This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. Learn more

Troubleshooting Insufficient Disk Space in tempdb

This topic provides procedures and recommendations to help you diagnose and troubleshoot problems caused by insufficient disk space in the tempdb database. Running out of disk space in **tempdb** can cause significant disruptions in the SQL Server production environment and can prohibit applications that are running from completing operations.

tempdb Space Requirements

The **tempdb** system database is a global resource that is available to all users that are connected to an instance of SQL Server. The **tempdb** database is used to store the following objects: user objects, internal objects, and version stores.

You can use the sys.dm_db_file_space_usage dynamic management view to monitor the disk space used by the user objects, internal objects, and version stores in the **tempdb** files. Additionally, to monitor the page allocation or deallocation activity in **tempdb** at the session or task level, you can use the sys.dm_db_session_space_usage and sys.dm_db_task_space_usage dynamic management views. These views can be used to identify large queries, temporary tables, or table variables that are using a large amount of **tempdb** disk space.

Diagnosing tempdb Disk Space Problems

The following table lists error messages that indicate insufficient disk space in the **tempdb** database. These errors can be found in the SQL Server error log, and may also be returned to any running application.

Error 1101 or 1105 **Is raised when** Any session must allocate space in **tempdb**.
Error 3959 **Is raised when** The version store is full. This error usually appears after a 1105 or 1101 error in the log.
Error 3967 **Is raised when** The version store is forced to shrink because **tempdb** is full.

The version store is forced to shrink because **tempdb** is full.
Error 3958 or 3966 **Is raised when** A transaction cannot find the required version record in **tempdb**.

tempdb disk space problems are also indicated when the database is set to autogrow, and the size of the database is quickly increasing.

Monitoring tempdb Disk Space

The following examples show how to determine the amount of space available in **tempdb**, and the space used by the version store and internal and user objects.

Determining the Amount of Free Space in tempdb

The following query returns the total number of free pages and total free space in megabytes (MB) available in all files in **tempdb**.

```
SELECT SUM(unallocated_extent_page_count) AS [free pages],
(SUM(unallocated_extent_page_count)*1.0/128) AS [free space in MB]
FROM sys.dm_db_file_space_usage;
```

Determining the Amount Space Used by the Version Store

The following query returns the total number of pages used by the version store and the total space in MB used by the version store in **tempdb**.

```
SELECT SUM(version_store_reserved_page_count) AS [version store pages used],
(SUM(version_store_reserved_page_count)*1.0/128) AS [version store space in MB]
FROM sys.dm_db_file_space_usage;
```

Determining the Longest Running Transaction

If the version store is using a lot of space in **tempdb**, you must determine what is the longest running transaction. Use this query to list the active transactions in order, by longest running transaction.

```
SELECT transaction_id
FROM sys.dm_tran_active_snapshot_database_transactions
ORDER BY elapsed_time_seconds DESC;
```

A long running transaction that is not related to an online index operation requires a large version store. This version store keeps all the versions generated since the transaction started. Online index build transactions can take a long time to finish, but a separate version store dedicated to online index operations is used. Therefore, these operations do not prevent the versions from other transactions from being removed. For more information, see Row Versioning Resource Usage.

Determining the Amount of Space Used by Internal Objects

The following query returns the total number of pages used by internal objects and the total space in MB used by internal objects in **tempdb**.

```
SELECT SUM(internal_object_reserved_page_count) AS [internal object pages used],
(SUM(internal_object_reserved_page_count)*1.0/128) AS [internal object space in MB]
FROM sys.dm_db_file_space_usage;
```

Determining the Amount of Space Used by User Objects

The following query returns the total number of pages used by user objects and the total space used by user objects in **tempdb**.

```
SELECT SUM(user_object_reserved_page_count) AS [user object pages used],
(SUM(user_object_reserved_page_count)*1.0/128) AS [user object space in MB]
FROM sys.dm_db_file_space_usage;
```

Determining the Total Amount of Space (Free and Used)

The following query returns the total amount of disk space used by all files in **tempdb**.

```
SELECT SUM(size)*1.0/128 AS [size in MB]
FROM tempdb.sys.database_files
```

Monitoring Space Used by Queries

One of the most common types of **tempdb** space usage problems is associated with large queries that use a large amount of space. Generally, this space is used for internal objects, such as work tables or work files. Although monitoring the space used by internal objects tells you how much space is used, it does not directly identify the query that is using that space.

The following methods help identify the queries that are using the most space in **tempdb**. The first method examines batch-level data and is less data intensive than the second method. The second method can be used to identify the specific query, temp table, or table variable that is consuming the disk space, but more data must be collected to obtain the answer.

Method 1: Batch-Level Information

If the batch request contains just a few queries, and only one of them is a complex query, this is typically enough information to know just which batch is consuming the space instead of the specific query.

