

Module 2

Designing for Performance

File Placement for Reliability and Performance

➤ Kimberly L. Tripp
➤ SQLskills.com

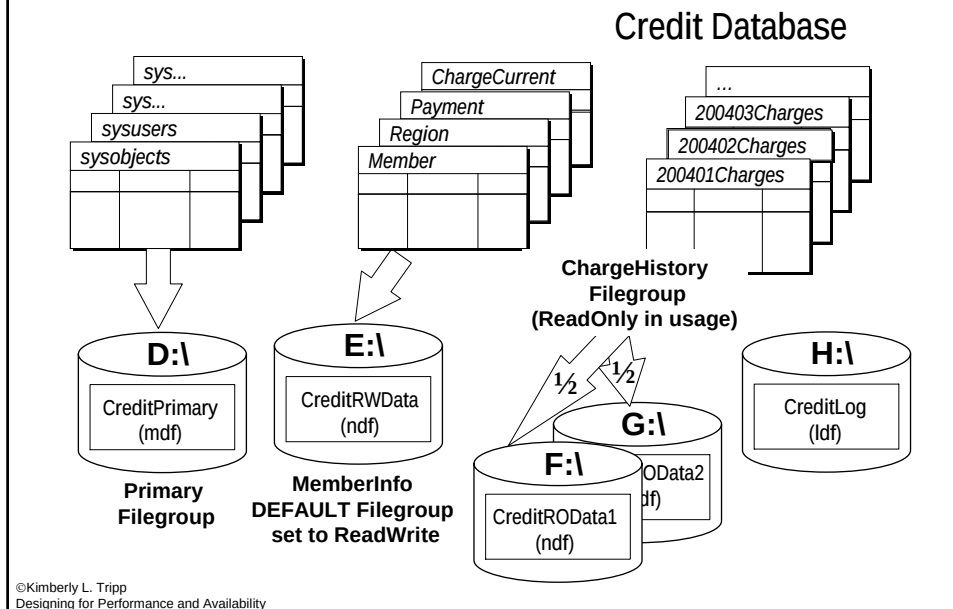


Overview

- Introduction to Filegroups
- Using Filegroups for Maintenance
- Using Filegroups for Availability
- Using Filegroups for Performance
 - Guidelines for Creating Filegroups
 - Distributing Data Across Disks by Using Filegroups
 - Combining Filegroups with Hardware RAID
 - Functional Partitioning Basics
- Common Questions/Best Practices

©Kimberly L. Tripp
Designing for Performance and Availability

Introduction to Filegroups



Benefits of Filegroups

Maintenance

- **Back Up or Restore Files or Filegroups Rather Than an Entire Database – Faster Restore when Isolated Failure**
- **Group Tables and Indexes with Similar Maintenance Requirements into Same Filegroups – Consider partitioning due to concurrency/maintenance requirements**
- **Assign an Individual High-Maintenance Table to Its Own Filegroup**

Benefits of Filegroups

Availability

- Separate data into secondary data files
- Files per logical drive – multiple files, multiple physical disks?
- Think of what is likely to fail – file(s) or disk
- Key types of data:
 - Single large table
 - Read write, critical v. non-critical
 - Read-only, critical v. non-critical (varying degrees)
 - Text/image and LOB Data (textimage_on)
includes: text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max)

©Kimberly L. Tripp
Designing for Performance and Availability

Creating Filegroups

Recommendations

- Do not consider Filegroups under 10GB
- One File per Logical Drive (regardless of physical drives)
- Isolate/pre-allocate Primary Filegroup
- Make User-Defined Filegroup the Default for Large Systems
- Determine Number of Filegroups by Maintenance Needs
- Spread Data Evenly Across Multiple Disks

©Kimberly L. Tripp
Designing for Performance and Availability

Files per Filegroup

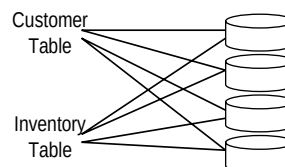
- **Simplicity in VLDB**
 - Small number of Filegroups
 - Large amounts of similar data
 - Create Multiple Files per Filegroup with Limited Filegroups for simplicity
- **Varying Data Access Patterns**
 - Multiple Filegroups
 - Create Multiple Files in a Filegroup when capacity warrants it – otherwise, create a Single File per Filegroup
 - Filegroups hold similar data – functionally partitioned

