SQL Server: Detecting and Correcting Database Corruption

Module 7: Simple Restore Techniques

Paul S. Randal
http://www.SQLskills.com/blogs/paul/
Paul@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **Restoring from backups is the best way to avoid data loss, as long as the right backups are available**
- **Minimizing the time required to restore is important, so knowing the restore options and how to use them is essential**

- **In this module we'll cover:**
  - Restore options
  - Point-in-time restore
  - Tail-of-the-log backups
  - Restore sequence

Recovering Using Backups

- **Although this is the best way to avoid data loss, it's not necessarily the best way to avoid downtime**
  - Depends what kind of backups are available
  - For example, with a backup strategy of full backup every Sunday and half-hourly transaction log backups:
    - If recovery is necessary towards the end of the week, lots of log backups need to be restored, which will take a long time
    - Better to have daily full or differential backups to allow faster restore
- **Remember:**
  - Backups have to exist to be useful
  - Backups have to be valid to avoid data loss
- **How often do you validate your backups?**
  - Perform a full test restore on another server at least once a week

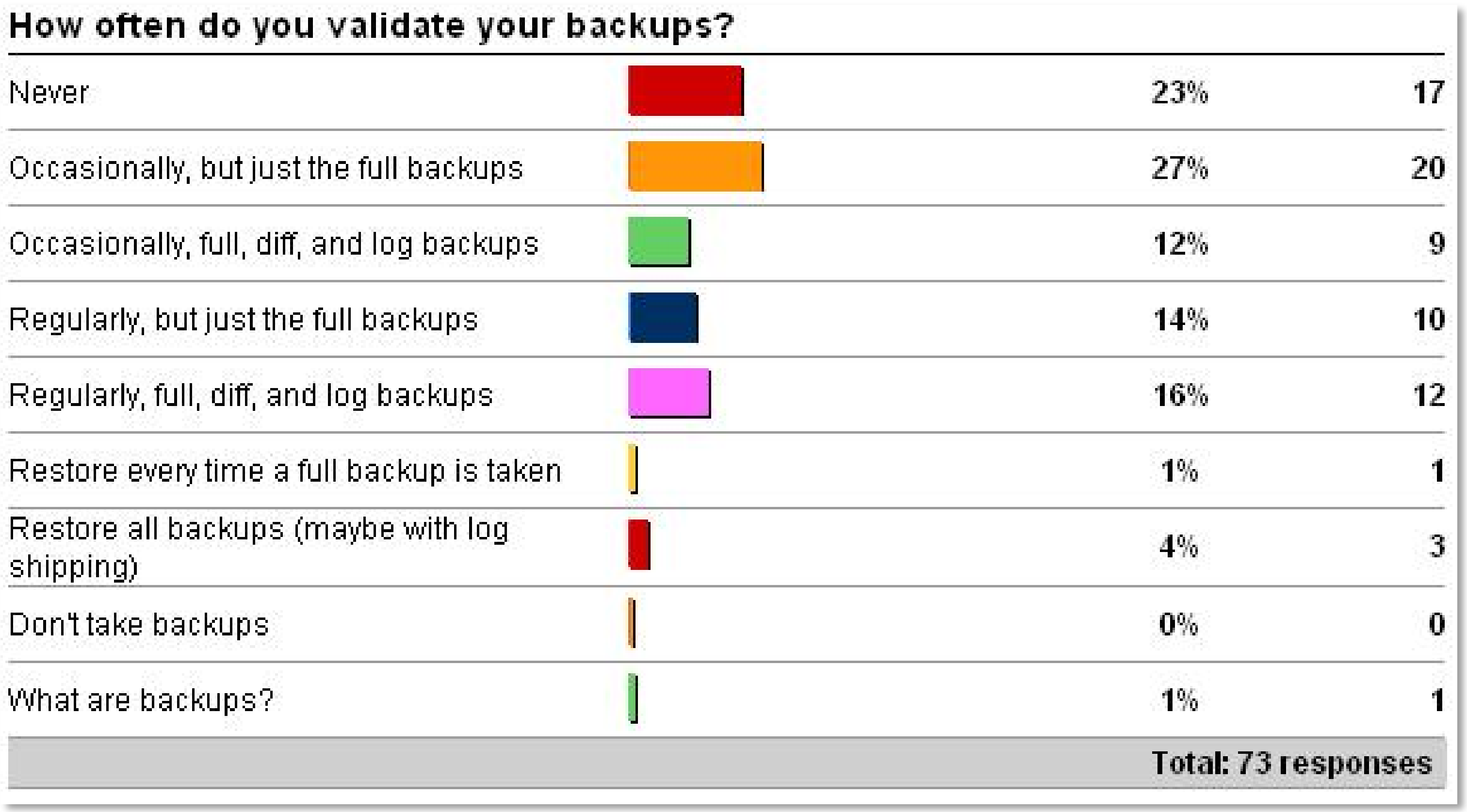
Backup Types (1)

