Fragmentation of datafiles in SQL Server

Can queries lose performance with Windows file fragmentation?

Among several doubts that are discussed in the day-to-day of the DBA is the need (or not) to perform defragmentation of the datafiles of the SQL Server databases by the Operating System. After all, is the structure of these files in Windows affected by the fragmentation of the Windows file recording? Is it necessary to perform periodic defragmentation on the disks? Does this type of fragmentation affect performance within the database?

This article will address these issues, conceptualizing how Windows stores information on disk with its NTFS file system and seeing what "responsibility" the operating system has over the information saved when it comes to SQL Server datafiles.

We will also analyze how the indexes behave before and after defragmenting the datafile, checking if the index improves its fragmentation and we will conceptualize internal fragmentation and external fragmentation.

We will analyze what effect the fragmentation causes in a simple query within the database, as well as, we will make a comparison between the fragmented datafile and a non-fragmented datafile. Within these situations, we will discuss some configurations in SQL Server that are very important to avoid disk fragmentation as much as possible in order to minimize any slowness due to this problem.

File Systems on SQL Server and NTFS

NTFS (New Technology File System) was developed by Microsoft in the 1980s for Windows NT, inheriting its features in IBM's HPFS. It manages volumes within the server's disks independently and whatever is on the volume are files or free space. And sses files free and spaces are recorded (divided) into clusters. The size of cluster can be 512 bytes to 64 K B ytes space (4 KB is the default size).

And what would be the ideal NTFS allocation unit (cluster) size for SQL Server? The SQL Server stores data rows in 8 KB pages. In general, eight of these 8 KB pages are read and written to the disc at once, and this is known as an extension. Doing a simple math, you may find that a good measure is for the cluster to be 64KB in size.

Best practices indicate that you should make sure that the volume used for SQL Server has been formatted with a block of minimum size of 64KB. These blocks are known as " file allocation units ". In some cases, such as large data storage servers, a size of 512KB (8 x 8 extensions) can also be used.

The smallest data storage unit in SQL Server is the page. The disk space allocated to a data file (*.mdf or *.ndf) in a database is logically divided into sequentially numbered pages and continues. Disk I/O operations are performed at the page level, that is, SQL Server reads or writes entire data pages.

Extensions are a collection of eight physically allocated and sequential pages that are used to manage the pages effectively. All pages are stored in extensions.

Pages

Because SQL Server has its page size to 8 KB, that means SQL Server databases have 128 pages per megabyte (1024KB/8KB). Each page begins with a 96-byte header used to store system information about the page. This information includes the page number, the page type, the amount of free space on the page, and the allocation unit ID of the object that owns the page. This type of structure is found in the data files (*.mdf). The log files (*.ldf) do not contain pages, they contain a series of records of the database data movement stored within that log file.

The data lines are placed in series on the page, starting immediately after the header. A row offset table starts at the bottom of the page, and each row offset table contains an entry for each row on the page. Each entry records the distance between the first byte of the line and the beginning of the page. The entries in the line offset table are in reverse sequence of the sequence of lines on the page.

The Figure 1 shows how this architecture is:

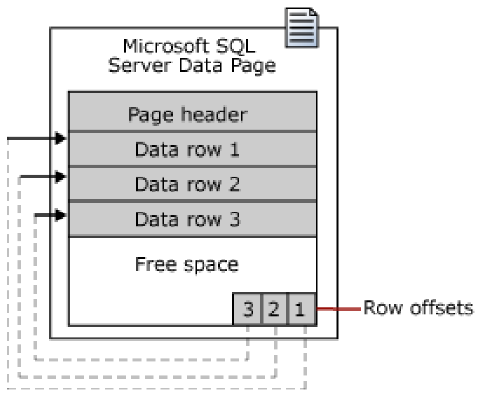


Figure 1. Structure of the data page in SQL Server

Large line support

The lines cannot be moved from one page to another, however, some parts of the line can be moved from the line page so that the line can be much longer. The maximum amount of data contained in one line of a page is of 8KB. However, this does not include stored data of the Text/Image type. This restriction is found for tables that contain varchar, nvarchar, varbinary or sql_variant columns. When the total row size of all fixed and variable columns in a table exceeds the 8,060 byte limitation, SQL Server will automatically move one or more variable length columns to the pages in the ROW_OVERFLOW_DATA allocation unit, starting with the column with the greatest width. When a column is moved to a page in the allocation unit ROW_OVERFLOW_DATA, a 24-byte pointer is maintained on the original page of the allocation unit IN_ROW_DATA to know its orientation. If a subsequent operation reduces the line size, SQL Server will return the page size to the original.

