

SQLskills Immersion Event

IE3: High Availability & Disaster Recovery

Module 3: Backup Strategy and Internals

Kimberly L. Tripp
Kimberly@SQLskills.com



Overview

- **Planning a backup strategy**
- **Backup types**
 - Full/differential/log
 - VLDB strategy concerns
 - Filegroup/file backups
 - Filegroup/file differential backups
- **Special features for backups**
 - COPY_ONLY
 - Parallel striped backup/multi-file backups
 - Mirrored backups
 - Backup integrity
 - Backup compression
- **Log shipping**



Why Take Backups?

- **To allow a restore to take place**
 - Disaster recovery
 - Log shipping
 - Initializing database mirroring or Availability Groups
 - Initializing replication
 - Transferring a database to development
 - Moving/upgrading a database
- **To manage the size of the transaction log**
- **Just in case...**
 - Before performing a database repair
 - Before performing an upgrade of any kind
 - Hardware, OS, SQL Server

How to Plan a Backup Strategy?

Don't Do It.

Plan A Restore Strategy

- Never plan a backup strategy
- Always plan the restore operations you need to be able to do, and that will dictate what backups you need and what recovery model to use
- This usually turns out to be FULL recovery model, with a full + log + differential backup strategy for most databases...
- VLDBs may look at more granular backups including:
 - Filegroup backups, filegroup differentials and log backups
 - File backups, file differentials and log backups
- Planning your backups without considering restore can lead to not meeting your SLAs (more downtime) and possibly data loss

Don't Forget System Databases

- If you need to do a bare-metal install, the system databases may be crucial
- master: info about all databases, logins, configuration, settings
- msdb: SQL Agent jobs and history, backup history, SSIS packages
- model: the blueprint for new databases
- resourcedb: essentially a DLL and can only be copied as a file
- Replication distribution databases (if applicable) should be backed up to allow replication to recover without a re-initialization
- You need the entire application ecosystem for recovery to be successful

Overview

- Planning a backup strategy
- **Backup types**
 - Full/differential/log
 - VLDB strategy concerns
 - Filegroup/file backups
 - Filegroup/file differential backups
- **Special features for backups**
 - COPY_ONLY
 - Parallel striped backup/multi-file backups
 - Mirrored backups
 - Backup integrity
 - Backup compression
- **Log shipping**

How to Backup

- **Manually use appropriate BACKUP statement for type of backup performing**
- **Use UI for one-time backup**
- **Use UI to create scheduled backups**
- **Use UI to create a script from which to build a backup**
- **Use Database Maintenance Plan Wizard**
- **Use SMO**
- **Create backup devices to simplify naming**
 - Local: do not use UNC names!
 - Network: use FULLY QUALIFIED UNC names

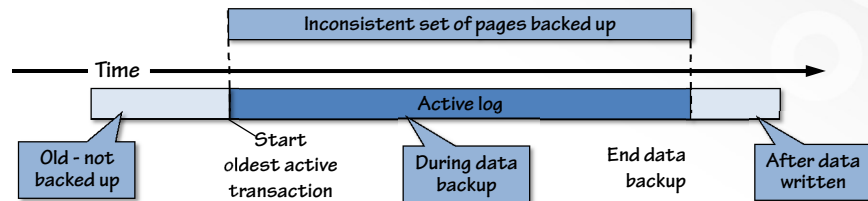
Backup Types

- **Full: backing up the entire set of data regardless of granularity. Can be performed at:**
 - Database-level
 - Called a full database backup or database backup
 - Filegroup-level
 - Called a full filegroup backup or filegroup backup
 - File-level
 - Called a full file backup or file backup
- **Differential: backing up all of the changes since the last time the SAME granularity of backup was performed. Can be performed at:**
 - Database-level
 - Called a differential database backup, database differential backup, differential backup, or just diff backup
 - Filegroup-level
 - Called a differential filegroup backup
 - File-level
 - Called a differential file backup

Full Database Backups

- Creates image for a possible recovery starting point
- Complete image of all data, etc. (less free space)
- Does not require a backup 'window'
- Usually NOT used alone as other backups must be used to roll-forward to a later point in time
- Backup image is backup completion, meaning that if restored, the image will be the committed changes up to the time that the backup finished...how?

How Full Database Backups Work



- Data is read sequentially
- Data is not changed, it's a straight copy
 - Pages with page checksums MAY be processed...
- Data is written directly to first available backup device (if parallel striped backup)
- Log backups can occur concurrently (from 2005), however, log clearing is deferred until full backup completes

How Full Database Backups Work

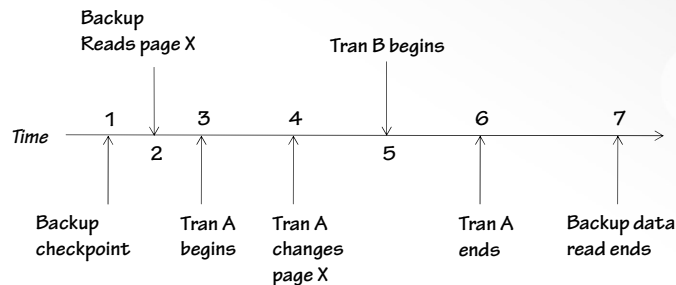
1. Checkpoint: to make sure that pages on disk are up to date with current changes
2. Mark the log: backup began here
3. Read from data files: reads pages directly from disk, even though they might be changing WHILE the backup is occurring therefore backup is considered an inconsistent set of pages (NOT a problem as the log will allow for transactional consistency)
4. Mark the log: backup ended here
5. Copy the transaction log: backup includes transaction log to make consistent. No point making the data consistent on the backup as we only need this information on the restore...

All of this means that transactions NEVER wait

How Much Log is Necessary?

