**[Automating DBCC DBTABLE – obtaining the disk Sector Size](http://www.sqlservercentral.com/blogs/discussionofsqlserver/2017/03/22/automating-dbcc-dbtable-obtaining-the-disk-sector-size/)**

Posted on 22 March 2017

FROM: <http://www.sqlservercentral.com/blogs/discussionofsqlserver/2017/03/22/automating-dbcc-dbtable-obtaining-the-disk-sector-size/?utm_source=SSC&utm_medium=pubemail>

I was recently reading [this msdn article](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2dzLm1zZG4uY29tL2Ivc3Fsam91cm5leS9hcmNoaXZlLzIwMTIvMDcvMjgvYW4taW4tZGVwdGgtbG9vay1hdC1naG9zdC1yZWNvcmRzLWluLXNxbC1zZXJ2ZXIuYXNweA%3D%3D&feed-stats-url-post-id=3100) on Ghost Records, and it mentioned that you could get the number of ghost records on a page with DBCC DBTABLE… and it also mentioned that you need to be sure that you enable Trace Flag 3604 in order to see the results. So, two things immediately jumped out at me. First, I wanted to look at this to see where the ghost records were located. Secondly, I’ve just written a few articles ([here](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOC9wZXJzaXN0aW5nLWRiY2Mtb3V0cHV0LWRhdGEv&feed-stats-url-post-id=3100), [here](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOS9hdXRvbWF0aW5nLWRiY2MtcGFnZS8%3D&feed-stats-url-post-id=3100), [here](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOS9hdXRvbWF0aW5nLWRiY2MtcGFnZS1wYXJ0LTItZGV0ZXJtaW5pbmctb2JqZWN0LW5hbWVzLW9uLXBhZ2VzLw%3D%3D&feed-stats-url-post-id=3100) and [here](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8xMC9hdXRvbWF0aW5nLWRiY2MtcGFnZS1wYXJ0LTMtaXMtYS1jbHVzdGVyZWQtaW5kZXgtcGh5c2ljYWxseS1zb3J0ZWQtb24tZGlzay8%3D&feed-stats-url-post-id=3100)) where I’ve been able to use the “WITH TABLERESULTS” option on the DBCC command to avoid using this trace flag and to provide automation for the process, and I wanted to see if that would work here also.

The good news is that “WITH TABLERESULTS” does indeed work with DBCC DBTABLE. The bad news is that I could not find the ghost record count in the results.

When I was looking for this information, I noted that the results meta-data are identical to the way DBCC PAGE has its output, so this means that the automation processes already developed will work for them. And as I was looking through the results, looking for a ghost record counter, I noticed two interesting fields:

| **Field** | **Value** |
| --- | --- |
| m\_FormattedSectorSize | 4096 |
| m\_ActualSectorSize | 512 |

Hmm, this is showing me the disk Sector Size of each database file. After checking things on a few different systems, it looks like the m\_ActualSectorSize is what the sector size is for the disk that the database file is currently on, and the m\_FormattedSectorSize appears to be the sector size for when the database was created – and it is copied from the model database, so it appears to be what the disk was like when Microsoft created the model database.

Since it’s a best practice to have the disk sector size (also known as the allocation unit size or block size) set to 64kb (see [this white paper](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL21zZG4ubWljcm9zb2Z0LmNvbS9lbi11cy9saWJyYXJ5L2RkNzU4ODE0JTI4dj1zcWwuMTAwJTI5LmFzcHg%3D&feed-stats-url-post-id=3100)), I decided to programmatically get this information. After digging through the Object and ParentObject columns, this script to get the current allocation using size (Sector Size) for each drive was developed:

USE master;

GO

IF OBJECT\_ID('tempdb.dbo.#DBTABLE') IS NOT NULL DROP TABLE #DBTABLE

CREATE TABLE #DBTABLE (

ParentObject VARCHAR(255),

Object VARCHAR(255),

Field VARCHAR(255),

Value VARCHAR(255));

INSERT INTO #DBTABLE

EXECUTE ('DBCC DBTABLE WITH TABLERESULTS');

WITH cte1 AS

(

-- get the objects for the dbt\_dbid. Distinct to return only one per database

SELECT DISTINCT Object

FROM #DBTABLE

WHERE Field = 'dbt\_dbid'

