

SQL Server Application Installation **from dev to professional quality**

Ken Lassesen





Developer's world

- Create a SQL Database
 - Run with default location
 - Run with default settings
 - May pick Simple Recovery Model
 - Load is typical single user (dev)
- Hardware crash
 - Replace hardware
 - Reinstall OS/Applications
 - Recover code and rebuild the database



Naïve move to production

- Dev is “the” IT source
 - Mimics his environment
 - Knows that backups are important
 - Disk image is backed up nightly
 - Stored off site
- Handed off to a hosting service...
 - All backups handled by hosting service
 - “Cloud computing” is similar.



Physical local machine -- oops

- Drive fails
 - All orders and activity for the day is lost
- Does not handle load
 - No load simulator written
 - Deadlocks, time outs
 - CPU is <30%



Protecting against drive failure

- Use Transaction Log and Backups
 - Must be on a different *physical* spindle from all database files.
 - Be aware of disk encryption issues
 - Beware of multiple logical drives on a single spindle
- Use data redundancy
 - Parity (RAID 5+) or Mirroring (RAID 1)
 - O/S supports it
 - Hardware is preferred
 - Allows low downtime for clean drive failure
 - Hot pluggable SATA enclosures are cheap



Messy failure

- Failure results in database corruption that forces a restore
 - You have two perfect mirrors of the *same corrupt database*.
- Can you recover?
 - If using Blobs, you need the FULL recovery model
- Time to recover
 - Can be many days or weeks!



Backups for fast recovery

- Three components
 - Full Backup
 - Differential Backup
 - Transaction Log
- A “Backup” empties the transaction log
 - I have seen transaction logs that have three years of transactions...
 - No one setup a back up!



The recommended pattern

- Use a N-division model
- Determine Full Backup schedule
- Divide time between into N parts and do Differential backup in each period
- Divide time between differential backup into N parts and do a transaction log backup in each period



Recovery in shortest time

- Steps (Average $2N$ steps)
 - Restore last Full Backup
 - Restore Differential backups after the Full Backup
 - Restore Transaction logs backup after the Differential
 - Restore the current transaction log.
- Best practice: Keep the backup and logs on different physical spindles.



Reference:

- <http://msdn.microsoft.com/en-us/library/ms189895.aspx> - FULL
- [http://msdn.microsoft.com/en-us/library/ms178105\(v=SQL.100\).aspx](http://msdn.microsoft.com/en-us/library/ms178105(v=SQL.100).aspx) - DIFFERENTIAL
- <http://msdn.microsoft.com/en-us/library/ms187607.aspx> - TRANSACTION LOG



Where are we for physical machine?

- Database Files
 - Redundant – two+ drives
- Transaction Log
 - One+ drive (Redundant recommended)
- Backups
 - One+ drive (Redundant recommended)



Performance: The physical world

- Designing a beautiful sports car with a Prius drive train does not win a road rally!
- Often performance problems are hardware problems.
 - Do not start a rewrite until you have verify the hardware environment is near-optimal. Hardware is cheap. Devs are not.



Partition Alignment

- If the drive was not formatted under Windows Server 2008 – you may have a problem.
- Can increase performance 30+%
- <http://www.brentozar.com/archive/2009/01/partition-alignment-in-virtual-machines/>

Physical Track #1	Logical Track #1
Logical Track #1	Physical Track #2



Latency – spindle heads

- Insert a record (HM – Head move)
 - Temp: HM- Record about to insert
 - Rollback may happen
 - Temp Log: HM – Record above
 - DB: HM – Write record
 - DB: HM – read RI index table
 - RI may fail
 - DB: HM – insert record in Index
 - Temp: HM – remove stuff
 - Temp Log: HM – Record above



What is the cost of head moves?

- 7+ HM
 - Normally a lot of this is cached
 - Under load with low memory may result in physical HM
 - 5400 RPM → 0.07 Seconds best time
 - Single user



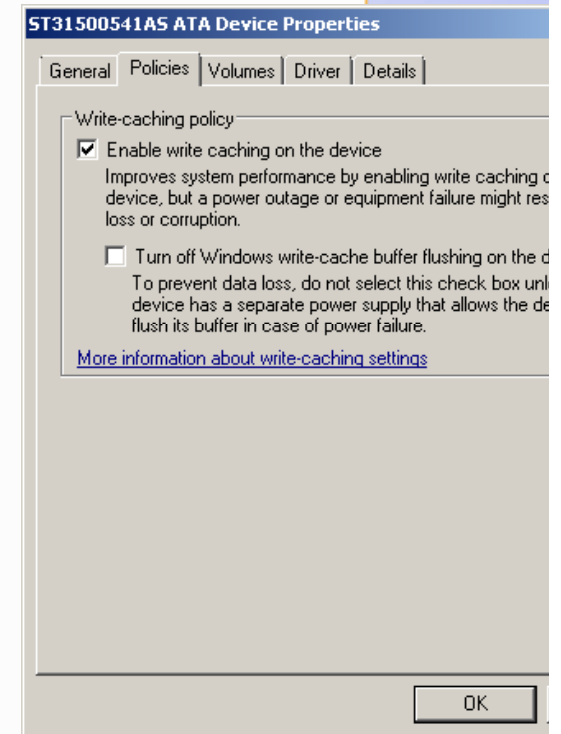
Why not just leave the head in place?

