

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

» summary

As the data in Microsoft SQL Server tables changes their indexes change. Over time these indexes become fragmented. This fragmentation will adversely affect performance. This technical white paper provides information to help you understand the detailed mechanics behind fragmentation. It will also help you understand the methods and approaches for performing defragmentation so you can improve your SQL Server's performance.

» overview

This technical white paper will help you understand SQL Server fragmentation and the performance benefits you can gain on your SQL Servers by continuously monitoring and managing index fragmentation.

The following is a summary of the key topics covered in this paper:

- The difference between disk and SQL Server internal and external fragmentation
- How fragmentation affects performance.
- The mechanics behind performance robbing data voids
- The pros and cons of various approaches to managing fragmentation
- How SQL defrag manager provides a better, more efficient and easier approach to solving fragmentation problems in SQL Server.
- How to judge the improvements gained by defragmenting your server.

Warning: This white paper will get a bit technical as it is intended for DBAs who want to truly understand the details and key components of fragmentation in SQL Server.

SQL Server fragmentation explained

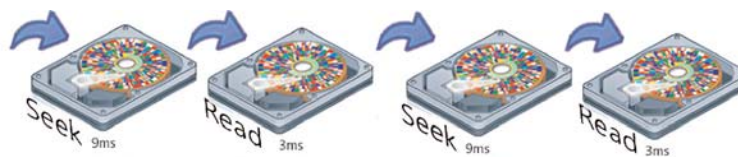
WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

Q. What is SQL Server fragmentation? Is it different than physical disk fragmentation?

A. SQL fragmentation is not physical disk fragmentation. Not all fragmentation is equal!

Physical disk fragmentation is likely what comes to mind when fragmentation is first discussed. Physical fragmentation is a side effect of how hard drives and Windows work. It is common knowledge that regular disk defragmentation is required to achieve optimal performance from your PC. Windows even includes a basic defragmentation utility.

Physical fragmentation slows down your PC because reading data is interrupted by head seek delay. Windows fits files into free space, often breaking the file into segments stored apart from one another. A hard drive's head relocates to read each individual segment. As it moves to each segment the head 'seeks' - often at a cost of 3-4 times the time it takes to read the segment itself. Physical fragmentation primarily affects desktop or laptop PCs containing one hard drive. The single drive must sequentially gather data - so on a fragmented disk it seeks, reads, seeks, reads - these 4 operations are performed one after another. Defragmented, the operation ends up as seek, read, read. We reduce the total cost of 24ms to 15ms in our simple example.



A single IDE or SATA drive is the most common workstation storage solution. Using the mfg's specs for this typical drive multiple reads take ~24ms to execute. When defragmented the same reads would take ~15ms (or ~38% faster) to execute.

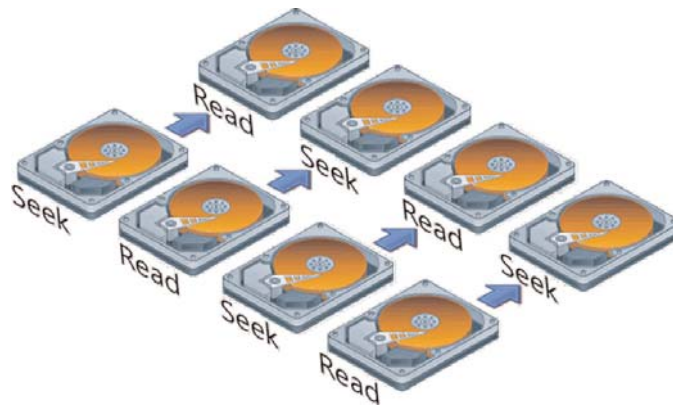
Shown above, in Figure 1, we see physical fragmentation forcing the drive to read the file as two segments. The seek costs us 18ms, while the read costs 6ms. Considering the average file is made up of hundreds of segments, the seek latency multiplies and becomes quickly evident as your system slows down over time.

Physical defragmentation products such as Windows defrag, Power Defrag™, Page Defrag™ (another Microsoft tool), or the granddaddy of them all, Diskeeper 2007™ work very well when repairing segmented files. Diskeeper's technology is licensed to Microsoft as the defragmentation tool internal to Windows. In fact, Diskeeper's latest innovations bring physical defragmentation capabilities to a completely new level. All of these products reorder the data on your disk, consolidating files into fewer segments to minimize "head seeks" - providing faster boot times, quicker file reads, and a more responsive system overall.