), cte2 AS

(

-- get the objects related to the dbt\_dbid for the m\_Startup% field

-- SQL 2005/2008/2008R2 - looking for m\_StartupState

-- SQL 2012+ - Looking for m\_StartupPhase

-- So use m\_Startup%

SELECT DISTINCT t1.Object

FROM #DBTABLE t1

JOIN cte1 ON cte1.Object = t1.ParentObject

WHERE t1.Field LIKE 'm\_Startup%'

), cte3 AS

(

-- get the filepath and sector size for each file

SELECT fcb\_filepath = MAX(CASE WHEN Field = 'fcb\_filepath' THEN Value ELSE NULL END),

m\_ActualSectorSize = MAX(CASE WHEN Field = 'm\_ActualSectorSize' THEN Value ELSE NULL END)

FROM #DBTABLE t1

JOIN cte2 ON cte2.Object = t1.ParentObject

WHERE t1.Field IN ('fcb\_filepath', 'm\_ActualSectorSize')

GROUP BY cte2.Object, t1.Object

)

-- and now get the distinct list of drives and their sector sizes

SELECT DISTINCT Drive,

m\_ActualSectorSize,

is\_64kb = CASE WHEN m\_ActualSectorSize % 65535 = 0 THEN 1 ELSE 0 END

FROM cte3

CROSS APPLY (SELECT Drive = UPPER(LEFT(fcb\_filepath, 2))) ca

ORDER BY Drive;

And here we have yet another way for how a process can be automated by using “WITH TABLERESULTS” on a DBCC command. I think that this one is a particularly good one to show the possibilities – to get this information you have to hit multiple parts of the DBCC results, and repeat it for each file in each database. Doing this by using the 3604 trace flag, finding the appropriate piece and then proceeding on to the piece would be very time consuming to do manually.

Finally, a quick note here: there are better ways of getting the disk sector size – since you can get it with WMI calls, you can get it with PowerShell (or even dos), and there are also command line utilities that will also get you this information. This is just a way to do it from within SQL Server. Note also that this only gets the drives that contain database files on this SQL Server instance – if you are looking for other drives, then this won’t work for you.

Take a look at the other fields that are available in DBTABLE – you just might find another item that you’d like to be able to retrieve.

**Previous related posts:**

[Persisting DBCC Output](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOC9wZXJzaXN0aW5nLWRiY2Mtb3V0cHV0LWRhdGEv&feed-stats-url-post-id=3100)  
[Automating DBCC Page](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOS9hdXRvbWF0aW5nLWRiY2MtcGFnZS8%3D&feed-stats-url-post-id=3100)  
[Automating DBCC Page, Part 2](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8wOS9hdXRvbWF0aW5nLWRiY2MtcGFnZS1wYXJ0LTItZGV0ZXJtaW5pbmctb2JqZWN0LW5hbWVzLW9uLXBhZ2VzLw%3D%3D&feed-stats-url-post-id=3100)  
[Automating DBCC Page, Part 3](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL2Jsb2cud2F5bmVzaGVmZmllbGQuY29tL3dheW5lL2FyY2hpdmUvMjAxNi8xMC9hdXRvbWF0aW5nLWRiY2MtcGFnZS1wYXJ0LTMtaXMtYS1jbHVzdGVyZWQtaW5kZXgtcGh5c2ljYWxseS1zb3J0ZWQtb24tZGlzay8%3D&feed-stats-url-post-id=3100)

***This post is re-published from*** [***my original post on SQL Solutions Group***](http://blog.waynesheffield.com/wayne/?feed-stats-url=aHR0cDovL3d3dy5zcWxzb2x1dGlvbnNncm91cC5jb20vb2J0YWluaW5nLXRoZS1kaXNrLXNlY3Rvci1zaXplLw%3D%3D&feed-stats-url-post-id=3100)***.***

http://blog.waynesheffield.com/wayne/?feed-stats-post-id=3100

The post [Automating DBCC DBTABLE – obtaining the disk Sector Size](http://blog.waynesheffield.com/wayne/archive/2017/03/automating-dbcc-dbtable-disk-sector-size/) appeared first on [Wayne Sheffield](http://blog.waynesheffield.com/wayne).