Operational Best Practices

SQL Server 2000: Recovery and Performance

Kimberly L. Tripp
President
SQLskills.com

SQL skills.com
immerse yourself in sql server

16 to 17 March 2005 | Interlaken, Switzerland
Microsoft TechDays 05

Speaker – Kimberly L. Tripp

- Independent Consultant/Trainer/Speaker/Writer
- Founder, *SY*Solutions, Inc. www.SQLskills.com

  email: Kimberly@SQLskills.com

  Become a subscriber on SQLskills.com and learn about new resources which can improve your productivity and server performance!
- SQL Server MVP http://mvp.support.microsoft.com/
- Microsoft Regional Director http://www.microsoftregionaldirectors.com/Public/
- Writer/Editor for SQL Magazine www.sqlmag.com
- Coauthor MSPress: *SQL Server 2000 High Availability*
- Presenter/Technical Manager for SQL Server 2000 High Availability Overview DVD

TechDays 05

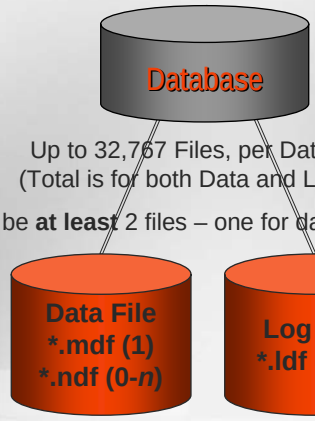
Overview

- Database Structures
- The Anatomy of a Data Modification
  - From Data Access to Locking to Logging
- Transaction Recovery
- Recovering a Damaged Database
  - Depends on backups available and strategies
- Minimizing Data Loss – Backup Strategies and Recovery Models
- Changing Recovery Models for Optimal Batch Processing

TechDays 05

Database Structure

Up to 32,767 Databases per Instance



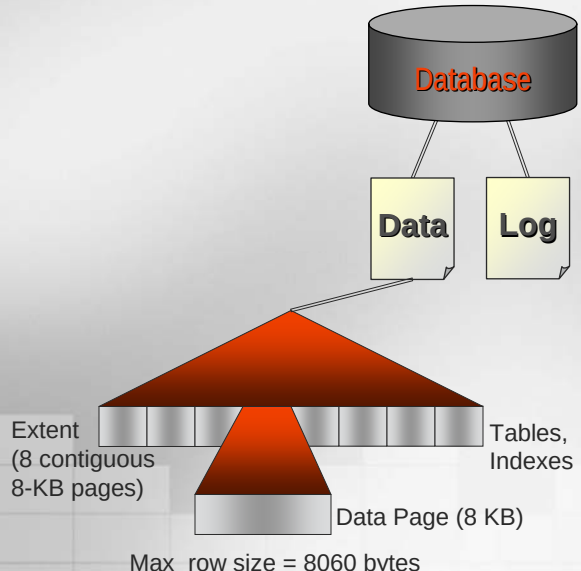
MDF = Primary (or **M**AIN) **D**ata **F**ile

NDF = **N**on-Primary **D**ata **F**ile

LDF = **L**og **D**ata **F**ile

Should be only 1 log file

How Data Is Stored



To get more information about the Transaction Log file see the BOL Topic: "Checkpoints and the Active Portion of the Log"

The Anatomy of a Data Modification

1) User sends UPDATE
   - Update is highly selective (only 5 rows)
   - Indexes exist to aid in finding these rows efficiently
   - The update is a SINGLE statement batch NOT enclosed in BEGIN TRAN…COMMIT TRAN block therefore this is IMPLICIT transaction

