**Additional Data Files and Proportional Fill**

**By** [**David Fundakowski**](http://www.sqlservercentral.com/Authors/Articles/David_Fundakowski/1929342/)**, 2016/07/25**

**FROM:** [**http://www.sqlservercentral.com/articles/Filegroups/143571/**](http://www.sqlservercentral.com/articles/Filegroups/143571/)

Author: The T-SQL I provided did not have code in the SELECT statement for the PercentUsed column present in the SSMS screenshot. I added that to both the block before 1-5 and the block before 4-4

**Introduction**

Additional files can be added to a filegroup configured with fixed-size data files to allow more data to be stored in a in the database.  If data is added while files are approaching max capacity, Extended Events can give insight as to how SQL Server spreads data across multiple files.

**Setup**

For this example, I have a blank database with a fixed 15mb database file on a server with AdventureWorks2014 .  The T-SQL code is below for setting up the db along with the SSMS walkthrough.

CREATE DATABASE [AdventureWorks2014\_Expand]

ON PRIMARY

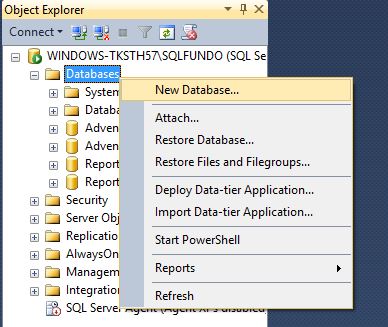
( NAME = N'AdventureWorks2014\_Expand', FILENAME = N'C:\Program Files\Microsoft SQL Server\AdventureWorks2014\_Expand\_Fixed1.mdf' , SIZE = 15360KB , MAXSIZE = UNLIMITED, FILEGROWTH = 0)

LOG ON

( NAME = N'AdventureWorks2014\_Expand\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\AdventureWorks2014\_Expand\_log.ldf' , SIZE = 10176KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

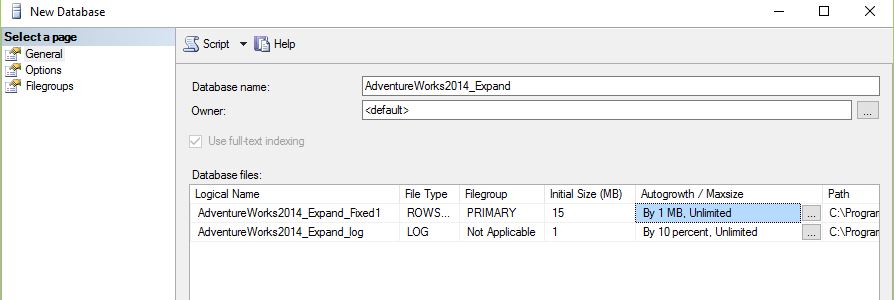
GO

Using SSMS to create a new database, start by right-clicking the Databases folder in the object explorer and select 'New Database...'  to get to the new database setup prompt



**Figure 1-1 Creating a new database using SSMS**

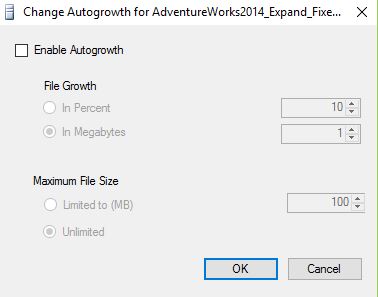
In the New Database setup, name the Database File, set the Initial Size at 15 MB and click the elipsis after the AutoGrowth/Maxsize cell to reach the prompt where Autogrowth can be disabled.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

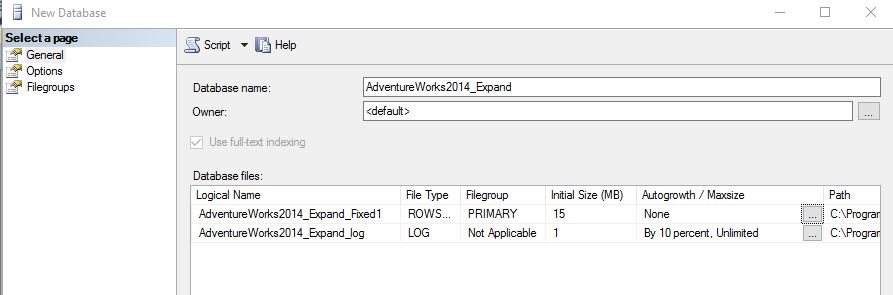
**Figure 1-2: New database dialog - General page**

In Figure 1-3, click the checkbox next to Enable Autogrowth to set the file to a fixed size that will not grow.  Select OK.



