Dynamics NAV/SQL Server Configuration Recommendations

This document describes SQL Server configuration recommendations that were gathered from field experience with Microsoft Dynamics NAV and SQL Server. This information applies to Microsoft Dynamics NAV executable versions 5.00 SP1 and later running on Microsoft SQL Server 2005 SP3 x64 or Microsoft SQL Server 2008 SP1 x64 or Microsoft SQL Server
2008 R2 x64

SQL Server 2008 SP1 / 2008 R2 or later

"Max Server Memory"

MAX_SERVER_MEMORY= TOTAL_SERVER_MEMORY - [OS/Apps Memory] + [Threads * Thread Size]

TOTAL_SERVER_MEMORY = 32GB
OS/Apps Memory = 2 GB to 4 GB (I use 2GB on system with <= 16GB of RAM and 4GB on systems with > 16GB of RAM)
Threads = select max_workers_count from sys.dm_os_sys_info
Thread Size =

Platform	**Size**
32 bit	512k
x64	2MB
IA64	4MB

Example: x64 SQL Server with 32GB of RAM

MAX_SERVER_MEMORY = (32GB - (4GB + (255 * 2MB)))

MAX_SERVER_MEMORY = (32GB - (4GB + (510MB)))

MAX_SERVER_MEMORY = (32GB - (4.5GB))

MAX_SERVER_MEMORY = (27.5GB)

```
sp_configure 'max server memory', 27500
RECONFIGURE
GO
```

"Auto - Create Statistics"

We strongly recommend that you enable "Auto Create Statistics" as this is one of the ways that SQL Server tunes itself for better performance. The overhead to run this is minimal compared to the performance issue you may experience if a necessary statistic is missing and SQL Server cannot create it. Statistics are a very important factor in SQL coming up with an efficient query plan.

"Auto - Update Statistics"

We strongly recommend that you enable "Auto Update Statistics" as this is one of the ways that SQL Server tunes itself for better performance. The overhead to run this is minimal compared to the performance issue you may experience if a statistic is out of date and SQL Server uses it to compile the query and create the execution plan. Statistics are a very important factor in SQL coming up with an efficient query plan. This is very important with C/AL FINDFIRST, FINDLAST, FIND('-'), FIND('+'), and FINDSET for [Entry No.], because if the statistic on [Entry No.] is out of date it will take SQL longer to find the first or last value meaning it will lock the resource longer having a negative impact on system concurrency and possibly resulting in Lock Time Outs.

"Auto-Grow"

Auto-Growths of the Database or Transaction Log in production can degrade performance as all transaction must queue up and wait for the SQL to Grow the Log or Data file before it can begin to process transactions again. This can create a bottleneck. We strongly recommend growing the Data and Log file during off peak periods and by 10 to 25% of the current size. We DO NOT recommend disabling "Auto-Grow" as in an emergency it is still better to have SQL Auto Grow the files than it is to run out of space and bring database down.

Database Compatibility Level (OPTIMZIE FOR UNKNOWN)

If you are running SQL Server 2008 (2008R2) and Microsoft Dynamics NAV 2009 SP1 the SQL Database Compatibility Level must be set to 100 in order for Microsoft Dynamics NAV to use the OPTIMZIE FOR UNKNOWN query hint to help diffuse parameter sniffing. Generally we recommend that the SQL Database Compatibility level be set to whatever version of SQL you are running for the Microsoft Dynamics NAV Database.

Trace Flag 4136

If you are running the Microsoft Dynamics NAV 2009 SP1 executables and the database compatibility level is set to 100 then you do not need Trace Flag 4136 and it will be ignored by any query using the OPTIMIZE FOR UNKNOWN hint (they both essentially do the same thing). If you are not using the Microsoft Dynamics NAV 2009 SP1 executables I would recommend enabling this trace flag. Also if you have other databases on the server and or other applications access the Microsoft Dynamics NAV database that do not use OPTIMIZE FOR UNKNOWN you may still wish to enable this trace flag to help prevent "parameter sniffing" for those databases and or applications.

http://support.microsoft.com/kb/980653
*Verify that you have the correct SQL Server service pack and cumulative update level to utilize this feature.

