Files and File Groups in SQL Server

August 29, 2016 · Klaus Aschenbrenner · 14 Comments

(Be sure to checkout the FREE SQLpassion Performance Tuning Training Plan - you get a weekly email packed with all the essential knowledge you need to know about performance tuning on SQL Server.)

In today's blog post I want to talk about a very important topic in SQL Server: how SQL Server handles Files and File Groups. When you create a simple database with a **CREATE DATABASE** command, SQL Server only creates 2 files for you:

- A Data File (.mdf)
- A Transaction Log File (.ldf)

The data file itself is created within the one and only PRIMARY file group. Within the PRIMARY file group by default SQL Server stores all the data (user tables, system tables, etc.). So what is the purpose of having additional files and File Groups? Let's have a look at it.

Multiple File Groups

When you create additional File Groups for your database, you are able to store your user defined tables and indexes in them. This helps you in multiple ways:

- You keep the PRIMARY file group small.
- You can split your data across multiple File Groups (e.g. when you use **Table Partitioning** in the Enterprise Edition).
- You can perform Backup and Restore operations at the File Group level. This gives you a more granular control over your Backup and Restore strategy.
- You can run DBCC CHECKDB commands at the File Group level instead of at the database level.

In general, you should always have at least one secondary File Group where you store your own database objects. You should never store anything in the PRIMARY file group besides the system objects that SQL

Server creates for you.

Multiple Files

When you have created your own File Group you also have to place at least one file into it. In addition, you can add additional files into the File Group. This can also improve the performance of your workload, because SQL Server evenly spreads the data between all files through a so-called **Round Robin Allocation Algorithm**. The first extent of 64K is stored in the first file, the second extent of 64K is stored in the second file, the third extent of 64K is again stored in the first file (when you have 2 files in your File Group).

With that approach SQL Server can latch multiple copies of the allocation bitmap pages (PFS, GAM, SGAM) in the Buffer Pool and increase the throughput of your workload. You solve the same problem that can occur in a default configuration of **TempDb**. In addition, SQL Server also makes sure that all the files within the File Group are full at the same point in time – through a so-called **Proportional Fill Algorithm**. Therefore, it is also very important that all your files within the File Group have the same initial size and the same Auto Growth factors. Otherwise the Round Robin Allocation can't work very well.

A concrete example

Now let's have a look at a concrete example of how you can create a database with an additional File Group with multiple files within it. The following code shows you the **CREATE DATABASE** command that is necessary to accomplish this task.

```
-- Create a new database
CREATE DATABASE MultipleFileGroups ON PRIMARY
(
    -- Primary File Group
    NAME = 'MultipleFileGroups',
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DA
    SIZE = 5MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
),
-- Secondary File Group
FILEGROUP FileGroup1
(
    -- 1st file in the first secondary File Group
    NAME = 'MultipleFileGroups1',
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DA
    SIZE = 1MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
),
(
    -- 2nd file in the first secondary File Group
    NAME = 'MultipleFileGroups2',
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DA
    SIZE = 1MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
)
LOG ON
(
    -- Log File
    NAME = 'MultipleFileGroups_Log',
    FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DA
    SIZE = 5MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
)
GO
```

After the creation of the database, the question is then how to put a table or index into a specific file group? You can specify the File Group manually with the **ON** keyword as the following code shows.

```sql
CREATE TABLE Customers
(
    FirstName CHAR(50) NOT NULL,
    LastName CHAR(50) NOT NULL,
    Address CHAR(100) NOT NULL,
    ZipCode CHAR(5) NOT NULL,
    Rating INT NOT NULL,
    ModifiedDate DATETIME NOT NULL,
)
ON [FileGroup1]
GO
```

The other option is that you mark a specific File Group as the **Default File Group**. Then SQL Server automatically creates every new database object within that File Group without specifying the **ON** keyword.

```sql
-- FileGroup1 gets the default filegroup, where new database objects
-- will be created
ALTER DATABASE MultipleFileGroups MODIFY FILEGROUP FileGroup1 DEFAULT
GO
```

This is the approach that I normally recommend, because then you don't have to think anymore, when you afterwards create your database objects 😉 So now let's create a new table that is automatically stored in the **FileGroup1** File Group.

