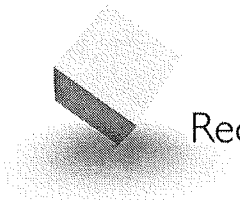


Recursosinformáticos

SQL Server 2016

Aprender a administrar una base de datos transaccional

con SQL Server Management Studio (con ejercicios y correcciones)

ISBN: 978-2-409-00862-7
Edición original: 978-2-409-00587-9

Ediciones ENI

P° Ferrocarriles Catalanes, 97-117, 2a pl. of. 18
08940 - Cornellà de Llobregat (Barcelona)

Tel: 934 246 401
Fax: 934 231 576

e-mail: info@ediciones-eni.com
http://www.ediciones-eni.com

Autor: Jérôme GABILLAUD
Edición española: Juan Francisco DIEZ LÉGLISE,
Angel Mª SÁNCHEZ CONEJO
Colección **Recursos Informáticos** dirigida por Joëlle MUSSET

Prólogo

Este libro presenta la administración de SQL Server y, de manera más particular, ofrece una introducción precisa a los diferentes conceptos de administración de una base de datos.

Se estudian las acciones necesarias para la realización de las tareas de administración diarias, en forma de concepto, y después se detalla la implementación de la funcionalidad tal y como se aplica en SQL Server. En el marco de las operaciones más frecuentes, se ha elegido presentar las operaciones con la interfaz gráfica SQL Server Management Studio, además de las instrucciones Transact SQL asociadas. Respecto a las operaciones puntuales, solo se presenta el modo operativo con la interfaz gráfica.

Este libro se dirige particularmente a cualquier persona que desee aprender la administración de bases de datos SQL Server. No es necesario tener conocimientos relativos a la administración de bases de datos antes de abordar este libro.

Para sacar el máximo partido de las diferentes instrucciones presentes en este libro, es aconsejable tener conocimientos de Transact SQL y del lenguaje SQL. Este conocimiento de Transact SQL permitirá al lector entender mejor y aprovechar las diferentes instrucciones Transact SQL que recogemos aquí. Por otro lado, este conocimiento también permitirá entender mejor cómo se pueden utilizar los datos y hacer las elecciones correctas en términos de administración. Por tanto, es necesario saber cómo se usan los datos presentes en la base de datos para administrarlos mejor.

Después de detallar las posibles opciones durante la instalación, se presentan las posibilidades que se ofrecen en términos de almacenamiento, seguridad, copias de seguridad, restauración y reparto de datos. Todo ello se completa con un capítulo que permite ofrecer alguna pista sobre la optimización de servidor.

Podrá descargar algunos elementos de este libro en la página web de Ediciones ENI: **http://www.ediciones-eni.com**. Escriba la referencia ENI del libro **RIT16SQLA** en la zona de búsqueda y valide. Haga clic en el título y después en el botón de descarga.

Capítulo 3
Gestión de la base de datos

Capítulo 4
Gestión de la seguridad del acceso

Capítulo 5
Tareas planificadas

Capítulo 6
Transferencia de datos

Capítulo 7
Replicación

Capítulo 8
Copia de seguridad

Capítulo 9
Restauración

Capítulo 10
Herramientas adicionales

Anexo

Capítulo 1
Presentación

1. Introducción

Desde hace ya algunas versiones, SQL Server ha alcanzado una cierta madurez en lo referente a la gestión de una base de datos relacional, y las evoluciones más destacadas se sitúan en la parte de BI aunque también en la integración de SQL Server en Windows Azure (estos dos puntos son objeto de este libro).

Sin embargo, esto no significa en ningún caso que no existen mejoras a nivel del propio SQL Server.

Primero, en lo referente a la distribución, la instalación de la herramienta cliente SQL Server Management Studio está completamente desligada de la instalación del motor de la base de datos, lo cual es algo bueno dado que el puesto donde se ejecuta SQL Server Management Studio no es, en general, el mismo sobre el que funciona la base de datos.

Como complemento a la evolución de algunos comandos, las herramientas mejoran con la intención de integrarse. Así, SQL Server Profiler ya no existe (al menos es recomendable no utilizarlo), sino que se propone una solución completamente integrada en SQL Server Management Studio.

De la misma forma, el almacén de consultas permite un mejor seguimiento de los tiempos de ejecución de las mismas además de asegurar la buena salud del servidor con indicadores como el tiempo consumido por el procesador, el número de lecturas lógicas, el tiempo de ejecución de las consultas...

SQL Server 2016 propone también las tablas temporales, lo que permite seguir la evolución de los datos. SQL Server gestiona el tiempo de conservación del histórico así como el volcado de la información de la tabla de datos hacia la tabla de histórico.

SQL Server se encarga de toda la parte técnica para asegurar la puesta en marcha de esta funcionalidad.

2. Presentación de SQL Server

El objetivo de este capítulo es presentar SQL Server de manera global y adquirir una visión general de SQL Server en su conjunto, a saber:

- Entender el concepto de SGBDR y el modo de funcionamiento cliente/servidor.
- Presentar los componentes de SQL Server y las plataformas de ejecución.
- Presentar la arquitectura de administración y de programación.
- Presentar el concepto de base de datos y las bases de datos instaladas en el servidor SQL.

SQL Server es un SGBDR (sistema de gestión de base de datos relacional) completamente integrado en Windows, lo que permite realizar numerosas simplificaciones a nivel de administración, ofreciendo un máximo de posibilidades.

2.1 ¿Qué es un SGBDR?

SQL Server es un sistema de gestión de base de datos relacional (SGBDR), lo que le confiere una gran capacidad de gestionar los datos, conservando su integridad y su coherencia.

SQL Server se encarga de:

- Almacenar los datos.
- Verificar las restricciones de integridad definidas.
- Garantizar la coherencia de los datos que almacena, incluso en caso de error (parada repentina) del sistema.
- Asegurar las relaciones entre los datos definidos por los usuarios.

Este producto está completamente integrado en Windows a varios niveles:

- Observador de eventos: se utiliza el diario de las aplicaciones para registrar los errores generados por SQL Server. Windows centraliza la gestión de errores, lo que facilita el diagnóstico.
- Analizador de rendimientos: mediante la adición de numerosos contadores, es sencillo detectar los cuellos de botella y reaccionar de manera más adecuada para evitar estos problemas. Se utiliza toda la potencia del Analizador de rendimiento y es posible, dentro de la misma herramienta, tener los contadores tanto sobre SQL Server como sobre Windows y, de esta manera, poder detectar cuál es y dónde se encuentra el verdadero problema.

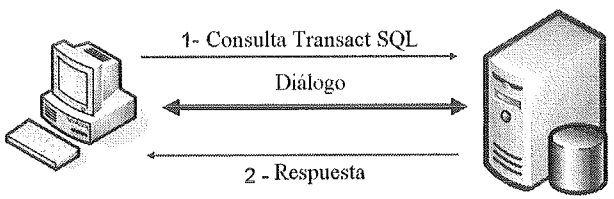
- Tratamientos en paralelo: SQL Server es capaz de aprovechar las capacidades de las arquitecturas mutiprocesador. Cada instancia de SQL Server dispone de su propio proceso de ejecución, y los threads de Windows o bien los hilos (si la opción está activada) se ejecutan con el objetivo de explotar al máximo la arquitectura material disponible. Cada instancia de SQL Server ejecuta siempre varios threads de Windows. Para tener en cuenta todos los procesadores presentes en el sistema, el parámetro de configuración **max degree of parallelism** debe tener el valor 0. Es el valor por defecto. Para evitar la generación de ejecuciones en paralelo, es suficiente con poner a 1 el valor de este parámetro. Por último, asignando a este parámetro un valor comprendido entre 2 y el número de procesadores, es posible limitar el grado de paralelismo. El valor máximo soportado por el parámetro es 64.
- Seguridad: SQL Server es capaz de basarse completamente sobre la seguridad gestionada por Windows, con el objetivo de permitir a los usuarios finales tener un único nombre de usuario y una única contraseña. No obstante, SQL Server gestiona su propio sistema de seguridad para todos los clientes no Microsoft.
- Se utilizan los servicios de Windows para la ejecución de los componentes de software correspondientes al servidor. La gestión del servidor (parada, inicio y suspensión) es más fácil y es posible disfrutar de todas las funcionalidades asociadas a los servicios de Windows (inicio automático, ejecución en el contexto de una cuenta de usuario de dominio...).
- Active Directory: los servidores SQL y sus propiedades se registran automáticamente en el servicio de directorio Active Directory. De esta manera, es posible efectuar búsquedas en Active Directory para localizar las instancias de SQL Server que están funcionando.

SQL Server puede gestionar dos tipos de bases de datos diferentes:

- Las bases OLTP (*OnLine Transactional Processing*), que corresponden a las bases de datos en las cuales la información se almacena de manera directa, con el objetivo de reutilizarla más tarde tal y como fue almacenada.
- Las bases OLAP (*OnLine Analytical Processing*), que contienen información estadística con el objetivo de poder extraer información en forma de cubos multidimensionales para ayudar en la toma de decisiones, por ejemplo. Las estadísticas contenidas en las bases OLAP están basadas en la información incluida en una base OLTP.

2.2 Modo de funcionamiento cliente/servidor

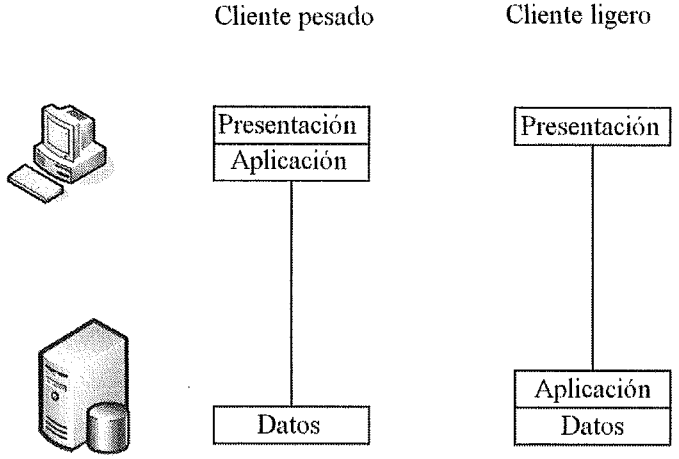
Todas las aplicaciones que utilizan SQL Server para gestionar los datos se apoyan en una arquitectura cliente/servidor. La aplicación cliente se encarga de la puesta en marcha de la interfaz de usuario. Esta aplicación se ejecuta generalmente en varios puestos clientes al mismo tiempo. En lo que respecta al servidor, este se encarga de la gestión de los datos y del reparto de los recursos del servidor entre las diferentes solicitudes (consultas) de los clientes. Las reglas de gestión de la empresa se reparten entre el cliente y el servidor.



Modo de funcionamiento cliente/servidor

Podemos distinguir tres casos:

- Las reglas se implementan completamente en el cliente, llamado **cliente pesado**. Esta solución permite liberar recursos a nivel del servidor, aunque aparecen problemas de actualización de los clientes y de desarrollo de otras aplicaciones.
- Las reglas se definen completamente en el servidor. El cliente es, por lo tanto, un **cliente ligero**. Esta solución permite obtener clientes que consumen pocos recursos materiales y una centralización de las reglas, lo que flexibiliza las actualizaciones. Sin embargo, el servidor consume muchos recursos y el nivel de interacción con el usuario corre el riesgo de degradarse, ya que el conjunto de restricciones se verifica en el momento en que el usuario envía su solicitud (consulta) al servidor.
- Las reglas de la empresa se definen en una tercera máquina, llamada **Middle Ware**, con el objetivo de disminuir los recursos utilizados por el cliente y el servidor, conservando en todo momento la centralización de las reglas.



La arquitectura cliente/servidor permite un despliegue óptimo de las aplicaciones clientes en numerosos puestos siempre conservando una gestión centralizada de los datos (en el servidor), lo que hace posible compartir la información en el interior de la empresa.

También es posible tener varias aplicaciones clientes en el mismo servidor de base de datos. Esta posibilidad ofrece numerosas funcionalidades, aunque debe asegurarse que la carga de trabajo a la que se somete el servidor no sea demasiada para las capacidades de la máquina. Esta arquitectura cliente/servidor se respeta en todas las herramientas que permiten acceder a la información contenida en el servidor SQL; por lo tanto, también en las herramientas de administración, aunque estas no estén instaladas en el servidor.

Todas las peticiones que provienen de los clientes y se dirigen al servidor deben escribirse en Transact-SQL. Este lenguaje de consulta de base de datos respeta la norma ANSI SQL-92. SQL proporciona un conjunto de comandos para gestionar los objetos y manipular los datos de las bases de datos. Transact SQL está enriquecido con numerosas funcionalidades, no normalizadas, con el objetivo de ampliar las capacidades del servidor. De esta manera, es posible definir procedimientos almacenados en el servidor.

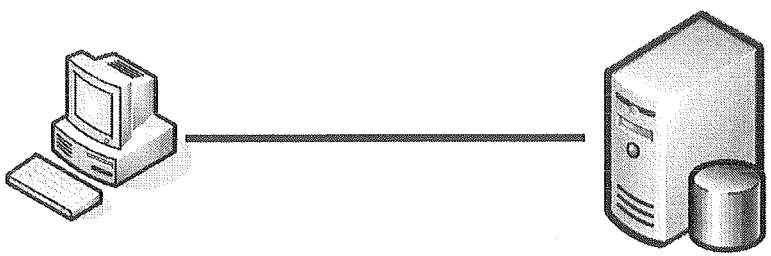
2.3 Las posibles plataformas

Es importante distinguir dos casos: por un lado, las posibles plataformas para el cliente. Por otro, las plataformas para el servidor.

Las plataformas clientes que se presentan aquí son los puestos sobre los que se pueden instalar las herramientas de administración de SQL Server. No se trata de los puestos que albergan una aplicación que se conecta a una instancia de SQL Server para gestionar los datos.

De una manera sintética, las herramientas clientes de administración se pueden instalar en todos los sistemas operativos Windows Server o puestos de trabajo.

Por el contrario, para la parte servidor, la disponibilidad en términos de plataforma se relaciona con la edición SQL Server que se haya elegido. Sin embargo, para albergar una instancia de base de datos en producción, es necesario disponer de un servidor de alto rendimiento. Por lo tanto, se recomienda una plataforma Windows Server. La edición de Windows se realizará en función de las restricciones que imponen tanto la edición de SQL Server que se haya elegido como el entorno técnico. La instalación de una instancia bajo Windows Client se reservará para puestos móviles.



Cliente Servidor

Si el cliente tiene instalada una aplicación específica, la gama de plataformas aumenta significativamente gracias, en particular, a que el controlador jdbc permite el acceso a una instancia de SQL Server desde una aplicación escrita en Java. La gama se ampliará en el caso de una aplicación ASPX que ofrezca una interfaz de Internet. Un simple navegador de Internet permite iniciar la aplicación.

2.4 Los componentes de SQL Server

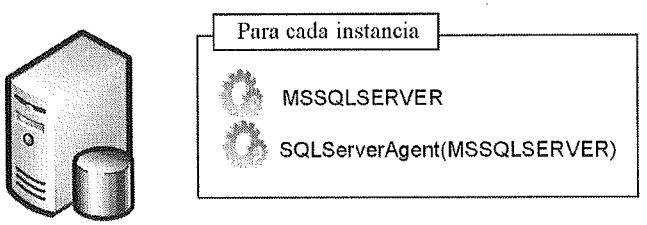
El motor de base de datos de SQL Server o Database Engine está compuesto por varios programas. Algunos se ejecutan en forma de servicios y otros tienen una interfaz de usuario gráfica o una línea de comandos.

Componentes de servidor

SQL Server se ejecuta en forma de servicios de Windows. Según las opciones de instalación elegidas, puede tener más o menos servicios. Los principales servicios son:

- SQL Server: es el servidor de base de datos propiamente dicho. Si no se inicia este servicio, no es posible acceder a la información. Gracias a él, SQL Server asegura la gestión de las consultas de usuario. Se hace referencia a este servicio con el nombre MSSQLSERVER para la instancia por defecto y MSSQLSERVER$*nombreInstancia* en el caso de cualquier otra instancia con un nombre dado.

- SQL Server Agent: este servicio se encarga de la ejecución de las tareas planificadas, la vigilancia de SQL Server y el seguimiento de las alertas. Está directamente unido a una instancia de SQL Server. En el gestor de servicios, se hace referencia a él con el nombre SQLServerAgent(MSSQLSERVER) para la instancia por defecto y SQLServerAgent(*nombreInstancia*) en el caso de cualquier otra instancia con un nombre dado.
- Microsoft Full Text Search: este servicio se encarga de gestionar la indización de los documentos de tipo texto almacenados en SQL Server y gestionar igualmente las búsquedas por palabras clave.



Es posible instalar varias instancias de SQL Server en el mismo puesto.

Conectividad de cliente

La instalación de los componentes de conectividad en los puestos clientes permite administrar la red y soportar OLE-DB.

Herramientas de gestión

Las tareas de administración se realizan mediante el uso de herramientas. La mayor parte de ellas tienen una interfaz gráfica de fácil manejo y uso intuitivo. Sin embargo, las tareas administrativas se deben meditar muy bien antes de realizarse. El uso de algunas herramientas implica que el componente de servidor correspondiente esté instalado.

Estas herramientas son:

- SQL Server Management Studio, para realizar todas las operaciones a nivel del servidor de base de datos.
- Administración de configuración de SQL Server para administrar los servicios relacionados con SQL Server.
- SQL Server Profiler, para el seguimiento y análisis de la carga de trabajo de una instancia de SQL Server.
- Asistente de parametrización del motor de base de datos, que permite una optimización del funcionamiento del servidor de base de datos.

Todas estas herramientas y el funcionamiento de SQL Server están profusamente documentados.

Los componentes

Las diferentes piezas de software que proporciona SQL Server se articulan siempre en torno al motor de base de datos relacionales, que trata de manera eficiente la información almacenada en formato relacional y en formato xml.

- SQL Server Integration Service (SSIS) es una herramienta de importación y exportación de datos fácil de utilizar y muy parametrizable.
- La replicación de los datos en diferentes instancias permite colocar los datos lo más cerca posible de los usuarios y reducir los tiempos de procesamiento.
- SQL Server integra ahora el lenguaje estadístico R. Es decir, ahora se pueden consultar datos almacenados en una base de datos SQL Server directamente con el lenguaje R. Este lenguaje estadístico está particularmente orientado a las bases de datos decisionales, que no se describen en este libro.
- La integración de CLR en SQL Server permite desarrollar procedimientos y funciones utilizando los lenguajes VB.NET y C#. La integración de CLR no pretende sustituir a Transact SQL, aunque se presenta como un complemento con el objetivo de poder realizar una codificación simple y de buen rendimiento para el conjunto de las funcionalidades que deben estar presentes en el servidor.

CLR

La integración de CLR (*Common Language Runtime*) en SQL Server permite aumentar considerablemente las posibilidades ofrecidas en términos de programación. La presencia de CLR no afecta a Transact SQL. Ambos son complementarios. Transact SQL es perfecto para escribir los procedimientos o funciones para los que hay un tratamiento intensivo de los datos. Por el contrario, en los casos en que el volumen de datos manipulados no es grande, CLR permite escribir con sencillez tratamientos complejos, ya que se beneficia de toda la riqueza de CLR.

CLR permite igualmente definir sus propios tipos de datos o bien nuevas funciones de cálculo agregado.

Por último, CLR permite a los desarrolladores de aplicaciones desarrollar los procedimientos y funciones en SQL Server conservando sus lenguajes favoritos (VB.NET o C#, por ejemplo) y, por lo tanto, sin necesidad de dominar Transact SQL.

En caso de que el código esté escrito en Visual Studio, la integración de la versión compilada en SQL Server y el mapeo CLR-Transact SQL se realizan de manera automática. Es posible realizar el desarrollo sin utilizar Visual Studio, pero la integración con SQL Server se llevará a cabo de manera manual, lo que es una tarea tediosa.

PowerShell

SQL Server, como todos los servidores de Microsoft, integra completamente PowerShell como lenguaje de scripting. A diferencia de otros servidores, SQL Server dispone de pocos cmdlets específicos ya que PowerShell ofrece la posibilidad de ejecutar instrucciones con formato Transact SQL, que ya es el lenguaje de script utilizado en SQL Server. Pero como PowerShell está basado en el framework .Net, también es posible ejecutar las operaciones de administración a través de la biblioteca SMO.

Por estas dos razones, los comandos PowerShell no se presentan con cada instrucción Transact SQL. Por el contrario, en el capítulo Herramientas adicionales, se dedica una parte al PowerShell, los cmdlets específicos a SQL Server y una segunda parte a SMO.

3. Arquitectura

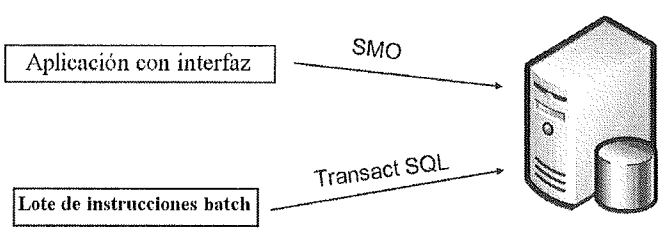
3.1 Administración

El lenguaje natural de SQL Server es Transact SQL. Por tanto, es necesario transmitir las instrucciones en este lenguaje. Como este lenguaje no es forzosamente natural para el usuario, es posible realizar la instrucción de manera gráfica con SQL Server Management Studio, y después ejecutarla en el servidor con los botones **Aceptar**, **Aplicar**... Las herramientas gráficas utilizan la biblioteca SMO (*SQL Server Management Object*) para establecer un diálogo eficaz con el servidor.

SQL SMO engloba y extiende SQL DMO. Por lo tanto, la biblioteca SMO es compatible con SQL Server 7, SQL Server 2000, 2005, 2008, 2012, 2014 y 2016.

Incluso aunque las tareas de administración se tratan en este libro, un conocimiento adecuado de Transac SQL y de las posibilidades ofrecidas por este lenguaje es muy aconsejable para el lector que quiera administrar una base de datos.

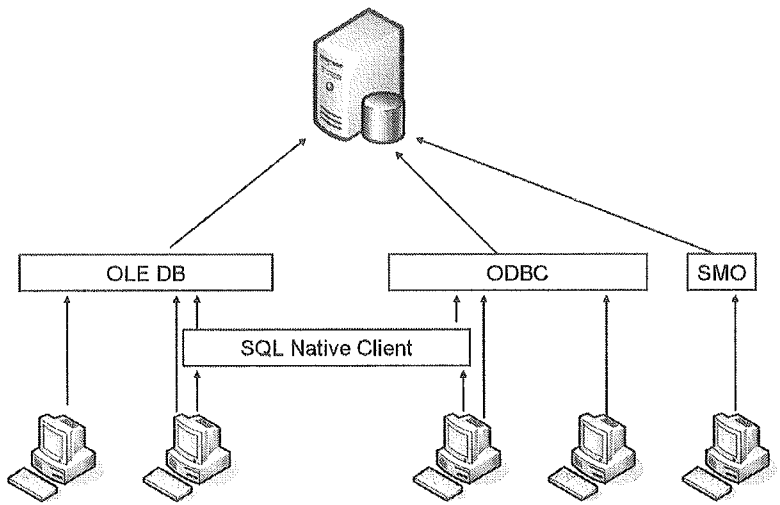
Es posible escribir scripts Transact SQL para ejecutar operaciones administrativas en forma de procesamientos batch.



Administración de SQL Server

3.2 Programación

El desarrollo de aplicaciones clientes para visualizar los datos del servidor se puede apoyar en diferentes tecnologías.



La DLL SQL Native Client es un método de acceso a los datos que está disponible para tecnologías OLE-DB u ODBC de acceso a datos. Con este nuevo API, se puede usar el conjunto de funcionalidades de SQL Server, como los tipos personalizados definidos CLR (UDT: *User Defined Type*), MARS o el tipo XML.

SQL Native Client es un API que permite aprovechar al máximo las funcionalidades de SQL Server y disponer de un programa de acceso óptimo al servidor.

Es posible utilizar los objetos ADO para acceder a la información. Esta elección es más estándar, ya que un programa que accede a una fuente de datos ADO puede trabajar correctamente tanto con una base Oracle como SQL Server, aunque no permite la misma gestión de todas las funcionalidades que ofrece SQL Server. El API SQL Native Client permite escribir programas clientes optimizados, aunque solo pueden acceder a los datos contenidos en un servidor SQL Server.

SQL Native Client será adoptado como modelo de acceso a datos en los nuevos programas escritos en C# o VB.NET que deseen trabajar con SQL Server, y en los programas existentes cuando estos últimos deseen trabajar con elementos específicos de SQL Server, como el tipo XML, por ejemplo.

Este modelo de programación corresponde a una aplicación cliente que desea gestionar datos. En caso de que la aplicación quiera poder hacer operaciones de administración, será necesario utilizar la biblioteca SMO.

4. Base de datos SQL Server

4.1 Objetos de base de datos

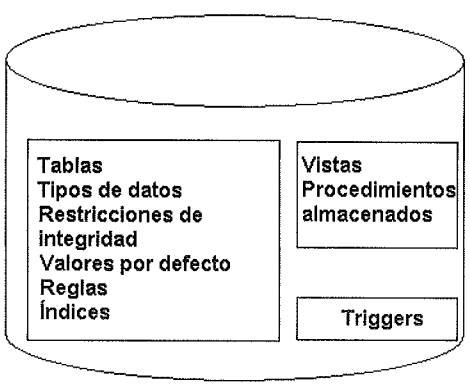
Las bases de datos contienen cierto número de objetos lógicos. Es posible agrupar estos objetos en tres grandes categorías:

- Gestión y almacenamiento de los datos: tablas, tipos de datos, restricciones de integridad, valores por defecto, reglas e índices.

Observación

Los valores por defecto y las reglas definidas en la base de datos estaban marcadas como obsoletas (deprecated) desde hace ya algunas versiones y ya no están disponibles en SQL Server 2016.

- Acceso a los datos: vistas y procedimientos almacenados.
- Gestión de integridad compleja: triggers (procedimientos almacenados que se ejecutan automáticamente en el momento de la ejecución de una orden SQL que modifique el contenido de una tabla: INSERT, UPDATE y DELETE). El trigger está siempre asociado a una tabla y a una instrucción SQL. Permite establecer reglas de integridad complejas entre varias tablas o mantener datos no normalizados.



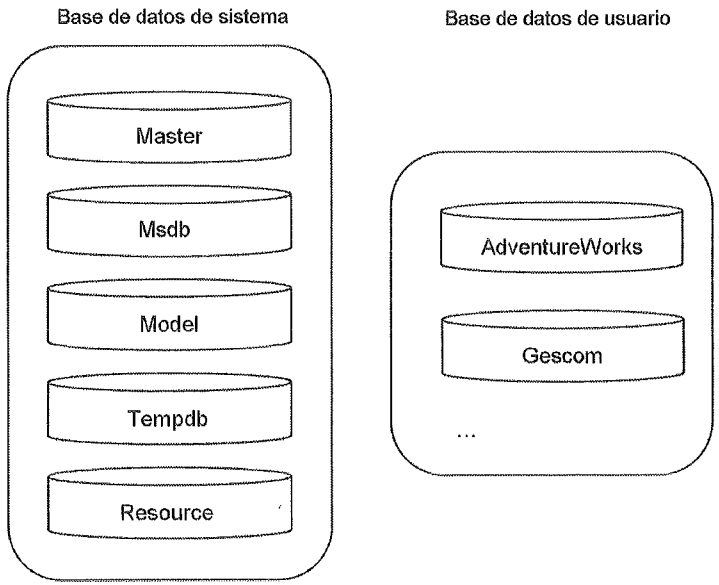
Objetos de base de datos

Nombre completo de los objetos

La regla aplicada para denominar a los objetos permite una identificación perfecta. El nombre completo se compone de la siguiente manera: *servidor.nombreBase.propietario.objeto*. Por defecto, solo es necesario el nombre de los objetos. Este concepto se explica en profundidad en el capítulo Gestión de la base de datos.

4.2 Bases de datos de sistema y tablas de sistema

Para gestionar el conjunto de datos almacenados, SQL Server se utiliza a sí mismo. Por lo tanto, existen bases de datos de sistema y, en cada base de datos de usuario, algunas tablas de sistema. La inserción y la actualización de datos en estas tablas no se efectúan nunca de manera directa, sino a través de comandos Transact SQL o de procedimientos almacenados.



Organigrama de bases de datos

Observación

Los nombres de base de datos y de tablas de sistema son fijos y conocidos por SQL Server. Por lo tanto, no es necesario renombrar una tabla o una base de datos de sistema.

Master

Es la base de datos principal de SQL Server, donde está almacenado el conjunto de datos estratégicos para el buen funcionamiento del servidor (cuentas de conexión, opciones de configuración, existencia de bases de datos de usuario y las referencias a los archivos que componen estas bases...). Por tanto, se hará regularmente copia de seguridad de esta base de datos, como parte de un plan de copias de seguridad.

Model

Esta base contiene el conjunto de elementos incluidos en toda base de datos de usuario nueva. Por defecto, solo están las tablas de sistema, aunque es posible añadir otros elementos.

Por tanto, la base de datos **Model** se utiliza con fines de personalización de las futuras bases de datos de usuarios, creadas a continuación. No es posible eliminar los elementos existentes por defecto en la base de datos **Model**.

Tempdb

La base **Tempdb** es un espacio temporal de almacenamiento compartido. Permite gestionar las tablas temporales locales o globales, las tablas de trabajo intermediarias para hacer ordenaciones, por ejemplo, pero también los conjuntos de resultado de los cursores. La base **Tempdb** se crea nuevamente, con su tamaño inicial, en el momento del inicio de la instancia. Por lo tanto, no se puede conservar de manera permanente ninguna información en la base de datos **Tempdb**. Los objetos temporales se eliminan cuando su propietario se desconecta.

Se pueden especificar los parámetros físicos de la base tempdb al instalar una instancia SQL Server. Por defecto esta base de datos se compone de ocho archivos de 8 MB.

Msdb

Contiene la información utilizada por el servicio SQL Server Agent para lanzar una alerta, avisar a un operador o ejecutar una tarea planificada. **Msdb** contiene también el histórico de ejecución de las tareas.

Resource

Esta base de datos de solo lectura contiene la definición de todos los nuevos elementos definidos a partir de SQL Server 2014. Los objetos de sistema se definen aquí de manera lógica y aparecen en el esquema de **sys**.

Esta base de datos no se puede incluir en un proceso de copia de seguridad clásica. Es necesario hacer una copia de seguridad directamente del archivo mdf que representa a la base de datos.

Base de datos de usuario

Las bases de datos de usuario van a contener los datos proporcionados por los usuarios. Las bases presentes en el esquema anterior (AdventureWorks y GESCOM) son las bases de ejemplo utilizadas en la documentación oficial de SQL Server y en este manual.

4.3 Las tablas de sistema

Las tablas de sistema siempre están presentes en SQL Server. Sin embargo, se recomienda no trabajar directamente con ellas. Para buscar información, es necesario pasar por el esquema de información y, más exactamente, por las vistas definidas en el esquema del usuario **sys** siempre que sea posible.

El motor de SQL Server usa directamente las tablas de sistema. Las aplicaciones que utilizan SQL Server no deben nunca acceder directamente a estas tablas, ni siquiera en modo de solo lectura. De hecho, como la estructura de estas tablas evoluciona con las versiones de SQL Server, si algunas aplicaciones acceden de manera directa a las tablas de sistema, podemos encontrar una imposibilidad a la hora de migrar a una nueva versión de SQL Server, a no ser que la aplicación se escriba de nuevo.

Sin embargo, se puede omitir esta regla cuando no se dispone de otro medio para conseguir la información.

Observación

SQL Server no tiene en cuenta los triggers que puedan estar definidos sobre tablas de sistema, ya que pueden entorpecer el buen funcionamiento de algunas operaciones.

En la tabla siguiente, se hace referencia a algunas tablas de sistema

Vista de sistema	Objetivo
sys.server_principals	Lista las conexiones definidas en el servidor.
sys.messages	Una línea para cada mensaje o advertencia que se define en el servidor, y esto para cada idioma soportado.
sys.databases	Una línea por cada base de datos (de sistema o usuario) que hay en el servidor.
sys.configurations	Cada línea corresponde a una opción de configuración. La consulta de esta vista permite conocer el valor de cada opción de configuración.
sys.database_principals	Una línea por cada usuario definido en la base de datos actual.
sys.columns	Una línea por cada columna de cada tabla, vista y parámetro de los procedimientos almacenados.
sys.objects	Una línea por cada elemento definido en la base de datos actual.

Vista de sistema	Objetivo
sys.databse_files	Esta vista devuelve una línea por cada archivo que compone la base de datos desde la cual se la llama. Esta vista de sistema es, por lo tanto, específica a cada base de datos y devuelve unos resultados específicos para cada base de datos.

4.4 Extracción de metadatos

Para consultar los datos contenidos en las tablas de sistema, se desaconseja realizar directamente una consulta de tipo SELECT. Es preferible hacer, a través de procedimientos almacenados, funciones de sistema y vistas del esquema de información.

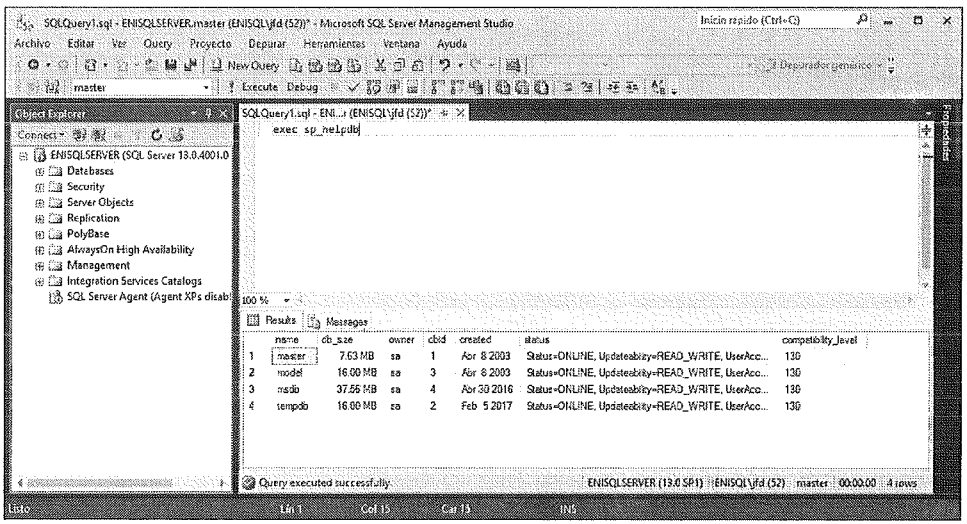
Procedimientos de sistema almacenados

Los procedimientos de sistema almacenados permiten consultar las tablas de sistema y conocer el estado del servidor, de la base de datos, etc. También se pueden usar para realizar operaciones de configuración.

Hay muchos procedimientos de sistema almacenados a nivel de servidor que no están recogidos en la documentación en línea de SQL Server. No se trata de un olvido, sino que esto permite identificar fácilmente un procedimiento almacenado presente en la versión actual cuya presencia no garantiza Microsoft en futuras versiones de SQL Server.

Para consultar las tablas de sistema, hay numerosos procedimientos almacenados. Todos ellos comienzan por **sp_**. Entre todos los procedimientos almacenados que existen, mencionamos los siguientes:

Procedimiento almacenado	Descripción
sp_help [nombre_objeto]	Información del objeto indicado.
sp_helpdb [nombre_base_datos]	Información de la base de datos indicada.
sp_helpindex [nombre_tabla]	Información de los índices de la tabla indicada.
sp_helplogins [nombre_conexion]	Información de la conexión indicada.
sp_who y sp_who2	Lista de los usuarios conectados actualmente.



El catálogo

SQL Server tiene vistas de sistema que permiten obtener la información de sistema. Todas estas vistas se presentan en el esquema **sys**.

La estructura de las vistas de sistema se podría enriquecer en futuras versiones de SQL Server. Por este motivo, es recomendable no utilizar el carácter genérico * en una instrucción SELECT, sino indicar de manera precisa el nombre de las columnas. Si se hace esto, el script no se verá afectado por los posibles cambios que se produzcan en la estructura de la vista.

Con el objetivo de navegar lo mejor posible en estas vistas, estas se agrupan en temas:

- Alta disponibilidad (AlwaysOn)
- Seguimiento de cambios
- CLR
- Colección de datos (vistas específicas de la base de datos tempdb)
- Espacios de almacenamiento
- Gestión de los correos electrónicos (Database Mail)
- Espejo de bases de datos
- Archivos
- EndPoints
- Propiedades extendidas
- Servidores asociados

- Mensajes de error
- Objetos
- Particiones
- Reglas (Policy-Based)
- Administrador de recursos
- Consultas
- Tipos de datos escalares
- Esquemas
- Seguridad
- Service Broker
- Configuración del servidor

Funciones de sistema

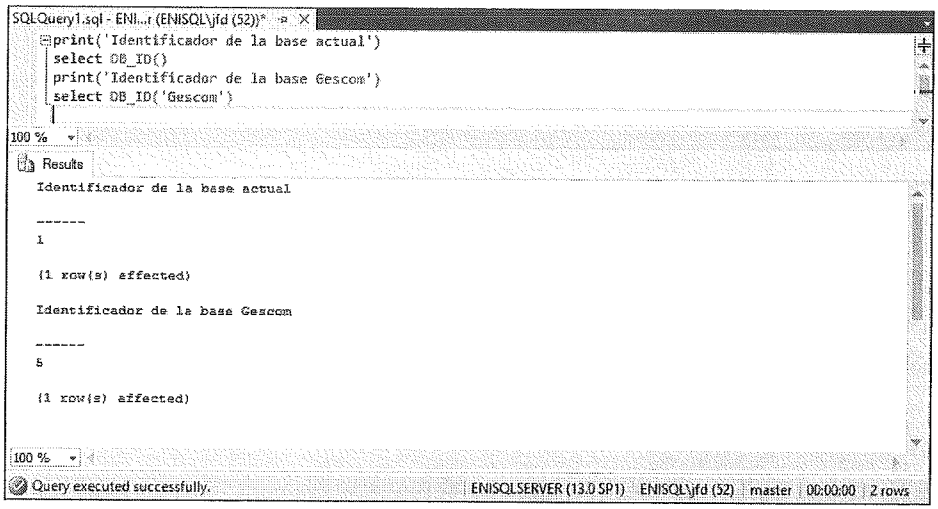
Las funciones de sistema se utilizan con los comandos Transact SQL. De esta manera, es posible recuperar los valores relativos a la base de datos en la que se trabaja, del servidor o de los usuarios.

Mencionamos algunas de ellas:

Funciones de sistema	Parámetro	Descripción
DB_ID	Nombre	Devuelve el identificador de la base de datos.
USER_NAME	ID	Devuelve el nombre de usuario a partir de su identificador.
COL_LENGTH	Nombre de la tabla y de la columna	Longitud de la columna.
STATS_DATE	Identificadores de la tabla y de las estadísticas	Fecha de última actualización de las estadísticas.
DATALENGTH	Expresión	Longitud de la expresión.

Ejemplo

En el ejemplo siguiente, la función DB_ID se presenta en dos ámbitos de uso diferentes. Si se utiliza sin argumentos, esta función permite conocer el identificador de la base de datos en la que estamos trabajando. Si se utiliza con argumentos, esta función permite conocer el identificador de esta base de datos.



Esquema de información

Es una serie de vistas que ofrecen una visualización de los parámetros de manera independiente de las tablas de sistema. El hecho de no hacer referencia directamente a las tablas de sistema impide posibles modificaciones que puedan aparecer en su estructura en próximas versiones. Además, estas vistas cumplen con la definición del estándar ANSI SQL para los esquemas de información. Cada vista presenta metadatos para el conjunto de los objetos presentes en la base de datos.

Las vistas del esquema de información:

CHECK_CONSTRAINTS

Visualiza el conjunto de restricciones del tipo CHECK definidas en la base de datos.

COLUMN_DOMAIN_USAGE

Visualiza el conjunto de columnas definidas en un tipo de datos de usuario.

COLUMN_PRIVILEGES

Visualiza el conjunto de privilegios, a nivel de columna, para el usuario actual o para otro usuario de la base de datos.

COLUMNS

Visualiza la definición del conjunto de columnas accesibles, en la base de datos actual, para el usuario actual.

CONSTRAINT_COLUMN_USAGE

Visualiza el conjunto de columnas para las que existe una restricción.

CONSTRAINT_TABLE_USAGE

Visualiza el conjunto de tablas para las que hay, al menos, una restricción definida.

DOMAIN_CONSTRAINTS

Visualiza el conjunto de tipos de datos de usuario que son accesibles para el usuario actual y que están asociados a una regla.

DOMAINS

Visualiza el conjunto de tipos de datos de usuario que son accesibles para el usuario actual.

KEY_COLUMN_USAGE

Visualiza el conjunto de columnas para las que hay definida una restricción de clave.

PARAMETERS

Visualiza las propiedades de los parámetros de las funciones definidas por el usuario y de los procedimientos almacenados accesibles para el usuario actual. Para las funciones, también se visualiza la información relativa al valor de retorno.

REFERENTIAL_CONSTRAINTS

Visualiza el conjunto de restricciones de clave foránea definidas en la base de datos actual.

ROUTINE_COLUMNS

Visualiza las propiedades de cada columna reenviadas por las funciones accesibles para el usuario actual.

ROUTINES

Visualiza el conjunto de procedimientos almacenados y de funciones accesibles para el usuario actual.

SCHEMATA

Visualiza las bases de datos para las que el usuario actual tiene permisos.

SEQUENCES

Permite conocer las diferentes secuencias con el detalle de los parámetros de cada secuencia.

TABLE_CONSTRAINTS

Visualiza las restricciones a nivel de tabla de la base de datos actual.

TABLE_PRIVILEGES

Visualiza el conjunto de privilegios del usuario actual en la base de datos actual y el conjunto de privilegios que el usuario actual otorga a otros usuarios de la base de datos actual.

TABLES

Visualiza el conjunto de tablas de la base de datos actual para las que el usuario actual tiene derechos.

VIEW_COLUMN_USAGE

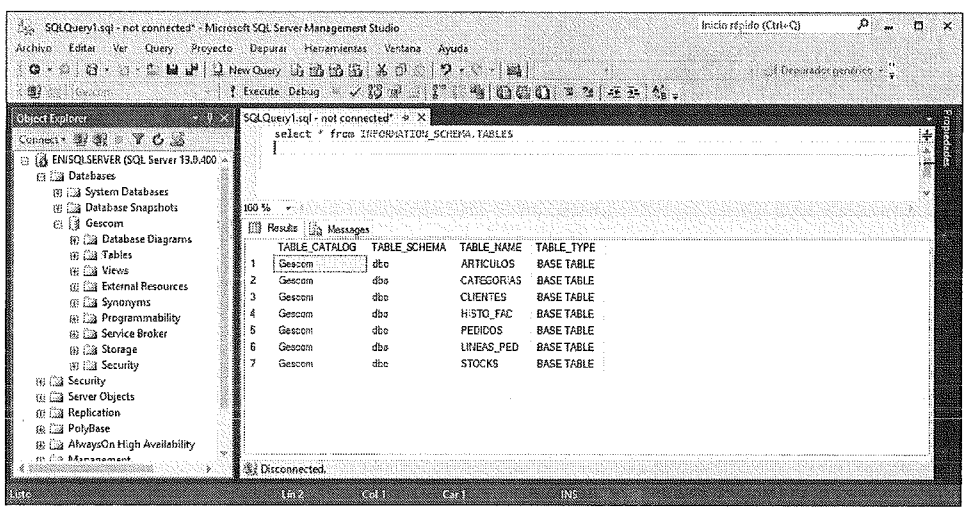
Visualiza el conjunto de columnas de tablas que se utilizan en la definición de una o varias vistas.

VIEW_TABLE_USAGE

Visualiza el conjunto de tablas de la base de datos actual que se utilizan a la hora de definir las vistas.

VIEWS

Visualiza el conjunto de vistas accesibles para el usuario actual en la base de datos actual.



4.5 Las tareas del administrador

El administrador de la base de datos tiene como objetivo principal mejorar el funcionamiento de la base de datos. Aunque SQL Server ofrece numerosas herramientas y algoritmos de autooptimización, todavía queda mucho trabajo para el administrador.

Las principales tareas del administrador son:

- Gestionar los servicios SQL Server.
- Gestionar las instancias de SQL Server.
- Establecer los procesos de copia de seguridad y restauración.
- Configurar la disponibilidad de los datos de acuerdo con la política de la empresa.
- Gestionar las configuraciones de red.
- Importar y exportar datos.

Además de las competencias en sistemas que debe tener el administrador para ser capaz de gestionar lo mejor posible la instancia de SQL Server, es importante que conozca las diferentes posibilidades ofrecidas por SQL Server para la automatización de las tareas con SQL Agent.

Para medir el resultado de su trabajo y comparar las diferentes opciones de configuración que puede tener que realizar, el administrador debe ser capaz de usar las herramientas y métodos asociados a SQL Server.

Por último, y sin duda el aspecto más importante, la administración de la base debe inscribirse en un proceso más global que implique al administrador desde el diseño de la base de datos para tomar las mejores decisiones en términos de arquitectura desde el momento de su diseño. El administrador podrá, de esta manera, intervenir en la creación de la base de datos y en las decisiones que se tomen, como, por ejemplo, el número de archivos que se ha de utilizar, los índices, las vistas y los procedimientos almacenados que es necesario definir para optimizar el tráfico de red, pero también para simplificar la gestión de derechos de acceso. De la misma manera, será el administrador el que aconseje sobre la partición o no de una tabla.

Capítulo 2
Instalación y configuración

1. Instalación de SQL Server

La instalación de SQL Server permite presentar las diferentes ediciones de SQL Server, así como detallar las posibles opciones de instalación. Se resaltará un aspecto particular para detallar la ejecución de los servicios asociados a SQL Server (software de servidor). Una vez que se ha instalado el servidor, es necesario asegurarse de que la instalación se ha llevado a cabo de forma correcta y posteriormente configurar el servidor y ciertas herramientas clientes para completar las diferentes etapas de administración.

Aunque es fácil de realizar, la instalación de SQL Server debe ser una operación meditada. En efecto, existen numerosas opciones de instalación y su elección debe corresponder a una necesidad real o bien permitir cubrir una evolución futura del sistema.

1.1 Las ediciones de SQL Server

Hay disponibles varias ediciones de SQL Server. Cada edición tiene características específicas. En función de las posibilidades elegidas, se optará por seleccionar una edición u otra.

SQL Server ofrece tres ediciones principales, que responden a la mayoría de los casos que se pueden presentar en una empresa. De manera adicional a estas soluciones, existen dos ediciones que responden a necesidades concretas en términos de disponibilidad o uso.

Para permitir una mejor agrupación de los servicios, es posible instalar varias instancias de SQL Server en el mismo servidor. Esta noción de instancia permite optimizar el uso de las licencias, separando la gestión de las diferentes instancias.

Para terminar, la puesta en marcha de SQL Server en un servidor Windows Core tiene características y especificaciones no tratadas en este libro.

Enterprise

La edición Enterprise es la más completa. Ofrece el conjunto de funcionalidades disponibles con SQL Server. Esta edición es conocida por ser capaz de gestionar volúmenes muy importantes de datos y transacciones, con muchos usuarios conectados.

Ofrece funcionalidades avanzadas en lo relativo a operaciones de *business intelligence* y alta disponibilidad.

Standard

Esta edición, más simple que la edición Enterprise, tiene como objetivo responder a las necesidades de una empresa que busque un motor de base de datos con alto rendimiento y no necesite funcionalidades específicas de la edición Enterprise.

Algunas limitaciones se encuentran sobre todo a nivel de las soluciones de alta disponibilidad. La gestión de volúmenes de datos muy grandes también tiene un rendimiento más bajo que en la edición Enterprise.

Esta edición representa una buena elección, pues que permite una fácil evolución hacia otras conservando la herramienta de administración SQL Server Management Studio.

Express

La edición Express de SQL Server tiene la particularidad de poder utilizarse en producción sin que sea necesario adquirir una licencia de SQL Server. No se trata de una versión reducida de SQL Server, sino de un motor SQL Server totalmente funcional. No existe límite respecto al número de usuarios conectados. Las únicas limitaciones son relativas al volumen de datos, 10 GB, y el hecho de que el motor no pueda explotar más de un gigabyte de memoria. Es razonable pensar que, cuando la aplicación alcance estos límites, la empresa disponga de los medios necesarios para adquirir una versión completa de SQL Server.

Esta edición Express está aconsejada para los desarrolladores de aplicaciones, ya que será posible migrar de una manera sencilla a versiones superiores de SQL Server.

Este tipo de edición también se adapta bien a las aplicaciones autónomas. Efectivamente, la edición Express se puede instalar en una plataforma Windows de usuario, como por ejemplo el ordenador portátil de un comercial que lleva consigo la base de todos los artículos de la empresa. Esta base se puede actualizar mediante el proceso de replicación de SQL Server que permite sincronizar los datos en el catálogo e inyectarlos en el sistema de información de la empresa.

Esta edición también es útil en el marco de una aplicación monopuesto que necesite una gestión robusta y fiable de los datos. La elección de SQL Server para este tipo de aplicaciones permite dejar abierto el camino hacia una gestión multiusuario.

Developer

Además de todas estas ediciones de producción, SQL Server también ofrece una edición Developer. Esta incluye el conjunto de las funcionalidades propuestas por la edición Enterprise. Sin embargo, con una edición Developer, la puesta en producción de aplicaciones no es legal. Como su propio nombre indica, la versión Developer permite al equipo de desarrollo de aplicaciones hacer sus pruebas sobre una base completamente funcional sin tener la obligación de adquirir una licencia de producción.

La edición Developer está disponible gratuitamente descargándola desde el sitio web de Microsoft en la siguiente URL: https://www.microsoft.com/en-us/sql-server/sql-server-editions-developers. El uso de esta edición es una buena manera de probar las diferentes funcionalidades de SQL Server.

Web

Centrada en la gestión de datos, esta edición permite ofrecer un motor de base de datos destinado a los sitios web con un coste bajo. Las posibilidades en términos de administración son reducidas. Las funcionalidades decisionales y la creación de informes no están disponibles en esta edición, que se halla limitada a los centros de servicios.

1.2 Desarrollo del proceso de instalación

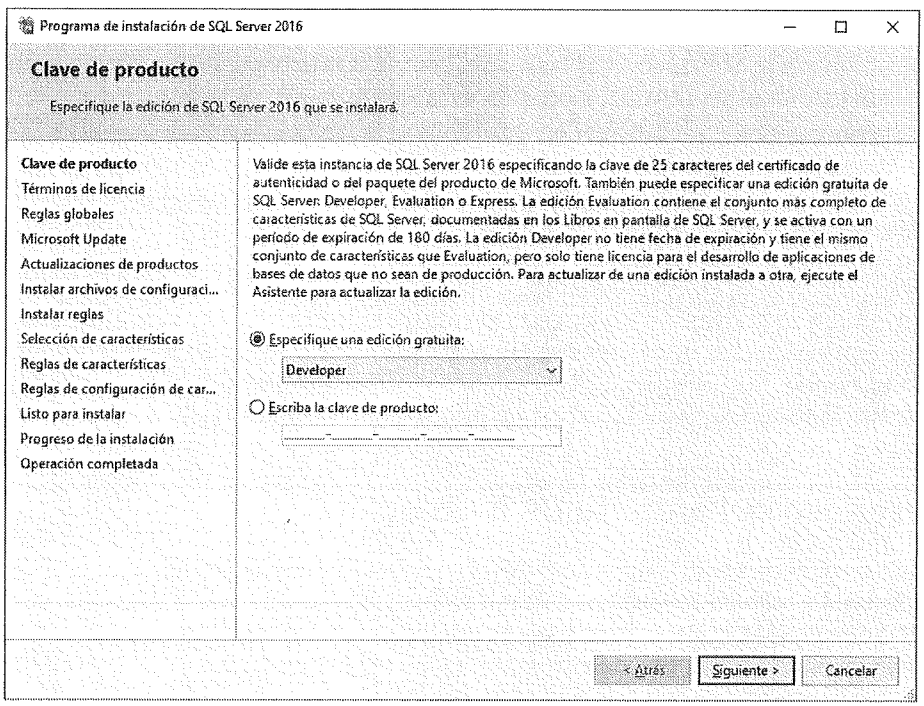
Como sucede con muchas herramientas, la instalación se divide en dos fases muy diferenciadas. La primera consiste en solicitar al usuario la configuración exacta de la instalación que desea realizar. Esta etapa es relativamente rápida. De hecho, en función de las opciones del usuario, los elementos y las opciones se cargan o no. Las opciones que se definen durante esta fase de preguntas-respuestas son muy importantes porque algunas de ellas solo se pueden tomar durante la instalación del motor SQL Server.

El proceso de instalación hace referencia al motor de base de datos. La instalación de la consola de administración SQL Server Management Studio se realiza de forma independiente.

Observación

Las etapas detalladas para una instalación del motor también son válidas en lo que respecta a la instalación de una nueva instancia. La etapa relativa a la ubicación del programa es la única que no se presenta.

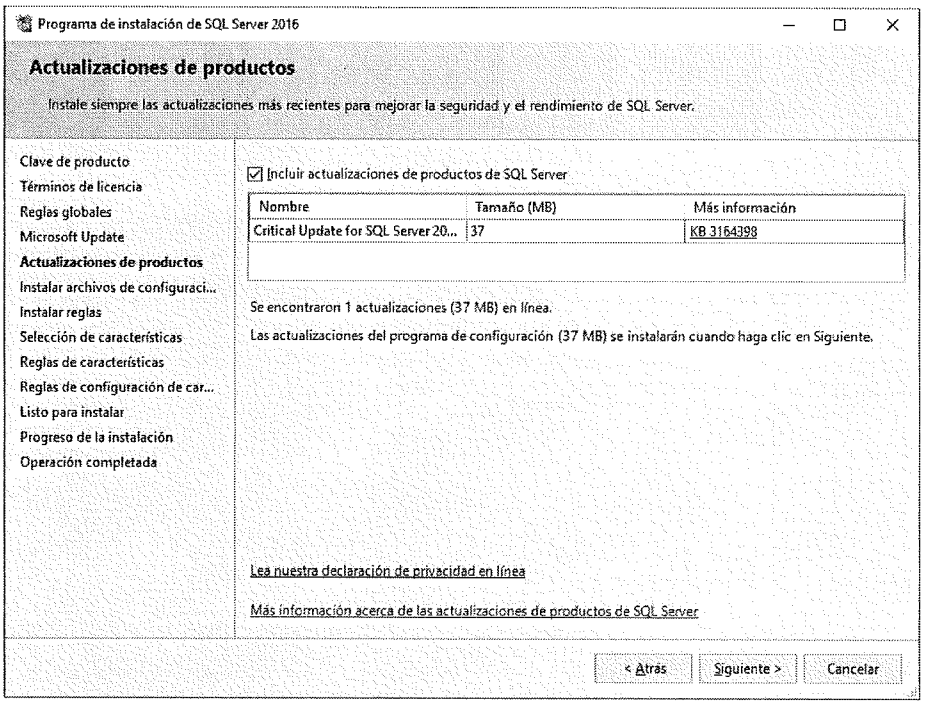
Después de introducir la clave del producto y aceptar el contrato de licencia, se instalan los archivos de soporte del programa de instalación de SQL Server.



El proceso de instalación de SQL Server comienza ejecutando una serie de reglas para validar la configuración de la plataforma.

Si estas reglas no se satisfacen completamente, se obtiene como resultado advertencias o errores. Una advertencia indica que, aunque es posible instalar una instancia de SQL Server, ciertos componentes no se podrán instalar.

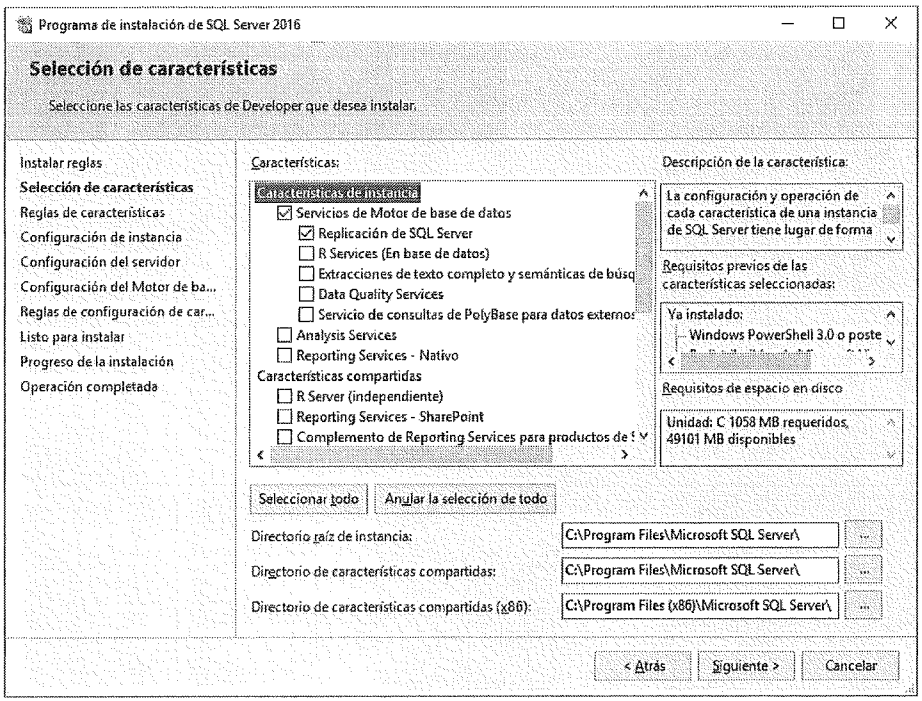
El proceso de instalación comprueba si existen actualizaciones pendientes para el producto.



1.2.1 Elección de los componentes

Antes de parametrizar la instalación, SQL Server solicita la elección de los componentes que se desea instalar en el puesto local. Aquí es cuando se puede determinar si se desea instalar solo las herramientas o también el motor de base de datos.

Se trata de afinar la selección de los componentes que se desea instalar. No se trata de marcar todas las opciones, sino de seleccionar los componentes (cliente o servidor) que se van a utilizar realmente. Limitando el número de componentes instalados, se reduce la superficie de ataque del sistema y se evita la sobrecarga de este con componentes que no se utilizan. Por supuesto, si hay componentes que no se han seleccionado en el momento de la instalación inicial y son necesarios más adelante, se pueden añadir posteriormente.

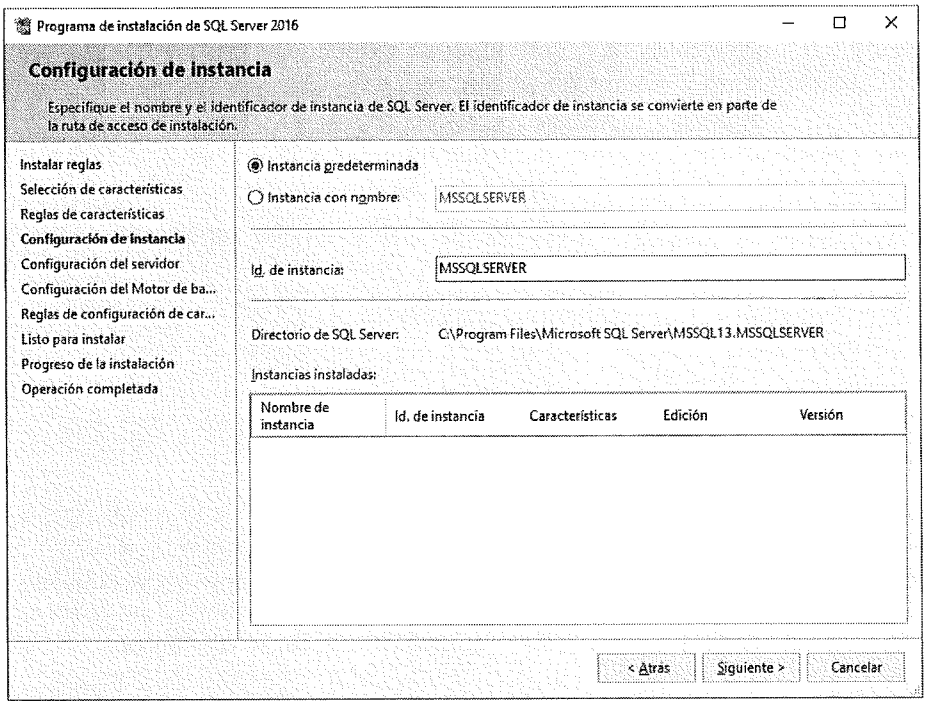


Para cada componente, es posible definir su ubicación física en el disco duro. Para acceder a esta pantalla de configuración, es necesario pulsar el botón ... .

En función de los elementos seleccionados, puede que sea necesario instalar algunos componentes adicionales. La instalación de estos componentes adicionales se puede integrar dentro del proceso de instalación de SQL Server o realizarse de manera independiente.

1.2.2 Nombre de la instancia

MS SQL Server ofrece la posibilidad de instalar una o varias instancias del motor de base de datos en el mismo servidor. Cada instancia es totalmente independiente de las otras instaladas en el mismo servidor y se gestiona de manera autónoma. Cuando hay varias instancias en el mismo servidor, es necesario utilizar el nombre de cada una de ellas para distinguir las unas de las otras. La primera instancia instalada normalmente es la instancia predeterminada y tiene el mismo nombre que el servidor sobre el que se instala.



Las otras instancias tienen su propio nombre. Después de seleccionar el nombre de la instancia que se va a instalar, el proceso de instalación verifica la capacidad de disco disponible del sistema.

Observación

Un mismo servidor solo puede contener una única instancia predeterminada.

Cada instancia está perfectamente identificada por su nombre. El nombre de la instancia debe respetar las reglas siguientes:

- El nombre de la instancia está limitado a 16 caracteres.
- No se diferencia entre mayúsculas y minúsculas.
- El nombre de la instancia no puede contener las palabras DEFAULT y MSSQLSERVER, así como ninguna otra palabra reservada.
- El primer carácter del nombre de la instancia debe ser una letra (A a Z) o bien el guion bajo (_).
- Los otros caracteres pueden ser letras, números o bien el guion bajo (_).

- Los caracteres especiales tales como el espacio, la barra inversa (\), la coma, los dos puntos, el punto y coma, la comilla simple, el ampersand (&) y la arroba (@) no están permitidos en el nombre de una instancia.

1.2.3 Los servicios de SQL Server

En función de las elecciones realizadas en el proceso de instalación, se pueden haber creado varios servicios. Entre ellos, los más habituales son los siguientes:

- SQL Server Database Services.
- Agent SQL Server.
- Analysis Services.
- Reporting Services.
- Integration Services.
- Búsqueda por texto completo.
- SQL Server Browser.

Algunos de estos servicios están asociados a la instancia de SQL Server que se haya instalado. Por ejemplo, es el caso de SQL Server Database Services y Agent SQL Server. Algunos servicios, como Reporting Services, están directamente asociados a servicios de uso particular de SQL Server, como es el caso para Business Intelligence.

Los servicios SQL Server Database Services y Agent SQL Server son los principales servicios que se van a estudiar en este libro.

Servicio	Nombre para la instancia por defecto	Nombre para otra instancia dada
Microsoft SQL Server	MSSQLSERVER	MSSQL$*NombreInstancia*
Agent SQL Server	SQLSERVERAGENT	SQLAgent$*NombreInstancia*

Cuenta Local del sistema y cuenta de usuario

Los programas en el lado servidor se ejecutan en forma de servicios. Como todos los servicios, para acceder a los recursos de la máquina, utilizan el contexto de una cuenta de usuario de dominio. Por defecto, los servicios se ejecutan en el contexto de la cuenta Local System. Esta cuenta permite obtener todos los recursos de la máquina local, pero no permite acceder a los recursos del dominio.

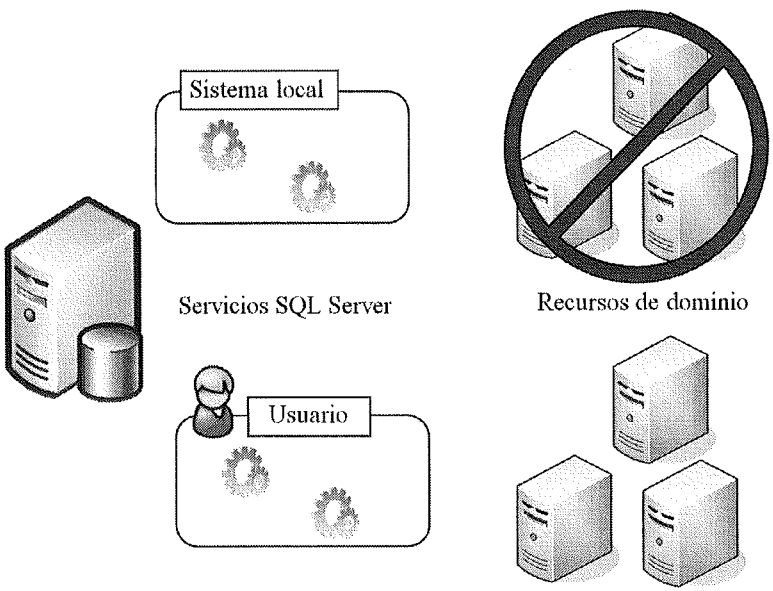
Los dos servicios MS SQL Server y SQL Server Agent deben poder acceder a los recursos del dominio con el objetivo de poder utilizar todas las funcionalidades que ofrece SQL Server (gestión de tareas planificadas, replicación...). En el momento de la instalación es posible precisar la cuenta de usuario de dominio que será utilizada por estos dos servicios.

Para simplificar las operaciones de gestión, se recomienda utilizar la misma cuenta de Windows para los dos servicios.

Observación

La misma cuenta también se podrá utilizar para el servicio de búsqueda por texto completo.

Si la empresa tiene varios servidores SQL en varios dominios, es preferible que todos los servicios SQL Server se ejecuten utilizando una cuenta de usuario de dominio con el mismo nombre y con la misma contraseña.

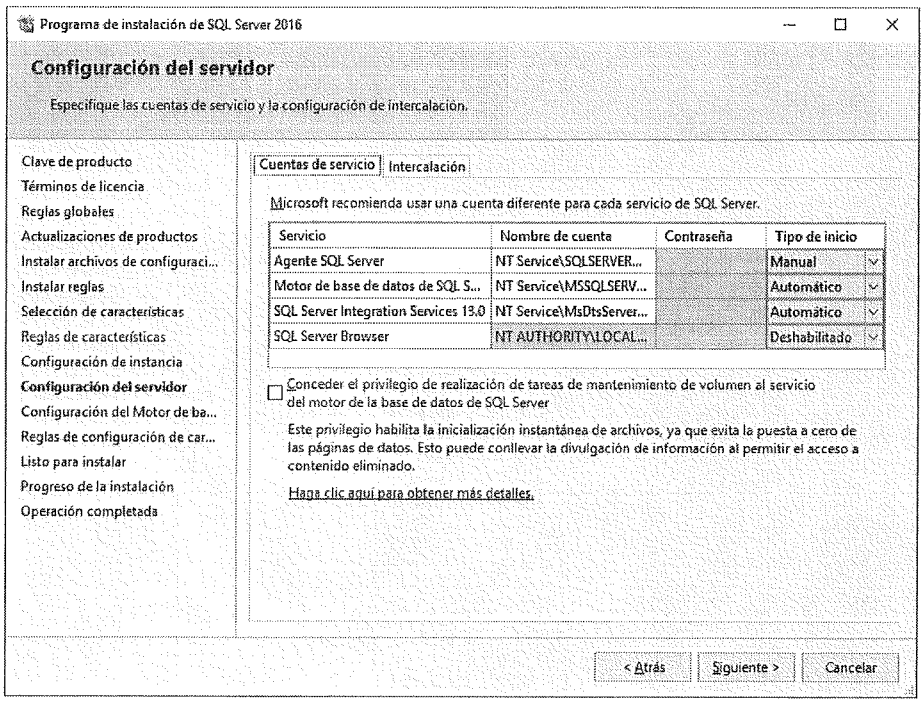


Cuenta de inicio de sesión

Inicio automático de los servicios

Es posible parametrizar los servicios de SQL Server de manera que se inicien automáticamente en el momento de arrancar Windows. La ventaja de esta opción es que no se precisa abrir una sesión en el puesto como administrador para lanzar los servicios del SGBDR. En el inicio de Windows, las bases de datos gestionadas por el SGBDR son accesibles inmediatamente.

Ambas opciones, **Cuentas de servicio** e **Inicio automático del servicio**, se pueden fijar en el momento de la instalación de SQL Server o bien una vez que SQL Server está instalado, a través de las herramientas de administración.



1.2.4 Parámetros de intercalación

El idioma por defecto de la instancia de SQL Server tiene una incidencia directa sobre la intercalación seleccionada.

Es posible instalar SQL Server independientemente del idioma definido a nivel del sistema operativo.

No hay que confundir la página de códigos y la intercalación.

La página de códigos es el sistema de codificación de caracteres seleccionados. La página de códigos permite identificar 256 caracteres diferentes. Existen numerosas páginas de códigos, teniendo en cuenta la diversidad de caracteres que se usan en uno u otro idioma. Para poder expresar datos en diferentes idiomas, es posible utilizar el sistema Unicode. Este sistema de codificación de caracteres usa dos bytes para codificar cada carácter. Por lo tanto, el sistema Unicode permite codificar 65.536 caracteres diferentes, lo que es suficiente para codificar todos los caracteres que se usan en los distintos idiomas occidentales.

Esta solución con páginas de códigos solo es válida si en la base de datos solo se almacenan datos en un único idioma. Con el aumento de las aplicaciones de comercio electrónico, cada vez es más normal que las bases de datos contengan información (como nombres, apellidos y dirección de los clientes) en diferentes idiomas. Para soportar los caracteres específicos de cada idioma, es necesario utilizar el tipo de datos Unicode. Los datos de tipo **Unicode** se guardan en los tipos **nchar** y **nvarchar**. La aplicación cliente, que es la que permite rellenar y visualizar la información, también debe soportar el tipo de datos Unicode.

Observación

El espacio necesario para el tipo de datos Unicode es dos veces mayor que para los datos no Unicode. Sin embargo, esta ligera desventaja se compensa por el ahorro de tiempo a la hora de visualizar los datos en el puesto cliente. De hecho, con el tipo de datos Unicode, ya no es necesario realizar un mapeo entre la página de códigos que se usa en la base de datos para almacenar la información y la página de códigos que se usa en el puesto cliente para visualizar la información.

La intercalación corresponde a un modelo binario de representación de los datos, que permite definir las reglas de comparación y de ordenación. Por ejemplo, la intercalación permite definir cómo se deben tener en cuenta los caracteres acentuados cuando se ejecutan operaciones de comparación u ordenación para el idioma español.

Por defecto, existen tres tipos de intercalación:

- Las intercalaciones Windows, que se basan en los parámetros de los idiomas definidos en Windows. Con este tipo de intercalación, las sentencias de ordenación y comparación se adaptan automáticamente al idioma del servidor.
- Las intercalaciones binarias son muy adecuadas por su rapidez de tratamiento. Se basan en el código binario que se usa para registrar cada carácter de información en formato unicode o no.
- Las intercalaciones SQL Server aseguran la compatibilidad ascendente con las versiones anteriores de SQL Server. Por lo tanto, es preferible no seleccionar esta opción en una instalación nueva.

Se debe indicar una intercalación cuando se crea la instancia. Esta se convertirá en la intercalación predeterminada de la instancia y las bases de datos **master**, **msdb**, **tempdb**, **model** y **distribution** la usarán.

Cuando se crean las bases de datos de usuario, se podrá indicar otra intercalación con la cláusula **COLLATE**.

Observación

Como las intercalaciones controlan el orden de las ordenaciones de los datos Unicode y no Unicode, no es posible definir sentencias de intercalación incompatibles.

El nombre de las diferentes sentencias de ordenación se estructura siempre de la misma manera: nombre del idioma, por ejemplo *Spanish*, seguido de varios pares de caracteres que representan las opciones de la sentencia de ordenación. Estas opciones son las siguientes:

_CS Es decir, *Case Sensitive*, por lo que es sensible a mayúsculas y minúsculas. Dicho de otra manera, la ordenación distingue entre mayúsculas y minúsculas.

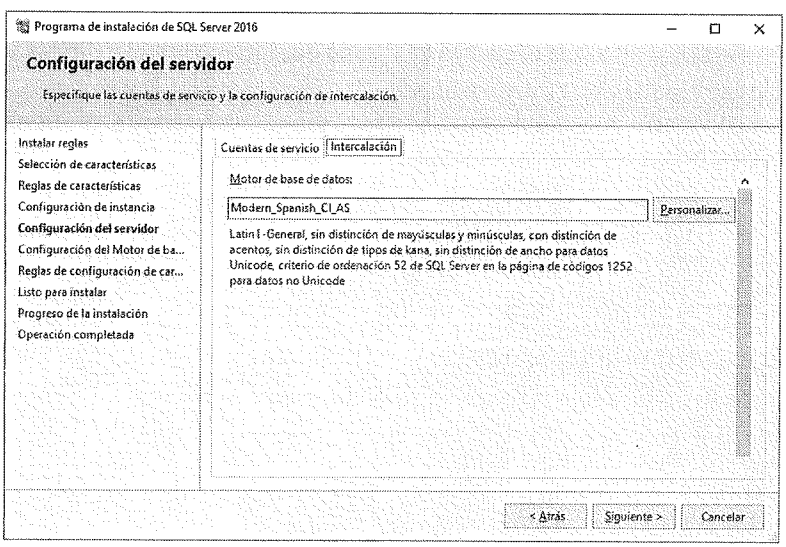
_AS Es decir, *Accent Sensitive*, por lo que se distingue entre caracteres acentuados o no.

_KS Es decir, *Kana Sensitive*, por lo que se mantiene la diferencia entre los juegos de caracteres japoneses.

_WS Es decir, *Width Sensitive*, por lo que la ordenación considera como diferentes el mismo carácter codificado en unicode (nchar o nvarchar), con 2 bytes, y el codificado con 1 byte, en código ASCII.

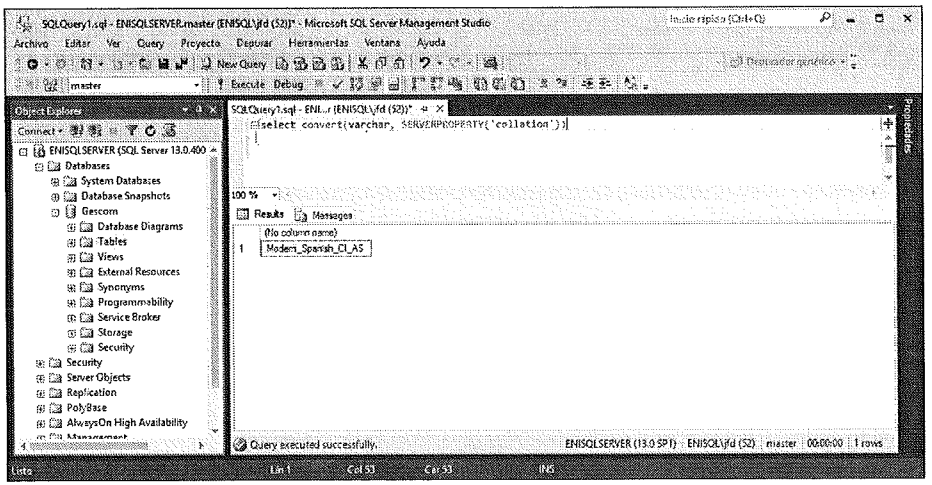
Observación

Estas opciones pueden aparecer con I en lugar de S. I significa Insensitive e indica que la ordenación no hace ninguna distinción respecto a ese criterio.

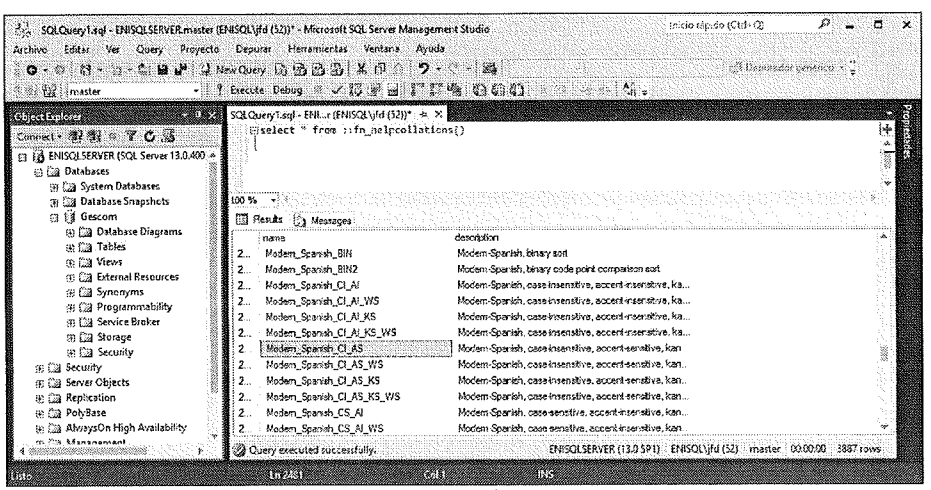


Fijar los parámetros de intercalación

Es posible conocer la intercalación del servidor con la función SERVERPROPERTY, como se muestra en el siguiente ejemplo.



También es posible conocer el conjunto de intercalaciones disponibles en el servidor con la función: fn_helpcollations().



Observación

CUIDADO: la diferenciación entre mayúsculas y minúsculas se aplica tanto a los identificadores como a los datos. Si se elige una sentencia de ordenación binaria respetando las mayúsculas/minúsculas, entonces todas las referencias a los objetos se deben hacer utilizando la misma combinación de mayúsculas/minúsculas que la especificada cuando se crearon estos objetos.

1.2.5 Modo de autenticación

La administración de las cuentas de usuario se puede basar completamente en las cuentas de usuario de Windows. También es posible definir cuentas de usuario y gestionarlas totalmente en SQL Server. En este caso, es necesario especificar la contraseña del usuario SQL Server que tendrá los privilegios de administrador SQL Server.

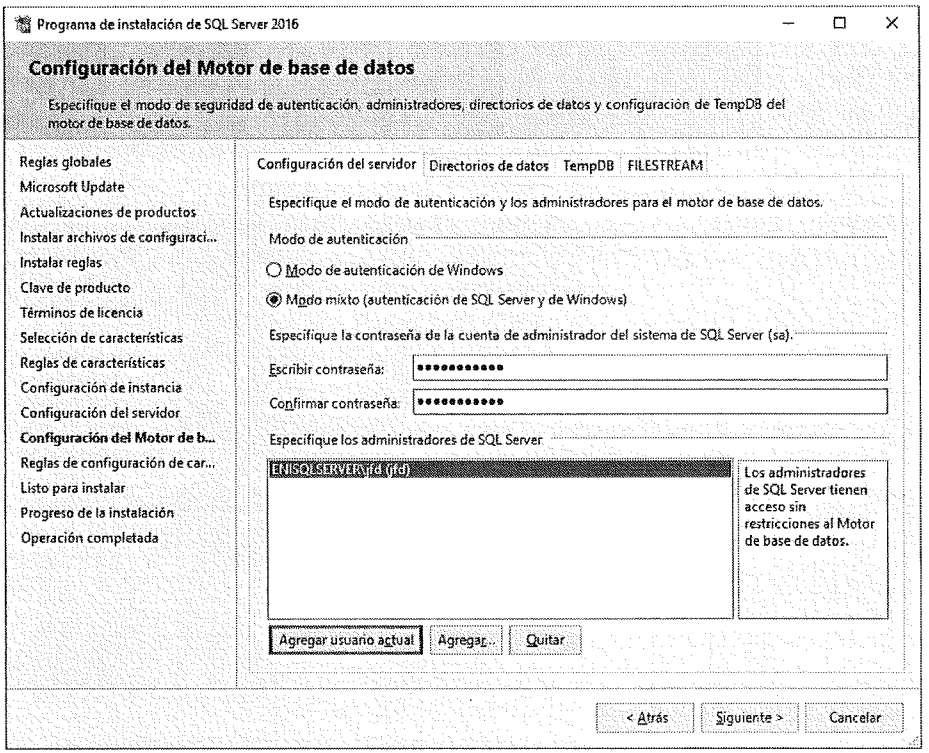
Es muy recomendable utilizar solo el modo de autenticación de Windows y recurrir al modo mixto solo cuando sea estrictamente necesario. En este último caso, es imprescindible especificar una contraseña para el usuario **sa** (administrador de la instancia de SQL Server que se está instalando).

Como se puede corregir este tipo de configuración después de la instalación de la instancia, en caso de dudas es mejor mantener únicamente el modo de autenticación de Windows.

1.2.6 Configuración del motor de base de datos

La configuración del motor de base de datos permite especificar el modo de seguridad elegido y el emplazamiento de los archivos de datos, así como la activación o no de la opción FILESTREAM.

Estas diferentes opciones se pueden personalizar usando las fichas correspondientes. Por ejemplo, para poder usar la opción FILESTREAM desde el mismo momento en que se termina la instalación, hay que activar esta opción en la ficha correspondiente.



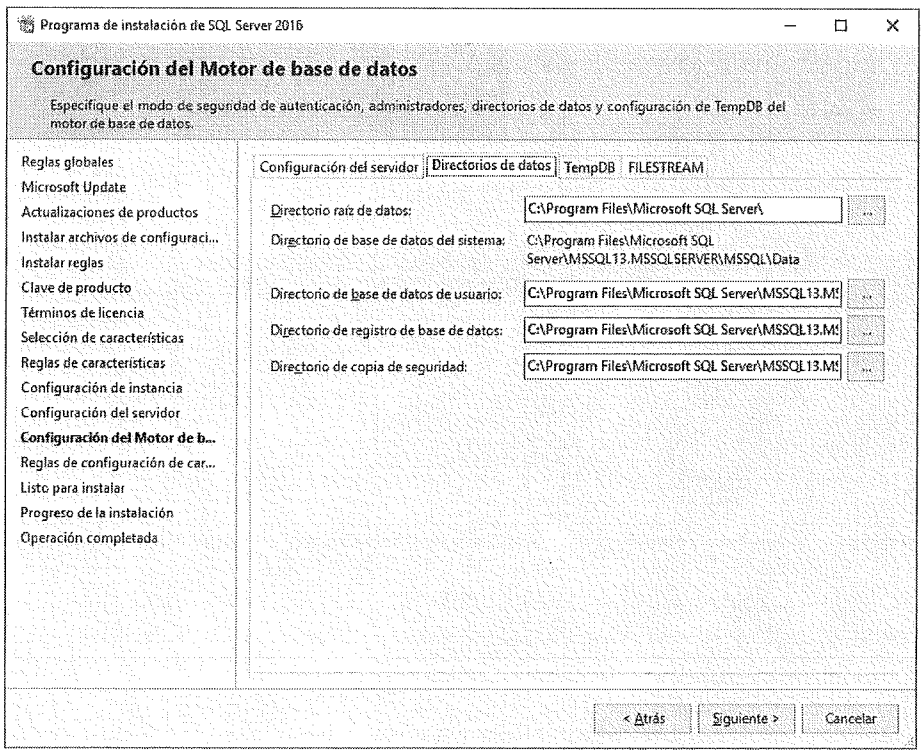
Estas diferentes opciones se pueden modificar/anular posteriormente en el momento de la configuración del servidor o cuando se gestionen los archivos de la base de datos.

Para una mayor seguridad a nivel de los usuarios, se recomienda utilizar el contexto de seguridad de Windows y prohibir la seguridad SQL Server. Esto evitará, por ejemplo, que los desarrolladores codifiquen «en duro» un nombre y una contraseña en una aplicación o en un archivo de configuración.

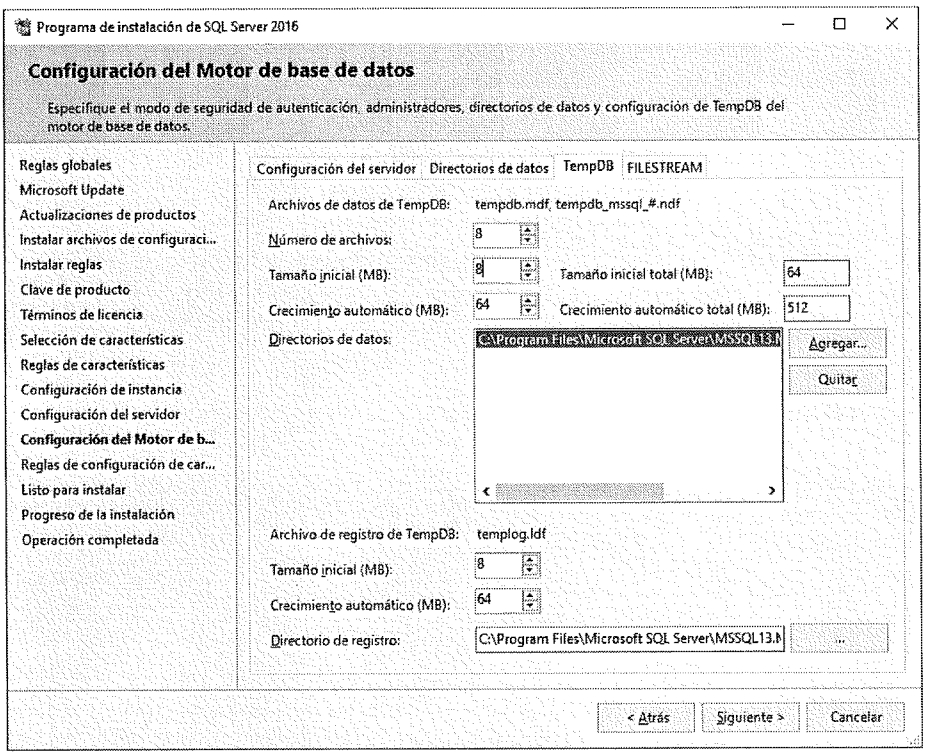
Observación

Para las instancias de SQL Express, se añade automáticamente una conexión que corresponde al grupo BUILTIN\Users. De esta manera, todos los usuarios del puesto de trabajo se pueden conectar a la instancia. Se puede eliminar esta conexión sin que esto suponga un problema respecto al correcto funcionamiento de la instancia.

También es a este nivel cuando es posible definir el directorio por defecto para los archivos de datos de las bases de datos de usuario y sistema. De hecho, el proceso de instalación que se propone por defecto es almacenar esta información en el directorio **Program Files**. Por tanto, es muy recomendable especificar en este lugar el directorio de sistema que va a contener los archivos de las bases de datos.



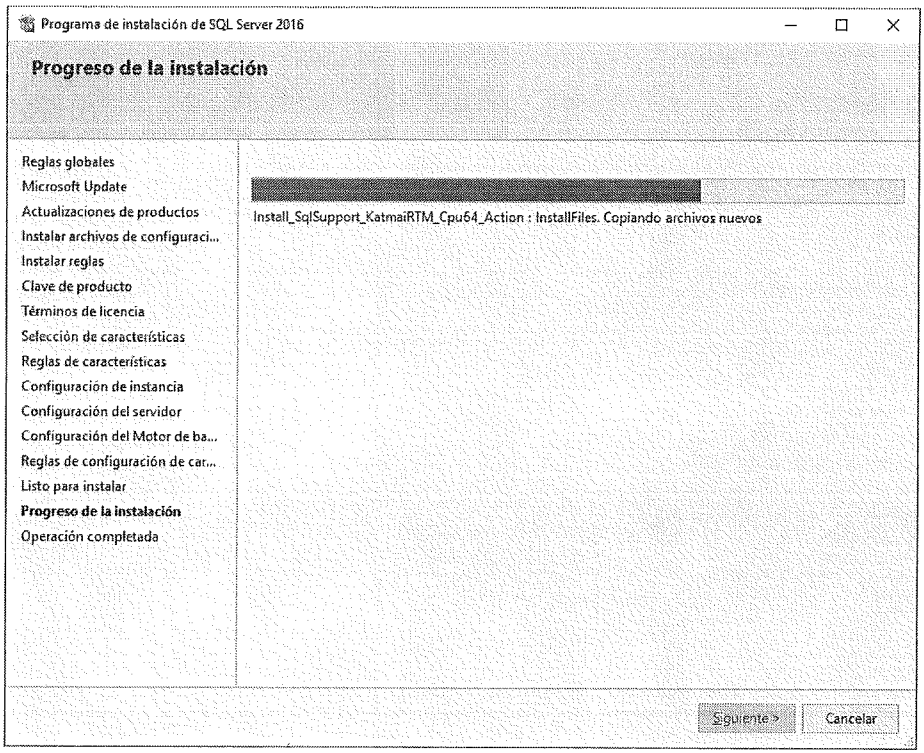
Llegados a este nivel también se puede configurar la base de datos tempdb. Por defecto, la base de datos tempdb está compuesta por tantos archivos como cores tengan los procesadores. Este número de archivos no puede, sin embargo, ser inferior a 8.



La opción **FILESTREAM** se puede aplicar fácilmente en una instancia ya instalada. Pero, evidentemente, si durante la instalación sabemos que esta opción se debe aplicar, entonces es mejor activarla en el proceso de instalación.

1.2.7 Resumen del proceso de instalación

Se verifican las reglas de instalación para comprobar que nada bloqueará el proceso de instalación. Después se vuelven a mostrar los elementos que se han seleccionado para instalarse. Tras validar esta última pantalla, empieza la instalación propiamente dicha.



1.3 Gestión de la red

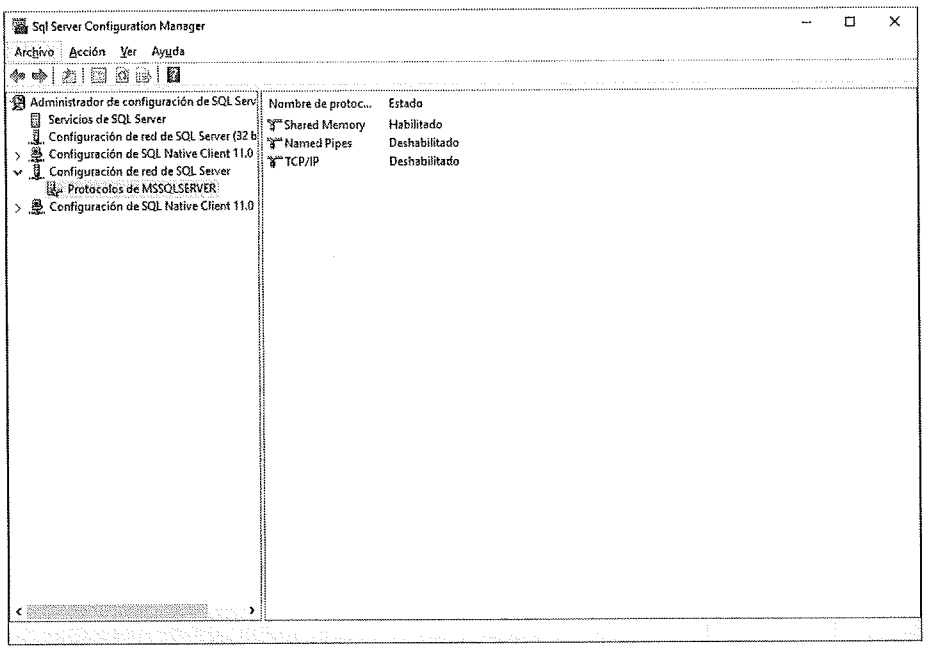
SQL Server utiliza las bibliotecas de red con el objetivo de asegurar la gestión de la transmisión de paquetes entre el servidor y el cliente. Estas bibliotecas de red existen en forma de DLL (*Dynamic Link Library*) y aportan todas las operaciones necesarias para establecer el diálogo entre el servidor y el cliente, incluso aunque estos dos procesos se encuentren en el mismo puesto.

La aplicación utiliza estas bibliotecas de red a través del mecanismo IPC o Comunicación Inter Proceso.

Un servidor puede escuchar al mismo tiempo varias bibliotecas y aceptar las peticiones provenientes de clientes con los que dialoga con protocolos de red distintos. El único requisito para que el servidor pueda responder a los clientes es que la biblioteca de red correspondiente a la del cliente esté instalada en el servidor.

Una vez las bibliotecas de red están instaladas en el servidor, es necesario configurar las net library para que el servidor pueda tenerlas en cuenta.

La gestión de la red entre el puesto cliente y el servidor pasa principalmente por TCP/IP. Es por este motivo por lo que la gestión de este protocolo se incluye por defecto en el momento de la instalación del servidor o de las herramientas clientes.



Bibliotecas disponibles:

Canales nombrados

Los canales nombrados están desactivados en todas las ediciones de SQL Server. Su uso se limita al diálogo entre las herramientas gráficas y el servicio SQL Server en el puesto servidor.

La instancia predeterminada abre el canal de comunicaciones que se llama \\.\pipe\sql\query.

Sockets TCP/IP (predeterminados)

Esta net library permite utilizar los sockets Windows tradicionales.

Para poder utilizar correctamente la net library TCP/IP, es importante indicar el número del puerto por el que responde SQL Server. Por defecto, se trata del puerto 1433, número oficial asignado por la IANA (*Internet Assigned Number Authority*) a Microsoft. También es posible utilizar un proxy. En este caso, hay que indicar la dirección del proxy cuando se configure la net library TCP/IP.

SQL Server usa el puerto 1433 si ninguna otra aplicación o proceso lo utiliza al mismo tiempo.

En algunos casos, como el acceso al servidor por medio de un firewall, se aconseja utilizar un puerto libre con un número inferior a 1024.

Observación

En caso de que SQL Server esté configurado para utilizar un puerto dinámico, el número del puerto puede cambiar en cada inicio de SQL Server.

El resto de las instancias diferentes a las predeterminadas utilizan un número de puerto dinámico. El servicio SQL Server Browser proporciona este número de puerto correspondiente a la instancia sobre la que queremos abrir una conexión.

1.4 Modos de licencia

El modo de licencia está directamente relacionado con la edición que se haya elegido y el modo de uso. Existen diferentes modos de licencia, tanto para el servidor asociado a las licencias de acceso cliente como para el procesador.

La edición Enterprise solo ofrece un modo de licencia por procesador, la edición Business Intelligence está disponible solo en modo de licencia por servidor, mientras que la edición Standard ofrece los dos modos.

Las licencias por procesador tienen en cuenta los núcleos del procesador, por lo que se trata de una licencia por núcleo que usa SQL Server. A nivel de licencia, no puede haber más de cuatro núcleos para este tipo de licencia.

La siguiente tabla resume las diferentes posibilidades de licencia respecto a la edición elegida:

	Edición		
	Enterprise	**Standard**	**Business Intelligence**
Licencia por núcleo	X	X	
Licencia por servidor		X	X

Cuando se utiliza una licencia de servidor, es necesario completarla con licencias de acceso cliente.

Cuando los servidores de base de datos están virtualizados, es necesario tener una licencia para cada servidor (por cada servidor virtual) y licencias de acceso cliente. Otra opción es tener una licencia por núcleo virtual. Esto permite aprovechar sin límite las posibilidades de virtualización de un servidor.

Observación

La gestión de licencias es un elemento sensible, por lo que es conveniente estar seguros de que la solución adoptada responde a las reglas dictadas por Microsoft. Más adelante se resumen estas reglas.

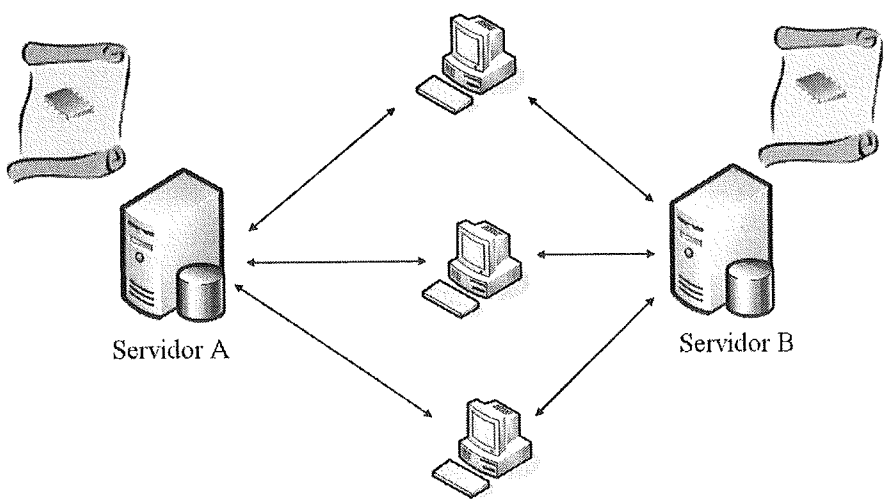
Licencia por núcleo

Con este modo de gestión de licencias, será necesario tener una licencia de SQL Server por cada uno de los procesadores del servidor que se desee. De esta manera, un número ilimitado de usuarios se pueden conectar al servidor sin que sea preciso tener una licencia SQL Server. En principio se usaba en las aplicaciones de tipo Internet/intranet. Ahora este modo de licencia también se utiliza con cualquier otro tipo de aplicación. Su principal ventaja consiste en una gestión simplificada de los derechos de uso.

Ejemplo: una empresa tiene un servidor SQL y 10 estaciones de trabajo que deben acceder a este servidor. Las estaciones de trabajo se reparten en 2 grupos de 5 puestos: trabajo de día y trabajo de noche. Los puestos de estos grupos solo se pueden conectar al servidor durante horarios predeterminados y no puede existir una superposición. ¿Cuántas licencias de acceso a SQL Server serán necesarias en modo de licencia por procesador?

Respuesta: solo se necesita una licencia de acceso. La gestiona el servidor y autoriza a las estaciones de trabajo a conectarse al mismo tiempo en el servidor SQL.

Para adquirir una licencia que cubra la totalidad del servidor, hay que adquirir licencias para cubrir los núcleos de procesador presentes en el servidor físico.



Modo de licencia por procesador

Licencia por usuario

Con este modo de gestión de licencias, cada usuario debe tener una licencia para trabajar con SQL Server. Este modo de gestión será interesante si los usuarios se pueden conectar a SQL Server usando diferentes herramientas.

El uso de un servidor de aplicaciones que gestiona en un momento determinado un conjunto de conexiones no reduce el número de licencias necesarias. Se necesita una licencia de acceso por cada usuario físico.

Una misma licencia de acceso cliente o CAL (*Cliente Access License*) permite acceder a múltiples servidores. El nivel de licencia del cliente debe corresponder al servidor más exigente.

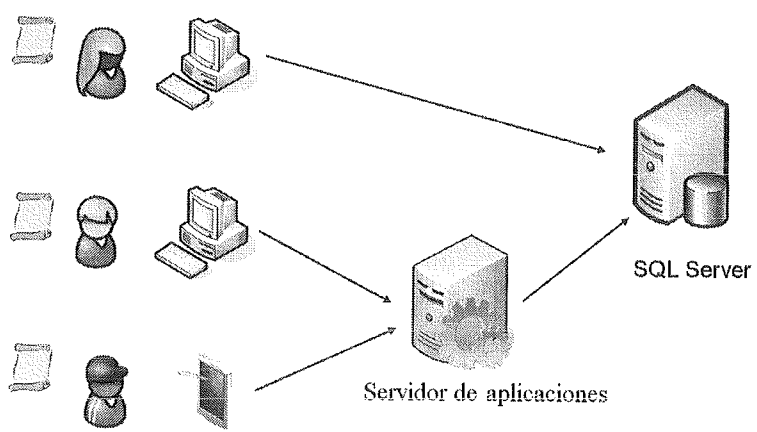
En el modo de gestión de licencias por usuario, además es necesario adquirir una licencia de tipo servidor para la máquina en la que está instalada la instancia de SQL Server.

Observación

Los usuarios indicados aquí son usuarios físicos. Todos ellos pueden utilizar el mismo usuario de base de datos, aunque no es recomendable.

Ejemplo

En el siguiente ejemplo, tres usuarios diferentes se conectan al servidor. Por lo tanto, esta solución necesita tres licencias de acceso cliente.



Una licencia de acceso a SQL Server 2016 también permite acceder a un servidor SQL Server 2014. Como es lógico, a la inversa no es válido.

Licencia por puesto

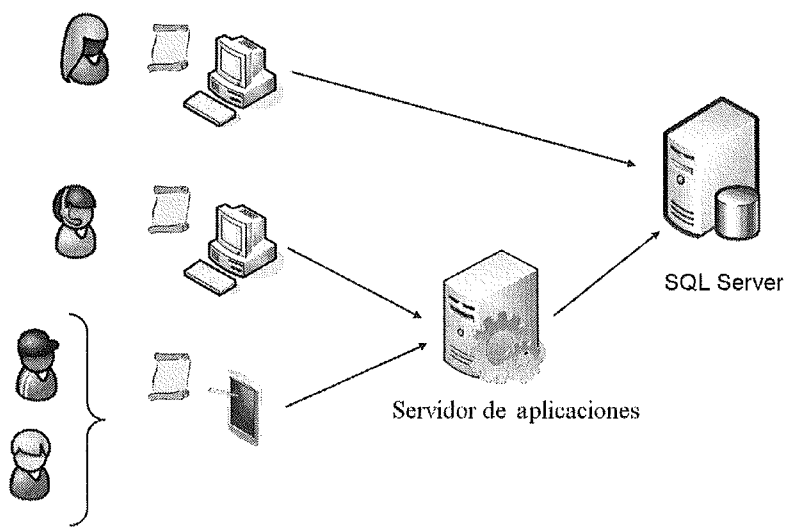
Con la licencia por puesto (o por asiento), cada periférico que establece una conexión a un servidor SQL Server necesita una licencia. Esta licencia no está asociada al número de usuarios potenciales que puedan utilizar el puesto. De esta manera, si varios usuarios comparten el mismo puesto de trabajo, pueden utilizar la misma licencia de acceso.

Un puesto equipado con esta licencia puede conectarse a varias instancias de SQL Server. La licencia debe ser compatible con la instancia de SQL Server más exigente.

En el modo de gestión de las licencias por puesto, además es necesario adquirir una licencia de tipo servidor para la máquina en la que está instalada la instancia de SQL Server.

Ejemplo

En el ejemplo siguiente, dos usuarios comparten el mismo periférico. Por lo tanto, es necesario disponer de tres licencias de acceso por puesto.



1.5 SQL Server y la virtualización

En SQL Server, existen dos casos distintos de virtualización. El primero consiste en virtualizar los servidores, y en este caso hay que tener licencias para cada máquina virtual. El otro caso en que se puede usar la virtualización corresponde a los entornos de tipo cloud privado. Aquí solo se aborda el primer caso.

En el caso de que los servidores SQL Server estén virtualizados para implementar una gestión más sencilla de las máquinas, se aplica el mismo modo de gestión de licencias que el que se ha presentado para los servidores físicos. En el caso de una licencia por núcleo, todos los núcleos implicados en la máquina virtual deben tener una licencia SQL Server. En el caso de las licencias de acceso, no cambia nada entre la máquina física y la virtual.

1.6 Ejecutar el programa de instalación

El programa de instalación situado en el DVD de SQL Server se ejecuta automáticamente cuando se inserta el DVD en la máquina. Si el procedimiento de instalación no se inicia automáticamente, conviene ejecutar el programa SETUP.EXE situado en la raíz del DVD.

Instalación automática

Es posible realizar una instalación automática.

El procedimiento de instalación automática pasa por el uso de la línea de comandos y permite indicar en la línea de comandos los valores de los diferentes argumentos. Aunque la definición de esta línea de comandos puede ser relativamente complicada y laboriosa, en caso de que haya instalaciones SQL Server que se deban hacer de manera repetitiva o automática, el ahorro de tiempo es importante.

Sin embargo, durante una instalación de SQL Server en modo interactivo, todas las elecciones hechas a nivel de las opciones de instalación están guardadas en un archivo de configuración llamadó ConfigurationFile.ini, en el log de la instalación (para SQL Server 2016 en C:\Program Files\Microsoft SQL Server\130\Setup Bootstrap\Log) y en el subdirectorio correspondiente al día de la instalación. Si el proceso de instalación se ejecuta varias veces en el mismo día, es posible distinguir los directorios gracias a la hora y los minutos.

Este modo de funcionamiento para una instalación interactiva genera un archivo de argumentos que permite configurar rápidamente una instalación automática.

Observación

Independientemente del modo de instalación seleccionado, siempre es necesario aceptar el contrato de licencia, salvo algunas excepciones, como, por ejemplo, si usa un modo de licencia por volumen.

Sintaxis

```
setup.exe /settings nombreCompletoArchivo.ini [/qn] [/qb]
```

`nombreCompletoArchivo.ini`

Ruta absoluta y nombre del archivo de parametrización.

`/qn`

Conmutador para hacer una instalación completamente silenciosa (sin ningún cuadro de diálogo).

`/qb`

Conmutador para hacer una instalación visualizando únicamente los cuadros de diálogo que permiten efectuar un seguimiento de la progresión de la instalación.

Las opciones del programa **setup.exe** van a ser diferentes en función del tipo de instalación deseada. Por ejemplo, en el caso de que SQL Server se instale en una imagen para un desarrollo, hay opciones particulares relativas a la preparación de la imagen (sysprep).

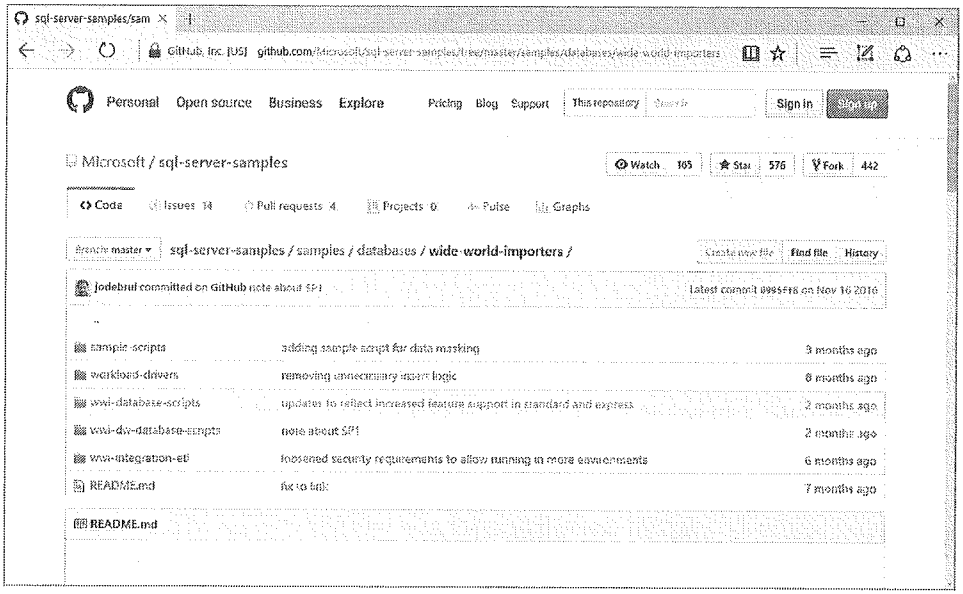
1.7 Las bases de datos de ejemplo

En el momento de la instalación del motor de base de datos, no se establece ninguna base de datos predeterminada de ejemplo. De hecho, en un servidor de producción, las bases de datos de ejemplo no son necesarias y solo pueden introducir fallos de seguridad. Sin embargo, en un servidor de pruebas o de formación, estas bases de datos de ejemplo sí tienen sentido.

Para SQL Server, Microsoft propone una base de datos de ejemplo llamada Wide World Importers. En esta base de datos se estructuran los ejemplos presentados en la ayuda en línea. Encontrará asimismo numerosos ejemplos basados en la otra base de datos de ejemplo de SQL Server: Adventure Works.

Para descargar la base de datos de ejemplo Wide World Importers, debe acceder a GitHub:
https://github.com/Microsoft/sql-server-samples/tree/master/samples/databases/wide-world-importers.

En realidad existen tres versiones de la base de datos de ejemplo. Una versión correspondiente a una base de datos OLTP (este tipo de bases de datos es la que se utiliza en este libro) llamada wwi, una versión destinada a la parte decisional llamada wwi-dw y una versión para las herramientas de importación llamada wwi-etl.



Adicionalmente a estas bases de datos de ejemplo, también es posible descargar e instalar muchos otros ejemplos relacionados con el uso correcto de SQL Server. La mayor parte de ellos están escritos en Transact SQL.

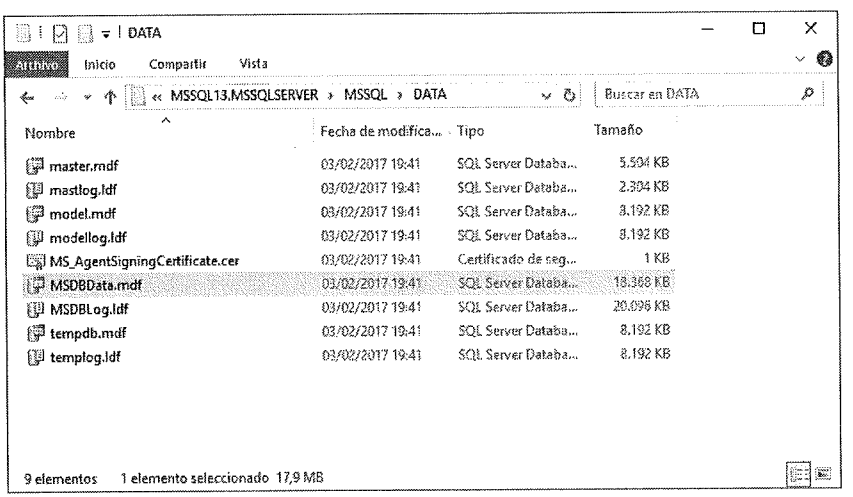
Por defecto, y para permitir el nivel de seguridad más alto posible e impedir las conexiones anónimas, no se define la cuenta huésped (guest). En un servidor de pruebas, algunas veces puede resultar interesante autorizar estas conexiones anónimas para permitir a los usuarios acceder libremente a esta base de datos de ejemplo.

2. Comprobación de la instalación

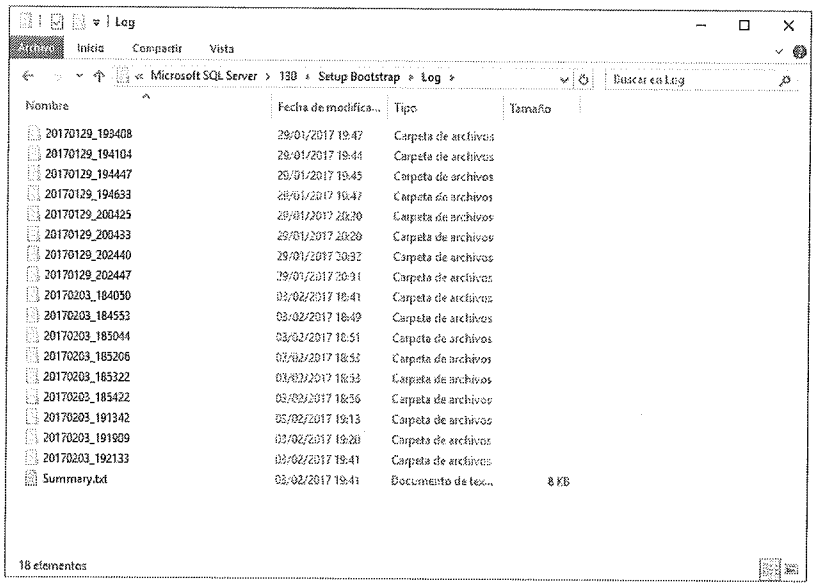
Como para todos los productos, es importante asegurarse de que la instalación se ha ejecutado correctamente y de que el servidor está operativo.

2.1 Verificar los elementos instalados

Con el explorador de archivos se puede verificar si está bien definida la arborescencia que reagrupa todos los archivos necesarios para el correcto funcionamiento del motor de base de datos. Si la instalación se ha hecho utilizando los parámetros predeterminados, esta arborescencia es c:\Program Files\Microsoft SQL Server\130. Los archivos de datos se encuentran en el directorio c:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA, excepto si se ha definido otra ruta durante la instalación.



A nivel de archivos, se puede consultar el diario de instalación (summary.txt), que está en el directorio c:\Program Files\Microsoft SQL Server\130\Setup Bootstrap\LOG. La documentación es útil cuando el proceso de ejecución termina con algún error. Así, se podrá entender mejor el origen del problema. Este archivo de resumen se gestiona por cada ejecución del programa de instalación. Una nueva ejecución del programa de instalación va a renombrar (con fecha y hora) el archivo resumen existente para conservar la traza de todas las instalaciones.



2.2 Verificar el arranque de los servicios

Para un uso clásico de la base, los dos servicios MS SQL Server y SQL Server Agent deben estar iniciados y en posición de inicio automático. El servicio MS SQL Server representa el motor de la base de datos y, mientras este servicio no esté iniciado, es imposible conectarse al servidor y trabajar con los datos que contiene. El servicio SQL Server Agent toma a su cargo, entre otras, la ejecución y la gestión de todas las tareas planificadas. Para gestionar los servicios, es posible hacerlo con las herramientas propuestas de manera estándar por Windows, aunque SQL Server ofrece también sus propias herramientas.

3. Las herramientas

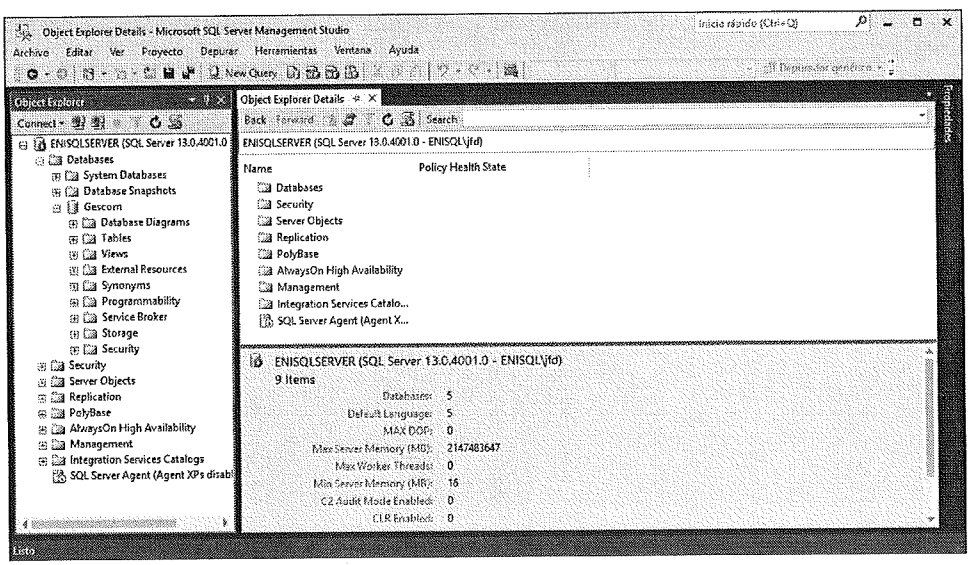
SQL Server tiene muchas herramientas. Estas son complementarias y cada una se adapta a un tipo de problema o acción. Es importante tener una idea del objetivo de cada una para saber cuál emplear a la hora de resolver un problema determinado. Es posible hacer una analogía con el bricolaje: si solo tiene un destornillador, intentará hacerlo todo con él. Si su caja de herramientas está bien surtida, hay muchas posibilidades de encontrar la herramienta que soluciona exactamente el problema que tenga.

En SQL Server el enfoque es similar. Prácticamente se puede hacer todo en Transact SQL, aunque SQL Server ofrece herramientas gráficas que permiten solucionar problemas precisos. El uso de la mayoría de ellas se detalla a lo largo de este libro, pero un vistazo general permite tener una mejor visión del interés de cada una.

SQL Server Management Studio

Se trata de la herramienta principal de SQL Server y está destinada a los desarrolladores y administradores.

SQL Server Management Studio (SSMS) es la consola gráfica de administración de las instancias de SQL Server. Con esta herramienta es posible administrar varias instancias locales o remotas. SQL Server Management Studio también es la herramienta principal de los desarrolladores de bases de datos, que la usan para definir scripts de creación de tablas, vistas, procedimientos, funciones, triggers de base de datos...



Observación

Esta herramienta se puede iniciar desde línea de comandos con la aplicación ***ssms***.

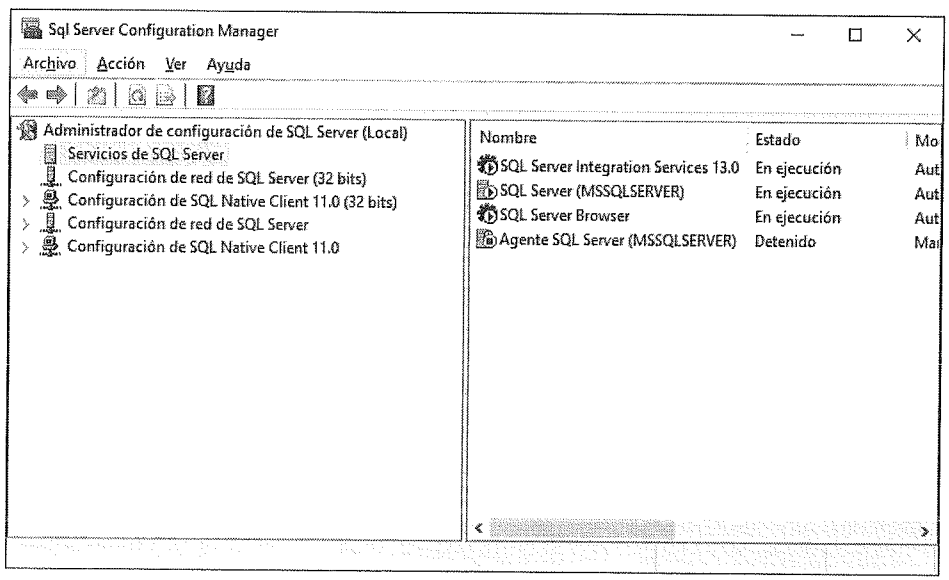
SSMS ofrece un editor de scripts SQL y Transact SQL evolucionado, que integra la asignación sintáctica de colores para distinguir claramente las palabras clave, las cadenas de caracteres, los comentarios, etc. SSMS también ofrece la funcionalidad de autocompleción, que permite completar, de manera automática, las palabras clave cuando se escribe la consulta.

Esta funcionalidad de autocompleción se activa con la tradicional combinación de teclas [Ctrl][Espacio]. Además de las palabras claves, gracias a la esta funcionalidad están disponibles las referencias a las tablas, columnas, nombres de procedimientos y funciones. Esta funcionalidad permite ganar un tiempo precioso cuando se escriben scripts, limitando los errores en la codificación.

Para facilitar la gestión correcta de las bases de datos, SQL Server Management Studio ofrece la generación de informes. Estos informes permiten tener una vista general y sintética de uno o varios elementos de la base o del servidor. SQL Server 2016 ofrece un cierto número de informes predefinidos, aunque puede definir sus propios informes de manera adicional.

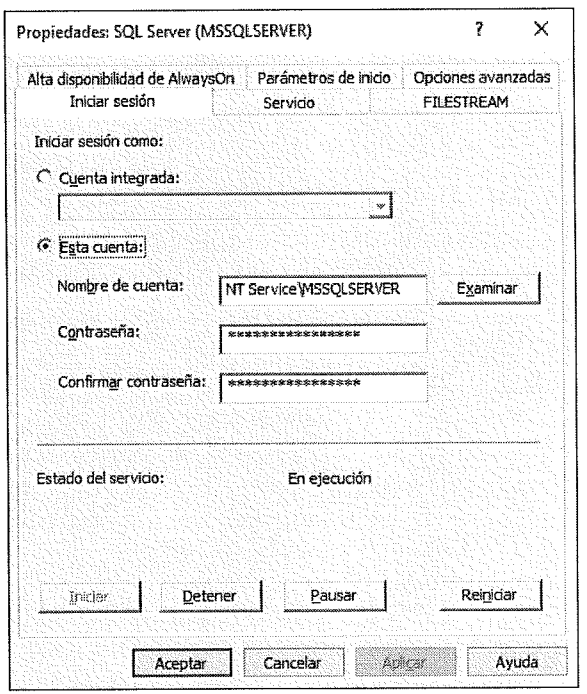
Administrador de configuración SQL Server

El administrador de configuración de SQL Server permite gestionar el conjunto de los elementos relativos a la configuración de los servicios y de la red del lado cliente y del lado servidor.



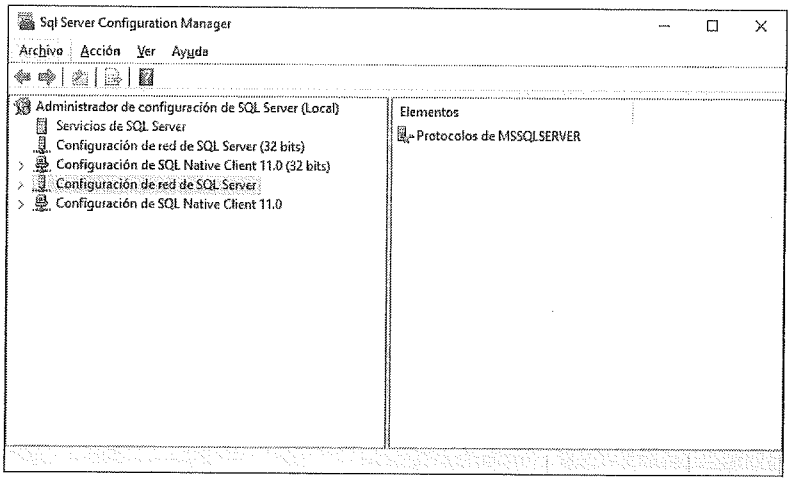
La configuración de los servicios

Los diferentes servicios relativos a SQL Server se pueden administrar directamente desde esta herramienta. Además de las operaciones clásicas de inicio y parada, es posible configurar el tipo de inicio (automático, manual, desactivado), así como la cuenta de seguridad en la cual se debe ejecutar el servicio.



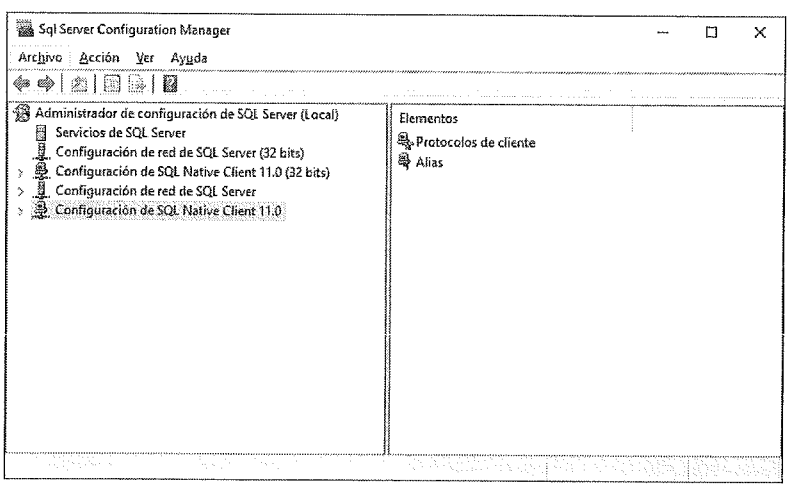
Configuración de la red SQL Server

El administrador de configuración permite gestionar cuáles son los protocolos que se tienen a nivel del servidor. También es posible a este nivel modificar las propiedades específicas de cada protocolo, como el número de puerto de escucha del protocolo TCP/IP.



Configuración de SQL Native Client

Establece la configuración de las herramientas clientes instaladas de forma local y, más exactamente, define de manera concreta los protocolos que tiene a su disposición para comunicarse con el servidor. Además, cuando sea necesario, se pueden definir alias. Esta funcionalidad es particularmente interesante cuando el nombre del servidor está registrado en una aplicación y no es posible, o sencillo, modificarlo. La definición de un alias permite redirigir todas las peticiones hacia un servidor distinto.



Asistente de parametrización de base de datos

Este asistente permite, entre otras cosas, a partir de una carga de trabajo capturada con SQL Profiler, validar o no la estructura de la base de datos. Como resultado del análisis, el asistente va a aconsejar la creación/eliminación de índices o bien la partición de tablas con el objetivo de obtener ganancias en términos de rendimiento.

SQLCmd

SQLCmd es una herramienta en línea de comandos que permite ejecutar los scripts SQL. Esta herramienta va a utilizarse en el momento de la solicitud de ejecución de tareas administrativas relativas a SQL Server a partir de Windows.

Esta herramienta permite conectarse a una instancia local o no de SQL Server. La autenticación de esta instancia puede efectuarse basándose en el modo de seguridad de Windows o bien en SQL Server.

SQLCmd permite ejecutar los scripts SQL que contienen instrucciones tanto de DML (*Data Manipulation Language*) como de DDL (*Data Definition Language*).

osql

Es otra herramienta en línea de comandos para ejecutar scripts. Esta herramienta se mantiene por razones de compatibilidad, aunque, dado que se apoya en tecnología odbc, probablemente desaparecerá. Ahora es necesario utilizar `sqlcmd`.

bcp

Se trata de una herramienta en línea de comandos que permite extraer fácil y rápidamente los datos desde la base y llevarlos a un archivo o bien hacer la operación inversa.

sqlps

Permite ejecutar el entorno PowerShell específico de SQL Server. Esta herramienta todavía está presente por razones de compatibilidad, pero ahora se recomienda integrar las extensiones PowerShell para SQL Server en el entorno PowerShell del servidor Windows.

sqldiag

Esta herramienta de diagnóstico puede ejecutarse con el objetivo de proporcionar información al servicio de soporte.

sqllogship

Esta aplicación permite preparar un archivo diario como copia de seguridad antes de enviarlo hacia otra instancia de SQL Server.

Tablediff

Como su nombre indica, esta herramienta permite comparar el contenido de dos tablas. Es muy útil para responder a problemas de sincronización que puedan aparecer en una replicación de fusión.

Dta

Esta herramienta en línea de comandos corresponde al asistente de parametrización del motor de base de datos, que permite ejecutar esta herramienta desde los scripts.

SqlLocalDB

Esta herramienta permite crear una instancia de SQL Express LocaDB. Es una herramienta en línea de comandos que permite a los desarrolladores crear una instancia de SQL Express utilizando un script.

sqlagent90

Esta herramienta corresponde al servicio SQL Server Agent. Sin embargo, este servicio solo se debería ejecutar como aplicación cuando existan problemas de actualización.

sqlmaint

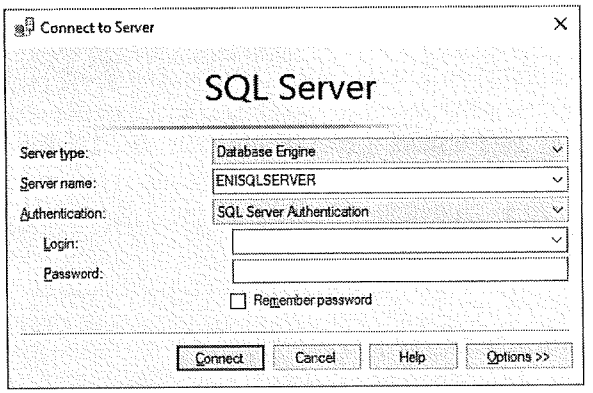
Esta herramienta permite crear planes de mantenimiento. Solo se conserva en SQL Server por razones de compatibilidad. En SQL Server 2014, los planes de mantenimiento se crean con Integration Services y es Agent SQL Server el responsable de la correcta ejecución de las tareas planificadas.

sqlservr

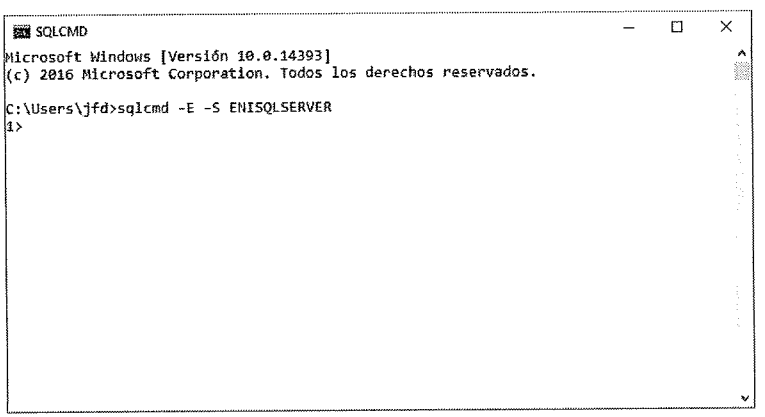
Normalmente esta herramienta no se conoce demasiado. Permite iniciar SQL Server no como un servicio, sino como una aplicación. Como aplicación en línea de comandos, existen numerosas opciones que permiten definir el conjunto de parámetros activos cuando se inicia la aplicación. Por ejemplo, la opción **-f** permite iniciar SQL Server con una configuración mínima, es decir, ignorando la configuración que se haya definido. En algunas ocasiones, este tipo de inicio en modo mínimo permite retomar el control después de una elección de configuración no deseada.

Utilizar una herramienta cliente para conectarse a SQL Server

Para establecer la primera conexión al servidor, hay varias herramientas. En modo gráfico, conviene lanzar SQL Server Management Studio. Cuando se ejecuta la herramienta, aparece la siguiente ventana de conexión.



Es posible establecer la conexión desde una herramienta en línea de comandos, como sqlcmd. La pantalla siguiente permite conectarse al servidor utilizando el modo de seguridad de Windows.



4. La configuración

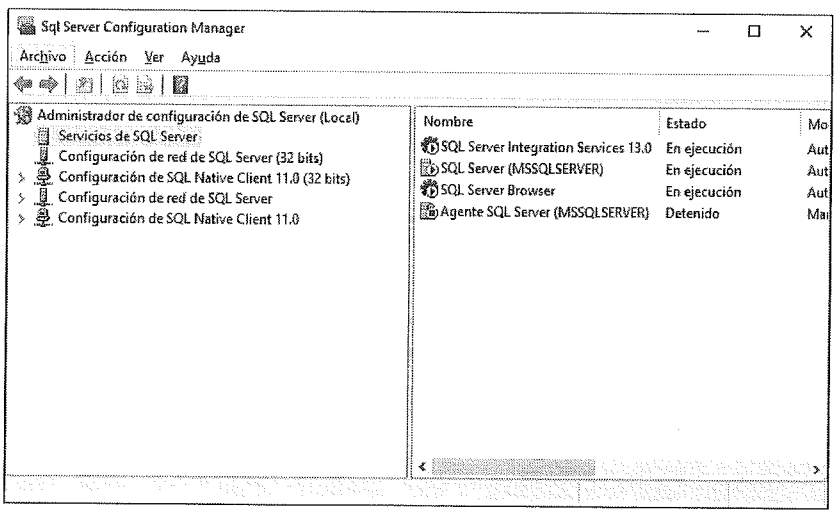
Antes de poner en funcionamiento el servicio del servidor SQL y hacerlo accesible para todos los usuarios, es importante realizar cierto número de operaciones de configuración del servidor y de las herramientas de administración cliente con el objetivo de protegerse contra cualquier operación delicada.

4.1 Los servicios

Los diferentes componentes del servidor se ejecutan en forma de servicios. Por lo tanto, es necesario que estos servicios sean iniciados para poder trabajar con el servidor. Estos servicios se pueden gestionar con el administrador de configuración de SQL Server, aunque también se pueden gestionar como todos los servicios de Windows.

Desde el administrador de configuración, es sencillo visualizar el estado del servicio y modificar sus propiedades.

El administrador de configuraciones también se identifica con el término SQL Server Configuration Manager. SQL Server Configuration Manager no es una aplicación, sino un componente de la consola mmc. Se puede arrancar también SQL Server Configuration Manager ejecutando SQLServerManagement13.msc.



Como todos los servicios de Windows, se pueden gestionar de manera centralizada en el servidor de Windows.

Por último, es posible actuar sobre estos servicios directamente en línea de comandos por medio de los comandos **net start** y **net stop**. En el momento de un inicio por medio de línea de comandos, es posible anular la configuración por defecto del servicio especificando la configuración que se debe utilizar en forma de parámetros. Por ejemplo, la opción m (net start mssqlserver m) permite iniciar el servidor en modo monousuario.

En caso de problemas de inicio, es posible iniciar el servidor SQL Server como una aplicación con la ayuda de sqlservr.exe. La utilización de este ejecutable permite iniciar la instancia sin tener en cuenta todas las opciones de configuración definidas.

Los diferentes estados de los servicios

Iniciado

Cuando el servicio MSSQL Server está iniciado, los usuarios pueden establecer nuevas conexiones y trabajar con los datos contenidos en la base. Cuando el servicio SQL Server Agent está iniciado, el conjunto de tareas planificadas, alertas y replicación está activo.

Suspendido

Si el servicio MSSQL Server está suspendido, el usuario no puede establecer ninguna conexión con el servidor. Los usuarios conectados no se ven afectados por esta medida. La suspensión del servicio SQL Server Agent desactiva la planificación de todas las tareas y las alertas.

Detenido

La parada del servicio MSSQL Server desactiva todas las conexiones de usuario y lanza un proceso de CHECKPOINT (el conjunto de los datos validados presentes en memoria persisten en el disco duro y se inscribe el punto de sincronización en el diario). Este mecanismo permite asegurar que el próximo inicio del servidor será correcto. Sin embargo, el servicio espera a que terminen todas las instrucciones en curso de ejecución antes de detener el servidor. La parada del servicio SQL Server Agent desactiva la ejecución planificada de todas las tareas y la gestión de las alertas.

4.2 SQL Server Management Studio

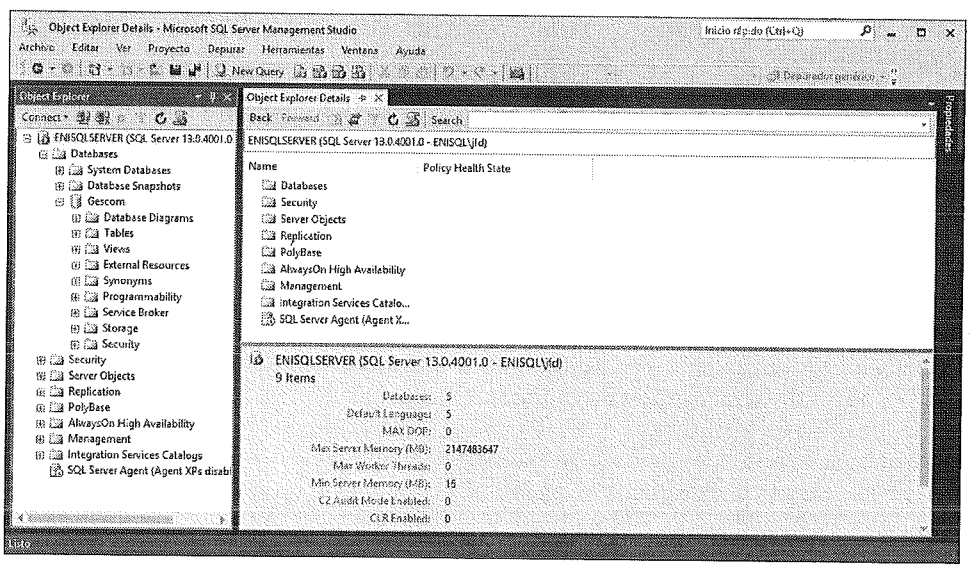
SQL Server Management Studio es la herramienta de gestión gráfica de SQL Server que permite realizar las tareas administrativas y todas las operaciones de desarrollo. La utilización de la misma herramienta permite reducir la distinción entre los grupos de usuarios, administradores y desarrolladores. Al compartir la misma herramienta, es más fácil saber lo que se puede hacer de otra manera.

Para poder navegar de una instancia a otra de SQL Server, posiblemente con servidores diferentes, es necesario registrar cada servidor en la consola de administración. Este registro no es necesario para la instancia local de SQL Server, ya que en el momento de crear la instancia la información relativa a esta instancia se añade a SQL Server Management Studio.

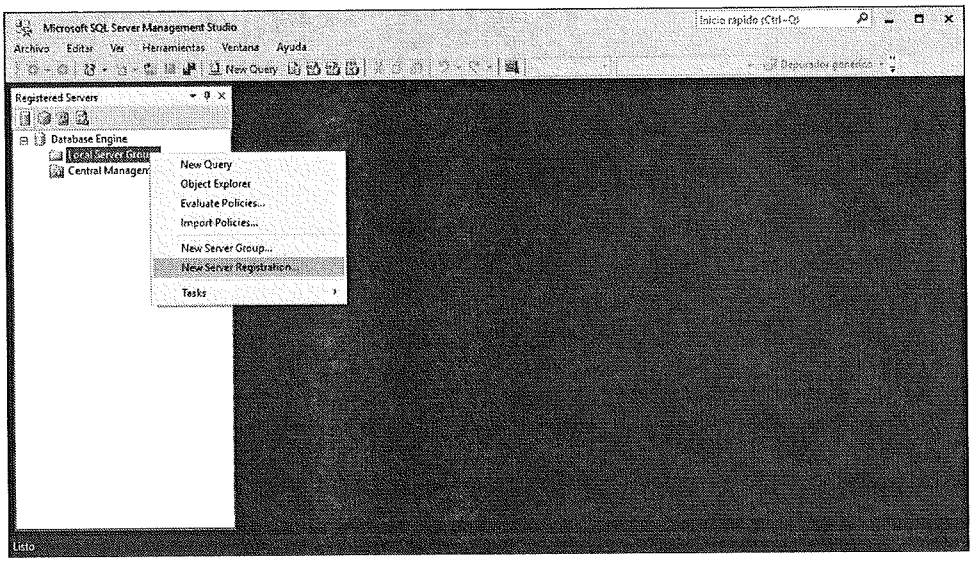
Registrar un servidor

La ventana **Registered Servers** permite conocer la lista de servidores registrados en SQL Server Management Studio. Si no se ve esta ventana, es posible iniciarla mediante la opción de menú **Ver - Registered Servers** o con la combinación de teclas [Ctrl][Alt] **G**.

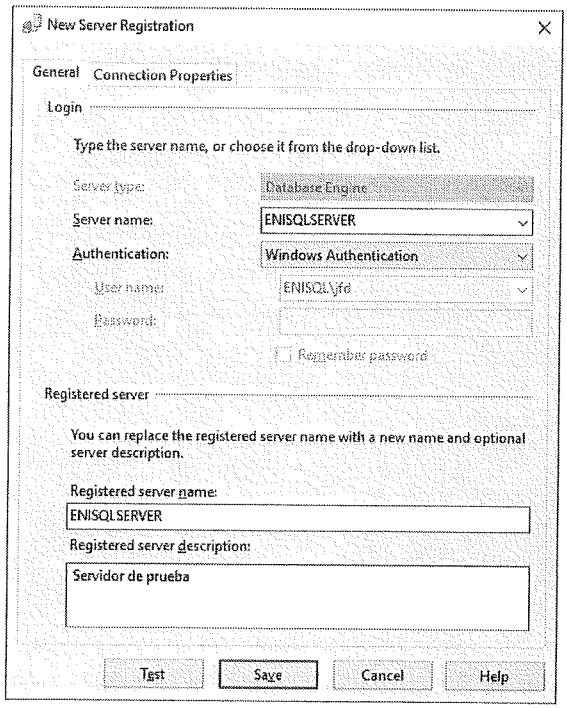
Los servidores se agrupan por tipos. Para cada tipo se pueden definir grupos de servidores con objeto de agruparlos por algún otro criterio, por ejemplo la ubicación física. Los grupos de servidores no tienen ninguna influencia en el registro del servidor. Es posible cambiar de lugar un servidor y situarlo en otro grupo con la opción **Tasks - Move to...**, desde el menú contextual asociado al servidor. Por analogía, es posible comparar los grupos de servidores con los directorios y los servidores registrados con los archivos. Los archivos no se ven afectados cuando se desplazan de un directorio a otro. Sucede lo mismo con el registro de servidores. Los directorios se definen para reagrupar los archivos siguiendo una lógica; es exactamente el papel que juegan los grupos de servidores.



Para registrar un servidor nuevo utilizando SQL Server Management Studio, es necesario seleccionar la opción **New Server Registration**, desde el menú contextual asociado al nodo **Local Server Groups** de la ventana **Registered Servers**.



El cuadro de diálogo que permite realizar el registro tiene dos fichas. La primera permite completar toda la información general de registro, como el nombre del servidor o el tipo de autenticación que se usa para establecer la conexión con el servidor.



El botón **Test** sirve para garantizar que la conexión elegida permite trabajar correctamente con el servidor seleccionado.

Observación

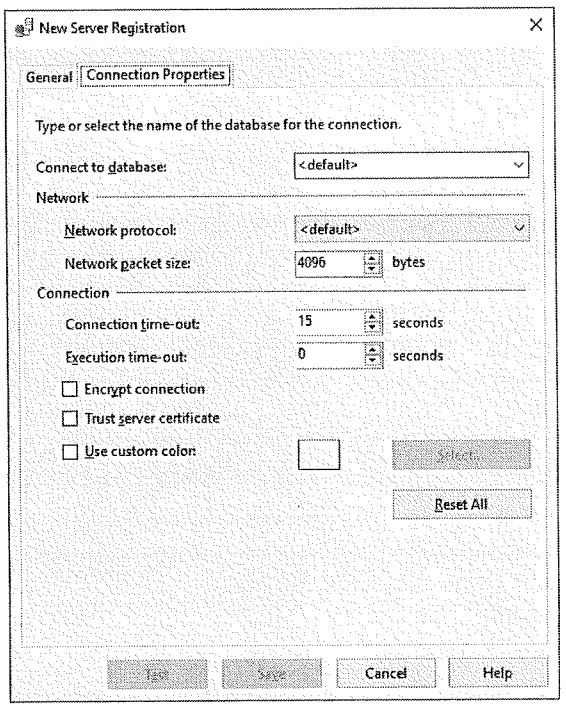
Es posible registrar un servidor con un nombre diferente al que tiene dicho servidor.

Respecto a la segunda ficha, esta permite fijar las opciones más avanzadas, como la base de datos por defecto o el tipo de protocolo de red que se usa para establecer la conexión con el servidor.

Observación

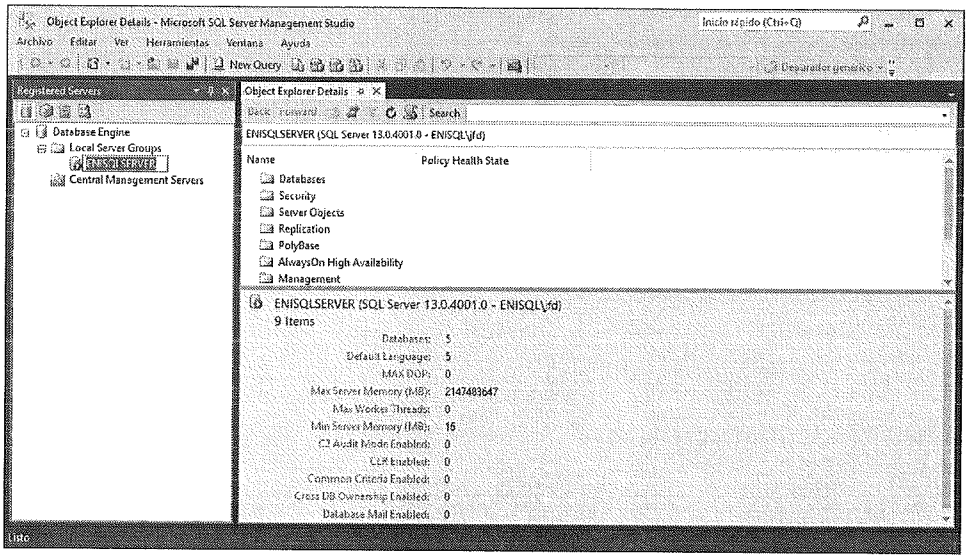
Por razones de seguridad, es preferible elegir, siempre que sea posible, el modo de autenticación de Windows.

Cuando se guarda el servidor, se pueden seleccionar algunas opciones, tales como el tiempo de expiración de la conexión.



Las bases de datos de sistema se agrupan en un directorio para facilitar el trabajo del administrador. De esta manera, las bases de datos de los usuarios son las que se visualizan en primer lugar. Esta separación permite no desordenar la consola con las bases del sistema, que no son importantes más que para SQL Server.

Para una visualización más funcional, puede usar el explorador de objetos presente en la parte izquierda de SSMS. Si el explorador de objetos no está, es posible verlo con el menú **Ver - Object Explorer** o la tecla [F8].



El mismo tipo de separación se efectúa en las bases de datos entre las tablas de sistema y las tablas creadas por los usuarios. Estas últimas son las que contienen la información y en las que el administrador debe volcar todos sus esfuerzos.

