

Module 1

SQL Server IO

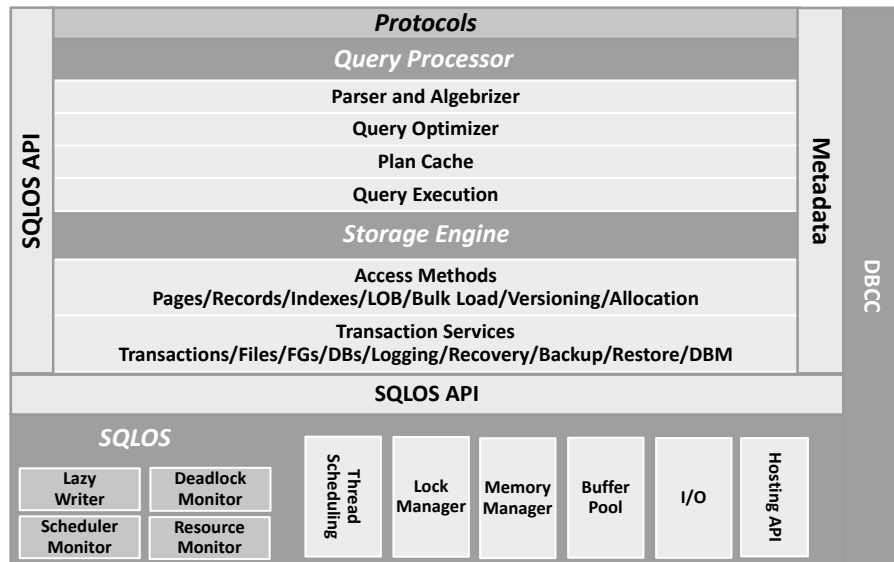
Storage Needs



Overview

- Files, filegroups, and physical DB layout
- Data: writing, logging, reading, restoring
- Always a special case: tempdb
- Basic monitoring

Server Architecture

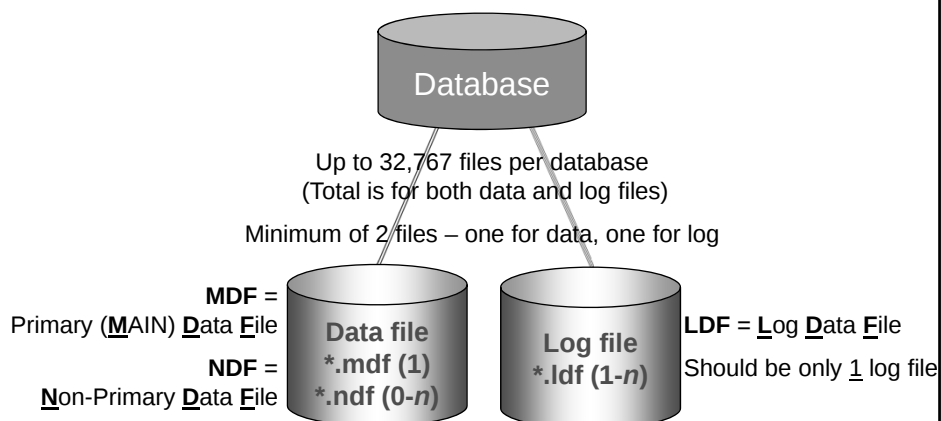


©SQLskills.com
SQLskills Immersion Event

3

Database Structure

Up to 32,767 databases per instance



➡ Just because you can, doesn't mean you should...

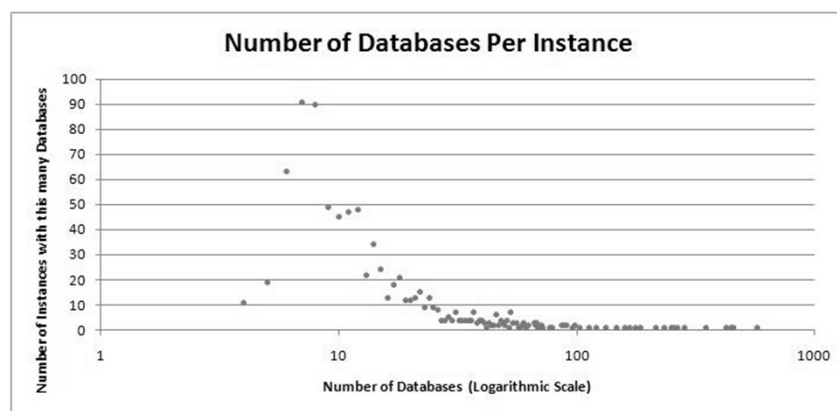
©SQLskills.com
SQLskills Immersion Event

