Guía de Seguridad y Telemantenimiento de Microsoft® SQL Server®

Por

AWERTY

Clientes | Tecnología | Asistencia

AWERTY
Clientes | Tecnología | Asistencia

0. Microsoft SQL Server – Puesta en marcha

SQL Server es el sistema para la gestión de las bases de datos (BBDD) producida por Microsoft. Cuando un ingeniero de sistemas o el propio departamento de IT de una compañía trabajan con esta tecnología, debe definir previamente dos aspectos:

1. Puesta en escena. Diseñar la arquitectura del entorno SQL.
2. Problemas del día a día. Crear un entorno de producción.

SQL Server requiere unos mínimos conocimientos de un entorno que no sólo está afectado en la parte de sus bases de datos. En su desarrollo entran en juego otras variables como los consumos de memoria, de CPU o del disco del propio servidor.

Como técnicos expertos en adecuación de soluciones SQL y solución de incidentes derivados, en Awerty Servicios Informáticos hemos detectado que, a menudo, los problemas con esta tecnología vienen derivados de los servidores, que arrastran problemas endémicos o bien plantean nuevos retos para tratar de mejorar su rendimiento.

Entonces, ¿cómo afrontar los problemas con SQL? Primero de todo conviene determinar qué tipo de problema ha surgido, y sobre todo, discernir si atañe al **equipo** o al propio **SQL Server**. Antes de asumir una inversión de tiempo (y dinero) en la posible solución, esas dos partes deben quedar bien diferenciadas.

Afrontamos esta guía como un fiel reflejo de nuestra forma de trabajar día a día. Por eso, si bien es cierto que el documento puede resultar muy útil para muchos perfiles, destacamos que nos centramos sobre todo en estas empresas:

Perfil de empresa: PYME	**Núm. de usuarios:** 10-100	**Bases de Datos:** 10GB - <100GB	**Entorno:** SQL Standar o Enterprise No clusterizado

1. Lentitud, el problema más generalizado

Cuando los usuarios se quejan de la lentitud de Windows SQL Server -en cualquiera de sus formas de conexión, bien sea trabajando en aplicativo en Terminal Server, en aplicativo cliente, etc.)- en un elevado porcentaje de casos el problema se centra, como comentábamos anteriormente, en el propio servidor. Por lo tanto, deberemos aprender a buscar qué causa esa lentitud en el servidor. Enumeramos a continuación las causas más frecuentes con las que nos hemos topado.

1.1. Cuello de botella (hardware)

Un mal endémico del servidor. Desbordado, destina recursos a SQL, pero no se los puede proporcionar en su totalidad por varias razones: falta de disco, de memoria, de CPU o incluso de red.

Para estar seguros de que el cuello de botella existe realmente se deben seguir unas pautas concretas de obtención de **mediciones** y resultados. Para familiarizarse con esta práctica, o incluso llevarla a cabo, es muy recomendable leer este artículo de Microsoft Technet. Pese a la antigüedad, el texto sigue plenamente vigente. Las recomendaciones son aplicables a un servidor actual. Sólo se debe tener en cuenta que si se trata de un servidor virtual, aunque las mediciones servirán, después se deberán cotejar con las que proporciona la plataforma de virtualización en cuestión (Hyper-V, VMWare, Citrix, etc.).

En el citado artículo se hace mención al cuello de botella de proceso y a cómo los resultados pueden proporcionarnos información útil.

Para llevar a cabo las mediciones a las que hacíamos referencia es necesario conocer el entorno del monitor de rendimiento: tanto de 2003, 2008 como de 2012.

- Performance Monitor en Windows Server 2003
- Performance Monitor en Windows Server 2008 y 2012 (este artículo es el inicial, debe seguirse toda la guía de subtemas).