- Enough transaction log must be included in the backup to allow the database to be restored and recovery performed to make the database transactionally consistent
- Minimum of all log from the end of the data-reading portion of the backup all the way back to the oldest of:
 - Last checkpoint
 - Beginning of oldest active transaction
 - Beginning of oldest un-replicated transaction
 - I.e. not yet scanned by the replication Log Reader Agent

Log Contained in a Full Backup



1. Checkpoint and begin reading from the database.
2. The read operation reads page X
3. Transaction A starts
4. Transaction A makes a change to page X. The copy in the backup is now out-of-date. Note that the backup will not read page X again - it's already passed that point in the database.
5. Transaction B starts. It does not complete before the data read operation completes.
6. Transaction A commits. This commits the changes to page X.
7. The backup data read operation completes and transaction log reading starts. Log is read back to the oldest of (last checkpoint, begin of oldest active tran): point 1

Full Database Backup Concerns

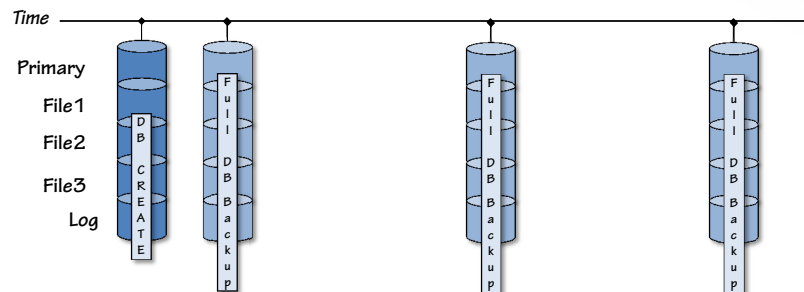
- **Large database**
 - Time: reading a lot of data takes a lot of time
 - Space: wasting space on backup media with data that has not changed
 - Solution: partitioning or backup compression
 - Cost: in terms of time, space and media storage
- **Log backups do not clear the log during full (or diff) backups**
 - Log can fill and/or cause significant auto-growth
 - Log needs to be as large as space required to hold active transactions during full backup
 - Longer the full, larger the log
 - Log clearing is deferred until the data backup completes

Concurrent Log and Data Backups

- **Before SQL Server 2005, log backups would wait for database-level full or differential backups to complete**
 - This is because a log backup clears the log, potentially including the log required by the full/differential backup
- **SQL Server 2005 fixed the backup logic so that concurrent log backups can be taken during full and differential backups**
 - The log clearing is deferred until the concurrent log or differential backup completes
 - Upon completion of the full/differential backup, the log manager performs the log clearing algorithm, giving rise to the misconception that a data backup clears the log

Full Database Backup Only Strategy

- Full backups only provide a single point in time to which the database can be restored
- All work since the last full backup is lost

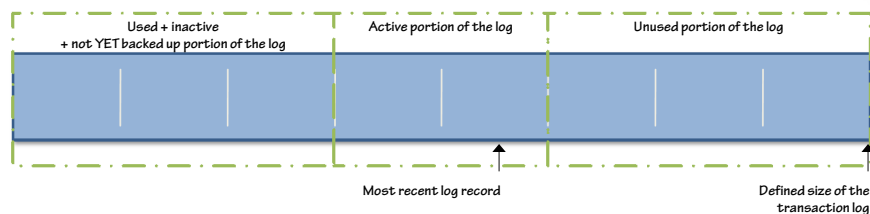


Transaction Log Backups

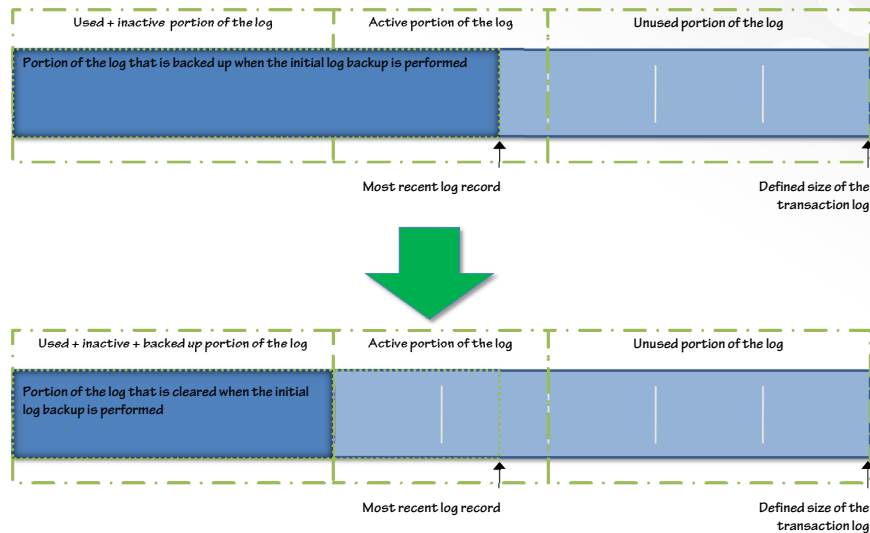
- Information about the changes since the last transaction log backup (can be described as incremental)
- Taken at frequent intervals therefore usually fairly small
- Once backed up, SQL Server clears the inactive portion (the inactive VLFs) of the log, if possible
 - This can help to keep the transaction log file small and manageable but SQL Server does NOT release the file's free space
- Minimal impact to current users because only inactive information may be cleared
- Key purposes
 - Allow roll-forward to point of failure
 - Allow recovery to an arbitrary point in time
 - Support log shipping, either through automation wizard or manually backed up and copied/shipped to a secondary location

Initial Log Backup

- Remember the active and inactive portions of the transaction log
- The first log backup will back up everything from the start of the transaction log to the most recent log record
- Only the inactive portion of the transaction log can be cleared
 - Note: the white vertical lines are VLF boundaries in the transaction log

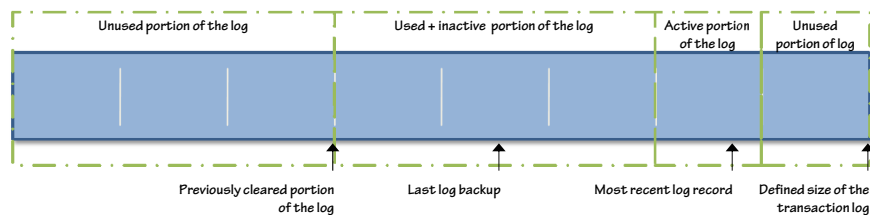


Initial Log Backup (2)