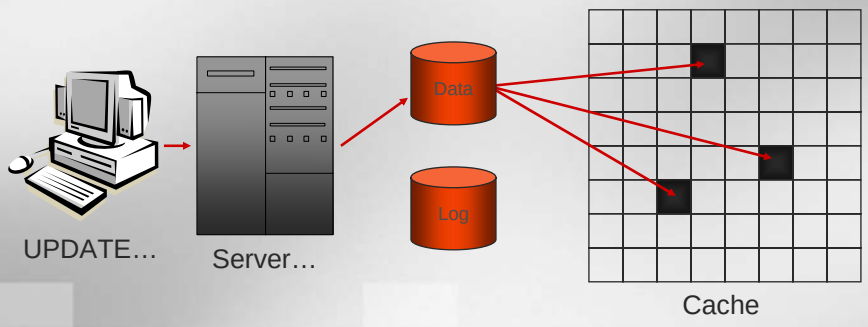
TechDays 05

The Anatomy of a Data Modification

2) Server receives the request and locates the data in cache OR reads the data from disk into cache
   - Since this is highly selective only the necessary pages are read into cache (maybe a few extra but that's not important here)
   - Let's use an example where the 5 rows being modified are located on 3 data pages

TechDays 05

What it looks like - Data



The Anatomy of a Data Modification

3) SQL Server proceeds to lock the necessary data
   - Locks are necessary to give us a consistent point FOR ALL rows from which to start
   - If any other transaction(s) have ANY of these rows locked we will wait until ALL locks have been acquired before we can proceed.
   - In the case of this update (because it's highly selective and because indexes exist to make this possible) SQL Server will use row level locking.

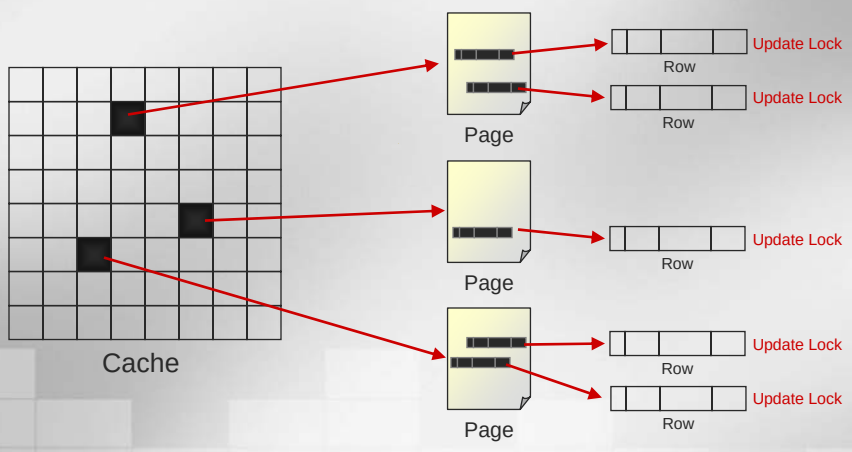
The Anatomy of a Data Modification (continued)

- The rows are locked but there are also "intent" locks at higher levels to make sure other larger locks (like page or table level locks) are not attempted (and fail)
  - 1 Database-level Shared Lock
  - 1 Table-level Intent eXclusive Lock
  - 3 Page-level Intent eXclusive Lock
  - 5 Row-level Update Locks
- There are a few locks that have already occurred – within indexes, etc. to read the data – but they are not significant here

This sounds complex but it's not too bad…

TechDays 05

What it looks like - Locks



TechDays 05

The Anatomy of a Data Modification

4) SQL Server can now begin to make the modifications – for EVERY row the process will include:
   1. Change the lock to a stricter lock (eXclusive lock)
      - An update lock helps to allow better concurrency by being compatible with other shared locks (readers). Readers can read the pre-modified data as it is transactionally consistent
      - The eXclusive lock is required to make the change because once modified no other reads should be able to see this un-committed change