1.2. Almacenamiento | Tipo de RAID y formateo del volumen

Dado que las necesidades de cada uno de estos elementos con respecto al almacenamiento son distintas, lo ideal sería separarlos en conjuntos de discos diferentes y dedicados en exclusividad. Como norma general, se debe eliminar el uso de RAID 5 para SQL puesto que la escritura en este tipo de RAID está muy penalizada: calcula y graba una paridad por cada escritura de datos.

Siempre que sea posible, conviene utilizar RAID 10 o RAID 50, pero esto sería muy caro, así que podríamos considerar lo siguiente como línea base:

Ficheros LOG

- **RAID 10** con 4 discos debería ser suficiente.
- RAID 1 si el espacio en disco es justo
- Formateado en **sectores de 4KB**

Ficheros MDB

- **RAID 10** con máximo de discos posibles. Aquí es donde no se deben escatimar recursos. SSD a ser posible
- Formateando en **sectores de 64KB**.
  Nota: Vigilar con el formateo NFTS de Windows por defecto porque lo hace en sectores de 4.096 bytes y SQL trabaja con páginas de 8.192. Es decir, cada operación de SQL obligaría a leer dos sectores de disco. Además, SQL lee siempre 8 páginas de la BBDD, independientemente de la información que se haya solicitado y retiene en memoria la información no solicitada en lugar de descartarla para mejorar el rendimiento.
  Ocho páginas de 8 KB cada una = 64 KB. ¡Perfecto!

Sistema operativo y pagefile

- Un RAID 1 sería suficiente, si podemos separarlos. Tendríamos un RAID uno para cada uno de ellos. Sería preferible a RAID 5 puesto que este penaliza mucho la escritura aleatoria
- Formateo nativo de Windows

Almacén de backups

- RAID 1 con dos discos sería suficiente a menos que tengamos una necesidad de throughput muy elevada. En cuyo caso RAID 10 o discos SSD.

1.3. Almacenamiento | Sobre la capa de software

Antes de diseñar el almacenamiento se debe tener claro cómo nuestro software va a utilizar el hardware de dicho almacenamiento. Evidentemente, también es determinante la intensidad con la que va a utilizarlo.

El diseño variará radicalmente en funcion de si pensamos en una aplicación X para 50 usuarios o si pensamos en la misma aplicación para 1.000 usuarios concurrentes. Tratándose de Windows + SQL Server, como norma general, entrarán en juego los siguientes elementos de software:

Ficheros LOG de SQL	Ficheros MDB de SQL	Sistema operativo	Archivo de paginación SO	Almacén de Backup
•Escritura •Acceso secuencial •Pocos KB por operación	•Lectura •Acceso aleatorio	•Lectura y escritura aleatoria	•Lectura y escritura aleatoria	•Escritura •Acceso secuencia

1.4. Almacenamiento | Sobre los controladores

Dejando al margen los discos, hay otro elemento que dificulta la toma de decisión de la velocidad que debe tener el RAID. Las **controladoras de disco.** Pueden tener más o menos memoria caché y esta puede ser más o menos rápida. Incluso, la caché de cada controladora puede trabajar de forma síncrona con la otra a través de un canal UDMA, o no.

En definitiva, hay tantos escenarios posibles que es fácil no llegar a obtener ningún cálculo que garantice un 100% de seguridad. Por eso, inicialmente, lo mejor es hacer los cálculos basándose en los valores que proporcionan las unidades de disco, que siempre serán razonablemente más aproximadas, y dejar que después las controladoras hagan su trabajo.

