

SQLskills Immersion Event

IEPTO1: Performance Tuning and Optimization

Module 2: Data File Internals and Maintenance

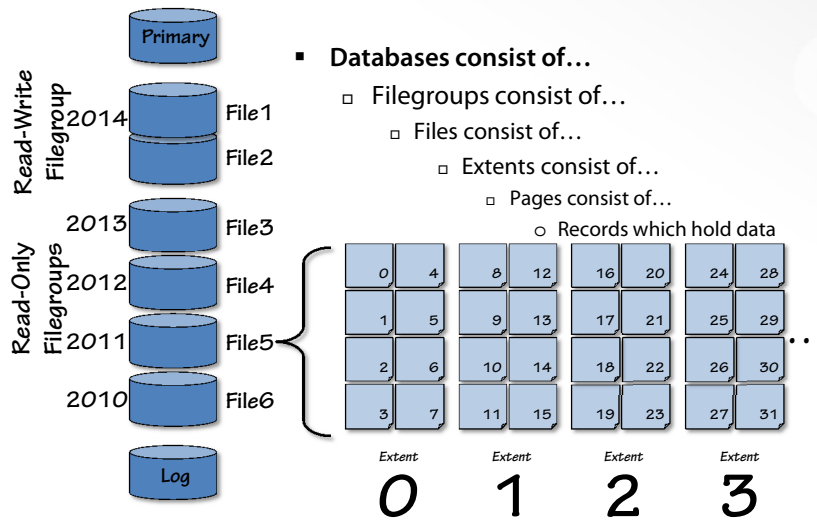
Paul S. Randal
Paul@SQLskills.com



Overview

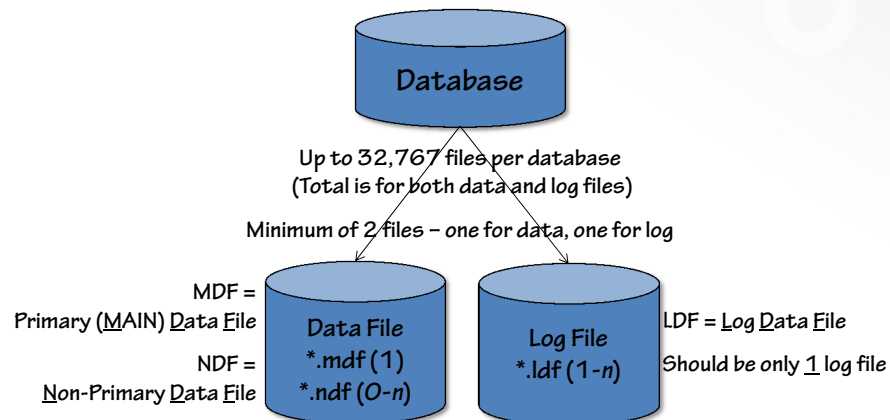
- Physical layout considerations
- Allocation algorithms
- Instant initialization
- Auto-grow
- To shrink or not to shrink?
- Data compression
- Tempdb

Database Components



Database Structure

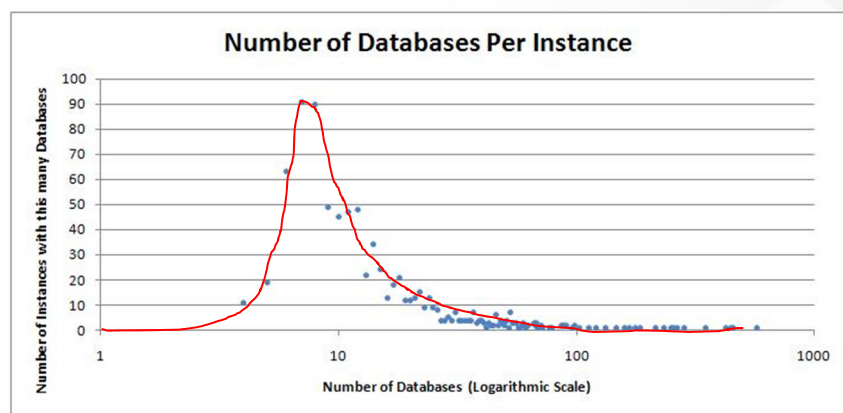
Up to 32,767 databases per instance



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 I/O and CPU load from maintenance, DBCC 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



