# Infor LN Performance, Tracing, and Tuning Guide for SQL Server

**Publication Information**

Release: Infor LN 10.x

Publication date: November 28, 2014

Document code: B0079C

# Contents

# About this guide

This document provides guidelines to optimize the performance of an Infor LN environment on a SQL Server database by tracing and tuning the environment.

Note: This document is a comprehensive compilation; however there may be instances wherein relevant information or procedures may have been omitted. Therefore, we recommend verifying the proposed changes in a test environment before moving to production. The information provided may not hold true for future versions of the SQL Server database.

# Intended audience

This document is intended for intermediate to expert Infor LN and database Administrators and Technical Consultants and aims to better the performance of the Infor LN system.

# Related documents

Certain sections in this document are described in more detail in other documents. The following documents help to extend the knowledge in particular areas.

- *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*
- *Infor LN - Sizing guide (B0045 US)*
- *Infor LN - Data compression (B0050 US)*
- *Infor Enterprise Server - Technical Reference Guide for* Microsoft SQL Server *Database Driver (U8173 US)*

You can find the documents in the product documentation section of the Infor Xtreme Support portal, as described in "Contacting Infor" on page 6.

A good reference to SQL Server documentation about tracing and tuning is important. The following site has proven content: http://msdn.microsoft.com/en-us/library/ms130214.aspx

# Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at www.Infor.com/Inforxtreme.

If we update this document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@Infor.com.

# Chapter 1 Tuning SQL Server for Infor LN

1

The performance of Infor LN highly depends on the database performance. This chapter describes the important tuning areas of SQL Server.

## Introduction

Tuning SQL Server is not the standard method to repair your system. This chapter describes what can be done based on the performance issues you have identified.

Usually, Infor LN runs efficiently with an out-of-the box SQL Server database, if the setup of the SQL Server database is planned appropriately. Additional tuning further improves performance.

Regarding SQL Server Performance Tuning, it is important to note that performance tuning is not a search for the “magic” switch. Each area of tuning is important to the overall performance, and must be approached using a logical methodology. Proper organization, logical changes, and documentation classify the difference between effective and ineffective tuning. Performance tuning along with capacity planning and sizing allow you to design, setup, implement, and maintain a smooth and optimal operating system.

In this chapter, the term CPU is used frequently and refers to the Windows Operation systems description of this term. In practice, it can be a CPU core or CPU thread. Only when an exception applies, it is explicitly stated.

The following topics are discussed in additional detail to make the database “tuning” process easier:

- SQL Server Information resources
- I/O Setup
- Storage Setup
- Network setup
- Database statistics
- (N)VARCHAR
- SQL Server Parameters

# SQL Server Information resources

For Generic SQL Server Information and SQL Server Tuning, information is available on the internet and from various Microsoft SQL Server books. However, most of these are copies of the SQL Server Books Online; Therefore, SQL Server Books Online (BOL) is a valuable information source for the SQL Server.

For more Information, see these books (Series):

- *Inside Series MSQL Server by the Microsoft Press*.
- *Microsoft SQL Server Internals* by the Microsoft Press.
- *SQL Server Internals and Troubleshooting* by the Wrox Press
- *SQL Server Magazine*. This magazine lists all the Microsoft SQL Server updates on a monthly basis.
- MSDN blogs of the SQL Server Customer Advisory Team (SQLCAT) Web site, where SQL Server Developers include additional information (not part of the official SQL Server build).

# I/O Setup

The *Infor LN - Performance, Tracing and Tuning Guide (U9357)* provides guidelines for a correct I/O setup. For the SQL Server, the following additional guidelines are important:

- Configure Stripe sizes for the SQL Server volumes to the recommended size of 64K or a value divisible by 8K.
- Configure NTFS Allocation Unit for the SQL Server used file systems to the recommended size of 64K or a value divisible by 8K.
- Always use the new configured and formatted disks for the SQL Server data and the log storage. Most SQL Administrators tend to check only the internal SQL Server data and index fragmentation. It is also important to regularly check for the existence of highly fragmented SQL Server file storage on the Windows file system level.

# Storage Setup

A good storage setup is important for the SQL Server performance. The following guidelines help design and implement the storage setup:

- Usage of filegroups to improve performance is not recommended. It is recommended that you spread the I/O across multiple disks, based on your requirement. Only “hotspots” (for example for a specific, heavily used, table as the device queue) in the Infor LN database that exist for an extended period of time are possible scenarios for I/O separation using filegroups. The %BSE%/lib/msql/msql_storage_param fully supports this process.

The following example displays the appropriate setup of the %BSE%/lib/msql/msql_storage_param to separate a table:

```
ttaad320:000:T:group:0400:FILEGROUP HOTSPOT
*:*:T:group:0400:FILEGROUP PRIMARY
*:*:I:FILEGROUP PRIMARY
```

- Split the Infor database into multiple data files. For CPU cores up to 8, it is recommended that you use one datafile per core. Above 8 CPU cores: use 8 datafiles and add an additional datafile for every two additional cores, because each data file starts with an own header page, PFS, SGAM, and GAM page. Contention on these pages is reduced by creating multiple copies of these pages. This is done by creating multiple data files.
- The I/O demand for the Tempdb is very low in normal circumstances. Additional functionality that also includes the use of Infor LN databases increases the I/O demand of Tempdb; in such a scenario, separation of Tempdb to its own volume is recommended. High I/O demand in Tempdb is usually due to poorly written SQL or complex queries.
- Split the SQL Server Tempdb database into multiple data files. It is recommended that for 2 CPU cores you use 1 data file, up to a maximum of 8 data files.
- The Infor LN database must be created with the appropriate size. For example, if at least 150 GB is required for a year, create the database with the required size from the beginning. Do not let the size increase to the required size and cause additional fragmentation in the database and the NTFS volume.
- Set the autogrowth to a minimum of 10 percent or 1 GB (for each extent) to avoid fragmentation on file system level. Never use the default extent size of 1 MB.
- SQL Server does not use circular transaction logging, such as Oracle wherein the oldest transaction log file is overwritten when the transactions are flushed to the data files, however the SQL Server transaction log file data increases. You must set up an efficient log backup procedure to free up log space and maintain the log within acceptable values.