To continue with this method, a SQL Server Agent Job must be set up to poll from the sys.dm_db_session_space_usage and sys.dm_db_task_space_usage dynamic management views by using a polling interval in the range of few minutes. A polling interval of three minutes is used in the following example. You must poll from both views because **sys.dm_db_session_space_usage** does not include the allocation activity of the current active task. Comparing the difference between the pages allocated at two time intervals lets you calculate how many pages are allocated in between the intervals.

The following examples provide the queries that are required for the SQL Server Agent job.

A. Obtaining the space consumed by internal objects in all currently running tasks in each session.

The following example creates the view `all_task_usage`. When queried, the view returns the total space used by internal objects in all currently running tasks in **tempdb**.

```
CREATE VIEW all_task_usage
AS
    SELECT session_id,
      SUM(internal_objects_alloc_page_count) AS task_internal_objects_alloc_page_count,
      SUM(internal_objects_dealloc_page_count) AS task_internal_objects_dealloc_page_cou
```

```
    FROM sys.dm_db_task_space_usage
    GROUP BY session_id;
GO
```

B. Obtaining the space consumed by internal objects in the current session for both running and completed tasks

The following example creates the view `all_session_usage`. When queried, the view returns the space used by all internal objects running and completed tasks in **tempdb**.

```
CREATE VIEW all_session_usage
AS
    SELECT R1.session_id,
        R1.internal_objects_alloc_page_count
        + R2.task_internal_objects_alloc_page_count AS session_internal_objects_alloc_pa
        R1.internal_objects_dealloc_page_count
        + R2.task_internal_objects_dealloc_page_count AS session_internal_objects_deallo
    FROM sys.dm_db_session_space_usage AS R1
    INNER JOIN all_task_usage AS R2 ON R1.session_id = R2.session_id;
GO
```

Assume that when these views are queried at a three-minute interval, the result sets provide the following information.

- At 5:00 P.M., session 71 allocated 100 pages and deallocated 100 pages since the start of the session.
- At 5:03 P.M., session 71 allocated 20100 pages and deallocated 100 pages since the start of the session.

When you analyze this information, you can tell that between the two measurements: The session allocated 20,000 pages for internal objects, and did not deallocate any pages. This indicates a potential problem.

Note

As the database administrator, you may decide to poll more frequently than three minutes. However, if a query runs for less than three minutes, the query probably will not consume a significant amount of space in **tempdb**.

To determine the batch that is running during that time, use SQL Server Profiler to capture the RPC:Completed and SQL:BatchCompleted event classes.

An alternative to using SQL Server Profiler is to run DBCC INPUTBUFFER once every three minutes for all the sessions, as shown in the following example.

```
DECLARE @max int;
DECLARE @i int;
SELECT @max = max (session_id)
FROM sys.dm_exec_sessions
SET @i = 51
  WHILE @i <= @max BEGIN
         IF EXISTS (SELECT session_id FROM sys.dm_exec_sessions
                    WHERE session_id=@i)
         DBCC INPUTBUFFER (@i)
         SET @i=@i+1
         END;
```

Method 2: Query-Level Information

Sometimes just looking at the input buffer or the SQL Server Profiler event **SQL:BatchCompleted** does not always tell which query is using most of the disk space in **tempdb**. The following methods can be used to find this answer, but these methods require collecting more data than the procedures defined in Method 1.

To continue with this method, set up a SQL Server Agent Job job that polls from the **sys.dm_db_task_space_usage** dynamic management view. The polling interval should be short, once a minute, as compared to Method 1. This short interval is because **sys.dm_db_task_space_usage** does not return data if the query (task) is not currently running.

In the polling query, the view defined on the **sys.dm_db_task_space_usage** dynamic management view is joined with sys.dm_exec_requests to return the **sql_handle, statement_start_offset**, **statement_end_offset**, and **plan_handle** columns.

```
CREATE VIEW all_request_usage
AS
  SELECT session_id, request_id,
      SUM(internal_objects_alloc_page_count) AS request_internal_objects_alloc_page_coun
      SUM(internal_objects_dealloc_page_count)AS request_internal_objects_dealloc_page_c
  FROM sys.dm_db_task_space_usage
  GROUP BY session_id, request_id;
GO
CREATE VIEW all_query_usage
AS
  SELECT R1.session_id, R1.request_id,
      R1.request_internal_objects_alloc_page_count, R1.request_internal_objects_dealloc_
      R2.sql_handle, R2.statement_start_offset, R2.statement_end_offset, R2.plan_handle
  FROM all_request_usage R1
  INNER JOIN sys.dm_exec_requests R2 ON R1.session_id = R2.session_id and R1.request_id
GO
```

If the query plan is in cache, you can retrieve the Transact-SQL text of the query and the query execution plan in XML showplan format at any time. To obtain the Transact-SQL text of the query that is executed, use the **sql_handle** value and the sys.dm_exec_sql_text dynamic management function. To obtain the query plan execution, use the **plan_handle** value and the sys.dm_exec_query_plan dynamic management function.

```
SELECT * FROM sys.dm_exec_sql_text(@sql_handle);
SELECT * FROM sys.dm_exec_query_plan(@plan_handle);
```

If the query plan is not in cache, you can use one of the following methods to obtain the Transact-SQL text of the query and query execution plan.