4.3 Configuración del servidor

Antes de abrir completamente el acceso al servidor y permitir a los usuarios trabajar con él, conviene vigilar atentamente los dos puntos siguientes:

Contraseña del administrador

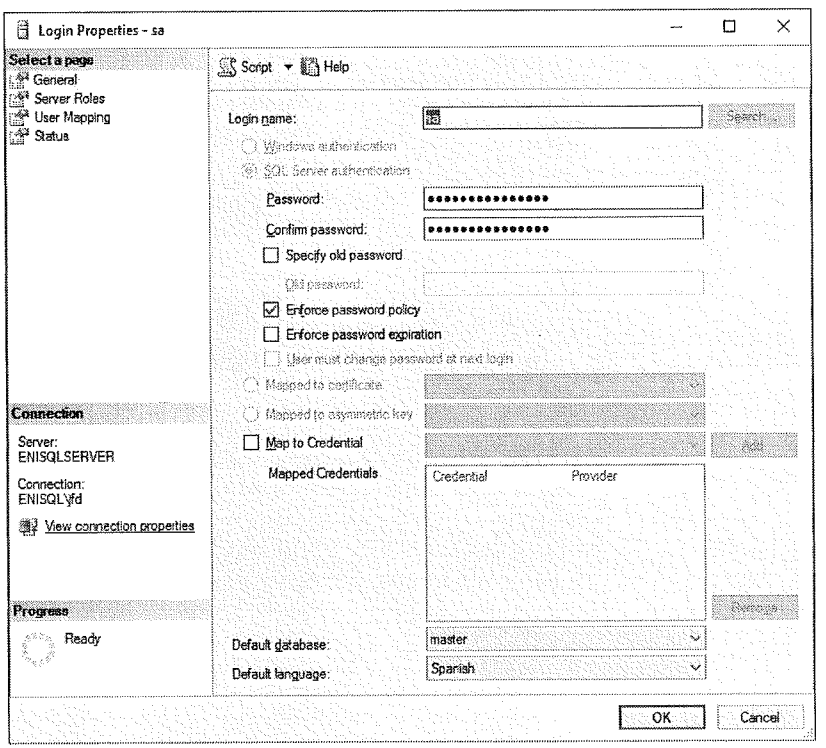
Esta preocupación afecta únicamente a los servidores que son configurados en modo de seguridad mixta. Si se ha seleccionado esta opción durante la instalación, es necesario asegurarse de que la contraseña del administrador SQL Server (sa) es suficientemente segura. Si no es el caso, es necesario modificarla.

Durante la instalación de SQL Server, se predefinen dos usuarios. El primero es el grupo local de los administradores (utilizado con la seguridad de Windows). El segundo es el usuario **sa**. Estos dos usuarios tienen los derechos de administrador del servidor SQL. El usuario **sa** se basa en la seguridad de SQL Server y su contraseña se solicita durante el procedimiento de instalación.

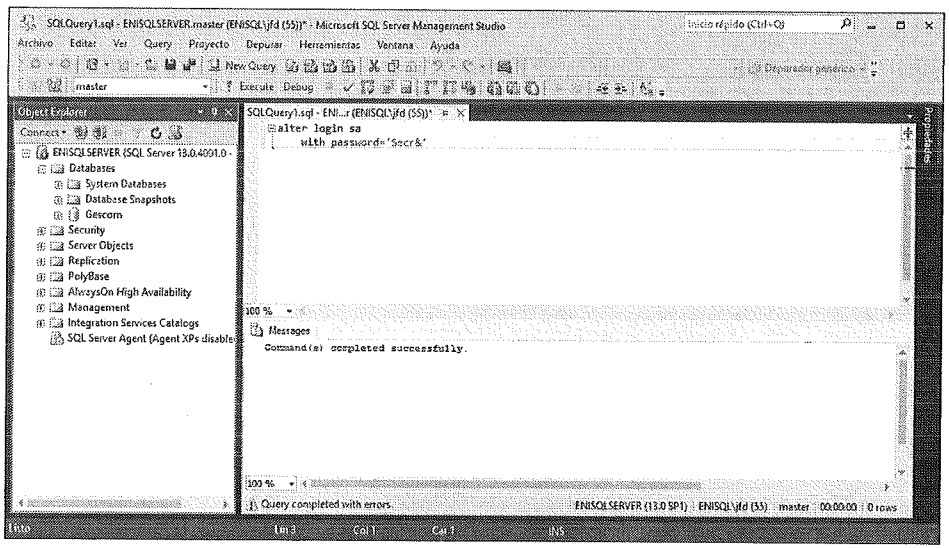
Si en el momento de la instalación solo se activa el modo de seguridad de Windows, la conexión **sa** no está activada. Es necesario definir una contraseña segura antes de activar la conexión. Sin embargo, la conexión solo podrá utilizarse si el servidor está configurado en modo de seguridad mixta.

Se presentan dos posibilidades.

Por SQL Server Management Studio

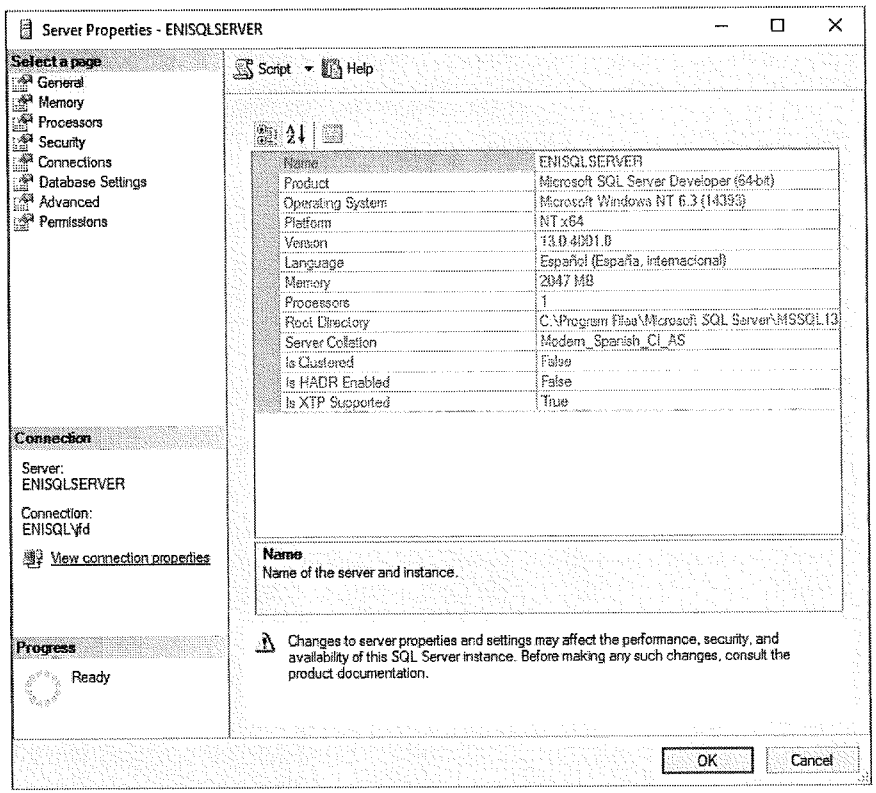


Por el Transact SQL

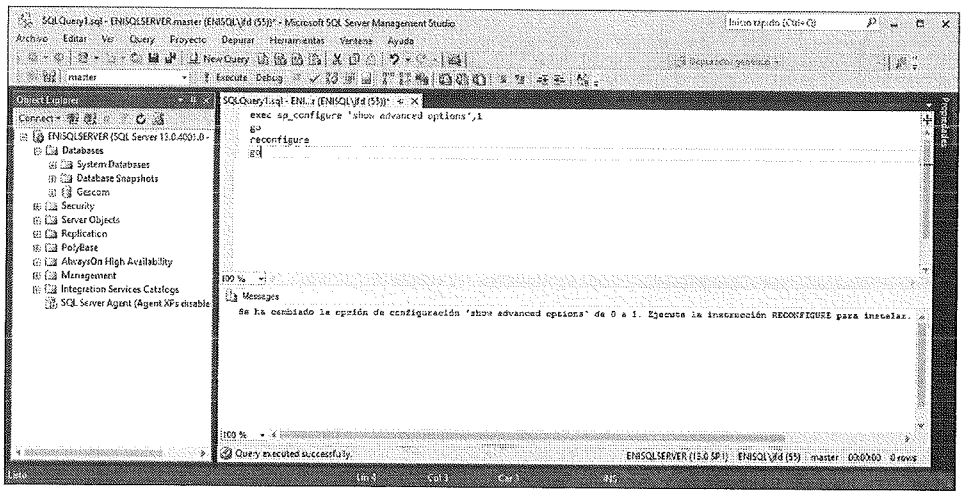


Gestión de los recursos

Los recursos de la máquina se gestionan dinámicamente. Esta gestión automática de los recursos permite ofrecer las mejores funcionalidades posibles del servidor realizando un mínimo de tareas administrativas. Sin embargo, algunas veces puede ser interesante gestionar manualmente algunos recursos para optimizar la utilización de los recursos del servidor. Algunos de estos ajustes se pueden realizar a través de SQL Server Management Studio.



Para acceder a la totalidad de los parámetros del servidor, es necesario utilizar el procedimiento almacenado **sp_configure**.



Si se modifica una opción con la ayuda del procedimiento **sp_configure**, esta no tendrá efecto hasta que se vuelva a iniciar el servidor SQL. Es posible aplicar la modificación de manera inmediata ejecutando el comando **RECONFIGURE WITH OVERRIDE**.

4.4 La gestión de los procesos de SQL Server

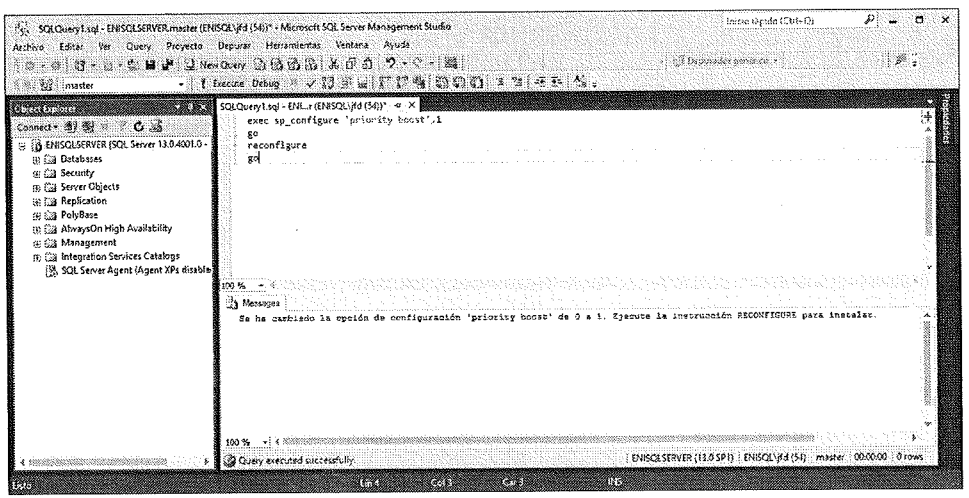
Para el sistema operativo cada aplicación se ejecuta como un proceso. Cada proceso dispone de sus propios threads, que corresponden a las unidades de trabajo que el sistema operativo debe enviar al procesador. A un proceso le corresponde siempre un mínimo de un thread.

Cada instancia de SQL Server gestiona sus propios threads y también su sincronización, sin pasar por el núcleo de Windows. El objetivo de SQL Server es responder eficaz y rápidamente a las peticiones que aumentan la carga de manera brusca y repentina. Para poder estar disponible, SQL Server gestiona su propio pool de threads, por lo que el número máximo se controla por el parámetro **max worker threads**. Con el valor por defecto (0) SQL Server se encarga de gestionar él mismo este número de threads, aunque también se puede fijar el número máximo de threads. El objetivo de estos threads es tratar las peticiones de los usuarios. Teniendo en cuenta que un usuario no trabaja el 100 % de su tiempo en el servidor, sino que debe leer o modificar los datos antes de enviar una nueva consulta, es preferible para SQL Server compartir un mismo thread entre varios usuarios.

Observación

El valor máximo de este parámetro era 255 con SQL Server 2000. En una migración de servidor se recomienda utilizar el valor cero (0).

Por último, para programar la ejecución de los diferentes threads, Windows asigna a cada proceso una prioridad diferente. El rango de valores va de 1 (el menos prioritario) a 31 (el más prioritario). Esta gestión de prioridad no afecta a los procesos de sistema. Por defecto, el proceso SQL Server recibe un nivel de prioridad igual a 7, es decir, un proceso normal. Es posible dar una prioridad superior por medio de la opción de configuración **priority boost**. Esta opción puede ser interesante cuando varias instancias de SQL Server se ejecutan en el mismo puesto y se quiere dar prioridad a una de ellas.



Para conocer los valores de las diferentes opciones de configuración de la instancia, es posible consultar la vista sys.configurations. Esta vista contiene, entre otras, las siguientes columnas:

name	Representa el nombre de la opción.
value	Valor actual configurado. Es el valor que se tendrá en cuenta la próxima vez que se inicie la instancia.
value_in_use	Valor de la opción en la instancia que se está ejecutando.

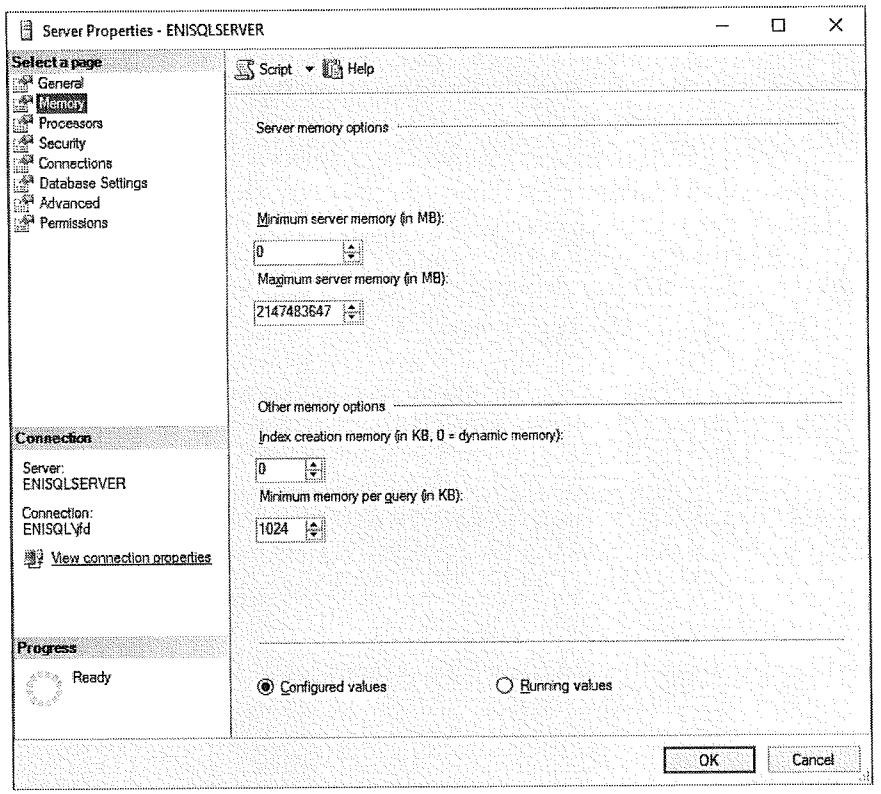
4.5 La gestión de la memoria

Por defecto, SQL Server gestiona dinámicamente la cantidad de memoria que necesita. Es muy recomendable conservar una gestión dinámica de la memoria, ya que permite un reparto óptimo de esta entre los diferentes procesos que se ejecutan en el servidor.

Es importante que el servidor tenga memoria suficiente, ya que esto permite minimizar el número de lecturas físicas y favorecer las lecturas lógicas. Cuanto más numerosas sean estas últimas, mejor es el tiempo de respuesta del servidor. El ratio entre estos dos tipos de lecturas se puede obtener con el Analizador de rendimiento. Este punto se trata en el capítulo Herramientas adicionales - El monitor de rendimiento (monitor de sistema).

Con objeto de permitir la gestión dinámica de la cantidad de memoria utilizada, SQL Server se apoya en el API (*Application Programming Interface*) de gestión de la memoria de Windows (*Query Memory Resource Notification*), para adquirir el máximo de memoria sin privar al sistema de la cantidad de memoria que necesita.

Esta gestión dinámica se puede limitar usando los parámetros **min server memory** y **max server memory**. La instancia de SQL Server, aunque se utilice poco, conservará siempre la cantidad de memoria especificada en **min server memory**. En caso de carga de trabajo, es posible adquirir memoria, sin sobrepasar nunca el valor especificado en **max server memory**.



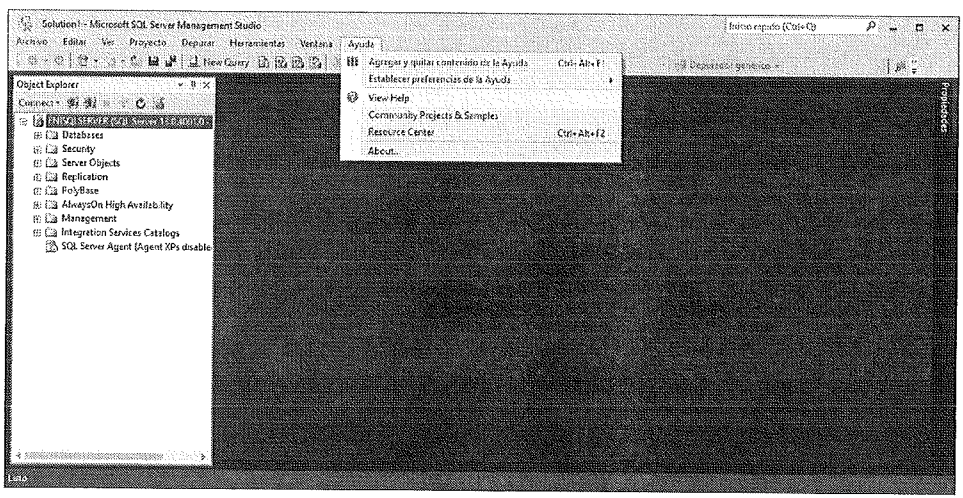
4.6 La documentación on-line

Durante la instalación de SQL Server, no se instala la documentación on-line en el servidor, sino que hay que acceder al sitio web MSDN, que la contiene. Esta solución permite garantizar que siempre se acceda a la última versión de la documentación de SQL Server. En caso de que sea necesario administrar servidores de diferentes versiones, esto permite tener un punto único para la documentación de todas las versiones de SQL Server. Sin embargo, esta solución requiere un acceso a Internet desde el servidor o el puesto de administración.

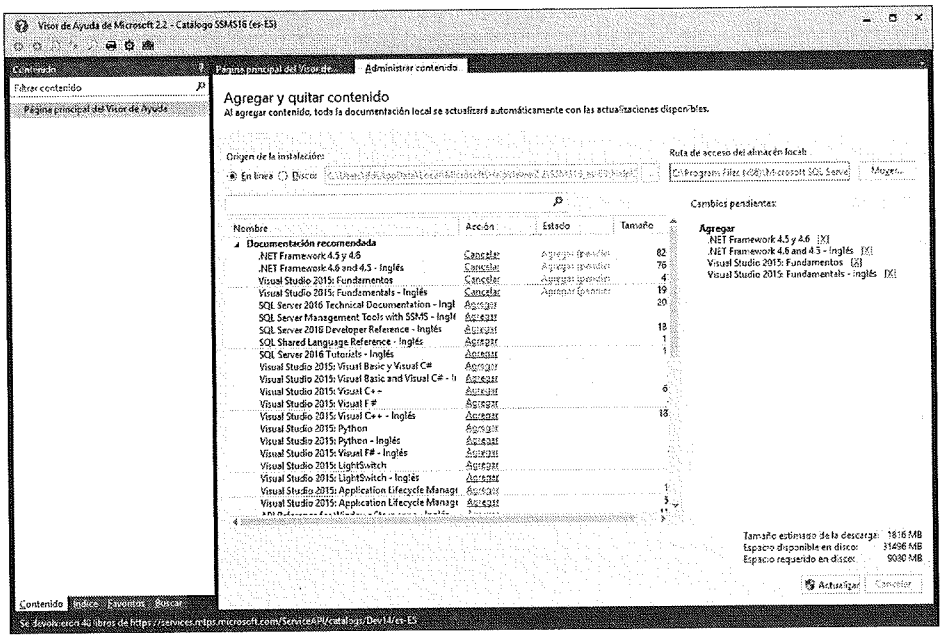
Si esto no resulta posible, es mejor descargar e instalar la documentación de SQL Server 2016.

El administrador de bibliotecas de ayuda permite, de una manera muy sencilla, recuperar la documentación de forma local.

Desde SSMS, hay que activar el menú **Ayuda - Agregar y quitar contenido de la Ayuda,** para pedir al administrador de bibliotecas de ayuda que muestre la documentación que puede instalar de manera local. Después de unos instantes de recuperación de información, se muestran los diferentes contenidos disponibles. Solo falta seleccionar el contenido o los contenidos de ayuda que deseamos tener localmente.



Se muestra entonces la pantalla de configuración de la ayuda local donde conviene agregar de manera local el o los elementos deseados.



5. El servicio de texto completo

El objetivo del servicio de búsqueda por texto completo es mejorar la precisión y velocidad de las consultas que se llevan a cabo sobre campos que contienen textos de grandes dimensiones, es decir, en las columnas de tipo char, nchar, varchar, nvarchar, varbinary y xml. En caso de que la tabla tenga muchas líneas, una consulta con el operador like puede tardar varios minutos en ejecutarse. Si la columna tiene un índice por texto completo, la extracción de las líneas tarda solo unos instantes.

Para poner en marcha este servicio, que se usa en la indexación, la consulta y la sincronización, se necesita una clave única (o clave primaria) en todas las tablas que puedan ser utilizadas en una búsqueda por texto completo. El índice por texto completo conserva una traza de todas las palabras significativas que se han empleado, así como su ubicación. Para que la búsqueda sea acertada y rápida, solo se deben utilizar palabras relevantes y con sentido. Para identificar las palabras sin sentido, SQL Server utiliza una lista de palabras irrelevantes. Esta lista se conserva directamente en la base de datos.

Esta lista de palabras irrelevantes se puede modificar libremente para añadir palabras irrelevantes específicas de una empresa o de un contexto de trabajo. Por ejemplo, el nombre de la empresa se puede considerar como una palabra irrelevante, ya que es probable que aparezca con frecuencia.

El servicio de búsqueda por texto completo se apoya en algunos aspectos precisos para describir su puesta en marcha y funcionamiento:

- Índice por texto completo: almacena la información relativa a las palabras significativas. Las búsquedas se efectúan a partir de esta información.
- Catálogo por texto completo: asociado a una instancia de SQL Server, el catálogo puede contener de 0 a n índices.
- Analizador léxico: en función del idioma y sus reglas léxicas, el analizador léxico va a definir las fichas.
- Ficha: se trata de una cadena de caracteres marcada por el analizador léxico.
- Generador de formas derivadas: en función del idioma, el generador de formas derivadas permite gestionar las diferentes formas que puede tomar un término, como, por ejemplo, la conjugación para un verbo o bien la concordancia para un nombre o un adjetivo.
- Filtro: este elemento permite extraer el texto a partir de un archivo específico (.doc, por ejemplo) y registrar este texto en una columna de tipo varbinary(max), por ejemplo.
- Análisis o Alimentación: se trata del proceso que permite inicializar y mantener actualizado el índice.
- Palabras irrelevantes: se trata de una lista de palabras que no tienen significado ni sentido para las búsquedas en modo texto. Esta lista de palabras es específica de cada idioma. El objetivo es reducir el número de palabras que se deben tratar, eliminando todas las palabras de unión, artículos o términos desprovistos de significado dependiendo del contexto; por ejemplo, el nombre de la empresa o un acrónimo que se use habitualmente.

Este servicio también está disponible para todas las bases que alberga el servidor.

Implementación

La búsqueda lingüística en datos de tipo texto solo es posible en las tablas activadas para la búsqueda por texto completo. La búsqueda lingüística, al contrario que el operador LIKE, que se basa en los caracteres, efectúa una comparación sobre las palabras y expresiones.

La puesta en marcha de una búsqueda por texto completo en una base de datos obliga a efectuar las operaciones siguientes:

- Precisar las tablas y las columnas que deben inscribirse en la búsqueda por texto completo.
- Realizar la indexación de los datos de las columnas inscritas y completar los índices por texto completo con las palabras relevantes.
- Ejecutar las consultas en las columnas inscritas para la búsqueda por texto completo.

- Asegurarse de que todas las modificaciones realizadas en estas columnas se han propagado a los índices.

Estos índices no se completan en tiempo real, sino de manera asíncrona, ya que:

- El tiempo de indexación es generalmente mucho más largo.
- Las búsquedas por texto completo son, por regla general, bastante menos precisas que las búsquedas en modo estándar.

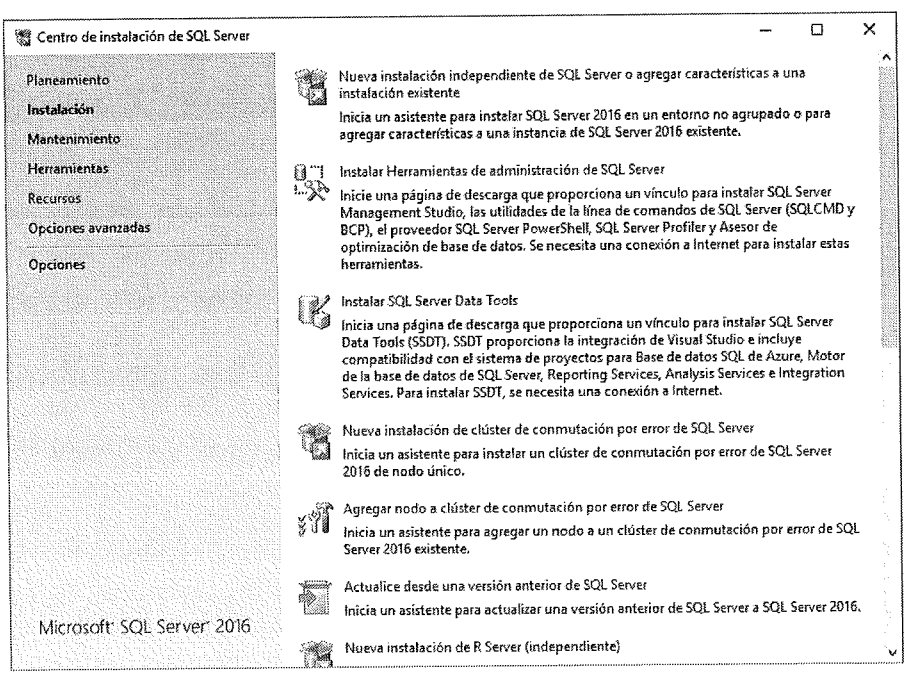
Puesta en marcha

El servicio de búsqueda integral de texto no se instala por defecto con el motor SQL Server ya que es un componente opcional de la instalación de SQL Server.

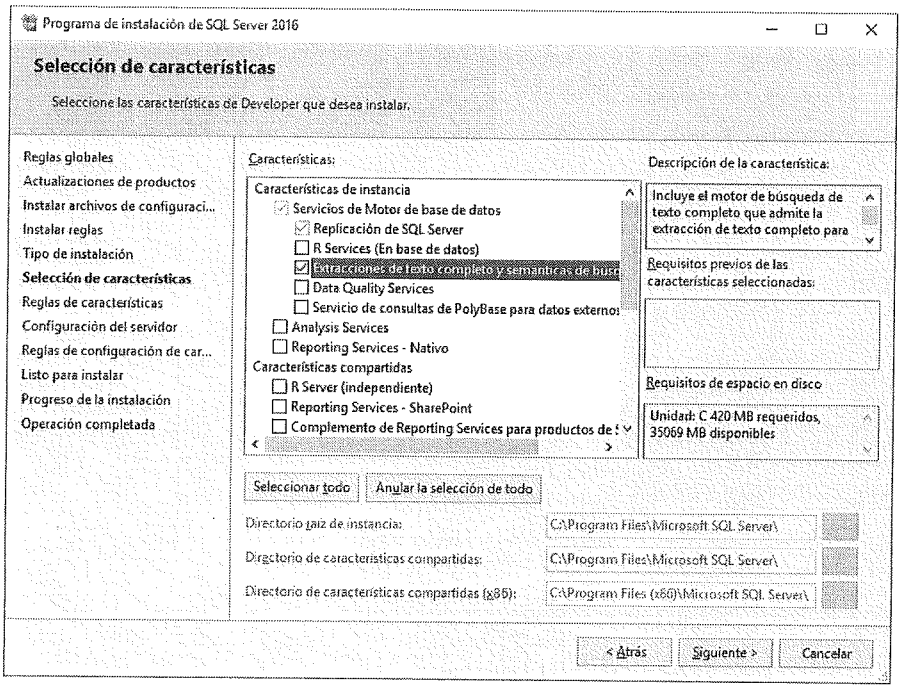
En caso de que no esté instalado este servicio, es necesario volver a ejecutar el proceso de instalación de SQL Server, solicitando esta vez la modificación de la instancia instalada para añadirle el servicio requerido.

Ejemplo

Solicitar la modificación de la instancia existente y añadirle funcionalidades.



Después de seleccionar la instancia en la que se aplica este cambio de funcionalidad, es necesario seleccionar el componente llamado **Extracciones de texto completo y semánticas de búsqueda**, como se muestra abajo.



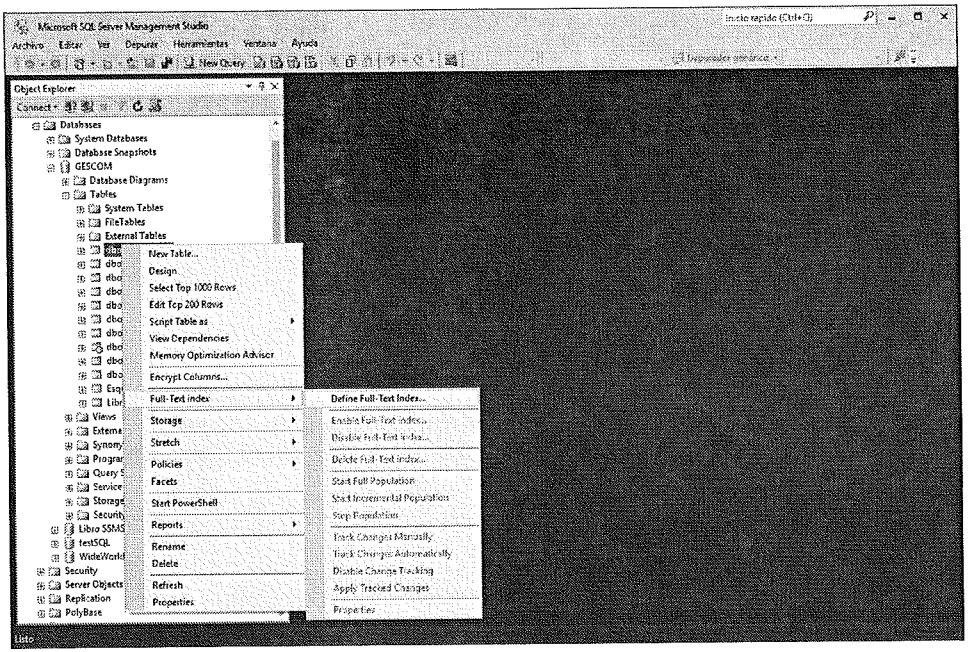
Este proceso para agregar funcionalidades es el mismo que se aplica cuando la funcionalidad deseada no se encuentra en la instalación inicial de SQL Server.

La puesta en marcha del servicio de búsqueda por texto completo puede efectuarse desde SQL Server Management Studio en forma de scripts Transact SQL.

Las etapas del trabajo que hay que realizar son:

- Crear un catálogo.
- Crear uno o varios índices que utilicen este catálogo.
- Definir la lista de las palabras irrelevantes.

La puesta en marcha del servicio por texto completo puede realizarse tanto de manera gráfica desde SQL Server Management Studio como por medio de scripts Transact SQL, o bien por medio del asistente de indexación por texto completo. Teniendo en cuenta que la creación de este tipo de índices es una operación puntual, la utilización del asistente puede ser muy útil. Se puede acceder desde el menú contextual asociado a la tabla en la que se va a definir el índice.



Esta puesta en marcha también puede realizarse etapa por etapa por medio de comandos Transact SQL.

5.1 El catálogo

En el momento de su creación inicial, el índice por texto completo va a hacer muchas lecturas y escrituras en el disco duro. Por lo tanto, es necesario que el índice se defina en un sistema de archivos de alto rendimiento. Esto significa que el índice por texto completo se va a definir en un catálogo que le es propio. El catálogo se define obligatoriamente en la misma base que el índice. Así, como un índice se define siempre en un catálogo, un catálogo puede contener uno o varios índices. Cuando la tabla indexada contiene muchos registros, es mejor que el índice por texto completo se defina en su propio catálogo. Por el contrario, si las tablas tienen un número razonable de registros, entonces es posible definir un número razonable de índices en un mismo catálogo. Con el objetivo de que el catálogo esté optimizado, es mejor agrupar en un mismo catálogo los índices que tienen una frecuencia de actualización parecida.

El catálogo se gestionará con las instrucciones Transact SQL CREATE FULLTEXT CATALOG, ALTER FULLTEXT CATALOG y DROP FULLTEXTCATALOG. Aquí se detalla solo la creación de un catálogo con la instrucción CREATE FULLTEXT CATALOG.

Observación

No es posible definir catálogos en las bases master, model y tempdb.

```
CREATE FULLTEXT CATALOG nombreCatálogo
  WITH ACCENT_SENSITIVITY = {ON|OFF}]
  [AS DEFAULT]
  [AUTHORIZATION nombrePropietario]
```

`nombreCatálogo`

Nombre del catálogo. Cada catálogo tiene un nombre único. El nombre está limitado a 120 caracteres y respeta las reglas de nomenclatura de los identificadores en SQL Server. El nombre del catálogo se usará para nombrar el archivo. El nombre del archivo también debe ser único.

`ACCENT_SENSITIVITY`

Permite precisar si el catálogo debe distinguir o no los caracteres acentuados.

`AS DEFAULT`

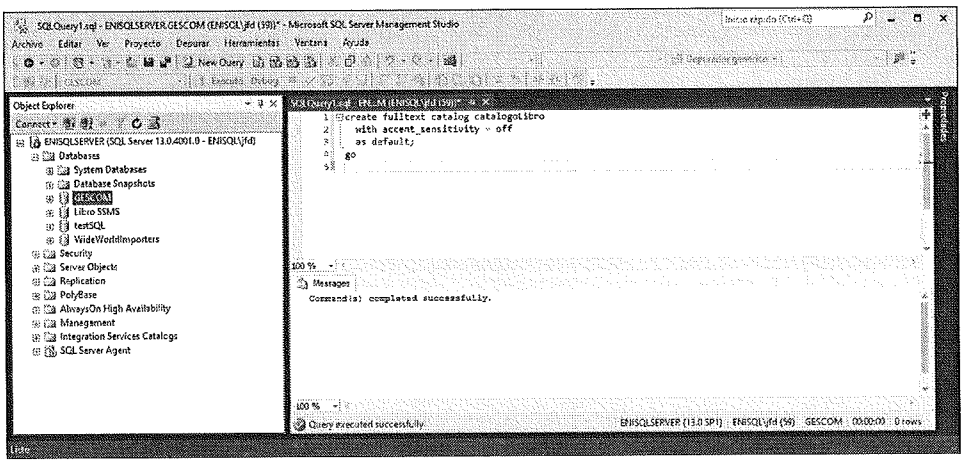
El catálogo se convierte en el catálogo por defecto para los índices por texto completo que se puedan definir a partir de ese momento.

`nombrePropietario`

Cuando el creador del catálogo no es el propietario, es posible indicar el nombre del propietario.

Ejemplo

En el siguiente ejemplo, se define un catálogo de texto completo en la base de datos GESCOM.

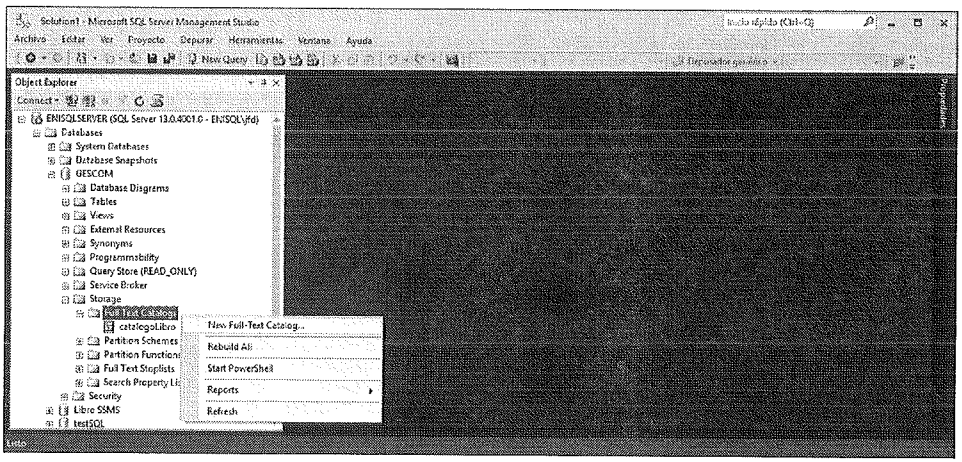


Observación

Las opciones ON FILEGROUP e IN PATH todavía están presentes a nivel de la instrucción, aunque no tienen ningún efecto cuando se trabaja en una instancia de SQL Server 2012.

Para efectuar las operaciones de modificación y eliminación, es necesario utilizar las instrucciones ALTER FULLTEXT CATALOG y DROP FULLTEXT CATALOG.

También es posible realizar estas operaciones desde la consola SQL Server Management Studio, seleccionando la opción **New Full - Text Catalog** desde el menú contextual asociado al nodo **Storage - Full Text Catalogs**, en el explorador de objetos.



El índice por texto completo

El índice único utilizado por el índice por texto completo para hacer referencia a las líneas de información deberá ser lo más compacto posible con el objetivo de limitar la carga de trabajo. Un dato de tipo entero (int) encaja perfectamente.

El índice por texto completo se define con la instrucción CREATE FULLTEXT INDEX.

Naturalmente, este tipo de índice puede definirse en columnas que contienen datos de tipo texto, pero también en las columnas de tipo varbinary(max) que almacenan textos directamente en un formato de datos específico (por ejemplo, .doc). Este tipo de índice también se puede crear en las columnas de tipo xml.

```
CREATE FULLTEXT INDEX ON nombreTabla
  [(nombreColumna [TYPE COLUMN tipoDocumento]
    [LANGUAGE indicadorDeIdioma] [,...])]
  KEY INDEX nombreÍndice
    [ON nombreCatálogo]
  [WITH CHANGE_TRACKING
    {MANUAL | AUTO | OFF [, NO POPULATION]}]
[,STOPLIST={OFF|SYSTEM|nombreListaPalabrasIrrelevantes}]
```

`nombreTabla`

Se trata del nombre de la tabla en la que se ha definido el índice por texto completo.

`nombreColumna`

Nombre de la columna o de las columnas que participan en el índice.

`tipoDocumento`

Cuando la columna indexada contiene un documento, esta columna permite conocer el tipo de documento.

`indicadorDeIdioma`

Permite especificar el idioma en el que se guarda la información en la columna indexada. Este indicador solo es útil en caso de que el idioma utilizado para almacenar la información difiera del idioma por defecto definido en SQL Server. Por ejemplo, para indexar una columna que contiene textos en inglés cuando SQL Server está configurado con español como idioma por defecto.

`nombreÍndice`

Nombre del índice único que se utilizará para hacer referencia a las líneas de información en la tabla.

`nombreCatálogo`

Nombre del catálogo utilizado por el índice. Si no se especifica ningún nombre de catálogo, se utiliza el catálogo por defecto (el creado con la cláusula AS DEFAULT).

`WITH CHANGE_TRACKING`

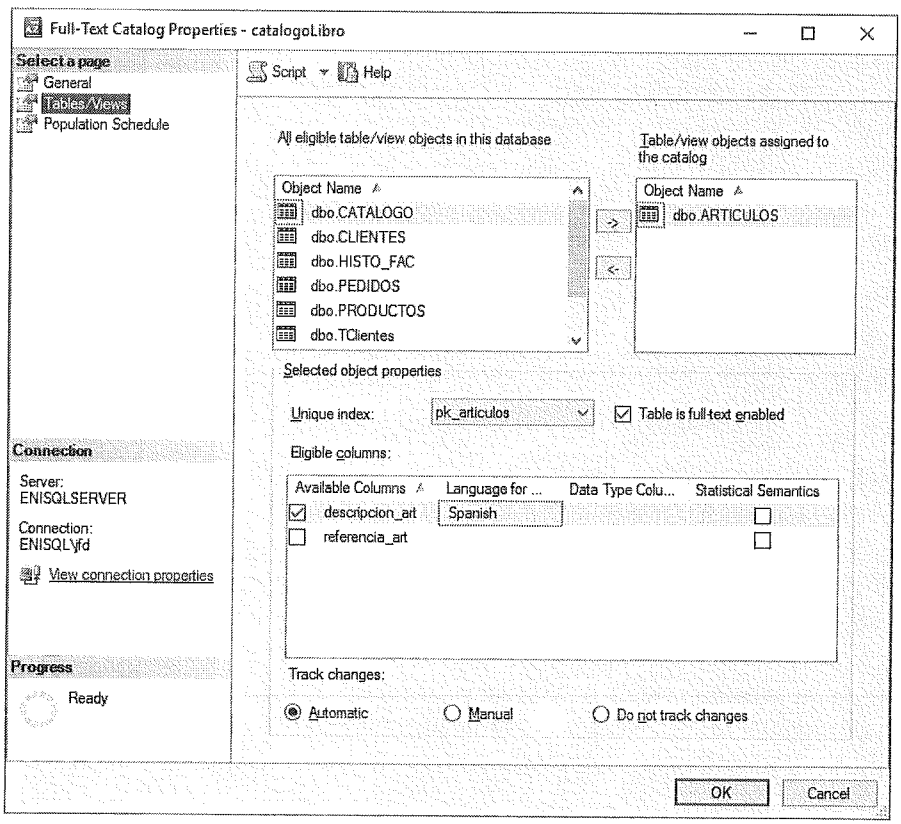
Esta cláusula permite especificar cómo se informa al índice de las modificaciones realizadas en las columnas indexadas.

`STOPLIST`

Esta opción permite precisar la lista de palabras irrelevantes asociadas al índice: si no hay ninguna (OFF), si es la lista por defecto (SYSTEM) o si es una lista personalizada, en cuyo caso es necesario indicar el nombre.

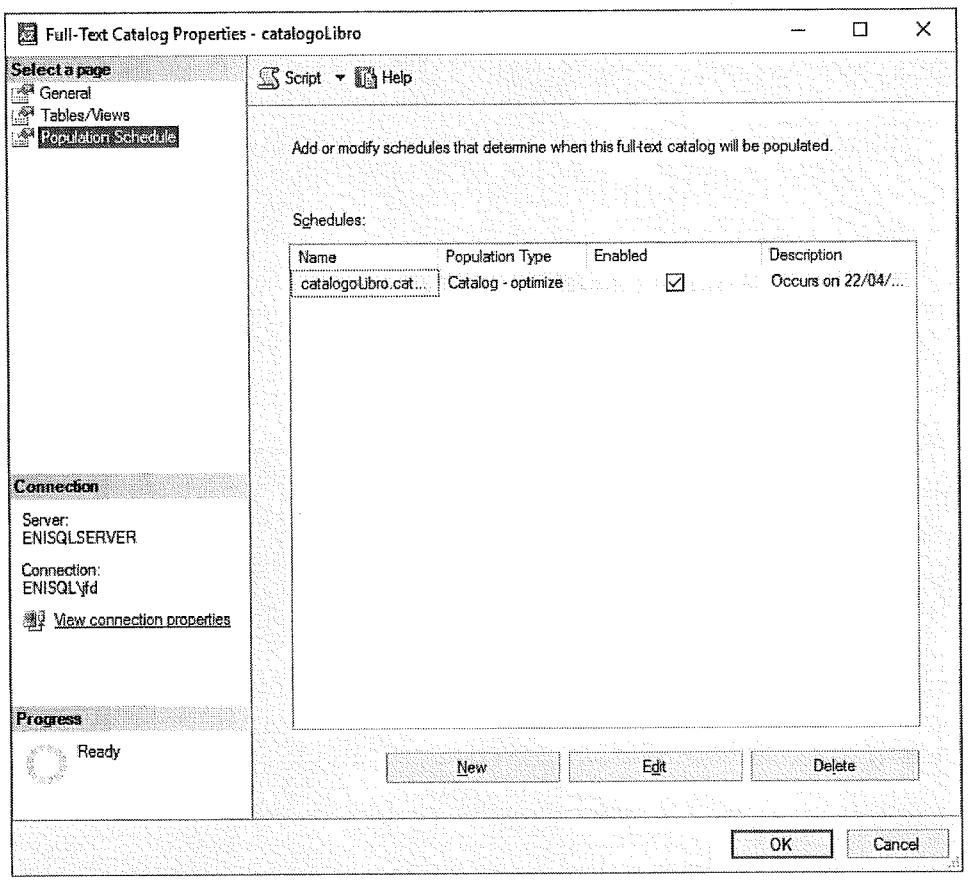
Ejemplo

En el ejemplo siguiente, la columna NOMBRE_ART de la tabla de los artículos está indexada. Este índice utiliza el catálogo **catalogolibro** y realiza el seguimiento de las modificaciones de manera automática.



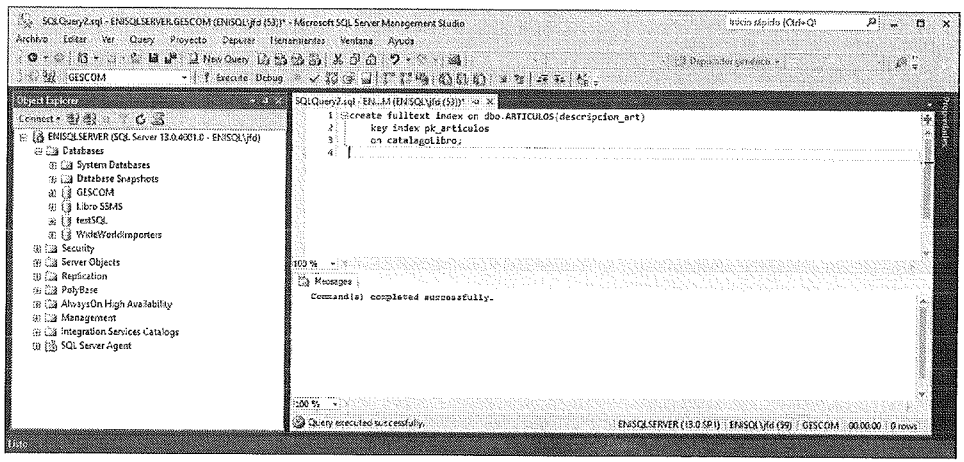
Desde SQL Server Management Studio, es posible visualizar los detalles de este índice por medio de las propiedades del catálogo.

Las propiedades del catálogo están accesibles al seleccionar **Propiedades** desde el menú contextual asociado al catálogo.



Ejemplo

También es posible definir el índice con la instrucción Transact SQL CREATE FULLTEXT INDEX como se muestra a continuación:



5.2 La lista de palabras irrelevantes

Esta lista permite definir las palabras vacías de significado y, por lo tanto, que el índice no debe tener en cuenta.

Esta lista de palabras vacías está disponible únicamente desde SQL Server 2008. Por lo tanto, para poder utilizarla, la base de datos debe estar a un nivel 100 o superior de compatibilidad (110 para SQL Server 2012).

Sintaxis

```
CREATE FULLTEXT STOPLIST nombreListaPalabrasIrrelevantes
[FROM {nombreListaPalabrasIrrelevantesFuente|SYSTEM STOPLIST}]
[AUTHORIZATION propietario];
```

`nombreListaPalabrasIrrelevantes`

Se trata del nombre asignado a la lista de palabras irrelevantes que se está creando.

`nombreListaPalabrasIrrelevantesFuente`

Se trata del identificador de una lista de palabras irrelevantes cuyo contenido se va a copiar en la lista de palabras que se está creando.

`SYSTEM STOPLIST`

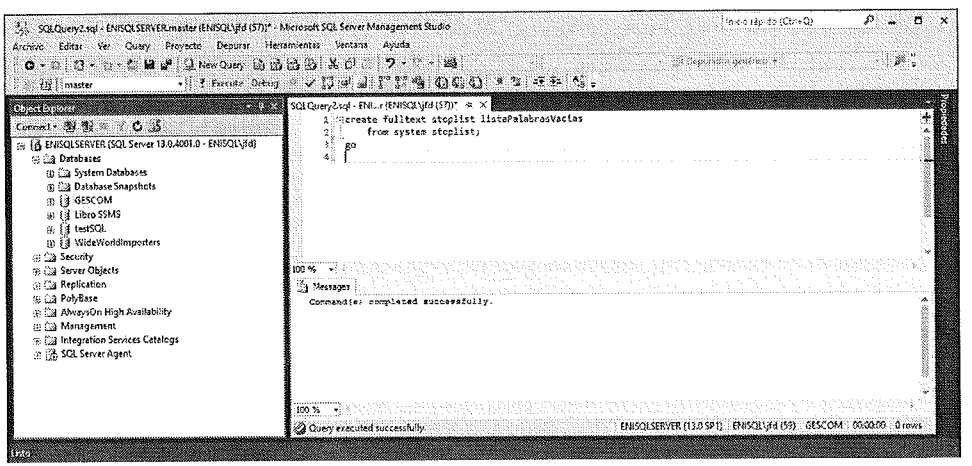
Con esta opción, la lista de las palabras irrelevantes se define a partir de la lista situada en la base de datos de recursos.

`propietario`

Se trata esta vez de especificar explícitamente el propietario de esta lista. Esta opción solo es necesaria en caso de que la persona que ejecute el comando no sea el propietario del índice.

Ejemplo

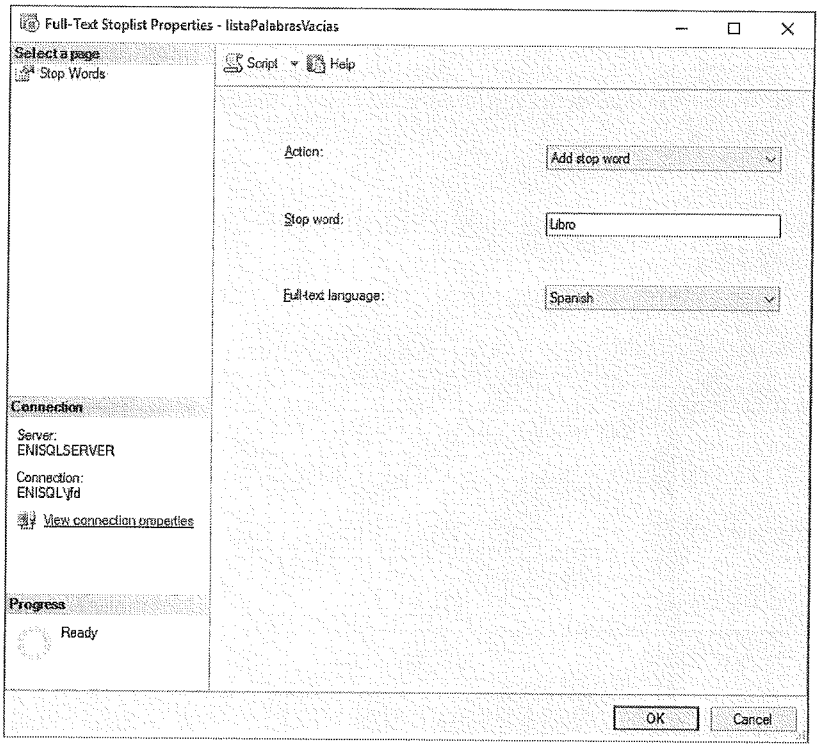
El ejemplo siguiente muestra la creación de una lista de palabras irrelevantes desde un script Transact SQL.



Esta lista también puede gestionarse desde SQL Server Management Studio a partir del nodo **Storage - Full Text Stoplists**.

Ejemplo

Se añade a la lista la palabra Libro.



Una vez definida, esta lista de palabras irrelevantes puede modificarse usando la instrucción ALTER FULLTEXT STOPLIST y eliminarse con DROP FULLTEXT STOPLIST. El comando ALTER permite modificar la lista tanto para añadir como para eliminar palabras desprovistas de significado.

Sintaxis

```
ALTER FULLTEXT STOPLIST listaPalabrasIrrelevantes
ADD 'nuevaPalabra' LANGUAGE idioma;
```

`listaPalabrasIrrelevantes`

Representa el identificador de la lista que se va a modificar.

`nuevaPalabra`

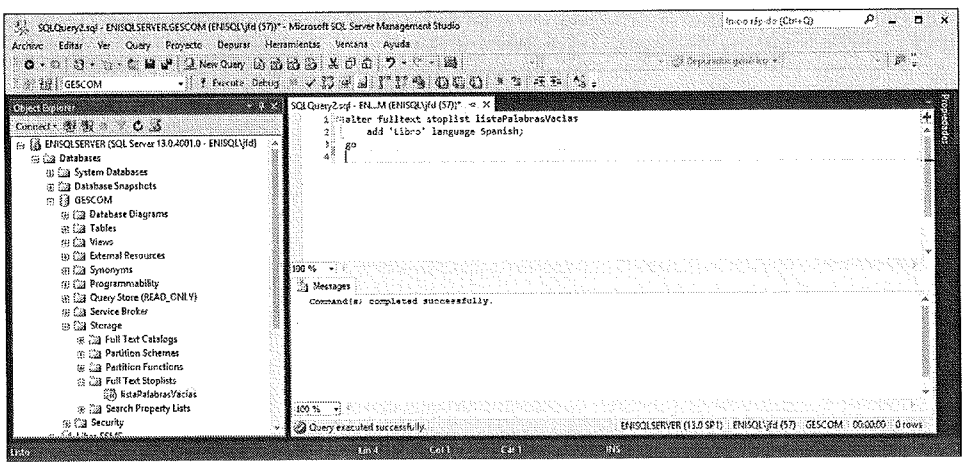
Corresponde al nuevo término vacío de significado. Los términos añadidos representan palabras específicas de un vocabulario determinado que están presentes muy habitualmente para hacer eficaz la indexación.

`idioma`

Representa el idioma para el que se define esta palabra. La codificación de los diferentes idiomas registrados en SQL Server está disponible consultando la tabla **sys.syslanguages**.

Ejemplo

En el ejemplo siguiente, la palabra libro se considera desprovista de significado para el idioma español.



5.3 Inicializar el índice

Después de crear el índice, es necesario enriquecerlo con datos. Esta operación no se realiza de manera instantánea, ya que puede requerir una fuerte carga de trabajo en el lado del servidor. Por lo tanto, es preferible planificar el rellenado del índice para una época de poca actividad.

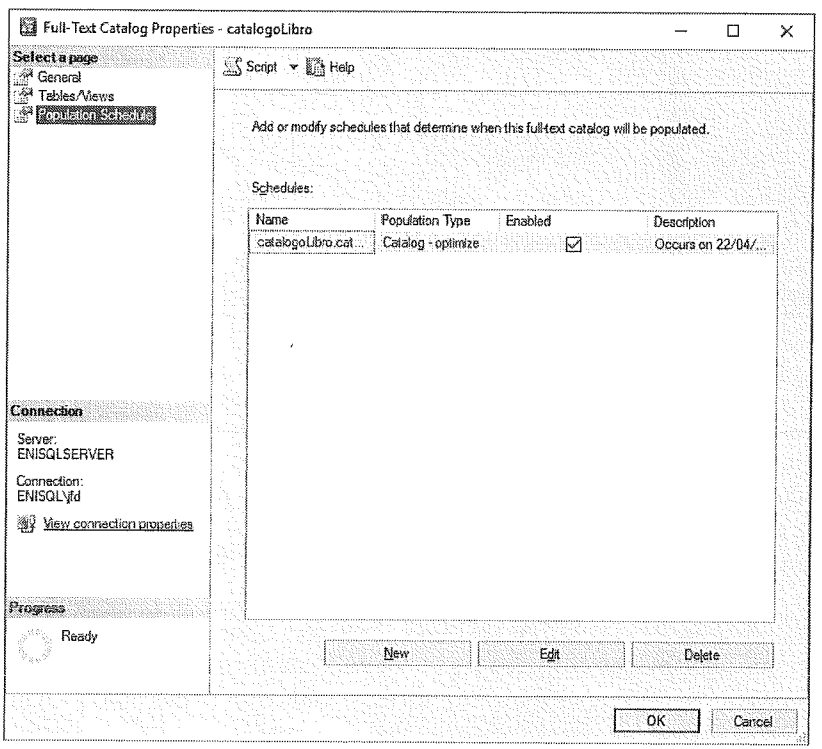
Existen tres maneras de inicializar y mantener actualizado el índice completo: en función de las modificaciones o de forma periódica.

La inicialización completa del índice es, sin duda, la manera la más natural de trabajar con los índices, ya que se indexan todos los términos de manera global, ya sea en el momento de la creación del índice o bien basándose en un incremento temporal determinado.

Con el modo de rellenado basado en las modificaciones, SQL Server guarda una traza de todos los datos añadidos o modificados. El informe de estas modificaciones hacia el índice de texto completo puede efectuarse de manera manual, utilizando un script con una ejecución planificada, o bien de manera continua.

Por último, el rellenado basado en incrementos temporales se apoya en un valor de tipo timestamp. Cuando se añade información a la tabla, la columna de tipo timestamp toma valor. Si la tabla no tiene una columna de tipo timestamp, no es posible utilizar este tipo de rellenado. Al final del proceso de rellenado, el valor timestamp actual se conserva en los metadatos, de manera que en el próximo proceso de rellenado será posible tener en cuenta únicamente los datos más recientes.

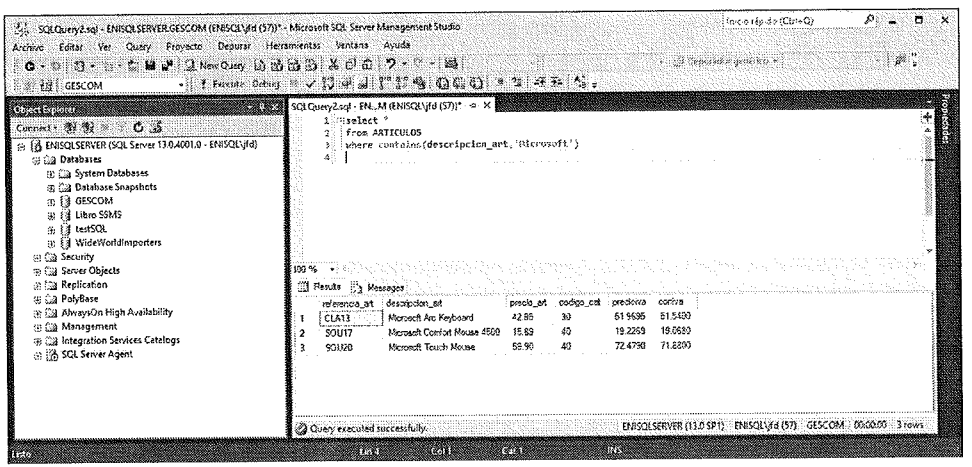
Es posible definir esta operación a partir de la ventana de propiedades del catálogo, como se ilustra en la pantalla siguiente.



Esta operación también puede realizarse en Transact SQL con la instrucción `ALTER FULLTEXT INDEX ON nombreTabla START FULL POPULATION` para una inicialización completa. Esta instrucción tiene diferentes opciones que permiten, por ejemplo, activar o desactivar el índice.

Utilización dentro de las consultas

La utilización de los índices de texto completo se efectúa por medio de dos predicados, **CONTAINS** y **FREETEXT**, que pueden utilizarse en cualquier cláusula WHERE. También existen dos funciones Transact SQL que combinan un conjunto de líneas y pueden utilizarse en una cláusula FROM de una consulta SELECT, a saber, **CONTAINSTABLE** y **FREETEXTTABLE**.



5.4 Encontrar la información relativa a los índices de texto completo

Toda la información relativa a estos índices se puede encontrar consultando las diferentes vistas del catálogo de sistema.

Las cinco vistas que se presentan a continuación son las que se utilizan con más frecuencia.

sys.fulltext_index_catalog_usages

Permite obtener la lista de los catálogos definidos.

sys.fulltext_index_column

Permite identificar las columnas que participan en un índice de texto completo.

sys_dm_fts_index_population

Permite obtener la información de completitud en los índices activos de tipo texto completo.

sys.fulltext_indexes

Permite conocer la lista de índices de texto completo definidos en la base de datos actual.

sys.fulltext_languages

Permite conocer la lista de idiomas que soporta SQL Server a nivel de los índices de texto completo.

Las palabras irrelevantes se pueden consultar en las vistas sys.fulltext_stoplists, sys.fulltext_stopwords y sys.fulltext_system_stopwords.

`sys.dm_fts_index_keywords_position_by_document` permite saber la posición de una palabra clave en un documento.

6. Ejercicio: instalar una nueva instancia

6.1 Enunciado

Creamos una nueva instancia de SQL Server. Esta instancia se llamará Libro y su emplazamiento por defecto de los ficheros de datos será c:\datos\Libro. Se llama a este emplazamiento el directorio raíz de datos. Después, SQL Server organiza su propia arborescencia para los ficheros de datos, los ficheros de log y los de backup.

Observación

Para más simplicidad de los nombres de los ficheros, es preferible evitar utilizar caracteres acentuados.

Para esta instancia, puede usar una edición Developer.

Además del motor de base de datos, es necesario instalar los servicios de replicación.

El servicio SQL Server Agent asociado a esta instancia se configurará en modo de arranque automático. Haremos lo mismo para el servicio SQL Server Browser, ya que este es necesario para acceder a las instancias nombradas de SQL Server.

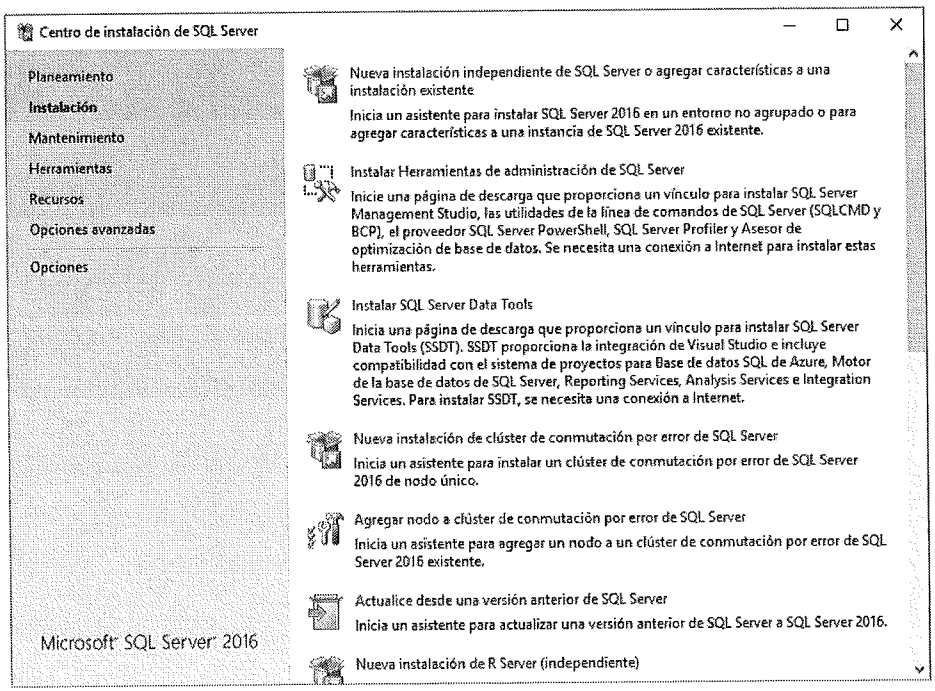
En cuanto a la clasificación, se recomienda elegir Spanish_CI AS, es decir insensible a la diferenciación entre mayúsculas y minúsculas pero si a los acentos.

Elegiremos el modo de autenticación Windows como modo de seguridad y se añadirá el usuario que realiza esta instalación como administrador SQL Server.

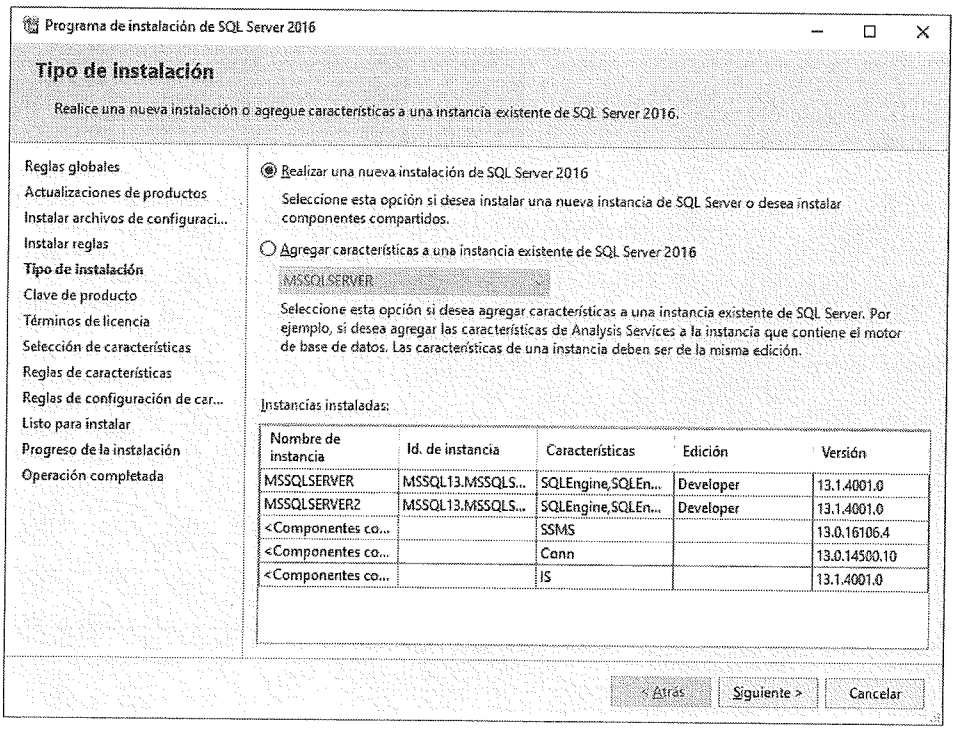
Se recomienda activar la función FILESTREAM. Esta función permitirá almacenar fuera de la estructura de la base de datos los ficheros pesados (como las imágenes, vídeos y sonidos) y disponer de un acceso más rápido a dichos ficheros.

6.2 Solución

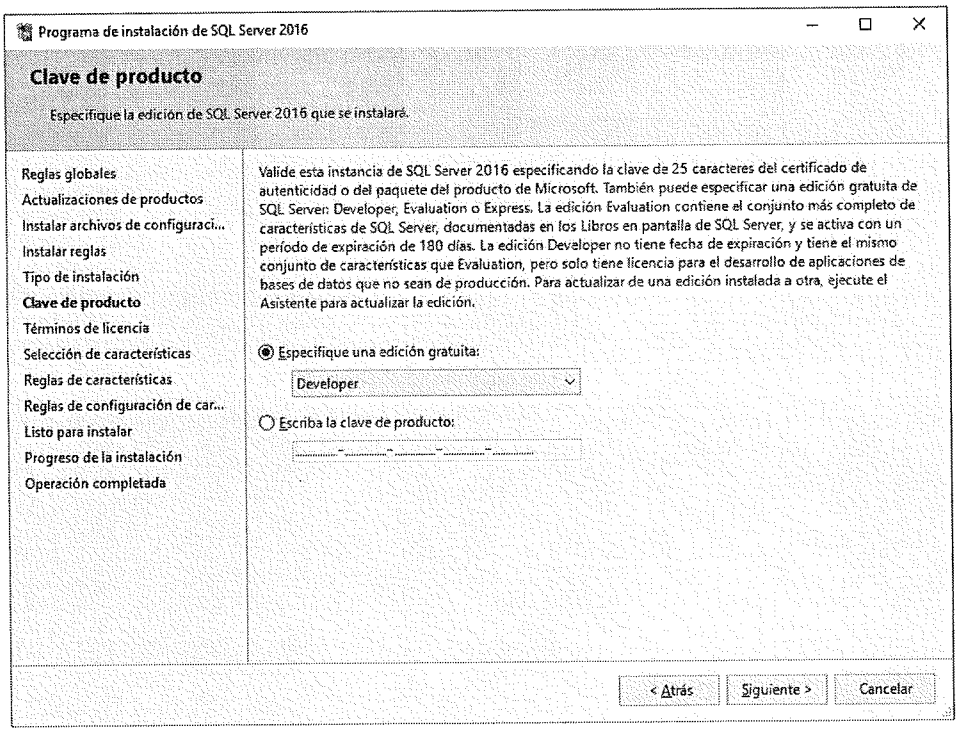
Desde la herramienta de instalación de SQL Server, podemos reutilizar el setup.exe en la raíz del DVD de instalación. Para ello debe elegir la instalación de una nueva instancia.



Después del control de los requisitos, el instalador le pide elegir entre: añadir nuevas funcionalidades sobre una instancia existente o instalar una nueva instancia.

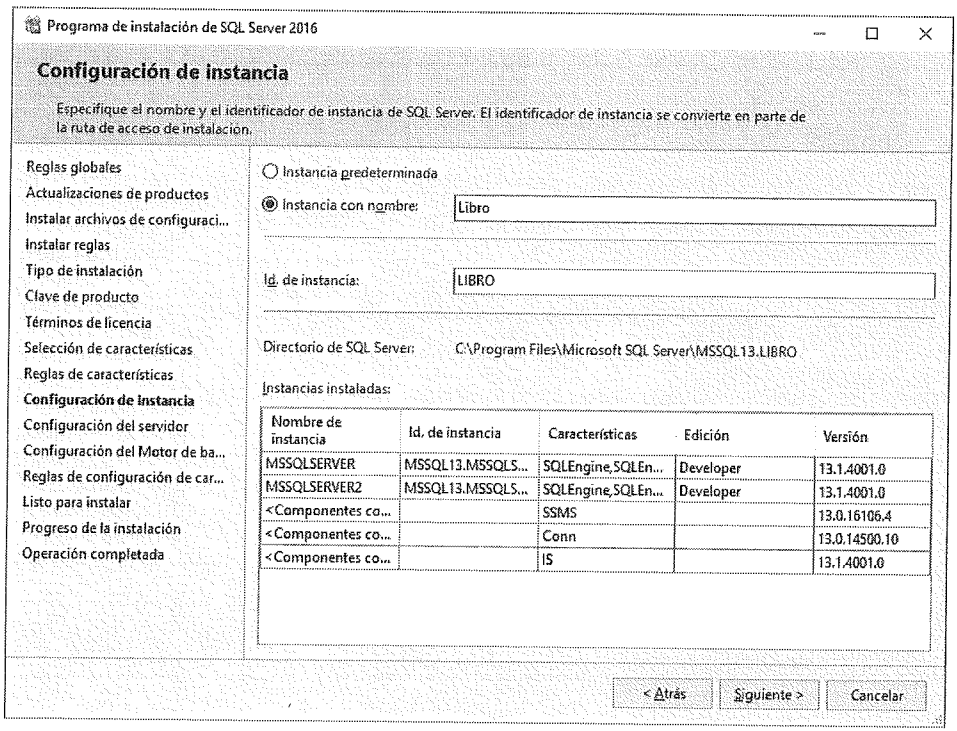


Para más facilidad, debe elegir la edición Developer, que presenta la doble ventaja de disponer de todas las funcionalidades y ser gratuita, lo que resulta positivo para el aprendizaje. Sin embargo, esta edición no corresponde a una puesta en producción.

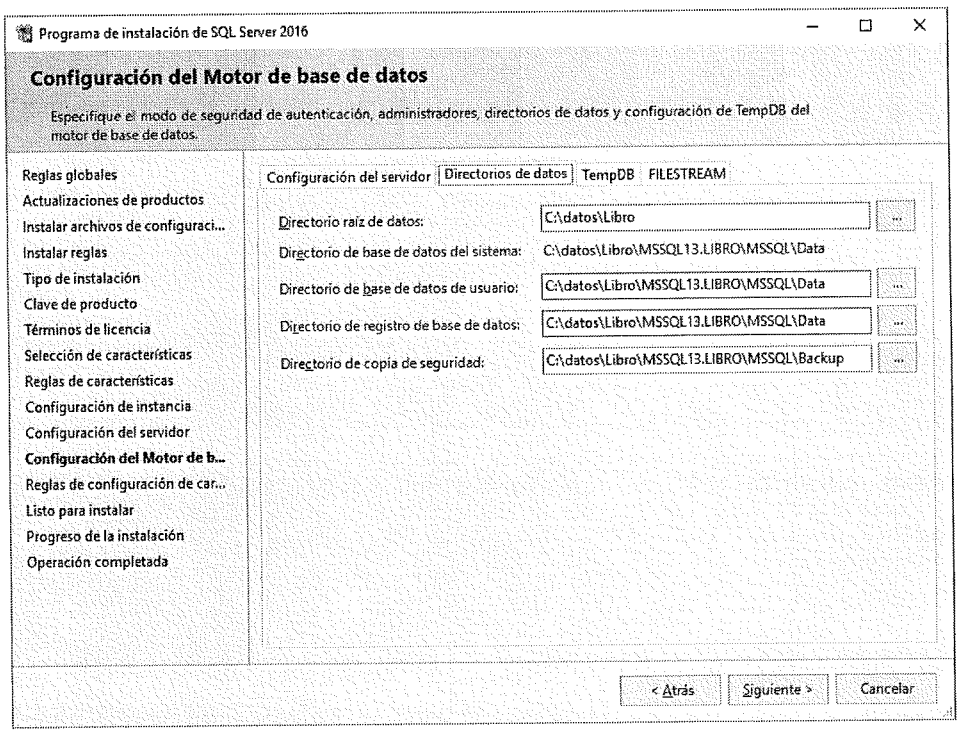


La siguiente etapa consiste en seleccionar las funcionalidades a instalar que, en el caso presente, son el motor de base de datos y la replicación.

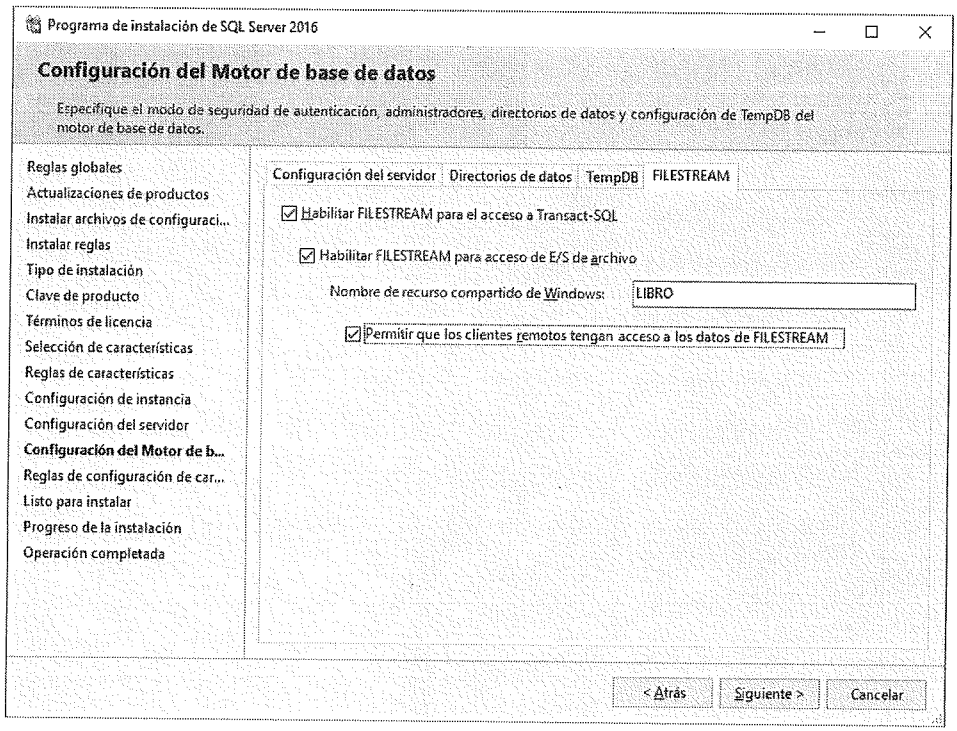
Esta instancia lleva el nombre de Libro.



Por fín, se configura el directorio de almacenamiento de datos por defecto.



Y se activa la funcionalidad FILESTREAM.



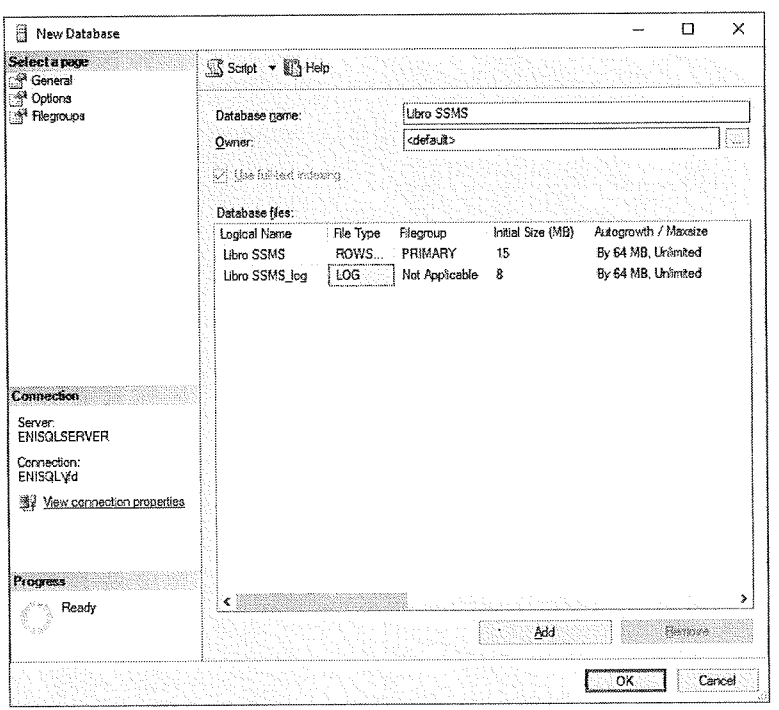
7. Ejercicio: instalar la base de datos de ejemplo por defecto

7.1 Enunciado

Instalar en la instancia Libro la base de datos de ejemplo por defecto World Wide Importers.

7.2 Solución

La primera etapa consiste en ir a la página de GitHub para recuperar los scripts llamados wwi-scripts, es decir los correspondientes a la versión OLTP de la base de datos World Wide Importers. Estos scripts están disponibles en:
https://github.com/Microsoft/sql-server-samples/tree/master/samples/databases/wide-world-importers/wwi-database-scripts



El fichero readme, también disponible en esta página web, muestra los pasos a seguir para instalar la base de datos World Wide Importers en su instancia.

Capítulo 3
Gestión de la base de datos

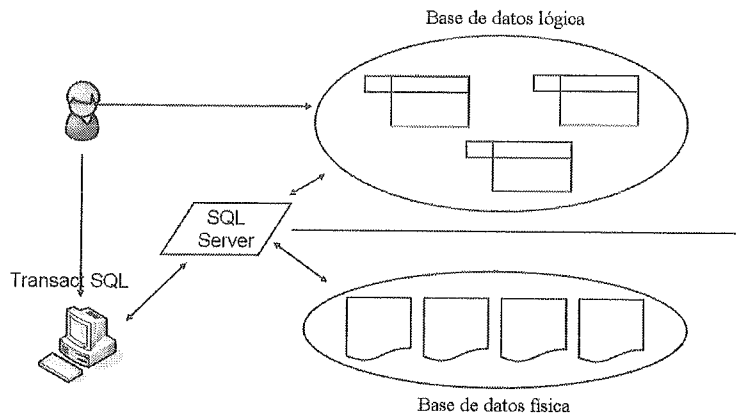
1. Nociones generales

Una vez realizada la instalación del servidor SQL, conviene definir los espacios lógicos de almacenamiento con el objetivo de reagrupar bajo un mismo nombre el conjunto de datos correspondientes a un mismo proyecto. Este conjunto es la **base de datos**, que va a permitirnos trabajar lógicamente con los objetos, tales como las tablas, sin tener que preocuparnos del almacenamiento físico. SQL Server permite realizar asociaciones entre los archivos físicos y las bases de datos. En este capítulo, se tratarán la creación y la gestión de los archivos físicos al mismo tiempo que las bases de datos.

1.1 Relaciones entre la base de datos y la organización física

En el momento de la creación de una base de datos es necesario precisar, al menos, dos archivos. El primero servirá para almacenar los datos y el segundo será utilizado por el diario con el objetivo de almacenar las imágenes antes y después de una modificación de los datos.

Estos dos archivos son obligatorios y propios de cada base de datos. En SQL Server, no es posible compartir un archivo de datos o el diario entre varias bases de datos.



Separación entre los esquemas lógico y físico

1.2 El concepto de transacción

1.2.1 ¿Qué es una transacción?

Una transacción es un conjunto indivisible de sentencias Transact SQL. Una transacción se ejecuta de manera completa y no se admite la ejecución de una sentencia individual de manera aislada. El motor SQL debe ser capaz, mientras la transacción no ha terminado, de restaurar los datos al estado inicial. Si la transacción no ha terminado, ningún otro usuario puede intervenir sobre los datos, ni en modo lectura ni en modo escritura. Es peligroso basarse en los datos, ya que se ignora si las modificaciones en curso van a persistir en el tiempo o no. Para garantizar la coherencia de los datos, todas las líneas que se modifican en el interior de una transacción son bloqueadas para que ningún otro usuario pueda intervenir sobre ellas. El bloqueo se efectúa automáticamente y los bloqueos se liberan cuando el usuario indica el fin de la transacción, ya sea con éxito o con errores. El bloqueo de los datos lo gestiona de manera óptima SQL Server para minimizar el número de líneas de datos bloqueadas (bloqueo de la línea posible), así como el número de bloqueos realizados (bloqueos de línea, de bloques o de tabla). Cuando un dato está bloqueado y una transacción diferente a aquella que causó el bloqueo lo solicita, esta debe esperar a la liberación del bloqueo para acceder a la información. Las filas de espera para el acceso a los datos son gestionadas por las listas FIFO (del inglés *First In First Out*, es decir, «primera en entrar, primera en salir»).

Ejemplo de transacción: un ejemplo seguro conocido y representativo del concepto de transacción es el de la retirada de dinero de un cajero automático. La transacción se basa, por lo tanto, en dos operaciones: el cargo en la cuenta corriente y la retirada del dinero. No es posible realizar una de las dos operaciones de manera aislada.

Si no es posible realizar una de las operaciones, hay que anular las dos. No parece razonable hacer el cargo en la cuenta del usuario si la suma correspondiente no le ha sido entregada.

1.2.2 Las sentencias Transact SQL

Son cuatro, BEGIN TRAN, COMMIT TRAN, ROLLBACK TRAN, SAVE TRAN, e indican respectivamente el comienzo de la transacción, la finalización con éxito, la finalización con errores y la definición de los puntos de parada.

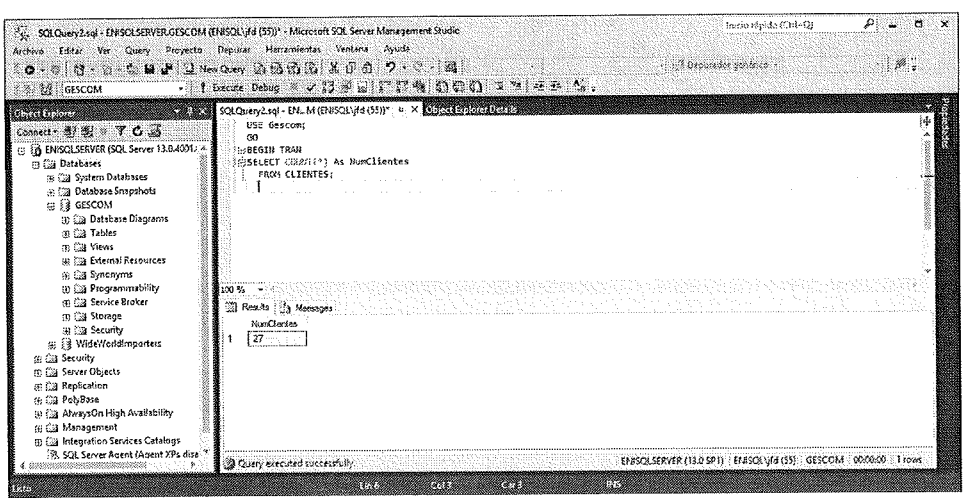
BEGIN TRAN[SACTION]

Esta instrucción permite iniciar de manera explícita una transacción. Si este comando no está presente, toda instrucción SQL es una transacción implícita que se valida (COMMIT) tan pronto como se efectúa la modificación sobre los datos. Se trata por lo tanto de un modo autocommit. En el momento del inicio de la transacción, es posible dar un nombre a la transacción y también marcar dicho inicio en el diario de la base de datos. Esta marca podrá explotarse durante un proceso de restauración de los datos para restaurar la transacción.

```
BEGIN { TRAN | TRANSACTION } [nombreTransacción]
   [ WITH MARK [ 'descripción' ] ]
[;]
```

Ejemplo

En el ejemplo siguiente, se inicia una nueva transacción.



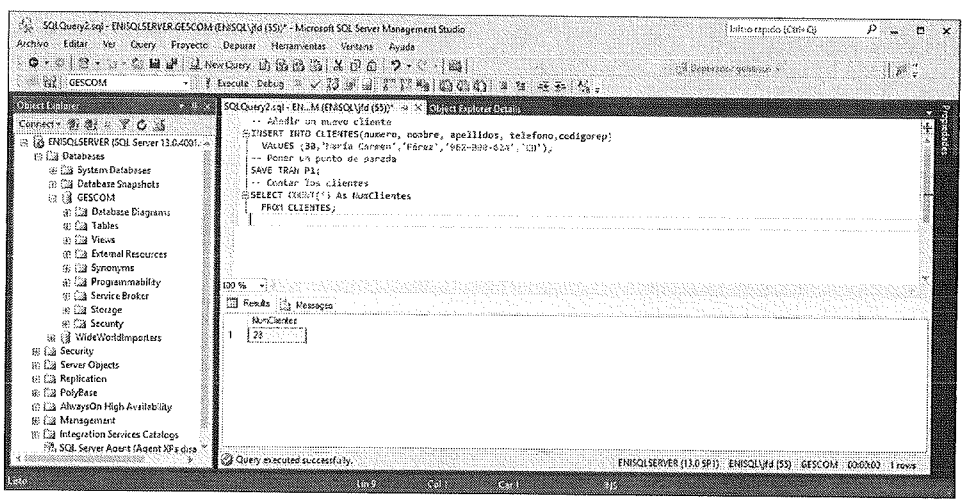
SAVE TRAN

Esta instrucción permite definir los puntos de parada y por lo tanto permite anular una parte de la transacción en curso. Es posible definir varios puntos de parada en una misma transacción. Para permitir la anulación hasta un punto de parada preciso, generalmente se identifican por un nombre.

```
SAVE { TRAN | TRANSACTION } {nombrePuntoParada}[;]
```

Ejemplo

En el ejemplo siguiente, se define el punto de parada P1 y después se añade un nuevo cliente.



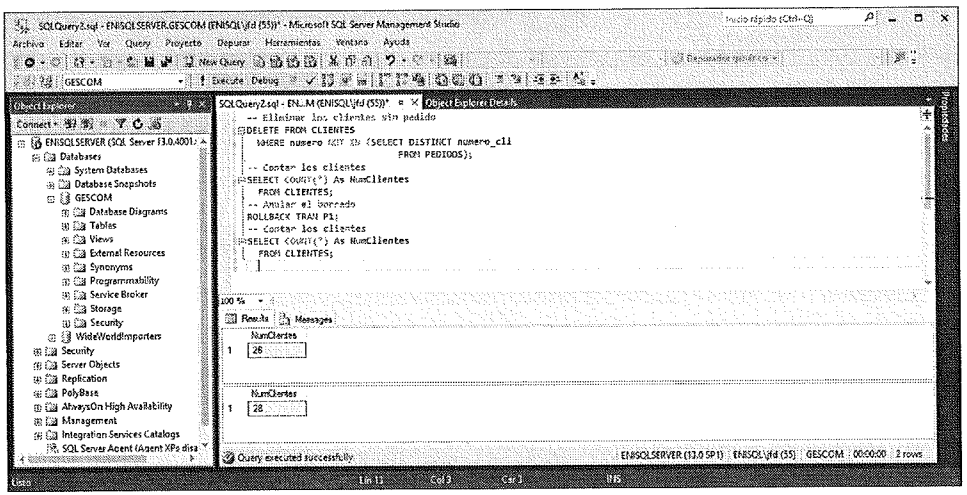
ROLLBACK TRAN[SACTION]

La instrucción ROLLBACK permite anular una parte o la totalidad de la transacción, es decir, parte de las modificaciones realizadas sobre los datos. La anulación parcial de una transacción solo es posible si se han definido puntos de parada con la ayuda de la instrucción SAVE TRAN. No es posible detener la anulación entre dos puntos de parada.

```
ROLLBACK { TRAN | TRANSACTION }
        [nombreTransacción|nombrePuntoParada][;]
```

Ejemplo

En el ejemplo siguiente, se eliminan los clientes sin pedido y después una consulta cuenta el número de clientes definidos en la base de datos. Por último, la eliminación se anula por medio de la instrucción ROLLBACK, que cancela todas las modificaciones efectuadas sobre los datos desde la definición del punto de parada P1.



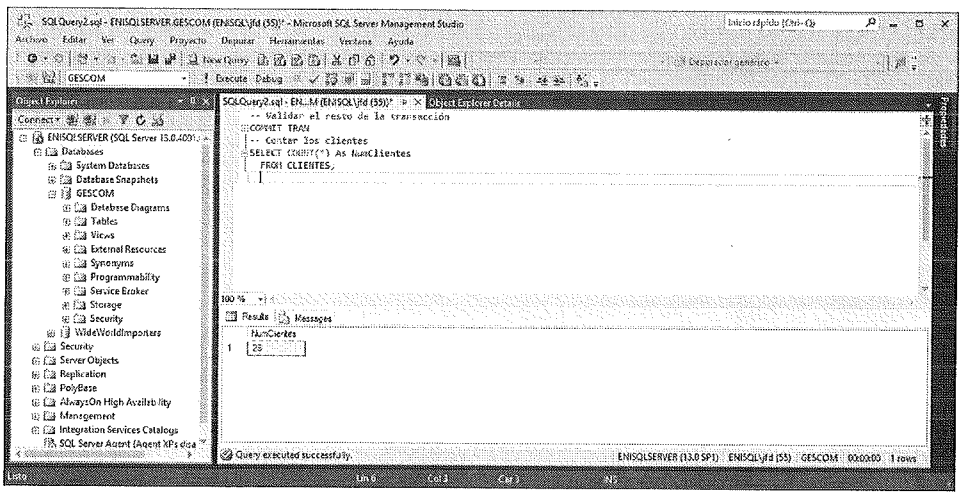
COMMIT

Esta instrucción permite poner fin con éxito a una transacción: es decir, conservar el conjunto de modificaciones efectuadas en la transacción. Para el resto de los usuarios de la base de datos, las modificaciones son visibles al final de la transacción.

```
COMMIT { TRAN | TRANSACTION } [nombreTransacción] [;]
```

Ejemplo

Se validan las modificaciones y finaliza la transacción.



Para SQL Server, si la transacción no ha sido comenzada explícitamente por el comando BEGIN TRAN, toda instrucción SQL constituye una transacción que se valida automáticamente (commit).

Observación

Si en el curso de una transacción explícita el cliente que ha originado la transacción interrumpe su conexión con el servidor, la transacción se anula (ROLLBACK) automáticamente.

Además del modo clásico de gestión de las transacciones, SQL Server propone un modo de transacción diferido. De hecho, en el modo clásico de control de las transacciones, cuando se está ejecutando la instrucción `COMMIT` para validar la transacción, la información relativa a la transacción se registra en el log y solo al final de esta escritura el usuario puede retomar el control para realizar otras operaciones. Este modo de funcionamiento permite garantizar la durabilidad de la información guardada en SQL Server.

El modo de transacción diferido permite responder más rápidamente al usuario sin tener que esperar a que la información se guarde físicamente en el log de las transacciones. SQL Server guarda la información en el log pero posteriormente.

Por tanto, SQL Server guarda la información en el log cuando se valida una transacción normal. La escritura física en el log afectará, por tanto, a varias transacciones. En caso de que todas las transacciones sean diferentes, entonces SQL Server escribirá la información en el log cuando la memoria RAM de escritura del log esté llena o durante la ejecución del procedimiento almacenado `sp_flush_log`.

Este será el modo mejor adaptado cuando la escritura en el log representa un cuello de botella a nivel del rendimiento, si se cumple la condición de que la pérdida de datos sea posible a nivel de la aplicación.

Siempre en el mismo marco de aplicación, es decir, que la pérdida de datos esté permitida, el modo de transacción diferido puede representar una buena solución en caso de que el bloqueo de datos entre usuarios sea frecuente. De hecho, este modo permite liberar más rápidamente los bloqueos, que se liberan justo después de la ejecución de la instrucción `COMMIT TRAN`, sin esperar la escritura en el log.

El uso de este tipo de transacción necesita una configuración específica de la base de datos mediante la opción `DELAYED_DURABILITY` de la instrucción `ALTER DATABASE`.

Sintaxis

```
ALTER DATABASE nombre_base SET DELAYED_DURABILITY={DISABLED | ALLOWED |
FORCED}
```

`DISABLED`: las transacciones diferidas no están autorizadas en la base de datos. Se trata de la opción por defecto.

`ALLOWED`: las transacciones diferidas están autorizadas, pero no es el modo por defecto de las transacciones. El usuario debe indicar que desea trabajar en modo de transacción diferida al inicio de la transacción, con la instrucción `DELAYED_DURABILITY=ON`.

`FORCED`: todas las transacciones son en modo diferido.

También es posible definir bloques de instrucciones atómicas, es decir, no divisibles, a nivel de los procedimientos almacenados. Para estos bloques de instrucciones, es posible definir un comportamiento específico a nivel de las transacciones, con la instrucción `DELAYED_DURABILITY`, que puede tomar el valor `ON` u `OFF`.

Sintaxis

```
BEGIN ATOMIC WITH (DELAYED_DURABILITY={ON|OFF}
```

La instrucción `COMMIT` también permite gestionar el tipo de transacción, respetando la elección definida a nivel de la base de datos. Por tanto, es posible definir el tipo de la transacción a nivel de la instrucción `COMMIT`, solo cuando la base de datos se configura en modo `DELAYED_DURABILITY=ALLOWED`.

Sintaxis

```
COMMIT [TRAN] [WITH DELAYED_DURABILITY={ON|OFF}]
```

1.3 Los archivos de diario

1.3.1 El cometido

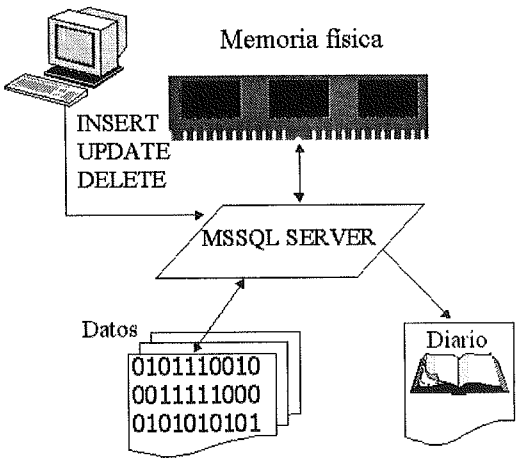
Los archivos de diario o registro permiten almacenar las imágenes antes y después de la modificación de los datos de la base de datos. Solo las operaciones DML (lenguaje de manipulación de datos) y las sentencias SQL INSERT, UPDATE y DELETE provocan un registro en los diarios de las operaciones que se efectúan sobre los datos de la base de datos. Las operaciones de gran envergadura, como la creación de un índice, también se registran en el diario. El diario se utilizará principalmente durante las operaciones de restauración automática posteriores a una parada repentina del servidor, o bien durante las operaciones de copias de seguridad cuando estas últimas se basan en el diario. También se utilizará cuando la base participe en la replicación.

El objetivo del diario es permitir al servidor garantizar en todo momento la coherencia de los datos; es decir, que todas las transacciones validadas (COMMIT) continúen aunque se produzca un problema importante que provoque la parada repentina del servidor.

En cada reinicio del servidor, SQL Server verifica que la última instrucción del diario sea un punto de sincronización. Si no es el caso, se reproducen todas las transacciones validadas (COMMIT), mientras que todas las transacciones no validadas se anulan (ROLLBACK).

SQL Server también ofrece transacciones llamadas diferidas, en el sentido en que la inscripción de la información en el log de las transacciones no está relacionada con la ejecución de la instrucción `COMMIT`. Con este tipo de transacciones, SQL Server responde más rápidamente a los usuarios, porque, cuando el usuario envía el comando `COMMIT`, esto no provoca la escritura física en el disco duro. SQL Server procederá a la escritura de la información en el log durante el siguiente acceso a disco. Lógicamente, esta rapidez tiene la otra cara de la moneda, porque esta transacción, aunque validada correctamente, se puede perder en caso de parada repentina del servidor.

1.3.2 Funcionamiento



Funcionamiento del diario

Cuando una sentencia SQL se ejecuta en el servidor SQL, este intenta ejecutarla lo más rápidamente posible. Después de analizar la sentencia y establecer un plan de ejecución, si los datos afectados por la sentencia no están en memoria, el motor SQL va a leer los archivos en disco duro para encontrar la información necesaria. Cuando están en memoria, se pueden realizar las modificaciones sobre los datos. La modificación se registra siempre en el diario antes de hacerse realmente en los datos de la base de datos. Un diario de este tipo se llama diario de escritura anticipada.

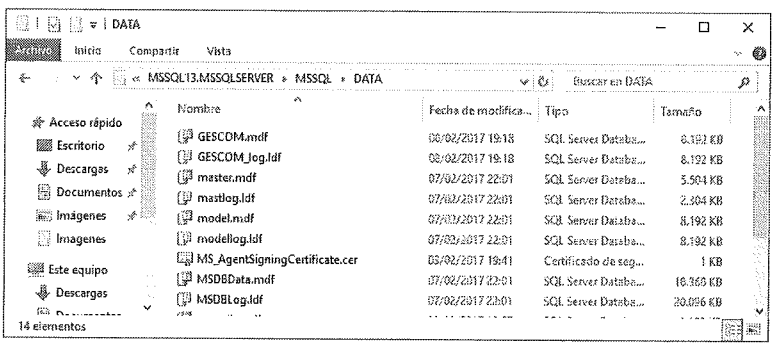
De manera lógica, los datos se registran unos detrás otros en el diario. Cada información está perfectamente identificada por su LSN (*Log Sequence Number*) o número secuencial de registro en el diario. Cada registro en el diario también contiene el identificador de la transacción que originó la modificación de los datos. Los registros de una misma transacción no se registran de manera contigua.

Observación

*Por defecto, los archivos de diario se sitúan en el directorio C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Data, y tienen la extensión *.ldf. Se trata de una extensión recomendada que no es obligatoria. El diario puede estar formado por uno o varios archivos. El tamaño de estos archivos puede ser fijo o variar de manera automática o manual. Se tratará la gestión de los archivos en este capítulo en la parte Creación, administración y eliminación de una base de datos.*

El diario de las transacciones puede estar formado por varios archivos físicos. La gestión del diario se hace de manera independiente de la que se efectúa sobre los datos. SQL Server gestiona una caché de escritura específica del diario.

En función del uso que se haga de la información del diario, es posible truncar regularmente el diario para usar siempre los mismos archivos físicos, controlando en todo momento el espacio que ocupa en disco.



Para hacer que los accesos a disco tengan mejor rendimiento, es imprescindible tener en cuenta este fraccionamiento de datos en páginas de 8 KB y el agrupamiento en extensiones de 64 KB. Por este motivo, para los archivos de datos es mejor elegir asignaciones de 64 KB, mientras que las de 8 KB son suficientes para el archivo de traza.

Observación

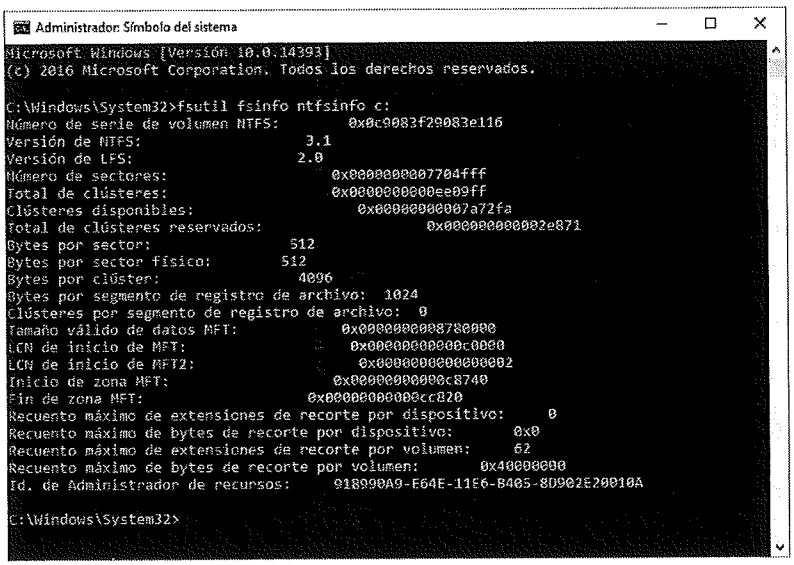
Para saber con precisión el rendimiento de los discos, Microsoft proporciona la utilidad Diskspd. Esta utilidad reemplaza SQLIO y se puede descargar desde la technet de Microsoft.

La unidad de asignación de espacio en disco para los archivos se define durante el formateo y particionamiento NTFS, usando el comando **format/A**. La opción **A** permite definir el tamaño más pequeño de esta unidad de asignación, que varía entre 512 bytes y 64 KB. Cuanto más pequeño es el tamaño, el espacio mínimo que ocupa un archivo se calcula de manera más ajustada. Pensando en la optimización del uso del espacio en disco, es normal utilizar una granularidad muy fina, por lo que la opción adecuada sería una unidad de asignación de 512 o 1024 bytes. Cuando la unidad de almacenamiento se formatea sin el uso de la opción /A, Windows define las unidades de asignación con un tamaño de 4096 bytes.

Es posible conocer el tamaño de la unidad de asignación predeterminada en una unidad de almacenamiento ya existente, usando la herramienta en línea de comandos fsutil. Por ejemplo, para obtener la información del disco C:, se utilizaría el siguiente comando: **fsutil fsinfo ntsinfo c**: Esta instrucción se detalla completamente en technet.

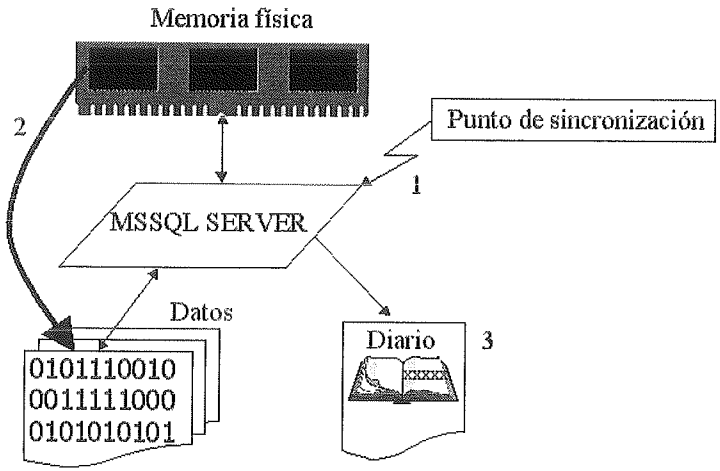
Ejemplo

En el siguiente ejemplo, el disco C: tiene una granularidad de 4096 bytes (información que se muestra en la línea Bytes por clúster).



1.3.3 Los puntos de sincronización

Regularmente, SQL Server va a desencadenar un punto de sincronización. Consiste en volcar sobre archivos todos los datos almacenados en memoria que corresponden a datos validados. Los puntos de sincronización también se llaman CHECKPOINT. El número de datos que se han tocado entre dos puntos de sincronización va a determinar el tiempo de restauración automática después de una parada repentina del servidor.



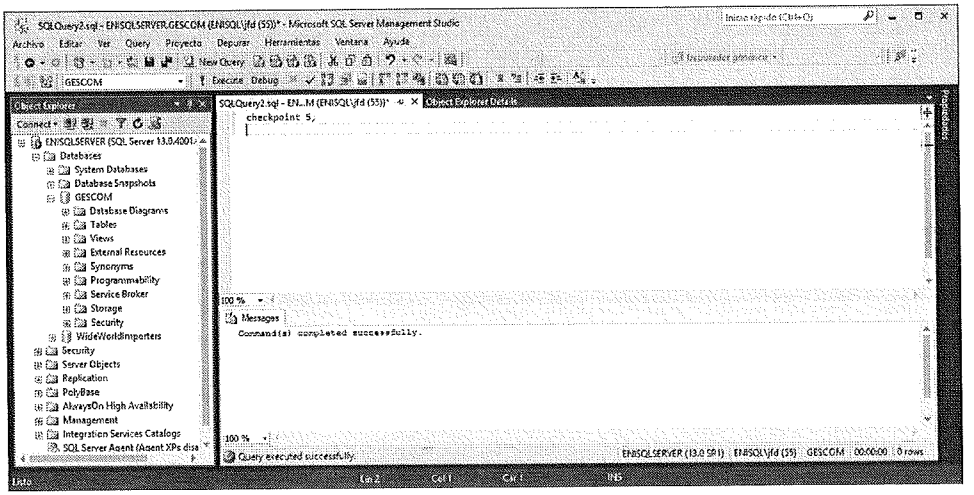
Principio de funcionamiento de un punto de sincronización

SQL Server optimiza los puntos de sincronización de manera que se garantice la mejor gestión de los datos sin deteriorar los tiempos de respuesta del servidor. Se puede intervenir sobre esta optimización por medio de de la instrucción CHECKPOINT.

```
CHECKPOINT [tiempoRealización]
```

`tiempoRealización`

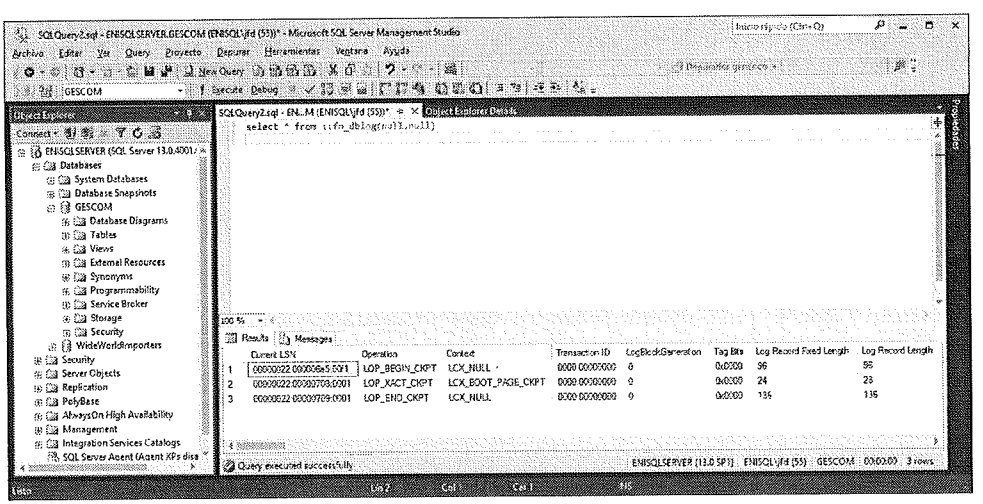
Permite precisar el tiempo acordado en segundos para terminar el punto de sincronización. SQL Server se encarga de desencadenar el punto de sincronización para terminar a tiempo.



Observación

Solo una parada de la instancia por medio de la instrucción Transact SQL SHUTDOWN WITH NOWAIT permite parar la instancia sin desencadenar un punto de sincronización.

La lectura del archivo de traza se realiza mediante la función **::fn_dblog**. Este tipo de función puede ser especialmente interesante para definir el LSN (*Log Sequence Number*) o la fecha y hora en que se realiza una transacción. Este tipo de información será necesaria durante una restauración hasta un punto concreto.



1.4 Los archivos de datos

1.4.1 Su cometido

Cada base de datos tiene, al menos, un archivo de datos. Este archivo contiene el conjunto de datos almacenados en la base. Cada archivo de datos solo puede contener los datos procedentes de una única base de datos, por lo que hay especialización de los archivos en relación con la base de datos.

1.4.2 Estructura de los archivos de datos

Los archivos de datos se estructuran para responder de manera óptima a todas las solicitudes del motor y, sobre todo, para poder almacenar más datos optimizando el espacio en disco utilizado.

Para optimizar el espacio del que dispone, el servidor va a formatear los archivos de datos con objeto de optimizar su estructura.

El trabajo realizado por SQL Server sobre estos archivos de datos se parece al trabajo que realiza el sistema operativo sobre los discos disponibles en la máquina. Por supuesto, es posible que el sistema operativo formatee el espacio de disco del que dispone para dividirlo en bloques. Posteriormente, serán estos bloques los que se asignen a los archivos. SQL Server realiza el mismo tipo de trabajo sobre los archivos de datos y a continuación asigna el espacio disponible a las diferentes tablas e índices.

Las páginas

Antes de poder trabajar con un archivo de datos, SQL Server estructura el archivo fraccionándolo en bloques o páginas de 8 KB. El tamaño de 8 KB lo fija SQL Server para reducir las pérdidas de espacio, simplificando las operaciones de asignación, pero también de lectura y escritura. La página representa la unidad de trabajo de SQL Server. Siempre es una página completa de datos la que se carga en memoria y siempre es una página la que se escribe en disco. No es posible cargar en memoria caché una parte de una página. Una página puede contener varias líneas de una tabla o bien varias entradas de índice y toda la información se carga en una única lectura de disco.

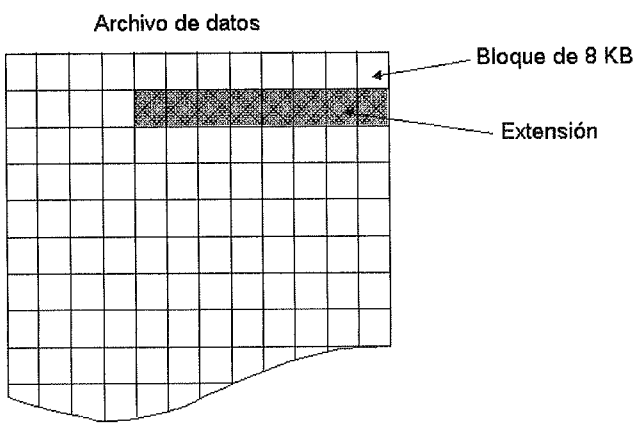
Este tamaño de 8 KB también permite gestionar las bases de datos de tamaño más grande con un número de bloques menor.

Por último, para evitar la fragmentación de las líneas de datos en varias páginas, una línea debe estar entera (excepto el tipo texto e imágenes) en una página. Por tanto, el tamaño máximo de una línea es de 8060 bytes. Este valor de 8060 bytes es ligeramente inferior a 8 KB porque SQL Server reserva algunos bytes para gestionar el encabezado de la página con el objetivo de controlar la gestión.

Si se almacenan varias líneas en la misma página, se hace de manera secuencial. En caso de cambio del tamaño de las líneas, SQL Server gestiona dinámicamente el desplazamiento de las líneas en la página o el establecimiento de un puntero hacia otra página para leer el final de la línea de datos.

Habida cuenta de que la página es la unidad de trabajo de SQL Server, cada página contiene un tipo preciso de datos. Es posible distinguir los tipos de página siguientes:

- Datos: estas páginas contienen la información en formato numérico, de texto o de fecha.
- Texto/Imagen: estas páginas contienen tanto textos voluminosos como objetos en formato binario.
- Índices: estas páginas contienen las entradas de los índices.
- GAM/SGAM o *Global Allocation Map* y *Shared Global Allocation Map*: estas páginas contienen la información relativa a la asignación de las extensiones.
- PFS o *Page Free Space*: estas páginas contienen la información relativa a la asignación de las páginas y el espacio disponible en estas páginas.
- IAM o *Index Allocation Map*: estas páginas contienen la información relativa a la utilización de páginas por las tablas y los índices.
- BCM o *Bulk Change Map*: estas páginas contienen la lista de las extensiones modificadas por las operaciones de copia en bloque desde la última copia de seguridad del diario.
- DCM o *Differential Change Map*: estas páginas contienen la lista de todas las extensiones modificadas desde la última copia de seguridad de la base.



La fragmentación en bloques de los archivos de datos

Las extensiones

Las extensiones son los agrupamientos lógicos de 8 bloques contiguos. Su tamaño es de 64 KB (8 x 8 KB). El papel de las extensiones es evitar una dispersión grande de los datos para un mismo objeto dentro de los archivos de datos.

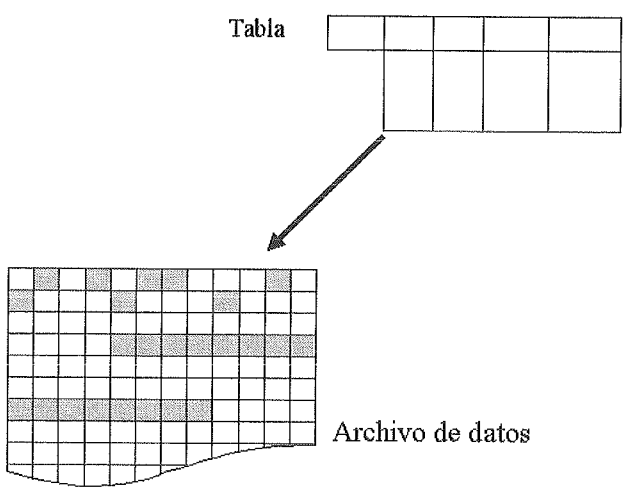
Existen dos tipos de extensiones: las extensiones mixtas y las extensiones propias o uniformes. Estos dos tipos de extensiones permiten limitar lo mejor posible el consumo del espacio en disco por parte de una tabla en función del volumen de datos que hay que almacenar.

Las extensiones mixtas

Inicialmente, las extensiones son comunes a varias tablas o varios índices. Cuando uno de los objetos definidos sobre la extensión solicita espacio, SQL Server asigna el espacio bloque por bloque mientras el objeto no utilice los 8 bloques. Una vez que un objeto supera este límite, SQL Server le asigna espacio extensión por extensión. La ventaja de este método es que impide la pérdida de espacio inútilmente con tablas o índices que contienen pocos datos.

Las extensiones uniformes

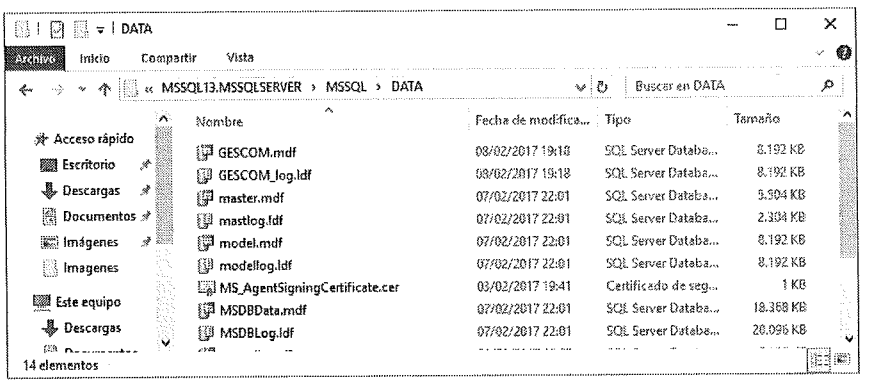
Si un objeto ocupa más de 64 KB de información, entonces el espacio se asignará extensión por extensión. Cada una de las extensiones está especializada para un objeto y las lecturas de disco serán más rápidas, ya que todos los datos se almacenan de manera contigua en paquetes de 64 KB.



Asignación de extensiones

1.4.3 Funcionamiento

Cada base de datos tiene, al menos, un archivo de datos que además es específico de la base. Su ruta de acceso por defecto es C:\Program Files\Microsoft SQL Server\MSSQL13. MSSQLSERVER\MSSQL\Data. El primer archivo de datos de una base tiene la extensión mdf y todos los siguientes tienen la extensión ndf. Estas son las extensiones recomendadas, aunque no obligatorias.



2. Creación, administración y eliminación de una base de datos

Una base de datos gestiona un conjunto de tablas de sistema y de tablas de usuario. La información contenida en estas tablas de sistema representa, entre otras cosas, la definición de las vistas, los índices, los procedimientos, las funciones, los usuarios y los privilegios. Las tablas de usuario contienen la información introducida por los usuarios.

Observación

Una instancia de SQL Server no puede contener más de 32.767 bases de datos.

2.1 Crear una base de datos

La creación de una base de datos es una etapa puntual, realizada por un administrador de SQL Server. Antes de intentar crear una base de datos, es importante definir cierto número de elementos de manera precisa:

- El nombre de la base de datos, que debe ser único en el servidor SQL.
- El tamaño de la base de datos.
- Los archivos utilizados para el almacenamiento de los datos.

Para crear una nueva base de datos, SQL Server se basa en la base Model, que contiene todos los elementos que van a ser definidos en las bases de datos de usuario. Por defecto, esta base Model contiene las tablas de sistema. Sin embargo, es perfectamente posible añadir elementos a esta base. Todas las bases de usuario creadas posteriormente dispondrán de estos elementos adicionales.

Una base se puede crear de dos maneras diferentes:

- Por medio de la instrucción Transact SQL CREATE DATABASE.
- Por medio de SQL Server Management Studio.

Una base de datos siempre está compuesta, como mínimo, de un archivo de datos principal (extensión mdf) y de un archivo diario (extensión ldf). Se pueden definir archivos de datos secundarios (ndf) en el momento de la creación de la base o bien posteriormente.

Esta operación de creación de base de datos afecta a la base Master. Por lo tanto, es absolutamente necesario crear una copia de seguridad de esta base de sistema para poder trabajar con la nueva base de datos tras una restauración.

La información relativa a los archivos de datos se registra en la base Master y en el archivo primario de la base de datos.

2.1.1 Sintaxis Transact SQL

```
CREATE DATABASE nombreBaseDeDatos
[ ON
  [PRIMARY] [ <especificaciónArchivo> [,n]]
  [LOG ON < especificaciónArchivo > [,n]]
]
[ COLLATE intercalación ]
[;]
```

Para especificaciónArchivo existen los siguientes elementos de sintaxis:

```
(NAME = nombreLógico,
FILENAME = 'rutaYNombreArchivo'
[,SIZE = tamaño [KB|MB|GB|TB]]
[,MAXSIZE={tamañoMáximo[KB|MB|GB|TB]|UNLIMITED}]
[,FILEGROWTH = pasoIncremento [KB|MB|GB|TB|%]]
) [,n]
```

PRIMARY

Permite precisar el primer grupo de archivos de la base de datos. Este grupo es obligatorio, ya que el conjunto de tablas de sistema se crea obligatoriamente en el grupo primario. Si la palabra clave PRIMARY se omite, el primer archivo de datos indicado en el comando CREATE DATABASE corresponde obligatoriamente al archivo primario. Normalmente tiene la extensión mdf.

NAME

Este atributo, obligatorio, permite precisar el nombre lógico del archivo. Este nombre se usará para hacer las manipulaciones en el archivo gracias a los comandos Transact SQL; nos referimos fundamentalmente a los comandos DBCC o a la gestión del tamaño de los archivos.

FILENAME

Especificación del nombre y ubicación física del archivo en el disco duro. El archivo de datos se crea siempre en un disco local del servidor.

SIZE

Cuando no se precisa el tamaño para el archivo principal de la base de datos (mdf) entonces se toma el tamaño del archivo principal de la base de datos Model, es decir 8 MB. En ningún caso una base de datos puede tener un archivo principal cuyo tamaño sea inferior a la del archivo de la base de datos Model en su creación.

Para los archivos secundarios y los archivos de logs, el tamaño mínimo es de 1 MB, aunque el tamaño por defecto es de 8 MB. Se utiliza este mismo tamaño por defecto también en la creación del archivo de log.

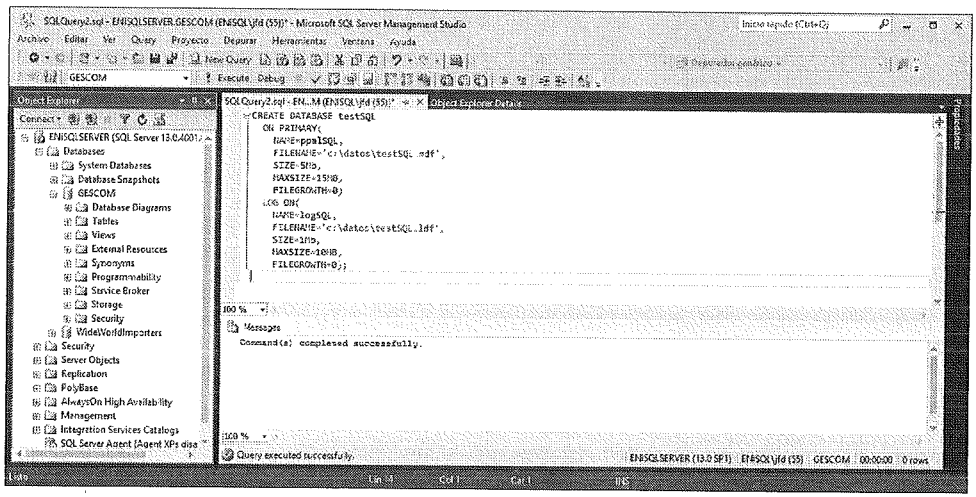
El tamaño de un archivo siempre se expresa utilizando un número entero. Si deseamos indicar un valor no entero, es necesario cambiar de unidad. Por ejemplo, no es posible decir que un archivo tiene un tamaño de 2,5 MB. Para ello, hay que indicar un tamaño de archivo de 2560 KB.

MAXSIZE

Este atributo opcional permite indicar el tamaño máximo en megabytes (por defecto) o en kilobytes que puede tener el archivo. Si no se especifica nada, el archivo crecerá hasta la saturación del disco duro, con el límite de 16 TB para un archivo de datos y 2 TB para un archivo de log.

FILEGROWTH

Indica el factor de extensión del archivo. El tamaño de cada una de las extensiones puede corresponder a un porcentaje (%), un tamaño en megabytes (por defecto) o en kilobytes. Si se indica el valor cero, el factor de extensión automática del archivo es nulo y el tamaño del archivo no cambia automáticamente. El factor de extensión definido por defecto es de 64 MB. Este valor se aplica tanto a los archivos de datos como a los archivos de log.

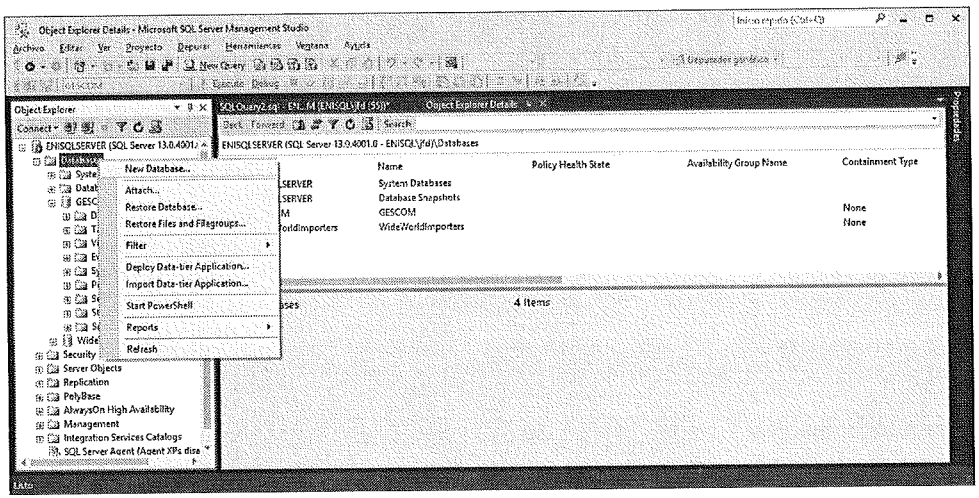


Observación

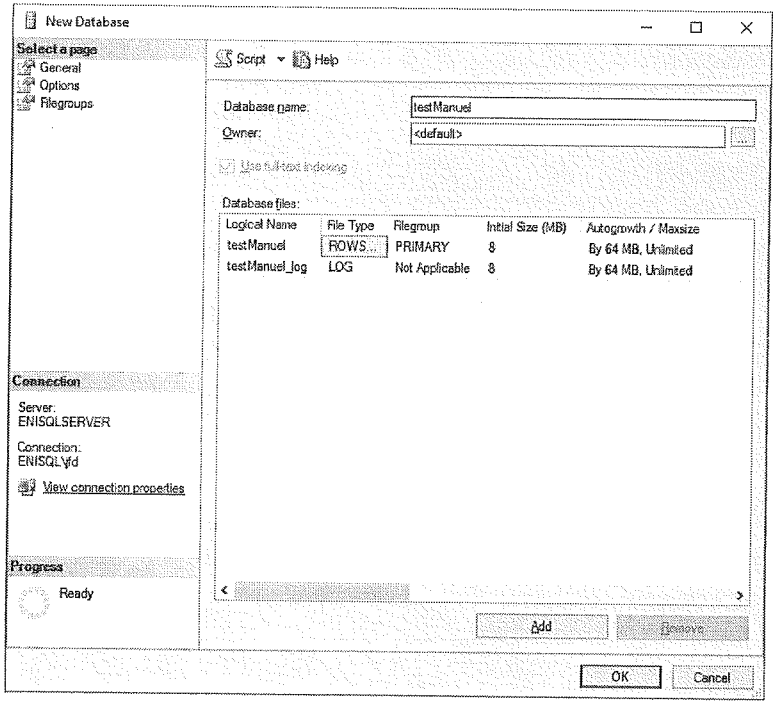
Aquí no se han presentado todas las opciones de la instrucción CREATE DATABASE, sino solo las que se usan más habitualmente cuando se crea una nueva base de datos.

2.1.2 Uso de SQL Server Management Studio

También es posible crear una base de datos de manera gráfica desde SQL Server Management Studio. Es necesario seleccionar **New Database** en el menú contextual asociado al nodo **Databases** del explorador de objetos, como se ilustra a continuación.



SQL Server Management Studio presenta un cuadro de diálogo con las propiedades de la base de datos en modo creación. Después de haber completado opciones como el nombre de la base de datos, así como el nombre y la ubicación de los archivos, es posible solicitar la creación de la base.



Este cuadro de diálogo permite a un usuario crear rápida y fácilmente una base de datos, manteniendo el control sobre todos los parámetros.

Es posible crear archivos en particiones sin formatear. Sin embargo, esta técnica solo permite una pequeña ganancia de rendimiento en relación con la creación de archivos en particiones formateadas NTFS. Además, el sistema de archivos no puede manipular los archivos creados en particiones sin formatear (fundamentalmente en el marco de las copias de seguridad de base de datos parada) y solo puede tener un archivo por partición.

Cuando se usan particiones sin formatear, solo se puede crear un archivo en cada partición de tipo.

Los archivos de base de datos no se deben crear en particiones comprimidas. En algunos casos extremos, es posible situar los archivos secundarios en particiones comprimidas. Los archivos deben estar en modo de solo lectura. En contraposición, los archivos de diario de traza no se deben definir bajo ningún concepto en una partición comprimida.

Si los archivos de datos se guardan en una partición EFS, los datos se encriptan con la cuenta que ejecuta el servicio SQL Server. Solo esta cuenta podrá desencriptar los datos. Cuando se planifique un cambio de cuenta, será necesario transferir los datos en primer lugar.

2.2 Gestionar una base de datos

En el momento de gestionar una base de datos, se toman en cuenta varios criterios. En esta sección se aborda la gestión del espacio utilizado por los archivos físicos que constituyen la base de datos. Los puntos principales respecto a la gestión de los archivos son: el crecimiento dinámico o manual de los archivos, la adición de nuevos archivos y la reducción del tamaño de los archivos.

Observación

En una base de datos, hay un máximo de 32.767 grupos de archivos, y no más de 32.767 archivos por cada grupo.

2.2.1 Aumentar el espacio de disco disponible para una base de datos

Los archivos de datos y los archivos diarios almacenan la información. Como la base contiene normalmente cada vez más información, en un momento dado estos archivos estarán llenos. En ese instante será necesario encontrar más espacio.

Los diferentes métodos expuestos a continuación para aumentar el espacio de almacenamiento para la base de datos son complementarios, ya que cada método tiene sus ventajas y sus inconvenientes.

Archivo con crecimiento dinámico

En el momento de crear la base, es posible fijar ciertos criterios referentes al tamaño máximo de los archivos (MAXSIZE) y a la tasa de crecimiento (FILEGROWTH). Si estas opciones se omiten, el tamaño máximo será infinito y la tasa de crecimiento será del 10 % para los diarios y de 1 MB para los archivos de datos.

Si utiliza archivos de crecimiento dinámico, el servidor nunca se bloqueará por el tamaño del archivo, excepto si se satura el disco o se alcanza el tamaño máximo.

El factor de crecimiento permite fijar el tamaño de todos los añadidos. Este tamaño debe fijarse de manera óptima, teniendo en cuenta el hecho de que, si el factor de crecimiento es muy pequeño, introducirá muchas fragmentaciones físicas del archivo y las peticiones de extensión de este serán frecuentes, mientras que un factor de crecimiento demasiado grande necesitará una carga de trabajo importante por parte del servidor cuando este establezca la estructura de los bloques de 8 KB en el interior del archivo.

Observación

Si la tasa de crecimiento se fija en cero, el archivo no podrá crecer dinámicamente.

Observación

El crecimiento del archivo tiene siempre un tamaño que es múltiplo de 64 KB (el tamaño de una extensión).

Si bien la parametrización de los archivos permite gestionar automáticamente su crecimiento, esta solución presenta al mismo tiempo la desventaja de que no es posible controlar cuándo tendrá lugar este crecimiento. Si esta operación interviene en plena carga de trabajo, se corre el riesgo de ralentizar el servidor.

Por el contrario, si se realiza la parametrización del crecimiento automático de los archivos, en caso de problema los archivos adaptarán su tamaño en función del volumen de datos, sin bloquear a los usuarios.

Archivo con crecimiento manual

El crecimiento manual de los archivos permite controlar el momento en el que el archivo va a crecer y, por lo tanto, el momento en el que el servidor va a asumir una carga de trabajo adicional para adaptar el archivo si se trata de un archivo de datos.

Adición de archivos

Para permitir a una base de datos obtener más espacio, es posible añadir archivos. Esta solución presenta la doble ventaja de controlar el momento en el que el servidor va a sufrir una sobrecarga de trabajo y al mismo tiempo impedir la fragmentación física de los archivos, sobre todo si estos últimos se almacenan en una partición NTFS. Sin embargo, esta solución necesita que el administrador vigile de cerca la utilización de los archivos diarios y de datos para poder intervenir antes de que el sistema se bloquee por falta de espacio en disco.

El diario de las transacciones

El diario de las transacciones está compuesto por uno o varios diarios. Con el objetivo de que el servidor funcione correctamente, es indispensable que el diario no se sature nunca. El diario está vacío cuando se realizan copias de seguridad y a veces en cada punto de sincronización si la base se configura en modo de restauración simple.

Para asegurar una cantidad de espacio suficiente para el diario, la solución más fácil consiste en asignar a los archivos que lo componen un crecimiento automático.

Modificar un archivo en Transact SQL

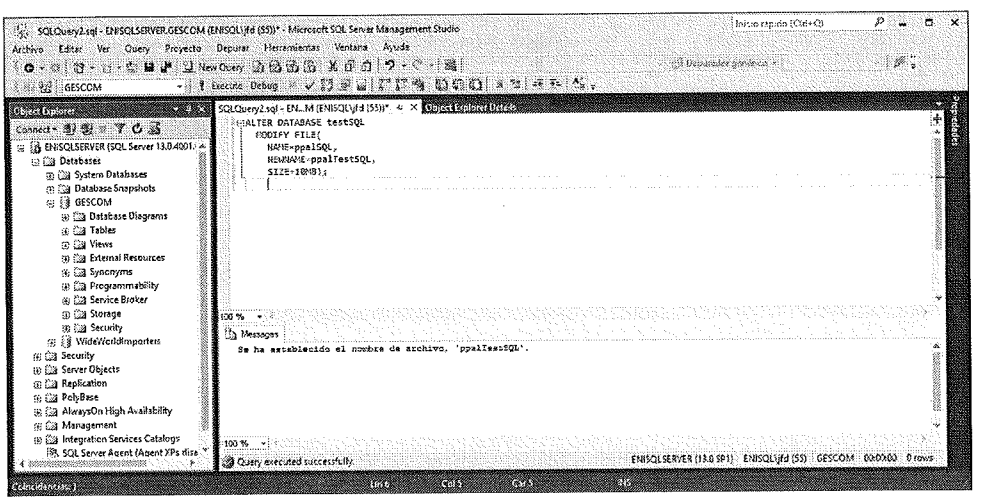
La instrucción ALTER DATABASE permite efectuar todas las operaciones relativas a la manipulación de los archivos de base de datos, tanto para los archivos de datos como para los archivos del diario de transacciones.

Se pueden modificar todas las características de los archivos, aunque las modificaciones realizadas deben seguir algunas reglas, como por ejemplo que el nuevo tamaño del archivo debe ser superior al tamaño inicial.

```
ALTER DATABASE nombreBaseDeDatos
MODIFY FILE (especificaciónArchivo [;]
```

Para `especificaciónArchivo` los elementos de sintaxis son los siguientes:

```
(NAME = nombreLógico,
NEWNAME = nuevoNombreLógico,
FILENAME = 'rutaYNombreArchivo'
[,SIZE = tamaño [KB|MB|GB|TB]]
[,MAXSIZE={tamañoMáximo[KB|MB|GB|TB]|UNLIMITED}]
[,FILEGROWTH = pasoIncremento [KB|MB|GB|TB|%]]
)
```

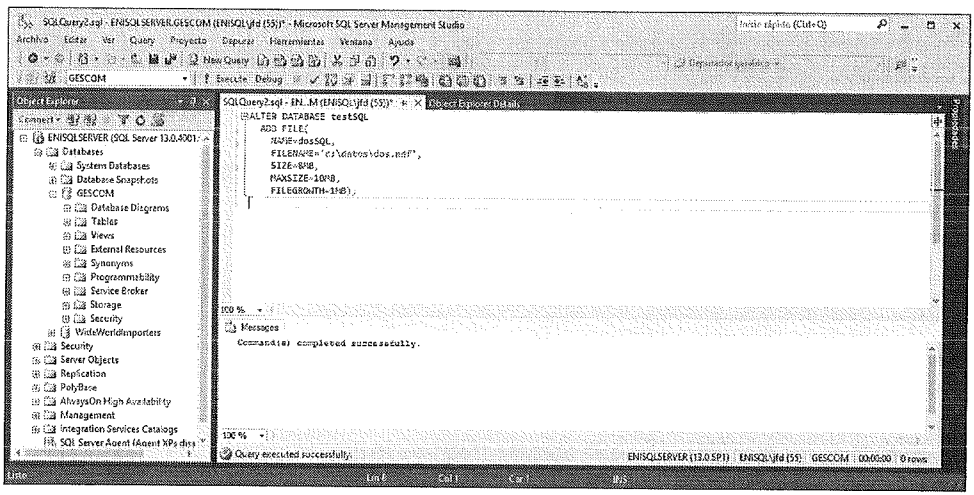


Añadir un archivo en Transact SQL

El comando ALTER DATABASE asociado a la opción ADD FILE permite añadir un archivo de datos o un archivo diario.

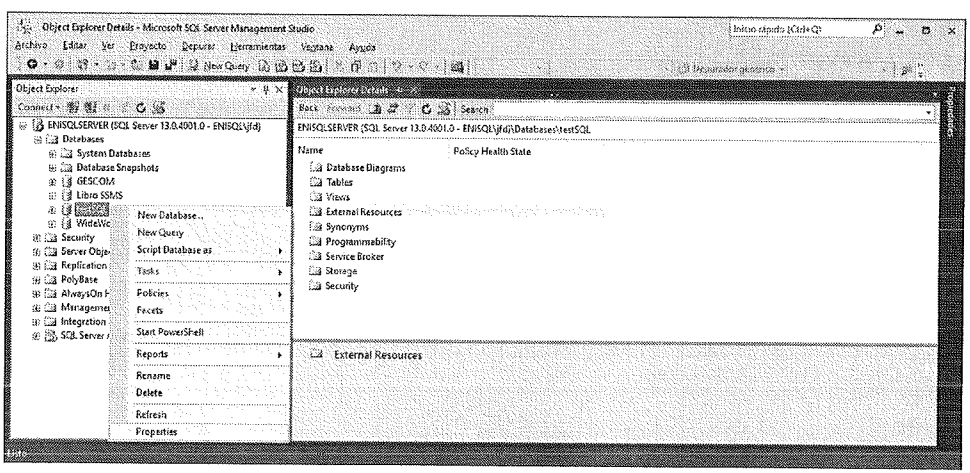
Sintaxis

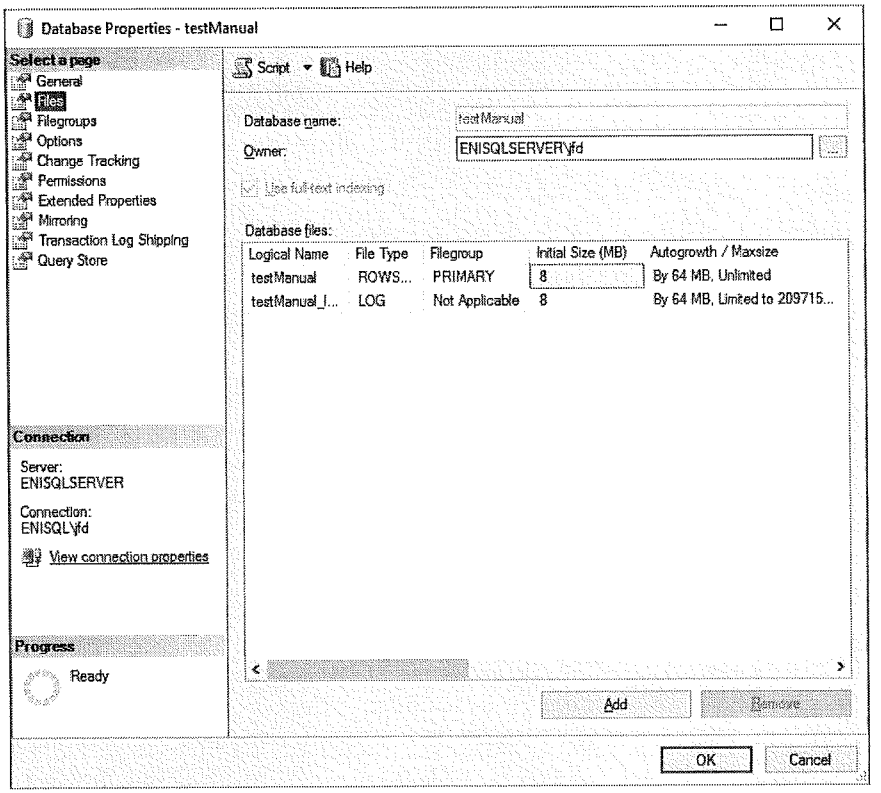
```
ALTER DATABASE nombreBaseDeDatos
ADD FILE especificaciónArchivo[;]
```



Modificar y añadir archivos desde SQL Server Management Studio

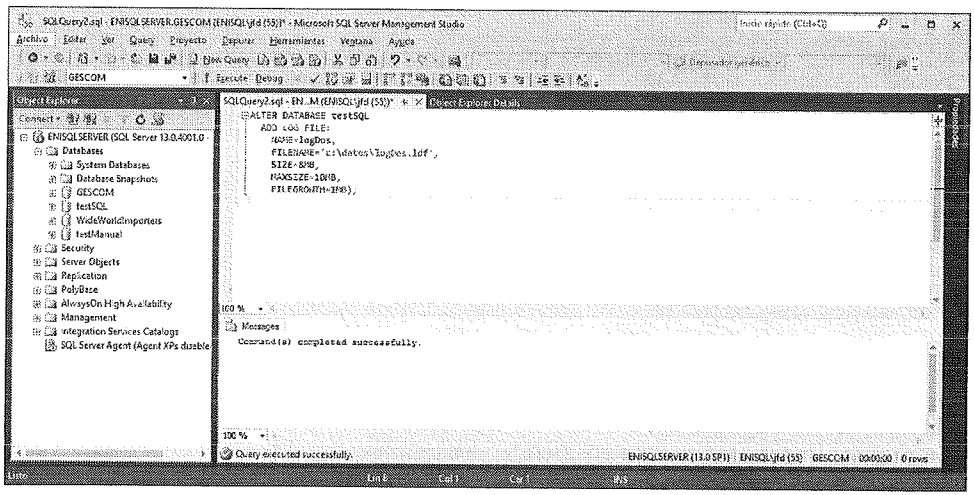
Mediante el cuadro de diálogo que indica las propiedades de la base de datos, es posible gestionar las operaciones sobre el tamaño de los archivos y sobre su propio tamaño.





Los archivos del diario

Igual que para los archivos de datos, es posible redimensionar el archivo diario, así como añadir un archivo al diario de transacciones. Si la modificación puede efectuarse para los archivos de datos por medio del comando ALTER DATABASE nombre-BaseDatos MODIFY FILE..., la adición necesita la utilización de la opción ADD LOG FILE. Evidentemente, igual que para los archivos de datos, todas estas operaciones pueden efectuarse desde SQL Server Management Studio.



2.2.2 Liberar el espacio en disco que usan los archivos de datos vacíos

Cuando las tablas se vacían de sus datos con los comandos DELETE o TRUNCATE TABLE, las extensiones ocupadas por las tablas e índices se liberan. Sin embargo, el tamaño de los archivos no se reduce. Para realizar esta operación es importante asegurarse de que la totalidad del espacio libre se reagrupa al final del archivo. Una vez que se realiza esta operación, es posible truncar el archivo sin situarse nunca por debajo del tamaño inicial.

Antes de reducir el tamaño de los archivos, es importante asegurarse de que la base de datos no necesitará este espacio en el futuro. Por ejemplo, si sabemos que a fin de mes o de trimestre un cierre cualquiera permite eliminar muchos registros de datos, no es necesario liberar espacio en disco. De hecho, durante el siguiente mes o trimestre se va a registrar información nueva que va a necesitar espacio en disco y los archivos se van a utilizar de nuevo. Si los archivos no contienen información, no afectan al tiempo de la copia de seguridad a nivel de SQL Server.

La aplicación de la reducción de los archivos se lleva a cabo utilizando dos comandos DBCC.

SHRINKDATABASE

Esta instrucción permite compactar el conjunto de archivos que conforman la base de datos (diarios y datos). Para los archivos de datos, todas las extensiones utilizadas se almacenan de manera contigua en la parte inicial del archivo. Para los archivos diarios, esta operación de compactación se realiza en diferido y SQL Server intenta dar a los archivos diarios un tamaño lo más cercano posible al tamaño alcanzado cuando los diarios se truncan.

Sintaxis:

```
DBCC SHRINKDATABASE {nombre_base_datos|id_base_datos|0}
[,porcentaje_determinado]
[,{NOTRUNCATE|TRUNCATEONLY}])
```

`nombre_base_datos`

Nombre de la base de datos sobre la que se va a ejecutar el comando.

`id_base_datos`

Identificador de la base de datos. Este identificador puede averiguarse mediante la ejecución de la función db_id() o bien consultando la vista sys.databases desde la base master.

`0`

Permite precisar que la ejecución se realizará sobre la base actual.

`porcentaje_determinado`

Permite precisar el porcentaje de espacio libre deseado en el archivo después de la compactación.

`NOTRUNCATE`

Permite no entregar al sistema operativo el espacio obtenido después de la compactación. Por defecto, el espacio obtenido de esta manera se libera.

`TRUNCATEONLY`

Permite liberar el espacio inutilizado en los archivos de datos y compactar el archivo a la última extensión asignada. No se prevé ninguna reorganización física de los datos: se desplazan las líneas de datos con el objetivo de completar lo mejor posible las extensiones utilizadas por el objeto. En este caso, el parámetro `porcentaje_determinado` se ignora.

SHRINKFILE

Esta instrucción, parecida a DBCC SHRINKDATABASE, permite realizar las operaciones de compactación y reducción de archivo de datos por archivo.

Sintaxis:

```
DBCC SHRINKFILE ([nombre_archivo|id_archivo]
{[[,tamaño_determinado]
[,{NOTRUNCATE|TRUNCATEONLY}]]|EMPTYFILE}
```

`tamaño_determinado`

Permite precisar el tamaño final deseado expresado en megabytes en forma de número entero. Si no se especifica ningún tamaño, el tamaño del archivo se reduce a su máximo.

`EMPTYFILE`

Este comando permite realizar la migración de todos los datos contenidos en el archivo hacia otros archivos del mismo grupo. Además, SQL Server ya no utiliza este archivo y, por tanto, es posible eliminarlo de la base de datos usando un comando ALTER DATABASE.