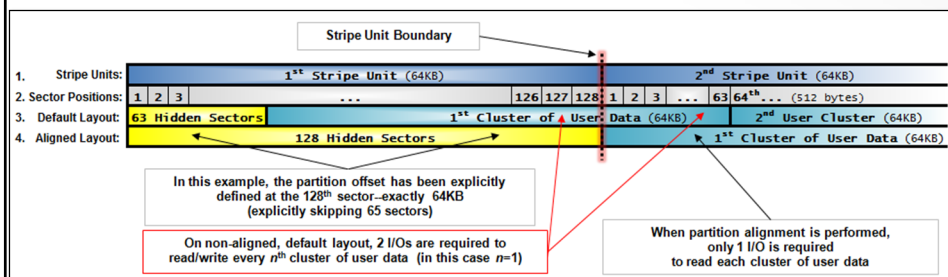
- Source: <http://www.sqlskills.com/blogs/paul/log-file-configuration-metrics-for-17000-databases/> (<http://bit.ly/bffg7W>)

Files and Filegroups

- **A filegroup contains one or more files**
 - A table or index is wholly contained in a single filegroup
- **Every database has at least one filegroup – the PRIMARY filegroup**
 - Contains at least one file – the MDF
 - Multiple data files can be added to the filegroup
- **Multiple secondary filegroups can be defined**
 - Each secondary filegroup can have multiple data files
- **Many reasons for doing this**
 - Manageability, performance, availability...

Partition Alignment

- With the recommended 64KB RAID stripe size, and the recommended 64KB NTFS cluster size, but with bad alignment, two I/Os are required to read/write every 64KB of data



- With correct alignment, only a single I/O is required

Volume Configuration

- **Disk partition offsets (when not on Windows Server 2008+)**
 - Before WS2008, default offset is 31.5KB, WS2008+ uses 1MB
 - This is not RAID stripe or NTFS allocation unit aligned
 - Leads to extra I/Os to read a portion of data
 - This can cause up to 30-40% performance drop
 - As documented by MS with many customers
 - On WS2008+, still must check after formatting to ensure I/O subsystem did not intercept and use incorrect alignment
 - Upgrading to WS2008+ does not fix alignment of existing partitions
- **RAID stripe size should be 64KB or higher**
- **NTFS cluster (allocation unit) size**
 - Should usually be 64KB
- **See WP: Disk Partition Alignment Best Practices For SQL Server**
 - <http://msdn.microsoft.com/en-us/library/dd758814.aspx>

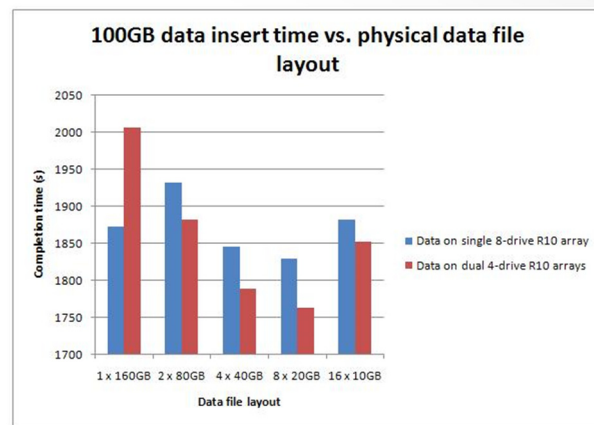
Physical Layout Considerations (1)

- **RAID array configuration**
 - Know your workload and choose an appropriate RAID level (e.g. not RAID-5 for a high volume OLTP system)
 - Different kinds of data onto different storage
 - E.g. rarely used LOB data on RAID 5, OLTP data on RAID 1/10
- **Defrag the volumes**
 - NTFS-level fragmentation DOES affect performance of large scans
 - Not needed if the volumes are not being shared
 - Do not do volume defragging while SQL Server is running
- **Pre-allocate the files to prevent them having to grow**
- **Capacity planning considerations?**

SSDs

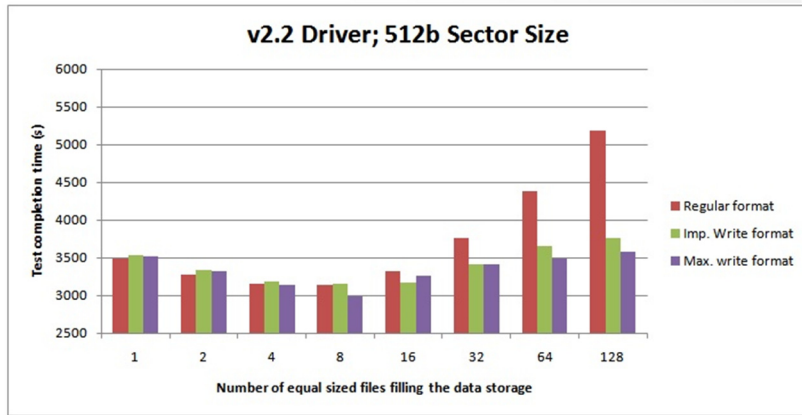
- SSD = Solid State Drive
- Extremely low I/O latency and high throughput
- Can be expensive if using Enterprise-class
- Beware of 'best practices'
 - Don't just put tempdb or transaction logs on them!
 - Don't ignore index fragmentation when using them!
- You must have multiple SSDs in a RAID configuration
- Consider Buffer Pool Extension in SQL Server 2014
 - Ability to have unchanged data stored on SSD
 - Too early to give real-world guidance
 - See MSDN at <http://bit.ly/RFNfeu>
- See Paul's SSD blog series:
 - <http://www.sqlskills.com/blogs/paul/category/ssds/>

Multiple Data Files?



