Temp DB - Troubleshooting Documentation (SQL Server)

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:27 AM PST

Contents

IMPORTANT NOTE

The content below is outdated and does not apply to Azure SQL Database. It is a 1:1 copy from a SQL Server 2008 blog article which has not been updated to current troubleshooting features.

Current, detailed tempdb troubleshooting steps for Azure SQL Database are available at Temp DB - Resolve tempdb related errors and exceptions.

.

.

.

.

.

.

.

.

.

Original content

.

.

TempDB Monitoring and Troubleshooting: Out of Space

> From https://blogs.msdn.microsoft.com/sqlserverstorageengine/2009/01/11/tempdb-monitoring-and-troubleshooting-out-of-space/

One of the key challenges in TempDB is that it is a common resource for all applications running on an instance and any misbehaving application or rouge user command can take up all the space in TempDB bringing down other applications with it. In my discussions with customer during various conferences, I often hear of following suggestions

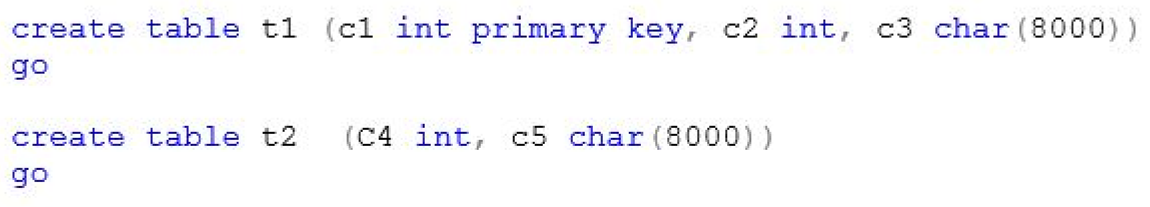
1. Provide a way to control how much TempDB space can be allocated by various applications on an instance of SQL Server. Clearly, this will provide a very good way to isolate applications from misbehaving ones. In this case, if an application exceeds its limit, it may come to a stop even if there was space on TempDB. To address this, the SQL Server can possibly provide some alternatives like to allow space allocation if the

TempDB is not in-use by other aplications and then do force deallocations when pressure from other applications mount.

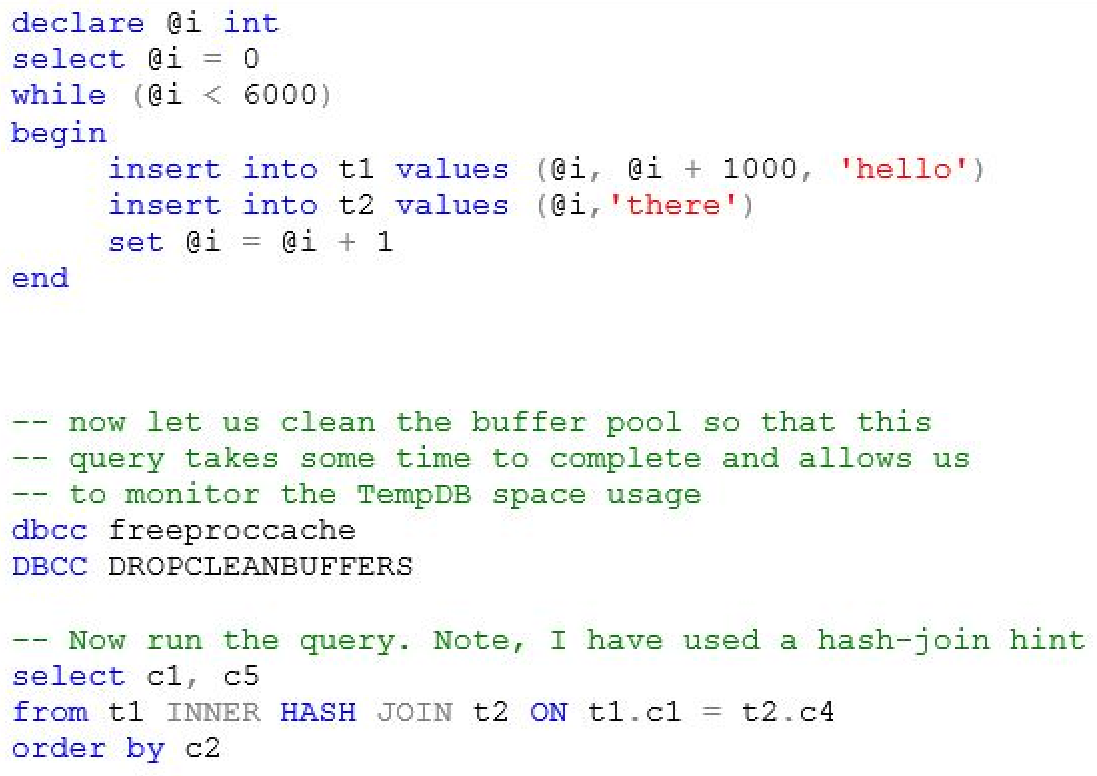
2. Provide multiple TempDBs and then assign different TempDBs to different applications. In my opinionm if SQL Server could do (1) well, then this may not be as use useful.