- Transaction logs are sequential
 - Move Transaction Log to a dedicated disk
 - Move Temp Db Transaction Log to a dedicated disk
- You can't break the logging speed limit!
 - Should be drives with fastest sequential write speed



SQLIO

- Utility to test performance of drives
- Write Caching OFF for
 - Disk Drives
 - Controllers
- If enabled, writes are reported as written *before physical write*.
 - Increases performance
 - Lost of data/database corruption
- http://sqlserverpedia.com/wiki/SAN_Performance_Tuning_with_SQLIO





Putting the database on multiple drives

- FileGroups are the physical files used for the database.

```
CREATE TABLE [eAudit].[Client](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [MetaData] [xml] NOT NULL,
    [AddDateTime] [datetime] NOT NULL,
    CONSTRAINT [PK_Client] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH ( STATISTICS_NORECOMPUTE = OFF)
    ON [PRIMARY]
) ON [PRIMARY]
```



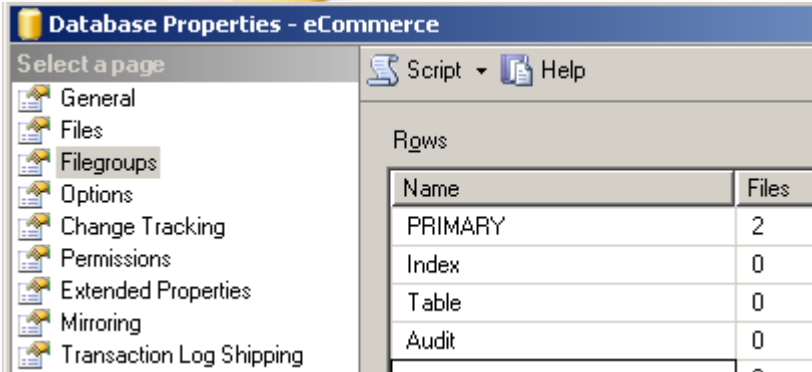
Assign to File Group when designing

- Some obvious divisions:
 - Tables and Indexes on different drives
 - Serial Tables (Audit Logs) on different drive

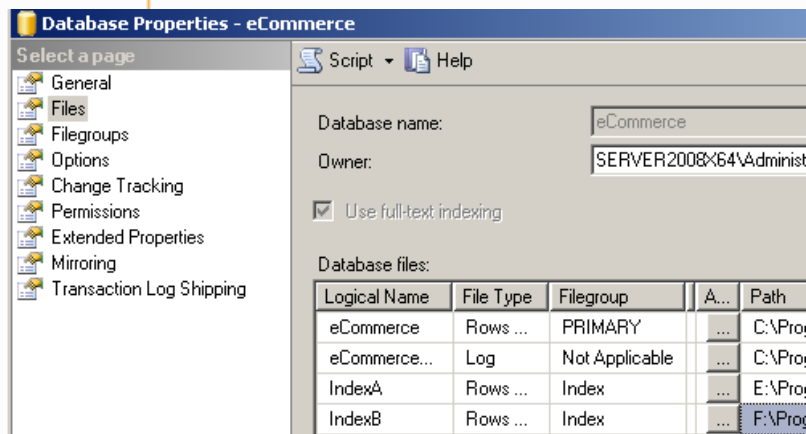
```
CREATE TABLE [eAudit].[Client](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [MetaData] [xml] NOT NULL,
    [AddDateTime] [datetime] NOT NULL,
    CONSTRAINT [PK_Client] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH ( STATISTICS_NORECOMPUTE = OFF)
    ON [INDX]
) ON [TABLES]
```



How to add File Groups



- Add the File Groups



- Add the Files for each group



Packaging: Install to one drive

- Create file groups as appropriate
- File groups do not compel different drives
- Very simple to improve performance by moving file groups to different physical spindles
 - Deliver a *built-in scalability plan*



Monitoring Physical Performance

- SQL records the IO Stall times

```
SELECT
    name as [Database Name],
    type_desc as [File Type],
    physical_name ,
    case when A.NumberReads > 0 then A.IoStallReadMS/A.NumberReads
else null end as [Read Latency],
    case when A.NumberWrites > 0 then A.IoStallWriteMS/A.NumberWrites
else null end as [Write Latency],
    case when A.NumberReads > 0 then A.BytesRead/A.NumberReads else
null end as [Read Size],
    case when A.NumberWrites > 0 then A.BytesWritten/A.NumberWrites else
null end as [Write Size]
FROM fn_virtualfilestats(NULL,NULL) A
JOIN sys.master_files C
    On database_id=DbId
    and file_id=FileId
Order by A.NumberReads desc, A.NumberWrites desc
```



Hosting and Virtual issues

- Same issues apply
 - Virtual:
 - Use Pass-Thru to physical drives.
 - Hosted:
 - If you own machine
 - Verify that logical drives are physical spindles
 - If you rent a database, make sure SLA covers:
 - Backups
 - Transaction Logs
 - “Because they are in the business does not mean best practises”



One little, two little, three little hard drives...

- Good Installation (all redundant)
 - File Groups for database
 - Index – 2+ drives
 - Blobs – 2+ drive
 - Audit – 2+ drive (Serialized tables)
 - Primary – 2+ drive
 - Transaction Log – 2+ drives
 - Temp DB:
 - 2N+ drives --1 logical drive per CPU
 - Transaction Log -- 2 Drives
- QUAD box: 20 drives.