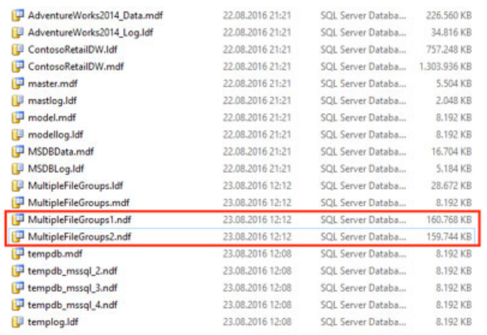
```sql
-- The table will be created in the file group "FileGroup1"
CREATE TABLE Test
(
    Filler CHAR(8000)
)
GO
```

And now we perform a simple experiment: we insert 40 000 records into the table. Every record is about 8K in size. Therefore we insert about 320 MB of data into the table. And this is where the Round Robin Allocation that I have mentioned previously kicks in: SQL Server spreads the data evenly between both files: the first file gets 160 MB of data, and the second file gets also 160 MB of data.

```sql
-- Insert 40.000 records, results in about 312MB data (40.000 x 8KB / 1024 = 31
-- They are distributed in a round-robin fashion between the files in the file
-- Each file will get about 160MB
DECLARE @i INT = 1
WHILE (@i <= 40000)
BEGIN
    INSERT INTO Test VALUES
    (
        REPLICATE('x', 8000)
    )

    SET @i += 1
END
GO
```

Then when you look into Windows Explorer you can see that both files have the same size:



AdventureWorks2014_Data.mdf	22.08.2016 21:21	SQL Server Databa...	226.560 KB
AdventureWorks2014_Log.ldf	22.08.2016 21:21	SQL Server Databa...	34.816 KB
ContosoRetailDW.ldf	22.08.2016 21:21	SQL Server Databa...	757.248 KB
ContosoRetailDW.mdf	22.08.2016 21:21	SQL Server Databa...	1.303.936 KB
master.mdf	22.08.2016 21:21	SQL Server Databa...	5.504 KB
mastlog.ldf	22.08.2016 21:21	SQL Server Databa...	2.048 KB
model.mdf	22.08.2016 21:21	SQL Server Databa...	8.192 KB
modellog.ldf	22.08.2016 21:21	SQL Server Databa...	8.192 KB
MSDBData.mdf	22.08.2016 21:21	SQL Server Databa...	16.704 KB
MSDBLog.ldf	22.08.2016 21:21	SQL Server Databa...	5.184 KB
MultipleFileGroups.ldf	23.08.2016 12:12	SQL Server Databa...	28.672 KB
MultipleFileGroups.mdf	23.08.2016 12:12	SQL Server Databa...	8.192 KB
MultipleFileGroups1.ndf	23.08.2016 12:12	SQL Server Databa...	160.768 KB
MultipleFileGroups2.ndf	23.08.2016 12:12	SQL Server Databa...	159.744 KB
tempdb.mdf	23.08.2016 12:08	SQL Server Databa...	8.192 KB
tempdb_mssql_2.ndf	23.08.2016 12:08	SQL Server Databa...	8.192 KB
tempdb_mssql_3.ndf	23.08.2016 12:08	SQL Server Databa...	8.192 KB
tempdb_mssql_4.ndf	23.08.2016 12:08	SQL Server Databa...	8.192 KB
templog.ldf	23.08.2016 12:08	SQL Server Databa...	8.192 KB

When you put those file onto different physical drives, you can even access them simultaneously. That's the power of having multiple files within a File Group.

Summary

In today's blog post I have shown you how multiple File Groups and multiple Files within a File Group can help you to make your database more manageable and how multiple files within a File Group are used through the Proportional Fill Algorithm.

Please feel free to leave a comment if you are already using a customized physical database design as described here.

Like or share to get the source code.