Extensions

The extension is the basic unit in which the space is controlled. An extension has eight physically contiguous pages, which returns 64 KB of space. This means that SQL Server databases have 16 extensions per megabyte (1024KB/64KB). To make space allocation useful, SQL Server does not allocate entire extensions to tables with small amounts of data. With this rule, SQL Ser ver has a uniform and mixed extension to manage this type of allocation. In uniform extents that belong to a single object, all eight pages in the extent can be used only by the object, and hile and mixed xtensões can be shared by up to eight objects. Each of the eight pages of the extension can belong to a different object. A new table or index is usually pages that are allocated mixed extensions. When the table or index grows to eight pages, it is switched to use uniform extensions for subsequent allocations. If an index is created on an existing table that has enough rows to generate eight pages in the index, all allocations to the index are in uniform lengths. That is, depending on the size of the allocation of the extensions they will be continuous (large allocations of data) or mixed (as soon as the object is created).

Following the Figure 2 which illustrates this behavior:

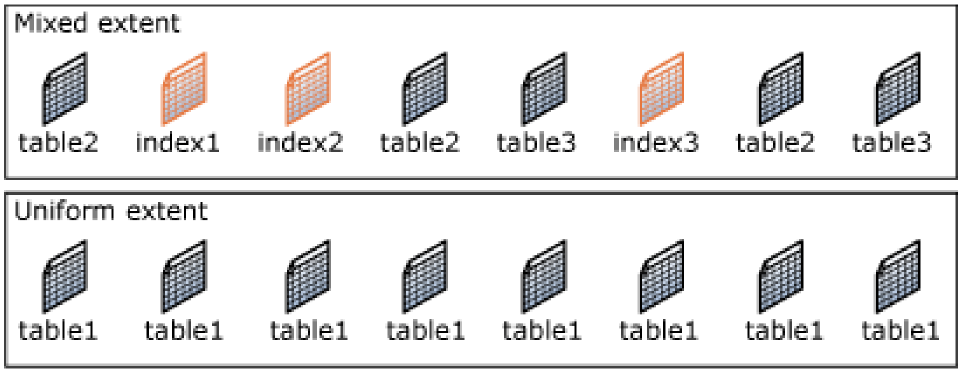


Figure 2. Structure of the two types of extensions

Datafile fragmentation

Fragmentation occurs on a volume over time as you save, change, or delete files. Changes that you save to a file are usually stored in a different volume location than the original file. This does not change the location where the file appears in Windows, only the location where the pieces of information that make up the file are stored on the volume itself. Over time, both the file and the volume itself become fragmented, and the computer slows down by having to look in different locations to open a single file. Disk Defragmenter is a tool that reorganizes data on the volume and gathers fragmented data so that the computer can work more efficiently. In this version of Windows, Disk Defragmenter runs on a schedule so you don't have to remember to run it, although you can still run it manually or change the schedule used. Disk defragmentation is the process of consolidating fragmented data onto a volume (such as a hard drive or storage device) to make it work more efficiently. Disk fragmentation can occur in two ways with SQL Server:

❑ Internal fragmentation occurs when data gets old and has been the subject of many insertions, updates and deletions. This will be seen in Fragmentation within the datafile.

❑ File fragmentation occurs when a file is created and the file system does not have enough continuous disk space to create the file in a single fragment. You end up with a single file spread across multiple file fragments on the surface of the disc. An important point to consider is that SQL Server files do not become more fragmented once they have been created (with the exception of a point that we will see later). If files are created when there is not enough contiguous free space, they are created in multiple fragments. If the disk is defragmented (and the OS has enough space to fully defragment all files) right after they are created, they are no longer fragmented and will never fragment again. The ideal scenario is that you have dedicated disks for your SQL Server files, with the correct size for each datafile (*.mdf), and disable automatic growth (autogrow option). In this situation, you start with "clean" discs, create non-fragmented files and they will stay that way forever. So you just need to deal with internal (index) fragmentation. The most common scenario, however, is that automatic growth is enabled and is invoked to continue growing the database files as needed. Autogrow as a resource is great for helping to avoid the emergency situation (SQL Server Error 161 Mg) of running out of space in a file and having to manually add more space. However, it should only be activated as a last resort (when you have numerous databases to manage, for example), since it can create fragmented files on the disk, which will affect performance. If you do not have dedicated units for the databases and you are starting from scratch the best method is:

1. Install the Operating System;
2. Defragment and the disk;
3. Install all applications (including SQL Server);
4. Defragment the disc again;
5. Create the SQL Server datafiles with their maximum size that is expected to be achieved.
6. Stop the SQL Server service, check the datafiles for fragmentation and defrag again if necessary.
7. Re-enable the SQL Server service and disable the autogrow option or, at least, make the growth increments large enough for this to happen rarely.
8. Routinely defragment the disk to preserve contiguous free space in case you need to add more SQL database files in the future. In most cases, the operating system's Disk Defragmenter does a great job and is all you need, but there are a number of third-party tools that can give you more control, speed and centralized management for coordinating defragmentation across multiple servers, in addition to power defragment datafiles without stopping the SQL Server service. If you have a large database on a dedicated drive and you are concerned about fragmenting the datafiles in that database, a tip is to copy the files off the drive and then copy them back to the previous drive. This will remove any fragmentation. This action requires sufficient free disk space on both sides and also stops the SQL Server service. However, it is a "quick" way to defragment SQL Server datafiles.

Note: Plan the growth of your databases and adjust their sizes accordingly, in addition to creating a verification routine, comparing the used space
and available space of each database, thus preventing the autogrow option from being used.

How does the autogrow option affect performance?

When you run a command on SQL Server that fetches more space than what the data log has available, the transaction log autogrow option is enabled, increasing the log size in the proportion that has been configured. In this case, the time the transaction will take will be longer, as it will have to, in addition to recording the information, increase the log file on the disk.

If the growth configured in the log is very large and there is some other factor that makes the process take too long, the query may fail, giving a timeout error. This same type of error can affect data files (*.mdf).

In addition, whenever this new growth of the datafile file occurs, the other transactions must wait for the completion of this growth, directly affecting all sessions running in the instance.

If you combine the autogrow and autoshrink options (automatically reducing free database space), you may needlessly generate processing spikes on the server. Therefore, we advise you to disable the autoshrink option in any database. Leaving it activated in combination with the autogrow, you can generate, for example, a transaction that causes the data log to grow 10 MB until the time of recording (generating I/O for the increase of the datafile). After a temp the the autoshrink option starts automatic amen you and reduces the transaction log to 10 MB, for the transaction has already been transferred to the datafile data (more I/O desnecessesário). Then, you run the same transaction, which causes the transaction log to increase 10 MB again.

In this example, you are creating unnecessary disk overhead and are probably causing the log file to fragment. In addition, a higher CPU usage is necessary, as the data must be read and discarded several times to finish recording to disk.

If there are many log file augmentation operations, you may have an excessive amount of virtual log files (VLF). This can lead to performance problems in database initialization or replication, mirroring, CDC (Change Data Capture) and online operations. In addition, this can sometimes cause performance problems when modifying data.

Physical fragmentation resulting from changing the size of data or log files can have a major impact on performance. This will happen if you use automatic settings or manually increase and decrease files frequently.

If you enlarge the database by small increments or if you enlarge and then shrink it, you can cause disk fragmentation. The disk fragmentation can result in performance problems with this scenario of small growth increments affect ing all the system.

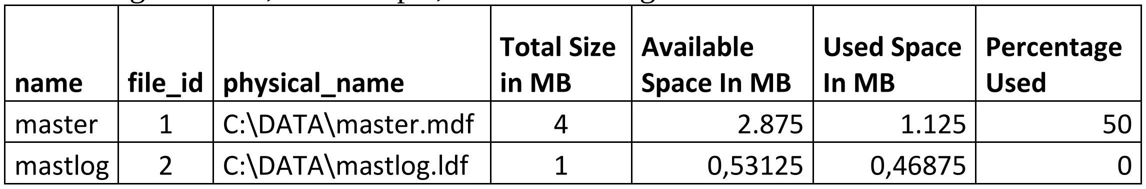
One way to prevent automatic growth is by placing databases with fixed sizes and controlling occupied spaces. This is done by disabling the autogrow option and periodically checking your datafiles to see the number of megabytes used compared to the amount of space available within the datafiles.

The query in Listing 1 shows the space used by each SQL Server datafile. With it it is possible to create a job that executes the same and shows the datafiles that are, for example, with 80% of the occupied space and send an email to the database administrator to take the necessary actions such as, increase the size of the datafile again, add a new datafile, check new storage for the data or even suggest a purge of data in the database.