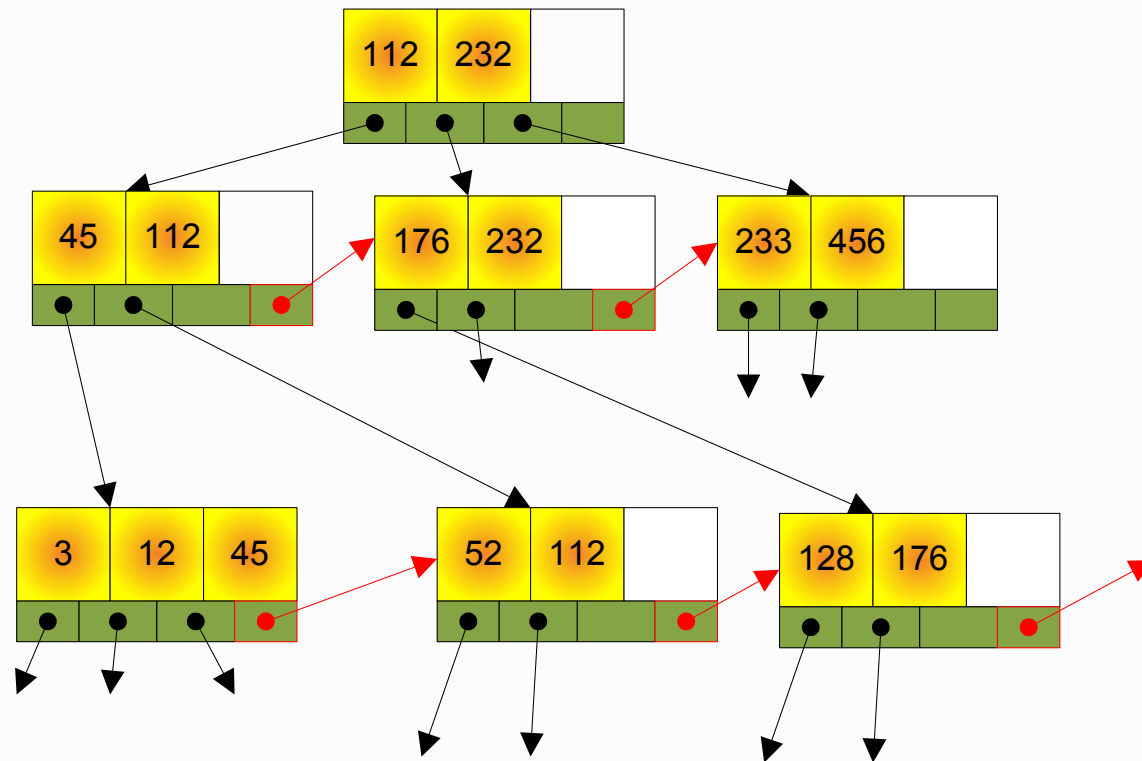
Index Fill Factor

- Once upon a time, long, long ago most primary keys were integers...
 - Identity(1,1)
 - Fill Factor = 0 (100%) was logical because *every new key was appended to the end of the index..*
 - This because the installed default value for SQL Server
 - Once set, screams will arise if MSFT dare to change it!