- Testing on 15k 300GB SCSI in Dell MD3000i iSCSI, 8-core server
- Source: <http://www.sqlskills.com/blogs/paul/benchmarking-do-multiple-data-files-make-a-difference/> (<http://bit.ly/as3ZMV>)

Multiple Data Files With SSDs?



- Testing on Fusion-io 640GB ioDrive Duo, 8-core server
- Source: <http://www.sqlskills.com/blogs/paul/benchmarking-multiple-data-files-on-ssds-plus-the-latest-fusion-io-driver/> (<http://bit.ly/isXIkW>)

Physical Layout Considerations (2)

- **Separation of files**
 - Degree of separation depends on I/O workload and I/O subsystem
 - E.g. SAN can do a better job of spreading I/O costs than a single drive
 - Consider separating:
 - Log files from data files, even separate databases on different LUNs
 - 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?
- **Note: SQL Server 2012+ allows files to be on SMB shares**
- **WP: Physical Database Storage Design**
 - <http://www.microsoft.com/technet/prodtechnol/sql/2005/physdbstor.mspx>
(<http://bit.ly/T28kef>)

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/physical-database-layout-vs-database-size/> (<http://bit.ly/1deVt4S>)



15

© SQLskills. All rights reserved.
<http://www.sqlskills.com>

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/physical-database-layout-vs-database-size/> (<http://bit.ly/1deVt4S>)



16

© SQLskills. All rights reserved.
<http://www.sqlskills.com>

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/physical-database-layout-vs-database-size/> (<http://bit.ly/1deVt4S>)



17

© SQLskills. All rights reserved.
<http://www.sqlskills.com>

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/physical-database-layout-vs-database-size/> (<http://bit.ly/1deVt4S>)



18

© SQLskills. All rights reserved.
<http://www.sqlskills.com>

Overview

- Physical layout considerations
- Allocation algorithms
- Instant initialization
- Auto-grow
- To shrink or not to shrink?
- Data compression
- Tempdb

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
- **Misconception: you cannot add another data file to a filegroup and rebalance the allocations across all files**
 - You have to create a new filegroup to do that

How Does the Buffer Pool Work?

- Sometimes called the 'buffer cache'
- Memory block used to hold in-memory copies of pages from data files
- Pages are read into the buffer pool when requested by other parts of the Storage Engine and not already in memory
 - When a page is already in memory, this is a logical I/O
 - When a page has to be read from disk, this is a physical I/O
 - All I/Os start as logical and may become physical
- Page lifetime in memory depends on many things
 - Two last access times are tracked and provide an LRU indication
 - Tracked as smallint, not a full datetime
 - When memory pressure is felt, the least-recently-used pages are 'tossed' out of the buffer pool to make way for needed pages
 - Also done proactively by the lazy writer background process
- Hash tables exist to quickly find page images in memory

More on the Buffer Pool

- What's in the buffer pool?
 - `SELECT * FROM sys.dm_os_buffer_descriptors`
 - Be aware that large amounts of index fragmentation can lead to lots of wasted buffer pool space
 - Blog posts with scripts:
<http://www.sqlskills.com/blogs/paul/category/buffer-pool/>
- Memory management
 - Buffer pool does full-extent reads when ramping up in Enterprise Edition
 - And with TF840 in other editions: <http://support.microsoft.com/kb/912322/>
 - As long as the I/O subsystem can keep up, no downsides to enabling

Creating or Growing Data Files

- **Whenever a data file is created or grown, its 'high-water mark' must be set in NTFS**
 - This is the point in the file up to which NTFS knows the data is trustworthy and will allow it to be read
- **By default, this is done by zeroing out the new/additional space**
 - This is done by SQL Server, single-threaded, by issuing successive writes of blocks of zeroes to the file
 - The new /additional space cannot be used until the process completes and the allocation that triggered it will pause until it is done
- **In SQL Server 2005+ this process can be skipped by enabling instant file initialization**
 - Only applies to data files