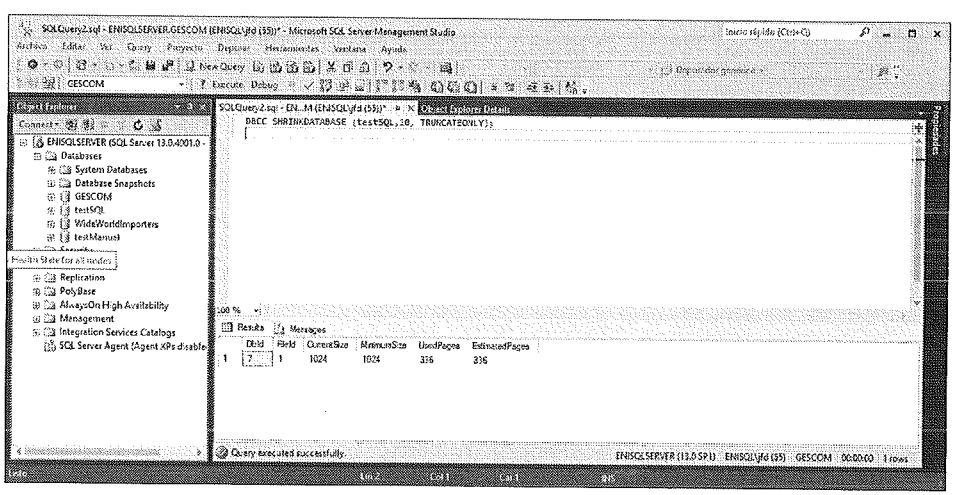
`NOTRUNCATE`

Permite no entregar al sistema operativo el espacio libre obtenido después de la compactación. Por defecto, el espacio obtenido de esta manera se libera.

`TRUNCATEONLY`

Permite liberar el espacio inutilizado en los archivos de datos y compactar el archivo a la última extensión asignada. No se prevé ninguna reorganización física de los datos: se desplazan las líneas de datos con el objetivo de completar lo mejor posible las extensiones utilizadas por el objeto. En este caso, el parámetro `tamaño_determinado` se ignora.

Ejemplo



El tamaño final debe ser superior al tamaño de la base **Model** más el tamaño de los datos.

Observación

Antes de realizar una operación de compactación, es prudente llevar a cabo una copia de seguridad completa de la base de datos que se va a compactar, así como de la base ***Master****.*

Las instrucciones DBCC SHRINKDATABASE y DBCC SHRINKFILE se ejecutan en modo diferido. Por lo tanto, es posible que el tamaño de los archivos no disminuya de manera instantánea.

Solo la instrucción DBCC SHRINKFILE permite reducir el tamaño de un archivo a un tamaño inferior al indicado en el momento de la creación del archivo. Evidentemente, no es posible bajar del tamaño ocupado por los datos.

2.2.3 Configuración de la base de datos

Es posible parametrizar numerosas opciones a nivel de base de datos. Esta parametrización se efectúa con la instrucción ALTER DATABASE en Transact SQL o desde la ventana de propiedades de la base de datos en SQL Server Management Studio. Todos los parámetros listados a continuación se deben fijar para cada base de datos. No es posible fijar las opciones de varias bases de datos con un único comando, pero sí se pueden precisar varias opciones de una base en un solo comando.

Si se debe hacer alguna selección para todas las bases de usuario, es preferible cambiar las opciones de la base de datos **Model**. De esta manera, toda base de datos de usuario nueva funcionará con los parámetros definidos en **Model**.

```
ALTER DATABASE nombreBaseDeDatos
SET opción [;]
```

Entre todas las opciones disponibles, se detallan las más interesantes:

Opción	Descripción
AUTO_SHRINK {ON|OFF}	Si esta opción está activada, los archivos se reducen cuando disponen de más del 25 % de espacio libre.
READ_ONLY	Permite poner la base de datos en modo de solo lectura.
READ_WRITE	La base de datos se pone en modo de lectura/escritura.
SINGLE_USER	Solo un usuario puede trabajar en la base de datos.

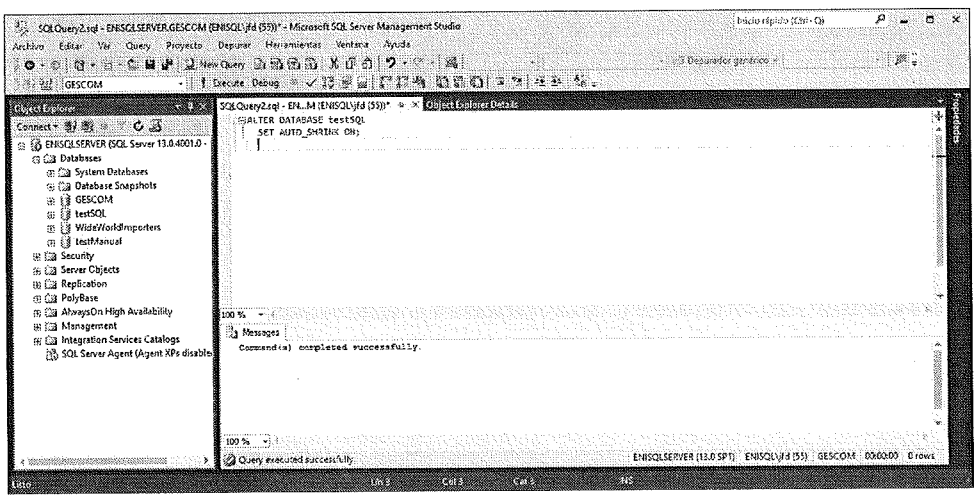
Opción	Descripción
RESTRICTED_USER	Solo se pueden conectar a la base de datos los usuarios miembros de los roles **db_owner**, **db_creator** y **sysadm**.
MULTI_USER	Es el modo de funcionamiento estándar de una base de datos, autorizando a varios usuarios a trabajar al mismo tiempo.
AUTO_CREATE_ STATISTICS { ON | OFF }	Cuando esta opción está en ON, se calculan de manera automática las estadísticas ausentes en el momento de la optimización de la consulta.
AUTO_UPDATE_ STATISTICS { ON | OFF}	Cuando esta opción está en ON, se calculan de manera automática las estadísticas obsoletas.

El uso de la opción AUTO_SHRINK puede ocasionar demasiadas operaciones de modificación del tamaño de los archivos de base de datos, por lo que, si la base de datos se administra en detalle, es mejor desactivar esta opción.

Las opciones AUTO_CREATE_STATISTICS y AUTO_UPDATE_STATISTICS se utilizan mucho, ya que permiten mantener actualizadas las estadísticas, midiendo la pertinencia de los índices. Cuando el optimizador de consultas traza un plan de ejecución, va a usar las estadísticas para determinar si el uso del índice permite o no ahorrar tiempo. Si las estadísticas no están actualizadas, el optimizador corre el riesgo de elegir un plan no óptimo, ya que la información de la que dispone inicialmente es errónea. Es posible imaginar esto con un GPS para coches. Cuando queremos ir de un punto A a un punto B, el GPS decide el itinerario que hay que seguir. La primera generación de GPS aplica este itinerario sin tener en cuenta las posibles dificultades en una dirección concreta. Este itinerario, aunque sea válido, no tiene por qué ser el más rápido porque no se dispone de estadísticas de uso de la red de carreteras. Pero si una segunda generación de GPS conoce en tiempo casi real las estadísticas de uso de la red de carreteras, entonces podrá elegir el itinerario más corto en tiempo para llegar al destino. Es evidente que esto tiene un coste, ya que el aparato necesita un tiempo para recibir, analizar y emitir esta información. En SQL Server pasa lo mismo: habilitar las opciones de creación y actualización de las estadísticas de manera automática consume recursos del servidor, pero permite trazar planes de ejecución más eficientes.

Observación

Las opciones AUTO_CREATE_STATISTICS y AUTO_UPDATE_STATISTICS no afectan a las tablas de sistema o al uso interno de SQL Server, como por ejemplo los índices XML, los índices de texto completo o las filas de espera de Service Broker. Para todas estas tablas, las estadísticas se crean y se actualizan sea cual sea el valor de las opciones AUTO_CREATE_STATISTICS y AUTO_UPDATE_STATISTICS.



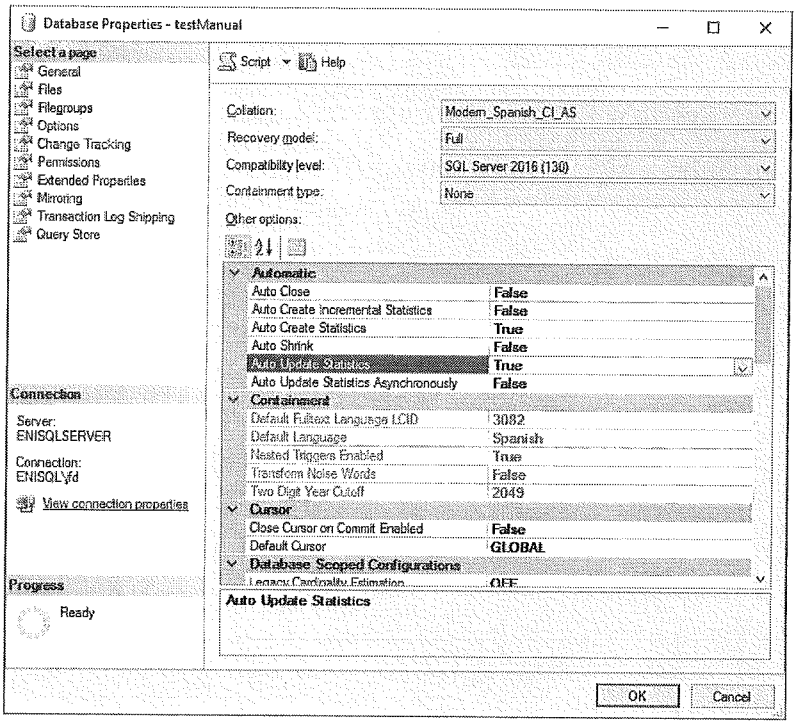
Observación

La función DATABASEPROPERTYEX permite conocer el valor actual de la opción que se le pasa por parámetro.

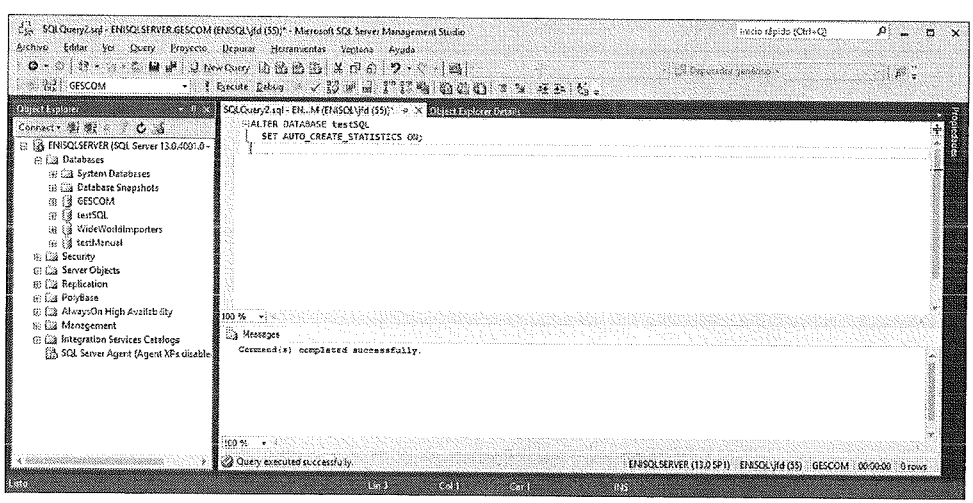
Para ajustar los parámetros de la base es posible utilizar:

SQL Server Management Studio

Para averiguar y si es preciso modificar los parámetros de una base de datos, es necesario visualizar la ventana que presenta las propiedades de la base. Esta ventana se muestra seleccionando **Properties** en el menú contextual asociado a la base de datos.



ALTER DATABASE

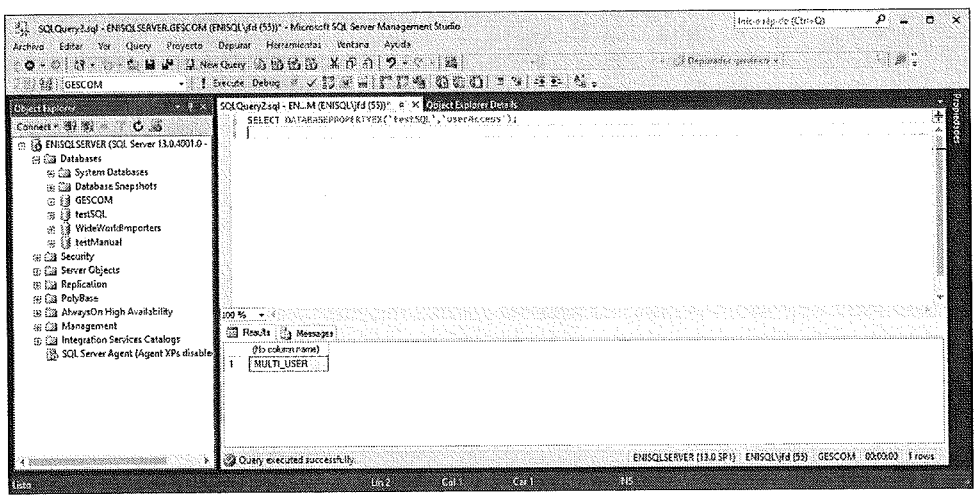


Opciones actualmente gestionadas

Antes de fijar una nueva parametrización de la base de datos, es importante averiguar la parametrización actual. El conocimiento de esta parametrización también puede ayudar a la comprensión del funcionamiento de la base. Es posible leer los valores de las diferentes opciones desde SQL Server Management Studio, aunque el conocimiento de la parametrización de la base también se puede hacer con scripts Transact SQL.

databasepropertyex

Para averiguar el valor de un parámetro.

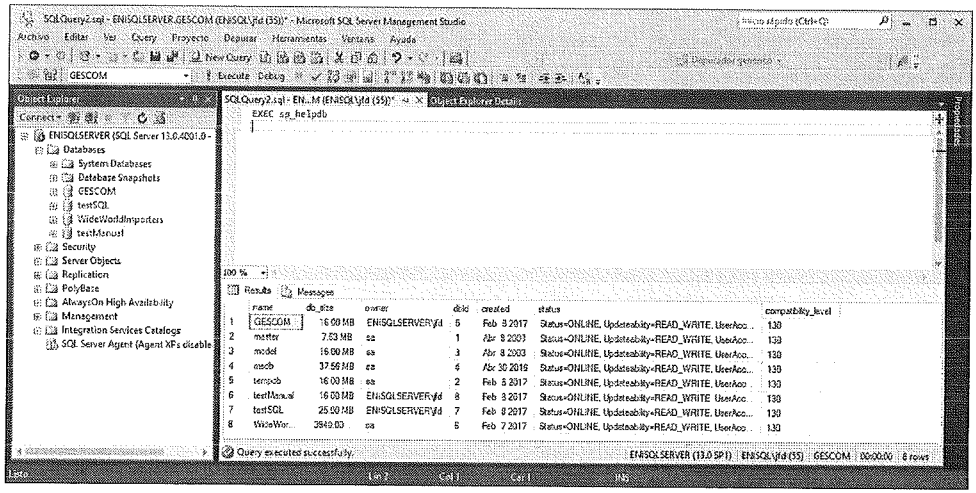


sp_helpdb

Si no se ha pasado ningún parámetro, este procedimiento permite averiguar el conjunto de bases de datos que existen en el servidor. Los datos proporcionados son el nombre, el tamaño, el propietario, el identificador, la fecha de creación y las opciones.

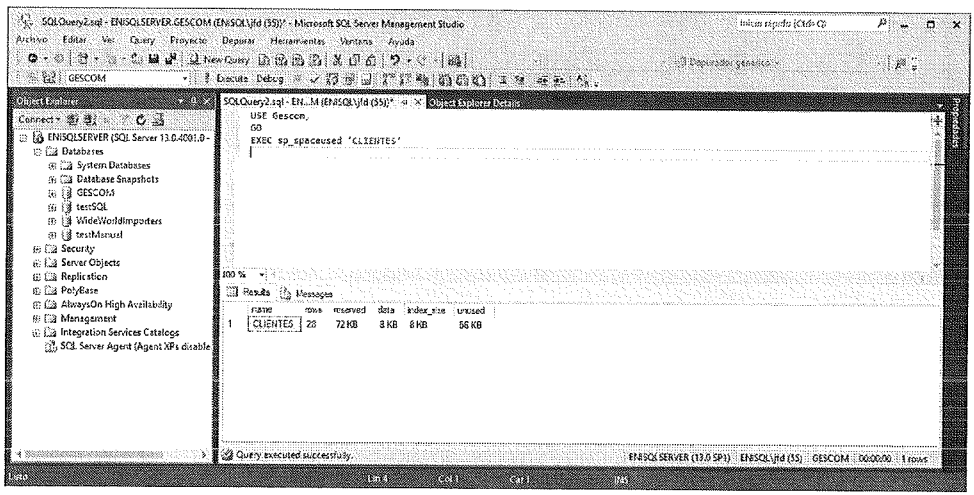
Si se pasa como parámetro un nombre de base de datos, el procedimiento permite averiguar la información general de la base de datos, así como el nombre y la ubicación de los archivos de datos y de los archivos de diario.

El procedimiento sp_helpdb utiliza la tabla **sys.databases** para establecer la lista de bases de datos y las diferentes opciones de cada una de ellas.



sp_spaceused [nombre_objeto]

Permite averiguar el espacio de almacenamiento utilizado por una base de datos, un diario o los objetos de la base (tablas...).

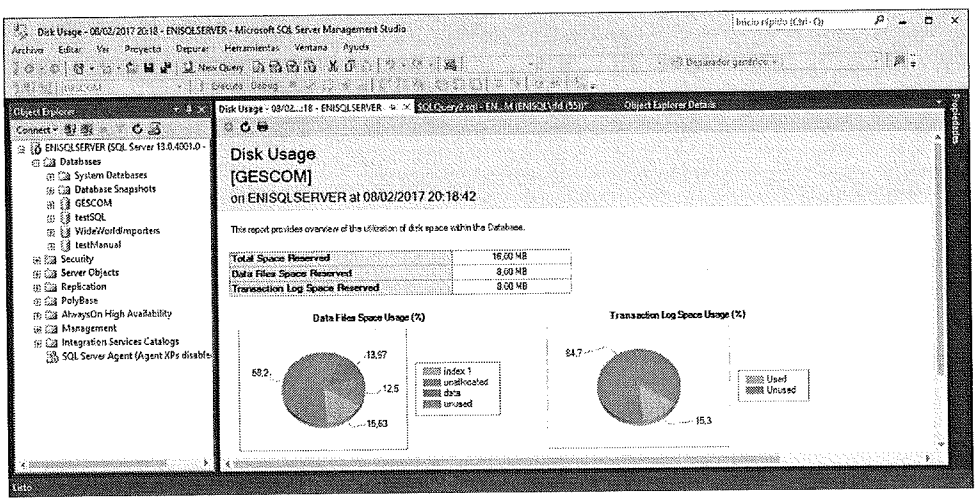


Es posible, en SQL Server Management Studio, obtener un informe sobre la utilización del espacio en disco por las diferentes tablas de la base de datos actual. Para ello, después de situarse sobre una base de datos determinada, hay que seleccionar la opción **Informes - Informes estándares**. A este nivel, se muestra la lista de informes predefinidos. Son numerosos, aunque solo algunos tienen que ver con el uso de disco. Si bien la noción de informe está disponible en apartados más específicos como Tablas o también a nivel de cada tabla, los informes predefinidos solo están disponibles a nivel de la base de datos.

En este caso, solo los cuatro primeros informes tienen que ver con el uso de disco.

Ejemplo

El siguiente informe presenta el uso de disco para la base de datos GESCOM.



2.3 Eliminar una base de datos

La eliminación de una base de datos de usuario es una operación puntual que puede realizarse, como la mayor parte de las operaciones de administración, con SQL Server Management Studio o con Transact SQL. La eliminación de la base de datos tiene como consecuencia la eliminación de todos los archivos que corresponden a la base, así como todos los datos contenidos en ella. Esta operación es irreversible y en caso de una manipulación errónea es necesario recuperar una copia de seguridad.

Observación

Después de la eliminación de una base de datos, cada conexión que utilizaba esta base predeterminada se queda sin base de datos predeterminada.

Observación

Con el objetivo de poder realizar posibles restauraciones del servidor, es importante efectuar una copia de seguridad de la base ***Master*** *después de eliminar una o varias bases de usuario.*

Los límites

Por supuesto, no siempre es posible eliminar una base. Los principales límites son:

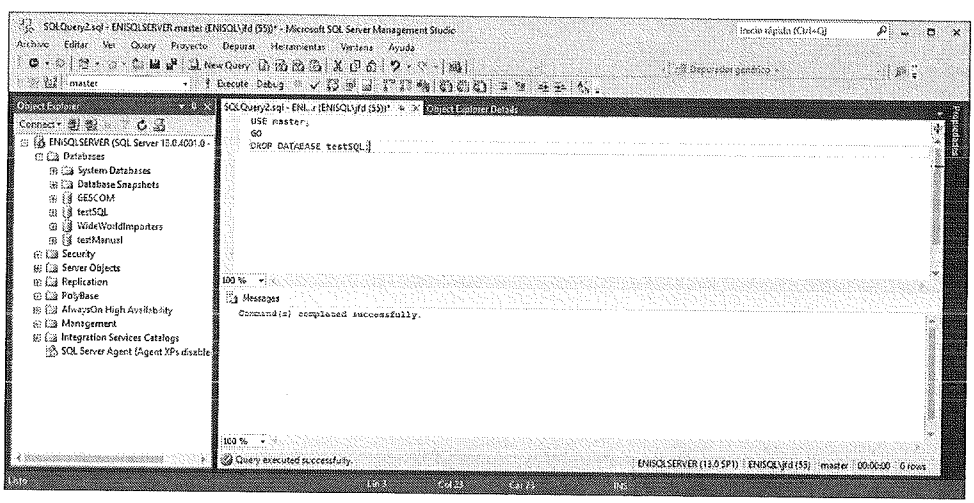
- Cuando está en curso de restauración.
- Cuando está abierta por un usuario, en lectura o en escritura.
- Cuando forma parte de una publicación en el marco de la replicación.

Observación

No es posible eliminar las bases de datos de sistema.

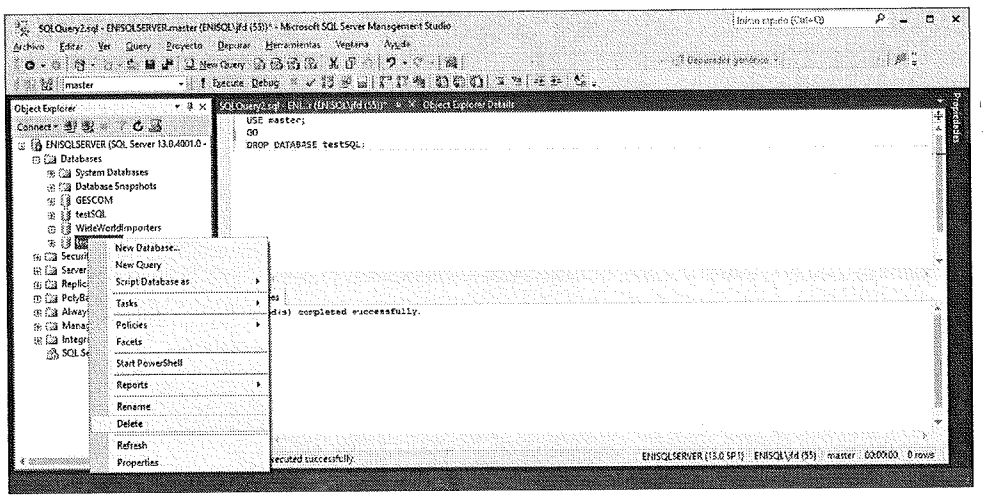
2.3.1 Transact SQL

El comando DROP DATABASE permite eliminar una base de datos.



2.3.2 SQL Server Management Studio

Con un simple clic del botón derecho del ratón sobre la base de datos, se consigue acceso al menú **Delete**, en el menú contextual. Con este método, solo es posible eliminar las bases de datos una a una.



Para eliminar una base desde SQL Server Management Studio, es necesario realizar las operaciones siguientes:

- Situarse en el explorador de objetos.
- Seleccionar el nodo correspondiente a la base de datos que se va a eliminar.
- Seleccionar la opción **Delete** desde el menú contextual asociado a la base de datos.

En el cuadro de diálogo que solicita la confirmación de la supresión, la opción **Close existing connections** existente no está activada. Entonces, si alguna persona se encuentra conectada a la base de datos, entonces la supresión no es posible.

La activación de esta opción permite, por el contrario, forzar la eliminación de la base de datos aunque existan conexiones activas al ejecutar el comando.

2.4 Las bases de datos de relación continente-contenido

SQL Server ofrece las bases de datos de relación continente-contenido. El objetivo de este tipo de base de datos es hacerlas independientes del servidor en el que se encuentren. De esta manera, es mucho más sencillo y rápido transferir la base de datos de un servidor a otro. Para que esta operación sea posible, se van a almacenar muchos metadatos directamente en la base de datos de sistema. Todos estos datos usan la misma clasificación. Otro aspecto importante de estas bases de datos es la autenticación, porque la puede hacer directamente la base datos. Los usuarios definidos en la base de datos continúan existiendo, incluso si la base de datos se transfiere a otro servidor.

Cuando se crea o modifica la base de datos, se puede definir la relación continente-contenido con la opción CONTAINMENT.

Sintaxis

```
CREATE DATABASE nombreBaseDatos CONTAINMENT={NONE|PARTIAL}...
```

Por defecto, las bases de datos que se crean con SQL Server o que pertenecen a una versión anterior se crean con la opción NONE. Esta opción especifica que no se trata de una base de datos de relación continente-contenido.

SQL Server ofrece únicamente la opción PARTIAL, que autoriza a definir relaciones, procedimientos que no están totalmente contenidos en la base de datos porque hacen referencia a objetos externos o están almacenados en una base de datos de sistema.

Sin embargo, este tipo de base de datos tiene algunas limitaciones, como, por ejemplo, no poder formar parte de un procedimiento de replicación, como resultado de las modificaciones o incluso de un proceso de captura.

3. Establecer grupos de archivos

Es posible precisar los archivos de datos que usa la base de datos, aunque desafortunadamente es imposible indicar sobre qué archivo se crea un objeto particular. Para resolver este problema, existe la posibilidad de crear una tabla o un índice sobre un conjunto de archivos. Este conjunto de archivos, también llamado **grupo de archivos**, se gestiona de manera muy sencilla.

¿Por qué es necesario trabajar con varios grupos de archivos? Porque en un sistema operativo es normal separar los archivos de sistema de los programas y datos de usuario. ¿Por qué lo será en una base de datos? Conviene reunir en un mismo grupo de archivos los datos del mismo tipo. Por ejemplo, los datos estables (como clientes, artículos), los que evolucionan (como pedidos, facturas...) simplemente porque los volúmenes no son los mismos y la manera de trabajar con ellos, tampoco.

También puede ser interesante definir los índices y las tablas sobre grupos de archivos distintos para reducir los tiempos de actualización de los datos y de los índices. En el caso de datos sensibles, el reparto por grupos de archivos también se puede regir por la política de copias de seguridad adoptada.

3.1 Creación de un grupo de archivos

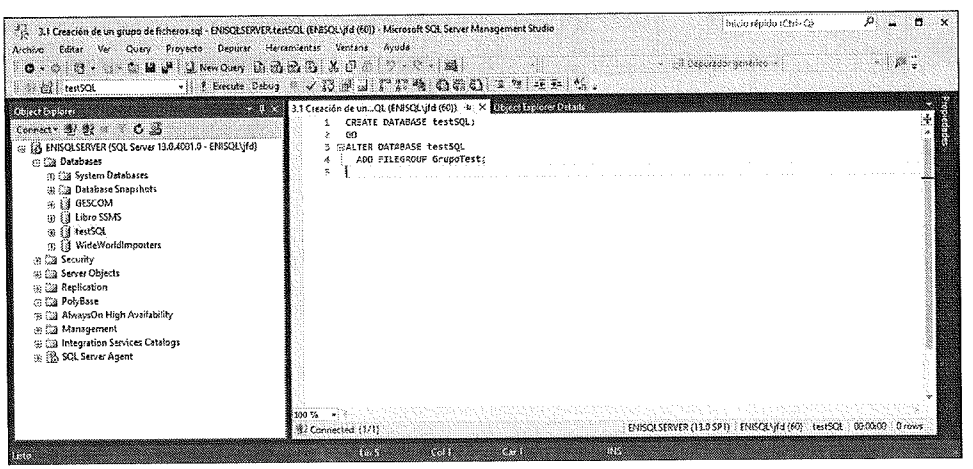
Antes de poder utilizar los grupos de archivos, es necesario crearlos. En el momento de la creación de la base, se crea el grupo de archivos PRIMARY.

El comando ALTER DATABASE permite crear un grupo de archivos. Este comando va a permitir añadir los archivos en el interior del grupo.

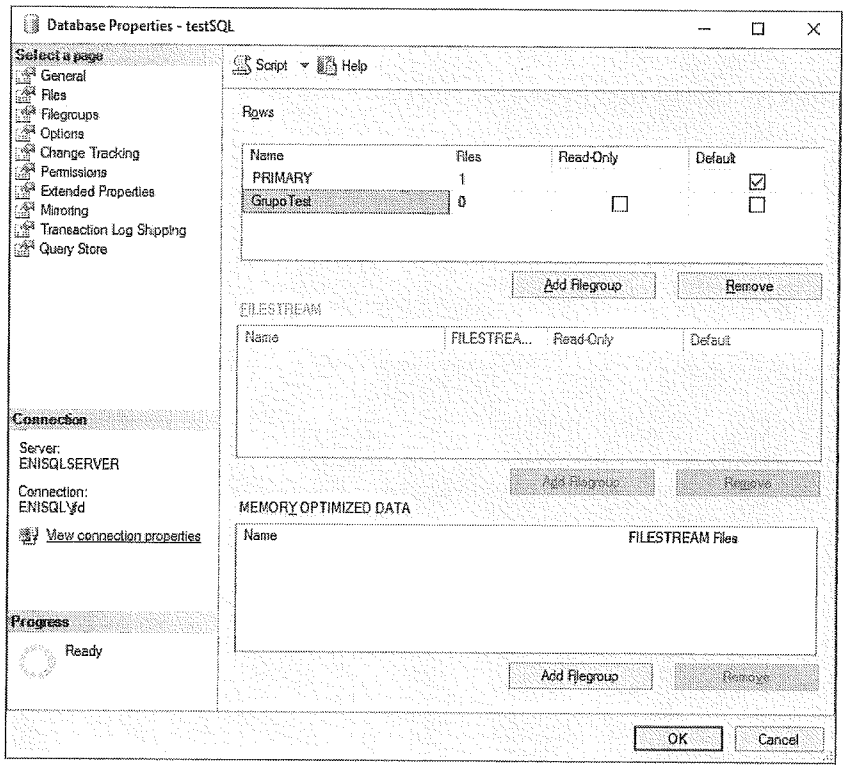
Sintaxis

```
ALTER DATABASE nombre_base_datos
ADD FILEGROUP nombre_grupo_archivos[;]
```

Ejemplo



Desde SQL Server Management Studio, la gestión de los grupos de archivos se realiza por medio de la ventana de propiedades de la base de datos.



3.2 Añadir archivos

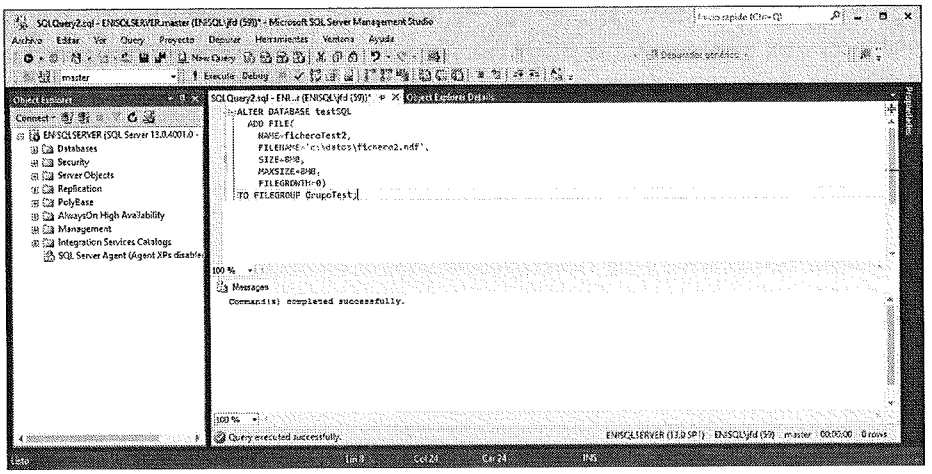
La adición de archivos se realiza con el comando ALTER DATABASE.

Un archivo solo puede pertenecer a un único grupo de archivos. Por defecto, si no se precisa ningún grupo de archivos en el momento de la adición del archivo a la base, este se añade al grupo PRIMARY.

Sintaxis

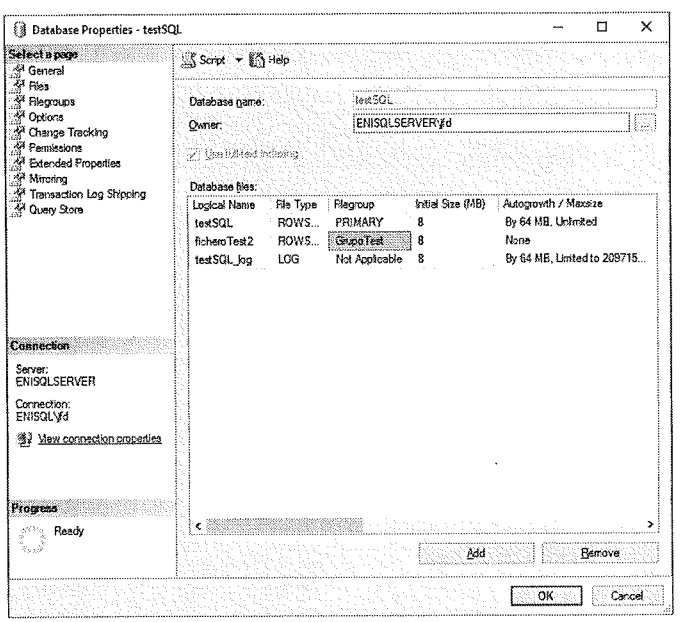
```
ALTER DATABASE nombre_base_datos
ADD FILE especificación_archivo
TO FILEGROUP nombre_grupo_archivos
```

Ejemplo



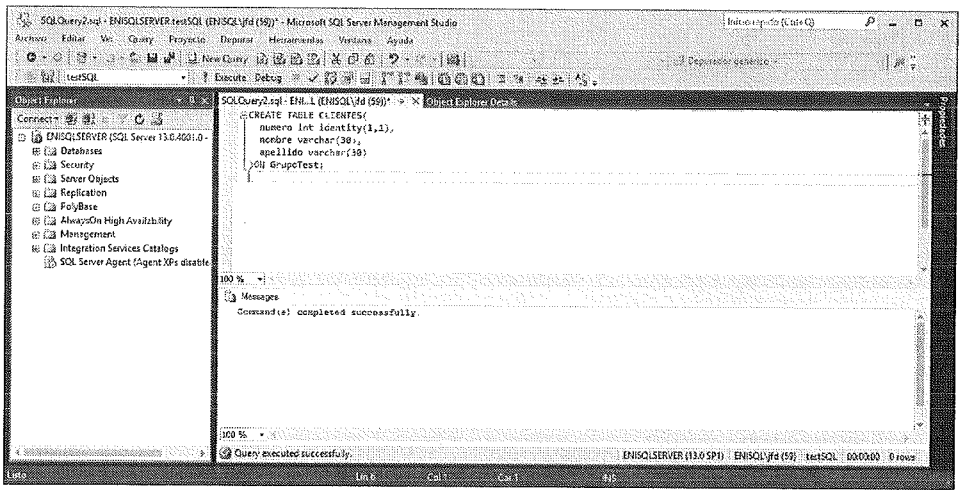
Los archivos que forman parte de un grupo de archivos se gestionan de la misma manera que los que forman parte del grupo de archivos **Primary**.

Desde SQL Server Management Studio, la gestión de los archivos se realiza desde la ventana de las propiedades de la base de datos.



3.3 Utilización de un grupo de archivos

La utilización de un grupo de archivos por parte de un objeto debe especificarse en el momento de la creación de este último. Solo pueden utilizarse los objetos que contienen información, ya sean tablas o índices. La instrucción ON nombre_grupo_archivos se añade al final del comando SQL y antes de su ejecución. Un archivo solo puede pertenecer a un único grupo de archivos.



Observación

Por defecto, los objetos se crean en el grupo PRIMARY.

El procedimiento `sp_help` permite conocer los detalles de la tabla y, entre otros, el grupo de archivos asociado.

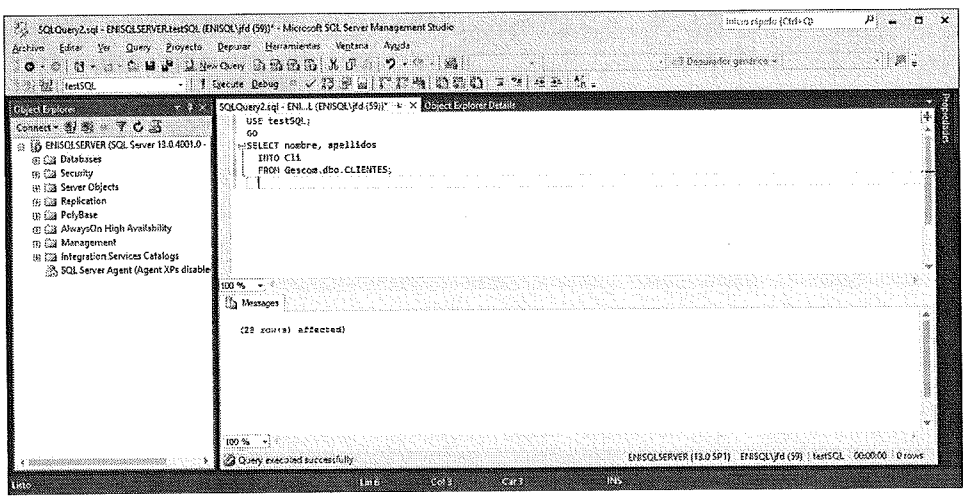
También se puede conocer el grupo de un archivo usado por una tabla mostrando la página **Storage** desde el cuadro de diálogo que muestra las propiedades de una tabla.

4. Instrucciones INSERT, SELECT... INTO

La instrucción SELECT INTO permite proyectar el resultado de un comando SELECT en una tabla. La tabla se crea en ese momento y de manera dinámica para contener el resultado de SELECT.

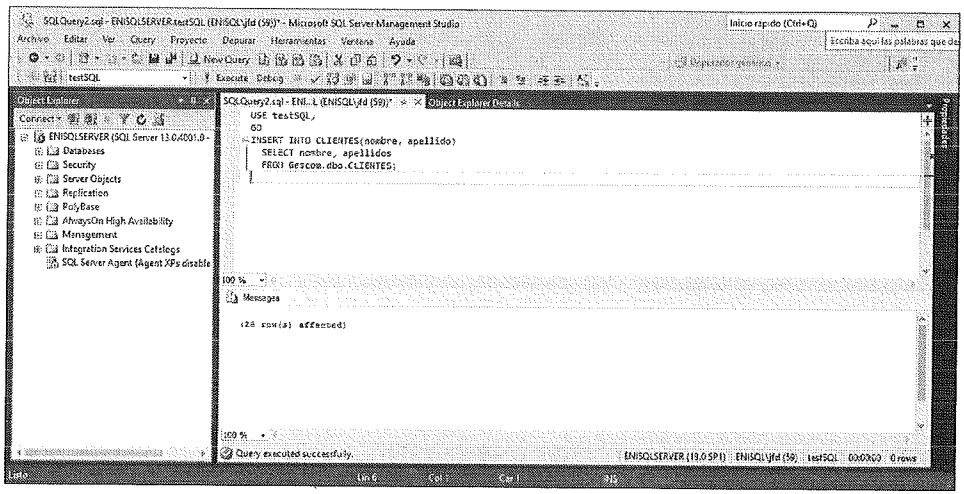
Observación

La tabla creada no dispone de ninguna restricción de integridad.



El comando INSERT también es capaz de tener en cuenta el resultado de un comando SELECT para proyectarlo en una tabla ya creada anteriormente mediante la instrucción SQL DDL CREATE TABLE.

En esta ocasión, es perfectamente posible incluir la definición de restricciones de integridad en el momento de la creación de la tabla.



5. Estructura de los índices

SQL Server ofrece dos tipos de índices:

- Los índices organizados o clúster.
- Los índices no organizados o no clúster.

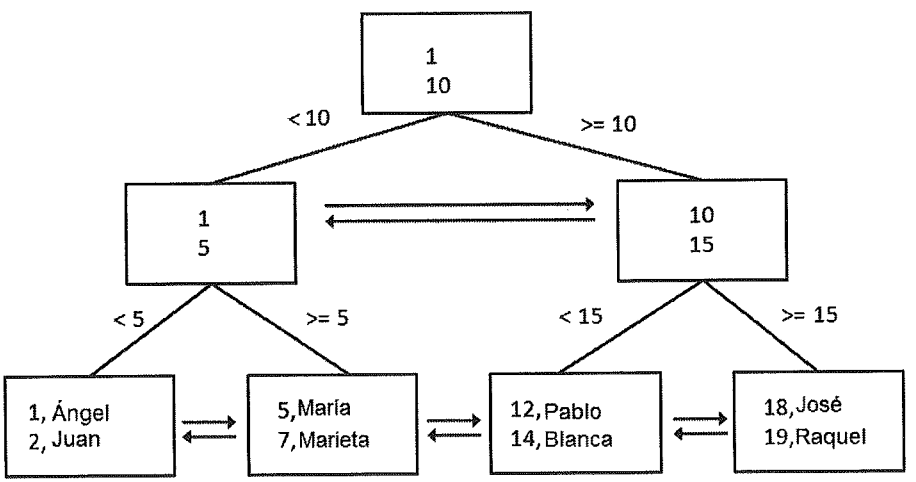
Habida cuenta de que el índice organizado (u ordenado) organiza físicamente los datos almacenados en la tabla, por lo general está asociado a la clave primaria de la tabla, ya que se trata de datos estables y poco voluminosos. Sin embargo, no es obligatorio. Algunas veces puede ser útil organizar según otro criterio, por ejemplo cuando la clave primaria está desprovista de significado. Cada tabla tiene a lo sumo un índice organizado. Los índices no organizados no afectan a la estructura física de la tabla. En cambio, como se basan en la organización física de los datos, es necesario definirlos en un momento posterior.

5.1 Los índices ordenados

Estos índices organizan físicamente la tabla y están formados por un árbol en el que las páginas del nivel de las hojas contienen los datos de la tabla subyacente. Los niveles superiores del árbol permiten ordenar la información en función del valor indexado. Cuando se añade una línea de información, esta se inserta en función del valor de su clave.

Habida cuenta de que la clave del índice organiza físicamente la tabla, es necesario basar este índice en un valor estable, y por eso tradicionalmente se mantiene el índice por clave primaria.

El esquema siguiente ilustra de manera sintética la estructura de un índice ordenado. Además de las uniones que permiten recorrer el árbol de abajo hacia arriba (desde la raíz a las hojas), existe una doble unión que permite recorrer todas las páginas de un mismo nivel.



El índice ordenado se crea por defecto cuando se define una restricción de clave primaria sobre una tabla. Si el administrador desea que la definición de la restricción no vaya acompañada de la creación de un índice tal, es necesario especificar la palabra clave NONCLUSTERED cuando se define la restricción.

Sintaxis

```
ALTER TABLE nombreTabla
ADD CONSTRAINT nombreRestricción PRIMARY KEY
[CLUSTERED|NONCLUSTERED] (listaColumnas);
```

La segunda posibilidad es definir un índice con la opción CLUSTERED.

Sintaxis

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX nombreÍndice
ON nombreTabla (listaColumnas)
[ON grupoDeArchivos];
```

5.2 Los índices no ordenados

El otro tipo de índices que es posible definir en SQL Server tiene que ver con los índices llamados NONCLUSTERED. Es decir: la definición de estos índices no reorganiza físicamente la tabla. De esta manera, es posible definir varios índices de este tipo en una misma tabla.

Observación

No es posible definir más de 249 índices no ordenados sobre una misma tabla.

Si se planifica la creación de un índice ordenado (CLUSTERED) para una tabla, es recomendable que esta definición intervenga antes de la definición de los índices no ordenados (NONCLUSTERED).

Igual que para los índices ordenados, estos índices pueden contener una o varias columnas, con una ordenación ascendente (por defecto) o descendente para cada columna. El orden se especifica con los caracteres **ASC** para ascendente o bien **DESC** para descendente detrás del nombre de cada columna.

Contrariamente al índice ordenado, que debe crearse sobre datos relativamente estables, el índice no ordenado puede definirse sin tener en cuenta la estabilidad de los valores indexados. Es más bien la utilización que se haga de los datos lo que va a permitir definir estos índices. Las operaciones de ordenación y de unión se pueden acelerar notablemente gracias a la definición del índice.

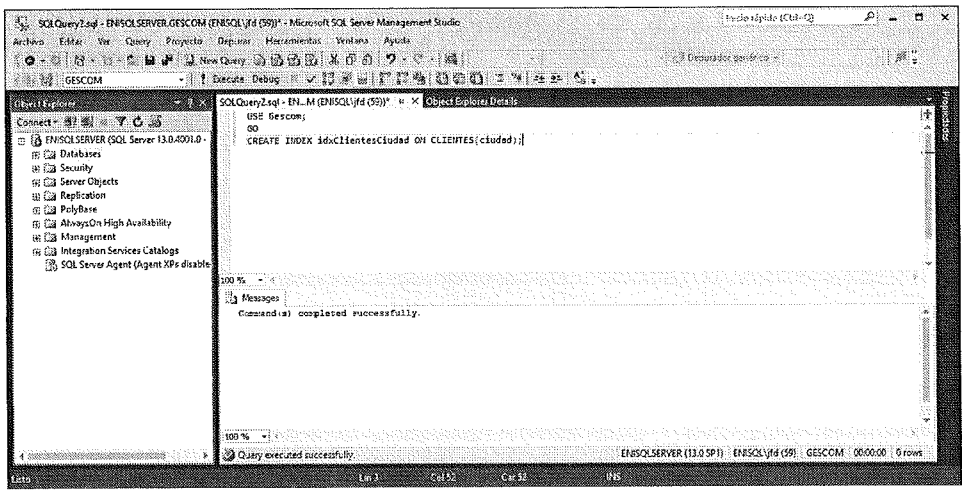
Si se define un índice ordenado sobre una tabla que tiene índices no ordenados, los índices no ordenados se reconstruyen.

Observación

Los criterios que permiten definir o no los índices pueden establecerse a través del asistente de parametrización de base de datos. La utilización de esta herramienta se detalla en el capítulo Herramientas adicionales.

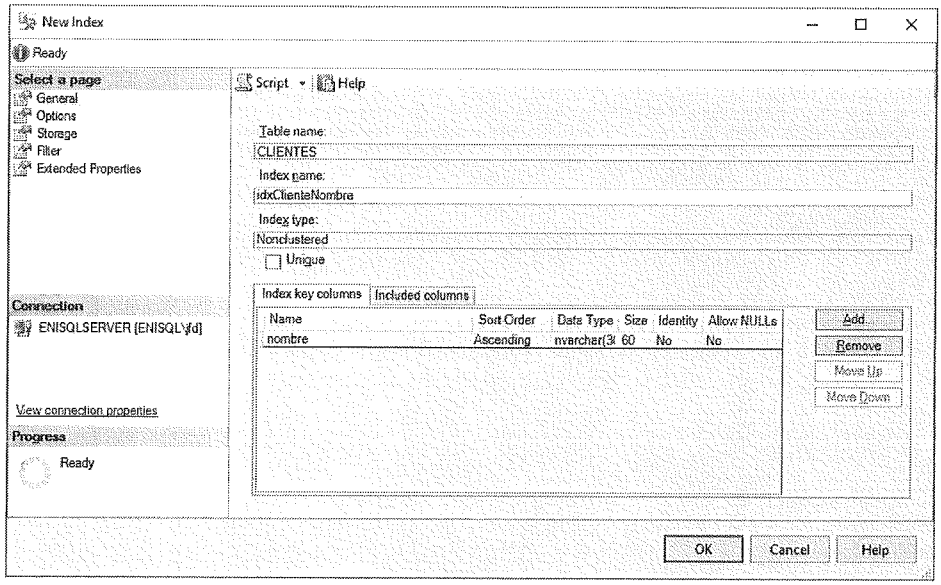
Ejemplo

El ejemplo siguiente ilustra la definición de un índice sobre la columna ciudad de la tabla clientes.



También es posible definir este tipo de índices directamente desde SQL Server Management Studio.

Hay que seleccionar el menú **New Index** - **Non-Clustered Index** desde el menú contextual asociado al nodo índice en la descripción de la tabla que hace el explorador de objetos.



5.3 Los índices de recubrimiento

Esta vez se trata de una particularidad de SQL Server que consiste en definir índices que van a contener, a nivel de las hojas, la clave del índice, así como los valores resultantes de una o varias columnas. El objetivo de estos índices es permitir al motor SQL Server recorrer únicamente el índice, sin que sea necesario acceder a la tabla para responder a las necesidades de datos de una consulta.

En términos del volumen de datos manipulados, la ganancia puede ser consecuente, pero además se pondera con relación al volumen de disco ocupado y también al tiempo adicional necesario para llevar a cabo las acciones de actualización (INSERT, UPDATE y DELETE).

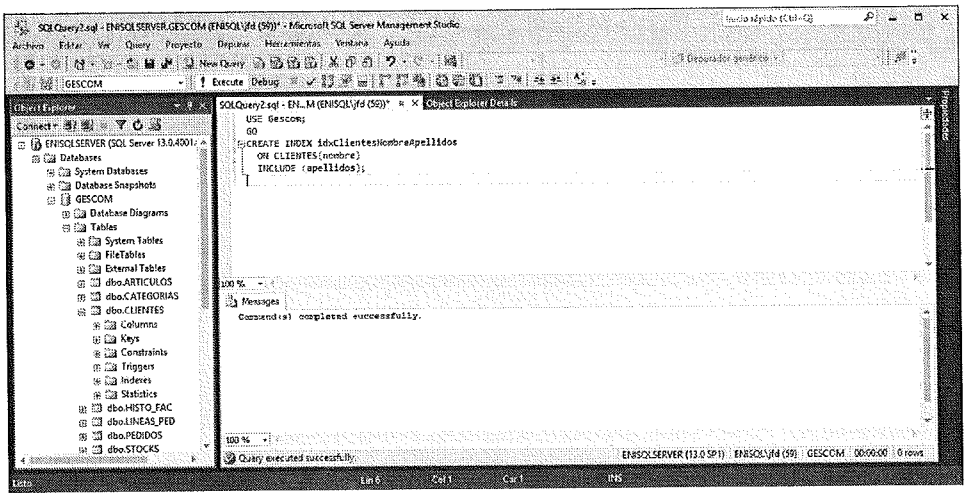
Los índices de este tipo se definen con la ayuda de la instrucción CREATE INDEX seguida de la palabra clave INCLUDE para precisar la(s) columna(s) que se debe(n) incluir en el nivel de las columnas.

Sintaxis

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX nombreÍndice
ON nombreTabla (listaColumnas)
INCLUDE (listaColumnas)
[ON grupoDeArchivos];
```

Ejemplo

En el ejemplo siguiente, se define un índice sobre el apellido del cliente y el nombre se incluye en el nivel de la hoja.

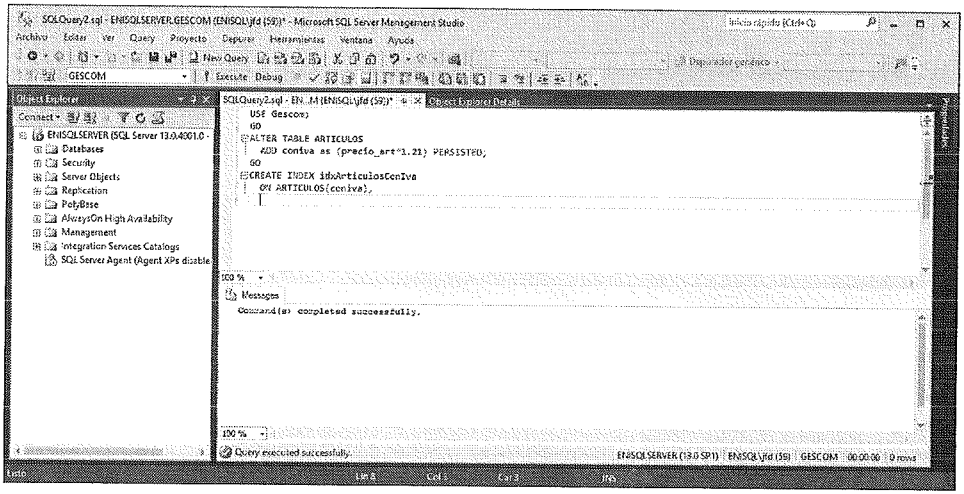


5.4 Indexar las columnas calculadas

Siempre con el objetivo de responder de manera rápida a los usuarios, es posible definir columnas calculadas en una tabla. Sin embargo, para poder definirlas, el cálculo deberá ser elemental, es decir, un cálculo realizado sobre cada línea de datos, y no como resultado de una agrupación. Todos los datos deben provenir de la misma tabla y la función de cálculo debe ser determinista.

En este caso, es posible definir un índice sobre estas columnas calculadas.

Ejemplo



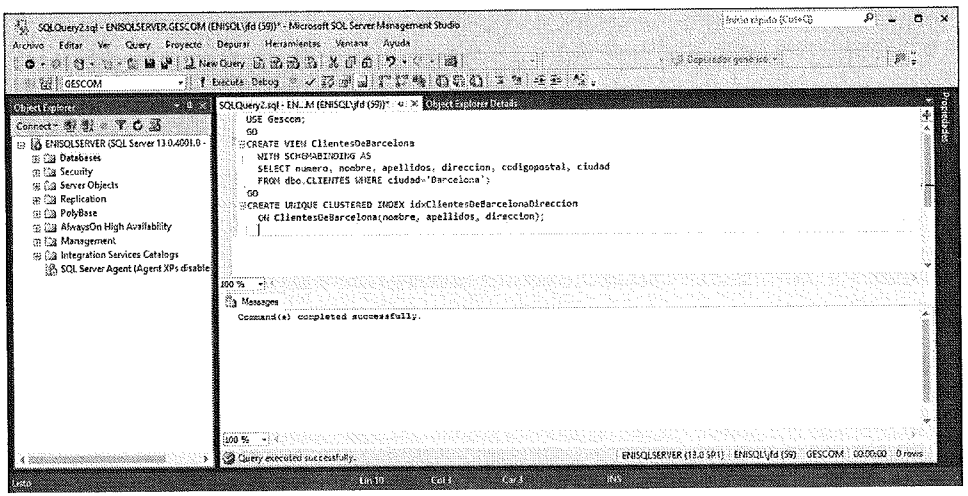
5.5 Indexar las vistas

Las vistas se utilizan frecuentemente en las consultas de extracción, ya que permiten, entre otras cosas, simplificar la escritura de las consultas. Para mejorar el rendimiento de las consultas que utilizan vistas, es posible definir uno o varios índices sobre las vistas.

El punto de partida consiste en definir un índice ordenado (CLUSTERED) único con el objetivo de materializar la vista. Seguidamente, se pueden definir otros índices no ordenados sobre la vista. Las columnas que presentan el resultado de un cálculo también pueden ser indexadas.

Las sintaxis de definición de un índice sobre una tabla o sobre una vista son idénticas.

Ejemplo

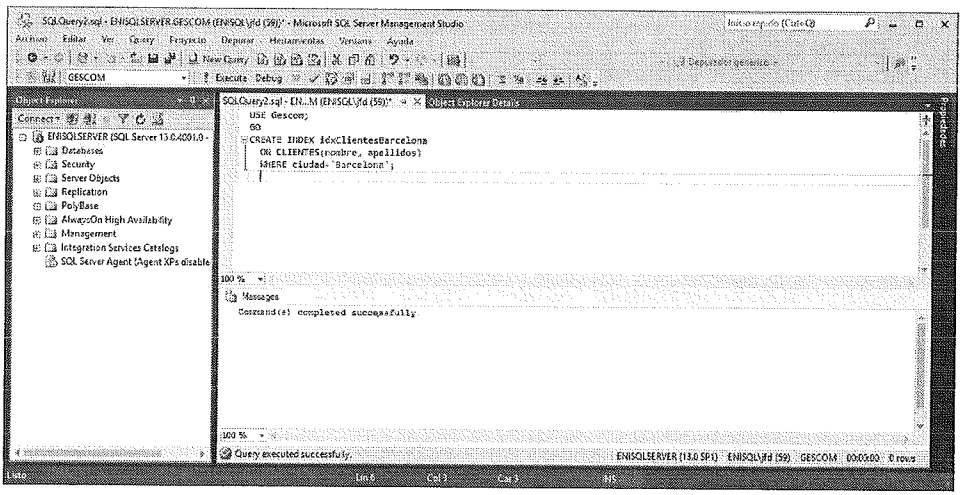


5.6 Los índices filtrados

Los índices filtrados no son un nuevo tipo de índice, sino una solución ofrecida por SQL Server para permitir definir los índices limitando el espacio ocupado en el disco duro por los índices y, por tanto, el tiempo de actualización de los índices. Durante la definición del índice filtrado, se añade una cláusula WHERE a la definición del índice para que este no afecte a todos los registros de la tabla, sino solo a los que cumplen los criterios de selección. De esta manera, es posible definir índices solo en algunos casos en los que las agrupaciones son numerosas.

Ejemplo

Para la tabla de clientes, se define un índice sobre los nombres y apellidos, solo para los clientes que viven en Barcelona.



5.7 Los índices XML

Como sucede con las otras columnas, es posible indexar las columnas de tipo XML. Sin embargo, la indexación de datos XML depende de la propia estructura de los datos. Al tratar una consulta, la información XML se analiza en cada línea, lo que puede dar lugar a tratamientos largos y costosos si el número de líneas es importante o cuando la información en formato XML es abundante.

El mecanismo habitual de indexación, que se basa en un árbol balanceado, va a utilizarse para definir el índice llamado principal sobre la columna de tipo XML. Pero los índices que van a permitir acelerar el tratamiento de las consultas son los índices que se basan en este índice principal. Estos índices secundarios se definen con relación a los tipos de consultas ejecutadas con frecuencia:

- Índice PATH para consultas ejecutadas sobre la ruta de acceso.
- Índice PROPERTY para consultas sobre las propiedades.
- Índice VALUE para consultas sobre los valores.

Observación

También es posible definir un índice de tipo texto completo sobre las columnas de tipo XML.

5.7.1 Índice principal

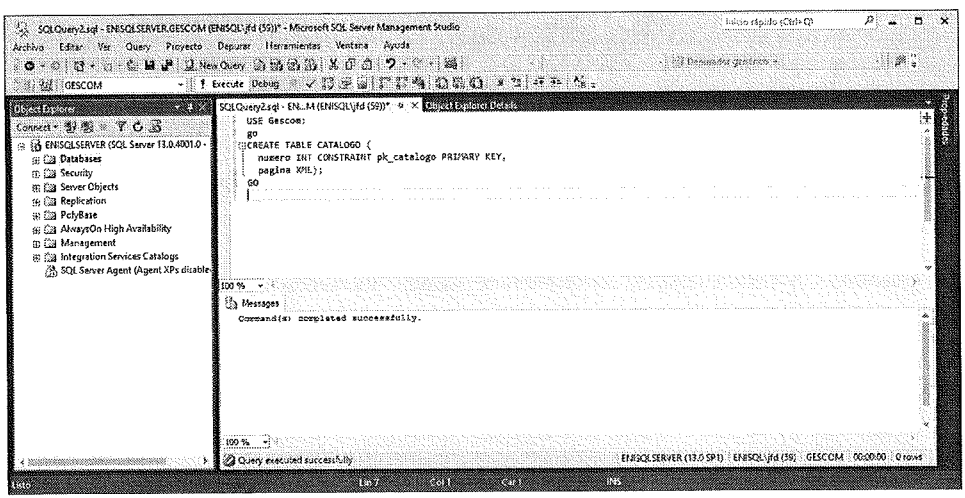
El índice principal representa el punto de partida de toda indexación de una columna de tipo XML. Solo se puede definir sobre una tabla que tenga una restricción de clave primaria asociada a un índice que organice físicamente la tabla, lo que suele ser frecuente.

Sintaxis

```
CREATE PRIMARY XML INDEX nombreÍndice ON tabla(columnaXML)[;]
```

Ejemplo

Un índice principal se define sobre la columna pagina de la tabla catalogo.



5.7.2 Índice secundario

La definición de un índice secundario solo es posible si y solo si existe un índice primario ya definido sobre la columna.

El documento XML solo puede contener 128 niveles como máximo. Los documentos que tienen una jerarquía más compleja son rechazados en el momento de la inserción o la modificación de las columnas.

En lo que respecta a la indexación, esta afecta a los 128 primeros bytes del nodo. Los valores más largos no se toman en cuenta en el índice.

Sintaxis

```
CREATE XML INDEX nombreÍndice ON tabla(columnaXML)
USING XML INDEX nombreÍndiceXMLPrincipal
FOR {PATH|PROPERTY|VALUE}[;]
```

PATH

Permite construir un índice sobre las columnas path y value (ruta y valor) del índice XML principal. Este índice puede mejorar sensiblemente los tiempos de respuesta en la utilización del método **exist()** en una cláusula where, por ejemplo.

PROPERTY

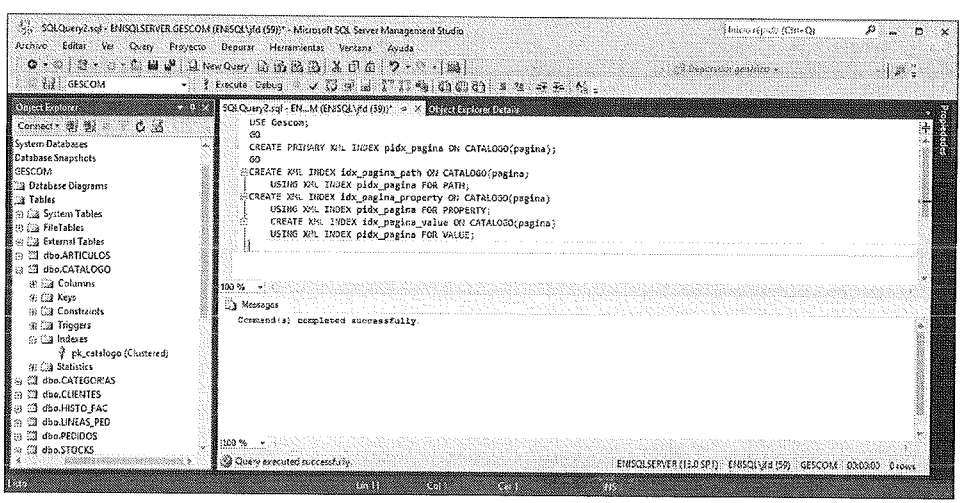
Permite construir un índice sobre las columnas PK, path y value del índice XML principal. El símbolo PK corresponde a la clave primaria de la tabla. Este tipo de índice es útil al utilizar el método **value()** en las consultas SQL de manipulación de datos.

VALUE

Permite construir un índice sobre las columnas value y path del índice XML principal. Este tipo de índices se utiliza en las consultas para las que el valor del nodo se conoce independientemente de la ruta de acceso. Este puede ser el caso al utilizar el método **exist()**, por ejemplo.

Ejemplo

En el ejemplo que se presenta a continuación, se crean los tres tipos de índice en relación con el índice XML principal definido sobre la columna page de tipo XML de la tabla Catalogo.



5.8 Los índices espaciales

SQL Server permite almacenar datos espaciales de tipo geométrico o geográfico. Como para toda la información conservada en SQL Server, el motor debe proporcionar un acceso rápido a esta información. Para alcanzar este objetivo, los índices tienen un papel crucial. Por lo tanto, es lógico que SQL Server ofrezca la posibilidad de indexar las columnas de tipo espacial.

La estructura de estos índices difiere de la de los índices clásicos. Para este tipo de índice, SQL Server define una estructura compatible con los datos espaciales estableciendo una matriz de cuatro niveles con el objetivo de acceder rápidamente a la celda deseada. Cada nivel corresponde a una matriz definida sobre 16, 64 o 256 celdas. Cada celda de un nivel se detalla por una casilla de nivel inferior. Por ejemplo, si se decide trabajar con una casilla de 16 celdas en el nivel 1, entonces el nivel 4 (el más detallado) tendrá 65.536 celdas. El número de celdas definidas en las casillas de los diferentes niveles representa la densidad del índice. Esta densidad puede tomar los valores LOW (16 celdas), MEDIUM (64 celdas) o bien HIGH (256 celdas).

Las zonas geográficas contenidas en la tabla se indexan así y el recorrido de los diferentes niveles del índice permite localizar rápidamente la celda o las celdas que contienen la zona objetivo de la búsqueda.

Para limitar la extensión de las casillas de índice, en el momento de la definición del índice, la zona geográfica que se debe tener en cuenta para la indexación se define con la opción BOUNDING BOX.

Sintaxis

```
CREATE SPATIAL INDEX nombreÍndice
ON nombreTabla(nombreColumna)
WITH (
BOUNDING_BOX=(xMin, yMin, xMax, yMAX),
GRIDS(LEVEL_1=densidad1, LEVEL_2=densidad2, LEVEL_3=densidad3,
LEVEL_4=densidad4)
);
```

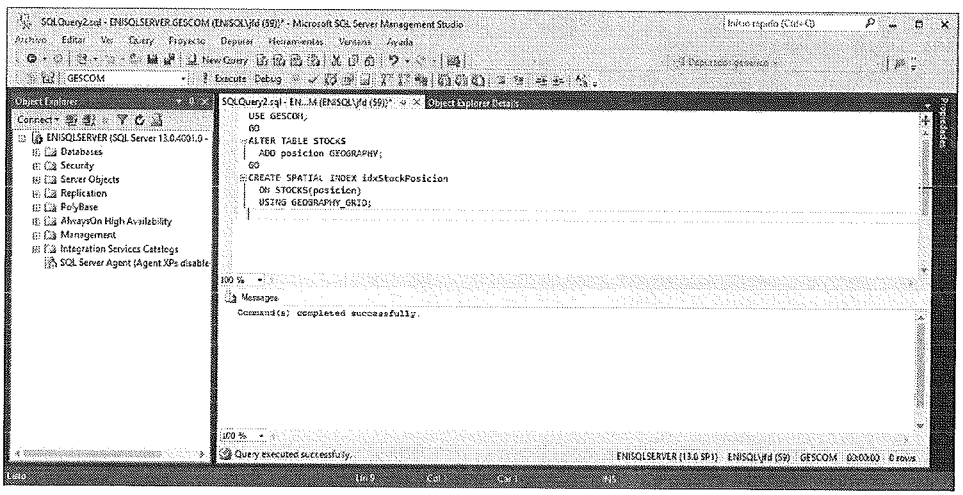
o

```
CREATE SPATIAL INDEX nombreÍndice
ON nombreTabla(nombreColumna)
USING {GEOMETRY_GRID|GEOGRAPHY_GRID}
WITH (
BOUNDING_BOX=(xMin, yMin, xMax, yMAX),
GRIDS(densidad)
);
```

La opción USING permite especificar al crear índice si los datos indexados son de tipo geométrico o geográfico.

Como se ha precisado anteriormente, la densidad puede tomar los valores LOW, MEDIUM o HIGH. Si el nivel de densidad no se especifica en el momento de crear el índice, entonces se define el índice por defecto con una densidad de tipo MEDIUM.

Ejemplo



Observación

Los índices de tipo espacial se pueden definir sobre un grupo de archivos diferente del que contiene la tabla indexada.

6. La partición de tablas y de índices

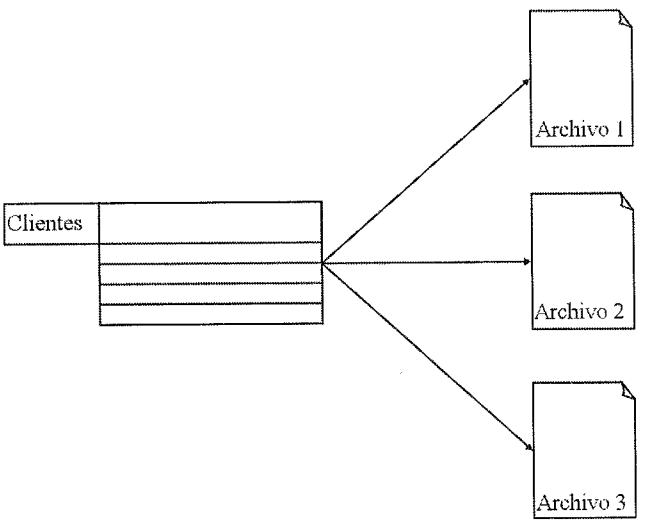
El objetivo de la partición es ofrecer un mejor rendimiento al trabajar con tablas muy voluminosas en términos de datos y a las que acceden muchos usuarios.

La partición puede ser útil para dividir el método de acceso a los datos. Por ejemplo, sobre la tabla de peticiones, solo es posible modificar (update) los pedidos del año contable en curso. Los pedidos de los años anteriores deben estar en modo de solo lectura.

La partición de una tabla permite dividir una tabla de grandes dimensiones en varias tablas. Cada una de estas subtablas es más pequeña que la tabla inicial y, por lo tanto, más fácil de gestionar para SQL Server. Es la unión de los datos presentes en todas estas subtablas lo que se corresponde con la tabla de origen.

Cada una de estas subtablas se puede crear sobre un grupo de archivos diferente, lo que hace posible la gestión de parámetros de almacenamiento distintos para cada partición de la tabla inicial y adaptar, de esta manera, las condiciones de almacenamiento de los datos en función de su utilización.

La tabla sobre la que se ha realizado una partición permite optimizar el almacenamiento de la información sin que el número de tareas administrativas adicionales sea elevado. Es más: los elementos tales como las restricciones de integridad y los triggers se definen en la tabla sin tener en cuenta el espacio de almacenamiento físico utilizado.



El reparto de los datos entre las diferentes particiones de la tabla se efectúa automáticamente en función de los criterios de reparto definidos en el momento de la creación de la tabla.

También es posible dejar que SQL Server reparta automáticamente los datos entre los diferentes grupos de archivos. Aunque esta solución es fácil de poner en práctica, no presenta demasiado interés, ya que los datos no se agrupan siguiendo criterios lógicos. Por lo tanto, las extracciones raramente se centran en un único grupo de archivos.

Así pues, para hacer una partición en la tabla, es necesario definir una función de partición. Esta función va a dar una clave de partición que va a utilizarse para seleccionar la partición que se usará, en función del plan de partición definido.

Si dos tablas utilizan la misma función de partición y el mismo esquema de partición, entonces los datos relacionados se almacenarán en el mismo grupo de archivos. En este caso, se dice que los datos están alineados.

Es posible crear índices sobre una tabla con particiones. El índice creado de esta manera también tiene particiones. Como en cualquier tabla, es posible definir un índice organizado (CLUSTERED). Si se escoge definir este tipo de índice, entonces la organización física de los datos debe estar en relación con los elementos pasados a la función de partición. Es decir, que las columnas pasadas como parámetro a la función de partición deben estar presentes en la definición del índice. De esta manera, la organización física de la tabla sigue el reparto de los datos entre los diferentes grupos de archivos.

6.1 La función de partición

Es la función de partición la que va a permitir dirigir los datos a un grupo de archivos o a otro. Para repartir los datos entre las diferentes particiones, la función utiliza rangos de valores. Cada rango está limitado por valores. En la función de partición solo se indican estos valores límite. En caso de que el valor límite deba estar incluido en el rango de valor inferior, es necesario utilizar la palabra clave LEFT. La palabra clave RIGHT permite incluir el valor límite en el espacio siguiente.

Observación

SQL ordena siempre los datos de manera ascendente.

Sintaxis

```
CREATE PARTITION FUNCTION nombrefunción ( parámetro)
AS RANGE [ LEFT | RIGHT ] FOR VALUES ( [ valorLímite [ ,... ] ] )
```

`parámetro`

Columna de cualquier tipo salvo timestamp, varchar(max), nvarchar(max) y varbinary utilizada para calcular la clave de partición.

`valorLímite`

Valor que marca el límite de cada partición.

Ejemplo

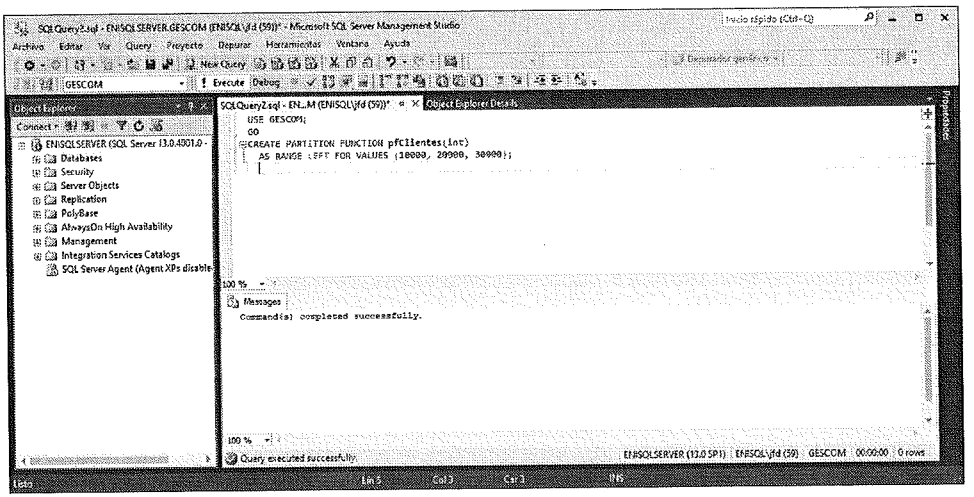
En el ejemplo siguiente, se define una función de partición sobre un valor de tipo entero. Esta función define cuatro particiones diferentes:

Primera partición para los valores inferiores o iguales a 10.000.

Segunda partición para los valores estrictamente superiores a 10.000 e inferiores o iguales a 20.000.

Tercera partición para los valores estrictamente superiores a 20.000 e inferiores o iguales a 30.000.

Cuarta partición para los valores estrictamente superiores a 30.000.



6.2 El esquema de partición

El esquema de partición permite establecer una tabla de correspondencia entre los valores devueltos por la función de partición y la utilización de una partición u otra. Cada partición está asignada a un grupo de archivos. Es posible, aunque no recomendable, utilizar el mismo grupo de archivos para todas las particiones.

También es posible precisar más grupos de archivos de los que necesita la partición. En este caso, el primer grupo de archivos adicional se marcará NEXT USED; es decir, como el siguiente grupo de archivos que se va a utilizar si la función de partición define una nueva partición.

Sintaxis

```
CREATE PARTITION SCHEME nombreEsquemaPartición
AS PARTITION nombreFunciónPartición
[ ALL ] TO ( { grupoDeArchivos | [ PRIMARY ] } [,_] )
[ ; ]
```

nombreEsquemaPartición

Identificador del esquema de partición.

nombreFunciónPartición

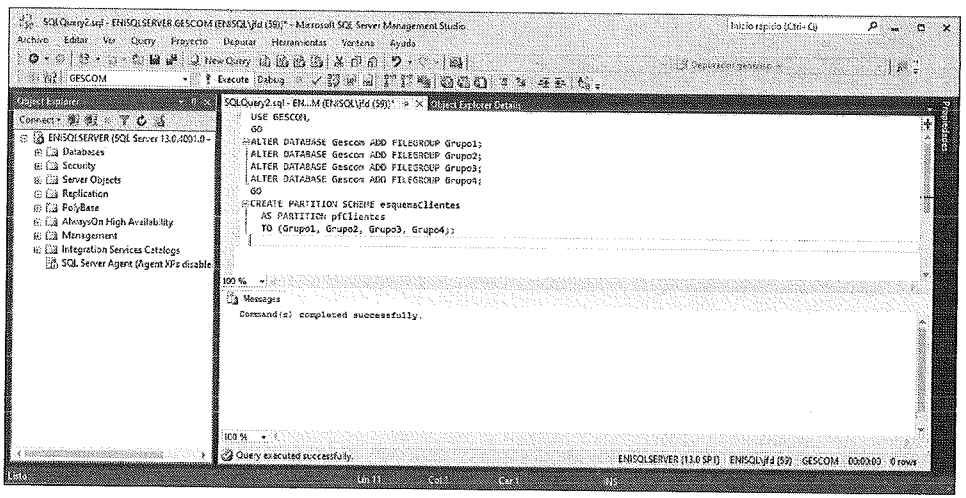
Nombre de la función de partición relativa al esquema. Un esquema solo puede relacionarse con una única función. Por el contrario, una misma función puede ser utilizada por varios esquemas.

grupoDeArchivos

Nombre del grupo o de los grupos de archivos utilizados por las diferentes particiones.

Ejemplo

En el ejemplo siguiente, se define un esquema de partición con relación a la función de partición definida en el ejemplo anterior.



6.3 La tabla con particiones

El reparto de los datos entre las diferentes particiones de la tabla será una operación transparente para los usuarios de esta. Por el contrario, en la creación de la tabla, es necesario indicar el esquema de partición que se va a usar, así como la columna que será utilizada por la función de partición, para decidir qué partición se asigna a cada línea de información. El tipo de esta columna debe coincidir con el tipo de parámetro de la función. En caso de que la columna utilizada para el cálculo de la partición sea una columna calculada, entonces deberá ser de tipo PERSISTED.

Sintaxis

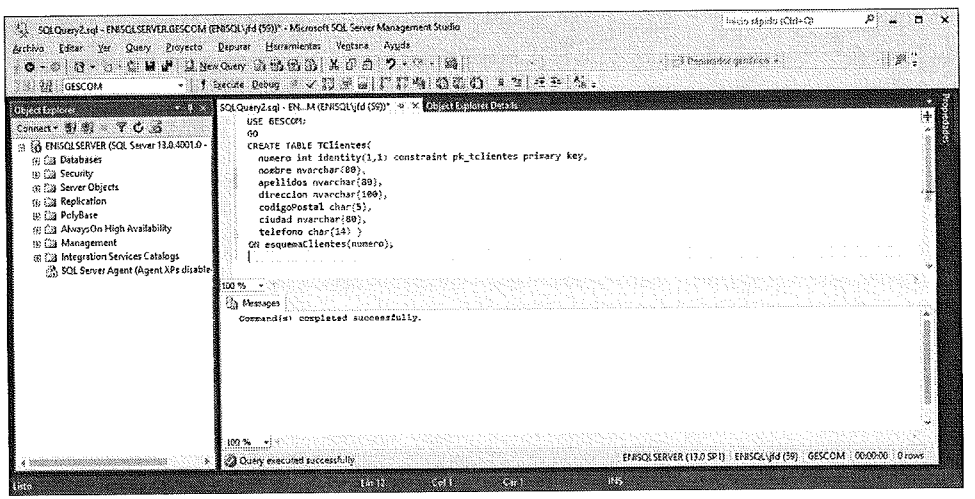
```
CREATE TABLE nombreTabla(
  definiciónColumna [,...]
) ON
nombreEsquemaPartición(columnaUtilizadaParaCalcularLaPartición)[;]
```

Observación

La información dada aquí para las tablas es igualmente válida para los índices.

Ejemplo

En el ejemplo siguiente, la tabla TClientes se define utilizando el esquema de partición establecido en el ejemplo anterior.



6.4 Los índices con particiones

Como para las tablas, es posible particionar los índices. El proceso de partición es el mismo, es decir: se basa en un esquema y una función de partición. Se recomienda encarecidamente que el índice que se quiere particionar esté definido sobre una tabla que ya tenga particiones y se base en los mismos esquemas y funciones de partición.

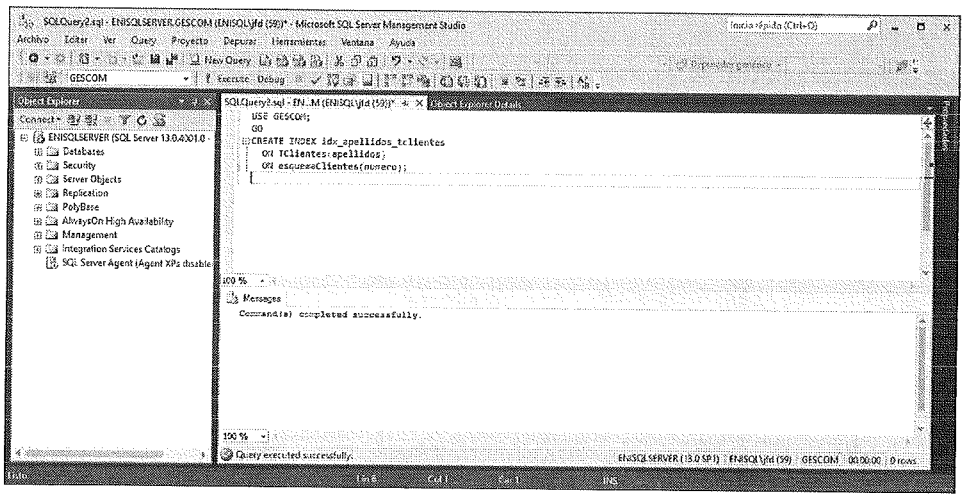
Si la columna de partición no forma parte de las columnas indexadas, entonces se incluye en el índice como una columna que permite definir índices de recubrimiento.

Sintaxis

```
CREATE INDEX nombreÍndice
ON nombreTabla(columna1,...)
ON nombreEsquemaPartición(columnaDePartición);
```

Ejemplo

Se define un índice sobre la columna nombre de la tabla particionada TClientes.



7. Compresión de datos

SQL Server 2012 ofrece la posibilidad de activar la compresión a nivel de tablas e índices. Si la compresión se puede definir sobre las tablas e índices existentes, no se tomará en cuenta hasta después de la reconstrucción de la tabla (ALTER tabla nombreTabla REBUILD) o del índice en cuestión. Si la compresión de la tabla implica la compresión del índice organizado (CLUSTERED), los índices no organizados no se ven afectados y es necesario habilitar la compresión sobre cada uno de ellos, uno a uno. En el caso de las tablas con particiones, la compresión puede tener lugar partición a partición.

Observación

La compresión solo es posible para los datos de usuario. Las tablas de sistema no se pueden comprimir.

El objetivo de la compresión es reducir el espacio de disco que utilizan los datos de la tabla. La compresión de los datos va a permitir almacenar más líneas de información en el mismo bloque de 8 KB. La compresión no permite aumentar el tamaño máximo de las líneas. De hecho, el mecanismo debe ser reversible.

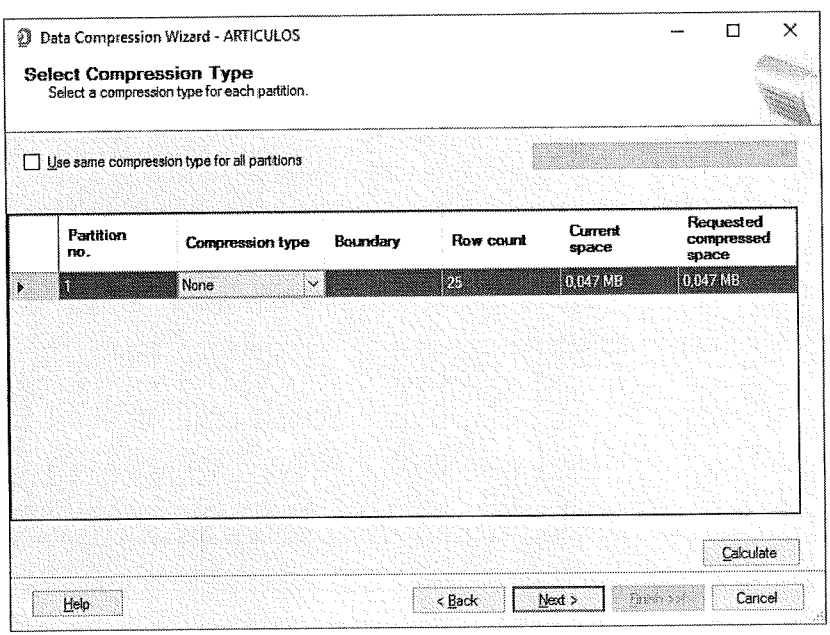
En las tablas valorizadas, es posible conocer el impacto de la compresión de los datos ejecutando el procedimiento almacenado **sp_estimate_data_compression_savings**.

La compresión es una operación puntual. Por eso es preferible pasar por el asistente que propone SQL Server Management Studio. A nivel de Transact SQL, la compresión se realiza con las instrucciones CREATE TABLE/CREATE INDEX y ALTER TABLE/ALTER INDEX.

El asistente de compresión de los datos se lanza desde el menú contextual asociado a la tabla, seleccionando la opción **Storage - Manage Compression**.

El asistente también le permite visualizar el resultado estimado en términos de ahorro de espacio de la compresión gracias al botón **Calculate**, que permite tener una estimación correcta en función del tipo de compresión elegido.

El número de registros presentes en la tabla debe ser consecuente para que la ganancia de compresión sea interesante. En función de los tipos de datos elegidos para las diferentes columnas de la tabla, la compresión puede resultar más o menos interesante.



8. Cifrado de datos

SQL Server permite encriptar los archivos de datos y los diarios de log. Este encriptado es dinámico y se efectúa en el momento de cada escritura en disco. Es igual para la operación de desencriptado de datos. Esta funcionalidad de encriptado/desencriptado es transparente y se llama TDE o *Transparent Data Encryption*.

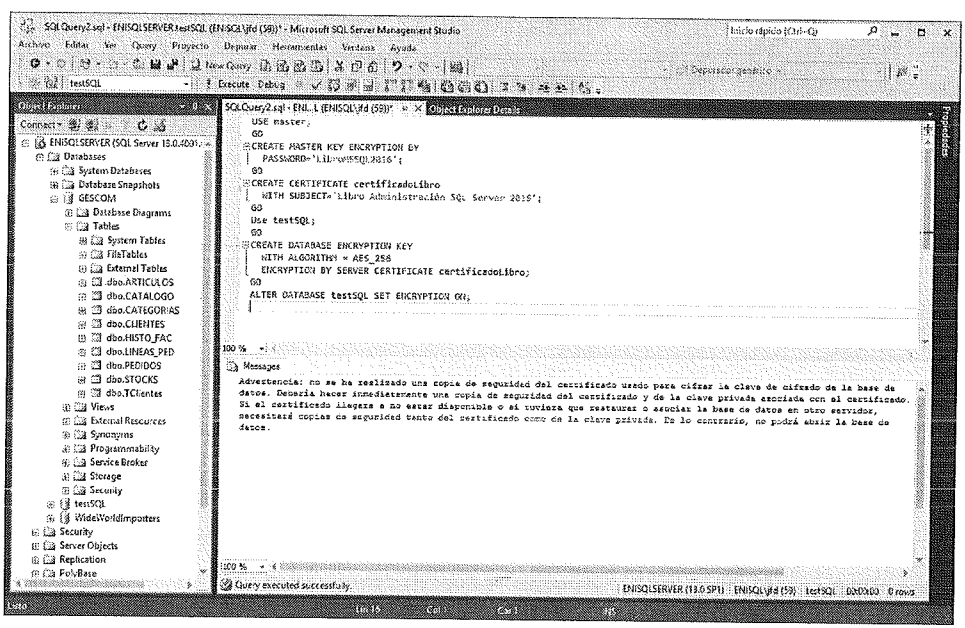
Establecer esta operación de encriptado permite garantizar una opacidad más grande de los archivos de datos y diarios para las diferentes herramientas de sistema de análisis de los archivos, o evitar una asociación/anulación de asociación de base de datos no autorizada. Sin embargo, esta operación de encriptado no tiene ninguna garantía adicional en lo que respecta a la comunicación entre el proceso cliente y el servidor. Cuando el encriptado de la base de datos está habilitado, las copias de seguridad también se encriptan con la misma clave. Por lo tanto, es necesario tener esta clave para poder restaurar los datos.

El encriptado de los datos se apoya en una clave de encriptado (DEK: *Database Encryption Key*) que se registra en la base master en forma de un certificado, por ejemplo.

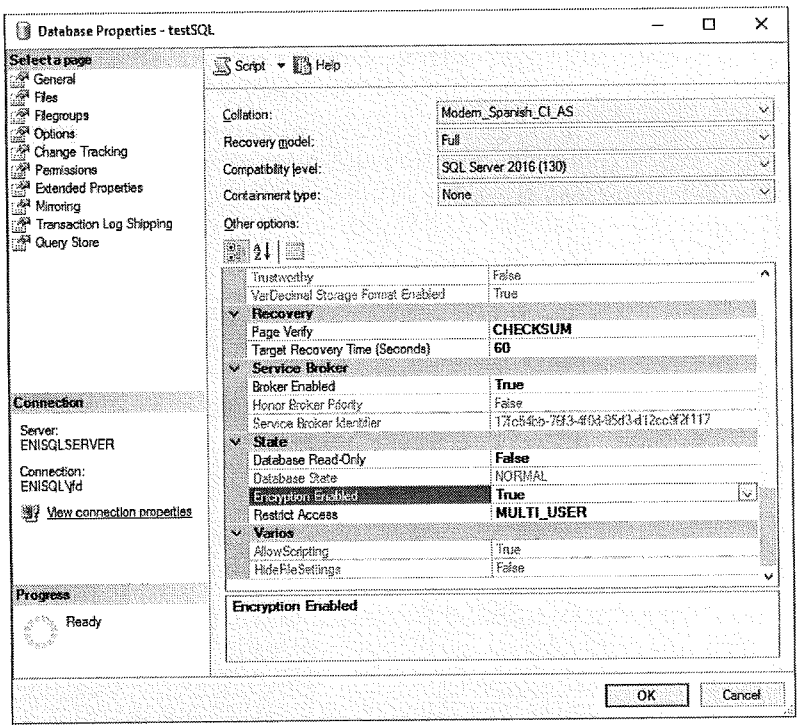
Antes de poder establecer las funciones de encriptado en una base de datos, es necesario comenzar por definir una clave principal para obtener un certificado válido. Entonces, el certificado se utiliza para definir la clave de encriptado. La generación de esta clave se puede realizar con diferentes algoritmos. El nombre del algoritmo elegido se pasa como argumento a la instrucción CREATE DATABASE ENCRYPTION KEY.

Ejemplo

En el siguiente ejemplo, el encriptado se define sobre la base TestSQL.



Con SQL Server Management Studio, desde la página **Options** de las propiedades de la base de datos, es posible habilitar o deshabilitar el soporte del proceso de encriptado a nivel de la base de datos.



La activación del encriptado sobre una base de datos afecta a todos los archivos de datos. Si un grupo de archivos está en modo de solo lectura, esta activación falla.

9. Las tablas temporales

SQL Server propone hacer el seguimiento automático de las diferentes versiones de la información almacenada en una tabla. Es decir que SQL Server guardará una versión de cada fila antes y después de su modificación.

El nombre exacto de este mecanismo porpuesto por SQL Server es el de "Tablas de sistema temporales por versión" ya que SQL Server gestiona el periodo de validez de cada fila.

Para gestionar esta noción de temporalidad, SQL Server se apoya en:

- Dos columnas de tipo datetime2. Estas dos columnas corresponden a las fechas y horas de inicio y final de validez.
- Una tabla espejo (en términos de estructura) a la tabla que contiene los datos activos para conservar el histórico. A esta tabla se la conoce a menudo con el nombre de "tabla de histórico". Puede ser creada automáticamente por SQL Server o por el administrador.

Estas tablas, aunque sean singulares en su sistema de gestión de la información, se pueden administrar como cualquier otra tabla de la base de datos.

Sintaxis

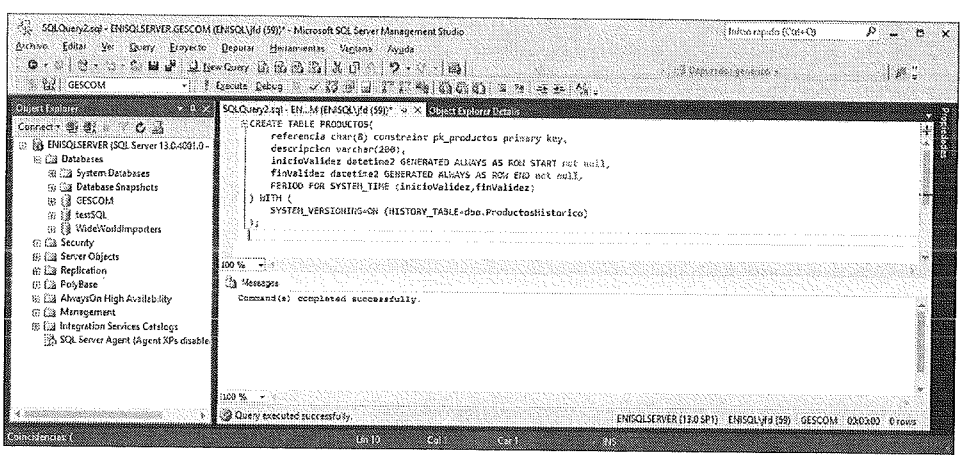
```
CREATE TABLE nombreTabla(
nombreColumnatipo, ...
inicioValidez datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
finValidez datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
PERIOD FOR SYSTEM_TIME (inicioValidez, finValidez)
) WITH (SYSTEM_VERSIONING=ON (HISTORY_TABLE= nombreTablaHistorico)) ;
```

Observación

Esta tabla necesita una clave primaria.

Ejemplo

En el siguiente ejemplo, se genera una tabla de productos con una noción de temporalidad.



10. Planificación

10.1 Dimensionar los archivos

Si se desea evaluar el tamaño de los archivos necesarios para almacenar la información contenida en la base de datos, es necesario tener en cuenta numerosos criterios.

Para los archivos de datos

- Distinguir las tablas de sistema y de usuario.
- Tener en cuenta el número de líneas de las tablas.
- Identificar los valores indexados (clave, número de línea, factor de llenado).

Solo después de una detallada evaluación de la cantidad del espacio ocupado, es posible fijar el tamaño inicial de los archivos de datos. El método más simple consiste en evaluar la longitud media de una línea, calcular cuántas líneas pueden almacenarse en un bloque de 8 KB y por último encontrar el número de bloques necesarios para almacenar todas las líneas de la tabla. A partir de este número de bloques utilizados por la tabla, conviene tomar el múltiplo de 8 inmediatamente superior y a continuación dividir por 8 para obtener el número de extensiones.

Para los archivos diarios

- La actividad.
- La frecuencia.
- El tamaño de las transacciones.
- Las copias de seguridad.

Si tenemos en cuenta estos criterios y consultamos las opciones de la base de datos, podremos fijar un tamaño óptimo para el archivo diario. Inicialmente, puede ser útil fijar el tamaño del diario entre el 10 % y el 25 % del tamaño de los datos en la base de datos. Este porcentaje disminuye si la base soporta principalmente consultas de selección SELECT, que no utilizan el diario.

10.2 Nombrar la base de datos y los archivos de manera explícita

Es importante pensar el nombre de la base, los nombres lógicos, físicos y el tamaño de los archivos, así como la gestión dinámica o no del crecimiento.

10.3 Ubicación de los archivos

La ubicación de los archivos de datos y diarios debe especificarse con precisión, con el objetivo de reducir al máximo los conflictos de acceso al disco y de hacer trabajar a todos los discos de la máquina de manera equitativa.

10.4 Uso de los grupos de archivos

Para limitar los conflictos de acceso a disco, puede ser interesante crear varios grupos de archivos en el servidor y especializar cada uno de ellos. Un buen ejemplo consiste en dejar sobre el grupo de archivos PRIMARY el conjunto de tablas de sistema de la base y crear un segundo grupo para las tablas de usuario. Un método como este permite separar los datos y las tablas de sistema. Algunas veces es posible crear un tercer grupo para los índices.

10.5 Nivel de compatibilidad

Un servidor puede albergar diferentes bases de datos de usuario con diferentes niveles de compatibilidad. Una base creada con SQL Server 2016 tiene un nivel de compatibilidad de 130. Una instancia SQL Server 2016 puede administrar bases de nivel 100 para SQL Server 2008, 110 para SQL Server 2012, 120 para SQL Server 2014 y 130 para SQL Server 2016.

Todas las instrucciones Transact SQL se interpretan dentro de una base de datos en relación con el nivel de compatibilidad fijado.

Es posible modificar el nivel de compatibilidad de una base con la instrucción `ALTER DATABASE` y la opción `SET COMPATIBILITY LEVEL`.

Syntaxis

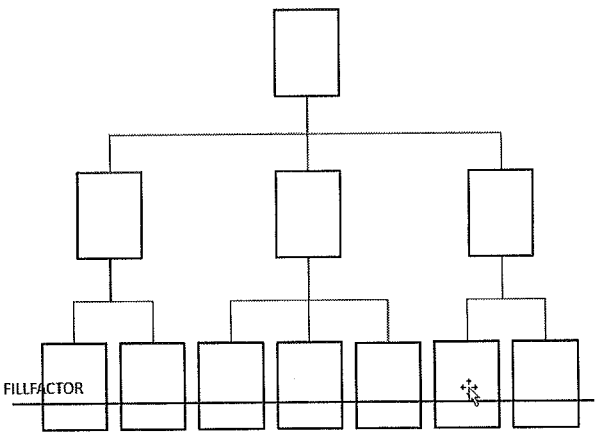
```
ALTER DATABASE nombreBaseDeDatos SET COMPATIBILITY LEVEL= { 100 | 110 |
120 | 130 }
```

10.6 Establecer el parámetro FillFactor

Durante la construcción o reconstrucción de los índices, está presente el parámetro FillFactor. Este parámetro permite definir la tasa de ocupación de las páginas durante la aplicación del índice. El parámetro FillFactor (también conocido como "tasa de ocupación" en SQL Server Management Studio) solo afecta a las páginas que se encuentran en un nivel hoja. Las demás páginas constituyentes del índice no están afectadas por esta tasa de ocupación. Esta tasa de ocupación se expresa como un porcentaje. Por ejemplo, si se asigna el valor 70 al parámetro FillFactor, significa que la página se llenará al 70 % y el 30 % quedará libre.

Este espacio libre se usará para gestionar la adición de datos en el índice sin que esto haga que se desequilibre la estructura del índice. De esta manera, el índice va a conservar una estructura equilibrada más tiempo. El valor del parámetro FillFactor se debe elegir en función de la tasa de actualización.

El siguiente esquema ilustra la estructura de un índice y la configuración de FillFactor a nivel de hoja.



Un valor alto de FillFactor significa que las páginas de nivel hoja van a contener mucha información ; por lo que el índice va a ser más compacto, ya que serán necesarias menos páginas. Además, el uso del índice será más rápido. Por el contrario, la adición de información en la tabla indexada va a desequilibrar con rapidez el índice (más concretamente la estructura de B-tree del índice), por lo que será necesario reconstruirlo.

Un valor pequeño de FillFactor va a provocar un aumento en el número de páginas que componen el índice y su uso tendrá un rendimiento más bajo porque será necesario manipular más páginas. Por el contrario, este tipo de índices será resistente a las inserciones/modificaciones de datos, por lo que el riesgo de desequilibrar el índice es más bajo.

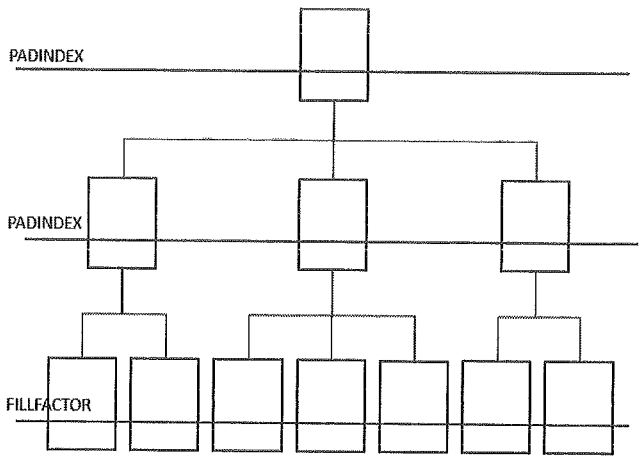
Por último, para hacer una configuración más profunda de los índices, la tasa de ocupación que se define con FillFactor se puede aplicar a las páginas de nivel hoja, usando el parámetro PADINDEX.

Este parámetro FillFactor se puede usar durante la reconstrucción del índice usando la instrucción SQL ALTER INDEX.

Observación

La instrucción DBCC DBREINDEX se mantiene por razones de compatibilidad pero no se debe utilizar ya que se eliminará en próximas versiones de SQL Server.

El siguiente esquema ilustra la estructura de un índice y la configuración de FillFactor a nivel de hoja y de Padindex en las demás páginas de índice.

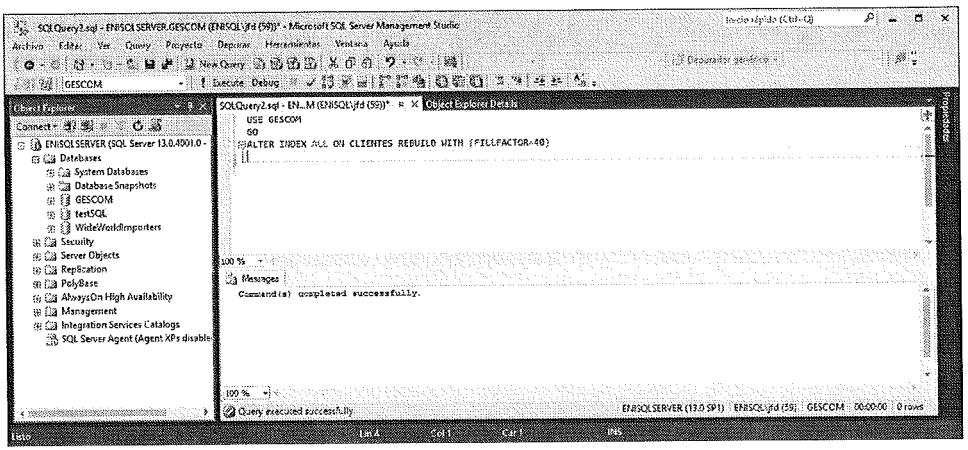


Sintaxis

```
ALTER INDEX {nombre_índice | ALL} ON nombre_tabla
REBUILD WITH ([PAD_INDEX=ON,] FILLFACTOR=valor);
```

Ejemplo

El siguiente ejemplo ilustra la reconstrucción de todos los índices de la tabla de clientes, asignando el valor 40 % al parámetro FillFactor.



11. Ejercicio: crear una base de datos

11.1 Enunciado

La primera etapa consiste por lo tanto en crear la base de datos LibroTSQL y, como su nombre indica, se creará con la ayuda de un script Transact SQL. Crearemos una segunda base de datos, llamada LibroSSMS, desde la interfaz gráfica del Management Studio.

Los archivos de datos se definirán en un directorio específico. Conviene crear el directorio c:\datos. Evidentemente, guardar los datos en el disco c:\ es producto del ejemplo pedagógico.

Parámetros de la base de datos LibroTSQL

Archivo de datos

- Tamaño: 10 MB
- Nombre físico: c:\datos\LibroTSQL.mdf
- Nombre lógico: LibroTSQL

Archivo de log
- Tamaño: 8 MB
- Nombre físico: c:\datos\LibroTSQL_log.ldf
- Nombre lógico: LibroTSQL_log

Parámetros de la base de datos LibroSSMS

Archivo de datos
- Tamaño: 15 MB
- Nombre físico: c:\datos\LibroSSMS.mdf
- Nombre lógico: LibroSSMS

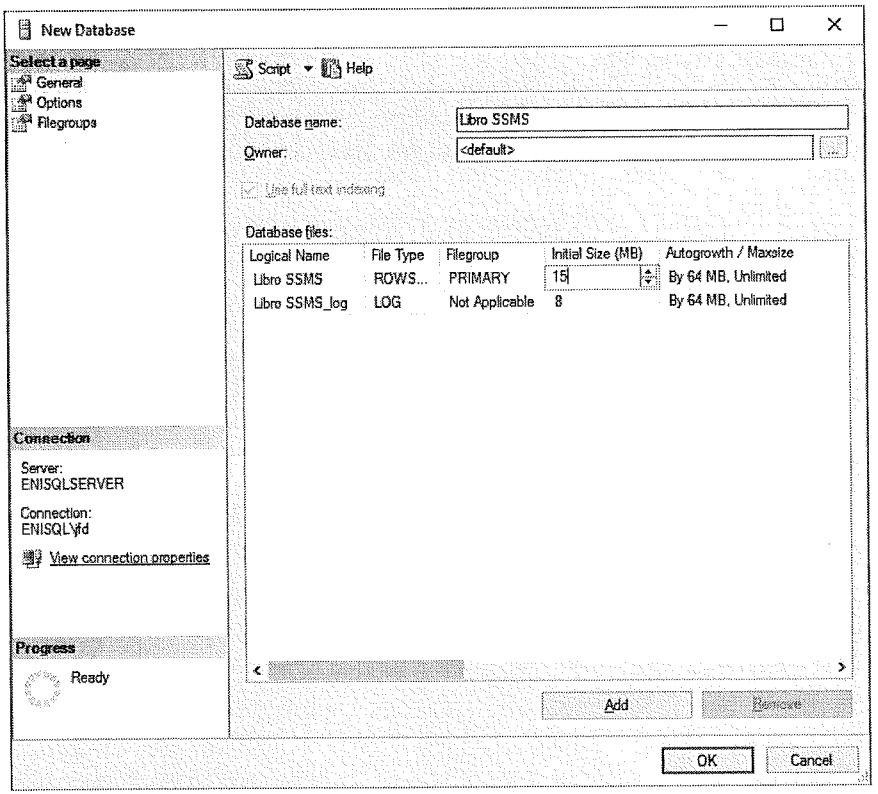
Archivo de log
- Tamaño: 8 MB
- Nombre físico: c:\datos\LibroSSMS_log.ldf
- Nombre lógico: LibroSSMS_log

11.2 Solución

Creamos la base de datos LibroTSQL con ayuda del siguiente script:

```
CREATE DATABASE LibroTSQL
    ON PRIMARY (
           NAME=LibroTSQL,
           FILENAME='c:\datos\LibroTSQL.mdf',
           SIZE=10Mb
    )
    LOG ON (
           NAME=LibroTSQL_log,
           FILENAME='c:\datos\LibroTSQL_log.ldf',
           SIZE=8Mb
);
```

Para crear la base de datos LibroSSMS debemos, desde el explorador de objetos de SQL Server Management Studio, acceder al menú contextual asociado a la rama de NuestroServidor - Base de datos y seleccionar la opción **New Database**. Aparece la siguiente pantalla. Debe tener cuidado al indicar el directorio de creación de los archivos así como el nombre físico de dichos archivos.



12. Ejercicio: añadir un grupo de archivos

12.1 Enunciado

En la base de datos LibroTSQL, añadimos el grupo de archivos Data. Este grupo de archivos se compone de dos archivos: data1.ndf y data2.ndf.

El archivo data1.ndf posee un tamaño fijo de 50 MB mientras que el archivo data2.ndf tiene un tamaño inicial de 10 MB que puede después crecer hasta un tamaño de 50 MB en pasos de 10 MB.

12.2 Solución

Para añadir un grupo de archivos y definir los archivos pertenecientes a este grupo, puede hacerlo a través del cuadro de diálogo que presenta las propiedades de la base de datos desde SSMS, o también a través de un script Transact SQL. A continuación presentamos este script.

En un primer paso, el grupo de archivos se define con la instrucción ALTER DATABASE.

```
ALTER DATABASE LibroTSQL
     ADD FILEGROUP Data;
```

El grupo de archivos está definido pero todavía no es posible utilizarlo ya que ningún archivo pertenece a este grupo. El primer archivo (data1) se crea con la ayuda del siguiente script.

```
ALTER DATABASE LibroTSQL
     ADD FILE (
           NAME=data1,
           FILENAME='c:\datos\data1.ndf',
           SIZE=50Mb,
           MAXSIZE=50Mb
     )
     TO FILEGROUP Data;
```

El segundo archivo (data2) se crea de una forma parecida pero añadiendo los parámetros MAXSIZE y FILGROWTH para controlar el tamaño máximo del archivo y su paso de crecimiento.

```
ALTER DATABASE LibroTSQL
     ADD FILE (
           NAME=data2,
           FILENAME='c:\datos\data2.ndf',
           SIZE=10Mb,
           MAXSIZE=50Mb,
           FILEGROWTH=10Mb
     )
     TO FILEGROUP Data;
```

Capítulo 4
Gestión de la seguridad del acceso

1. Introducción

El control de acceso representa una operación importante en la gestión de la seguridad sobre un servidor de bases de datos. La seguridad de los datos requiere una organización de los objetos de manera independiente de los usuarios, y esto es posible gracias a los esquemas. La seguridad pasa también por un mejor control de las autorizaciones y la posibilidad de asignar los privilegios necesarios a cada usuario para que puedan trabajar de manera autónoma.

Para la organización de esta política de seguridad, es necesario tener en cuenta la organización jerárquica de los elementos de seguridad, de manera que la gestión de los derechos de acceso sea simple y eficaz.

SQL Server se apoya sobre tres elementos claves, que son:

- Las entidades de seguridad.
- Los objetos asegurables.
- Las autorizaciones.

Las entidades de seguridad son las cuentas de seguridad que disponen de un acceso al servidor SQL.

Los objetos asegurables representan los objetos gestionados por el servidor. Aquí, un objeto puede ser una tabla, un esquema o una base de datos, por ejemplo.

Las autorizaciones se conceden a las entidades de seguridad para que puedan trabajar con los objetos asegurables.

La organización jerárquica permite asignar una autorización (por ejemplo, SELECT) a un objeto asegurable de nivel elevado (por ejemplo, el esquema) para permitir a la entidad de seguridad que recibe la autorización ejecutar la instrucción SELECT sobre todas las tablas contenidas en el esquema.

Las vistas del catálogo de sistema permiten obtener un informe completo y detallado sobre las conexiones existentes, los usuarios de base de datos definidos y los privilegios asignados. Algunas de estas vistas se presentan a continuación:

- sys.server_permissions: lista de permisos de nivel servidor y sus beneficiarios.
- sys.sql_logins: lista de las conexiones.
- sys.server_principals: entidad de seguridad definida en el servidor.
- sys.server_role_members: lista de los beneficiarios de un rol de servidor.
- sys.database_permissions: lista de permisos y sus beneficiarios en la base de datos.
- sys.database_principals: entidad de seguridad a nivel de la base de datos.
- sys.database_role_members: lista de los beneficiarios de un rol de base de datos.

Para simplificar la gestión de los derechos de acceso, es posible utilizar tres tipos de roles. Los **roles de servidor** agrupan las autorizaciones a nivel del servidor. Estas autorizaciones son válidas para todas las bases de datos instaladas. Los **roles de base de datos** agrupan los derechos a nivel de la base de datos sobre la que se definen. Por último, los **roles de aplicaciones**, definidos sobre las bases de datos de usuario, permiten agrupar los derechos necesarios para la correcta ejecución de una aplicación cliente.

2. Gestión de los accesos al servidor

Antes de poder trabajar con los datos gestionados por las bases de datos, es necesario en primer lugar conectarse al servidor SQL. Esta etapa permite hacerse identificar por el servidor SQL y utilizar posteriormente todos los derechos que se asignan a la conexión. En SQL existen dos modos de gestión de los accesos al servidor de base de datos.

Atención: en esta sección únicamente se aborda la parte de conexión al servidor. Es importante distinguir bien la conexión al servidor de la utilización de bases de datos. La conexión al servidor permite hacerse identificar por el servidor SQL como un usuario válido para utilizar, posteriormente, una base de datos: los datos y los objetos. El conjunto de estos derechos se definirá más adelante. Estos derechos se asocian a un usuario de base de datos al que corresponde una conexión.

Observación

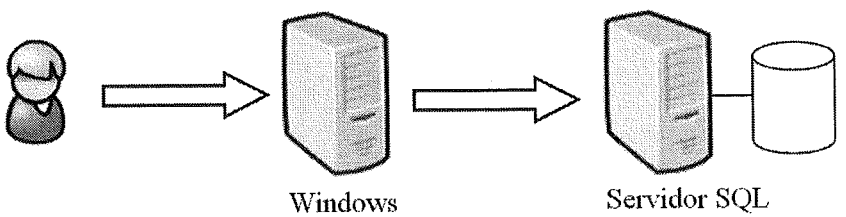
Hablaremos de conexión al servidor o de login.

2.1 Modo de seguridad de Windows

Este tipo de gestión de seguridad permite apoyarse sobre los usuarios y los grupos de Windows para el dominio y el puesto local. SQL Server utiliza la gestión de los usuarios de Windows (gestión de las contraseñas...) y recupera únicamente los nombres para crear conexiones al servidor.

Una funcionalidad muy importante de SQL Server es poder autorizar a los grupos de Windows a conectarse. La gestión de los grupos permite realizar una administración más flexible que la de los usuarios. El método más simple para gestionar las conexiones es pasar por la creación de un grupo local. Este grupo local está autorizado a conectarse al servidor SQL y es utilizado por los usuarios o los grupos globales.

Con este método de funcionamiento, como un usuario de Windows puede pertenecer a varios grupos, puede tener varios derechos de conexión a SQL Server.



Autenticación de Windows

En modo de seguridad de Windows, solo se almacenan los nombres de usuario. La gestión de las contraseñas y la pertenencia a diferentes grupos se deja a Windows. Este modo de funcionamiento permite separar las tareas que son responsabilidad de cada uno y especializar a SQL Server en la gestión de los datos, dejando a Windows la gestión de los usuarios, que sabe hacerla bien. Además, con este esquema de funcionamiento, es posible aplicar la política siguiente: un usuario = una contraseña. El acceso al servidor SQL es transparente para los usuarios aprobados por Windows.

Observación

SQL Server se apoya sobre los grupos a los que pertenece el usuario en el momento de la conexión al servidor. Si se efectúan modificaciones de pertenencia a los grupos desde Windows, estas modificaciones no se tomarán en cuenta hasta la próxima conexión del usuario al servidor SQL.

Observación

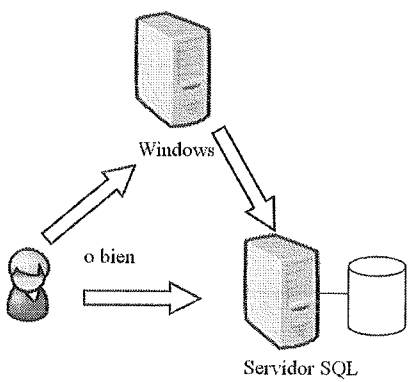
SQL Server se basa en el SID de Windows para identificar el grupo. Si un grupo se elimina y posteriormente se vuelve a crear en Windows, es importante realizar la misma operación en lo que se refiere a la conexión del grupo en SQL Server.

2.2 Modo de seguridad mixta

El modo de seguridad mixta se basa en una autenticación de Windows y posteriormente en una autenticación de SQL Server. Este modo de autenticación es el que se va a detallar aquí.

2.2.1 Definición

Se trata del funcionamiento más conocido para la seguridad de los servidores de bases de datos. En este modo de funcionamiento, es SQL Server quien se encarga de verificar que el usuario que solicita conectarse está bien definido y posteriormente se encarga también de verificar la contraseña.



Modo de seguridad mixta

Todos los usuarios son completamente gestionados por SQL Server (nombre y contraseña). Este tipo de gestión de las conexiones se adapta bien a los clientes que no se identifican desde Windows.

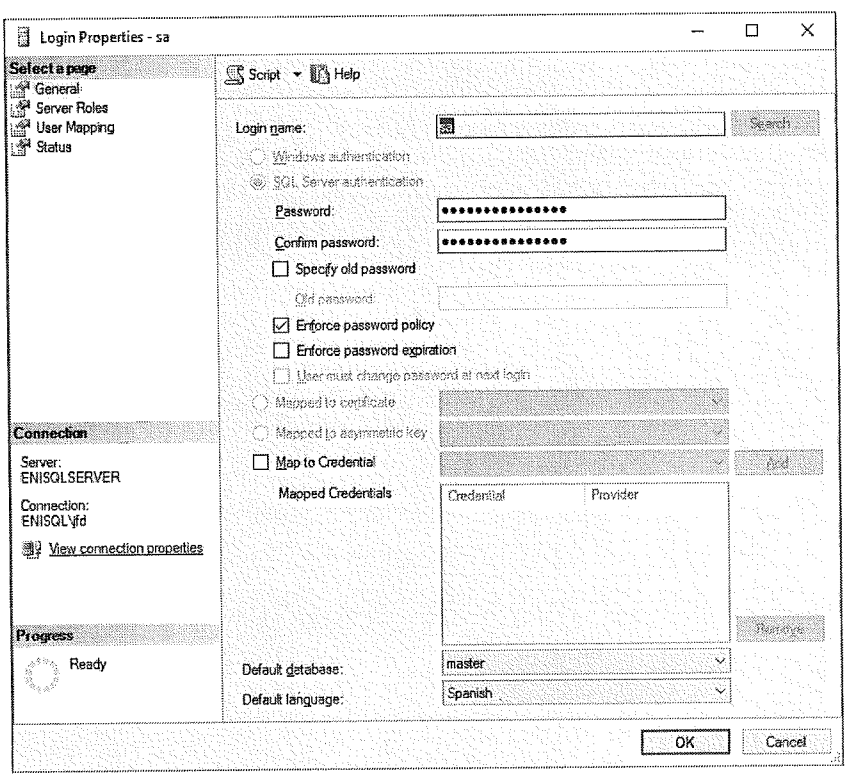
2.2.2 Principio de funcionamiento

No es correcto pensar que en modo de funcionamiento mixto solo es el servidor el que se encarga de la verificación de los nombres de usuario y de las contraseñas. En un primer momento, cuando un usuario de red intenta conectarse al servidor SQL, se realiza una prueba utilizando la seguridad de Windows.

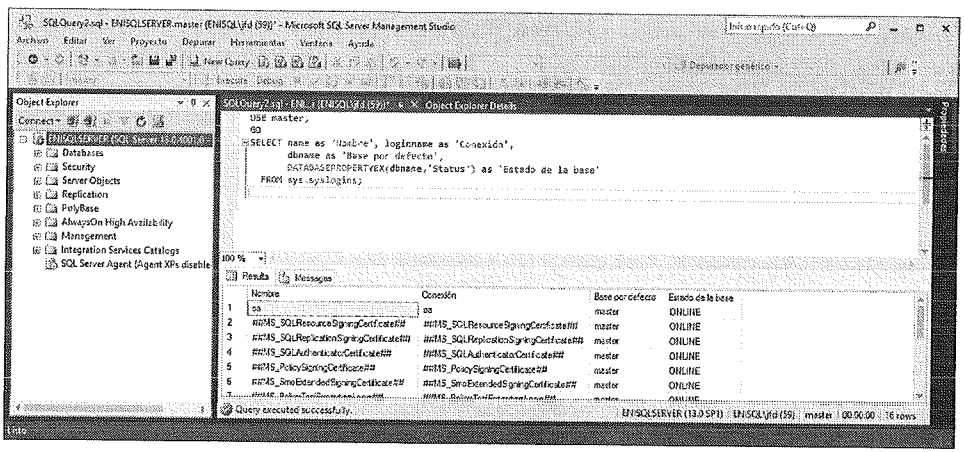
Si el usuario de red es aprobado en SQL Server o si pertenece a un grupo que es aprobado por SQL Server, entonces el usuario podrá conectarse al servidor SQL. En caso contrario (error de conexión utilizando la seguridad de Windows), SQL Server se encarga de solicitar un nombre de conexión y una contraseña.

2.3 Base de datos predeterminada

Después de la definición de las conexiones (logins) al servidor, es importante definir en las diferentes bases de datos de usuario los usuarios que corresponden a estas conexiones. Los permisos de uso de la base de datos se asignan a los usuarios de dicha base. También es importante definir, para cada conexión o login, una base de datos predeterminada. Sobre esta base de datos se situará cualquier usuario que utilice esta conexión. Atención, es necesario tener en cuenta que definir una base de datos predeterminada no otorga ningún permiso para usar esta base de datos.



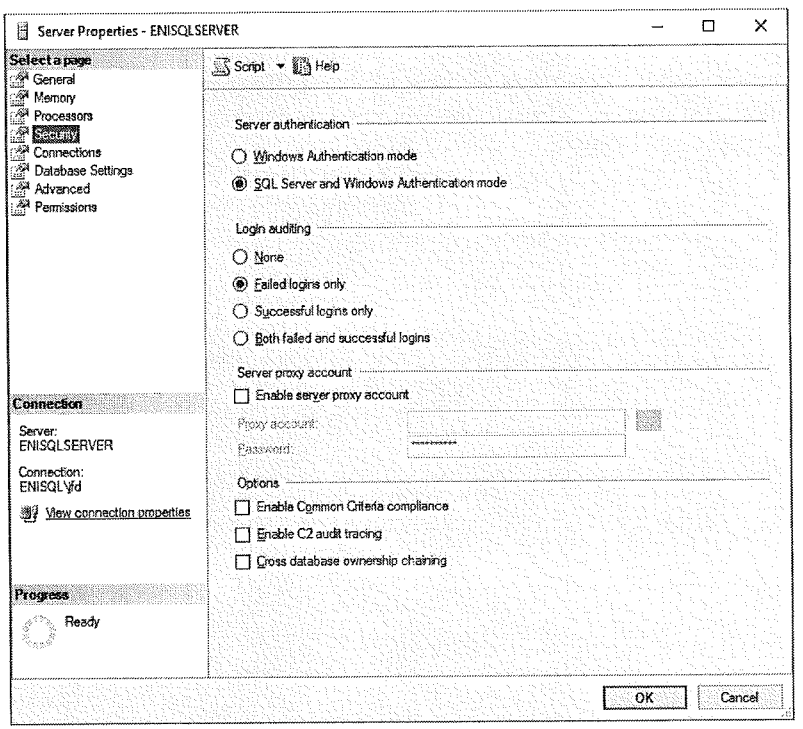
Es posible conocer la base de datos predeterminada para cada conexión consultando la vista sys.syslogins de la siguiente manera:



En caso de que la base de datos predeterminada asociada a la conexión ya no sea válida, o si deseamos modificar esta base de datos predeterminada, es necesario volver a las propiedades de la conexión desde SQL Server Management Studio o usar la instrucción ALTER LOGIN (instrucción que se presenta más adelante en este capítulo).

2.4 ¿Cómo elegir un modo de seguridad?

Es posible modificar el modo de seguridad utilizado por el servidor SQL directamente desde SQL Server Management Studio modificando las propiedades de la instancia, como se muestra a continuación.



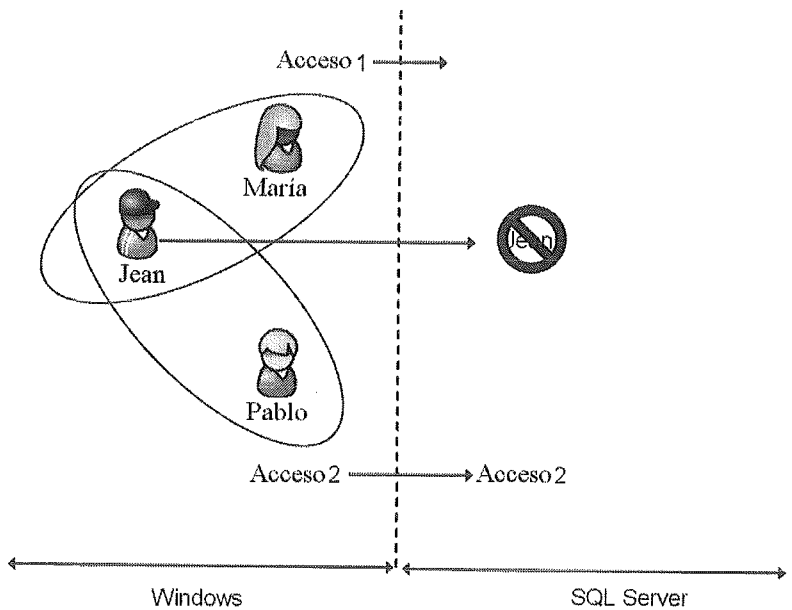
En el momento de instalar el servidor SQL, se predefinen dos conexiones: una en modo Windows, en la que se autoriza al grupo local de los Administradores a conectarse, y otra en modo de seguridad SQL Server, en la que el usuario **sa** puede conectarse al servidor. Estos dos usuarios tienen privilegios de administrador del servidor SQL.

Observación

Se recomienda utilizar la seguridad de Windows, que ofrece una mayor flexibilidad para la gestión de los usuarios.

2.5 Administrar una conexión en SQL Server

Teniendo en cuenta que un usuario de Windows puede pertenecer a varios grupos, se le puede asignar varias veces el permiso de conexión al servidor SQL. Esto puede presentar un problema cuando un usuario o un grupo de usuarios no deben tener nunca la capacidad de conectarse al servidor SQL. Para remediar esta situación, Microsoft proporciona, dentro del conjunto de órdenes Transact SQL para la gestión de las conexiones, la orden DENY, que permite denegar explícitamente la conexión a un usuario o un grupo de Windows. DENY constituye un rechazo explícito y es prioritario frente a los permisos de conexión.



Creación de una conexión

En el esquema anterior, hay tres usuarios de Windows y dos grupos. Se ha establecido una conexión para el grupo Acceso 2, el grupo Acceso 1 no tiene conexión y el usuario Jean tiene prohibida la conexión.

Observación

*Es necesario tener un permiso de administrador (**sysadm**) o un permiso de gestor de seguridad (**securityadmin**) para poder realizar las diferentes operaciones relativas a la gestión de las conexiones.*

Las conexiones que sean de tipo SQL Server o bien Windows se deben definir en la instancia de SQL Server para permitir a los usuarios la conexión.

La gestión de las conexiones se puede realizar de manera gráfica con el explorador de objetos desde SQL Server Management Studio o con scripts Transact SQL. Todas las operaciones de gestión de las conexiones se realizan en Transact SQL con las instrucciones CREATE LOGIN, ALTER LOGIN y DROP LOGIN. Cada solución tiene sus ventajas y sus inconvenientes.

Observación

Los procedimientos sp_addlogin y sp_grantlogin no se deben utilizar más. Todavía están presentes en SQL Server para asegurar la compatibilidad de los scripts.

Las cuentas de conexión son necesarias para permitir el acceso al servidor de base de datos. Sin embargo, son insuficientes para permitir trabajar sobre una base de datos. Es necesario asociarles una base de datos predeterminada, es decir, la base de datos de trabajo predeterminada en la que se situarán después del inicio de la conexión. Para acceder a esta base de datos, se debe definir una cuenta de usuario de base de datos para la conexión sobre la base de datos.

2.5.1 En modo de seguridad de Windows

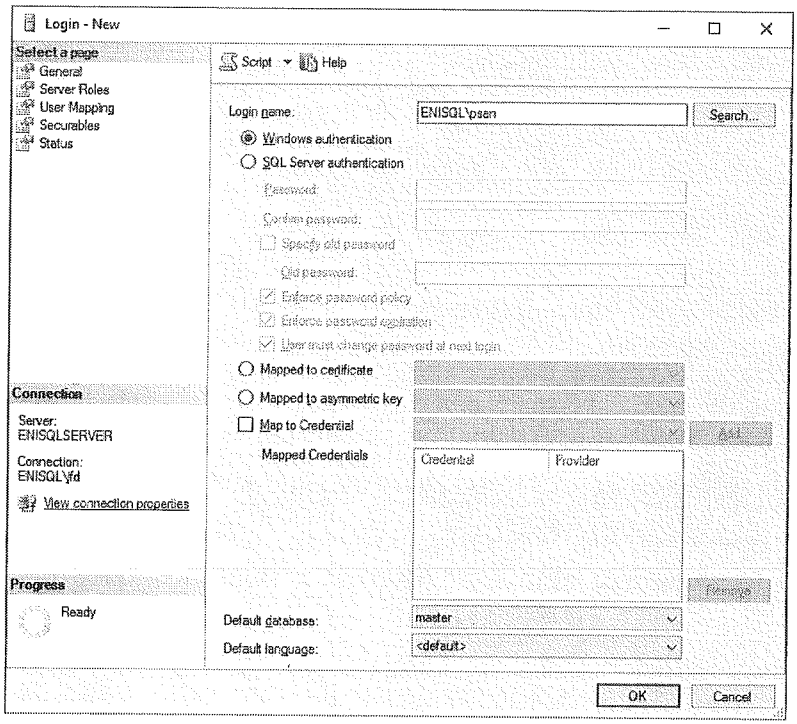
Los nombres de los grupos o de los usuarios deben corresponder a los que se definen en Windows.

SQL Server Management Studio

Es posible crear conexiones desde la interfaz gráfica de SQL Server Management Studio, procediendo de la siguiente manera:

- Desde el explorador de objetos, situarse sobre el nodo **Security - Logins**.
- Desde el menú contextual asociado al nodo conexión, seleccionar **New Login**.

Se muestra la pantalla siguiente. Solo falta seleccionar la cuenta de usuario de Windows o el grupo al que se autoriza a conectarse.



La interfaz gráfica de SQL Server Management Studio permite también modificar simplemente una cuenta de conexión a partir de la ventana de las propiedades o eliminar una conexión.

Transact SQL

```
CREATE LOGIN nombre_conexión
FROM WINDOWS
[WITH DEFAULT_DATABASE=baseDeDatos|DEFAULT_LANGUAGE=idioma]
```

`nombre_conexión`

Nombre del usuario o del grupo de Windows para añadir. Se debe indicar en forma de dominio\usuario.

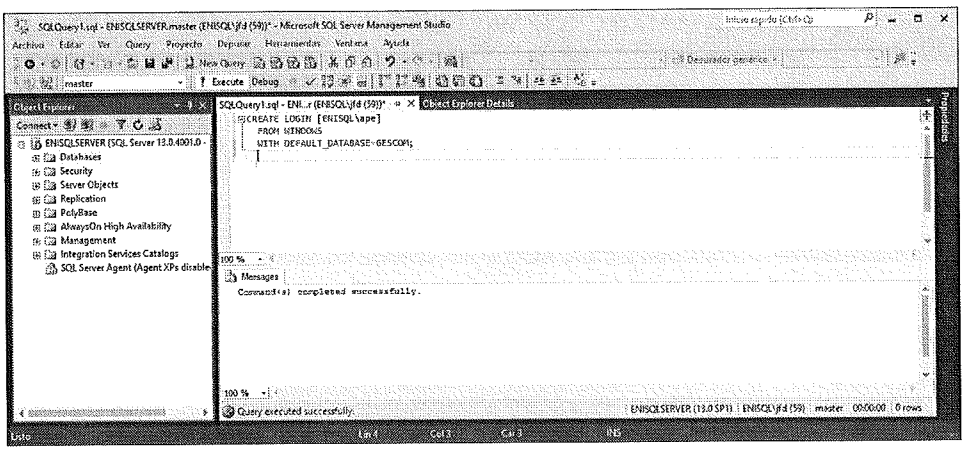
`baseDeDatos`

Indica la base de datos que se utilizará por defecto. Si no se indica ninguna, la base de datos por defecto es la base Master.

`idioma`

Idioma de trabajo por defecto para esta conexión. Solo es necesario indicar esta opción si el idioma relativo a esta conexión es distinto al definido por defecto sobre el servidor SQL.

Ejemplo



2.5.2 En modo de seguridad mixto

Como para la seguridad en modo Windows, existen dos medios para gestionar las conexiones.

Además del conjunto de reglas de gestión de las contraseñas, la gestión de una conexión en modo de seguridad de SQL Server es en todo momento similar a la de una conexión en modo de seguridad de Windows.

Una vez creados, los dos tipos de conexión son idénticos. No hay una solución que presente más funcionalidades que otra.

Para las conexiones que utilicen una seguridad de SQL Server, es posible definir reglas de gestión de contraseñas.

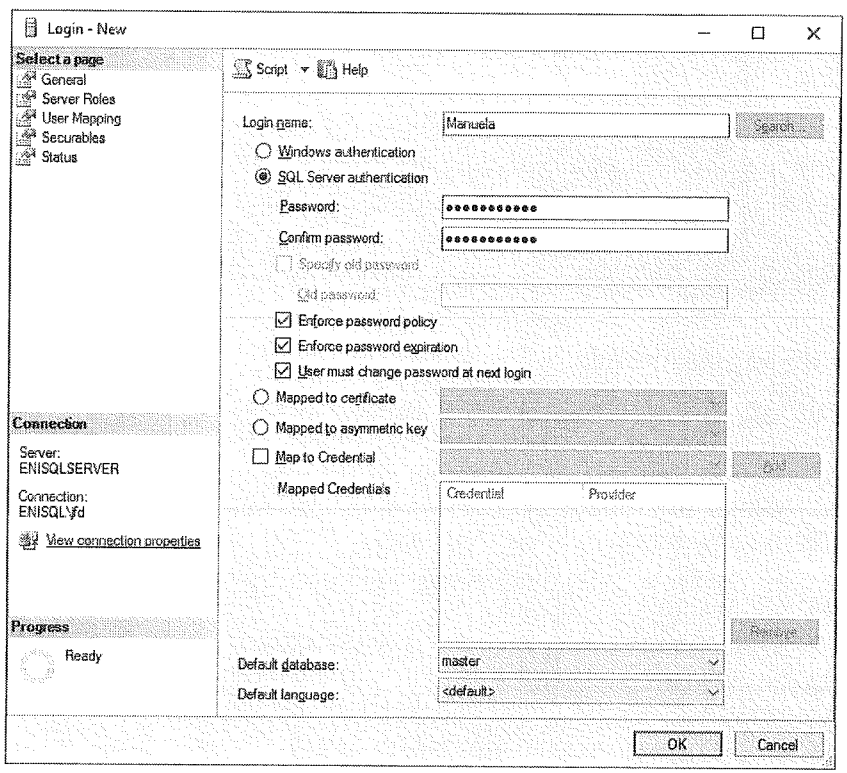
Para las conexiones que utilicen una seguridad de Windows, es el sistema operativo el que gestiona las reglas de seguridad relativas a la contraseña.

SQL Server Management Studio

Es posible crear conexiones desde la interfaz gráfica de SQL Server Management Studio, procediendo de la siguiente manera:

- Desde el explorador de objetos, situarse sobre el nodo **Security - Logins**.
- Desde el menú contextual asociado al nodo de conexión, seleccionar **New Login**.

Se muestra la pantalla siguiente. Solo falta definir el nombre de la nueva conexión, así como la contraseña asociada. A este nivel, también se puede indicar la política de gestión de los pasos que hay que establecer. Las reglas de gestión de las contraseñas se heredan de las establecidas en Windows.



La interfaz gráfica de SQL Server Management Studio también permite modificar simplemente una cuenta de conexión a partir de la ventana de propiedades o eliminar una conexión.

Transact SQL

```
CREATE LOGIN nombre_conexión
WITH {PASSWORD='contraseña' |ContraseñaHash HASHED}[MUST_CHANGED]
[,SID=sid |
,DEFAULT_DATABASE=baseDeDatos |
,DEFAULT_LANGUAGE=idioma |
,CHECK_EXPIRATION={ON | OFF} |
,CHECK_POLICY={ON | OFF}
,[CREDENTIAL=nombre_credencial]]
```

`nombre_conexión`

Nombre de la conexión que será definida en SQL Server.

Cuando la conexión corresponde a una cuenta de Windows, el nombre de la conexión tendrá el formato [nombreDominio\nombreCuenta] y no será posible utilizar el nombre en formato UPN, es decir, nombreCuenta@nombreDominio. En el marco de la seguridad mixta, cuando se crea una conexión SQL Server, el nombre de esta conexión debe respetar las reglas de nomenclatura de los identificadores SQL Server, por lo que no es posible utilizar el carácter \.

`contraseña`

Contraseña asociada a la conexión. Esta contraseña se almacena encriptada en la base Master y se verifica en cada nueva conexión al servidor. Es obligatorio definir una contraseña para cada conexión.

Las contraseñas tienen un mínimo de 8 caracteres y un máximo de 12; además, está habilitada la distinción entre mayúsculas y minúsculas para las contraseñas. Debido a razones obvias de seguridad, se aconseja elegir una contraseña robusta.

`ContraseñaHash`

Indica que la cadena proporcionada corresponde a la versión encriptada de la contraseña. Por lo tanto, se debe proporcionar, no la contraseña tal cual la ha rellenado el usuario, sino la contraseña tal cual la almacena SQL.

`MUST_CHANGED`

Con esta opción SQL Server solicita al usuario rellenar una nueva contraseña durante su primera conexión al servidor. Esta opción no es posible más que si `CHECK_EXPIRATION` y `CHECK_POLICY` están habilitadas (puestas a ON).

`baseDeDatos`

Indica la base de datos predeterminada que se utilizará. Si no se indica ninguna, entonces la base de datos predeterminada es la base Master.

`Idioma`

Idioma de trabajo predeterminado para esta conexión. Esta opción solo se indica si el idioma relativo a esta conexión es distinto al predeterminado que se ha definido en el servidor SQL.

`CHECK_EXPIRATION`

Si esta opción toma el valor ON (OFF predeterminado), indica que la cuenta de conexión va a seguir la regla relativa a la expiración de las contraseñas definida en el servidor SQL. La activación de esta opción solo se puede hacer cuando `CHECK_POLICY` también está habilitada.

`CHECK_POLICY`

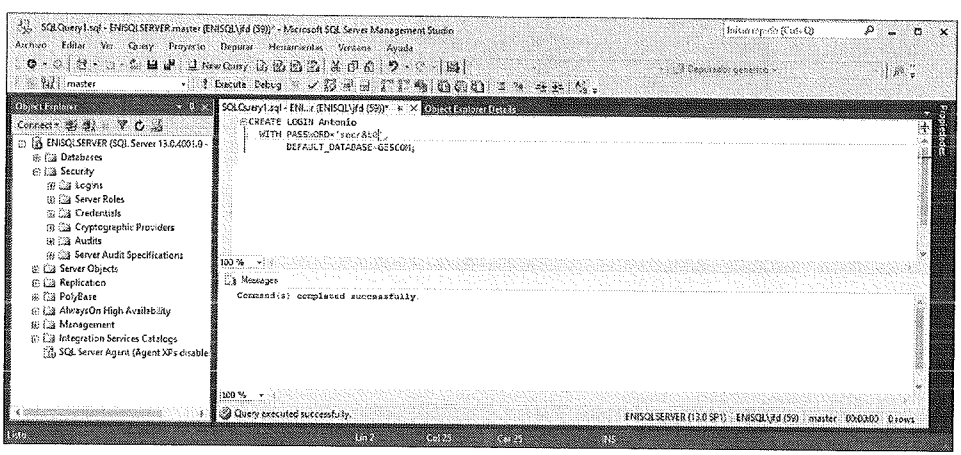
Está habilitada de manera predeterminada. Esta opción permite repercutir en SQL Server las reglas de gestión de las contraseñas definidas en Windows en el puesto en el que está la instancia de SQL Server.

`CREDENTIAL`

Permite relacionar la conexión a una credencial creada anteriormente con la instrucción `CREATE CREDENTIAL`.

Ejemplo

En el siguiente ejemplo, se crea la conexión Antonio con la contraseña secr&t0. El usuario no deberá cambiar su contraseña y se configura su base de datos predeterminada.



Uso de un certificado

SQL Server ofrece la posibilidad de crear conexiones al servidor basadas en certificados (CERTIFICATE). Estos certificados permiten identificar a un usuario de manera segura basándose en información como:

- Una clave pública.
- Una identificación, como el nombre y la dirección de correo electrónico.
- Un periodo de validez.

Para ser precisos, los certificados de SQL Server cumplen el estándar X.509.

La creación de un certificado en SQL Server se puede basar en un certificado definido en un archivo o un assembly o solicitar a SQL Server generar las claves públicas y privadas.

Los certificados de seguridad van a permitir a las aplicaciones o los servicios abrir una sesión segura en el servidor. Este tipo de inicio de sesión seguro por parte de un servicio se utiliza por el servicio Service Broker, que usa un certificado para enviar un mensaje sobre una base de datos remota.

Si el certificado se crea sin basarse en un archivo externo, la clave privada se encripta usando la clave principal de la base de datos.

2.6 Información de identificación

Estos objetos permiten a las conexiones en modo de seguridad de SQL Server acceder a los recursos externos del servidor de base de datos. La información de identificación (credenciales) contiene un nombre de cuenta de Windows y una contraseña. Las conexiones de SQL Server están asociadas a una credencial por medio de la instrucción CREATE LOGIN o ALTER LOGIN.

La modificación y la eliminación se realizan mediante las instrucciones ALTER CREDENTIAL y DROP CREDENTIAL.

Sintaxis

```
CREATE CREDENTIAL nombreDeCredencial
WITH IDENTITY = 'identidad' [, SECRET = 'secreta'];
```

`nombreDeCredencial`

Permite identificar la credencial de manera única en el servidor. Este identificador no puede comenzar por el carácter #. Las credenciales de sistemas comienzan siempre por los caracteres ##.

`identidad`

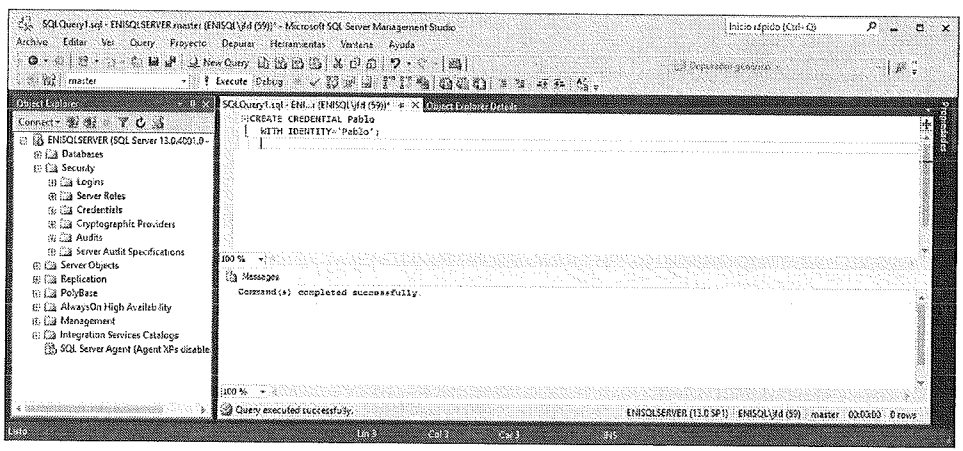
Nombre de la cuenta de Windows que estará asociada a la credencial y permitirá el acceso a los recursos externos a SQL Server.

`secreta`

Opcional. Este parámetro permite especificar una contraseña para poder acceder a los recursos externos.

Ejemplo

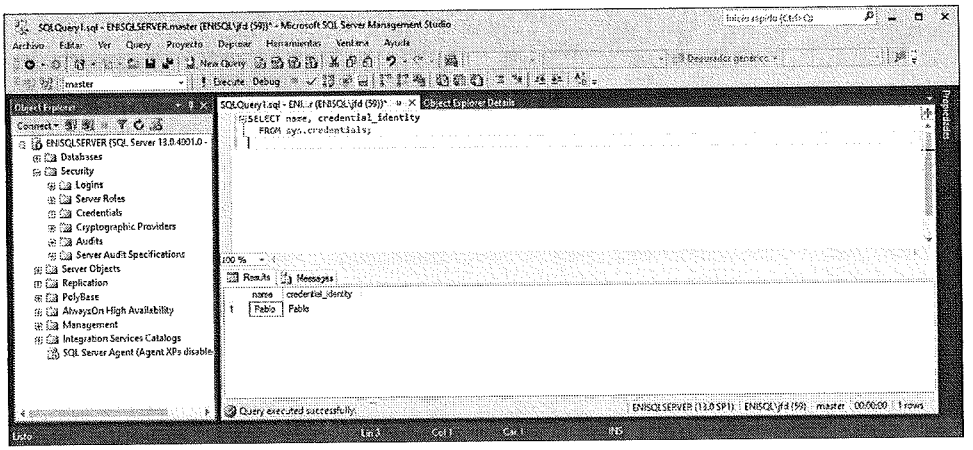
Se crea una credencial que corresponde a la cuenta de usuario Pablo definida a nivel de Windows.



Es posible tener toda la información relativa a las credenciales ya definidas sobre el servidor consultando la vista **sys.credential**.