Indexes are B+ Trees

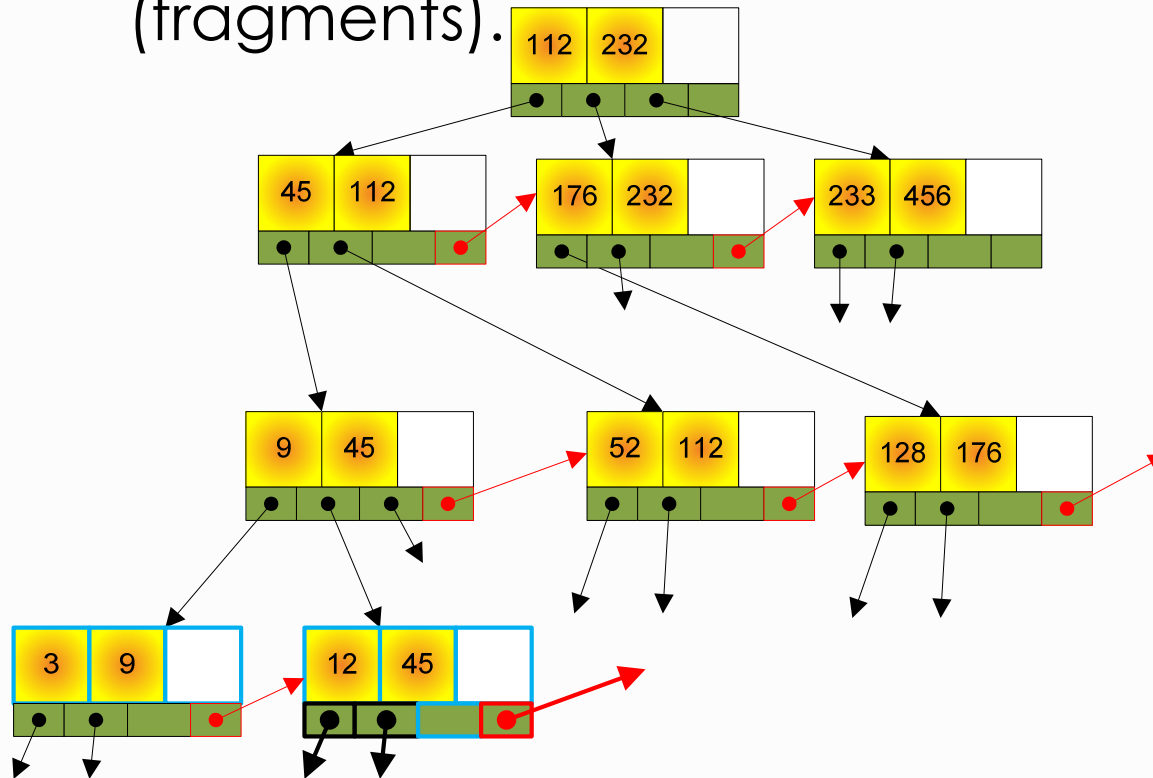
- B+ date back to the 1970's





Index Split

- When a key record is filled, it splits (fragments).





Common fragmentation reality

- If an index is NOT Identity(1,1), for example a GUID...

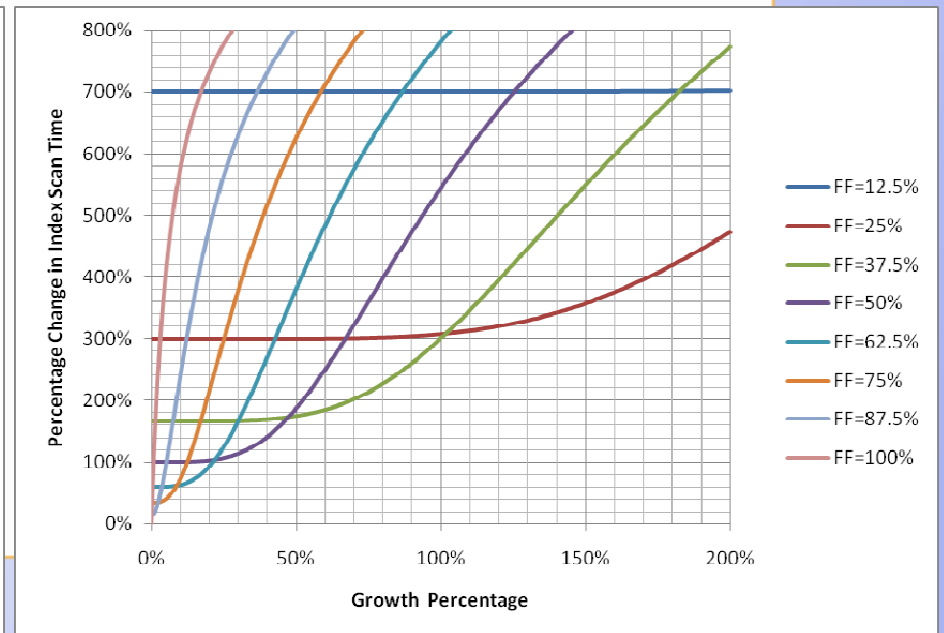
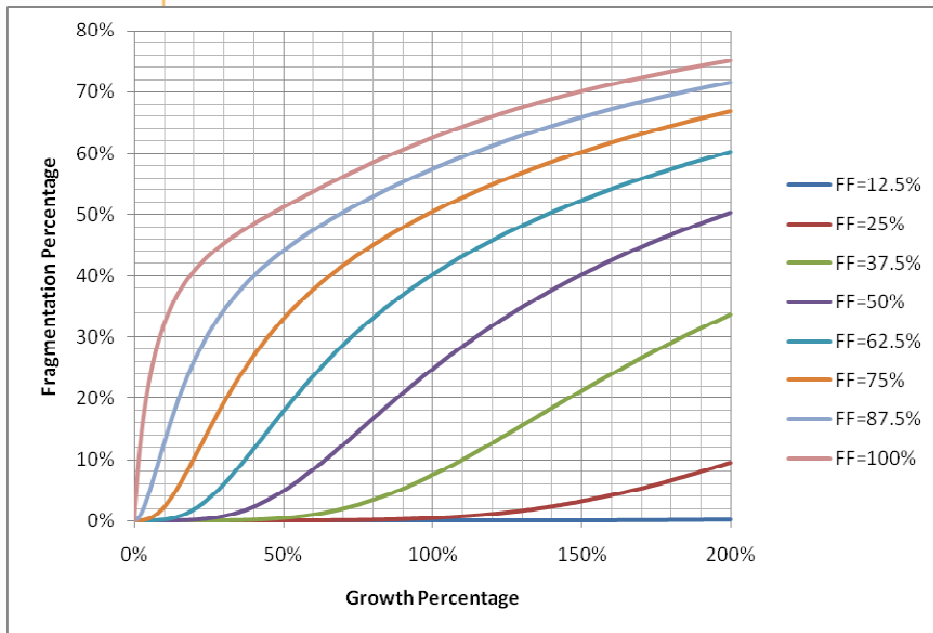
```
SELECT
    object_id,
    index_level,
    avg_fragmentation_in_percent,
    fragment_count,
    page_count,
    avg_page_space_used_in_percent,
    record_count
FROM sys.dm_db_index_physical_stats(
    db_id(), NULL, NULL, NULL, 'DETAILED')
Where index_type_desc <> 'HEAP'
AND alloc_unit_type_desc='IN_ROW_DATA'
```

	object_id	index_level	avg_fragmentation_in_percent	fragment_count	page_count	avg_page_space_used_in_percent	record_count
1	180911716	0	99.6390615600561	9974	9974	86.434815913022	428974
2	180911716	1	62.962962962963	27	27	59.3068569310601	9974
3	180911716	2	0	1	1	4.31183592784779	27