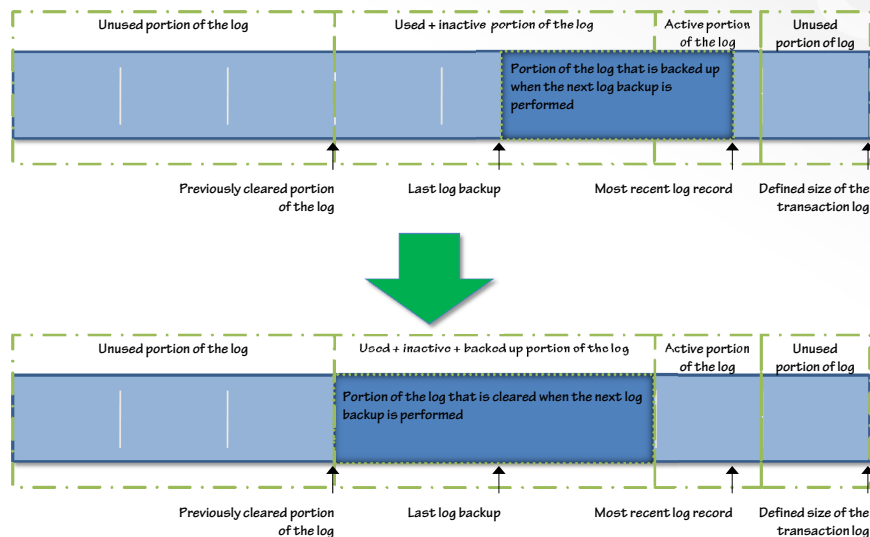


Subsequent Log Backups

- Once logging activity moves forward in the transaction log then the previously active VLFs are now essentially unneeded and can be cleared the next time log clearing happens
- The next log backup will back up all of the activity since the last log backup and then attempt to clear the currently unneeded portion of the transaction log
 - VLFs do not become inactive until log clearing occurs

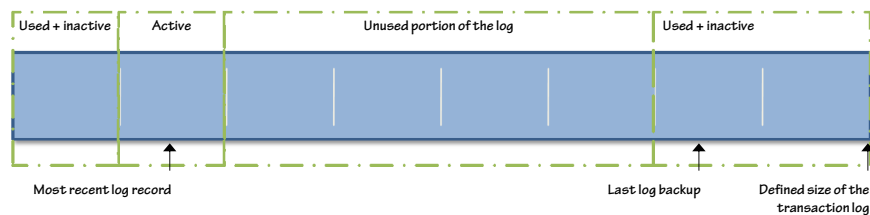


Subsequent Log Backups (2)

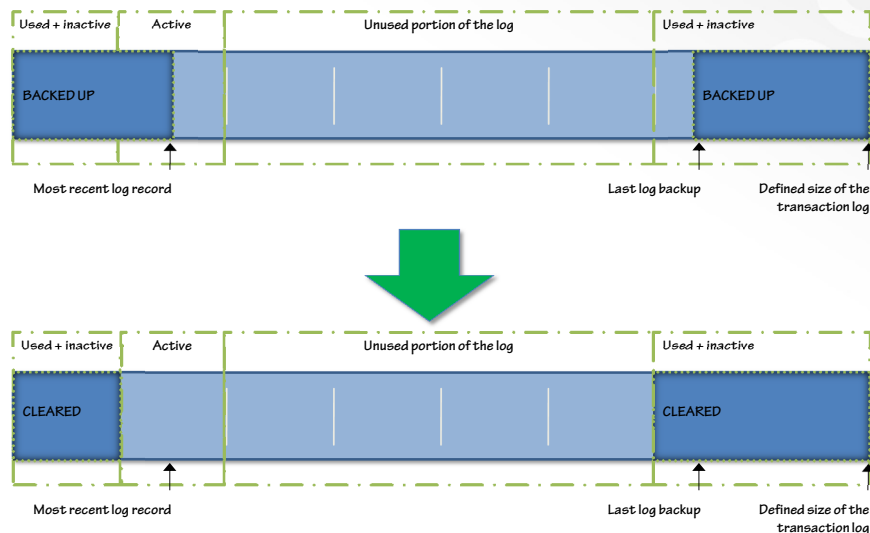


The Transaction Log Eventually Wraps

- Eventually the transaction log wraps and starts to reuse the cleared and inactive VLFs at the beginning of the transaction log file
 - The transaction log is circular nature
 - If log backups had not been performed, the transaction log would not have been cleared and the wrap operation would not have been possible, necessitating transaction log growth



The Transaction Log Eventually Wraps (2)



Demo

When does the log get cleared?

Full Recovery Model Transaction Log Management Required

- If you're **ONLY** performing full database backups then FULL (or BULK_LOGGED) recovery model is **NOT** recommended as the log is **NOT** being maintained (it will eventually fill)
- **Manage the transaction log:**
 - Using SIMPLE recovery model; OR
 - Perform transaction log backups
- **Some people use manual commands to clear the log:**
 - BACKUP LOG WITH TRUNCATE_ONLY or
 - BACKUP LOG WITH NO_LOG
- **NOTE: These do NOT exist in SQL Server 2008+**

Trace Flags To Stop Log Dumping

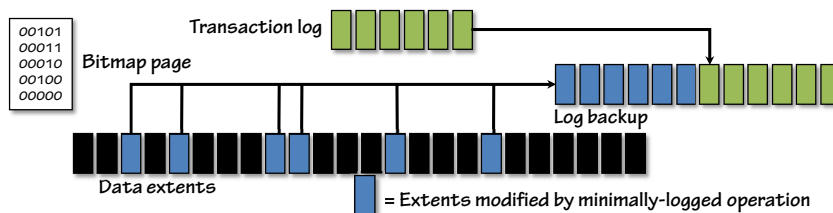
- To change the behavior of BACKUP LOG WITH TRUNCATE_ONLY or NO_LOG:
- In SQL Server 2000
 - Trace flag –T3231
 - In FULL or BULK_LOGGED recovery model they do nothing
 - In SIMPLE recovery model they will clear the log
- In SQL Server 2005
 - Trace flag –T3231 works the same way as in 2000
 - Trace flag –T3031 does a checkpoint
- In SQL Server 2008
 - Manually clearing the log is no longer possible, but some people may still do it by switching back-and-forth to SIMPLE recovery model