- **See my blog post at http://bit.ly/15SCN5e for more details on backup types and recovery models**
- **Full database backup**
  - The starting point for any restore sequence is usually the most recent full database backup
  - This provides a single point in time to restore to, or the basis for further restore operations
- **Transaction log backup**
  - Commonly known as just a “log backup”
  - The set of transaction log records generated since the last log backup completed, using the full or bulk_logged recovery models
  - Must be used to roll forward a database to the desired point in time

Backup Types (2)

- **Differential database backup**
  - The set of data file pages that have changed since the most recent full database backup preceding this backup
  - A differential backup is the net effect of all transaction log backups between the most recent full backup and this differential backup
  - Where possible, the restore sequence will start with restoring a full backup and then restore the most recent differential backup, before finishing with transaction log backups
  - This makes the restore sequence shorter and faster
- **There are also full file, full filegroup, differential file , and differential filegroup backups**
  - Their use is more advanced and we will not consider them here
  - If they exist in your environment, it is imperative that you practice restore sequences with them

Backup Validation 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

- Source: my blog post at http://bit.ly/xZHb9

How to Restore?

- **In an emergency, always go to the operations guide/DR handbook/run book (do you have one?)**
- **Operations guide should define:**
  - Location of backups
  - How to put together the restore sequence, depending on what is damaged
  - Commands or scripts to use
  - Amount of time expected per step during restore process
- **There may be a 3rd-party backup/restore tool that you must use**
  - We're just talking about SQL Server native backups in this module
- **Make sure to use the correct RESTORE options**
  - We'll discuss these later

Continuity of the Log Backup Chain

- **The log backup chain is an unbroken sequence of log backups covering the time between the most recent full backup and the point in time to which a restore is required (typically the current time)**
- **If the log backup chain is not complete (meaning one or more log backups are damaged or missing) then complete recovery is not possible unless the “gap” can be bridged with a differential backup**
- **Consider creating copies of all backups, with some stored off-site**
- **What breaks the log chain?**
  - Clearing the transaction log manually
    - Using the WITH TRUNCATE_ONLY or WITH NO_LOG options on BACKUP LOG prior to SQL Server 2008
  - Changing the database to the simple recovery model
  - Reverting from a database snapshot or rebuilding the transaction log

Restore Options

- **WITH RECOVERY is the default**
  - Does not allow further restore options as database is fully recovered
  - Frustrating that it is the default, as it causes problems
- **Use WITH NORECOVERY for safety and make it your default for all restore operations in the restore sequence**
  - Postpones fully recovering the database
  - Then use RESTORE DATABASE dbname WITH RECOVERY to complete the restore sequence
  - If you make a mistake, you must restart the restore sequence
- **WITH STANDBY is the same as WITH NORECOVERY but creates a recovered, read-only database for you to look in**
  - Creates an UNDO file to save info from transactions that are pending at the end of the restore