Como norma general en cuanto a las controladoras de disco podríamos tener en cuenta:

Cuanta **MÁS MEMORIA CACHE** tenga la controladora, más información podrá almacenar y menos hará trabajar a los discos	Si la memoria caché es **CONFIGURABLE**, tratándose de SQL Server deberíamos configurarla para **70% lectura y 30% escritura**	La memoria **NO DEBE HABILITARSE**: en caso de problema eléctrico o del propio disco no garantiza que la información almacenada

1.5. Almacenamiento | Otros factores a tener en cuenta

A parte del número de IOPS, otros dos valores a tener en cuenta en nuestro almacenamiento son:

Ancho de banda en Mbps. No es tan determinante en un entorno SQL, puesto que este requiere muchas operaciones de IO pero de tamaño muy pequeño. Sería distinto si estuviésemos diseñando un almacenamiento para streaming de audio o vídeo o, en definitiva, para recoger o almacenar datos de forma masiva. El ancho de banda vendrá determinado por el número de IOPS, el tamaño del búfer y el método de acceso (aleatorio o secuencial).

- Latencia. Cuánto se demora en hacerse efectiva cada operación que pedimos a nuestro almacenamiento. Es decir, cuántos milisegundos pasan desde que el SO lanza la escritura de un dato y el almacenamiento confirma que el dato ya está escrito.

1.6. Almacenamiento | ¿Qué protocolo de acceso es mejor?

En lo relativo al protocolo de acceso al almacenamiento sería necesario disponer previamente de los datos sobre los requerimientos que va a tener nuestra infraestructura. Sin disponer de esta información, como norma general, deberíamos tener esto en mente:

- Evitar siempre el acceso o almacenamiento vía SMB/CIFS.
- No subestimar el rendimiento que podemos obtener mediante iSCSI o NFS. Más ahora que el ethernet a 10 GB empieza a extenderse. Está bien pensar en ethernet siempre como punto de partida, sabemos que tenemos cosas mejores si las necesitamos.
- La fibra óptica siempre será mejor que el acceso ethernet, aunque probablemente no tanto como uno puede pensar *a priori*. Es más caro, eso seguro, y como se suele decir '*no todo el mundo necesita un Ferrari*'.

2. ¿Cómo saber si el almacenamiento supone un cuello de botella?

Averiguar si tenemos o no un cuello de botella en algún punto y cuál es, puede requerir mucho tiempo de análisis. Para saber si este cuello de botella está en el acceso al almacenamiento, inicialmente, podríamos fijar nuestra atención en tres puntos:

1. Longitud de cola de disco (de lectura o de escritura) a nivel de SO.
Siempre que hay una operación en cola indica que está esperando algo. ¿A qué espera? (cuando sepamos esto, misterio resuelto).

2. Longitud de la cola de disco a nivel de controladora.
Hay que pensar que la controladora en una operación de escritura: almacena el dato en caché y automáticamente le indica al disco que está escrito aunque realmente no sea así. Todavía se tiene que escribir realmente. La controladora puede experimentar lentitud (cuello de botella) si los discos que tiene por debajo no pueden dar todo lo que se les pide.

3. Latencia en ms por operación.
Como norma general, en un entorno SQL no se debe trabajar por encima de 10/15 ms. Aunque, esa cifra variará mucho en función de qué medio utilicemos para acceder al almacenamiento. La fibra óptica, por ejemplo, da menos latencia. A más milisegundos, la experiencia del usuario se irá degradando. Por encima de 30/40 ms pueden comenzar a oírse quejas

2.1. Sistema operativo (SO)

La lentitud o el mal rendimiento también pueden venir originados por problemas en el propio sistema operativo. Hay diversas formas de tratar de obtener información de un servidor para llegar a conclusiones acerca de su salud. Para empezar, es bueno aprender a leer, interpretar y filtrar su visor de sucesos.

Microsoft publica varios artículos al respecto. Lo que sucede es que habiendo todavía bastantes servidores con la plataforma Windows 2003 Server es conveniente separarlos. Su funcionamiento varía un poco respecto a las versiones más nuevas 2008 y 2012.