However, **physical disk fragmentation is not the same as SQL Server defragmentation!** SQL Server is different. SQL Servers use advanced storage systems with multiple drives working in tandem, changing the way files are read. Physical fragmentation is something solved with hardware - not with defragmentation scripts or tools.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™



The fault-tolerance in database storage overcomes the vast majority of physical disk fragmentations' impact. Best practices universally prescribe multi-drive storage subsystems for production SQL Servers. Most SQL Servers use multi-drive storage such as RAID arrays, SANs, and NAS devices; there are always multiple drives acting in tandem. Hard disk controllers supporting drive arrays are aware of the alternate seek/read dynamic and tailor communications with the array for maximum I/O.

As a result, files are distributed across many drives inherently becoming segmented. Working in tandem, however, allows one drive to seek while the others read. With the common configuration of 5 drives, a seek delay of 9ms per drive allows 2 drives reading for 3ms with no seek delay impact at all. Data storage drives are generally much faster than workstation drives, so seek times of 4ms and read times of 1.5ms are not unusual.

There are many DBAs who run a traditional physical defragmentation program in tandem with their intelligent drive controller which results in limited improvement. Physically defragmenting a file in an array implicitly leaves the file segmented across the virtual unison of tandem drives. It's by design. The goal is to gain the most performance while incurring the least overhead – so don't run physical defrags if they slow the storage by 50% while running, and ultimately improve read speeds 1-2%.

The most important concept to understand is that the controller, physical defragmentation programs, and multi-drive arrays are unaware of what SQL Server is doing with the file data internally. By focusing on SQL Server's representation of data - how SQL Server has laid out the database itself, how full each page is, and how effectively we're utilizing available SQL Server resources, we can optimize to the 'next level' of SQL Server performance, solidly trumping any benefit to physical defragmentation by orders of magnitude. In a nutshell, SQL Server's performance can be most improved by focusing on its internals. In fact, once you start focusing on defragmentation at the SQL Server level – whether with manual defragmentation or with the automated defragmentation provided with SQL defrag manager, you may decide that physical defragmentation is no longer needed!

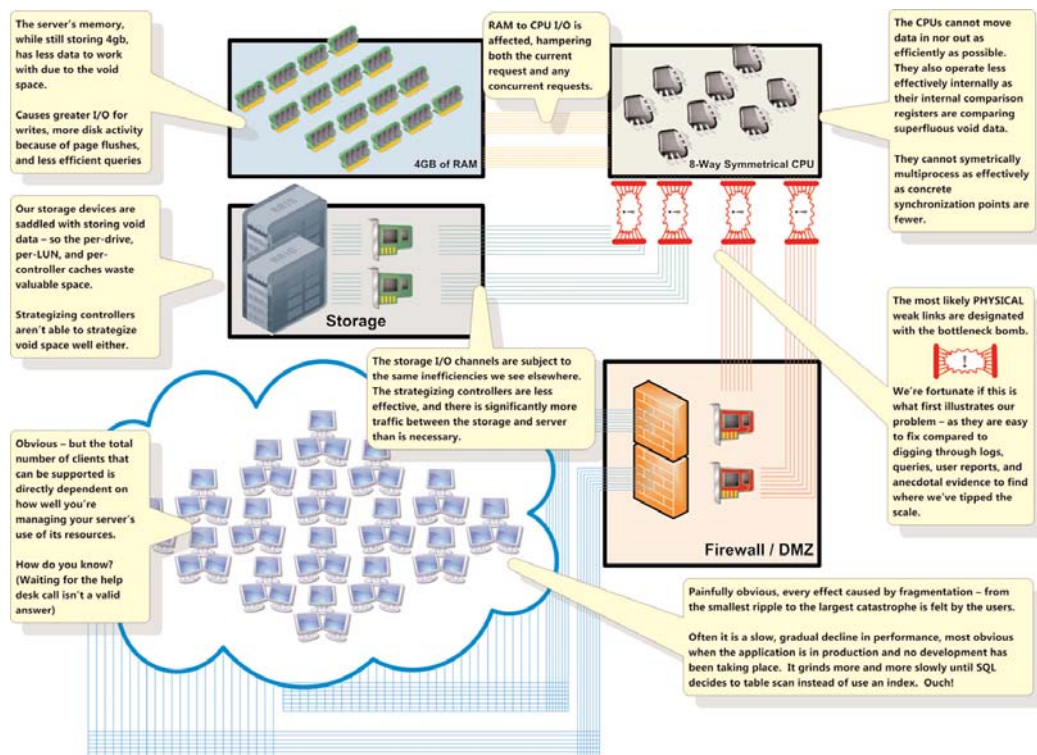
SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

Q. How is SQL Server's fragmentation affecting my Server?

A. Fragmentation of SQL Server indexes mainly creates wasted space that can affect your server performance much more than one might expect.