Ejemplo

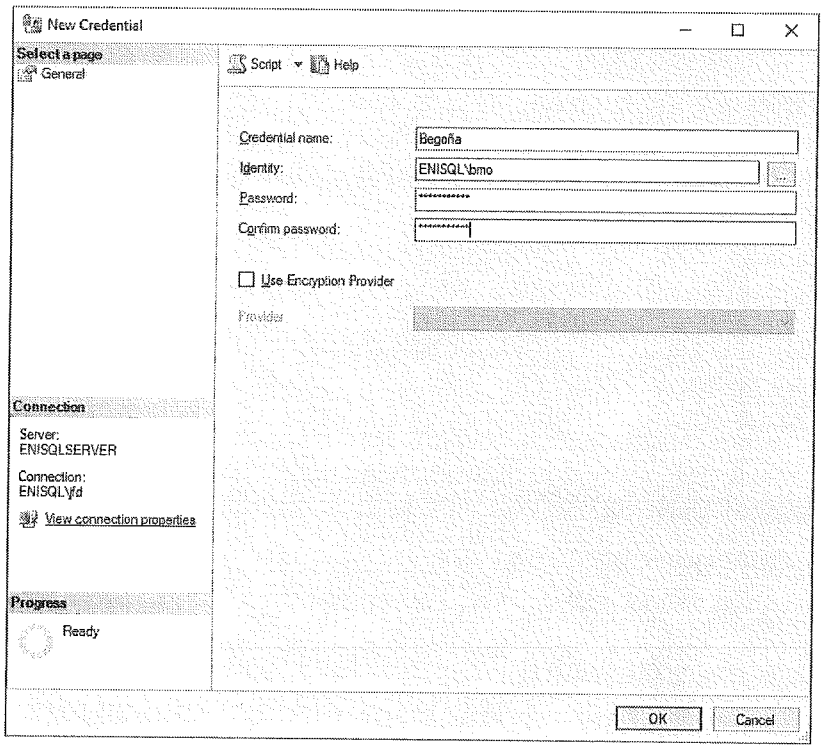
El siguiente script permite conocer las credenciales definidas y la conexión asociada.



SQL Server Management Studio

Es posible la gestión de una credencial de manera gráfica procediendo de la forma siguiente:

- Desde el explorador de objetos, situarse sobre el nodo **Security - Credentials**.
- Desde el menú contextual asociado al nodo **Credentials**, solicitar la creación de una credencial seleccionando la opción **New Credential...**.



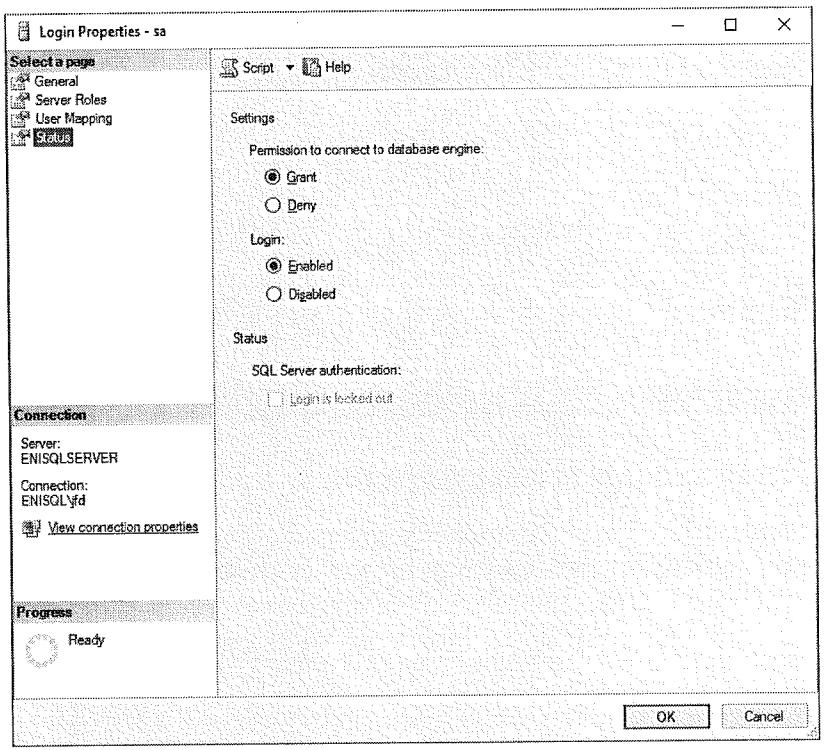
Ventana de creación de una nueva credencial

2.7 Activar y desactivar una conexión

En una conexión definida, es posible desactivarla para deshabilitar temporalmente su utilización. Si se conservan toda la información y las propiedades de la conexión, no es posible utilizar la conexión desactivada para conectarse a SQL Server. La desactivación de la conexión sirve para fortalecer la seguridad en cuentas de conexión que, aunque están definidas, no se utilizan, o bien cuando el administrador define inicialmente las conexiones. Solo tiene que activar estas conexiones cuando sea necesario.

SQL Server Management Studio

Desde la ventana de las propiedades de la conexión sobre la página **Status**, es posible habilitar o deshabilitar la conexión.



Transact SQL

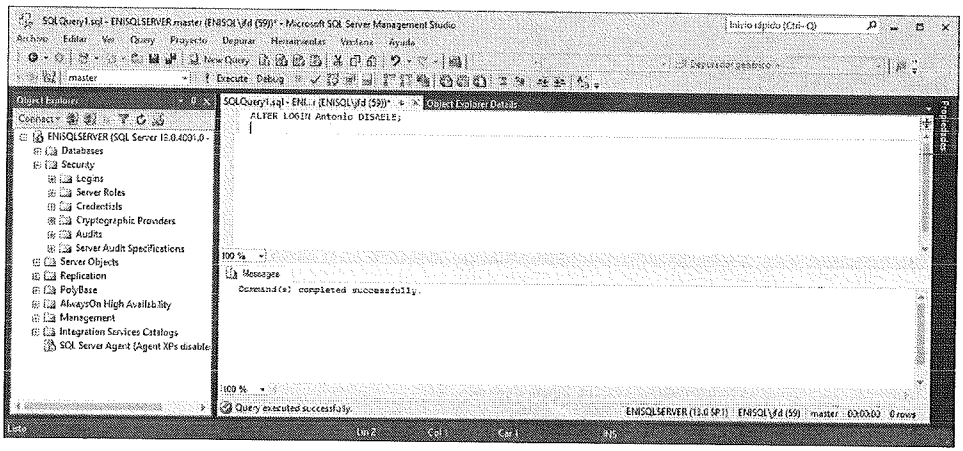
La habilitación y la deshabilitación de una conexión se efectúan por medio de la instrucción ALTER LOGIN.

Sintaxis

```
ALTER LOGIN nombreConexión {ENABLE|DISABLE}[;]
```

Ejemplo

Se desactiva la conexión Antonio.



2.8 Información relativa a las conexiones

Toda la información relativa a las conexiones se guarda en la base de datos, en las tablas de sistema de la base de datos Master. Es posible conocer esta información consultando las vistas de sistema que se enumeran a continuación. Se desaconsejan las acciones directas en las tablas para garantizar la ejecución correcta de los scripts en las diferentes versiones de SQL Server. Un script que se prueba y valida funcionará en las siguientes versiones de SQL Server.

sys.server_principals

Lista de las conexiones definidas en el servidor.

sys.sql_logins

Lista de las conexiones definidas con el modo de seguridad de SQL Server.

sys.server_permission

Lista de permisos asignados a las conexiones directamente en el servidor.

sys.server_role_member

Lista de los roles de servidor y de las conexiones que los usan.

sys.system_components_surface_area_configuration

Devuelve la lista de los objetos de sistema que se pueden ver o no, en función de la configuración efectuada. De esta manera, si la visibilidad está habilitada, un usuario puede ver los metadatos de un objeto, incluso si no tiene permisos para ello.

3. Gestión de los usuarios de la base de datos

Después de la definición de las conexiones (login) a nivel del servidor, es necesario definir los usuarios en las diferentes bases de datos.

Los permisos de uso de los objetos definidos en la base de datos se asignan a nivel de los usuarios de base de datos. Cuando se define una conexión, la base de datos predeterminada permite situar la cuenta de conexión sobre una base de datos para comenzar a trabajar. Sin embargo, la conexión solo podrá trabajar sobre la base de datos si existe una cuenta de usuario definida a nivel de la base de datos y asociada a la conexión. Este es un punto de paso obligatorio, salvo si se asignan a la conexión los privilegios de alto nivel.

Observación

Si no se define ninguna base de datos predeterminada a nivel de la conexión, entonces la base Master se considera la base de datos predeterminada, lo que no es una buena elección.

Los usuarios de base de datos se asocian a una conexión a nivel del servidor. Sin embargo, algunos usuarios, como **guest**, **sys** e **INFORMATION_SCHEMA**, no se mapean a ninguna conexión.

Si un usuario tiene una conexión a SQL Server pero no existe usuario de base de datos que le permita trabajar sobre las bases de datos, el usuario solo puede realizar operaciones muy limitadas:

- Seleccionar la información contenida en las tablas de sistema y ejecutar algunos procedimientos almacenados.
- Acceder a todas las bases de datos de usuario con una cuenta de usuario **guest**.
- Ejecutar las instrucciones que no necesitan autorización, como la función **PRINT**.

Existen dos tipos de permisos: los permisos de uso de los objetos definidos en una base de datos y los de ejecución de las instrucciones de SQL, que añaden nuevos objetos a la base de datos.

Los usuarios de base de datos corresponden a una conexión. Son los usuarios de base de datos quienes reciben los diferentes derechos que permiten utilizar las bases de datos.

Para ser capaz de gestionar los accesos a la base de datos, es necesario tener los permisos correspondientes al propietario de base de datos (**db_owner**) o al administrador de acceso (**db_accessadmin**).

Observación

*Los usuarios de base de datos se almacenan en la tabla de sistema **sysusers** de la base en la que se ha definido el usuario.*

Observación

Cuando se crea un usuario de base de datos, este no tiene ningún permiso. Es necesario asignar todos los permisos que el usuario necesite.

El usuario **guest** (invitado) es un usuario particular que se puede definir en las bases de datos de usuario. Este nombre de usuario se utilizará en las conexiones al servidor que no estén mapeadas con un usuario de base de datos. La gestión de este usuario es la misma que la de un usuario cualquiera de la base de datos.

La cuenta privada (guest) está habilitada en las bases de datos Master y tempdb. Esto forma parte del modo de funcionamiento normal de SQL Server y no es posible hacer excepciones a esta regla.

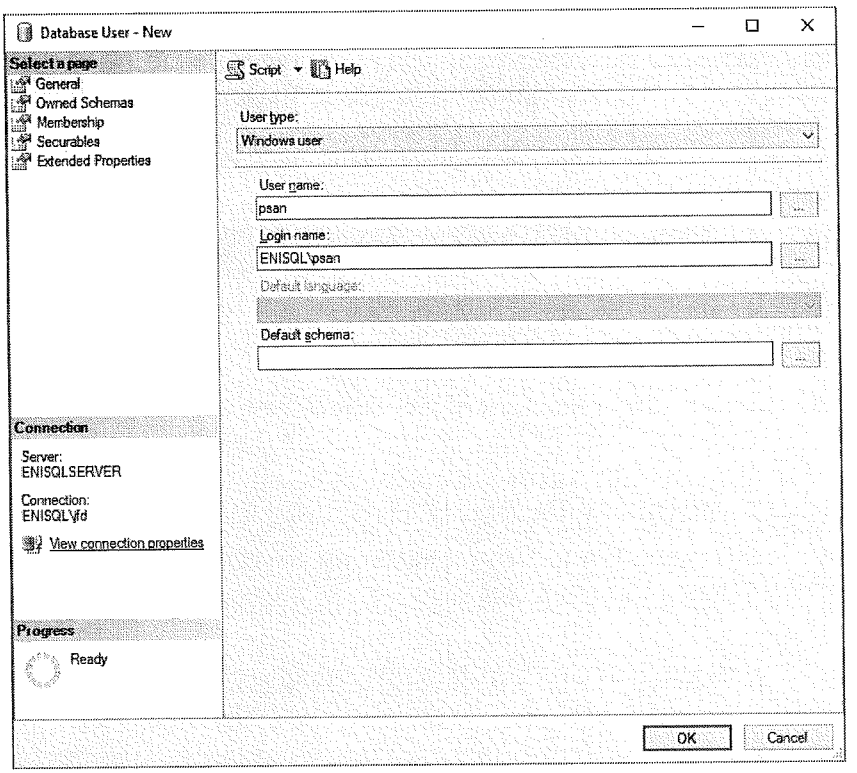
3.1 Crear un usuario

La creación de un usuario de base de datos va a permitir unir una conexión a un usuario y, por tanto, autorizar la utilización de la base de datos.

SQL Server Management Studio

Para crear un usuario de base de datos es necesario, desde el explorador de objetos, situarse sobre la base de datos en la que se va a operar y realizar las acciones siguientes:

- Localizar el nodo **Security** y situarse sobre el nodo **Logins**.
- Desde el menú contextual asociado al nodo **Logins**, seleccionar **New Login**.
- Seleccionar la conexión asociada al usuario y posteriormente definir el nombre de este último.
- Finalmente, es posible indicar los roles de base de datos asignados a este usuario.



En el ejemplo anterior, se crea el usuario psan en la base de datos GESCOM. Este usuario se mapea a la conexión psan.

Transact SQL

La instrucción CREATE USER permite definir cuentas de usuarios en la base de datos.

Sintaxis

```
CREATE USER nombreUsuario
[ FOR {LOGIN nombreConexión |
   CERTIFICATE nombreCertificado|
   ASYMMETRIC KEY nombreClaveAsimétrica}]
[ WITH DEFAULT_SCHEMA = nombreEsquema ]
```

`nombreUsuario`

Nombre del nuevo usuario de base de datos.

`nombreConexión`

Nombre de la conexión a la que se asocia el usuario de base de datos. En caso de no asignarse ningún nombre de conexión, SQL Server resuelve el problema de la siguiente manera:

- 1: SQL Server intenta asociar la cuenta de usuario de base de datos a una conexión que tenga el mismo nombre.
- 2: Si el paso anterior falla, si el nombre de usuario es **guest** y no existe aún una cuenta **guest** en la base de datos, entonces se crea la cuenta de usuario **guest** (invitado).
- 3: En el resto de los casos, la instrucción falla.

`nombreCertificado`

Nombre del certificado que se va a asociar al usuario de base de datos.

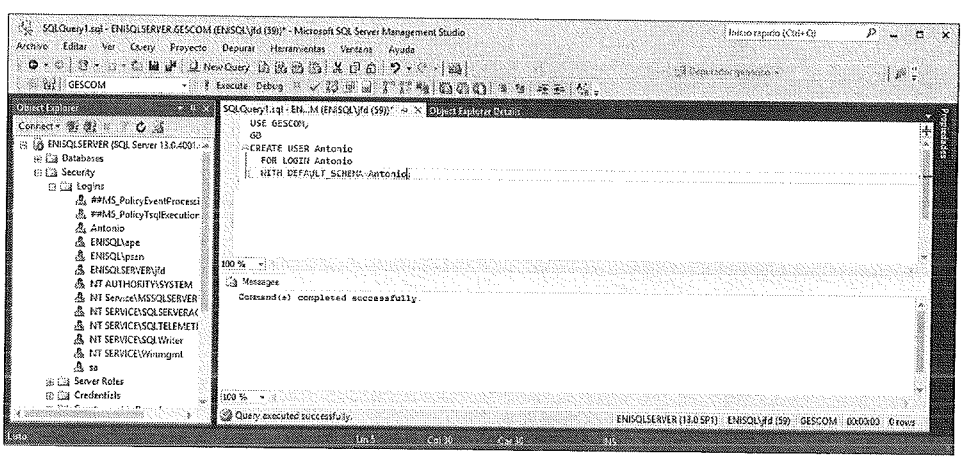
`nombreClaveAsimétrica`

Nombre de la clave asimétrica asociada al usuario de base de datos.

`nombreEsquema`

Nombre del esquema asociado a este usuario de base de datos. Es posible asociar varios usuarios al mismo esquema.

Ejemplo



Observación

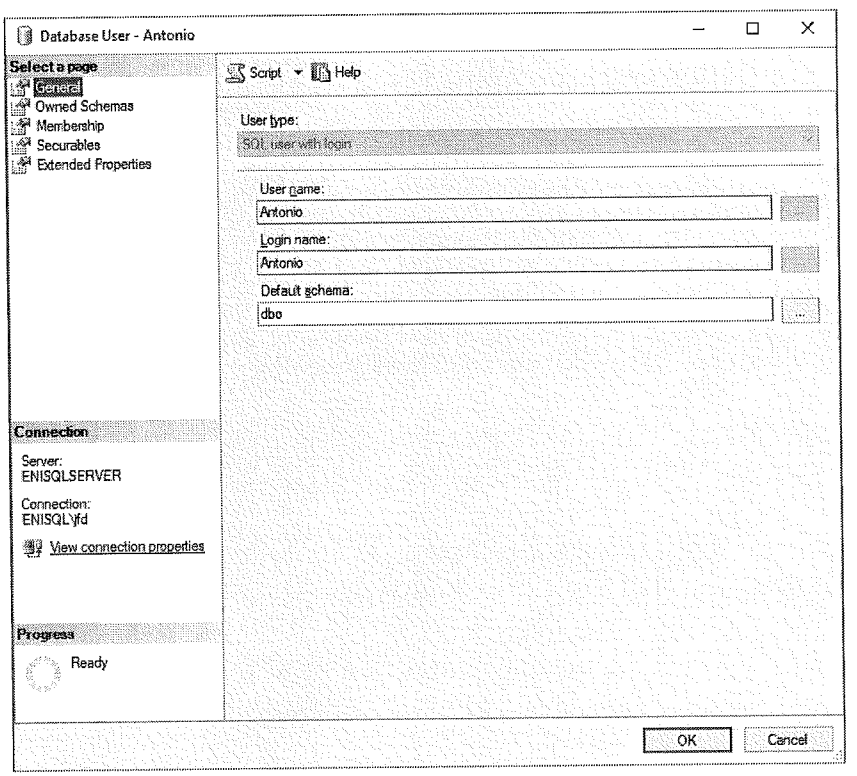
Los procedimientos sp_grantdbaccess y sp_adduser se mantienen en SQL Server únicamente por razones de compatibilidad. No se recomienda utilizarlos.

3.2 Información

SQL Server Management Studio

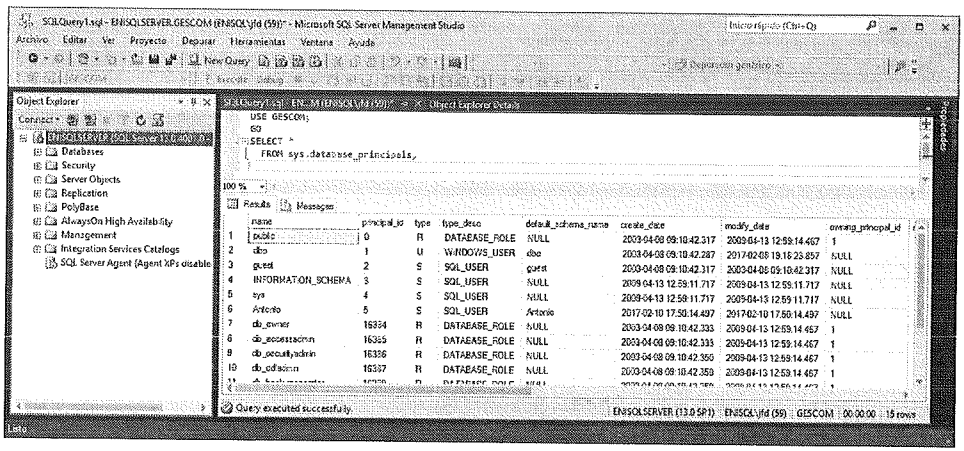
Para obtener el conjunto de información relativa a un usuario de base de datos, es necesario proceder de la siguiente manera:

- Desde el explorador de objetos, situarse sobre la base de datos.
- Localizar los nodos **Security - Logins**.
- Situarse sobre el usuario para el que se desea obtener información y solicitar que se muestren las propiedades a partir del menú contextual asociado.



Transact SQL

Es posible obtener la información de los usuarios de base de datos consultando la vista **sys.database_principals**.



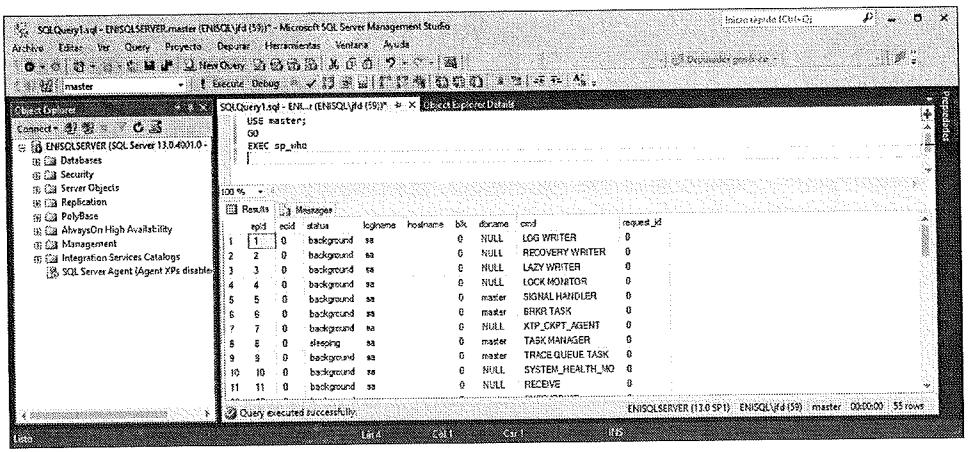
Observación

El procedimiento almacenado sp_helpuser se mantiene por razones de compatibilidad, pero es preferible utilizar la vista ***sys.database_principals.***

El procedimiento **sp_who** permite conocer los usuarios actualmente conectados y los procesos en curso.

Para cada conexión, el procedimiento **sp_who** permite conocer, entre otra, la siguiente información:

- El nombre de conexión utilizado (loginame).
- El nombre del equipo desde el que se establece la conexión (hostname).
- El nombre de la base de datos actual (dbname).



3.3 Establecer la lista de conexiones y usuarios

Para garantizar la seguridad de la instancia de SQL Server, es necesario controlar quién puede acceder a la información. La situación se complica rápidamente, ya que el número de usuarios y de bases de datos va en aumento. Por este motivo, es interesante establecer una lista de conexiones y de usuarios de base de datos para cada una de ellas, así como de los permisos que se asignan a cada usuario.

Para realizar esta lista, lo más sencillo es usar las vistas de sistema con objeto de hacer el proceso automático. Esta consulta se basa en las vistas de sistema **sys.database_principals** de la base de datos, para obtener la lista de usuarios que se han definido en ella, y la vista **sys.server_principals**, que proporciona la lista de conexiones definidas.

Ejemplo

La siguiente consulta permite establecer la lista de usuarios de la base de datos GESCOM y su conexión asociada:

```
SELECT s.name as "Conexion", p.name as "Usuario"
FROM sys.database_principals p
INNER JOIN sys.server_principals s
ON s.sid=p.sid;
```

La vista **sys.databases** también se usará para recorrer todas las bases de datos que haya en la instancia de SQL Server.

El siguiente script permite construir esta lista completa. Para esto, se utiliza un cursor que recorre todas las bases de datos. Para limitar la lista que se devuelve y ganar en claridad, solo se tienen en cuenta las bases de datos de usuario, lo que explica la exclusión de la base de datos Master, model, tempdb y msdb. Para cada valor que devuelve el cursor, se ejecuta la consulta anterior. Sin embargo, como una parte de la consulta es dinámica (el nombre de la base de datos), es necesario construir la consulta como una cadena de caracteres (lo que se hace dando valor a la variable @consulta). Para ejecutar esta consulta dinámica, se utiliza el procedimiento almacenado **sp_executesql**.

```
DECLARE cLasBases cursor for select name from sys.databases where name
not in(
'master', 'model', 'tempdb', 'msdb');
DECLARE @nombreBase sysname,
DECLARE @consulta nvarchar(500)
BEGIN
   OPEN cLasBases;
   FETCH cLasBases INTO nombreBase;
   While  (@@fetch_status = 0) begin
     set @consulta = 'SELECT ''' + @nombreBase + ''' as Base,'
     set @consulta = @consulta + 's.name' as Conexion, p.name as Usuario '
     set @consulta = @consulta + 'FROM MASTER.SYS.DATABASE_PRINCIPALS p '
     set @consulta = @consulta + 'INNER JOIN ' + @nombreBase + '
.sys.server_principals s'
     set @consulta = @consulta + 'ON s.sid = p.sid'
     exec sp_executesql @consulta
     FETCHcLasBases INTO @nombreBase;
   End
   Close cLasBases;
   DEALLOCATE cLasBases
End;
```

Observación

Estos scripts, como el resto presente en este libro, están disponibles para su descarga desde el sitio de Ediciones ENI.

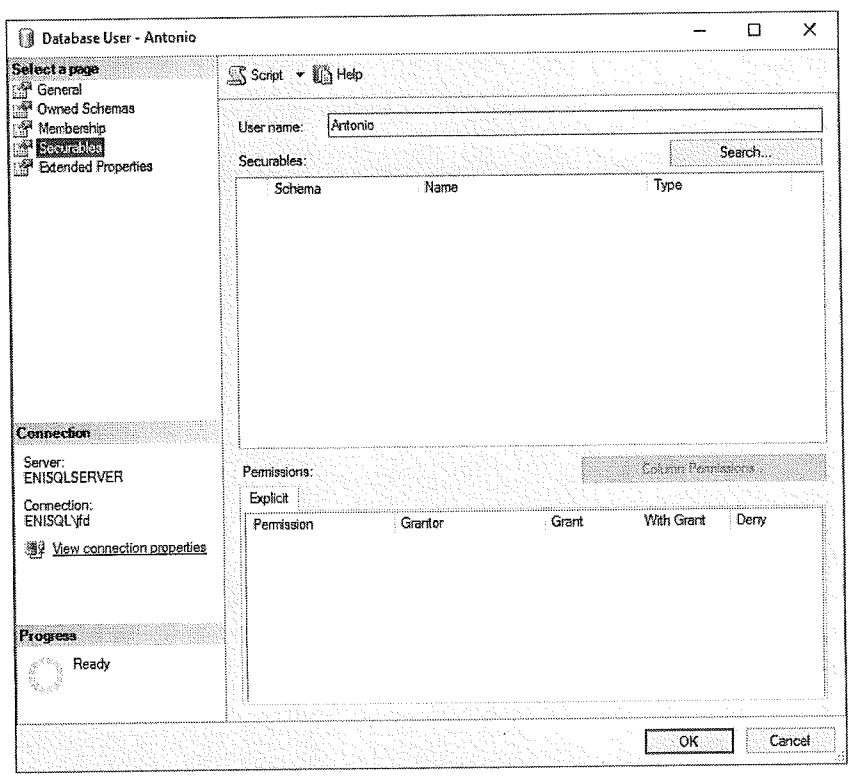
3.4 Modificación

Es posible modificar un usuario de base de datos para cambiar su nombre o el esquema asociado a él.

SQL Server Management Studio

Es necesario situarse en el explorador de objetos, sobre la base de datos afectada por la modificación de la cuenta de usuario, y realizar las acciones siguientes:

- Localizar el nodo **Security** y posteriormente **Logins**.
- Situarse sobre la cuenta de usuario que hay que modificar.
- Visualizar la ventana de propiedades desde la opción **Properties**, situada en el menú contextual asociado al usuario.



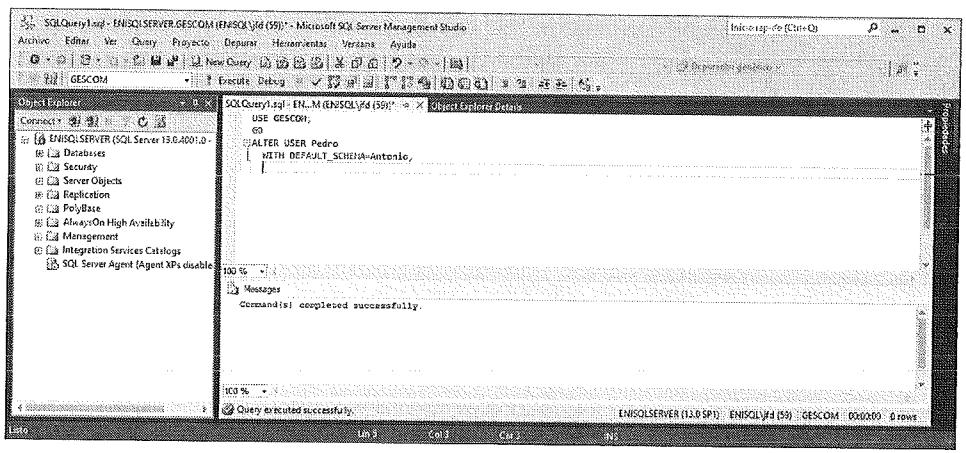
Transact SQL

La instrucción ALTER USER permite realizar esta operación.

Sintaxis

```
ALTER USER nombreUsuario
WITH NAME=nuevoNombreUsuario,
   DEFAULT_SCHEMA = nombreNuevoEsquema
```

Ejemplo



3.5 Eliminación

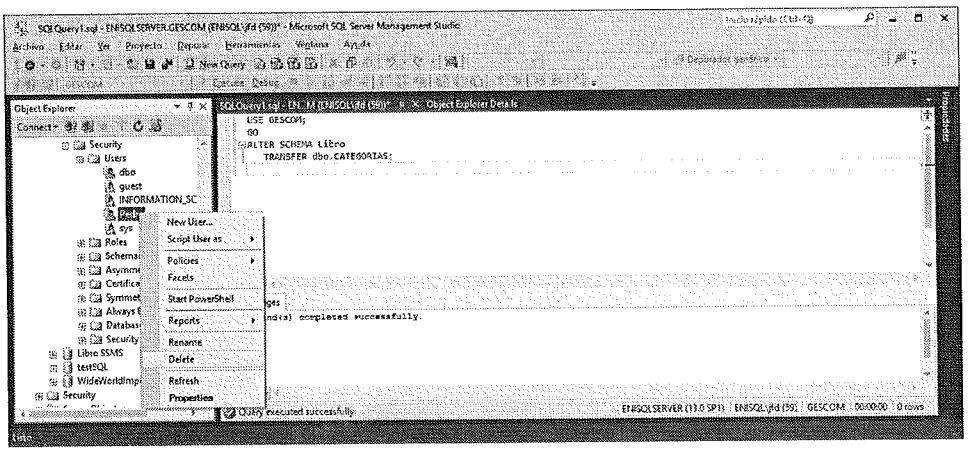
Teniendo en cuenta que los usuarios no tienen objetos, su eliminación siempre es posible sin ningún riesgo de pérdida de datos. La eliminación de un usuario no implica la eliminación del esquema asociado al usuario.

SQL Server Management Studio

Es necesario situarse, desde el explorador de objetos, sobre la base de datos afectada por la eliminación de la cuenta de usuario y realizar las acciones siguientes:

- Localizar el nodo **Security** y a continuación **Logins**.
- Situarse sobre la cuenta de usuario que hay que eliminar.

– Seleccionar la opción **Delete** desde el menú contextual asociado a la cuenta de usuario.



La eliminación no es posible sobre las cuentas de usuario predefinidas, a saber: dbo, guest, INFORMATION_SCHEMA y sys.

Transact SQL

La instrucción DROP USER permite realizar esta operación.

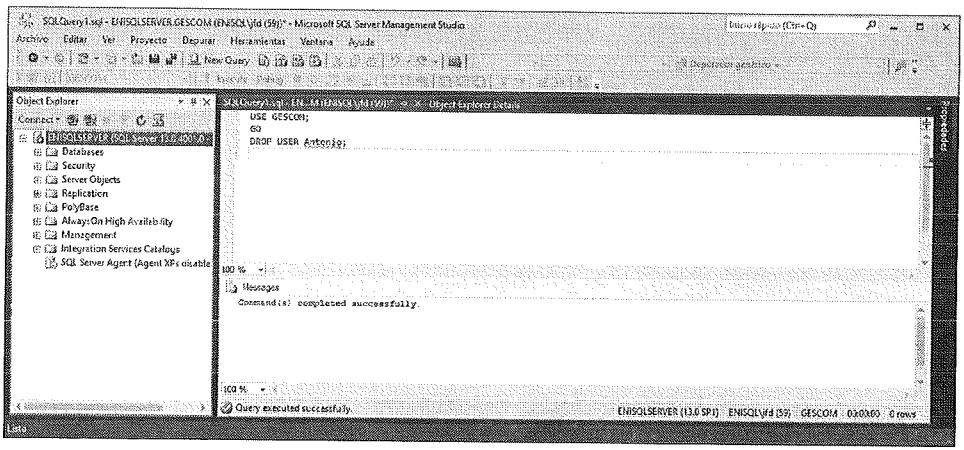
Observación

*Los procedimientos **sp_dropuser** y **sp_revokedbaccess** se mantienen en SQL Server únicamente por razones de compatibilidad.*

Sintaxis

```
DROP USER nombreUsuario
```

Ejemplo



4. Administración de los esquemas

El objetivo de los esquemas es separar los usuarios de base de datos de los objetos que pueden crear. Sin embargo, los objetos no se dejan tal cual, sino que se agrupan de manera lógica en el esquema. De esta manera, es posible definir un esquema como un conjunto lógico de objetos de una base de datos.

Permitiendo esta agrupación lógica de tablas, vistas, funciones y procedimientos en la base de datos, los esquemas ofrecen una mejor legibilidad sin complicar la estructura.

Los esquemas facilitan la compartición de información entre varios usuarios sin perder el nivel de seguridad. Por ejemplo, si varios desarrolladores trabajan en conjunto sobre un mismo proyecto, todos ellos se pueden conectar utilizando su propia conexión y usuario de base de datos, lo que no impide trabajar sobre el mismo esquema y, de esta manera, compartir las tablas, vistas, procedimientos, funciones... que se definen sobre la base dentro del proyecto.

Los esquemas facilitan la administración de los permisos de uso de los objetos que los componen, ya que es posible asignar los permisos de uso de los objetos directamente a nivel de estos.

Es responsabilidad del propietario del esquema gestionar los permisos de uso del esquema y de los objetos presentes en él. El propietario del esquema puede transferir la gestión de las tablas/vistas a otros usuarios.

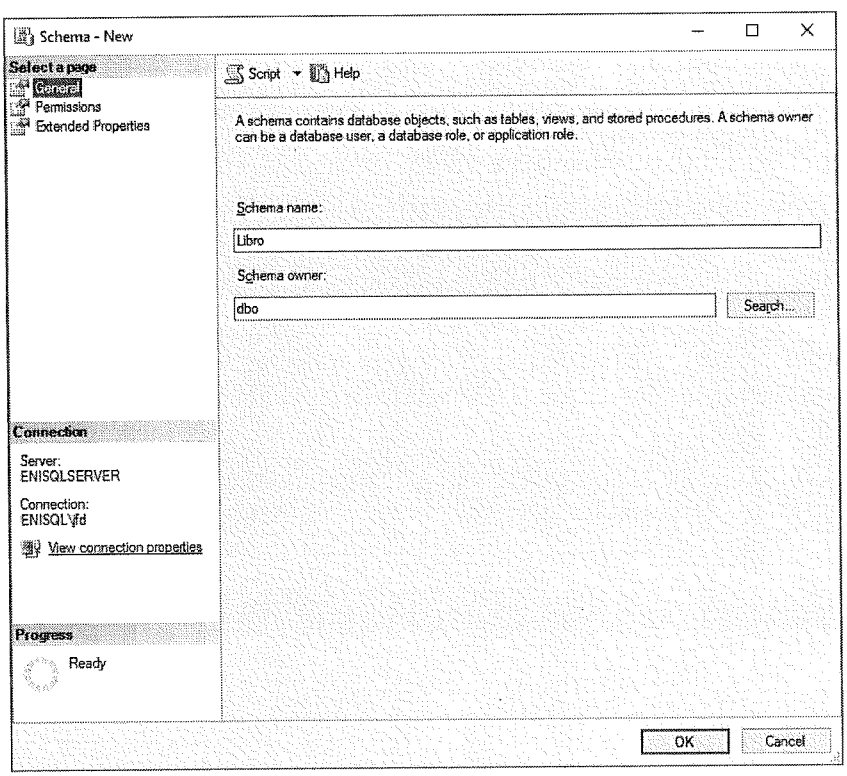
Para acceder a los objetos situados fuera de su esquema predeterminado, un usuario de base de datos debe utilizar el nombre completo de un objeto, es decir, nombreEsquema.nombreObjeto. Cuando se usa un nombre corto (simplemente el nombre del objeto, sin indicar el nombre del esquema), SQL Server busca ese objeto únicamente en el esquema actual del usuario.

4.1 Creación

SQL Server Management Studio

Para crear un esquema de base de datos, es necesario situarse sobre la base de datos afectada y realizar las acciones siguientes desde el explorador de objetos:

- Localizar el nodo **Security** y situarse sobre el nodo **Schemas**.
- Desde el menú contextual asociado al nodo **Schemas**, seleccionar **New Schema**.



Transact SQL

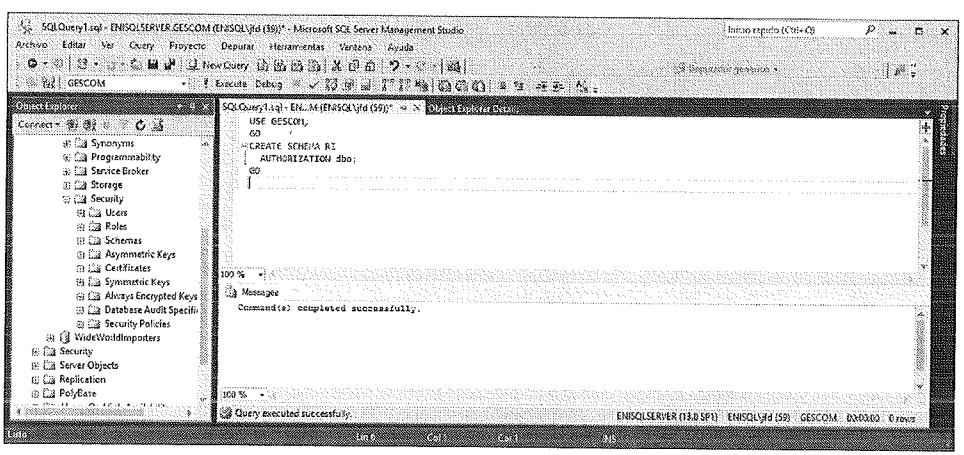
```
CREATE SCHEMA nombreEsquema
 AUTHORIZATION nombrePropietario|
 [definiciónDeTablas |
  definiciónDeVistas |
  gestiónDePrivilegios]
```

De manera más sencilla, es posible decir que un esquema se caracteriza por un nombre. La opción AUTHORIZATION permite definir el usuario de base de datos que es propietario del esquema. Un mismo usuario de base de datos puede ser el propietario de varios esquemas. El propietario de un esquema no ha de tener necesariamente como esquema predeterminado uno del que es propietario. También es posible crear las tablas y las vistas de un esquema desde la construcción del propio esquema.

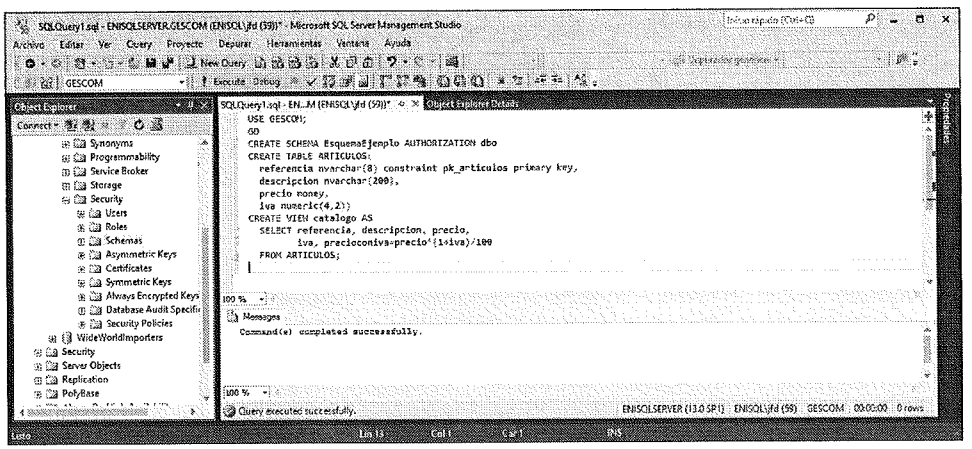
Este caso es el mismo que el de las operaciones DDL, es decir, la acción de la instrucción CREATE es indivisible, por lo que se crean todos los objetos o no se crea ninguno. De esta manera, un problema de creación sobre una tabla o vista impide la creación del esquema completo.

Ejemplo

En el ejemplo siguiente, se crea el esquema RI.



En este segundo ejemplo, se define el esquema EsquemaEjemplo, así como las tablas y las vistas específicas de este esquema.



4.2 Modificación

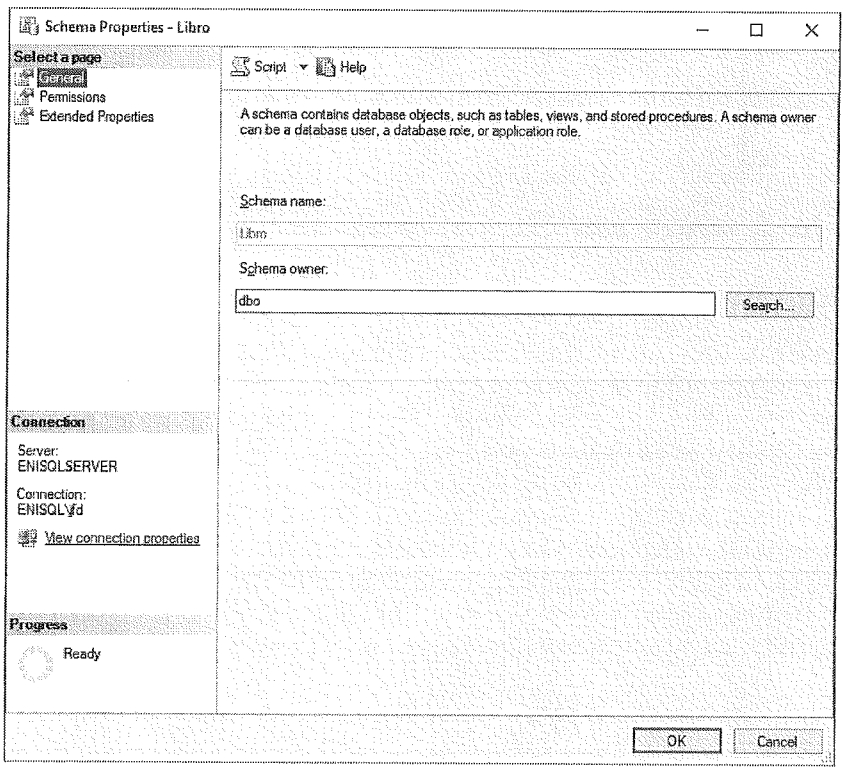
La modificación de un esquema consiste en modificar los objetos contenidos en este esquema. Cuando un objeto se transfiere de un esquema a otro, los permisos relativos a estos objetos se pierden. La transferencia de un objeto entre dos esquemas es posible en el interior de una misma base de datos.

La transferencia de un objeto de un esquema a otro tiene algunas consecuencias. Si no se especifica explícitamente el propietario, es decir, que el objeto pertenece al propietario del esquema (SCHEMA OWNER), entonces después de la transferencia es el propietario del esquema de destino el que se convierte en propietario del objeto. Por otra parte, si los permisos de uso del objeto se han definido antes de la transferencia hacia el nuevo esquema, estos permisos se eliminan como consecuencia de la transferencia.

SQL Server Management Studio

Para modificar un esquema de base de datos, es necesario situarse sobre la base de datos en cuestión y realizar las acciones siguientes desde el explorador de objetos:

- Localizar el nodo **Security** y situarse sobre el nodo **Schemas**.
- Situarse sobre el esquema que hay que modificar.
- Visualizar la ventana de propiedades desde la opción **Properties**, disponible en el menú contextual asociado al esquema.



Transact SQL

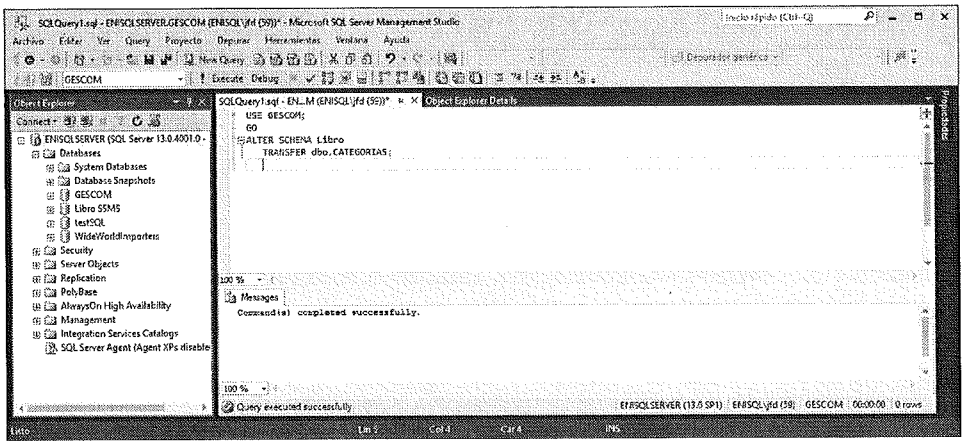
```
ALTER SCHEMA nombreEsquema TRANSFER nombreObjeto
```

`nombreObjeto`

Representa el nombre del objeto que va a transferirse al esquema actual. El nombre de este objeto sigue el formato nombreEsquema.nombreCortoObjeto.

Ejemplo

En el ejemplo siguiente, la tabla CATEGORIAS presente en el esquema dbo se transfiere al esquema libro.



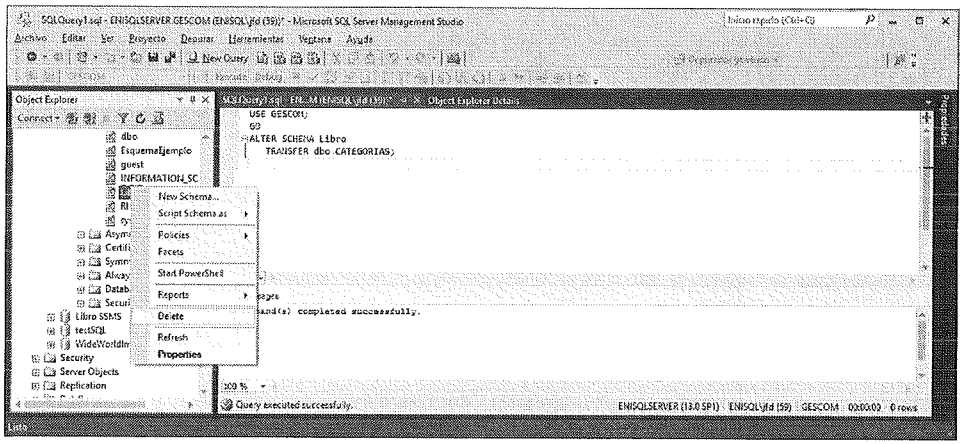
4.3 Eliminación

La eliminación de un esquema es posible solo para los esquemas que no contienen objetos. Por ejemplo, si el esquema contiene tablas o vistas, es necesario eliminarlas o transferirlas a otro esquema antes de poder eliminar el esquema actual.

SQL Server Management Studio

Para eliminar un esquema de base de datos, es necesario situarse sobre la base de datos en cuestión y realizar las acciones siguientes desde el explorador de objetos:

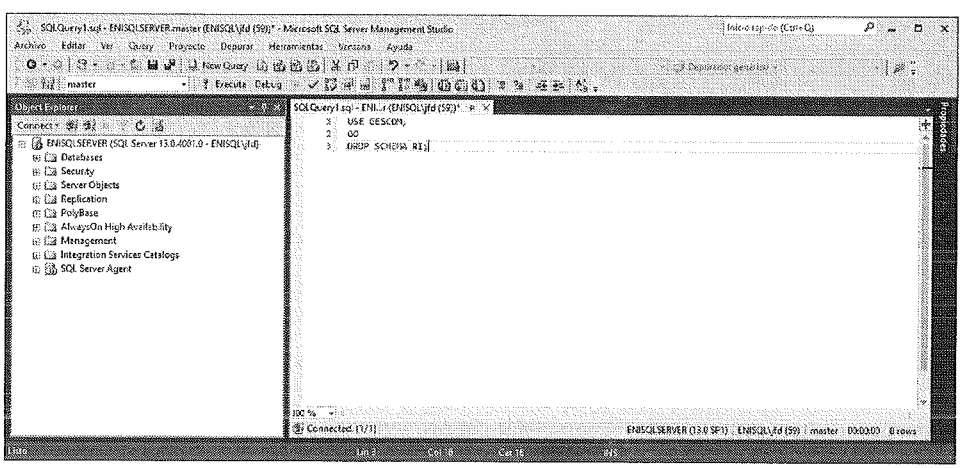
- Localizar el nodo **Security** y situarse sobre el nodo **Schemas**.
- Situarse sobre el esquema que hay que eliminar.
- Seleccionar la opción **Delete** desde el menú contextual asociado al esquema.



Transact SQL

```
DROP SCHEMA nombreEsquema
```

Ejemplo



4.4 La información relativa a los esquemas

Es posible encontrar la información de cada esquema consultando la vista **sys.schemas**. Esta vista tiene las siguientes columnas:

Name

Representa el nombre del esquema. No puede haber dos esquemas con el mismo nombre en una base de datos.

Schema_id

Identificador numérico que se asigna al esquema.

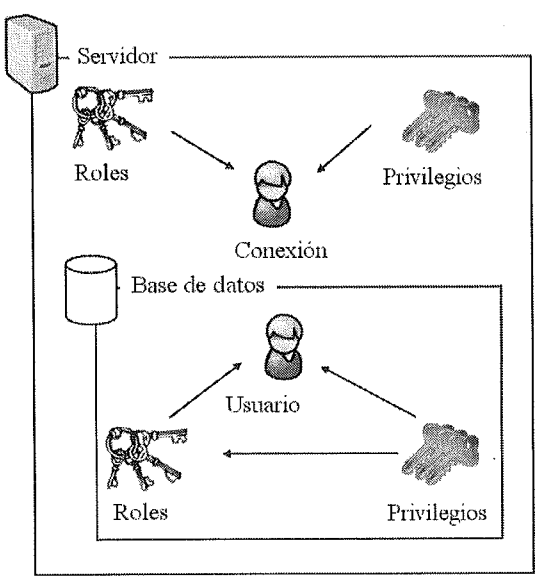
Principal_id

Identificador numérico que corresponde al propietario del esquema.

5. Administración de los permisos

Todos los usuarios de base de datos, incluido **guest** (el invitado), pertenecen al grupo **public**. Los derechos que se detallan a continuación se pueden asignar directamente a **public**.

Los derechos se organizan de manera jerárquica con relación a los elementos del servidor a los que se puede dar seguridad.



Es posible gestionar la asignación de privilegios a nivel del servidor, de la base de datos, del esquema o directamente a nivel del objeto. De esta manera, los privilegios se pueden asignar tanto a un usuario de base de datos como a una conexión.

SQL Server gestiona los privilegios con tres tipos de palabras claves:

- GRANT
- REVOKE
- DENY

Es decir, un privilegio se puede asignar (GRANT) o retirar (REVOKE) si se ha asignado previamente. La instrucción DENY permite prohibir el uso de un privilegio particular, aunque el privilegio en cuestión haya sido asignado directamente o por medio de un rol.

5.1 Permisos de uso de las instrucciones

Los permisos de uso de las instrucciones SQL para crear nuevos objetos de la base de datos son los permisos para ejecutar algunas sentencias SQL. Un usuario que dispone de estos permisos puede, por ejemplo, crear sus propias tablas, procedimientos... La asignación de estos permisos no debería poder llegar a ser peligrosa y, como para todos los permisos, se deben asignar solo cuando sea necesario.

Los permisos principales de instrucciones disponibles son:

- CREATE DATABASE
- CREATE PROCEDURE
- CREATE FUNCTION
- CREATE TABLE
- BACKUP DATABASE
- CREATE VIEW
- BACKUP LOG

Observación

Es posible obtener una vista gráfica de todos los permisos disponibles en SQL Server 2014 descargando el documento pdf del sitio Web de Microsoft en la siguiente dirección: http://go.microsoft.com/fwlink/?linkid=229142

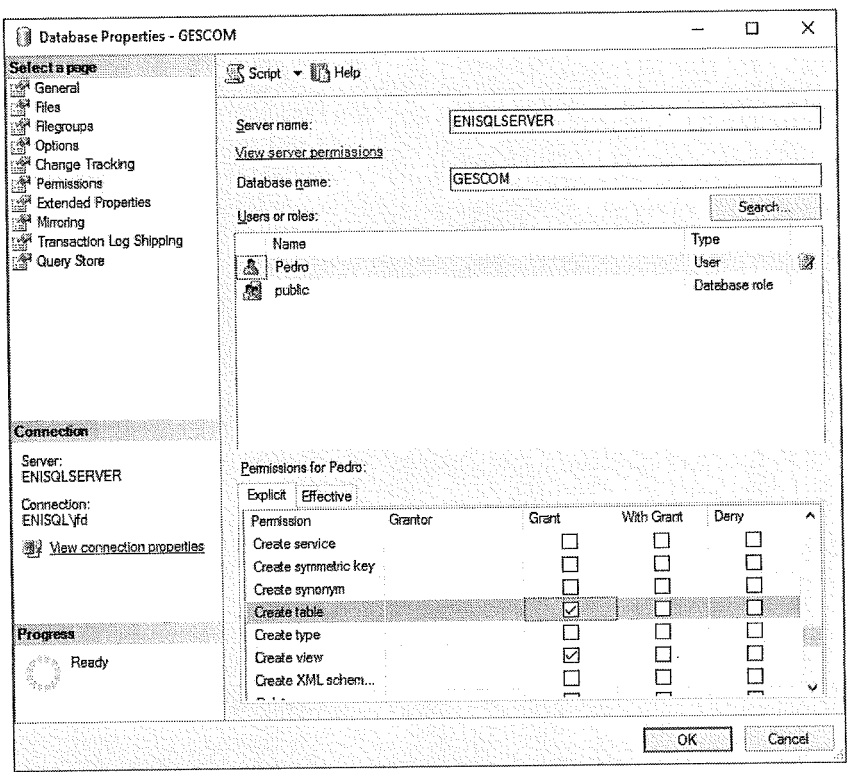
5.1.1 Autrizar

SQL Server Management Studio

Estos derechos se administran a nivel de la base de datos mediante la ventana de propiedades.

Ejemplo

El privilegio CREATE TABLE se asigna al usuario de base de datos Pedro por medio de la ventana de propiedades de la base GESCOM.



Transact SQL

La asignación de privilegios se efectúa utilizando la instrucción GRANT, cuya sintaxis se detalla a continuación.

```
GRANT permiso [,...]
    TO usuario[,...]
    [ WITH GRANT OPTION ]
```

`permiso`

Nombre del/de los permiso(s) relativo(s) a esta autorización. También es posible utilizar la palabra clave ALL en lugar de citar explícitamente el permiso o los permisos asignados. Sin embargo, este término ALL no permite asignar los privilegios de ejecución de todas las instrucciones, sino simplemente sobre las instrucciones para crear las bases de datos, tablas, procedimientos, funciones y para efectuar las copias de seguridad de la base de datos y del diario.

`usuario`

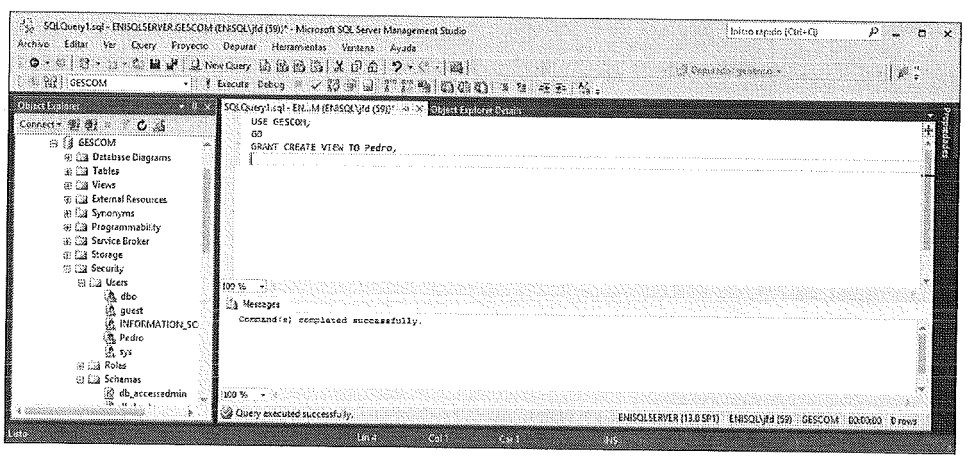
Nombre del usuario o de los usuarios de base de datos que recibe(n) los permisos.

`WITH GRANT OPTION`

Si el permiso se recibe con este privilegio, el usuario puede asignar el permiso a otros usuarios de base de datos.

Ejemplo

En el ejemplo siguiente, se asigna el privilegio CREATE VIEW al usuario de base de datos Pedro.



5.1.2 Retirar

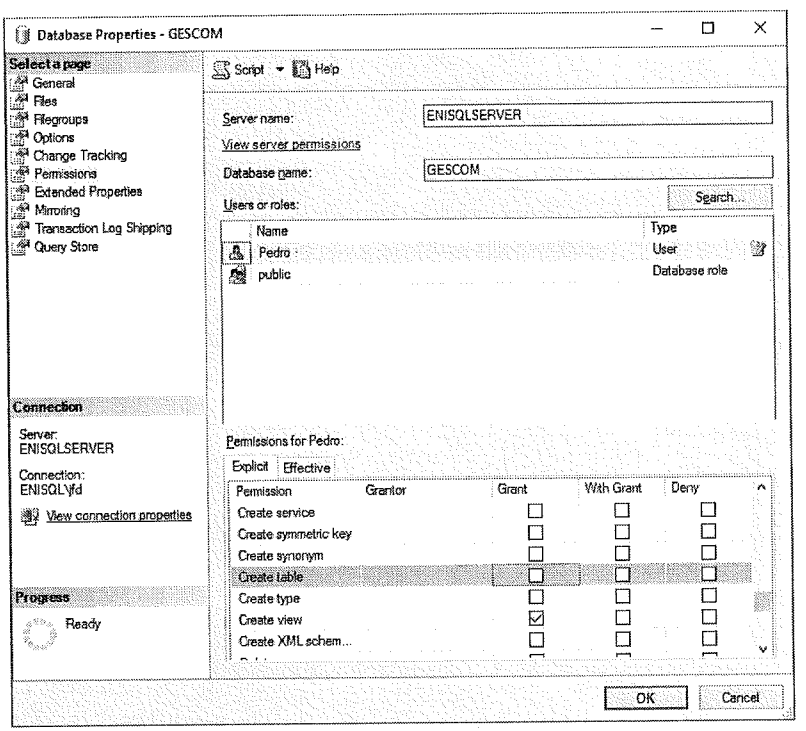
Es posible retirar un privilegio que ha sido asignado a una entidad de seguridad. Si el privilegio no ha sido asignado previamente a la entidad de seguridad, la instrucción no realiza ninguna acción.

SQL Server Management Studio

La gestión de los privilegios se efectúa a nivel de las propiedades de la base de datos.

Ejemplo

Se retira el permiso CREATE TABLE al usuario Pedro.



Transact SQL

```
REVOKE [GRANT OPTION FOR]
       permiso [,...]
       FROM usuario [,...]
       [CASCADE]
```

`GRANT OPTION FOR`

Si el permiso se ha asignado con el privilegio de administración GRANT OPTION, es posible, por medio de esta opción, retirar simplemente el parámetro de administración.

`permiso`

Lista de permisos que hay que retirar.

`usuario`

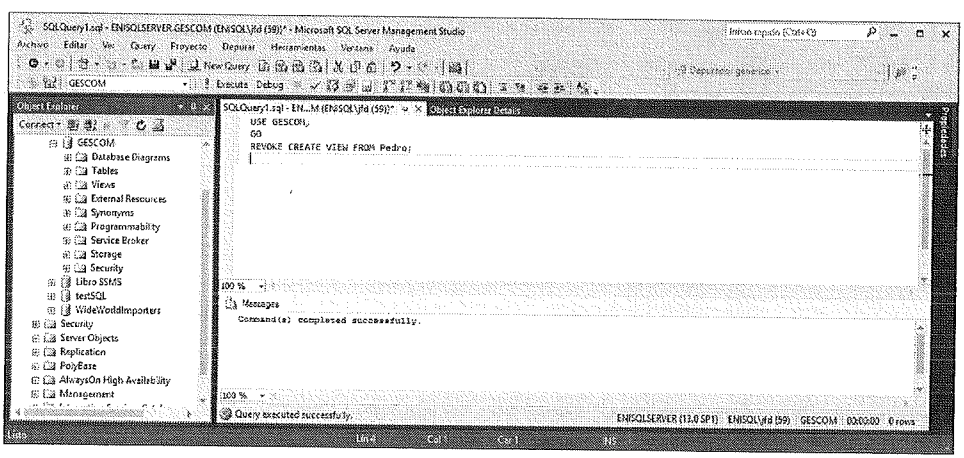
Lista de usuarios afectados por esta eliminación.

`CASCADE`

Si el permiso retirado se ha asignado con el privilegio de administración WITH GRANT OPTION, la opción CASCADE permite retirar el privilegio a los usuarios que lo han recibido a través del usuario afectado por la eliminación inicial del privilegio.

Ejemplo

Se retira el privilegio CREATE VIEW al usuario Pedro.



5.1.3 Prohibir

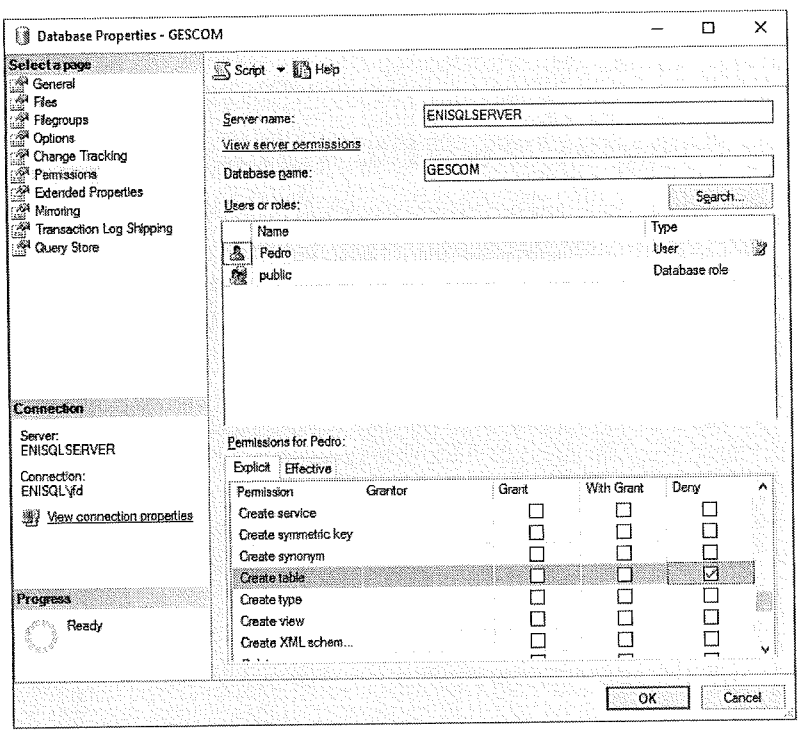
La instrucción DENY permite prohibir a un usuario la utilización de un privilegio, aunque haya recibido el permiso directamente o por su pertenencia a un grupo.

SQL Server Management Studio

Como para la asignación y la eliminación, este tipo de privilegio se gestiona de manera centralizada a nivel de las propiedades de la base de datos.

Ejemplo

A través de las propiedades de la base de datos, se va a prohibir al usuario Pedro crear una tabla.

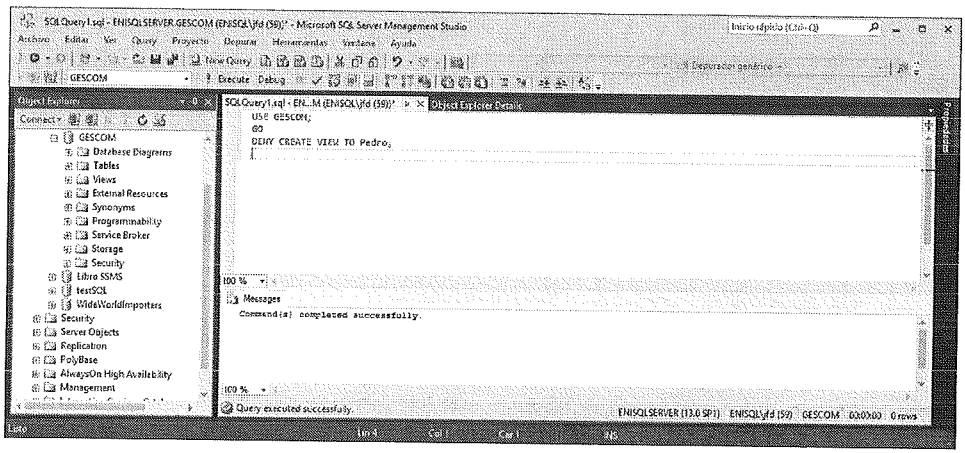


Transact SQL

```
DENY permiso [,...]
      TO usuario [,...]
      [CASCADE]
```

Ejemplo

Se prohíbe crear vistas al usuario Pedro.



5.2 Derechos de utilización de los objetos

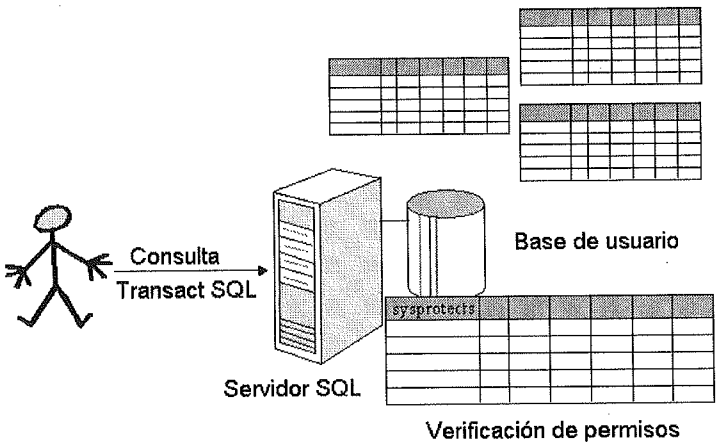
Estos derechos permiten fijar las operaciones (lectura, modificación, adición o eliminación) que el usuario puede realizar sobre los datos contenidos en una tabla, o bien dar el derecho de ejecución de un procedimiento almacenado. Estos derechos son, en general, gestionados por el propietario del objeto.

En lo que respecta al derecho de lectura de los datos contenidos en una tabla (utilización de la instrucción SELECT), es posible indicar aquellas columnas que el usuario puede visualizar. Por defecto, se trata de todas las columnas.

Los principales derechos de utilización de los objetos se refieren a las tablas, las vistas y los procedimientos almacenados, y corresponden a las sentencias:

- INSERT
- UPDATE
- DELETE
- SELECT
- EXECUTE (que solo se utiliza para los procedimientos almacenados).

Las instrucciones SELECT y UPDATE se pueden limitar a algunas columnas de la tabla o la vista. Sin embargo, es preferible no utilizar demasiado esta posibilidad, ya que da lugar a más trabajo administrativo de gestión de los derechos. Es preferible pasar por una vista o por un procedimiento almacenado para limitar el uso de la tabla.



Principio de funcionamiento de las autorizaciones

5.2.1 Autorizar

SQL Server Management Studio

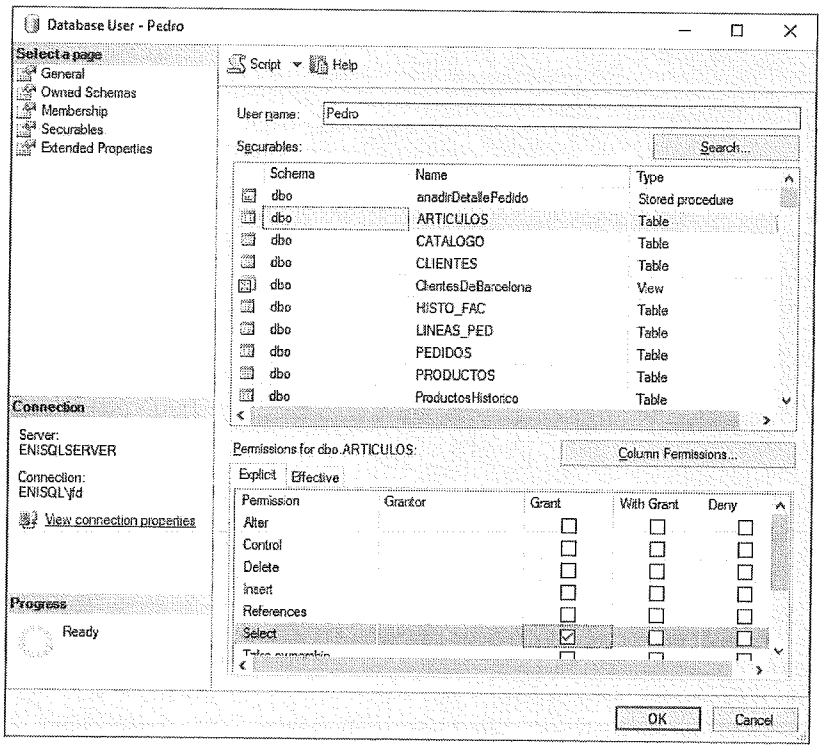
Los privilegios de utilización de los objetos se pueden gestionar en dos niveles con SQL Server Management Studio:

- A nivel usuario, lo que permite averiguar las posibilidades de trabajo que tiene cada usuario.
- A nivel de los objetos, para saber cuáles son los usuarios que pueden utilizar el objeto en cuestión y cómo pueden utilizarlo.

En cualquiera de los dos casos, las autorizaciones se gestionan mediante la ventana de propiedades.

Ejemplo

En el ejemplo siguiente, se va a asignar al usuario Pedro la posibilidad de ejecutar la instrucción SELECT sobre la tabla clientes del esquema dbo.



Transact SQL

```
GRANT {ALL [PRIVILEGES]|permiso[(columna,[,...])] [,...]}
    ON objeto[,...]
    TO usuario [,...]
    [WITH GRANT OPTION ]
```

ALL

Permite asignar todos los privilegios de utilización del objeto. Los privilegios asignados son siempre en función del tipo del objeto afectado por esta asignación del privilegio.

PRIVILEGES

La palabra clave se ha añadido para respetar la norma ANSI-92. No aporta ninguna modificación en lo que respecta a la utilización de la palabra clave ALL.

`permiso`

Permite especificar la operación o las operaciones que se asignarán a los usuarios.

`objeto`

Corresponde al nombre completo del objeto o de los objetos sobre los que se realiza la asignación del privilegio de utilización.

`usuario`

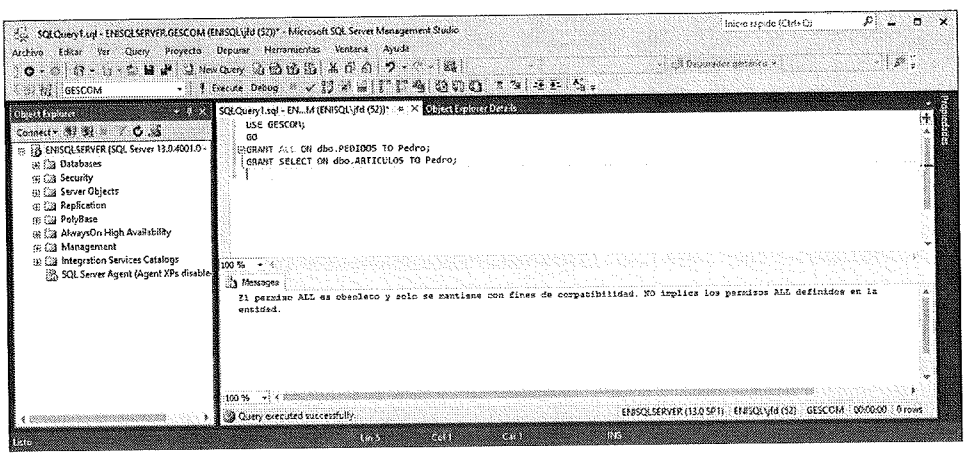
Corresponde al usuario o, más concretamente, a la entidad de seguridad que va a poder beneficiarse del privilegio o de los privilegios.

`WITH GRANT OPTION`

El privilegio se asigna con una opción de administración que autoriza al beneficiario del privilegio a asignar este mismo privilegio a otros usuarios de la base de datos.

Ejemplo

En el ejemplo siguiente, el usuario Pedro recibe todos los privilegios de utilización sobre la tabla de pedidos, así como la posibilidad de ejecutar consultas sobre la tabla de artículos.



5.2.2 Retirar

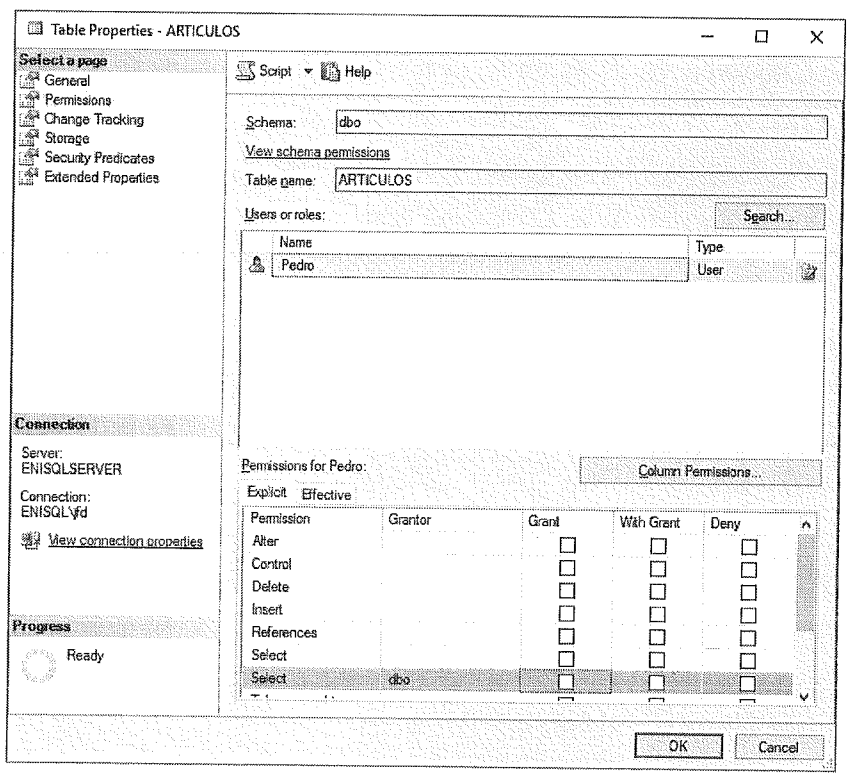
Cuando se ha asignado un permiso, es posible retirar esta asignación. No se puede retirar la utilización de un permiso a una entidad de seguridad si este permiso no ha sido previamente asignado a dicha entidad de seguridad.

SQL Server Management Studio

Por medio de la ventana de propiedades de la entidad de seguridad o del objeto, es posible retirar un permiso que ha sido asignado anteriormente.

Ejemplo

A partir de las propiedades de la tabla dbo.ARTICULOS, se va a retirar al usuario Pedro la posibilidad de efectuar sentencias de tipo SELECT sobre esta tabla.



Transact SQL

```
REVOKE [GRANT OPTION FOR ]
      {ALL [PRIVILEGES]|permiso[(columna,[,...])] [,...]}
       ON objeto [(columna [,...])]
      {FROM | TO} usuario [,...]
      [ CASCADE ]
```

`GRANT OPTION FOR`

Con esta opción, es posible retirar simplemente el privilegio de administración del privilegio.

`permiso`

Permite precisar el permiso o los permisos afectados por la acción de retirada de permisos. Igual que para la asignación, es posible indicar las columnas afectadas por la operación, aunque la gestión de este nivel de derecho es laboriosa.

`objeto`

Es necesario precisar el nombre completo de los objetos en la base de datos; es decir: nombreEsquema.nombreObjeto.

`FROM, TO`

Estos dos términos son sinónimos. Normalmente se utiliza TO al asignar un privilegio y FROM al eliminar un privilegio.

`usuario`

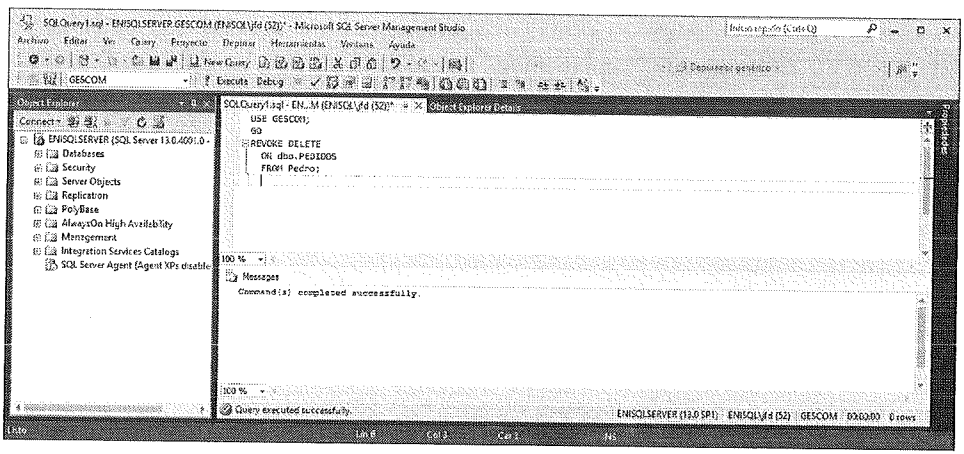
Se trata de la entidad de seguridad a la que se retira el privilegio. Lo más habitual es que esta entidad de seguridad sea un usuario, pero se puede tratar de un rol, por ejemplo.

`CASCADE`

Al retirar un permiso asignado con el privilegio de administración, esta opción permite retirarlo en cascada; es decir, efectuar la eliminación del permiso a todas las entidades de seguridad que lo han recibido a través de aquella a la que se le retira el permiso.

Ejemplo

En el ejemplo siguiente, al usuario Pedro se le retira la posibilidad de eliminar pedidos.



5.2.3 Prohibir

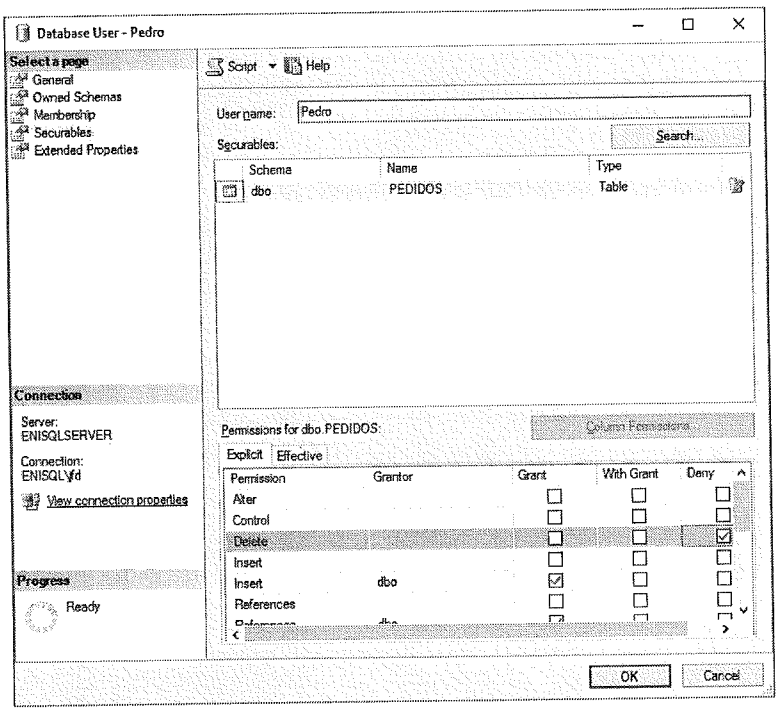
La prohibición de utilizar un permiso para usar un objeto es una instrucción más fuerte que la eliminación, porque puede aplicarse a una entidad de seguridad, incluso aunque esta última no haya recibido todavía el privilegio de utilización del objeto de manera directa o no.

SQL Server Management Studio

Como para la adición o cancelación, la prohibición se gestiona en las ventanas de propiedades, tanto de la entidad de seguridad como del objeto. La elección de una solución respecto a la otra depende del tipo de vista que se quiera adoptar y del tipo de operación que se quiera realizar. ¿Qué se desea, controlar mejor el uso de una tabla o controlar las acciones que puede realizar un usuario de base de datos?

Ejemplo

En el ejemplo siguiente, usando la ventana de propiedades de la tabla se va a prohibir al usuario Pedro la posibilidad de eliminar la información de la tabla dbo.PEDIDOS.



Transact SQL

```
DENY {ALL [PRIVILEGES]|permiso[(columna,[,...])] [,...]}
      ON object [(columna [,...])]
      TO usuario [,...]
     [CASCADE]
```

`columna`

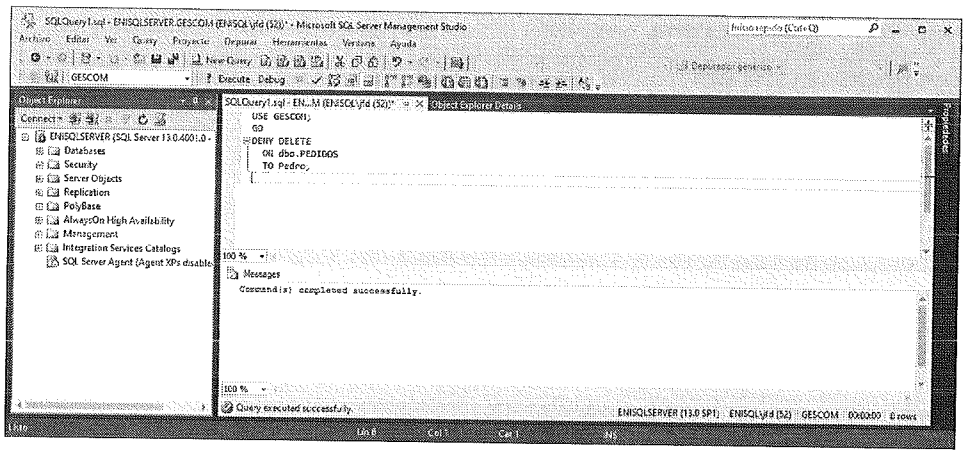
La prohibición a nivel de columna no tiene efecto sobre una autorización asignada a nivel de la tabla. Esta inconsistencia de seguridad se mantiene en SQL Server por razones de compatibilidad anterior.

`CASCADE`

La prohibición se transmite a las entidades de seguridad que han recibido el permiso de utilizar este privilegio a través de la entidad de seguridad a la que se prohíbe la utilización de este privilegio.

Ejemplo

En el ejemplo siguiente, se va a prohibir al usuario de la base de datos Pedro la posibilidad de eliminar la información de la tabla dbo.PEDIDOS.



5.3 Derechos a nivel de la base de datos

Los privilegios asignados a nivel de la base de datos ofrecen la posibilidad al usuario que los recibe de realizar las acciones autorizadas sobre el conjunto de la base de datos. Por ejemplo, el privilegio ALTER ANY USER permite a un usuario crear, eliminar y modificar cualquier cuenta de la base de datos.

A nivel de la base de datos, es posible asignar privilegios a un usuario pero también a un esquema, un assembly o un objeto service broker.

La gestión de estos privilegios utiliza las instrucciones Transact SQL GRANT, REVOKE y DENY, pero también se puede realizar de manera gráfica a partir de las propiedades de la base de datos.

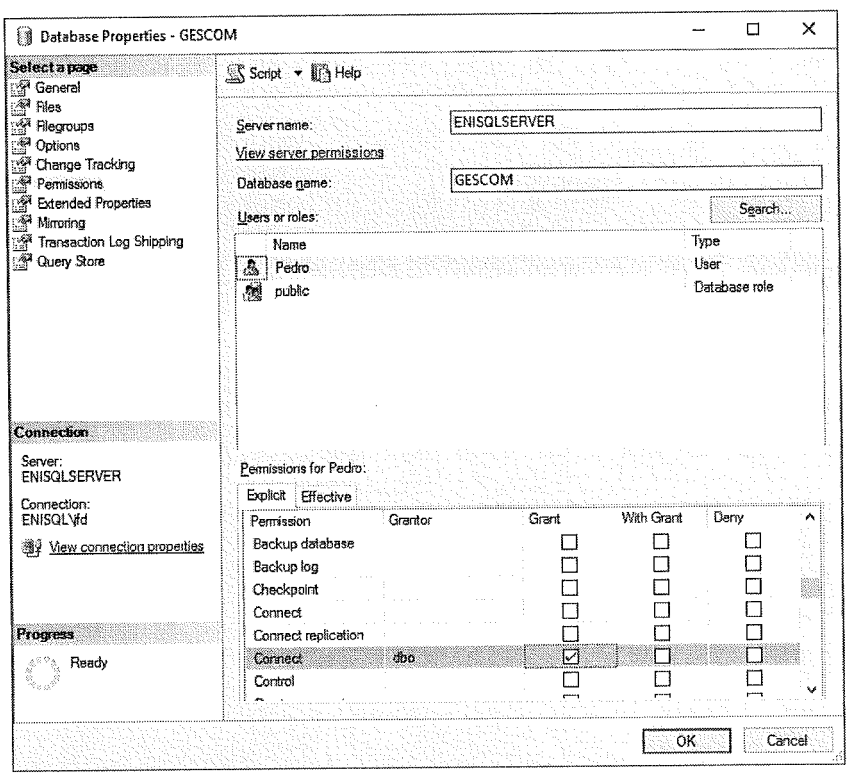
Observación

Solo se ilustran aquí los métodos para asignar privilegios. Las operaciones de eliminación y prohibición son muy parecidas a las ilustradas en puntos anteriores.

SQL Server Management Studio

Los derechos relativos a la utilización se pueden asignar de la manera siguiente desde SQL Server Management Studio:

- Desde el explorador de objetos, localizar el nodo que representa a la base de datos sobre la que se va a realizar la modificación del privilegio.
- Visualizar las propiedades de la base seleccionando **Properties** desde el menú contextual.



También es posible asignar estos permisos de manera gráfica modificando las propiedades de los usuarios de base de datos.

Transact SQL

La misma operación puede realizarse en Transact SQL con la instrucción GRANT.

```
GRANT permisoBaseDeDatos [ ,... ]
TO usuario [ ,...n ]
[ WITH GRANT OPTION ]
[ AS { grupo | rol } ]
```

`permisoBaseDeDatos`

El permiso o los permisos de base de datos asignados.

`Usuario`

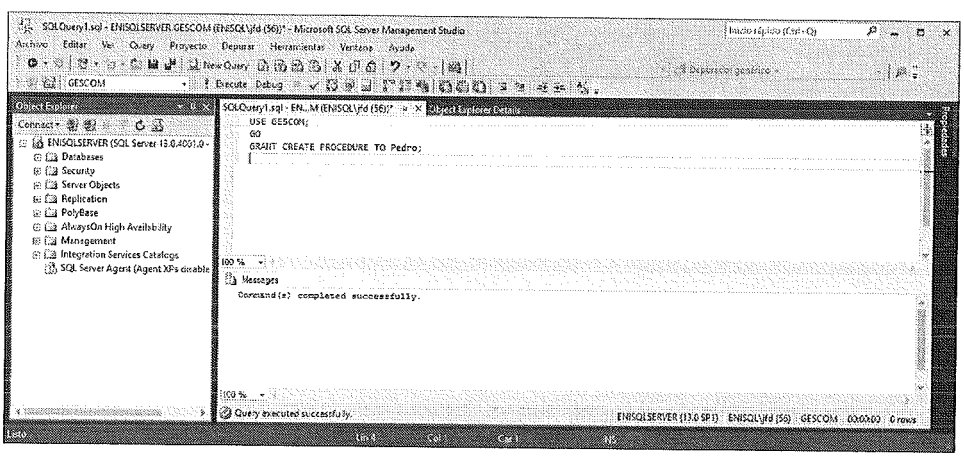
Cuenta de usuario de base de datos que recibe el privilegio.

`AS {grupo|rol}`

Contexto de seguridad utilizado para poder asignar el privilegio.

Ejemplo

En el ejemplo siguiente, se asigna el privilegio CREATE PROCEDURE al usuario Pedro.



Los privilegios que es posible asignar a este nivel son:

ALTER	CREATE DATABASE
ALTER ANY APPLICATION ROLE	CREATE DATABASE DDL EVENT NOTIFICATION
ALTER ANY ASSEMBLY	CREATE DEFAULT
ALTER ANY ASYMMETRIC KEY	CREATE FULLTEXT CATALOG
ALTER ANY CERTIFICATE	CREATE FUNCTION
ALTER ANY CONTRACT	CREATE MESSAGE TYPE
ALTER ANY DATABASE DDL TRIGGER	CREATE PROCEDURE
ALTER ANY DATABASE EVENT NOTIFICATION	CREATE QUEUE
ALTER ANY DATASPACE	CREATE REMOTE SERVICE BINDING
ALTER ANY FULLTEXT CATALOG	CREATE ROLE
ALTER ANY MESSAGE TYPE	CREATE ROUTE
ALTER ANY REMOTE SERVICE BINDING	CREATE RULE
ALTER ANY ROLE	CREATE SCHEMA
ALTER ANY ROUTE	CREATE SERVICE
ALTER ANY SCHEMA	CREATE SYMMETRIC KEY
ALTER ANY SERVICE	CREATE SYNONYM
ALTER ANY SYMMETRIC KEY	CREATE TABLE
ALTER ANY USER	CREATE TYPE
AUTHENTICATE	CREATE VIEW
BACKUP DATABASE	CREATE XML SCHEMA COLLECTION
BACKUP LOG	DELETE
CHECKPOINT	EXECUTE
CONNECT	INSERT
CONNECT REPLICATION	REFERENCES
CONTROL	SELECT
CREATE AGGREGATE	SHOWPLAN

CREATE ASSEMBLY	SUBSCRIBE QUERY NOTIFICATIONS
CREATE ASYMMETRIC KEY	TAKE OWNERSHIP
CREATE CERTIFICATE	UPDATE
CREATE CONTRACT	VIEW DATABASE STATE
ALTER ANY DATABASE EVENT SESSION	VIEW DEFINITION

5.4 Derechos a nivel del servidor

SQL Server permite asignar privilegios a nivel del servidor. Estos privilegios no se asignan a un usuario de base de datos, sino a una conexión.

Como para los derechos de utilización de los objetos y de las instrucciones, es posible asignar estos privilegios con una opción de administración. Así, la conexión que tiene este derecho puede asignar este mismo derecho a una o varias conexiones adicionales.

Los diferentes derechos que es posible asignar a nivel del servidor son:

ADMINISTER BULK OPERATIONS	CONNECT SQL
ALTER ANY CONNECTION	CONTROL SERVER
ALTER ANY CREDENTIAL	CREATE ANY DATABASE
ALTER ANY DATABASE	CREATE DDL EVENT NOTIFICATION
ALTER ANY ENDPOINT	CREATE ENDPOINT
ALTER ANY EVENT NOTIFICATION	CREATE TRACE EVENT NOTIFICATION
ALTER ANY LINKED SERVER	EXTERNAL ACCESS ASSEMBLY
ALTER ANY LOGIN	IMPERSONATE ANY LOGIN
ALTER RESOURCES	SELECT ALL USERS SECURABLES
ALTER SERVER STATE	SHUTDOWN
ALTER SETTINGS	UNSAFE ASSEMBLY
ALTER TRACE	VIEW ANY DATABASE
AUTHENTICATE SERVER	VIEW ANY DEFINITION
CONNECT ANY DATABASE	VIEW SERVER STATE

Observación

Toda la información relativa a los privilegios asignados a nivel del servidor están accesibles a través de la vista ***sys.server_permissions.***

También es posible asignar derechos de utilización de objetos definidos a nivel del servidor tales como las terminaciones http o http endpoint.

Observación

Para poder asignar estos privilegios, es necesario trabajar sobre la base Master.

5.5 Consultar las vistas de sistema

Consultando las vistas de sistema del diccionario, y más concretamente las que se ocupan de la seguridad, es posible ver las diferentes autorizaciones y prohibiciones relativas a las diferentes entidades de seguridad.

Las principales vistas son:

- sys.database_permissions, que permite listar las diferentes autorizaciones de utilización de los privilegios.
- sys.database_principals, que permite listar las diferentes cuentas de seguridad.

sys.database_principals

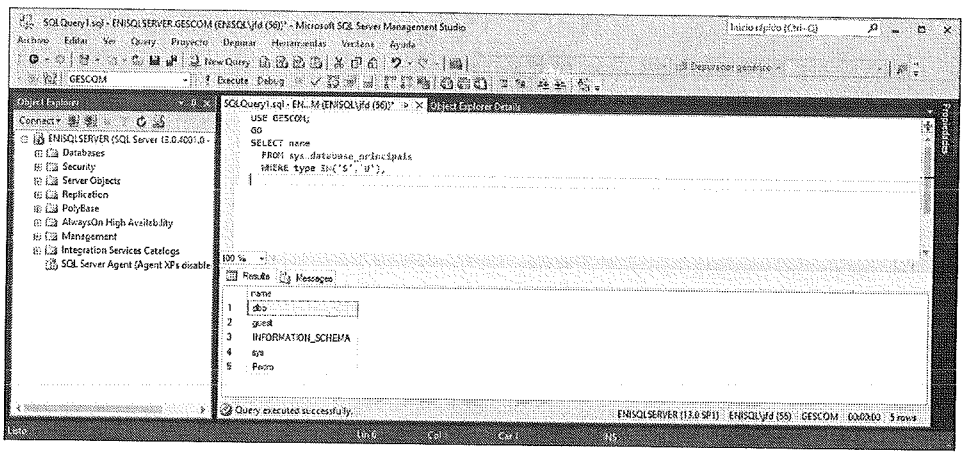
Esta vista lista todas las entidades de seguridad definidas localmente en la base de datos. Las principales columnas de esta vista son:

- **name**: nombre de la entidad de seguridad. Este nombre es único dentro de la base de datos.
- **principal_id**: identificador numérico que permite identificar de manera única una entidad de seguridad en la base de datos.
- **type**: código relativo al tipo del contexto de seguridad: usuario de SQL, usuario de Windows, rol, grupo de Windows...
- **type_desc**: descripción para comprender el tipo mencionado en la columna anterior.
- **default_schema_name**: nombre del esquema asociado por defecto a la entidad de seguridad.
- **create_date**: fecha de creación de la entidad de seguridad.
- **modify_date**: fecha de la última modificación sobre la entidad de seguridad.
- **owning_principal_id**: identificador del contexto de seguridad que tiene la cuenta en cuestión. Todos, a excepción de los roles de base de datos, deben pertenecer a **dbo**.

- **sid**: o Security Identifier. En este campo se informa de si la entidad de seguridad está asociada a un externo.
- **is_fixed_role**: este atributo es 1 cuando la entidad de seguridad se asocia a un rol fijo predefinido sobre el servidor.

Ejemplo

La consulta siguiente permite ver las cuentas de usuarios definidos en la base GESCOM.



sys.database_permissions

Esta vista permite obtener la información de los diferentes permisos asignados a la base de datos. Existe un registro por cada permiso asignado a cada entidad de seguridad. En caso de administración de permisos a nivel de columna, existe un registro para cada columna afectada por el permiso.

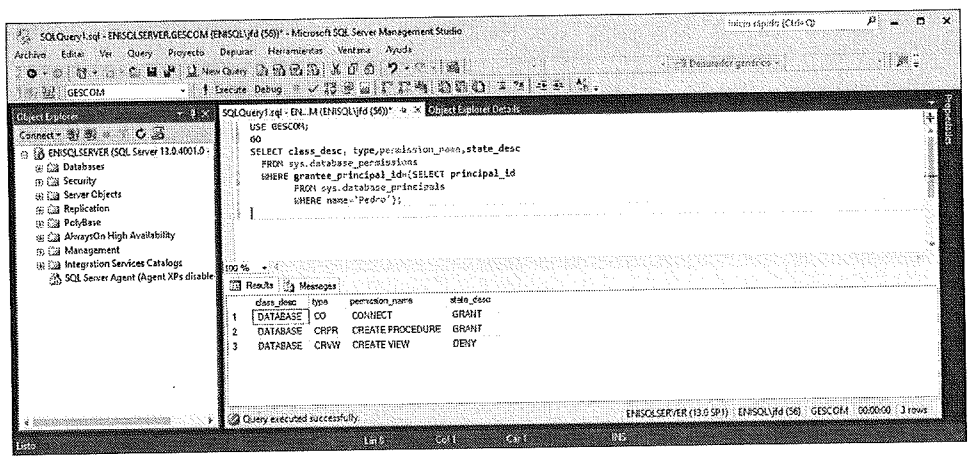
Las columnas más importantes de esta vista son:

- **class**: código de la clasificación del permiso, es decir, si se trata de un permiso a nivel de base de datos que se aplica al uso de un objeto o columna, etc.
- **class_desc**: descripción de la clasificación del permiso.
- **mayor_id**: identificador del objeto sobre el que se asigna el permiso.
- **minor_id**: identificador de la columna sobre la que se asigna el permiso.
- **grantee_principal_id**: identificador del contexto de seguridad al que concierne el permiso.
- **grantor_principal_id**: identificador del contexto de seguridad en el origen del permiso.

- **type**: tipo del permiso.
- **permission_name**: nombre del privilegio al que concierne el permiso.
- **state**: código relativo al estado del permiso.
- **state_desc**: descripción del estado actual del permiso: GRANT, REVOKE, DENY o bien GRANT_WITH_GRANT_OPTION.

Ejemplo

Con ayuda de la siguiente consulta es fácil conocer los permisos que tiene el usuario de base de datos Pedro.



El resto de las vistas de sistema que se pueden consultar para obtener los datos de los diferentes tipos de permisos asignados son:

- **sys.database_role_members**: lista de beneficiarios (principal_id) de un rol de base de datos.
- **sys.master_key_passwords**: información de cada clave principal de la base de datos creada con el procedimiento sp_control_dbmasterkey_password.

Observación

Como la gestión de permisos y de los diferentes niveles es relativamente compleja, Microsoft proporciona un póster (es decir un documento en formato pdf) disponible en GitHub: http://go.microsoft.com/fwlink/?LinkId=229142 o https://github.com/Microsoft/sql-server-samples/blob/master/samples/features/security/permissions-posters/Permissions_Poster_SQL_Server_2016_and_SQLDB.pdf. Este documento se encuentra también disponible entre los ficheros en descarga asociados a este libro.

6. Contexto de ejecución

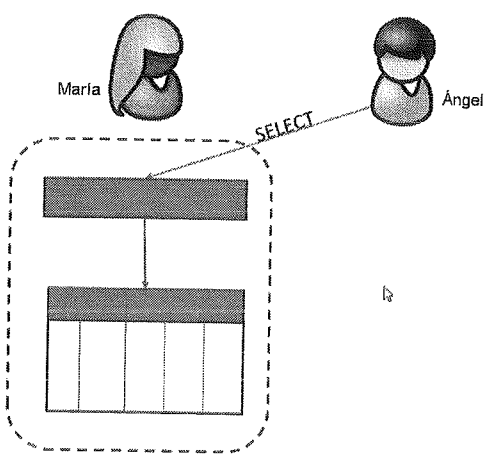
El contexto de ejecución está directamente relacionado con la conexión y el usuario de bases de datos asociado. La conexión de ejecución permite establecer la lista de posibles acciones y aquellas que no se pueden realizar. Esta lista se crea a partir de los permisos asignados a los usuarios directamente o a través de los roles.

En algunos casos puede ser necesario y deseable modificar el contexto de ejecución para aprovechar los permisos extendidos, pero solo en el ámbito de un script, procedimiento o función.

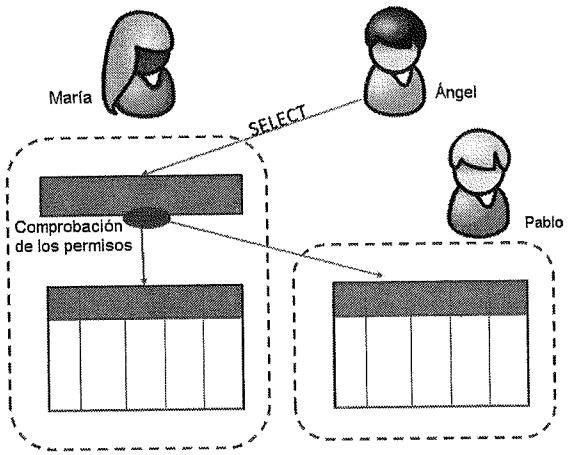
Asociada al contexto de ejecución, es necesario entender bien la noción de encadenamiento de propiedades en SQL Server.

En primer lugar, cuando un usuario accede a los objetos de los que es propietario, esto no representa ningún problema porque no hay ruptura de encadenamiento de propiedades. Este caso es relativamente raro, ya que los usuarios de las bases de datos que crean objetos raramente los utilizan a diario.

Otro caso sencillo se produce cuando el usuario ha recibido de su propietario permisos para usar un objeto. Por ejemplo, el usuario Ángel recibe de María el permiso para usar (SELECT) una vista de María. La consulta SELECT de definición de la vista hace referencia a una tabla también de María. Ángel no tiene ningún permiso sobre esta tabla y podrá utilizar esta vista sin problemas, ya que los dos objetos (vista y tabla) tienen el mismo propietario.



Si la vista de María que utiliza Ángel accede a una tabla que no es propiedad de María, se comprueban los permisos asignados a Ángel para saber si este puede ejecutar la consulta.



EXECUTE AS

Con esta instrucción se puede solicitar la conexión a la base de datos usando una conexión diferente a la actual. Esta instrucción se puede ejecutar de manera autónoma en un script Transact SQL o como una cláusula durante la creación de un procedimiento, función o trigger.

Para los procedimientos, funciones o triggers, la cláusula EXECUTE AS ofrece mucha flexibilidad en términos de programación y permiten a un usuario realizar acciones para las que no tiene permisos. Para el desarrollador, el cambio de contexto de ejecución también permite garantizar que no se impedirá la correcta ejecución del código debido a un problema de permisos de acceso a los datos.

Por ejemplo, durante la ejecución de un script Transact SQL, la instrucción EXECUTE AS permite realizar de manera puntual operaciones que necesiten permisos extendidos, mientras que el resto del script no lo necesita.

El contexto de ejecución elegido de esta manera debe ser lo más cerrado posible para permitir simplemente la instrucción especificada. Por ejemplo, no es conveniente conectarse como propietario de la base de datos si la operación solicitada es simplemente la creación de una tabla. Ejecutando una consulta con permisos más importantes se está abriendo una brecha de seguridad.

Por lo tanto, es posible especificar el nombre de una conexión (LOGIN) o el de un usuario de la base de datos actual (USER). La opción CALLER solo se usa en la ejecución de un procedimiento o módulo, para especificar que los permisos que se deben utilizar son los asignados al contexto que ejecuta el módulo.

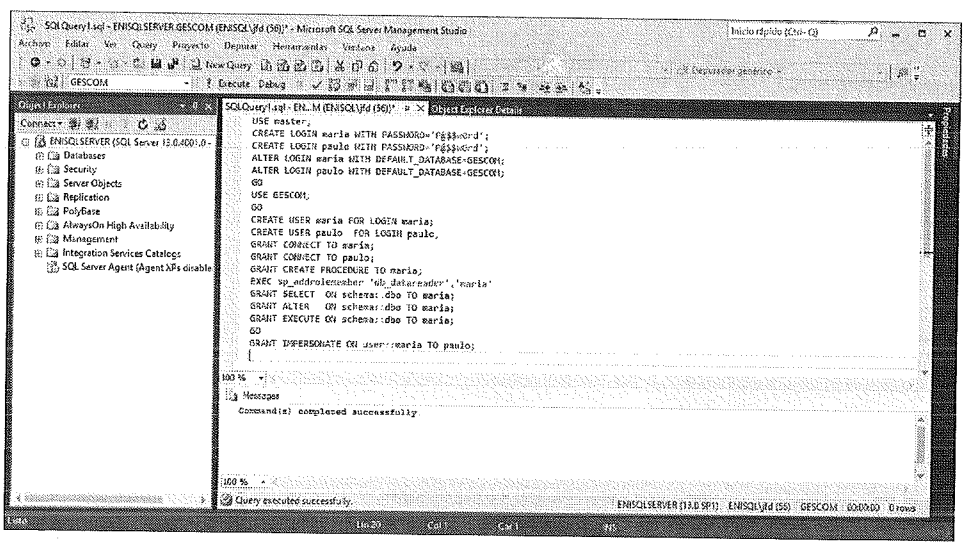
Si se especifica la opción WITH NO REVERT durante la ejecución de la instrucción EXECUTE AS, no es posible eliminar el nuevo contexto de ejecución usando una instrucción REVERT o EXECUTE AS. Es necesario desconectarse.

Sintaxis

```
EXECUTE AS {LOGIN|USER} = 'identificador' |CALLER;
EXECUTE AS {LOGIN|USER} = 'identificador' |CALLER WITH NO REVERT;
```

Ejemplo

En esta primera parte, el contexto se prepara definiendo los componentes de los usuarios Maria y Paulo y se asignan los derechos de impersonalización a Paulo sobre el esquema de Maria.

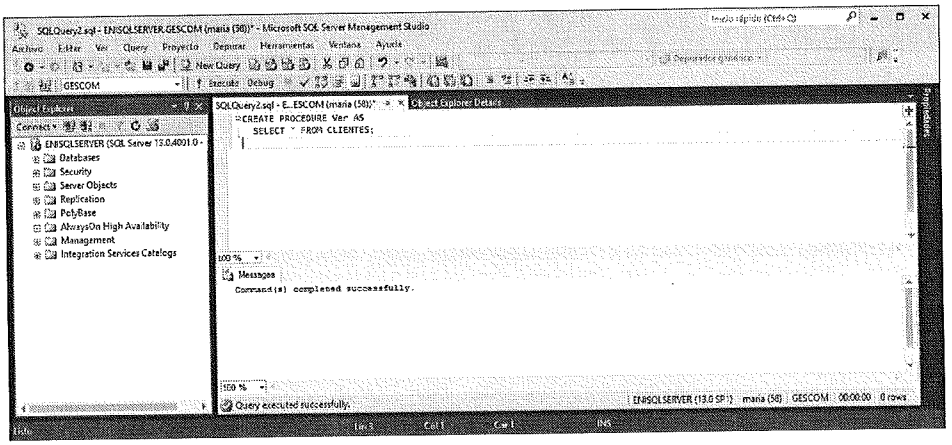


El usuario Maria define en su esquema el procedimiento **Ver**, que permite visualizar el contenido de la tabla CLIENTES.

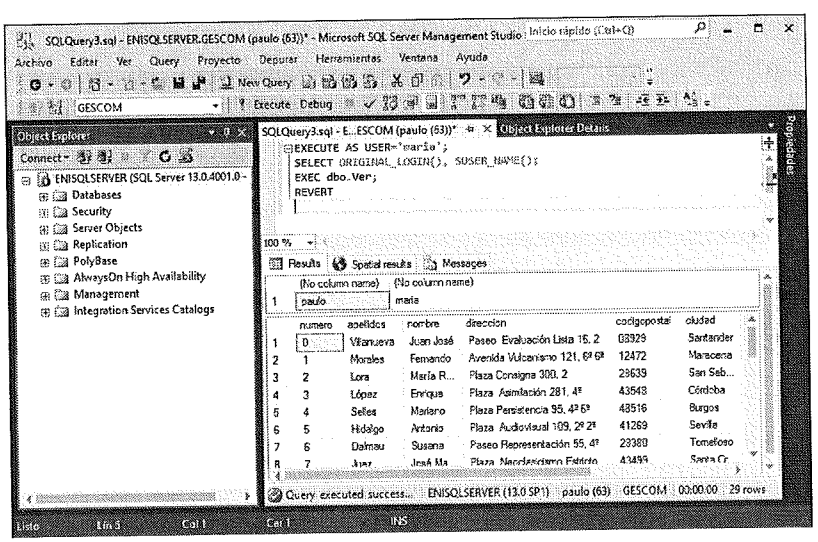
Antes de ejecutar el siguiente script, debemos conectarnos con los identificadores del usuario Maria.

Observación

El nombre de la conexión utilizada para un script se indica en la barra de estado de la consulta.



Por último, el usuario Paulo intenta usar el procedimiento **Ver** de Maria. Al inicio, Paulo solicita asumir la identidad de Maria con la instrucción EXECUTE AS. Se verifican las identidades con las funciones ORIGINAL_LOGIN y SUSER_NAME, y después el usuario Paulo puede ejecutar sin problemas el procedimiento **Ver** que ha creado Maria.



Otra versión del comando EXECUTE_AS permite indicar el contexto de ejecución de un procedimiento, función o trigger de base de datos. El ámbito de este cambio de contexto de ejecución está limitado a la ejecución del módulo.

Sintaxis

```
Procedimiento o función o trigger
EXECUTE AS {CALLER|SELF|OWNER|'nombreUsuario'}
...
```

CALLER

Se tiene en cuenta el contexto del que ejecuta el módulo.

SELF

Se tiene en cuenta el contexto del usuario que ha creado o modificado el módulo.

OWNER

Se tiene en cuenta el contexto del propietario del módulo.

'nombreUsuario'

Permite indicar el contexto de usuario que se debe utilizar.

SETUSER

Al contrario que la instrucción EXECUTE AS, que no modifica la conexión inicial del servidor, la instrucción SETUSER permite cambiar de conexión en un script Transact SQL. Es decir, se cierra el contexto de ejecución actual y se abre uno nuevo. No es posible volver al primer contexto de ejecución.

Original_Login

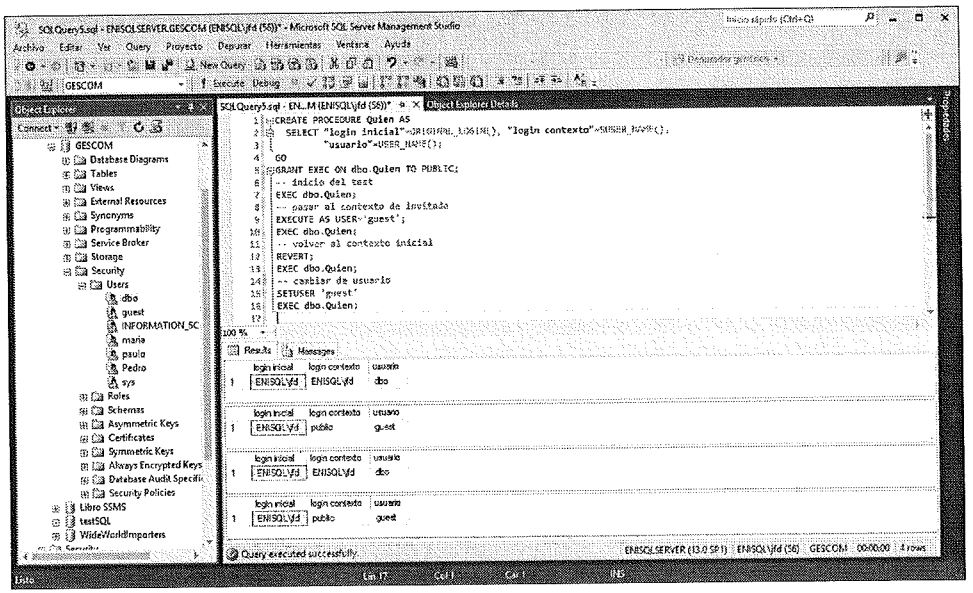
Esta función permite determinar el nombre exacto de la conexión que se utiliza inicialmente para conectarse al servidor. Conocer este nombre solo es interesante cuando el contexto de ejecución es diferente de la conexión inicial. El contexto de ejecución se puede modificar con la instrucción EXECUTE AS.

REVERT

Después de un cambio de contexto de ejecución con la instrucción EXECUTE AS, la instrucción REVERT permite volver al contexto de ejecución que había cuando se cambió con EXECUTE AS.

Ejemplo

En el siguiente ejemplo, se define el procedimiento **dbo.quien** para mostrar toda la información de la conexión inicial al servidor, de la conexión que se usa para ejecutar la consulta y de la cuenta de usuario que se utiliza:



7. Los roles

Los roles son los conjuntos de permisos. Estos conjuntos existen a tres niveles distintos: servidor, base de datos y aplicación. Los roles permiten agrupar los derechos y gestionar más fácilmente los diferentes usuarios y las conexiones. Siempre es preferible asignar los derechos a los roles y posteriormente asignar los roles a los usuarios. Con una estructura como esta, la adición y la modificación de permisos o de usuarios son más sencillas.

Es posible definir un rol como un conjunto identificado de permisos. Para facilitar la gestión de los permisos, SQL Server ofrece los roles predefinidos, también llamados fijos, ya que no es posible añadir o eliminar privilegios en estos roles.

Estos roles fijos se definen en dos niveles:

- Servidor.
- Base de datos.

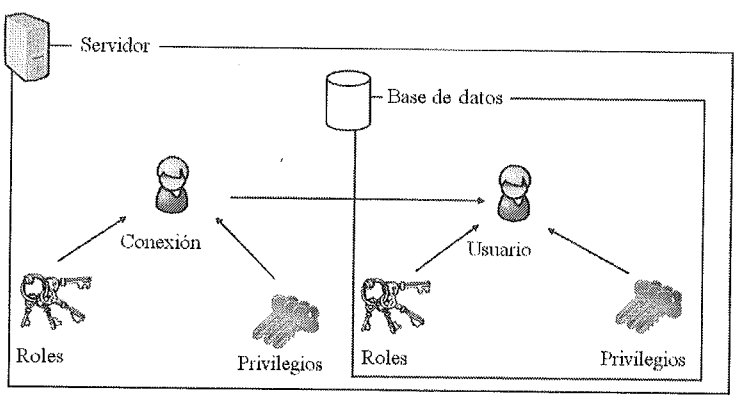
Además de estos roles fijos, es posible gestionar otros roles. Es conveniente establecer un nombre único para definir un rol y posteriormente asignar uno o varios permisos respetando un procedimiento en todo punto similar al utilizado para asignar los permisos a los usuarios. Estos roles se pueden definir en tres niveles:

- Servidor.
- Base de datos.
- Aplicación.

Los roles permiten una gestión simplificada de los privilegios, ya que también es posible definir los perfiles tipo de privilegios y posteriormente asignar a cada usuario de base de datos uno o varios perfiles tipo con objeto de darle todas las autorizaciones que necesita para trabajar en la base de datos.

El usuario dispone al final del conjunto de permisos que se le asignan:

- Directamente a la conexión utilizada.
- Indirectamente por medio de un rol fijo de servidor asignado a la conexión.
- Indirectamente por medio de los roles asignados al usuario de base de datos.
- Directamente al usuario de base de datos.



Como complemento de estos diferentes roles, existe el rol **public**. Este rol es adicional, ya que todos los usuarios reciben el rol public y no pueden ignorarlo. Este rol también es particular, ya que se le pueden asignar, retirar o prohibir permisos. Todas las modificaciones hechas a nivel de los permisos sobre el rol public son válidas para todos los usuarios. Por lo tanto, no es recomendable trabajar con este rol, aunque, en algunos casos, añade flexibilidad a la gestión de los permisos.

7.1 Roles de servidor

Son los roles predefinidos que dan a las conexiones un cierto número de funcionalidades. Su uso facilita la gestión en caso de que haya muchos usuarios.

7.1.1 Los roles predeterminados

Hay 8 roles de servidor:

sysadmin

Ejecuta cualquier operación sobre el servidor. Es el administrador del servidor.

serveradmin

Permite configurar los parámetros a nivel del servidor.

setupadmin

Permite añadir/eliminar los servidores asociados y ejecutar algunos procedimientos de sistema almacenados como **sp_serveroptions**.

securityadmin

Permite gestionar las conexiones de acceso al servidor.

processadmin

Permite gestionar los tratamientos que se ejecutan sobre SQL Server.

dbcreator

Permite crear y modificar las bases de datos.

diskadmin

Permite gestionar los archivos sobre el disco.

bulkadmin

Puede ejecutar la instrucción BULK INSERT para insertar los datos por bloques. El interés de este tipo de inserción reside en el hecho de que la operación no se realiza en modo conectado y los tiempos de inserción son mucho más rápidos.

Observación

*Solo los miembros del rol **sysadmin** pueden asignar un rol de servidor a una conexión.*

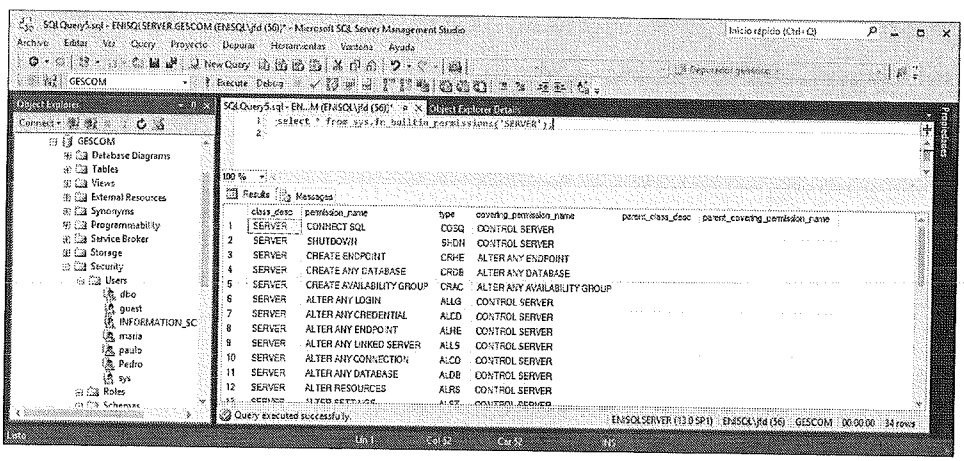
El rol **public** recibe el permiso VIEW ANY DATABASE. De esta manera, los usuarios que se conecten a SQL pueden listar las bases definidas sobre la instancia. Solo podrán acceder si una cuenta de usuario de base de datos está mapeada a su conexión o si se define el usuario **guest** (invitado) a nivel de la base de datos.

7.1.2 Crear un rol de servidor

SQL Server ofrece la posibilidad de crear sus propios roles de servidor. Estos roles solo pueden contener los permisos disponibles a nivel de servidor y se van a asignar a las conexiones de la misma manera que los roles predeterminados.

Antes de crear un rol, es necesario identificar los permisos que deseamos asignarle. Para identificar estos permisos, lo más sencillo es consultar la función sys.fn_builtin_permissions, pasándole como argumento SERVER. Esta consulta se ilustra en el siguiente ejemplo. Tenga cuidado con subestimar los roles predeterminados, ya que permiten dar respuesta a la mayor parte de los casos de uso.

Ejemplo



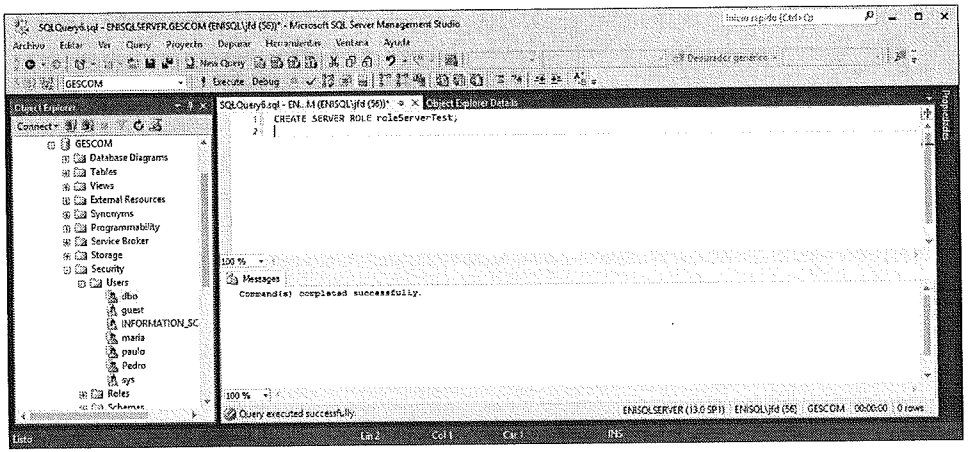
El rol se crea usando la instrucción CREATE SERVER ROLE. Como sucede con todos los objetos SQL Server, las instrucciones ALTER y DROP permiten modificar y eliminar los roles de servidor.

Sintaxis

```
CREATE SERVER ROLE nombreRoleServidor;
```

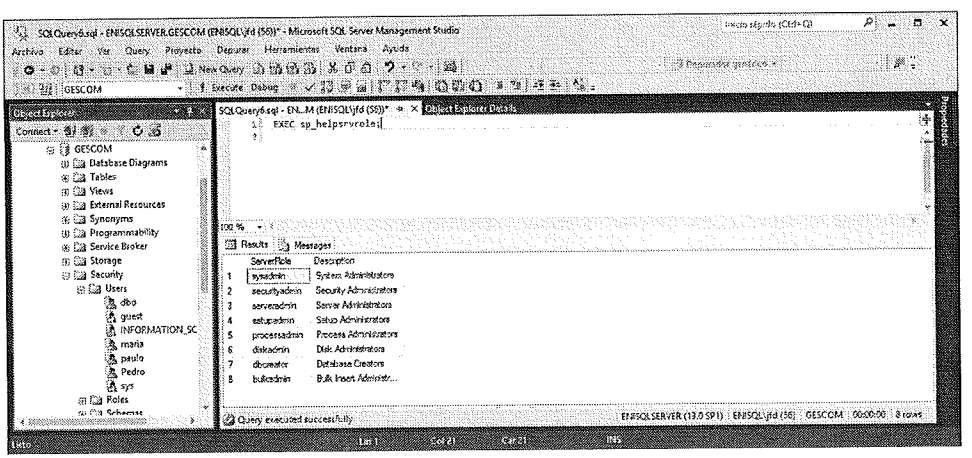
Ejemplo

Creación de un rol a nivel de servidor.



7.1.3 Asignar los roles

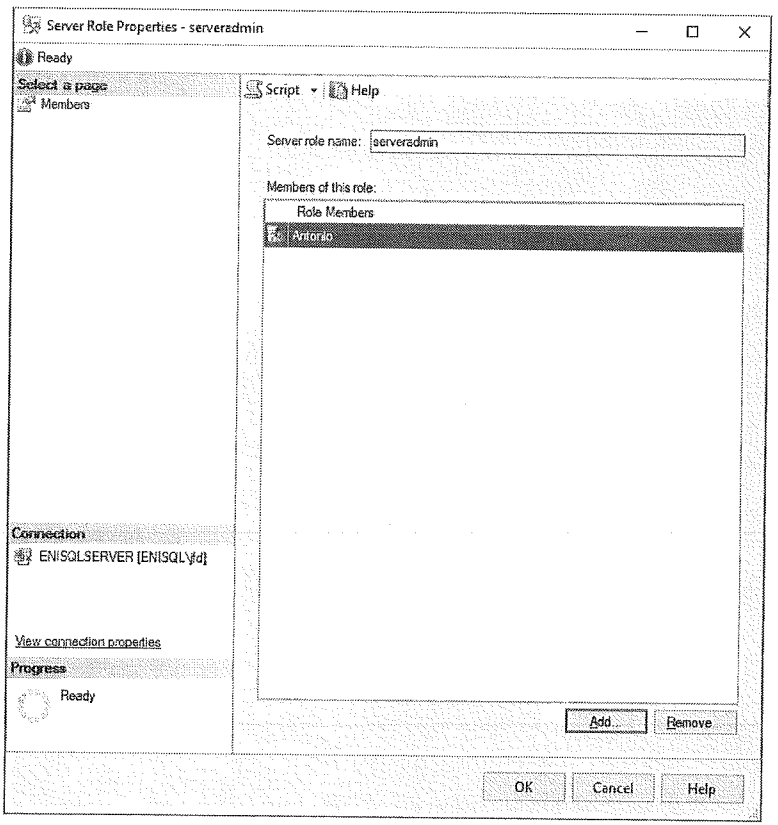
Los procedimientos **sp_helpsrvrole** y **sp_helpsrvrolemember** permiten obtener toda la información necesaria de los diferentes roles fijos del servidor y su asignación.



Ejecución de sp_helpsrvrole para averiguar los roles de servidores fijos

SQL Server Management Studio

La gestión de los roles fijos de servidor, y más concretamente la gestión de las conexiones en el interior del rol, se realiza desde la ventana de propiedades del rol.



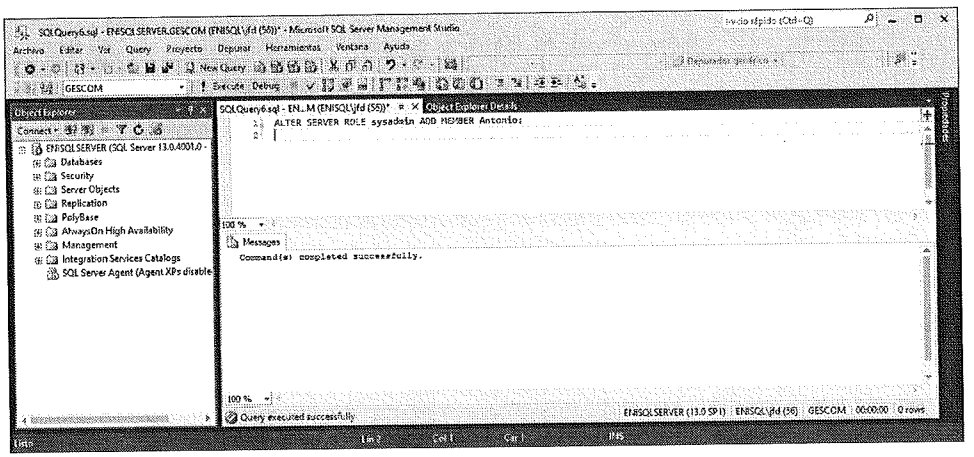
Transact SQL

Para añadir una conexión a un rol de servidor, hay que utilizar la instrucción ALTER ROLE. El hecho de que una conexión se beneficie de un rol o de que se elimine este beneficio, se considera como una modificación.

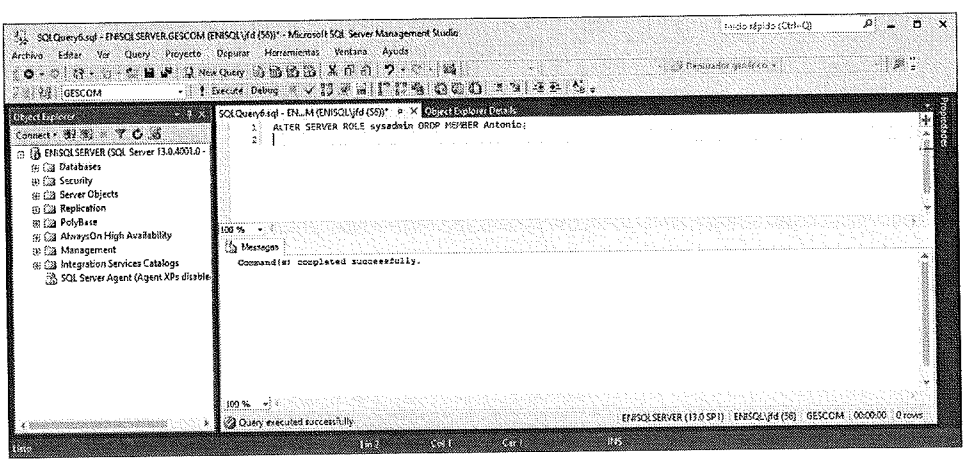
Sintaxis

```
ALTER SERVER ROLE nombreRolServidor ADD MEMBER nombreConexión;
ALTER SERVER ROLE nombreRolServidor DROP MEMBER nombreConexión;
```

Ejemplo



En este segundo ejemplo, la conexión Antonio no se va a beneficiar más del rol sysadmin.



Observación

*Los procedimientos almacenados **sp_addsrvrolemember** y **sp_dropsrvrolemember** se mantienen por razones de compatibilidad, pero ya no se utilizan porque Microsoft no garantiza su presencia en las próximas versiones de SQL Server.*

7.2 Roles de base de datos

Los roles de bases de datos permiten siempre agrupar las diferentes autorizaciones o denegaciones de instrucciones o de objetos para, de esta manera, facilitar la gestión de los derechos. Existen roles predefinidos y es posible definir roles propios.

Las modificaciones sobre los roles son dinámicas y los usuarios no tienen necesidad de desconectarse/volverse a conectar de/a la base para disfrutar de los cambios.

Observación

Los roles predefinidos no pueden eliminarse.

Solo los miembros de los roles fijos de base de datos **db_owner** y **db_securityadmin** pueden gestionar la pertenencia de los usuarios a un rol fijo o no de la base de datos.

7.2.1 El rol public

Se trata de un rol predefinido particular, ya que todos los usuarios de la base tienen este rol. Así, cuando se asigna, se elimina o se prohíbe una autorización de instrucción a **public**, instantáneamente todos los usuarios de la base se benefician de las modificaciones.

El rol **public** contiene todas las autorizaciones por defecto que tienen los usuarios de la base de datos.

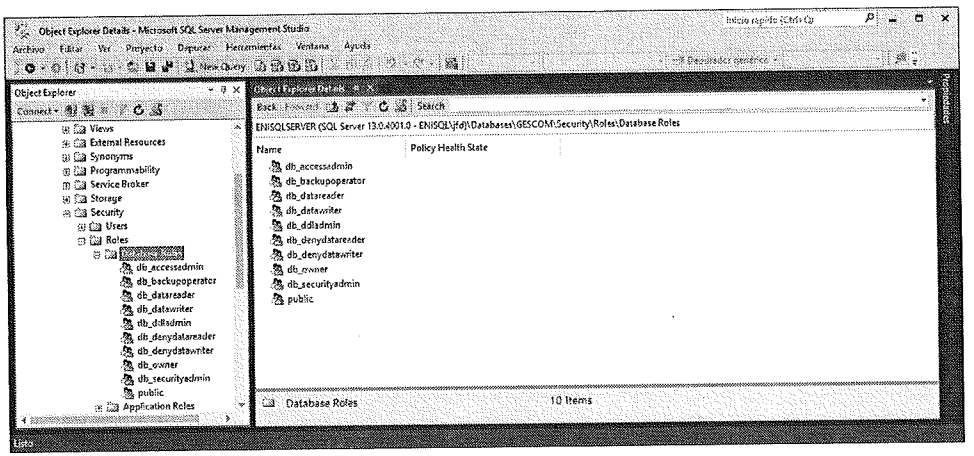
No es posible asignar miembros a este rol, ya que todo el mundo lo tiene implícitamente. Este rol está presente en todas las bases del servidor, tanto en las de sistema como en las de usuario.

Observación

Al crearse, un usuario no dispone de ninguna autorización, salvo aquellas asignadas a public.

7.2.2 Los roles predefinidos

Aparte del rol **public**, existen roles predefinidos en la base de datos.



db_owner

Conjunto de derechos equivalentes a ser el propietario de la base. Este rol contiene todas las actividades posibles en los otros roles de la base de datos, así como la posibilidad de ejecutar algunas tareas de mantenimiento y de configuración de la base. Todos los miembros de este grupo van a crear objetos cuyo propietario es **dbo**. Se trata del propietario por defecto de todos los objetos y no es necesario especificarlo para manipular sus objetos.

db_accessadmin

Permite añadir o eliminar usuarios en la base de datos. Estos usuarios corresponden a conexiones SQL Server o a grupos de usuarios de Windows.

db_datareader

Permite consultar (SELECT) el contenido de todas las tablas de la base de datos.

db_datawriter

Permite añadir (INSERT), modificar (UPDATE) o eliminar (DELETE) datos en todas las tablas de usuario de la base de datos.

db_ddladmin

Permite añadir (CREATE), modificar (ALTER) o eliminar (DELETE) objetos de la base de datos.

db_securityadmin

Permite gestionar los roles, los miembros de los roles y las autorizaciones sobre las instrucciones y los objetos de la base de datos.

db_backupoperator

Permite efectuar una copia de seguridad de la base de datos.

db_denydatareader

Prohíbe la visualización de los datos de la base.

db_denydatawriter

Prohíbe la modificación de los datos contenidos en la base.

Observación

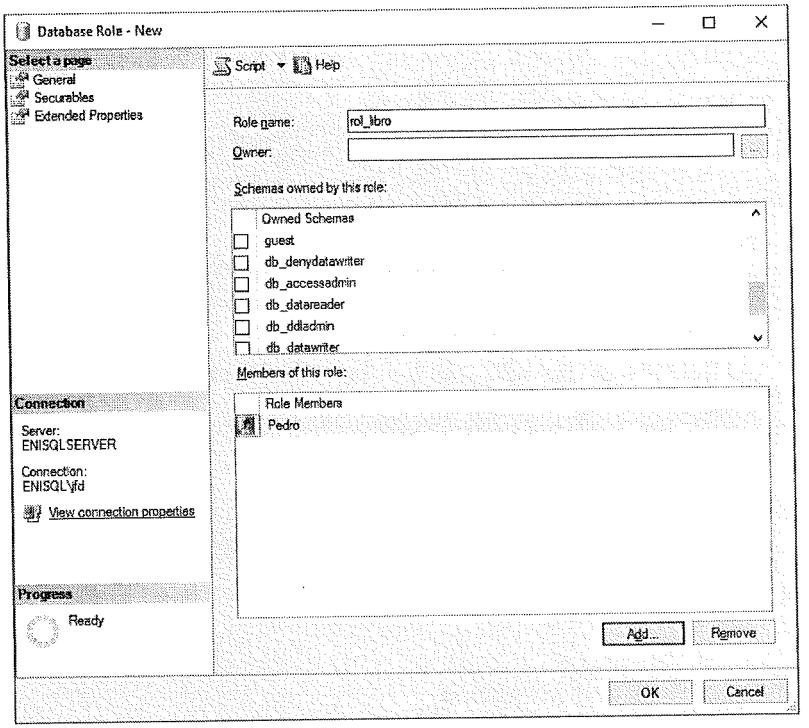
Todos los roles predefinidos están presentes en todas las bases de sistema.

7.2.3 Los roles de base de datos definidos por los usuarios

Es posible definir los propios roles con el objetivo de facilitar la administración de los derechos en el interior de la base.

El rol SQL Server se va a establecer cuando varios usuarios de Windows deseen efectuar las mismas operaciones en la base de datos y no exista el grupo Windows correspondiente, o cuando los usuarios autentificados por SQL Server y los autentificados por Windows deban compartir los mismos derechos.

Para los roles definidos a nivel de base de datos, es posible saber qué usuarios se benefician de ellos, viendo la ventana de propiedades del rol desde SQL Server Management Studio.



Los roles se pueden asignar directamente a un usuario o a otro rol. Sin embargo, la superposición repetida de roles puede afectar al rendimiento. Además, no es posible crear roles recursivos.

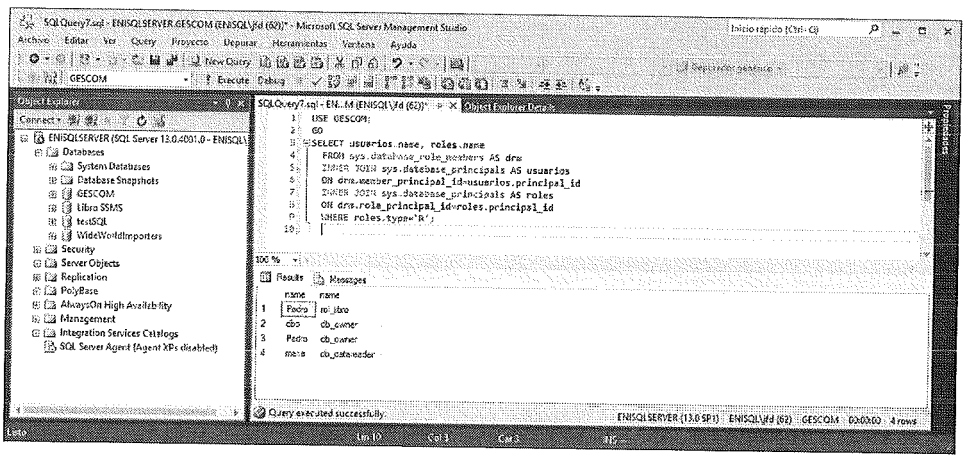
Observación

El uso de roles puede parecer pesado algunas veces en el momento de la creación, pero facilita enormemente las evoluciones que puedan tener lugar (modificación de las autorizaciones, de los usuarios).

Observación

Para definir un rol es necesario ser miembro del rol ***sysadmin*** *o miembro de* ***db_owner*** *o* ***db_securityadmin****.*

La consulta de las tablas de sistema **sys.database_principals** y **sys.database_role_member** permite conocer la lista de los usuarios que pertenecen a cada rol de base de datos. El ejemplo siguiente ilustra este aspecto:



Como con el conjunto de operaciones de administración, es posible gestionar los roles de dos maneras diferentes.

7.2.4 Creación de un rol de base de datos

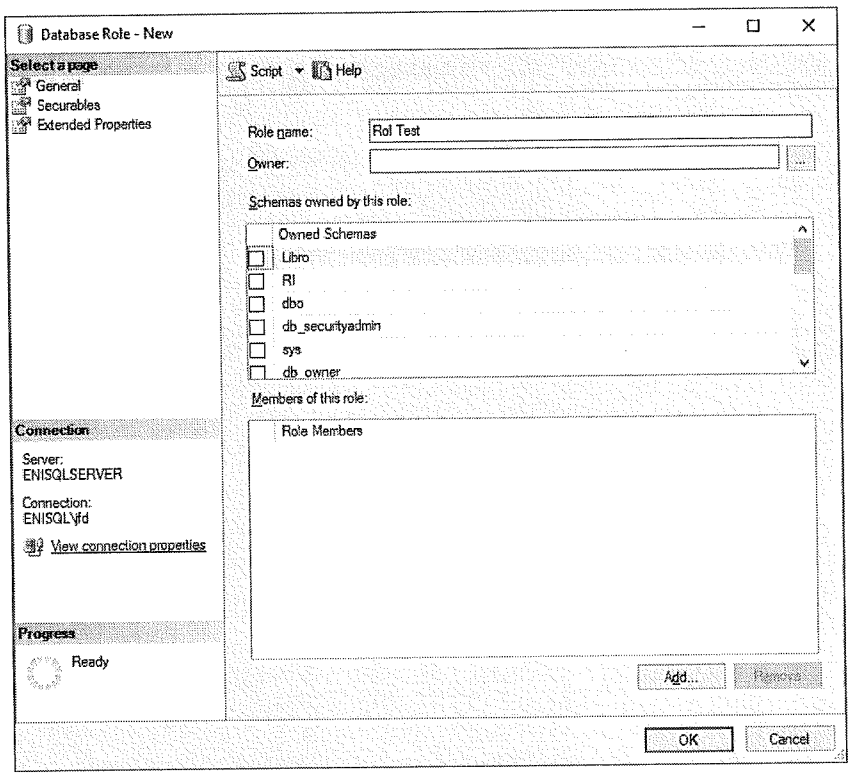
SQL Server Management Studio

La creación de los roles de base de datos es posible realizando las operaciones siguientes:

- Desde el explorador de objetos, sitúese sobre la base de datos de usuario en la que se va a realizar la operación.
- Sitúese sobre el nodo **Security - Roles - Application Roles**.
- Desde el menú contextual asociado al nodo **Application Roles**, seleccione **New Application Role**.

Ejemplo

En la pantalla siguiente, se crea el rol Rol Test.



Es posible modificar el rol en la ventana que presenta sus propiedades.

Transact SQL

Sintaxis

```
CREATE ROLE nombreRol
[ AUTHORIZATION propietario ]
```

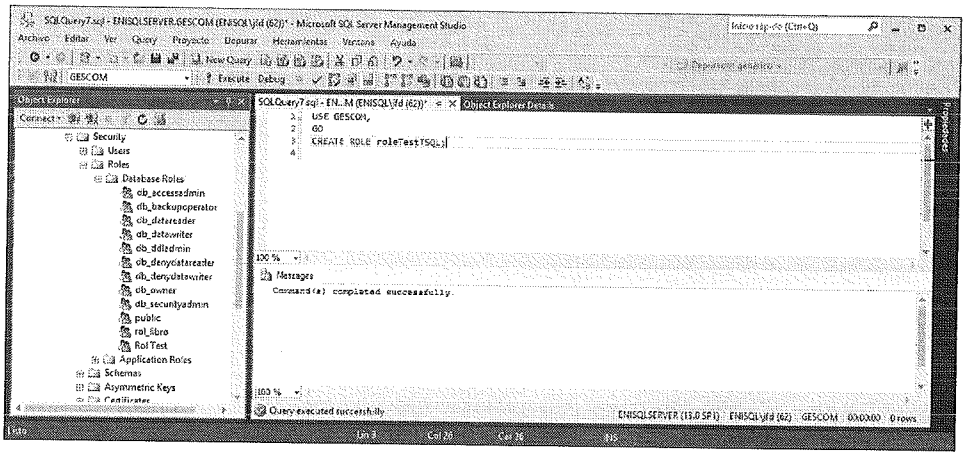
`nombreRol`

Nombre del rol que acaba de crearse.

`propietario`

Es posible indicar un propietario para el rol.

Ejemplo



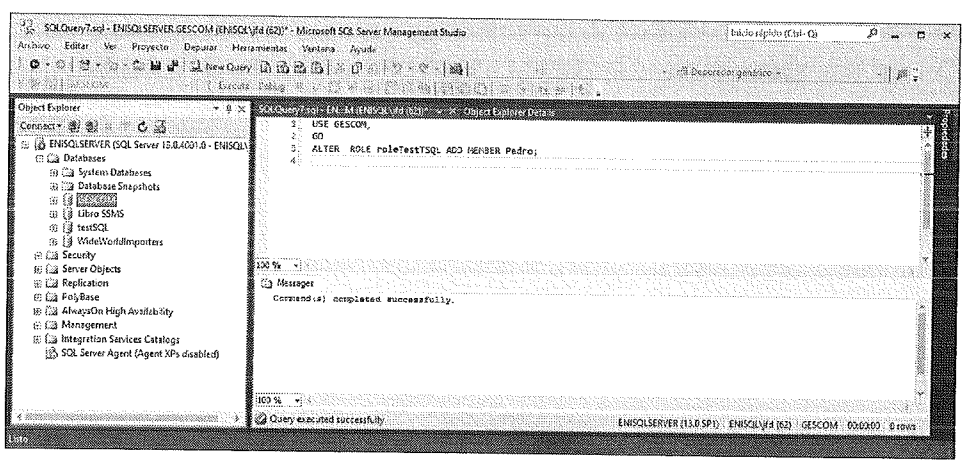
7.2.5 Administración de miembros de un rol

La administración de los miembros de un rol se hace en Transact SQL con la instrucción ALTER ROLE.

Sintaxis

```
ALTER ROLE nombreRolServidor ADD MEMBER nombreUsuario;
ALTER ROLE nombreRolServidor DROP MEMBER nombreUsuario;
```

Ejemplo



Observación

*Los procedimientos almacenados **sp_addrolemember** y **sp_droprolemember** se mantienen por razones de compatibilidad, pero no se recomienda utilizarlos.*

7.2.6 Eliminación de un rol

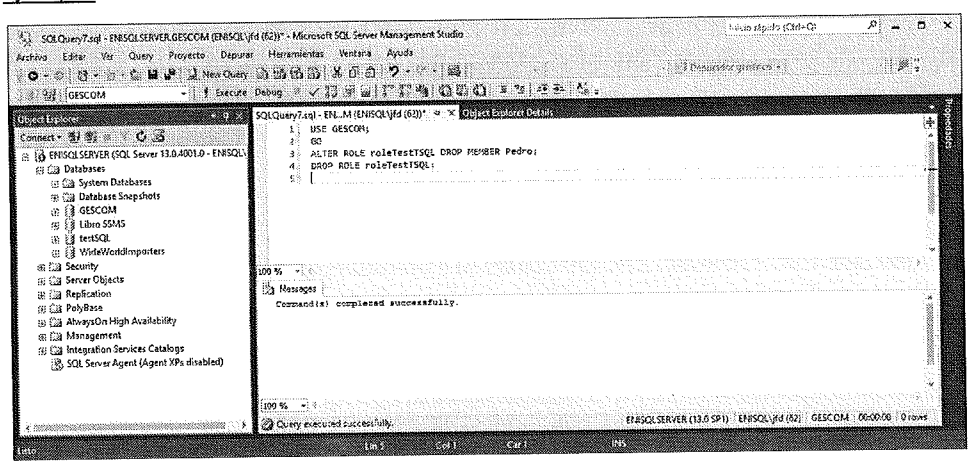
La opción de eliminación está disponible desde el menú contextual asociado al rol que se desea eliminar.

Transact SQL

Sintaxis

```
DROP ROLE nombreRol
```

Ejemplo



7.3 Roles de aplicación

Los roles de aplicación son los roles definidos a nivel de la base de datos sobre la que existen. Como todos los roles, los roles de aplicación permiten agrupar autorizaciones de objetos y de instrucciones. Sin embargo, los roles de aplicación se distinguen porque no tienen ningún usuario y están protegidos por una contraseña.