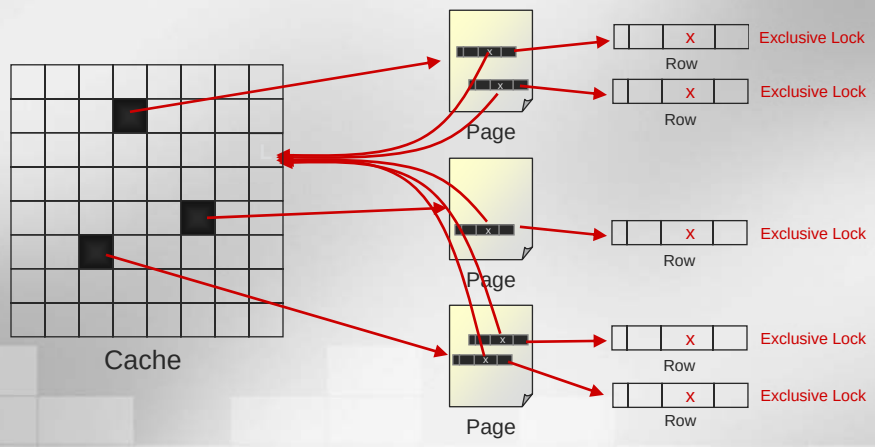
TechDays 05

The Anatomy of a Data Modification (continued)

   2. Make the modification to the data row
      - The modification occurs in cache
      - The data row has ONLY the after image information
   3. Log the modification to the transaction log pages
      - Also in cache
      - The log has an intermediate state which includes before and after information

TechDays 05

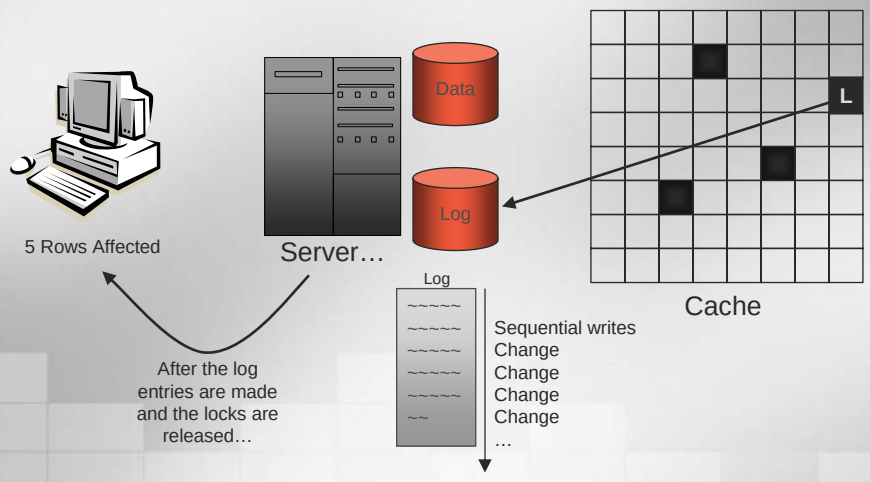
What it looks like - Modifications



The Anatomy of a Data Modification

5) Finally, the transaction is complete – this is the MOST critical step
   - All rows have been modified
   - There are no other statements in this transaction – i.e. Implicit transaction
   - Steps are:
     1. Write all log pages to transaction log ON DISK
     2. Release the locks
     3. Send a message to the user: (5 Rows Affected)

What it looks like
Write-Ahead Logging



TechDays 05

So now what?

- The transaction log ON DISK – is up to date
- The data in CACHE – is up to date
- But when does the data get written from cache to disk?

Checkpoint

TechDays 05

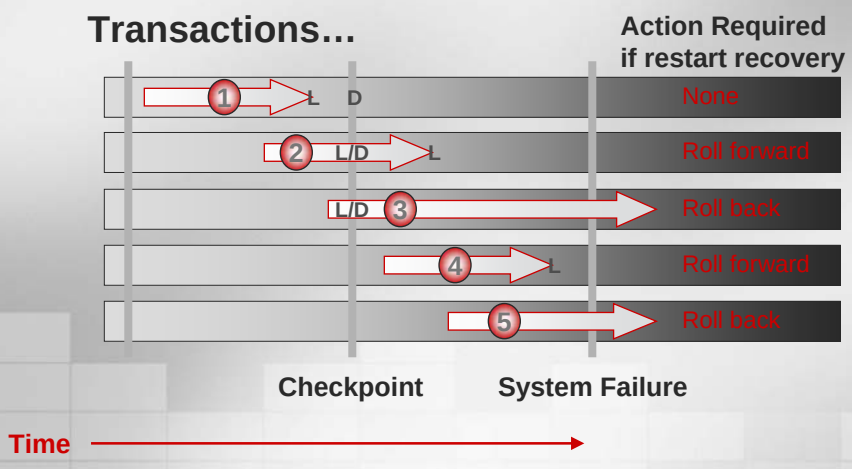
Checkpoint

*It's important to realize that a checkpoint does NOT just to write committed pages… Instead a checkpoint writes **ALL** pages which have changed since they were brought into cache – **regardless** of the state of the transaction which changed them!*