Fragmentation of your SQL Server's internal allocations and page structures result in 'gaps' or 'void' space that is dead weight carried along with valid data. Your backups, storage, I/O channels, buffer memory, cached data, logs, tempdb, CPUs and query plans are impacted by these unnecessary voids. SQL's fragmentation continually eats away at these resources with nearly every update, delete, insert, and table/index change. If ignored, fragmentation can be the proverbial 'death by a thousand cuts' to a server's performance and scalability.



Shown above is a detailed diagram of how SQL Server fragmentation can affect your SQL Server performance and overview of the affected areas. As you identify how the fragmentation affects your server, you'll see that fragmentation effects are cumulative and nearly impossible to predict. SQL defrag manager, however, uses sophisticated algorithms to predict and detect SQL Server fragmentation "hot spots" and to defragment indexes on a continuous basis.

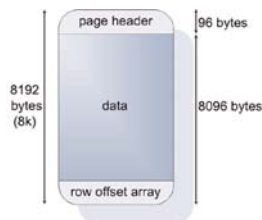
SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

Q. What creates the voids and other adverse effects and how do I get a handle on them?

A. Typical, day to day activity causes SQL Servers to fragment over time. Changes to your data – inserts, updates, deletes, and even changing varchar values contribute to fragmentation. The full list of actions that cause fragmentation is long and the rate of fragmentation varies across different indexes and tables. Sometimes there is a pattern resulting from seasonal or annual peaks (e.g., when financials run). But more often than not, it is difficult to find, predict and proactively manage fragmentation manually.

Let's dive into the details of where these voids sit, how they are created, and how they propagate throughout your server:



SQL Server stores all data, objects, and internal structures in 8192 byte data pages shown in Figure 2. These pages are known only to SQL Server and may be stored in one or more physical files on disk. Data gets a maximum of 8096 bytes per page – the rest of the page contains the page header and row locations. When creating a table or index, SQL Server pages fill according to the fill factor you specify (or the closest approximation.)

Over time, insert, deletes, and modifications (such as widening the value in varchar fields) fill the page and ultimately overflow the page creating a 'page split'. Splitting divides the full page evenly, putting half of its data on a newly allocated page, and may negate any fill factors you designate. For example, if you designate a fill factor of 80%, over time, due to splitting, your pages may reach a fill factor of 50% or less.

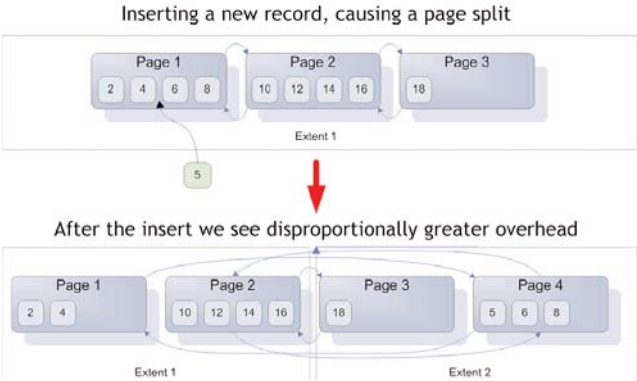
The more that heavy, spiked, or continuous changes occur on a table, the faster and further it and its indexes drift. Since the indexes are based on variants of data in the table, they have their own unique drift profile.

The net result of drifts is waste – lots of it – waste of your disk, I/O channels, server's caches and buffers, and CPU utilization. The waste may also skew your query plans.