A. Using the polling method

Poll from the view `all_query_usage`, and run the following query to obtain the query text:

```
SELECT R1.sql_handle, R2.text
FROM all_query_usage AS R1
OUTER APPLY sys.dm_exec_sql_text(R1.sql_handle) AS R2;
```

Because **sql_handle** should be unique for each unique batch, you do not have to save duplicate **sql_handle** entries.

To save the plan handle and XML plan, run the following query.

```
SELECT R1.plan_handle, R2.query_plan
FROM all_query_usage AS R1
OUTER APPLY sys.dm_exec_query_plan(R1.plan_handle) AS R2;
```

B. Using SQL Server Profiler events

As an alternative to polling the **sys.dm_exec_sql_text** and **sys.dm_exec_query_plan** functions, you can use SQL Server Profiler events. There are profiler events that can be used to capture the query plan and query text that is generated. For example, Event 165 returns performance statistics for trace, SQL text, query plans, and query statistics.

Monitoring Space Used by Temp Tables and Table Variables

You can use an approach similar to polling queries for monitoring the space used by temp tables and temp variables. Applications that acquire a large amount of user data inside temp tables or temp variables can cause space use problems in **tempdb**. These tables or variables belong to the user objects. You can use the **user_objects_alloc_page_count** and **user_objects_dealloc_page_count** columns in the **sys.dm_db_session_space_usage** dynamic management view and follow the methods described earlier.

Monitoring Page Allocation and Deallocation by Session

The following table shows the results returned by the **sys.dm_db_file_space_usage**, **sys.dm_db_session_space_usage**, and **sys.dm_db_task_space_usage** dynamic management views for a specified session. Each row represents an allocation or deallocation activity in **tempdb** for a specified session. The activity is listed in the **Event** column. The remaining columns show the values that would be returned in the columns of the dynamic management views.

For this scenario, assume that the **tempdb** database starts with 872 pages in unallocated extents, and 100 pages in user-object reserved extents. The session allocates 10 pages for a user table, and then deallocates all of them. The first 8 pages are in mixed extent. The remaining 2 pages are in uniform extent.

Event Start **dm_db_file_space_usageunallocated_extent_page_count column** 872 **dm_db_file_space_usageuser_object_reserved_page_count column** 100 **dm_db_session_space_usageand dm_db_task_space_usageuser_object_alloc_page_count column** 0 **dm_db_session_space_usageand dm_db_task_space_usageuser_object_dealloc_page_count column** 0
Event Allocate page 1 from existing mixed extent **dm_db_file_space_usageunallocated_extent_page_count column** 872 **dm_db_file_space_usageuser_object_reserved_page_count column** 100

dm_db_session_space_usageand dm_db_task_space_usageuser_object_alloc_page_count column
1

dm_db_session_space_usageand dm_db_task_space_usageuser_object_dealloc_page_count column
0

Event
Allocate pages 2 to 8: consuming one new mixed extent

dm_db_file_space_usageunallocated_extent_page_count column
864

dm_db_file_space_usageuser_object_reserved_page_count column
80

dm_db_session_space_usageand dm_db_task_space_usageuser_object_alloc_page_count column
8

dm_db_session_space_usageand dm_db_task_space_usageuser_object_dealloc_page_count column
0

Event
Allocate page 9: consuming one new uniform extent

dm_db_file_space_usageunallocated_extent_page_count column
856

dm_db_file_space_usageuser_object_reserved_page_count column
108

dm_db_session_space_usageand dm_db_task_space_usageuser_object_alloc_page_count column
16

dm_db_session_space_usageand dm_db_task_space_usageuser_object_dealloc_page_count column
0

Event
Allocate page 10 from existing uniform extent

dm_db_file_space_usageunallocated_extent_page_count column
856

dm_db_file_space_usageuser_object_reserved_page_count column
108

dm_db_session_space_usageand dm_db_task_space_usageuser_object_alloc_page_count column
16

dm_db_session_space_usageand dm_db_task_space_usageuser_object_dealloc_page_count column
0

Event
Deallocate page 10 from existing uniform extent

dm_db_file_space_usageunallocated_extent_page_count column
856

dm_db_file_space_usageuser_object_reserved_page_count column