- Why?
  - To reduce roll-forward recovery time on startup
  - To batch I/Os to disk and reduce disk thrashing

TechDays 05

Transaction Recovery and Checkpoints



TechDays 05

Key Points

- Data Portion mostly random reads – except at checkpoint
- Log Portion mostly sequential writes
- Separate physical disks minimizes contention at the drive level – first choice in tuning
- Log is critical in recovery

TechDays 05

Key Points (continued)

- Log is written AHEAD of the data portion
- Log is the ONLY place where transactional consistency is known (i.e. guaranteed)
- Once a checkpoint occurs SQL Server doesn't need the information in the log – for committed (or inactive) transactions (the log could even be cleared however…)
- Without the transaction log a database cannot function (i.e. marked suspect)
- Then what?

TechDays 05

Bringing it all together
Recovering from Disaster

- A Database has two primary components
  - 

Data Portion
  - 

Log Portion
- Disaster varies by version
  - In SQL Server 2000: If either are damaged entire database is marked SUSPECT and is inaccessible
  - In SQL Server 2005: If the primary or log are damaged then the database is offline. If a secondary data file is damaged then only that filegroup is offline
- Damage to the data portion is not disastrous

TechDays 05

Bringing it all together
Recovering from Disaster

- Damage to the log IS devastating…
- Best case scenario first
  - If you can backup the transaction log
  - Accessing the "tail" of the log captures the changes since the last log backup
- Losing the log
  - Recovery from Backups – preferred (but how long ago and/or do they exist)
  - Recovering Data from damaged database – tricky, not recommended – yet, possible!

TechDays 05

Best Case Scenario

Up-to-the-minute 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
- THEN Up-to-the-minute Recovery is possible and with NO DATA LOSS!

When is a log backup possible?

1. Most critical, the LDF file must be accessible! The transaction log portion of the database cannot be damaged in any way. If any log file(s) is damaged then a backup of the transaction log will fail.
2. The Database Recovery model must NOT be set to SIMPLE
   - If the Database Recovery Model is Full – there are no other restrictions, backup the log!
   - If the Database Recovery Model is Bulk_logged then a log backup is ONLY possible IF no bulk operations have occurred since the last transaction log backup.

TechDays 05

How is a log backup possible?

- Through TSQL

  BACKUP LOG *dbname*
  TO *device*
  WITH NO_TRUNCATE

- Through Enterprise Manager
  - Select Database, right click Tasks, Select Backup Database
  - On Backup Database dialog, choose Transaction Log Backup and then on the Options dialog, DESELECT the option:
    - ☐ Remove inactive entries from transaction log

TechDays 05

If the Log is not accessible

- You must recover from your database backups; the data portion is not (and should not be) accessible…
- How do you recover – from your backups…
- When was your last backup?

TechDays 05

demo

Accessing Orphaned Data

Without the log you, data integrity is NOT guaranteed

16 to 17 March 2005 | Interlaken, Switzerland

Microsoft TechDays 05

Orphaned Data portion not accessible for good reason!

Why can't you access the data portion?

- The data is NOT guaranteed to be accurate!
- What if a transaction – which spanned a checkpoint – was pending at the time of the failure (remember transaction #3)… You would have inconsistent data in your database (at the time of checkpoint they wrote BOTH the log and data for transaction #3). During recovery the log would have made it consistent – but you don't have the log…

TechDays 05

Recovering the Database

- If you've got a FULL Database backup…at least you have that. It's a good starting point for recovery!
- If you've also got a series of log backups…even better – you can get even closer to the point in time where the drives/controller failed and possibly up-to-the-minute
- In the worst case scenario you have nothing. You have no backups and only an inaccessible and suspect database…do you have any choice?