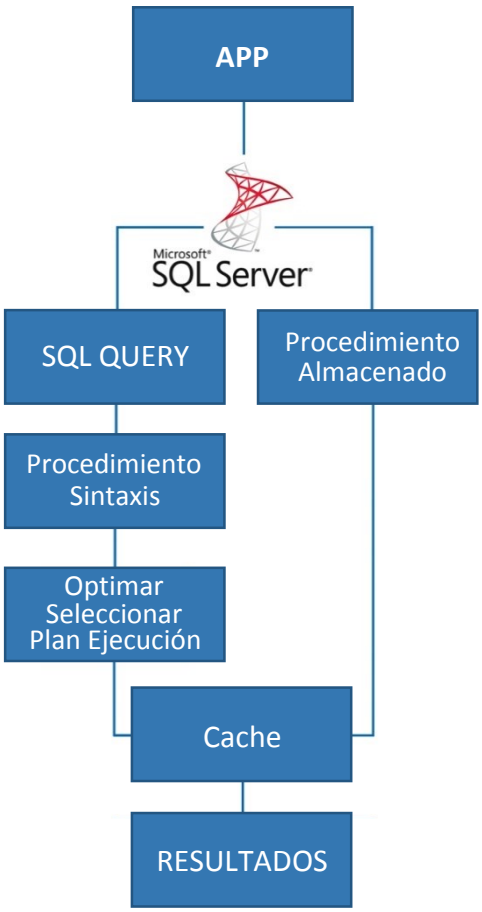
Visor de sucesos en Windows Server 2003

Visor de sucesos en Windows Server 2008 y 2012

Lo que debemos aprovechar de la utilización del visor de sucesos es el filtrado por origen, fechas, errores y advertencias, etc. De este modo, tendremos una información más centrada en lo que nos interesa y el espectro de opciones se reducirá.

2.2. Aplicación (software)

En esta capa se debe tener en cuenta que tanto si se trabaja con una aplicación que hace procesos contra el SQL (automáticos y o invisibles para el usuario), como si es el propio usuario quien genera las consultas, reports, etc., hay una serie de recomendaciones para optimizar consultas en SQL server en Aplicaciones.



2.1.1. Best practices consultas SQL Server

- Si la consulta utiliza cursores determina antes si se puede escribir la consulta con un tipo de **cursor más eficaz** (como uno de avance rápido) o con una **única consulta**. Las consultas únicas normalmente mejoran las operaciones de cursor. Dado que un conjunto de instrucciones de cursor suele constituir una operación de bucle externo, en la que cada fila del bucle se procesa una vez con una instrucción interna, se puede contemplar la posibilidad de utilizar en su lugar una instrucción GROUP BY o CASE. Quizá incluso una subconsulta.

- No utilizar varios alias para una sola tabla en la misma consulta para simular la intersección de índices. Ya no es necesario debido a que SQL Server tiene en cuenta automáticamente la intersección de índices y puede utilizar varios en la misma tabla de la consulta.

- Utilizar la parametrización de consultas para permitir la reutilización de los planes de ejecución de consulta almacenados en la memoria caché. Si un conjunto de consultas tiene el mismo hash de consulta y hash de plan de consulta, podría mejorar el rendimiento creando una **consulta parametrizada**. Llamar a una consulta con parámetros en lugar de a varias consultas con valores literales permite reutilizar el plan de ejecución de consulta almacenado en la memoria caché.

- Uso de Exists. Cuando queramos hacer una sub-consulta en una base de datos utilizando la sentencia NOT IN, analicemos si podemos cambiar nuestro queries con el uso de la sentencia Exists que es mucho más eficiente que la anterior. O en todo caso, utilizar IN en vez de NOT IN, ya que esto hace un escaneo completo en la tabla descartando opciones a omitir.

- Uso de Distinct. Utilizar distinct para excluir datos duplicados es muy usado por los programadores para evitar errores de diseño de base de datos y así esconder algunos duplicidad de información, pero esto es un grave error. Es una de las sentencias que más necesita hacer I/O en el disco y forzar bastante el procesador. Por tal motivo, si es necesario evitemos utilizarla.

