DBCC FreeProcCache

Being day 12 of the DBCC Command month at SteveStedman.com, today's featured DBCC Command is DBCC FREEPROCCACHE.

Description:

DBCC FREEPROCCACHE is used for purge all of the parsed query plans out of memory. This is commonly used in development environments, but not as common in a production environment.

Use in a development environment is common, for instance when you are working on performance tuning, or parameterization of queries. You can clear the procedure cache with DBCC FreeProcCache, run the program or web page that may be using the database, then see what is in the procedure cache. This can be useful in finding queries that may need to be parameterized. Another way to use would be to find out what queries are being run by some program. To do this you would start by working with a database that is not being used by others, clearing the procedure cache with DBCC FreeProcCache, then run the program you are trying to figure out, then look at what is in the cache, again this is something that could be done in a development or test environment, but I wouldn't recommend doing it in production.

Use in a production environment should be rare, this is one of the common things to try when SQL Server is having difficulty. If you are are the point that SQL Server is extremely slow to respond and you have been unable to find the cause, one thing to try is to free the procedure cache with DBCC FreeProcCache and see if that fixes the problem.

DBCC FreeProcCache Syntax:

Example:

The following example is from a development envioronment using the AdventureWorks2012 Database.

First we connect to AdventureWorks2012 and see what is in the cache.

Editor | Results | Messages

	size_in_byt...	text
1	180224	CREATE PROCEDURE sp_jobhistory_row_limiter @job_id UNI...
2	40960	CREATE PROCEDURE [dbo].[sp_syspolicy_check_membershi...
3	303104	CREATE PROCEDURE [dbo].[sp_syspolicy_purge_history] @inc...
4	73728	CREATE PROCEDURE sp_verify_job_time @time INT, ...
5	40960	CREATE PROCEDURE sp_verify_job_date @date INT, ...
6	98304	CREATE PROCEDURE sp_verify_jobproc_caller @job_id ...
7	507904	CREATE PROCEDURE sp_sqlagent_log_jobhistory @job_id ...
8	147456	CREATE FUNCTION fn_syspolicy_is_automation_enabled() RE...
9	229376	CREATE PROCEDURE sp_verify_job_identifiers @name_of_n...
10	262144	CREATE PROCEDURE sp_help_jobstep @job_id UNIQUEID...
11	40960	CREATE FUNCTION dbo.SQLAGENT_SUSER_SNAME(@user...
12	753664	CREATE PROCEDURE sp_sqlagent_has_server_access @log...
13	24576	CREATE FUNCTION agent_datetime(@date int, @time int) RE...

Here we see that there is plenty in the cache. Next we cleare the cache with DBCC FreeProcCache and take another look at what is in the cache.

After running DBCC FreeProcCache you can see that there is nothing left in the cache.

Editor | Results | Messages

	size_in_bytes	text

When the very next query is run, it will need to be reparsed rather than using an already parsed query in the cache. This will take a bit longer than if there was already a parsed plan to run. Lets run 3 queries, then take a look at the cache.

Notice the GO Statement between each query. This tells SSMS to run each query as a separate batch. Without the GO statement the 3 queries would have been parsed as a single batch.

Editor | Results | Messages

	FirstName	MiddleName	LastName
1	John	NULL	Arthur
2	John	NULL	Berry
3	John	NULL	Emory

	FirstName	MiddleName	LastName
1	Mary	NULL	Bishop
2	Mary	NULL	Alexander
3	Mary	NULL	Peterson
4	Mary	NULL	Turner
5	Mary	NULL	Phillips

	FirstName	MiddleName	LastName

Here we see the results from the three queries. The first two returned results, and the third one had no rows in the result set. Now lets take a look at the cache

Below we see a total of 20 items in the cache now. the top item highlighted in blue is the query we used to see what was in the cache, the second block highlighted in red contains the 3 queries from above, and the third rest of them are queries being run by SQL or other supporting queries. For instance line 13 is the dm_exec_sql_text which is called from the query above that checks the plan.