Tweet

Thanks for your time,

-Klaus

SQLSERVER

14 COMMENTS

Taiba
08/29/2016

Thank you , very clear explanation

Reply

Darko
08/29/2016

Very good. It is necessary to fix just one small inaccuracy in the code.
Create table on file group "FileGroup1", instead of "MyCustomFileGroup".

Reply

Klaus Aschenbrenner
08/30/2016

I've fixed that – thank you 🙂

Reply

Tibor
08/30/2016

Hi Klaus,

Nice article, one question though. I keep hearing that we should not store user objects on PRIMARY and we should have a separate FG for that. So what is the real benefit of this, since this adds a bit of complexity too. It seems we just move everything into another FG/file what needs to handled as well.

Thanks,
Tibor

Reply

Klaus Aschenbrenner
08/30/2016

Hello Tibor,

One reason is the following: in the Enterprise Edition your database will be online as soon as the PRIMARY file group and your transaction log is online. The smaller your PRIMARY file group is, the faster your database will be online.

Thanks,

-Klaus

Reply

Amit Kumar
08/30/2016

It's very helpful.

Reply

Klaas Vandenberghe
08/31/2016

Hi Klaus

Thank you for another great insight.
Do we still benefit from multiple data files with a SAN, and on a SQL Server in Azure?
Also, what's the procedure when we inherit a database with 400+ tables on Primary or on 1 file in FG1, and we want to move to multiple files?

Reply

Klaus Aschenbrenner
09/01/2016

Hello Klaas,

Thanks for your comment.
You will always benefit from multiple data files, because then SQL Server is able to latch multiple system pages in parallel in the Buffer Pool.
For migrating tables into a different file group, you have to rebuild the indexes into the new file group.

Thanks,

-Klaus

Reply

Kevin
09/08/2016

I've used a secondary filegroup (and made it default) on all databases I've created for a while now. However, I'd always only created one physical data file in that filegroup.

If I add a second data file to this file group, will it begin using the Proportional Fill Algorithm?

Reply

Klaus Aschenbrenner
09/08/2016

Hello Kevin,

Thanks for your comment.
If you add a fresh new file (only one), then almost all allocations are happening in the newly added file.
In that case it would be better to add a few more additional files, because then you have the Round-Robin Allocation between the new added files.

Thanks,

-Klaus

Reply

Mötz Jensen
09/27/2016

Hi,

I have a quick question regarding your answer to Kevin.

When adding multiple new files, would you create multiple file groups and keep only one file per group or would you just have 2 file groups (primary) and then a second on, adding only files the second one?

Or would you divide the files across the 2 file groups, so they contain the same number of files?

Thanks,
Mötz

Reply

Klaus Aschenbrenner
09/27/2016

Hello Mötz,

Thanks for your comment.
I suggest to have at least one additional file group with multiple files within it.
How many file you should add to this file group depends on the number of CPU cores you have, and if you encounter Latch Contention problems on system pages.

Thanks,

-Klaus

Reply

Girish
01/10/2019

Nice article, how do I do this in existing database?

I have added multiple data files but how to I move data from single MDF file to this other data files to make it even?

Reply

Daniel Adeniji
02/24/2019

Klaus Aschenbrenner,

Nice one.

One of the most in-depth blog posts I have read.

It is good as it provides needed context and affirming feedback as we evaluate adding complexity.

Much needed science in the consideration for adding additional filegroups and files.

Good participation in the comment section, as well.

God Blessings and pay back for placing such a treasure in the public space.

Daniel Adeniji

Reply

IT`S YOUR TURN

Your email address will not be published. Required fields are marked *

Your comment ... *

Your Name *

Your E-Mail *

Your Website

SEND

SQL First Aid Kit - Signup

Be sure to checkout the FREE SQLpassion Performance Tuning Training Plan – you get a weekly email packed with all the essential knowledge you need to know about performance tuning on SQL Server.

By confirming your subscription, we will store your email address and send you additionally emails about our upcoming trainings and about our published blog postings.

Your E-Mail

YES, I WANT IT