Optimal Fill factor

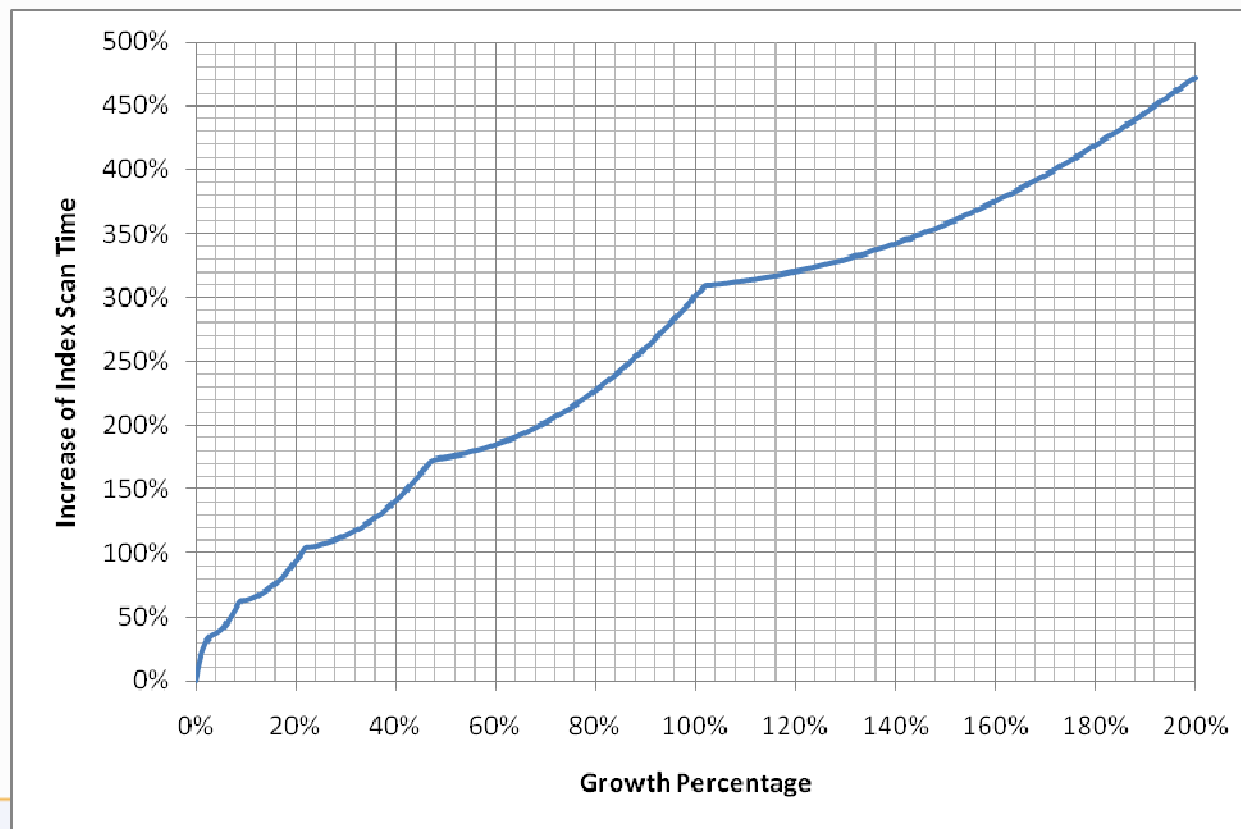
- Mathematics boil down to 2 items
 - Growth Percentage between defragmentation
 - Key Size.





What's the best that you can do?