Editor | Results | Messages

	size_in_byt...	text
1	40960	SELECT size_in_bytes, text FROM sys.dm_exec_cached_pl
2	40960	SELECT FirstName, MiddleName, LastName FROM person.P...
3	40960	SELECT FirstName, MiddleName, LastName FROM person.P...
4	40960	SELECT FirstName, MiddleName, LastName FROM person.P...
5	483328	WITH RingBuffer AS (SELECT CAST(replace(dorb.record, '...
6	114688	SELECT 6, cast(st.text as nvarchar(4000)), s.loginame,...
7	16384	SET XACT_ABORT OFF
8	32768	() select table_id, item_guid, oplsn_fseqno, oplsn_bOffset, opls...
9	32768	() select table_id, item_guid, oplsn_fseqno, oplsn_bOffset, opls...
10	32768	() select table_id, item_guid, oplsn_fseqno, oplsn_bOffset, opls...
11	32768	() select table_id, item_guid, oplsn_fseqno, oplsn_bOffset, opls...
12	32768	() select table_id, item_guid, oplsn_fseqno, oplsn_bOffset, opls...
13	24576	CREATE FUNCTION sys.dm_exec_sql_text(@handle varbinary(...
14	24576	create view sys.dm_exec_cached_plans as select * from Open...
15	49152	CREATE VIEW sys.dm_os_sys_info AS SELECT cpu_ticks, ...
16	16384	CREATE VIEW sys.dm_os_ring_buffers AS SELECT * FROM ...
17	24576	CREATE FUNCTION sys.dm_exec_sql_text(@handle varbinary(...
18	98304	CREATE VIEW sys.dm_exec_requests AS SELECT session...
19	65536	CREATE VIEW sys.sysprocesses AS SELECT spid, kpid, b...
20	8192	sp_reset_connection

If we wanted to filter this down to just the queries we had written you could do it by adding a WHERE text LIKE ... clause to the query as shown here.

Here we see that only the three queries show up, and that each of those three takes up about 40K of memory on SQL Server.

Editor | Results | Messages

	size_in_byt...	text
1	40960	SELECT FirstName, MiddleName, LastName FROM p...
2	40960	SELECT FirstName, MiddleName, LastName FROM p...
3	40960	SELECT FirstName, MiddleName, LastName FROM p...

So why are there three copies of the same SELECT statement, this seems a bit wasteful. Indeed it does, for more information see an earlier post called [How much Procedure Cache memory does one SQL Statement take up]? There are ways to correct this.

Using DBCC FreeProcCache For A Specific Plan Handle

If you wanted to clear just a single plan handle, and not all the plan handles, you could use the optional parameter called @handle.

To get the plan handle, we start by modifying our earlier query to show us what is in the plan cache. You could leave out the WHERE clause on your own system, but I have it here to show us just the three queries in question from above.

Editor | Results | Messages

	size_in_byt...	text	plan_handle
1	40960	SELECT FirstNa...	0x0600070034E32B37F0460D700600000001000
2	40960	SELECT FirstNa...	0x0600070076C2951550BFDA7D060000000100
3	40960	SELECT FirstNa...	0x060007000100FF3310B8DA7D060000000100

Here we see the same three query plans from earlier, with an addition column called plan_handle. To free a single plan handle, we would just copy the numeric plan handle, and add that into the DBCC FreeProcCache query.

Editor | Results | Messages

	size_in_byt...	text	plan_handle
1	40960	SELECT FirstName, Middl...	0x0600070034E32B37F0460D7006000
2	40960	SELECT FirstName, Middl...	0x0600070076C2951550BFDA7D06000

Where we only see 2 of the three original queries in the plan cache.

How big is my Procedure Cache?

You can run the following query to check the size of your procedure cache.