The void/waste space is known as "internal fragmentation." Internal fragmentation lowers page density and as a result our server resources trickle slowly away now being increasingly consumed by empty space. SQL does try to fill the voids in split pages – however there is rarely the Tetris™-like fit necessary to reach optimal population post split. The common practice of using an identity column as your clustered index, forces inserts into new pages at the bottom of the table, preventing recovery of the voided space.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

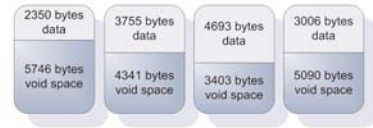


The space used by actual data is reflected in a metric called “page density.” The denser a page, the more data vs. void it contains. A page density of 100% would mean the data page is completely full. Even if the pages had no void, Figure 3 illustrates how the split has introduced other inefficiency in contiguously accessing the pages after the split. Interestingly this parallels physical fragmentation – although it is a completely isolated variant in SQL Server’s management of data vs. the way files are segmented on disk. This type of fragmentation is called ‘external fragmentation’.

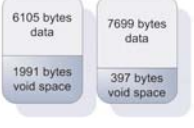
The vast majority of the time, the void grows, instead of filling up neatly. When void space becomes too much (your page density becomes too low), SQL Server will discard the index due to excessive overhead. At this point, fragmentation becomes very evident as very few systems will tolerate discarding indexes in favor of table scans.

For example, shown in Figure 4, the four pages require four logical reads. Defragmentation would condense the data by reorganizing it into two pages and two reads. A 42% reduction in void space is gained by defragmenting the data. This concretely reflects in improved performance and increased scalability. In the example, two times the data could fit in the defragmented space vs. the fragmented space. By reclaiming the voids, we return capacity to our server.

Fragmented, 32384 bytes are allocated for 13804 bytes of data (57% unused void space)



Defragmented, 16192 bytes are allocated for 13804 bytes of data (15% unused void space)



While it may seem trivial on such a small scale, when your average page density is low, you are wasting disk space, incurring more physical I/O, increased logical reads, wasting precious server memory while computing and comparing data unnecessarily. Further, if you are fortunate enough to have an intelligent I/O controller, you are also wasting the benefit of its optimization strategies. It becomes obvious that this process of splitting, voids, progressive order, and rates of decay requires non-stop attention to insure the server is running with as much free resource as it can.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

Q. Ok, so clearly the maintenance benefits are real, and we want to obtain them, now what?

A. Besides SQL defrag manager from Idera, there are two approaches most commonly used for fragmentation today and both have serious disadvantages

If, for a moment, we ignore SQL defrag manager, there are two existing methods for managing SQL Server fragmentation. Neither is ideal, or gives you the information you need to stay informed and on top of the fragmentation challenge. Both leave you completely blind — you won't know if they helped, hurt, stepped on, or blocked your busiest table.

First method: plain old damage control. The server performance degrades slowly and is ignored. All of a sudden, a spot in the database reaches critical mass, performance craters, and is eventually addressed. This is how the majority of DBAs are introduced to the fragmentation problem. They just fix it and wait for the next hotspot or for SQL Server performance to run down again and again. Unfortunately, you will never know when your server is going to act up or how severe the impact will be. Furthermore, there may be cascade effects caused by inadvertent query plan disruption due to fragmentation.

Second: Run a blind maintenance script. These scripts are often quite complex with unpredictable results. They usually work, but may often cause after effects such as blocking or locking and can generate considerable overhead. You have no idea how long it will take the script to find every index — and it must query each one, every time. The script does not track performance benefits, does not track how defragmentation varies each time, and offers no notification of potential problems.

All-purpose SQL Server defragmentation scripts:

- Request information that can cause long blocks or deadlocks.
- Rarely have internal logic to know when to defragment— instead they just steamroll your servers every day whether they need it or not (perhaps many times a day.)
- If you make a minor change to a defragmentation script, you will be required to re-deploy to the new script to all of the servers in your enterprise.
- Should be tailored to each database — but to do this would require near constant “hand-tuning.” A very time-consuming and practically impossible process.
- Aren't able to report when the script was run, what performance enhancements were gained, or how many resources they've reclaimed on your server since you started running them.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

A new third option: Idera SQL defrag manager.

SQL defrag manager offers a totally new way to identify, optimize, manage and automate SQL Server defragmentation. It is designed specifically to overcome the compromises DBAs have to make regarding the important task of fragmentation maintenance.

Consider this: If you are able to eliminate void space, every page of void reclaimed is money back in your corporation's pocket. Those reclaimed resources are regained server capacity that had been lost unnecessarily. SQL defrag manager will reclaim these resources and track the total improvement on every object in your enterprise daily or over a year. You can even produce an annual report showing how much money has been saved through the use of defragmentation technology – and we guarantee that it will be impressive!