Instant File Initialization in Action

- **Hardware configuration – Dell Precision 670**
Dual Proc (x64) with Dual Core, 4 GB Memory RAID 1+0 array w/4-142 GB, 15000rpm disks
- **Software configuration: SQL 2005**
 - With zero initialization
 - CREATE DATABASE with 20 GB Data file = 14:02 minutes
 - ALTER DATABASE BY 10 GB = 7:01 minutes
 - RESTORE 30 GB DATABASE (EMPTY Backup) = 21:07 minutes
 - RESTORE 30 GB DATABASE (11GB Backup) = 38:28 minutes
 - With instant file initialization
 - CREATE DATABASE with 20 GB Data file = 1.3 seconds
 - ALTER DATABASE BY 10 GB = 0.4 seconds
 - RESTORE 30 GB DATABASE (EMPTY Backup) = 5 seconds
 - RESTORE 30 GB DATABASE (11 GB Backup) = 19:42 minutes

Instant File Initialization

- Allows SQL Server to skip the zeroing process
- Instead of doing the zeroing, SQL Server calls the Windows API SetFileValidData, which sets the file highwater mark
- Disabled by default because there is a potential security risk of enabling it:
 - If a volume is being shared by a database and a file server storing 'secure files' and if the file server deletes some files, and then the database grows, it may pick up some of the space used by the deleted files
 - As instant file initialization skips zeroing the new space, the old raw bytes of the deleted files will be accessible to a DBA using DBCC PAGE
 - If this is not the case then no security risk, or if the DBA is a Windows Administrator then the risk is already there
- If you can, turn it on!

Enabling Instant File Initialization

- Cannot be enabled from within SQL Server
- Requires:
 - NTFS volumes with Windows XP SP2+ or Windows 2003+ Server
 - SQL 2005+, any edition
 - "Perform Volume Maintenance Tasks" security permission granted to SQL Server service account or group
- Use Local Security Policy Editor to grant the permission
 - Administrative Tools -> Local Security Policy and then Local Policies -> User Rights Assignment
- Defaults to Local Administrators Group
 - Can grant this to lower privileged account
- Instant initialization happens automatically once SQL Server is restarted after granting the permission
- Use TF 3605 + TF 3004 to watch initialization process in the error log

Auto-Grow

- **When a file is full, it will grow to provide more space**
 - By default, new space is zero-initialized, which takes time
 - Allocation pauses while the grow takes place
 - Commonly the auto-growth default is not changed leading to massive amounts of auto-growth
 - 1MB for data files (we'll discuss log files later this week)
- **Best practice to manually manage file sizes**
 - Manually growing files allows YOU to decide which files to grow, by how much, and when
 - Allows file fragmentation to be minimized
 - Monitor file usage and alert when threshold reached
- **Auto-grow should always be left enabled 'just in case'**
 - If it's off, and no-one monitors space, the workload could stop
 - Monitor auto-growth using SQL Trace or Extended Events
 - Always have file growth to be a fixed amount, not a percentage

Auto-Shrink

- **When to use? NEVER!**
 - About the only thing there is NOT an 'it depends' answer for
 - This should be one of the first things that gets checked when you take over a new database
- **Why not?**
 - Data file shrink (same code as auto-shrink) is almost guaranteed to introduce index fragmentation within the data files
 - The database will most likely just auto-grow again, and then auto-shrink, auto-grow in a cycle that wastes resources
 - You can't control when it kicks in and affects performance
 - Blog post: <http://www.sqlskills.com/blogs/paul/auto-shrink-turn-it-off/>
(<http://bit.ly/fyIBzd>)
- **Again, always make sure auto-shrink is turned off**

Demo

Impact of shrink on index fragmentation

When to Use Data File Shrink?

- **Should be a very rare operation**
 - Because it causes index fragmentation
- **Three scenarios:**
 - Emptying a file before removing it
 - When a large amount of data has been deleted AND the space won't be reused
 - Moving a file/filegroup/database to read-only
- **Even then, shrink may not be the best method**
 - Remember – it causes fragmentation
- **So, how to perform a data file shrink?**

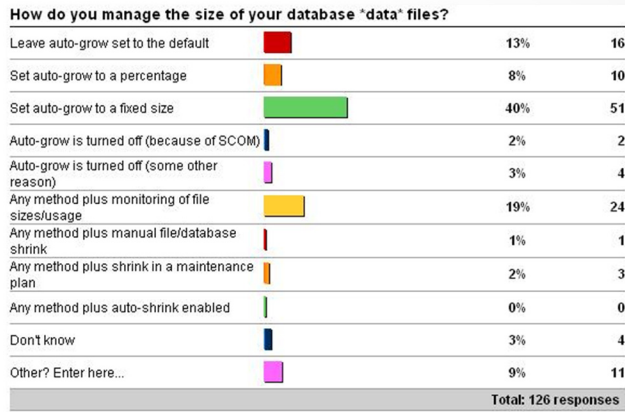
How to Shrink a Data File?