Listing 1. Query to check available space

```
IF convert(varchar(20),SERVERPROPERTY('productversion')) LIKE '8%'
SELECT [name],
       fileid,
       filename,
       [size]/128 AS 'Total Size in MB',
       [size]/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0
AS 'Available Space In MB',
       CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0 AS 'Used Space
In MB',
       (100-((([size]/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS
int)/128)/([size]/128))*100)) AS 'percentage Used'
FROM sysfiles ELSE
SELECT [name],
       file_id,
       physical_name,
       [size]/128 AS 'Total Size in MB',
       [size]/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0
AS 'Available Space In MB',
       CAST(FILEPROPERTY(name, 'SpaceUsed') AS int)/128.0 AS 'Used Space
In MB',
       (100-((([size]/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS
int)/128)/([size]/128))*100)) AS 'percentage Used'
FROM sys.database_files
```

The Listing 1 returns, for example, the result of Figure 3:



name	file_id	physical_name	Total Size in MB	Available Space In MB	Used Space In MB	Percentage Used
master	1	C:\DATA\master.mdf	4	2.875	1.125	50
mastlog	2	C:\DATA\mastlog.ldf	1	0,53125	0,46875	0

Figure 3 Query result with the used percentage.

Fragmentation within the datafile

Inside the SQL Server data file, external or internal index fragmentation can occur. The f ragmentação d and index occurs over time as the data are added, changed, and deleted in a table. It is part of the c otidiano of one database and it is inevitable. When fragmentation is severe, it will affect the performance because inserts require I/O page s intensiv ely and the readings become ineffective because there is no data query performed on each page 8 KB of data recovered n the disc.

Internal fragmentation refers to data pages of 8KB that have plenty of free space (it means that they contain no more data). External fragmentation refers to adjacent data pages, which means that the data pages are not side by side on the disk. This results in increased movement in the disc head, which leaves any costly consultation. The sys.dm_db_index_physical_stats DMV (dynamic management view) will provide all the information you need about current fragmentation levels.

Indexes can only be rebuilt on tables with clustered indexes. Fragmentation cannot be removed from tables without clustered indexes (HEAP tables). Unclustered indexes can be reconstructed at the same time or separately. The best way to remedy fragmentation of content is with a command ALTER INDEX. The reorganization index is lighter, but it can not resolve data inte r calad the s and fragmentation index pages (external).

Rebuild an index is a tarefade intensive I/O that will solve amba s the fragmentaç s internal and external and can be done so online in the

Enterprise or Developer editions of SQL Server. Other editions allow q ue the same result but requires that the table be "re taken down " during the maintenance operation of the index of the same.

Regular maintenance and pro-active n the index reorganization will keep the contents in good condition and should m reduce the frequency with which they are required the reconstruction of the same. It is possible to create a script that will check the DMV mencionad the earlier and take corrective measures necessary - to reorganize or rebuild an index based on fragmentation levels identified by the DMV. This approach, together with some kind of audit trigger for activity history, is best practice and will mean that you no longer have to worry about index or table fragmentation. The m Yearly l the SQL Server nline contains a script sample to reorganize and rebuild indexes * based on the levels of fragment action. This can provide a useful starting point for customizing for your environment.

Note: Maintaining the index is essential for continuing SQL Server performance. Fragmentation occurs through daily inserts, updates and
deletions. Use the DMV and sys.dm_db_index_physical_stats index maintenance to ensure that your indexes are maintained properly..

Do I need to defrag my datafile?

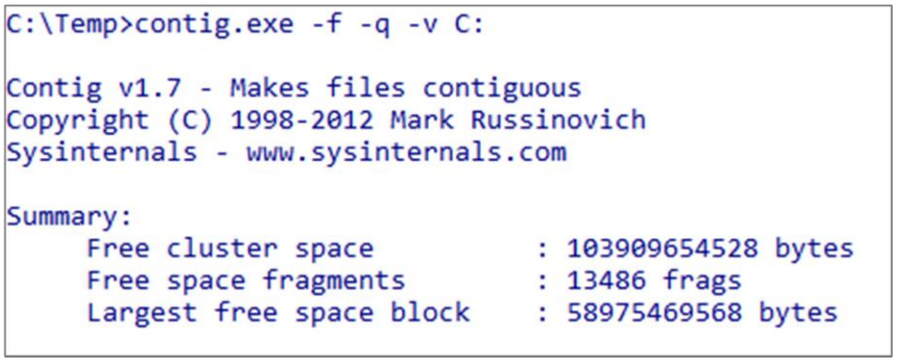
Defragmentation is cleaning work done at the file system level to reduce the constant growth of file system fragmentation. Disk fragmentation is a slow and continuous process that occurs when a file is divided into pieces to fit a volume in its available spaces. As files are constantly being written, deleted and resized, moving from one location to another etc., fragmentation ends up being a natural occurrence. When a file is spread across multiple locations, it takes more time for a disk to complete reading or writing I/O. Thus, defragmentation is necessary to obtain a better yield. For example, when SQL Server backup is performed, it needs a minimal differential area or MinDiffArea to prepare a snapshot of the file (a file with no information but the size previously allocated on the server). The problem is that you don't have to have a piece of contiguous free space without fragmentation.