Continuity of the Log Backup Chain

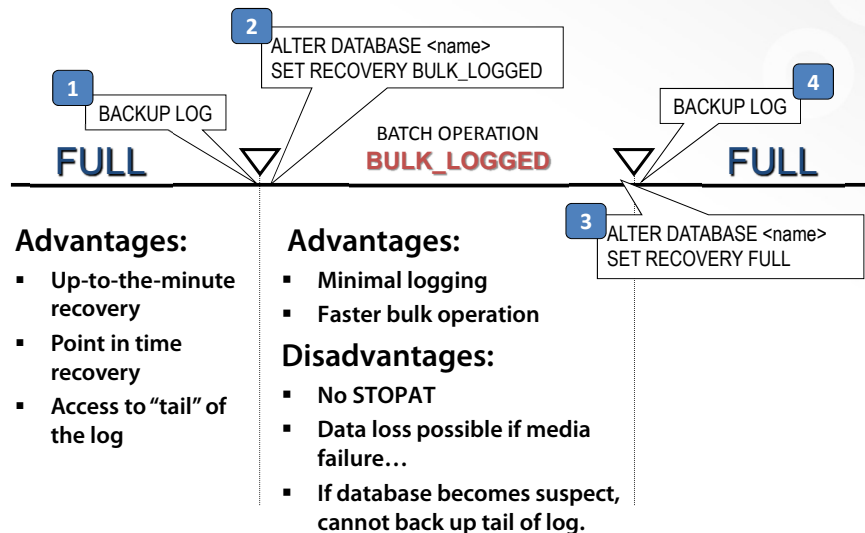
- If the log backup chain is not complete (meaning one or more log backups are damaged or missing) then complete recovery is not possible
 - Must have complete chain of log backups from which to restore
- Consider using mirrored backups of the transaction log and/or storing them on redundant disks
- What breaks the log chain?
 - Clearing the transaction log manually (using WITH TRUNCATE_ONLY or WITH NO_LOG)
 - Changing the database to SIMPLE recovery model (and then changing back)
 - Deleting, overwriting or throwing away a log backup that was not made as COPY_ONLY
 - Reverting from a database snapshot

Log Backups After Minimally-Logged Operations

- If there has been a minimally-logged operation since the previous log backup, the next log backup will also back up all data extents modified by the minimally-logged operation
 - This means the minimally-logged operation can be fully reconstituted
- The resulting log backup will be roughly the same size as if the operation was performed in the FULL recovery model
 - A point-in-time restore operation cannot stop at any point between the start and end of the log backup



Switching Recovery Models

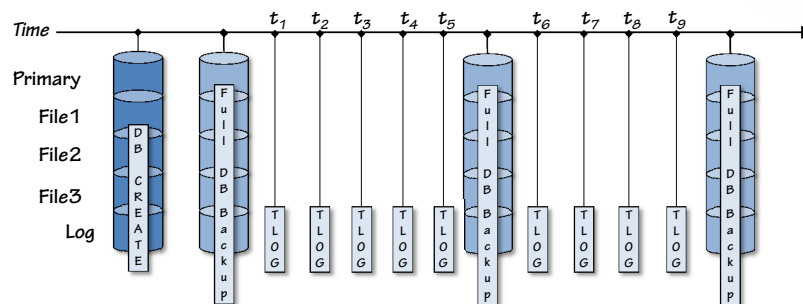


Full Database + Log Backup Strategy

- **Relatively common configuration**
 - Easy to setup
 - Easy to control
 - Easy to recover
- **Costly to manage**
 - Takes more time
 - Requires more space than just full backups

Full Database Backup Plus Log Backups Strategy

- Up-to-point-of-crash recovery is possible with no data loss
- Possible to work around a damaged full backup if prior log backups are still available
- Restoring a large number of log backups can take a significant amount of time so may need to include differential backups



Tail-Of-The-Log Backups

Up-To-The-Point-Of-Crash Recovery

- This is the best case!
- If the transaction log is accessible and the data portion of the database is damaged (in part or entirely) then you might be able to back up the transaction log
- IF you can backup the transaction log AND you have all of the transaction log backups since the last full AND you're in FULL recovery model (or BULK_LOGGED but no minimally logged operations have occurred) THEN up-to-the-minute recovery is possible and with NO DATA LOSS!

Demo

Tail-of-the-log backups

How to Backup the Tail of the Log?

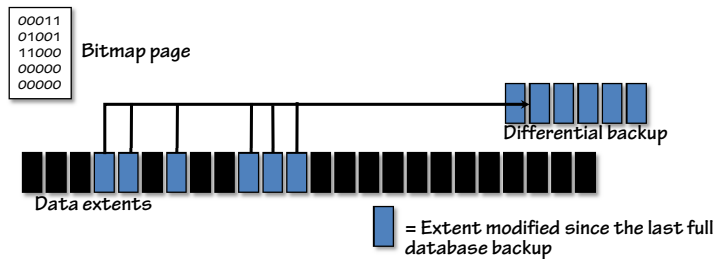
- **Use the NO_TRUNCATE clause with the BACKUP LOG command**
 - `BACKUP LOG dbname TO device1 WITH NO_TRUNCATE`
- **Through SSMS**
 - Right-click on a working database, choose Tasks, Backup Database... , select the damaged database from drop-down
 - Choose the Options 'page' on backup dialog
 - Under the Transaction Log section
 - Truncate the transaction log by removing inactive entries
 - **SELECT THIS ONE >>>** Backup the tail of the log, and leave the database in a restoring state
 - i.e. `BACKUP WITH NO_TRUNCATE, NORECOVERY`
- **Important: If you want to keep your database 'partially available' you should use the Transact-SQL syntax instead. You do NOT want your database put into the restoring state**

Differential Database Backups

- **Similar to full backup except only includes the extents modified since the last full backup**
 - I.e. differential backups are NOT incremental, they are cumulative
- **Time to create the backup is proportional to the number of extents modified since last full backup.**
- **Excellent to reduce roll-forward time on restore**
- **Faster than restoring the comparable transaction log backups for same timeframe**
- **How big will the next differential backup be?**
 - <http://tinyurl.com/njd6jr>

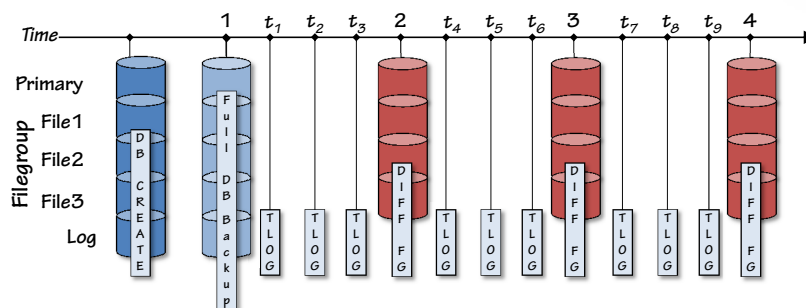
How Differential Backups Work

- Modified extents in files are 'marked' in the DIFF map
 - Similar to GAM page – one per GAM interval
- Bitmap for file is reset when file backup, filegroup backup or full backup – backs up that portion of the database
- TIP: Use COPY_ONLY for a full database backup copy which does NOT impact the differential bitmap!