**Figure 1-3: Autogrowth setttings - Autogrowth disabled**

The test Database ready is ready to deploy.  It has a custom file name, fixed file size, and autogrowth disabled.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 1-4: Final database settings**

I also have a sample table to replicate a table from AdventureWorks2014.  I am going to fill this table with data from AdventureWorks2014.

USE [AdventureWorks2014\_Expand]

GO

CREATE TABLE [dbo].[PersonPhone](

[BusinessEntityID] [int] NOT NULL,

[PhoneNumber] [nvarchar](25) NOT NULL,

[PhoneNumberTypeID] [int] NOT NULL,

[ModifiedDate] [datetime] NOT NULL

) ON [PRIMARY]

GO

This table is going to be populated with data from AdventureWorks2014 until the database .mdf file (AdventureWorks2014\_Expand\_Fixed1) is about 85% full.

USE [AdventureWorks2014\_Expand]

GO

INSERT INTO PersonPhone (

BusinessEntityID

, PhoneNumber

, PhoneNumberTypeID

, ModifiedDate)

SELECT \*

FROM AdventureWorks2014.Person.PersonPhone

GO 9

The SQL Server master\_files sys table provides information about the capacity and current size of the data file after the PersonPhone insert.

SELECT

  db\_name(database\_id) as databaseName,

  name AS FixedFileName,

  physical\_name AS FixedFilePath,

  [file\_id] AS fileID,

  (size \* 8.0)/1024 AS FileSize,

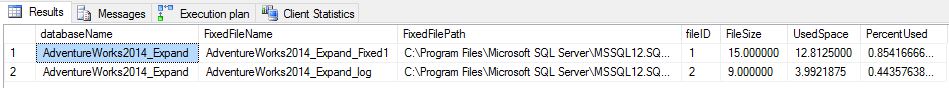
  FILEPROPERTY(name, 'spaceused') \*8/1024 AS [UsedSpace]

  , (FILEPROPERTY(name, 'spaceused') \*8/1024)/((size \* 8.0)/1024) AS PercentUsed

FROM sys.master\_files

WHERE database\_id = db\_id()

The fixed size database file on AdventureWorks2014\_Expand is 85% full after the PersonPhone insert based on the code above.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

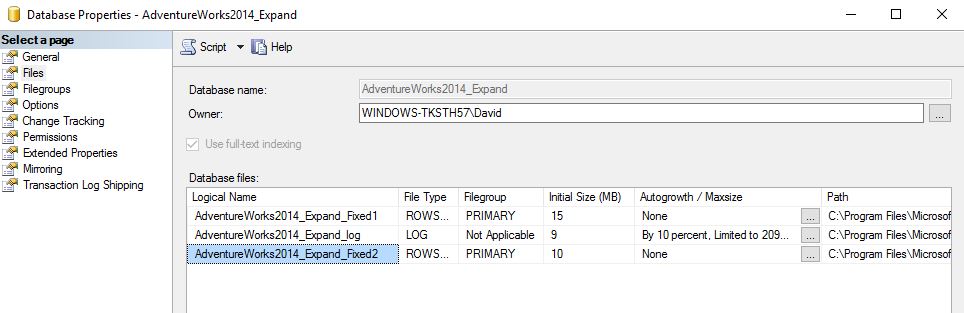
**Figure 1-5: Results of database space query**

If I try to run the insert again, I will throw an error since the data file cannot accomodate the data I want to add.

Could not allocate space for object 'dbo.PersonPhone' in database 'AdventureWorks2014\_Expand' because the 'PRIMARY' filegroup is full. Create disk space by deleting unneeded files, dropping objects in the filegroup, adding additional files to the filegroup, or setting autogrowth on for existing files in the filegroup.