La lógica de un rol de aplicación es permitir a todos los usuarios de una aplicación tener suficientes derechos para el correcto funcionamiento de dicha aplicación, donde las operaciones que pueden realizarse sobre la base de datos son controladas por el programa cliente. Sin embargo, los usuarios no disponen de suficientes privilegios para realizar el conjunto de las operaciones directamente en SQL.

Cuando un rol de aplicación se activa desde una aplicación cliente o desde un script, las autorizaciones contenidas en este rol tienen prioridad sobre todas las autorizaciones asignadas directamente o por medio de roles al usuario de la base de datos.

Los roles de aplicación permiten obtener un comportamiento estándar de la aplicación sea cual sea el usuario de Windows que ejecute dicha aplicación.

Como los roles de aplicación se definen a nivel de la base de datos, no es posible asignarles privilegios sobre otras bases. Es más, la conexión a otras bases de datos solo es posible por medio de la cuenta **guest**.

7.3.1 Creación de un rol de aplicación

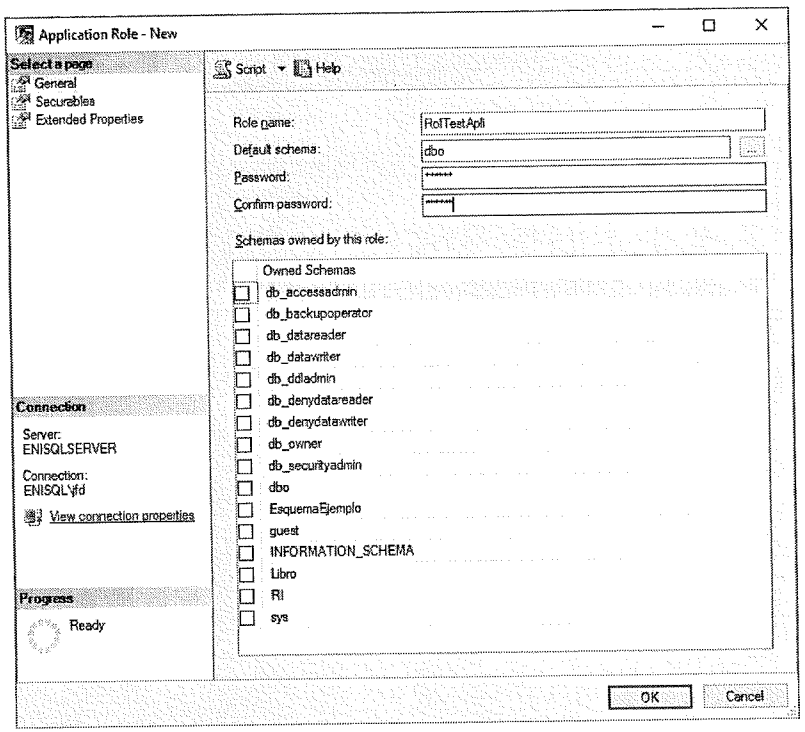
SQL Server Management Studio

Los roles de aplicación se gestionan de manera similar a la utilizada para los roles de base de datos.

- Desde el explorador de objetos, sitúese sobre la base de datos de usuario en la que se va a realizar esta operación.
- Sitúese sobre el nodo **Security - Roles - Application Roles**.
- Desde el menú contextual asociado al nodo **Application Roles**, seleccione **New Application Role**.

Ejemplo

Se crea el rol de aplicación RolTestApli.

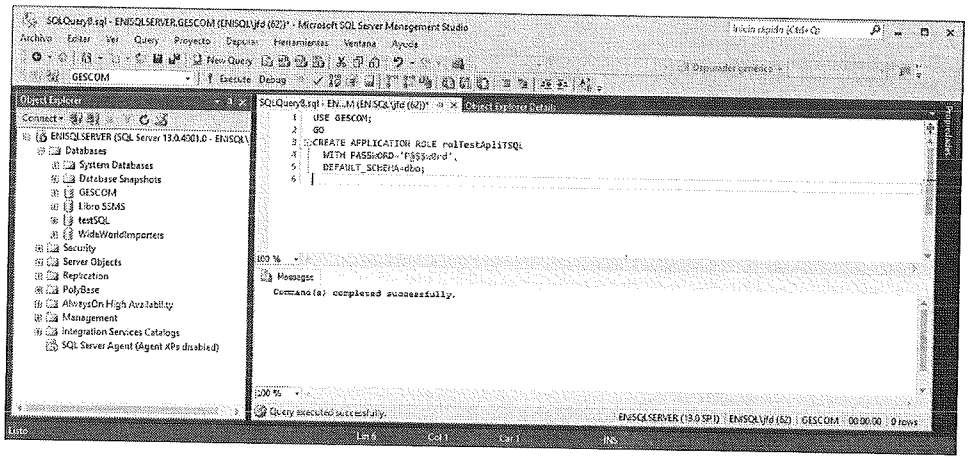


Transact SQL

Sintaxis

```
CREATE APPLICATION ROLE nombreRolAplicación
       WITH PASSWORD = 'contraseña'
       [, DEFAULT_SCHEMA = esquema ]
```

Ejemplo



7.3.2 Eliminar un rol de aplicación

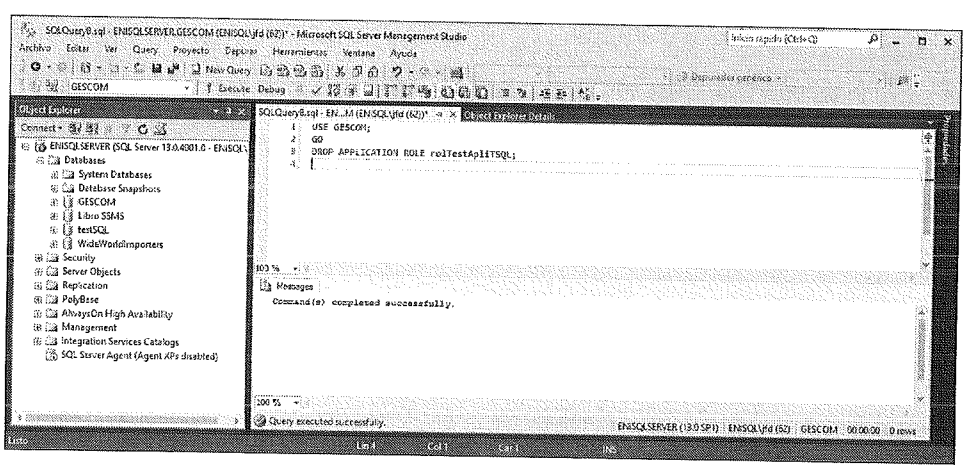
La eliminación de un rol de servidor se realiza con el menú contextual asociado al rol, seleccionando la opción **Delete**.

Transact SQL

Sintaxis

```
DROP APPLICATION ROLE nombreRolAplicación
```

Ejemplo



7.3.3 Modificar un rol de aplicación

Es posible modificar la contraseña del rol por medio de las **Propiedades del rol**.

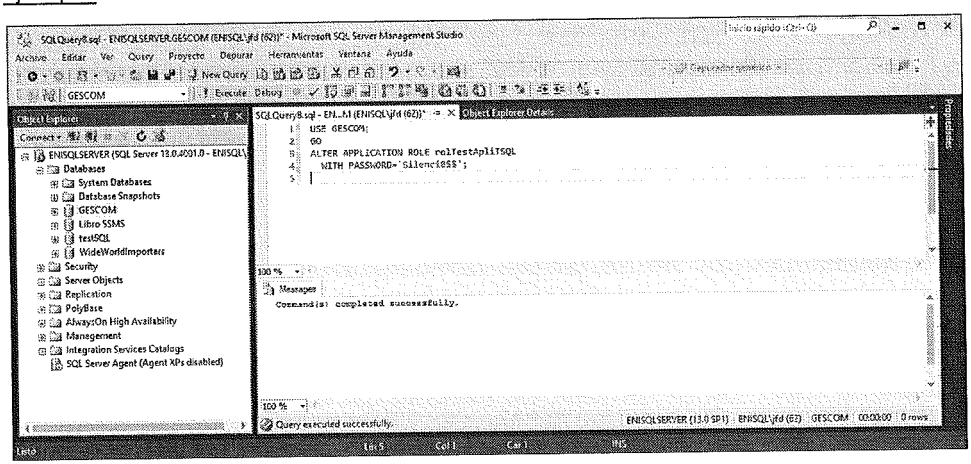
La adición de permisos sobre las instrucciones u objetos se realiza con las mismas etapas que la gestión de los derechos a nivel de los usuarios.

Transact SQL

La instrucción ALTER permite modificar la contraseña, pero también el esquema por defecto y el nombre del rol.

```
ALTER APPLICATION ROLE nombreRolAplicación
        WITH {NAME = nuevoNombre|
              PASSWORD = 'nuevaContraseña'|
              DEFAULT_SCHEMA = nuevoNombreEsquema}
```

Ejemplo



7.3.4 Activación de un rol de aplicación

Transact SQL

La activación de un rol de aplicación se realiza con el procedimiento almacenado **sp_setapprole**. Desde que se activa un rol de servidor, los permisos del usuario se ignoran y solo cuentan los asignados al rol de aplicación.

Sintaxis

```
sp_setapprole 'rol',
[Encrypt N] 'contraseña'
[, 'estilo_cifrado']
```

`rol`

Nombre del rol de aplicación que se va a activar.

`contraseña`

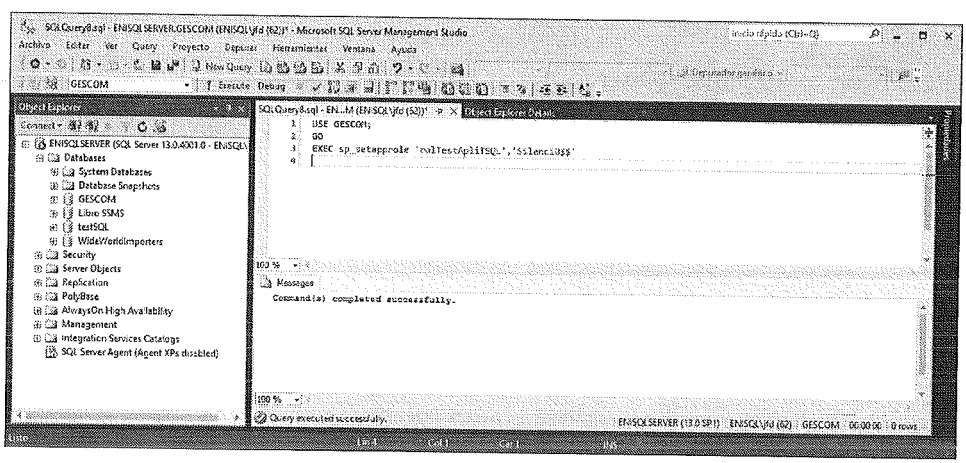
La activación de un rol de aplicación está condicionada por la contraseña. La contraseña se puede encriptar con la función canónica ODBC Encrypt. La opción N permite convertir la contraseña al formato unicode.

`estilo_cifrado`

Permite especificar un estilo de encriptado para la contraseña. Se dispone de dos estilos:

- none: La contraseña se transmite a SQL Server sin codificación inicial. Es la opción predeterminada.
- odbc: La contraseña se codifica con la ayuda de la función Encrypt de ODBC. Esta funcionalidad solo es posible en un cliente ODBC que se comunique con el servidor OLE DB de SQL Server. Esta opción permite enmascarar la contraseña (ofuscación), pero la contraseña no se encripta.

Ejemplo



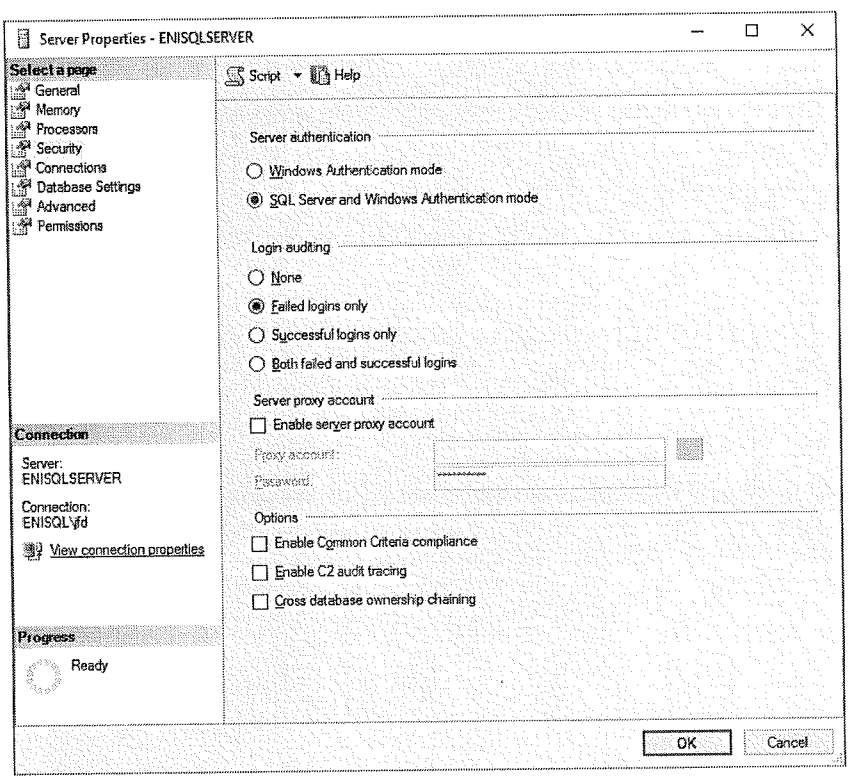
8. Ejercicio: modo de seguridad

8.1 Enunciado

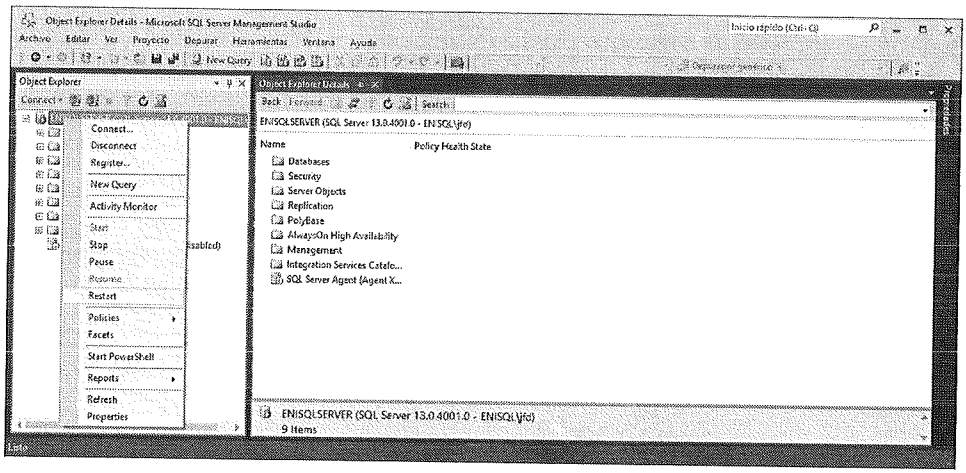
Configure el modo de seguridad mixto (SQL Server y Windows) sobre la instancia Libro.

8.2 Solución

Para activar el modo de seguridad mixto en una instancia SQL Server, debemos mostrar la ventana que presenta las propiedades de la instancia desde el explorador de objetos en SQL Server Management Studio. Desde la ventana de propiedades, seleccionaremos la opción **Security** y después la opción **SQL Server and Windows Authentication mode** en la zona **Server authentication**.



Para que este nuevo modo de configuración sea tenido en cuenta debemos reiniciar el servicio asociado a esta instancia. Esta operación se puede realizar directamente desde SQL Server Management Studio seleccionando la opción.



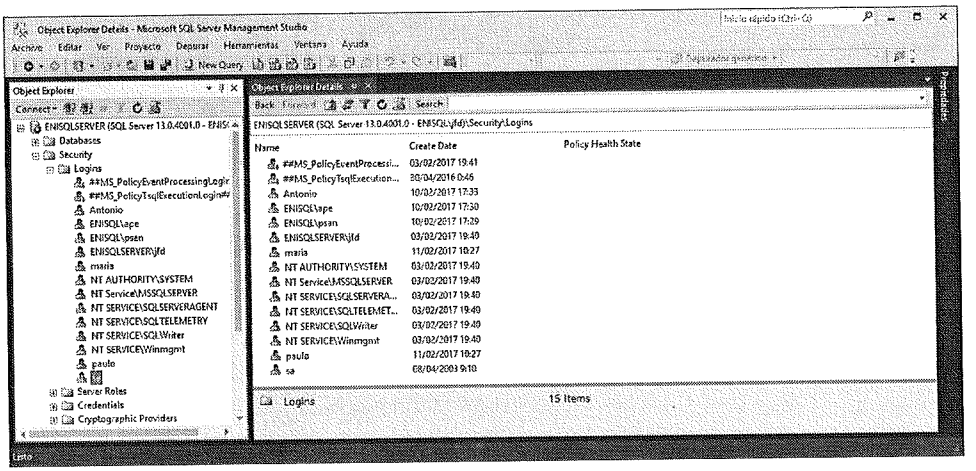
9. Ejercicio: cuenta sa

9.1 Enunciado

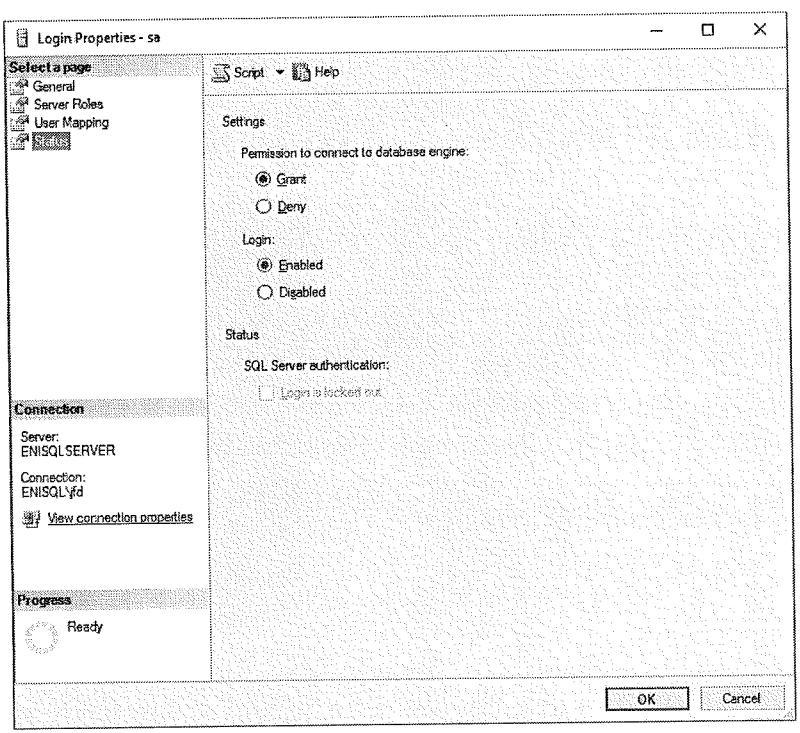
Para la instancia Libro, active la cuenta sa y defina una contraseña para esta cuenta de nivel administrador.

9.2 Solución

Desde el explorador de objetos, desplegando el árbol **Security - Logins**, es fácil identificar las cuentas desactivadas ya que el ícono que las representa contiene un flecha apuntado hacia abajo, como es el caso para la cuenta sa.



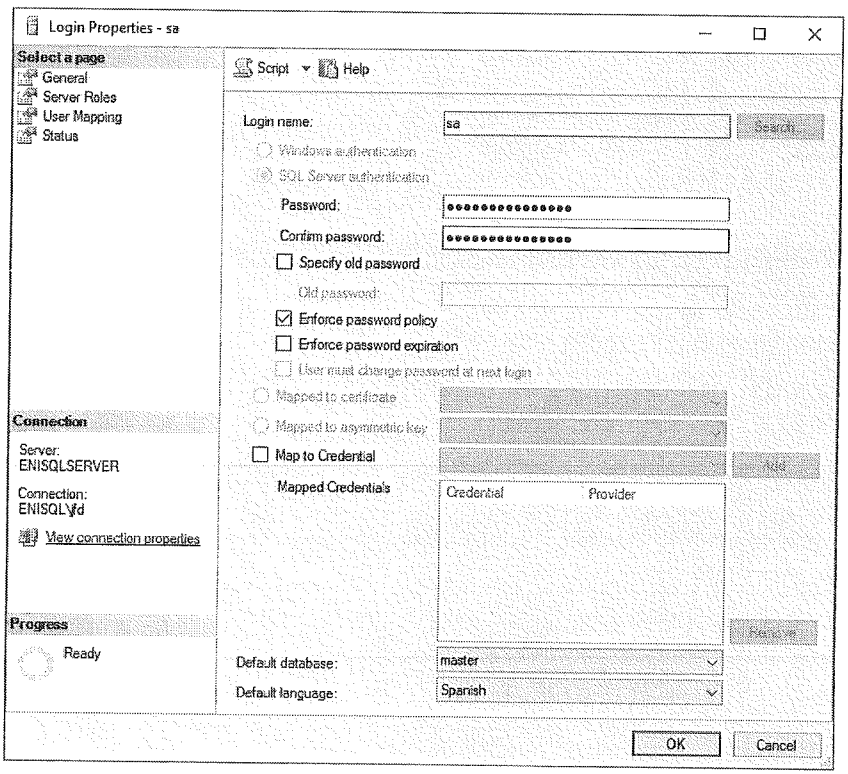
Desde la ventana de propiedades de esta conexión, debemos mostrar la página de **Status** y seleccionar la opción **Enabled** en la zona de **Login**.



Podemos realizar la misma operación con ayuda del commando ALTER LOGIN de la siguiente manera:

```
ALTER LOGIN sa ENABLE;
```

La siguiente etapa consiste en definir una contraseña para esta conexión. Esto podemos hacerlo desde la página **General**.



El cambio de contraseña podemos relizarlo también con el siguiente script Transact SQL:

```
ALTER LOGIN sa WITH PASSWORD='P@$$w0rd';
```

10. Ejercicio: crear usuarios SQL Server

10.1 Enunciado

Cree las conexiones Pablo y Juan para la instancia Libro. Estas dos cuentas poseen las siguientes características.

La cuenta Pablo se crea mediante un script Transact SQL mientras que la cuenta Juan se crea desde SQL Server Management Studio.

Cuenta Pablo

Contraseña: P@$$w0rd

Rol servidor: ninguno

Base de datos por defecto: ninguna

Cuenta Juan

Contraseña: P@$$w0rd

Rol servidor: ninguno

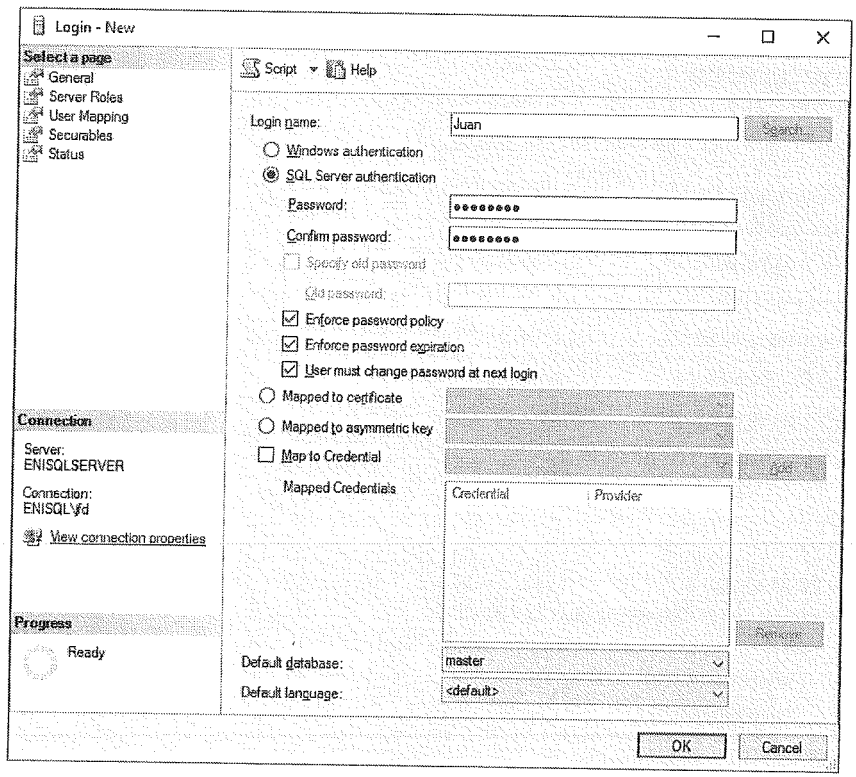
Base de datos por defecto: ninguna

10.2 Solución

La creación de la cuenta Pablo se realiza con un script Transact SQL. Por lo tanto es posible introducir directamente la instrucción.

```
CREATE LOGIN Pablo
  WITH PASSWORD='P@$$w0rd';
```

Para realizar el mismo tipo de operación desde SQL Server Management Studio y por lo tanto crear la conexión Juan, basta con seleccionar la opción **New Login** desde el menú contextual de la rama **Security - Logins** del explorador de objetos.



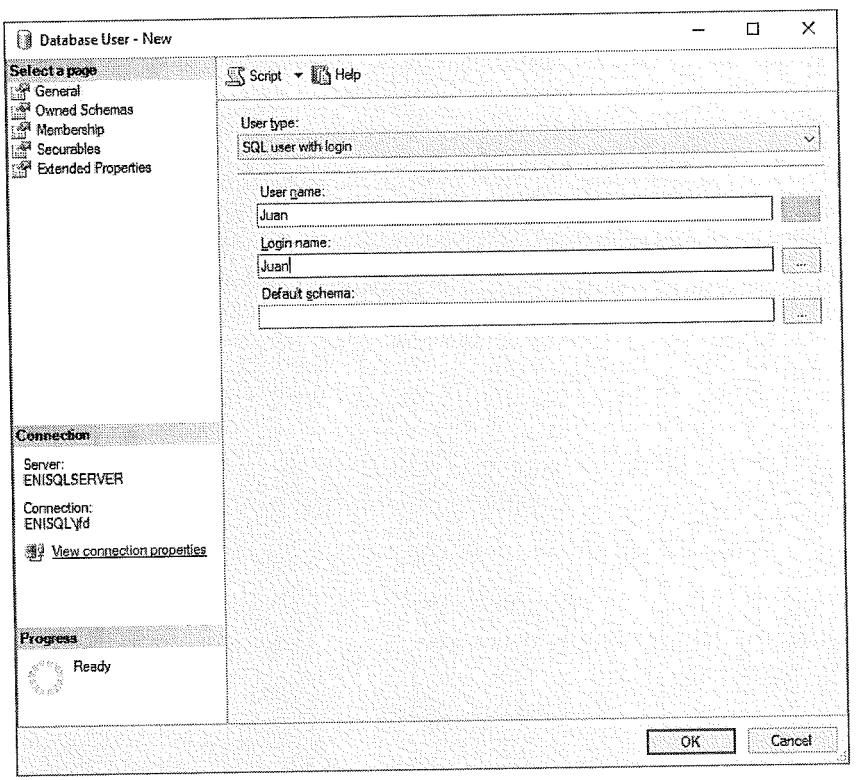
11. Ejercicio: crear usuarios de base de datos

11.1 Enunciado

En la base de datos LibroSSMS, cree el usuario Juan correspondiente a la conexión Juan definida para el servidor.

11.2 Solución

Desde SQL Server Management Studio, seleccionamos la opción **New Login** desde el menú de la rama **Databases - LibroSSMS - Security - Logins**, y configuramos el nuevo usuario como se puede ver a continuación:



También se puede usar la instrucción CREATE USER, lo que produce el siguiente script:

```
USE LibroSSMS;
GO
CREATE USER Juan FOR LOGIN Juan;
```

Es necesario comenzar el script con la instrucción USE LibroSSMS para posicionarse sobre la base de datos correcta.

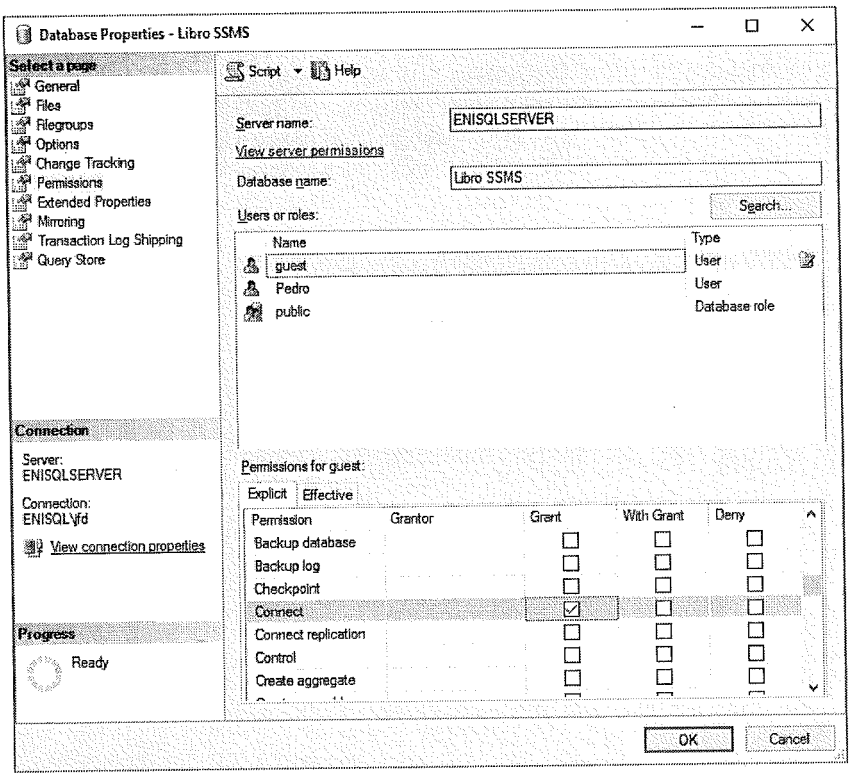
12. Ejercicio: activar la cuenta de invitado

12.1 Enunciado

En la base de datos LibroSSMS, active la cuenta de invitado.

12.2 Solución

La cuenta invitado (o guest) está por defecto desactivada en las bases de datos. Es una buena medida de seguridad. Debemos por lo tanto activarla si deseamos utilizarla. Podemos realizar esta operación desde SQL Server Management Studio mostrando las **propiedades de la base de datos** y seleccionando la sección **Permissions**. Desde allí debemos añadir la cuenta de invitado en la zona **Users or roles** y, después de haberla seleccionado, asignar, en la zona **Permissions for guest**, el permiso de **Connect**.



13. Ejercicio: crear un rol de base de datos

13.1 Enunciado

En la base de datos LibroSSMS, cree el rol de base de datos rolSSMS y concédale los permisos de crear tablas y vistas.

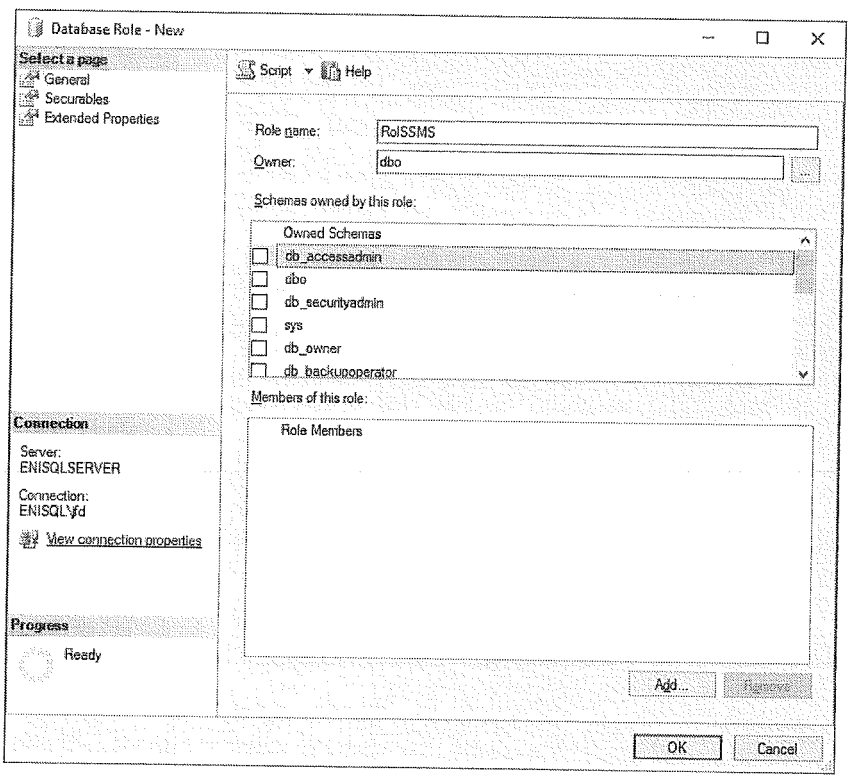
13.2 Solución

Debemos realizarlo en dos etapas; en primer lugar creamos el rol de base de datos y después concedemos los permisos.

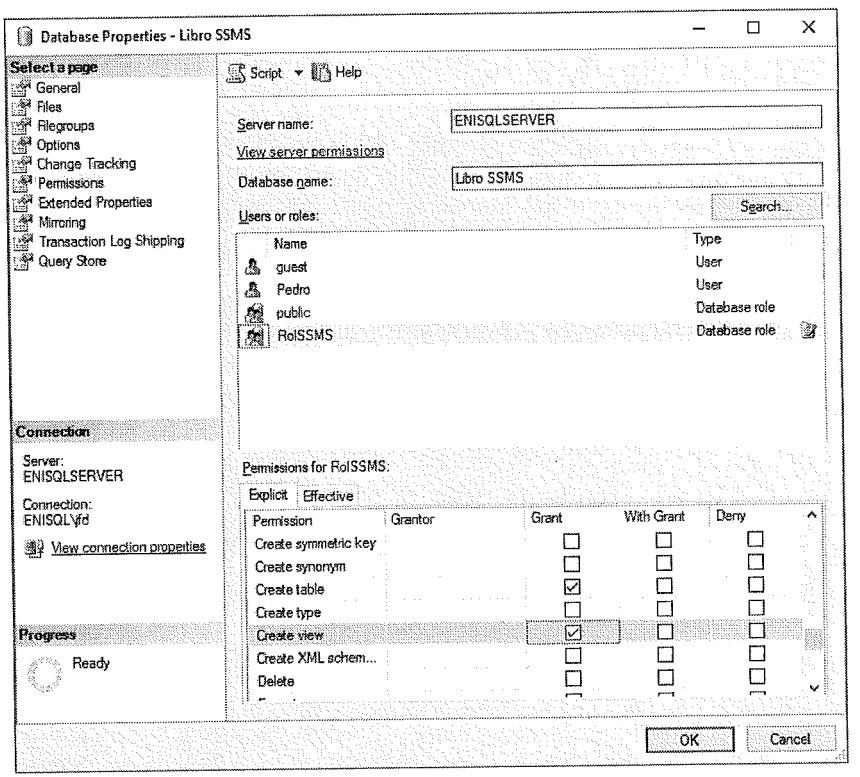
Estas dos etapas las podemos realizar desde SQL Server Management Studio o con ayuda de un script Transact SQL.

Desde SQL Server Management Studio

Desde el explorador de objetos, seleccionamos la rama **Security - Roles - Database Roles** en la base de datos LibroSSMS, y en el menú contextual de esta rama elegimos la opción **New Database Role**. Se abre la siguiente ventana de propiedades, que se permite crear un nuevo rol y debemos completarla como sigue:



Los permisos de creación de tablas y vistas son permisos que se pueden conceder desde la sección **Permissions** en el cuadro de diálogo de **Properties** de la base de datos LibroSSMS.



Con un script Transact SQL

Debemos utilizar las instruccione CREATE ROLE para crear el rol y después GRANT para conceder permisos. El script presentado a continuación muestra una posible solución:

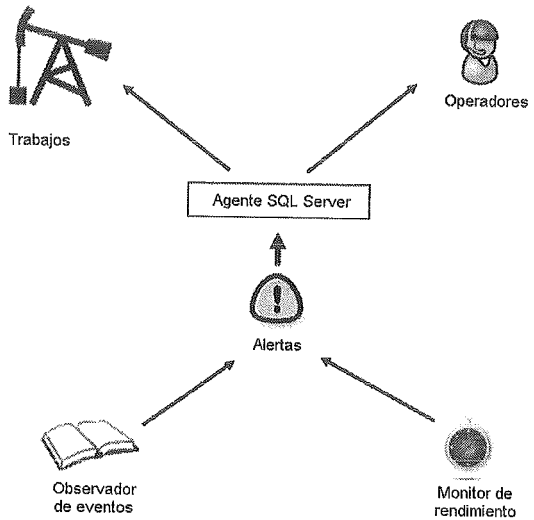
```
USE LibroSSMS;
GO
CREATE ROLE RolSSMS;
GO
GRANT CREATE TABLE, CREATE VIEW to RolSSMS;
```

Capítulo 5
Tareas planificadas

1. Introducción

SQL Server ofrece la posibilidad de automatizar las tareas administrativas. Por supuesto, no es posible automatizar todas las tareas, pero las tareas planificadas son un buen complemento a la optimización que SQL Server hace por defecto. Además, con estas tareas predefinidas, el administrador tiene un rol de anticipador, lo que le da más posibilidades para salir adelante de la mejor manera posible, tanto en términos de rendimiento como de fiabilidad.

La gestión de tareas planificadas, las alertas y las operaciones son servicios gestionados por el agente SQL Server. Este servicio debe iniciarse para que todos estos elementos sean administrados. El agente SQL Server trabaja con el Observador de eventos para gestionar los errores de SQL Server, con el Monitor de rendimiento para gestionar las alertas relativas a las condiciones de rendimiento y con la base de datos MSDB para averiguar la respuesta que hay que aplicar ante una alerta o las tareas planificadas que hay que ejecutar.



Principio de funcionamiento

Cuando se produce una alerta, el agente puede reaccionar de dos maneras diferentes; ejecutando un trabajo determinado o previniendo a un operador para que esté al corriente del problema que acaba de aparecer. Obviamente, la ejecución de una tarea puede conducir a la activación de nuevas tareas y así sucesivamente.

Otras tareas planificadas van a ejecutarse por medio del servicio SQL Server Agent, no como respuesta a un error, sino utilizando criterios de tiempo. Por ejemplo, una reconstrucción de índices puede planificarse una vez por semana, la noche del sábado al domingo.

El agente SQL Server permite realizar una administración preventiva de problemas que pueden aparecer durante la operación rutinaria de un servidor de base de datos.

2. Configuración de los servicios

Como la ejecución automática de trabajos administrativos se apoya en el servicio SQL Server Agent, es importante que este último esté correctamente configurado.

Observación

La configuración del servicio MSSQL se ha tratado durante la instalación.

Excepcionalmente, es posible iniciar este servicio como una aplicación con **sqlagent90.exe**.

Esta aplicación en línea de comandos tiene las tres opciones siguientes:

- **-c**: para indicar si el agente SQL Server se ejecuta de manera independiente al servicio dedicado de Windows.
- **-v**: para detallado, es decir, que el agente SQL Server se ejecuta en modo documentado y se muestra la información directamente en la ventana que sirve para iniciar la aplicación.
- **-i nombreInstancia**: permite indicar sobre qué instancia de SQL Server queremos iniciar el agente SQL Server en modo línea de comandos.

2.1 Cuenta de inicio para SQL Server Agent

El servicio SQL Server Agent se ejecuta en el contexto de una cuenta de usuario. Esta cuenta puede ser Sistema Local, Servicio de Red o bien una cuenta de usuario (local o sobre un dominio). Los dos primeros casos corresponden a cuentas predefinidas que permiten hacer un determinado número de acciones. Si el servicio SQL Server Agent está preparado para acceder a los recursos disponibles en la red, es preferible que se ejecute en el contexto de una cuenta de usuario de domino. Durante la instalación en Windows 2008 R2 o una versión superior, es posible usar las cuentas virtuales o de servicios administrados (MSA), que se ajustan perfectamente a esta tarea. Esta opción se fija inicialmente durante la instalación, aunque es posible modificarla más adelante. En caso de utilizar una cuenta de usuario, no se asigna ningún privilegio directamente al usuario, sino que se crea el grupo SQLServer AgentUser$nombreOrdenador$MSSQLServer (para la instancia por defecto). A este grupo se le asignan los siguientes permisos:

- Abrir una sesión como un servicio.
- Abrir una sesión como un programa de tratamiento de lotes.
- Sustituir una ficha a nivel de proceso.
- Anular el control del recorrido.
- Cambiar las cuotas de memoria de un proceso.

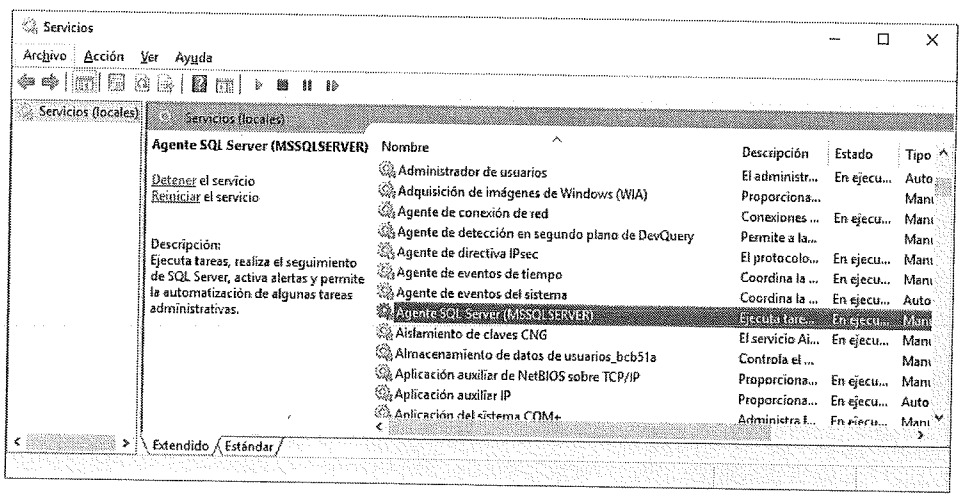
El hecho de pasar por un grupo simplifica la gestión de privilegios durante el cambio de la cuenta utilizada.

Observación

Es posible configurar este servicio para que se inicie de forma automática. Adicionalmente, esta opción permite gran flexibilidad a la hora de usar el agente SQL Server.

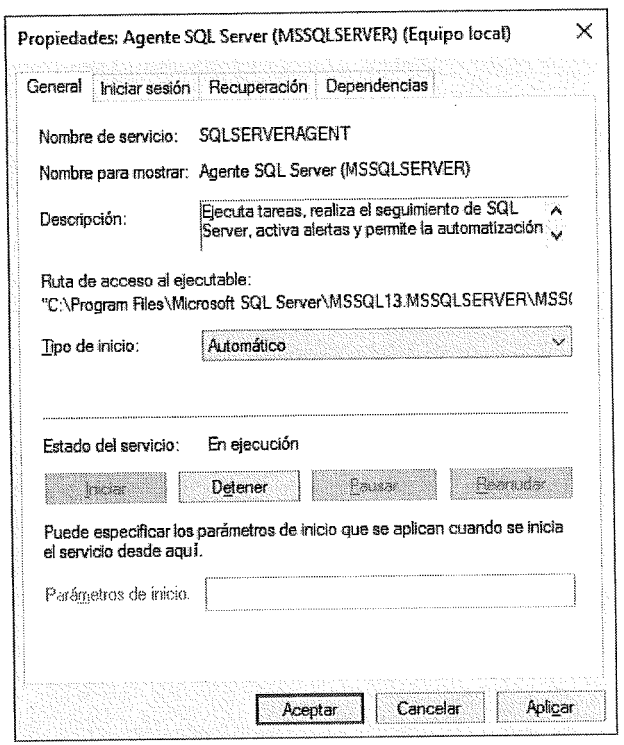
2.1.1 Configuración del servicio en Windows

Para configurar los servicios relativos a MSSQLServer y al Agente SQL Server, es posible utilizar la consola de gestión de servicios de Windows. Sin embargo, es preferible configurar los servicios que se deben ejecutar en el contexto de una cuenta de usuario de dominio en la instalación de SQL Server.



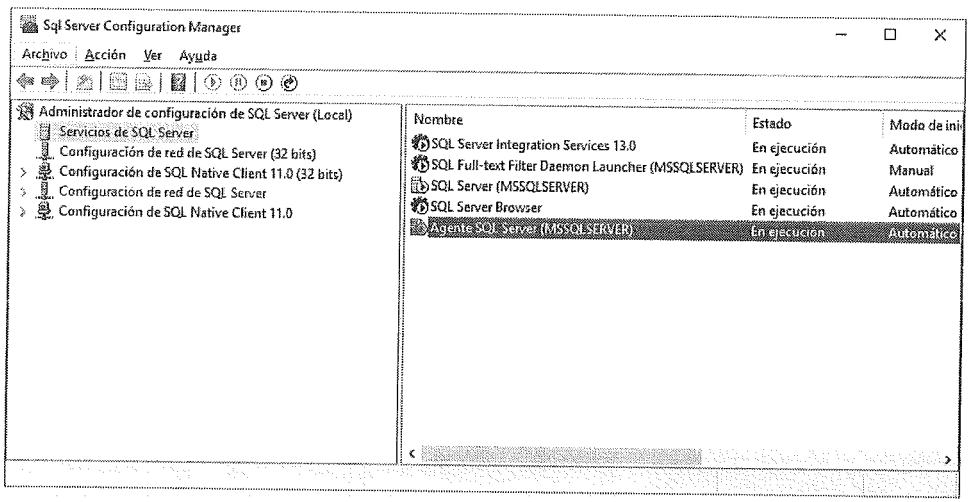
Después de haber seleccionado el servicio SQLSERVERAGENT, es necesario llamar al cuadro de diálogo que muestra las propiedades del servicio con el fin de poder averiguarlas y modificarlas.

Las propiedades pueden visualizarse utilizando el icono de la barra de herramientas, a través del menú **Acción - Propiedades** o seleccionando **Propiedades** en el menú contextual.

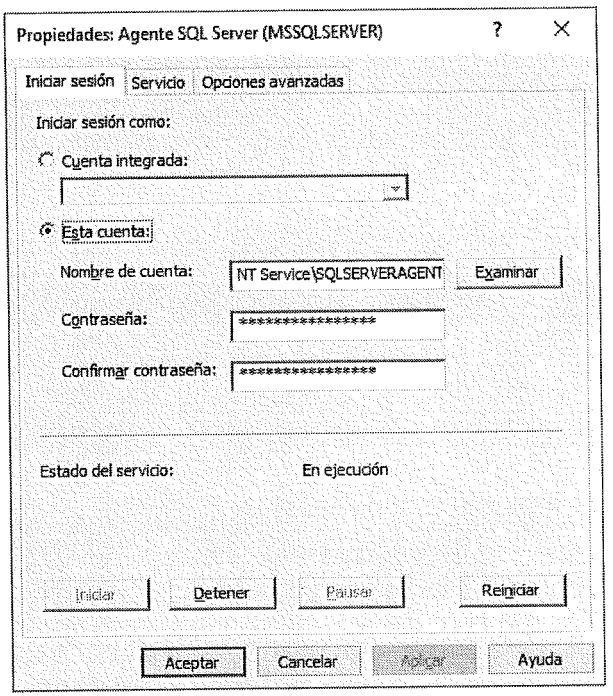


2.1.2 Configuración del servicio en SQL Server Configuration Manager

Es posible gestionar la configuración del servicio desde SQL Server Configuration Manager. Esta utilidad permite visualizar exclusivamente los servicios de SQL Server.



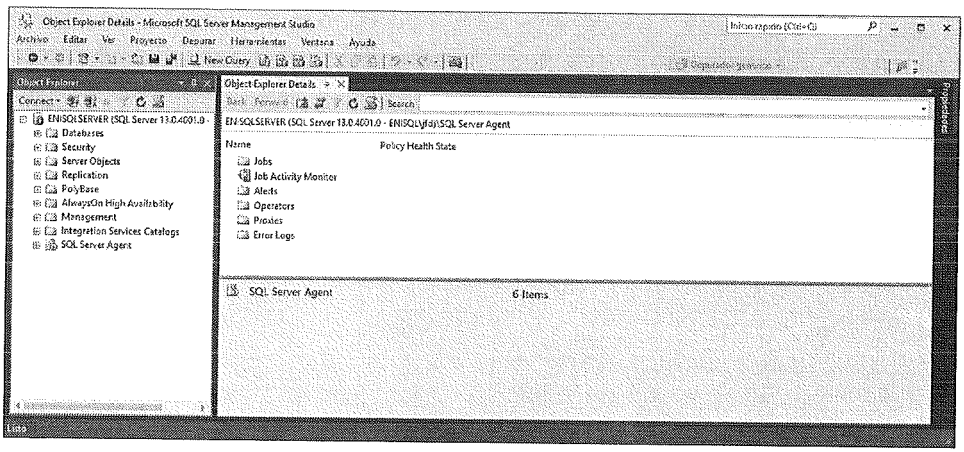
Para visualizar y modificar las propiedades, es necesario hacer clic sobre el servicio o bien seleccionar **Propiedades** en el menú contextual asociado al servicio.



Observación

Para facilitar la gestión de los servicios, es preferible que los servicios MSSQL Server y SQLServerAgent se ejecuten en el contexto de una misma cuenta de usuario de dominio, aunque esto no es obligatorio. De la misma manera, si varios servidores SQL deben ponerse en contacto en uno o varios dominios, todos los servicios se ejecutarán en una cuenta de usuario con el mismo nombre y la misma contraseña.

Igualmente, es posible averiguar el estado del servicio (iniciado o no) desde SQL Server Management Studio. Desde aquí es posible iniciar, detener o suspender el servicio.



2.1.3 La seguridad de SQL Server Agent

El servicio SQL Server Agent permite administrar numerosos elementos. En caso de que el servicio deba tener derechos importantes sobre el servidor para poder realizar correctamente todas las tareas que le son asignadas, la utilización de este servicio debe controlarse de la manera más precisa posible. Este control está asegurado por los tres roles de base de datos **SQLAgentUserRole**, **SQLAgentReaderRole** y **SQLAgentOperatorRole** definidos en la base de datos msdb. La pertenencia a estos roles es necesaria únicamente para los usuarios que no sean miembros del rol de servidor sysadmin.

Por ejemplo, en caso de que un usuario se conecte a la consola gráfica SQL Server Management Studio sin ser miembro de uno de estos tres roles, la herramienta no presentará el nodo relativo a SQL Server Agent. De esta manera, el usuario no es capaz de modificar, ni siquiera de saber, el trabajo realizado a nivel de la automatización de tareas. El mismo nivel de seguridad se define a nivel de Transact SQL.

Además de los roles de base de datos, el servicio dispone de subsistemas y proxies para gestionar lo mejor posible todos los elementos de seguridad.

Los subsistemas permiten presentar la funcionalidad disponible para una etapa del trabajo.

Un proxy para SQL Server Agent permite gestionar la seguridad relativa a varios subsistemas y las conexiones asociadas.

De esta manera, el propietario de una etapa de trabajo solo podrá indicar un proxy para esa etapa si y solo si pertenece a dicho proxy.

2.2 Configuración de la mensajería electrónica

La gestión de los correos electrónicos con SQL Server pasa por el servicio de correo electrónico de la base de datos. Este servicio mejora la gestión de los correos electrónicos dentro de la empresa, ya que ofrece una mayor flexibilidad en términos de puesta en práctica y rendimiento y también garantiza una mayor seguridad.

El servicio de correo electrónico de la base de datos utiliza el protocolo estándar SMTP para enviar correos electrónicos. No se apoya sobre MAPI, por lo que es opcional instalar un cliente de mensajería como Outlook. A través de este protocolo, el servicio de correo electrónico de la base de datos se encarga del envío de correos electrónicos en formato HTML.

El servicio de correo electrónico se ejecuta en un proceso distinto al de SQL Server. De esta manera, si este servicio falla, no va a perturbar el buen funcionamiento de la base de datos. Los correos electrónicos se sitúan en la cola de espera del proceso asociado al servicio de correos electrónicos de la base de datos.

El servicio de mensajería de la base de datos no está habilitado por defecto. El asistente de configuración se encarga de hacerlo antes de configurarlo. Esta activación es obligatoria para que funcione correctamente el servicio.

Como sucede con cualquier operación de configuración, es posible hacer esta parametrización gráficamente con SQL Server Management Studio o usando scripts Transact SQL que llamen a instrucciones y procedimientos almacenados concretos.

Teniendo en cuenta el carácter puntual de esta operación de configuración, en este libro solo se presenta el modo gráfico.

El uso de la mensajería electrónica para enviar correos está limitado a determinados elementos.

Es necesario pertenecer al rol DataBaseMailUserRole si deseamos enviar un mensaje electrónico.

Para permitir una mayor robustez del servicio de mensajería electrónica, se pueden establecer perfiles. Cada perfil corresponde a un conjunto de cuentas de mensajería electrónica que se van a utilizar para enviar un mensaje. Existen dos tipos de perfiles:

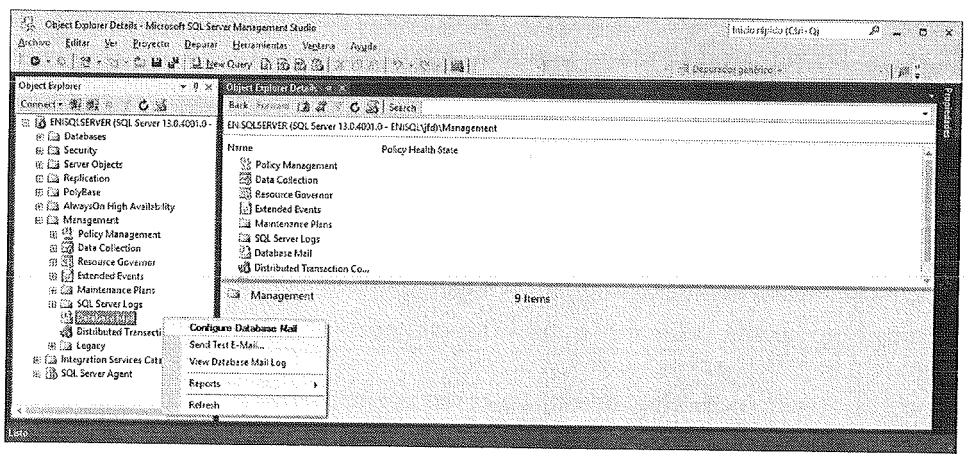
- Los perfiles públicos, a los que pueden acceder todos los usuarios que pertenezcan al rol **DatabaseMailUserRole**, en la base de datos msdb. Un usuario puede mandar mensajes si es miembro de este rol.
- Perfiles privados, que también se definen en la base de datos msdb, pero esta vez cada usuario se debe identificar de manera explícita para poder usar el perfil.

2.2.1 Configuración desde SQL Management Studio

El asistente de configuración de la mensajería de base de datos nos va a guiar en la realización de una de las siguientes acciones:

- Configurar el correo electrónico de base de datos.
- Gestionar las cuentas del correo electrónico de base de datos.
- Gestionar los perfiles de seguridad.
- Gestionar los parámetros de sistema.

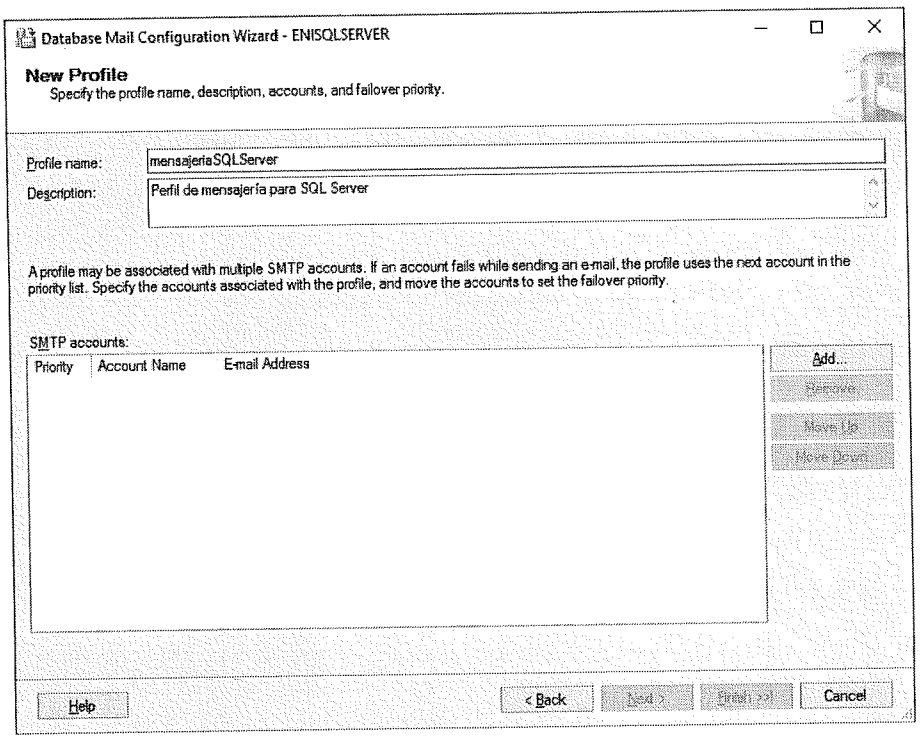
El asistente se lanza desde SQL Server Management Studio seleccionando **Configure Database Mail** en el menú contextual asociado al nodo **Management - Database Mail** de la instancia de SQL Server sobre la que se realiza la configuración.



La primera pantalla del asistente (después de la pantalla de bienvenida) permite seleccionar la acción que se desea realizar con él.

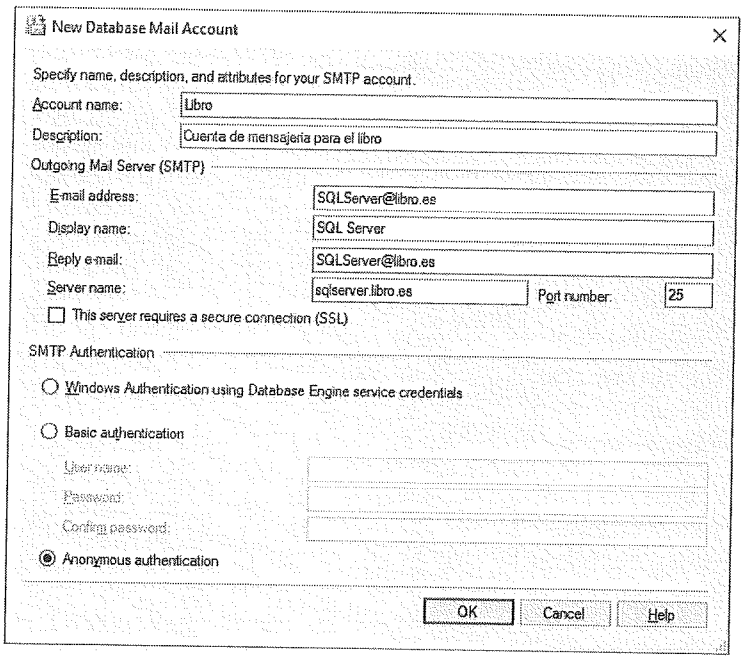
En caso de que la selección se efectúe sobre la configuración del servicio, el asistente se encarga de iniciar el servicio.

Para que el servicio de correo electrónico de base de datos pueda funcionar, debe disponer de una cuenta de correo electrónico. Esta cuenta se define en el perfil. Los perfiles permiten agrupar de manera lógica las diferentes cuentas de correo.



Definición de un primer perfil

Utilizando el botón **Add** es posible definir una o varias cuentas de correo.



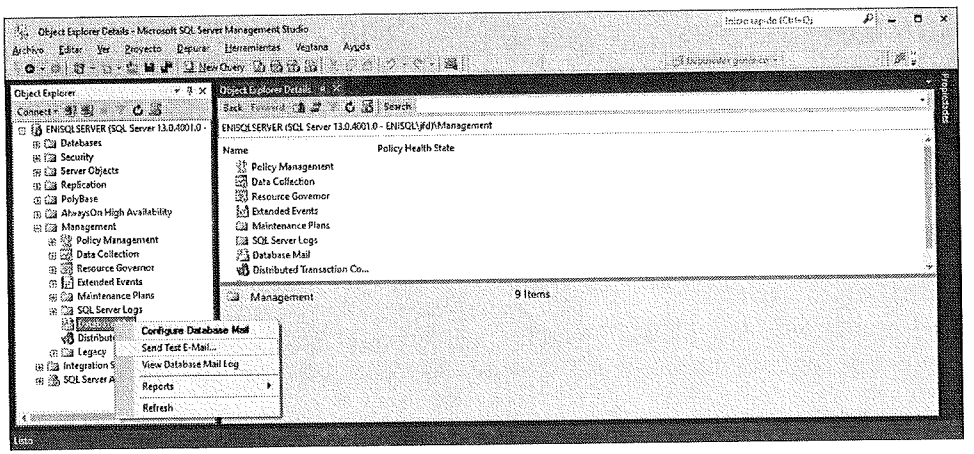
Definición de una cuenta de correo electrónico

Posteriormente, el asistente ofrece hacer público el perfil, es decir, accesible al conjunto de los usuarios del servidor. Por último, el asistente termina en una pantalla de síntesis que resume las diferentes operaciones solicitadas. La validación de este resumen implica la creación del perfil.

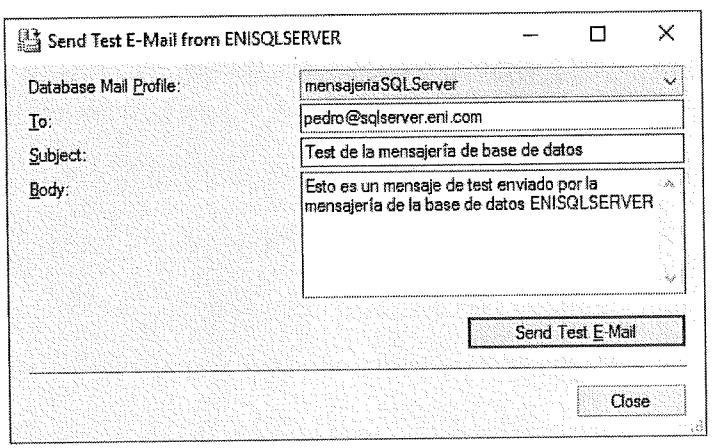
2.2.2 Probar el servicio

Solo los usuarios miembros del rol de servidor **sysadmin** o del rol de base de datos **databaseMailUserRole** definido sobre la base **msbd** pueden enviar correos electrónicos.

Es fácil probar el perfil seleccionando **Send Test E-Mail** desde el menú contextual asociado al nodo **Database Mail**, desde el explorador de objetos de SQL Server Management Studio.



Una pantalla permite precisar el perfil que se va a utilizar, así como el destinatario del mensaje de prueba.



3. Los operadores

Un operador puede corresponder a una persona física o bien a una función de la empresa. En función del tamaño de la empresa, una misma persona física puede desempeñar la función de varios operadores, o bien un mismo operador puede corresponder a varias personas físicas.

Los operadores serán utilizados por el agente SQL Server para avisar del fin de la ejecución de un trabajo o del desencadenamiento de una alerta para informar de la gravedad de la situación.

Para establecer la comunicación con los operadores, el agente SQL Server dispone de tres canales de comunicación: el correo electrónico, la radiomensajería y el mensaje a través de la red (net send).

La entrega de un correo electrónico al operador pasa por el servicio de mensajería y es el servidor de mensajería el que se encarga de enviar el correo electrónico hasta su destinatario.

Para los mensajes de tipo radiomensajería, el agente SQL Server también se basa en el servicio de mensajería: es el servidor de correo electrónico el que se encarga de transmitir el mensaje al destinatario en función de la pasarela configurada en el servidor. El mensaje inicial proporcionado por el agente SQL Server deberá respetar las restricciones impuestas por esta pasarela.

Por último, los mensajes a través de la red, que corresponden a la ejecución del comando net send, se basan en los servicios de mensajería instantánea de Windows Messenger. Este servició debe ejecutarse sobre el servidor y sobre el puesto del operador.

La definición de todos los operadores se almacena en la base **msdb**.

3.1 Creación

Es preferible definir los operadores antes que las alertas y los trabajos, ya que las operaciones administrativas se encadenan en un orden lógico.

Consideraciones generales

Antes de crear los operadores, es preciso tener en cuenta algunos puntos importantes:

- Si varias personas están afectadas por el mismo problema, es necesario utilizar un alias de grupo para el correo electrónico. Este método es más flexible que definir tantos operadores como personas posibles.
- Probar todos los métodos de notificación de operadores.
- El comando **net send** solo se puede utilizar para avisar a los operadores y servidores que ejecutan Windows.

Observación

La gestión de los nombres de mensajería presenta algunas limitaciones y, para evitar problemas, conviene dar el nombre completo de la dirección de correo electrónico (ejemplo: libro@eni.es) con objeto de evitar conflictos si varios operadores tienen el mismo nombre.

Las principales características de un operador son:

- Su nombre.
- La información que lo describe.
- Los medios de que dispone SQL Server Agent para avisarle.

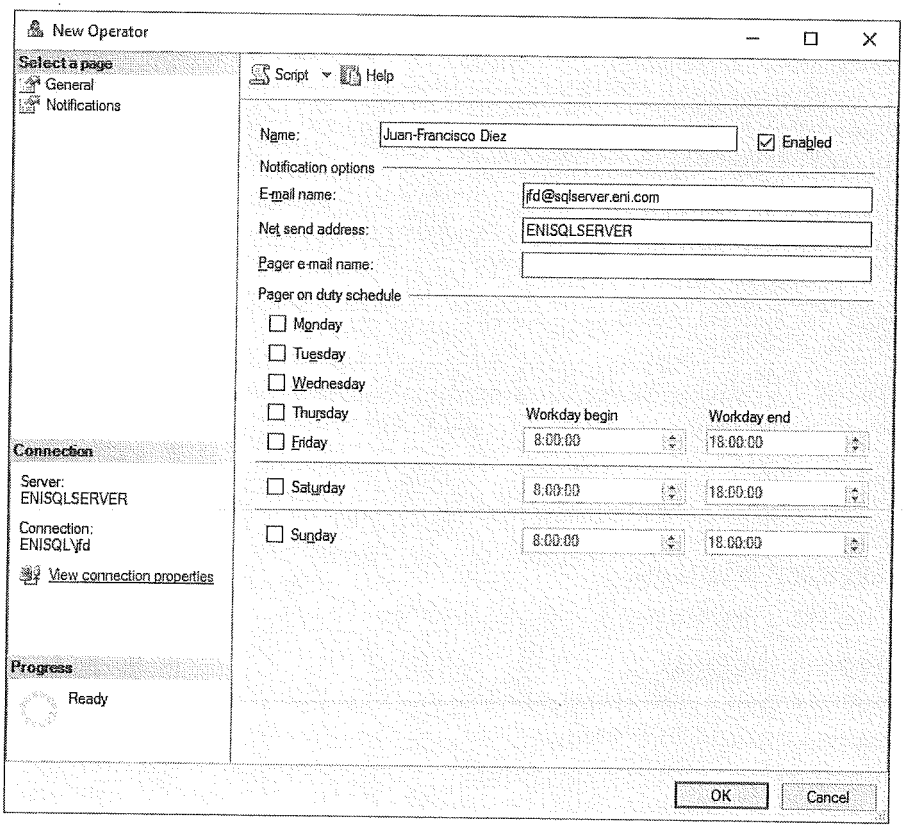
El operador suplente

Es posible definir un operador suplente. Este último recibirá el mensaje únicamente si el operador inicial está ilocalizable.

Este operador también se llama operador de prevención contra fallo y solo está disponible para las alertas enviadas por radio mensajería.

SQL Server Management Studio

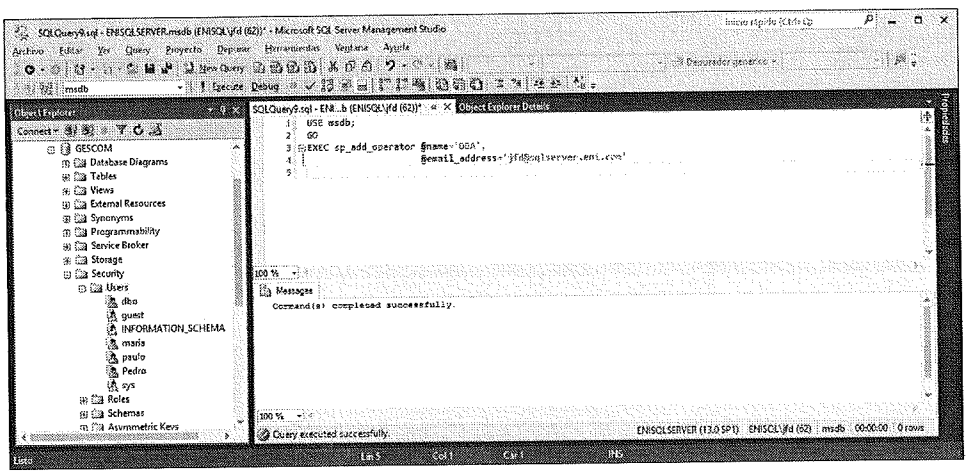
Para crear un nuevo operador, es necesario seleccionar **New Operator** desde el menú contextual asociado al nodo **SQL Server Agent - Operators** del explorador de objetos de SQL Server Management Studio.



Transact SQL

Es posible crear un nuevo operador a través del procedimiento almacenado **sp_add_operator**.

Ejemplo



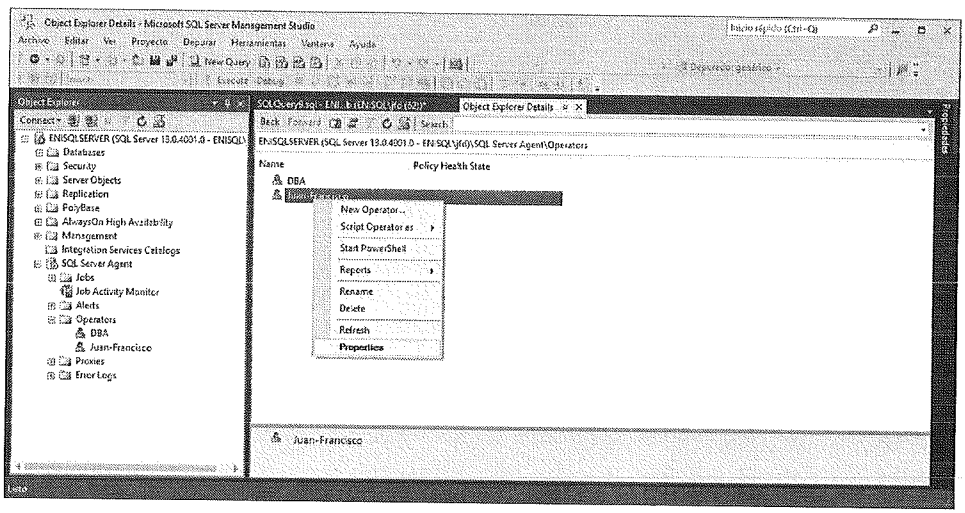
Observación

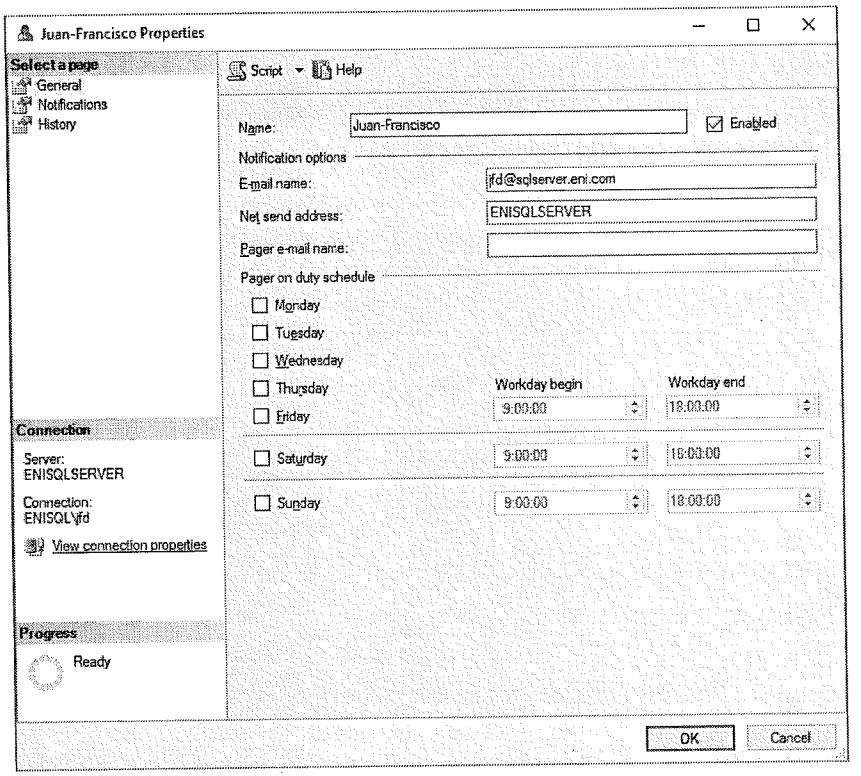
*Todos los procedimientos almacenados están en la base de datos **msdb**. La primera instrucción del script debe ser, por lo tanto, **use msdb**.*

3.2 Modificación

SQL Server Management Studio

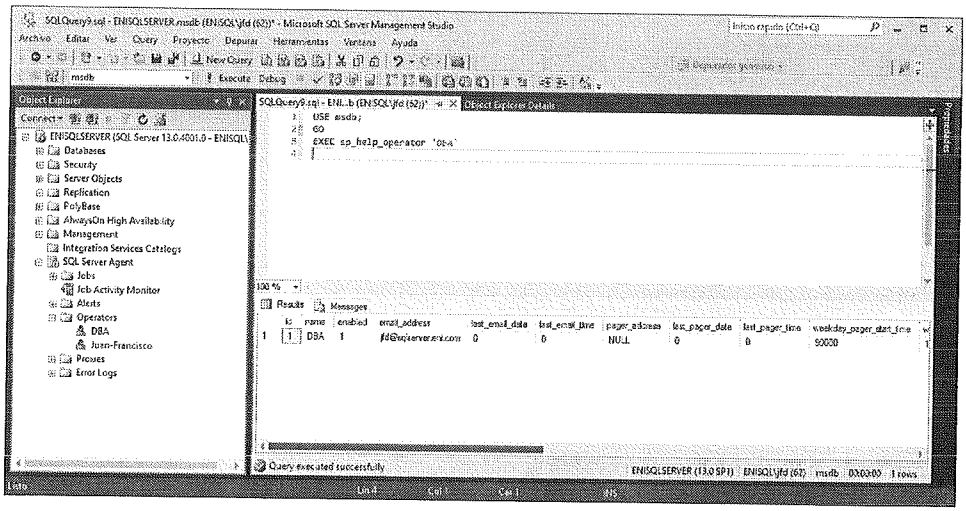
A través del cuadro de diálogo **Properties** del operador, es posible averiguar la información relacionada y modificar algunos parámetros.



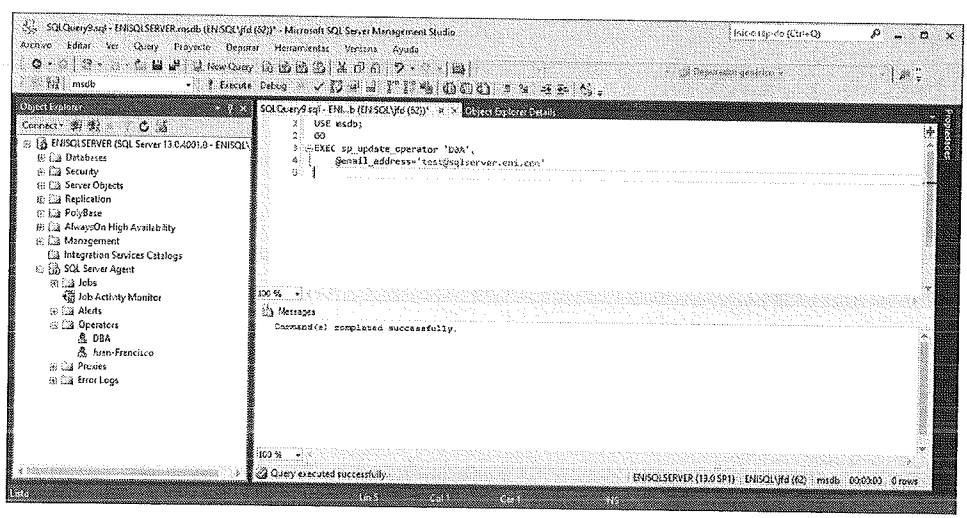


Transact SQL

El procedimiento almacenado **sp_help_operator** permite averiguar los parámetros actuales del operador.



Para realizar modificaciones sobre los operadores ya definidos, es suficiente con ejecutar el procedimiento **sp_update_operator**.

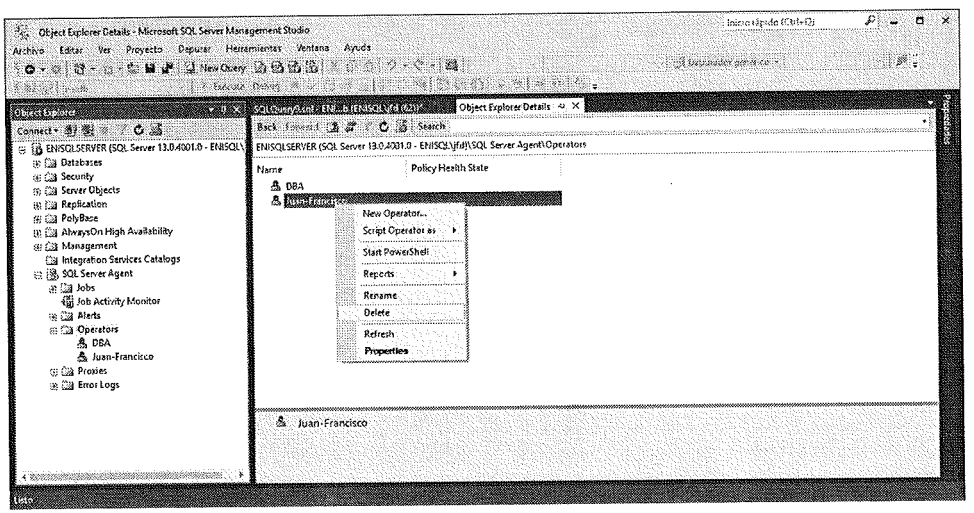


3.3 Eliminación

La eliminación de un operador únicamente es posible si este último no es un operador de prevención contra fallos.

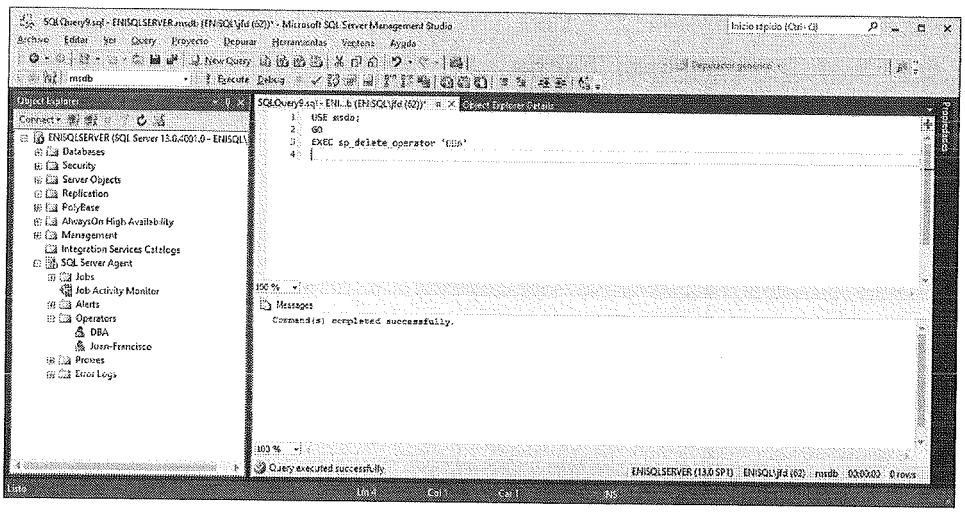
SQL Server Management Studio

En el menú contextual asociado al operador, seleccione la opción **Delete**.



Transact SQL

Es suficiente con ejecutar el procedimiento **sp_delete_operator**.



4. Los trabajos

Es posible automatizar algunas tareas de administración repetitivas a través de trabajos y de la planificación de su ejecución. Por supuesto, también es posible lanzar manualmente su ejecución.

Los trabajos están formados por un conjunto de tareas. Al final de cada tarea se pueden presentar dos casos: o bien la tarea se ha ejecutado con éxito o bien ha habido un fallo.

El trabajo, que es un encadenamiento de tareas, debe definir todas las soluciones posibles para que pueda ejecutarse correctamente.

Todos los trabajos se almacenan en la tabla **sysjobs** de la base **msdb**.

4.1 Implantación

Las principales características de un trabajo son:

- El nombre: debe ser único en el servidor y está limitado a 128 caracteres.
- La categoría: permite organizar los trabajos en función de las operaciones que realizan. Existen, desde la instalación del servidor, categorías predefinidas, como búsqueda de texto completo, mantenimiento de la base de datos...

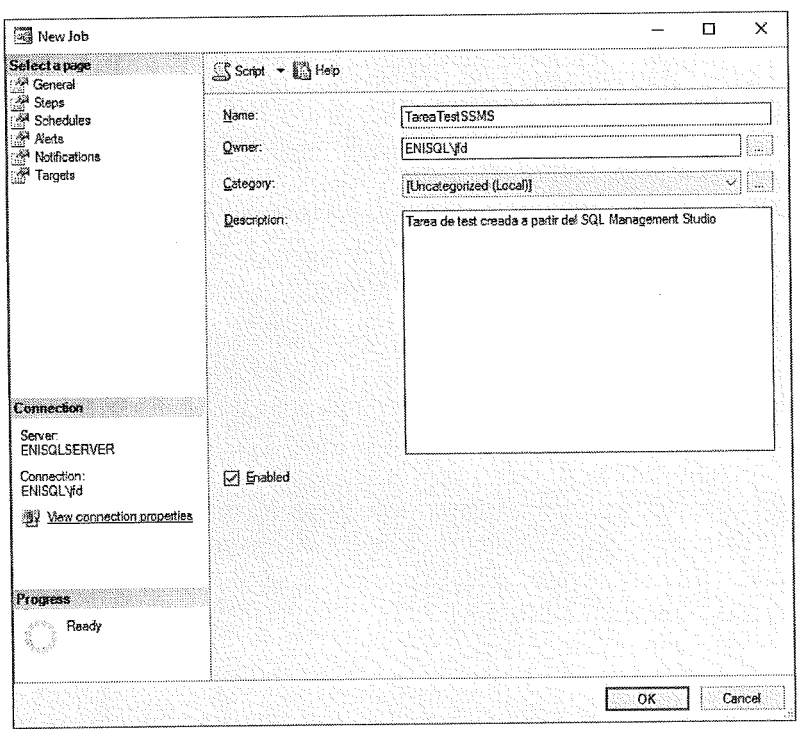
- El propietario: este puede ser diferente del usuario que lo ha creado.
- La descripción.
- Las etapas del trabajo.
- La planificación.
- La notificación.

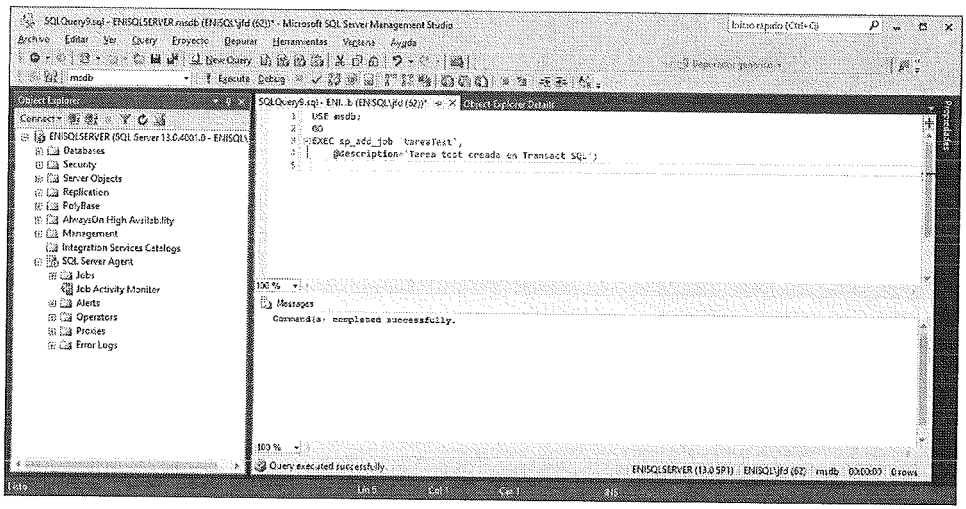
Cada trabajo puede estar asociado a una categoría. La agrupación por categoría permite una agrupación lógica de los diferentes trabajos y, por lo tanto, una presentación de mejor calidad en SQL Server Management Studio.

La creación de un trabajo solo es posible para un administrador del sistema (sysdamin) o bien para un usuario miembro de uno de los tres roles de base de datos asociados a SQL Server Agent.

La modificación y la eliminación de un trabajo solo las puede efectuar el propietario o un administrador del sistema.

La definición de un trabajo puede realizarla bien SQL Server Management Studio o bien el procedimiento almacenado **sp_add_job**.





En el momento de crear un trabajo, es necesario:

- Verificar que el trabajo está habilitado.
- Determinar el lugar en que se ejecuta el trabajo en caso de tener múltiples servidores.

Observación

Los trabajos solo pueden ejecutarse si el servicio SQLServerAgent está iniciado.

4.2 Definición de las etapas de un trabajo

Es posible definir las etapas de un trabajo ya sea a través de SQL Server Management Studio o bien mediante el procedimiento almacenado **sp_add_jobstep**.

Todas las etapas se almacenan en la base de datos **msdb** y la tabla **sys_add_jobstep**.

Las etapas del trabajo están caracterizadas, lo que permite seleccionar el subsistema apropiado para la ejecución de la etapa. Existen siete tipos de etapas. Cuatro de ellas se detallan a continuación; los otros tipos de tareas son las relativas a los scripts Microsoft ActiveX, las tareas Analysis Services y las tareas Integration Services.

El resultado de la ejecución de las diferentes etapas se almacena en la tabla **sysjobstepslogs** de la base de datos **msdb**.

4.2.1 Transact SQL (TSQL)

Este es el tipo por defecto de una etapa de trabajo.

Es posible ejecutar instrucciones Transact SQL, procedimientos almacenados o procedimientos almacenados extendidos. Es necesario tener en cuenta los puntos siguientes:

- Al ejecutar un procedimiento, se debe asignar valor a todos los parámetros obligatorios.
- El resultado de la ejecución se puede enviar a un archivo de salida. Sin embargo, no es posible utilizar el resultado de salida de una etapa como entrada para la etapa siguiente.

4.2.2 Comando del sistema operativo (CMDEXEC)

En una etapa, es posible ejecutar un comando del sistema operativo o iniciar la ejecución de un programa o de un archivo de comandos. Es necesario, sin embargo:

- Identificar un código de salida para indicar que el comando ha terminado con éxito.
- Indicar la ruta completa del programa o del archivo de comandos.

La etapa del trabajo debe hacer referencia a un elemento ejecutable del sistema, es decir, un archivo que tenga la extensión cmd, bat o exe. La ruta de acceso a este archivo debe ser una ruta completa.

4.2.3 PowerShell

Windows Server ofrece un lenguaje de scripting: el PowerShell. Todos los servidores Microsoft ofrecen bibliotecas específicas para administrar el servidor con scripts. Con este tipo de etapa se pueden realizar tareas de gestión del servidor o del motor de base de datos. Cada etapa de este tipo ejecuta un proceso **sqlps** que consume 20 MB de memoria RAM. Si se ejecutan muchas tareas de este tipo de manera simultánea, puede tener una incidencia importante en el rendimiento global del servidor.

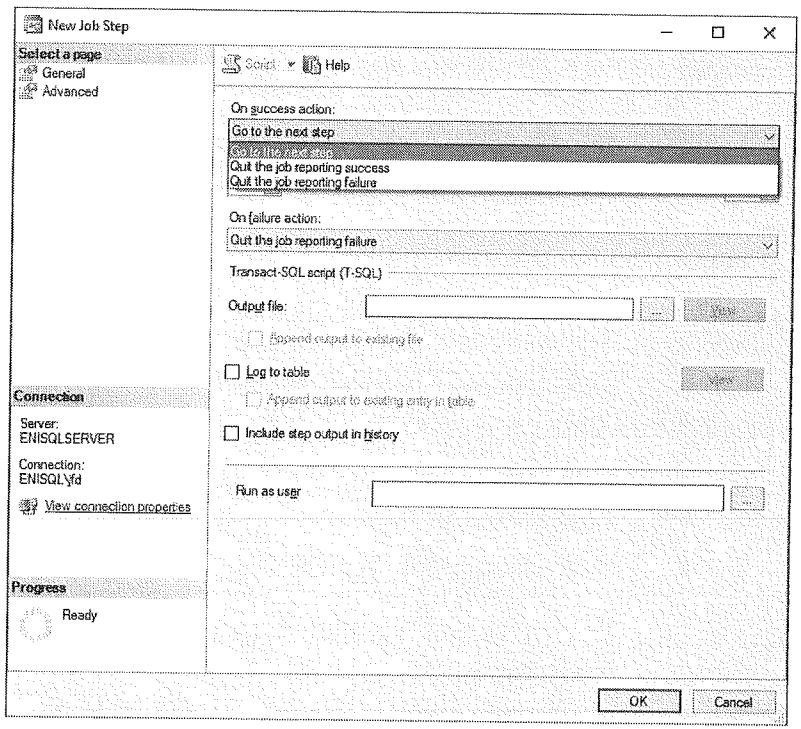
4.2.4 Replicación

El proceso de replicación se llama por medio de los agentes y se pone en marcha en forma de trabajos.

Las tareas relativas a la replicación se representan por medio de tipos de etapa muy diferentes, como SNAPSHOT o DISTRIBUTION, que corresponden, respectivamente, a las tareas de captura instantánea o de distribución.

4.3 Encadenamientos entre las etapas

Al final de cada etapa, hay dos situaciones posibles: la etapa ha sido un éxito o bien se ha producido un error.



Por defecto, en caso de éxito, SQL intenta ejecutar la etapa siguiente. Para cada etapa, se puede precisar un número de intentos de ejecución, así como el intervalo de espera entre cada intento.

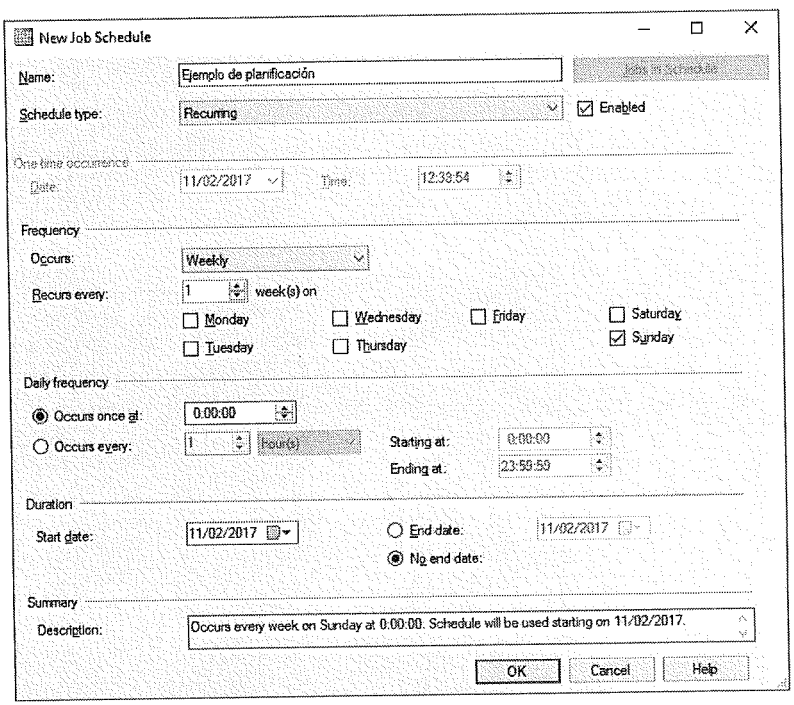
Por defecto, en caso de error en una etapa se termina el trabajo. Si se produce un error en una etapa, es posible notificar a un operador o registrar un mensaje en el Observador de eventos. Incluso es posible encadenar con una etapa cualquiera del trabajo.

Otra posibilidad consiste en configurar el trabajo para que sea eliminado automáticamente después de su ejecución.

4.4 La planificación

La ejecución de los trabajos puede ser planificada. Esta planificación será importante para todas las operaciones de mantenimiento de bases de datos, que podrán ejecutarse en momentos de baja actividad del servidor.

Las planificaciones las realiza SQL Server Management Studio o bien el procedimiento almacenado **sp_add_jobschedule**.



Los trabajos se ejecutan bien como respuesta a una alerta o bien según las planificaciones definidas. Si se trabaja en un entorno multiservidor, es posible precisar varios servidores de destino para un mismo trabajo.

La planificación de un trabajo solo se tiene en cuenta si esta está activa y el trabajo solo se ejecutará si el agente SQL Server está iniciado.

La planificación puede solicitar la ejecución del trabajo:

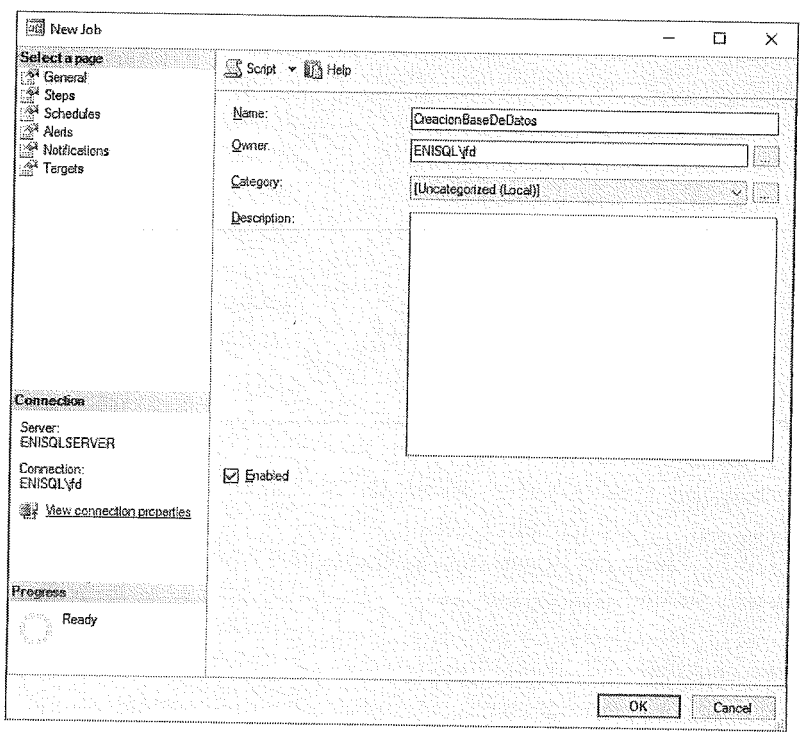
- A una fecha y hora determinados (una única ejecución).
- De manera regular: cada minuto, hora, día, semana, mes... o una combinación de estos criterios (ejemplo: ejecución del trabajo cada 2 horas de lunes a viernes de 08:00 h a 19:00 h).

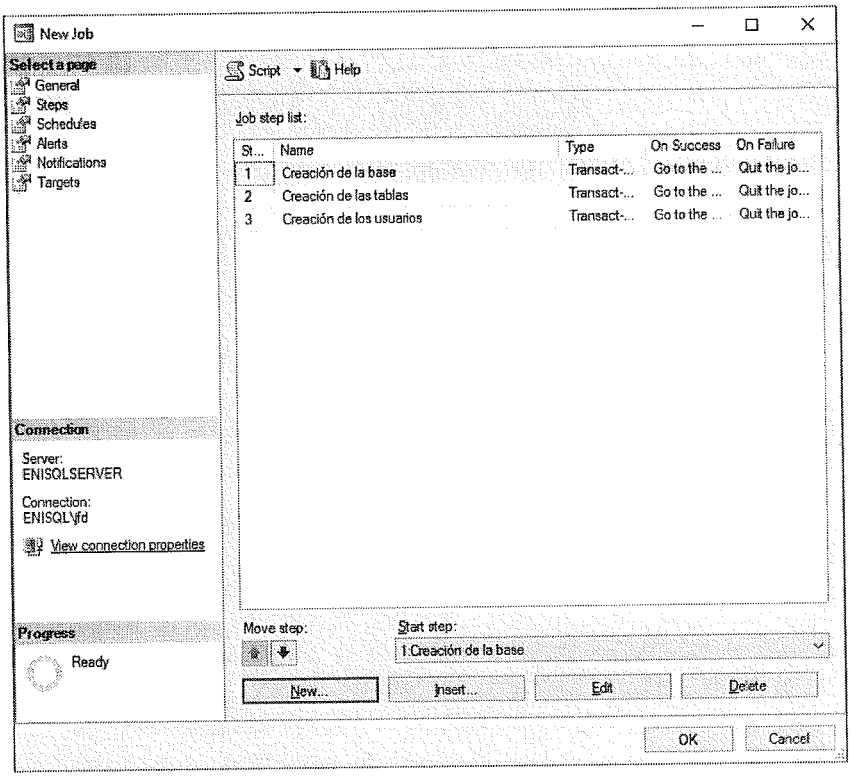
- Cuando el proceso está inactivo. Esta opción solo está disponible si el contexto de la cuenta de usuario utilizada por SQLServerAgent es miembro del grupo local de Administradores.

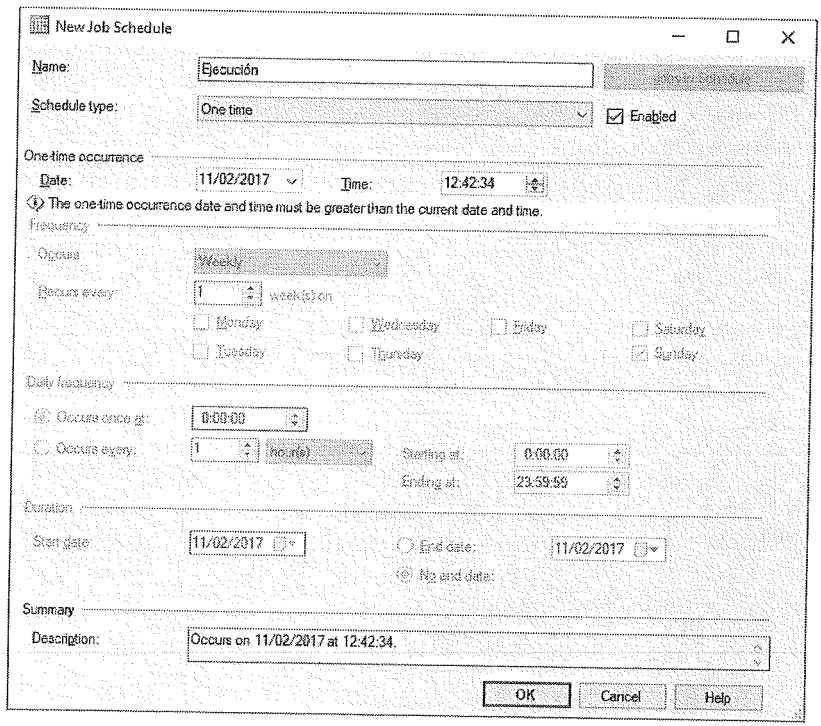
Si se planifica correctamente un trabajo, la ejecución de los trabajos no implicará ninguna sobrecarga para el servidor cuando esté muy solicitado por los usuarios. La planificación del conjunto de todos los trabajos administrativos permite optimizar los tiempos de respuesta del servidor.

4.5 Ejemplo de trabajo

El siguiente trabajo crea una base, una tabla y un usuario de base de datos. Se ha planificado para que se ejecute una única vez. Se avisará a un operador al final de la ejecución del trabajo.



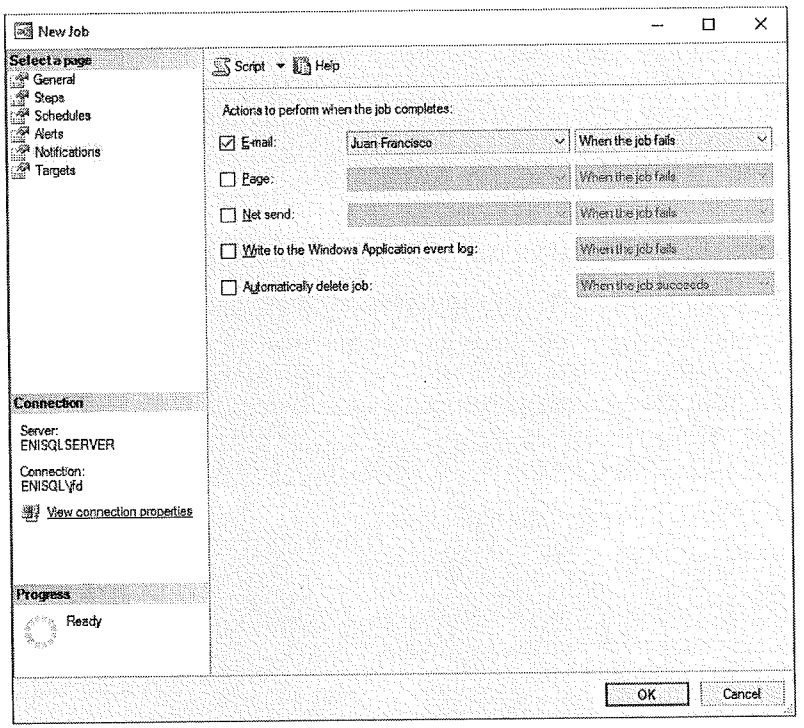

New Job
Select a page
General
Steps
Schedules
Alerts
Notifications
Targets
Script
Help
Job step list:
St...
Name
Type
On Success
On Failure
1
Creación de la base
Transact-...
Go to the ...
Quit the jo...
2
Creación de las tablas
Transact-...
Go to the ...
Quit the jo...
3
Creación de los usuarios
Transact-...
Go to the ...
Quit the jo...
Connection
Server:
ENISQLSERVER
Connection:
ENISQL\jfd
View connection properties
Progress
Ready
Move step:
Start step:
1.Creación de la base
New...
Insert...
Edit
Delete
OK
Cancel



Evidentemente, es posible definir varias planificaciones para el mismo trabajo.

Los trabajos se guardan en la tabla sysjobs de la base de datos msdb. Incluso si es mejor usar la interfaz gráfica o los procedimientos almacenados de gestión de trabajos, algunas veces también es necesario trabajar directamente con esta tabla.

Por ejemplo, una consulta de actualización (UPDATE) sobre esta tabla permitirá modificar fácilmente el propietario de varios trabajos. Esta información se guarda en la columna owner_sid. El identificador de la conexión (sid) propietaria del trabajo se obtiene consultando la columna sid de la vista syslogins, de la base de datos Master.



5. Las alertas

Las alertas se definen con el objetivo de disparar un tratamiento automático para solucionar el problema o advertir a un operador, que será capaz de reaccionar rápidamente para resolver el problema.

5.1 Presentación

Cuando un servidor está funcionando, los errores, mensajes o eventos generados por SQL Server se registran en el Observador de eventos de Windows. El agente SQL Server lee el diario de Aplicación de eventos para buscar la información que puede tratar, comparándola con las alertas que han sido definidas en la tabla **sysalerts** de la base de datos **msdb**.

También se puede disparar una alerta cuando un contador de rendimiento supera un valor límite (fijo). Por último, una alerta puede dispararse tras un evento WMI (*Windows Management Instrumentation*) concreto. En este último caso, el agente SQL Server es un cliente del espacio de nombres WMI y en el momento de definir la alerta es necesario especificar el evento WMI que va a dispararla.

Las alertas asociadas a un error SQL Server son las más frecuentes.

5.1.1 ¿Cómo registrar la información en el diario de Aplicación?

Hay tres tipos de eventos que se pueden registrar en el Observador de eventos:

- Los mensajes con nivel de gravedad superior a 19. Los mensajes se almacenan en la tabla **sysmessages** de la base **Master**. Para obligar a que se registre en el diario un mensaje cuando el nivel de gravedad es inferior a 19, es necesario ejecutar el procedimiento **sp_altermessage**.
- Todas las instrucciones RAISERROR con la opción **WITH LOG**.
- Todo evento registrado con **xp_logevent**.

Observación

El tamaño del diario de aplicación del Observador de eventos de Windows debe ser suficiente como para contener todos los mensajes.

5.1.2 ¿Cómo reacciona el agente SQL Server?

El agente SQL recorre el diario de Aplicación buscando mensajes que provengan de SQL Server. Entonces compara el error con las alertas definidas en la tabla **sysalerts** que está cargada en memoria caché para mejorar los rendimientos.

Esta alerta dispara la ejecución de una tarea planificada o la notificación al operador.

5.2 Gestión de las alertas

Cada alerta tiene un nombre único. Este nombre está limitado a 128 caracteres.

Observación

Solo los mensajes registrados en el Observador de eventos pueden disparar una alerta.

5.2.1 Como respuesta a errores de SQL Server

Para los errores de SQL Server, es posible vincular una alerta bien al número del error o bien a su gravedad.

Si para un evento dado (número de error y nivel de gravedad) es posible ejecutar dos alertas, el agente SQL Server ejecuta la alerta más específica posible; en este caso, aquella asociada al número del error. Como respuesta a un evento, solo se puede disparar una alerta.

Antes de crear una alerta, es necesario tener en cuenta los criterios siguientes:

- El número de error debe estar registrado en el Observador de eventos si se desea que se dispare una alerta.
- El nivel de gravedad puede ser el factor que dispare la alerta. Solo los errores que tienen un nivel de gravedad entre 19 y 25 se registran automáticamente en el Observador de eventos. Los niveles de 20 a 25 corresponden a los errores irrecuperables. Por lo tanto, es necesario definir el operador al que avisar en caso de un error de este tipo. Las alertas que SQL proporciona como ejemplo corresponden a la gestión de estos niveles de gravedad.
- La base de datos: es posible precisar la base de datos origen del evento para especializar las alertas. De esta manera, se pueden crear varias alertas para un mismo número de error.
- Texto del evento: con el objetivo de limitar la ejecución de alertas, es posible precisar el texto que debe contener el mensaje del evento.

5.2.2 La transferencia de eventos

Basándose en un nivel de gravedad, es posible transferir todos los mensajes que tengan un nivel de gravedad superior o igual al que se haya definido hacia otro servidor SQL. La transferencia de los eventos se puede realizar con el objetivo de centralizar el tratamiento de las alertas para un grupo de servidores que ejecuten SQL Server.

Ventajas

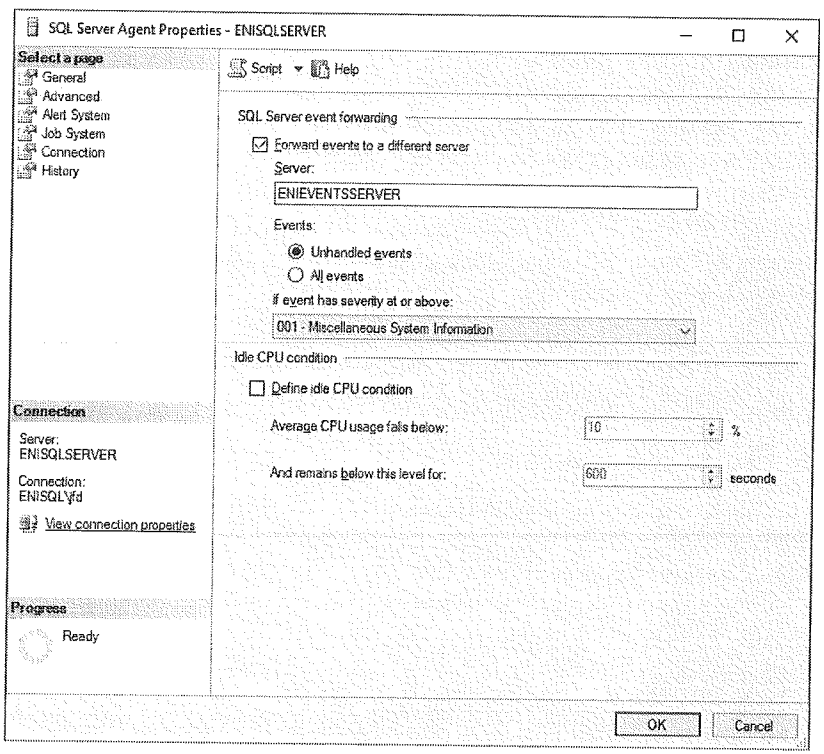
- La centralización permite una gestión simplificada de las alertas.
- La administración de un servidor SQL adicional implica una carga de trabajo inferior para el administrador.
- El tiempo para la puesta en práctica se reduce, ya que todas las alertas se definen una única vez.

Inconvenientes

- La transferencia de eventos aumenta el tráfico de red.
- Punto de fallo único.
- El servidor que trata todas las alertas sufre una carga de trabajo importante y está menos disponible para el tratamiento de los datos que gestiona.

Puesta en marcha

La elección de un servidor de transferencia solo se puede hacer con SQL Server Management Studio, usando la ventana de propiedades del servicio SQL Server Agent.

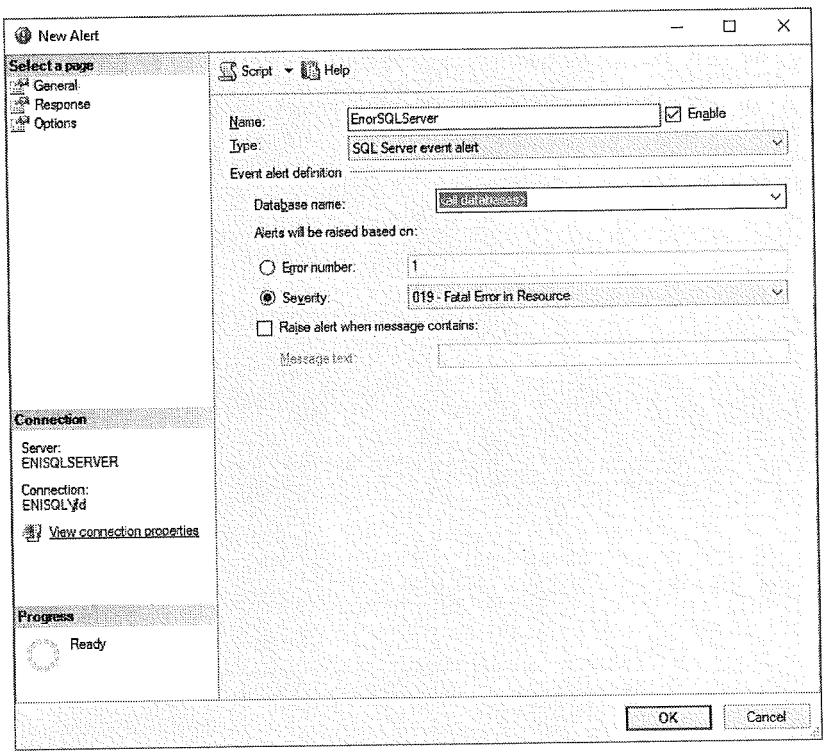


Desde aquí también es posible definir los criterios correspondientes a la inactividad de la CPU. Esta definición es importante porque es posible planificar los trabajos basándose en la inactividad del procesador en lugar de en el tiempo.

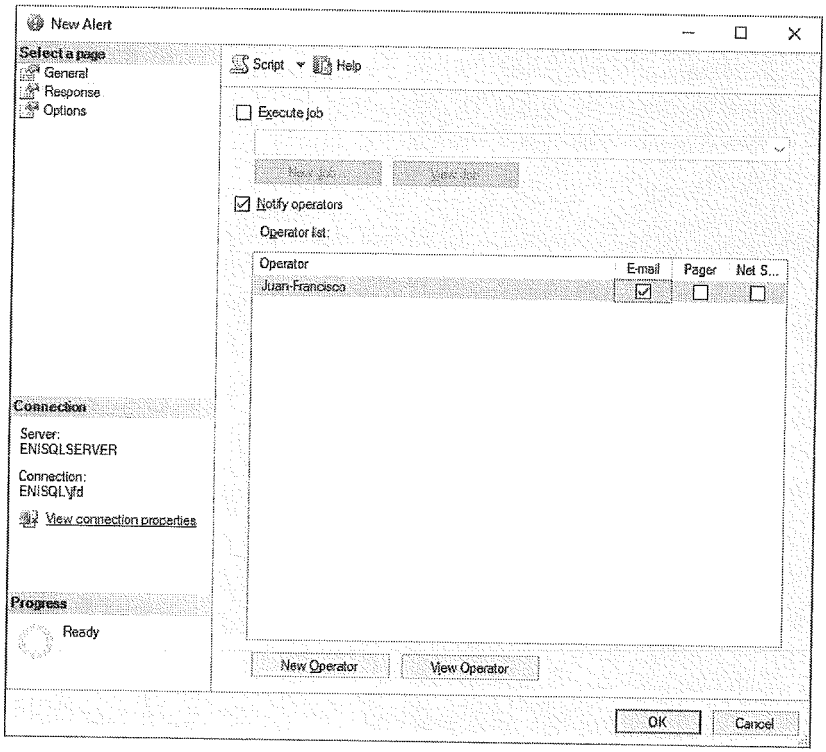
5.2.3 Implantación

SQL Server Management Studio

Para definir una nueva alerta es necesario seleccionar **New Alert** desde el menú contextual asociado al nodo **SQL Server Agent - Operators** del explorador de objetos.



Falta completar este cuadro de diálogo indicando el nombre de la alerta, su vinculación a un determinado número de error o a un nivel de gravedad. Es posible restringir el alcance de la alerta especificando una base de datos o el texto que debe contener el evento.

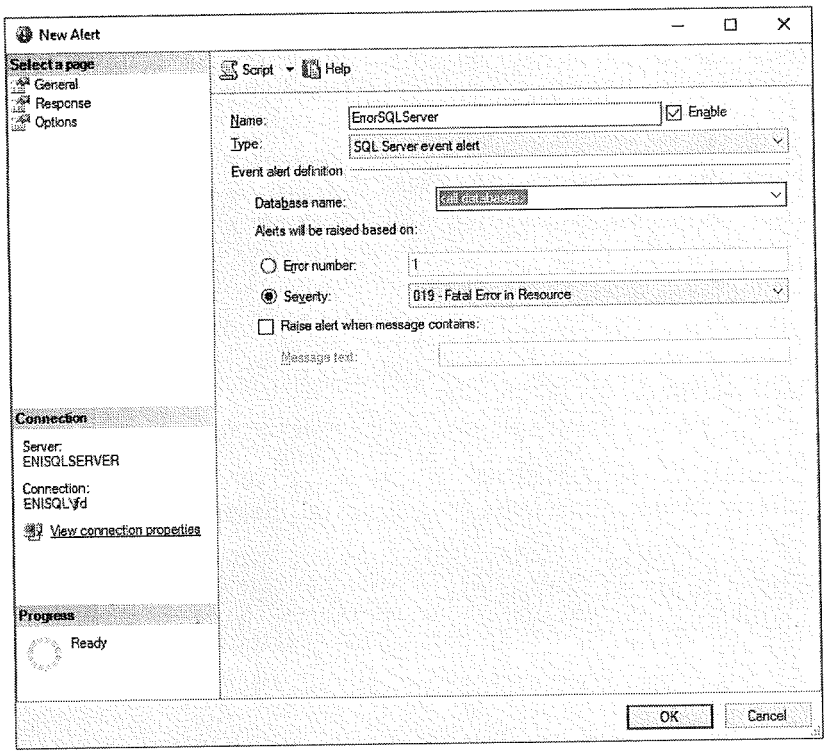


La página **Response** permite precisar lo que debe hacer la alerta.

Es posible indicar un trabajo que se ha de ejecutar y los operadores a los que hay que avisar. Para cada operador, conviene indicar el medio utilizado para el aviso. Las notificaciones a los operadores contienen el texto indicado a nivel de la alerta.

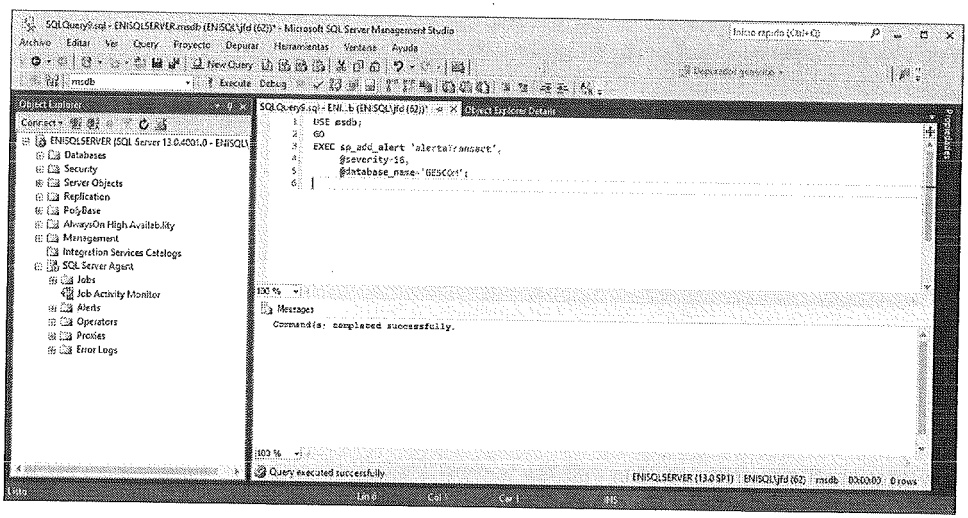
Activación/desactivación de la alerta

Por medio de **Propiedades de la alerta**, es posible desactivarla o reactivarla.



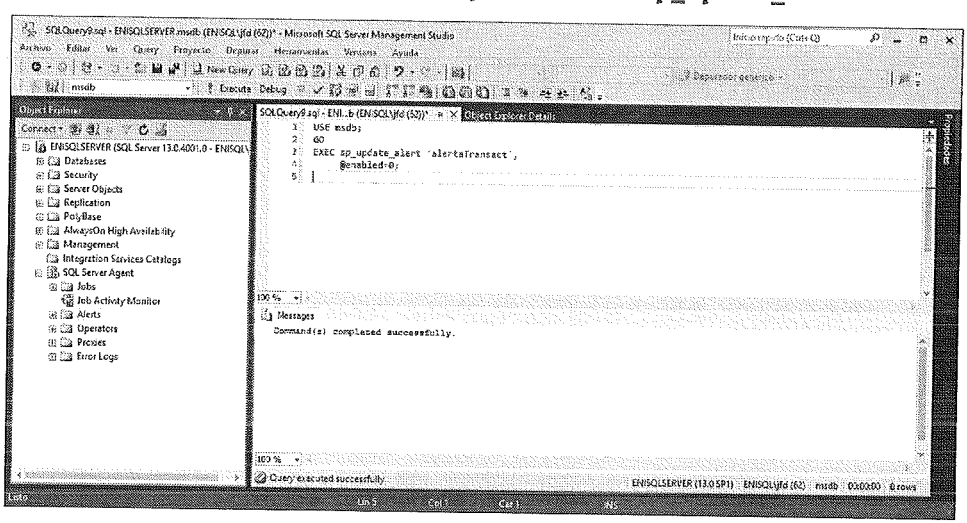
Transact SQL

El procedimiento **sp_add_alert** permite definir nuevas alertas asociadas bien a un número de error o bien a una gravedad. Este procedimiento debe utilizarse mientras se está situado sobre la base **msdb**.



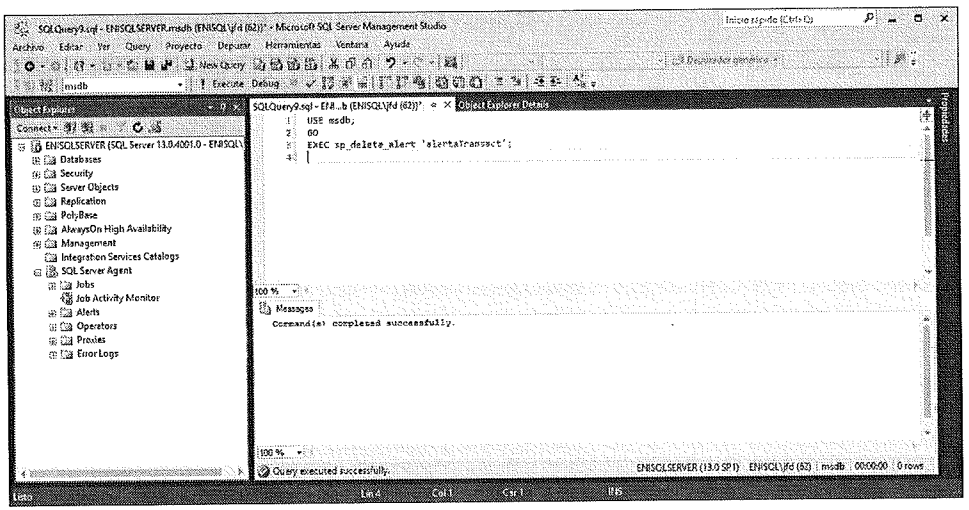
Activación/desactivación de la alerta

Esta operación se efectúa por medio del procedimiento **sp_update_alert**.



Eliminación

Es necesario, en este caso, utilizar el procedimiento **sp_delete_alert**.

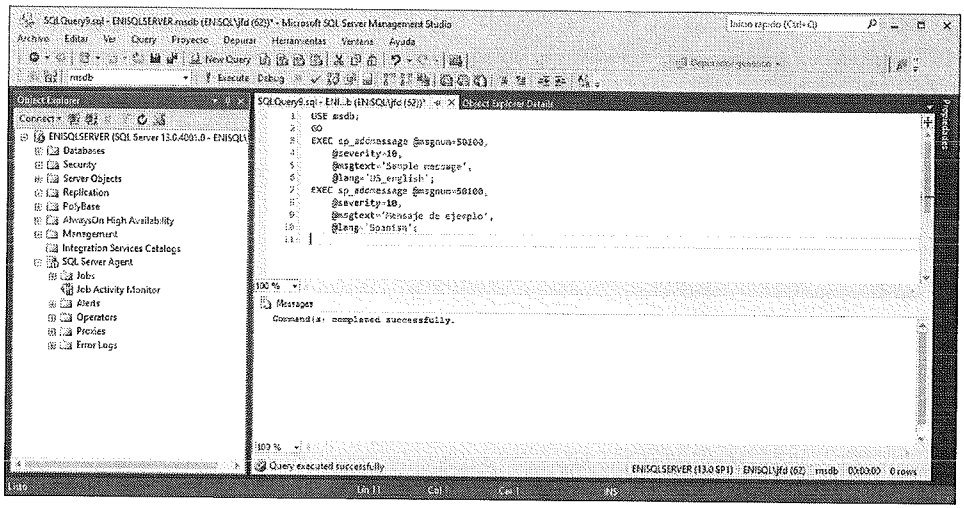


5.2.4 En respuesta a los errores de usuario

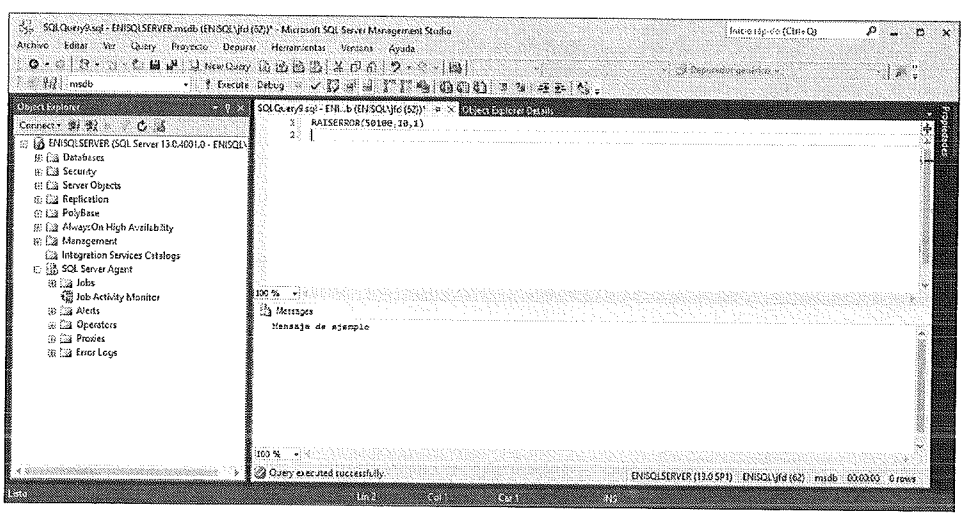
Es posible definir mensajes propios a una aplicación. Estos mensajes se pueden añadir a los mensajes ya definidos en SQL Server. La gestión de estos mensajes se realiza con los tres procedimientos almacenados **sp_addmessage**, **sp_altermessage** y **sp_dropmessage** para crear, modificar o eliminar un mensaje.

Por ejemplo, el procedimiento **sp_addmessage** se ejecuta para definir el mensaje de error que tiene el número 50.001, ya que todos los mensajes definidos por el usuario tienen un número superior a 50.000. Cuando se define el mensaje, también es necesario definir una gravedad, un idioma y un mensaje de error. A este nivel, también es posible especificar si el mensaje se registrará o no en el diario de eventos.

Para poder definir el mensaje en español, hay que crear un mensaje con el mismo número de error, pero en inglés.



Para desencadenar el error a partir de la aplicación de base de datos, es suficiente con ejecutar el comando RAISERROR.



La gestión de alertas utiliza los mismos métodos que se han explicado al abordar la gestión de alertas como respuesta a las alertas de SQL Server.

5.2.5 Como respuesta a umbrales de rendimiento

Es posible definir alertas sobre umbrales de rendimiento. Estas alertas están asociadas a los contadores SQL Server disponibles en el Analizador de rendimiento de Windows.

Se pueden crear alertas sobre los elementos siguientes:

- Método de acceso.
- Administrador del búfer.
- Administrador de caché.
- Base de datos.
- Bloqueos.
- Estadísticas de SQL Server.

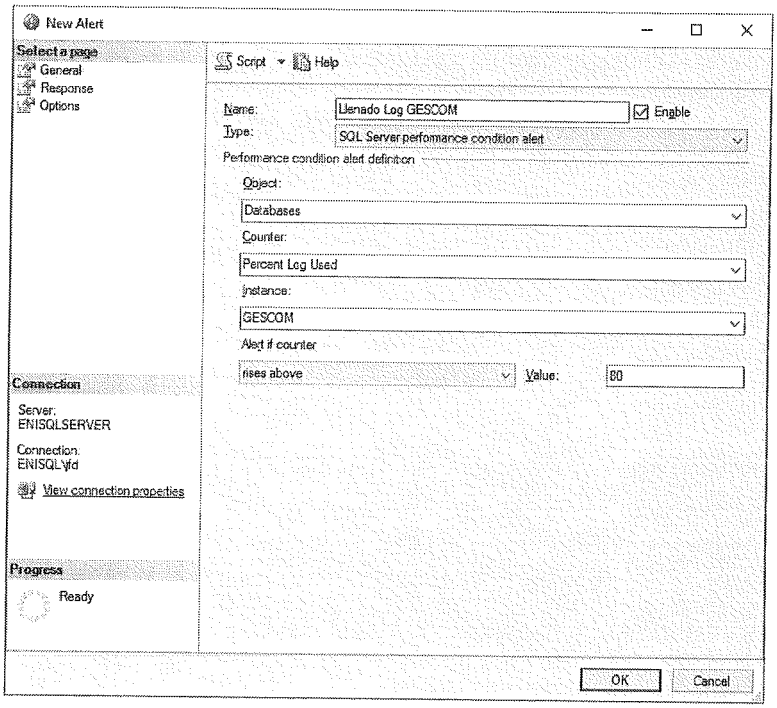
Observación

No es necesario que el Analizador de rendimiento se ejecute sobre el puesto donde está instalado el servidor de SQL.

La definición de las alertas es prácticamente idéntica, aunque, en lugar de asociar la alerta a un número de error o un nivel de gravedad, se asocia a un objeto, un contador, una instancia (si se da el caso) y un valor por encima o por debajo del cual se ejecuta la alerta (los contadores de SQL Server se detallan en el capítulo Herramientas adicionales).

Ejemplo

Definición usando SQL Server Management Studio de una alerta sobre una tasa de ocupación del diario. Esta alerta se notificará a un operador, con el fin de que este último realice una copia de seguridad y después trunque el diario.



Para este tipo de alerta, normalmente es necesario ajustar, a nivel de la página de opciones, el intervalo entre el desencadenamiento de dos alertas. De hecho, cuando el umbral se sobrepasa, es necesario un tiempo importante para prever el procedimiento correctivo para volver a quedar por debajo del umbral definido. En el caso del ejemplo anterior, hay que aumentar el tamaño del log o desencadenar una copia de seguridad del log antes de truncar la parte no activa. Si no se define ningún intervalo antes de un nuevo desencadenamiento de la alerta, el operador de destino se puede ver rápidamente lleno de mensajes.

6. Ejercicio: planificar tareas

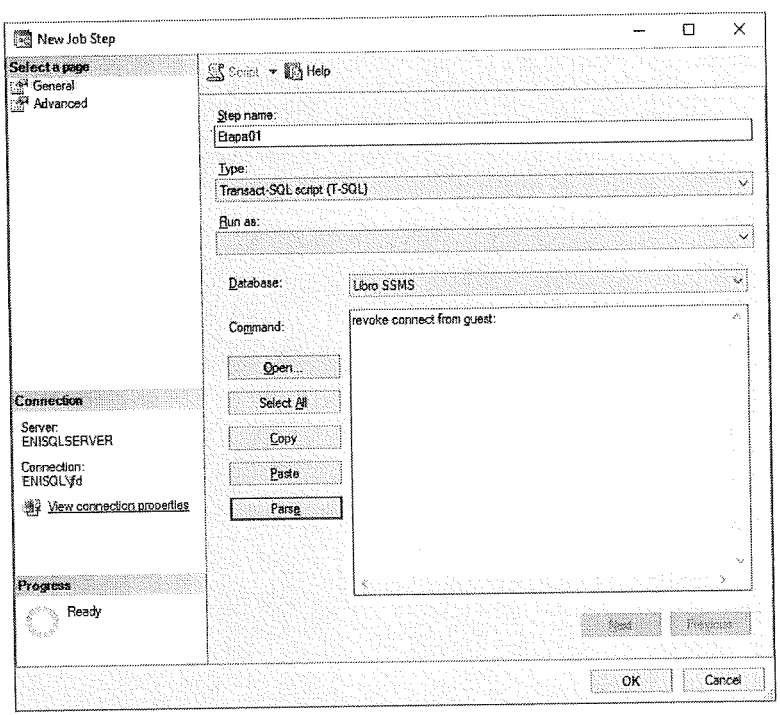
6.1 Enunciado

Defina una tarea planificada DesactivarGuest en la base de datos LibroSSMS que se ejecutará todos los días de la semana (de lunes a viernes) a las 23 horas y cuya finalidad será desactivar la cuenta de invitado para impedir conexiones por usuarios no expresamente autorizados a trabajar en esta base de datos.

6.2 Solución

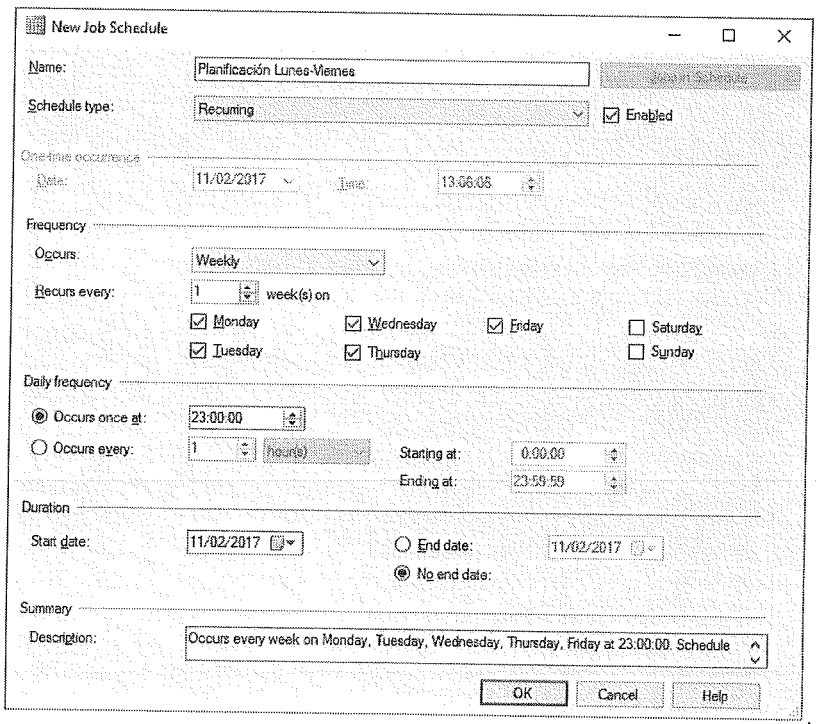
Después de verificar que el servicio SQL Server Agent está arrancado, debemos posicionarnos en la rama **Jobs** y elegir la opción **New Job** desde el menú contextual.

En la sección **General**, indicamos el nombre de la tarea y después nos situamos en la sección **Steps**. Llegados allí debemos crear una nueva etapa (botón **New**) y completarla como sigue:



En caso de duda sobre la sintaxis Transact SQL, el botón **Parse** permite asegurar que el script es sintácticamente correcto.

A continuación debemos situarnos en la sección **Schedules** para programar la ejecución de esta tarea de lunes a viernes a las 23 horas. La siguiente imagen ilustra esta planificación:



Capítulo 6
Transferencia de datos

1. Importación y exportación de datos

1.1 Presentación

La importación de los datos consiste en recuperar los datos desde una fuente externa a SQL Server (archivo ASCII, base de datos de Access...) y almacenar estos datos en una o varias tablas de una base SQL Server. La exportación representa la operación inversa: el contenido de una tabla o el resultado de una consulta se proyecta en un archivo ASCII, o directamente en una base de datos de Access, por ejemplo.

La importación es, en general, una operación puntual que interviene entre la creación y el diseño de una base de datos y la puesta en servicio de la base. En esta etapa de importación, todos los datos que provienen de un sistema de gestión de datos antiguo se integran en SQL Server. Una vez que termina la migración, es posible trabajar con los datos presentes en SQL Server.

Sin embargo, algunas veces la importación puede ser una operación que se realice regularmente. Esto sucede sobre todo cuando la base SQL sirve para editar análisis sobre los datos almacenados en sistemas diferentes.

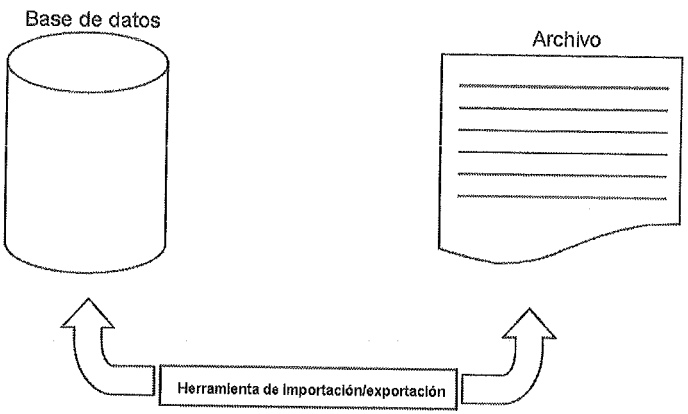
Las operaciones de exportación son por lo general menos frecuentes. Naturalmente, si los datos tienen que ser manipulados desde herramientas como Microsoft Access o Excel, es preferible trabajar directamente sobre los datos almacenados en SQL Server en lugar de realizar una exportación para trabajar con una copia local de los datos. La exportación sigue siendo la solución ideal cuando un usuario desea establecer cálculos sobre los datos trabajando en modo desconectado (en el ordenador portátil, por ejemplo).

SQL Server proporciona varias herramientas de importación y exportación para poder realizar estas operaciones entre fuentes ODBC, OLE DB, hojas de cálculo de Excel y archivos de texto ASCII.

Si los datos almacenados por SQL Server deben ser distribuidos a otros servidores SQL de la empresa, es preferible establecer la replicación de datos, que ofrece más flexibilidad y se encarga de sincronizar las bases de datos replicadas.

Observación

La replicación se abordará en el capítulo Replicación.



Presentación de la transferencia de datos

1.2 Las herramientas

La transferencia de información de una base de datos hacia otra es una operación corriente que debe realizarse de manera rápida y segura. Para responder a las diferentes situaciones de utilización que se puedan presentar, SQL Server ofrece distintas opciones. No todas tienen las mismas características, pero todas tienen en común el hecho de transferir un volumen importante de información de una fuente de datos hacia otra.

1.2.1 SSIS (SQL Server Integration Services)

Con SSIS, SQL Server ofrece mucho más que una simple herramienta de importación y exportación de datos. SSIS permite también definir transformaciones más o menos complejas sobre los datos manejados. ETL (*Extract Transform and Load*) permite trabajar con fuentes de datos externas accesibles mediante OLEDB/ODBC para importar los datos en una base SQL Server realizando las operaciones de adecuación de formato de los datos, como por ejemplo un trabajo sobre fechas.

SSIS es también una herramienta perfectamente adaptada para la alimentación de datos de una base OLAP. La operación se define en forma de trabajo planificado.

1.2.2 Replicación

La replicación de datos permite copiar datos y sincronizarlos con el objetivo de que todas las copias contengan los mismos valores de datos. Es posible establecer la replicación entre SGBDR funcionando sobre la misma red, LAN o WAN, aunque la replicación puede realizarse también a través de Internet.

El establecimiento de los diferentes tipos de replicaciones se detallará en el capítulo Replicación.

1.2.3 BCP

Esta herramienta de línea de comandos permite importar y exportar los datos entre un archivo y SQL Server. Se trata de una herramienta de base de datos que permite realizar rápidamente numerosas operaciones.

1.2.4 SELECT INTO e INSERT

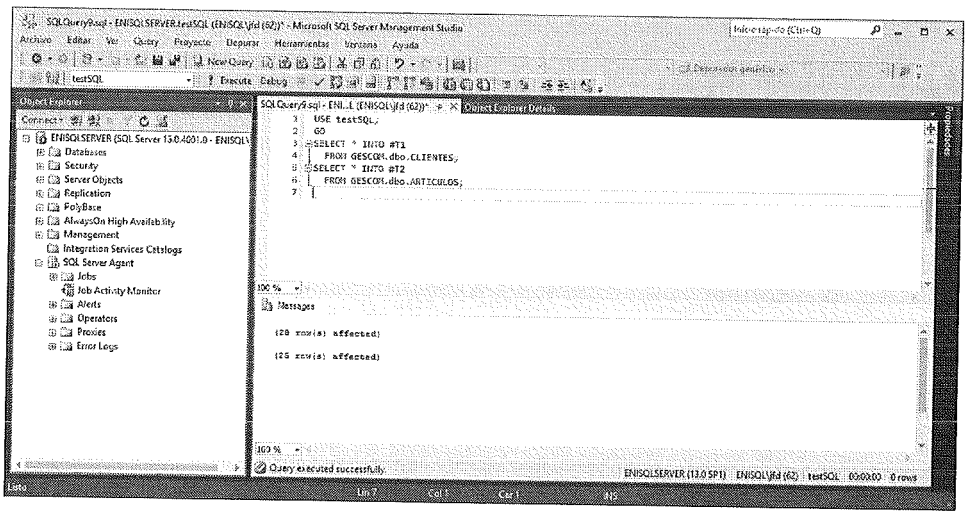
Estas instrucciones SQL permiten, respectivamente, crear una nueva tabla que contenga el resultado de una consulta en la base de datos e insertar datos en una tabla de la base. La utilización de estos comandos se ha detallado en el capítulo Gestión de la base de datos.

Los datos pueden provenir de otro servidor diferente de aquel donde se ejecuta la operación, por lo que se debe proporcionar el nombre completo del objeto de la siguiente forma: Servidor.base.esquema.objeto.

De manera predeterminada, se utilizan los valores:

- Servidor: donde se ejecuta la consulta.
- Base: la indicada por el último comando use o bien la base de datos por defecto del usuario si todavía no se ha ejecutado ningún comando use.
- Esquema: por defecto es **dbo**.

Ejemplo



1.2.5 Los criterios de selección

La selección de una herramienta depende de ciertos criterios. Los más habituales son:

- El formato de los datos.
- La ubicación de los datos.
- La frecuencia de la operación: ¿se trata de una operación puntual o va a desarrollarse de manera regular?
- ¿Utilización de una herramienta en línea de comandos o que posea una interfaz gráfica?
- El rendimiento.

Funcionalidades	SSIS	Replicación	BCP	SELECT INTO INSERT
Importación				
- de datos de texto	Sí		Sí	Sí
- fuente de datos ODBC	Sí	Sí		
- fuente de datos OLEDB	Sí	Sí		Sí

Funcionalidades	SSIS	Replicación	BCP	SELECT INTO INSERT
Exportación				
- de datos de texto	Sí		Sí	
- fuente de datos ODBC	Sí	Sí		
- fuente de datos OLEDB	Sí	Sí	Sí	
Interfaz				
- gráfica	Sí	Sí	Sí	
- línea de comandos			Sí	Sí
Planificación posible	Sí	Sí	Sí	
Transformación de datos	Sí			
Rendimiento máximo		Sí	Sí	
Operación puntual	Sí		Sí	Sí
Operación regular	Sí	Sí	Sí	

2. La herramienta BCP

BCP (*Bulk Copy Program*) es una potente herramienta de línea de comandos. Es muy conocida entre los usuarios de las versiones anteriores de SQL Server. Se utiliza cuando el volumen de las transferencias entre archivos de texto y la base SQL Server es elevado.

BCP permite exportar los datos de una tabla o de una consulta SQL hacia un archivo de texto o bien importar un archivo de texto en una tabla. Cuando se utilice, es necesario precisar la fuente y el destino de los datos, así como un nombre de usuario y una contraseña para conectarse al servidor.

La utilización de BCP no necesita un dominio especial de Transact SQL, salvo en caso de que la exportación se base en una consulta SQL. Por el contrario, la descripción del archivo de datos es un requisito obligatorio.

2.1 La sintaxis

Al principio puede parecer algo pesada, pero es posible fijar muchas opciones, lo que añade mucha flexibilidad cuando se usa **bcp**.

```
bcp {nombre_completo_objeto|"consulta"}
{in|out|queryout|format} archivo_de_datos
[-m máximo_de_errores] [-f archivo_de_formato] [-x]
[-e archivo_de_errores] [-d nombreBaseDatos]
[-F primera_línea] [-L última_línea]
[-b tamaño_del_lote_de_instrucciones]
[-n] [-c] [-w] [-N] [-V {80|90|100}] [-q] [-n]
[-C página_de_código]
[-t fin_de_campo] [-r fin_de_línea]
[-i archivo_de_entrada] [-o archivo_de_salida]
[-a tamaño_del_paquete] [-k ReadOnly]
[-S nombre_del_servidor] [-U id_conexión]
[-P contraseña] [-q]
[-T] [-v] [-R] [-k] [-E] [-h "opción [,...n]"]
nombre_completo_objeto|"consulta"
```

Se trata de dar el nombre completo del objeto (tabla o vista) o una consulta cuyo resultado se exportará a un archivo.

En una exportación es posible tomar como objeto una tabla, una vista o solicitar la ejecución de una consulta. Los usuarios deben tener los derechos de selección apropiados.

En caso de una importación de datos, no se puede realizar a través de una consulta. Algunas veces es posible realizar las inserciones a través de una vista. Para ello se deben tomar algunas precauciones en el momento de la definición de la vista última. Durante la importación es necesario que el usuario sea miembro del rol **db_owner**.

```
{in|out|queryout|format} archivo_de_datos
```

Las opciones indicadas son una operación de exportación (`out`) o importación (`in`) de datos. Si se especifica una consulta en lugar de un nombre de objeto, hay que utilizar `queryout`. Por último, `format` permite crear un archivo de formato siguiendo las opciones especificadas; se podrá usar la opción `f`.