Full, Diff Database, and Log Backup Strategy

- Differential database backups are only the extents changed since the last full database backup
- Simplifies recovery time by allowing you to recover the database, the last differential, and only the logs after that, until the time of the disaster

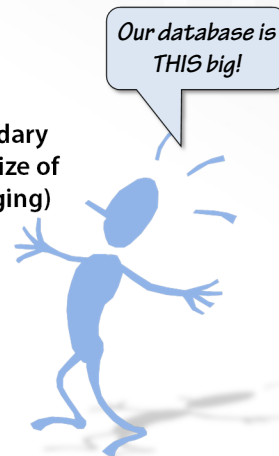


Granular Backups

- Full and differential backups can be performed at three levels:
 - Database
 - Filegroup (or list of filegroups)
 - File (or list of files)
- Can specify just **READ_WRITE_FILEGROUPS** to create a partial backup (either full or differential) of ALL the filegroups NOT set to read-only
 - Used when you want to run with SIMPLE recovery model but want to do more granular backups
 - If a filegroup becomes read/write after the partial full backup was taken, but was not included in the partial full backup, a partial differential will fail
- Can make for faster recovery, but more complex to manage

VLDB Issues

- VLDB: Very large database (100+ GB)
- VLT: very large table (well into double digit GB)
- More space/time required for many operations
- Typically large portion not changing often
- Large database can add to importance of secondary site (although this is not really size related but size of the database makes this problem more challenging)



VLDB Creation Techniques

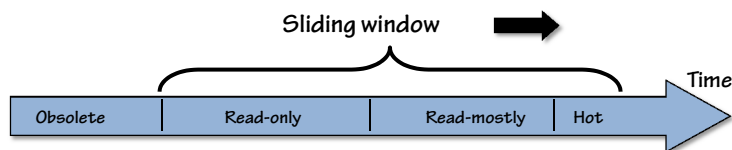
- **Partitioning can offer:**
 - Smaller, more granular backups (i.e. table backups/table restores)
 - Smaller, more granular (and possibly little to no) index maintenance for some partitions
 - Less contention due to partition-level lock escalation
- **Partitioned views (SQL Server 7.0+)**
- **Updateable partitioned views (SQL Server 2000+)**
- **Partitioned tables (SQL Server 2005+)**
- **Best architecture often combines techniques**

How is This Possible?

- Fine grain operations are based on 'partitioning' datasets for VLDB
- Partitioning in this sense does not require the SQL Server 2005 partitioned tables feature however, this feature significantly benefits from these capabilities
- Partitioning for fine grain operations just means strategically placing objects within filegroups to recover the prioritized combination at time of disaster
- Strategies...

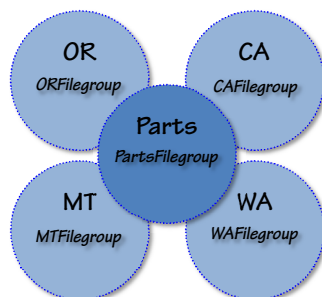
Time-Based Data Placement

- Structures designed for sliding-window scenario
- Tables created and data flows on regular/consistent basis, e.g. weekly, monthly, yearly, etc.
- Data may be archived/removed to keep only 'current' timeframe, e.g. year, two years, etc.
- Uses partitioned views or partitioned tables defining 'ranges' using date-based criteria



Related-Object Groupings

- Region-specific Data:
- Customers, Sales, SRs are found within the region-specific filegroup
- If CAFilegroup is damaged, customers in Oregon, Washington and Montana are not affected...
- However, damage to 'Parts' would mean downtime



Related-object groups = List-based or functional

- Regionally-based with some shared components
- Functionally-based – could use separate tables (with PV) OR Partitioned Tables using a list-based partition function

Database Structures

Partitioning Basics (1)

- **Database has at least two files, ALWAYS**
- **Data file**
 - First data file is the 'Primary' data file and stores system tables critical to this database's accessibility
 - A database will NOT remain available if this is damaged!
 - Critical to isolate (from other data, in a VLDB), create and locate on redundant array
- **Log file**
 - Where changes are stored until backed up
 - Also critical to database availability, a database will NOT remain available if this is damaged AND you can have data loss if/when this portion of the database is damaged
 - Critical to isolate and make redundant

Database Structures

Partitioning Basics (2)

- **Additional/secondary, non-primary, data files**
- **Basis for 'partitioning' data**
- **Exist in filegroups and are available for placement of objects by specifying the filegroups during object creation**
 - A file can ONLY be a member of one filegroup
 - Once added to the database, the filegroup CANNOT be changed (however, the number of files, size and location can; the file can always be removed from the filegroup)
- **Contain user-defined data (tables/indexes) strategically created/placed on one or more filegroups**
- **Contain complete objects, when the object has not been partitioned (an object CANNOT exist over multiple filegroups unless partitioned [created on a partition scheme])**
- **Contain a partition of a partitioned object**

Partitioned Tables and Indexes

Types and Implementation

- **Types of partitioning = 'range'**
 - Date ranges = defined through boundary cases
 - Uses deterministic values that are hard-coded values or calculated at creation time (based on a function(s))
 - Simulate list-based partitioning by using boundary point that have no values between them
- **Implementing partitioned tables and indexes**
 - Partition function
 - Partition scheme
 - Partitioned table
 - Partitioned index

Demo

Range-based partitioning: scenario setup/configuration

Benefits of Partitioning

- **Speed in managing sliding window**
 - ↳ Partition manipulation outside of active table
- **Piecemeal backup**
 - ↳ Backup active components more frequently, inactive less frequently
- **Availability**
 - ↳ If a secondary, non-primary, file (therefore filegroup) becomes unavailable, the other filegroups can still be accessed and recovery can occur concurrent with a workload
- **But what about data access or certain maintenance operations that create locking? Blocking scenarios limit availability...**

Demo

Recovering a damaged database while the database/system is online

Damaged Partition: Summary

- Does not render the database unavailable as only the damaged filegroup is unavailable
- Does not render the partitioned table or partitioned view unavailable as only the damaged data is unavailable but the partitioned object can still be queried/accessed
(NOTE: In SQL Server 2005, a partitioned view that has tables that are inaccessible used to generate an error when accessing the view; however, the tables on which the view are defined are still accessible.)
- Can proceed with recovery, also [almost] ONLINE!
 - There are a couple of hiccups here...
 - Minimal impact to downtime as files are brought offline and/or when file is restored
- Users can access the database during restore

Demo

Accessing a damaged database while part of it is damaged
and before it is repaired

Backup Strategy Survey

What kind of backups do you take of your SQL data, and in what recovery model?