4

Consolidation Issues

- Too many databases per instance can lead to:
 - Performance problems (lack of resources)
 - Constantly fighting for buffer pool resources
 - I/O subsystem placement difficulties
 - Maintenance problems (lack of resources)
 - Constant background IO/CPU load from maintenance, CHECKDB, backups
- Too many database files can lead to long startup time
 - Every file has to be opened and read to get the file header page

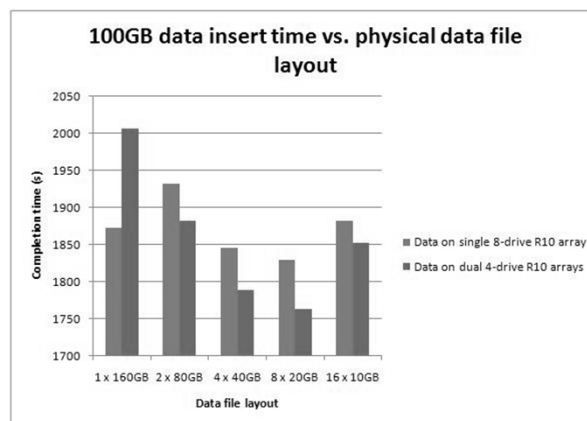
Databases per Instance



Physical Layout Considerations

- Degree of separation depends on I/O workload and I/O subsystem
 - E.g. it may be fine to have several databases on one volume
 - E.g. SAN can do a better job of spreading I/O costs than a single drive
- Consider separating:
 - Log files from data files
 - Different databases on different volumes
 - SQL Server files separate from other uses (e.g. OS files)
- Decide on file/filegroup layout
 - tempdb (see later in this module)
 - Multiple data files for increased performance?
 - Multiple filegroups for partitioned maintenance?
 - Multiple filegroups to allow partial database availability?
 - Usually don't put all logs on one LUN and all data on another LUN
- WP: Physical Database Storage Design
 - <http://www.microsoft.com/technet/prodtechnol/sql/2005/physdbstor.mspx>

Multiple Data Files?



- Source: <http://www.sqlskills.com/BLOGS/PAUL/post/Benchmarking-do-multiple-data-files-make-a-difference.aspx>

Data File Layout Survey: <10GB

What's the physical layout of your database, and why? (Databases under 10GB)

Single filegroup, single file		71%	259
Single filegroup, multiple files on a single drive		5%	18
Single filegroup, multiple files spread over multiple drives/arrays/LUNs		13%	47
Multiple filegroups, one per drive/array/LUN, for performance		3%	11
Multiple filegroups, one per drive/array/LUN, for recoverability		0%	1
Multiple filegroups, one per drive/array/LUN, for manageability		1%	3
Multiple filegroups, one per drive/array/LUN, for multiple reasons		3%	11
Multiple filegroups, all on same drive/array/LUN, for any reason		4%	14
Total: 364 responses			

Source: <http://www.sqlskills.com/BLOGS/PAUL/post/physical-database-layout-vs-database-size.aspx>

©SQLskills.com
SQLskills Immersion Event

9

SQL

Data File Layout Survey: 10-100GB

What's the physical layout of your database, and why? (Databases between 10GB and 100GB)

Single filegroup, single file		43%	135
Single filegroup, multiple files on a single drive		9%	27
Single filegroup, multiple files spread over multiple drives/arrays/LUNs		19%	58
Multiple filegroups, one per drive/array/LUN, for performance		11%	34
Multiple filegroups, one per drive/array/LUN, for recoverability		0%	1
Multiple filegroups, one per drive/array/LUN, for manageability		2%	5
Multiple filegroups, one per drive/array/LUN, for multiple reasons		11%	34
Multiple filegroups, all on same drive/array/LUN, for any reason		5%	17
Total: 311 responses			

Source: <http://www.sqlskills.com/BLOGS/PAUL/post/physical-database-layout-vs-database-size.aspx>