©Kimberly L. Tripp
Designing for Performance and Availability

Proportional Fill Performance & Simplicity for some VLDB

- **Evenly balancing Data Load Across Multiple Disks**

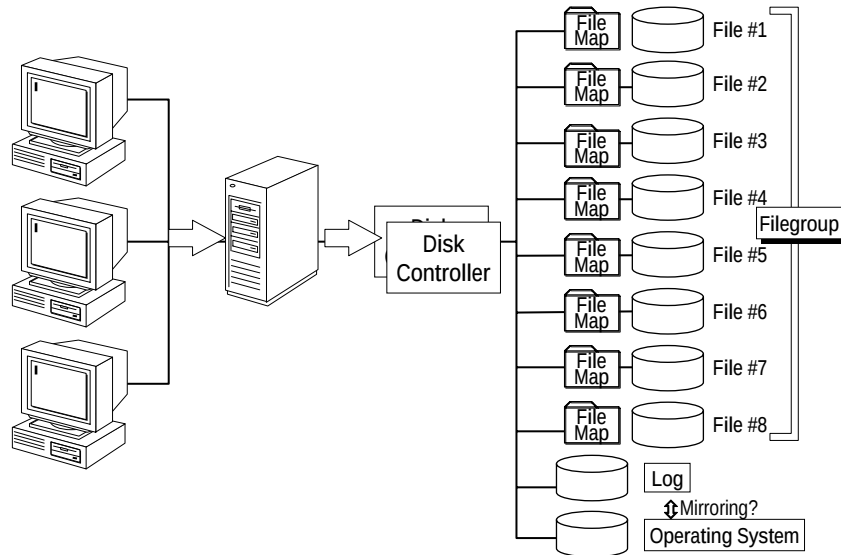
Four Logical Disks
1 File per Logical Disk
One Filegroup



- **No design changes**
- **Ideal when varying access patterns do NOT exist**

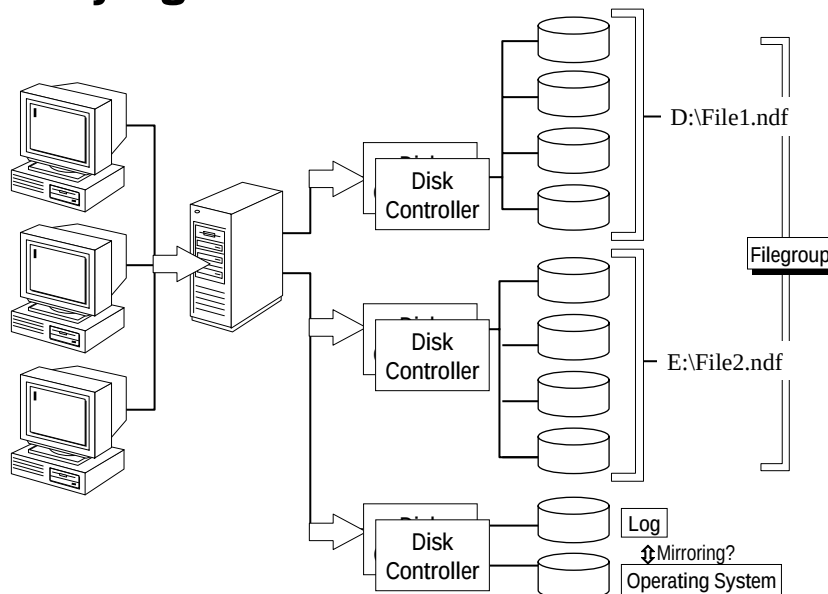
©Kimberly L. Tripp
Designing for Performance and Availability

Simplicity in VLDB



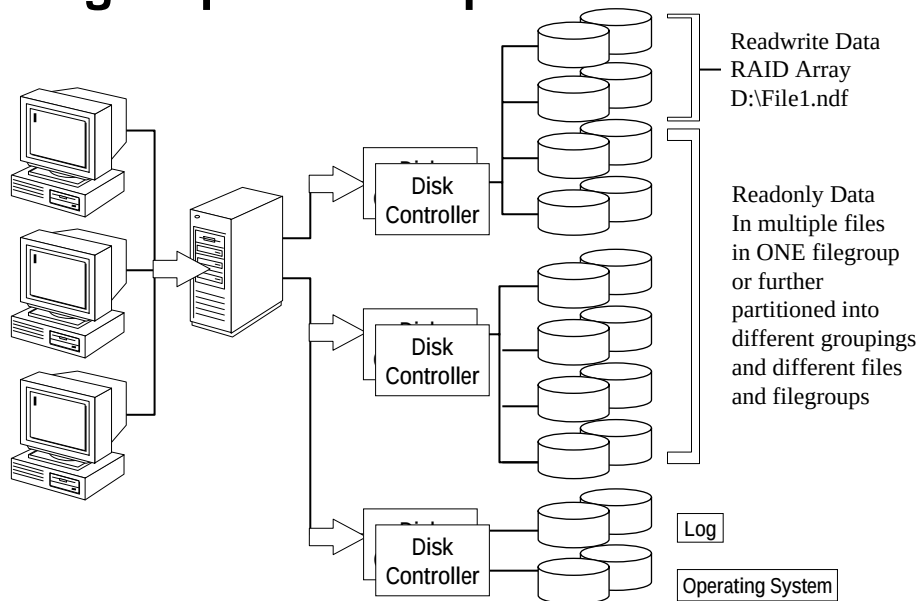
©Kimberly L. Tripp
Designing for Performance and Availability

