Manage File Growth in Blackbaud CRM Databases

Len Wyatt
Enterprise Performance Team

Audience

- Database administrators (DBAs) who support ***Blackbaud CRM***, regardless of whether they are self-hosted or hosted by Blackbaud Service Delivery Operations (SDO)
- Blackbaud Professional Services, which advises customers on system configurations
- Developers, whether they write customizations or are members of the Blackbaud Products group. They should follow appropriate practices and understand what an optimal deployment looks like.

Introduction

A SQL Server database stores its data as files in the file system of the server, and the management of those files has a non-trivial impact on the performance of the database. This article is about managing the growth of files for ***Blackbaud CRM*** databases. Files can be set to auto-grow, which means they expand automatically when they need more space. That is an important fallback mechanism, but as discussed below, it is a sub-optimal practice, especially for log files.

It is important to keep in mind that best practices for SQL Server file management are well known and widely available on the web. While they are not unique to ***Blackaud CRM*** applications, this article summarizes recommended practices in a ***Blackbaud CRM*** context.

File Management in General

Here is an overview of the main points for managing SQL Server database files:

- You should separate data and log files from each other and isolate them from other disk usage. Because of how physical disks work, you should separate data and log files for performance and reliability reasons. Even for sites with solid-state drives, you should separate them for reliability.
- You should size files for current needs plus future growth. Most sites plan for space some number of months in advance.

- You should allow auto-growth to occur if needed but should manage it not to occur – especially for log files. Instead, you should pre-allocate space during non-prime hours. Part of optimizing file growth is to ensure that *instant file initialization* is configured.
- You should avoid shrinking database files whenever possible. Shrinking files is a performance killer and heavily fragments your indexes. Since most files will continue to grow, it is usually best to leave free space open for future growth.

Note: Sites that follow standard deployment practices from Blackbaud most likely already separate data and log files and do not shrink files. However, it is likely that file growth is managed by auto-growth. And sites also probably have multiple log files, which are rarely needed as I explain below.

What Auto-Growth Means to Users

While a data or log file is being grown (enlarged), the database is locked. Users may observe delays or even time-outs. This is why it is important to grow database files outside of prime usage hours and to enable instant file initialization for data files. (Instant file initialization cannot be used for log files).

Avoid Auto-Growth in Data Files

The best way to manage storage is to plan ahead and allocate space before storage runs low. For data files, this is not too hard. Blackbaud can give space recommendations to new customers based on the experience of similarly sized customers. Once in operation, customers can monitor their disk space usage and plan accordingly. Keep in mind that you can't just track the total size of the database to know to allocate more space to files. A ***Blackbaud CRM*** database has a number of logical filegroups mapped to different files, and any of them might be low on space. Here is a T-SQL command that can be used in SQL Server Management Studio to list each file the database uses along with the amount of space used, the amount of space that is still available, and the percentage of space that is free.

```
select
	FILEID,
	FILE_SIZE_MB = size/128,
	SPACE_USED_MB = fileproperty(name,'SpaceUsed')/128,
	FREE_SPACE_MB = (size-fileproperty(name,'SpaceUsed'))/128,
	PCT_FREE = cast((size-fileproperty(name,'SpaceUsed'))*100.0/size as int),
	AUTO_GROWTH = case
		when growth = 0 then 'none'
		when status & 0x100000 = 0x100000
			then cast(growth as varchar(20)) + '%'
		else cast(growth/128 as varchar(20)) + ' MB' end,
	NAME,
	FILENAME
from dbo.sysfiles
```

It is really a best practice to track this information over time. You can plan ahead if you have an idea how fast various files grow at your site. When you are out of space or low on space, it's hard to know how much more is needed without some historical information.

While it is best to allocate space to data files in advance of the need, the catastrophic effects of auto-growing *data* files (not *log* files) on demand is largely relieved through the use of instant file initialization. This is enabled by granting the **Perform Volume Maintenance Tasks** privilege to the account that SQL Server runs under.

If SQL Server does not run under an administrative account (and *not* using an administrative account is recommended for security reasons), you can grant the **Perform Volume Maintenance Tasks** privilege with the Local Security Policy tool. (Under Security Settings on the Local Security Policy screen, select Local Policies and then select User Rights Assignment. Double-click Perform volume maintenance tasks and then click Add User or Group). The privilege takes effect when SQL Server restarts. It can then use instant file initialization to grow data files.

Remember that even if you use instant file initialization and auto-growth for data files, you still want to track the rate of growth to predict when more disk space will be needed. Files cannot continue to grow if the disk is full. Windows Performance Monitor (PerfMon) is an excellent tool to track the overall growth of the data files in your database. In the object **SQLServer:Databases** (there is an instance per database), find the counter **Data File(s) Size (KB)**. Record this value over time for a history of the cumulative growth of your data files.

Avoid Auto-Growth in Temporary Data Files

Blackbaud CRM and SQL Server both use temporary table space during normal operations. This space shows up in SQL Server as the tempdb database; it is completely invisible to ***Blackaud CRM*** users. For performance reasons, it is best to put the data files for tempdb on a separate physical drive (or drives) from the data and log files. In addition, Microsoft recommends creating one data file for tempdb for each physical CPU in the server (not for each core).