©SQLskills.com
SQLskills Immersion Event

10

SQL

Data File Layout Survey: 100GB-1TB

What's the physical layout of your database, and why? (Databases between 100GB and 1TB)

Single filegroup, single file		20%	45
Single filegroup, multiple files on a single drive		5%	11
Single filegroup, multiple files spread over multiple drives/arrays/LUNs		17%	39
Multiple filegroups, one per drive/array/LUN, for performance		21%	48
Multiple filegroups, one per drive/array/LUN, for recoverability		1%	3
Multiple filegroups, one per drive/array/LUN, for manageability		1%	2
Multiple filegroups, one per drive/array/LUN, for multiple reasons		25%	57
Multiple filegroups, all on same drive/array/LUN, for any reason		11%	25
Total: 230 responses			

Source: <http://www.sqlskills.com/BLOGS/PAUL/post/physical-database-layout-vs-database-size.aspx>

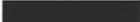
©SQLskills.com
SQLskills Immersion Event

11

SQL

Data File Layout Survey: 1TB+

What's the physical layout of your database, and why? (Databases over 1TB)

Single filegroup, single file		11%	16
Single filegroup, multiple files on a single drive		2%	3
Single filegroup, multiple files spread over multiple drives/arrays/LUNs		8%	12
Multiple filegroups, one per drive/array/LUN, for performance		11%	17
Multiple filegroups, one per drive/array/LUN, for recoverability		2%	3
Multiple filegroups, one per drive/array/LUN, for manageability		3%	4
Multiple filegroups, one per drive/array/LUN, for multiple reasons		52%	78
Multiple filegroups, all on same drive/array/LUN, for any reason		11%	16
Total: 149 responses			

Source: <http://www.sqlskills.com/BLOGS/PAUL/post/physical-database-layout-vs-database-size.aspx>

©SQLskills.com
SQLskills Immersion Event

12

SQL

SSDs

- SSD = Solid State Drive
- Extremely low IO latency and high throughput
- Expensive, but coming down in price
- Yes, enterprise-ready
 - We have enterprise clients running on them
- Excellent for random IOs, but also good anywhere there is an IO bottleneck
- See Paul's Benchmarking blog series:
 - <http://www.sqlskills.com/BLOGS/PAUL/category/Benchmarking.aspx>

Configuration Options

- Auto-shrink OFF!
 - And rarely shrink
- Auto-grow ON
 - Set to something other than the defaults
- Instant file initialization usually ON
 - Not applicable to log files
 - *Very* useful for cutting down restore times
- Be careful of:
 - Index fragmentation causing wasted space (extra IOs) and preventing efficient range scan read-ahead
 - Incorrect partition alignment causing double reads/writes

Data Writing

- Data file writes can be:
 - Single/multiple pages
 - Single/multiple extents (for bulk operations)
- Data file pages are written when:
 - A checkpoint occurs (for whatever reason)
 - The lazywriter forces a dirty page from the buffer pool
 - A bulk operation flush occurs (a.k.a 'eager writes')
- Write performance can be dramatically affected by:
 - Number of files and file placement
 - Incorrect I/O subsystem configuration

Read/Write Latency

- Many systems these days are I/O bound, but is the problem the I/O subsystem or your queries?
- If you've optimized your queries and performance is still slow, look into the I/O subsystem
- `sys.dm_io_virtual_file_stats`
 - Replaces `fn_virtualfilestats`
 - Gives total stall time (aggregated latencies) for reads and writes along with read/write counts
 - Better than Physical Disk performance counters as these are per database file and give SQL Server's view of the I/O subsystem
- `sys.dm_io_pending_io_requests`
 - Lists all pending I/Os
 - Join with `sys.dm_io_virtual_file_stats`

Demo

I/O latencies

SQL
skills

Data Logging

- Every database change is logged to some degree
 - With the single exception of the version store in tempdb
- Writes: always sequential
- Reads: mixture of sequential and occasional random
 - Sequential reads from backups, replication log reader, and others
 - Random from transaction rollback following backwards chain of log records
- No performance gain from multiple log files
- Log records are written prior to the data file pages they describe being changed
- Most important part of the database
 - Protect it with redundancy and make sure it gets good performance
- Log file performance can be affected by *VLF fragmentation*
 - See <http://www.sqlskills.com/blogs/kimberly/post/8-Steps-to-better-Transaction-Log-throughput.aspx>
 - Alleviated somewhat by recent hotfix...