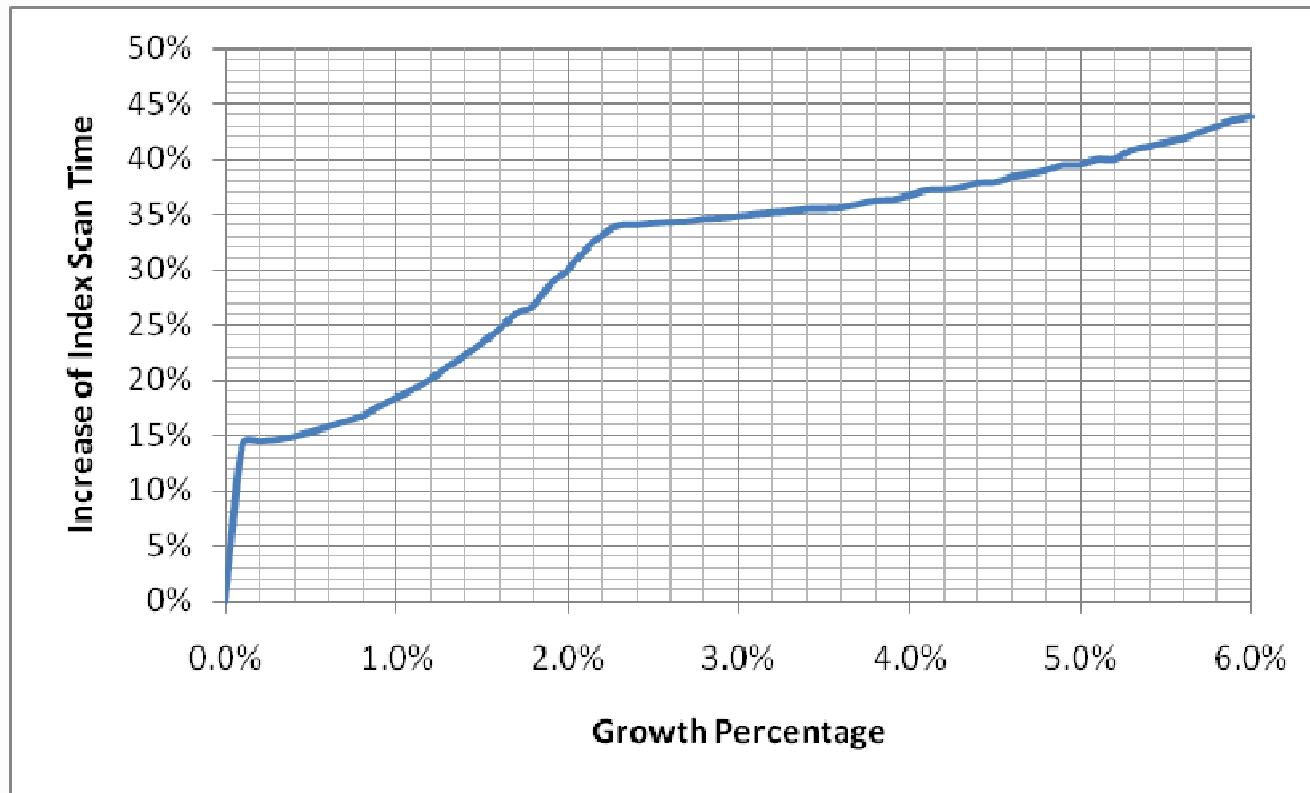
- You want to defrag often...





How often? Try once a week...