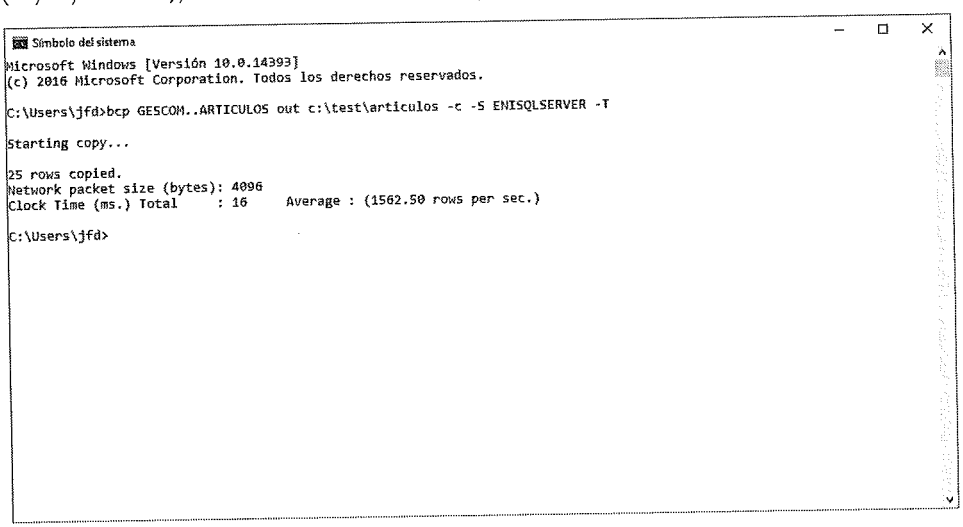
Las otras opciones permiten ajustar el tipo del archivo (texto...), los delimitadores de campos y de líneas...

-S, -U, -P y -T

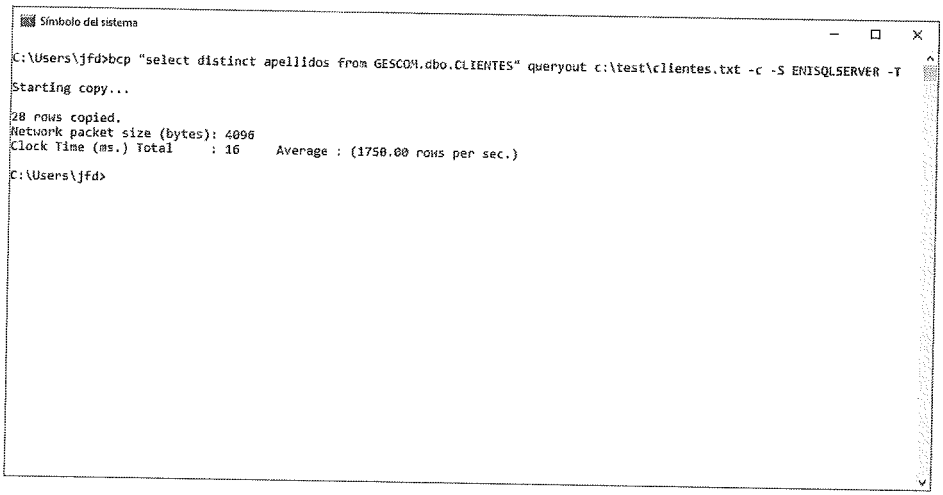
Estas cuatro opciones se encuentran en numerosas herramientas de línea de comandos tales como isql. Permiten conectarse a un servidor SQL y abrir una conexión. El nombre del servidor lo indica la opción **-S**. Además, es posible abrir una sesión utilizando una autenticación de Windows (**-T**) o una autenticación de SQL Server. Es necesario proporcionar el nombre de la conexión (**-U**) y la contraseña (**-P**).

2.2 El uso de bcp en modo interactivo

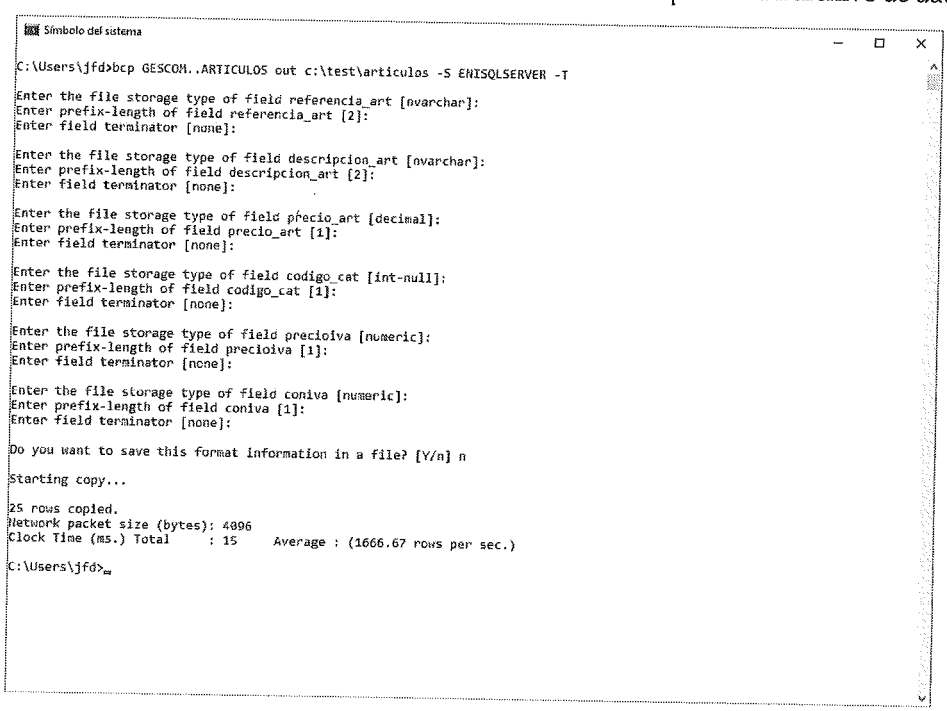
Si cuando se ejecuta el comando **bcp** no se especifica alguna de las siguientes opciones (**-n**, **-c**, **-w** o **N**), entonces la herramienta pregunta de manera interactiva al usuario.



En el ejemplo anterior, los datos contenidos en la tabla **articulos** de la base **GESCOM** se exportan al archivo articulos.txt. Gracias a la opción `-c`, los datos se exportan en modo carácter. La base se almacena en el servidor IvanMDW81 y se utiliza una autenticación Windows (opción `-T`) para conectarse al servidor.



En el siguiente ejemplo, el resultado de una consulta se exporta a un archivo de datos.



Utilización de bcp en modo interactivo

Durante la importación de datos con las herramientas de copia por bloques (**bcp** y **bulk copy**), se deshabilitan los triggers de tipo **instead of** e **insert**. Es posible solicitar su ejecución eliminando la opción **FIRE TRIGGERS**. En este caso hay que asegurarse de que los triggers son capaces de tratar un conjunto de registros, ya que solo se habilitan una vez por bloque.

Para facilitar la descripción de los datos, es posible utilizar un archivo de formato. Si no existe durante la primera importación, **bcp** pregunta con el objetivo de generar el archivo de formato bcp.fmt.

3. SSIS

3.1 Presentación

Cada vez es más importante consolidar en un punto central los datos repartidos en diferentes lugares y después mostrarlos en un formato u otro. El principal problema para este tipo de operación de centralización de datos es el formato de los datos. Además, cuando la información está repartida en diferentes sistemas y servidores, hay muy pocos cambios que se hagan en el mismo formato. Con SQL Server Integration Services, es posible importar, exportar y transformar los datos entre varias fuentes heterogéneas.

SSIS también se puede utilizar para exportar los datos desde la base de datos hacia una herramienta de análisis. Por ejemplo, los datos integrados con SSIS se pueden consolidar a nivel de la base de datos y después se puede llamar de nuevo a SSIS para exportarlos a un archivo de Excel.

SSIS es una herramienta destinada principalmente al análisis decisional (*Business Intelligence*). En las bases de datos OLAP existe un uso simplificado de esta herramienta. Este es el comportamiento que se presenta aquí, ya que permite hacer operaciones de importación y exportación de datos entre diferentes formatos.

SSIS es un ETL que permite almacenar una secuencia de operaciones de extracción y transformación de los datos para que se respeten las restricciones que imponga el destino de los datos. Por último, la información se carga en el destino. Como estas operaciones pueden ser largas y costosas de implementar, con SSIS se puede hacer una copia de seguridad del conjunto en un lote o paquete SSIS.

Observación

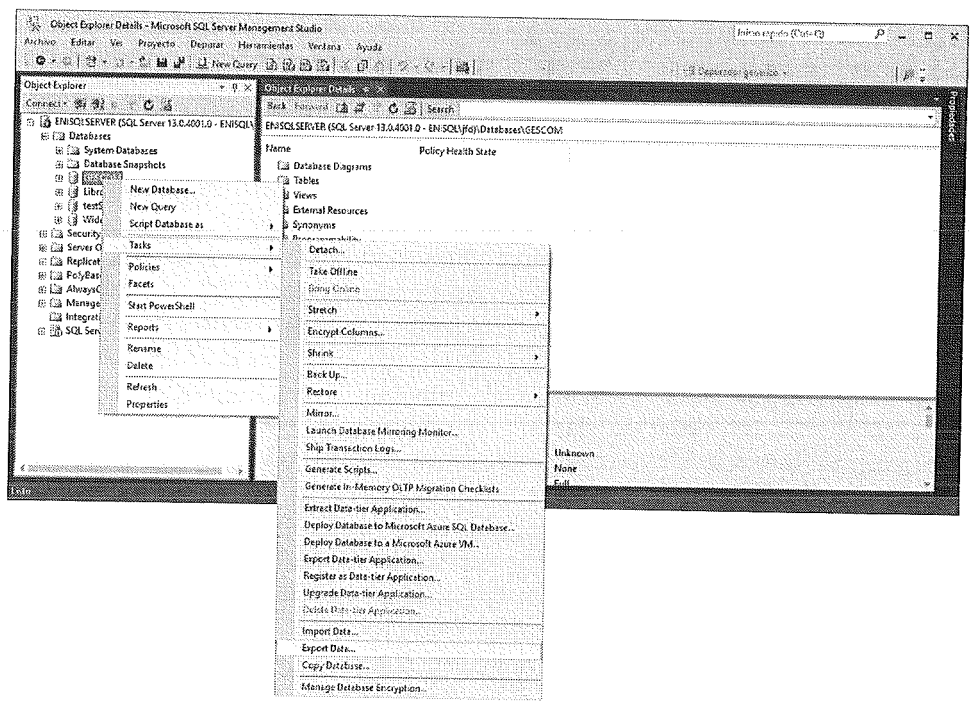
Para poder realizar las operaciones de transferencia de datos, es necesario tener el permiso de lectura (SELECT) sobre la fuente de datos y ser propietario de la base de datos de destino.

3.2 Asistentes de importación y exportación

SQL Server Management Studio dispone de asistentes de importación y de exportación de datos con el objetivo de definir paquetes SSIS simples. Estos paquetes responden a la mayor parte de las operaciones de transferencia de datos deseadas sobre una base de datos OLTP, ya que hacen posible tanto la extracción como la integración de datos externos.

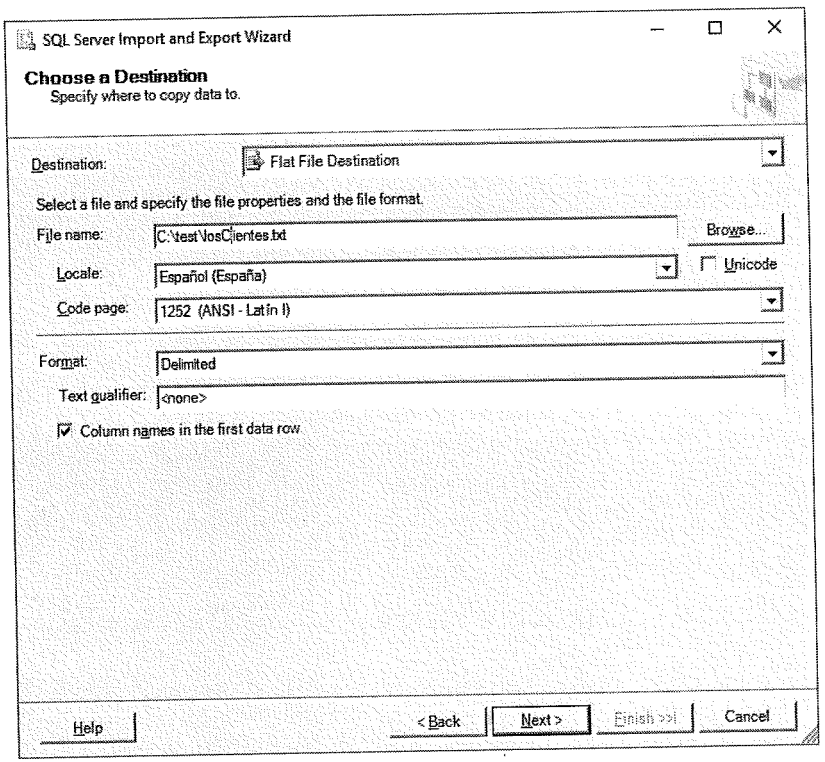
Las principales funciones propuestas por el asistente son:

- La transferencia de datos entre bases de datos heterogéneas.
- La transferencia de objetos entre dos bases SQL Server.
- La creación dinámica del destino de la transferencia.
- La transformación de los datos.

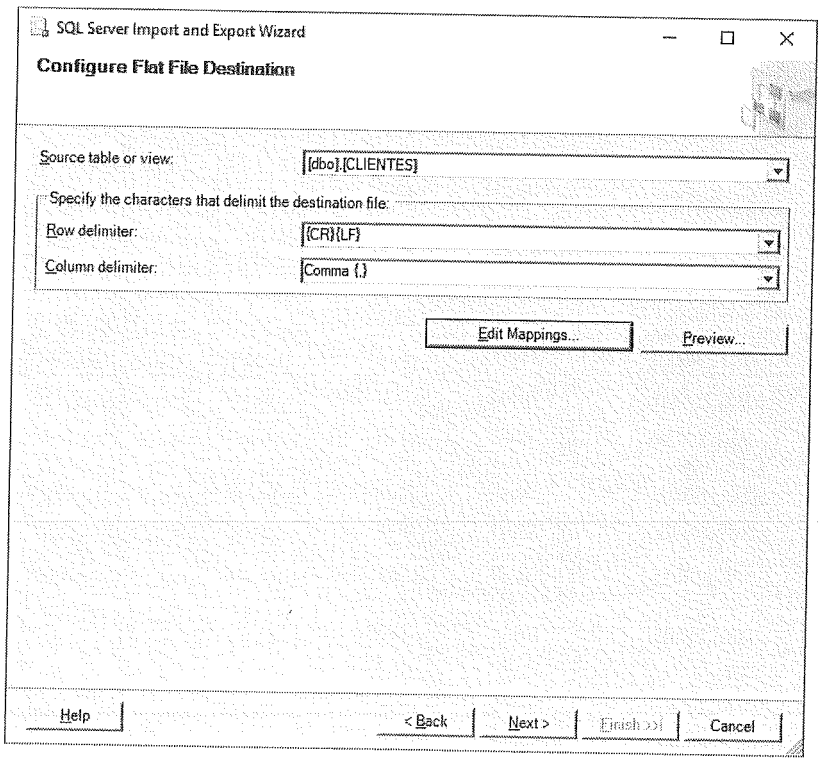


En un primer momento, el asistente va a solicitar la fuente y el destino de los datos.

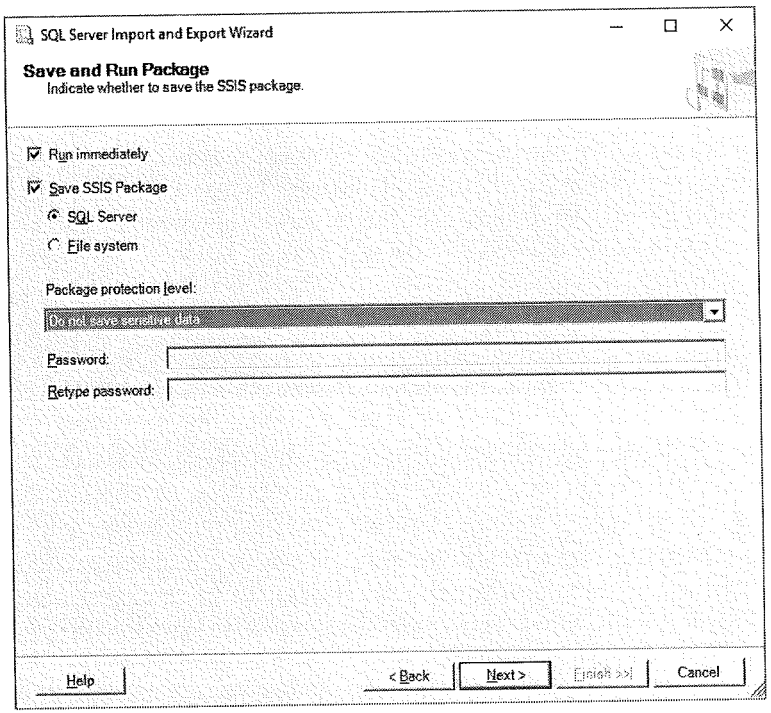
En el ejemplo que se presenta a continuación, la fuente corresponde a la base GESCOM y el destino es un archivo de texto.



El asistente solicita a continuación seleccionar la tabla o bien introducir la consulta que va a proporcionar la información que se ha de almacenar en el archivo. También es posible a este nivel definir las transformaciones y especificar el formato exacto del archivo de texto.



Por último, el asistente permite almacenar el paquete tanto en SQL Server como en forma de archivo.



Sistema de archivos

Eligiendo este tipo de copia de seguridad, toda la definición del paquete y su parametrización se guarda en un archivo fuera de cualquier instancia de SQL Server. Esta solución es muy flexible en términos de despliegue de paquetes. A nivel de copia de seguridad, es necesario consultar el archivo MsDtsSrvr.ini.xml de configuración del servicio SSIS para conocer la lista de directorios que controla el servicio.

Base de datos msdb

Usando esta base de datos de sistema para almacenar la definición de los diferentes paquetes, es posible basarse en la política de copia de seguridad de los paquetes. Sin embargo, la parametrización de los lotes se guarda en archivos externos a la base de datos.

4. Adjuntar y separar una base de datos

Es posible mover fácil y rápidamente los archivos de datos y los logs de una base de datos SQL Server entre dos instancias SQL Server. El formato de los archivos de datos es el mismo, tanto si el servicio SQL Server se ejecuta en modo 32 bits como si lo hace en 64 bits.

Para transferir estos archivos, hay que proceder en dos etapas: separar la base de la instancia donde se ejecuta y adjuntar los archivos que forman la base a la nueva instancia. Estas operaciones se desarrollan fácilmente si se respetan algunos criterios que se indican a continuación a nivel de la separación de una base de datos.

Es posible migrar una base de datos de una version a otra de SQL Server. Durante esta migración, la indexación de texto completo se puede reconstruir o importar en función del valor asignado al argumento de nivel `server upgrade_option`. La importación solo será posible si los archivos son accesibles durante la asignación. En caso de una reconstrucción, es importante no subestimar el tiempo necesario para la reconstrucción de los índices de texto completo.

En todos los casos, después de la migración de la base de datos, el nivel mínimo de compatibilidad es de 100, porque es el nivel mínimo soportado por SQL Server 2014.

El hecho de mover una base de datos de una instancia a otra, va a implicar modificaciones en muchos metadatos y esto hace que se corra el riesgo de sufrir efectos importantes en la correcta utilización de la base. Por ejemplo, todas las tareas planificadas que forman parte de la correcta administración de la base se almacenan en la base de datos MSDB y el movimiento de una base de usuario no implica la migración de sus trabajos.

Las conexiones definidas a nivel de la base se asocian a los logins del lado servidor. Durante el movimiento de la base, los logins no se mueven y, por tanto, las conexiones definidas en la base ya no están asociadas a los logins existentes.

4.1 Separación de una base de datos

Una base de datos normalmente se puede separar, pero también existen algunas restricciones.

Si la base participa en una replicación, no es posible separarla. En este caso, hay que preparar la base con el procedimiento almacenado `sp_replicationdboption`.

Si la base participa en una replicación en espejo, entonces no es posible separar la base. En primer lugar, hay que borrar la base del proceso de replicación en espejo.

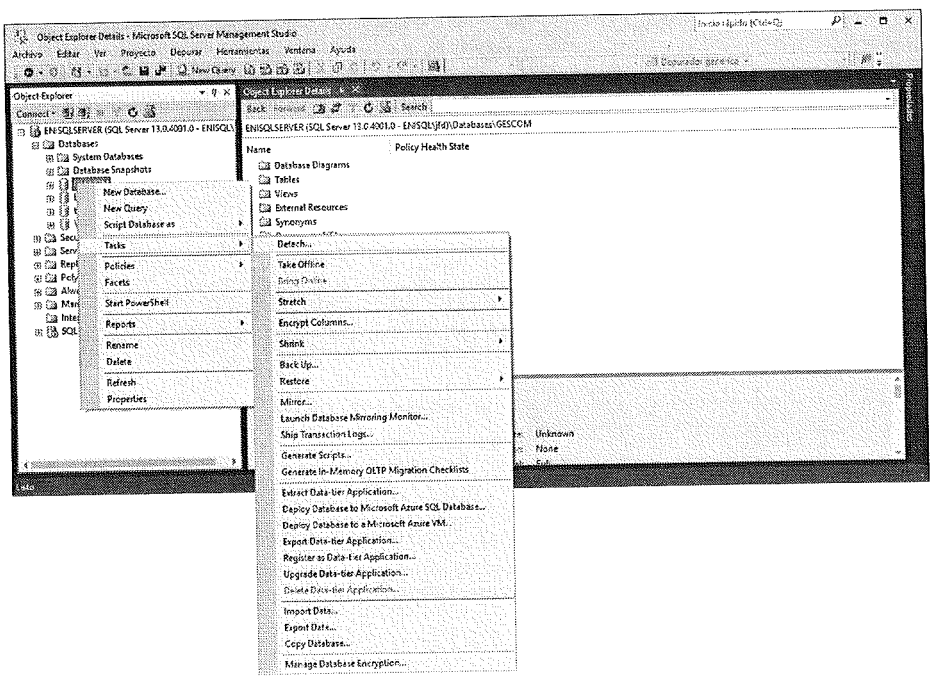
Si un snapshot existe en la base, entonces hay que eliminar esta copia.

El proceso de separación solo es posible para las bases de datos de usuario. No es posible separar las bases de datos de sistema, incluida MSDB. Igualmente, para poder separarla, la base de datos debe estar en línea, es decir, estar abierta a los diferentes usuarios que quieran trabajar en ella.

Para separar una base de datos, el método más sencillo consiste en usar el menú desde SQL Server Management Studio. Durante la separación, la interfaz gráfica ofrece dos opciones que corresponden a la eliminación de las conexiones (lo que permite desconectar a los usuarios todavía conectados a la base) y actualizar las estadísticas.

Ejemplo

La base de datos GESCOM se separa.



4.2 Adjuntar una base de datos

Durante la unión de una base, todos los archivos deben estar disponibles. Si la ruta física de acceso a los archivos se modifica, lo que es frecuente, entonces es necesario indicar la ubicación exacta de cada archivo durante la unión. El archivo de traza también se debe mover, aunque algunas veces es necesario proporcionar los archivos de traza de la base separada.

Por tanto, es conveniente conservar todos los archivos (datos y logs) que forman la base de datos separada.

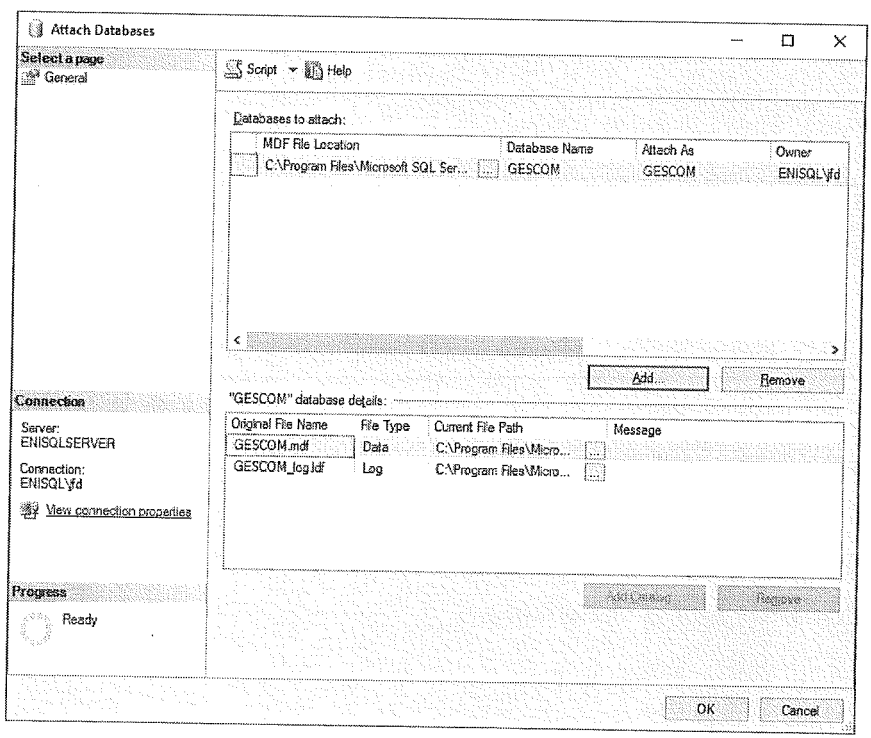
Durante esta operación, si el archivo MDF está en modo de solo lectura en el disco, entonces la base está accesible en modo de solo lectura, incluso si los datos de usuario están almacenados en los archivos NDF que son accesibles en modo de lectura/escritura.

Para tener éxito en esta operación, es posible usar la instrucción SQL `CREATE DATABASE` con la opción `FOR ATTACH` o bien usar SQL Server Management Studio. La operación en SQL Server Management Studio se ilustra a continuación.

Para adjuntar una base de datos, hay que seleccionar la opción **Attach** en el menú contextual del nodo **Databases** del explorador de objetos.

Ejemplo:

La base de datos GESCOM está unida a la instancia actual.



Observación

En caso de que la base de datos que se mueve esté cifrada, es necesario que el propietario de la base la abra con la instrucción `OPEN MASTER KEY DECRYPTION BY PASSWORD='contraseña'`. *Esta operación puntual se debe completar con la descodificación automática de los datos, para que la base sea más fácilmente accesible. Esto se hace con la instrucción* `ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY`.

Capítulo 7
Replicación

1. Presentación

La replicación es una poderosa funcionalidad de SQL Server que permite distribuir datos y ejecutar los procedimientos almacenados sobre varios servidores de la empresa. La tecnología de replicación ha evolucionado considerablemente y ahora permite copiar, trasladar los datos a diferentes lugares y sincronizarlos automáticamente. La replicación puede establecerse entre las bases de datos residentes sobre el mismo servidor o sobre servidores diferentes. Los servidores pueden estar sobre una red local (LAN) o global (WAN) o sobre Internet.

SQL Server distingue dos grandes categorías de replicación:

- La replicación de servidor a servidor.
- La replicación de servidor a clientes.

En el caso de replicación de servidor a servidor, la replicación permite una mayor integración o aproximación de los datos entre varios servidores de base de datos. El objetivo de este tipo de replicación es efectuar un intercambio de información entre servidores de base de datos. Los usuarios que trabajan sobre las bases de datos que participan en la replicación pueden, de esta manera, consultar datos de mayor calidad.

La replicación de servidor a cliente afecta principalmente a los usuarios desconectados de la red de la empresa que desean trabajar con todos los datos de la empresa o parte de ellos. Los usuarios trabajan con una aplicación específica y utilizan SQL Server como servidor de base de datos local. Cuando los usuarios se conectan a la red, la sincronización de los datos entre sus puestos y la instancia de SQL Server central la realiza el mecanismo de replicación de SQL Server.

La gestión de la replicación se ha simplificado para permitir un establecimiento y un mantenimiento más fáciles. Para las soluciones más complejas y que necesiten una integración completa a un programa, SQL Server ofrece API RMO (*Replication Management Object*). Esta API permite manipular mediante programación todos los elementos de la replicación. API RMO está disponible para los lenguajes que se basan en el framework .NET.

El concepto de esquema que permite un agrupamiento lógico de los objetos en la base de datos es tomado en cuenta por la replicación con la posibilidad de realizar cambios de esquemas.

2. Las necesidades para la replicación

La replicación es una tecnología compleja y no puede existir una solución única para cubrir todas las necesidades. SQL Server ofrece diferentes tecnologías de replicación que se pueden adaptar y combinar para responder lo más fielmente posible a las necesidades de las aplicaciones. Cada tecnología tiene ventajas e inconvenientes. Los tres criterios principales para seleccionar una tecnología de replicación son:

- La coherencia de los datos replicados.
- La autonomía de los sitios.
- El particionamiento de datos para evitar conflictos.

No es posible optimizar los tres criterios al mismo tiempo, de manera que una solución que favorezca la coherencia de los datos deberá dejar poca autonomía a los sitios para averiguar en todo momento el conjunto de modificaciones que tienen lugar sobre los datos.

2.1 Coherencia de los datos replicados

Existen dos tipos principales de coherencia:

- La homogeneidad de las transacciones.
- La convergencia de los datos.

La coherencia de las operaciones distribuidas, como la replicación, es mucho más complicada de mantener en comparación con la coherencia de las transacciones locales, por lo que es suficiente con respetar el test ACID (Atomicidad, Coherencia, Aislamiento y Durabilidad).

La homogeneidad de las transacciones en la replicación obliga a que los datos sean idénticos en todos los sitios que participan en la replicación, como si la transacción fuera a ejecutarse sobre todos los sitios.

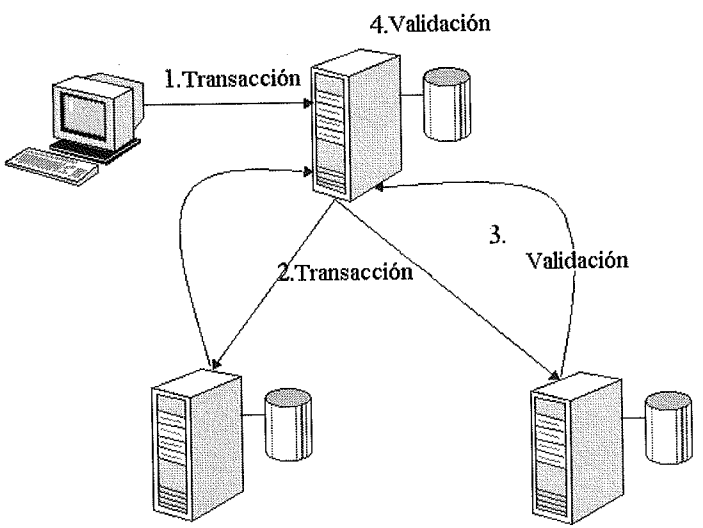
La convergencia de los datos significa que todos los sitios que participan en las replicaciones deben tener el mismo juego de datos, que no es necesariamente el que se obtendría si todas las replicaciones se hubieran desarrollado sobre el mismo servidor.

2.1.1 Coherencia de las transacciones

En todos los casos, los sitios contienen un juego de valores idéntico a aquel sobre el que han hecho o se podían haber hecho todas las operaciones de modificación.

Coherencia transaccional inmediata

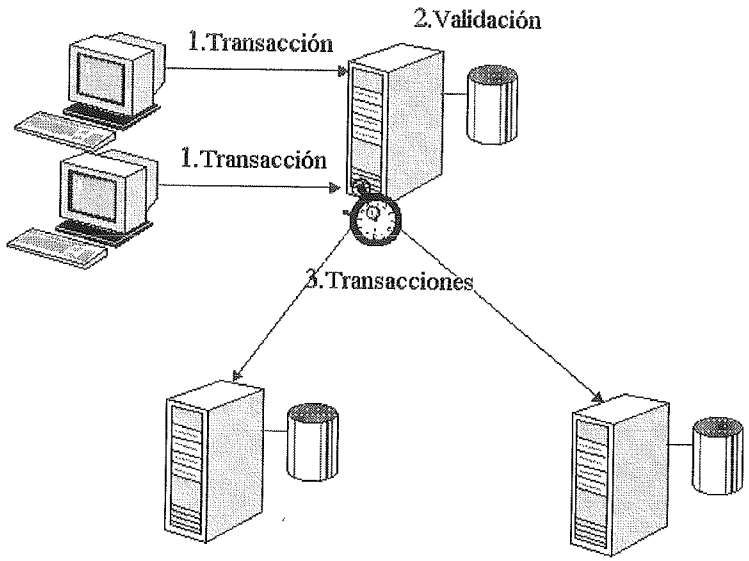
Con esta coherencia, todos los sitios que participan en la replicación tienen la garantía de ver siempre los mismos valores en el mismo momento. Para asegurar la coherencia transaccional, SQL Server dispone de un protocolo de validación en dos fases con todos los sitios participantes. Las modificaciones se efectúan en todos los sitios o en ninguno. Esta solución está muy limitada en la realidad, ya que los problemas de red prohíben toda validación de una transacción mientras el servidor no se conecte de nuevo a la red.



En el ejemplo anterior, el cliente efectúa una transacción sobre el servidor al que está conectado. Esta transacción solo se validará si se ha ejecutado con éxito en todos los servidores que participan en la replicación.

Coherencia transaccional latente

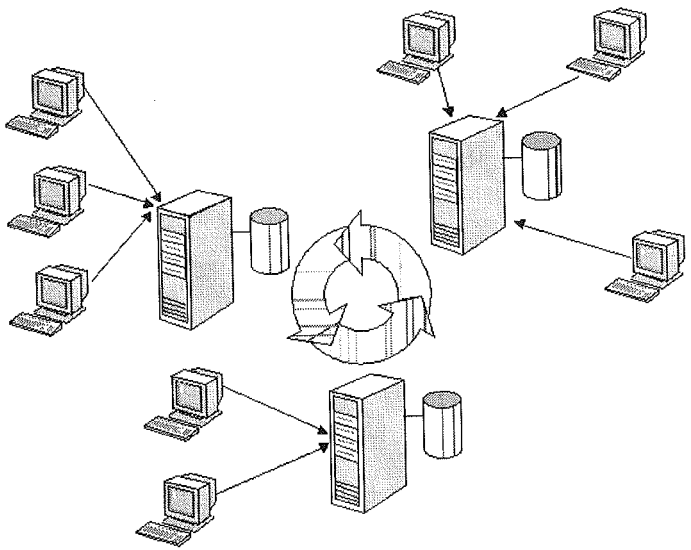
La coherencia latente de las transacciones garantiza que todos los participantes obtendrán los mismos valores que los contenidos en el sitio de publicación en un momento dado. Puede transcurrir un período de tiempo entre el instante en que se efectúa la transacción en el servidor de publicación y el instante en que las modificaciones se reflejan en los otros sitios.



En este ejemplo, el cliente envía una transacción al servidor y esta se ejecuta de manera local inmediatamente. Después, de forma periódica, los servidores que participan en la replicación repiten localmente el conjunto de transacciones efectuadas sobre el servidor principal.

2.1.2 Convergencia de los datos

Con este proceso, todos los sitios acaban obteniendo el mismo juego de datos, lo que no habría sido posible si todas las modificaciones se hubieran realizado sobre un único servidor. Todos los sitios evolucionan libre e independientemente, unos de otros. La convergencia de los datos se establece con ayuda de la replicación de fusión que tiende a llevar a todos los sitios que participan a gestionar el mismo juego de datos.



Todos los servidores son accesibles en lectura/escritura y las transacciones se ejecutan localmente. Entonces el proceso de replicación realiza las transacciones sobre los otros servidores teniendo en cuenta las modificaciones que han podido intervenir localmente.

2.2 Autonomía de los sitios

La autonomía de los sitios se mide por el impacto que tiene una operación efectuada sobre un sitio en el resto de los sitios. La autonomía es perfecta cuando un sitio puede evolucionar totalmente de manera libre e independiente. El sitio autónomo no se preocupa por las operaciones que pueden intervenir en los otros sitios. Por ejemplo, cuando la replicación garantiza la convergencia de los datos, la autonomía de los sitios está en su máximo, ya que cada servidor SQL puede evolucionar libremente con relación a los otros sitios. A la inversa, la coherencia transaccional inmediata impone una autonomía casi nula de los sitios que participan en ella, ya que una transacción debe ser aprobada por todo el mundo antes de ser validada. Si un solo servidor no puede, entonces la transacción no se valida en ninguno.

2.3 Particionamiento de los datos

Es posible repartir los datos sobre varios sitios con el objetivo de que cada sitio trabaje con su propio juego de datos, estrictamente distinto de los demás. De esta manera, las transacciones que intervienen sobre cada sitio solo ponen en juego los datos del sitio y la coherencia global se conserva.

Si, por ejemplo, cada agencia tiene un archivo de clientes sobre una zona geográfica bien determinada, un cliente solo puede ser gestionado por una única agencia y toda fuente de conflicto queda excluida.

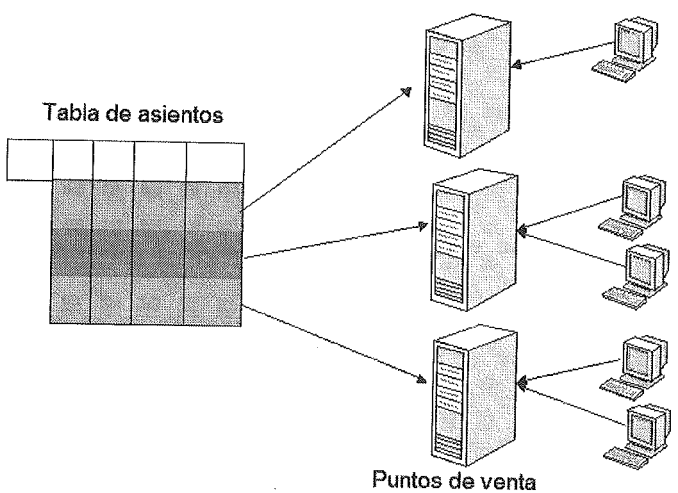
El particionamiento de los datos permite evitar todo conflicto de datos, lo que es preferible, ya que la resolución de los conflictos es un proceso pesado que demanda mucho tiempo de máquina. Cuanto más numerosos son los conflictos, más difícil es gestionar la situación.

El particionamiento de los datos permite funcionar con una coherencia de datos latente, ya que cada sitio solo modifica su propio juego de datos. La aplicación de esta coherencia es menos laboriosa que la coherencia transaccional inmediata, que se basa en un proceso de validación en dos fases.

Observación

Este tipo de particionamiento no se debe confundir con el particionamiento de tablas. En el marco de la replicación, se define una partición lógica, mientras que en el marco de una tabla particionada se define una partición física de la tabla.

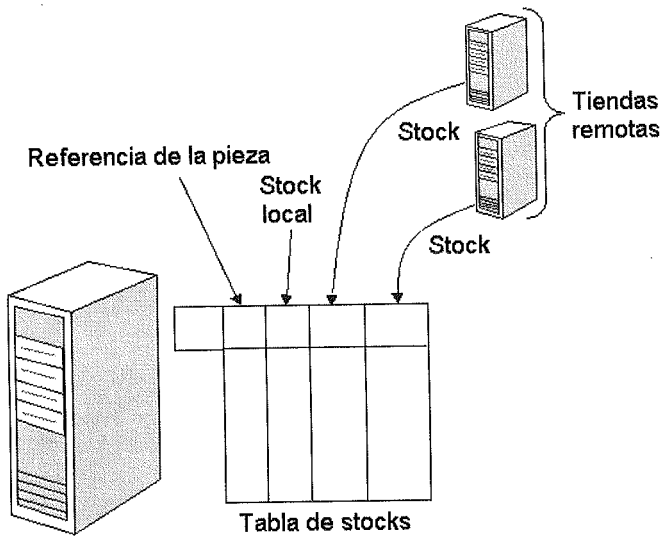
Ejemplos



Ejemplo de particionamiento horizontal

Las entradas a espectáculos se venden en varios puntos de venta. Para que el mismo asiento no se venda dos veces, la operación más sencilla de ejecutar consiste en asignar a cada punto de venta un número de asientos. Por lo tanto, la sala se particiona en función de los puntos de venta.

Como consecuencia, cada punto de venta gestiona de manera autónoma los asientos que tiene asignados. Sin embargo, cada punto de venta puede saber qué asientos aún no han sido vendidos por los otros puntos de venta. Aquí la coherencia transaccional latente es suficiente.



Ejemplo de particionamiento vertical

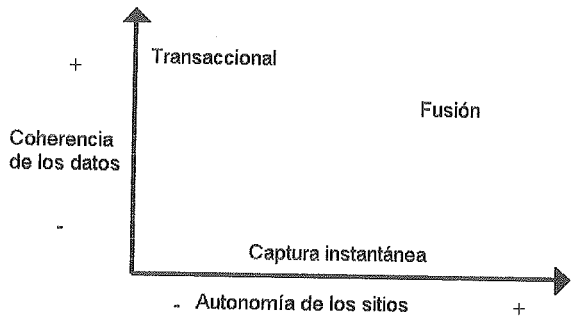
Cada punto de venta de una cadena de tiendas de reparación de automóviles gestiona su propio stock de mercancías y por medio de la informática conoce el stock de los puntos de venta más próximos. El conocimiento de este stock se utilizará cuando falte una pieza y sea necesario satisfacer al cliente lo más rápido posible. Aquí la coherencia transaccional inmediata no es necesaria, ya que lo que se requiere es un conocimiento general del stock de la tienda más cercana (accesible únicamente en modo de lectura).

2.4 Tipos de replicación

Existen tres tipos de replicación proporcionados por SQL Server:

- Captura instantánea.
- Transaccional: captura instantánea y actualización inmediata de los suscriptores.
- Fusión.

Cada uno de los tipos de replicación responde a una necesidad bien concreta. Siguiendo el método seleccionado, o bien convergen los datos, o bien se garantiza la coherencia de las transacciones.



Todos los métodos de replicación proporcionan funciones y atributos para gestionar tanto la coherencia de los datos como la autonomía de los sitios. La gestión del particionamiento es responsabilidad absoluta del programador y ningún método de replicación afecta a este criterio.

Es importante señalar que una misma aplicación puede establecer al mismo tiempo, aunque en datos diferentes, varios modos de replicación. No todos los datos necesitan, sin duda alguna, la coherencia transaccional inmediata. Algunas veces una coherencia transaccional latente será suficiente y, para algunos valores, puede ser oportuno establecer una replicación de fusión para que las bases tiendan a gestionar el mismo juego de datos.

El tipo de replicación seleccionado dependerá de las exigencias en términos de coherencia de los datos y de autonomía de los sitios, pero también de tener en cuenta recursos materiales tales como la capacidad de red, siendo esta última muy utilizada en la coherencia transaccional inmediata.

3. Los modelos de replicación

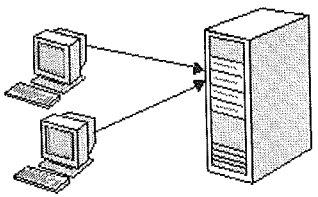
En SQL Server, los diferentes modelos de replicación utilizan la metáfora «editor-suscriptor» para diseñar lo mejor posible dichos modelos.

3.1 Los principales componentes

3.1.1 El editor

Igual que un editor de libros o periódicos, un servidor editor pone a disposición de los otros servidores los datos para instalar la replicación.

El editor conserva todos los datos publicados (los que participan en la replicación) y mantiene al día las modificaciones realizadas sobre estos datos. Para los datos publicados, el editor es siempre único.

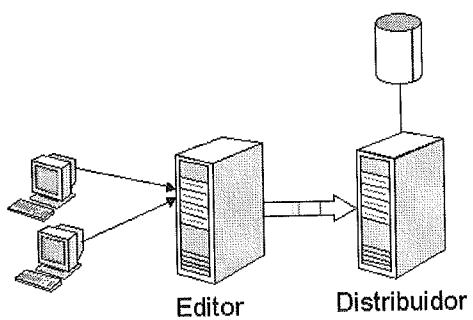


3.1.2 El distribuidor

Se trata del servidor SQL que contiene la base de distribución; es decir, aquella que contiene toda la información utilizada por los suscriptores para tener actualizados los datos que incluyen.

Observación

Estos dos roles pueden ser realizados por la misma máquina.



3.1.3 Los suscriptores

Son los servidores SQL los que almacenan una copia de la información publicada y después reciben las modificaciones de estos datos. En las versiones 6.x de SQL Server, no era posible modificar los datos sobre los suscriptores. Ahora es posible modificar los datos publicados sobre el suscriptor. Un suscriptor puede convertirse en editor para otros suscriptores.

Igual que en una revista, los suscriptores deben contar con una suscripción para recibir los datos publicados. Existen dos tipos de suscripción:

Suscripción enviada

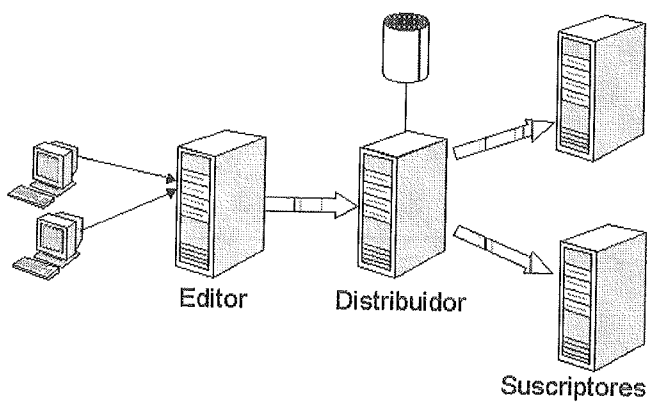
En este caso, es el distribuidor el que se encarga de enviar la actualización de los datos distribuidos a todos los suscriptores. Este tipo de suscripción es especialmente adecuada cuando el tiempo de actualización de los suscriptores debe reducirse al mínimo.

Observación

Con las suscripciones enviadas, los suscriptores pueden ser bases de datos diferentes de SQL Server, como Oracle, por ejemplo.

Suscripción de extracto

Es el suscriptor el que decide suscribirse o no. Es, por tanto, el suscriptor el que va a solicitar regularmente actualizaciones. Este tipo de suscripción resulta muy adecuado cuando los suscriptores son muy numerosos, ya que la carga de trabajo del servidor distribuidor será muy importante. Las suscripciones de extracto también se adaptan bien a los usuarios poco asiduos, ya que cuando el usuario se conecta a la red de la empresa es su propio servidor el que solicita la actualización de sus datos.



3.1.4 Los agentes

Para funcionar correctamente, la aplicación necesita que algunos programas se ejecuten de manera recurrente. Estos programas reciben el nombre de agentes y su ejecución repetitiva es posible porque aparecen en forma de tareas planificadas. Los agentes de replicación funcionan, por tanto, bajo el control de SQL Server Agent.

Los agentes de replicación son:

- **El agente de instantáneas**: su función es proporcionar una imagen exacta de la base replicada en un momento determinado. Esta imagen tiene en cuenta las estructuras y los datos. La totalidad de esta captura se almacena en los archivos de captura instantánea y la información de sincronización se almacena en la base de distribución. Este agente se ejecuta sobre el servidor distribuidor.
- **El agente de lectura del diario**: este agente, activo únicamente en el marco de la replicación transaccional, escruta el diario de base de datos al que está unido y copia las transacciones que tienen que ver con la replicación hacia la base de distribución. Si se definen replicaciones transaccionales sobre varias bases, entonces cada una de ellas tiene su propio agente de lectura. Este agente se ejecuta sobre el distribuidor.
- **El agente de distribución**: ejecutado sobre el servidor de distribución (suscripción enviada) o bien sobre el suscriptor (suscripción de extracto), el agente de distribución se encarga de aplicar la captura instantánea y las transacciones registradas en la base de distribución. Cada suscriptor dispone de una instancia específica del agente de distribución.
- **El agente de mezcla**: específico de la replicación de fusión, se encarga en primer lugar de aplicar la captura instantánea sobre el suscriptor. Seguidamente, se conecta al servidor de publicación y al suscriptor para unir la información. Por defecto, descarga las modificaciones del suscriptor en el servidor de publicación y después traslada al suscriptor las modificaciones que han intervenido sobre el servidor de publicación. Cada suscriptor a una replicación de fusión tiene su propio agente de fusión.
- **El agente de lectura de la fila de espera**: específico de la replicación transaccional con la opción de actualización pendiente, este agente se ejecuta sobre el servidor de distribución. No existe más que una única instancia independientemente del número de suscriptores. El agente de lectura de la fila de espera va a enviar las modificaciones efectuadas a nivel de los suscriptores sobre el servidor de publicación.

Los agentes de replicación se gestionan tanto por SQL Server Management Studio como por el monitor de replicación de SQL Server.

3.1.5 Los elementos que participan en la replicación

Es posible considerar que todos los elementos relativos a los datos y objetos creados de los usuarios pueden participar en la replicación. Es decir, que se pueden replicar las tablas, vistas, procedimientos almacenados, funciones y assemblies CLR. Solo la replicación de fusión es algo más restrictiva, ya que no permite replicar la ejecución de procedimientos almacenados y vistas indexadas.

3.2 Replicación de instantáneas

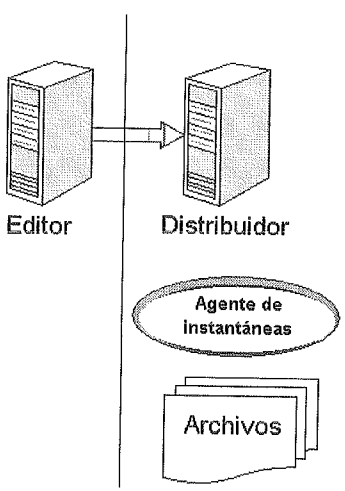
Esta replicación consiste en tomar una imagen instantánea de los datos publicados en la base de datos. Este tipo de replicación exige una sobrecarga de trabajo poco importante para el servidor editor, ya que la operación es puntual. Los suscriptores se actualizan copiando la totalidad de los datos publicados en lugar de efectuar únicamente las modificaciones (INSERT, UPDATE y DELETE). Esta replicación es adecuada para publicaciones de pequeño volumen, ya que en caso contrario las actualizaciones de los suscriptores pueden necesitar importantes recursos de red.

Esta replicación también es adecuada para grandes volúmenes de información actualizada de manera puntual y global. Por ejemplo, es el caso de un catálogo de productos editado por un proveedor. Este catálogo es voluminoso y se actualiza (cambio de tarifas, modificación de descripciones, adición de nuevos productos, eliminación de productos antiguos) de manera global y puntual (por ejemplo, dos veces al año). La difusión de estos numerosos cambios es más rápida mediante una captura instantánea.

Esta replicación es la más simple y garantiza una coherencia transaccional latente.

La replicación de captura instantánea se utiliza a menudo cuando los suscriptores tienen necesidad de acceder a la información únicamente en modo de lectura y no necesitan conocer la información en tiempo real.

Es el agente de instantáneas el que se encarga de efectuar el trabajo para preparar los archivos que contienen los esquemas y los datos de las tablas publicadas. Estos archivos se almacenan en el distribuidor.



Cada vez que el agente de captura instantánea se ejecuta, comienza por verificar la existencia de nuevas suscripciones. Si no es el caso, entonces no se genera ningún script ni ningún archivo de datos.

Si la publicación se crea con la opción de creación inmediata de una primera captura instantánea, se crean nuevos archivos de datos y nuevos esquemas cada vez que se ejecuta el agente de captura instantánea.

Todos los archivos y esquemas se almacenan en la carpeta de captura instantánea y después el agente de distribución o de fusión los transfiere hacia el suscriptor, a menos que se decida hacer esta etapa manualmente.

El agente de captura instantánea añade información a la tabla **MSrepl_commands** para indicar la ubicación del juego de sincronización y a la tabla **MSrepl_transactions** para precisar la tarea de sincronización del suscriptor. Estas dos tablas se sitúan en la base de datos de distribución.

Observación

*El procedimiento **sp_replcmds** permite ver la lista de los comandos para las transacciones marcadas para la replicación.*

La captura instantánea es el punto de inicio de la replicación.

Los scripts correspondientes a esta captura instantánea se almacenan normalmente sobre el distribuidor. Es posible definir otra ubicación adicional o alternativa a la predeterminada para reducir la carga de trabajo del distribuidor. Esta otra ubicación puede corresponder a un directorio compartido sobre otro servidor o bien a un soporte extraíble, como un CD-ROM. Este último tipo de soporte puede ser muy valioso en el caso de la replicación de una base de datos voluminosa para no sobrecargar la red. Esta otra ubicación se conserva como propiedad de la publicación.

Con este tipo de replicación, la base de datos de distribución no se utiliza y no contiene ningún dato de usuario.

Según la configuración de la replicación para la que se efectúa la captura instantánea, los archivos generados no serán iguales.

Si la captura instantánea es relativa a una replicación transaccional o una replicación de fusión sin publicación con filtro parametrizado, entonces la captura instantánea contiene la estructura y los datos en los archivos en formato bcp (copia por bloque).

En caso contrario (replicación de fusión con una publicación que tiene un filtro parametrizado), entonces la captura instantánea se realiza en dos pasos. En primer lugar, se captura la estructura y después se capturan los datos para cada suscriptor a la fusión con el objetivo de tener en cuenta el filtro particular de los datos.

En caso de que la configuración de esta instantánea no se haga de manera gráfica sino en forma de script, se utilizará la herramienta snapshot.

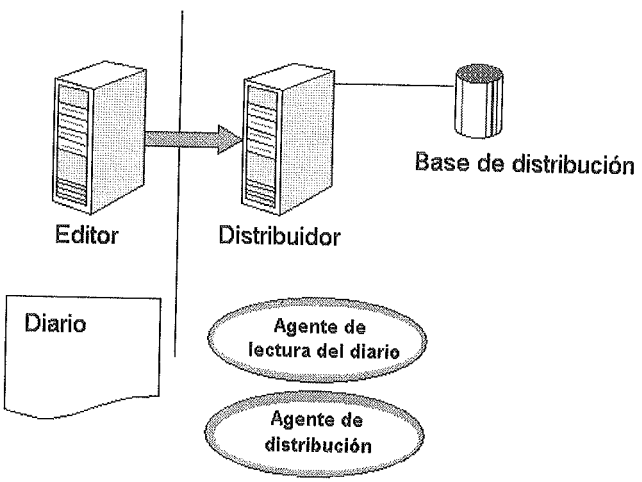
3.3 Replicación transaccional

Puede utilizarse para replicar tablas (total o parcialmente) y procedimientos almacenados.

Las replicaciones transaccionales utilizan el diario para ver las modificaciones aportadas a los datos publicados. Estas modificaciones se almacenan inicialmente sobre la base de distribución antes de ser enviadas a los suscriptores.

Todas las publicaciones tienen un editor. Las modificaciones aportadas al editor se copian de nuevo en la base de distribución con un lapso de tiempo más o menos amplio. En un entorno en el que las conexiones son óptimas, el tiempo de latencia entre el editor y los suscriptores puede ser muy pequeño.

Esta replicación funciona tanto con las suscripciones enviadas como con las suscripciones de extracto.



Observación

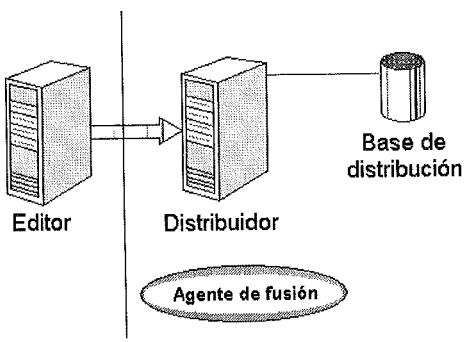
La replicación transaccional siempre se establece después de una captura instantánea para que todos los suscriptores partan de la misma base de información.

En función del tipo de publicación, la replicación transaccional puede permitir la actualización de la información directamente sobre los suscriptores con un informe de la transacción sobre el servidor, con el objetivo de distribuirlo hacia otros suscriptores en el marco clásico de la replicación transaccional.

3.4 Replicación de fusión

Se trata aquí de vigilar las modificaciones en una base de datos fuente y de sincronizar los valores entre el editor y los suscriptores. Estos últimos pueden efectuar operaciones de actualización sobre los datos distribuidos. Si el editor conserva el control de la publicación, no son siempre las operaciones efectuadas sobre el editor las que tienen prioridad sobre las efectuadas sobre el suscriptor. Todas las modificaciones realizadas en la base destino son trasladadas a la base origen.

Todas las tablas que participan en una publicación de fusión tienen una columna **unique** identifier con la propiedad ROWGUIDCOL. En caso contrario, se añade una columna **rowguid** automáticamente.



Observación

La replicación de fusión no siempre comienza por una captura instantánea.

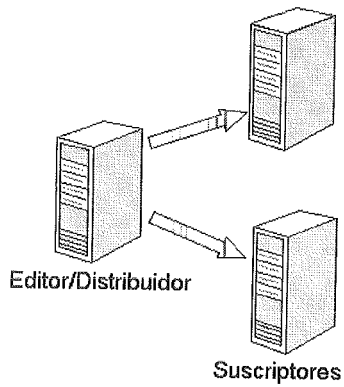
3.5 Los modelos físicos de replicación

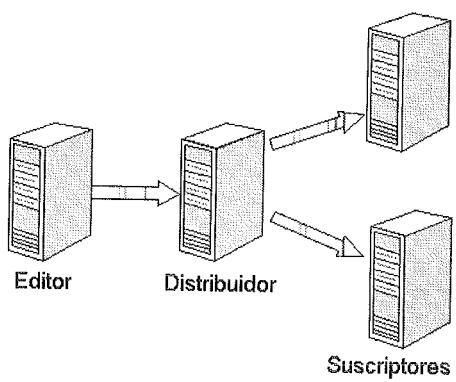
Todos los modelos físicos presentados a continuación son independientes del tipo de replicación seleccionado. Además, su presentación se basa completamente en la metáfora «editor/distribuidor/suscriptor».

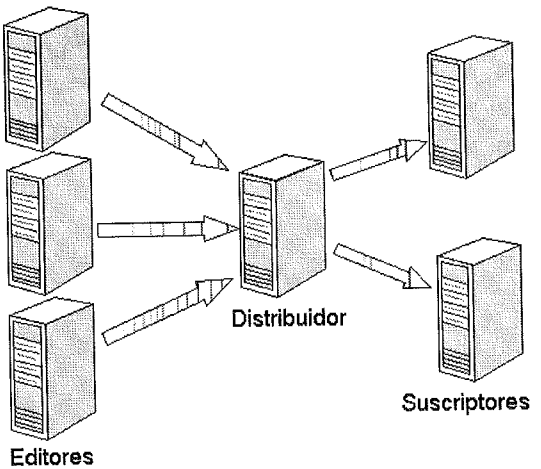
3.5.1 Editor central-suscriptores múltiples

Este es el modelo por defecto en SQL Server. El servidor editor desempeña también el papel de distribuidor. El editor y el distribuidor pueden ejecutarse sobre dos servidores diferentes. También es posible utilizar el mismo distribuidor para varios editores.

El servidor de publicación (el editor) es el propietario de todos los datos replicados. El servidor de distribución almacena los datos en espera de que sean enviados a los suscriptores. Los datos recibidos sobre los suscriptores deben estar accesibles en modo de solo lectura: los usuarios tienen únicamente el permiso SELECT.







Observación

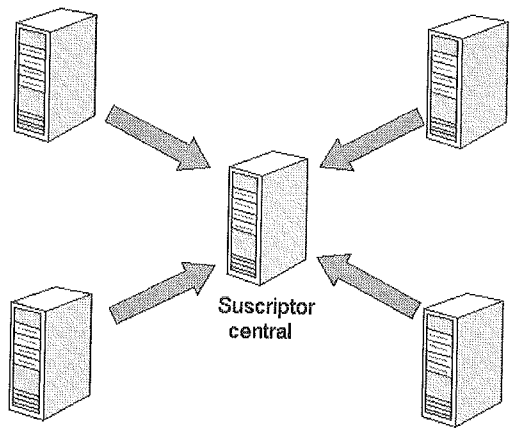
Si el servidor y el distribuidor están en dos máquinas distintas, la parte más grande del trabajo de replicación la realiza el distribuidor.

3.5.2 Suscriptor central-editores múltiples

En este caso, varios editores replican los datos hacia un suscriptor central. El suscriptor centraliza la información de todos los editores. Tiene una vista global de la situación, mientras que cada editor tiene una vista local de la información. Dado que varios editores van a registrar información en la misma tabla de suscripción, para evitar la pérdida de información es importante que cada dato sea propiedad de un único editor. La manera más sencilla es establecer un filtrado horizontal de los datos.

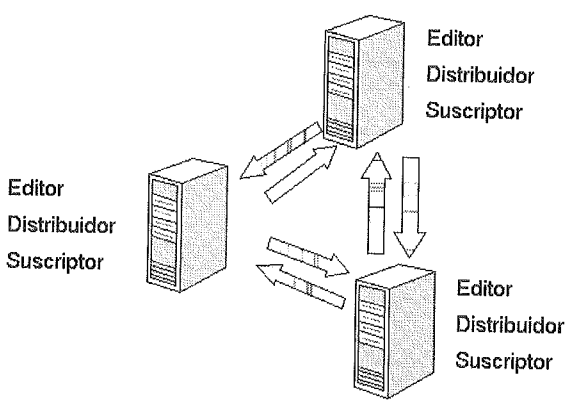
Ejemplo

Este modo de replicación puede ser establecido por una empresa que desea estar al corriente de las operaciones que se llevan a cabo sobre todos los puntos de venta.



3.5.3 Editores múltiples-suscriptores múltiples

En este caso, los editores son también suscriptores múltiples. Este caso puede producirse, por ejemplo, entre varios puntos de venta que trabajan de manera independiente unos de otros, pero que desean conocer los diferentes stocks de los otros puntos de venta.



Todos los modelos físicos de replicación se pueden asociar a los diferentes tipos de replicación que existen.

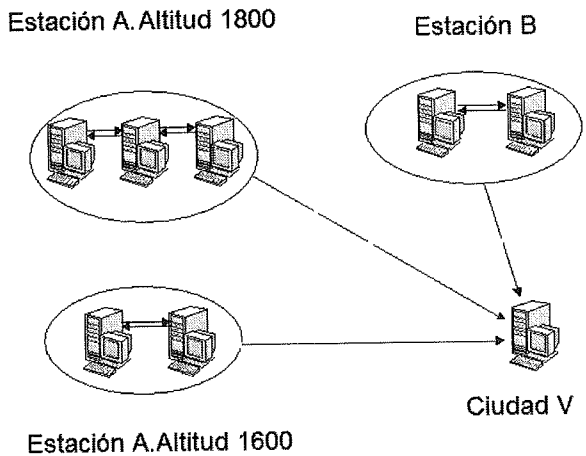
Ejemplo

Un propietario de tiendas de esquí tiene siete tiendas repartidas como sigue:

- Tres tiendas en la estación A altitud 1800.
- Dos tiendas en la estación A altitud 1600.
- Dos tiendas en la estación B.

El propietario vive en la ciudad V y desea ver al final de cada día los resultados de sus diferentes tiendas.

Con el objetivo de responder lo mejor posible a las exigencias de los clientes, las tiendas situadas en una misma estación y a una misma altitud conocen todo el stock para su ciudad y altitud. ¿Qué modelo y tipo de replicación es necesario establecer?



Se va a establecer, en cada estación, una replicación transaccional sin autorizar las actualizaciones sobre los suscriptores y después cada tienda va a establecer una replicación transaccional hacia el servidor utilizado por el propietario.

4. Planificación

La implementación de la replicación necesita una planificación rigurosa de las tareas que hay que realizar con el fin de utilizar lo mejor posible los recursos proporcionados por SQL Server, reduciendo los recursos materiales (tiempo de CPU, red) utilizados por la replicación.

4.1 Opciones generales de planificación

4.1.1 Opción NOT FOR REPLICATION

La opción **NOT FOR REPLICATION** permite definir un comportamiento diferente de las opciones cuando el tratamiento se hace en el marco de la replicación. Esta opción es posible sobre:

- Las columnas de tipos identity.
- Las restricciones de validación (CHECK).
- Las restricciones de clave extranjera (FOREIGN KEY).
- Los triggers de base de datos.

4.1.2 Tipo de datos uniqueidentifier

El tipo de datos **uniqueidentifier** se utiliza con la columna ID y con la función NEWID(), que permite generar un nuevo ID para cada nueva línea.

Ventajas de GUID:

- GUID es siempre única y de esta manera se evitan numerosos conflictos.
- El número de GUID no está limitado, contrariamente a los identificadores.

Sin embargo, esta opción no es interesante cuando los usuarios no ven o no utilizan los valores de GUID. Efectivamente, los valores de tipo **uniqueidentifier** presentan inconvenientes que no pueden pasarse por alto al poner en marcha una aplicación.

- La manipulación por un usuario es complicada (formato demasiado largo).
- Los valores son aleatorios y no tienen ningún sentido.
- Los valores **uniqueidentifier** no están necesariamente disponibles para las aplicaciones existentes, que se construyen a partir de valores de identificadores incrementales.

En resumen, es posible precisar que los valores de tipo **uniqueidentifier** se adaptan bien cuando son utilizados directamente por SQL Server o por la aplicación, ya que permiten manejar un identificador único, pero su manejo por parte del usuario final es difícil. Estos valores son muy utilizados en el proceso de replicación para el almacenamiento interno de la información.

4.1.3 Filtrado de los datos

Por defecto, cuando una tabla participa en una publicación, la totalidad de los datos que contiene se replica sobre los puestos remotos. Sin embargo, los puestos remotos no tienen obligatoriamente necesidad de conocer la totalidad de los valores. Algunas veces solo les interesa un subconjunto de líneas o de columnas. Hablamos en esos casos de filtrado.

El filtrado permite reducir el volumen de datos publicados y por lo tanto la replicación necesita menos recursos de los servidores sobre los que se ha establecido (tiempo de CPU, espacio en disco...) y de red, ya que el volumen de información que hay que transmitir a cada suscriptor es menor.

Filtrado horizontal

Se trata en este caso de publicar un subconjunto de líneas contenidas en la tabla publicada. El suscriptor solo recibirá las líneas de la tabla del editor que contiene la información que le interesa.

Sin embargo, el filtrado horizontal presenta algunos inconvenientes. En el caso de una replicación transaccional, la cláusula de filtrado debe ser evaluada para cada línea relativa a la publicación. Si la evaluación es positiva, entonces la información se transmite al distribuidor; en caso contrario, no sucede nada. Esta sobrecarga de trabajo está asegurada por el editor. El coste de esta carga suplementaria de trabajo sobre el servidor debe compensarse con las ganancias sobre los suscriptores y la aligeración del tráfico de red. Si no es el caso, es preferible evitar un filtrado horizontal.

Filtrado vertical

Se trata en este caso de publicar únicamente algunas columnas de la tabla que participa en la publicación. Este filtrado tiene una influencia directa sobre la estructura de la tabla generada sobre el suscriptor, que es diferente de la tabla del editor.

Contrariamente al filtrado horizontal, el filtrado vertical no va a afectar de manera significativa al rendimiento del editor, ya que la carga suplementaria de trabajo es baja.

Filtrado mixto

Es posible combinar un filtrado horizontal y un filtrado vertical sobre una misma tabla.

4.2 Replicación de instantáneas

Existen dos criterios que hay que tener en cuenta para que esta replicación se desarrolle correctamente.

Planificación rigurosa de las capturas instantáneas

Es el agente de instantáneas el que se encarga de dirigir la captura instantánea de la publicación. Cuando ejecuta su trabajo, el agente bloquea la tabla y efectúa una copia en bloque de los datos. Mientras la tabla esté bloqueada por el agente, ningún otro usuario puede acceder para modificar (INSERT, UPDATE, DELETE) los datos que contiene. Para reducir la no disponibilidad de la tabla para los usuarios, la captura instantánea debe ejecutarse en un momento en que los usuarios no necesiten modificar los valores de la tabla.

Para planificar lo mejor posible la ejecución de la captura instantánea, es preciso conocer la duración aproximada del tiempo necesario para realizar la captura. Como el agente de captura utiliza **bcp**, lo más simple es hacer una copia por bloques de las tablas publicadas para estimar el tiempo necesario. Si el juego de datos es muy voluminoso, resulta más sencillo hacer un **bcp** sobre una muestra y evaluar el tiempo global.

Espacio en disco para la carpeta de captura

El resultado de la captura instantánea se almacena en forma de archivos en el distribuidor, en un directorio compartido. Como los archivos contienen todos los datos presentes en la tabla, su tamaño será cercano al de la propia tabla y es fácil, de esta manera, averiguar el espacio en disco necesario para los archivos de captura instantánea.

Los archivos de captura instantánea se almacenan por defecto en un directorio local, generalmente dedicado a contener los archivos de captura, del tipo: C:\Program Files\Microsoft SQL Server\MSSQL12.LIBRO\MSSQL\ReplData

El agente de instantáneas debe disponer de los privilegios necesarios para escribir en esta carpeta. Por el contrario, la cuenta de Windows asociada al agente de fusión y replicación debe disponer de privilegios de lectura.

4.3 Replicación transaccional

En este caso hay que tener en cuenta cuatro elementos.

Observación

No hay que olvidar que el comienzo de una replicación transaccional empieza siempre por una captura instantánea.

Espacio del diario

El agente de lectura del diario escruta el diario de la base publicada y transfiere todas las transacciones sobre los datos publicados hacia la base de distribución. El tamaño solicitado por el diario es, como regla general, ligeramente superior al de una base que no tiene replicación.

Efectivamente, si la base de distribución no está accesible o si el agente de lectura del diario está desactivado, el diario no puede vaciarse hasta que la transacción más antigua relativa a la replicación haya sido transferida hacia la base de distribución.

La base de datos de distribución

Sobre el distribuidor, la base de distribución permite almacenar y transmitir a los suscriptores toda la información necesaria para actualizar los datos replicados. La primera operación realizada sobre el suscriptor es la captura instantánea de la publicación. Esta operación de captura instantánea siempre se gestiona en forma de archivos almacenados fuera de la base de distribución.

La base de datos de distribución va a conservar todas las operaciones de modificación de los datos replicados desde la última captura instantánea. De esta manera, si llega un nuevo suscriptor, se podrán actualizar las tablas que contienen los datos publicados a partir de la captura instantánea presente sobre los servidores de distribución y después añadir todas las transacciones, por lo que la base de distribución conserva una traza.

Para que la base de distribución conserve un tamaño razonable, es prudente planificar periódicamente una nueva captura instantánea. El trabajo de limpieza de la base de distribución comienza en cuanto termina la nueva captura instantánea.

Las claves primarias

La gestión de las claves primarias está asegurada si todas las tablas que participan en la replicación tienen una. Es posible establecer una clave primaria sobre una tabla existente mediante el comando ALTER TABLE o bien utilizando SQL Server Management Studio.

Los tipos texto e imagen

La replicación transaccional tiene en cuenta los datos de texto de grandes dimensiones y las imágenes.

Para evitar problemas, es importante recordar que la replicación transaccional se apoya en el diario de las transacciones, por medio del agente de monitorización del archivo de log. Por lo tanto, solo se pueden tener en cuenta en el proceso de replicación las operaciones registradas en el diario de transacciones.

Observación

*El parámetro del servidor **max text repl size** permite especificar el tamaño máximo (en bytes) de los datos de texto e imagen que se pueden replicar. Si un comando sobrepasa este límite, la operación fracasa. El valor por defecto es de 65.536 bytes. Para replicar completamente estos datos de tipo texto, este parámetro debe valer -1.*

4.4 Replicación de fusión

Columnas timestamp

La replicación de fusión no tiene en cuenta las columnas de tipo **timestamp** (fecha/hora), ya que estas columnas son generadas automáticamente por el servidor local y solo garantizan la unicidad en una base de datos específica. Por lo tanto, no es posible que una modificación del valor **timestamp** sobre un servidor se aplique a la columna **timestamp** de otro servidor. Es la razón por la que las columnas de tipo **timestamp** deben ser eliminadas de todas las tablas que participan en la replicación de fusión.

Integridad de los datos

Como la replicación de fusión publica las modificaciones en el suscriptor, la base de datos de este último debe permitir garantizar la integridad de los datos y, por lo tanto, todas las tablas necesarias para garantizar dicha integridad deben estar presentes en el suscriptor.

Referencias de clave primaria

Si las tablas que participan en la replicación tienen restricciones referenciales, entonces las tablas referenciadas también deben participar en la publicación. Sería el caso, por ejemplo, de una tabla **pedidos** que hace referencia a la clave primaria de la tabla **clientes**. Esta última tabla solo debe participar en la replicación de fusión para autorizar la creación de pedidos sobre el suscriptor. Si el suscriptor solo necesita modificar los registros de la tabla **pedidos** sin tocar la columna que hace referencia al cliente, entonces la presencia de la tabla **clientes** no es necesaria.

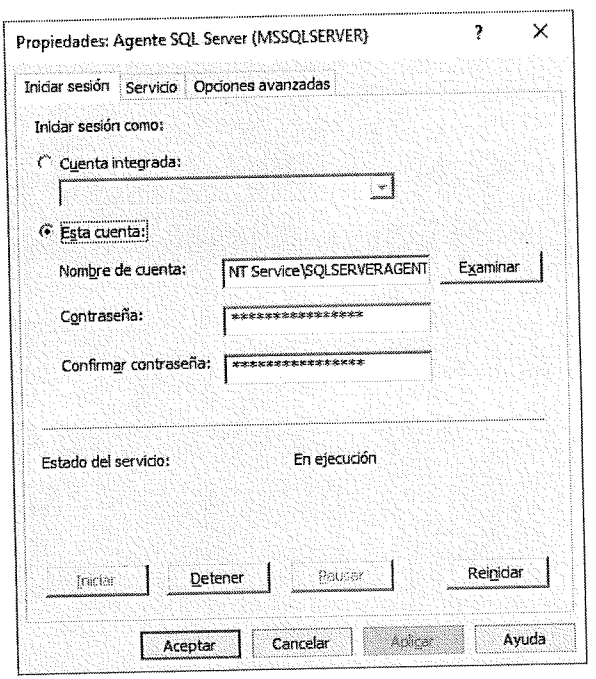
En una replicación de fusión, es posible añadir manualmente nuevas tablas en todo momento.

5. El acceso a la red

Para que el proceso de replicación se desarrolle sin problemas, se deben satisfacer algunas condiciones elementales sobre el acceso a la red:

- Si los servidores SQL que participan en la replicación están en dominios diferentes, se deben establecer relaciones de aprobación entre estos dominios.
- El agente SQL Server debe ejecutarse en el contexto de una cuenta de usuario del dominio. Si es posible, el agente SQL Server de los diferentes servidores SQL que participan en la replicación utiliza siempre la misma cuenta de usuario. Esta cuenta de usuario de dominio debe ser miembro del grupo local de administradores para que SQLServerAgent se beneficie de los privilegios administrativos.

Es posible averiguar y modificar la configuración de este servicio por medio de SQL Server Configuration Manager.



Verificación de las propiedades del agente SQL Server

Observación

El agente SQL Server no debe utilizar ni una cuenta local system ni una cuenta de usuario local, ya que estas dos cuentas no permiten acceder a los recursos de la red.

6. Puesta en marcha

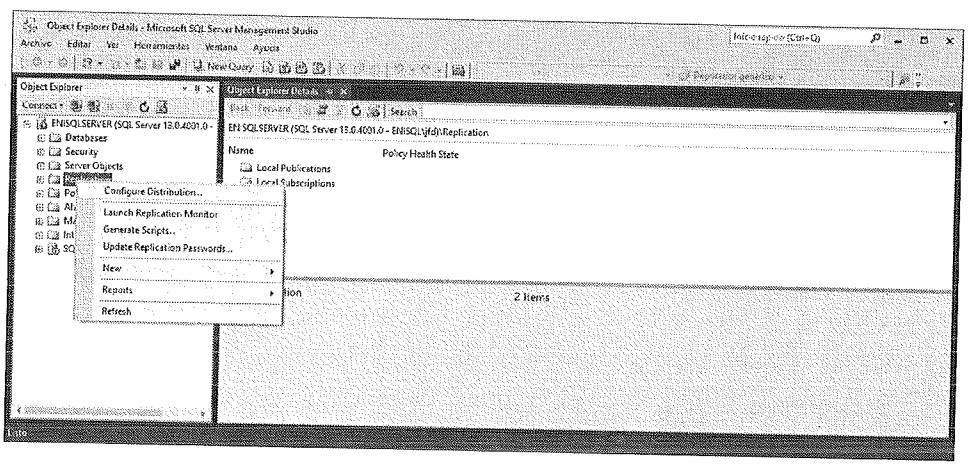
Independientemente de cuál sea el modelo y el tipo de replicación elegidos, la puesta en marcha de esta técnica siempre debe seguir las mismas etapas:

- La puesta en marcha del distribuidor.
- La puesta en marcha del editor.
- La puesta en marcha de los suscriptores.
- La definición de las publicaciones.

Como SQL Server considera por defecto que el editor y el distribuidor residen en el mismo servidor, la instalación de estos dos componentes se mezcla.

Para poder poner en marcha la replicación a partir de SQL Server Management Studio, todos los servidores SQL que participan en la replicación deben estar inscritos en ella.

SQL Server Management Studio ofrece diferentes asistentes gráficos para instalar, vigilar y parametrizar el entorno de replicación. Se accede a todos estos elementos desde el nodo **Replication** del explorador de objetos de SQL Server Management Studio.



6.1 El distribuidor

Se trata del puesto que va a gestionar el directorio compartido para la captura instantánea, así como la base de distribución. El distribuidor almacena las modificaciones efectuadas sobre los datos y los transmite a los suscriptores.

6.1.1 Conceptos

El distribuidor debe estar instalado antes de establecer los editores que lo utilizan. Para crear un distribuidor, es necesario ser administrador del sistema (miembro del grupo local Administradores y conectarse a SQL Server como administrador a través de las conexiones aprobadas). Una vez que el distribuidor esté instalado, es posible conocer sus propiedades locales y remotas.

La base de datos de distribución

Contiene todas las transacciones que están a la espera de ser enviadas hacia los suscriptores. En el marco de una replicación transaccional, es alimentada por el agente de vigilancia del diario que transfiere a ella todas las transacciones que intervienen sobre los datos replicados. Esta base se crea automáticamente durante la configuración del distribuidor, pero es posible:

- Especificar un servidor de distribución remoto durante la instalación del editor.
- Definir varias bases de distribución para un mismo editor.

Acceso al directorio compartido

El directorio de distribución, utilizado por la replicación de captura instantánea, debe estar disponible para todos los agentes de distribución que sean capaces de utilizarlo. Su uso depende del tipo de replicación establecida y solo se plantea cuando se utiliza un servidor de distribución remoto.

La memoria

La memoria física presente en el distribuidor debe existir en cantidad suficiente para responder adecuadamente a las diferentes peticiones de los suscriptores. La cantidad de memoria necesaria depende del volumen de datos replicados y del número de suscriptores.

La desinstalación

Es posible desinstalar un distribuidor por medio del asistente **Replicación**, **Desactivar el asistente de Publicación** y **Distribución**. Los efectos son los siguientes:

- Las bases de datos de distribución del servidor son eliminadas.

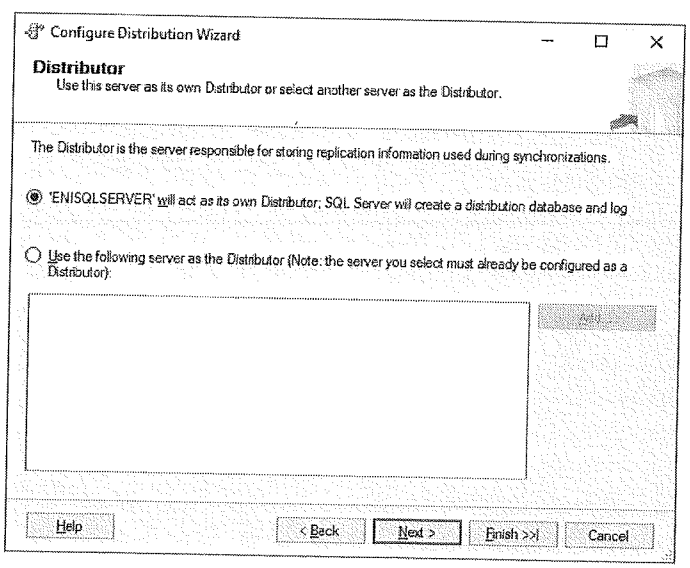
- Todos los editores que utilizan este distribuidor son desactivados y se eliminan todas las publicaciones.
- Todas las suscripciones son eliminadas, pero los datos de suscripción permanecen en los suscriptores.

6.1.2 El establecimiento

El establecimiento de un distribuidor necesita privilegios de administrador; es decir, ser miembro del rol de servidor **sysadmin**. La creación del distribuidor se hará por medio de los procedimientos almacenados o bien a través de SQL Server Management Studio.

Se accede a la configuración del distribuidor por dos rutas diferentes en SQL Server Management Studio.

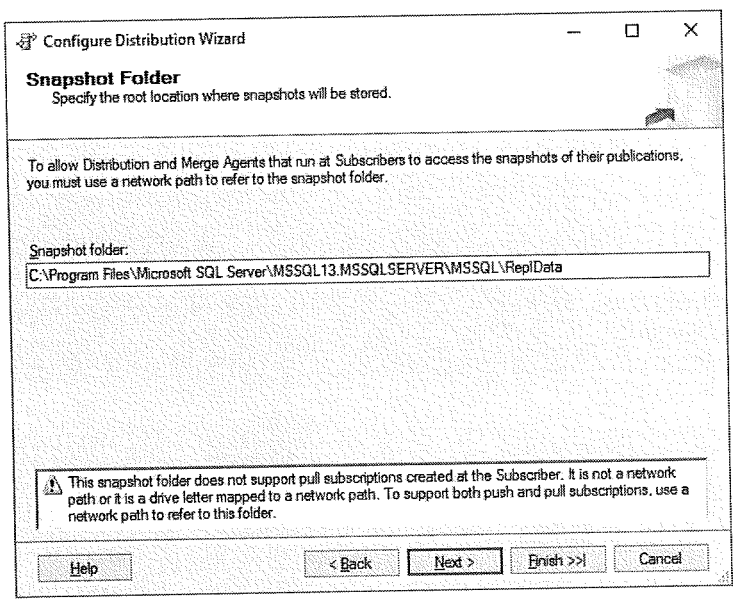
- Bien mediante la selección explícita de la opción **Configure Distribution** en el menú contextual asociado al nodo **Replication** desde el explorador de objetos.
- Bien solicitando la creación de una nueva publicación (opción **New - Publisher**) desde el menú contextual asociado al nodo **Replication**. Entonces se ejecuta el asistente de creación de una publicación y este asistente permite seleccionar/configurar un distribuidor para la replicación.



El asistente de publicación creando un distribuidor

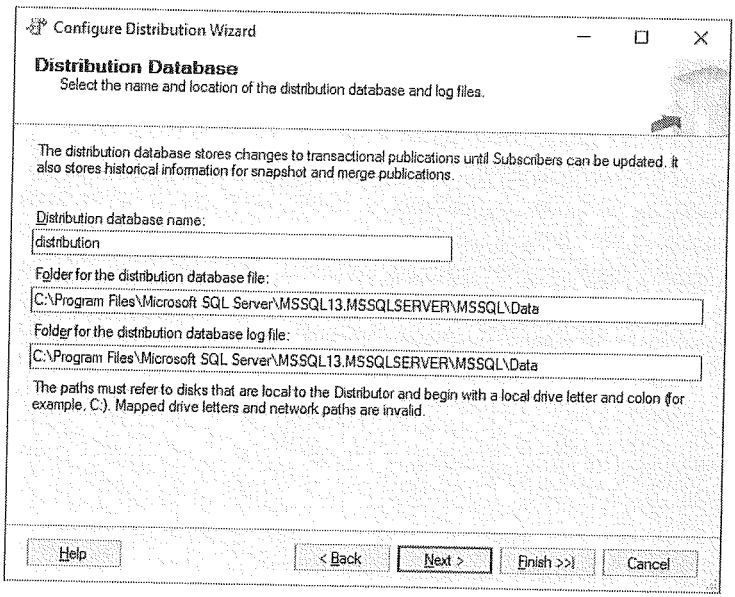
En caso de que la petición se lleve a cabo sobre la configuración de la distribución, se ejecuta el asistente de configuración de la distribución. Este asistente en primer lugar va a solicitar que se indique el servidor que tiene el papel de distribuidor. Por lo tanto, es posible seleccionar el servidor que contendrá también las publicaciones (editor/distribuidor) o bien seleccionar un distribuidor remoto.

En la etapa siguiente, el asistente permite especificar la carpeta utilizada para los archivos de instantáneas. Se propone una carpeta local por defecto. Si la replicación de instantáneas debe ser accesible desde la red, es necesario indicar un nombre de ruta de red. La advertencia situada en la parte inferior de la ventana alerta precisamente sobre este punto.

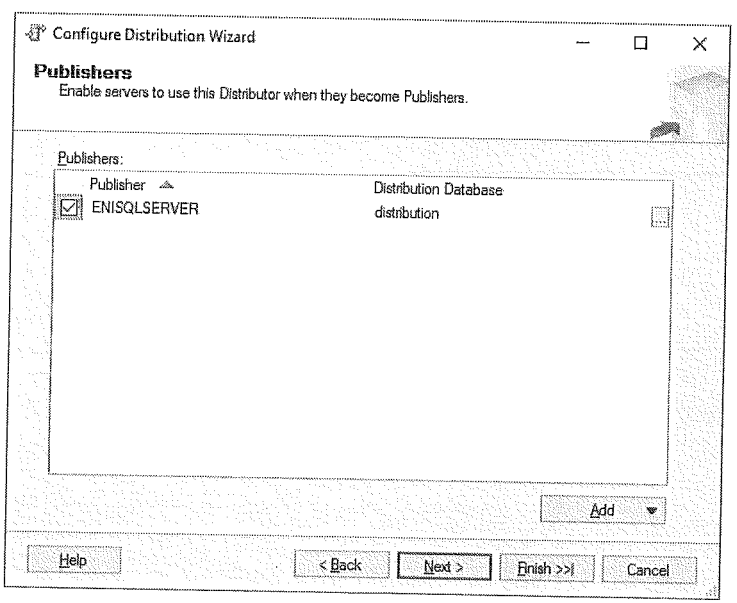


Después, el asistente permite precisar el nombre de la base de distribución, así como la ubicación física del archivo de datos y del diario. Como muestra la pantalla siguiente, la base se llama por defecto "distribution" y las carpetas propuestas son las especificadas por defecto.

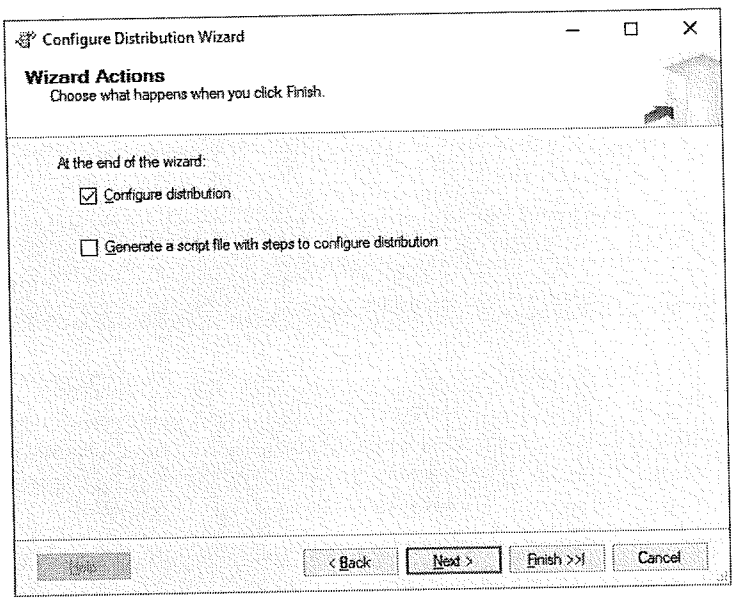
Es importante que la cuenta de seguridad utilizada para realizar el snapshot pueda escribir en esta carpeta.



A continuación, el asistente solicita indicar cuáles son los editores autorizados para trabajar con este distribuidor.

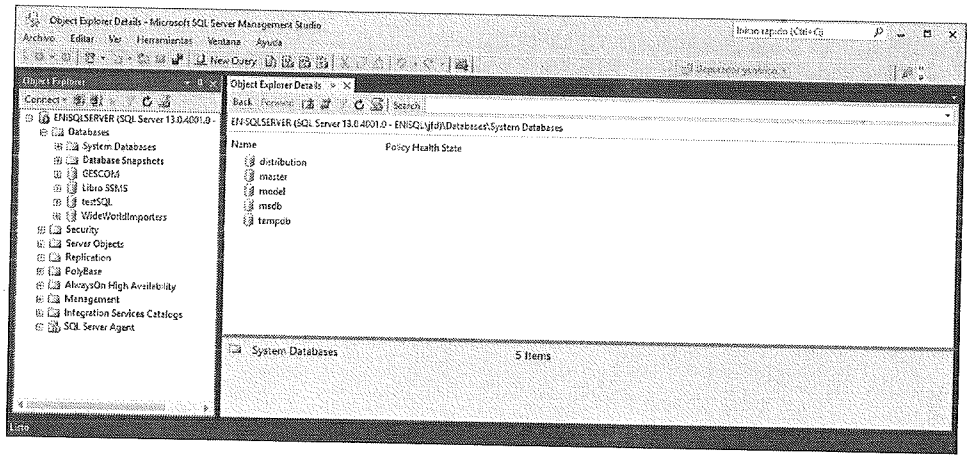


Por último, el asistente ofrece la posibilidad de realizar la configuración inmediatamente o bien de generar los scripts Transact SQL correspondientes a las diferentes opciones especificadas con el asistente, para poder poner en marcha más tarde el distribuidor.



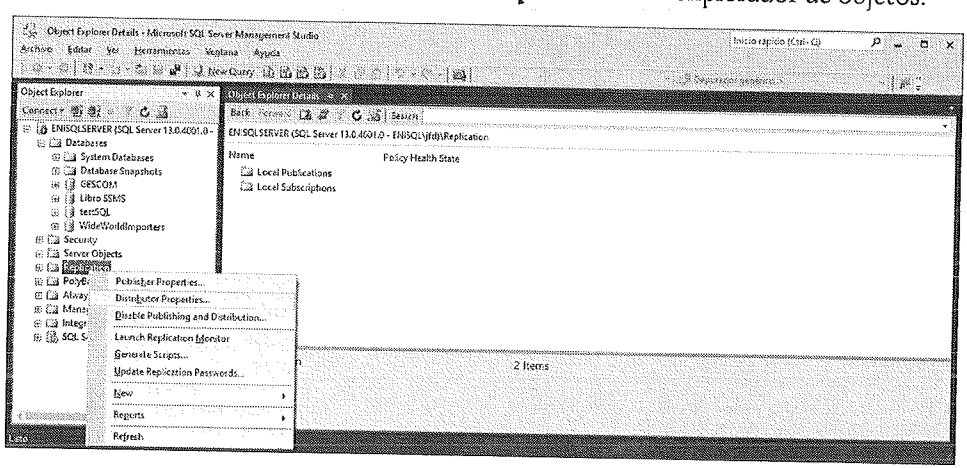
Observación

No se puede efectuar ninguna modificación sobre el servidor hasta que el asistente termine.

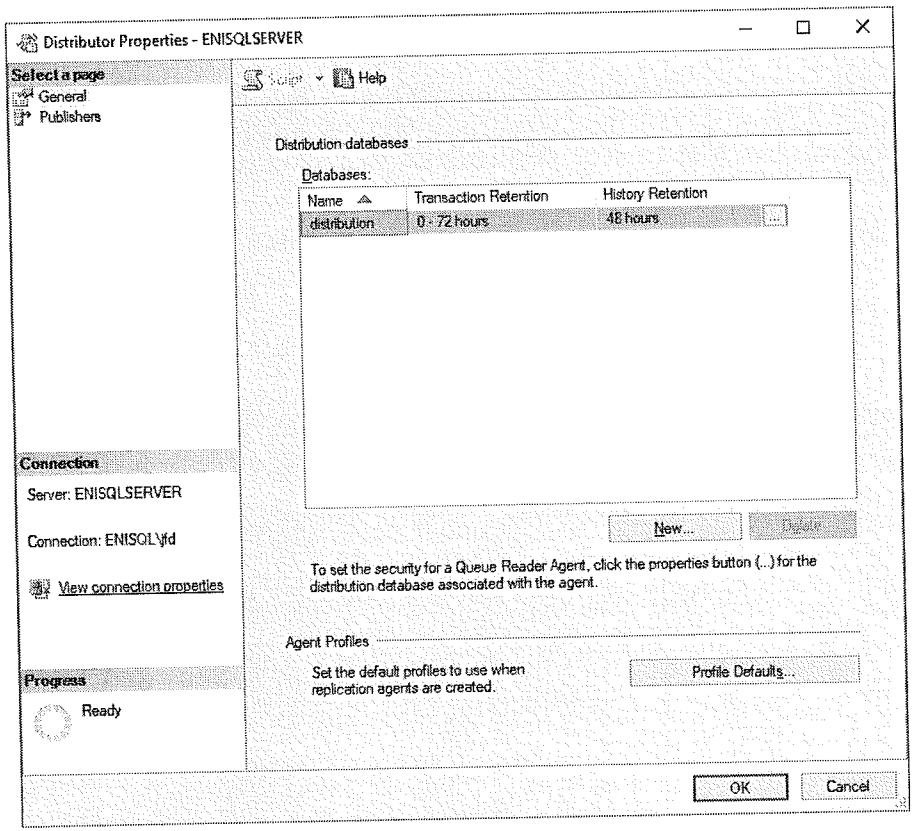


Aparición de la base de datos de distribución

Las opciones accesibles desde SQL Server Management Studio son ligeramente diferentes para poder realizar todas las operaciones asociadas a la administración del servidor de distribución. A estas nuevas funcionalidades se accede fundamentalmente por el menú contextual asociado al nodo **Replication** del explorador de objetos.



El menú contextual permite visualizar las propiedades del distribuidor. Estas permiten configurar la distribución en sí misma, como la duración de retención de las transacciones y del histórico. También es posible modificar la lista de los editores autorizados a utilizar este distribuidor.



También es posible lanzar el monitor de replicación con el objetivo de seguir el trabajo efectuado por cada agente que participa en la replicación. El monitor de replicación permite conocer el estado de cada suscriptor y seguir las operaciones de sincronización.

El menú contextual también ofrece la posibilidad de desactivar la distribución.

Siempre desde el explorador de objetos, es posible visualizar la base de datos del sistema de distribución que se ha creado con el asistente.

6.2 El editor

Después de la creación del distribuidor, es posible crear un editor que utilice este distribuidor.

Un distribuidor puede ser común a varios editores y un editor puede utilizar varios distribuidores.

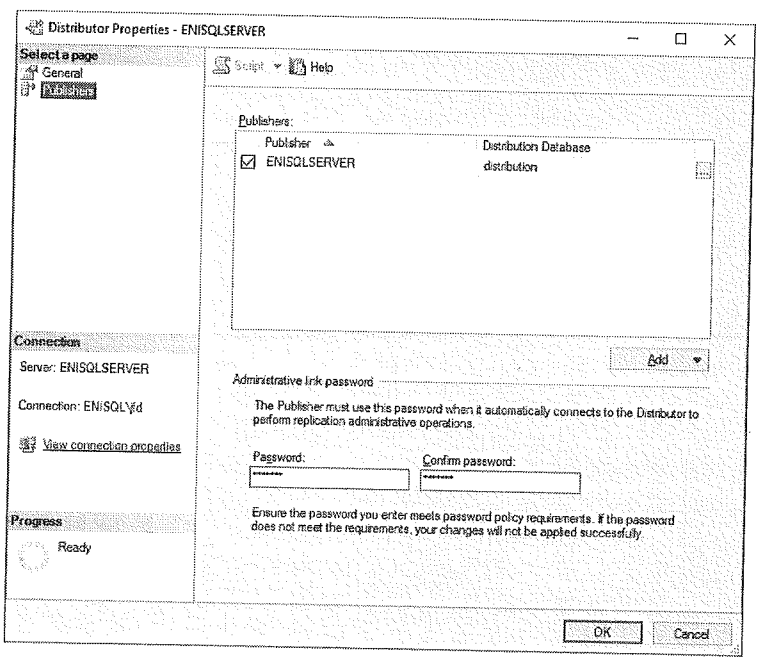
Al agregar un editor, es posible:

- Activar el editor y autorizarle a utilizar el distribuidor.
- Visualizar o modificar las opciones del editor.

La adición, la eliminación y la modificación de un editor se pueden realizar de diferentes maneras:

SQL Server Management Studio

A partir de los propiedades del distribuidor es posible gestionar (añadir/eliminar) los publicadores (editor) que tienen permiso para el distribuidor.



Es posible añadir un distribuidor SQL Server u Oracle. Aunque los procedimientos de inscripción sean diferentes, el botón **Add** ofrece la posibilidad de elegir el tipo del editor antes de autorizarlo a utilizar el distribuidor.

Para eliminar la autorización asignada a un distribuidor, es necesario desactivar el publicador desde las propiedades del distribuidor.

Transact SQL

Las operaciones relativas a la adición y la eliminación de un editor del distribuidor son posibles mediante los procedimientos:

- **sp_adddistpublisher** para añadir un nuevo publicador en el distribuidor. La utilización de este procedimiento solicita la ejecución del procedimiento sp_get_distributor para determinar si la instancia de SQL Server está configurada o no como un distribuidor.
- **sp_dropdistpublisher** para eliminar el publicador del distribuidor.

Modificación de la base de datos de distribución

Si un editor cambia de base de distribución, esto implica comenzar todo desde el principio: desactivar el editor, eliminarlo de la base de datos de distribución inicial y activar el editor con otra base de datos de distribución. Sobre el editor, conviene crear las publicaciones y los artículos, así como activar los suscriptores que pueden recibir los datos que provienen del editor a través del distribuidor.

6.3 Las publicaciones

Existen dos tipos de publicaciones: las de datos y las de procedimientos almacenados.

Una publicación, en el marco de la replicación SQL Server, corresponde a un conjunto de elementos (artículos) que participan en la replicación. Dependiendo del tipo de replicación seleccionado, estos artículos pueden ser tablas, tablas particionadas, procedimientos almacenados, funciones, vistas, vistas indexadas o tipos de datos definidos por el usuario.

Cuando una tabla participa en una publicación, se llama **artículo**. El artículo corresponde a la totalidad de la tabla o bien a un subconjunto de líneas o de columnas. Un artículo también puede corresponder a la ejecución de un procedimiento almacenado; sin embargo, esta funcionalidad no está disponible en el marco de una replicación de fusión.

Cuando se cree una publicación es cuando el editor va a utilizar los recursos puestos a su disposición por el distribuidor para albergar los archivos de instantáneas, así como la copia de las transacciones sobre la base de distribución.

Cuando se crea una publicación, deben resolverse los elementos siguientes:

- El nombre de la publicación y su descripción.
- El servidor de distribución que se va a utilizar.

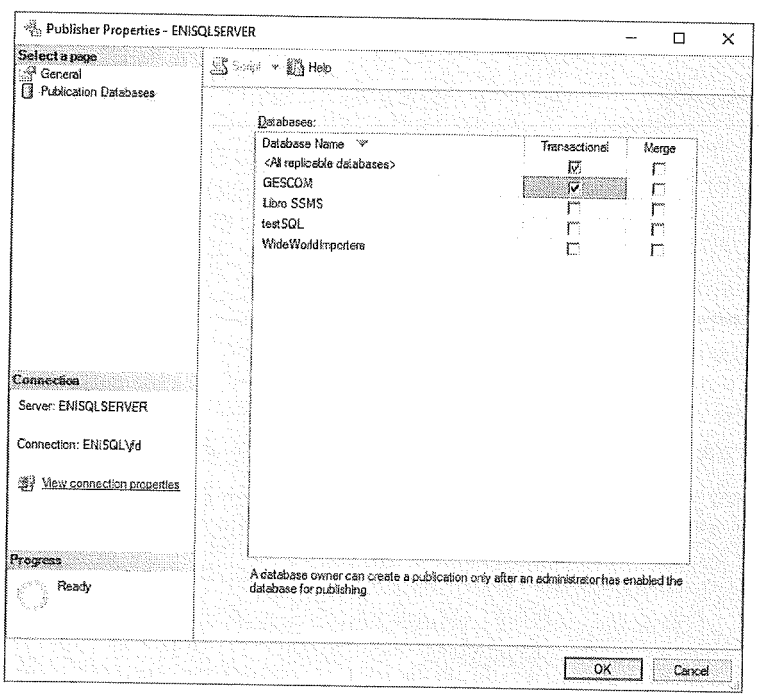
- El tipo de replicación (captura instantánea, transaccional o fusión) en la que la publicación va a participar.
- Los artículos.

Todos estos elementos deberán especificarse cuando se ejecute el asistente de creación de una publicación.

Creación y modificación

La creación de una publicación por parte del propietario de la base de datos (o un usuario miembro del rol db_owner) solo es posible si el administrador del servidor ha habilitado la base para participar en una replicación. En el momento de esta activación, el administrador indica si las publicaciones definidas sobre la base de datos pueden inscribirse en una replicación transaccional o de fusión.

La activación de la publicación sobre las bases de datos es posible desde la ventana que muestra las propiedades del publicador. La opción **Publisher Properties** desde el menú contextual asociado al nodo **Replication** del explorador de objetos permite visualizar esta ventana.

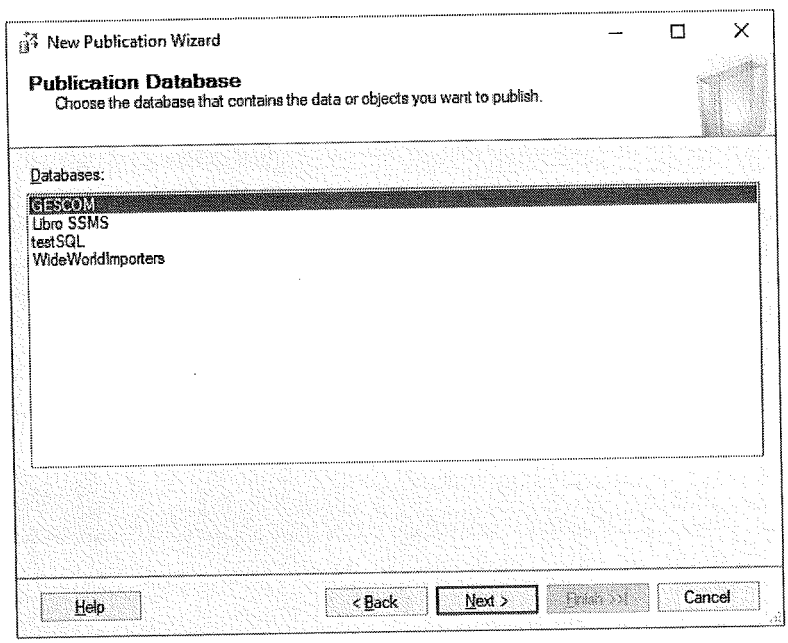


La base GESCOM está activada para la replicación transaccional

La creación de una publicación también se realiza por medio de un asistente en SQL Server Management Studio. Para crear una nueva publicación, es necesario seleccionar **New Publication** en el menú contextual asociado al nodo **Replication - Local Publications** del explorador de objetos.

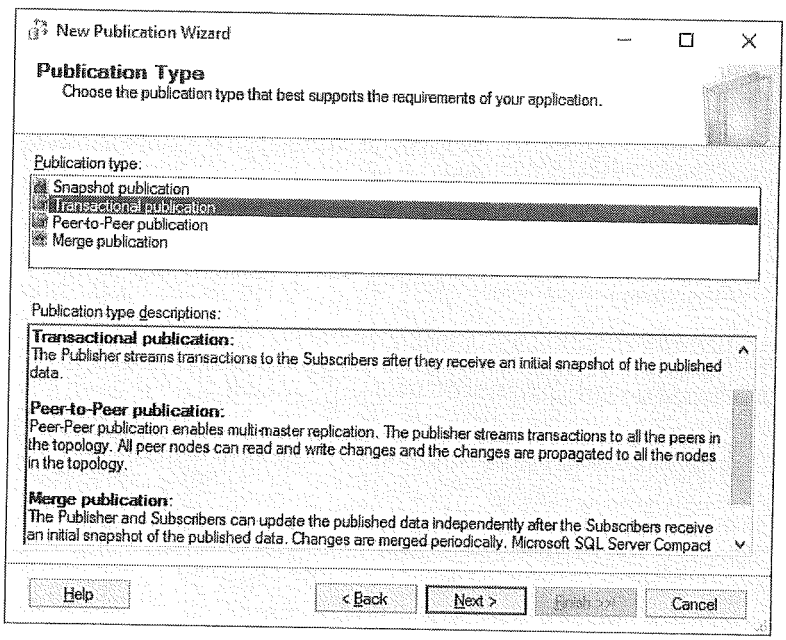
La primera etapa del asistente consiste en seleccionar la base de datos que contiene el objeto o los objetos que va(n) a participar en la publicación. Este cuadro de diálogo muestra todas las bases de datos de usuario, estén o no habilitadas para replicar.

En el ejemplo siguiente, solo la base GESCOM está habilitada para replicar, pero en la ventana se muestran todas las bases de datos.



El asistente solicita entonces indicar el tipo de replicación en el que va a participar esta publicación.

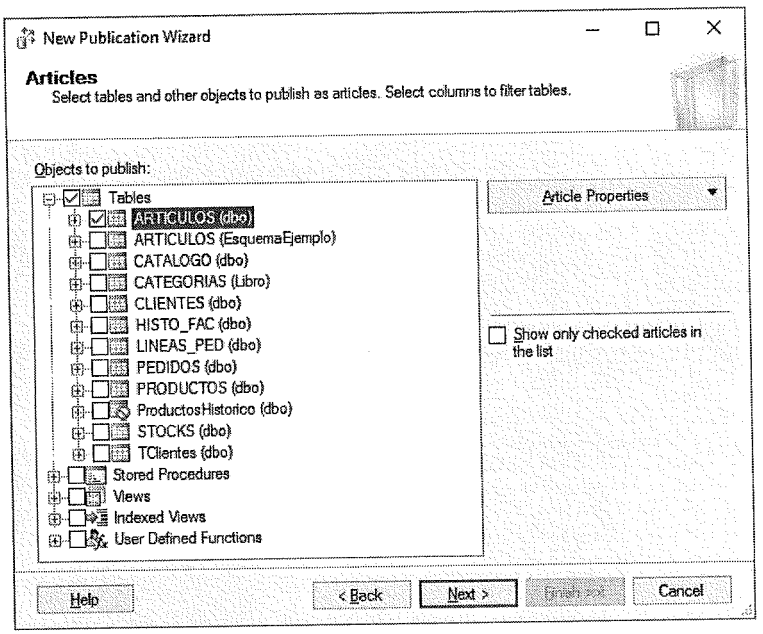
En el ejemplo siguiente, la publicación se define en el marco de una replicación transaccional; es decir, que los datos se modifican sobre el editor y estas modificaciones se envían a los suscriptores por el proceso de replicación.



La elección se centra en una publicación transaccional

Una vez definido el tipo de replicación, es posible seleccionar los elementos de la base que participan en esta publicación; es decir, definir los artículos.

En el ejemplo siguiente, la publicación va a tener un único artículo: la tabla de los artículos.

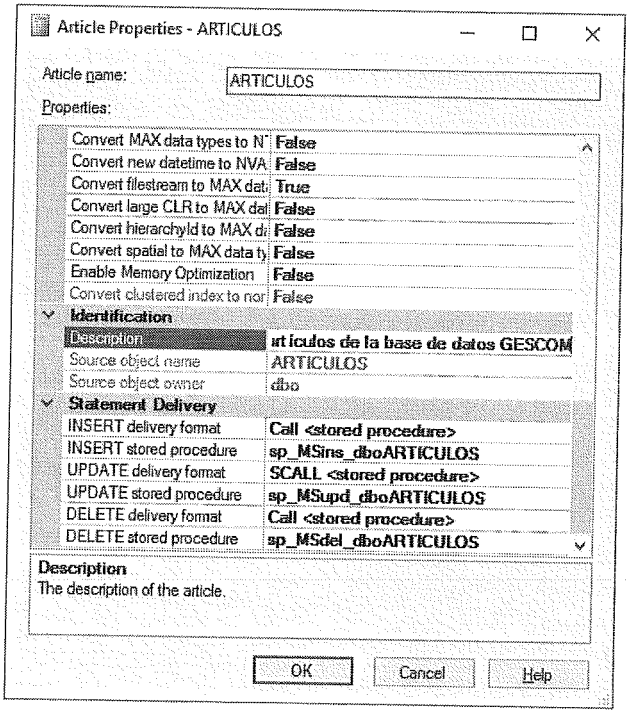


Selección de los objetos que se van a publicar

Observación

Solo las tablas que tienen una clave primaria pueden participar en una publicación como artículo.

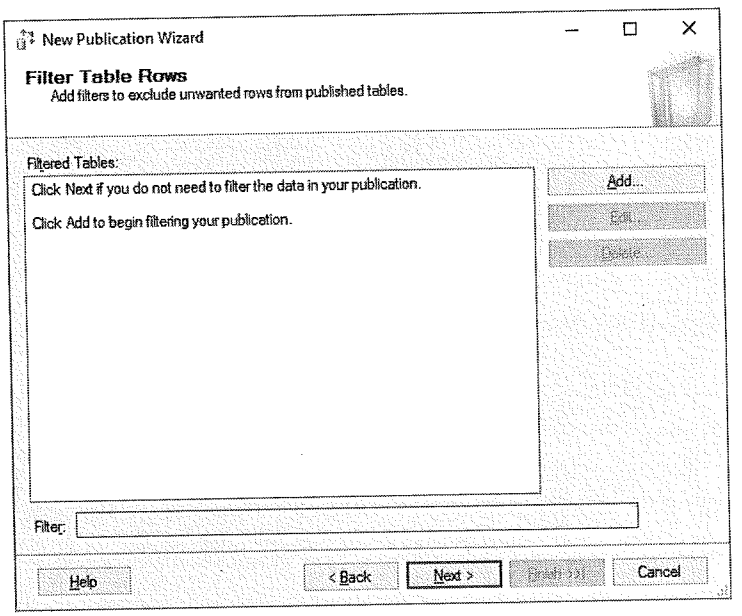
Es interesante visualizar las propiedades de cada artículo publicado para introducir una descripción y fijar el esquema y el nombre de la tabla de destino.



Fijar las propiedades generales de cada artículo

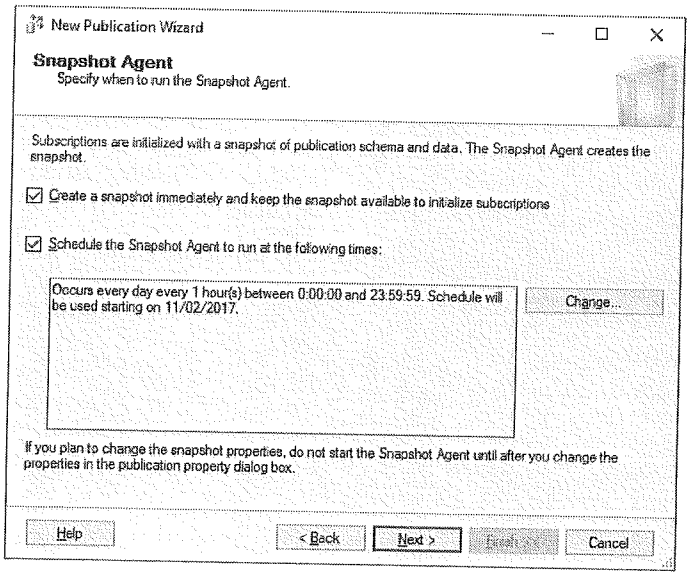
Las propiedades del artículo publicado permiten definir el comportamiento que hay que adoptar sobre los índices, los índices XML, los valores por defecto y las autorizaciones.

El asistente ofrece posteriormente la posibilidad de definir filtros para limitar el volumen de datos publicados. Por ejemplo, ¿es necesario publicar la totalidad de la tabla de artículos o bien hay que limitar la replicación a los artículos con un nombre determinado?

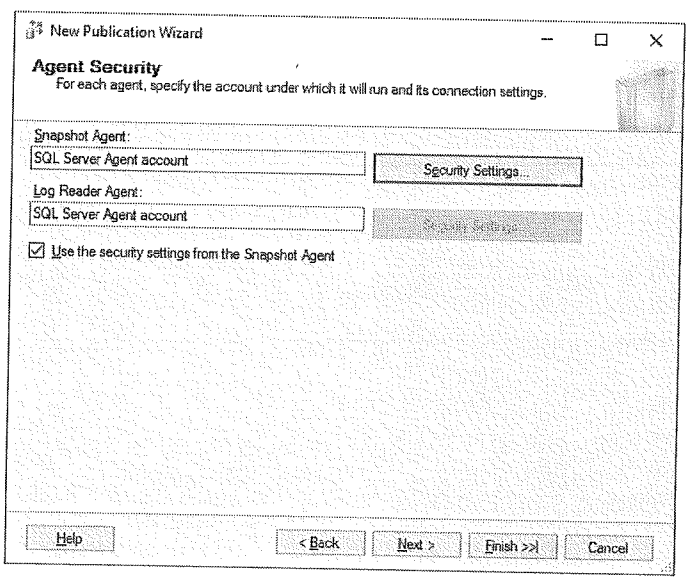


La etapa siguiente permite definir si es necesario o no realizar instantáneas y, si es preciso, planificar la creación periódica de nuevas instantáneas. Este último punto es particularmente interesante cuando el volumen de datos modificados es importante o bien cuando las suscripciones no se han realizado en el mismo período.

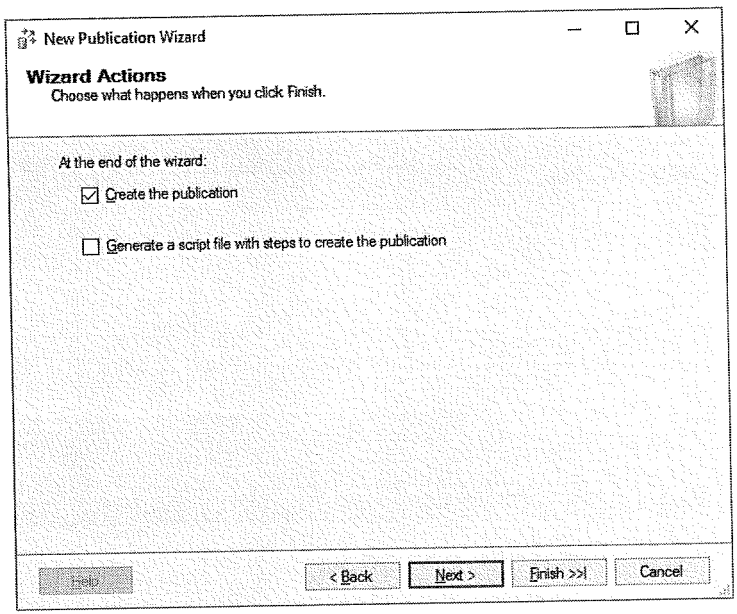
En el ejemplo siguiente, se solicita la creación de una instantánea y esta se efectuará una vez al día.



A continuación, el asistente solicita especificar las cuentas de seguridad que van a ser utilizadas por los agentes de instantáneas y de registro del LOG para conectarse al servidor.

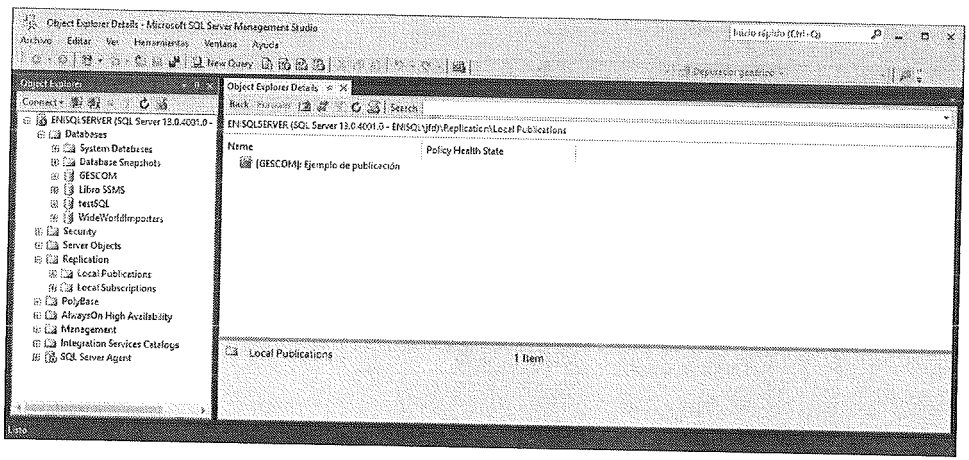


Antes de hacer clic en **Finish** para crear la publicación, el asistente ofrece la posibilidad de crear los scripts Transact SQL relativos a la creación de esta nueva publicación.



La última etapa del asistente permite dar nombre a la publicación y a continuación solicitar su creación.

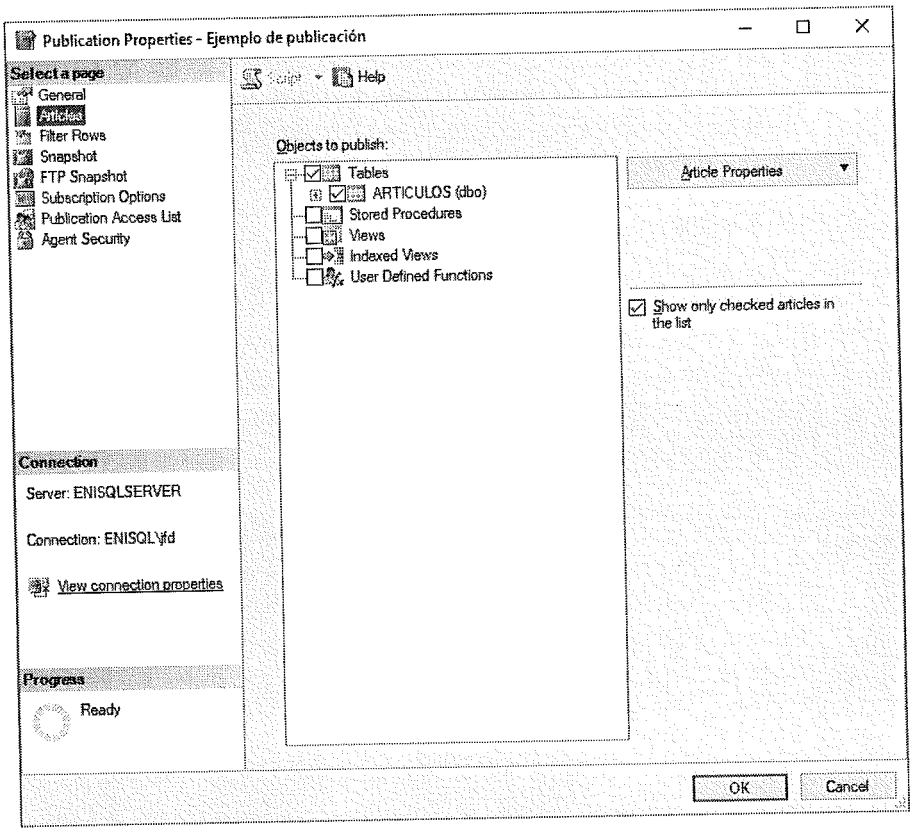
Después de su creación, es posible visualizar y modificar las publicaciones desde SQL Server Management Studio. Es posible, por medio de las propiedades de la publicación, modificar las elecciones hechas durante la ejecución del asistente de creación de la publicación.



Las diferentes publicaciones en Enterprise Manager

Los artículos

Los artículos de una publicación pueden ser de diferentes tipos. Es posible modificar los artículos que participan en una publicación a través de las propiedades de la publicación. A este nivel, es fácil añadir, modificar o eliminar artículos.



6.4 Las suscripciones

Suscribirse a una publicación significa que el suscriptor acepta los datos replicados sobre una base de datos de destino.

Al establecer la suscripción, es necesario definir el tipo del suscripción: de inserción o de extracción.

La suscripción de inserción significa que el agente de distribución se ejecuta sobre el servidor de distribución. La información entonces es enviada desde el servidor de distribución hacia el suscriptor por el agente de distribución.

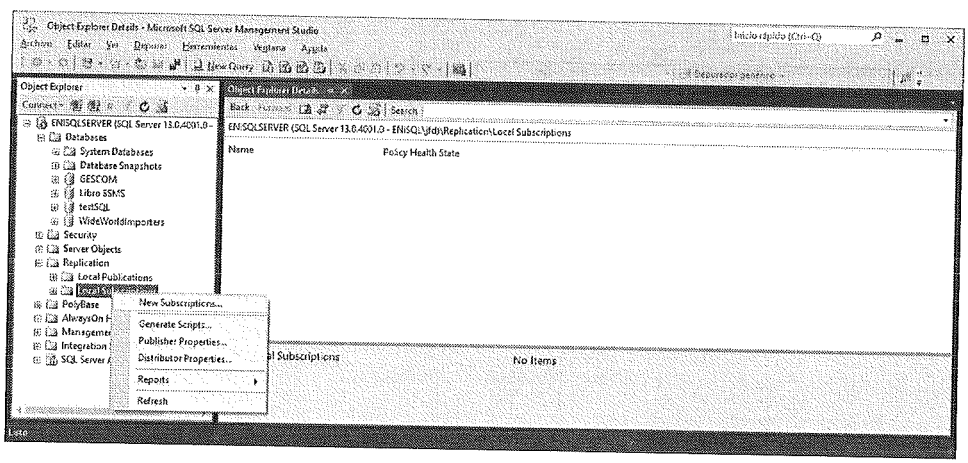
En el caso de la suscripción de extracción, cada suscriptor ejecuta su propio agente de distribución. Por lo tanto, es el suscriptor el que soporta esta carga de trabajo adicional. El agente de distribución envía la información desde el servidor de distribución hacia el suscriptor.

Cuando se implementan las suscripciones, interviene una etapa de sincronización para situar al mismo nivel al editor y al suscriptor. Su base de destino va a recibir el esquema y los datos publicados. La base de destino puede contener otras tablas.

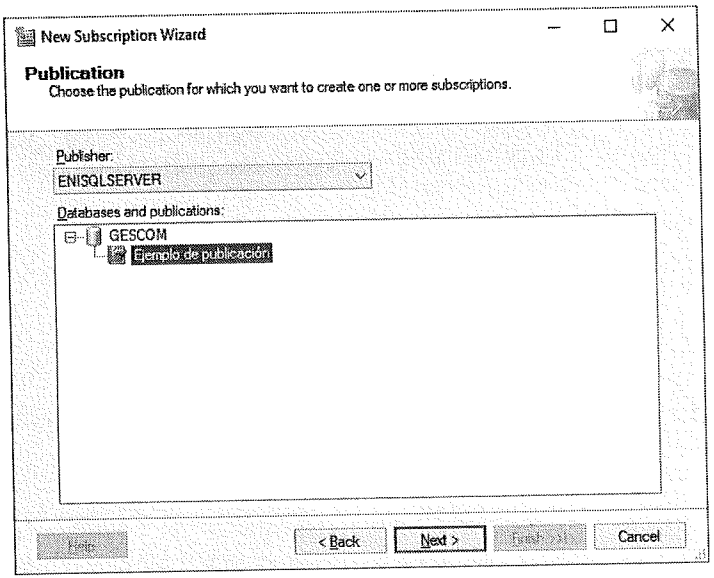
Solo se podrá crear una suscripción si existen la publicación y la base de datos de destino.

6.4.1 Utilización de los asistentes

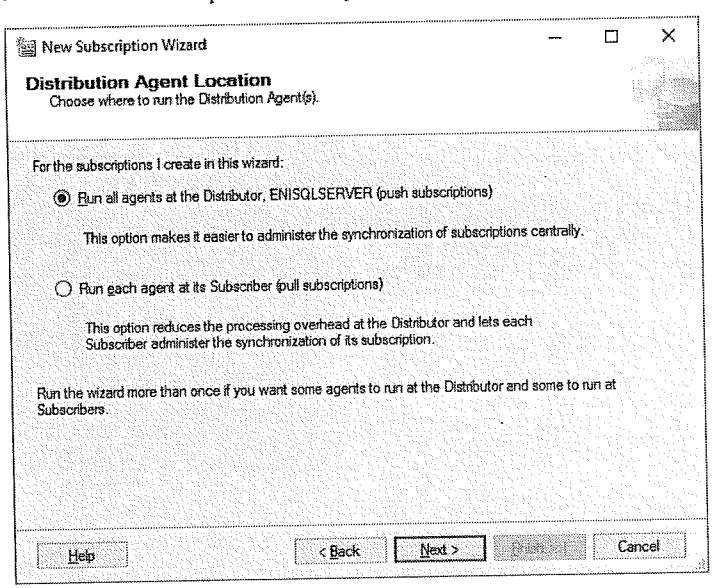
SQL Server Management Studio ofrece un asistente para crear nuevas suscripciones. Es posible lanzar este asistente seleccionando la opción **New Subscriptions** en el menú contextual del nodo **Replication - Local Subscriptions**, o bien asociado a una publicación. La siguiente pantalla ilustra este segundo método:



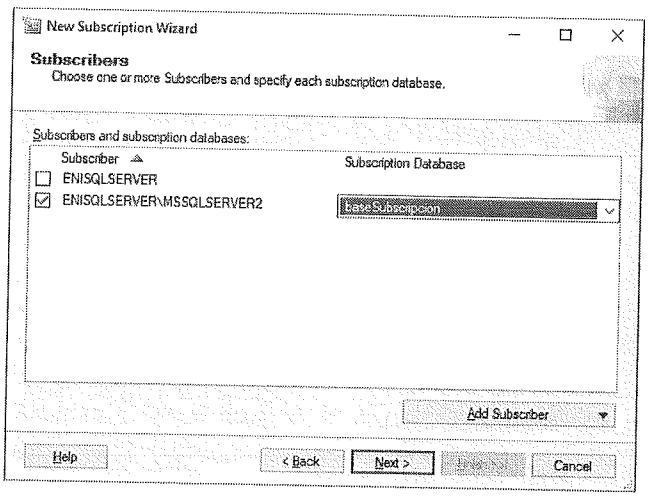
Después de la pantalla de bienvenida, el asistente solicita seleccionar la publicación relacionada con esta nueva suscripción. En el ejemplo siguiente, es la publicación creada anteriormente la relacionada con la nueva suscripción.



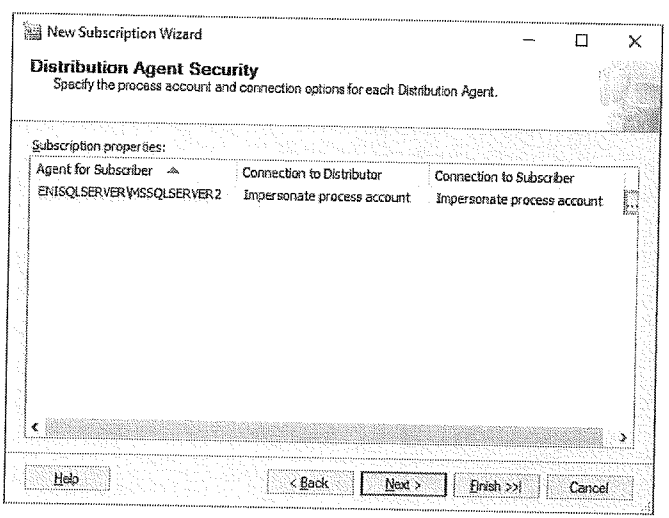
Posteriormente, el asistente solicita indicar el tipo de suscripción, es decir, si la suscripción será de tipo inserción (con ejecución del agente sobre el distribuidor) o de tipo extracción (con ejecución del agente sobre el suscriptor). En el ejemplo siguiente, se selecciona un tipo de suscripción de inserción.



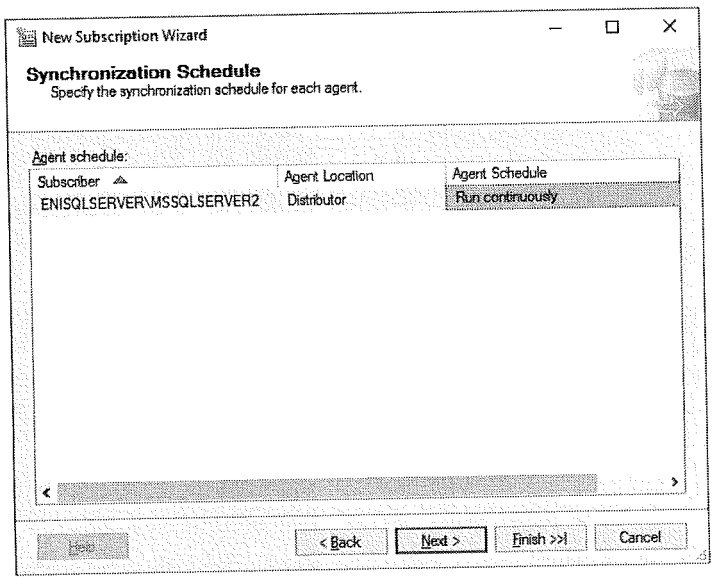
El asistente permite posteriormente seleccionar el suscriptor. Si el suscriptor no aparece en la lista propuesta, es necesario utilizar el botón **Add Subscriber** para inscribir el servidor y de esta manera poder seleccionarlo como un suscriptor. En el momento de seleccionar un servidor suscriptor, es necesario también especificar la base de datos destinataria. Si la base no existe, es posible crearla entonces. En el ejemplo siguiente, se selecciona la misma instancia de SQL Server como suscriptor.



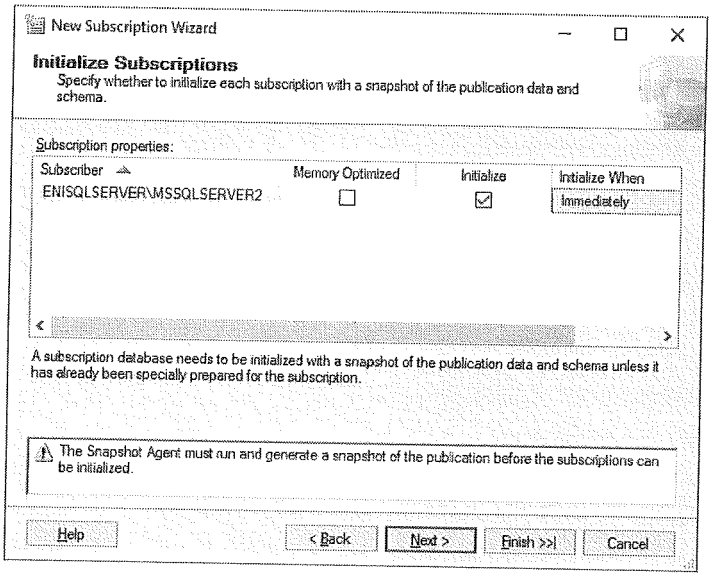
Posteriormente, el asistente solicita precisar las cuentas de seguridad que utilizarán los agentes de distribución para conectarse al servidor de distribución y al suscriptor.



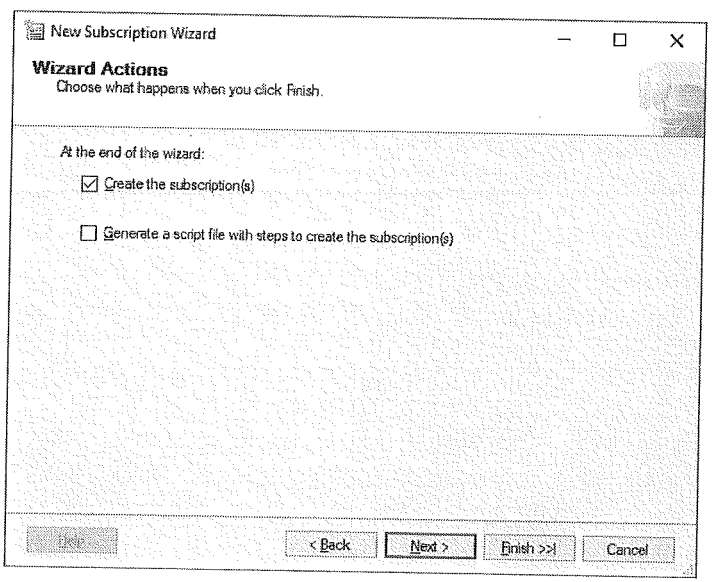
Ahora, el asistente solicita indicar el modo de ejecución del agente de distribución. En el ejemplo siguiente, el agente se ejecuta de manera continua, con el objetivo de informar lo más rápido posible de las modificaciones que tienen lugar sobre el servidor de publicación dirigidas hacia los suscriptores. Sin embargo, con la ayuda de una planificación, el agente puede ejecutarse con una periodicidad determinada, como por ejemplo durante las noches, cuando la carga de trabajo es menor.



El asistente hace el mismo tipo de pregunta para la ejecución de la instantánea, que puede ejecutarse bien de manera inmediata o bien en el momento de la primera sincronización.

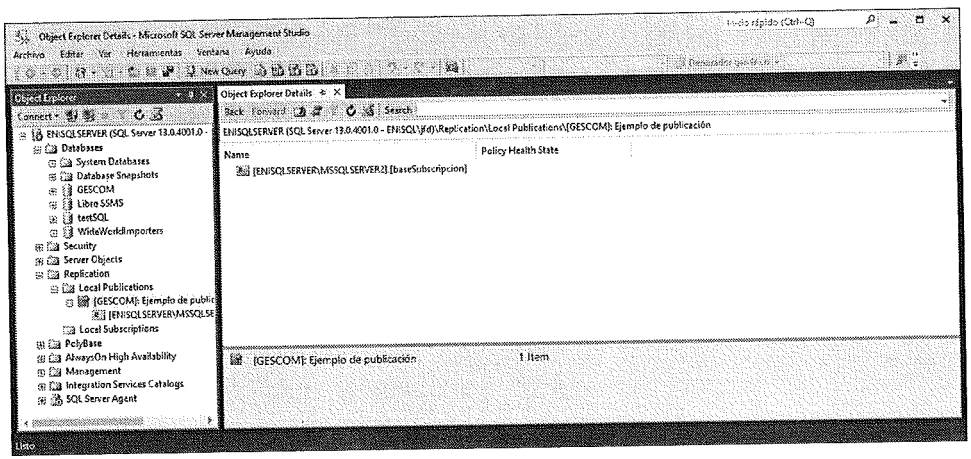


Como para la creación de la publicación, el asistente da la posibilidad de generar los scripts Transact SQL correspondientes.



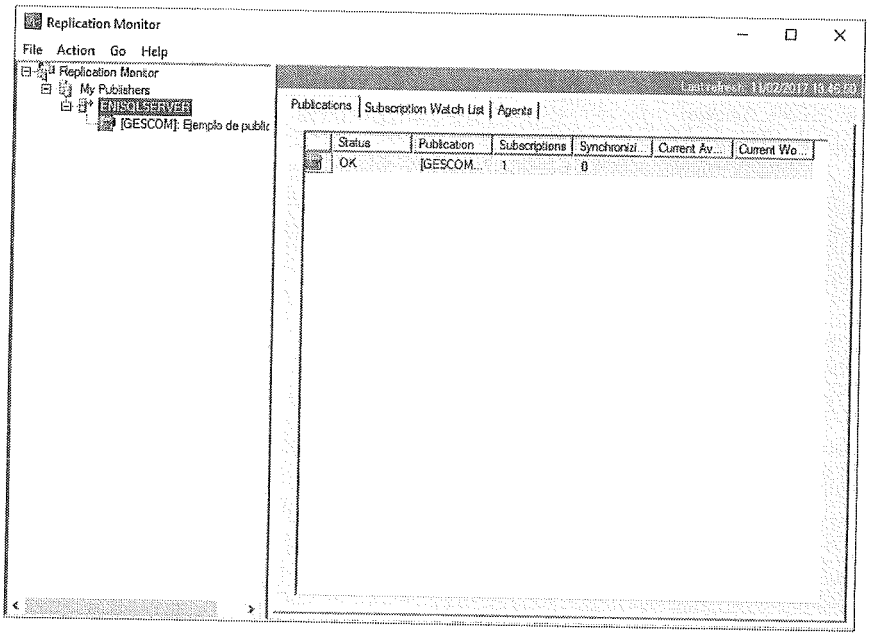
Después, el asistente presenta una pantalla de resumen antes de terminar la creación de la suscripción.

Desde SQL Server Management Studio, es posible visualizar el conjunto de las suscripciones definidas con relación a una publicación, pero también las suscripciones efectuadas localmente por el servidor.

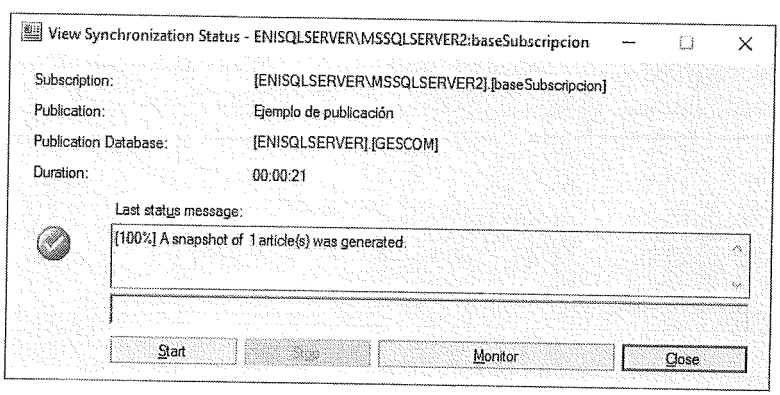


6.4.2 Vigilar la replicación

El monitor de replicación puede iniciarse desde el distribuidor para seguir la ejecución y la evolución de los diferentes agentes de replicación. Se accede a este monitor seleccionando **Launch Replication Monitor** desde el menú contextual asociado al nodo **Replication** del explorador de objetos de SQL Server Management Studio.



El menú contextual asociado a la publicación también permite obtener el resumen de la ejecución de los agentes de instantáneas y de lectura del diario.



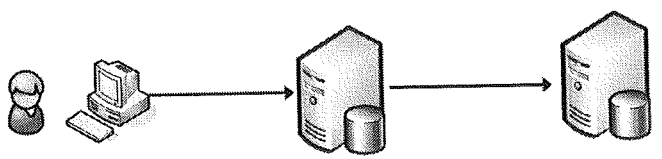
6.4.3 Eliminación

La eliminación de una suscripción entorpece las nuevas actualizaciones de la base de destino, pero esta última no se elimina ni se limpia. Será un administrador del servidor de destino el que elimine el esquema creado por la puesta en marcha de la publicación.

7. El acceso a los datos remotos

SQL Server permite a un usuario conectado localmente ejecutar una consulta sobre un servidor remoto. Este proceso se basa en la unión entre los servidores. Cuando la unión se establece entre dos servidores, el servidor que recibe por parte de alguno de sus usuarios una petición de ejecución de una consulta sobre otro servidor transmite la consulta al servidor remoto.

En el esquema siguiente, un usuario conectado a servidor1 puede solicitar la ejecución de una consulta sobre servidor2 sin tener que conectarse a servidor2. Efectivamente, como los dos servidores están asociados, es servidor1 el que se encarga de ejecutar la consulta sobre servidor2.



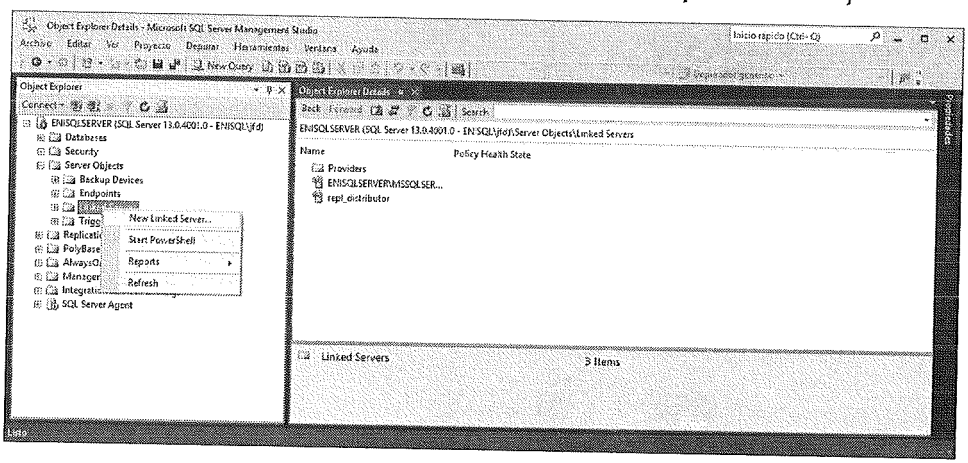
La ventaja de los servidores asociados radica en que es el servidor el que se encarga de la conexión sobre el servidor remoto. Esta operación es transparente para el usuario final.

Antes de poder trabajar con un servidor asociado, es necesario inscribirlo sobre el servidor local y después definir una política de gestión de las conexiones establecidas sobre el servidor asociado.

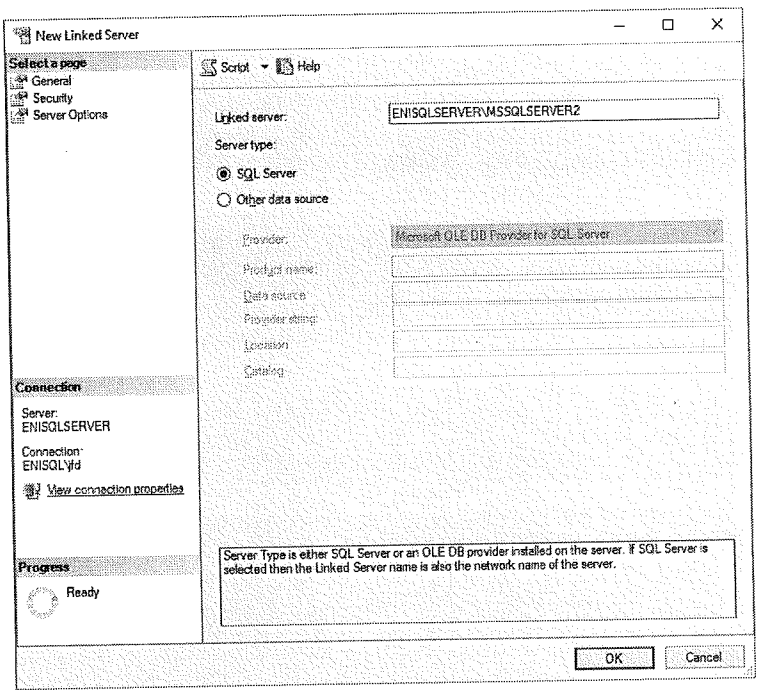
El concepto de servidores asociados permite a SQL Server establecer una relación de confianza con orígenes OLE DB que tienen la ventaja de acceder a los servidores remotos, emitir las consultas, operaciones de actualización, comandos y transacciones compartidas sobre los orígenes de datos heterogéneos.

7.1 Añadir un servidor asociado

Desde SQL Server Management Studio, es fácil definir un nuevo vínculo con un servidor remoto seleccionando la opción **New Linked Server** en el menú contextual asociado al nodo **Server Objects - Linked Servers** del explorador de objetos.



Se ejecuta el cuadro de diálogo relativo a la inscripción de un nuevo servidor asociado. Las tres páginas de la ventana permiten configurar completamente esta unión. Si se produce un error o un olvido en esta etapa, es posible remediarlo modificando las propiedades del servidor vinculado.

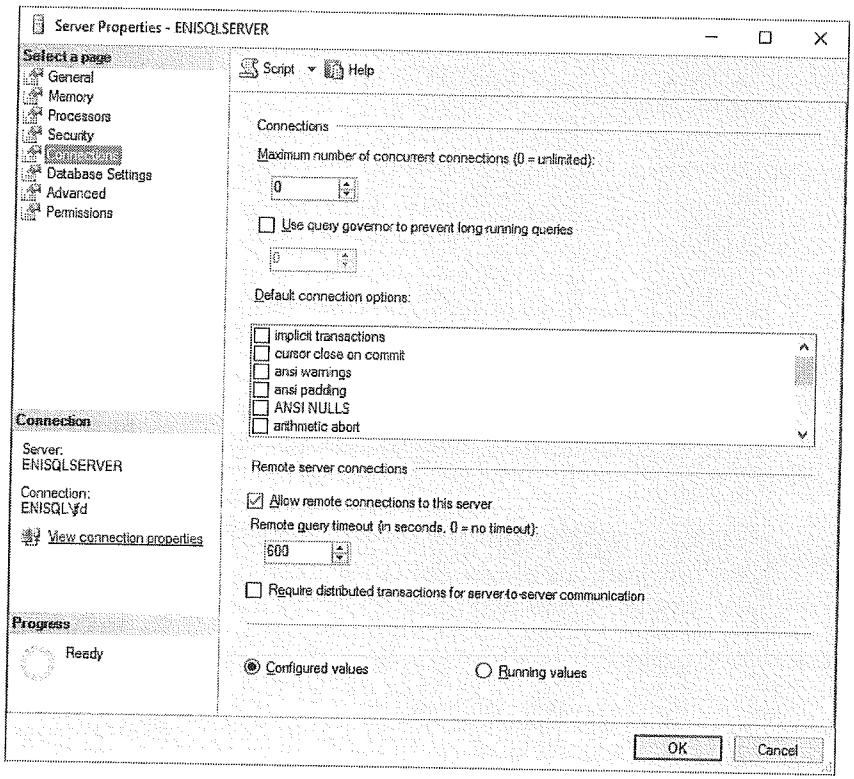


También es posible realizar estas operaciones en Transact SQL con los procedimientos **sp_addlinkedserver** para añadir un nuevo servidor vinculado y **sp_dropserver** para eliminar la vinculación con un servidor.