See the SQLCAT website and the Microsoft knowledge base, for more Information on I/O and Storage Setup best practices.

# Network Setup

## 2-Tier

For an optimal SQL Server (network) performance, the Infor LN SQL Server database driver uses the SNAC (SQL Native Access Client) driver instead of MDAC/WDAC (Microsoft/Windows Data Access Components). Therefore, the Infor LN database driver accesses SQL Server using the shared memory instead of the named pipes. This delivers approximately 10 percent performance increase when compared to the named pipes protocol in a 2-Tier configuration. Shared memory is always enabled on a SQL Server installation. When not specified, the SNAC enabled Infor LN

database driver always uses shared memory. The following example in $BSE/lib/defaults/db_resource enables the shared memory when the same is not specified:

```
msql_serverhost=<local systemname>
```

Alternatively, you can specify a dot instead of the systemname:

```
msql_serverhost=.
```

## 3-Tier

TCP/IP is the preferred protocol when the SQL Server and Infor LN are installed on separate servers (3-Tier configuration). This is applicable for both the SNAC enabled SQL Server database drivers and the older WDAC/MDAC based database drivers. To use the SNAC driver, you must ensure that the SQL Server client software is installed on the application server.

see http://msdn.microsoft.com/en-us/library/ms810810.aspx#bkmk_MDAC_WDAC, for more information about data access technologies.

# Database Maintenance

An optimized and well-performing database requires maintenance. A lot of tasks are automatically done by the database engine, but others must be done using the automated scripts, preferably during a maintenance window. During this activity, the system can be utilized, but the user activity must be minimal. You must ensure there is no overlap between your batch window and your database maintenance window, to prevent locking issues. The following maintenance tasks are required and are described in additional detail:

- Index maintenance
- Database integrity check
- Create/update statistics

## Index maintenance

As data changes over time, indexes are fragmented. An index is fragmented when the physical ordering of the index pages on the disk do not match the logical order of the data (as defined by the index key). Heavily fragmented indexes can reduce the speed of data access and decrease query performance. Index fragmentation can be corrected by reorganizing or rebuilding the index.

During index reorganization, the index is still accessible. However, offline rebuilding or re-creating the index does prevent user access to the index. To allow user access during the index rebuild, an online rebuild is required. Note: Rebuilding indexes online requires the Enterprise edition of SQL Server.

Checking index fragmentation periodically and taking the required corrective actions is important to maintain the performance of your Infor LN environment. The fragmentation rate depends on the user activity, but as a general rule, it is recommended that you check for index fragmentation during your weekly maintenance window.
For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) Web site. These scripts check the fragmentation rate of the index and, if required, take the corrective action based on certain thresholds.

## Database Integrity

Although a SQL Server ensures data integrity, data corruption, because of a hardware problem, can occur. Therefore, it is recommended to regularly check the integrity of the database. This can be done using automated scripts during the weekly maintenance window. For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) web site. If any problems are detected, you can restore data from the backups or use one of the REPAIR options on DBCC CHECKDB.

## Database Statistics

Creating database statistics on a SQL Server is important. In most circumstances, the default **auto create statistics** database option creates the required statistics. When disabled, you are required to create statistics manually. Create missing statistics on the newly created data at regular intervals. You must create and update statistics during the maintenance hours, when activity on the system is minimal.

The following procedure creates missing statistics in a particular database:

```
sp_createstats @fullscan = 'fullscan', @indexonly = 'indexonly'
```

This procedure scans all the data and statistics created only for index. In SQL Server terminology, index means statistics for key distribution of the index, that is, the first column of an index key.

The following example creates missing statistics for all columns, including columns in a composite index key. This approach is recommended for creating statistics for a SQL Server. However, additional storage is required for statistics. Using the fullscan option the SQL Server Query optimizer can create an optimal plan in most cases:

```
sp_createstats @fullscan = 'fullscan'
```

The following procedure can be used to update statistics in a database:

```
sp_updatestats
```

To prevent shared schema (SCH-S) locks on the database during normal operation hours, it is recommended to set the SQL Server **AUTO_UPDATE_STATISTICS_ASYNC** option to ON. This updates statistics asynchronously with a minimal impact on the database concurrency.

```
ALTER DATABASE YourDBName SET AUTO_UPDATE_STATISTICS_ASYNC ON
```

Apart from the auto create/update facilities, it is recommended to update database statistics on a weekly basis. For a list of commonly used index maintenance scripts, see Ola Hallengren's (https://ola.hallengren.com) Web site.

# Data Volume

The volume of data can increase faster than expected, especially when too much history is stored or financial logging option is used. To reduce the data and save storage usage, consider the following:

- Database compression
- Varchar data type

## Database Compression

From portingset 9.0a onwards, Infor LN supports table and index page compression. The benefits of Data compression:

- Significant savings in disk storage space.
- Fewer page reads, because more rows can fit on a page.
- Reduced disk I/O activity, because more compressed rows than uncompressed rows are included in a page.
- Ability to compress older data that is not accessed often, such as archived tables, while the frequently accessed data can be stored in the uncompressed form.
- Possibility to free space no longer required for a table.
- Faster backup and restore.

Compression ratios of more than 5x are possible, depending on your data. See *Infor LN Data compression (B0050 US)* for more information.

## (N)VARCHAR Data Type

From porting set 8.4b onwards, the standard data types used by the Infor LN SQL Server database are VARCHAR and NVARCHAR. This significantly reduces data size and data growth. Consequently, the amount of I/O reduces and the overall performance also improves.

# SQL Server parameters

In most cases, Infor LN runs efficiently with an out-of-the box SQL Server database with standard parameters. Modifications of parameters can eliminate a few issues or influence resource usage on high-end SQL Server systems. The parameters used by benchmarks and customers are listed in this table.

| **sp_configure parameter** | **recommended value** | **importance** |
|---|---|---|
| Affinity (64) I/O Mask | Default (0) | Optional (High-end only) |
| Affinity (64) mask | Default (0) | Optional (Special purpose) |
| Blocked process threshold | Default (0) | Optional (Tracing purpose) |
| Max degree of parallelism | 1 | Recommended |
| Min / Max Server memory | Fixed value | Recommended |
| Max worker threads | Lowest possible value | Recommended |
| Priority boost | Default (0) | Recommended |
| Recovery interval | Default (0) | Optional (Special purpose) |
| **“special” settings** | **recommended value** | **importance** |
| Lock Pages in Memory | According value | Recommended |
| Large Pages | Trace flag 834 | Optional (High-end only) |
| Plan cache size | Trace flag 8032 | Optional (High-end only) |

**Caution:** Although this does not occur often, setting one or more of these parameters in the database can lead to issues in other products or Infor LN sessions. Therefore, it is recommended that you perform a validation test before implementation.