TempDB

The number of Data Files for TempDB should be equal to the number of CPU cores utilized by SQL Server up to a maximum of 8. This prevents bottlenecks at the SGAM page allocation process when new Temp objects are created. The files for TempDB should be on a drive separate from the SQL Data and Log files.

We recommend splitting data files for TempDB because each temporary object that is created in TempDB needs pages allocated for it. These pages must be allocated via the SGAM page and there is only one SGAM page per data file (not really but close enough). So if you have only one data file and you are creating a lot of temp objects such as is possible with SSRS and RCSI (or other *database on the server as they all use the same TempDB) you can bottleneck on the SGAM page because it is trying to handle too many allocations at the same time. That is why there is a recommendation to have 1 TempDB data file per CPU core or active thread to alleviate the possibility of an allocation bottleneck. Granted this is rare but it does happen.*

NOTE: Each TempDB file DOES NOT need to be on its own drive they just need to be separated from all SQL Data and Log files

Disk Alignment

Disk alignment can have a significant impact on disk performance as one block can span 2 physical disks. Since proper configuration can vary greatly by manufacturer please consult your hardware manufacturing to verify the proper disk alignment and offset settings. The allocation or stripe size should be >=8k to prevent torn pages we recommend 64k to match the size of the SQL Server Extent (8 – 8K pages).

To check disk alignment

1. Click "Start" - "Run" and type MSINFO32
2. Click "Components" - "Storage" - "Disks"
3. Look at the value for "Partition Starting Offset" for the drive you wish to check

"Partition Starting Offset" / (64 * 1024) = a whole number then the disks are aligned for that drive
"Partition Starting Offset" / (64 * 1024) = a decimal number then the disks are not aligned for that drive

Example

Partition Starting Offset = 117,020,033,024 bytes

117,020,033,024 / 65,536 = 1,785,584 (a whole number so the disk is aligned

117,020,033,153 / 65,536 = 1,785,584.001968384 (a decimal number so disk is not aligned)

NOTE: All drives formatted via Window Server 2008 and later automatically will align the disks.

Read Committed Snapshot Isolation (RCSI)

With the release of SQL Server 2005 a new isolation level was introduced called Read Committed Snapshot. This isolation level mimics the “Level 1” Read Committed which is the SQL Server default, but allows for greater concurrency and less blocking as it uses a versioning principle for locked records. Readers and writers will receive different “versions” of the record so they do not block each while maintaining transactional consistency. These row versions are help in the Version Store which is kept in TempDB. From a Microsoft Dynamics NAV application perspective this is not very useful as NAV sends isolation level hints with most statements which will override RCSI, but will have a benefit when outside applications access records within the NAV production database such as reports, views, and secondary applications. (SSRS, ETL processes, etc…)

```
ALTER DATABASE DatabaseName
SET READ_COMMITTED_SNAPSHOT ON
GO
```

NOTE: TempDB must be configured per recommendations or page allocation bottlenecks could occur at the TempDB level.

"Max Degree of Parallelism"

Microsoft Dynamics NAV communicates to SQL Server with client side cursors via ODBC. The C/AL is converted to simple TSQL statements via the NDBCS.DLL. These statements are very simple Select, Update, Insert, and Delete SQL statements and do not require parallel execution in SQL Server. This does not stop the compiler from sometimes attempting to use parallel execution when planning queries and this can sometimes negatively impact performance. This is why I recommend setting the - Max degree of Parallelism in SQL Server to 1 to prevent this.

```
sp_configure 'max degree of parallelism', 1
RECONFIGURE
GO
```

NOTE: If you are running SQL Enterprise Edition Maintenance Jobs will be multi-threaded, but if you set "Max Degree of Parallelism" to 1 they will be executed single threaded. We recommend that in the Maintenance Job you flip "Max Degree of Parallelism" back to 0 run the Maintenance Job and then flip it back to 1. This can be done with TSQL code inside the Job.