Restoring an Entire Database to a Point in Time

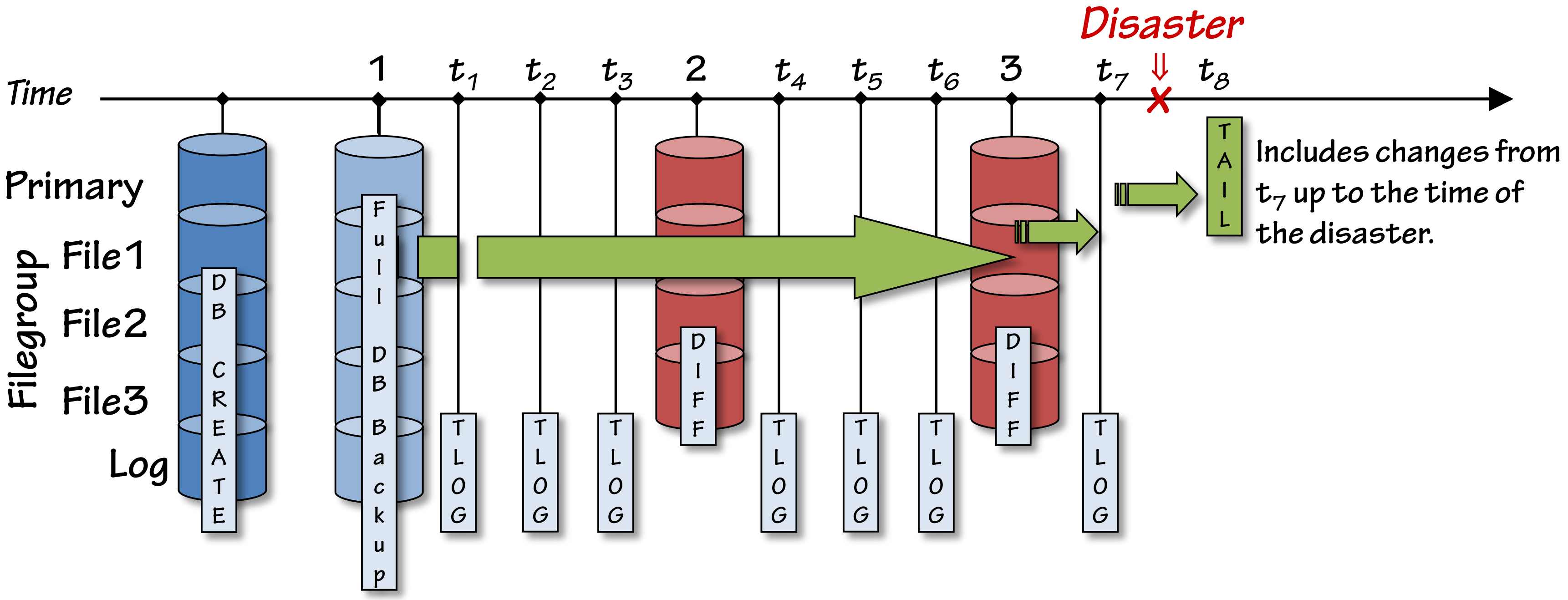
- **Common examples:**
  - Restore a database to the time just before a user or application error occurred (e.g. dropping a table)
  - Restore a database to the point before a known corruption-causing event occurred
- **Start with most recent full backup and proceed with smallest set of backups to the desired point**
- **Best practice is to use WITH STOPAT on all operations in the restore sequence**
  - Date and time specific, or
  - Transaction name (i.e. marked transaction)
- **Note: All work after the stop point is lost**

Tail-Of-The-Log Backups

- **After a disaster, where it is necessary to restore from backups, the final portion of the transaction log must first be backed up to allow restoration with zero data loss**
- **This backup is called a tail-of-the-log (or tail-log) backup**
- **I recommend performing a tail-of-the-log backup manually rather than through SSMS**
- **If the data files are inaccessible you can only perform a tail-of-the-log backup using the WITH NO_TRUNCATE option**
- **If there was a minimally-logged operation since the last log backup, and the data files are inaccessible:**
  - Until SQL Server 2008 R2, the tail-of-the-log backup will fail
  - From SQL Server 2008 R2 onwards, the tail-of-the-log backup will succeed, but restoring it will result in a corrupt database
    - As the log backup will contain the minimally-logged operation, but not the data extents that were changed as a result of it