## Affinity (64) I/O Mask

The **affinity I/O mask** option binds the SQL Server disk I/O to a specified subset of CPUs. In high-end SQL Server online transactional processing (OLTP) environments, this extension can enhance the performance of SQL Server threads issuing I/Os by storing the threads on the CPUs which processes the I/O and also, at the Windows level. This must be used in conjunction with the “affinity mask” setting. This enhancement does not support hardware affinity for individual disks or disk controllers.

## Affinity (64) mask

By segregating the SQL Server threads to run on particular processors, Windows can better evaluate the system's handling of processes. For example, on an 8-CPU server running two instances of SQL Server (instance A and B), the system administrator can use the **affinity mask** option to assign the first set of 4 CPUs to instance A, and the second set of 4 to instance B. The same is applicable when to segregating the SQL Server from Infor LN in a 2-Tier setup.

## Blocked process threshold

Use the **blocked process threshold** option to specify the threshold, in seconds, at which blocked process reports are generated. The threshold can be set from 0 to 86,400. By default, no blocked process reports are produced. This event is not generated for system tasks or tasks waiting on resources that do not generate detectable deadlocks.

You can define an alert to be implemented when this event is generated. For example, you can (choose to) page the administrator to take appropriate action to handle the blocking situation.

## Max degree of parallelism

When the SQL Server runs on a computer with more than one CPU, the server detects the degree of parallelism, that is, the number of processors employed to run a single statement, for each parallel plan execution. You can use the **max degree of parallelism** option to limit the number of processors that must be uses in the parallel plan execution. The default value of 0 uses all available processors. Set the **max degree of parallelism** to 1 to suppress parallel plan generation. Set the value to a number greater than 1 (up to a maximum of 64) to restrict the maximum number of processors used by a single query execution. Even if a value greater than the number of available processors is specified, only the actual number of available processors is used. If the computer has only one processor, the **max degree of parallelism** value is ignored.

When SQL Server is only used for Infor LN, it is recommended to set the **max degree of parallelism** to 1. Infor LN queries are usually short queries and do not benefit from parallelism. When Infor LN is used with different applications, such as reporting; it is recommended to set the **max degree of parallelism** to a value greater than 1. The default value of 0 can affect the Infor LN Performance if a heavy query or a query with performance issues is used on the SQL Server.

## Min / Max Server memory

By default, SQL Server dynamically allocates system memory. However, the memory size can increase to the system maximum which can be an issue, because memory is released only when the memory pressure is high.

Use the **min server memory** and **max server memory** options, to reconfigure the SQL Server instance memory (in megabytes). It is recommended to fix the memory size of the instance by setting the **max server memory** and **min server memory** to the same value.

**Note:** You must review your current **max server memory** and **min server memory** values after you upgrade to SQL Server 2012, the server includes and accounts for more memory allocations compared to earlier versions. See http://support.microsoft.com/kb/2663912.

## Max worker threads

Use **the max worker threads** option to configure the number of worker threads available for the SQL Server processes. The SQL Server uses the native thread services of the Windows operating system, so that one or more threads support each network that SQL Server supports simultaneously. Another thread manages the database checkpoints, and a pool of threads manages all users.

Set this SQL Server parameter to the lowest possible value to optimize the database. Setting **the max worker threads** parameter to a high value results in excessive context switches and decreased performance as most of the Infor LN database requests are short queries.

## Priority boost

Use the **priority boost** option to specify if the SQL Server must run at a higher Windows scheduling priority when compared to the other processes on the same computer. If you set this option to 1, the SQL Server runs at a priority base of 13 in the Windows scheduler. The default is 0, for which the priority base is 7. This parameter can also be used, in specific scenarios, for a 2-Tier system with a high load.

## Recovery internal

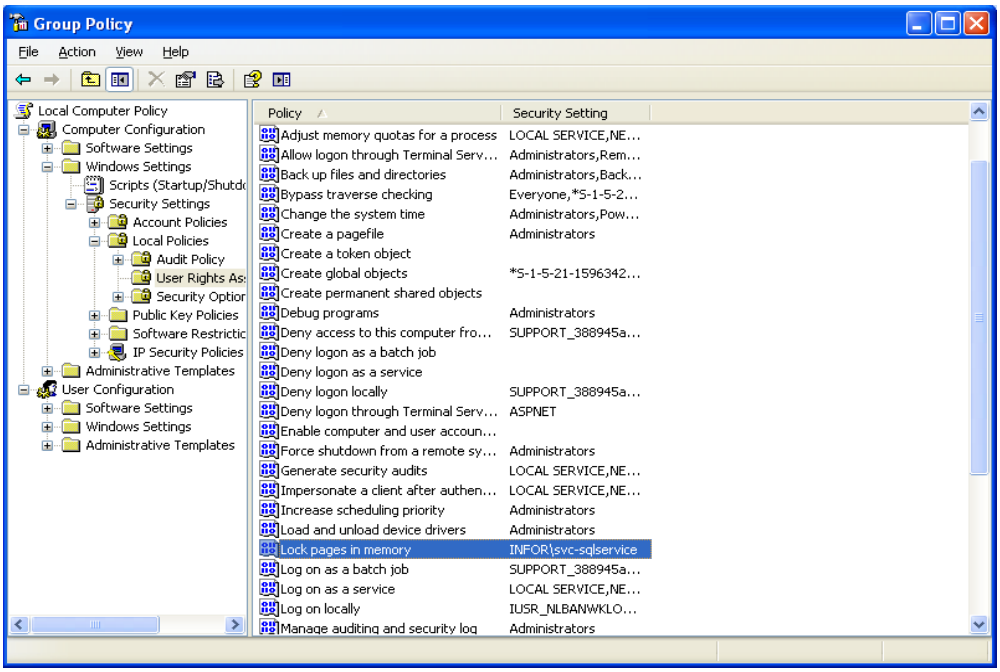
Use the **recovery interval** option to set the maximum number of minutes per database that SQL Server requires to recover databases. Each time an instance of SQL Server starts, each database is recovered, transactions that did not commit are rolled back, and transactions that are committed but the changes are not yet written to the disk when the SQL Server stopped are rolled forward. This configuration option sets an upper limit on the time it takes to recover each database. The default is 0, indicating an automatic configuration by the SQL Server, which means a recovery time of less than one minute and a checkpoint approximately every one minute for the active databases.