Microsoft Dynamics NAV Default Isolation Level

With the release of the Microsoft Dynamics NAV 5.00 SP1 (Build 30482 / KB 979135) and later executables we now have the option to change NAV default SQL isolation level from SERIALIZABLE to REPEATABLEREAD.

SERIALIZABLE

This is the most restrictive isolation level. When it's used, the phantom values cannot occur. It prevents other users from updating or inserting rows into the **data set** until the transaction will be completed.

REPEATABLEREAD

When it's used, the dirty reads and non-repeatable reads cannot occur. It means that locks will be placed on all **data** that is used in a query, and another transaction cannot update the data.

In summary SERIALIZABLE is more restrictive and locks more records that REPEATABLEREAD which can cause more Lock Timeouts and Deadlocks. REPEATABLEREAD is less restrictive and allows for higher overall concurrency.

Setting the Dynamics NAV default isolation level from SERIALIZABLE to REPEATABLEREAD

```
UPDATE [$ndo$dbproperty] SET diagnostics = 4194304
```

Microsoft Dynamics NAV "Lock Timeout"

This setting essentially means that if a user attempts to access a resource and it is locked by another user it will wait however many seconds you have specified and try again before issuing an error message. We recommend this setting not to be used at all or to be set to between 10 – 20 seconds, having a very high value for the static lock time-out can cause its own issues. Usually when a customer has a high value set for this is because there is an underlying performance issue. This is really up to the discretion of the Customer and Partner and can be set higher and there are specific circumstances where this does need to be set higher. This setting can be found under [File] – [Database] – [Alter] on the “Advanced” tab.

Dynamics NAV "Always Rowlock"

The Microsoft Dynamics NAV “Always” rowlock setting was a holdover from SQL 2000 and 32 bit architecture. In 99% of the cases on SQL 2005 and later this can be disabled which will free up lock memory that SQL can use for other purposes without any loss of performance. This setting can be found under [File] – [Database] – [Alter] on the “Advanced” tab.

NOTE: In some benchmarks not having this setting enabled actually increased concurrency and performance.

Maintenance Jobs

Statistics

Even with "Auto Update Statistics" enabled we still strongly recommend running a periodic SQL Maintenance Job to update statistics. This is because "Auto Update Statistics" will only be triggered when 5% of the data in that statistic has changed and on tables such as Value Entry, Item Ledger Entry, G/L Entry, and such can have 10s of millions of records a small percentage of data in a given statistic such as [Entry No.] can change and have a material effect on the overall data distribution in that statistic causing inefficient query plans long before the 5% threshold is reached resulting in degraded performance. We recommend the use of "sp_updatestats" to accomplish this task as it will only update the statistics where data has been changed. We would recommend creating a SQL Job that runs daily or weekly (depending on transaction volume) during off peak hours to update all statistics where data has changed.

NOTE: Do not use "FULLSCAN" as this is a waste of time as it will be overwritten the second the statistic is "auto updated."