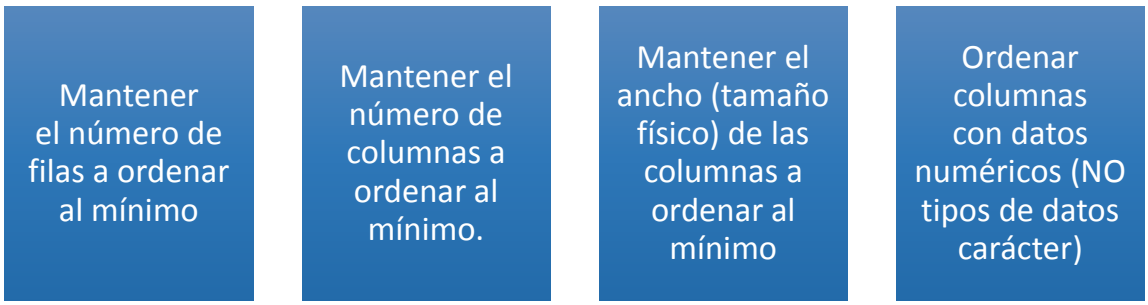
- Uso de Top. Cuando se quiere traer un grupo de registros es mejor utilizar la sentencia Top y no Rowcount, ya que esta última presenta inconvenientes con **listas no ordenadas**. En cambio, si la lista es ordenada es más eficiente que la sentencia Top.

- Uso de*. Cuando se realizan consultas que van a devolver muchos campos es mejor definir todos los campos que queremos devolver en nuestro queries, ya que el uso de * o All impide el uso de índices de forma eficiente.

- Verificar si existe un registro. Muchos programadores utilizan el count(*) para ver si un registro existe en la base de datos, pero una forma más eficiente de hacerlo es con Exists. Cuando éste encuentra un registro detiene la búsqueda del mismo.

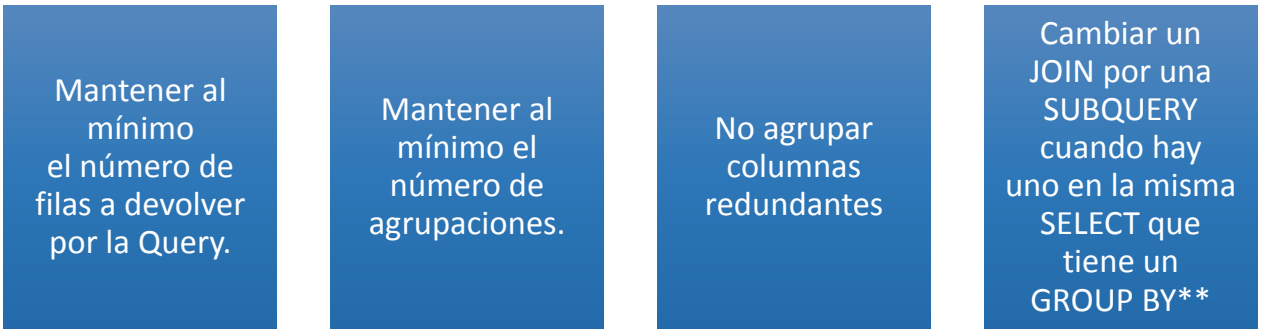
- Uso de ORDER BY. Usar ORDER BY en las QUERIES que se lancen sólo si es absolutamente indispensable. Es decir, que si es posible realizar la ordenación en el

lado del cliente siempre será mejor que realizarla desde el lado del servidor SQL Server.
En caso de que sea absolutamente necesario realizar la ordenación en el lado del servidor SQL Server deberemos atender a las siguientes recomendaciones:



Cuando usemos cualquier mecanismo de ordenación en Transact –SQL, debemos tener en mente todas estas recomendaciones para la mejora del rendimiento.
Si se ha de ordenar por una columna a menudo, debemos considerar el realizar un "Clustered Index" sobre esa columna para la mejora del rendimiento.