- **ALTER INDEX ... DISABLE nonclustered indexes**
- **Create a new filegroup and move all clustered indexes into it**
 - Use CREATE INDEX... WITH DROP_EXISTING and specify the location to be the new filegroup
 - Doesn't move LOB data, but see <http://bit.ly/H75AWL>
 - Can be performed online
- **If any table are heaps, use shrink to move them**
 - Heaps are unordered so shrink does not fragment them
 - Beware shrink is very slow for heaps and LOB data
- **ALTER INDEX ... REBUILD nonclustered indexes**
- **Drop old filegroup**
- **If you have to shrink a data file, use ALTER INDEX ... REORGANIZE to remove index fragmentation**
 - Or consider disabling then rebuilding them after the shrink

Common Shrink Misuse

- **Using shrink in a regular maintenance plan (or auto-shrink and auto-grow enabled)**
 - Leads to shrink-grow-shrink-grow cycle
 - Causes massive fragmentation
 - MS Dynamics CRM does this – be aware!
 - <http://bit.ly/x9a8Pa>
- **Using shrink after an index rebuild to reclaim space**
 - As we'll see in the index fragmentation module, an index rebuild needs space to build a new copy of the index, maybe growing the file
 - Shrinking after a rebuild will reclaim that space BUT will undo the effects of the index rebuild, wasting a lot of time and resources

Data File Management Survey



- Source: <http://www.sqlskills.com/blogs/paul/importance-of-data-file-size-management/> (<http://bit.ly/ZtvSD>)

Overview

- Physical layout considerations
- Allocation algorithms
- Instant initialization
- Auto-grow
- To shrink or not to shrink?
- Data compression
- Tempdb

Data Compression: The Problem

- **Cost of storage rises with database size**
 - High-end storage is expensive
 - Multiple copies required – test, HA, backups
- **Cost of managing storage rises with database size**
 - Time taken for backups and maintenance operations (I/O bound)
 - Time taken to restore backups in a disaster
- **Migration from other platforms (Oracle or DB2) that support compression is hard without compression**
 - Reduced TCO of SQL Server can be outweighed by increased storage costs from 3-5x increase in database size
- **Compressing data on-page leads to better memory utilization**
 - And even log space reduction
 - <http://www.sqlskills.com/blogs/joe/a-small-scale-test-measuring-the-impact-of-data-compression-on-the-transaction-log/> (<http://bit.ly/T26GsS>)

Do You Need Data Compression?

- **Data compression vs. backup compression**
 - Data can be compressed “at rest”, which removes the need to do backup compression every time the database is backed up
 - But, there is a trade-off... compressed data needs more CPU to access, but fewer I/Os
 - It all depends on the data being compressed and the access patterns
- **Data compression vs. removing fragmentation**
 - Indexes may get significantly smaller just by removing fragmentation rather than turning on compression
- **WP: Data Compression: Strategy, Capacity Planning and Best Practices**
 - <http://msdn.microsoft.com/en-us/library/dd894051.aspx>

Is Data Compression Suitable?

- Gain from data compression comes from having to perform fewer I/Os, tradeoff against CPU
- Data is compressed on disk AND in memory so each access of the data must decompress it
- Compression is best suited to access patterns where data is read into memory and used once
 - Data warehouse/reporting applications are most suitable
 - Highly-volatile OLTP applications are not usually suitable
- Questions to ask:
 - Is the I/O vs. CPU tradeoff worthwhile?
 - Will there be a significant space saving?
 - Are cost savings more important than potential workload impact?

How Far to Go?

- Data compression is a trade-off between storage size decrease and CPU increase
 - No point compressing as far as possible if the CPU increase proves detrimental to workload throughput
- Data compression ratio depends most heavily on the schema and data distribution
 - A 4-byte column with a 1-byte value can be compressed to 1 byte
 - A 4-byte column with a 4-byte value cannot be compressed
 - If all values are 1-byte in a 4-byte column, compression ratio is 75%
 - If all values are 4-bytes in a 4-byte column, compression ratio is 0%
- When comparing compression ratios, consider the CPU cost AND data being compressed

What Can Be Compressed?

- A whole heap
- A whole clustered index
- A whole non-clustered index
- A whole indexed view
- **Single partitions of partitioned tables and indexes**
 - Different partitions can have different compression settings
- System tables cannot be compressed
- How do you choose what/when to compress?