Index defragmentation

Indexes become fragmented just like hard drives and can have a negative impact on overall Performance. We recommend using the following script which use the REBUILD and REORGANIZE functions for index defragmentation. We would recommend creating a SQL Job that runs daily or weekly (depending on transaction volume) during off peak hours to update all indexes.

```
SET NOCOUNT ON;
DECLARE @objectid int;
DECLARE @indexid int;
DECLARE @partitioncount bigint;
DECLARE @schemaname sysname;
DECLARE @objectname sysname;
DECLARE @indexname sysname;
DECLARE @partitionnum bigint;
DECLARE @partitions bigint;
DECLARE @frag float;
DECLARE @command varchar(8000);
DECLARE @percentage int

SET @percentage = 25  --You can change the precentage
threshhold here

IF EXISTS (SELECT name FROM sys.objects WHERE name =
'work_to_do')
    DROP TABLE work_to_do;

SELECT
    object_id AS objectid,
    index_id AS indexid,
    partition_number AS partitionnum,
    avg_fragmentation_in_percent AS frag
INTO work_to_do
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL,
NULL , NULL, 'DETAILED')
WHERE avg_fragmentation_in_percent > @percentage AND
index_id > 0;

DECLARE partitions CURSOR FOR SELECT * FROM
work_to_do;


OPEN partitions;


FETCH NEXT
   FROM partitions
   INTO @objectid, @indexid, @partitionnum, @frag;

WHILE @@FETCH_STATUS = 0
    BEGIN;
        SELECT @objectname = o.name, @schemaname =
s.name
        FROM sys.objects AS o
        JOIN sys.schemas as s ON s.schema_id =
o.schema_id
        WHERE o.object_id = @objectid;

        SELECT @indexname = name
        FROM sys.indexes
        WHERE  object_id = @objectid AND index_id =
@indexid;

        SELECT @partitioncount = count (*)
        FROM sys.partitions
        WHERE object_id = @objectid AND index_id =
@indexid;


IF @frag < (@percentage)
    BEGIN;
    SELECT @command = 'ALTER INDEX ' + '[' +
@indexname + ']' + ' ON ' + '[' + @schemaname + ']' +
'.' + '[' +@objectname + ']' + ' REORGANIZE';
    --IF @partitioncount > 1
    --      SELECT @command = @command + ' PARTITION='
+ CONVERT (CHAR, @partitionnum);
    EXEC (@command);
    END;

IF @frag >= (@percentage)
    BEGIN;
    SELECT @command = 'ALTER INDEX ' + '[' +
@indexname + ']' + ' ON ' + '[' + @schemaname + ']' +
'.' + '[' +@objectname + ']'  + ' REBUILD';
    --IF @partitioncount > 1
    --      SELECT @command = @command + ' PARTITION='
+ CONVERT (CHAR, @partitionnum);
    EXEC (@command);
    END;
PRINT 'Executed ' + @command;

FETCH NEXT FROM partitions INTO @objectid, @indexid,
@partitionnum, @frag;
END;

CLOSE partitions;
DEALLOCATE partitions;

IF EXISTS (SELECT name FROM sys.objects WHERE name =
'work_to_do')
    DROP TABLE work_to_do;
GO
```

NOTE: See "Max Degree of Parallelism"

Instant File initialization

In SQL Server 2005 and later versions, data files can be initialized instantaneously, allowing for fast execution of database or file group restore operations. Instant file initialization reclaims used disk space without filling that space with zeros. Instead, disk content is overwritten as new data is written to the files. Log file initialization still requires zeroing, but it will happen in parallel with the transfer of the data from the backup. The roll forward step of restore will not start until all of the data has been transferred and the whole log has been initialized.

To use instant file initialization, you must run the MSSQLSERVER service account under a Windows account and assign the Windows SE_MANAGE_VOLUME_NAME special privilege to that Windows account. This privilege is assigned to the Windows Administrators group by default. If you have system administrator rights, you can assign this privilege by adding the Windows account to the Perform Volume Maintenance Tasks security policy.

"Optimize for Ad-Hoc Workloads"

The “optimize for ad hoc workloads” option is used to improve the efficiency of the plan cache for workloads that contain many single use ad hoc batches. When this option is set to 1, the Database Engine stores a small compiled plan stub in the plan cache when a batch is compiled for the first time, instead of the full compiled plan. This helps to relieve memory pressure by not allowing the plan cache to become filled with compiled plans that are not reused. I have seen a few situations with Dynamics installs where the plan cache was "bloating" due to the number of single use plans causing memory pressure and degraded performance. This usually manifests itself in the customer reporting that server performance degrades over time and then they reboot the SQL Server the performance rebounds. In these cases enabling Optimize for Ad-Hoc Workloads can help. This is not a very common occurrence in the "NAV World" but it does happen. The choice of whether to enabling this setting is up to you.

```
sp_configure 'optimize for ad hoc workloads', 1
RECONFIGURE
GO
```