**Adding a Second Data File**

Another fixed-size data file can be added to AdventureWorks2014\_Expand to allow more data insertions. Right-clicking on the AdventureWorks2014\_Expand database, selecting Properties, and clicking the Files page gets me to the prompt for adding a new file. Click Add and configure the new database file the same as our original file (Fig 1-1 through 1-3).  The new file in this case has a 10mb size.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 2-1: Adding a new file to the database**

The T-SQL for adding a new database file can be scripted out from the Figure 2-1 window by selecting the Script button.

ALTER DATABASE [AdventureWorks2014\_Expand]

ADD FILE ( NAME = N'AdventureWorks2014\_Expand\_Fixed2'

, FILENAME = N'C:\Program Files\Microsoft SQL Server\AdventureWorks2014\_Expand\_Fixed2.ndf'

, SIZE = 10240KB

, FILEGROWTH = 0)

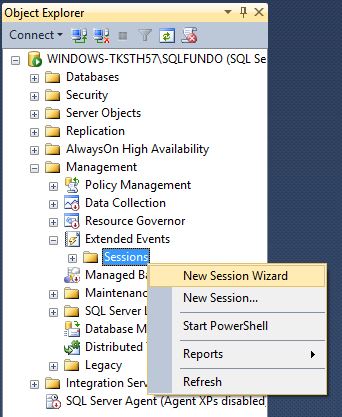
TO FILEGROUP [PRIMARY]

GO

**Setting up an Extended Events Session to Observe Proportional Fill**

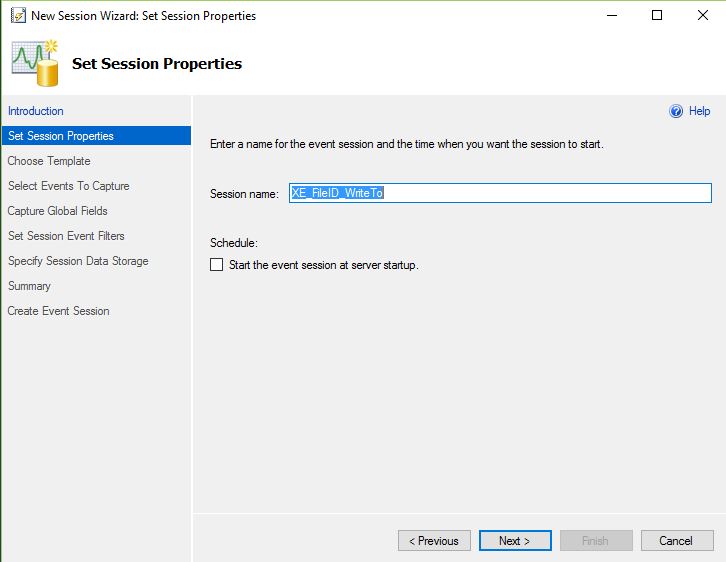
I now have around 10mb more space to add data to AdventureWorks2014\_Expand.  Before I run another insert, I want to see how SQL Server adds data when there are two fixed-size files, one almost full and one empty.

In Figure 3-1 we can set up a new Extended Events session for capturing information about our database inserts can be created through a wizard under the Management folder.



**Figure 3-1: Creating a new Extended Events session**

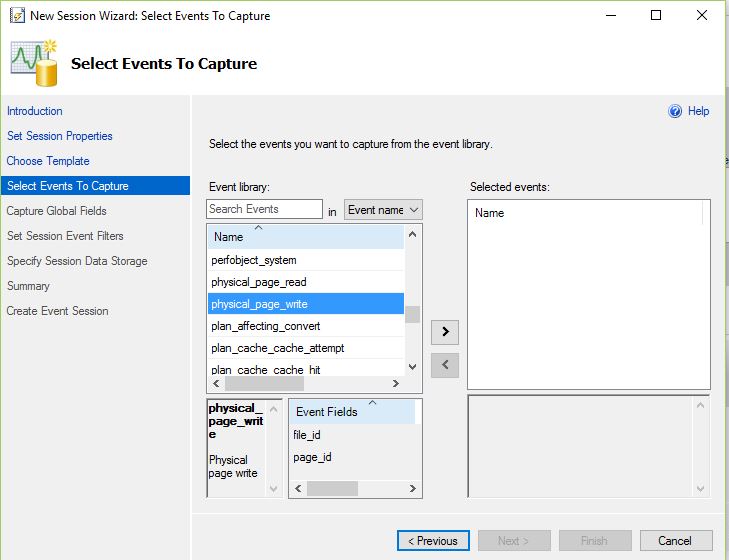
Figure 3-2 I am naming my session in a way that allows me to easilty identify it in the list of sessions under Extended Events in the Object Explorer.  I am also going to skip choosing a template since I only want to capture a specific event for this example.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 3-2: New EE session name**