SQL defrag manager not only tracks the improvement achieved on each object, it maintains dozens of statistics on each table and index. This information guides SQL defrag manager to determine how often it should check for fragmentation, and if you wish, the method it will use to correct the fragmentation. SQL defrag manager eliminates defragmentation overhead and risk on your servers – there is no agent required on any managed server. There is no job scheduled or script deployed. SQL defrag manager simply runs as a service, quietly in the background with no affect to your production servers.

Unlike scripts, SQL defrag manager's fragmentation detection routines are non-blocking. Defragmentation is also non-blocking, given the DBA has not chosen to rebuild the fragmented object. Rebuilt objects are often not needed. SQL defrag was invented by two DBAs frustrated with the scripts and the handholding their 99.999% available enterprises required. SQL defrag manager will shed light on the fragmentation levels across your entire SQL Server environment — allowing you to quickly detect and manage fragmentation with ease. It will also give you assurance that defragmentation is being handled in exactly the way it should be for that particular database – no more guessing!

You be the judge of how we've done.

SQL Server fragmentation explained

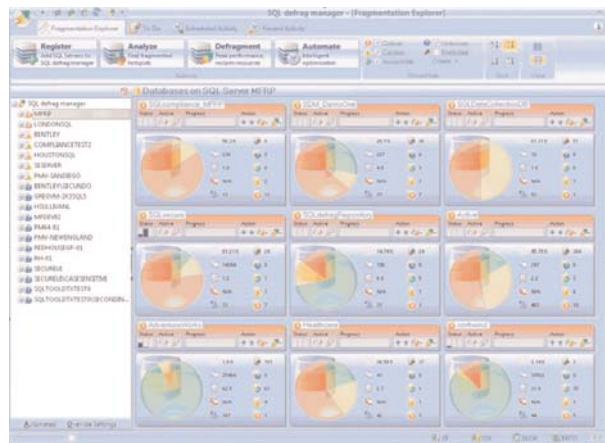
WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™

SQL defrag manager™

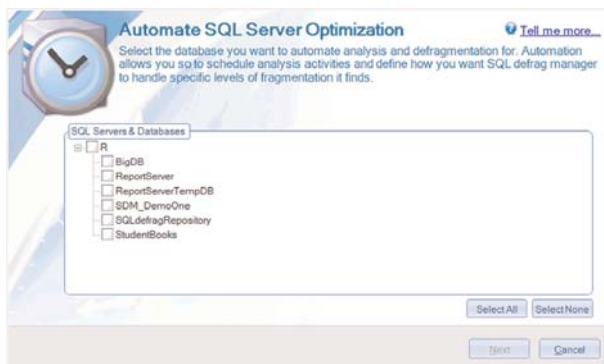


Within moments of installation you'll see a screen like that to the right. Fragmentation level detection in SQL defrag is non-blocking. Unlike scripts or manual queries, risk of unpredictable impact is eliminated. You have immediate and complete visibility into your enterprise and how fragmentation is affecting it.

Displaying all the servers in your enterprise, each is color coded to reflect how much impact fragmentation is having on it, its databases, tables and indexes. You can easily sort to bring the items most in need of your attention to the forefront.



Quickly drill down into the server, database, and locate an offending table/index by clicking the pie chart. If you want to fix it, one click safely fixes it (again, using non-blocking techniques) and reflects the improvement right away.



Once you find the fragmented spots, let SQL defrag manager to make sure it never becomes a problem again. You can set it to send you a summary and let you know if there is a problem, or it can simply defragment as needed – automatically.

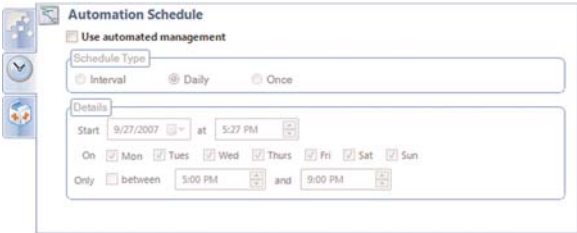
To the left is the automation screen. This instructs SQL defrag manager to take a look at this object regularly. If you wish, you can specify a custom range for responses. This is generally set at the server and all child objects inherit the setting – but you have the control. Simply override this function on any object in your environment.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™



Intelligent Scheduling



Setting the schedule you wish for the object to be checked is straightforward. Start and end ranges can be specified to insure that all activity is performed to your instruction and can be restricted to predefined lull or 'safe' windows. You can also have it run a one time job at a convenient time for you — such as on the weekend.