Estimating Space Savings

- **Enabling data compression on existing data can be very costly**
 - Basically the partition, index, or table is rebuilt
- **It makes sense to evaluate the potential savings before enabling compression, using the SP**
 - `sp_estimate_data_compression_savings`
- **Creates a 5% sampled subset of the data in tempdb and compresses it using the requested compression mechanism**
- **Returns the estimated size of the table/index/partition with the requested compression, plus details of the sample size**
 - Can also be used when turning OFF compression
- **Note: A storage decrease from compression is not guaranteed**

ROW Compression

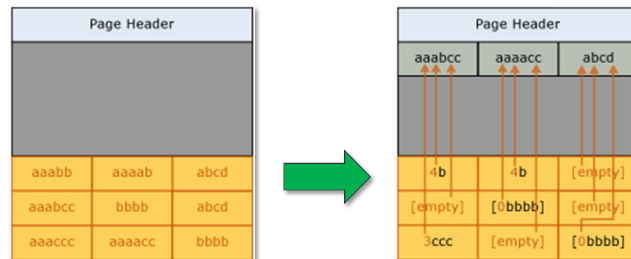
- **Stores fixed-length numeric data as variable-length**
 - E.g. integer, decimal, float, datetime, money
 - Super-set of vardecimal storage format (added in SQL Server 2005 SP2)
 - E.g. a bigint holding the value 34 will only store 1 byte
- **Stores fixed-length character data as variable-length**
 - Blank characters are not stored
 - E.g. a char (50) holding 'Paul Randal' will only store 11 bytes
- **A new variable-length row format is used to bring the per-column metadata overhead down from 2-bytes to 4-bits**
 - Null and zero values can be stored only using the 4-bits of metadata
- **Details of row compression can be found in Books Online**
 - 'Row Compression Implementation' in BOL index

PAGE Compression

- **Page compression does three things:**
 - ROW compression
 - Per-column prefix compression
 - Per-page dictionary compression
- **When compressing a page, these three operations are done in the order listed above**
- **Page compression adds a record at the start of each page called a compression information structure, or CI**
- **Details of page compression can be found in Books Online**
 - 'Page Compression Implementation' in BOL index

PAGE Compression: Prefix Compression

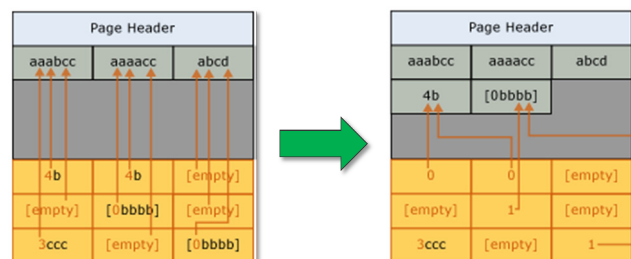
- For each column, a value is chosen that allows space reduction and is stored in the compression information structure (CI)
- The in-row values are replaced with indicators of full or partial matches with the value in the CI



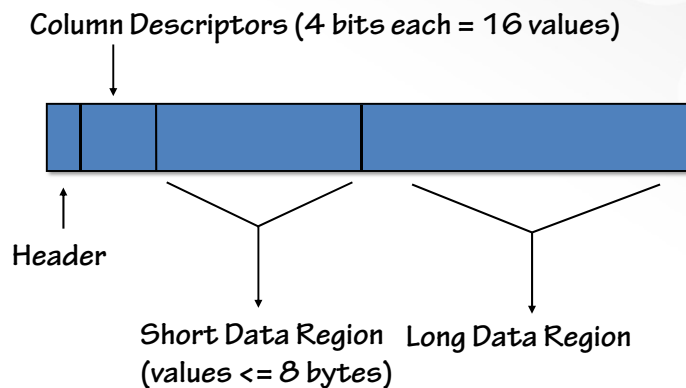
- Note that the largest value for each column is stored in the CI
- The process uses byte-level comparisons across all data types

PAGE Compression: Dictionary Compression

- Dictionary compression is done AFTER prefix compression
- The whole page is scanned looking for common values, which are stored in the CI area on the page
- The in-row values are replaced with pointers to the CI area



Record Structure



- Column descriptors give length, plus special values such as zero or null
- 'Jump points' every 32 columns to speed up access

When is Data Compressed?

- After PAGE compression is enabled, new pages are only ROW compressed until they fill up, then the next row insertion triggers page compression
- The page is PAGE compressed, and if there's 20% space savings after compression has occurred (and the CI structure added), the next row is compressed and inserted
 - If there's not enough space saving, the page is not compressed and the row will be inserted on a new page
- When a table or index is rebuilt with PAGE compression on, PAGE compression occurs as part of the rebuild, but follows the same process
 - i.e. there may be some uncompressed pages if PAGE compression was not worthwhile
- Per-page modification counter triggers re-compression of page

Enabling and Disabling

- **Compression can be enabled by specifying compression ROW or PAGE through:**
 - CREATE TABLE or CREATE INDEX
 - ALTER TABLE ... REBUILD (new syntax)
 - ALTER INDEX ... REBUILD
 - Data Compression Wizard in SSMS
- **Compression can be disabled by specifying compression NONE through:**
 - ALTER TABLE ... REBUILD
 - ALTER INDEX ... REBUILD
 - Data Compression Wizard in SSMS
- **Both can be done ONLINE if required**