7.2 Gestionar los usuarios remotos

El servidor de origen no puede abrir una sesión sobre el servidor de destino si este último no autoriza las conexiones remotas. Para abrir la sesión sobre el servidor remoto, el servidor de origen utilizará la información de seguridad registrada a nivel del vínculo de los servidores.

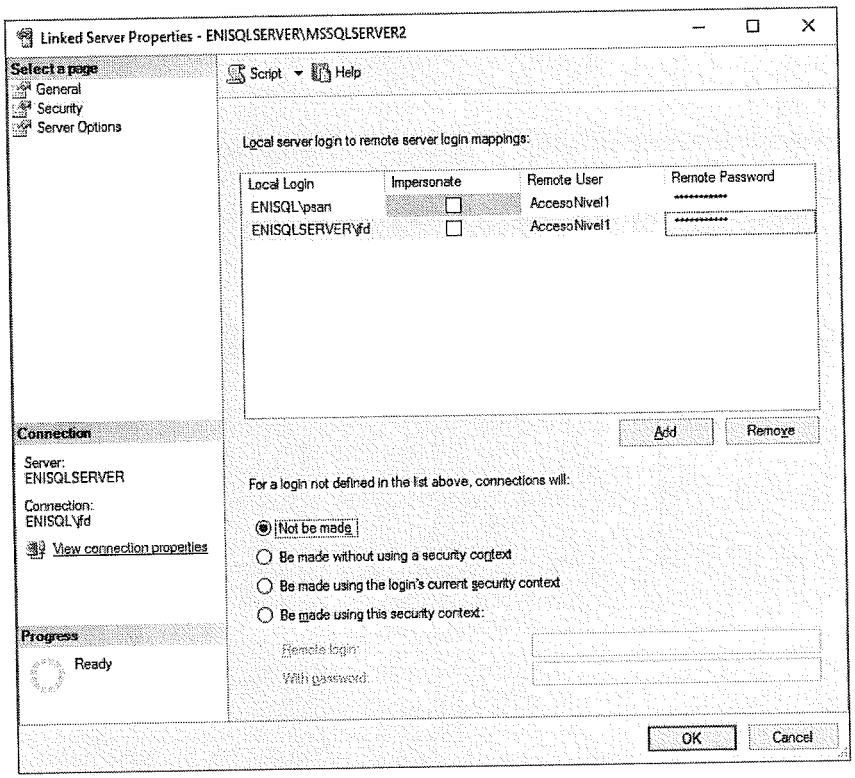
Con objeto de configurar el servidor de destino para que acepte las conexiones realizadas desde otro servidor, es necesario activar la propiedad **Allow remote connections to this server** en el servidor. Esta operación se realiza desde la ventana de visualización de las propiedades del servidor, página **Connections**, en la zona **Remote server connections**.



Para que el servidor de origen pueda abrir una conexión sobre el servidor vinculado, es necesario definir un mapeo entre los usuarios locales y los usuarios remotos. Así, solo algunos usuarios locales pueden acceder a los datos remotos. Además, las operaciones realizadas sobre el servidor vinculado están en el contexto de seguridad utilizado para establecer la conexión remota. Varios usuarios locales pueden acceder al servidor vinculado sirviéndose de la misma cuenta de conexión.

Desde SQL Server Management Studio, este mapeo de cuentas se realiza a partir de la ventana de las propiedades del servidor vinculado, en la página **Security**.

En el ejemplo siguiente, las conexiones locales ENISQL\psan y ENISQLSERVER\jfd están asociadas a la conexión remota AccesoNivel1. Los otros usuarios locales no pueden acceder al servidor vinculado.



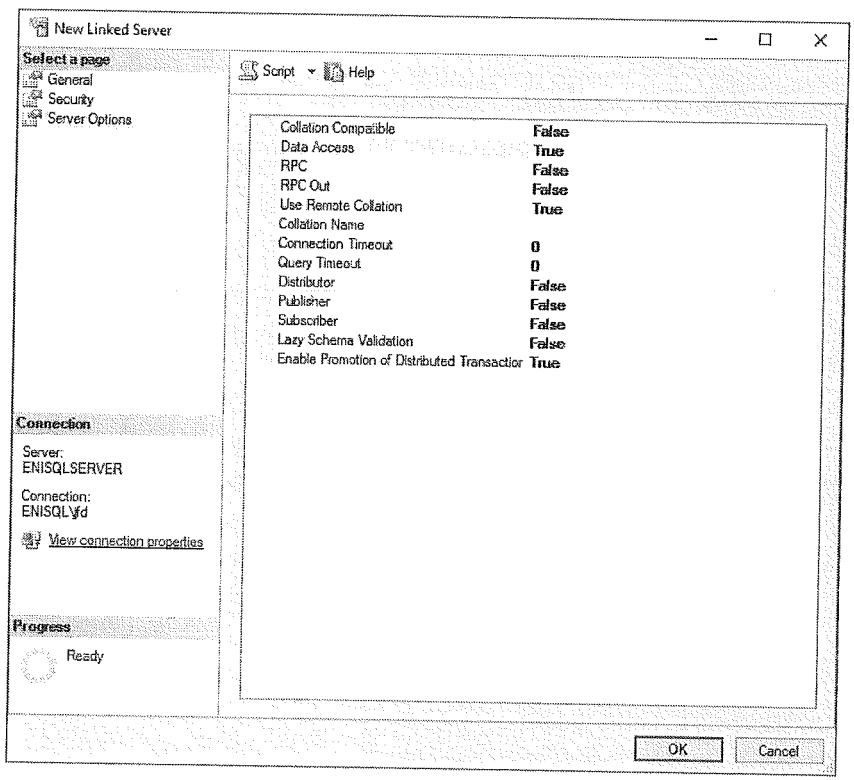
También se pueden realizar todas estas operaciones en Transact SQL con los procedimientos almacenados **sp_addlinkedsrvlogin** y **sp_droplinkedsrvlogin**.

7.3 Ejecución de una consulta distribuida

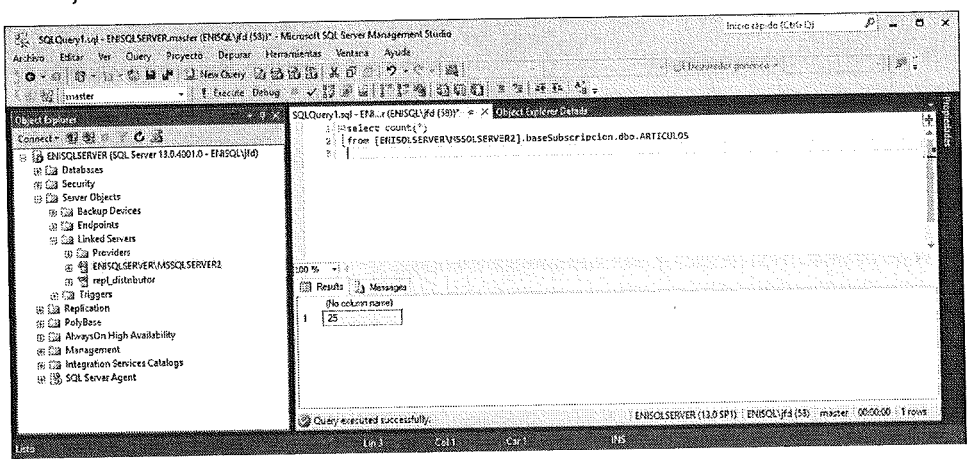
Los clientes que se conectan al servidor SQL sobre el que se definen los servidores vinculados pueden ejecutar consultas que van a utilizar los diferentes orígenes de datos gestionados directamente por el servidor o accesibles por medio de un servidor vinculado. Dentro de una misma consulta, es posible llamar a datos que provengan de varios orígenes de datos.

Para acceder a datos almacenados sobre un servidor vinculado, es necesario dar el nombre completo de los objetos a los que se desea acceder.

El acceso a los datos solo es posible si el servidor vinculado se configura para permitir este acceso.



Desde el momento en que se efectúa esta configuración, es posible acceder a la información indicando el nombre completo de los objetos; es decir, servidor.base.esquema.objeto.



Ejemplo de consulta que utiliza datos en servidores vinculados

Observación

Las funciones OPENQUERY y OPENROWSET permiten acceder a orígenes de datos OLE DB sin tener que vincular los servidores.

Capítulo 8
Copia de seguridad

1. Introducción

La gestión de las copias de seguridad es una de las tareas más importantes que debe realizar el administrador de bases de datos. Si las operaciones de copia de seguridad se planifican con exactitud y rigor, es perfectamente posible realizarlas en forma de trabajos automatizados y el responsable será avisado por correo electrónico del correcto desarrollo de las operaciones.

Las copias de seguridad se realizan para prevenir las posibles pérdidas de datos como consecuencia de:

- Un problema de soporte.
- Errores de usuario.
- Una pérdida permanente del servidor.

SQL Server permite hacer una copia de seguridad de la base de datos incluso cuando los usuarios están conectados. Esta copia de seguridad va a tener en cuenta todos los archivos que forman la base de datos y va a registrar su ubicación. El proceso de copia de seguridad asegura la coherencia de los datos y los diarios, capturando todas las actividades que se producen durante dicho proceso.

Aunque la base de datos permanece accesible durante la copia de seguridad, algunas operaciones son imposibles, a saber:

- Crear o modificar una base de datos (fundamentalmente la extensión automática del diario (o registro) de transacciones).
- Crear un índice.

- Ejecutar operaciones que no están trazadas por un fichero de log, ya que el proceso de copia de seguridad utiliza el diario para garantizar la coherencia de los datos.

2. Planificación

La planificación de las operaciones de copia de seguridad implica también las de restauración, porque la una no se hace sin la otra. Son las exigencias en materia de disponibilidad de datos las que van a determinar los criterios para el establecimiento de las copias de seguridad. Para realizar esta planificación, es necesario reflexionar sobre todos los incidentes que pueden surgir y saber cuáles son las necesidades de copia de seguridad para restaurar lo mejor posible la información. Será posible automatizar todos estos trabajos de copia de seguridad y probar la restauración, validando el buen funcionamiento de los procedimientos de copia de seguridad.

2.1 Preguntas

Las principales preguntas que es necesario plantearse para planificar lo mejor posible las copias de seguridad son:

- ¿Cuál es el tamaño de cada base de datos?
- ¿Cuál es el volumen de las modificaciones de datos?
- ¿Algunas tablas están sujetas a más modificaciones que otras?
- ¿Cuánto tiempo puede permanecer no operativa la base de datos?
- ¿La pérdida de modificaciones es crucial?
- ¿Es fácil recrear los datos perdidos?
- ¿Cuáles son los períodos importantes de utilización de la base de datos?
- ¿La base sufre sobrecargas de trabajo puntuales durante las que no es posible lanzar una copia de seguridad?
- ¿Los usuarios deben continuar accediendo a los datos durante las operaciones de copia de seguridad?
- ¿Cuál es el lapso de tiempo entre las operaciones de truncado (eliminación de la parte inactiva) del diario de transacciones?
- ¿Las copias de seguridad se conservan de manera cíclica?
- ¿El servidor SQL está en un clúster?
- ¿El servidor SQL está en un entorno multiservidor con una administración centralizada?

La duración de las operaciones de copia de seguridad va a depender fundamentalmente del soporte sobre el que efectúen las copias de seguridad.

Son posibles dos soportes: las bandas (con la condición de que el lector de bandas se instale en el servidor SQL) y los archivos.

2.2 Elegir una estrategia de copia de seguridad

Para cada base de datos, es necesario elegir una estrategia de copia de seguridad, que va a ser una combinación de copias de seguridad completas de bases de datos y copias de seguridad del diario de transacciones. Para mejorar el rendimiento de las copias de seguridad, SQL Server propone copias de seguridad diferenciales. Además, todas estas operaciones pueden realizarse sobre la totalidad de la base o únicamente sobre un grupo de archivos.

Un buen conocimiento de los diferentes métodos de copia de seguridad permite establecer el plan de copia de seguridad que mejor responda a las exigencias encontradas en la planificación.

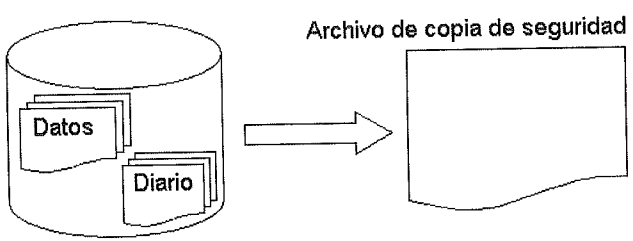
Observación

La estrategia de copia de seguridad debe fijarse para cada base de datos en función de las necesidades y de las evoluciones de cada una de ellas.

2.2.1 Copia de seguridad de una base de datos

La copia de seguridad total de una base de datos permite proporcionar un punto de partida para las restauraciones. Si únicamente se realizan copias de seguridad completas de las bases de datos, en caso de problema las transacciones validadas desde la última copia de seguridad completa se perderán.

Las copias de seguridad completas exigen relativamente mucho tiempo y ocupan un espacio considerable en el soporte de copias de seguridad. Es por esto por lo que, aunque constituyan un punto de partida obligatorio para toda estrategia de copia de seguridad, las copias de seguridad completas de base de datos se adaptan mejor a las bases de datos de pequeño volumen y para las que es posible reproducir fácilmente todas las transacciones que tienen lugar desde la última copia de seguridad completa.

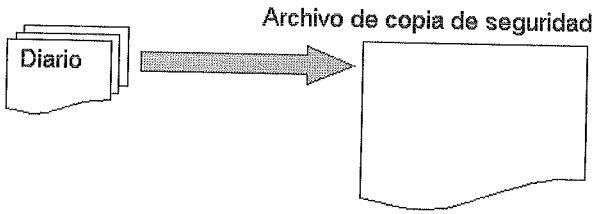


2.2.2 Copia de seguridad del diario de transacciones

Como complemento de las copias de seguridad completas, siempre es posible establecer una política de copia de seguridad de los diarios de transacciones. La copia de seguridad de los diarios presenta dos ventajas importantes.

Es posible recuperar la totalidad o una gran parte de las transacciones validadas desde la última copia de seguridad completa de la base.

El tamaño del archivo de diario no corre el riesgo de crecer de manera anárquica, ya que en cada copia de seguridad del diario es posible solicitar truncarlo. De esta manera, disminuye el riesgo de saturar el disco después de que aumente la extensión del archivo diario. La copia de seguridad de los diarios puede realizarse por medio de un trabajo planificado para una ejecución regular.



SQL Server genera puntos de control de sincronización (CHECKPOINT) de manera automática. El objetivo de estos puntos de sincronización es almacenar en disco el conjunto de transacciones validadas para las que las modificaciones todavía están en memoria. Así, en caso de restauración automática posterior a una parada repentina del servidor, el volumen de datos que hay que restaurar será menor, ya que solo afecta a los datos pertenecientes a las transacciones validadas desde el último punto de sincronización.

El punto de sincronización puede dispararse puntualmente por medio de la instrucción Transact SQL CHECKPOINT.

Sintaxis

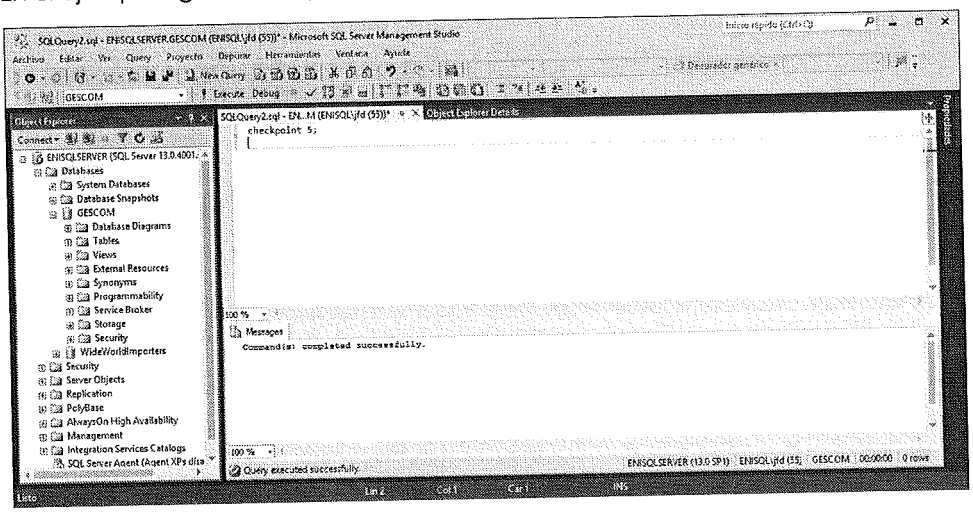
```
CHECKPOINT [valor][;]
```

`Valor`

Precisa el número de segundos de los que dispone SQL Server para terminar el punto de sincronización.

Ejemplo

En el ejemplo siguiente, el punto de sincronización se realiza en 5 segundos.



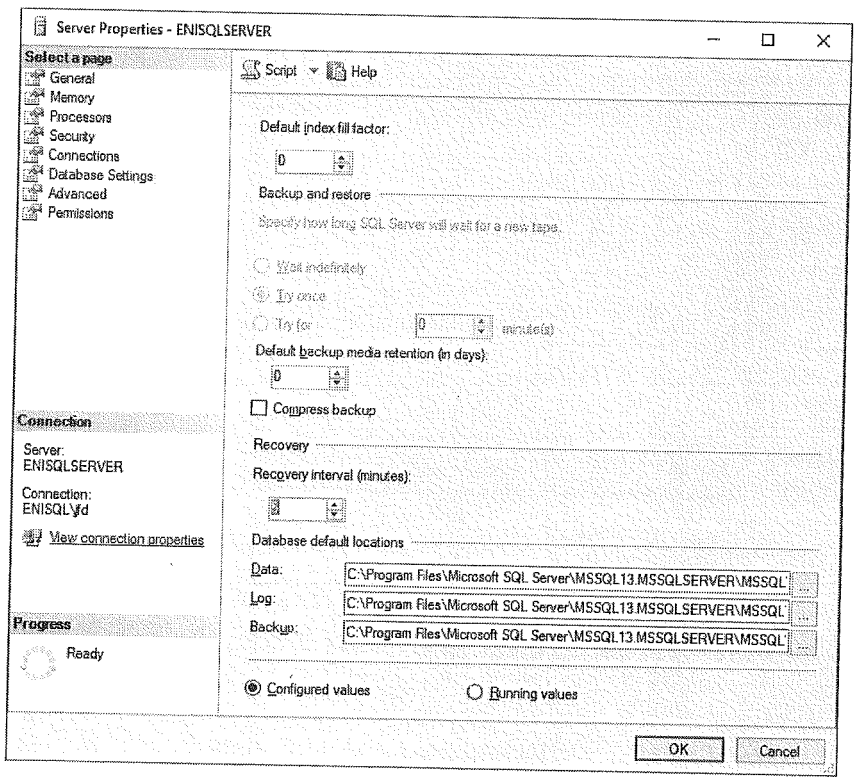
El punto de sincronización puede también realizarse de manera automática en función de algunos criterios:

- Observancia de los parámetros fijados con ayuda de la opción recovery interval.
- El diario está completo a más del 70 %, mientras que la base está configurada para truncar el diario durante la ejecución de los puntos de sincronización.
- El motor de base de datos se para, salvo si esta parada se realiza mediante la instrucción SHUTDOWN WITH NOWAIT.

La opción **RECOVERY INTERVAL** permite definir, a nivel del servidor, el número máximo de minutos que puede durar una restauración automática de la base desde el último punto de sincronización. La frecuencia de los puntos de sincronización la fija SQL Server en función de la actividad de actualización de la base. Por defecto, el valor de este parámetro es 0, lo que significa que SQL Server gestiona automáticamente la frecuencia de los puntos de sincronización. Se recomienda conservar este valor.

Sin embargo, si los puntos de sincronización están mal adaptados en comparación con la actividad del servidor, es posible modificar el valor de este parámetro, bien con el procedimiento **sp_configure**, bien desde la ventana de las propiedades del servidor desde SQL Server Management Studio.

Como se ilustra en la pantalla siguiente, el intervalo de recuperación se fija en la página **Database Settings**.

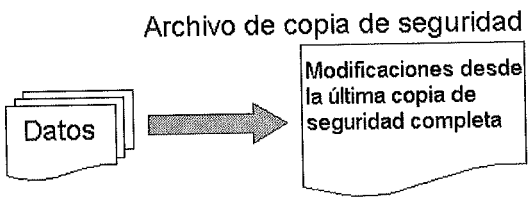


Si la base de datos usa el modo de recuperación simple, el diario solo se utiliza durante las recuperaciones automáticas, por lo que es inútil conservar la información sobre las transacciones que han finalizado antes del último punto de sincronización. Para evitar que el tamaño del diario crezca de manera ilimitada, SQL Server elimina la parte inactiva del diario en cada punto de sincronización.

2.2.3 Las copias de seguridad diferenciales

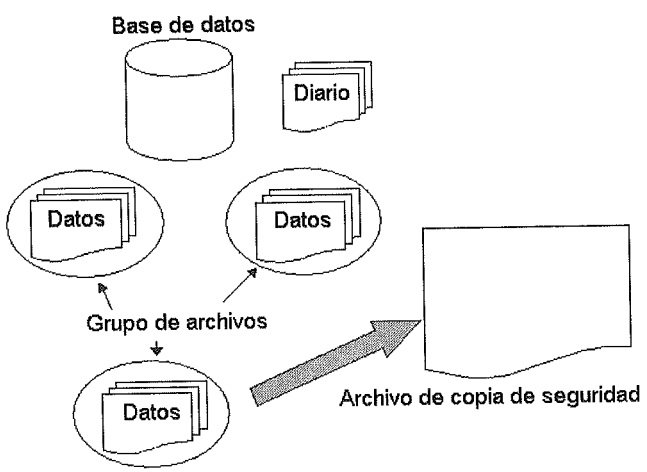
Si las copias de seguridad del diario son operaciones pesadas debido a que el volumen de los diarios puede crecer rápidamente, una alternativa interesante puede ser la puesta en marcha de copias de seguridad diferenciales. Las copias de seguridad diferenciales solo van a tener en cuenta los datos modificados desde la última copia de seguridad completa.

Las copias de seguridad diferenciales son más rápidas y menos voluminosas que las copias de seguridad completas y, asociadas a las copias de seguridad del diario de transacciones, pueden constituir una solución de copia de seguridad a la vez rápida y con buen rendimiento.



2.2.4 Las copias de seguridad por grupos de archivos

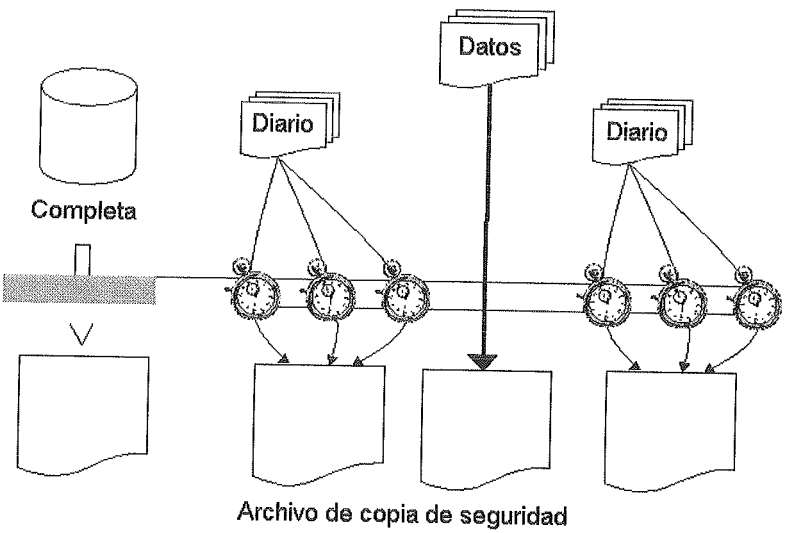
Si la base de datos tiene un volumen importante de datos, las copias de seguridad completas y diferenciales pueden tardar mucho en ejecutarse. Para reducir este tiempo, se pueden hacer copias de seguridad de los datos por grupos de archivos. Esta operación es posible si se han definido grupos de archivos en el momento de crear la base de datos.



2.2.5 Las combinaciones posibles

La mejor solución para realizar copias de seguridad se basa en una combinación de los diferentes métodos de copia de seguridad.

Por ejemplo, es posible combinar las copias de seguridad completas, las copias de seguridad del diario de transacciones y las copias de seguridad diferenciales para minimizar a la vez el volumen y el tiempo de las copias de seguridad y la pérdida de transacciones validadas. Poniendo en marcha estas operaciones en forma de trabajos planificados con notificación de fin de ejecución, las operaciones de copias de seguridad pueden ejecutarse automáticamente sin esfuerzo por parte del administrador.



Si una base de datos se extiende sobre varios grupos de archivos (primary, datos...), es posible realizar copias de seguridad por grupos de archivos. Para cada grupo de archivos, es necesario aplicar una estrategia de copia de seguridad (completa, diferencial y diario de transacciones). No es preciso efectuar simultáneamente el mismo tipo de copia de seguridad sobre todos los grupos de archivos.

En la combinación anterior, se utilizan los tres tipos de copia de seguridad. La copia de seguridad completa constituye el punto de partida obligatorio. Seguidamente, los diarios se salvan de forma regular para perder el mínimo de datos posible. Para minimizar los tiempos de restauración, se efectúa una copia de seguridad diferencial con el objetivo de conservar todas las modificaciones que han tenido lugar desde la última copia de seguridad completa.

Observación

Todas las estrategias de copia de seguridad comienzan siempre por una copia de seguridad completa de la base.

3. Establecimiento de las copias de seguridad

Las operaciones de copia de seguridad pueden ponerse en práctica bien mediante SQL Server Management Studio o bien con la ayuda de procedimientos almacenados en Transact SQL. Como siempre, la solución gráfica ofrece facilidad de aplicación, pero los comandos Transact SQL permiten acceder a la totalidad de las opciones propuestas.

3.1 Los modos de recuperación

Las posibilidades que se ofrecen en materia de copia de seguridad, y por lo tanto de restauración, están directamente asociadas al modo de configuración definido en cada base de datos. Los tres modos de recuperación que es posible configurar son:

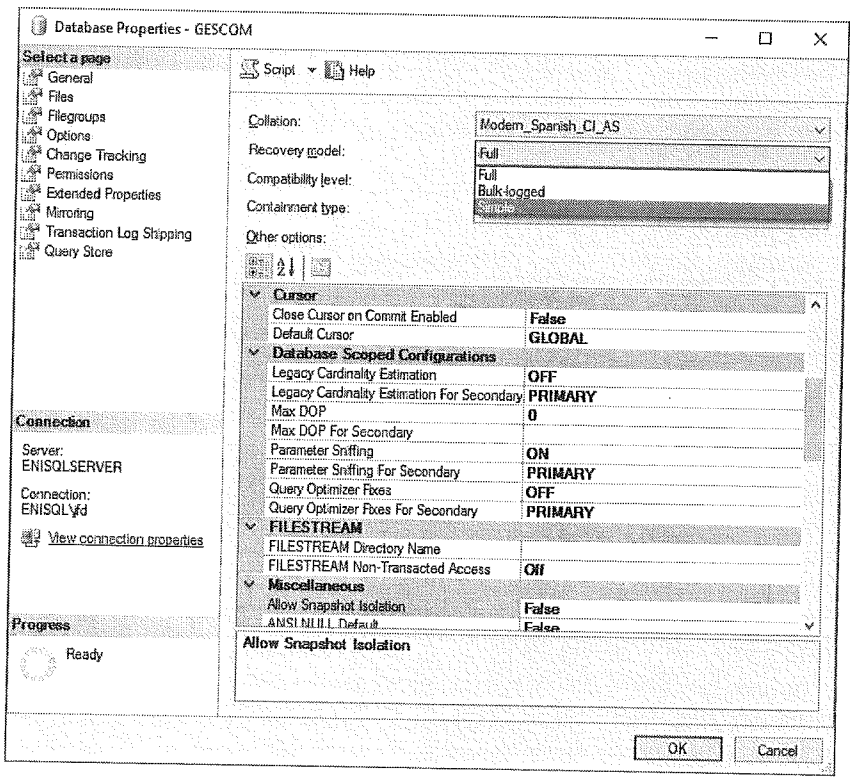
- **El modo de recuperación simple**: el diario se utiliza únicamente para garantizar la persistencia de las operaciones realizadas sobre los datos. Se trunca en cada punto de sincronización.
- **El modo de recuperación completo**: en este modo, todas las transacciones se registran en el diario y permanecen almacenadas incluso después del punto de sincronización. En caso de error, la pérdida de datos se reduce, ya que la operación de restauración es posible hasta el punto de fallo (si se ha adoptado una política correcta de copia de seguridad). Se trata del modo de recuperación por defecto definido en las bases de datos de usuario.
- **El modo de recuperación de registro masivo**: en este modo de recuperación avanzado, no solo se guarda en el diario la información relativa a las transacciones, sino también algunas operaciones que afectan a los datos, como la creación de índices.

Para configurar el modo de recuperación, es posible ejecutar la instrucción ALTER DATABASE.

Sintaxis

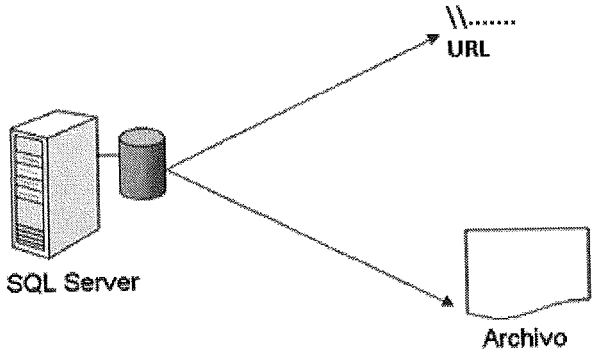
```
ALTER DATABASE nombreBaseDeDatos
SET RECOVERY { FULL | BULK_LOGGED | SIMPLE }
```

También es posible modificar las propiedades de la base desde SQL Server Management Studio. Este último caso se ilustra a continuación para configurar la base GESCOM en modo de recuperación completa.



3.2 El destino de las copias de seguridad

Las copias de seguridad pueden enviarse a diferentes soportes: disco duro, url.



3.2.1 Disco duro

Las copias de seguridad en disco se realizan en el archivo del sistema operativo. La copia de seguridad se puede realizar en un disco local o remoto. Para que SQL Server pueda acceder a un recurso compartido, este debe estar mapeado a un lector de red a nivel del usuario de Windows, en el contexto en el que se ejecuta el servicio. Es posible utilizar el procedimiento **xp_cmdshell** para definir este lector de red.

El procedimiento generado para la ejecución de **xp_cmdshell** usa el contexto de seguridad del servicio SQL Server. Para limitar el riesgo de un uso no deseado de este procedimiento, está deshabilitado por defecto. Además, en caso de que la cuenta de seguridad que ejecuta **xp_cmdshell** no sea miembro del rol sysadmin, la ejecución del procedimiento solo será posible si se ha definido una cuenta de proxy, usando **xp_cmdshell_proxy_account**.

Observación

Es preferible realizar las copias de seguridad sobre un disco diferente al que contiene los archivos de diario y de datos de la base de datos, para prevenir las averías en el disco.

Cuando se termina la copia de seguridad de un archivo, es prudente hacer copia de seguridad de este archivo sobre un soporte extraíble (banda) para almacenar las copias de seguridad en un lugar distinto a donde se encuentra el servidor.

También se puede especificar el destino de la copia de seguridad con una URL.

3.3 Los principales parámetros

La puesta en marcha de las copias de seguridad puede realizarse por medio de los procedimientos almacenados o bien por SQL Server Management Studio. Cuando las etapas que constituyen una operación de copia de seguridad son validadas, es posible reunirlas para formar un trabajo planificado y por lo tanto gestionado por el Agente SQL Server.

3.3.1 Los permisos

Para realizar una operación de copia de seguridad, el usuario debe tener algunos permisos. Hay tres roles predefinidos que contienen las autorizaciones suficientes para hacer la copia de seguridad de una base de datos. Por supuesto, es posible definir roles propios y asignarles las autorizaciones necesarias.

Los usuarios miembros de uno de los roles siguientes pueden realizar una copia de seguridad de base de datos:

- Rol de servidor **sysadmin**.
- Rol de base de datos **db_owner**.
- Rol de base de datos **db_backupoperator**.

3.3.2 La copia de seguridad de las bases de datos de sistema

La base de datos de sistema que contiene toda la información relativa al buen funcionamiento del servidor es la base **Master**. Se debe hacer una copia de seguridad de esta base después de cada modificación, sobre todo después de la creación de bases de datos de usuario. Efectivamente, si la base de datos no está referenciada en **Master**, es imposible acceder a ella. No hay que olvidar que la base **Master** no solo contiene la definición de todas las conexiones, sino también todas las referencias hacia los servidores vinculados.

Observación

Para reconstruir las bases de datos de sistema, es necesario pasar por el programa de instalación y elegir la opción REBUILDDATABASE.

La base **MSDB** contiene todos los trabajos planificados, los paquetes SIS y las operaciones de replicación, así como la definición de las alertas y de los operadores.

La base **MODEL** sirve de base de inicio para todas las bases de datos de usuario.

Se debe realizar una copia de estas bases de datos de sistema después de cada modificación. En período de funcionamiento, las modificaciones realizadas sobre estas bases normalmente son escasas y por lo tanto no es útil hacer copia de la base de datos si esta no evoluciona. De esta manera, es posible reducir los tiempos de copia de seguridad.

3.3.3 La copia de seguridad de las bases de datos de usuario

Estas son las bases de datos más oportunas para hacer copias de seguridad, ya que contienen toda la información de la empresa. Según la importancia de los datos que almacenan y el número de modificaciones que reciben, se adoptará un plan de copia de seguridad concreto. También es interesante ver qué etapas necesitan una copia de seguridad total de la base de datos:

- Después de la creación de la base.
- Después de la creación de un índice: el diario de transacciones graba la creación del índice, pero algunas veces la creación de un índice puede exigir más tiempo que la restauración de la totalidad de la base.
- Después de la eliminación del contenido del diario de transacciones, una copia de seguridad total permite no perder datos.
- Después de la ejecución de un comando que no está registrado en un archivo de log.

3.3.4 Los archivos de copia de seguridad

SQL Server no trabaja con el concepto de archivos de copia de seguridad, sino más exactamente con unidades de copia de seguridad. Existen dos categorías de unidades de copia de seguridad:

- Las unidades de copia de seguridad lógica.
- Las unidades de copia de seguridad física.

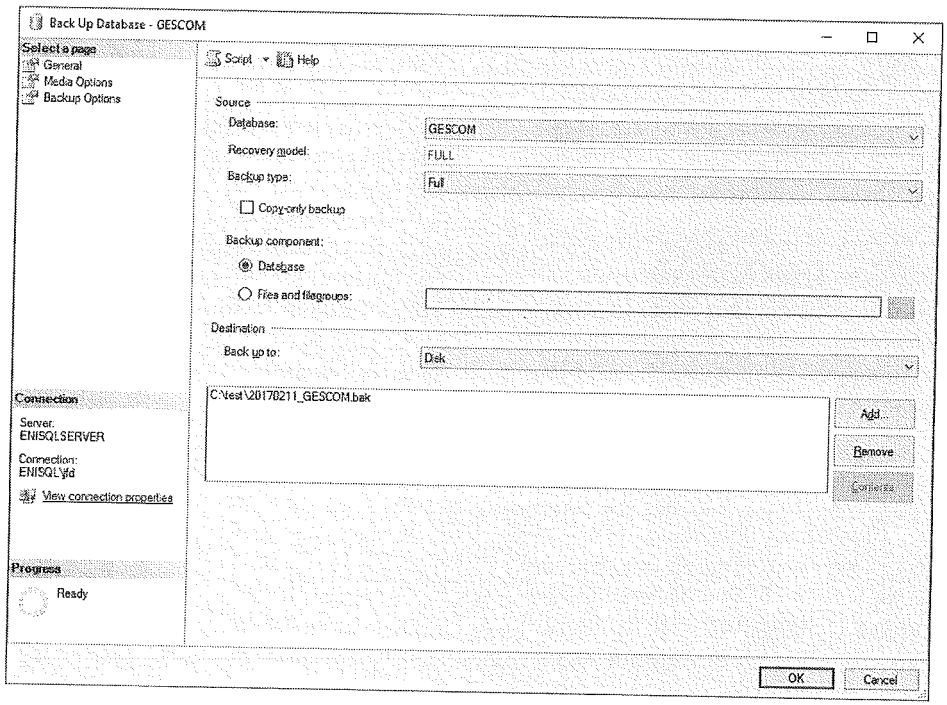
La distinción entre archivo y unidad de copia de seguridad proviene del hecho de que una unidad de copia de seguridad puede estar compuesta por varios archivos.

Una unidad de copia de seguridad puede contener varias copias de seguridad de una o de varias bases de datos.

Las unidades físicas

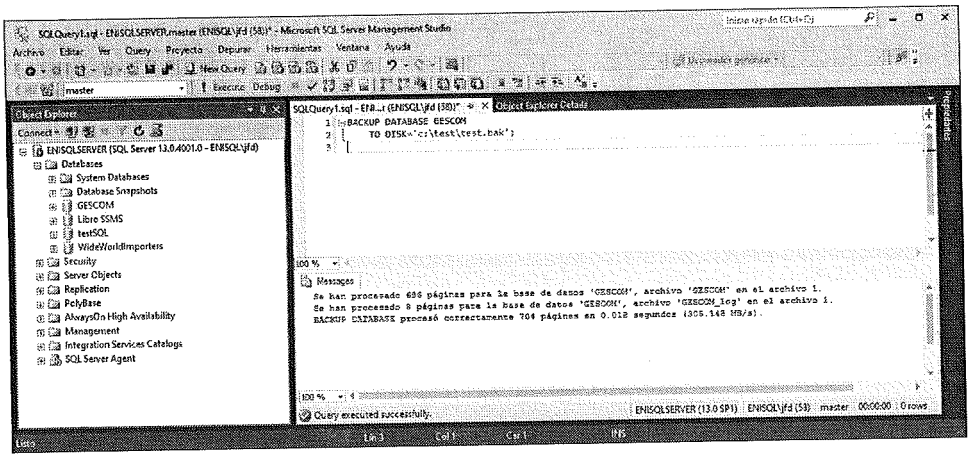
Una unidad de copia de seguridad física corresponde al nombre completo del archivo en Windows y, normalmente, a una utilización puntual de un soporte de copia de seguridad. Por ejemplo, antes de realizar una operación delicada sobre la base, puede realizarse una copia de seguridad completa de la base en una unidad de copia de seguridad física.

La unidad física se referencia directamente en las opciones de la instrucción BACKUP, o bien se indica el nombre de la unidad física en la ventana **Back Up Database** en SQL Server Management Studio.



Las diferentes opciones del juego de copias de seguridad, como la duración de su validez, la compresión y el cifrado, son accesibles desde la página **Backup Options** en la ventana anterior.

En Transact SQL, la unidad física se referencia directamente por la instrucción BACKUP.

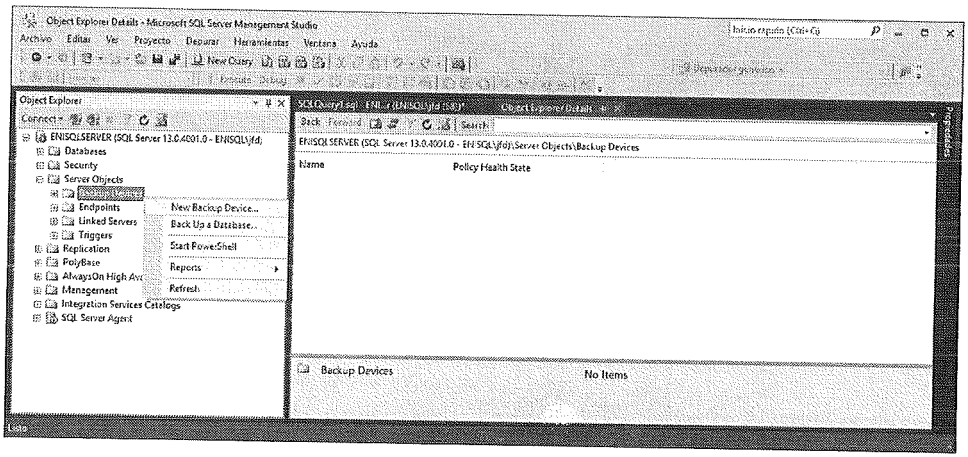


Si no se especifica la duración de la validez de la copia de seguridad, ya sea un número de días o una fecha de fin de validez, entonces SQL Server utiliza el valor definido en el servidor por el parámetro de configuración **media retention**. Esta propiedad es avanzada y por lo tanto solo es accesible después de la ejecución de la instrucción show advanced option.

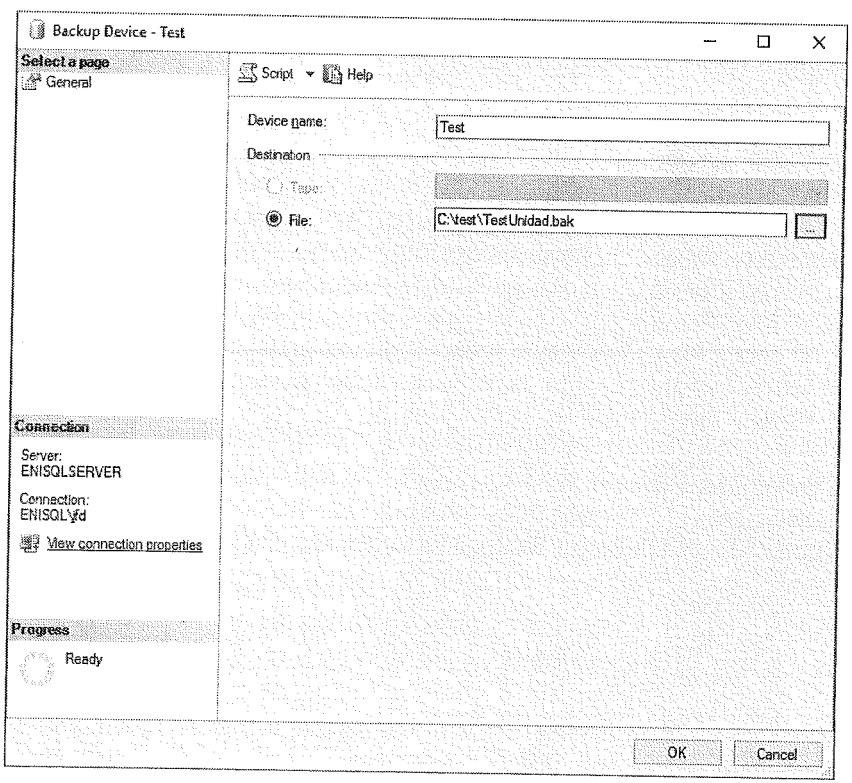
Las unidades lógicas

Detrás de una unidad de copia de seguridad lógica hay una unidad física. Las unidades de copia de seguridad lógica permiten referenciar de manera lógica los diferentes soportes de copia de seguridad que se pueden utilizar a nivel de la base de datos. Por ejemplo, en una política de copia de seguridad que contempla una copia de seguridad completa cada día y otra diferencial cada cuatro horas, es posible definir las unidades de copia de seguridad CompletaLunes, DiferencialLunes, CompletaMartes, DiferencialMartes... de manera que se utilicen unidades diferentes cada día de la semana. Las operaciones de copia de seguridad utilizan un nombre lógico, es decir, no es necesario conocer la ubicación física de los archivos.

Desde SQL Server Management Studio se puede crear una unidad lógica de copia de seguridad, seleccionando la opción **New Backup Device** en el menú contextual asociado al nodo **Server Objects - Backup Devices**, del explorador de objetos.



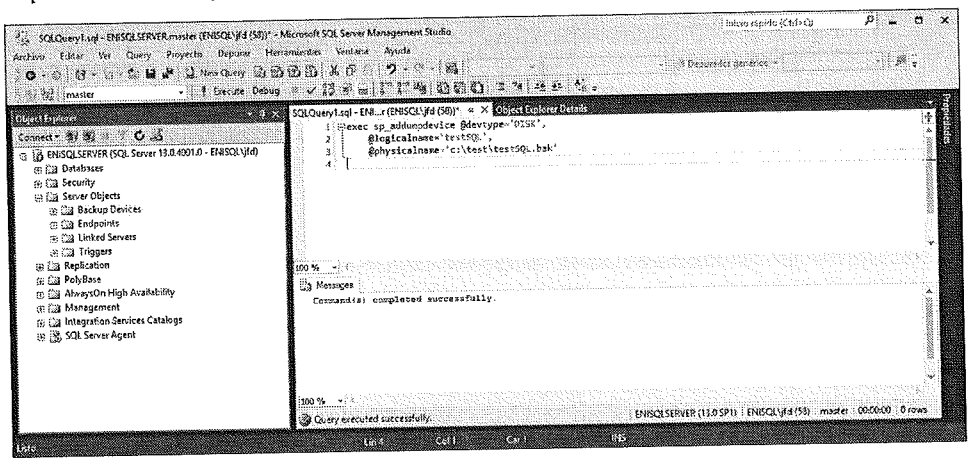
En el siguiente ejemplo se crea la unidad **Test**, basándose en un archivo físico.



La ventana de las propiedades de la unidad de copia de seguridad permite conocer el contenido de una unidad.

El mismo tipo de operación se puede realizar en Transact SQL.

El procedimiento que permite definir unidades lógicas nuevas es **sp_addumpdevice**.

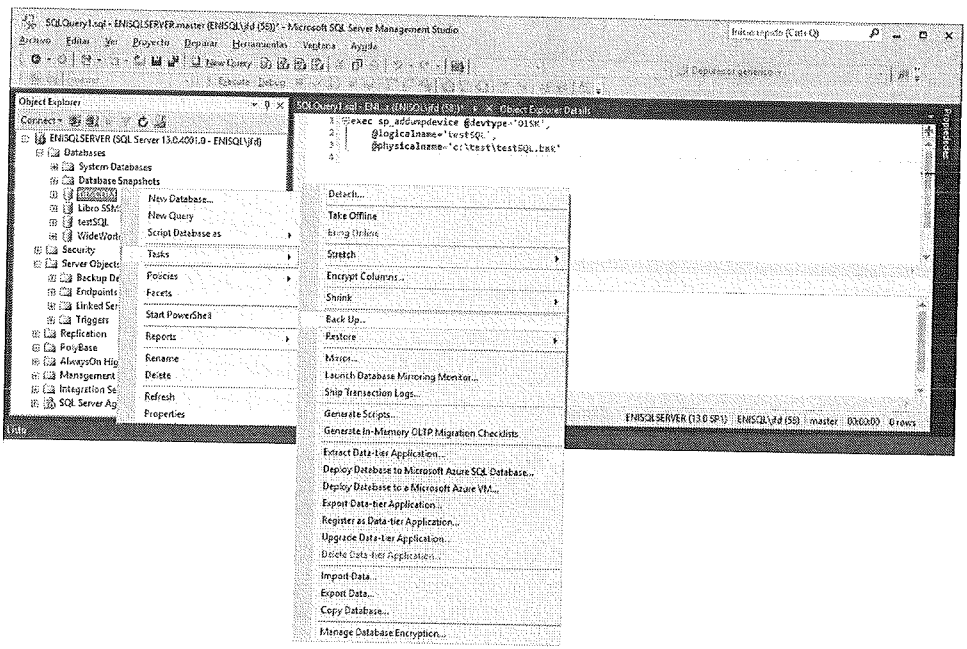


Las unidades lógicas permiten una gestión simplificada de las copias de seguridad a través de SQL Server Management Studio. También permiten reutilizar más eficientemente el espacio en disco de las antiguas copias de seguridad, facilitando la puesta en marcha de procedimientos automáticos sobre las operaciones de copia de seguridad.

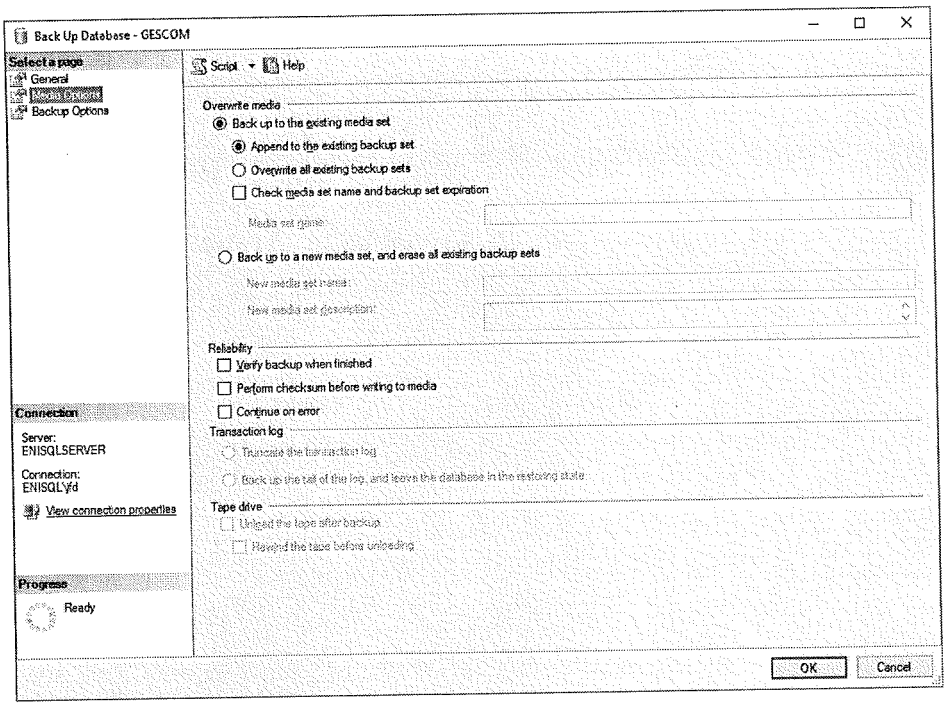
3.4 La instrucción BACKUP

La instrucción BACKUP permite hacer copias de seguridad tanto de la base de datos como del diario de transacciones. Las operaciones de copia de seguridad pueden realizarse en Transact SQL por medio de esta instrucción, pero también es posible pasar por la interfaz gráfica de SQL Server Management Studio.

La pantalla siguiente ilustra cómo solicitar la ejecución de una copia de seguridad desde SQL Server Management Studio.



La ventana de creación de una nueva copia de seguridad permite definir todas las opciones relativas a la ejecución de esta copia de seguridad.



Todas las opciones de la operación de copia de seguridad están disponibles mediante la página **Options**.

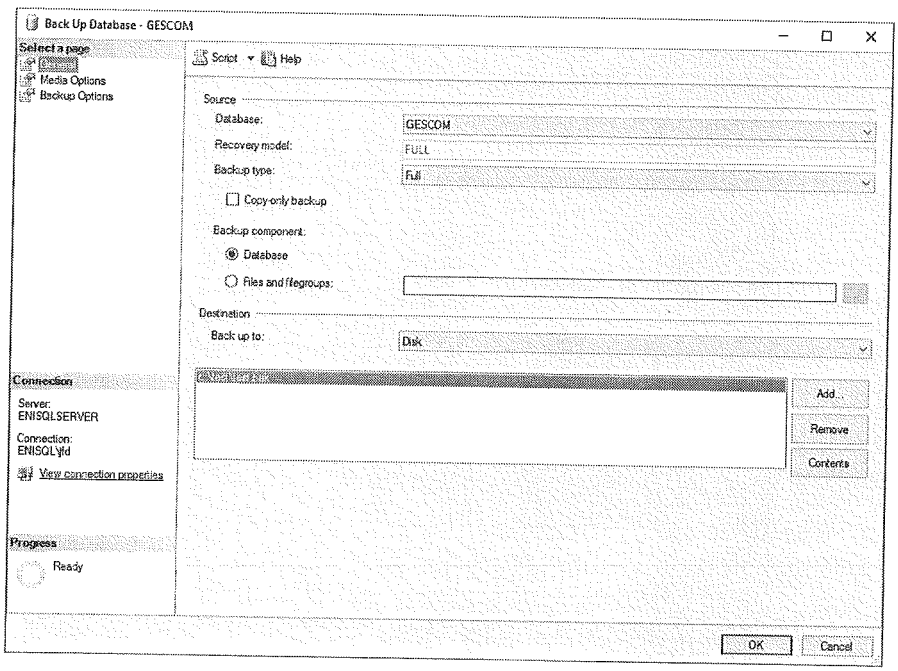
En el marco de reutilización de un archivo existente, es posible añadir la copia de seguridad a las ya contenidas en el archivo, o por el contrario borrar todos los datos presentes en el archivo.

En caso de que la instrucción BACKUP se reutilice directamente, la reutilización de un archivo conlleva tres opciones posibles:

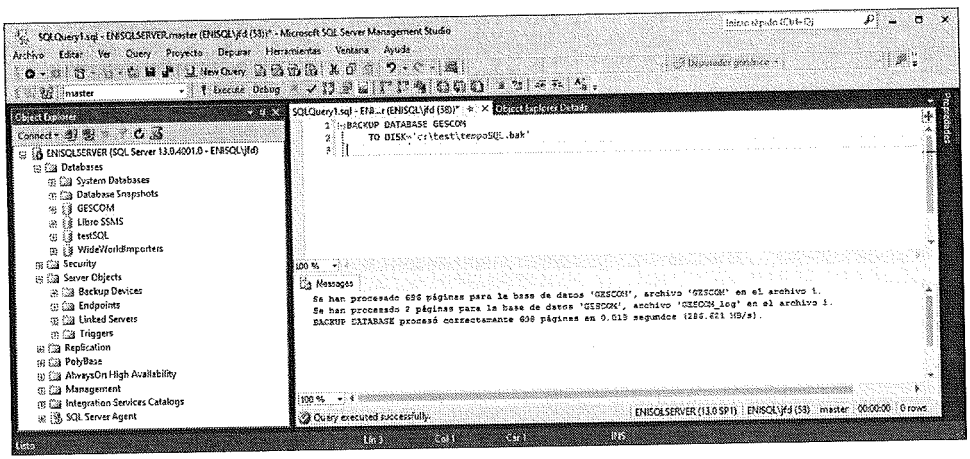
- INIT: para reemplazar el contenido de un archivo permanente, a condición de que la fecha de expiración de la copia de seguridad se haya superado (opción EXPIRATE) y que el archivo no sea miembro de un conjunto de copias de seguridad.
- NOINIT: para añadir la copia de seguridad a las ya presentes en el archivo.
- FORMAT: para poder reutilizar un archivo que ha participado en una copia de seguridad de varios archivos.

3.4.1 Copia de seguridad completa

Es el punto de partida de toda estrategia de copia de seguridad.



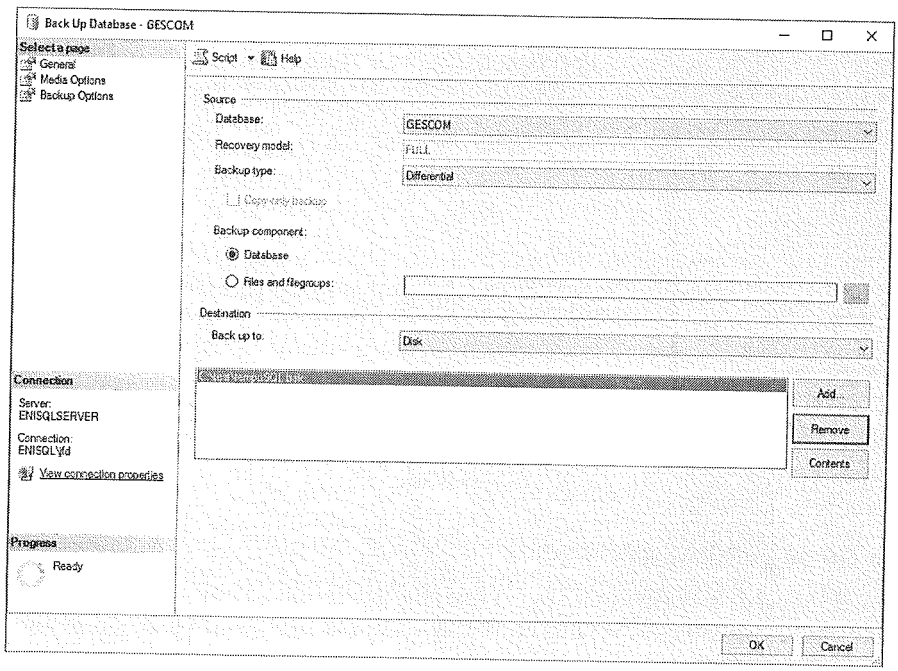
Solicitud de copia de seguridad completa



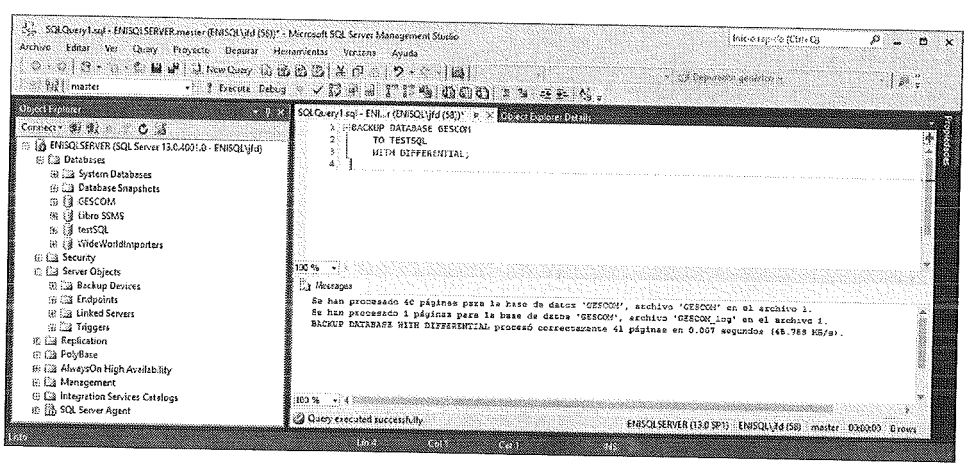
Copia de seguridad completa de GESCOM en Transact SQL

3.4.2 Copia de seguridad diferencial

Este tipo de copia de seguridad solo es posible si se ha efectuado una copia de seguridad completa anteriormente. Solo se guardan las páginas modificadas desde la última copia de seguridad completa. Para saber cuáles son estas páginas, SQL Server utiliza el número de secuencia del diario (LSN: *Log Sequence Number*) de la página y lo compara con el de la sincronización de la última copia de seguridad completa.



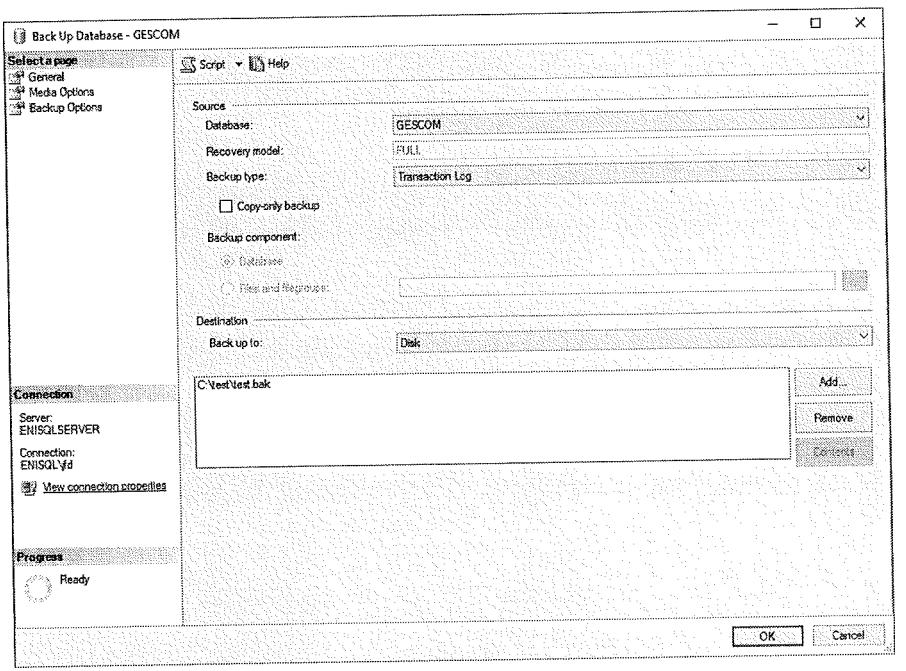
Copia de seguridad diferencial desde SQL Server Management Studio



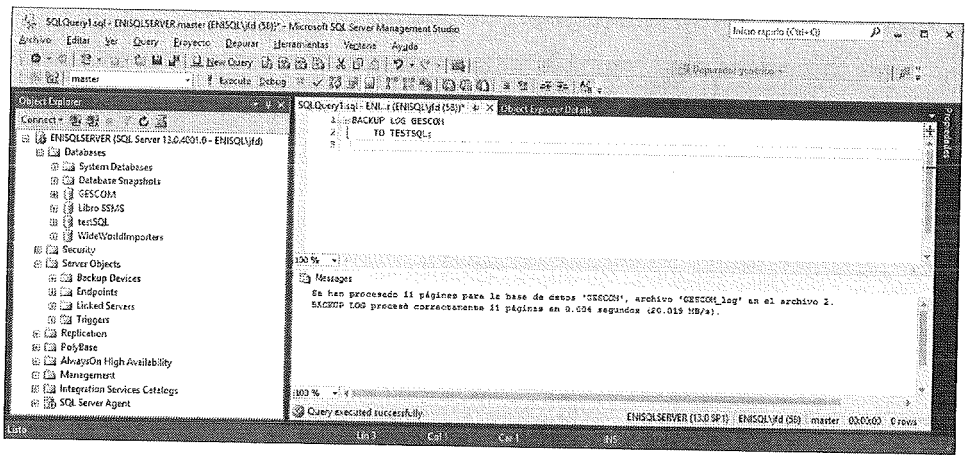
Copia de seguridad diferencial en Transact SQL

3.4.3 Copia de seguridad del registro de transacciones

Se puede hacer copia de seguridad del registro de transacciones. Se utiliza la instrucción BACKUP LOG. Una vez realizada la copia de seguridad, la parte inactiva del diario se trunca automáticamente. La opción NO_TRUNCATE permite no truncar el diario. Esta opción es particularmente útil en el marco de una operación de restauración.



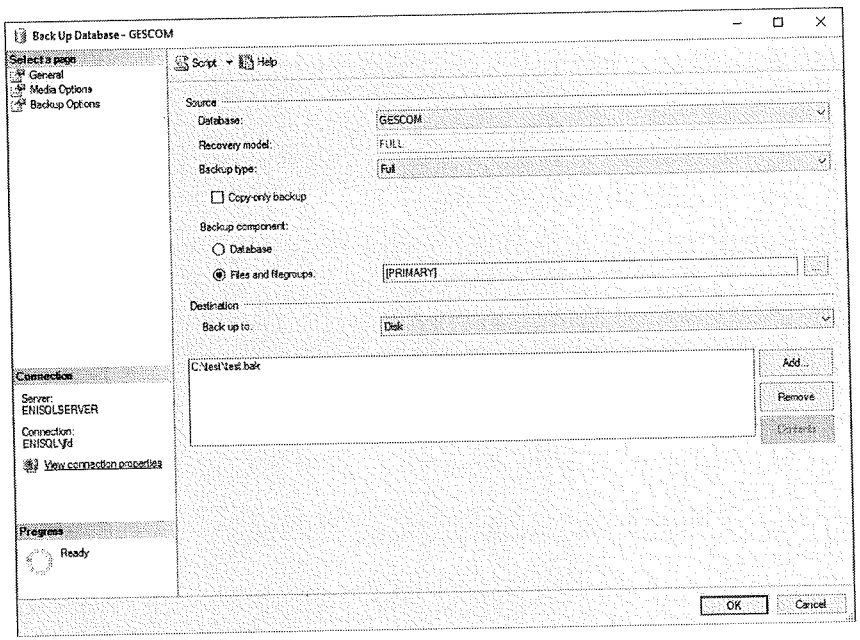
Copia de seguridad del registro de transacciones desde SQL Server Management Studio



Copia de seguridad del registro de transacciones en Transact SQL

3.4.4 Copia de seguridad de archivo o de grupo de archivos

Este tipo de copia de seguridad es especialmente interesante para las bases de datos muy voluminosas (VLDB: *Very Large DataBase*) cuando los tiempos de copia de seguridad son largos. Si los objetos se crean en grupos de archivos bien especializados, las copias de seguridad pueden ser óptimas en cuanto a tiempo. Las políticas de copia de seguridad posibles son las mismas que las de la base de datos.



Copia de seguridad de un archivo o de un grupo de archivos

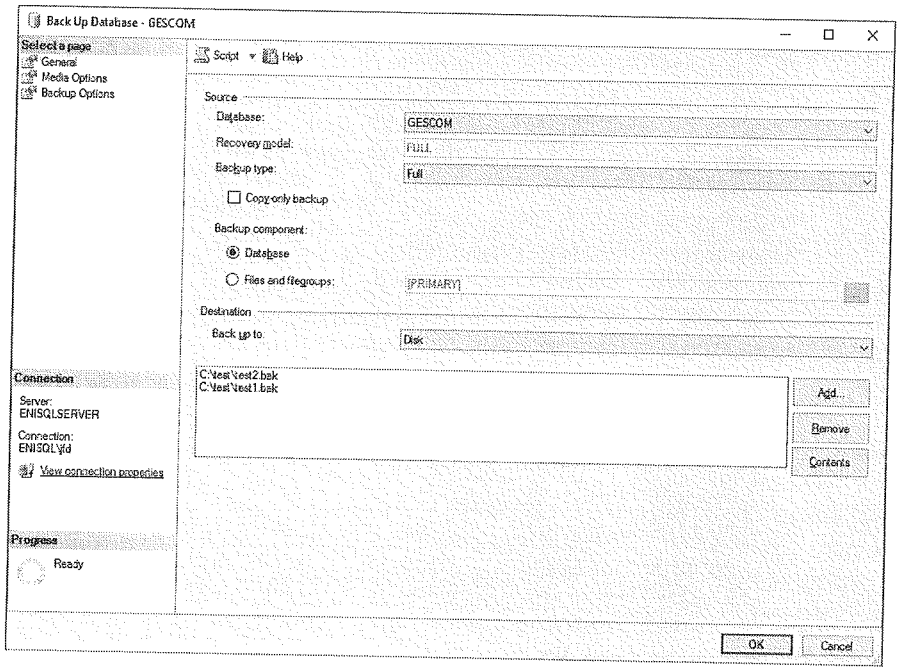
3.4.5 Copia de seguridad en varios archivos

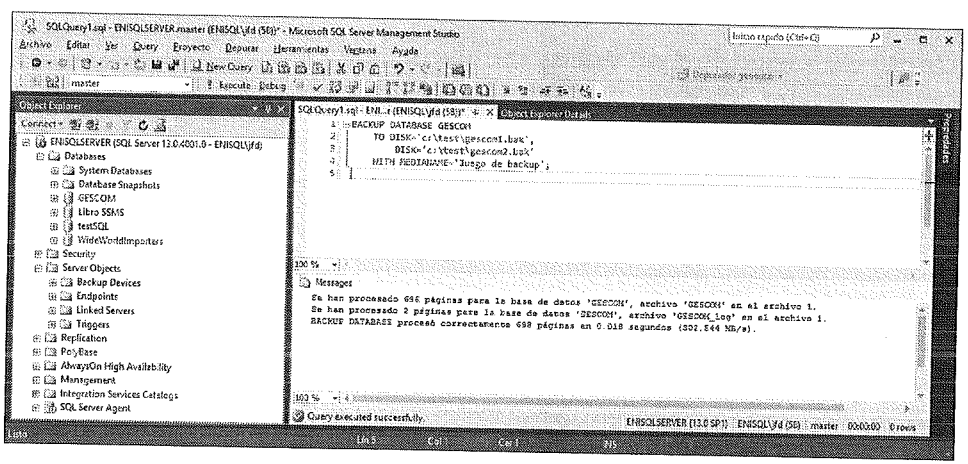
Siempre con el objetivo de acelerar las copias de seguridad, es posible realizarlas utilizando varios archivos al mismo tiempo. Las escrituras se realizan de manera paralela. Es el conjunto de los archivos lo que constituye la copia de seguridad. Aunque este método permite acelerar considerablemente los tiempos, las copias de seguridad son más frágiles, ya que la pérdida de un solo archivo hace inutilizable la totalidad de la copia de seguridad. Los archivos son indiferentemente temporales o permanentes.

Hay algunas limitaciones:

- Todos los archivos deben estar sobre el mismo tipo de soporte (banda, disco).
- Si un archivo es miembro de un conjunto de copias de seguridad, solo lo puede ser en el marco de esta copia de seguridad.

La instrucción BACKUP tiene la opción MEDIANAME, que permite dar un nombre a un conjunto de copias de seguridad. De esta manera, es más sencillo manipularlo. El nombre está limitado a 128 caracteres.



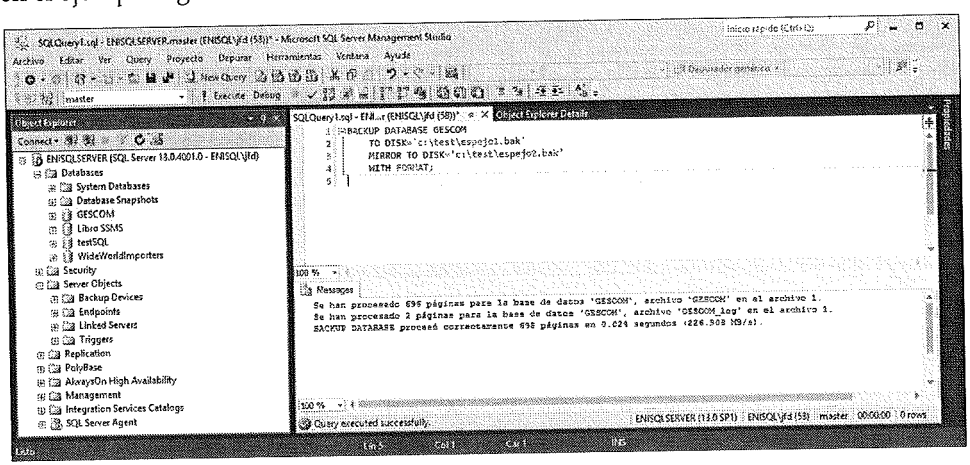


3.5 La replicación en espejo de las copias de seguridad

Tener una única versión del archivo de copia de seguridad puede ser peligroso. Es preferible disponer de varias. Así, si hay un archivo ilegible durante la restauración de la base de datos, es posible basarse en otro archivo con el mismo contenido. La replicación en espejo de las copias de seguridad permite establecer esta multiplicidad de los archivos.

La replicación en espejo de los archivos de copia de seguridad es una ganancia real de fiabilidad de los dispositivos de copia de seguridad. En efecto, una unidad de copia de seguridad permite repartir la copia de seguridad en varios archivos. Pero es el conjunto de los archivos lo que constituye la copia de seguridad. Un problema en un único archivo impide la restauración de la base.

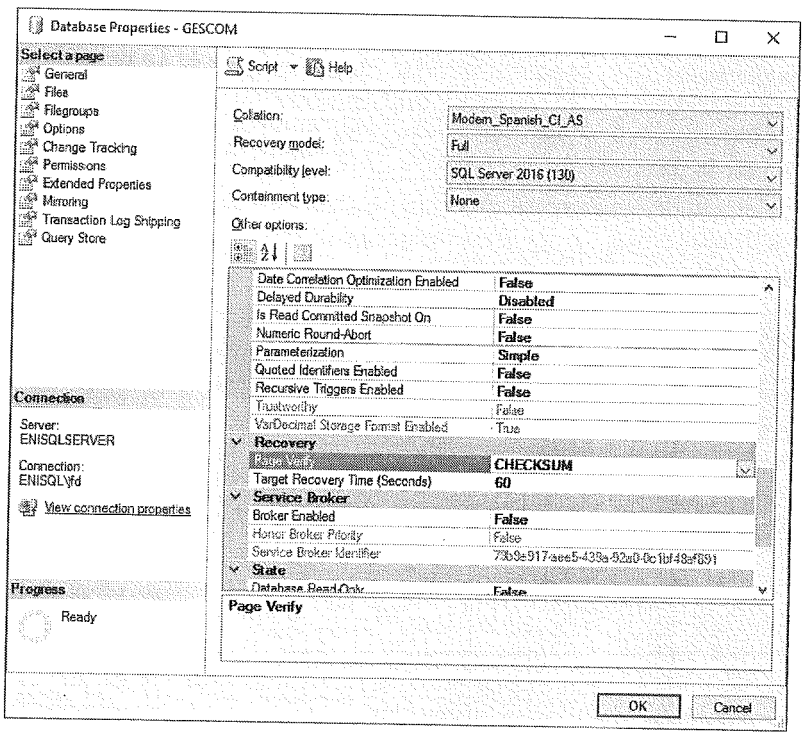
La replicación en espejo de los archivos de copia de seguridad se realiza mediante la opción MIRROR TO de la instrucción de copia de seguridad BACKUP, como se ilustra en el ejemplo siguiente:



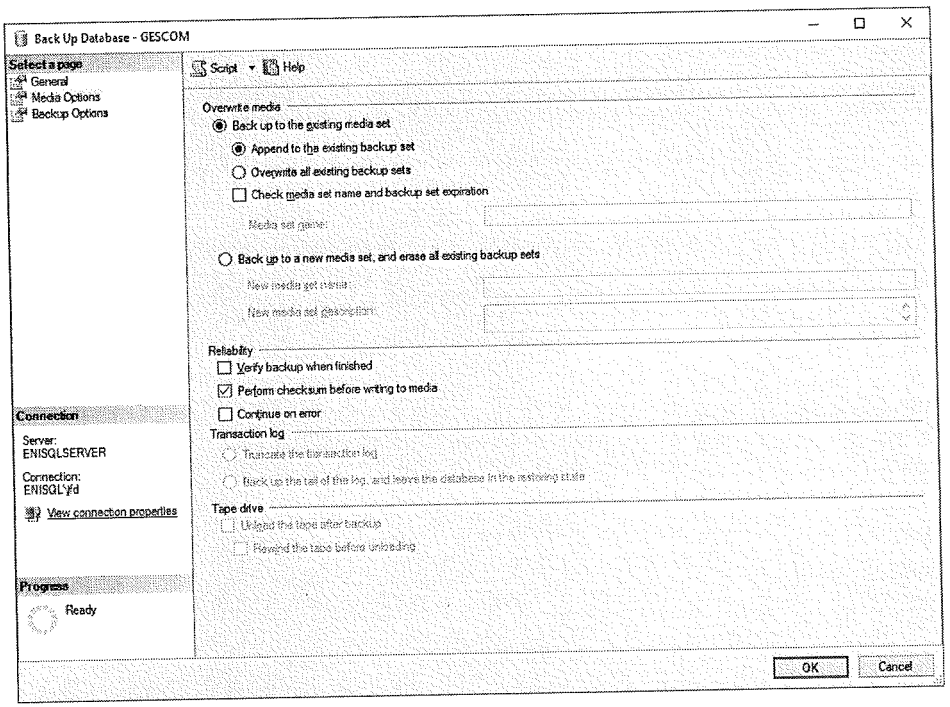
La replicación en espejo está disponible independientemente del soporte elegido para los archivos de copia de seguridad (disco o banda).

3.6 Verificar la integridad de una copia de seguridad

Si la opción TORN_PAGE_DETECTION o CHECKSUM está habilitada en la base de datos, las copias de seguridad realizadas a partir de esta base de datos conforman la validación de la integridad de las páginas de datos en la copia de seguridad. Esta opción se define en la base de datos.



Al definir una copia de seguridad, es posible solicitar la comprobación de la coherencia de esta copia de seguridad desde la página **Options**.

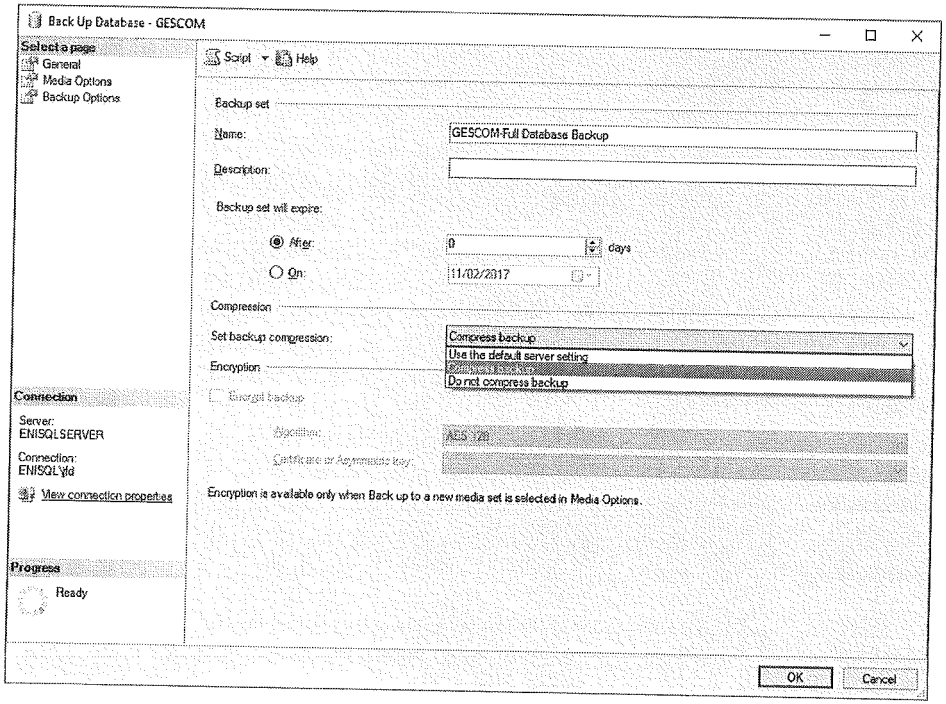


También es posible verificar la coherencia del conjunto de copias de seguridad mediante la instrucción RESTORE VERIFYONLY.

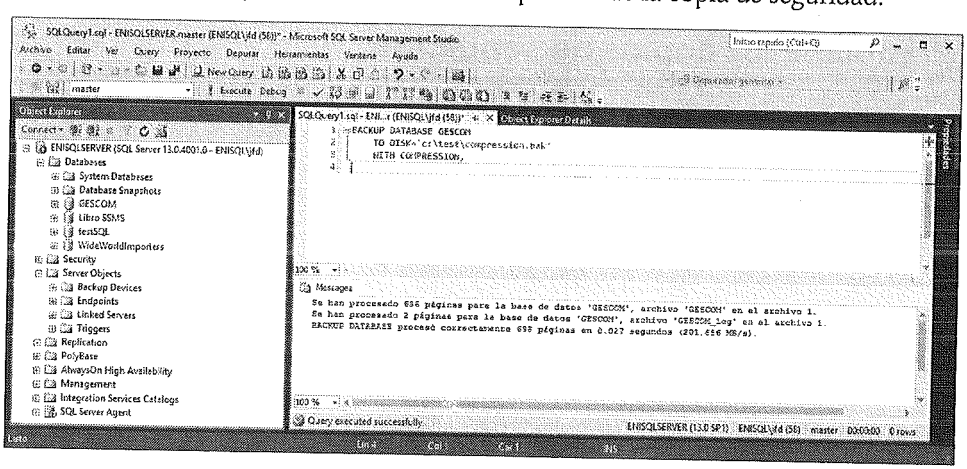
3.7 Comprimir las copias de seguridad

Con el objetivo de reducir el espacio que ocupan las copias de seguridad que ya están en disco o en banda, SQL Server ofrece la posibilidad de comprimir las copias de seguridad. El hecho de comprimir la copia de seguridad lleva implícita una carga de trabajo más importante a nivel del servidor. Afortunadamente, es posible definir esta opción de compresión para cada copia de seguridad. Además, el consumo del procesador se puede limitar desde el administrador de recursos.

En el momento de crear una copia de seguridad en modo gráfico, solo es necesario seleccionar un valor de la lista desplegable desde la página **Backup Options**.



En Transact SQL, es necesario utilizar los parámetros `WITH COMPRESSION` o `WITH NO_COMPRESSION` para activar o no la compresión de la copia de seguridad.



4. Ejercicio: copia de seguridad de la base de datos

4.1 Enunciado

Realice una copia de seguridad completa de la base de datos LibroSSMS en el archivo c:\copia\LibroSSMS.bak.

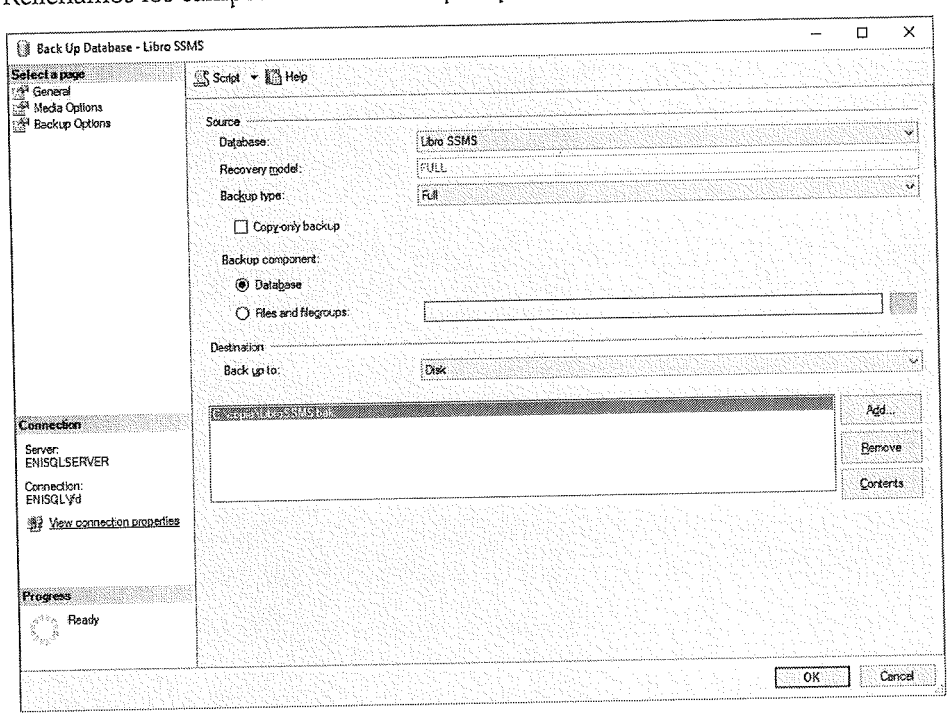
4.2 Solución

Antes de realizar esta copia, debemos asegurarnos de que existe la carpeta c:\copia y crearla si es necesario.

Con la ayuda de SQL Server Management Studio

Desde el explorador de objetos, nos situamos sobre la base de datos LibroSSMS y seleccionamos la opción **Tasks - Backup** del menú contextual.

Rellenamos los campos de la ventana que aparece de la siguiente manera:



Con la ayuda de un script TransactSQL

La instrucción BACKUP permite realizar la copia de seguridad de la base de datos:

```
USE master;
GO
BACKUP DATABASE LibroSSMS TO DISK='c:\copia\LibroSSMS.bak';
```

Capítulo 9
Restauración

1. Descripción general del proceso de restauración

La restauración representa la operación inversa de la copia de seguridad. El proceso de restauración no puede utilizarse para realizar migraciones de base de datos.

Esta operación puede realizarse con ayuda de los comandos Transact SQL (RESTORE) o mediante la consola de administración SQL Server Management Studio. Sea cual sea la base que se va a restaurar, es indispensable instalar SQL Server para cargar los datos en la máquina.

El simple hecho de reemplazar los archivos que contienen los datos y el diario no constituye en sí mismo una restauración, ya que SQL Server no es capaz de acceder a estos archivos mientras no estén referenciados en la base Master.

El proceso de restauración puede intervenir en dos casos:

- Tras una solicitud explícita por parte de un usuario.
- Tras un reinicio del servidor provocado por una parada repentina. Hablaremos entonces de restauración automática.

1.1 La restauración automática

Este proceso interviene en cada inicio del servidor. Se asegura de que la última operación inscrita en el diario sea un punto de sincronización. Si no es el caso, entonces el diario se lee de nuevo desde el último punto de sincronización y todas las transacciones validadas se recrean, mientras que el resto de las modificaciones se anulan. Esta operación es necesaria para garantizar la coherencia de los datos.

1.2 Operaciones ejecutadas automáticamente por SQL Server

SQL Server realiza automáticamente un cierto número de operaciones con el objetivo de acelerar el proceso de restauración y reducir al máximo el tiempo que el servidor no está disponible.

El control de seguridad

El interés principal de este control de seguridad es protegerse de las restauraciones accidentales, que pueden borrar una base actualizada para volver a una versión anterior de la base de datos. Igualmente, el control de seguridad va a asegurarse de que posee todos los archivos participantes en un conjunto de copia de seguridad.

El control de seguridad prohíbe también la restauración de la base si el conjunto de archivos que forman la base de datos es diferente al registrado por el conjunto de la copia de seguridad.

Reconstrucción de la base de datos y de los archivos asociados

En el marco de una restauración a partir de una copia de seguridad completa de la base, SQL Server se encarga de recrear la base y los archivos que la componen. Se crean los objetos y se transfieren los datos. El esquema de la base de datos se reconstruye automáticamente durante el procedimiento de restauración y no requiere de operaciones manuales.

1.3 Operaciones preliminares

Antes de realizar una restauración, es importante llevar a cabo algunas operaciones preliminares con el objetivo de asegurarse del correcto funcionamiento del proceso.

1.3.1 La verificación de las copias de seguridad

Esta operación se realizará normalmente al final de cada copia de seguridad. Aquí, el objetivo es encontrar la copia o las copias de seguridad que van a ser necesarias para restaurar la base reduciendo al máximo los tiempo de restauración y el volumen de datos perdidos.

Existen cuatro instrucciones, que se detallan a continuación, aunque también es posible utilizar SQL Server Management Studio.

RESTORE HEADERONLY

Permite conocer la información contenida en el encabezado de un archivo o de un juego de copia de seguridad:

- El nombre y la descripción del archivo o del conjunto de copia de seguridad.
- El soporte utilizado (disco o banda).
- La fecha y hora de la copia de seguridad.
- El método de copia de seguridad utilizado (completa, diferencial, del diario, por grupo de archivos o completo).
- El tamaño de la copia de seguridad.
- El número secuencial de la copia de seguridad en una cadena de archivos de copia de seguridad.

RESTORE FILELISTONLY

Esta instrucción permite obtener información sobre los archivos de datos y de diario que constituyen la base de datos que utiliza este archivo de copia de seguridad. La información devuelta es:

- El nombre lógico y físico de los archivos de datos y de los archivos de diario.
- El tipo de cada archivo (datos o diario).
- El grupo de archivos al que pertenece el archivo.
- El tamaño máximo de cada archivo en megabytes.
- El tamaño del conjunto de copia de seguridad en megabytes.

RESTORE LABELONLY

Permite obtener cierta información sobre el archivo de copia de seguridad.

RESTORE VERIFYONLY

Esta instrucción, que no verifica la coherencia de las copias de seguridad, solo se utiliza en el marco de un juego de copia de seguridad, para asegurarse de que todos los archivos que constituyen este juego están presentes.

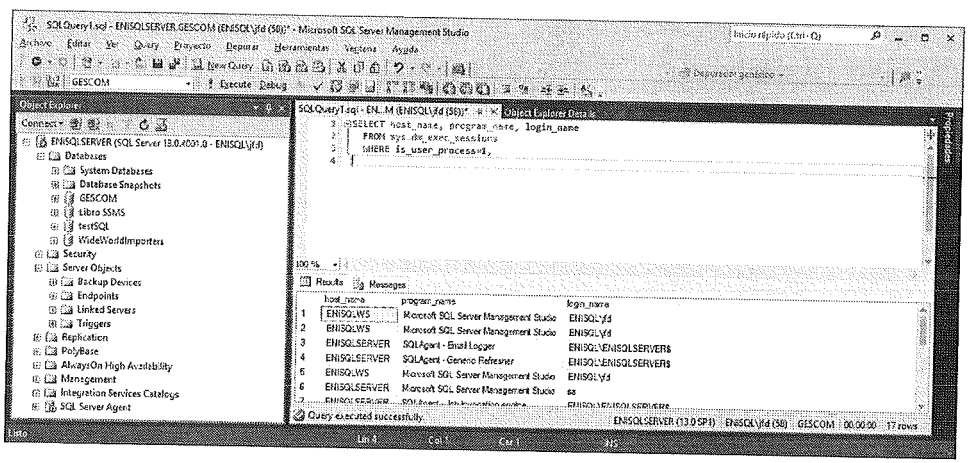
Estas cuatro instrucciones permiten visualizar el contenido de un juego de copia de seguridad y proporcionan, por lo tanto, mucha información sobre la estructura de la base de datos que hay que restaurar. Únicamente los usuarios que disponen del privilegio CREATE DATABASE pueden ejecutar estas instrucciones.

1.3.2 Las tareas específicas

Antes de lanzar una operación de restauración sobre una base de datos, es necesario asegurarse de que nadie trabaja en la base de datos y, posteriormente, hacer la copia de seguridad de la parte del diario de transacciones para la que no existe copia de seguridad con el objetivo de minimizar la pérdida de datos.

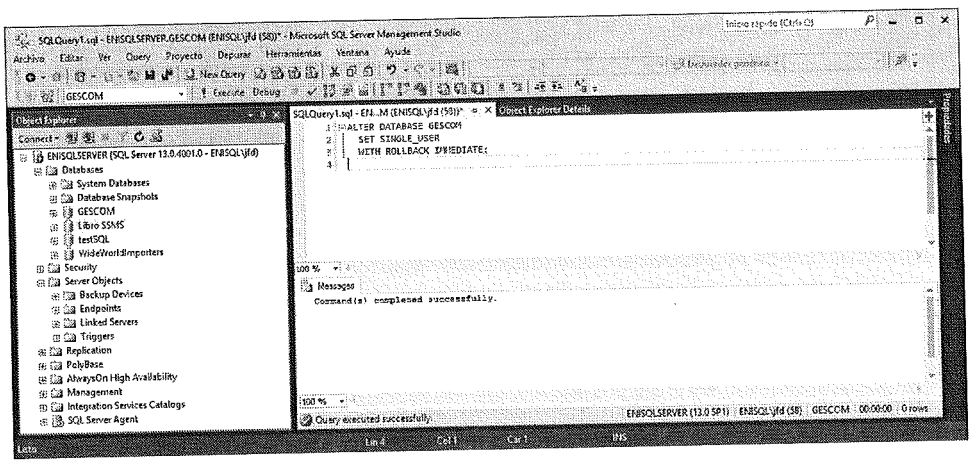
Asegurarse de que ningún usuario trabaje en la base

Consultando la vista de sistema **sys.dm_exec_sessions**, es posible establecer la lista de los usuarios actualmente conectados a la base. En el ejemplo siguiente se muestra cómo averiguar la conexión, el nombre del programa y el puesto utilizado para establecer conexiones de usuario (is_user_proces=1).



Después de asegurarnos de que no hay usuarios conectados a la base de datos, es necesario garantizar que no puedan establecerse nuevas conexiones, poniendo la base de datos en modo monousuario.

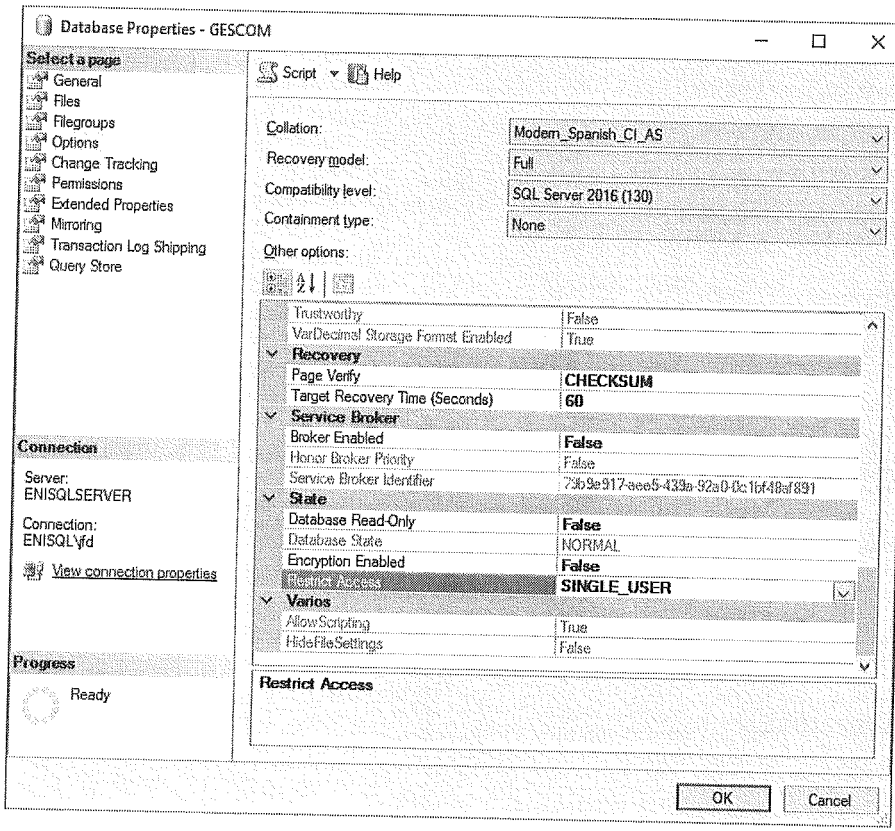
Es un administrador de la base de datos el que puede restringir el acceso, bien a un único usuario o bien solo a los usuarios que tienen el privilegio de abrir una sesión cuando la base se encuentra en modo restringido.



Cambio del modo de acceso en Transact SQL

Observación

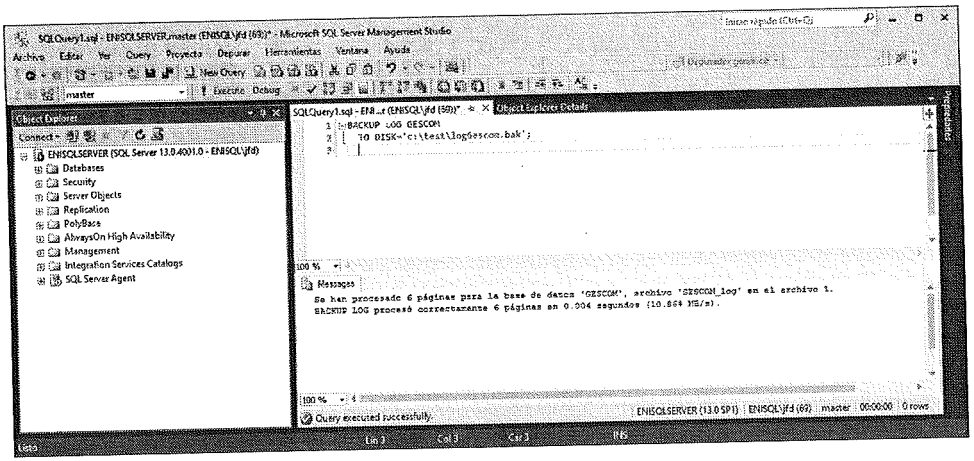
La instrucción ALTER DATABASE GESCOM SET MULTI_USER permite volver al modo de acceso normal.



Cambio del modo de acceso desde SQL Server Management Studio

Copia de seguridad del diario de transacciones

Cuando esta operación es posible, se realiza mediante una instrucción BACKUP LOG.



Copia de seguridad del diario de transacciones en curso

2. Restauración de las copias de seguridad

Según la copia de seguridad efectuada, el método de restauración será ligeramente diferente.

2.1 La instrucción RESTORE

En modo Transact SQL, la instrucción RESTORE permite recuperar una copia de seguridad hecha con SQL Server.

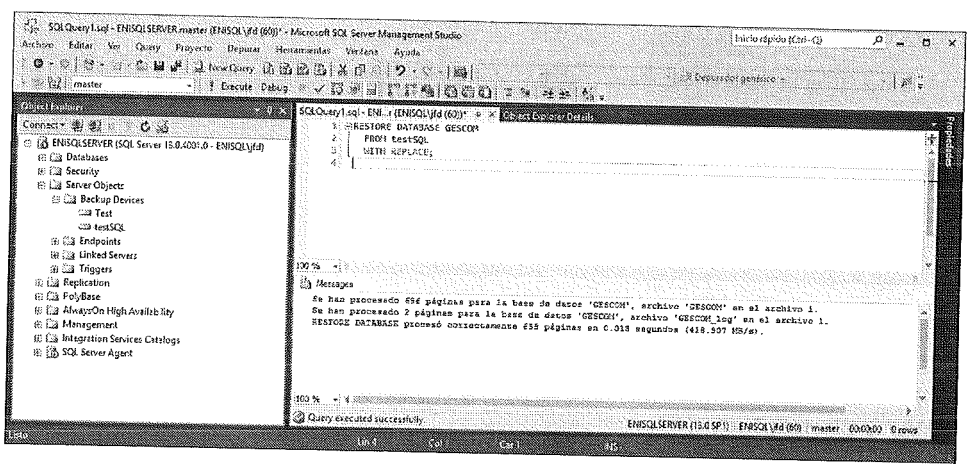
Sintaxis

```
RESTORE DATABASE {nombre_base |@var_nombre_base }
[FROM dispositivo_de_copia de seguridad [,...n ] ]
[WITH
[{CHECKSUM | NO_CHECKSUM }]
[[,] { CONTINUE_AFTER_ERROR | STOP ON ERROR } ]
[[,] FILE = número_archivo]
[[,] KEEP_REPLICATION]
[[,] MEDIANAME = nombre_soporte]
[[,] MEDIAPASSWORD = contraseña_media]
[[,] MOVE 'nombre_lógico' TO 'nombre_físico'] [,...n]
[[,] PASSWORD = contraseña]
[[,] PARCIAL]
[[,] {RECOVERY|NORECOVERY|STANDBY = nombre_archivo_anulación}]
[[,] REPLACE]
```

```
[[,] RESTRICTED_USER]
[[,] {REWIND|NOREWIND}]
[[,] STATS [ = porcentaje]]
[[,] {STOPAT = fecha_hora |
     STOPATMARK = {'marca'|'lsn:número_lsn' }
     [ AFTER fecha_hora] |
     STOPBEFOREMARK = { 'marca' | 'lsn:número_lsn' }
     [ AFTER fecha_hora]}]
[[,] {UNLOAD|NOUNLOAD}]
]
[;]
```

Ejemplo

En el siguiente ejemplo, se hace una restauración completa de la base GESCOM, a partir de las copias de seguridad presentes en la unidad de copia de seguridad testSQL. La opción `WITH REPLACE` se indica para confirmar el borrado de la base existente.



Observación

El comando RESTORE se debe lanzar siempre desde la base ***Master****.*

Para restaurar el diario de transacciones, es necesario utilizar la instrucción RESTORE LOG, que tiene una configuración similar a RESTORE DATABASE.

En una base de datos dañada, la restauración permite encontrar un conjunto coherente de datos. La etapa de restauración se encarga de recrear automáticamente los archivos y los objetos de la base de datos, sin que para esto sea necesario eliminar la base de datos con anterioridad.

2.2 Las opciones de la instrucción RESTORE

Existen numerosas opciones sobre la instrucción RESTORE, algunas de las cuales se detallan aquí.

Inicialmente, cuando la restauración utiliza varias copias de seguridad (una completa seguida de una diferencial y por último la de los diarios de transacciones), es importante que SQL Server no permita el acceso a la base de datos a los usuarios mientras no se haya efectuado la última restauración. Para esto, la instrucción RESTORE ofrece las opciones RECOVERY y NORECOVERY.

RECOVERY

Es la opción que SQL Server utiliza por defecto. Con esta opción, al final de la restauración, SQL Server revisa el diario de transacciones para anular todas las transacciones no validadas desde el último punto de sincronización y confirmar todas las validadas. Una vez que estas operaciones terminan, la base es accesible por los usuarios.

NORECOVERY

Esta opción debe especificarse para todas las etapas de la restauración excepto la última. SQL Server no toca el diario de transacciones y la base de datos no puede ser utilizada.

FILE

Esta opción se utiliza únicamente cuando el archivo de copia de seguridad contiene varias copias de seguridad. Permite precisar el número de la copia de seguridad que se desea restaurar.

MOVE... TO

Por defecto, los archivos de la base de datos se restauran en el mismo lugar que fue definido durante la copia de seguridad. Si deseamos especificar una ruta diferente, es necesario utilizar esta opción.

REPLACE

Esta opción permite restaurar una base eliminando la base existente inicialmente sobre el servidor. Esta opción está desactivada por defecto.

STOPAT

Esta opción, disponible únicamente si la base se configura en modo de restauración completa, permite recrear las transacciones registradas en el diario hasta una fecha y hora especificadas en formato varchar, char, smalldatetime o datetime.

STOPATMARK, STOPBEFOREMARK

Estas opciones están disponibles únicamente si la base se configura en modo de restauración completa. Permiten restaurar las transacciones hasta una transacción marcada o bien un número de registro (*Log Sequence Number*).

2.3 La restauración de los diferentes tipos de copia de seguridad

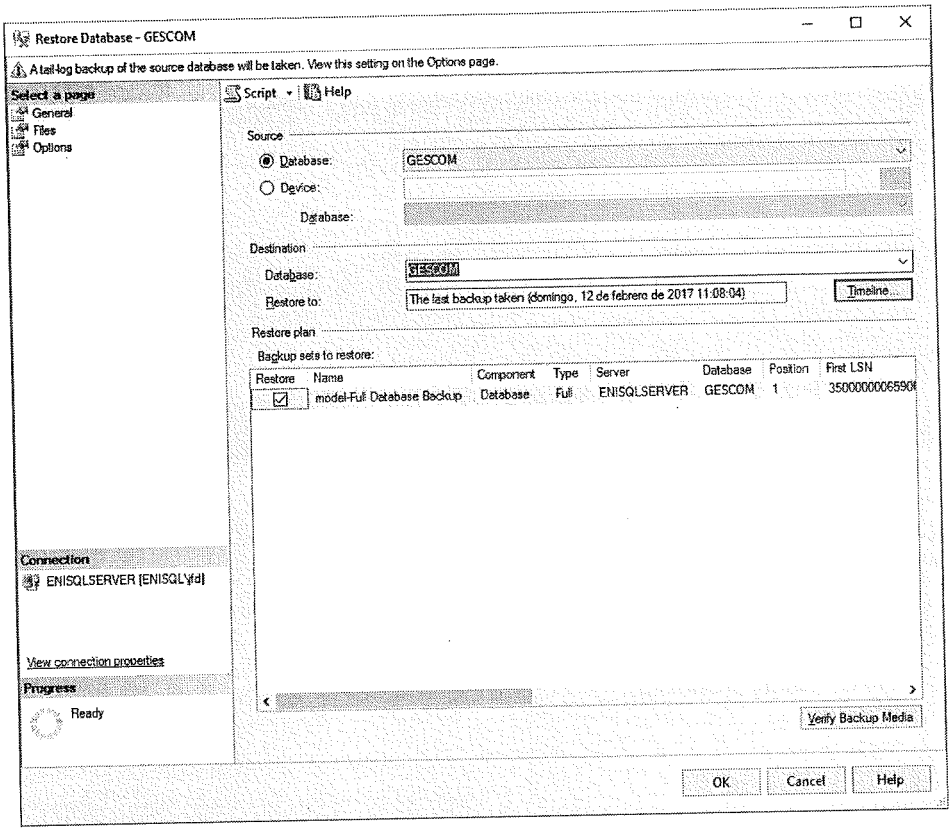
Todas las políticas de restauración comienzan necesariamente por la recuperación de una copia de seguridad completa de la base. Una vez establecido este punto de partida, es posible acumular diferentes restauraciones.

2.3.1 A partir de una copia de seguridad completa

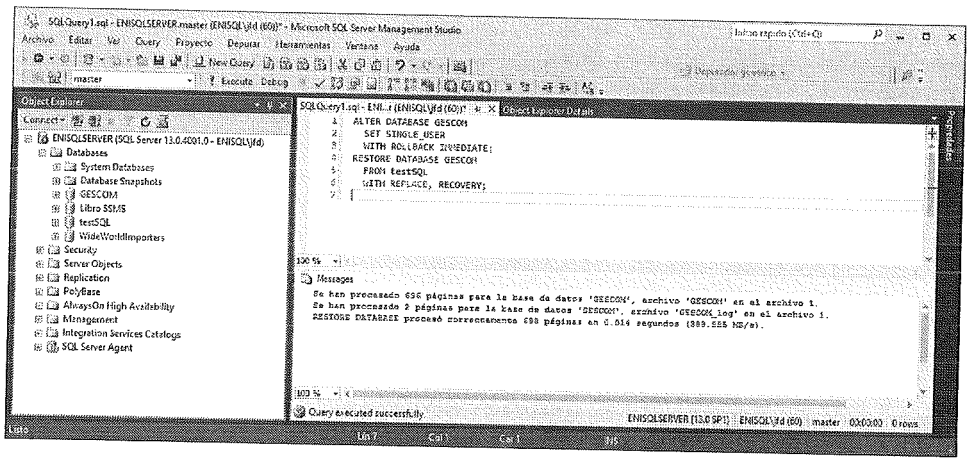
La restauración a partir de una base completa es una operación simple y rápida. Si es posible realizar una copia de seguridad completa todos los días, es necesario hacerlo, ya que este método permite obtener los tiempos de no disponibilidad del servidor más cortos.

SQL Server reemplaza todos los archivos de datos y los archivos asociados a sus ubicaciones de origen. De la misma manera, el conjunto del esquema de la base se recupera automáticamente.

Esta operación de restauración se realiza cuando el disco físico que contiene los archivos de la base de datos está dañado, o bien cuando una parte de los archivos está corrupta o se ha perdido. La restauración de una copia de seguridad completa también es posible cuando se desea restaurar los datos sobre un segundo servidor (este servidor puede convertirse en un servidor de seguridad, por ejemplo).



Restauración desde SQL Server Management Studio



Restauración sin poner en línea la base de datos

Las opciones de recuperación RECOVERY o NORECOVERY deberán estar presentes, sobre todo si después habrá otras restauraciones.

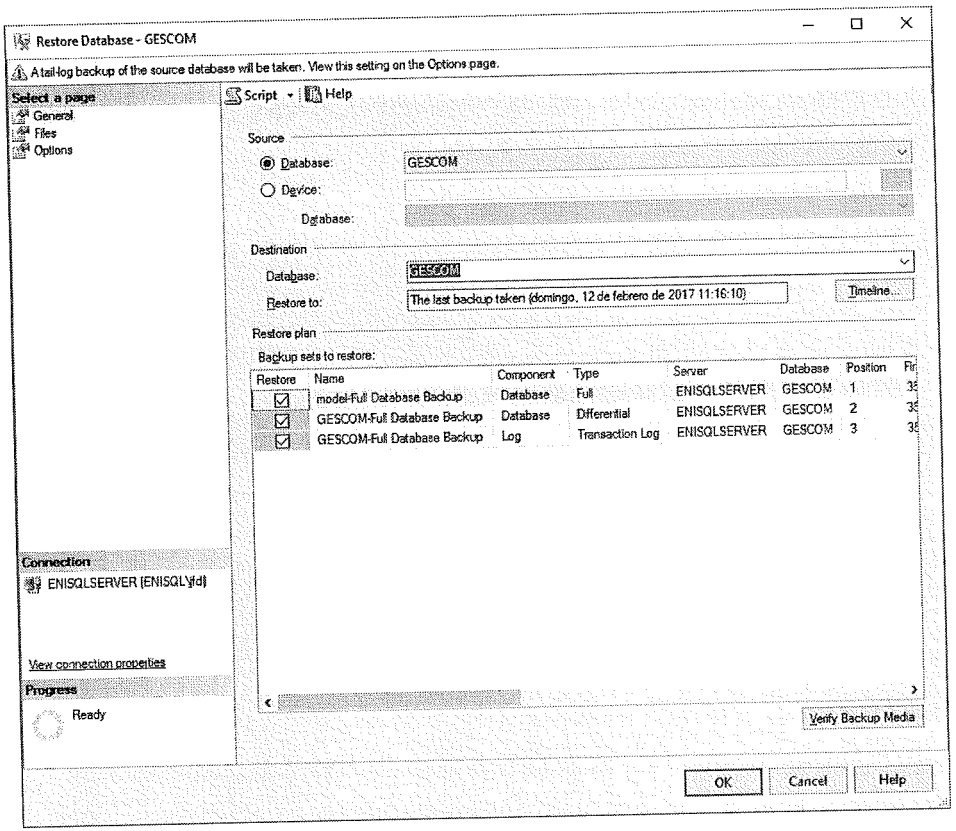
2.3.2 A partir de una copia de seguridad diferencial

Este tipo de restauración solo puede intervenir después de una copia de seguridad completa.

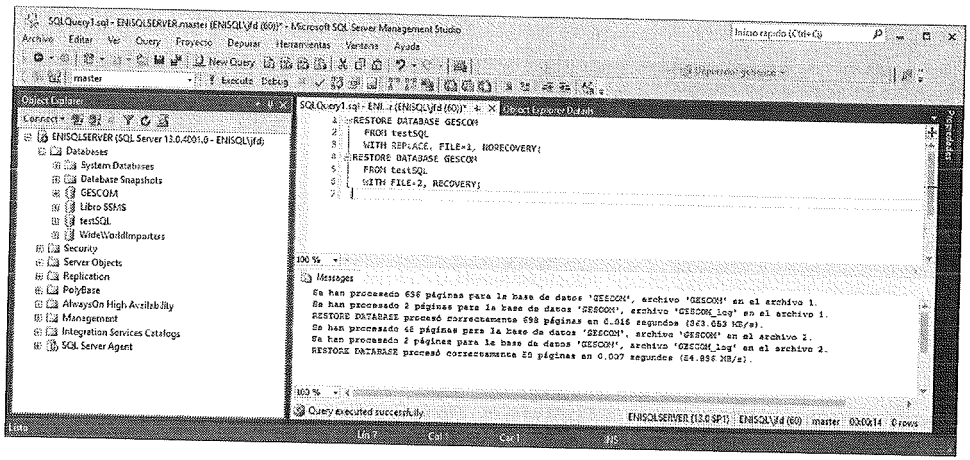
Si existen varias copias de seguridad diferenciales para la misma base de datos, es suficiente con restaurar la más reciente porque una copia de seguridad diferencial contiene todas las modificaciones realizadas sobre los datos desde la última copia de seguridad completa.

La restauración sitúa la base de datos en el estado en que estaba cuando se realizó la copia de seguridad.

La restauración de este tipo de copia de seguridad es mucho más rápida que la restauración de todos los registros de transacciones.



Restauración de una copia de seguridad diferencial



Restauración diferencial a partir de un dispositivo de copia de seguridad. Aquí se utiliza la primera copia de seguridad realizada en este dispositivo.

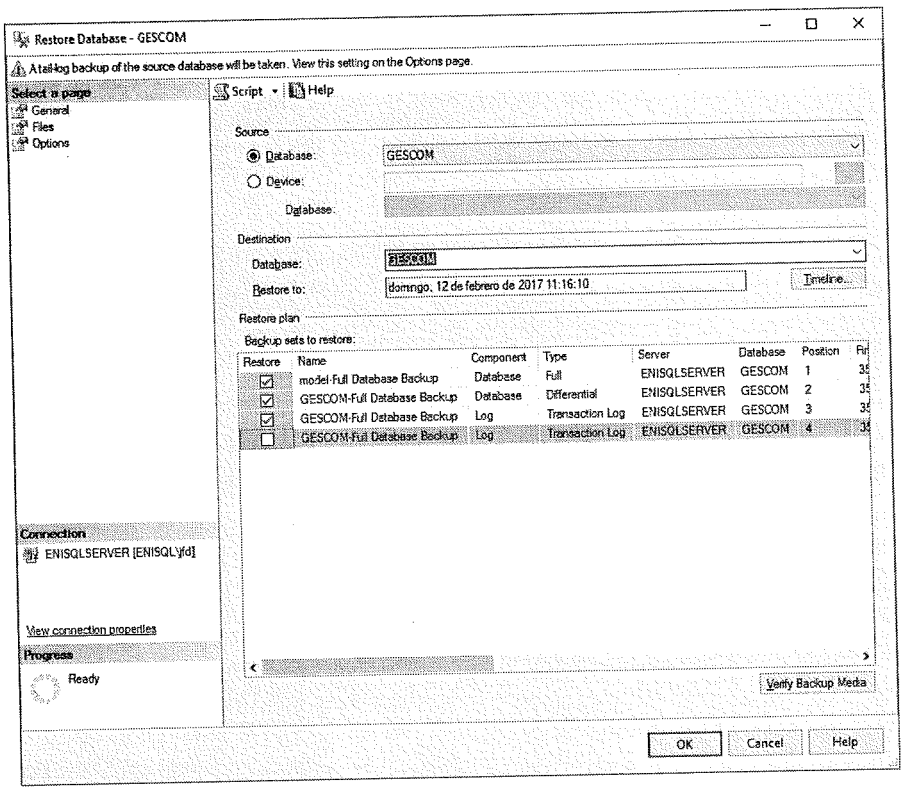
Una vez más, si la copia de seguridad diferencial es la última de la que se dispone para la base de datos, es necesario ejecutar la sentencia RESTORE con RECOVERY. En el resto de los casos (se dispone de copias de seguridad del diario de transacciones), es preciso utilizar la opción NORECOVERY para poder restaurar la base lo mejor posible.

2.3.3 A partir de una copia de seguridad del diario de transacciones

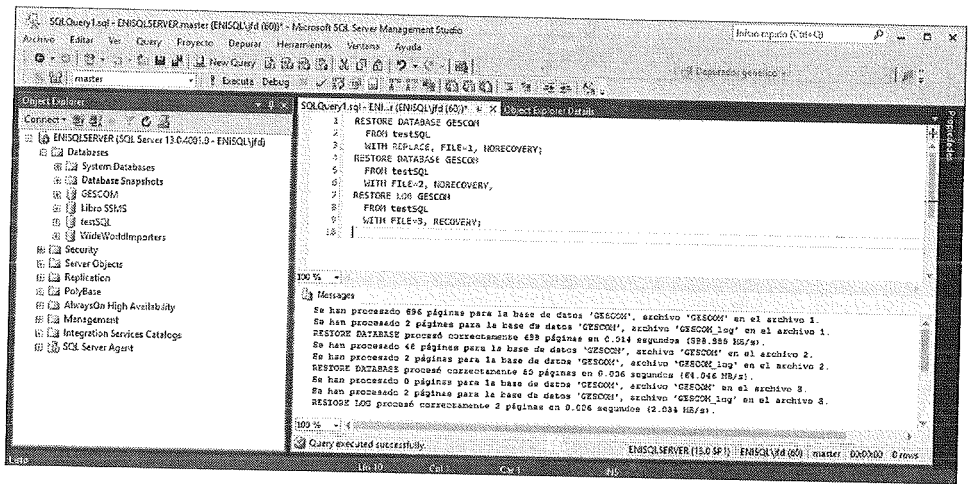
El diario de transacciones es el último elemento que se ha de restaurar. Va a permitir limitar la pérdida de información modificada en la base de datos desde la última copia de seguridad completa o diferencial.

Eventualmente, puede que tenga varios archivos de diarios para restaurar. En este caso se tienen que restaurar en orden cronológico.

Todas las sentencias de restauración de los diarios deben incluir la opción NORECOVERY, excepto el último diario, donde la opción se omitirá o será RECOVERY.



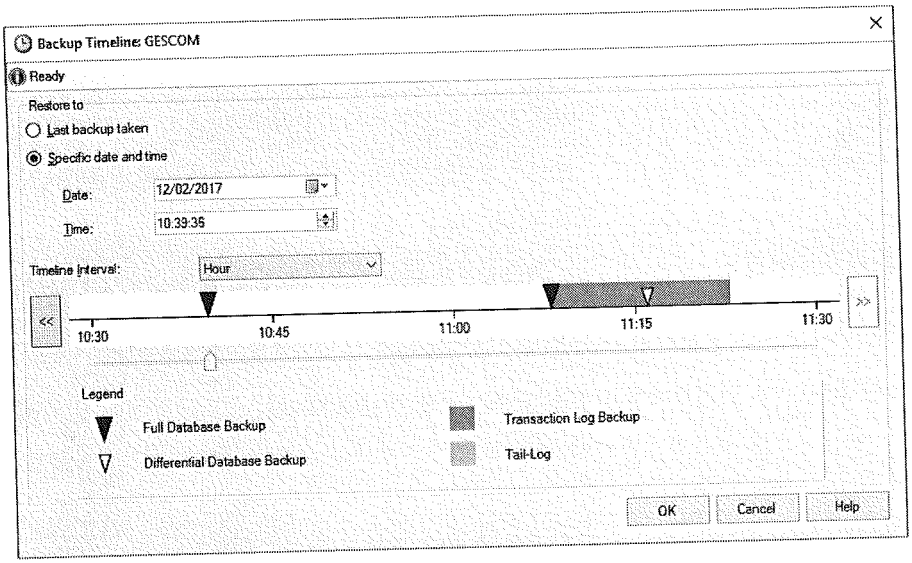
Solicitud de restauración del diario desde SQL Server Management Studio

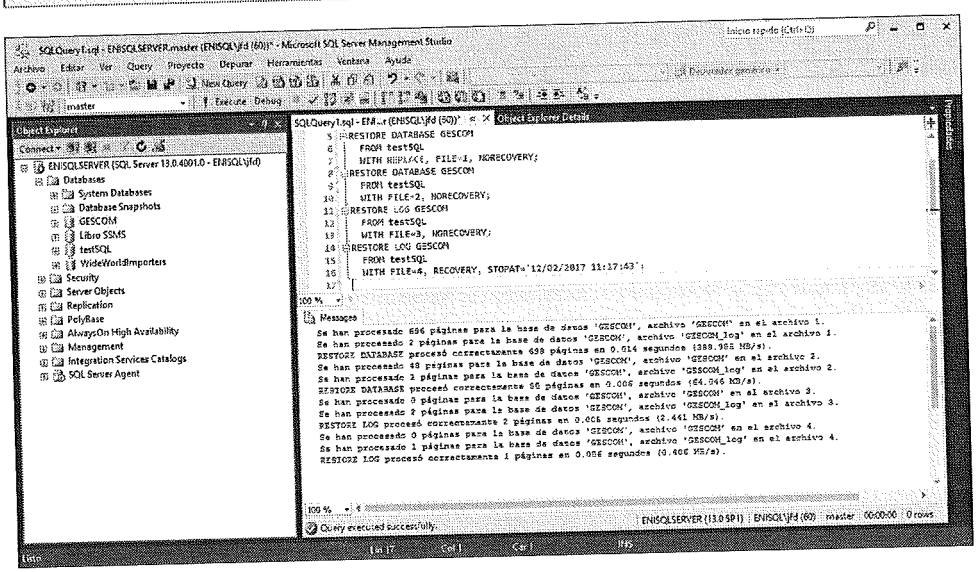


En el momento de restaurar los diarios de transacciones, es posible recrear las transacciones hasta una fecha y hora determinadas. Para aprovechar esta funcionalidad, es necesario utilizar la opción STOPAT junto con la opción RECOVERY.

Esta funcionalidad permite restaurar una base de datos parándose en un instante determinado y, por lo tanto, sin volver a crear las transacciones que se sabe que son perjudiciales.

Durante la restauración de un diario de transacciones desde SQL Server Management Studio, es posible especificar el límite en el tiempo en la ventana **Backup Timeline**, a la que se puede acceder pulsando el botón **Timeline** desde el cuadro de diálogo que presenta las opciones de restauración de la base de datos.


Backup Timeline: GESCOM
Ready
Restore to
Last backup taken
Specific date and time
Date:
12/02/2017
Time:
10:39:35
Timeline Interval:
Hour
10:30
10:45
11:00
11:15
11:30
Legend
Full Database Backup
Differential Database Backup
Transaction Log Backup
Tail-Log
OK
Cancel
Help


SQLQuery1.sql - ENISQLSERVER.master (ENISQL\jfd (60))* - Microsoft SQL Server Management Studio
Archivo Editar Ver Query Proyecto Depurar Herramientas Ventana Ayuda
Object Explorer
ENISQLSERVER (SQL Server 13.0.4001.0 - ENISQL\jfd)
Databases
System Databases
Database Snapshots
GESCOM
Libro SSMS
testSQL
WideWorldImporters
Security
Server Objects
Replication
PolyBase
AlwaysOn High Availability
Management
Integration Services Catalogs
SQL Server Agent
RESTORE DATABASE GESCOM
FROM testSQL
WITH REPLACE, FILE=1, NORECOVERY;
RESTORE DATABASE GESCOM
FROM testSQL
WITH FILE=2, NORECOVERY;
RESTORE LOG GESCOM
FROM testSQL
WITH FILE=3, NORECOVERY;
RESTORE LOG GESCOM
FROM testSQL
WITH FILE=4, RECOVERY, STOPAT='12/02/2017 11:17:43';
Messages
Se han procesado 696 páginas para la base de datos 'GESCOM', archivo 'GESCOM' en el archivo 1.
Se han procesado 2 páginas para la base de datos 'GESCOM', archivo 'GESCOM_log' en el archivo 1.
RESTORE DATABASE procesó correctamente 698 páginas en 0.014 segundos (389.985 MB/s).
Se han procesado 48 páginas para la base de datos 'GESCOM', archivo 'GESCOM' en el archivo 2.
Se han procesado 2 páginas para la base de datos 'GESCOM', archivo 'GESCOM_log' en el archivo 2.
RESTORE DATABASE procesó correctamente 50 páginas en 0.006 segundos (64.046 MB/s).
Se han procesado 0 páginas para la base de datos 'GESCOM', archivo 'GESCOM' en el archivo 3.
Se han procesado 2 páginas para la base de datos 'GESCOM', archivo 'GESCOM_log' en el archivo 3.
RESTORE LOG procesó correctamente 2 páginas en 0.006 segundos (2.441 MB/s).
Se han procesado 0 páginas para la base de datos 'GESCOM', archivo 'GESCOM' en el archivo 4.
Se han procesado 1 páginas para la base de datos 'GESCOM', archivo 'GESCOM_log' en el archivo 4.
RESTORE LOG procesó correctamente 1 páginas en 0.006 segundos (0.400 MB/s).
Query executed successfully.
ENISQLSERVER (13.0 SP1) ENISQL\jfd (60) master 00:00:00 0 rows

2.3.4 A partir de una copia de seguridad de archivo o de un grupo de archivos

Las operaciones que hay que realizar, y su orden, son las mismas que en el marco de una restauración total de la base de datos. La instrucción RESTORE se completa después del nombre de la base con el nombre del grupo de archivos afectados por la restauración.

2.4 La restauración de las bases de datos de sistema dañadas

Dependiendo de la gravedad de la situación, hay diferentes posibilidades.

2.4.1 Restauración a partir de una copia de seguridad

Si el servidor puede iniciarse, es posible restaurar las bases de datos de sistema usando un comando RESTORE DATABASE.

2.4.2 Reconstrucción de bases de datos de sistema

La reconstrucción de las bases de datos de sistema es posible lanzando de nuevo el proceso de instalación de la instancia de SQL Server.

Observación

La reconstrucción de las bases de sistema borra las bases ***master, msdb*** *y* ***model*** *existentes.*

Cuando las bases de sistema han sido reconstruidas, es posible reiniciar los servicios y proceder a la restauración desde una copia de seguridad válida.

2.5 La restauración en línea

La posibilidad de restauración en línea solo está disponible en la edición Enterprise de SQL Server.

La restauración en línea, es decir, efectuar un procedimiento de restauración de la base cuando los usuarios continúan trabajando en la base de datos, solo es posible si el grupo de archivos primario está intacto, es decir, que la operación de restauración va a afectar a otros grupos de archivos.

Como todo proceso de restauración, se desarrolla en dos momentos:

- Restaurar la información a partir de la última copia de seguridad completa.
- Recrear las transacciones presentes en el registro y restaurar el último registro con la opción RECOVERY.

El proceso de restauración en línea puede generar transacciones llamadas diferidas. Este tipo de transacción puede aparecer cuando alguna información no está accesible en el momento del inicio de la base. La transacción no puede, por lo tanto, recrearse de manera automática. El motor de SQL Server intentará ejecutar esta transacción, que manipula los datos afectados por la operación de restauración, cuando el proceso de restauración termine.

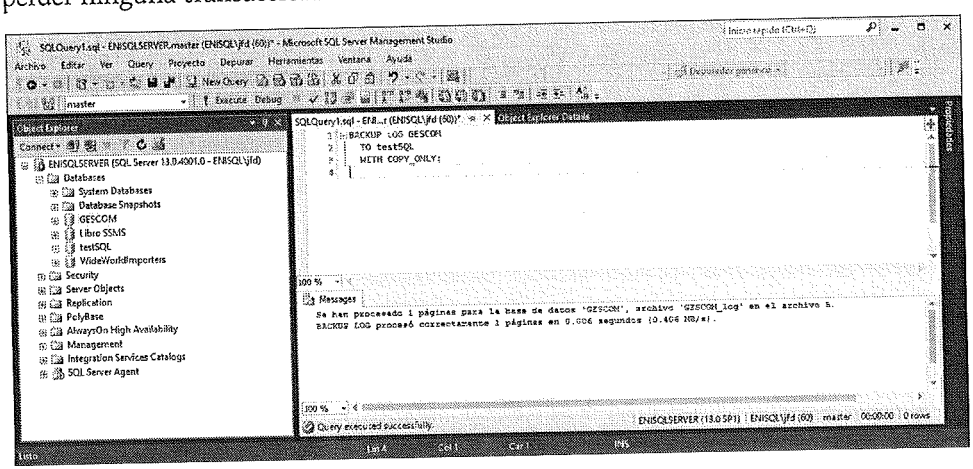
De hecho, es después de un inicio de la base de datos sin errores de entrada/salida cuando las transacciones diferidas pueden eliminarse.

Aplicación de una restauración en línea

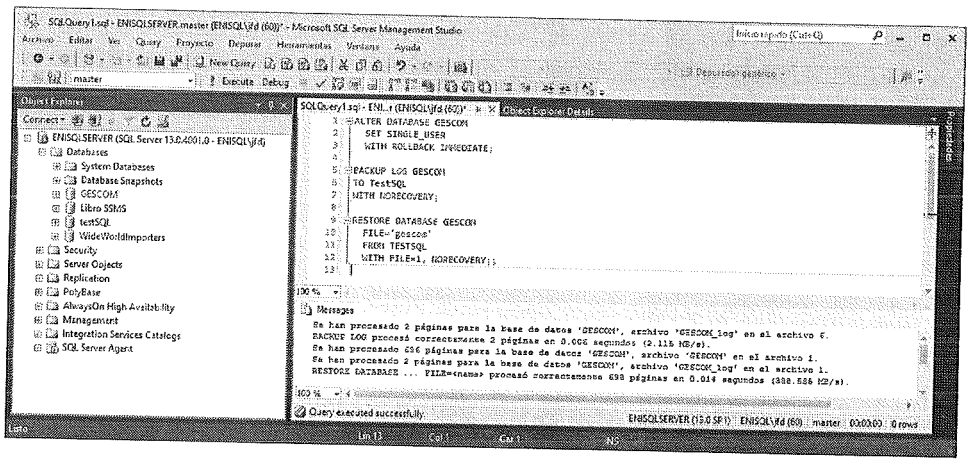
En caso de que el archivo esté dañado en una base de datos, es posible restaurar este archivo de manera aislada y después hacer una copia de seguridad del diario de transacciones actual. Por último, hay que restaurar los diferentes diarios de transacciones para actualizar la información que contiene el archivo.

Este tipo de proceso se va a ilustrar en la base GESCOM, simulando la pérdida de un archivo que no pertenece al grupo PRIMARY.

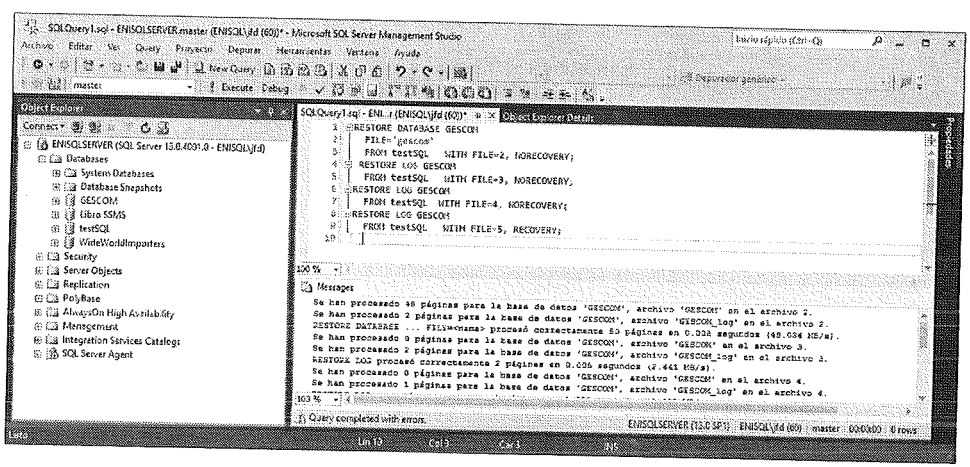
Es necesario hacer una copia de seguridad del diario de transacciones actual para no perder ninguna transacción.



La segunda etapa consiste en restaurar el archivo GESCOM a partir de la última copia de seguridad completa.



Para terminar, es posible restaurar los diarios de transacciones y hacer que la base de datos esté plenamente operativa.



3. Servidor de seguridad

Un servidor de seguridad presenta una doble ventaja. Para empezar, ofrece una garantía frente a la solución de problemas extremadamente rápida en caso de error del servidor principal, minimizando la pérdida de los datos. Además, el servidor de seguridad se alimenta de las copias de seguridad realizadas sobre el servidor principal, lo que permite validar todas las copias de seguridad, y esto es un punto importante. Como contrapartida, el servidor de seguridad representa una inversión elevada, ya que esta máquina no puede servir para un entorno de explotación si deseamos obtener una solución de reemplazo que se pueda establecer rápidamente.

3.1 Instalación del servidor de seguridad

El servidor de seguridad es un segundo servidor SQL que contiene una imagen en espejo del servidor de producción. El servidor de seguridad puede utilizarse en modo de solo lectura para reducir la carga de trabajo del servidor de producción o bien para sustituir al servidor de producción cuando este último esté dañado. El servidor de seguridad también puede utilizarse para ejecutar instrucciones DBCC, lo que permite verificar la coherencia de la base sin añadir una carga de trabajo adicional en el servidor de producción.

En primer lugar, es necesario restaurar una copia de seguridad de todas las bases del servidor de producción. Si los directorios son diferentes, es preciso utilizar la opción MOVE TO del comando RESTORE para cargar la ruta de los archivos de la base.

Todas las restauraciones deben tener la opción NORECOVERY o STANDBY. Esta última opción se detalla a continuación.

El servidor de seguridad será actualizado por medio de las copias de seguridad del registro de transacciones.

3.2 Uso del servidor de seguridad en modo de solo lectura

Dado que el servidor de seguridad es una imagen del servidor de producción, es posible utilizar este servidor para los clientes que realizan únicamente operaciones de lectura (SELECT). Esto va a permitir descargar, al servidor de producción, de la gestión de las consultas de lectura de datos, que pueden llegar a ser muy pesadas, fundamentalmente en el caso de análisis.

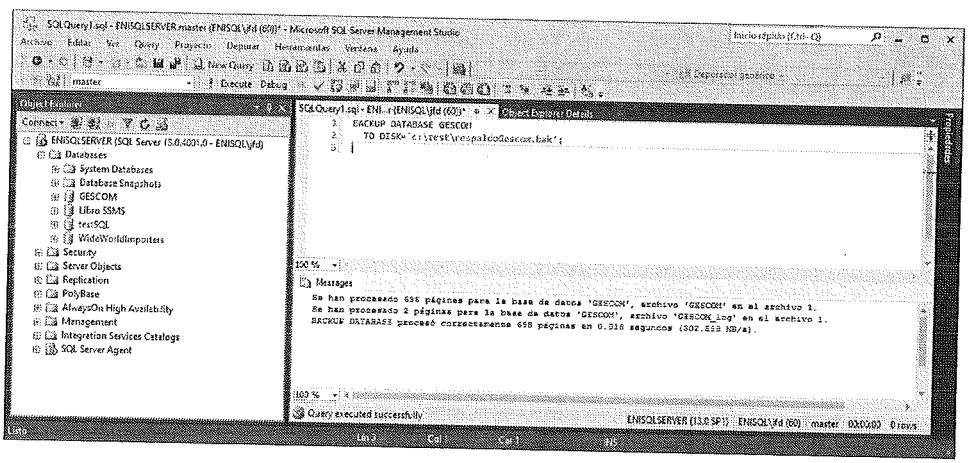
Con SQL Server, todos los servidores activos cubiertos con la SA (*Software Assurance*) permiten instalar un servidor de seguridad. De hecho, solo los servidores activos necesitan una licencia. Sin embargo, preste atención a este servidor de seguridad, porque no debe tener ninguna carga de trabajo. Las conexiones de usuario, incluidas las necesarias para las tareas de informes, no son posibles.

El problema que se plantea es el de la restauración. En efecto, la opción NORECOVERY, que permite restaurar los registros, no permite que nadie utilice la base de datos. Para evitar este problema, existe la opción STANDBY, que autoriza únicamente las operaciones de lectura sobre la base de datos. Esta opción necesita la utilización de un archivo que servirá para anular las modificaciones aportadas por la restauración de otros registros de transacciones.

3.3 Puesta en marcha de un servidor de seguridad

El punto de partida de la puesta en marcha de un servidor de seguridad es normalmente una copia de seguridad completa de la base de origen.

En el ejemplo siguiente, la copia de seguridad de la base GESCOM está realizada en una unidad física.



En una segunda etapa, esta copia de seguridad debe restaurarse en el servidor de seguridad. Dado que las ubicaciones de los archivos físicos no son necesariamente las mismas, se utiliza la opción MOVE TO en el proceso de restauración.

En el ejemplo siguiente, es otra instancia de SQL Server la que actúa de servidor de seguridad.

```
SQLQuery1.sql - ENI...r (ENISQL\jfd (60))*   Object Explorer Details
1  RESTORE DATABASE GESCOM
2    FROM DISK='c:\test\respaldoGescom.bak'
3    WITH MOVE 'Gescom'
4    TO 'C:\datos\gescom.mdf',
5    MOVE 'Gescom_Log'
6    TO 'C:\datos\gescom.ldf',
7    STANDBY='c:\test\StandByGescomRespaldo.ldf';
8
```

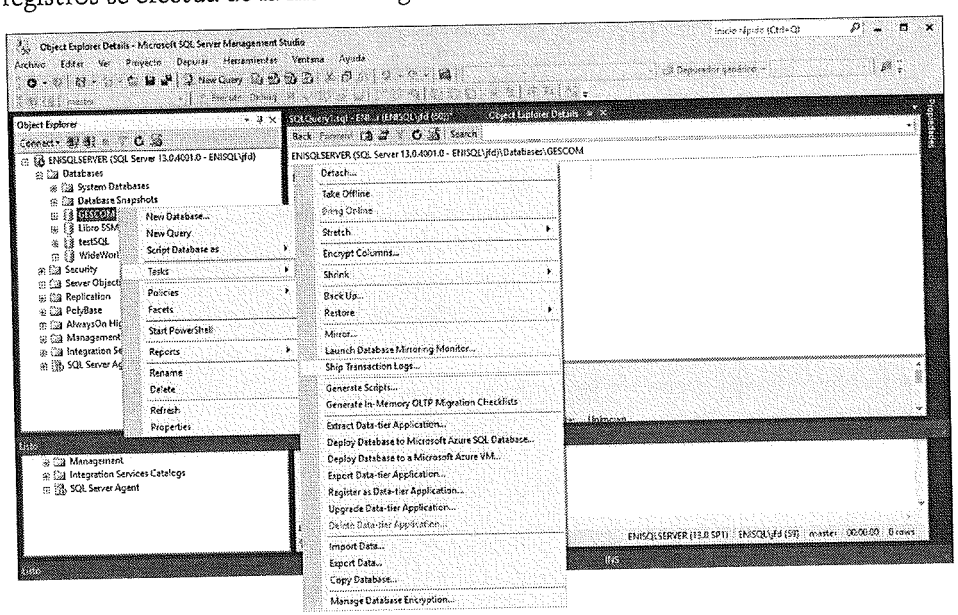
Observación

Con el objetivo de simplificar los elementos de sintaxis, la base GESCOM inicial no contiene el catálogo de texto completo.

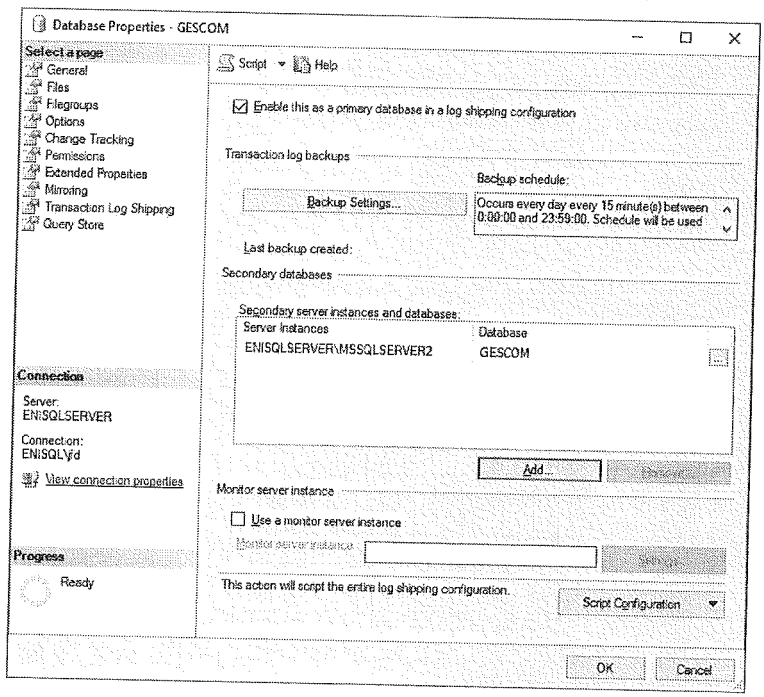
Seguidamente, las copias de seguridad de los registros de la base principal se restauran en el servidor de seguridad con la opción STANDBY o NORECOVERY.

Este proceso de envío de los archivos diarios puede automatizarse con SQL Server configurando el servidor principal para que sitúe una copia de seguridad del registro en una carpeta accesible por el servidor de seguridad.

Desde SQL Server Management Studio, la configuración del envío automático de los registros se efectúa de la manera siguiente.



Es necesario definir la base de datos como base de datos primaria, a continuación definir la ruta local o de red de acceso a la copia de seguridad del registro. Posteriormente, es posible definir el servidor o los servidores a los que se va a enviar la copia del registro, así como el modo de restauración deseado del registro. Las operaciones de copia de seguridad y de restauración se planifican.



Todas las operaciones relativas al envío de los registros pueden seguirse mediante una base de datos testigo que a va consignar todos los eventos relativos a esta operación.

El procedimiento de envío automático de los registros se establece para bases de datos, cuya estructura física permanece razonable en términos de complejidad (algunos archivos).

Evidentemente, este proceso se basa en los trabajos planificados y SQL Server Agent debe estar activo.

3.4 Cómo trabajar con el servidor de seguridad

Si el servidor de producción se avería, el servidor de seguridad debe tomar el relevo lo más rápidamente posible. Es indispensable un enfoque riguroso para perder la mínima cantidad de datos.

3.4.1 Conexión

Servidor de producción

Copia de seguridad del registro de transacciones

Si la operación es posible, debe realizarse una copia de seguridad del registro de transacciones para que las transacciones validadas desde la última copia de seguridad no se pierdan.

Desconexión

Con el objetivo de evitar cualquier conflicto, el servidor de producción debe estar físicamente desconectado de la red.

Servidor de seguridad

Nombre del servidor

Para no molestar a los puestos clientes que utilizan el servidor, el cambio de nombre permite una relativa transparencia en relación con las aplicaciones.

Restauración del registro

El registro que se ha salvaguardado en el servidor de producción debe restaurarse con la opción RECOVERY para hacer accesible la base de datos en modo de lectura y escritura.

3.4.2 Restauración del servidor de producción

Cuando se haya solucionado el problema en el servidor de producción, este debe retomar su rol inicial.

Servidor de seguridad

- Realizar una copia de seguridad de la base de datos y de todos los registros de transacciones.
- Desconectar físicamente el servidor de la red.

Servidor de producción

- Restaurar las copias de seguridad de la base de datos y de los registros de transacciones. Posteriormente, conectar de nuevo el servidor a la red.

3.4.3 Restablecimiento del ordenador SQL Server de seguridad

El servidor de seguridad debe cambiar de nombre antes de su conexión a la red. Para que se convierta en imagen del servidor de producción, lo más sencillo es realizar una copia de seguridad completa de la base de datos y restaurarla con la opción STANDBY o NORECOVERY.

Capítulo 10
Herramientas adicionales

1. La auditoría de la actividad de SQL Server

SQL Server ofrece numerosas herramientas de seguimiento del rendimiento del servidor y de la actividad de los usuarios. Esta vigilancia es importante, ya que permite detectar los cuellos de botella y mejorar los tiempos de respuesta del servidor.

Antes de iniciar la auditoría, conviene definir claramente:

- Los objetivos que se persiguen.
- La herramienta más apropiada. Las más flexibles son SQL Profiler y el Analizador de rendimiento de Windows.
- Si es necesario monitorizar SQL Server o su entorno para conseguir los objetivos.

Esta auditoría se puede definir en la instancia de SQL Server o en una o varias bases de datos. La auditoría va a seguir los eventos y registra el seguimiento de estos eventos en el diario de auditoría. Este diario de auditoría puede ser un archivo, el diario de eventos de seguridad o el diario de los eventos de aplicaciones de Windows. Sea cual sea el objetivo de la auditoría, el archivo se debe guardar regularmente con objeto de garantizar el espacio necesario para registrar los eventos.

La escritura en el diario de seguridad solo es posible si el contexto de seguridad en el que se ejecuta el servicio SQL Server dispone de este permiso. Es el caso por defecto de las cuentas de sistema local, servicio local y servicio de red. Para las otras cuentas, es necesario aplicar la regla **Crear auditorías de seguridad**.

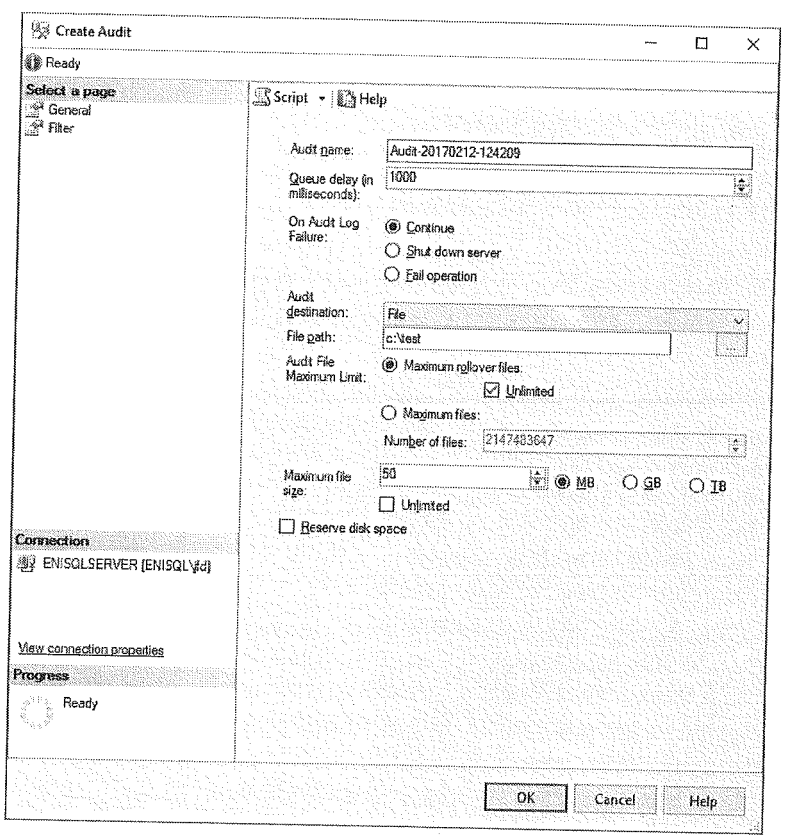
Para poder realizar o modificar una auditoría, es necesario ser miembro del rol **sysadmin**. También es posible auditar cada modificación de la auditoría.

Es importante notar que la puesta en marcha de una auditoría puede tener un efecto negativo en el rendimiento, ya que la activación de los contadores de auditoría consume recursos del servidor y, por lo tanto, hay menos recursos disponibles para responder a las peticiones de los usuarios.