NOTE: This is a SQL Server 2008 option only

"Page Verify" - Checksum vs. Torn Page Detection

We recommend use the Database Page Verify option to Checksum as Check sum is far more robust at detecting physical database corruption.

```
ALTER DATABASE DatabaseName
SET PAGE_VERIFY CHECKSUM;
GO
```

Lock Pages in Memory

We do recommend turning on “Lock Pages in Memory” so that OS doesn't page SQL Server out. However on 64 bit you only need to grant the right "Lock Pages in Memory" to the SQL account for SQL Server to utilize this feature. You do need to change any of AWE settings through sp_configure.

http://technet.microsoft.com/en-us/library/ms190730.aspx

SQL Server 2005 SP3 and later

"Max Server Memory"

MAX_SERVER_MEMORY= TOTAL_SERVER_MEMORY - [OS/Apps Memory] - [Threads * Thread Size]

TOTAL_SERVER_MEMORY = 32GB
OS/Apps Memory = 2 GB to 4 GB (I use 2GB on system with <= 16GB of RAM and 4GB on systems with > 16GB of RAM)
Threads = select max_workers_count from sys.dm_os_sys_info
Thread Size

Platform	**Size**
32 bit	512k
x64	2MB
IA64	4MB

Example: x64 SQL Server with 32GB of RAM

MAX_SERVER_MEMORY = (32GB - (4GB + (255 * 2MB)))

MAX_SERVER_MEMORY = (32GB - (4GB + (510MB)))

MAX_SERVER_MEMORY = (32GB - (4.5GB))

MAX_SERVER_MEMORY = (27.5GB)

```
sp_configure 'max server memory', 27500
RECONFIGURE
GO
```

"Auto - Create Statistics"

We strongly recommend that you enable "Auto Create Statistics" as this is one of the ways that SQL Server tunes itself for better performance. The overhead to run this is minimal compared to the performance issue you may experience if a necessary statistic is missing and SQL Server cannot create it. Statistics are a very important factor in SQL coming up with an efficient query plan.

"Auto - Update Statistics"

We strongly recommend that you enable "Auto Update Statistics" as this is one of the ways that SQL Server tunes itself for better performance. The overhead to run this is minimal compared to the performance issue you may experience if a statistic is out of date and SQL Server uses it to compile the query and create the execution plan. Statistics are a very important factor in SQL coming up with an efficient query plan. This is very important with C/AL FINDFIRST, FINDLAST, FIND('-'), FIND('+'), and FINDSET for [Entry No.] , because if the statistic on [Entry No.] is out of date it will take SQL longer to find the first or last value meaning it will lock the resource longer having a negative impact on system concurrency and possibly resulting in Lock Time Outs.

"Auto-Grow"

Auto-Growths of the Database or Transaction Log in production can degrade performance as all transaction must queue up and wait for the SQL to Grow the Log or Data file before it can begin to process transactions again. This can create a bottleneck. We strongly recommend growing the Data and Log file during off peak periods and by 10 to 25% of the current size. We DO NOT recommend disabling “Auto-Grow” as in an emergency it is still better to have SQL Auto Grow the files than it is to run out of space and bring database down.

Trace Flag 4136

If you are running the Microsoft Dynamics NAV 2009 SP1 executables and the database compatibility level is set to 100 then you do not need Trace Flag 4136 and it will be ignored by any query using the OPTIMIZE FOR UNKNOWN hint (they both essentially do the same thing). If you are not using the Microsoft Dynamics NAV 2009 SP1 executables I would recommend enabling this trace flag. Also if you have other databases on the server and or other applications that access the Microsoft Dynamics NAV database that do not use OPTIMIZE FOR UNKNOWN you may still wish to enable this trace flag to help prevent "parameter sniffing" for those databases and or applications.

http://support.microsoft.com/kb/980653
*Verify that you have the correct SQL Server service pack and cumulative update level to utilize this feature.