Retain the **recovery interval** set to 0 (self-configuring) unless you notice that the checkpoints hinder performance because of frequent occurrence. If so, it is recommended that you increase the value in small increments. Increasing the recovery interval results in “heavier” checkpoints and creates additional pressure on the I/O subsystem.

# Lock Pages in memory

Memory pages are locked in memory to enable faster allocations. SQL Server memory pages are non-pageable when locked in memory. Windows cannot use these pages when aggressively trimming the memory. To assign the “Lock pages in memory” privilege you must modify the Group Policy on the server with group policy editor (start with gpedit.msc).

Navigate to the User-Rights Assignment folder and add the (service) account to the ”Lock pages in memory” privilege.




# Large Pages

When using a larger page size for memory as organized by the kernel, the process of virtual address translation is faster. The normal page size for Windows memory is 4Kb on x64 systems. When using large pages, the page size is 2Mb.

SQL Server supports the concept of Large Pages when allocating memory for some internal structures and the buffer pool. But, the use of large pages for the buffer pool is not recommended for everyone and the usage must be tested and planned.

Large Pages are defined at two locations within the SQL Server:

- **LargePagesAllocator:** The SQL Server logic is used to decide if the Large Page Support from VirtualAlloc must be utilized.
- **Large Pages Used by Buffer Pool:** The Buffer Pool logic is used to decide if LargePageAllocator must be used to allocate the buffer pool memory.

## LargePagesAllocator

When the SQL Server is started for the first time, a decision on using large page support from Windows is made. This decision is based on the following three conditions, which must be set to true, when:

- using SQL Server Enterprise Edition
- the system has at least 8 GB physical RAM available after the boot process
- the “Lock Pages in Memory” privilege is set for the SQL Server service account

If these conditions are true, the SQL Server starts the “initialize” process for the LargePageAllocator for each memory node on the computer. The ERRORLOG displays the following messages:

```
2009-06-04 12:21:08.16 Server      Large Page Extensions enabled.
2009-06-04 12:21:08.16 Server      Large Page Granularity: 2097152
2009-06-04 12:21:08.21 Server      Large Page Allocated: 32MB
```

The Large Page Granularity is the minimum size of a “large page” on a specific Windows Platform. The next message indicates the purpose the LargePageAllocator for SQL Server is initialized. Using this option a 32Mb of large page memory is allocated to prime the system for any component that requires large pages memory. One of these messages for each memory node (NUMA) created by SQL Server are displayed.

## Large Pages

Large Pages are enabled by the trace flag 834. When enabled for 64-bit systems, the SQL Server Engine uses the LargePageAllocator to allocate SQL Server memory. From SQL Server 2012 onwards, large pages are allocated for all SQL Server memory. In previous versions, SQL Server allocated large pages only for the buffer pool.

If the engine detects the trace flag 834 is enabled when the database is started, memory is allocated using the LargePageAllocator. The server allocates all the memory when the database is started. This is because the allocation of large page with VirtualAlloc() can be very slow. If the buffer pool caching increases dynamically, this impacts the performance of the standard queries. The memory allocation is a one-time activity.

If these conditions are true, the SQL Server starts the “initialize” process for the Large Pages for each memory node on the computer. The ERRORLOG displays the following messages (SQL Server 2008 R2):

```
2009-06-04 14:20:31.13 Server      Large Page Extensions enabled.
2009-06-04 14:20:31.13 Server      Large Page Granularity: 2097152
2009-06-04 14:20:31.14 Server      Large Page Allocated: 32MB
2009-06-04 14:20:40.03 Server      Using large pages for buffer pool.
2009-06-04 14:27:56.98 Server      2048 MB of large page memory allocated.
```

The SQL Server attempts to allocate the size equal to the minimum of the **max server memory** setting and total physical memory on the computer. If **max server memory** is set to 0, SQL Server attempts to allocate the total RAM. Therefore, a suitable value for **max server memory** must be defined. The use of total physical RAM to allow the operating system and system cache to function is not recommended.

Use trace flag 834 in combination with Infor LN, if the following conditions are true, when using:

- Windows 2008 R2, or higher.
- SQL Server on a dedicated database server or settings ensure that sufficient memory is available for the SQL Server
- a heavily loaded Infor LN SQL Server database instance with memory usage above 16 Gb
- a NUMA based system with 2 CPU sockets or more