1.1 Definir una auditoría en el servidor

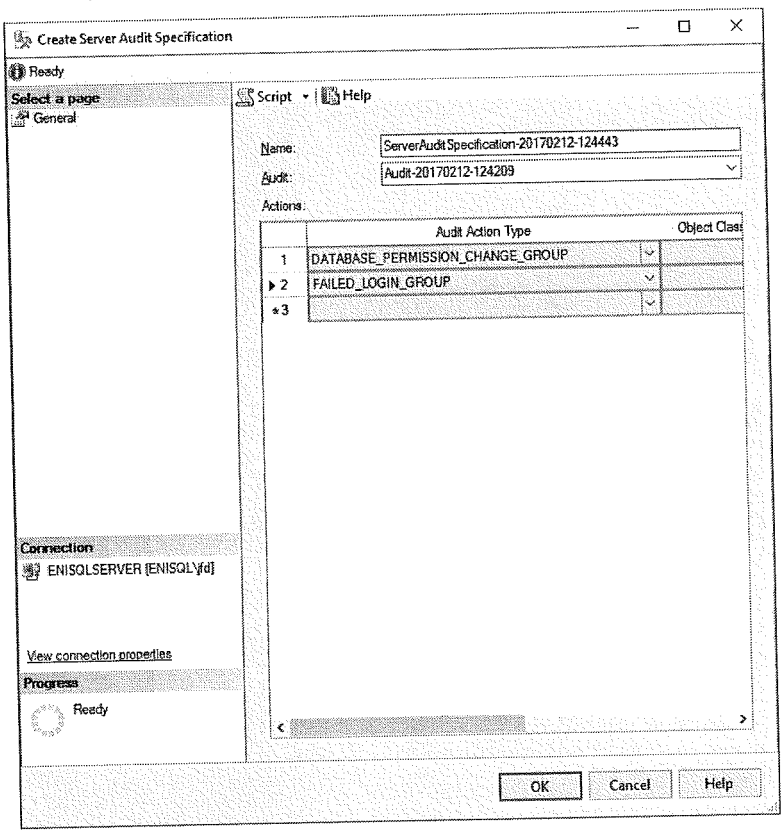
El establecimiento de este tipo de auditoría se inicia con la creación y configuración de un objeto de tipo SQL Server Audit. Este objeto podrá utilizarse posteriormente para definir que la especificación de esta auditoría sea de nivel servidor.

La creación de un objeto de tipo SQL Server Audit es posible desde el nodo **Security - Audits** del explorador de objetos. Es necesario seleccionar la opción **New Audit** desde el menú contextual asociado al nodo. Entonces aparece el cuadro de diálogo de definición de una nueva auditoría.



A este nivel se define el destino de la auditoría, así como el comportamiento de esta sobre la instancia si se produce falta de espacio en el registro.

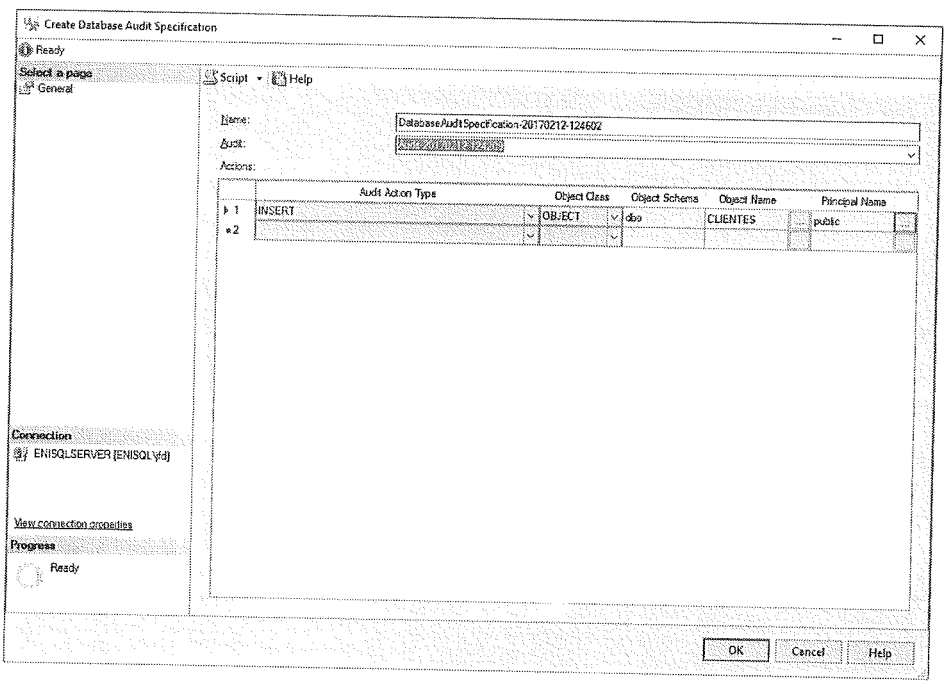
Es posible definir una especificación de auditoría de nivel servidor por medio de la opción **New Server Audit Specification** desde el menú contextual del nodo **Security - Server Audit Specifications**.



La definición de la especificación permite la selección de los eventos auditados, así como precisar la auditoría utilizada.

1.2 Definir una auditoría en la base de datos

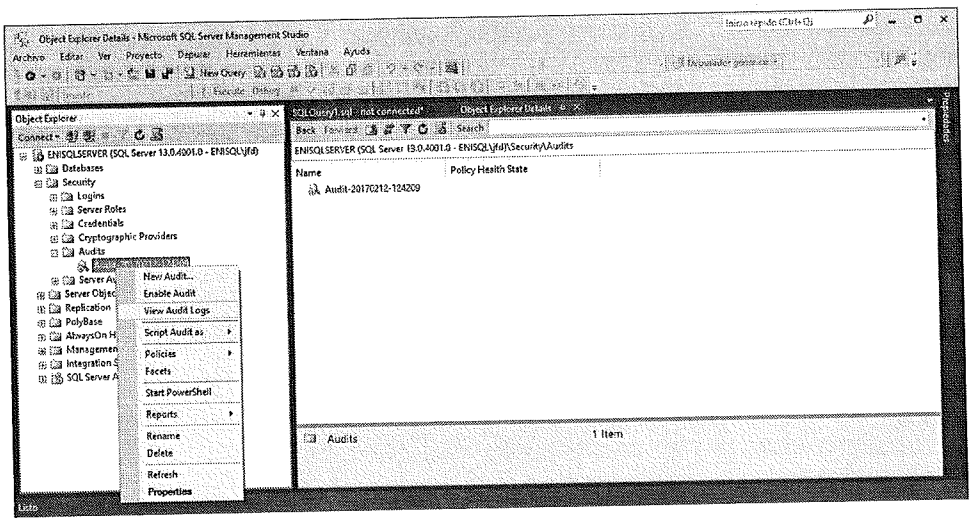
A nivel de la base de datos, como a nivel del servidor, la definición de una nueva auditoría se basa en los objetos SQL Server Audit y en las especificaciones. Si los objetos SQL Server Audit se definen a nivel del servidor, las especificaciones se definen a nivel de la base de datos. El cuadro de diálogo que permite crear una nueva especificación de auditoría de base de datos está accesible desde el nodo **Security - Database Audit Specifications**.



Después de asociar la especificación a un objeto SQL Server Audit, es posible seleccionar uno o varios eventos para auditar.

1.3 Visualizar el registro de auditoría

El registro de auditoría está accesible desde el nodo **Security - Audits** del explorador de objetos. A continuación, es necesario seleccionar la auditoría cuyo registro se desea visualizar. Para ello, hay que seleccionar la opción **View Audit Logs** desde el menú contextual asociado.



Se ejecuta la visualización y de esta manera es posible recorrer el registro de auditoría.

1.4 La auditoría C2

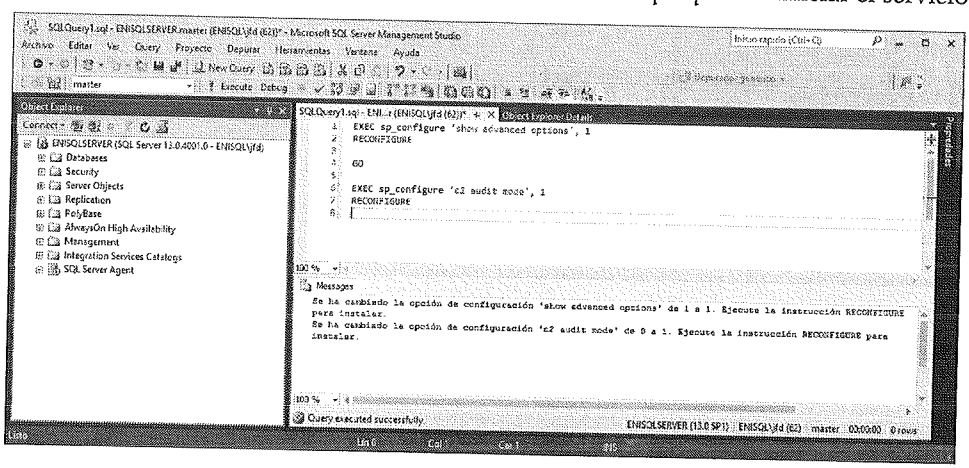
El generador de perfiles almacena el conjunto de datos auditados en los archivos de registro. El tamaño máximo de un archivo de registro es de 200 MB. Cuando se alcanza este límite, el archivo se cierra y se crea y abre otro de manera automática. Cuando no hay más espacio libre en disco, la instancia de SQL Server se para. Por tanto, será necesario liberar espacio en el disco donde se almacenan los archivos de auditoría antes de reiniciar la instancia de SQL Server.

En SQL Server es posible utilizar la opción **Modo auditoría C2** para auditar los intentos con éxito o fracaso de utilización de sentencias DML o de objetos de la base de datos. El análisis de la información recogida de esta manera permite establecer una estrategia de seguridad previendo todas las violaciones. Este modo de auditoría también es un buen medio para ver la actividad de la base de datos.

La información se almacena en archivos de 200 MB como máximo, en los directorios msql($nombre_instancia)\data. Si la auditoría permanece activa un cierto tiempo, la información se repartirá automáticamente en varios archivos (se crea un nuevo archivo cuando el archivo actual alcanza los 200 MB).

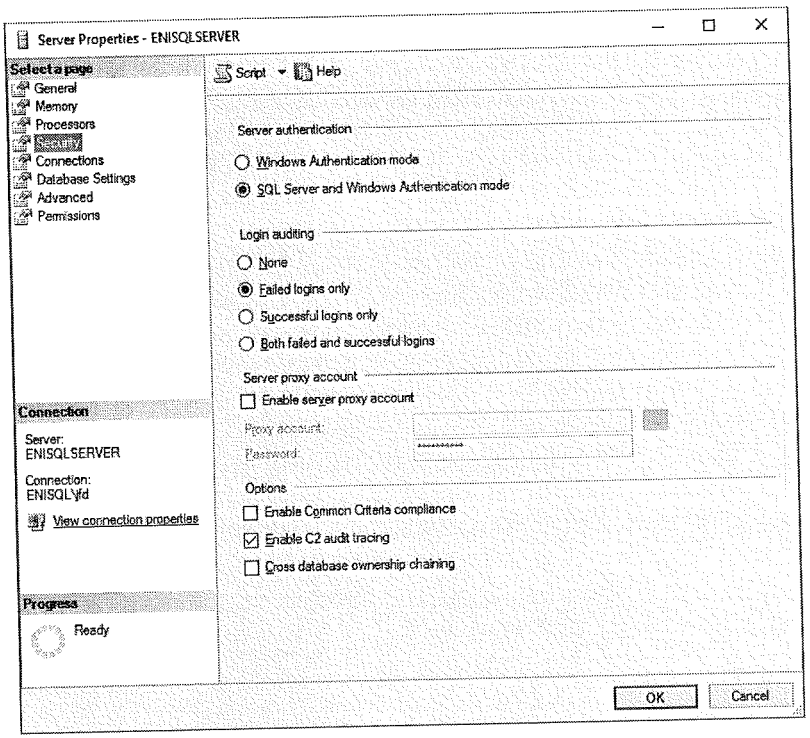
La auditoría de nivel C2 permanece activa mientras no se solicite su desactivación o mientras no se pare SQL Server. Únicamente los miembros del rol **sysadmin** pueden activar/desactivar la auditoría de nivel C2.

El establecimiento de la auditoría de nivel C2 se realiza con el procedimiento **sp_configure**. El comando RECONFIGURE permite tener en cuenta las modificaciones de configuración de manera inmediata sin tener que parar e iniciar el servicio.



El establecimiento de la auditoría de nivel C2

También es posible activar esta opción en la página **Security** de la ventana de propiedades del servidor, marcando la casilla **Enable C2 audit tracing**.



Observación

El establecimiento de una auditoría de nivel C2 puede degradar el rendimiento del servidor mientras la auditoría está activa.

Si el directorio de almacenamiento de los archivos de auditoría está lleno, SQL Server se para automáticamente. Si la auditoría no está configurada en modo de inicio automático, entonces es posible reiniciar SQL Server. En caso contrario, es necesario liberar espacio en disco duro con el objetivo de que la auditoría pueda registrar su información desde el mismo momento del inicio de SQL Server. Es posible iniciar SQL Server en línea de comandos con la opción **f** para impedir el inicio automático de la auditoría.

Si la auditoría impide el correcto inicio de la instancia de SQL Server (el evento MSG_AUDIT_FORCED_SHUTDOWN se escribe en el registro), entonces es necesario iniciar la instancia en modo monousuario con la opción **-m**. Efectivamente, en modo monousuario, la auditoría no puede bloquear el inicio, pero el evento MSG_AUDIT_SHUTDOWN_BYPASS sí se escribirá en el registro.

2. El generador de perfiles

Para capturar la actividad del servidor y así ser capaces de analizar la carga de trabajo a la que está sometido un servidor es necesario utilizar los eventos extendidos desde SQL Server Management Studio. Esta nueva funcionalidad de SQL Server 2016 permite unificar el comportamiento de las instancias SQL Server instaladas localmente y el de las instancias presentes en Azure.

En efecto, la noción de evento extendido está presente en los dos tipos de instancias SQL Server. En el siguiente ejemplo, presentamos la configuración a través de SQL Server Management Studio, pero es posible definir sesiones mediante consultas SQL. El análisis de los datos, aunque la información se almacene en archivos externos, puede hacerse bajo la forma de consultas SQL. Esta solución permite paliar la interfaz relativamente simple para realizar el análisis de la información.

Para identificar los elementos a capturar, debemos definir en primer lugar las sesiones. Las sesiones son comparables a filtros. Así, en vez de capturar el conjunto de la actividad del servidor, precisamos a nivel de las sesiones el o los eventos que deseamos analizar, y también la cantidad de detalle que deseamos disponer de cada evento.

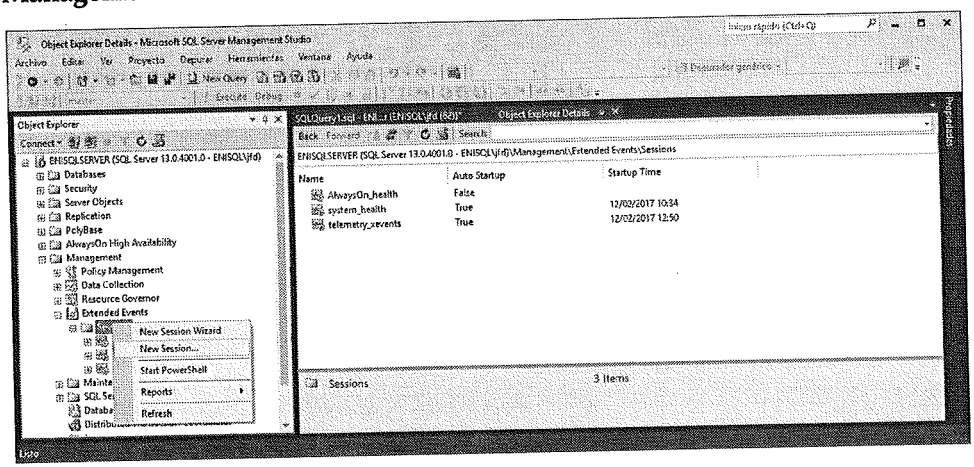
Para ayudarnos a definirlo, SQL Server proporciona plantillas de sesiones en las cuales nos podemos apoyar para definir más rápidamente la sesión.

Observación

Aunque la herramienta SQL Server Profiler siga estando disponible en SQL Server 2016, se desaconseja su uso. Esta herramienta se mantiene por razones de compatibilidad.

3. La creación de sesiones

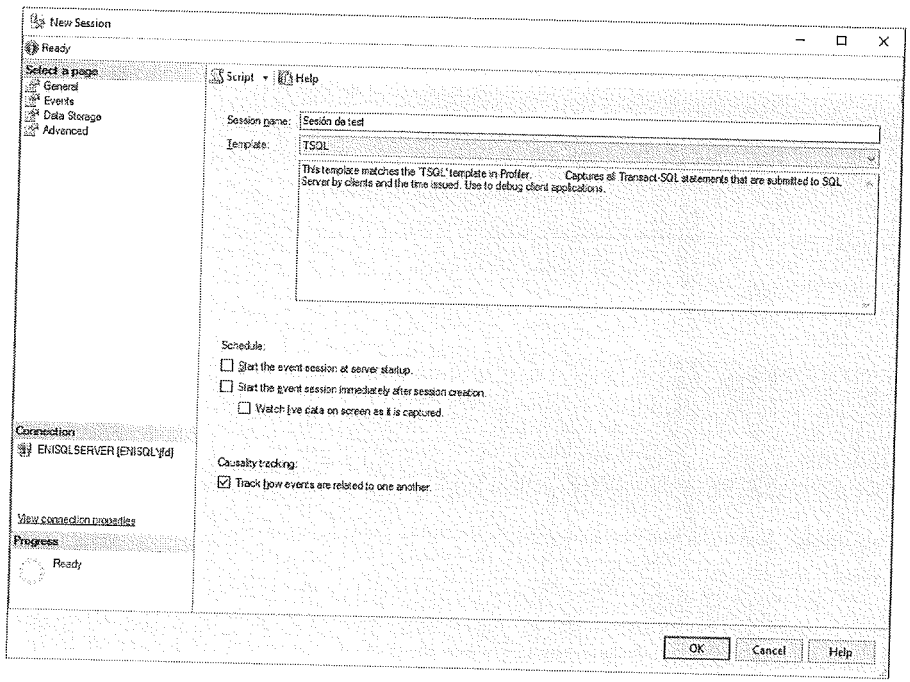
Para capturar la actividad, primero es necesario definir una nueva sesión desde la rama **Management - Extend Events - Sessions**.



La creación de una nueva sesión se puede hacer con el asistente o parametrizando directamente la sesión. El asistente permite construir paso a paso una plantilla mientras que el paso por el cuadro de diálogo permite crear la sesión definiendo las diferentes opciones en el orden deseado. La modificación de una sesión se realiza siempre desde el cuadro de diálogo de propiedades sea cual sea el modo de creación elegido.

Ejemplo

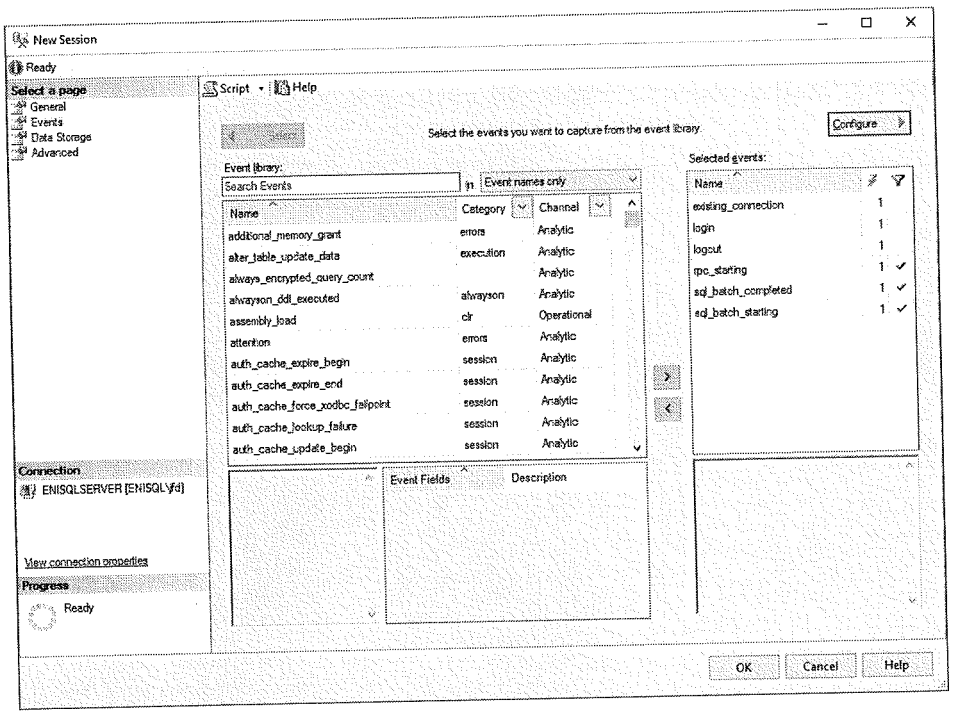
Creación de una sesión apoyándose en una plantilla TSQL



Desde la sección **Events** es posible seleccionar los eventos. En el caso de utilizar una plantilla, los eventos están predefinidos para su análisis.

Ejemplo

Eventos definidos por defecto en una plantilla TSQL



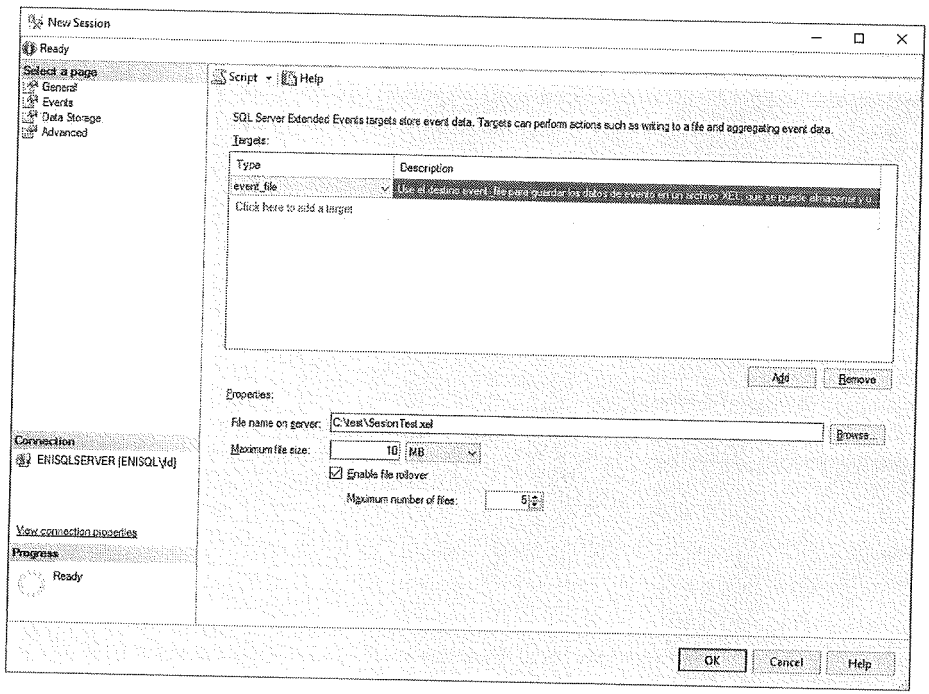
Por último, la sección **Data Storage** es importante ya que define la ubicación donde se almacenarán estos datos de tipo evento.

En el caso en el que queramos almacenarlos bajo la forma de un archivo, debemos elegir la opción **event file** e indicar el nombre y la ubicación del archivo así como su tamaño máximo y el número máximo de archivos almacenados para los eventos extendidos que se pueden crear.

Cuando todos los archivos están completos, esto no bloquea el servidor sino que el guardado de eventos se detiene.

Ejemplo

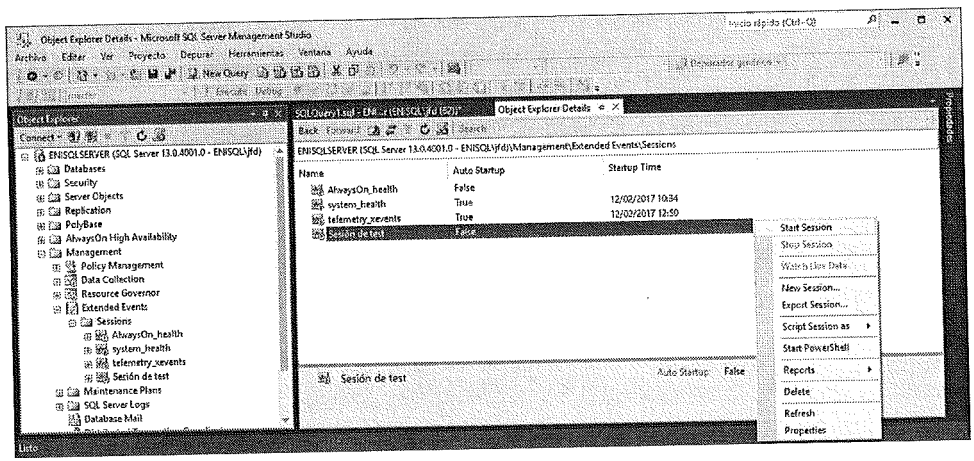
Creación de cinco archivos de 10 MB para el almacenamiento de eventos



4. Iniciar una sesión

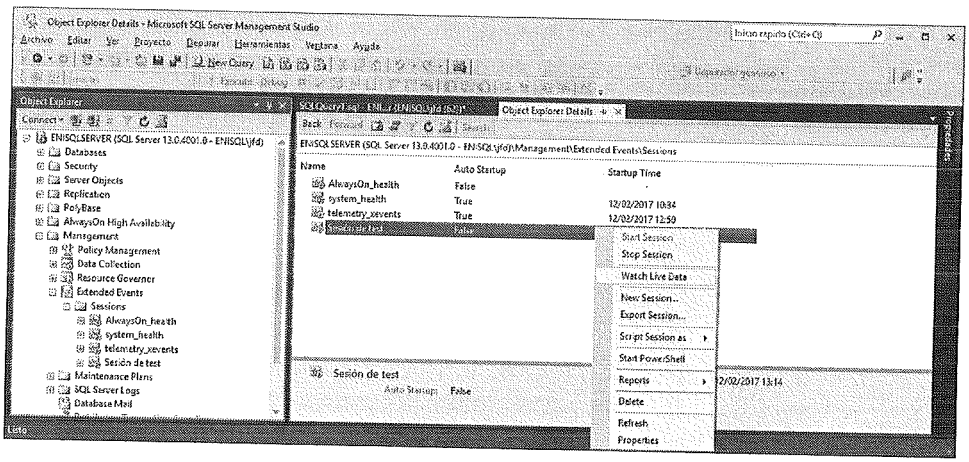
Para capturar la actividad, es necesario iniciar las sesiones.

Las sesiones están activas mientras no se detengan ni se alcance el límite del espacio de almacenamiento.

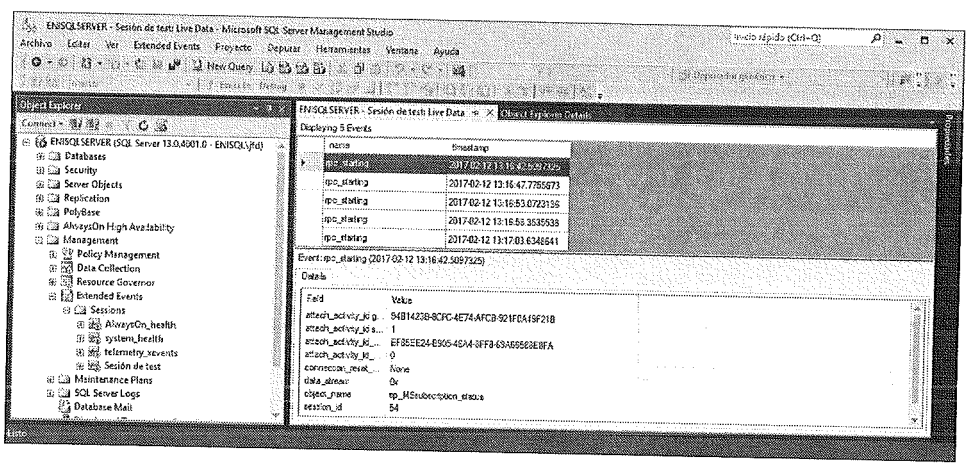


5. Analizar la información

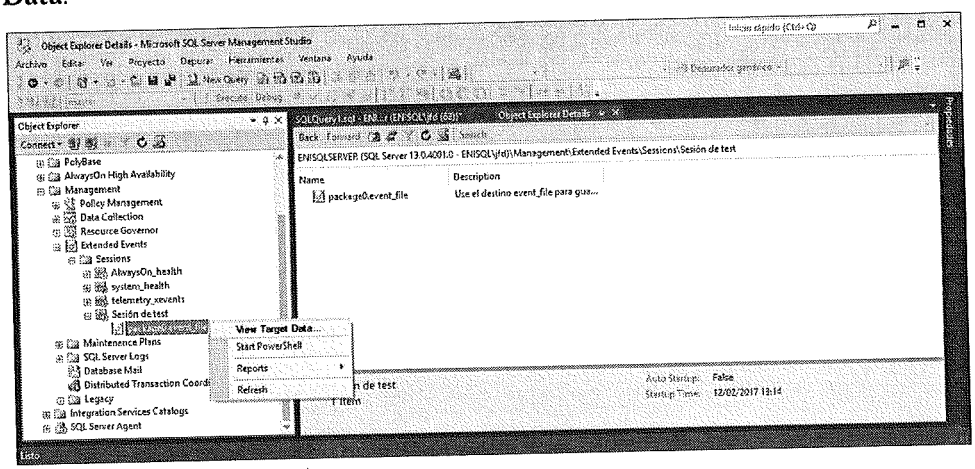
Mientras una sesión este activa y contenga información, es posible analizar esta información. Esta operación se realiza también desde SQL Server Management Studio. Para ello debemos ejecutar la herramienta de vigilancia de datos activos. Esta herramienta está accesible desde el menú contextual asociado a la sesión.



Se muestra la pantalla de presentación de datos y podemos visualizar el detalle de la información.



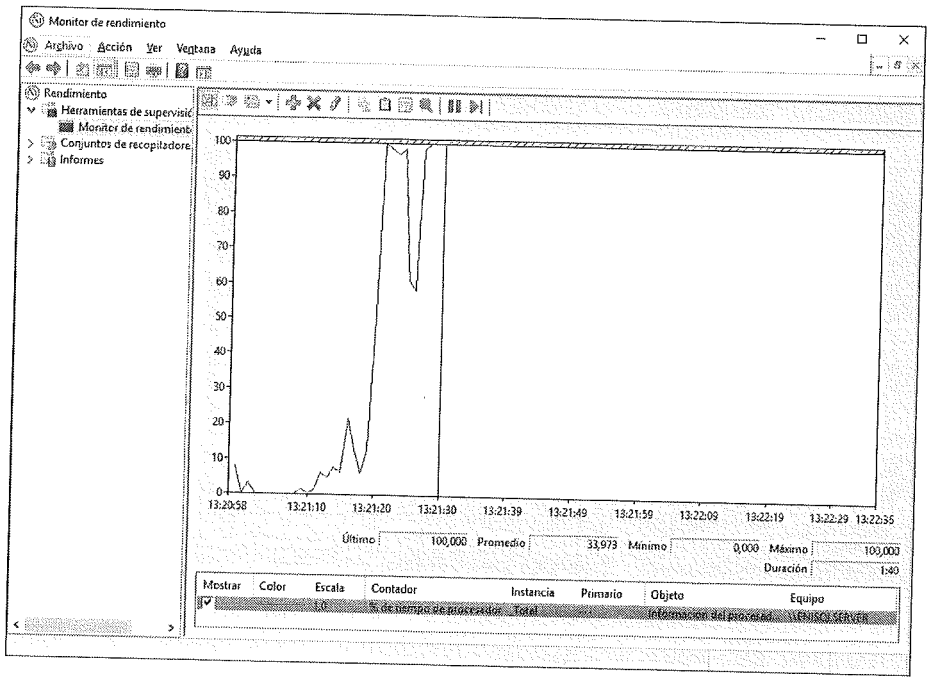
También es posible analizar los datos cuando la sesión ha dejado de estar activa. Este tipo de visualización es muy parecido al propuesto en el modo en vivo, aunque la forma de acceder es diferente ya que debemos solicitar en esta ocasión **View Target Data**.



6. El monitor de rendimiento (monitor de sistema)

Se trata del monitor de rendimiento de Windows, al que se han añadido numerosos contadores durante la instalación de SQL Server.

Además de estos contadores de serie, es posible definir diez contadores de usuario para poder adaptar la utilización del Monitor de rendimiento a las necesidades personales.



Los principales objetos específicos de SQL Server son:

- Agente de replicación: vigila a los agentes de replicación en curso de ejecución.
- Base de datos: vigila la utilización de la base de datos, como la cantidad de espacio de registro disponible y el número de transacciones activas.
- Instantáneas: vigila las instantáneas de las replicaciones.
- Distribución de replicación: vigila el número de comandos y transacciones leídas a partir de la base de datos de distribución.
- Fusión de replicación: vigila la ejecución de cada fusión que desplaza las modificaciones de datos, del suscriptor hacia el editor o bien a la inversa.
- Administrador de caché: permite vigilar la manera en la que SQL Server utiliza la memoria para almacenar objetos (procedimientos almacenados...).
- Administrador de la memoria intermedia: permite vigilar la manera en la que SQL Server utiliza la memoria para almacenar páginas de datos.
- Administrador de memoria: vigila la utilización global de la memoria.
- Lector del registro de transacciones: vigila al agente de lectura del registro de transacciones.
- Métodos de acceso: vigila el acceso a las páginas lógicas.

- Reservado para el usuario: la definición de los contadores (10 como máximo) es responsabilidad del usuario.
- Estadísticas generales: vigilan la actividad general del servidor.
- Estadísticas SQL: vigilan la compilación y el tipo de consultas enviadas a SQL Server.
- Dispositivo de copia de seguridad: vigila los dispositivos de copia de seguridad utilizados en las operaciones de copia de seguridad y de restauración.
- Bloqueos: un número mínimo de bloqueos es favorable para la concurrencia en el acceso, lo que mejora el rendimiento.
- Bloqueos internos: este objeto permite averiguar el tiempo medio de obtención de un bloqueo sobre un recurso, así como el número de solicitudes de bloqueos que no pueden ser satisfechas inmediatamente.

Los procedimientos almacenados reservados para la definición de los contadores de usuario son **sp_user_counter1** a **sp_user_counter10**. Las consultas definidas en estos procedimientos almacenados deben ser lo más simples posible para no aumentar la carga de trabajo ejecutado por el servidor. Es posible solicitar la ejecución de estos procedimientos en cualquier lugar (trigger, otros procedimientos...).

Los contadores personalizados están disponibles en el objeto SQL Server: **User Settable** del Monitor de rendimiento.

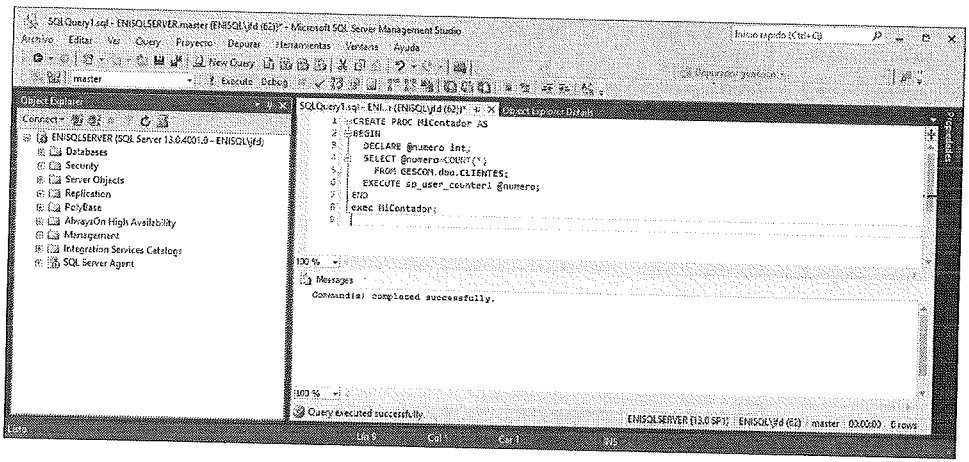
Para modificar el valor del primer contador de rendimiento, es necesario ejecutar el procedimiento **sp_user_counter1** que acepta como parámetro un único número entero.

Ejemplo

En el ejemplo siguiente, se crea un procedimiento almacenado para vigilar el número de líneas de la tabla de artículos.

Observación

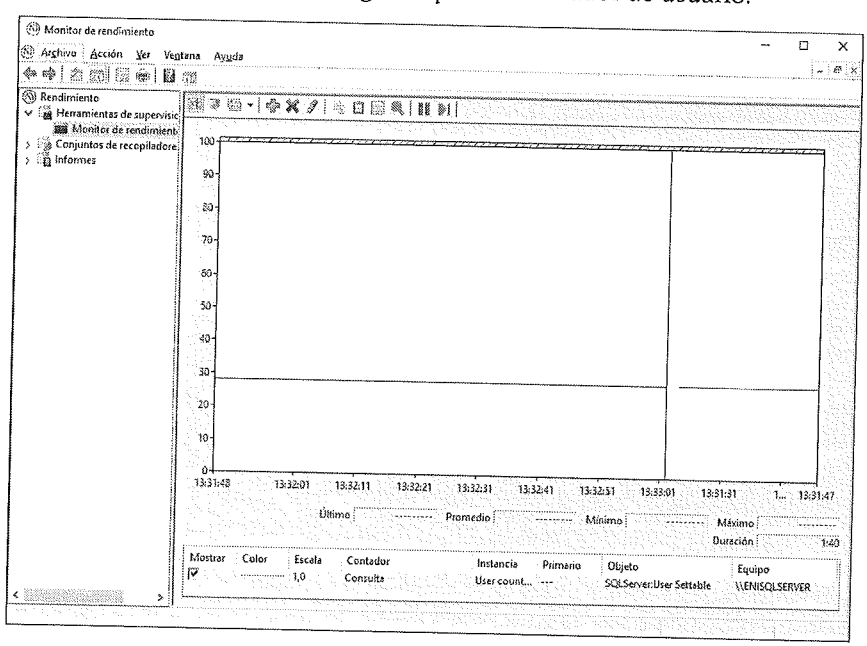
En el marco de un servidor de producción, es más recomendable hacer la llamada a este procedimiento en los triggers de tipo INSERT y DELETE asociados a esta tabla. De esta manera, el procedimiento no se ejecuta hasta que el número de líneas presentes en la tabla evoluciona.



Definición del procedimiento MiContador

Si el contador debe actualizarse de manera continua, es necesario integrar su ejecución en un bucle infinito.

En el monitor de sistema se vigila el primer contador de usuario.



Evolución del contador

SQL Server introduce también contadores de rendimiento para seguir la correcta aplicación de las nuevas sintaxis de comandos y evitar la utilización de las funcionalidades o elementos de sintaxis marcados como mantenidos por razones de compatibilidad anterior, pero destinados a desaparecer. Se trata de los contadores SQL Server Deprecated Features o Funcionalidades obsoletas.

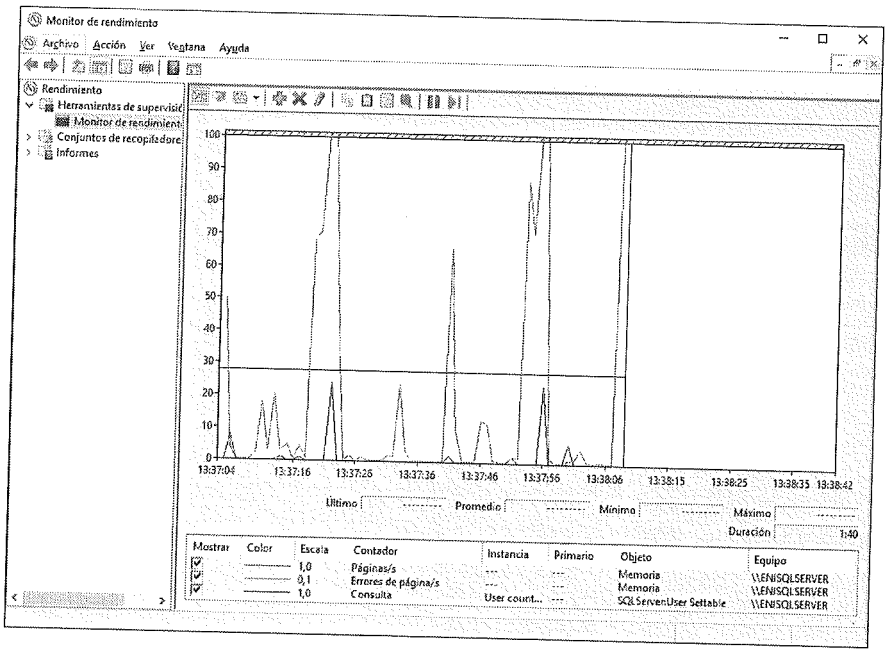
7. Optimización de la memoria y de la unidad central

Por defecto, SQL Server gestiona automática y dinámicamente la cantidad de memoria que necesita. Esta opción debe conservarse en la mayoría de los casos. Sin embargo, es posible fijar las cantidades de memoria mínima y máxima y el tamaño del trabajo.

El Monitor de rendimiento va a permitir vigilar la utilización de la memoria para asegurarse de que el consumo permanece dentro de los límites razonables y que ningún proceso incluido en SQL Server eche en falta memoria. Estos criterios corresponden a los contadores de Página/s y Bytes disponibles del objeto Memoria.

El contador Memoria: Página/s indica el número de páginas de memoria que son extraídas o bien escritas en disco por razones de paginación. Un valor elevado puede indicar una falta de memoria física.

El contador Memoria permite asegurarse de que la actividad del disco no está asociada al proceso de paginación de la memoria.

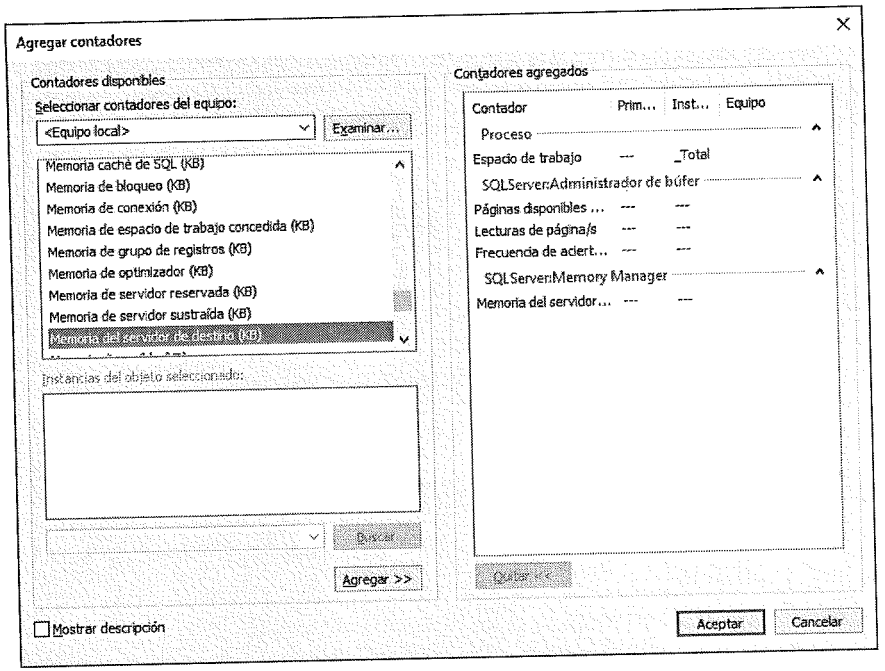


Vigilancia de la memoria del sistema

Para averiguar la cantidad de memoria utilizada por SQL Server, es necesario vigilar los contadores siguientes en el Monitor de rendimiento:

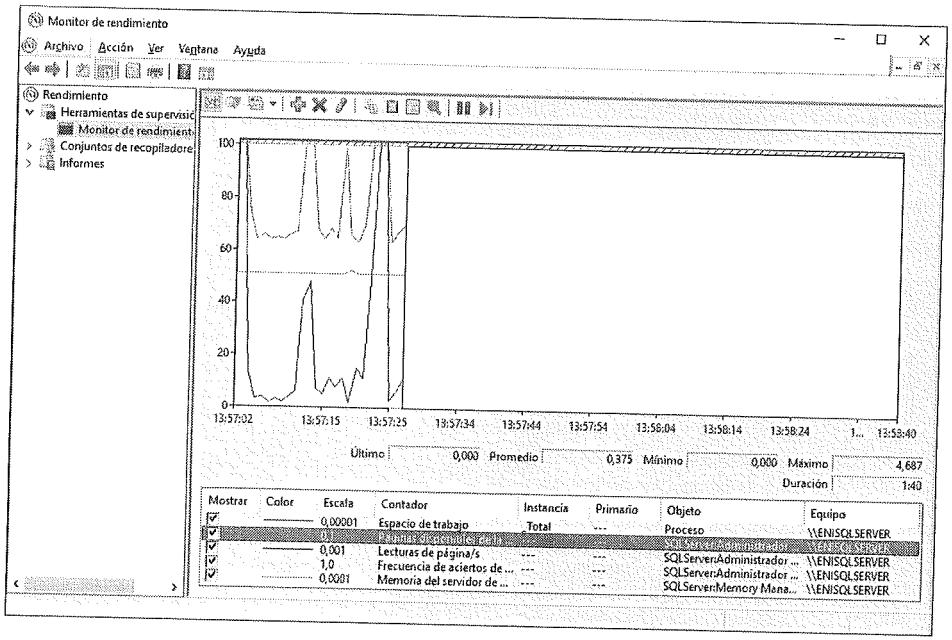
- Proceso\Franja de trabajo.
- SQL Server: Administrador de la memoria intermedia\Tasa de acceso al caché de memoria. Indica un porcentaje que representa el número de veces que las páginas de datos solicitadas se encuentran en la caché.
- SQL Server: Administrador de la memoria intermedia\Lectura de página/s. Número de páginas leídas por SQL Server cada segundo.
- SQL Server: Administrador de la memoria intermedia\Páginas libres de extensión. Número total de páginas de SQL Server no utilizadas.
- SQL Server: Memory Manager\Memoria del servidor de destino (KB). Este valor debe estar en relación con el volumen de memoria asignado a SQL Server.

Desde el monitor de rendimiento, estos contadores se presentan de la siguiente forma:

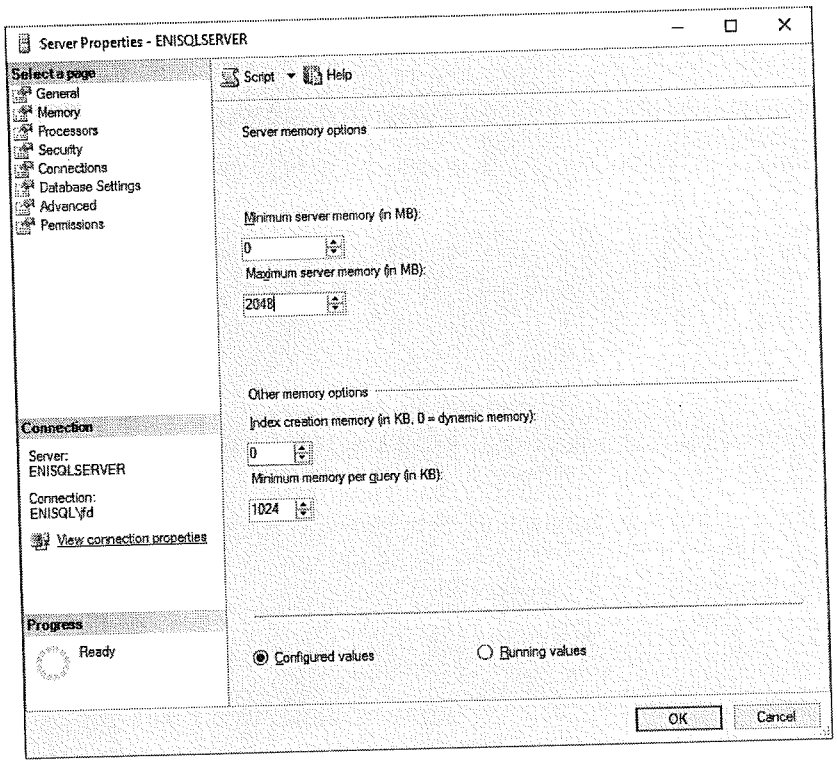


Para los contadores **Tasa de acceso al caché de memoria intermedia** del objeto **Administrador de memoria intermedia**, y **Memoria total del servidor** del objeto **Administrador de memoria**, los valores más significativos son:

- Tasa de presencia en la caché: 90 % es una tasa normal en curso de utilización, que asegura poca lectura física de los datos.
- Memoria total del servidor: si la memoria utilizada por SQL Server representa una parte importante de la memoria del puesto, puede que al puesto le falte memoria.



El tamaño de la memoria del servidor puede fijarse de manera estática por medio SQL Server Management Studio o mediante el procedimiento almacenado **sp_configure** con la opción **set working set size**.



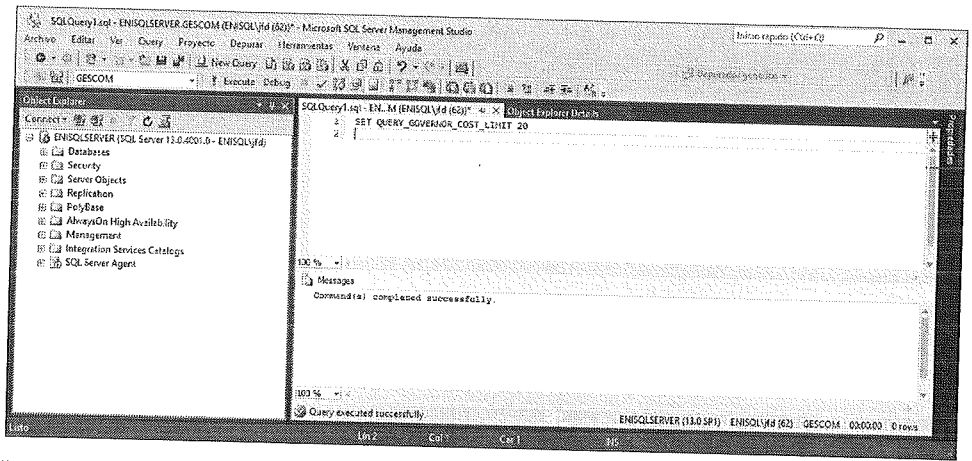
Configuración de la memoria de SQL Server

8. La limitación de los recursos utilizados por una consulta

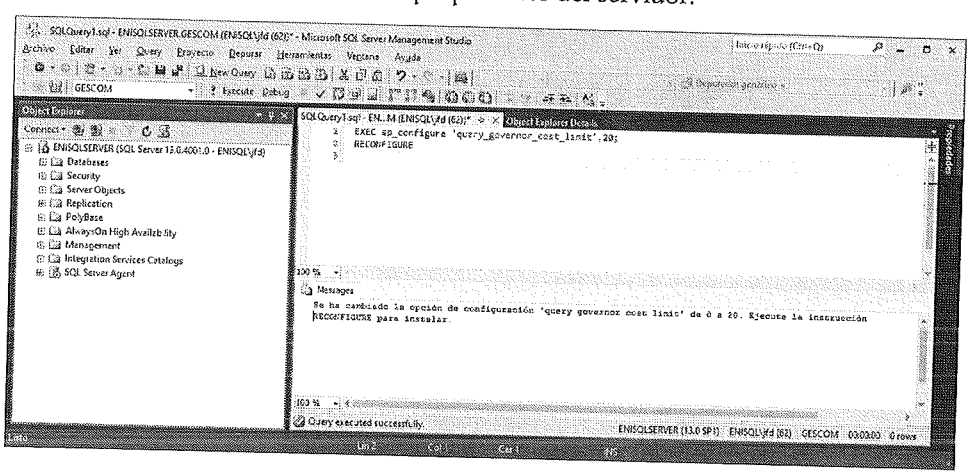
El coste de una consulta corresponde a la duración estimada (en segundos) de su ejecución. La opción **query governor cost limit** permite especificar un límite superior para la ejecución de una consulta.

Por defecto, esta opción recibe el valor 0, lo que autoriza la ejecución de todas las consultas. Si se asigna un valor positivo diferente de 0, entonces el administrador no autoriza la ejecución de todas las consultas, cuyo coste estimado es superior a este valor.

Esta limitación se puede especificar en el servidor por medio de **sp_configure** o bien en cada base de datos con **SET QUERY_GOVERNOR_COST_LIMIT**.



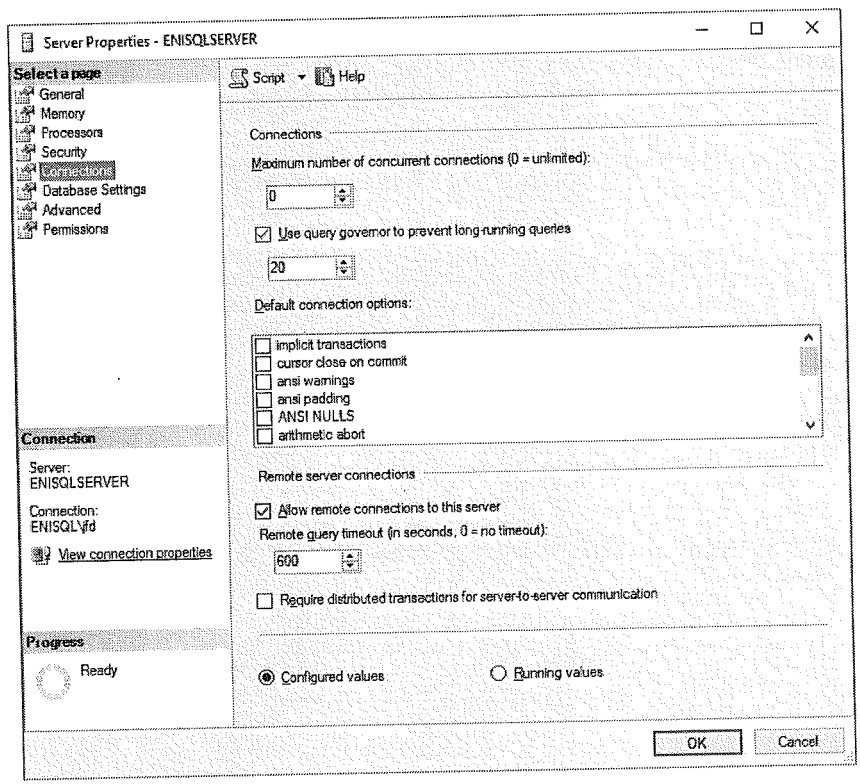
La configuración con la opción SET solo está disponible para el período actual de actividad de la instancia. Esta configuración no se conserva para el siguiente inicio de la instancia. Para conservar este valor, es necesario configurar la opción con **sp_configure** o bien mediante las propiedades del servidor.



Observación

La instrucción RECONFIGURE permite tener en cuenta los nuevos valores de las opciones de configuración sin tener que reiniciar el servidor.

Desde SQL Server Management Studio, esta opción es configurable en la ventana de las propiedades del servidor, en la página **Connections**, activando la casilla **Use query governor to prevent long-running queries** y precisando el coste máximo autorizado en la zona asociada.



9. El plan de ejecución de una consulta

SQL Server analiza las consultas, los procedimientos y los triggers. El optimizador de la consulta va a almacenar el plan de ejecución en la memoria de SQL Server, más concretamente en la zona de memoria llamada memoria caché del plan. Es posible analizar esta versión compilada de la consulta para comprender mejor las selecciones del optimizador de la consulta y reaccionar para permitir una ejecución más rápida de la consulta. Esto puede traducirse en una reescritura de la consulta, la adición de un índice, la actualización de las estadísticas...

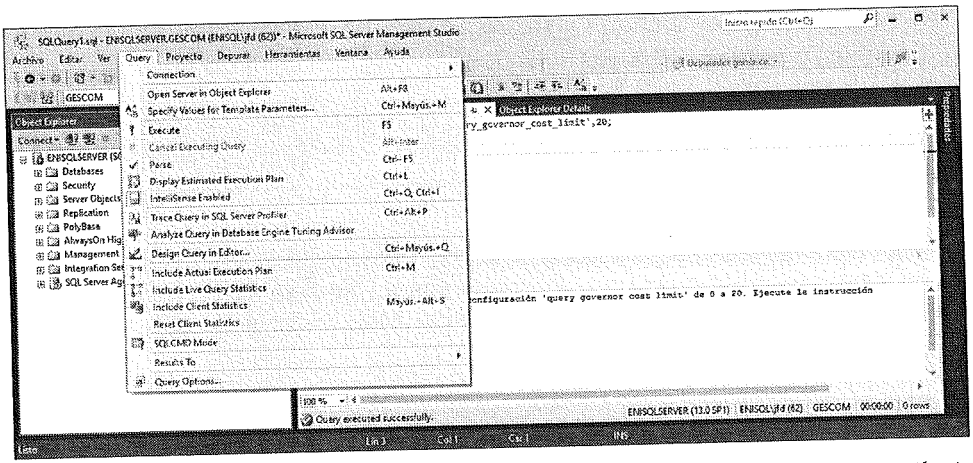
La optimización de las consultas no es el único punto que hay que tener en cuenta para resolver los problemas de rendimiento, pero sí es un aspecto importante. Efectivamente, centrarse en los problemas de memoria cuando la consulta está mal escrita puede ocultar momentáneamente los malos tiempos de respuesta, pero el problema aparecerá de nuevo cuando aumente el volumen de datos.

Observación

No es posible visualizar el plan de ejecución de un trigger o de un procedimiento almacenado.

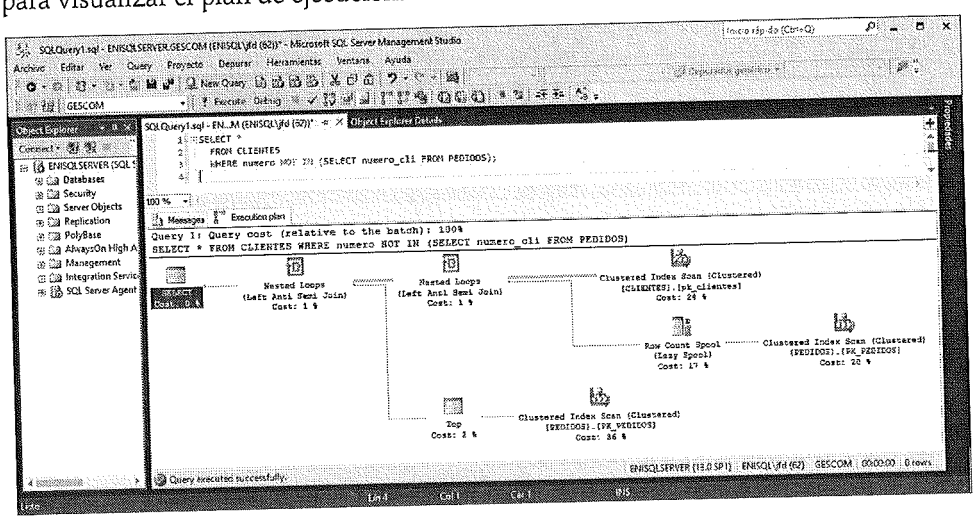
Para visualizar el plan de ejecución en SQL Server Management Studio, hay dos opciones:

- Mostrar el plan de ejecución estimado. El script Transact SQL no se ejecuta y el plan de ejecución mostrado procede del análisis de la consulta por el optimizador de la consulta.
- Incluir el plan de ejecución real. El script Transact SQL se ejecuta y se muestra el plan de ejecución utilizado.



La lectura del plan de ejecución solo es posible para los usuarios que tienen privilegios suficientes.

Tanto en un caso como en el otro, es necesario seleccionar la ficha **Execution Plan** para visualizar el plan de ejecución.



Para poder leer correctamente el plan de ejecución, es necesario saber lo siguiente:

- La lectura se hace de izquierda a derecha.
- El coste de cada consulta se expresa mediante un porcentaje en relación con el coste total del lote analizado.
- El icono presente en cada nodo representa el operador físico o lógico utilizado.
- Se muestra un plan de ejecución para cada consulta del lote Transact SQL analizado.

Si se pasa el cursor del ratón sobre los nodos, aparece información relativa a la operación realizada en forma de tooltips. Estos tooltips pueden contener la información siguiente:

- Operación física: muestra el operador físico utilizado.
- Operación lógica: presenta el operador lógico asociado al operador físico.
- Estimated Row Size: estimación del tamaño en bytes de la línea producida por el operador.
- Estimated I/O Cost: estimación de los costes de lectura/escritura.
- Estimated CPU Cost: estimación del coste de procesador para esta operación.
- Estimated Operator Cost: estimación del coste para el optimizador de consultas.
- Estimated Subtree Cost: suma de los costes de las operaciones anteriores y de esta operación.
- Estimated Number of Rows: disponible solo cuando la consulta se ejecuta. Este elemento permite saber el número de líneas tratadas.

Observación

Es posible visualizar el plan de ejecución de manera textual con SET SHOWPLAN_ TEXT o bien en forma de archivo XML con SET SHOWPLAN_XML.

Algunas consultas más complejas van a tener tiempos de respuesta más largos, ya que el volumen de datos manipulados es más importante y los tratamientos son numerosos. Para optimizar estas consultas, es necesario explorar las vías siguientes:

- Añadir memoria para asegurarse de que todos los datos se encuentran en memoria en el momento de la ejecución de la consulta. El objetivo es tener el máximo de lectura lógica con muy poca lectura física.
- Si el servidor dispone de varios procesadores, el sistema debe autorizar a SQL Server a utilizarlos para obtener un tratamiento paralelo de la consulta.

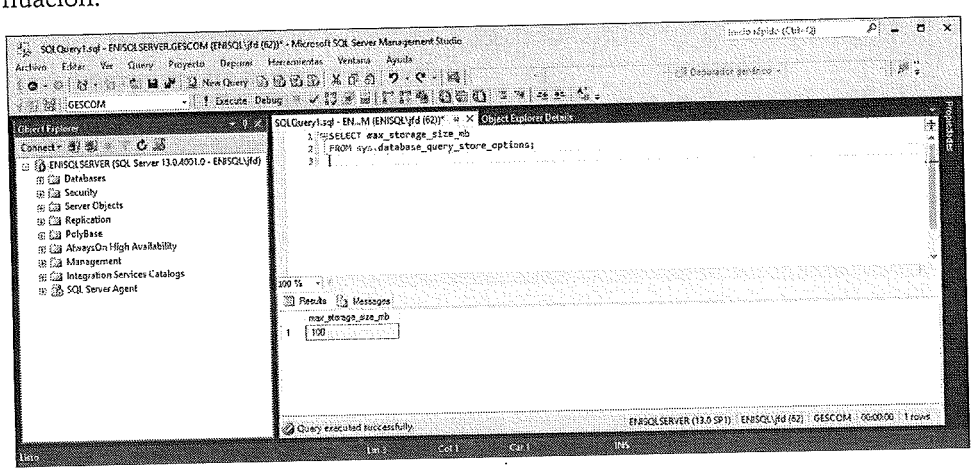
- Escribir la consulta de una nueva manera limitando al máximo la utilización de elementos que vuelven la ejecución más compleja, como los cursores, la ejecución de una consulta parametrizada en un bucle, la utilización de varios alias para una misma columna o la utilización de funciones que no son absolutamente necesarias.
- Modificar el valor de la opción de configuración **query governor** para limitar el tiempo de ejecución asignado a cada consulta.

10. El almacén de consultas

Los planes de ejecución evolucionan en función de varios criterios, y para una misma consulta pueden existir varios planes de ejecución posibles. Una de las dificultades es seleccionar el plan que mejor se adapte.

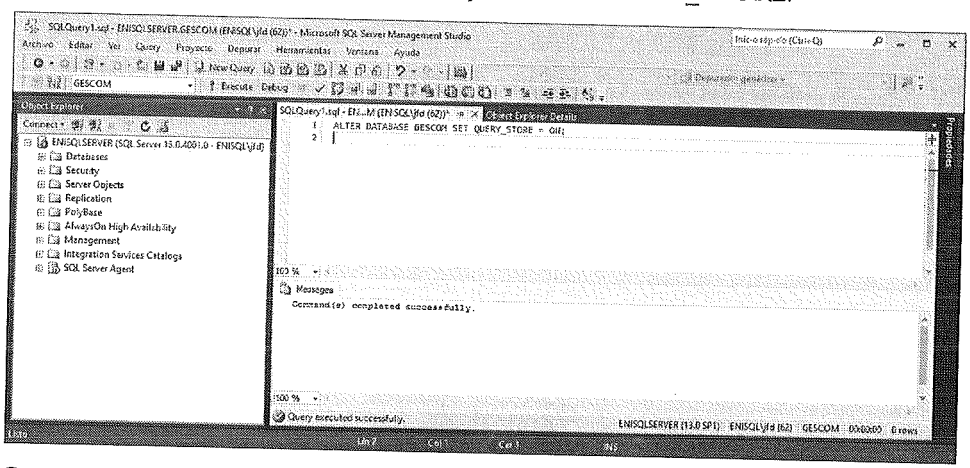
SQL Server propone el almacén de consultas. Como su nombre indica, en el seno de cada base de datos, SQL pretende almacenar las consultas y los planes de ejecución asociados.

Para no sobrecargar la base de datos, esta funcionalidad está desactivada por defecto, y el espacio máximo que puede utilizar el almacén de consultas está definido por el parámetro MAX_STORAGE_SIZE_MB. Es posible conocer el valor de este parámetro consultando la vista **sys.database_query_store_options** como se puede ver a continuación.



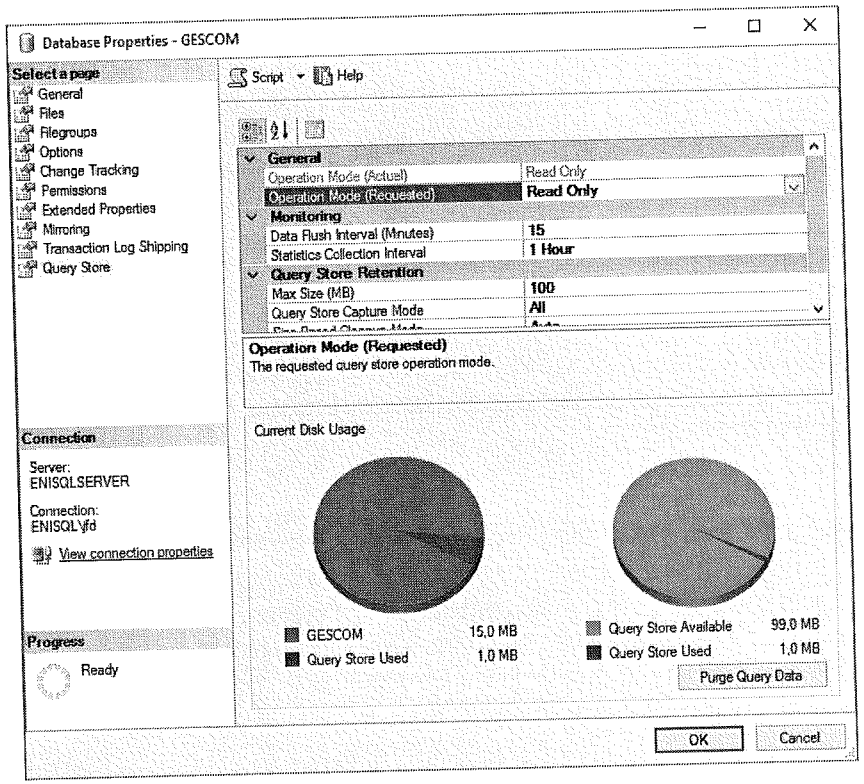
El ejemplo anterior muestra que el almacén de consultas de la base de datos GESCOM dispone de un espacio de 100MB (que es el tamaño por defecto). Este espacio de almacenamiento se realiza en el grupo de archivos Primary.

Para activar el almacén de consultas debemos ejecutar la instrucción ALTER DATABASE para configurar a ON el valor del parámetro QUERY_STORE.



Gracias a este mismo comando ALTER DATABASE, pero con la opción SET QUERY_STORE CLEAR, podremos liberar el espacio ocupado por el almacén de consultas.

También podemos ir a la sección **Query Store** de las **Database Properties** y allí modificar el valor **Operation Mode**.



El almacén de consultas proporciona cuatro informes predefinidos para responder a las peticiones más habituales. Estos cuatro informes son:

- Consultas anteriores: este informe muestra las veinticinco consultas más afectadas por las pérdidas de rendimiento en la última hora.
- Consumo global de recursos: este informe permite presentar, sobre la base de datos guardada en el último mes, la duración media de ejecución de las consultas, el número de ejecuciones de consultas, el consumo de tiempo de procesador y por último el número de lecturas lógicas.
- Principales consultas consumidoras de recursos: este informe permite identificar las veinticinco consultas que más consumen en el transcurso de la última hora.
- Seguimiento de consultas: este informe permite mostrar los detalles de la ejecución de una consulta en particular. Este análisis se utiliza habitualmente después de identificar una consulta con la ayuda de los informes presentados anteriormente.

11. Plan de mantenimiento

Para las operaciones clásicas del administrador, tales como las copias de seguridad regulares de la base de datos y de los diarios de transacciones o las operaciones de mantenimiento de los índices, es posible definir planes de mantenimiento. La ejecución de estas diferentes tareas se planifica durante la definición del plan de mantenimiento.

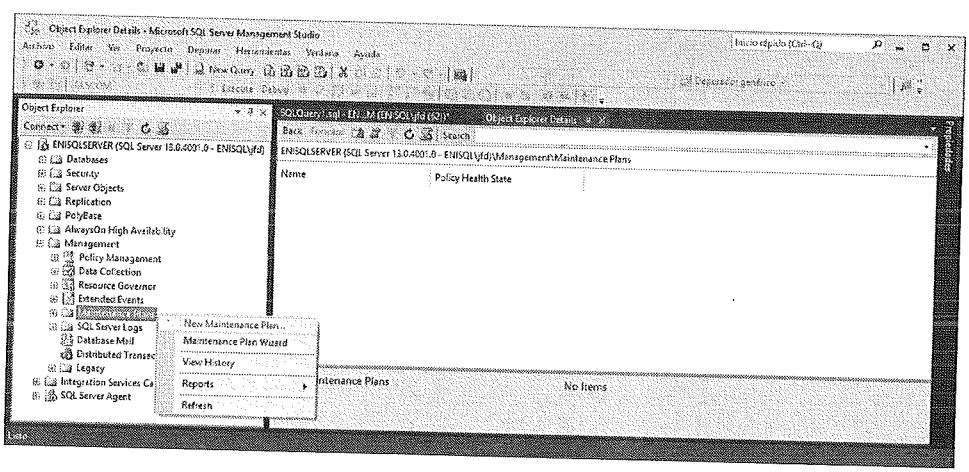
Los planes de mantenimiento se pueden definir con un asistente, pero la creación manual de un plan de mantenimiento ofrece más opciones de configuración. Por lo tanto, esta última opción es más interesante.

Los planes de mantenimiento se definen como paquete SIS y es SQL Server Agent quien se encarga de ejecutar el trabajo que lanza este paquete. Los planes de mantenimiento solo se pueden definir en las bases de datos que ofrecen un nivel de compatibilidad superior o igual a 80.

Observación

La herramienta de línea de comandos sqlmaint se mantiene por razones de compatibilidad con versiones anteriores, pero se recomienda no usarla.

La definición de un nuevo plan de mantenimiento se puede establecer fácilmente llamando al asistente de definición de un nuevo plan. Este asistente se puede ejecutar desde el menú contextual asociado al nodo **Management - Maintenance Plans**, desde el explorador de objetos.



El resultado de la ejecución de las diferentes tareas relativas al plan de mantenimiento se puede escribir en un archivo en formato texto o directamente en la base msdb.

En este último caso, las tablas que almacenan el detalle del plan de mantenimiento son `sysmaintplan_log` y `sysmaintplan_logdetail`. La información se puede leer fácilmente consultando el histórico desde SQL Server Management Studio.

Las operaciones principales que se incluyen dentro de un plan de mantenimiento son las siguientes:

- Reconstrucción de los índices, indicando si es necesaria una nueva tasa de completitud para las páginas hoja y no hoja, para conservar un índice correctamente balanceado el mayor tiempo posible (es decir, hasta la próxima reconstrucción).
- Limitar el tamaño de los archivos de datos, eliminando de estos la parte que no se usa. Este método es mejor que la reducción automática de los archivos de datos.
- Ejecutar la herramienta DBCC.
- Hacer copias de seguridad que cumplan el plan de copias de seguridad que se ha establecido.
- Por último, para obtener una mejor legibilidad de las diferentes tareas planificadas a nivel de SQL Server Agent, es posible solicitar la ejecución de algunos trabajos.

12. El asistente de configuración del motor de base de datos

El asistente de configuración del motor de base de datos tiene como objetivo ofrecer la estructura física óptima enfrentando la organización actual con una carga de trabajo.

Observación

Es posible solicitar la ejecución de esta herramienta en línea de comandos con dta.exe.

La carga de trabajo corresponde a una traza capturada con anterioridad por el generador de perfil de SQL Server o a un script Transact SQL.

A partir de esta carga de trabajo, la herramienta va a ofrecer una reorganización del esquema lógico añadiendo índices adicionales, particionando algunas tablas o bien proponiendo además la creación de vistas indizadas. Las propuestas hechas por el asistente tienen como objetivo reducir el coste estimado por el optimizador de consultas para la carga de trabajo analizada.

Durante el análisis de una carga de trabajo, es necesario configurar tres elementos:

- Nombrar de manera única el análisis.
- Hacer referencia a un archivo o una tabla que contenga una carga de trabajo.
- Seleccionar la base o las bases que van a ser utilizadas por este análisis.

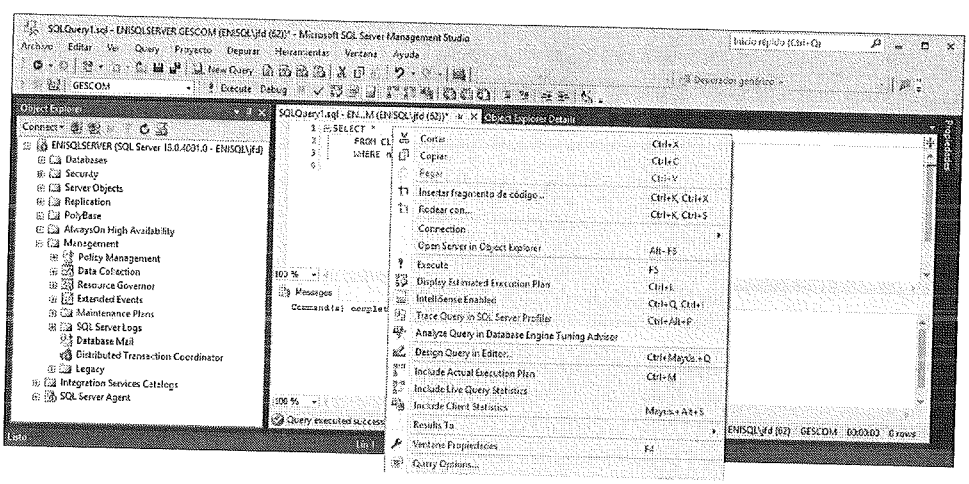
12.1 Inicialización del asistente de configuración

Toda la información relativa a la configuración va a ser almacenada en la base **msdb**. Para ello, durante la primera ejecución del asistente de configuración, se define una estructura en msdb. Solo un usuario que disponga de los derechos de administrador del sistema (**sysadmin**) puede realizar esta tarea. Como consecuencia, el usuario deberá ser, al menos, miembro del rol db_owner para analizar y realizar las propuestas en una base de datos.

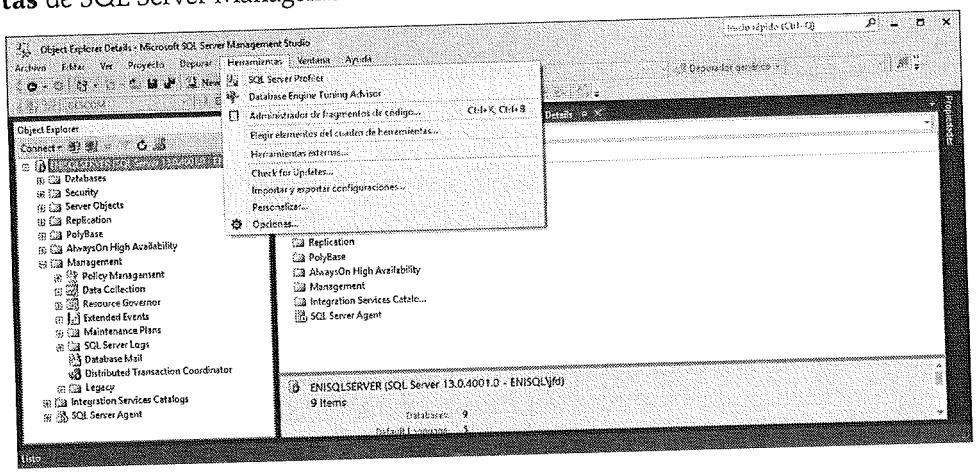
El asistente de configuración utiliza el procedimiento almacenado **xp_msver**.

El asistente se puede iniciar directamente desde el menú **Microsoft SQL Server - Herramientas de rendimiento** o bien desde SQL Server Management mediante **Herramientas - Database Engine Tuning Advisor**.

Pero la manera más simple de lanzar la herramienta es, sin duda, seleccionar una consulta o bien un lote de instrucciones en el editor de consultas de SQL Server Management Studio y escoger **Analyse Query in Database Engine Tuning Advisor** en el menú contextual.



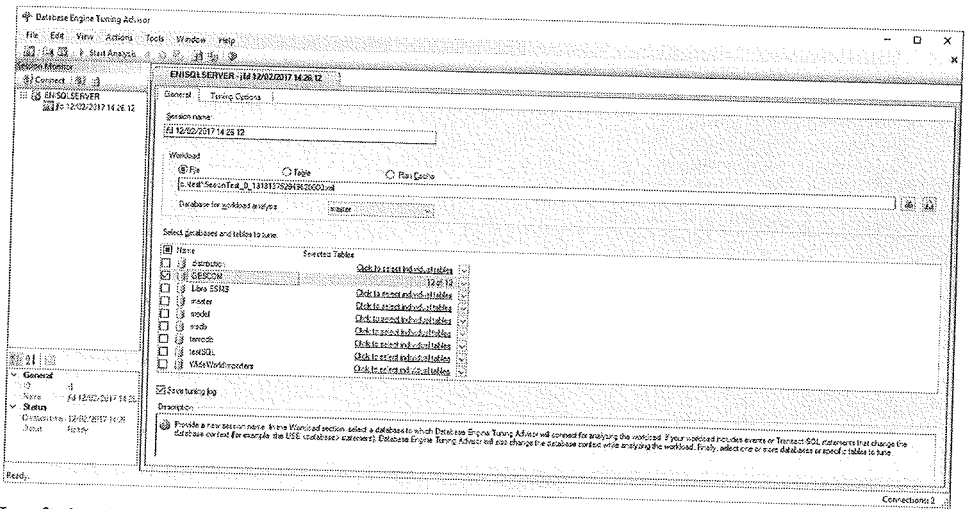
También es posible solicitar la ejecución de este asistente desde el menú **Herramientas** de SQL Server Management Studio.



12.2 Análisis de una carga de trabajo

Cuando se ejecuta la herramienta, hay que identificar la sesión especificando su nombre, el origen de la carga de trabajo y sobre qué base de datos se debe realizar el análisis.

Aquí es donde se pueden reutilizar los archivos generados por el almacenamiento de los eventos extendidos. Estos archivos corresponden a una carga de trabajo sometida a SQL Server. Analizando esta carga de trabajo podemos detectar si son posibles mejoras sobre la base de datos.



La ficha **Tuning Options** permite hacer una selección más exacta del tipo de estructura que va a formar parte del análisis y el margen de maniobra que se deja a la herramienta para construir su respuesta. Para iniciar el análisis, es necesario pulsar el botón **Start Analysis** de la barra de herramientas. Se muestra una nueva ficha con el objetivo de seguir la progresión del análisis.

Al final del análisis, se añaden las fichas **Recomendations** e **Reports**. Es posible aplicar las recomendaciones de manera inmediata o planificarlas por medio del menú **Actions - Apply Recomendations**.

Desde la ficha **Reports**, es posible visualizar algunos informes para saber si son adecuadas las elecciones actuales y las recomendadas.

13. Los triggers DDL

SQL Server ofrece triggers de tipo DDL. Estos triggers de base de datos funcionan como los triggers asociados a una acción insert, update o delete que puede producirse sobre una tabla dada. La ejecución del trigger DDL está asociada a la ejecución de una instrucción CREATE, ALTER, DROP, GRANT, REVOKE, DENY y UPDATE STATISTICS.

Los triggers DDL se ejecutan después de la instrucción DDL a la que están asociados y DDL permiten seguir las modificaciones realizadas sobre el esquema o bien prohibir algunas modificaciones que puedan realizarse sobre el esquema.

Para prohibir una instrucción DDL en el trigger asociado, es necesario anular la transacción con la instrucción ROLLBACK.

Al definir un trigger DDL, es necesario indicar la instrucción DDL que permite su ejecución, así como su ámbito, es decir, el lugar en que va a estar activo. El ámbito puede corresponder a una base de datos concreta o a la totalidad de la instancia.

Para permitir seguir lo mejor posible las diferentes ejecuciones de las instrucciones DDL, es posible utilizar la función EVENTDATA en los triggers DDL. Esta función permite capturar la información relativa a la ejecución del trigger y devuelve un flujo XML. Es posible extraer la información de este flujo XML utilizando una consulta XQuery.

Observación

No es posible definir triggers DDL de tipo instead of.

La información relativa a la ejecución actual del trigger DDL está disponible a través de EVENTDATA. No existe ninguna tabla **inserted** o **deleted**.

Los triggers de tipo DDL pueden definirse en una base de datos o bien en el servidor. En el caso de los triggers definidos sobre una base de datos, la definición del trigger se almacena en esta misma base. La definición de los triggers de nivel servidor se guarda en la base Master. En todos los casos, es posible obtener información relativa a estos triggers consultando la vista **sys.server_triggers**.

Sintaxis

```
CREATE TRIGGER nombreTrigger
ON { ALL SERVER | DATABASE }
[ WITH ENCRYPTION, [EXECUTE AS cláusula] ]
{ FOR | AFTER } {tipoInstrucción|grupoInstrucción }[ ,...]
AS instrucciones;
```

`nombreTrigger`

Corresponde al nombre del trigger. Cada trigger se identifica perfectamente por su nombre.

`ALL SERVER`

Con esta opción, el trigger está activo sobre toda la instancia de SQL Server en la que está definido.

`DATABASE`

Con esta opción, el trigger se dispara únicamente en la base sobre la que está definido.

`tipoInstrucción`

Tipo de la instrucción que va a disparar la ejecución del trigger. El trigger se ejecuta siempre después de la instrucción que lo dispara. El tipo de instrucción corresponde generalmente a la instrucción DDL de origen. Por ejemplo, para asociar un trigger a la instrucción que permite crear una tabla (CREATE TABLE), el tipo de la instrucción es CREATE_TABLE.

`grupoInstrucción`

Grupo de instrucciones que provocan la ejecución del trigger.

`instrucciones`

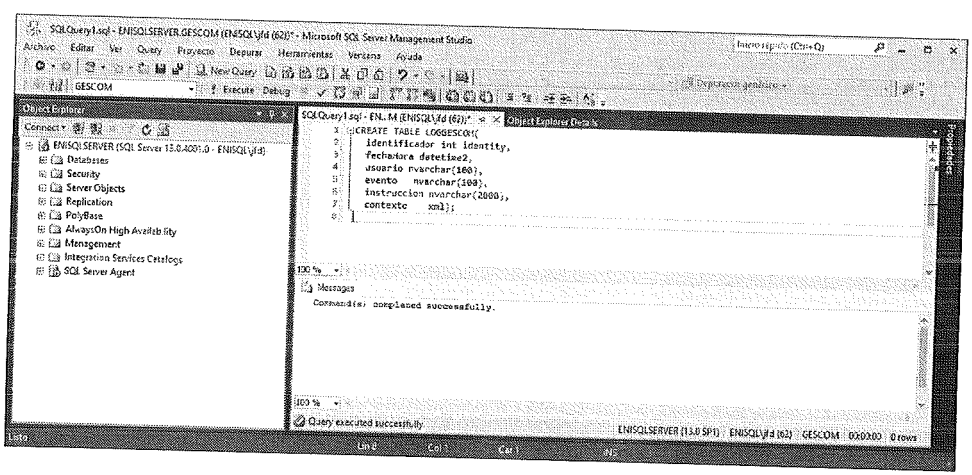
Representa el conjunto de instrucciones Transact SQL que van a ser ejecutadas. Es la lógica de aplicación del trigger.

Generalmente, la definición del ámbito del trigger como de base de datos o de instancia va a decidir los elementos o los grupos de elementos que pueden provocar la ejecución del trigger. La tabla con la descripción de los grupos de instrucciones se presenta en el anexo.

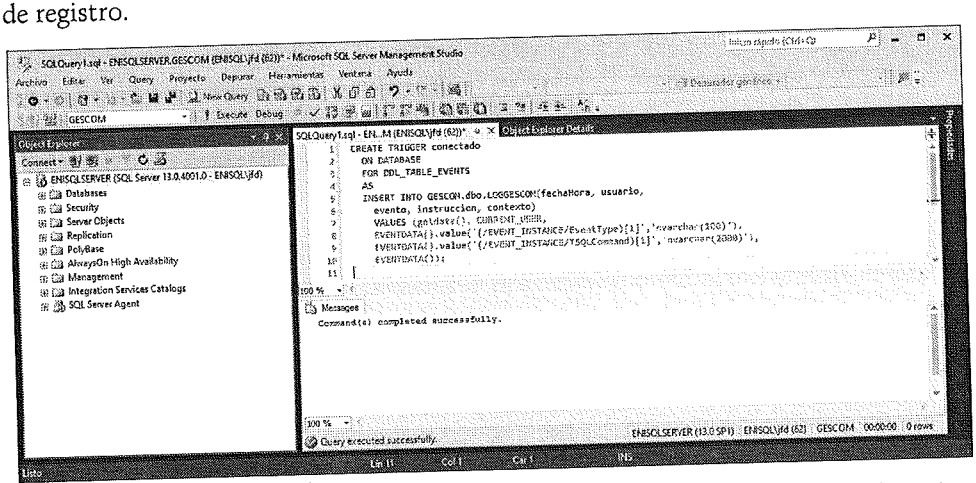
Ejemplo

Las siguientes pantallas ilustran la aplicación de un trigger DDL que permite monitorizar las acciones de modificación de estructura (create, alter y drop) que se realizan en la base de datos.

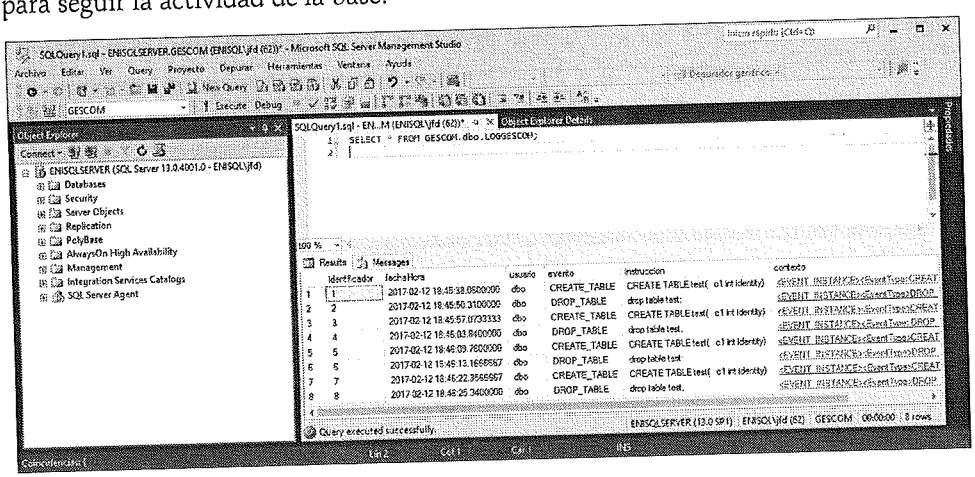
El primer paso consiste en definir la tabla que va a contener toda la información de seguimiento de los eventos. Esta tabla corresponde al registro de las acciones DDL ejecutadas.



El segundo paso consiste en crear un trigger DDL que estará activo a nivel de la instancia y que va a ser disparado por las operaciones de creación, modificación o eliminación de tabla en la base de datos GESCOM. La información se consigna en la tabla de registro.



Después de algún tiempo de funcionamiento, es posible consultar la tabla de registro para seguir la actividad de la base.



14. Los triggers de conexión

Los triggers de conexión permiten ejecutar una o varias instrucciones Transact SQL durante la conexión de un usuario al servidor. Este tipo de trigger puede considerarse un trigger DDL especial. Se ejecuta después de la validación de la autenticación del usuario en el servidor y antes de que el servidor permita el acceso al usuario. Por lo tanto, la conexión de usuario se establece y se abre cuando se ejecuta el trigger. Dado que la ejecución tiene lugar antes de que el usuario pueda operar directamente sobre la consola, los mensajes que pueden producirse (error, impresión) se redirigen directamente al registro de errores de SQL Server.

Los triggers de conexión pueden utilizarse para seguir o limitar las conexiones de usuario. Por ejemplo, a este nivel es posible limitar el número de sesiones abiertas simultáneamente por un mismo usuario. También es posible a este nivel registrar en una tabla la fecha y hora de establecimiento de la conexión. En todos los casos, la función **EVENTDATA** permite obtener toda la información relativa a la conexión que acaba de establecerse.

Es posible definir varios triggers de conexión. En este caso, el procedimiento almacenado **sp_settriggerorder** permite definir qué trigger se ejecutará primero y cuál el último. No es posible fijar el orden de ejecución de los otros triggers.

SQL Server establece una transacción implícita antes de ejecutar el primer trigger de conexión. Esta transacción se valida después de la ejecución del último trigger. Si durante la ejecución de los triggers de conexión la transacción se cierra o se produce un error con gravedad superior o igual a 20, entonces no es posible el establecimiento de la conexión del usuario.

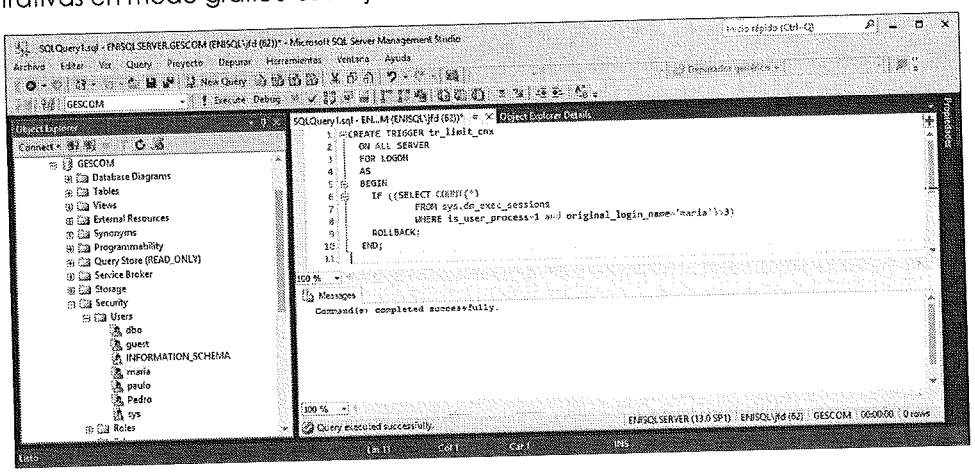
Sintaxis

```
CREATE TRIGGER nombreTrigger
ON ALL SERVER [WITH EXECUTE AS contextoEjecución]
FOR LOGON
AS
BEGIN
...
END;
```

Ejemplo

El trigger siguiente permite limitar a 3 el número de conexiones abiertas por un usuario llamado Ángel.

Hay que tener cuidado de no definir este trigger para que se ejecute con todos los usuarios, incluidos los administradores. De hecho, para estos últimos, el límite de 3 conexiones es demasiado bajo, ya que la realización de algunas tareas administrativas en modo gráfico es mejor con un número mayor de conexiones.



Los triggers pueden eliminarse con la ayuda de la instrucción:

```
DROP TRIGGER nombreTrigger ON {DATABASE | ALL SERVER}[;]
```

15. PowerShell

Este Shell se introdujo en Windows Server 2008 y permite definir potentes scripts de administración. Esta versión de PowerShell se enriquece de las herramientas específicas de cada aplicación servidor que edita Microsoft.

Este shell permite ejecutar las instrucciones de manera directa o en forma de script.

SQL Server no es una excepción a la regla y aporta su PowerShell, que integra todas las bibliotecas necesarias para definir scripts de administración eficaces, fiables y condensados. Las aportaciones de SQL Server al PowerShell son:

- La integración de un proveedor de acceso para poder navegar en la arborescencia del servidor de la misma manera que se hace en un sistema de archivos, es decir, principalmente con las instrucciones `cd` y `dir`. Este proveedor permite conectarse a instancias de SQL Server 2016, 2014, 2012 o 2008.

- La adición de applets de comando con el objetivo de poder integrar y ejecutar una acción SQL, fundamentalmente los scripts Transact SQL. Todos estos applets se apoyan en la tecnología SMO. De esta manera, PowerShell solo considera las instrucciones que toman en cuenta SMO. Estos applets de comando también se denominan cmdlet. Como para las otras instrucciones de PowerShell, no son sensibles a mayúsculas/minúsculas. Sin embargo, respetar las reglas de nomenclatura referentes a mayúsculas/minúsculas permite facilitar la lectura de las instrucciones.

La consola PowerShell se inicia por medio de la herramienta **sqlps**. Es posible ejecutar directamente los scripts PowerShell. El applet **Get-Help** permite, como siempre, encontrar toda la información relativa a los applets de comandos. Para SQL Server, es posible establecer inicialmente dos applets **Invoke-Sqlcmd** y **Invoke-PolicyEvaluation**.

Por ejemplo, para obtener la ayuda del proveedor SQL Server, es necesario ejecutar la instrucción Get-Help SQLServer. De manera más general, para obtener la ayuda del conjunto de proveedores se usa `Get-Help -category provider`.

Sobre los applets de comando, es posible utilizar las opciones `-Full`, `-Parameters *` y `-Examples` para visualizar diferentes niveles de ayuda.

Ejemplo

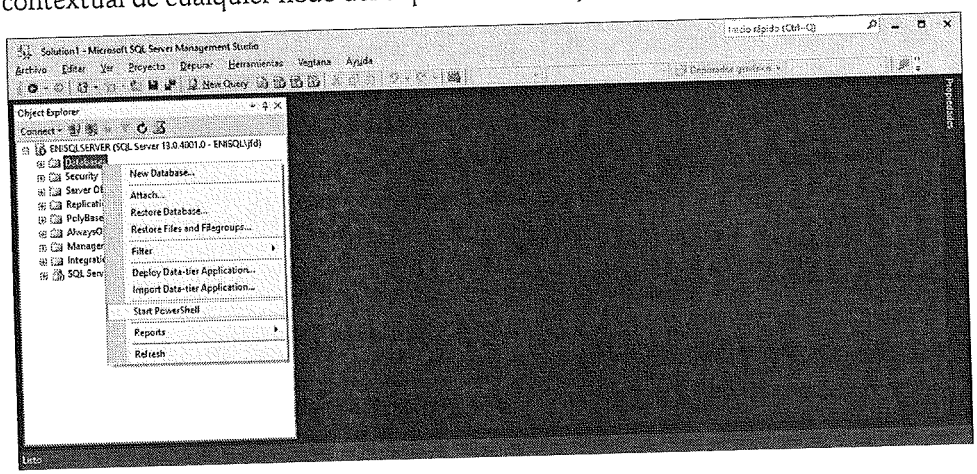
```
Seleccionar SQL Server Powershell
PS SQLSERVER:\SQL\ENISQLSERVER\DEFAULT> Get-Help Invoke-Sqlcmd

NOMBRE
    Invoke-Sqlcmd

SINOPSIS
    Runs a script containing statements supported by the SQL Server SQLCMD utility.


SINTAXIS
    Invoke-Sqlcmd [-ServerInstance <PSObject>] [-Database <String>] [-EncryptConnection <SwitchParameter>] [-Username
    <String>] [-Password <String>] [[-Query] <String>] [-QueryTimeout <Int32>] [-ConnectionTimeout <Int32>]
    [-ErrorLevel <Int32>] [-SeverityLevel <Int32>] [-MaxCharLength <Int32>] [-MaxBinaryLength <Int32>] [-AbortOnError
    <SwitchParameter>] [-DedicatedAdministratorConnection <SwitchParameter>] [-DisableVariables <SwitchParameter>]
    [-DisableCommands <SwitchParameter>] [-HostName <String>] [-NewPassword <String>] [-Variable <String[]>]
    [-InputFile <String>] [-OutputSqlErrors <Boolean>] [-IncludeSqlUserErrors <SwitchParameter>]
    [-SuppressProviderContextWarning <SwitchParameter>] [-IgnoreProviderContext <SwitchParameter>] [-OutputAs
    <OutputType>] [-InformationAction <ActionPreference>] [-InformationVariable <String>] [<CommonParameters>]

    Invoke-Sqlcmd [[-Query] <String>] [-QueryTimeout <Int32>] [-ErrorLevel <Int32>] [-SeverityLevel <Int32>]
    [-MaxCharLength <Int32>] [-MaxBinaryLength <Int32>] [-AbortOnError <SwitchParameter>] [-DisableVariables
    <SwitchParameter>] [-DisableCommands <SwitchParameter>] [-Variable <String[]>] [-InputFile <String>]
    [-OutputSqlErrors <Boolean>] [-IncludeSqlUserErrors <SwitchParameter>] [-OutputAs <OutputType>] -ConnectionString
    <String> [-InformationAction <ActionPreference>] [-InformationVariable <String>] [<CommonParameters>]

DESCRIPCIÓN
    The Invoke-Sqlcmd cmdlet runs a script containing the languages and commands supported by the SQL Server SQLCMD
    utility. The commands supported are Transact-SQL statements and the subset of the XQuery syntax that is supported
    by the database engine. This cmdlet also accepts many of the commands supported natively by SQLCMD, such as GO and
    QUIT. This cmdlet also accepts the SQLCMD scripting variables, such as SQLCMDUSER. By default, this cmdlet does
    not set SQLCMD scripting variables.

    This cmdlet does not support the use of commands that are primarily related to interactive script editing. The
    commands not supported include :!!, :connect, :error, :out, :ed, :list, :listvar, :reset, :perftrace, and
    :serverlist.
```

Observación

Cuando arranca sqlps, se muestra un mensaje referente a la versión de SQL Server. Estas tres líneas de mensajes se pueden eliminar ejecutando `sqlsp -nologo`.

La herramienta **sqlps** ejecuta la consola PowerShell en modo restringido por defecto, lo que impide la ejecución de scripts registrados en forma de archivos. Es posible modificar este nivel de ejecución llamando al applet de comando **Set-ExecutionPolicy**.

Por último, es posible iniciar la consola PowerShell directamente desde SQL Server Management Studio seleccionando la opción **Start PowerShell,** desde el menú contextual de cualquier nodo del explorador de objetos.



15.1 El proveedor PowerShell SQL Server

Con el proveedor PowerShell SQL Server, es posible navegar por la jerarquía del servidor con tanta facilidad como en un sistema de archivos.

Para permitir esto, el proveedor PowerShell SQL Server ofrece el lector SQLSERVER:. Este lector es comparable al lector C: que se encuentra habitualmente en el sistema de archivos.

El lector SQLSERVER está compuesto de los cuatro directorios siguientes:

SQL

Representa los objetos de base de datos. La navegación en estos objetos será siempre en la forma SQLSERVER:SQL\nombreServidor\nombreInstancia. Por ejemplo, para trabajar sobre la base GESCOM presente en la instancia por defecto del puesto IVANMDW81\LIBRO\SQLSERVER2012:
SQLSERVER:SQL\IVANMDW81\LIBRO\SQLSERVER2012\DEFAULT\Databases\GESCOM

SQLPolicy

Para gestionar la administración por reglas.

SQLRegistration

Para la inscripción de los servidores y la gestión de los grupos.

DataCollection

Para representar los objetos del recolector de datos y navegar en este árbol se van a utilizar los applets de comandos PowerShell o sus alias. Los principales applets de comando son:

Get-Location: visualizar el nodo actual.

Set-Location: situarse en el nodo cuya ruta ha sido configurada.

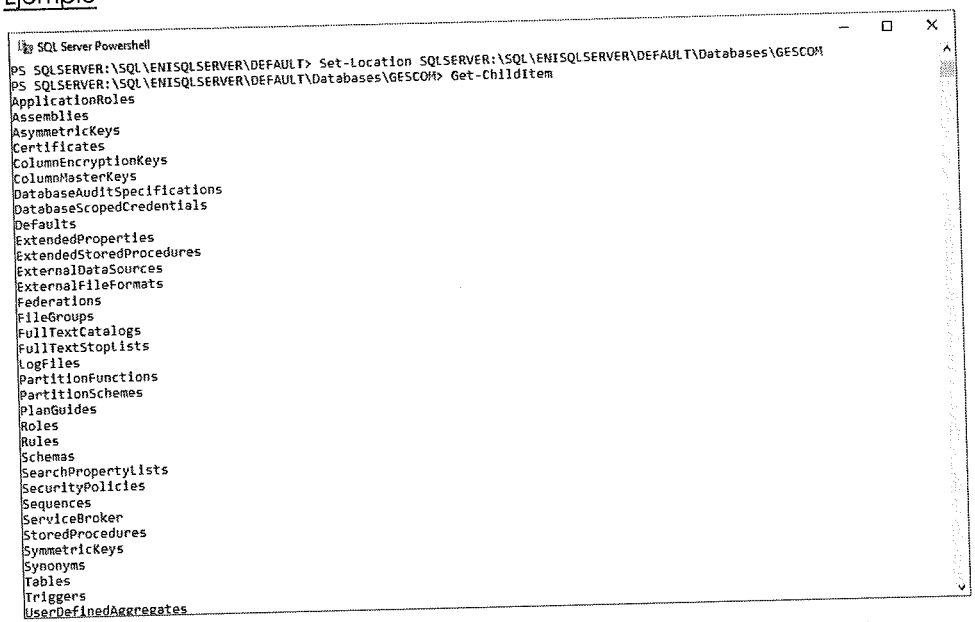
Get-ChildItem: recorre los hijos del nodo actual. Los objetos de sistema no son enumerados por defecto, excepto si la opción force está activada.

Get-Item: obtener la información relativa al nodo actual.

Rename-Item: renombrar el nodo actual.

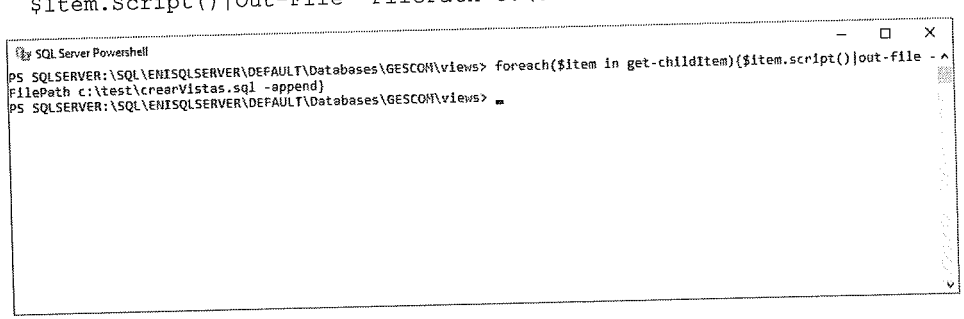
Remove-Item: eliminar el nodo actual.

Ejemplo



Es posible realizar numerosas operaciones de administración combinando los recursos de SQL Server con los elementos de sintaxis PowerShell. Por ejemplo, situándose sobre el nodo views, que representa la colección de las vistas, es posible recorrer cada vista para generar el script de creación.

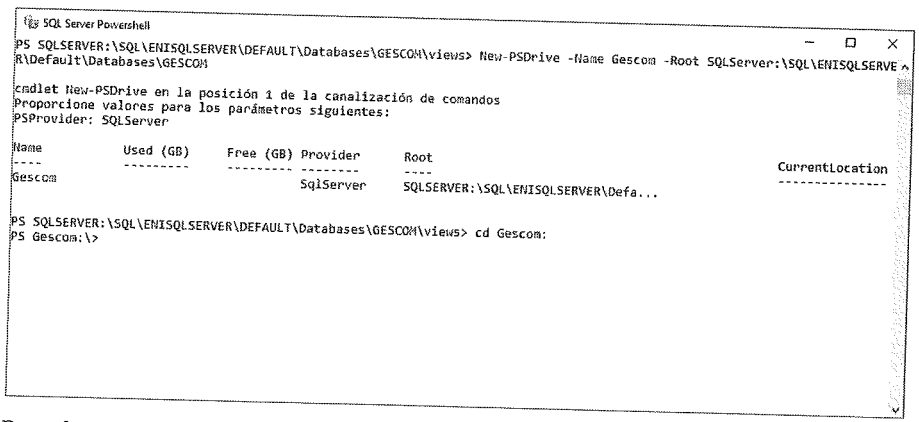
```
foreach ($Item in Get-ChildItem){
  $Item.Script()|Out-File -FilePath c:\CrearVistas.sql -append}
```



Las acciones en forma de scripts normalmente tienen la misma base de datos de destino. Para limitar la longitud y la complejidad del applet de comando **Set-Location**, es posible definir lectores que accedan directamente a una parte de la arborescencia del servidor. Por ejemplo, es posible definir el lector Gescom como acceso directo a la base GESCOM.

Ejemplo

```
New-PSDrive -Name GESCOM -Root SQLSERVER:\SQL\LIBRO\
DEFAULT\Databases\GESCOM
```



Para facilitar la introducción de las rutas, normalmente largas, es posible utilizar la tecla [Tab] para completar de manera automática el final de la ruta, seleccionando la opción correcta de la lista propuesta por PowerShell. En algunas bases de datos que tienen numerosos objetos, la lista propuesta puede ser larga. Es posible reducir la lista de propuestas configurando las variables siguientes:

`$SqlServerMaximumTabCompletion`, `$SqlServerMaximumChildItems` y `$SqlServerIncludeSystemObjects`

Todas estas conexiones a la instancia de SQL Server utilizan por defecto la seguridad integrada de Windows. Esto no es obligatorio y es posible trabajar con la seguridad de SQL Server para conectarse a una instancia de SQL Server. La función cuyo script se presenta a continuación permite definir un lector que se conecte por medio de una autenticación SQL Server (conexión u) a la instancia por defecto del ordenador local (LIBRO\DEFAULT). Se solicita la contraseña cuando se establece el lector.

```
Function lectorSQL{
  Param([string]$nombreLector,[string]$cnx="u",[string]
$ruta="SQLServer:\SQL\LIBRO\DEFAULT")
  $contraseña= read-host -AsecureString -Prompt "Contraseña"
  $credit=new-object System.Management.Automation.PSCredential
argumentlist $cnx,$contraseña
```

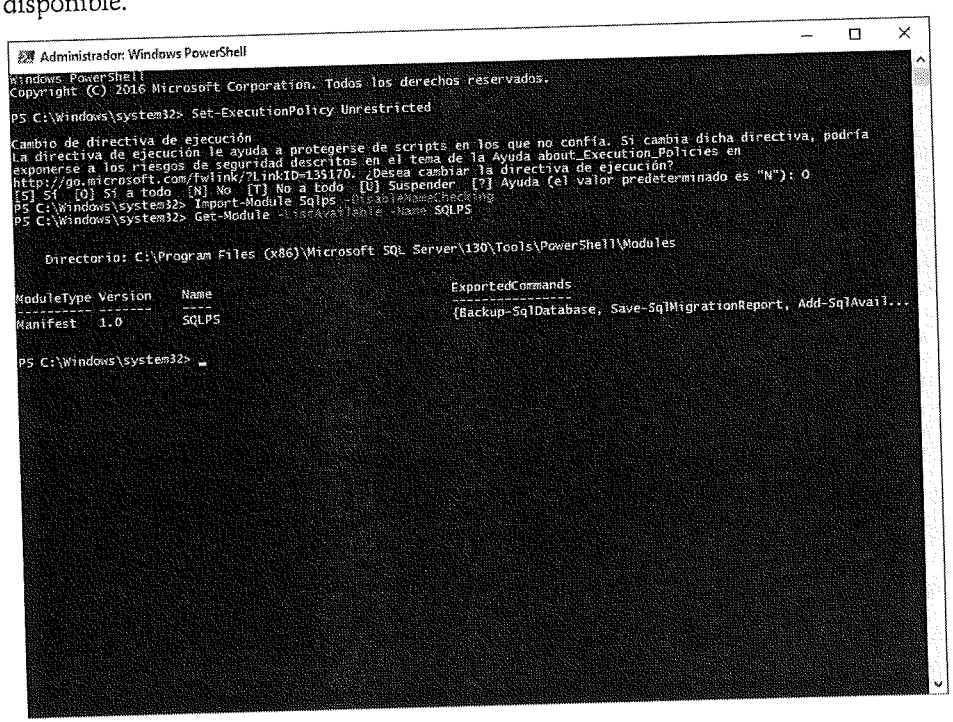
```
  New-PSDrive $nombreLector -PSProvider SqlServer -Root $ruta
-Credential $credit -Scope 1
}
lectorSQL SecurizadoSQL
```

15.2 Importar SQLPS

Una buena práctica para trabajar con PowerShell consiste en importar el módulo SQLPS en PowerShell. En efecto, resulta poco práctico arrancar PowerShell desde SQL Server Management Studio para realizar tareas administrativas.

Sin embargo, PowerShell no permitirá la ejecución de scripts. En efecto, PowerShell se ejecuta en modo restringido, lo que bloquea la ejecución de cualquier script. Afortunadamente, este modo se puede modificar con el comando Set-ExecutionPolicy. Sin embargo, deberemos aun así tener cuidado con el origen de los scripts para garantizar la integridad del sistema.

La importación se realiza fácilmente, mediante el comando Import-Module. Después podemos, mediante el comando Get-Module, verificar que el módulo está realmente disponible.



En el ejemplo anterior, Windows Powershell se ejecuta como administrador para permitir la ejecución del comando Set-ExecutionPolicy.

15.3 Los applets de comandos

Los applets de comandos PowerShell están formados por un verbo y un nombre. Esta construcción permite encontrar simple y rápidamente el applet correcto.

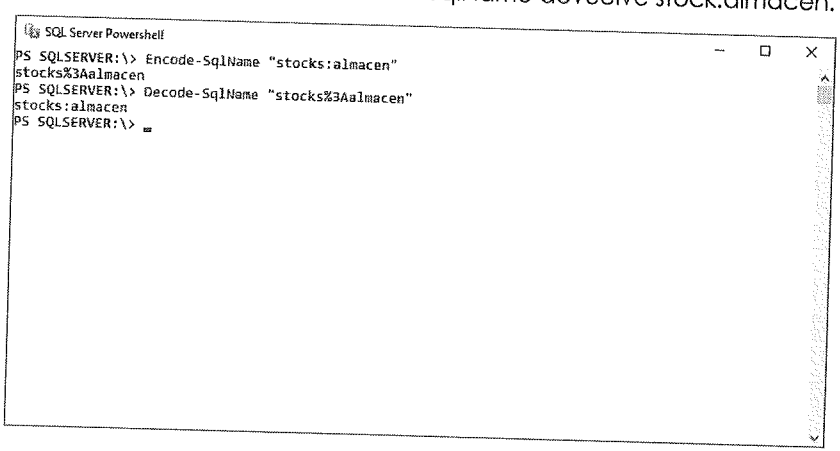
Como complemento a los applets de serie PowerShell, SQL Server aporta pocos applets de comandos específicos. Es posible destacar **Invoke-Sqlcmd**, **Invoke-PolicyEvaluation**, **Encode-SqlName**, **Decode-SqlName**.

15.3.1 Encode-SqlName, Decode-SqlName

Estos dos applets tienen como objetivo traducir los caracteres especiales que pueden estar presentes en algunos identificadores de objetos de SQL Server (aunque no se recomienda) y que no son tenidos en cuenta por PowerShell o bien no son correctamente interpretados.

Ejemplo

La cadena configurada para el applet Encode-SqlName devuelve stocks%3Almacen, y la de decode-SqlName devuelve stock:almacen.



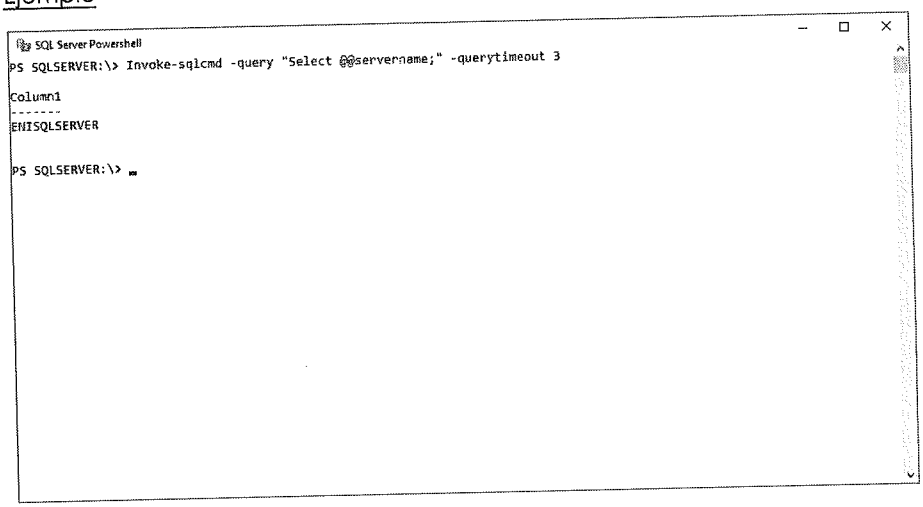
15.3.2 Invoke-PolicyEvaluation

Este applet de comando tiene como objetivo asegurarse de que el destino configurado es conforme con las reglas de administración definidas. Además de esta verificación, es posible hacer una llamada a este mismo applet para configurar las diferentes opciones de los objetos de destino de forma que respeten las reglas de administración definidas.

15.3.3 Invoke-Sqlcmd

Este applet de comando, que permite ejecutar scripts Transact SQL o bien XQuery, llama a sqlcmd para ejecutarlos. Este applet tiene sus propios parámetros, que no son los mismos que los de sqlcmd. Es posible fijar los parámetros de sqlcmd pasándolos en forma de cadena de caracteres al applet de comando.

Ejemplo



15.3.4 Convert-UrnToPath

Este applet de comando permite mejorar la comunicación entre la biblioteca SMO (*SQL Server Management Object*) y el proveedor de acceso de SQL Server. De hecho, el urn (*Uniform Resource Name*) que utiliza SMO no corresponde directamente a la estructura de pseudodirectorio que aplica el proveedor SQL. Asegurando la conversión, ese applet permite, por ejemplo, crear un objeto con ayuda de la biblioteca SMO y después, llamando a la propiedad URN, recuperar el urn de este objeto. Este urn se pasa como argumento al applet Convert-UrnToPath, que nos proporciona la ruta de acceso exacta a este objeto para manipularla en PowerShell, desde el proveedor SQL.

15.4 SMO

SQL Server ofrece pocos applets particulares de SMO porque ya existe la biblioteca SMO. Esta biblioteca está disponible para el framework .NET. Como PowerShell se basa en este framework, es posible usar esta biblioteca para trabajar con scripts de SQL Server. Los desarrolladores de aplicaciones conocen bien esta biblioteca. Se usa con mucha frecuencia para integrar tareas administrativa básicas, pero necesarias en la aplicación que se está programando. Añadiendo estas funcionalidades al programa, el usuario es autónomo desde su programa y no tiene que ejecutar SQL Server Management Studio.

En primer lugar, el script PowerShell que quiera trabajar con SMO tendrá que asegurarse de que se ha importado correctamente el espacio de nombres. Después tendrá que establecer la conexión con la instancia que desee.

El espacio de nombres que contiene SMO se llama Microsoft.SqlServer.SMO y se carga con la instrucción PowerShell:

[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.SMO")

SMO tiene un objeto de tipo **Microsoft.SqlServer.Management.Smo.Server** que representa a los servidores. Para acceder a una instancia, hay que crear un objeto de este tipo y pasarle el nombre de instancia como argumento.

Los objetos de tipo **Microsoft.SqlServer.Management.Smo.Server** tienen una propiedad llamada **Databases**. Esta propiedad es la lista de las bases de datos definidas en esta instancia.

Si se tienen los permisos necesarios, SMO permite la posibilidad de crear directamente una base de datos en una instancia de SQL Server. De este modo, se puede crear la base de datos de una aplicación sin recurrir a scripts externos, que no permiten crear una solución integrada.

Para crear esta base de datos, hay que usar un objeto de tipo **Microsoft.SqlServer.Management.Smo.Database**. Hay que definir dos propiedades:

- Parent, para indicar la instancia de SQL Server sobre la que se va a definir esta base de datos.
- Name, para definir el nombre de la base de datos.

El método **Create** permite solicitar la creación de la base de datos.

Ejemplo

A continuación se crea la base de datos, GescomSMO, en la instancia localhost.

```
$instancia='localhost';
$base='GescomSMO';

Set-psdebug -strict;
[System.Reflection.Assembly]::LoadWithPartialName
("Microsoft.SqlServer.Smo");
$servidor=new-object("Microsoft.SqlServer.Management.Smo.Server")
$instancia;
# Mostrar error si la instancia no es accesible
If ($server.Version -eq $null) {Throw "imposible encontrar
la instancia $instancia";}
$bdd=new-object ("Microsoft.SqlServer.Management.Smo.Database");
$dbb.Parent=$server;
$dbb.Name=$base;
$dbb.Create();
```

También es posible crear o modificar una tabla con los objetos de la biblioteca SMO. La creación de tablas es indisociable de la creación de columnas, ya que toda tabla está compuesta de, al menos, una columna. Las tablas y columnas se representan respectivamente por los objetos **Microsoft.SqlServer.Management.Smo.Table** y **Microsoft.SqlServer.Management.Smo.Column**.

Estas dos clases tienen las propiedades **Parent** y **Name**. La propiedad **Parent** permite definir el objeto de nivel superior que va a contener la tabla o columna. La propiedad **Name** representa el nombre de la tabla o columna.

En el caso de una tabla, también es necesario indicar el esquema sobre el que se va a definir la tabla. Para terminar, cuando la tabla esté completamente definida, hay que ejecutar el método **Create** para crear la tabla en la base de datos.

Antes de crearla, hay que añadir las columnas a la estructura de la tabla. El objeto de tipo Tabla tiene la colección **Columns**, que contiene el conjunto de columnas de la tabla. Se crean objetos columna y después se añaden a esta colección. Cuando se define una columna, aparte de las propiedades **Name** y **Parent**, hay que dar valor a la propiedad **DataType**. Para dar valor correctamente a esta propiedad, es necesario hacer referencia a la clase **Microsoft.SqlServer.Management.SmoSqlDataType**.

Ejemplo

En el siguiente extracto de código PoweShell, se crea un objeto de tipo Microsoft.SqlServer.Management.Smo.Table y se referencia con la variable $tabla. Las columnas de tipo Microsoft.SqlServer.Management.Smo.Column se definen de dos maneras diferentes. La diferencia tiene que ver con la forma en que se está acostumbrado a hacerlo, más que con el rendimiento. El objeto referenciado por la variable $columna1 utiliza el constructor predeterminado, es decir, sin argumento. Después hay que dar valor a las propiedades **Parent**, **Name** y **DataType**. Para el objeto referenciado por la variable $columna2, se utiliza otra versión del constructor de la clase **Column**, que permite pasar el nombre de la tabla, de la columna y el tipo de datos como argumentos.

```
$instancia='localhost';
$base='GescomSMO';

Set-psdebug -strict;
[System.Reflection.Assembly]::LoadWithPartialName
("Microsoft.SqlServer.Smo");
$servidor=new-object("Microsoft.SqlServer.Management.Smo.Server")
$instancia;
If ($server.Version -eq $null) {Throw "imposible encontrar la instancia
$instancia";}
$dbb=$servidor.Databases[$base];
$tabla= new-object("Microsoft.SqlServer.Management.Smo.Table");
$tabla.Parent=$bdd;
$tabla.Schema='dbo';
$tabla.Name='Personas';
$columna1=new-object("Microsoft.SqlServer.Management.Smo.Column")
$columna1.Parent=$tabla;
$columna1.Name='id'
$columna1.DataType= new-object Microsoft.SqlServer.Management.Smo.DataType(
[Microsoft.SqlServer.Management.Smo.SqlDataType]::Int);

$columna2=new-object(Microsoft.SqlServer.Management.Smo.Column($tabla,
"Nombre",[Microsoft.SqlServer.Management.Smo.DataType]::NVarChar(80));

$tabla.Columns.Add($columna1);
$tabla.Columns.Add($columna2)

$tabla.Create();
```

Para modificar una tabla, SMO ofrece la posibilidad de añadir columnas y restricciones de integridad, por ejemplo. Igual que el objeto Server tiene la lista de las bases de datos, el objeto Database tiene la propiedad Tables, que es la lista de las tablas que hay en la base de datos.

Las etapas de creación de una columna son las mismas que las de creación de una tabla. En cambio, ahora hay que llamar al método **Alter** de la tabla para aplicar las modificaciones en la base de datos.

La creación de restricciones de integridad se hace a través de un índice (para las restricciones de clave primaria y unicidad) o usando una clase específica, para las restricciones de validación y claves extranjeras.

Para las restricciones de clave primaria y unicidad es normal pasar por la creación del índice, ya que la creación de las restricciones implica la creación de un índice único. El índice que se crea es de tipo **Microsoft.SqlServer.Management.Smo.Index**. Para especificar que el índice sirve de soporte a una restricción de integridad, hay asignar a la propiedad **IndexKeyType** el valor **[Microsoft.SqlServer.Management.Smo.IndexKeyType]::DriPrimaryKey** para la clave primaria o **[Microsoft.SqlServer.Management.Smo.IndexKeyType]::DriUniqueKey** para la unicidad.

Las restricciones de validación se representan por la clase **Microsoft.SqlServer.Management.Smo.Check**, mientras que las restricciones de clave extranjera se representan por **Microsoft.SqlServer.Management.Smo.ForeignKey**.

Ejemplo

En el siguiente ejemplo se añade la columna apellidos a la tabla **Personas**, que se ha creado anteriormente. Estas modificaciones se aplican en la base de datos llamando al método Alter.

Se define la restricción de unicidad (**$unPersonas**) a través de un índice. Durante la construcción del objeto que representa al índice, se pasa como argumento la referencia a la tabla (**$tabla**). Se definen las propiedades específicas a los índices, como **FillFactor** y **Clustered**.

Se crean las columnas de tipo **Microsoft.SqlServer.Management.Smo.Index Column** y se añaden al índice antes de solicitar su creación.

```
$instancia='localhost';
$base='GescomSMO';
$esquema ='dbo';
$nombreTabla ='Personas';
Set-psdebug -strict;
[System.Reflection.Assembly]::LoadWithPartialName
("Microsoft.SqlServer.Smo");
$servidor=new-object("Microsoft.SqlServer.Management.Smo.Server")
$instancia;
If ($server.Version -eq $null) {Throw "imposible encontrar la instancia
$instancia";}
$dbb=$servidor.Databases[$base];
$tabla= $bdd.Tables[$nombreTabla];
$columna=new-object Microsoft.SqlServer.Management.Smo.Column
```

```
    ($tabla,
    "Apellido",
    [Microsoft.SqlServer.Management.Smo.DataType]::NVarChar(80));

$tabla.Columns.Add($columna);
$tabla.Alter();

$unPersonas = new-object Microsoft.SqlServer.Management.Smo.Index
($tabla, "UN_Personas");
$unPersonas.IndexKeyType =
[Microsoft.SqlServer.Management.Smo.IndexKeyType]::DriUniqueKey;
$unPersonas.IsClustered=$True
$unPersonas.FillFactor = 30
$columna1Index = new-object
Microsoft.SqlServer.Management.Smo.IndexedColumn
($unPersonas,"nombre");
$unPersonas.IndexedColumns.Add($columna1Index);
$columna2Index = new-object
Microsoft.SqlServer.Management.Smo.IndexedColumn
($unPersonas,"apellido");
$unPersonas.IndexedColumns.Add($columna2Index);
$unPersonas.Create();
```

16. La gestión de las reglas

SQL Server introduce la noción de gestión por reglas. Este tipo de gestión permite definir las reglas de administración y aplicarlas a una o varias instancias de SQL Server. Con la multiplicación de las instancias de SQL Server, incluso sobre un mismo puesto, se hace necesario tener una herramienta para simplificar la administración individual de las diferentes instancias. Las reglas también pueden servir para administrar las bases de datos de usuario.

Las reglas van a permitir definir un comportamiento común para todas las bases de datos, instancias y servidores de SQL Server haciendo obligatoria o no la activación de algunos servicios o forzando explícitamente las reglas de nomenclatura, por ejemplo. Entre otras cosas, las reglas van a permitir adoptar una estructura similar en bases de datos distintas, lo que permite una comprensión más fácil de las diferentes bases y facilita la administración.

La gestión por reglas se puede descomponer en tres componentes importantes:

- La definición y gestión de las reglas.
- El modo de ejecución de las reglas.
- La administración directa.

El término regla se puede precisar usando la noción de estrategia y de condición.

Una condición va a permitir asegurarse de que se respetan uno o varios criterios de configuración. La aplicación de las condiciones así definidas se confía a las estrategias. Cada estrategia permite verificar una única condición, pero una condición se puede referenciar por varias estrategias. Una estrategia permite definir los criterios de aplicación de la condición, como los destinos y el modo de aplicación de la estrategia.

La definición de las condiciones y estrategias queda registrada. Aunque es más fácil examinar las propiedades de las condiciones y estrategias desde SQL Server Management Studio, es posible consultar las vistas de sistema con el objetivo de trabajar en forma de scripts.

16.1 Las condiciones

Cada condición se compone de uno o varios criterios combinados entre sí por medio de los operadores lógicos Y y O, con el objetivo de poder evaluar si la condición se cumple.

Los diferentes criterios de evaluación se agrupan de manera lógica en forma de facetas. De esta manera, los diferentes parámetros de administración del servidor y de las bases se organizan de manera lógica y es más fácil encontrar el criterio o los criterios que se van a probar.

Cada faceta agrupa varias condiciones.

SQL Server ofrece facetas predefinidas. Cada faceta corresponde a un dominio bien definido de la administración de SQL Server.

16.2 Las estrategias

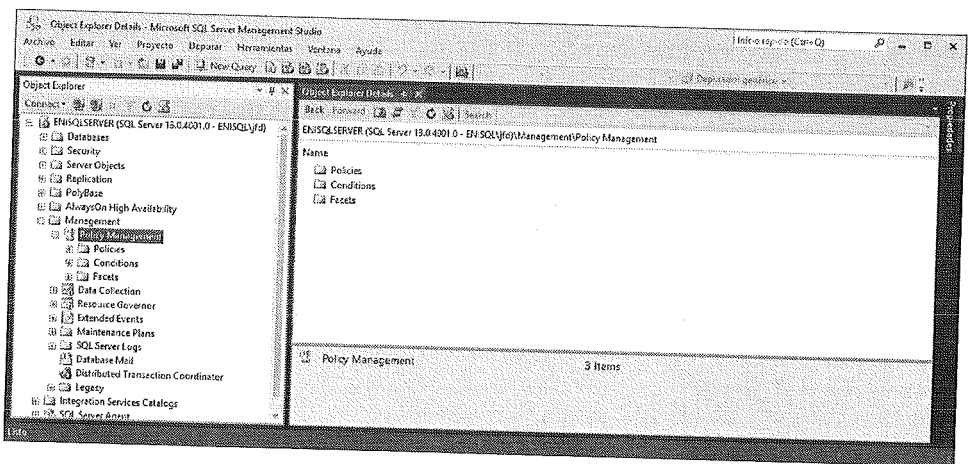
Una estrategia permite establecer una faceta (es decir, un conjunto de condiciones) y asegurarse de su cumplimiento. Cada estrategia está perfectamente identificada por su nombre. Una estrategia puede estar activa o no y tiene cuatro modos diferentes de aplicación:

- Bajo petición: la estrategia no está activa y es el administrador de la base de datos el que decide cuándo es necesario evaluarla y aplicarla.
- Según planificación.
- Sobre modificación - Registro.
- Sobre modificación - Evento.

Para identificar mejor las diferentes estrategias, es posible organizarlas en categorías e introducir una descripción para cada una de ellas. También es posible introducir la URL de una página web de ayuda o de explicación para permitir la explicación de la estrategia.

16.3 Puesta en marcha

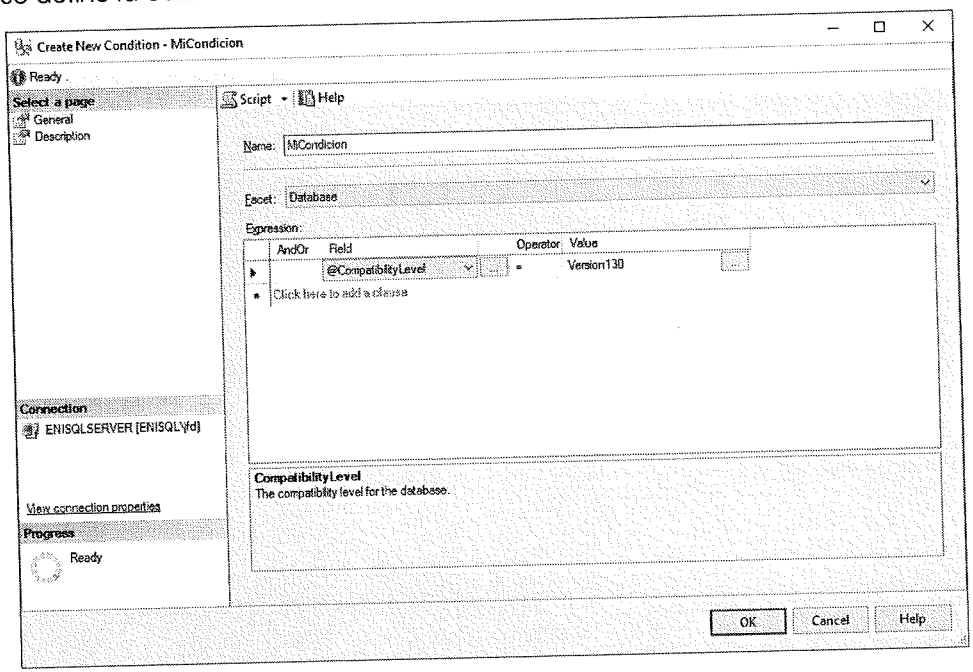
La puesta en marcha de las reglas la facilita SQL Server Management Studio. Se opera desde el nodo **Management - Policy Management** del explorador de objetos. A este nivel, aparecen tres carpetas que representan las facetas, las condiciones y las estrategias (directivas).



Mediante asistentes, es posible definir una o varias condiciones. Para crear una nueva condición, es necesario seleccionar la opción **New Condition** desde el menú contextual del nodo **Conditions**.

Ejemplo

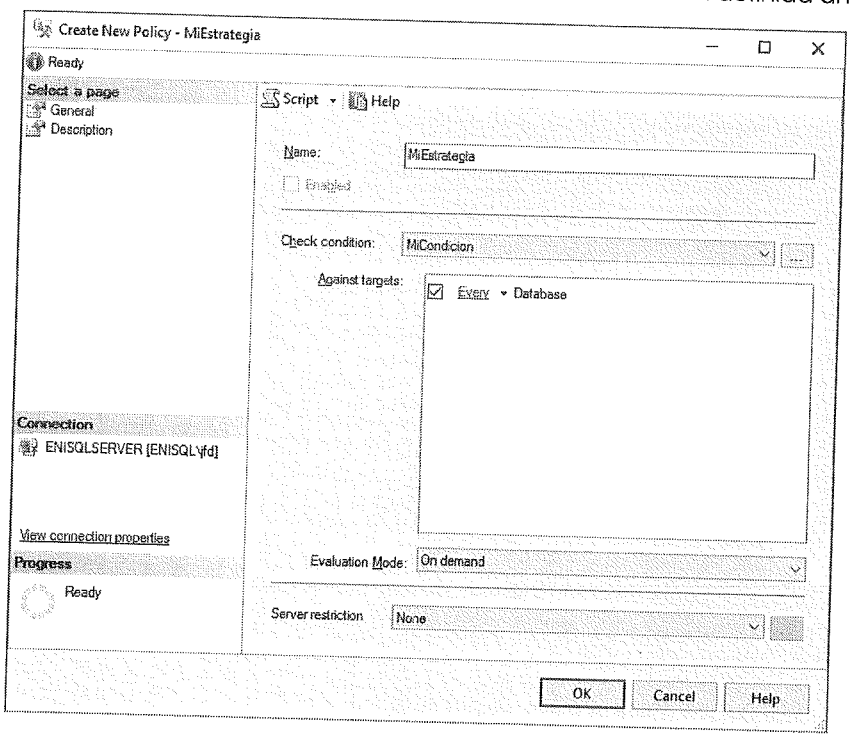
Se define la condición MiCondición.



Cuando están definidas las condiciones, es posible crear una estrategia propia seleccionando la opción **New Policy** desde el menú contextual del nodo **Policies**. Cada estrategia permite verificar una condición. Cuando se define la estrategia, se selecciona el modo de aplicación, así como la configuración de la condición.

Ejemplo

Se establece una estrategia de aplicación de la condición definida anteriormente.



17. Creación de copia en espejo

17.1 Principios de funcionamiento

La copia en espejo de las bases de datos permite garantizar una solución de alta disponibilidad conservando una inversión material razonable. En este tipo de alta disponibilidad, una base se copia en espejo con otra instancia de SQL Server. La base de datos inicial tiene el calificativo de principal, mientras que la base de datos de destino se llama espejo.

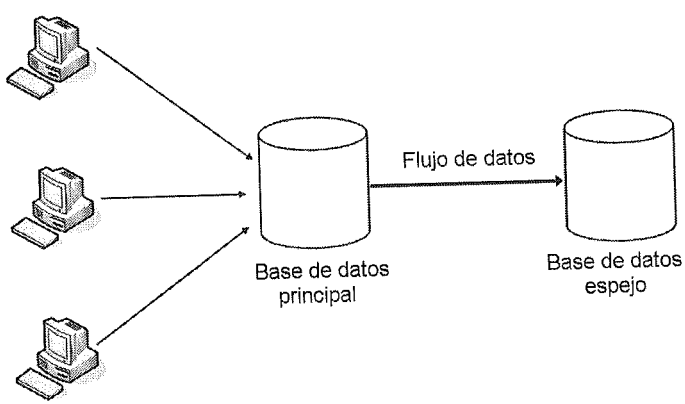
El papel desempeñado por cada una de las bases de datos es específico de cada relación de copia en espejo. El simple hecho de que una instancia contenga bases de datos que desempeñen el papel de principal en una copia en espejo no significa que no pueda contener bases de datos espejo en otras relaciones de copia en espejo.

Esta solución de alta disponibilidad no permite obviar las operaciones de copia de seguridad. Es preferible tener una copia en espejo como solución complementaria para garantizar la mejor disponibilidad posible de los datos.

La copia en espejo de las bases se efectúa mediante el registro de transacciones. Todas las transacciones efectuadas en la base de datos principal se recogen en el registro de transacciones. Esta información se registra inicialmente en memoria y luego se registra lo antes posible en disco. Es en el momento de esta escritura en disco cuando la información se envía a sus espejos. La base espejo registra esta información en disco. Las transacciones se recrean posteriormente. Las dos bases (principal y espejo) contienen, de esta manera, la misma información.

La copia en espejo puede utilizarse también para garantizar una sólida protección de los datos frente a un funcionamiento incorrecto del servidor principal. La diferencia entre alta protección y alta disponibilidad es que, en el primer caso, será una operación manual del administrador la que permitirá promover la base de destino como base principal, mientras que en el caso de una solución de alta disponibilidad esta basculación debe ser automática.

Una misma instancia de SQL Server puede contener bases que no participan en una copia en espejo, bases que desempeñan el papel de bases de datos principales y bases que desempeñan el papel de bases de datos espejo.



Conjuntamente con esta relación, puede existir un testigo que va a permitir automatizar la basculación hacia la base de datos espejo en caso de error en la base de datos principal. El testigo va a ayudar a la base principal o a la espejo a formar un quórum.

En este quórum, se pueden distinguir los casos siguientes:

- La principal no puede ponerse en contacto con la base espejo, aunque el testigo permanece accesible y por tanto el quórum está formado por la base principal y el testigo.
- La principal puede ponerse en contacto con la espejo pero no con el testigo, por lo que el quórum está formado por la principal y el testigo.
- El testigo y la espejo no pueden ponerse en contacto con la principal pero se ven entre sí. El quórum lo forman el testigo y la espejo. El testigo promueve la espejo a base de datos principal.

Con la utilización de un testigo, la solución de alta disponibilidad está garantizada y es transparente para las aplicaciones cliente, ya que es posible precisar la instancia de seguridad directamente en la cadena de conexión. De esta manera, los desarrolladores de aplicaciones no tienen que preocuparse por la basculación hacia la base de datos espejo, porque es SQL Server quien se encarga de esta operación.

Ejemplo

Ejemplo de cadena de conexión:

```
Data Source=angelsms;Failover Partner=espejo;Initial
Catalog=gescom;Integrated Security=SPI
```

La copia en espejo tiene algunas restricciones de configuración. Las principales son:

- Las dos instancias de SQL Server ejecutan SQL Server 2016. Las ediciones Workgroup y Express solo pueden tener el rol de testigo.
- La base principal debe configurarse en modo de restauración completa para poder seguir el registro de transacciones. Las importaciones en bloques (bulk logged) no pueden reproducirse de manera automática en la base espejo.
- La base espejo se instala a partir de una copia de seguridad de la base principal. Con el objetivo de poder reproducir el registro de la base principal, esta restauración debe dejar la base espejo en estado NOREVERY. Por lo tanto, la base espejo no estará accesible directamente para los usuarios. Estos tendrán la obligación de trabajar siempre con la base de datos principal.
- La base espejo debe tener exactamente el mismo nombre que la base principal.
- La copia en espejo va a afectar un poco al rendimiento de la base principal, ya que el envío de los registros a la base espejo se realiza de manera sincrónica. Es decir, la base principal se asegura de que la base espejo ha recibido y registrado correctamente el registro antes de continuar con su trabajo. Esta transferencia sobre los registros (porque el de la base espejo está sincronizado con el de la base principal) se configura de la siguiente manera en la base principal:

```
ALTER DATABASE basePrincipal SET SAFETY FULL;
```

El testigo va a permitir disparar la basculación automática hacia la base espejo en caso de error de la base principal.

La creación de reflejo de una base de datos se realiza siempre en el contexto de una sesión de creación de reflejo. En el transcurso de esta sesión, las dos bases comparten un estado para averiguar el nivel de sincronización. Estos estados son:

SYNCHRONIZING
Las dos bases de datos implicadas no contienen la misma información. La base de datos principal comunica los registros del registro de transacciones a la base de datos espejo para reproducir las modificaciones.

SYNCHRONIZED
Las dos bases de datos implicadas contienen la misma información.

SUSPENDED
La base principal trabaja sin enviar la copia del registro a la base espejo porque esta última está desconectada.

PENDING_FAILOVER
Una solicitud manual de basculación se ha realizado, pero todavía no ha sido aceptada por las bases de datos implicadas.

DISCONNECTED
El contacto con la base de datos asociada se ha perdido y eventualmente también con el testigo.

17.2 Puesta en marcha

La puesta en marcha de una solución de copia en espejo se compone de las siguientes etapas:

- Identificar la instancia principal y su espejo y la que va a jugar el papel de testigo, si existe en la solución de copia en espejo.
- Efectuar una copia de seguridad completa de la base principal. La base principal se habrá configurado en modo de restauración completa antes de efectuar la copia de seguridad (`ALTER DATABASE nombreBase SET RECOVERY FULL`). Esta copia de seguridad se restaurará sobre la espejo, aunque la base no basculará en modo recovery. Esto es así para permitir la restauración de archivos de diario. Los archivos de la base espejo deben poder crecer de la misma manera que lo hace la base principal, ya que las dos bases van a ocupar el mismo espacio en disco.

- La instancia principal y la instancia espejo se van a comunicar en este proceso de replicación. Por lo tanto, es necesario resolver todos los problemas de seguridad para permitir una conexión a la instancia remota. Si las instancias se sitúan en el mismo dominio o bien en dominios aprobados, es posible definir una conexión (LOGIN) sobre cada servidor después de asignar el derecho a esta conexión para conectarse a través de un punto de terminación (GRANT CONNECT ON ENDPOINT). Cuando las instancias se sitúen en dominios distintos sin relación de aprobación entre las dos, la conexión se apoyará en un certificado en lugar de sobre una cuenta de Windows.
- Se deben definir dos puntos de terminación (ENDPOINT) sobre los dos servidores. Estos puntos de terminación se dedicarán exclusivamente a la copia en espejo de la base.
- Aunque todas estas operaciones se pueden definir con scripts Transact SQL, y dando por hecho que la copia en espejo es una operación puntual, puede ser interesante apoyarse en el asistente de puesta en marcha que se encuentra en **Tasks - Mirror**, desde el menú contextual de la base de datos de la que se desea realizar la copia en espejo. Esta opción permite visualizar la página **Mirroring** de las propiedades de la base de datos. En esta página es posible ejecutar el asistente de configuración de seguridad, aunque también es posible especificar el servidor testigo (si existe) y el espejo.

Anexo

1. Puesta en marcha de la base de datos GESCOM

En todos los ejemplos de este libro, se utiliza a base de datos GESCOM. Esta base de datos es específica de los libros de SQL Server de Ediciones ENI, dentro de la colección Recursos Informáticos. Se crea paso a paso a lo largo del libro, a través de operaciones de SQL y Transact SQL.

Para que pueda seguir fácilmente los ejemplos que se presentan en este libro, puede descargar los scripts en el sitio Web de Ediciones ENI.

En esta descarga encontrará un script llamado **CreaciónGesCom.sql**. Hay que ejecutar este script desde SQL Server Management Studio para crear la base de datos GESCOM. Para añadir un juego de pruebas a esta base de datos GESCOM, hay que ejecutar el script **Datos_Gescom.sql**.

2. Recursos en la Web

Dirección	Descripción
http://blogs.technet.com/b/microsoft_partner_network_spain/archive/2011/01/13/grupo-de-usuarios-de-sql-server.aspx	Grupo de usuarios de SQL Server
http://www.microsoft.com/es-es/server-cloud/products/sql-server/	Sitio Web español de SQL Server

Dirección	Descripción
http://www.microsoft.com/en-us/server-cloud/products/sql-server/	Sitio Web de SQL Server
http://msdn.microsoft.com/es-Es/	MSDN
https://msdn.microsoft.com/en-us/sqlserver	Centro de recursos de SQL Server en MSDN
https://technet.microsoft.com/es-es/ms376608.aspx	Technet
http://microsoft.com/express/sql/default.aspx	SQL Express
http://support.microsoft.com	Soporte técnico de Microsoft

3. Glosario

ACL: *Access Control List*
ADF: *Application Definition File*
ADO: *ActiveX Data Object*
API: *Application Programming Interface*
AWE: *Address Windowing Extensions*
B2B: *Business To Business*
BCM: *Bulk Change Map*
BLOB: *Binary Large Object*
CLR: *Common Language Runtime*
COM: *Component Object Model*
CTE: *Common Table Expression*
DCM: *Differential Change Map*
DDL: *Data Definition Language*
DEK: *Database Encryption Key*
DML: *Data Manipulation Language*
DMV: *Dynamic Management View*
DSS: *Decision Support System*
DTA: *Database Tuning Advisor*
ETL: *Extract, Transform and Load*

GC: *Garbage Collector*
GAC: *Global Assembly Cache*
GAM: *Global Allocation Map*
IAM: *Index Allocation Map*
ICF: *Instance Configuration File*
LCID: *LocaleID*
MARS: *Multiple Active Result Set*
MDAC: *Microsoft Data Access Component*
ODS: *Open Data Services*
OLAP: *On Line Analytical Processing*
OLTP: *On Line Transaction Processing*
PFS: *Page Free Space*
RID: *Row Identifier*
RMO: *Replication Management Object*
SCC: *Setup Consistency Checker*
SGAM: *Shared Global Allocation Map*
SMO: *SQL Management Objects*
SMTP: *Simple Mail Transfer Protocol*
SOA: *Service Oriented Architecture*
SOAP: *Simple Object Access Protocol*
SQL: *Structured Query Language*
TDE: *Transparent Data Encryption*
TSQL: *Transact SQL*
TVF: *Table Valued Functions*
UDF: *User Defined aggregate Function*
UDT: *User Defined Types*
WMI: *Windows Management Instrumentation*
XML: *eXtensible Markup Language*

A

B

C

D

F

G

M

N

O

P

Q

R

S

T

U

W

X