Varying Data Access Patterns



©Kimberly L. Tripp
Designing for Performance and Availability

Filegroups with Separate Disks



©Kimberly L. Tripp
Designing for Performance and Availability

Disk Configuration

Better Availability/Best Performance

- **Disk Configuration for Better Availability**
 - **Partitioning**
 - Functional partitioning
 - Regional
 - Logical partitioning (date/time based data)
 - **Simplified approach – do not split up data thinking multiple files equals better performance (keep it simple)**
 - **Some extreme cases where having multiple files eliminates some internal bottlenecks on object allocation**
 - Files equal .25 to .50 the number of CPUs on multiproc machines greater than 8 CPUs (8-way s/b 2 to 4 files)
 - Multiproc machines with a lot of object creation (TempDB)
Review KB 328551

©Kimberly L. Tripp
Designing for Performance and Availability

Filegroups lead to better Availability

- Fine grain operations are based on “partitioning” datasets for VLDB
- Partitioning in this sense does NOT require Partitioned Tables however, this feature significantly benefits from these capabilities
- Partitioning for fine grain operations just means strategically placing objects within filegroups to ensure correct combination at time of disaster
- Strategies...

©Kimberly L. Tripp
Designing for Performance and Availability

Date/time-based

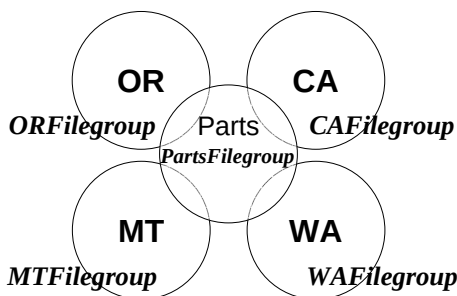
Time-based data placement

- Structures designed for sliding window scenario
- Tables created and data flows on regular/consistent basis – weekly, monthly, yearly, etc.
- Data may be archived/removed to keep only “current” timeframe – year, two years, etc.
- Uses SQL Server 2000 Partitioned Views or SQL Server 2005 Partitioned Tables defining “ranges” using date-based criteria

©Kimberly L. Tripp
Designing for Performance and Availability

Related-Object Groupings

- **Related-object groups = List-based or functional**
 - Regionally based with some shared components
 - Functionally based – could use separate tables OR Partitioned Tables using a list-based partition function



All Region-specific Data: Customers, Sales, ServiceRequests, etc. 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.

©Kimberly L. Tripp
Designing for Performance and Availability

Foundation – Database Structures

Partitioning Basics

- **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 (unless in Simple recovery model = truncate log on checkpoint)
 - Never break the continuity of the transaction log – the transaction should be cleared ONLY when a transaction log backup is performed (avoid TRUNCATE_ONLY/NO_LOG and/or use Trace Flag -T3231 (no_op), -T3031 (checkpoint))

©Kimberly L. Tripp
Designing for Performance and Availability

Foundation – Database Structures

Partitioning Basics (continued)

- **Additional non-Primary Data Files**
 - Each file **ALWAYS** exists in **ONE** filegroup
 - A file can **ONLY** be a member of one filegroup
 - Once added to the database, the filegroup **CANNOT** be changed
 - Contain user-defined data (tables/indexes) strategically created/placed on one or more filegroups
 - Contain complete objects (tables/indexes) – when the object has not been partitioned
 - Contain a partition of a partitioned object (an object **CANNOT** exist in multiple files in multiple filegroups unless partitioned)

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions

- **What is THE “right” way to configure your SAN for SQL Server?**
 - The I/O pattern of SQL Server is distinct and well-known; but very different from a file server, which is what most vendors optimize for!
 - Understanding I/O patterns and I/O volumes are key to success
 - Size matters, deploying a single large or hundreds or more small database requires a different strategy

©Kimberly L. Tripp
Designing for Performance and Availability

Thanks to Gert Drapers for a HUGE amount of tips and tricks and resources over the years (and still going!)
<http://www.SQLDev.NET>

Common Questions

- **General guidelines**
 - **Faster (RPM) disks are always better**
 - **More+smaller spindles are always better then less+ bigger spindles**
 - **Engaging the right engineers from the vendors early in the design is key**
 - **Don't micro manage disk placement; use the economy of scale of the SAN to perform optimal striping, therefore distributing you data over volumes (KISS) but do consider files for partitioning/management**
 - **Benchmark all storage configurations prior to deployment**

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

Five.