Transaction Log I/O

- There are specific limits on transaction log I/O
- Limit of outstanding I/Os (2005 SP1 onwards)
 - 64-bit: 32 outstanding I/Os
 - 32-bit: 8 outstanding I/Os
- Amount of outstanding I/O
 - SQL Server 2008: limit of 3840K at any given time
 - Prior to SQL Server 2008: limit of 480K at any given time
- This allows you to tell whether log throughput is being limited by the I/O subsystem performance
- Monitor by joining sys.dm_io_* DMVs or using perfmon or wait stats
- More details at <http://sqlcat.com/technicalnotes/archive/2008/12/09/diagnosing-transaction-log-performance-issues-and-limits-of-the-log-manager.aspx>

Transaction Log Usage

- sys.dm_tran_active_transactions
 - Lists all active transactions for the whole instance
 - Useful for debugging DTC issues
- sys.dm_tran_database_transactions
 - Lists information for all transactions with log information
 - Useful for determining who is using space in the log
 - Useful for determining how much log space a particular operation generates

Demo

Transaction log usage



How Does Allocation Work?

- With a single-file database, very simple
- When a table is stored in a filegroup with multiple files, two allocation algorithms kick-in
 - Round-robin allocation
 - Allocations are made from each data file in the filegroup in turn, essentially striping the data across multiple files
 - Proportional fill
 - Allocations are made from each data file during round-robin proportional to the amount of free space in each file

But If You Add Another Data File...

- SQL Server will NOT rebalance data across all files in a filegroup when you add another file
 - I was never allowed to implement my design ☹
- If you want to do this, consider:
 - Adding another filegroup
 - Rebuilding all indexes into the new filegroup
 - Drop the old filegroup
- However, this is tricky because:
 - You need to provision even **more** space
 - CREATE INDEX ... WITH (DROP_EXISTING=ON) doesn't move LOB data
- Best bet is to design up front before database grows

Data Reading

- Reads can be:
 - Single/multiple pages from a data file
 - Single/multiple extents from a data file
 - Variable size chunks of FILESTREAM files
 - Usually random, except for large scans and backups
- Misconception that SQL Server always reads extents
 - But it will do sometimes to 'ramp up' the buffer pool
- Reads can come from the buffer pool or backups
- Read performance can be dramatically affected by:
 - Number of files and file placement
 - Incorrect I/O subsystem configuration
 - Buffer pool memory and memory pressure
 - Ability to perform efficient read-ahead on indexes

LOB Data...

- ... is a special case
- Usually written once only and not read very often
 - Examples: documents, photos, medical images, videos
- Do you really need to have all that data on your fastest drives with the most redundant RAID level?
 - I.e. the most expensive storage for the least read, most unchanging data.... No.
- Variety of storage choices:
 - In-row, out-of-row, vertically partitioned, FILESTREAM
 - All with pros and cons
 - See <http://www.sqlskills.com/BLOGS/PAUL/post/Importance-of-choosing-the-right-LOB-storage-technique.aspx>

Buffer Pool Usage

- Unfortunately the query optimizer knows nothing about the contents of the buffer pool otherwise it might choose a less optimal index that's already in memory (to save physical I/Os)
- sys.dm_os_buffer_descriptors
 - Lists all pages currently in memory
 - Allows aggregating by database, table
 - Allows view of aggregate empty space in pages in memory
 - Can look at how memory pressure affects need to perform physical vs. logical I/Os

Demo

Buffer pool usage



Data Restoring

- One of the most critical operations you'll perform
- You want it to go fast!
- How to make it fast:
 - Enable instant file initialization
 - Restore as little as possible
 - Piecemeal restore – page, file, filegroup
 - Partial restore – only those filegroups necessary
 - Avoid very long-running transactions that require rollback
 - Use backup compression
 - Use backup striping so restore can parallelize reads

Tempdb: The Dumping Ground

- User objects:
 - Any user-created table
 - Table variables (@)
 - If they get large enough to spill to disk
 - #temp tables/indexes
 - Global ##temp tables/indexes
- Internal objects
 - Worktables (e.g. sort, hash joins, order by, ...) from memory spills to disk
 - Intermediate sort results from sorts
 - Intermediate sort results from index create/rebuilds
 - Only if SORT_IN_TEMPDB is specified
 - Intermediate DBCC CHECKDB results
 - Version store (main one + online index operations one)
- (Full list, with explanations, in Books Online: Capacity Planning for Tempdb)