Trace Flag 4119

If you are running SQL 2005 we would recommend that you enable Trace Flag 4119 for performance reasons. This is not necessary on SQL 2008)

http://support.microsoft.com/kb/942659
*Verify that you have the correct SQL Server service pack and cumulative update level to utilize this feature.

TempDB

The number of Data Files for TempDB should be equal to the number of CPU cores utilized by SQL Server up to a maximum of 8. This prevents bottlenecks at the SGAM page allocation process when new Temp objects are created. The files for TempDB should be on a drive separate from the SQL Data and Log files.

We recommend splitting data files for TempDB because each temporary object that is created in TempDB needs pages allocated for it. These pages must be allocated via the SGAM page and there is only one SGAM page per data file (not really but close enough). So if you have only one data file and you are creating a lot of temp objects such as is possible with SSRS and RCSI (or other databases on the server as they all use the same TempDB) you can bottleneck on the SGAM page because it is trying to handle to many allocations at the same time. That is why there is a recommendation to have 1 TempDB data file per CPU core or active thread to alleviate the possibility of an allocation bottleneck. Granted this is rare but it does happen.

NOTE: Each TempDB file DOES NOT need to be on its own drive they just need to be separated from all SQL Data and Log files

Disk Alignment

Disk alignment can have a significant impact on disk performance as one block can span 2 physical disks. Since proper configuration can vary greatly by manufacturer please consult your hardware manufacturing to verify the proper disk alignment and offset settings. The allocation or stripe size should be >=8k to prevent torn pages we recommend 64k to match the size of the SQL Server Extent (8 – 8K pages).

To check disk alignment

1. Click "Start" - "Run" and type MSINFO32
2. Click "Components" - "Storage" - "Disks"
3. Look at the value for "Partition Starting Offset" for the drive you wish to check

"Partition Starting Offset" / (64 * 1024) = a whole number then the disks are aligned for that drive
"Partition Starting Offset" / (64 * 1024) = a decimal number then the disks are not aligned for that drive

Example

Partition Starting Offset = 117,020,033,024 bytes

117,020,033,024 / 65,536 = 1,785,584 (a whole number so the disk is aligned

117,020,033,153 / 65,536 = 1,785,584.001968384 (a decimal number so disk are not aligned)

NOTE: All drives formatted via Window Server 2008 and later automatically will align the disks.

Read Committed Snapshot Isolation (RCSI)

With the release of SQL Server 2005 a new isolation level was introduced called Read Committed Snapshot. This isolation level mimics the “Level 1” Read Committed which is the SQL Server default, but allows for greater concurrency and less blocking as it uses a versioning principle for locked records. Readers and writers will receive different “versions” of the record so they do not block each while maintaining transactional consistency. These row versions are help in the Version Store which is kept in TempDB. From a Microsoft Dynamics NAV application perspective this is not very useful as NAV sends isolation level hints with most statements which will override RCSI, but will have a benefit when outside applications access records within the NAV production database such as reports, views, and secondary applications. (SSRS, ETL processes, etc…)

```
ALTER DATABASE DatabaseName
SET READ_COMMITTED_SNAPSHOT ON
GO
```

NOTE: TempDB must be configured per recommendations or page allocation bottlenecks could occur at the TempDB level.

"Max Degree of Parallelism"

Microsoft Dynamics NAV communicates to SQL Server with client side cursors via ODBC. The C/AL is converted to simple TSQL statements via the NDBCS.DLL. These statements are very simple Select, Update, Insert, and Delete SQL statements and do not require parallel execution in SQL Server. This does not stop the compiler from sometimes attempting to use parallel execution when planning queries and this can sometimes negatively impact performance. This is why I recommend setting the - Max degree of Parallelism in SQL Server to 1 to prevent this.

```
sp_configure 'max degree of parallelism', 1
RECONFIGURE
GO
```

NOTE: If you are running SQL Enterprise Edition Maintenance Jobs will be multi-threaded, but if you set "Max Degree of Parallelism" to 1 they will be executed single threaded. We recommend that in the Maintenance Job you flip "Max Degree of Parallelism" back to 0 run the Maintenance Job and then flip it back to 1. This can be done with TSQL code inside the Job.