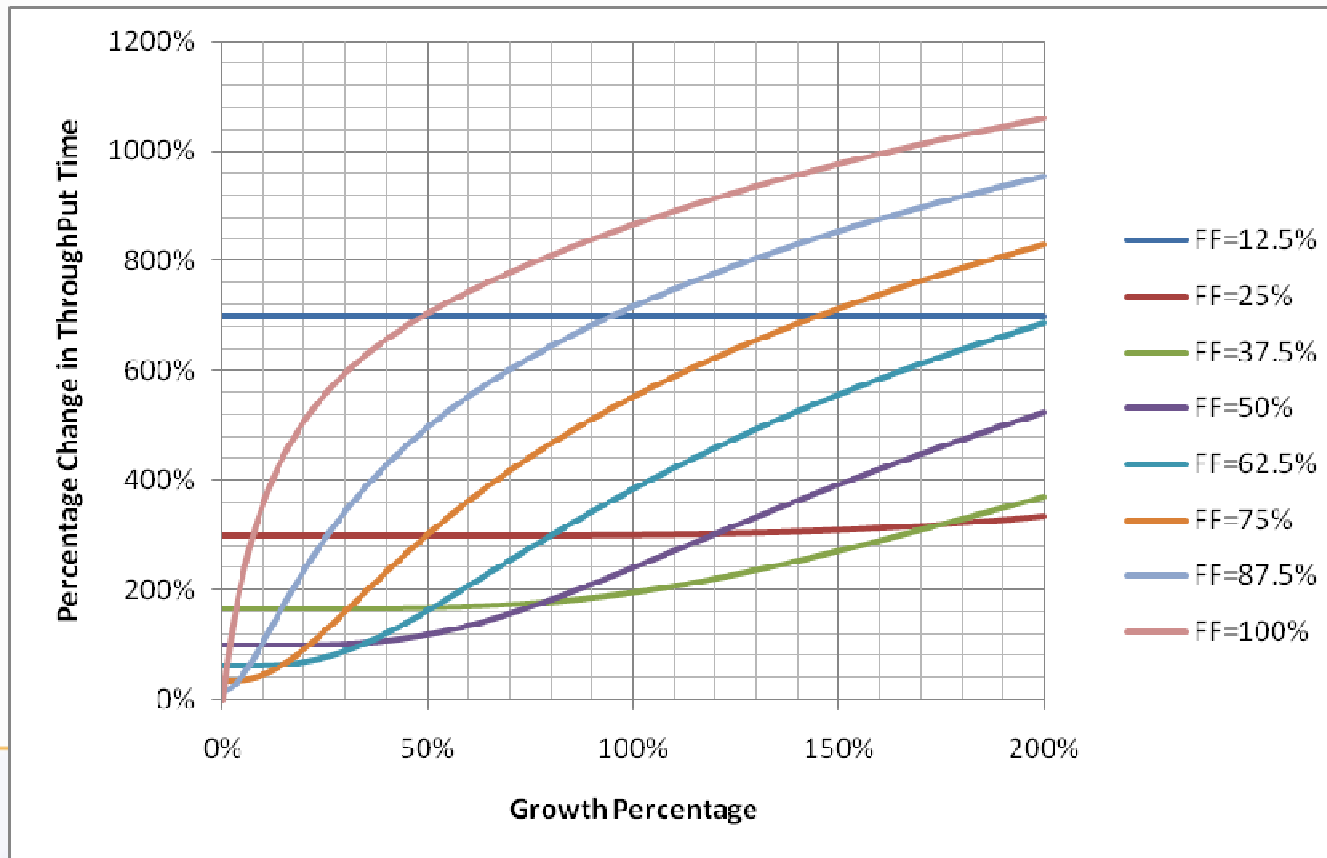
- Keep growth between defrag < 2%





Second dimension: Throughput

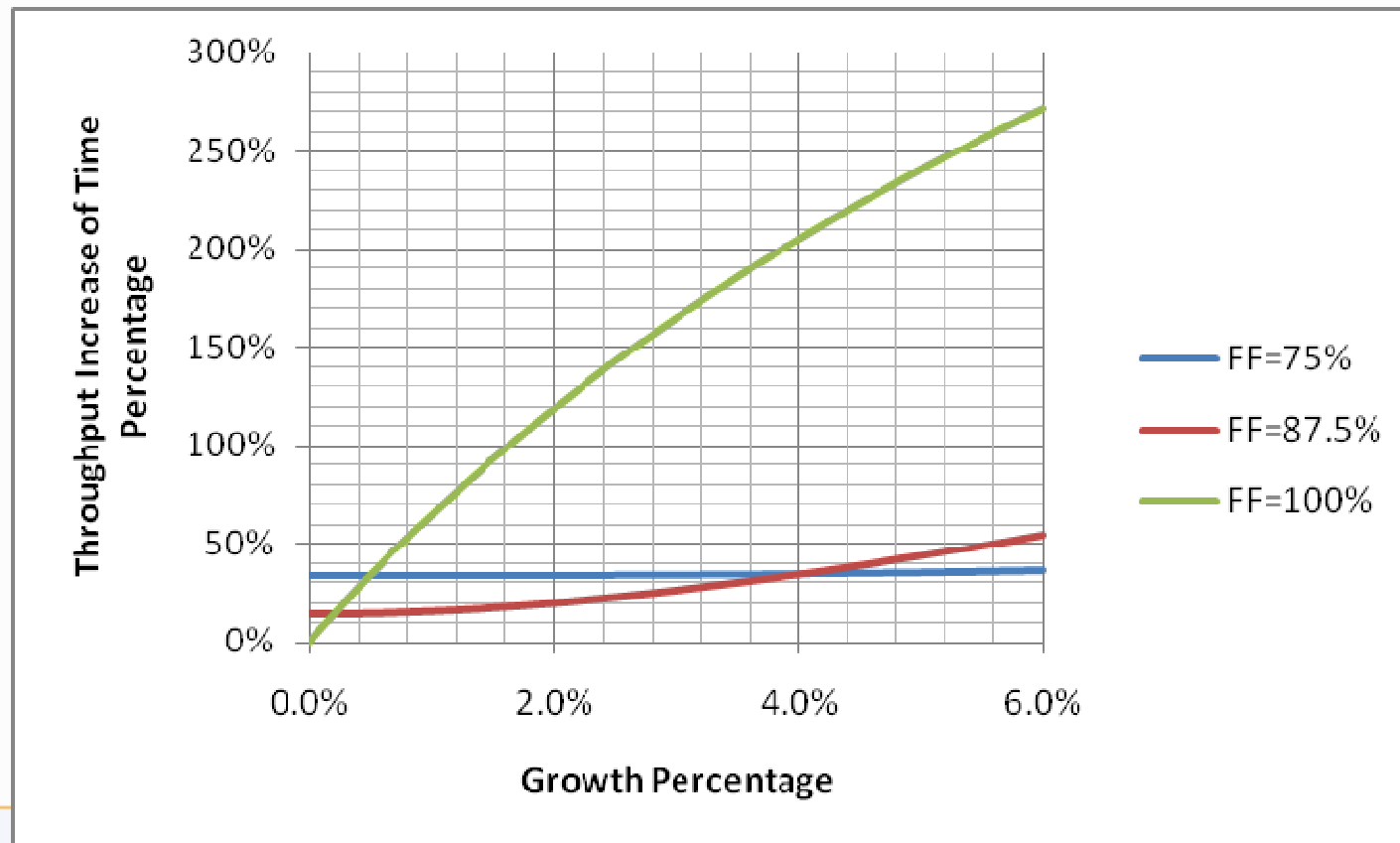
- You have empty slots to read with a fill factor....