Full Control – from “Tell Me” to “Fix”

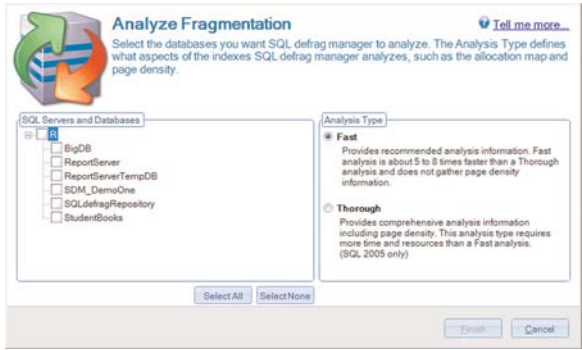


If SQL defrag manager finds an object outside of the acceptable range, you determine how SQL defrag manager will react. You can allow it to reorganize – non-locking and passive. If that fails, you can have it tell you then, or keep on going trying the more aggressive rebuild which blocks.



Automatic, Semi-Automatic or Ad-hoc

Set up any configuration of automated, intermittently automated, or ad-hoc query and remediation.



SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGER™



Defragment your way - while insuring maximum benefit.

- Have it notify you of approaching thresholds, so you can manually defragment when you wish.
- Have it try non-blocking remediation and if that cannot reach an acceptable level, tell you.
- Have it try non-blocking remediation, and if that doesn't reach acceptable, try more invasive blocking.



Custom tailor defragmentation to your enterprise – Schedule, Set the Range, and Set the Action.

- If you set only the server level all databases, tables, and indexes will inherit.
- Override any table you feel should be handled differently – perhaps with a custom range or time.
- Limit activity to outside specific windows on particular days – insure balanced operation even on weekends.



Let SQL defrag manager defragment for you.

- The tool is as interactive or automated as you wish it to be.
- There is no need for you to watch every potential problem in your enterprise. SQL defrag will let you know.



Central Enterprise Management Console – run the client from any machine.

- The SQL defrag manager Management Console provides a real-time window into fragmentation levels.
- Sit at your desk - view and manage defragmentation activity across all of your servers.
- Agent-less collection mechanism. There are no resident scripts on any of your monitored servers.



Lightweight Collection

- Fragmentation details are intelligently collected based on a customizable schedule, keeping overhead on your monitored servers low and controlled.
- If SQL defrag manager predicts a problem will occur before it is next scheduled, it'll let you know.



10,000 ft. to 1 ft. + 360° (Reporting - Coming in the next release, and currently available on request!)

- Comprehensive overview reporting covers every aspect of your enterprise at a glance.
- Detailed breakdown reports show exactly how each object has been managed, and the improvement.
- Summary improvement – see how much SQL defrag has benefitted you and your company – every day.
- All of the reports are in Reporting Services. Subscribe to your reports and read with your morning coffee.
- Model-based reporting allows for easy development of your own custom reports.

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGERTM

SQLdefrag Central
Home > SQLdefrag_Reports

Home | My Subscriptions | Site Settings | Help
Search for: [] Go

Contents: **Reports**

New Folder | New Data Source | Upload File | Report Builder | Show Details

- All Automated Objects** (new)
Shows all objects monitored by SQL defrag manager (servers, databases, tables and indexes) that are scheduled for automated analysis/defrag and their schedule type.
- All Monitored Objects**
All objects (servers, databases, tables and indexes) that have been registered with SQL defrag manager.
- All Non-Automated Objects**
Shows all objects monitored by SQL defrag manager (servers, databases, tables and indexes) that are NOT scheduled for automated analysis/defrag.
- Page Density Change Over Time by Server and Database - Parameterized**
Shows page density changes as a result of index operations over time, filtered by server, database, and minimum page count.
- Resources Reclaimed Over Time**
Graphs and tabular information for each server, database, and table showing the number of pages reclaimed over time using SQL defrag manager.
- Tables without Clustered Indexes (Heap)**
Tables registered with SQL defrag manager that do not have clustered indexes, also known as heap tables. These tables can cause significant overhead as they cannot be defragmented using SQL defrag manager or other traditional means.
- Top 10 Fragmented Clustered Indexes**
Top 10 report showing the clustered indexes with the greatest amount of fragmentation.
- Top 10 Fragmented Indexes - Parameterized**
Top 10 report showing the indexes with the greatest amount of fragmentation, filtered by server, minimum page count, and clustered/non-clustered index type.
- Top 10 Largest Databases - Parameterized**
Top 10 report showing the largest databases registered with SQL defrag manager, filtered by server.
- Top 10 Largest Indexes - Parameterized**
Top 10 report showing the largest indexes, filtered by server and clustered/non-clustered index type.
- Top 10 Largest Tables - Parameterized**
Top 10 report showing the largest tables, filtered by server.
- Table Subreport**
Sub-report showing a graph of all registered objects, their counts, and automated vs. non-automated status. Used in several other main reports.