- No usar el comando GROUP BY sin una función de agregación. La cláusula GROUP BY puede usarse con o sin una función de agregación. Pero si queremos obtener un mejor rendimiento, no usaremos la cláusula GROUP BY sin una función de agregación. Esto es porque produce el mismo resultado usar DISTINCT y es más rápido.
  Para **acelerar el uso** de la cláusula GROUP BY debemos seguir las siguientes recomendaciones:



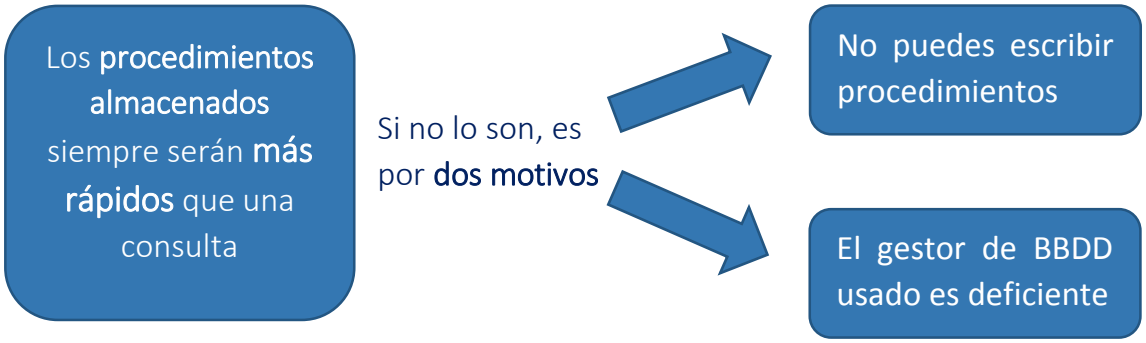
** Si es posible hacer esto, el rendimiento será mayor. Si se tiene que usar un JOIN, intentaremos hacer el GROUP BY por columna desde la misma tabla que la columna o columnas sobre la cual la se usa la función.

Consideraremos añadir un ORDER BY a la SELECT que ordena por la misma columna que el GROUP BY. Eso puede producir que el GROUP BY tenga mejor rendimiento.

2.1.2. Uso de Procedimientos Almacenados.

Reducen la carga de mantenimiento de código SQL, y aumentar la velocidad de la aplicación.

- El mantenimiento de las consultas SQL es más sencillo, porque las consultas SQL están centralizadas en el gestor de Base de Datos (BBDD).
- La seguridad se enriquece utilizando Procedimientos Almacenados, debido a que un buen DBA (Administrador de Bases de Datos) únicamente proveerá una lista de procedimientos (bien tipificados y documentados) que el programador llamará de acuerdo a sus necesidades. Así, evitamos que el programador deba preocuparse por verificar que los INSERT, DELETE, etc. estén 'bien'. Lo libramos de la carga de manipular directamente la base de datos. Y dejamos esa lógica donde debe ir: en el gestor.
- Supón que debes añadir una condición extra a la función de ejemplo 'ConsultarDatosUsuario'. Para ello, se debería cambiar la cadena que se retorna en la función, probar los cambios, compilar o publicar tu aplicación. Los SP se pueden modificar, sin necesidad de recompilar y hacer un deploy completo de la aplicación.



2.1.3. Gestión de datos en la aplicación

Si una aplicación utiliza un bucle, considera la posibilidad de colocar el bucle en la consulta. A menudo, una aplicación contendrá un bucle que, a su vez, contendrá una consulta con parámetros que se ejecutan muchas veces y será necesario realizar un viaje de ida y vuelta en la red entre el equipo que ejecuta la aplicación y SQL Server.

En su lugar, crea una sola consulta más compleja con una tabla temporal. Sólo se necesita un viaje de ida y vuelta en la red, y el optimizador de consultas puede optimizar mejor la consulta única.