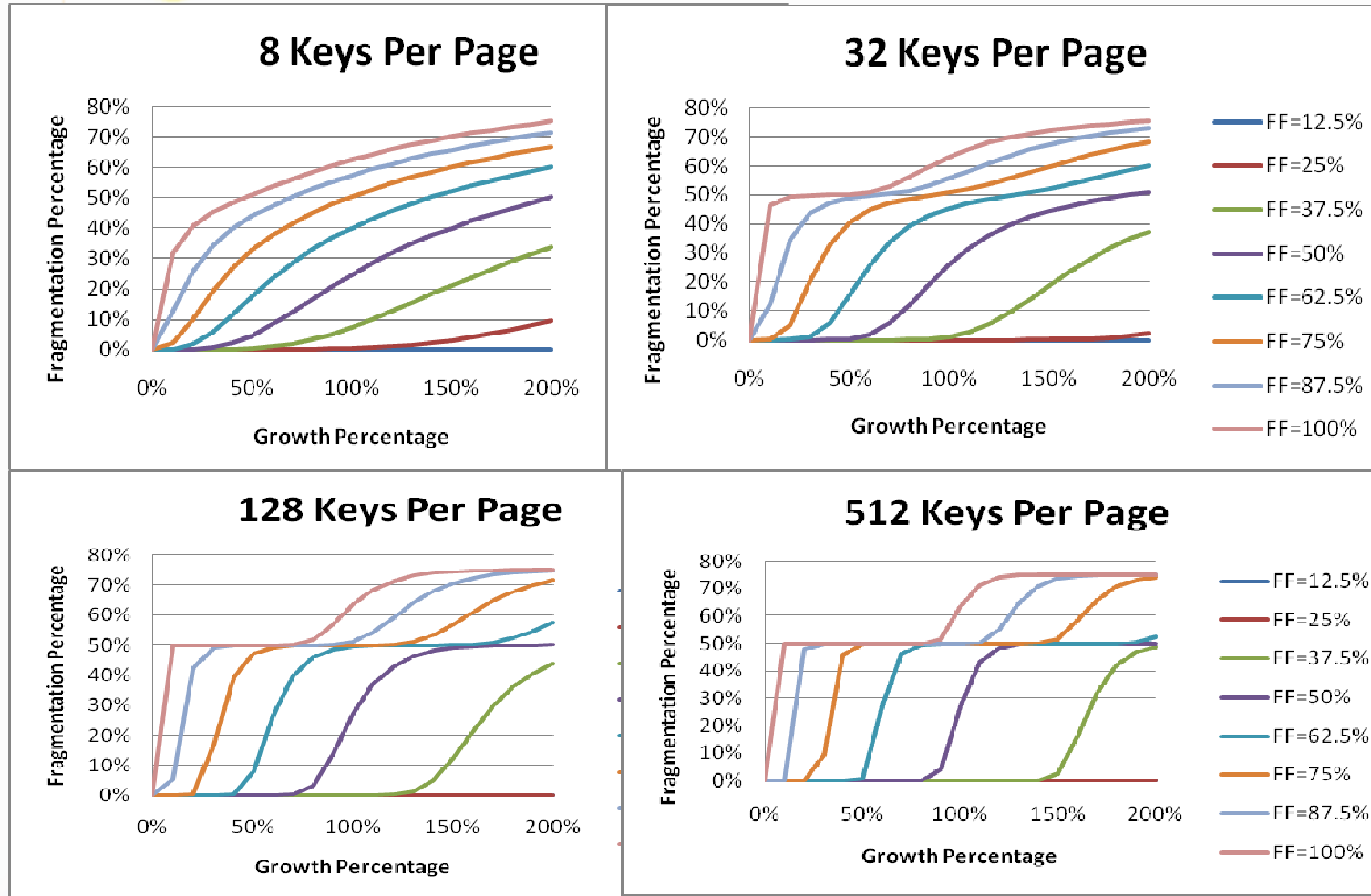
Micro-preformance

- A lower fill factor is often better





Key Size Impact





The mathematics... for optimal fill factor

```
CREATE FUNCTION [dbo].[Fn_OptimalThroughputFillFactorForRandom]
(
    @KeyBytes float,
    @GrowthPercentage float
)
RETURNS int
AS
BEGIN
    If @KeyBytes < 2
        SET @KeyBytes=2
    If @GrowthPercentage > 0.06
        SET @GrowthPercentage = 0.06
    If @GrowthPercentage < 0.001
        SET @GrowthPercentage = 0.001

    DECLARE @FillFactor float
    DECLARE @Rate float
    DECLARE @Offset float
    Set @Rate=- 5.2312 * Power(@keybytes,-0.244) -- R=0.95
    Set @Offset=1 - 0.2193 * Power(@keybytes, - 0.462) -- R :
    Set @FillFactor= CEILING(100 *
        (@Rate * @GrowthPercentage + @Offset))
    If @FillFactor < 50
        SET @FillFactor=50
    If @FillFactor > 99
        SET @FillFactor=99
    RETURN @FillFactor
END
```

```
Create FUNCTION Fn_OptimalThresholdFillFactorForRandom
(
    @KeyBytes float,
    @GrowthPercentage float
)
RETURNS int
AS
BEGIN
    If @KeyBytes < 2
        SET @KeyBytes=2
    If @GrowthPercentage > 0.06
        SET @GrowthPercentage = 0.06
    If @GrowthPercentage < 0.001
        SET @GrowthPercentage = 0.001

    DECLARE @FillFactor float
    DECLARE @Rate float
    DECLARE @Offset float
    Set @Rate=- 5.0189 * Power(@keybytes,-0.218) -- R=0.99
    Set @Offset=1 - 0.9774 * Power(@keybytes, - 0.574) --
    Set @FillFactor= CEILING(100 *
        (@Rate * @GrowthPercentage + @Offset))
    If @FillFactor < 50
        SET @FillFactor=50
    If @FillFactor > 99
        SET @FillFactor=99

    RETURN @FillFactor
END
```



Summary

- Know where your data is physically
- Plan for disaster.
 - Test for disaster if possible (i.e. **remove** one drive physically... and try to recover)
- Use FileGroups
- Schedule RegularDefragmentation
- Set optimal Fill Factor Size



More Readings:

- <http://lassesen.com/Publications/abid/55/Default.aspx>
 - [Best Practices for PTC Windchill on Microsoft SQL Server \[PDF\]](#)
 - [Best Maintenance Tools for PTC Windchill on Microsoft SQL Server 2005 \[PDF\]](#)
 - [Best Drive Configuration Practices for PTC Windchill on Microsoft SQL Server \[PDF\]](#)