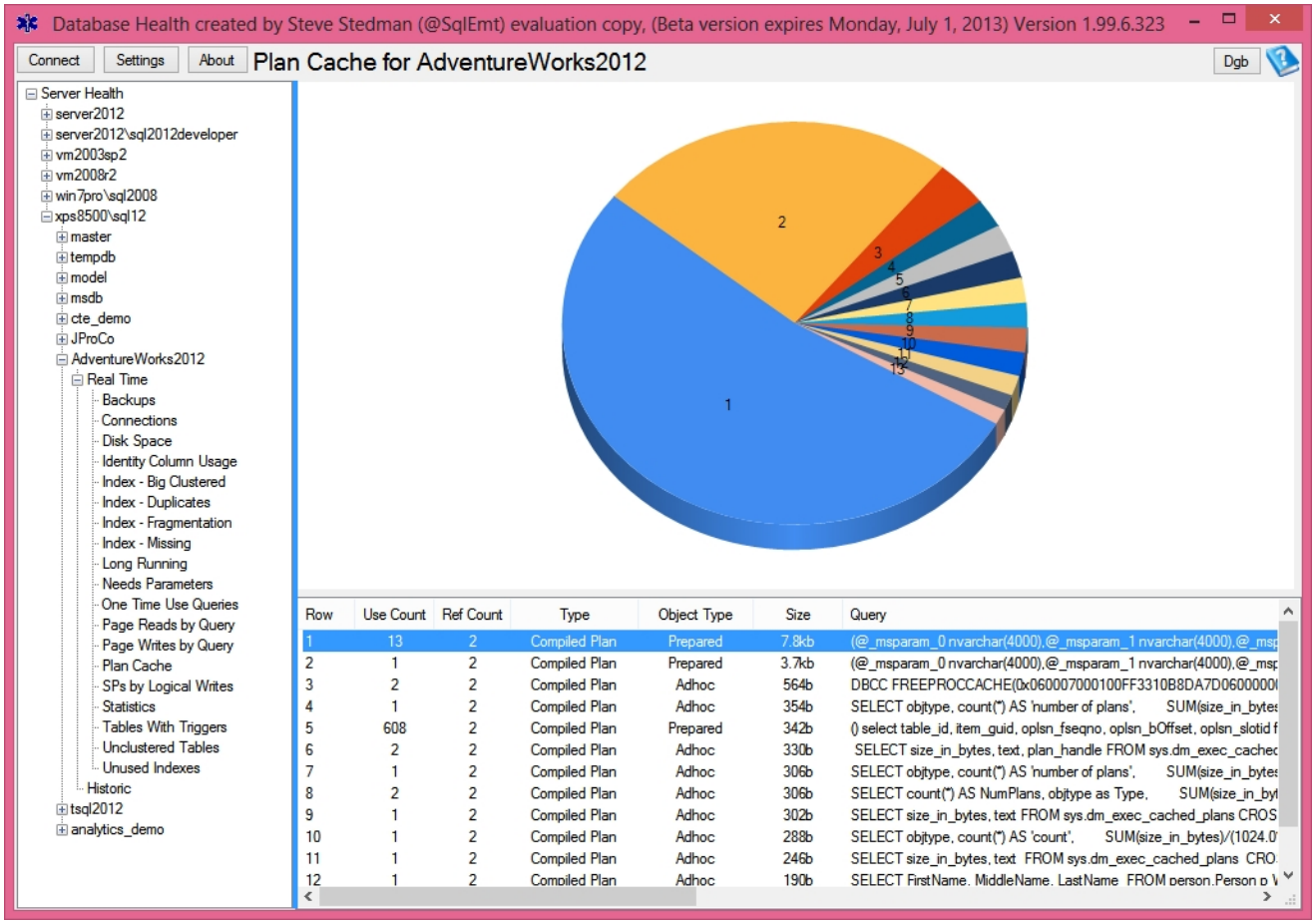
Which produces the following results on my test server.

Editor | Results | Messages

	NumPlans	Type	size_in_mb
1	17	Adhoc	1.531250000000
2	10	Prepared	3.007812500000
3	1	Proc	0.007812500000
4	33	View	2.703125000000

Database Health Reports and the Plan Cache

You can also view the plan cache using the Database Health Reports application as shown here.



Notes:

For more information see [TSQL Wiki DBCC freeproccache](#).

DBCC Command month at [SteveStedman.com](#) is almost as much fun as eating jello.