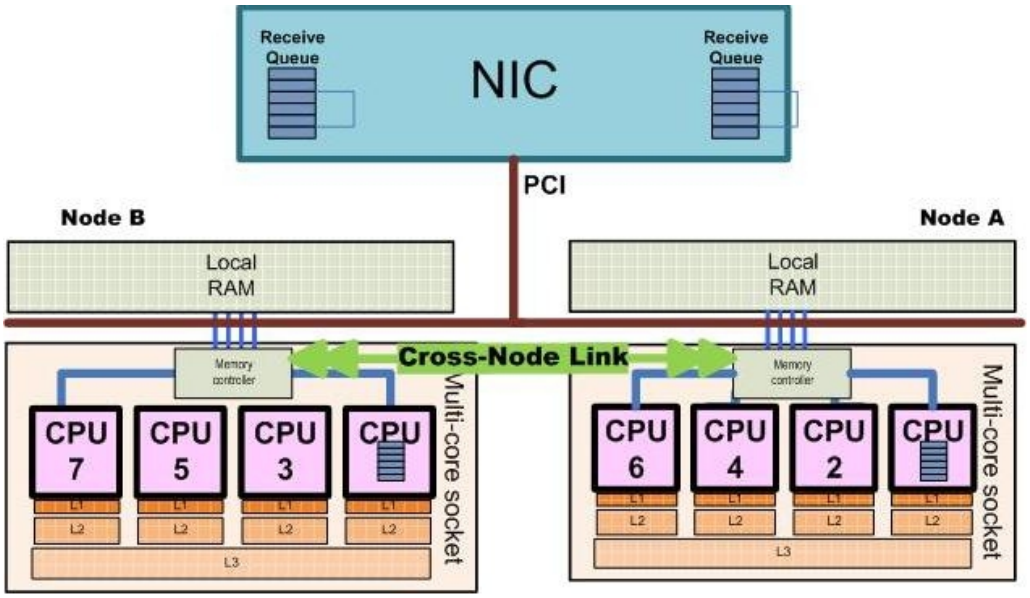
It is difficult to provide accurate figures and estimates for the increased performance levels when trace flag 834 is used. On an average, a performance increase in the range of 10 – 30 percent can be expected for heavily loaded SQL Server instances with Infor LN.

**Caution:** Do not over allocate memory for the SQL Server. When using the trace flag 834 ensure that the memory allocated by the SQL Server is at least 10% lower than the available system RAM. When additional memory is allocated by the SQL Server, the system might change to a “low memory” state, which triggers housekeeping activities such as trashing the plan cache. Therefore, it is recommended to carefully monitor the SQL Server **plan cache hit ratio** in perfmon, which must be > 95%.

# SQL Server and NUMA

SQL Server is designed to take advantage of NUMA-based computers without requiring any application changes. Earlier, NUMA was used by hardware vendors for “big iron” to resolve SMP-based (Symmetric Multi-Processor) scaling issues. Currently, all the latest CPUs include the interconnect technology which makes all systems NUMA based computers.

The diagram shows the NUMA architecture in detail:



In principle, a NUMA based system is split into nodes, with a NUMA-node for each Multi-core socket. This means that the system memory is also divided across the nodes. Local memory can be quickly accessed but memory on other node(s) must be accessed using the cross-node link with more latency.

There are various tuning recommendations for the SQL Server and NUMA, especially for the high-end usage. However, these are the basic recommendations:

- If the customer requirement is small, do not tune the database to prevent additional complexity.
- For Infor LN running in a 2-Tier configuration, assign the SQL Server instance to a dedicated NUMA node using the **Affinity mask** setting. SQL Server memory access on the other node(s) is always more expensive, and if the CPU capacity of all NUMA-nodes is not required, you can prevent expensive cross-node access.
- SQL Server memory (set by max server memory) is divided across the available NUMA nodes. On most systems, there is no (partial) memory sharing between NUMA nodes. This means additional total memory is required for the SQL Server to meet the per NUMA node memory requirements, when compared with a non-NUMA system.

## Plan cache size

Infor LN benchmarking of the SQL Server 2012 showed that SQL Server continually compiled query plans. This must occur during a period after a cold start of the database. Compiling SQL plans is an expensive CPU operation. In the benchmark, this occurred when more than 500 users accessed the database. If you encounter this issue on your SQL Server 2012 database, it is possible to implement the SQL Server trace flag 8032, which uses the SQL Server 2005 cache behavior and in general allows cache size to be larger. See http://msdn.microsoft.com/en-us/library/ms188396.aspx.

# Chapter 2 Tuning Infor LN for SQL Server

2

This chapter describes the important resource settings and performance parameters in Infor LN running on a SQL Server database.

## SQL Server db_resource parameters

From porting set 8.6a onwards, the default db_resource values are optimal for most customers. However, the following parameters are important for performance and in certain circumstances can be changed. For a 2-tier OLTP and batches (both 2-tier and 3-tier), some db_resource parameters increase the performance. See *Infor Enterprise Server Technical Reference Guide for SQL Server Database Driver (U8173 US).*

The following parameter must be added to the %BSE%/lib/defaults/db_resource file when the SQL Server is used as a database for the OLTP users:

```
msql_serverhost:<dbserver>\inforln
```

For batches, 2-tier and 3-tier, a separate db_resource file can be used to increase the performance of batches, without affecting the OLTP performance. This can be done using the environment variable USR_DBS_RES that must be set only for the user who executes the batches.

Example:

```
-set USR_DBS_RES=lib/defaults/db_resource.batch
```

The following parameters must be set for batches:

```
msql_opt_rows:50
msql_serverhost:<dbserver>\inforln
msql_max_arrsz:50
msql_retained_cursors:200
bdb_max_session_schedule:250          #See U9357, chapter 4 Tuning Infor LN
```

### msql_opt_rows

The resource msql_opt_rows allows you to specify a hint that the first **n** rows must be retrieved faster from the database. This allows the SQL Server to optimize the fetch request. Based on this value, the SQL Server determines a suitable communication buffer size to improve performance. By default, the value of this parameter is equal to the array fetch size.

## msql_retained_cursors

The resource msql_retained_cursors sets the number of inactive (breaked) cursors that must be retained in the list for reuse. These cursors can be reused and prevent a prepare/bind overhead. Additional resources, such as, memory until the cursors are closed and released are required.

The Default value is 20, which is a good starting point for most customers in a 2-tier OLTP. For batches, it is recommended to increase this parameter to 200.

## msql_serverhost

Use the msql_serverhost resource to specify a host name for the driver to locate the SQL Server instance that must be used. It is possible to specify a network protocol and a SQL Server Instance:

```
msql_serverhost:<dbserver>\inforln
```

In this example “dbserver\inforln” specifies the use of “inforln” SQL Server Instance on the server “dbserver”. See “Tuning SQL Server for Infor LN” on page 7 for optimal SQL Server network settings.

## msql_array_fetch

The resource msql_array_fetch is used to enable or disable the array fetch interface. The valid values are 0 and 1. When set to 0, the array fetch interface is disabled; when set to 1, the interface is enabled. The default value for this option is 1.