TechDays 05

Emergency Mode

- As per the demo, you CAN access an orphaned data portion
- Also, as per the demo, this data is NOT guaranteed for accuracy
- But, what if it's your last choice and your only choice…
  - If the database is suspect AND the log is NOT available AND you have NO OTHER OPTIONS for recovery…
  - Use emergency mode by setting the database status to 32768. This will allow you to access the "orphaned" data portion of the database in order to copy/verify the data and save to another database.

TechDays 05

Minimize Data Loss

- **Have a backup strategy!**
- Full Database Backups (only)
  - Highest amount of possible data loss
- Full Database Backups with Transaction Log Backups
  - Can provide up-to-the-minute recovery
  - Timely to recover a large number of logs
- Full Database Backups with Transaction Log Backups AND Differential Database Backups
- **Test your backup strategy!**

TechDays 05

Frequencies of Backups

- Full Database Backups (only)
  - Full database backups weekly or nightly
- Full Database Backups with Transaction Log Backups
  - Full database backups weekly or nightly
  - Transaction Log backups hourly, semi-hourly, every 15 minutes or even more frequently
- Full Database Backups with Transaction Log Backups AND Differential Database Backups
  - Full database backups weekly
  - Differential database backups nightly
  - Transaction Log backups hourly, semi-hourly, every 15 minutes or even more frequently

TechDays 05

Summary: Importance of the Log

- Only place where change information is written – ahead of the data (write-ahead logging)
- Only way to get up-to-the-minute recovery
- May grow quite large depending on type of activity and maintenance of log…
- How do you manage the log?

TechDays 05

Transaction Log Management

- How do you keep the log small?
- What "clears" the log? How?
  - Regular Log Backups – SQL Server clears the inactive portion of the log after it's backed up
  - NOTE: Database backups DO NOT EVER clear the log (disregard the UI BUG which shows this option even for a full database backup – on the options tab)
- What if the log fills?

  ⇨ Perform a NORMAL Log Backup!
  NO need for NO_LOG option on log backup

TechDays 05

What can impact your ability to recover? Recovery Models

- Full → Everything is Fully Logged
  - All operations allowed and FULLY logged
  - Operations like creating or rebuilding an INDEX may take more log space and more time than in previous versions even SQL Server 7.0
- Bulk_Logged → Minimal Logging for SOME Operations (not NON-LOGGED)
  - Operations whose performance is affected. This is a COMPLETE list:
    - CREATE/REBUILDING Indexes
    - Bulk Load Operations – BCP, DTS, BULK INSERT
    - SELECT INTO a permanent table
    - LOB Manipulation with WRITETEXT and UPDATETEXT
  - ALL other operations (i.e. updates, inserts, etc. take the same log space and time as the FULL recovery model
- Simple → Log Truncation on Checkpoint

TechDays 05

Recovery Models & Log Backups

- Full Recovery Model
  - All changes are fully logged therefore log backups will support:
    - up-to-the-minute recovery by backing up the “tail” of the
    - point-in-time recovery with the option STOPAP on restore (marked transactions or time-based stopat)
- Bulk_logged Recovery Model
  - Bulk Operations are “minimally” logged
    - faster bulk operations
    - NO point-in-time
    - NO up-to-the-minute
  - A log backup is possible but ONLY when the data portion is available at the time of the backup
- Simple Recovery Model does not support log backups

TechDays05

Minimize impact of logging

- Cannot clear the log unless the transactional information is inactive – if a transaction is pending then it stays in the log until it’s completed…
- Restart recovery will take longer if there were a lot of long running transactions pending at the time of failure
- Always try to avoid:
  - Long running transactions
  - Transactions which span more than one batch
  - Transactions that require user interaction
  - Nested Transactions (complex and usually longer running)

TechDays05

Logging, Recovery Models, Recovery and Performance

- Acceptable to Switch?
- How, Why and Why Not?
- Switching Recovery Models
- Batch Processing Scenario
- Batch Processing Script

TechDays 05

Acceptable to Switch?