Microsoft has come a long way from Windows XP/2003 to current versions in which it used to not have a defrag tool that specifically shows fragmentation on a volume or file.

In November 2012, microsoft released version 1.7 of a tool that specifically shows segmented fragmentation, even by file, so that we can have a more detailed view of how our datafiles are fragmented on the server.

So, how do I know that I need to defragment my SQL Server datafiles?

You can use the contig.exe tool to check the level of fragmentation before making the defragmentation decision. The microsoft tool called contig.exe *. For more information on how to execute execute your commands and parameters access the link in the links section. Below is na example of the output we can get in figure 3:



```
C:\Temp>contig.exe -f -q -v C:

Contig v1.7 - Makes files contiguous
Copyright (C) 1998-2012 Mark Russinovich
Sysinternals - www.sysinternals.com

Summary:
     Free cluster space          : 103909654528 bytes
     Free space fragments        : 13486 frags
     Largest free space block    : 58975469568 bytes
```

Figure 3. Checking volume C fragmentation:

From the result of this command, it is possible to check whether the SQL Server datafiles (or the entire drive volume) are defragmented. This type of analysis is important because with a high rate of "Free space fragments" can -If determine the cause of a slowdown in your instance.

It is interesting to note that it is important to create a maintenance window to perform defragmentation of SQL Server datafiles. When defragmentation is performed, the pages and clusters are transferred in a way that their organization becomes as optimized as possible, but this tends to have a cost, which is I/O. At this time of defragmentation, the processes will become slower and may cause system inactivity. And for the SQL Server datafiles to be effectively defragmented, download the SQL Server service in order to release the database files to the defragmenter.

There are third-party tools that perform defragmentation "ONLINE", that is, without downloading the SQL Server service. However, it is interesting to note that even so, we will still have a slowdown caused by the defragmentation of the files. Therefore, even if you have a defragmentation tool "ONLINE", place this process in your maintenance window so as not to cause any slow effect on the company's system.

Conclusion

We worked on this article with an analysis of the effects of fragmentation of SQL Server datafiles under Windows. We check how the structure of windows is and how it is associated in the structure of SQL Server datafiles.

We present how fragmentation can occur in SQL Server, being that it can be from Windows (from datafile), when the file is written in several “pieces” inside the hard disk so that the consultation time between the clusters ends up being much longer “Laborious” for the CPU to search for information and we also have the fragmentation of indexes (internal fragmentation in the database,) which occurs through insertions, deletions and updates of records, leaving empty spaces on the data pages, if it is necessary to rebuild the indexes so that they become more optimized. We also present a sequence of steps for when starting to install SQL Server and making the databases as fragmented as possible.

We studied how the autogrow option is harmful, and it can fragment datafiles in a way that affects the performance of queries, insertions and updates in databases. This option, combined with autoshrink can affect not only fragmentation but also how to increase the processing time, as the server needs to always consider when you need to increase or decrease Auto- en te its base.

With a script to analyze the space of the datafiles in view of the amount of available space, we can evaluate when it will be necessary to grow the datafies of the database manually, and the script can be executed from a job to be scheduled its execution each week to take the appropriate measures.

Finally, we show how to perform an analysis of the SQL Server datafile through Windows in order to find out if it is necessary to perform defragmentation on these files. The contig.exe command shows the spaces for cluster and fragmentation per block.

There are other ways to perform the analysis, as well as Windows defragmentation itself, but the interesting thing to remember is that if you use the defragmentation tool that Windows makes available, you will need to download the SQL Server services so that the datafiles are free for defragmentation. Still, there are third-party tools (paid tools) that do this job without the need to download any service that is running on Windows.

However, whatever the choice, it is important to keep in mind that this type of process demands a lot of I/O from the server, directly impacting performance. Therefore, choose a maintenance window that does not impact (or that has the least impact) possible for the company's business.

Links

* http://msdn.microsoft.com/pt-br/library/ms189858.aspx *(Reorganize and recreate indexes)*
* http://technet.microsoft.com/en-in/sysinternals/bb897428.aspx *(The Microsoft tool called contig.exe)*