SIMPLE recovery model, full backups only		17%	22
SIMPLE, full + differential backups		2%	3
FULL recovery model, full backups only		2%	3
FULL, full + differential backups		1%	1
FULL, full + log backups		45%	57
FULL, full + log + differential backups		26%	33
OS-level backups (e.g. VSS)		0%	0
I/O subsystem level backups (e.g. SAN snapshot)		1%	1
Shutdown SQL Server and copy the database files		0%	0
Don't take backups		0%	0
Other? Enter here...		6%	7
Total: 127 responses			

- <http://www.sqlskills.com/BLOGS/PAUL/post/Importance-of-having-the-right-backups.aspx> (<http://bit.ly/nd1Cv>)



53

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

Overview

- Planning a backup strategy
- Backup types
 - Full/Differential/Log
 - VLDB strategy concerns
 - Filegroup/file backups
 - Filegroup/file differential backups
- Special features for backups
 - COPY_ONLY
 - Parallel striped backup/multi-file backups
 - Mirrored backups
 - Backup integrity
 - Backup compression
- Log shipping



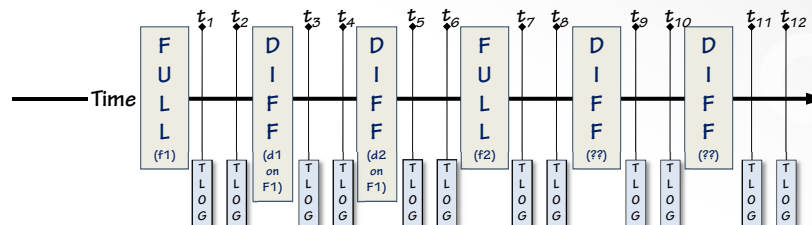
54

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

COPY_ONLY Backups

- A differential backup is everything since the last full
- A log backup is everything since the last log
- If an ad hoc full or log backup is taken and then lost, it may destroy your ability to recovery properly
 - E.g. taking a backup for a developer
- Taking a full backup resets the differential base GUID
- Taking a log backup clears that log
- Solution: use WITH COPY_ONLY for ad hoc backups

Differential Chain with/without COPY_ONLY



- If the full database backup at f2 is performed without using COPY_ONLY then
 - The differentials that follow cannot be used with full database backup performed at f1
 - The differential bitmap is reset at f2
- If the full database backup at f2 is performed using COPY_ONLY, then
 - The differentials that follow CAN be used with full database backup performed at f1
 - The differential bitmap is not reset at f2

COPY_ONLY vs. NO_TRUNCATE

- Sound similar... but, they're not the same!
- **COPY_ONLY will only work if the backup is not corrupt**
 - If it's corrupt you can use CONTINUE_AFTER_ERROR
- **NO_TRUNCATE**
 - A backup of the log without clearing the log
 - Designed for situations where the data portion is unavailable (tail of the log backups)
 - Works even if the log backup is damaged but **you won't know this** because NO_TRUNCATE implicitly does a CONTINUE_AFTER_ERROR even if you don't ask

+ NO_TRUNCATE should always work

- You won't really know if there's an error until you restore. You might be forced to use CONTINUE_AFTER_ERROR during the restore

Key point: Don't use NO_TRUNCATE for a copy



57

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

Parallel Striped & Multi-File Backups

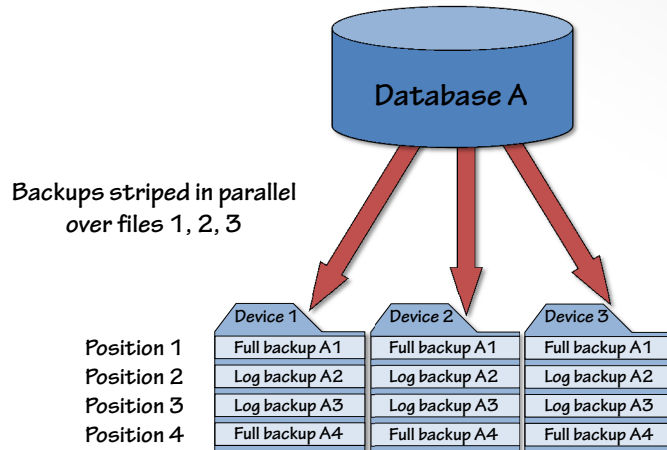
- **A backup can be striped across multiple backup files (a.k.a. devices)**
 - Backup will be written out in parallel
 - One thread per drive letter/mount point for reading and for writing
 - Usually increases backup performance
 - Also consider using BUFFERCOUNT and MAXTRANSFERSIZE to increase performance
- **Multiple backups can be contained within a single backup file (or files)**
 - Successive backups are appended after the previous ones
 - Oldest backup is in position 1 within the file(s), next is in position 2, etc
 - Backups can be different types and from different databases
 - We don't like to recommend this because of potential difficulties
 - Accidental overwriting, difficulty in finding correct backup



58

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

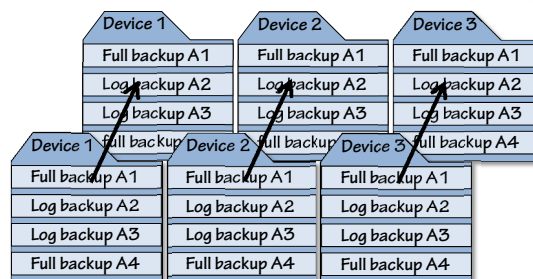
Parallel Striped/Multi-File Backups



Mirrored Backups

- Same device type for mirror
- Different disks and/or over the network
- Up to 4 total backups at one time (3 mirrors)

Mirror media set



Logical media set

How to Check Backup Integrity?

- **Use WITH CHECKSUM when taking a backup**
 - Test all page checksums in the database
 - Calculates a checksum over the entire backup stream and stamps it into the backup header
- **Using backup compression in SS2008 automatically switches on backup checksums**
- **Recommend you always use this option**
- **Possibly run a DBCC CHECKDB before taking a full database backup**
 - Make sure what you're backing up is not corrupt

How to Check Backup Integrity?