- Yes
  - If users are not allowed in database during batch processing
  - No modifications are made during batch process that are not recoverable through some other means (i.e. re-running the batch process)
  - Data loss during the batch process is OK
- No
  - Data loss is NOT acceptable
  - Modifications are made by users/processes that are not recoverable
  - OLTP System requiring 24x7x365

TechDays 05

Switching between Full/Bulk_Logged

```
ALTER DATABASE <name>
SET RECOVERY FULL
WITH <termination option>
```

Why?

- Minimally Logged Operations:
  - Take less time – i.e. better performance
  - Require Less "active" log space – the backup will NOT be smaller than if the database had been in Full Recovery Model
- Log shipping and automated log backups remain intact

TechDays 05

Full and Bulk_Logged

- Why Not?
  - If the database were to become corrupt WHILE the database is in a bulk logged state, you lose up-to-the-minute recovery *i.e. the tail of the log cannot be accessed*
  - You're only as good as your last LOG backup – when was that?
- You need to determine if this is acceptable?

TechDays 05

Switching Recovery Models

From → / To ↓	Full	Bulk_Logged	Simple
Full	*n/a*	No Change Required, Transaction Log Backup Recommended AFTER	Full Database or Differential backup after change
Bulk_Logged	No Change Required, Transaction Log Backup Recommended BEFORE	*n/a*	Full Database or Differential backup after change
Simple	Backup Log just prior to change	Backup Log just prior to change	*n/a*

- Transitions to/from SIMPLE are NEVER recommended as they break the continuity of the log chain. This eliminates your ability to go back to a prior full…meaning you're only as good as your last full backup!

TechDays 05

Switching Recovery Models

- Full ↔ BULK_LOGGED doesn't break automated backup procedures or log shipping. However, a log backup immediately before (when changing to Bulk Logged) and a log backup immediately after (when changing back to full) are recommended.
- Time running in "bulk_logged" should be minimized to reduce potential workloss exposure
- Backup before change to make sure you have a transaction log which has ALL options available in full recovery model
- Backup after changing back to full to make sure you have a backup which includes all minimally logged operations as well as make sure that the option has been reset!

TechDays 05

Switching Recovery Models

Advantages:

- Up-to-the-minute recovery
- Point in time recovery
- Access to "tail" of the log

Advantages:

- Minimal logging
- Faster Bulk Operation

Disadvantages:

- No *STOPAT*
- Data loss possible if media failure…
- If database becomes suspect, cannot back up tail of log.

Switching Recovery Models

Using ALTER DATABASE

```
-- Check the status before changing it!
IF DATABASEPROPERTYEX('dbname', 'Updateability') =
'READ_ONLY'
      ALTER DATABASE dbname
            SET RESTRICTED_USER,
            READ_WRITE WITH ROLLBACK AFTER 60
go
-- Perform Batch Operations…
go
ALTER DATABASE dbname
      SET READ_ONLY, MULTI_USER
      WITH ROLLBACK AFTER 30
go
```

Make sure to review the sample script with tons of comments!

Review

- Database Structures
- The Anatomy of a Data Modification
  - From Data Access to Locking to Logging
- Transaction Recovery
- Recovering a Damaged Database
  - Depends on backups available and strategies
- Minimizing Data Loss – Backup Strategies and Recovery Models
- Changing Recovery Models for Optimal Batch Processing

Resources

- Check out www.SQLskills.com for information about upcoming **SQL Immersion** events, useful links and event scripts. All of the scripts used in this presentation will be available on the Past Events page within one week.
- Read my blog: http://www.SQLskills.com/Blogs/Kimberly/
- Subscribe to SQLskills: http://www.sqlskills.com/login.aspx
- MSPress: *SQL Server 2000 High Availability*
  Authors: Allan Hirt with Cathan Cook, Kimberly L. Tripp and Frank McBath
  ISBN: 0-7356-1920-4
  On the SQLskills.com homepage you can download a sample chapter!

SQL skills

Please fill out your evaluation
Thank you!

Kimberly L. Tripp
Consultant . Trainer . Writer . Speaker

email: Kimberly@SQLskills.com
Make sure to register for special offers and other helpful information and resources!
www.SQLskills.com