## msql_array_insert

The resource msql_array_insert is used to enable or disable the array insert interface; the valid values are 0 and 1. When set to 0, the array insert interface is disabled; when set to 1, the interface is enabled. The default value for this option is 0.

**Note:** The database driver disables this option under specific circumstances. For example, if references must be checked or updated, or the application requires an immediate response from the driver, no array insert can be executed.

## msql_max_arrsz

If msql_array_insert is enabled, the msql_max_arrsz parameter is used to define the maximum number of rows inserted to the RDBMS. If msql_array_fetch is enabled, the msql_max_arrsz

parameter is used to define the maximum number of rows fetched from the RDBMS. The default value for this option is 5.

## msql_lock_timeout

The resource msql_lock_timeout parameter is used to determine the timeout value, in seconds, for queries, blocked by locks in the database server. The default value is 10, which means “one waits for 10 seconds for the lock to be released”. When set to -1, the driver waits indefinitely for the locks to be released; when set to 0, the driver does not wait for the locks to be released. It is recommended to use this resource carefully. When batches are executed during normal working hours, and in some high concurrent OLTP workloads, it can be beneficial to increase the timeout to 60 seconds. The transactions wait longer to acquire the lock rather than re-executing the query from the retry point.

## msql_no_index_hint

By default, the Infor LN database driver generates query hints to direct the SQL Server query optimizer with regards to the index that must be used. Based on the benchmarking, it can be beneficial to disable these index hints and allow the SQL Server optimizer to choose the optimal index to be used. To disable the index hint generation, set the msql_no_index_hint resource to 1.

**Caution:** For new customers, this setting is recommended. Existing customers must test this resource in a test environment by executing the critical business processes as the optimizer can make decisions that are not preferred, resulting in a performance decrease for specific sessions.

# Chapter 3 Troubleshooting SQL Server

3

Besides the Infor LN troubleshooting possibilities, SQL Server in combination with Windows includes powerful tools to troubleshoot for an application. A brief summary is provided. It is difficult to recommend the tool to be used in a specific situation. However, it is recommended that you start with a global tool (as Windows Task Manager) and start the detailed tracing based on symptoms identified using the global tools. See http://msdn.microsoft.com/en-us/library/ms179428.aspx, for more information on SQL Server performance tools.

## Windows Task Manager

The Windows Task Manager is recommended as a starting point for any tracing and tuning exercise. Do not use other “advanced” tools as the root cause is displayed in the Task manager.

## SQL Server Management Studio Activity monitor

The Activity Monitor available in the SQL Server Management Studio can be used by database developers and administrators for an overview of the SQL Server system performance.

See http://msdn.microsoft.com/en-us/library/cc879320.aspx for more information.

## SQL Server Management Studio Reports

SQL Server Management Studio generates built-in performance reports at both the server level and the database level. Right-click the Infor LN database icon in the Object Explorer pane and select Reports > Standard Reports to access the reports. Most reports are based on dynamic management views (DMVs). Apart from these standard reports, Microsoft provides performance dashboard reports, which can be downloaded from the Microsoft website. The performance dashboard reports allow a database administrator to quickly identify if there is a bottleneck on their system, and then capture additional diagnostic data that is required to resolve the problem.

# SQL Server Dynamic Management Views (DMV)

The DMVs are used to display the internal memory structures within the SQL Server address space. DMVs and dynamic management view functions (DMFs) reflect the server processes or all the sessions on the server. DMVs can be used for diagnostics, memory and process tuning, and monitoring the sessions on the server. DMVs alleviate the requirement to monitor the server with tools, such as Profiler and Performance Monitor (Perfmon), to diagnose performance issues. Additionally, DMVs can provide Information about performance issues, even after the issues occur. See http://msdn.microsoft.com/en-us/library/ms188754.aspx.

# Tracing with sqlcmd

Usually the graphical plan is easy to use but in some scenarios it is beneficial to use the command prompt utilities. The SQL query can be executed on the command prompt with the “sqlcmd” utility. The sqlcmd -? option is used to display the options available for the sqlcmd. See http://msdn.microsoft.com/en-us/library/ms162773.aspx and http://msdn.microsoft.com/en-us/library/hh213540.aspx.

These options provide useful information when analyzing query performance:

- SET_SHOWPLAN_TEXT {ON|OFF}. When set to ON, the query execution plan information is displayed. The query is not executed. Example:

```
sqlcmd -S <dbserver>\inforln -d inforlndb -U sa -P <password>
1> SET SHOWPLAN_TEXT ON
2> GO
1> select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
2> GO
StmtText
------------------------------------------------------------------------
 select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"

(1 rows affected)
StmtText
------------------------------------------------------------------------
 |--Nested Loops(Inner Join, OUTER REFERENCES:([inforlndb].[dbo].[ttcibd001090
   |--Index Seek(OBJECT:([inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_2a]),
   |--Clustered Index Seek(OBJECT:([inforlndb].[dbo].[ttcibd001090].[Ittcibd00

(3 rows affected)
```

  The example does not display the complete result set because due to wrapping the result set is unreadable. To prevent wrapping, it is recommended that you expand the command prompt window.

- SET_SHOWPLAN_ALL {ON|OFF}. When set to ON, the query execution plan information is displayed. The option is used to return information as a set of rows that form a hierarchical tree representing the steps executed by the SQL Server query processor, for the processing of each statement. Each statement displayed in the output contains a single row with the text of the

statement, followed by several rows with the details of the execution steps. The query is not executed. Example:

```
sqlcmd -S <dbserver>\inforln -d inforlndb -U sa -P <password>
1> set SHOWPLAN_ALL ON
2> go
1>  select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
2> go
StmtText
                                                                    DefinedValues
------------------------------------------------------------------------------------
--------
------------------------------------------------------------------ -----------------
--------
 select t_item, t_dsca from dbo.ttcibd001090 where t_seak="COST SET 1"
                                                                    NULL
  |--Nested Loops(Inner Join, OUTER REFERENCES:([
inforlndb].[dbo].[ttcibd001090].[t_item], [Ex
                                                                    NULL
       |--Index Seek(OBJECT:([ inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_2a]),
SEEK:([inforlndb
]=N'COST SET 1') ORDERED FORWARD
[inforlndb].[dbo].[ttcibd001
       |--Clustered Index Seek(OBJECT:([
inforlndb].[dbo].[ttcibd001090].[Ittcibd001090_1a]), S
]=[ inforlndb].[dbo].[ttcibd001090].[t_item]) LOOKUP ORDERED FORWARD
[inforlndb].[dbo].[ttcibd001

(4 rows affected)
```