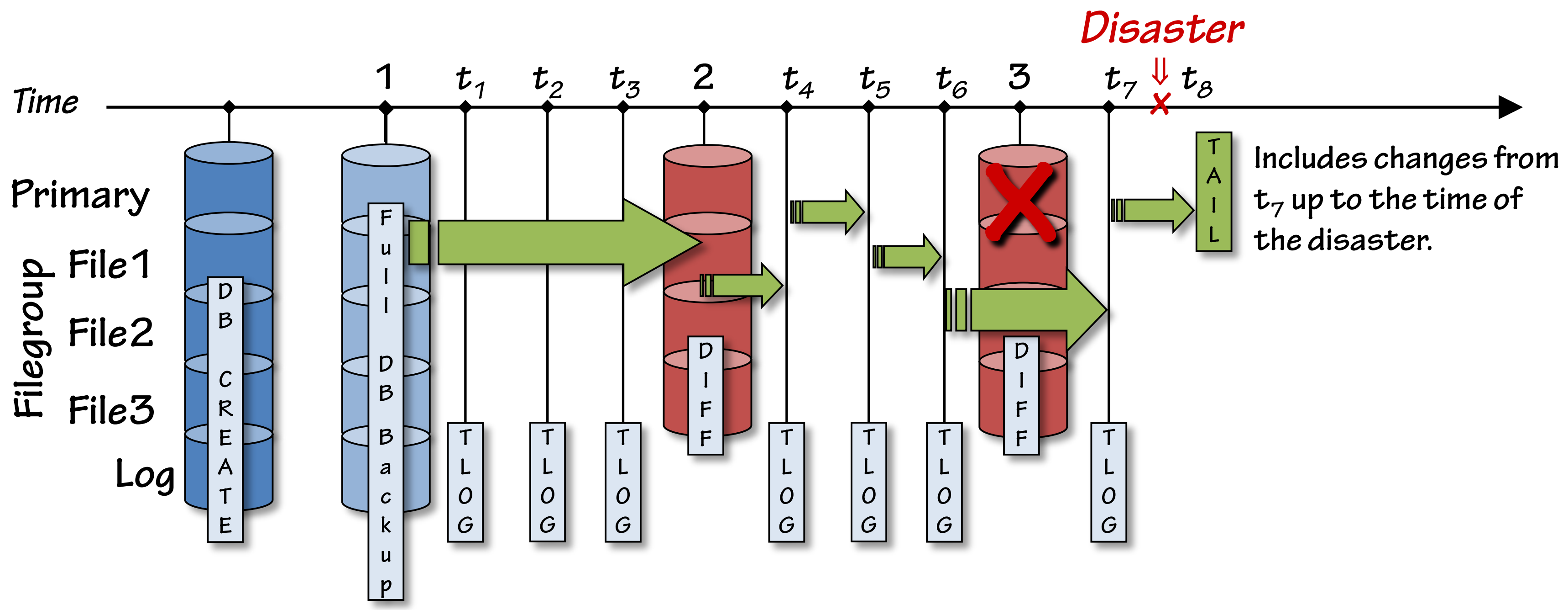
Recovery With Full, Differential, and Log Backups

- **Backup 1 is full, backups 2 and 3 are differential**
- **In the event of failure at time x:**
  - Backup tail-of-the-log
  - Restore full (1), last differential (3) and all log backups up to the point of the failure



Falling Back on Log Backups

- **What if differential backup 3 was bad?**
  - Restore full (1)
  - Restore latest working differential (2)
  - Restore all log backups up to the point of the failure (t4 through t7) and finally the tail-of-the-log (t8)



Determining the Restore Sequence

- **In a complex restore sequence, there may be differential and log backups that cover the same time interval**
- **No point restoring unneeded log backups, but also must make sure that the log backup covers the right time period (i.e. set of LSNs in the log)**
- **Finding the minimum required LSN or min-LSN can be done using:**
  - Backup history tables in msdb
    - Make sure you're backing up msdb (and other system databases too)
  - Output from RESTORE HEADERONLY
  - Best to have scripts written and tested in advance of having to use them
- **Restore sequence can also be found using the SSMS Database Recovery Advisor**
- **More information in Books Online at http://bit.ly/19RnAqn**

Restoring to Non-Enterprise Edition

- **SQL Server 2005 will not allow a database that contains table/index partitioning to restore on any Edition except Enterprise and Developer**
- **From SQL Server 2008 onward, this list has expanded to also include change data capture, data compression, transparent data encryption**
- **Enabling any of these features requires elevated privileges except data compression which only requires the ALTER TABLE permission**
  - This means a table owner can make the database only restorable on Enterprise Edition without your knowledge
- **This can be a serious setback in a disaster recovery situation where the database is restoring to Standard Edition, as the database must be restored BEFORE the server can tell whether any of these features are present and fail the restore operation**
- **Use the DMV sys.dm_db_persisted_sku_features to see whether any of your database are "Enterprise-only"**

Summary

- **Always try to restore as few backups as possible, to save time**
- **Make sure you have practiced determining and running restore sequences before a disaster occurs**
- **Make sure you know what the various restore options are and how to use them**

- **In the next module, we'll discuss:**
  - Simple techniques to recover using DBCC CHECKDB repair options