Example Syntax For Partitions

- **Create a partitioned index with different compression settings on various partitions (unspecified partitions will use NONE)**
 - CREATE CLUSTERED INDEX ClustIDX ON MyTable (Col1) WITH (DATA_COMPRESSION = ROW ON PARTITIONS (1), DATA_COMPRESSION = PAGE ON PARTITIONS (2 TO 4))
- **Modify a single partition's compression setting, with only that partition being rebuilt**
 - ALTER INDEX ClustIDX ON MyTable REBUILD PARTITION = 1 WITH (DATA_COMPRESSION = NONE)
- **Modify one or more partitions, with all partitions being rebuilt (unspecified partitions retain their current compression setting)**
 - ALTER INDEX ClustIDX ON MyTable REBUILD WITH (DATA_COMPRESSION = NONE ON PARTITIONS (1))
- **Check the compression state of partitions using the data_compression column in sys.partitions**

Notes on Options and Syntax

- Nonclustered indexes do not inherit the base table's compression setting – they must be compressed individually
- All partitions in an index or table will have the same compression setting unless otherwise specified
- When a clustered index is created on an existing table, it inherits the existing compression setting of the heap
- When a clustered index is dropped, the heap inherits the setting of the clustered index
 - So the pages at the leaf-level of the clustered index do not have to be uncompressed as part of dropping the clustered index
- **NOTE: Changing the compression setting of a heap will cause all non-clustered indexes to be rebuilt (as the heap rows move)**

Side Effects

- **Enabling or disabling compression rebuilds the heap, clustered index, or non-clustered index**
 - Same disk and log space requirements as an index rebuild
 - Changing the compression setting of a heap will cause all non-clustered indexes to be rebuilt
 - Beware of changing compression settings on systems with HA technologies because of extra log generation
- **When bulk exporting from a compressed table, the data will be exported uncompressed, which could be a lot larger than expected**
- **The transaction log only logs ROW compressed values, the entire page compression row is logged when created**
- **When a page splits, the new page will have the same compression info as the old one (initially at least)**

Limitations

- **Data compression is Enterprise Edition only**
 - Restoring a backup with compressed data in will fail on a lower edition (same as happened with partitioning in SQL Server 2005)
 - Check the sys.dm_db_persisted_sku_features DMV
- **Data compression cannot be used with sparse columns**
- **Data cannot be entered that would overflow the maximum row size when uncompressed**
 - This ensures that disabling compression will always succeed
- **LOB/FILESTREAM values out-of-row are not compressed**
- **Bulk-exported data will be exported uncompressed**
- **Remember: enabling or disabling requires a size-of-data rebuild!**

Customer Experiences with Data Compression

Customer	Space Savings	Throughput impact	Notes
Customer #1	40%	5%	PAGE compression. OLTP web application. Large volume of transactions.
Customer #2	62%	40%-60%	PAGE compression. DW application. Large sequential range queries .
Customer #3	38%	-1%	PAGE compression. OLTP with some reporting. 500 users, 1,500 trans/sec.
Customer #4	80%	-11%	PAGE compression. OLTP application. A lot of insert, update and delete activity.
Customer #5	52%	2% - 3%	PAGE compression. OLTP Application.
Customer #6	81%	3%	PAGE compression. ERP application – small transactions.

Your Mileage Will Vary!

Is Compression Working For You?

- You might be burning a lot of CPU attempting compression on non-compressible data
- Check in `sys.dm_db_index_operational_stats`
 - `page_compression_attempt_count`
 - `page_compression_success_count`
- Join with `sys.partitions` to limit results to only tables/indexes with compression enabled
 - `data_compression_desc = 'PAGE'`

Overview

- Physical layout considerations
- Allocation algorithms
- Instant initialization
- Auto-grow
- To shrink or not to shrink?
- Data compression
- Tempdb

Traditional Tempdb Uses

- **User objects:**
 - Table variables (@)
 - #temp and global ##temp tables/indexes
 - Regular tables
- **Internal objects**
 - Worktables (e.g. sort, intermediate results, ...) from memory spills to disk
 - Workfiles (only for hash joins and hash aggregates) from spills to disk
 - Intermediate sort results from index create/rebuilds
 - Only if SORT_IN_TEMPDB is specified
 - Intermediate DBCC CHECKDB results (in a worktable)
 - Version store (main one + online index operations one)
- **Full list and explanations in BOL: Capacity Planning for Tempdb**

Version Store

- **Stores all version records (as described in Module 1) created in all databases**
- **One new allocation unit created every minute**
 - Variable size depending on how many versions created in that minute
- **Cleanup task runs every minute or so**
 - If all version records in allocation unit are no longer required, it can be deleted
- **Non-logged**
- **Long-running transactions using snapshot isolation can interrupt cleanup and lead to tempdb growth**
- **See Books Online: Row Versioning Resource Usage**
 - <http://msdn.microsoft.com/en-us/library/ms175492.aspx>