These suggestions are well taken but unfortunately SQL Server does not support this functionality today. So you wonder what you can do. Well, the SQL Server exposes a way using DMVs to identify TempDB space allocations by currently executing queries. If you identify that the TempDB space is running awfully low, you can use this new way to identify currently executing requests. May be some user ran an adhoc query that took significant space in TempDB. You, as an administrator, can then make the decision if you need to kill one or more of these queries to get back the space in TempDB.

Let me illustrate this with an example. I will use two large (actually not so large) tables and then join them using a hash join. You may recall that during hash join, one of the tables in hashed in memory and is backed by persistence in TempDB.



```
create table t1 (c1 int primary key, c2 int, c3 char(8000))
go

create table t2  (C4 int, c5 char(8000))
go
```



```
declare @i int
select @i = 0
while (@i < 6000)
begin
     insert into t1 values (@i, @i + 1000, 'hello')
     insert into t2 values (@i,'there')
     set @i = @i + 1
end



-- now let us clean the buffer pool so that this
-- query takes some time to complete and allows us
-- to monitor the TempDB space usage
dbcc freeproccache
DBCC DROPCLEANBUFFERS

-- Now run the query. Note, I have used a hash-join hint
select c1, c5
from t1 INNER HASH JOIN t2 ON t1.c1 = t2.c4
order by c2
```

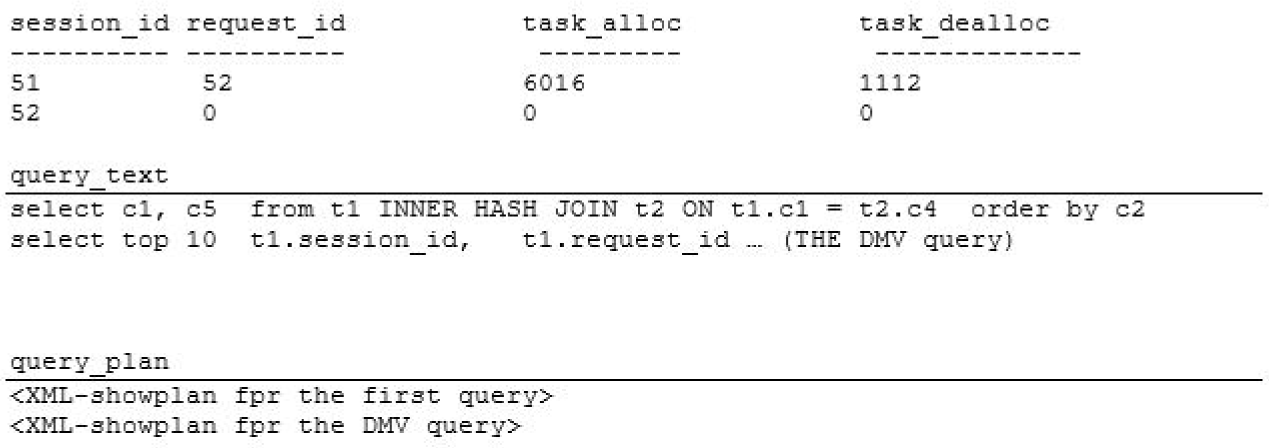
Now in another session, I will run the following DMV query