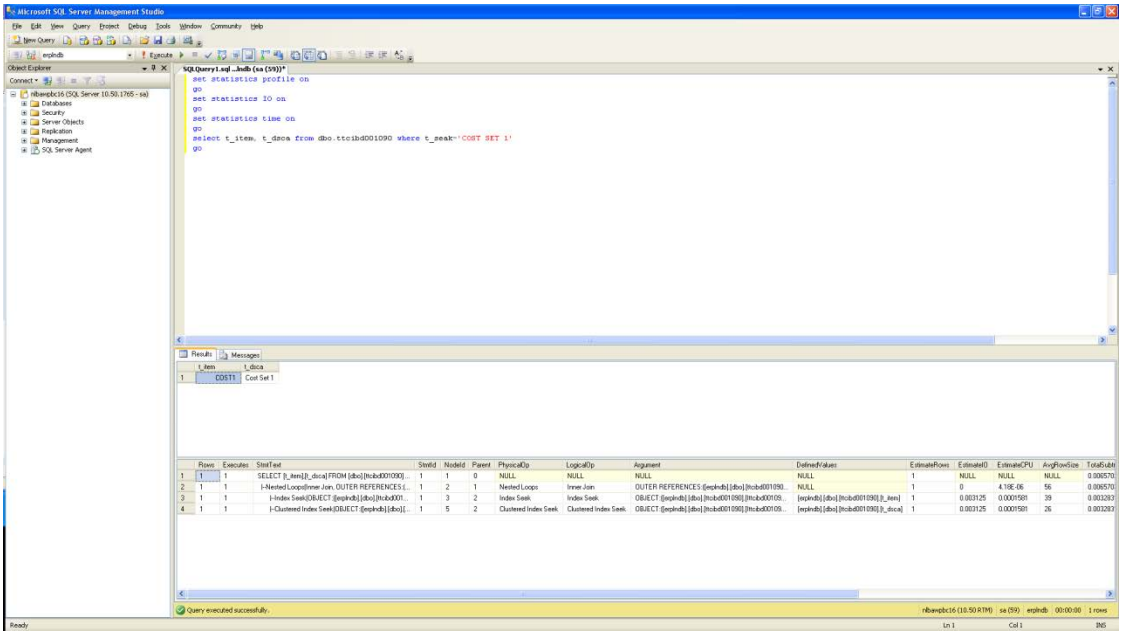
- SET STATISTICS PROFILE {ON|OFF} does not prevent the execution of the query. Apart from returning the same Information as SET SHOWPLAN_ALL, the query also displays two additional columns: Rows and executes. “Rows” contains the numbers of rows returned, and “Executed” contains the actual number of times the operator executed the query.
- SET STATISTICS IO {ON|OFF} displays the count of table accesses, logical and physical reads, and read ahead reads for each T-SQL statements.
- SET STATISTICS TIME {ON|OFF} displays the time in milliseconds to parse the statement, compile the query, and execute the query.

See http://msdn.microsoft.com/en-us/library/ms190356.aspx for more information about profiling options.

# SQL Server management studio

One of the best methods to trace the SQL Server is to use the SQL Server Management Studio.

A useful feature of the SQL Server management studio is when you can run a query in “sqlcmd mode” the same output as the “sqlcmd utility” is displayed. This ensures that the Information is better formatted:




The tab messages display the additional statistics Information:

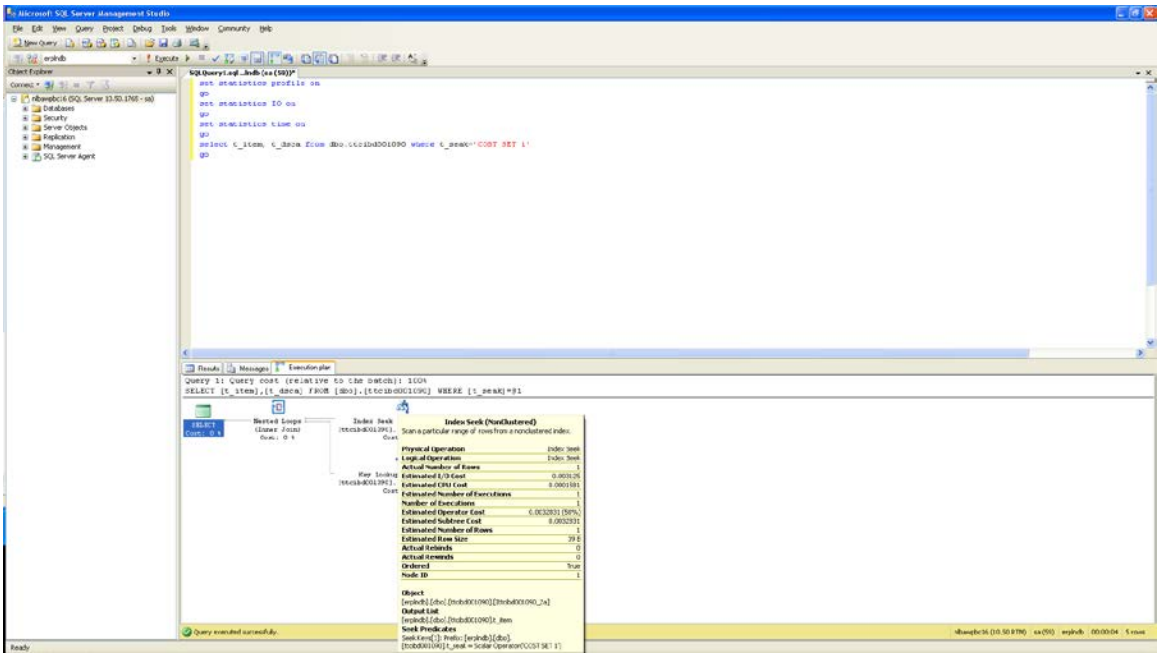
```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 1 ms.

(1 row(s) affected)
Table 'ttcibd001090'. Scan count 1, logical reads 7, physical reads 0, read-ahead
reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(4 row(s) affected)

SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 1 ms.
```