- **Use RESTORE ... WITH VERIFYONLY, CHECKSUM once the backup completes**
 - Does everything except actually restoring the database
 - Read all the pages in the backup and checks any page checksums
 - Re-calculates the backup stream checksum and checks against the one in the header
- **Alternatively, restore the backup on a separate server to help ensure it will restore during a disaster**
 - Maybe even run DBCC CHECKDB on it

Backup Integrity Survey

How often do you validate your backups?

Never		23%	17
Occasionally, but just the full backups		27%	20
Occasionally, full, diff, and log backups		12%	9
Regularly, but just the full backups		14%	10
Regularly, full, diff, and log backups		16%	12
Restore every time a full backup is taken		1%	1
Restore all backups (maybe with log shipping)		4%	3
Don't take backups		0%	0
What are backups?		1%	1
Total: 73 responses			

- <http://www.sqlskills.com/BLOGS/PAUL/post/Importance-of-validating-backups.aspx> (<http://bit.ly/xZHb9>)



63

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

What if the Database is Corrupt?

- A page checksum failure that's detected during a BACKUP ... WITH CHECKSUM will cause the backup to fail
- If taking the backup is imperative (e.g. the database is corrupt and you have no backup), use WITH CONTINUE_AFTER_ERROR
 - Forces the backup to complete no matter how corrupt the database is
 - Marks the backup as containing a corrupt database



64

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

What if the Backup is Corrupt?

- A page checksum failure (or backup stream failure) during a restore may cause the restore to fail
- If restoring the database is imperative (e.g. you've lost the database and this is the only backup), use `WITH CONTINUE_AFTER_ERROR`
 - Forces the backup to be restored no matter how corrupt the backup is
 - Recovery may not be able to complete
- Bottom line: in a disaster, you might be able to get some data back

Demo

Backup integrity

Making Backups Go Faster

- **Why?**
 - Reduce time that there's an I/O impact on the server
 - Reduce the likelihood that the log file will have to grow
- **Use I/O parallelism**
 - One reader or writer thread per physical volume
- **Use backup compression**
 - On the following slides
- **Use BUFFERCOUNT and MAXTRANSFERSIZE to optimize network traffic**
 - Be careful on 32-bit systems!
 - Amount of virtual address space is $\text{BUFFERCOUNT} * \text{MAXTRANSFERSIZE}$
 - See Jon's blog post links on resources slides at end
- **Backup to a local drive (or multiple drives in parallel) and then copy the file(s) over the network**

Backup Compression Syntax

- **Off by default on installation, simple syntax to enable**
 - Make it the default at the SERVER level:
 - EXEC sp_configure 'backup compression default', '1';
 - RECONFIGURE;
 - Explicitly during a BACKUP statement:
 - BACKUP DATABASE ... WITH COMPRESSION
 - Both of these can be done using SSMS
- **Log shipping log backups adhere to default setting**
- **RESTORE always does the right thing**
- **Compressing a backup always does backup checksums (but does not enable page checksums on the database)**

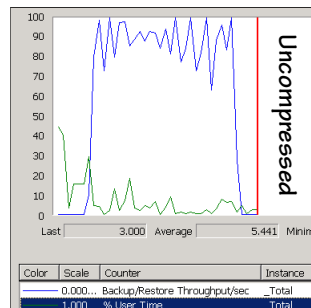
Backup Compression

Compression Ratio

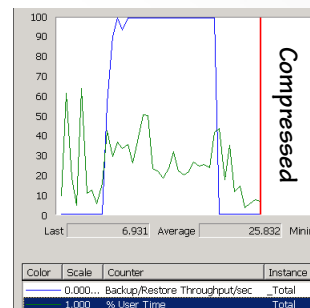
- Common questions:
 - 'How much compression will I see?'
 - 'Will it be comparable to, say, SQL Litespeed?'
- One simple answer: 'It depends!'
- All data compresses differently, so the compression ratio achieved depends on:
 - The type of data in the database
 - Whether the data in the database is already compressed
 - Whether the data/database is encrypted

Backup Performance

- System Monitor snapshot of backup of 322MB AdventureWorksDB



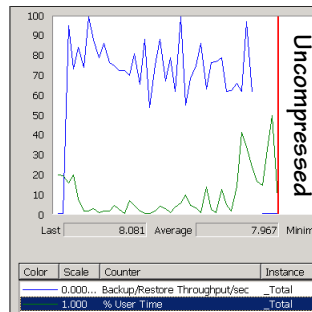
Hardly any CPU used (avg 5%), runtime = 39.5s, compression ratio of 0.



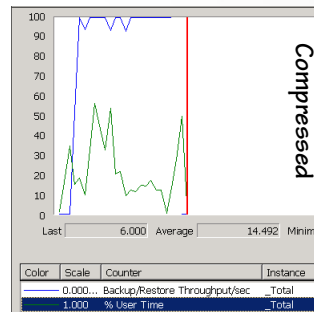
A LOT more CPU used (avg 25%) BUT runtime = 21.6s (45% improvement) and backup stored in 76.7MB (4.2x compression ratio)

Restore Performance

- System Monitor snapshot of restore of 322MB AdventureWorksDB



Hardly any CPU used (avg 8%), runtime = 71.0s



More CPU used (avg 14.5%) BUT runtime = 36s (almost 50% improvement)

Space Usage

- Backup compression will sometimes decide to pre-allocate space for the backup for performance
 - With piece-by-piece allocation, the writer threads can catch up with the allocation and be delayed
- This can lead to backup failures when the destination drive does not have enough space
- Trace flag 3042 prevents this behavior
 - Supported in SQL Server 2008 onwards
 - See KB 2001026