Now in another session, I will run the following DMV query

```
-- This DMV query shows currently executing tasks and
-- tempdb space usage
-- Once you have isolated the task(s) that are generating lots
-- of internal object allocations,
-- you can even find out which TSQL statement and its query plan
-- for detailed analysis

select top 10
t1.session_id,
t1.request_id,
t1.task_alloc,
     t1.task_dealloc,
    (SELECT SUBSTRING(text, t2.statement_start_offset/2 + 1,
          (CASE WHEN statement_end_offset = -1
              THEN LEN(CONVERT(nvarchar(max),text)) * 2
                   ELSE statement_end_offset
              END - t2.statement_start_offset)/2)
     FROM sys.dm_exec_sql_text(sql_handle)) AS query_text,
 (SELECT query_plan from sys.dm_exec_query_plan(t2.plan_handle))
as query_plan

from      (Select session_id, request_id,
sum(internal_objects_alloc_page_count +
user_objects_alloc_page_count) as task_alloc,
sum (internal_objects_dealloc_page_count +
user_objects_dealloc_page_count) as task_dealloc
       from sys.dm_db_task_space_usage
       group by session_id, request_id) as t1,
       sys.dm_exec_requests as t2
where t1.session_id = t2.session_id and
(t1.request_id = t2.request_id) and
      t1.session_id > 50
order by t1.task_alloc DESC
```

```
--This DMI..T query   shows currently executing tasks and
--tempdb space usage
--Once you have   isolated the task (s) that are generating lots
--of irlternal obi   ect allocations,
--you can even find   out which TSQL statement and its query plan
--for detailed   analysis

select top 1 Cl
Cl. session id,
Cl. request id,
Cl. task alloc,
t1 . task dealloc,
text, t2.statement start offset/2 +
(SELECT SUBSTRING   (1,
CASE WHEN   statement end offset -1
THEN LEN (CONVERT   (nvarchar (max) , text) ) *
2
ELSE statement end   offset
END – t2 .   statement start offset) /2)
FROM sys.dm exec   sq1 text (sql handle)) AS query text,
(SELECT query_plan   from sys.dm exec guery_plan (t2 . plan handle) )
as query_plan
f rom
Select session id,   request id,
sum (internal   objects alloc_page count
user obi acts   alloc_page count) as task alloc,
sum (internal   objects dealloc_page count
user obi acts   dealloc_page count) as task dealloc
from sys. dm db   task space usage
group by session   id, request id) as t1
sys.dm exec   requests as t2
where Cl. session
Cl. request id
Cl. session
order by Cl. task
id = t2.sess10n id
and
t2 . request id
) and
id > SO
alloc DESC
```

Here is the sample output of the query for my workload. I have simplified it by shortening the long DMV query and just put a symbolic name where XML show plan appears. This output shows that the query with hash-join is causing the most allocations in TempDB. Though in this case, we do know about the workload but you can run the above DMV query on any SQL Server without any knowlede of the workload and it can show you the top consumers (batches currently executing) of the space in TempDB. You can also take a look at the query plan to see what is causing the allocations in TempDB.



```
session_id request_id           task_alloc           task_dealloc
---------- ----------            ---------            -------------
51          52                  6016                 1112
52          0                   0                    0

query_text
select c1, c5  from t1 INNER HASH JOIN t2 ON t1.c1 = t2.c4  order by c2
select top 10  t1.session_id,   t1.request_id … (THE DMV query)


query_plan
<XML-showplan fpr the first query>
<XML-showplan fpr the DMV query>
```

Thanks

Sunil Agarwal

> From <https://blogs.msdn.microsoft.com/sqlserverstorageengine/2009/01/11/tempdb-monitoring-and-troubleshooting-out-of-space/ >

How good have you found this content?