Dynamics NAV Default Isolation Level

With the release of the Microsoft Dynamics NAV 5.00 SP1 (Build 30482 / KB 979135) and later executables we now have the option to change NAV default SQL isolation level from SERIALIZABLE to REPEATABLEREAD.

SERIALIZABLE
This is the most restrictive isolation level. When it's used, the phantom values cannot occur. It prevents other users from updating or inserting rows into the **data set** until the transaction will be completed.

REPEATABLEREAD
When it's used, the dirty reads and non-repeatable reads cannot occur. It means that locks will be placed on all **data** that is used in a query, and another transaction cannot update the data.

In summary SERIALIZABLE is more restrictive and locks more records that REPEATABLEREAD which can cause more Lock Timeouts and Deadlocks. REPEATABLEREAD is less restrictive and allows for higher overall concurrency.

Setting the Dynamics NAV default isolation level from SERIALIZABLE to REPEATABLEREAD

```
UPDATE [$ndo$dbproperty] SET diagnostics = 4194304
```

Microsoft Dynamics NAV "Lock Timeout"

This setting essentially means that if a user attempts to access a resource and it is locked by another user it will wait however many seconds you have specified and try again before issuing an error message. We recommend this setting not to be used at all or to be set to between 10 – 20 seconds, having a very high value for the static lock time-out can cause its own issues. Usually when a customer has a high value set for this is because there is an underlying performance issue. This is really up to the discretion of the Customer and Partner and can be set higher and there are specific circumstances where this does need to be set higher. This setting can be found under [File] – [Database] – [Alter] on the “Advanced” tab.

Microsoft Dynamics NAV "Always Rowlock"

The Microsoft Dynamics NAV “Always” rowlock setting was a holdover from SQL 2000 and 32 bit architecture. In 99% of the cases on SQL 2005 and later this can be disabled which will free up lock memory that SQL can use for other purposes without any loss of performance. This setting can be found under [File] – [Database] – [Alter] on the “Advanced” tab.

NOTE: In some benchmarks not having this setting enabled actually increased concurrency and performance.

Maintenance Jobs

Statistics

Even with "Auto Update Statistics" enabled we still strongly recommend running a periodic SQL Maintenance Job to update statistics. This is because "Auto Update Statistics" will only be triggered when 5% of the data in that statistic has changed and on tables such as Value Entry, Item Ledger Entry, G/L Entry, and such can have 10s of millions of records a small percentage of data in a given statistic such as [Entry No.] can change have a material effect on the overall data distribution in that statistic causing inefficient query plans long before the 5% threshold is reached resulting in degraded performance. We recommend the use of "sp_updatestats" to accomplish this task as it will only update the statistics where data has been changed. We would recommend creating a SQL Job that runs daily or weekly (depending on transaction volume) during off peak hours to update all statistics where data has changed.

NOTE: Do not use "FULLSCAN" as this is a waste of time as it will be overwritten the second the statistic is "auto updated."