As with data files, it is best to pre-allocate sufficient tempdb space to avoid auto-growth but to allow auto-growth "just in case." Blackaud's experience with SDO-hosted sites is that most have tempdb space at around 15 to 20 percent of the ***Blackbaud CRM*** database size. When you initially set the tempdb size, consider setting it to 20 percent of the database size until you have an opportunity to monitor your system.

As with data and log files, we recommend that you monitor tempdb space over time and that when it is necessary to grow, you do it outside of prime usage hours.

Avoid Auto-Growth in Log Files

The space needed for log files depends on the recovery model and the amount of activity between database backups. With the simple recovery model, log space is only needed until a given transaction is completed and written to the data drive(s). Most sites use the full recovery model because it is more secure: In the event of a drive failure, you can use log files and log backups to recover the database. The remainder of this discussion assumes that full recovery is used.

With the full recovery model, you need enough space to log all write transactions until a backup is completed. One purpose of the log is, after all, to provide a second record of all changes until a backup of the database is performed. This means that the amount of log space needed depends not only on the time between backups but also on how busy the system is. A busy season or a large business process increases the space needed. Upgrades to a new version of ***Blackbaud CRM*** can sometimes generate a lot of updates and thus grow the log file.

As you might expect from the previous paragraph, it's hard to predict log space needs. Since the consequences of growing the log are significant for end users, it's best to be generous with log file space. The good news is that you should allocate log files on the basis of physical drives anyway. Remember, nothing else should be on the same physical drive as the log file. Hence one mirrored drive (or the equivalent allocation in a SAN) suffices for most sites. Allocate most of that drive to the log file, and you are in business.

There is no reason to create multiple log files unless you physically run out of space on the log drive. In that case, a better strategy might be to back up more often. Multiple log files do not make the system run faster and could slow disaster recovery (which I hope you never need).

There is one more wrinkle when it comes to getting optimal performance at backup time. It has to do with the number of Virtual Log Files (VLFs) created. When a log file grows, multiple subdivisions of the log (VLFs) are created to allow faster backups. But too many VLFs can slow things down, and this frequently occurs when the log file is allowed to auto-grow. I just looked at a system that had 1000 VLFs, and I have read about systems with more. Without getting into all the details, here is a procedure that creates 1 GB VLFs up to the size of log file you wish. Use SQL Server Management Studio to run these commands, or perform the same operations using the menus and dialogs:

1. Back up the database. Seriously.
   ```
   -- Your statement may be more involved
   BACKUP LOG dbname TO device
   ```
2. If multiple log files exist, get rid of all but the primary log file.
   ```
   ALTER DATABASE dbname REMOVE FILE dbname_logname
   ```
3. Shrink the log file. This gets rid of as many of the old VLFs as possible.
   ```
   DBCC SHRINKFILE(dbname_log, TRUNCATEONLY)
   ```
4. Set the log file's size to 16 GB.
   ```
   ALTER DATABASE dbname
   MODIFY FILE ( NAME = dbname_log, SIZE = 16000 MB )
   ```
5. Extend the log by 16 GB to 32 GB.
   ```
   ALTER DATABASE dbname
   MODIFY FILE ( NAME = dbname_log, SIZE = 32000 MB )
   ```
6. Extend the log in 16 GB chunks until you reach the desired size.

This is a one-time procedure that should not be required again. Do it outside of regular business hours though because allocating and zeroing the pages takes some time and locks the database. In addition, I suggest another backup right after this change because truncating the log breaks the backup chain.

Things That Impact Log Usage

As mentioned previously, the log is a second record of everything that happens in the system, which allows you to recover a database even if a data drive fails. The log must be large enough to track everything that happens between backups. DBAs and developers can do some things to improve log usage:

- DBAs should be aware of when unusually large business processes run and take log backups sooner than usual.
- DBAs should consider taking a log backup after massive deletes (such as to clean up older audit data) because deleting data is a logged operation.
- DBAs and developers should be aware that when deleting ***all*** the data from a table, TRUNCATE TABLE is a minimally logged operation that barely impacts the log.
- DBAs and developers should be aware that dropping a table is minimally logged.
- Developers can provide code that allows customers to implement policies that periodically trim the audit data or other business process data. This could prevent the need for large deletes and could be done in a future version of the product or through customizations.

- Developers should use tempdb for temporary tables. Tempdb uses the simple recovery model, so log data does not persist for these tables.

Summary

By following a few key guidelines, you can have a database that performs better for your users and, most notably, does not "stall" while waiting for files to grow. In addition, by monitoring the growth of the database, you can plan ahead for storage space needs.

Further Reading

Most of the information for this article was found by simply searching the web. In particular, it comes from the very smart people at SQLskills and from MSDN articles. Here are some articles that you can read instead of this one or, even better, read for more information:

- Top Tips for Effective Database Maintenance has a section about managing data and log files.
- 8 Steps to Better Transaction Log Throughput is all about managing the transaction log file.
- Instant Initialization – What, Why and How? is just what it sounds like: an explanation of instant file initialization.
- Transaction Log VLFs – Too Many or Too Few? gives the rationale and procedure for creating virtual log files (VLFs).
- Chapter 9 of this book has an excellent discussion of database backups overall and is highly recommended background for all DBAs. Although the book is from 2004, the principles of backups remain the same.
- Why You Should Not Shrink Your Data Files explains why you should avoid shrinking database files whenever possible.

Books with similar information are also available, but these websites offer the content for free.