Tempdb Sizing

- No easy way to figure out initial size as so many operations can use tempdb space
- General practice is to create tempdb, run production workload and see how big tempdb gets
 - General query workload
 - Index maintenance operations
- Set tempdb size to resulting size
- Enable appropriate auto-growth, equal on all files
- Enable instant file initialization for instance
- We'll talk about the number of files to create in a few slides...
- Books Online:
 - Capacity Planning for Tempdb
 - <http://msdn.microsoft.com/en-us/library/ms345368.aspx>

Tempdb and I/O Subsystems

- Best practice is to separate tempdb from other databases due to I/O contention and put on high-performance I/O subsystem
 - Entirely depends on I/O subsystem and workload involved
- Favorite use of SSDs today
 - SSDs are best suited to random IO, but generally will give a perf boost to any I/O subsystem that's overwhelmed
- Page checksums weren't available for tempdb in 2005, but are in 2008
 - On by default for new instances of 2008
 - Must enable for tempdb for upgraded instances
- Can be put on RAM drive (no durability requirements)
- Supported to have non-shared tempdb in cluster in SQL 2012
- Books Online:
 - Optimizing Tempdb Performance
 - <http://msdn.microsoft.com/en-us/library/ms175527.aspx>

Contention in Tempdb

- Tempdb is very susceptible to contention issues because there's only one tempdb per instance to support all user database operations
- Contention is very often:
 - PAGEIOLATCH
 - Thread waiting for a page to be read from disk
 - PAGELATCH
 - Multiple threads competing for access to an in-memory page
- PAGEIOLATCH contention can be alleviated by:
 - Creating multiple data files, and/or
 - Putting tempdb on a high-performing I/O subsystem

In-Memory Tempdb Contention

- Some query workloads cause multiple concurrent threads to repeatedly create/drop small temp tables and/or worktables
- Causes PAGELATCH contention on allocation bitmaps and system catalogs
 - Use sys.dm_os_waiting_tasks to see waits on PAGELATCH_EX
 - SGAM page to manipulate mixed extents (resource 2:1:3)
 - PFS page to allocate/deallocate pages (resource 2:1:1)
 - System catalogs (usually resource 2:1:103 – sys.sysmultiobjrefs)
 - Don't use unique constraints
 - Fixed somewhat by 2008 R2 CU9
- Contention can occur in all versions
- This can sometimes (rarely) happen in user databases with VERY high-end allocation workloads

Alleviating In-Memory Tempdb Contention

- Trace flag 1118 (discussed in KB 328551) removes use of mixed extents (single-page allocations) in tempdb (and other DBs)
 - Removes SGAM contention
 - Still works and may be needed in 2005+, but not as much as on 2000
- Creating multiple data files to spread and reduce the contention
 - For 2000, # data files should be equal to # processor cores
 - For 2005+, you can still do 1:1 but you may not need to
 - Start with # data files = ¼ to ½ # processor cores and add more if needed
 - Bob Ward (CSS) at PASS 2011:
 - < 8 cores, use #files = #cores
 - > 8 cores, use 8 files, increasing by chunks of 4
 - Be careful of having too many data files
- Note: this does not apply to log files

Tempdb Troubleshooting (1)

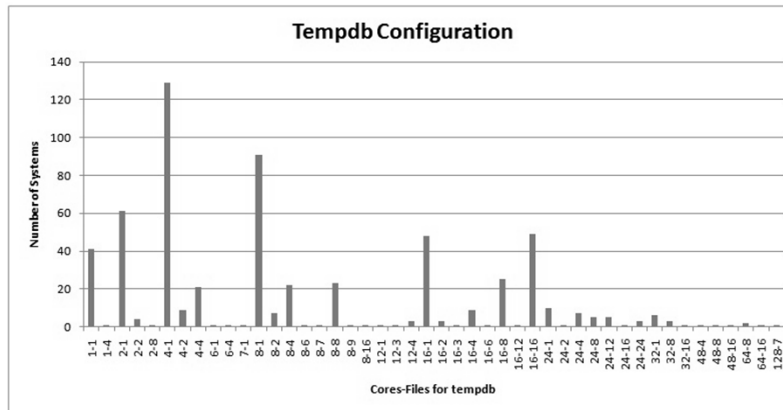
- **PAGELATCH/PAGEIOLATCH contention**
 - Allocation contention / I/O contention / user object DDL contention
 - Look at perf counters:
 - Temp tables in General Statistics perf object
 - Workfiles/worktables in Access Methods perf object
 - Look at wait stats:


```
SELECT [session_id], [wait_duration_ms],
       [resource_description]
FROM sys.dm_os_waiting_tasks
WHERE [wait_type] LIKE 'PAGELATCH_%'
AND [resource_description] LIKE '2:%';
```
 - I'll show you this in Module 5
 - Change workload / TF1118 / create multiple data files / limit auto-growth size / enable instant file initialization