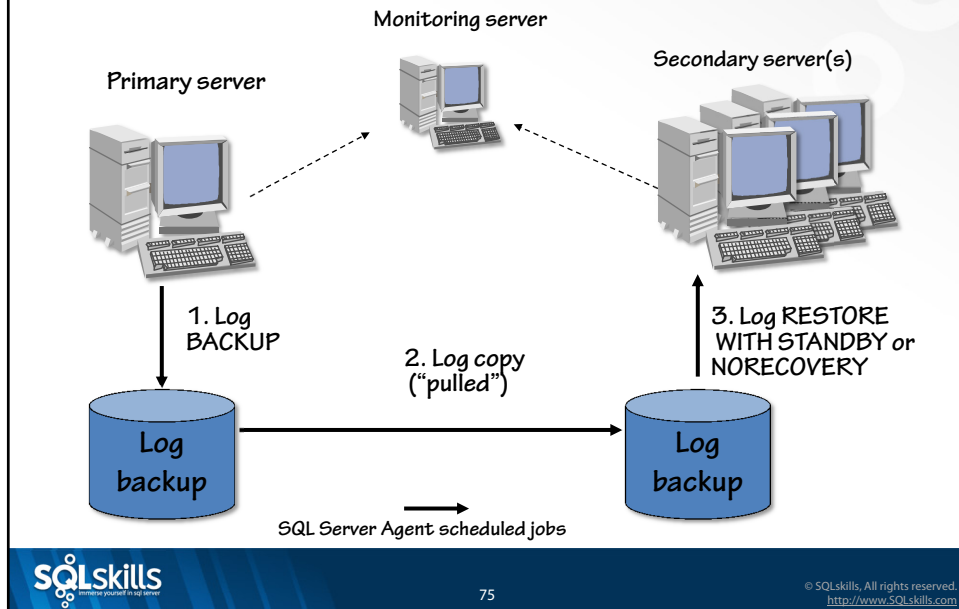
Restrictions

- **Compression available in Enterprise Edition only in 2008 but also in Standard Edition in 2008 R2 onwards**
 - All Editions can *de*compress a compressed backup
- **Backup sets need to be homogeneous**
 - Cannot mix compressed and uncompressed backups in a single backup media set
- **You cannot overwrite an existing backup media set with a different compression status backup without a WITH FORMAT**
- **All other existing functionality unaffected**
- **Best practice: make sure the backup compression ratio for a database is worth the CPU cost (and possibly extra time on busy systems) needed to compress the backup**
- **Use resource governor to limit amount of CPU being used**

Overview

- **Planning a backup strategy**
- **Backup types**
 - Full/Differential/Log
 - VLDB strategy concerns
 - Filegroup/file backups
 - Filegroup/file differential backups
- **Special features for backups**
 - COPY_ONLY
 - Parallel striped backup/multi-file backups
 - Mirrored backups
 - Backup integrity
 - Backup compression
- **Log shipping**

Principles of Log Shipping



Log Shipping

- **Automated backup/copy/restore of transaction logs**
- **Pros**
 - No hardware dependencies
 - Supports multiple secondary copies (for redundancy or reporting)
 - Supports a load delay to keep an earlier point in time version accessible (excellent for recovering from human error)
 - Allows access to secondary copies
 - Removes single point of failure when compared to failover clustering
- **Cons**
 - More potential for data loss (than synchronous database mirroring), data loss to the last log backup taken/copied/restored
 - Requires additional technologies to simplify client interaction

Key Log Shipping Parameters

- Delete transaction log files older than a certain time period
- Database load state
- Terminate user connections in database
- Transaction log backup schedule
- Copy frequency
- Load delay
- File retention period
- Backup alert threshold
- Out of sync alert threshold

Review

- Planning a backup strategy
- Backup types
 - Full/differential/log
 - VLDB strategy concerns
 - Filegroup/file backups
 - Filegroup/file differential backups
- Special features for backups
 - COPY_ONLY
 - Parallel striped backup/multi-file backups
 - Mirrored backups
 - Backup integrity
 - Backup compression
- Log shipping

Resources (1)

- **Backup Resources - Where, oh where, can they be?**
<http://www.sqlskills.com/BLOGS/KIMBERLY/post/Backup-Resources-Where-oh-where-can-they-be.aspx> (<http://bit.ly/9soJsV>)
 - Webcasts
 - Whitepapers
 - A Technical Case Study: Fast and Reliable Backup and Restore of Multi-Terabytes Database over the Network
 - Microsoft SQL Server 2008 Data and Backup
 - Compression Protecting SQL Server with System Center Data Protection Manager 2007
 - Articles
 - TechNet: Understanding Logging and Recovery in SQL Server, written by Paul
 - TechNet: Understanding SQL Server Backups, written by Paul
 - TechNet: SQL Server: Recovering from Disasters Using Backups, written by Paul
 - Paul's TechNet articles series from his blog category TechNet Magazine
 - SQL Server Magazine: Recovering from Isolated Corruption, written by me
 - SQL Server Magazine: The Best Place for Bulk_Logged, written by me
 - My SQLMag articles from the author's page on SQL Server Magazine
 - And a bunch more links....



79

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

Resources (2)

- **Articles:**
 - TechNet Magazine: Understanding SQL Server Backups
 - <http://technet.microsoft.com/en-us/magazine/dd822915.aspx>
 - SQL Server Magazine: Advanced Backup/Restore Options
 - <http://www.sqlmag.com/article/database-backup-and-recovery/advanced-backup-and-restore-options-129834>
 - SQLCAT: Tuning Performance of Backup Compression
 - <http://sqlcat.com/technicalnotes/archive/2008/04/21/tuning-the-performance-of-backup-compression-in-sql-server-2008.aspx>
 - SQL Server Central: Backup Monitoring and Reporting
 - <http://www.sqlservercentral.com/articles/Backup+%2f+Restore/66564/>



80

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

Resources (3)

- Jonathan's posts on tuning backup performance
 - <http://tinyurl.com/3qxay2r>
 - <http://tinyurl.com/3sberb7>
- Kimberly's backup/restore category
 - <http://www.sqlskills.com/BLOGS/KIMBERLY/category/Backup-Restore.aspx>
 - Understanding backups and log-related Trace Flags in SQL Server 2000/2005 and 2008
- Paul's backup/restore category
 - <http://www.sqlskills.com/BLOGS/PAUL/category/BackupRestore.aspx>
 - [http://www.sqlskills.com/BLOGS/PAUL/post/A-SQL-Server-DBA-myth-a-day-\(3030\)-backup-myths.aspx](http://www.sqlskills.com/BLOGS/PAUL/post/A-SQL-Server-DBA-myth-a-day-(3030)-backup-myths.aspx) (<http://bit.ly/cHupvz>)
 - <http://www.sqlskills.com/BLOGS/PAUL/post/Disaster-recovery-101-backing-up-the-tail-of-the-log.aspx> (<http://bit.ly/aDqvBf>)
- Amit Banerjee's post on VAS
 - How to find who is using/eating up the Virtual Address space on your SQL Server:
<http://troubleshootingsql.com/2010/02/16/how-to-find-who-is-using-eating-up-the-virtual-address-space-on-your-sql-server/> (<http://bit.ly/fxeETr>)



81

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

Questions?