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 Sizing and Restart

- Tempdb reverts to the last specifically set size after a server restart
- If tempdb grows during operations, that is not the same as specifically setting the size
- Avoid excessive auto-growth after a server restart by manually setting the size

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 I/O, but generally will give a performance boost to an I/O subsystem that's overwhelmed
- **Page checksums on tempdb weren't available until SQL 2008**
 - On by default for new instances of SQL 2008 onwards
 - Must enable for tempdb for upgraded instances
- **Can be put on RAM drive (no durability requirements)**
 - See KB 917047 (<http://support.microsoft.com/kb/917047>)
- **In 2012, tempdb in a cluster can be on non-shared storage**
- **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 (wait type)
 - Thread waiting for a page to be read from disk
 - PAGELATCH (wait type)
 - 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
 - Avoiding excessive use of tempdb

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_UP or _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 or primary key constraints (e.g. in TVPs)
 - Fixed somewhat from 2008 R2 CU9
- Contention can occur in all versions
- This can sometimes (rarely) happen in user databases with VERY high-end allocation workloads

Tempdb Allocation

- There is a cache of worktables and temp tables (from 2005 onwards)
 - Worktable cache is very small
 - Temp table cache is (relatively) unbounded
- Mechanism:
 - When a temp table/worktable is dropped, a single IAM page, a data page/extent, and its metadata entry remain
 - As long as temp table is less than 8MB when dropped
 - Only for temp tables NOT created by ad hoc statements
 - The next temp table/worktable creation pulls one out of the cache
 - All kinds of reasons a temp table may not be cached
- See Paul White's detailed blog post:
 - http://sqlblog.com/blogs/paul_white/archive/2012/08/17/temporary-object-caching-explained.aspx (<http://bit.ly/QrCYcG>)

Alleviating Tempdb In-Memory Contention

- Trace flag 1118 (discussed in KB 328551) removes use of mixed extents (SGAM contention) in all databases
 - **All instances across the world should have this enabled (and T3226)**
- Create multiple data files to spread and reduce contention
 - For 2000, guidance was # data files = # processor cores
 - For all other versions:
 - Older guidance: # data files = $\frac{1}{4}$ to $\frac{1}{2}$ # processor cores and add more if needed
 - Best guidance from Bob Ward (CSS) at PASS 2011:
 - < 8 cores, use #files = #cores
 - > 8 cores, use 8 files, increasing by chunks of 4
 - Documented as Microsoft official advice in KB 2154845
 - Be careful of having too many data files
- **Note: this does not apply to log files**
- Adding one file may not work (see <http://bit.ly/1uNmHbk>)
- Investigation article on Simple Talk: <http://bit.ly/16la3n8>



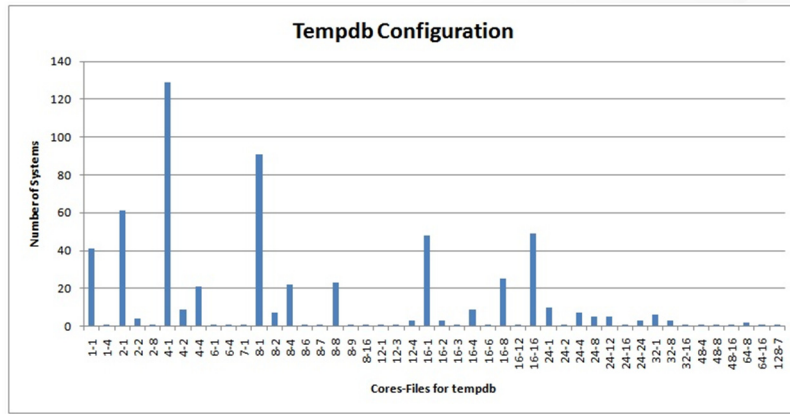
63

© SQLskills, All rights reserved.
<http://www.sqlskills.com>

Demo

Simple tempdb contention

Tempdb Configuration Survey



- Source: <http://www.sqlskills.com/blogs/paul/tempdb-configuration-survey-results-and-advice/> (<http://bit.ly/dRZzKb>)

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> (<http://bit.ly/oq0VH>)
- Blog post: Misconceptions Around TF 1118
 - <http://www.sqlskills.com/blogs/paul/misconceptions-around-tf-1118/> (<http://bit.ly/vNMLv>)
- Blog post series on tempdb
 - <http://www.sqlskills.com/blogs/paul/comprehensive-tempdb-blog-post-series/> (<http://bit.ly/T29Cpr>)
- 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>

Review

- Physical layout considerations
- Allocation algorithms
- Instant initialization
- Auto-grow
- To shrink or not to shrink?
- Data compression
- Tempdb

Questions!