SQL defrag managerTM All Automated Objects

Automated vs. Non-Automated Counts

Total Objects: Total # Automated: 2, Total Indexes: 14 of 14

Server	Database	Table	Index	Schedule	Automated	Non-Automated
BJ	SQLdefragRepository	ActivityLog	IX_ActivityLog	Internal	14 of 14	0
		AnalysisState	IX_AnalysisState	Internal	7 of 7	0
		Banking	IX_Banking	Internal	1 of 1	0
		ScheduleActivity	IX_ScheduleActivity	Internal	2 of 2	0
		SPR	PK_SPR_154330CB	Internal	5 of 5	0
			IX_SPR	Internal	1 of 1	0
		ToDo	PK_ToDo_30190CAF	Internal	1 of 1	0

SQL defrag managerTM All Monitored Objects

Automated vs. Non-Automated Counts

Total Objects: Total # Automated: 2, Total Indexes: 14 of 14

Server	Database	Table	Index	Schedule	Automated	Non-Automated
BJ	SQLdefragRepository	ActivityLog	IX_ActivityLog	Internal	14 of 14	0
		AnalysisState	IX_AnalysisState	Internal	7 of 7	0
		Banking	IX_Banking	Internal	1 of 1	0
		ScheduleActivity	IX_ScheduleActivity	Internal	2 of 2	0
		SPR	PK_SPR_154330CB	Internal	5 of 5	0
			IX_SPR	Internal	1 of 1	0
		ToDo	PK_ToDo_30190CAF	Internal	1 of 1	0

SQL defrag managerTM Page Density Change Over Time by Server & Database

Server Database: 1:SDM_DemoOne
Minimum Page Count: 25

Change in Page Density Over Time

Legend: City, CityEmployees, CityCustomers, CityOrders, RegionID, LastName, PK_EmployeeTerritories, CompanyID, OrderOrder_Details, ProductID, ProductSub_Product, PK_Customer, EmployeeID, ProductOrder_Details, ShipperID, ShipperOrders, ShipperOrder, CategoryProduct, EmployeeOrder, ProductID, SupplierProduct, PK_Territory

SQL defrag managerTM Tables without Clustered Indexes

Heap Table Count vs. Clustered Table Count

Legend: Clustered Tables, Heap Tables

Server	Database	Table	# Tables
BJ	AdventureWorks	DatabaseLog	12
		ProductProductPhoto	2
	ReportServer	ReportSchedule	1
	SDM_DemoOne	EmployeeTerritories	3
		Region	1
		Territories	1
BJR	ReportServer	ReportSchedule	1
	SDM_DemoOne	EmployeeTerritories	3
		Region	1
		Territories	1

SQL defrag managerTM Top 10 Fragmented Clustered Indexes

Servers: 1, R
Minimum Page Count: 25

Index Fragmentation Levels

Legend: PK_TESTRefTab4226, PK_Order_Details, PK_TESTRefTab4212, PK_TESTRefTab4211, PK_TESTRefTab4220, PK_TESTRefTab4224, PK_TESTRefTab4216, PK_TESTRefTab4203, PK_TESTRefTab4207, PK_TESTRefTab4206

Server	Database	Table	Index	Frag	# Pages
BJR					

SQL Server fragmentation explained

WITH AN INTRODUCTION TO SQL DEFRAG MANAGERTM



Resources Reclaimed Reports: Perhaps the most important report of all?

SQL defrag manager reports on all resources reclaimed during defragmentation and will translate this into memory, disk, I/O, CPU, and backups – and assign hard costs to each. You assign costs specific to your environment so that you have a concrete ROI value for defragmentation that you can provide management. Perhaps if you save your company \$50,000 they might consider that \$10,000 raise?