Index defragmentation

Indexes become fragmented just like hard drives and can have a negative impact on overall Performance. We recommend using the following script which use the REBUILD and REORGANIZE functions for index defragmentation. We would recommend creating a SQL Job that runs daily or weekly (depending on transaction volume) during off peak hours to update all indexes.

```sql
SET NOCOUNT ON;
DECLARE @objectid int;
DECLARE @indexid int;
DECLARE @partitioncount bigint;
DECLARE @schemaname sysname;
DECLARE @objectname sysname;
DECLARE @indexname sysname;
DECLARE @partitionnum bigint;
DECLARE @partitions bigint;
DECLARE @frag float;
DECLARE @command varchar(8000);
DECLARE @percentage int

SET @percentage = 25  --You can change the precentage
threshhold here
```

```
IF EXISTS (SELECT name FROM sys.objects WHERE name =
'work_to_do')
    DROP TABLE work_to_do;

SELECT
    object_id AS objectid,
    index_id AS indexid,
    partition_number AS partitionnum,
    avg_fragmentation_in_percent AS frag
INTO work_to_do
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL,
NULL , NULL, 'DETAILED')
WHERE avg_fragmentation_in_percent > @percentage AND
index_id > 0;

DECLARE partitions CURSOR FOR SELECT * FROM
work_to_do;


OPEN partitions;


FETCH NEXT
   FROM partitions
   INTO @objectid, @indexid, @partitionnum, @frag;

WHILE @@FETCH_STATUS = 0
    BEGIN;
        SELECT @objectname = o.name, @schemaname =
s.name
        FROM sys.objects AS o
        JOIN sys.schemas as s ON s.schema_id =
o.schema_id
        WHERE o.object_id = @objectid;

        SELECT @indexname = name
        FROM sys.indexes
        WHERE  object_id = @objectid AND index_id =
@indexid;

        SELECT @partitioncount = count (*)
        FROM sys.partitions
        WHERE object_id = @objectid AND index_id =
@indexid;
```

```
IF @frag < (@percentage)
    BEGIN;
    SELECT @command = 'ALTER INDEX ' + '[' +
@indexname + ']' + ' ON ' + '[' + @schemaname + ']' +
'.' + '[' +@objectname + ']' + ' REORGANIZE';
    --IF @partitioncount > 1
    --      SELECT @command = @command + ' PARTITION='
+ CONVERT (CHAR, @partitionnum);
    EXEC (@command);
    END;

IF @frag >= (@percentage)
    BEGIN;
    SELECT @command = 'ALTER INDEX ' + '[' +
@indexname + ']' + ' ON ' + '[' + @schemaname + ']' +
'.' + '[' +@objectname + ']'  + ' REBUILD';
    --IF @partitioncount > 1
    --      SELECT @command = @command + ' PARTITION='
+ CONVERT (CHAR, @partitionnum);
    EXEC (@command);
    END;
PRINT 'Executed ' + @command;

FETCH NEXT FROM partitions INTO @objectid, @indexid,
@partitionnum, @frag;
END;

CLOSE partitions;
DEALLOCATE partitions;

IF EXISTS (SELECT name FROM sys.objects WHERE name =
'work_to_do')
    DROP TABLE work_to_do;
GO
```

NOTE: See "Max Degree of Parallelism"

Instant File initialization

In SQL Server 2005 and later versions, data files can be initialized instantaneously, allowing for fast execution of database or file group restore operations. Instant file initialization reclaims used disk space without filling that space with zeros. Instead, disk content is overwritten as new data is written to the files. Log file initialization still requires zeroing, but it will happen in parallel with the transfer of the data from the backup. The roll forward step of restore will not start until all of the data has been transferred and the whole log has been initialized.

To use instant file initialization, you must run the MSSQLSERVER service account under a Windows account and assign the Windows SE_MANAGE_VOLUME_NAME special privilege to that Windows account. This privilege is assigned to the Windows Administrators group by default. If you have system administrator rights, you can assign this privilege by adding the Windows account to the Perform Volume Maintenance Tasks security policy.

"Page Verify" - Checksum vs. Torn Page Detection

We recommend use the Database Page Verify option to Checksum as Checksum is far more robust at detecting physical database corruption.

```
ALTER DATABASE DatabaseName
SET PAGE_VERIFY CHECKSUM;
GO
```

Lock Pages in Memory?

We do recommend turning on “Lock Pages in Memory” so that OS doesn't page SQL Server out. However on 64 bit you only need to grant the right "Lock Pages in Memory" to the SQL account for SQL Server to utilize this feature. You do need to change any of AWE settings through sp_configure.

http://technet.microsoft.com/en-us/library/ms190730.aspx

For more information on these setting please refer to “SQL Server Books Online” and MSDN.

These postings are provided "AS IS" with no warranties and confers no rights. You assume all risk for your use.

Michael De Voe
Senior Premier Field Engineer
Microsoft Dynamics
Microsoft Certified Master - SQL Server 2008