2.2. SQL Server (software)

La lentitud bien puede estar producida por el propio sistema SQL. Para empezar a conocer si al servidor de bases de datos le ocurre algo lo óptimo es centrarse en la información que nos reporta en forma de log. En este caso, también se deben separar las versiones de SQL para poder usar el visor de eventos, dado que varía según sea 2008 o 2012.

Visor de log's para SQL 2008

Visor de log's para SQL 2012

A partir de aquí, podremos ver si el problema de rendimiento del servidor SQL puede estar dado por alguna de sus funcionalidades o propiedades de configuración. Uno de los hechos que más se propaga en los problemas con SQL es su propiedad de **memoria asignada**.

Es bastante común ver que SQL devora toda la memoria RAM. Si ese fuera el problema (o queremos averiguar si lo es), se puede seguir esta guía que publicamos.

Otro aspecto sería comprobar si SQL Server está bien **configurado** (de forma estándar), y si lo que ya lo está, está funcionando bien.

A grandes rasgos, lo que es importante para un SQL Server, es que haya una serie de **tareas automatizadas** que sin lugar a dudas nos ayudarán a mejorar el rendimiento. Para familiarizarnos con estas tareas, debemos conocer lo que llamamos "Planes de Mantenimiento".

Planes de Mantenimiento en SQL 2008

Planes de mantenimiento en SQL 2012

Lo que harán los planes de mantenimiento será controlar factores y variables que, de otra forma, a medio y largo plazo supondrán un problema (seguro) para nuestro SQL Server. Por ejemplo, debemos familiarizarnos con:

El crecimiento de los ficheros LOG

Los ficheros log, son los ficheros del registro de transacciones y es el fichero que almacena todos los cambios y transacciones. Su crecimiento es exponencial, y se debe controlar y limitar.

El backup de la base de datos

Generar una copia de seguridad de la base de datos, nos dará la fiabilidad de tener todos los datos seguros, a la vez que estamos realizando un mantenimiento "higiénico" sobre la propia base de datos.

La reorganización (o también puede ser reconstrucción) del índice

Los índices son datos estructurados de la propia base de datos, por medio de varios identificadores. El hecho de que gocen de buena salud, o no, hará que las consultas contra la base de datos tengan mayor o menor respuesta.

Creadas todas estas tareas y ejecutadas de forma periódica o manual, nuestro servidor tendrá, por lo menos, unos cuantos frentes menos con los que lidiar en futuros problemas.

De forma ya bastante más profunda en el entorno, podemos proponer un inicio de investigación en la parte de auditoría SQL. Para familiarizarse con esta práctica, es necesario un nivel un poco más avanzado del entorno, pero lo cierto es que la auditoría podría darnos información tan valiosa como, por ejemplo, quién está haciendo cambios constantes en una tabla o qué ocasiona un cuello de botella en el proceso.

Auditoría SQL para Windows Server 2008

Auditoría SQL para Windows Server 2012

3. Conclusiones y recomendaciones

La información que se almacena en cualquier aplicación y/o empresa es cada vez mayor. Eso justifica la importancia de SQL Server, un entorno del que las empresas cada vez dependen más, y de dedicar a esta tecnología –como mínimo- unos recursos específicos para gestionarse. Tanto humanos como económicos.

Nuestra relación de partners de Microsoft nos avala para proporcionar todo el conocimiento necesario de esta tecnología, tanto en el diseño, como en su posterior desarrollo y en la resolución de los eventuales problemas que puedan surgir.

Trabajamos con confianza, confidencialidad y transparencia.

La **seguridad de la información** de la compañía

CON NUESTRO SERVICIO

Los **recursos humanos** y **económicos** de la compañía

Esperamos que la presente guía te haya sido de utilidad. Si quieres hacernos cualquier consulta, no dudes en ponerte en contacto con nosotros a través de nuestra página web.