Tempdb Troubleshooting (2)

- **Out of space**
 - Look at tempdb DMVs
 - sys.dm_db_file/session/task_space_usage
 - (Session = connection; task = unit of work; parallel query = multiple tasks)
 - Look at tempdb perf counters in Transactions perf object
 - Free space in tempdb, version store size, version generation/cleanup rates
- **Figure out what's using the most space**
 - Is a long-running transaction causing the version store to grow?
 - Version cleanup rate = 0
 - Is someone creating large user objects in tempdb?
 - See breakdown of results from sys.dm_db_file_space_usage
 - Is a plan causing lots of memory spills to tempdb?
 - More complex – see demo and whitepaper
- **See whitepaper:**
 - Troubleshooting Performance Problems in SQL Server 2008
 - Nasty link: see link from SQLskills.com/whitepapers.asp

Tempdb Layout Survey



Source: <http://www.sqlskills.com/BLOGS/PAUL/post/Tempdb-configuration-survey-results.aspx>

©SQLskills.com
SQLskills Immersion Event

37

Demo

tempdb troubleshooting

Temp Table Misuse

- Very common for us to see temp table problems
- Lack of filtering when populating temp table
 - Pulling columns into a temp table that are not used
 - Large waste of space and CPU
 - Minimize the column list in a temp table
- Incorrect temp table indexing
 - Create indexes after populating the table
 - Only create indexes that are required
- Temp table when none is required
 - Forcing an intermediate result set into a temp table could disrupt the efficient data pipeline through a query
 - Query may run much faster without a temp table

Tempdb Space Tracking (1)

- On several client systems I have an automated tempdb space tracking system (that I'll show you)
- sys.dm_db_file_space_usage
 - Total of all pages per file
 - Unused
 - Internal
 - Work tables for cursor or spool operations and temporary large object (LOB) storage
 - Work files for operations such as a hash join
 - Sort runs
 - User is everything else except version store
 - Version store

Tempdb Space Tracking (2)

- `sys.dm_db_session_space_usage`
 - Tracks cumulative page allocation and deallocation counts since the start of the session
 - Split between internal and user as on previous slide
- `sys.dm_db_task_space_usage`
 - Same as above, but for each worker (thread)
 - Can see parallelism happening

Demo

tempdb space tracking

Basic Monitoring

- Lots on this through the week
- Things to look for:
 - File growth and shrinks
 - PAGEIOLATCH_XX and other waits
 - Latencies in sys.dm_io_virtual_file_stats
 - Avg. Disk sec/Read and /Write in Physical Disk performance object
 - Do NOT look at disk queue lengths

Review

- Files, filegroups, and physical DB layout
- Data: writing, logging, reading, restoring
- Always a special case: tempdb
- Basic monitoring

More Info on tempdb

- Moving tempdb – see KB 224071
- WP: Working with tempdb in SQL Server 2005
 - <http://www.microsoft.com/technet/prodtechnol/sql/2005/workingwithtempdb.msp>
- Blog post: Misconceptions Around TF 1118
 - <http://www.sqlskills.com/BLOGS/PAUL/post/Misconceptions-around-TF-1118.aspx>
- Blog post series on tempdb
 - <http://www.sqlskills.com/BLOGS/PAUL/post/Comprehensive-tempdb-blog-post-series.aspx>
- Books Online:
 - Troubleshooting Insufficient Disk Space in Tempdb
 - <http://msdn.microsoft.com/en-us/library/ms176029.aspx>
 - Capacity Planning for Tempdb
 - <http://msdn.microsoft.com/en-us/library/ms345368.aspx>