- **Which RAID level should I be using?**
- **Best Practice:**
 - **Log files on RAID 1+0 disks**
 - **Isolate log files from data files at the physical disk level**
- **Depending on tempdb usage, performance may increase when placed on RAID 1+0**
 - **See "FIX: Concurrency enhancements for the tempdb database"**
 - **<http://support.microsoft.com/default.aspx?scid=kb;en-us;328551>**

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

Five.

- Test results indicate performance increase on RAID 1+0 for write intensive workloads but at a higher cost (\$)
- The performance difference between RAID 1+0 and RAID 5 can vary by vendor
- Benchmarking of the storage will give a clear indication of the performance differences between RAID levels before SQL Server is deployed

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

- How many & what size should my LUN's be?
 - LUN Size: Recommendation of the storage vendor balanced with the needs of the particular deployment
 - How many LUN's: There are some parallelism gains with SQL Server when multiple logical volumes (not necessarily LUNs) are used
 - Backup and restore – 1 reader, 1 writer thread per logical volume
 - Create Database – One thread per data file for initialization
 - May only be a consideration for very large databases
 - SQL Server 2005 alleviates this concern (data files created instantly, only log file requires initialization)
 - DBCC – Round robin I/O with equal amount issued to each logical volume
- Consider the number of processors on the host
- Use of logical volumes can offer better granularity of monitoring

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

- **How many data files and/or filegroups should I have?**
 - **More data files does not necessarily equal better performance – this is determined by hardware capacity & characteristics of SQL Server's I/O**
 - **Disaster recovery requirements**
 - **Will the target environment for a disaster recovery restore accommodate the file sizes?**

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

- **In a few, extreme cases (high performance insert scenarios or high object creation scenarios), the number of data files may impact scalability**
 - **Consider the number of processors on the host server (mainly be concerned for ≥ 8 processors)**
 - **Best Practice: have .25 to .5 data files (per filegroup) for each CPU on the host server**
 - **Consider Tempdb usage and apply recommendation to it as well**
- **Utilize maximum # of spindles - Multiple data files can be used to span as many underlying disks as necessary**
- **Multiple filegroups may be optimal for backup / recovery scenarios of larger datasets**

©Kimberly L. Tripp
Designing for Performance and Availability

Common Questions...

- Where do I place tempdb?
- Be aware of tempdb contention on internal allocation structures (SGAM, GAM, PFS, IAM) caused by:
 - Repeated create and drop of temporary tables (local or global)
 - Table variables that use tempdb for storage purposes
 - Work tables associated with CURSORS
 - Work tables associated with an ORDER BY clause
 - Work tables associated with an GROUP BY clause
 - Work files associated with HASH PLANS
- Resolution:
 - Affinitize allocation structures in file by CPU
- Solution: Install hotfix described in KB 328551/SP4
 FIX: Concurrency enhancements for the tempdb database
<http://support.microsoft.com/default.aspx?scid=kb;en-us;328551>

©Kimberly L. Tripp
Designing for Performance and Availability

Common Problems

- Firmware, driver issues and bugs
 - Stale Read – Data returned via a ReadFile or ReadFileScatter does not reflect the last successful write operation
 - Lost Write – Data sent via a WriteFile or WriteFileGather is never represented by stable storage
- Error 823 or 605
 - Trace flag –T818 tracks last 2048 page writes
 - PRB: Additional SQL Server Diagnostics Added to Detect Unreported I/O Problems
<http://support.microsoft.com/default.aspx?scid=kb;en-us;826433>
 - DBCC DROPCLEANBUFFERS
 - DBCC CHECKDB

©Kimberly L. Tripp
Designing for Performance and Availability

Review

- Introduction to Filegroups
- Using Filegroups for Maintenance
- Using Filegroups for Availability
- Using Filegroups for Performance
 - Guidelines for Creating Filegroups
 - Distributing Data Across Disks by Using Filegroups
 - Combining Filegroups with Hardware RAID
 - Functional Partitioning Basics
- Common Questions/Best Practices

©Kimberly L. Tripp
Designing for Performance and Availability