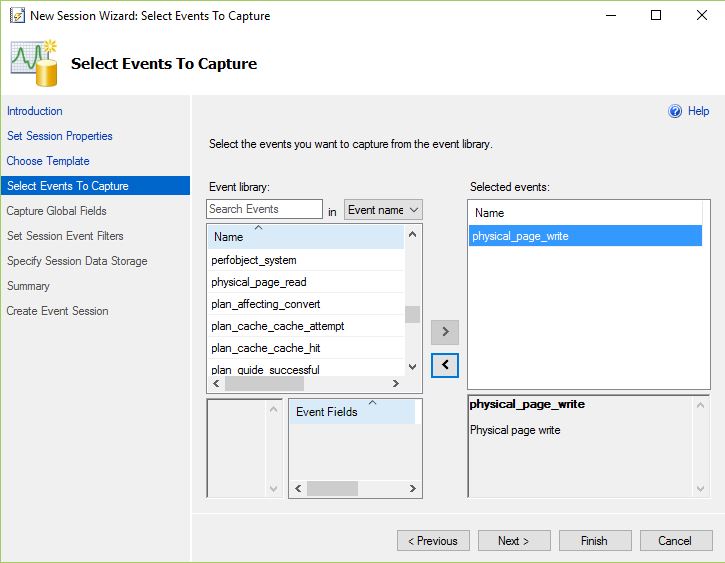
The 'Select Events to Capture' page gives me an event library that I can choose from.  There is a brief description below the library list along with a list of what event fields are included with the event from the library.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 3-3: Select the events to capture**

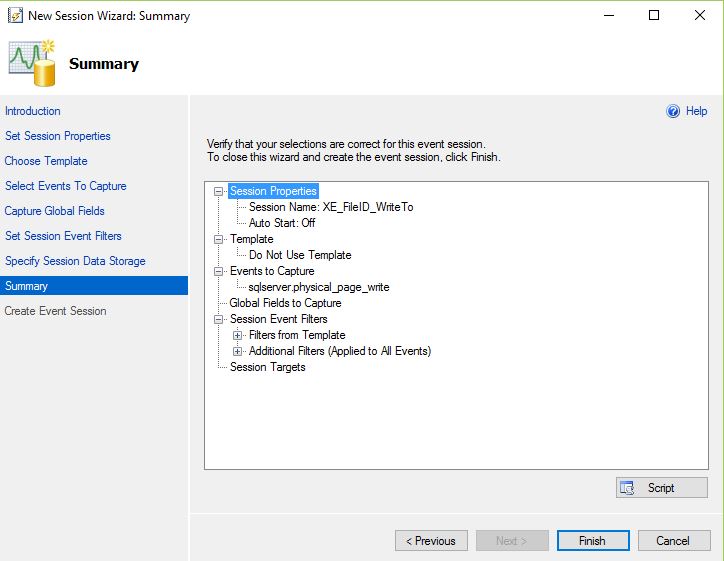
For this example, I am going to use physical\_page\_write as my event and move it to Selected Events by double-clicking or highlighting the event and clicking the right facing arrow.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 3-4: Selecting the physical\_page\_write event**

I am not setting any global fields, session filter, or data storage at this time.  On the summary page in Figure 3-5, there is a button to script out the whole Extended Events session if I want.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 3-5: Summary of the Extended Event session**

The T-SQL scripted out from the Summary page of the EE session:

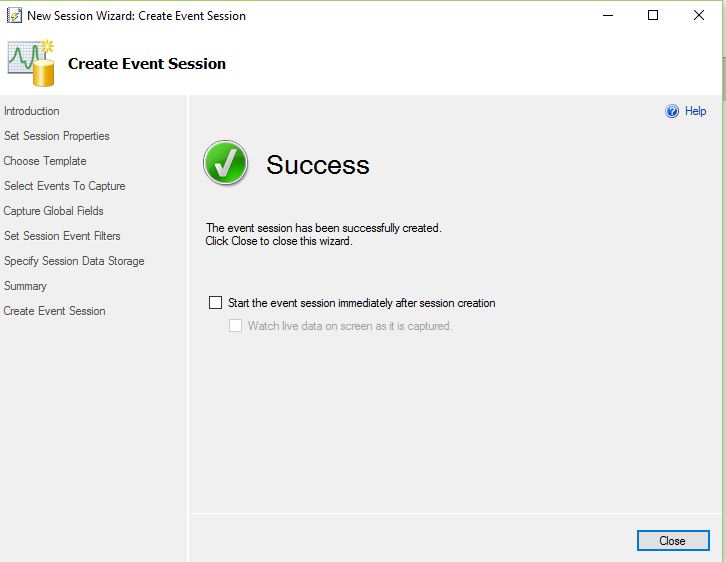
CREATE EVENT SESSION [XE\_FileID\_WriteTo] ON SERVER

ADD EVENT sqlserver.physical\_page\_write

WITH (STARTUP\_STATE=OFF)

GO

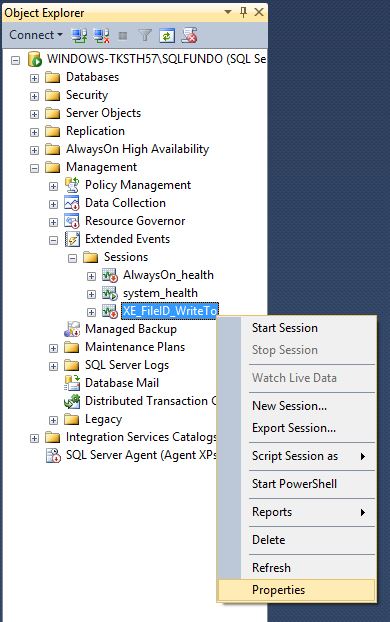
Lastly, I am not going to start the session right away.  I want to configure the output of the session first so I am leaving the box on the success page unchecked.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

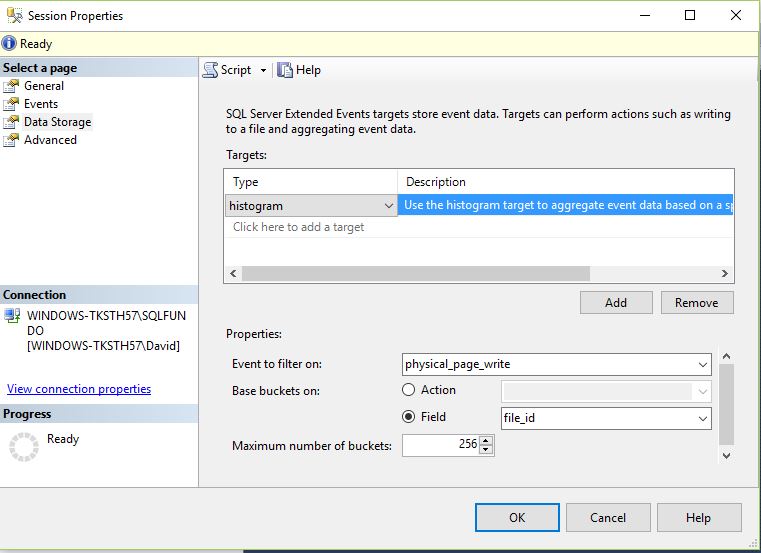
**Figure 3-6: The final step in the wizard**

I mentioned in Figure 3-6 I wanted to set up the output of the session.  The output options can be found through the session Properties.



**Figure 3-7: Get the EE session properties**

Under Data Storage, I select Histogram under Type to pick the output of my session.  Under Properties, I choose physical\_page\_write as the event to filter on, and file\_id as the field to base the buckets.  Essentially what this is doing it counting the number of physical page writes done in the session and groups them by file\_id. As with the Extended Event session wizard, there is a Script option on this page to script out the T-SQL for this session property.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 3-8: The data storage tab for the EE session**

The T-SQL for adding a histogram output to my extended events session scripted from Figure 3-8.