Switching the “Include Actual Execution Plan”, under the “query” tab, displays the graphical query execution plan. This diagram shows an example of the trace output in SQL Server Management Studio:



# Management Data Warehouse (MDW)

The Performance Data Collection and Warehouse can be used by database administrators to gather performance related data using the built-in data collectors which are also known as data collection containers. Using data collectors, you can gather performance data from multiple SQL Servers and store the data within a Management Data Warehouse (MDW). When you successfully configure the Management Data Warehouse (MDW) with the **Disk Space**, **Query Statistics,** and **Server Activity** Data Collection Sets, multiple built in reports can be generated for analysis. These reports can be executed from within the SQL Server Management Studio (SSMS).See http://msdn.microsoft.com/en-us/library/bb677179.aspx.

# SQL Server Profiler

The SQL Server Profiler is a powerful SQL Server tracing tool. Creating a SQL Server Profiler snapshot of a badly performing query or multiple statements helps you identify the reason for the performance issue. Infor Xtreme solution 22881401 provides detailed steps and a set of trace templates for using the SQL Server Profiler. A powerful feature of the SQL Server Profiler is the option to correlate a Profiler trace with Performance monitor (perfmon) data. The impact of performance on the database server correlates directly to the Profiler trace and vice versa. See http://msdn.microsoft.com/en-us/library/ms191152.aspx.

# SQL Server Extended Events

When compared to the SQL Server profiler, Extended Events is a light weight server-side performance monitoring system. Events in SQL Server can be captured using the SQL Server Management Studio (SSMS) management node. There are events for CPU, Memory, disk, database, cursor usage, locks, latches and so on. Results can be stored in a file or table. See http://msdn.microsoft.com/en-us/library/bb630282.aspx.

# Troubleshooting with SQLDiag utility

SQLDiag is a diagnostics collection utility that can run either from the command prompt or as a service. SQLDiag can be used to gather the following Information:

- Windows Performance logs
- Windows Event logs
- SQL Server Profiler traces
- SQL Server blocking Information
- SQL Server Configuration Information

To analyze the SQLDIAG output, you can also use the PAL utility. See http://msdn.microsoft.com/en-us/library/ms162833.aspx for more information about SQLDiag.

# Troubleshooting with PAL Tool

When you have a performance issue and are not sure which performance counters to gather or how to analyze the same, you must use the PAL tool (Performance Analysis of Logs). It is a powerful tool that reads and analyzes a performance monitor counter log using the known thresholds. Features:

- Includes thresholds for most of the major Microsoft products such as IIS, SQL Server, and so on.
- An easy to use GUI interface which enables creating batch files for the PAL.ps1 script.
- A GUI editor for creating or editing your own threshold files.
- Creates an HTML based report which is easy to copy/paste into other applications.
- Analyzes performance counter logs for thresholds using thresholds that change the criteria based on the computer's role or hardware specification.

See https://pal.codeplex.com.

# Performance monitor

To monitor the utilization of system resources, use the Performance Monitor (perfmon). Collect and view real-time performance data in the form of counters, for server resources such as processor and memory use, and for many SQL Server resources such as locks and transactions.

The most useful SQL Server resources in the Performance monitor are:

- SQLServer: Access Methods.
- SQLServer: Buffer Manager.
- SQLServer: Cursor Manager by Type.
- SQLServer: Cursor Manager Total.
- SQLServer: Databases.
- SQLServer: General Statistics.
- SQLServer: Latches.
- SQLServer: Locks.
- SQLServer: Memory Manager.
- SQLServer: SQL Statistics.
- SQLServer: Wait Statistics.

# Server-Side Traces

Server-Side tracing is tracing using stored procedures. The SQL Profiler is a powerful tool, but the graphical UI that displays all the captured events utilizes most of the CPU resources. For long running traces on loaded SQL Servers systems, Server-Side traces are a good solution.

The stored procedures used for these tasks are documented. However, the easiest way to create scripts that utilize these traces is to create a trace using the SQL Profiler graphical UI, and on the File menu, select Export.

The system stored procedures to be used when creating and managing a trace:

- sp_trace_create
- sp_trace_generateevent
- sp_trace_setevent
- sp_trace_setfilter
- sp_trace_setstatus

# SQL Nexus

SQL Nexus is a tool that helps you identify the root cause of SQL Server performance issues. It loads and analyzes performance data collected by SQLDiag. It can significantly reduce the amount of time you spend to analyze the data manually.

Before working with the SQL Nexus tool, it is recommended that you gain knowledge of other performance topics and tools of the SQL Server.

SQL Nexus features:

- Fast, easy data loading: You can quickly and easily load SQL Trace files; T-SQL script output, including SQL DMV queries; and Performance Monitor logs into a SQL Server database for analysis. All three facilities use bulk load APIs to insert data quickly. You can also create your own importer for a custom file type.
- View loaded data via reports: After the data is loaded, you can generate several different charts and reports to analyze the data.
- Trace aggregation to display the TOP N most expensive queries (using ReadTrace)
- Wait stats analysis to view blocking and other resource contention issues (based on SQL Perf Stats)
- Full-featured reporting engine: SQL Nexus uses the SQL Server Reporting Services (RS) client-side report viewer (this does not require an RS instance). You can create reports for Nexus from either the RS report designer or the Visual Studio report designer. You can also modify the reports that are included with Nexus using either facility. Zoom in/Zoom out to view server performance during a particular time window. Expand/collapse report regions (subreports) for easier navigation of complex data. Export or e-mail reports directly from SQL Nexus. Nexus supports exporting to Excel, PDF, and several other formats.
- Extensibility: You can use the existing importers to load the output from any DMV query into a table, and any RS reports you add to the Reports folder are automatically displayed in the reports task pane. If required, you can even add a new data importer for a new data type. SQL Nexus can automatically “fix up” the database references in your reports to reference the current server and database, and provide generic parameter prompting for any parameters your reports support.

See http://sqlnexus.codeplex.com.