ALTER EVENT SESSION [XE\_FileID\_WriteTo] ON SERVER

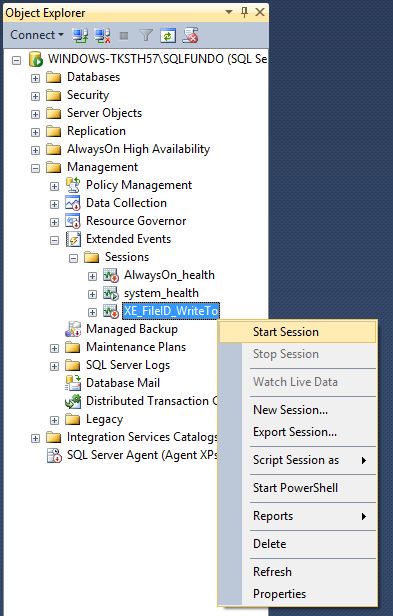
ADD TARGET package0.histogram(SET filtering\_event\_name=N'sqlserver.physical\_page\_write',source=N'file\_id',source\_type=(0))

GO

**Observing Proportional Fill through Extended Events**

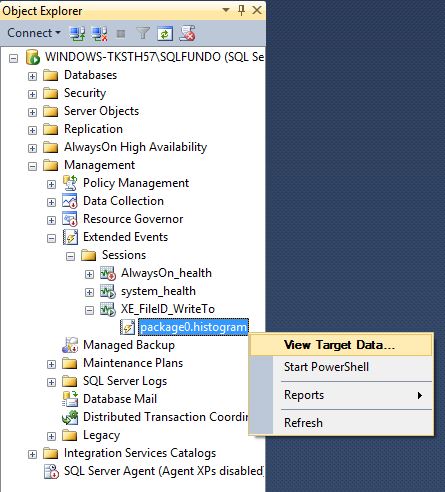
To see how SQL Server insert data and fills two data files, I am going to start my newly created session and run an insert.

To start the session to capture file writes, I right click the session and select 'Start Session' as shown in Figure 4-1.



**Figure 4-1: Starting an EE session**

To see the data from the session, I expand the session node, expand the XE\_FileID\_WriteTo session, right-click, and select View Target Data...  This will open a new window in SSMS.



**Figure 4-2: View the data for an EE session**

Next, I run my script again from earlier, but with fewer interations:

USE [AdventureWorks2014\_Expand]

GO

INSERT INTO PersonPhone (

BusinessEntityID

, PhoneNumber

, PhoneNumberTypeID

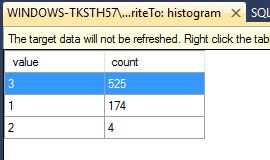
, ModifiedDate)

SELECT \*

FROM AdventureWorks2014.Person.PersonPhone

GO 5

In the window that launched from Figure 4-2, I can right click anywhere in the window and click refresh to see what was captured by the session. In this case I see the count of physical writes grouped by file ID in Figure 4-3.



**Figure 4-3: Target data window for the EE session**

The session output shows that SQL Server allocated more writes to the newest data file, file ID 3, and less to the original .mdf file, file ID 1. I can run my script from earlier to confirm this.

SELECT

  db\_name(database\_id) as databaseName,

  name AS FixedFileName,

  physical\_name AS FixedFilePath,

  [file\_id] AS fileID,

  (size \* 8.0)/1024 AS FileSize,

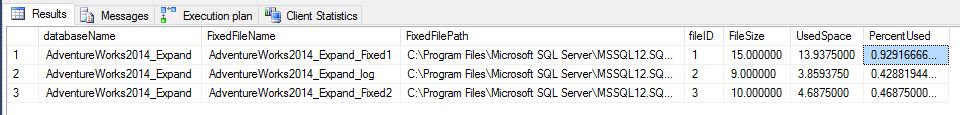
  FILEPROPERTY(name, 'spaceused') \*8/1024 AS [UsedSpace]

  , (FILEPROPERTY(name, 'spaceused') \*8/1024)/((size \* 8.0)/1024) AS PercentUsed

FROM sys.master\_files

WHERE database\_id = db\_id()

The results confirm that SQL Server did not completely fill the first data file before moving on to the next file.  It instead did the majority of the writes to the new data file (Fixed2, fileID 3) and did a smaller amount to the .mdf file.



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**Figure 4-4: Results from the database query**