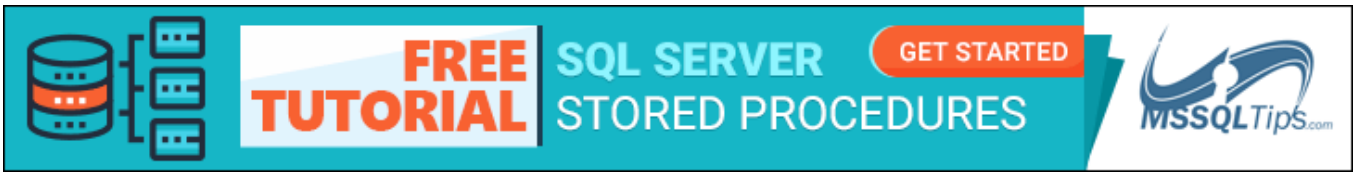
MSSQLTips Free SQL Server Conference Registration + Travel Expenses click to learn more

Understanding SQL Server Blocking



Free Webcast - Simplify SQL Server Administration

Problem

In my previous article, I provided you with a script that presented the current locks being held in the active database. While this script also provided limited information on blocking that was occurring, it did not give a fully-informative insight on that matter. I alluded to a different solution for reviewing blocking and this is the subject of this article.

For those of you who've not yet read the Understanding SQL Server Locking tip I suggest doing so as it gives a high-level view of sys.dm_tran_locks, locking, and the objects we'll be discussing in this tip.

Solution

Blocking is the logical outcome of locks being issued against various objects in a database. If a request is made against an object with a lock issued against it, then a block occurs.

For example...

Request A is attempting to insert a row in table X. Subsequently, before the Request A completes, Request B is submitted to update table X. If the row affected by Request A is also affected by Request B then a block is encountered because, among other things, the RID (row id) for the row affected by Request A has an exclusive (X) lock issued against it for the duration of the transaction involved in Request A.

This behavior will be used as the basis of observing blocking via the script this article is centered-on. The behavior is manifested via two scripts:

```
BEGIN TRAN INSERT_FOO
   INSERT INTO dbo.Categories([CategoryName], [Description])
   VALUES ('Foo', 'Description FOR Foo')
--ROLLBACK TRAN INSERT_FOO
```

Notice that I've left the transaction in a running state. This is so we can examine the blocking behavior that will be caused when we issue the subsequent update script that follows:

```
BEGIN TRAN
UPDATE dbo.Categories
SET [Description] = 'Foo'
WHERE [CategoryName] = 'Foo';
ROLLBACK TRAN
```

So how do we observe the blocking that is currently being experience as a result of these two open transactions? Well in the case of using the Dynamic Management Views (DMVs) which is the subject of this article, we query among other objects the sys.dm_trans_locks DMV as well as a few select other system objects using the following query:

```
--=============================================
--View Blocking in Current Database
--Author: Timothy Ford
--http://thesqlagentman.com
--=============================================
SELECT DTL.resource_type,
   CASE
       WHEN DTL.resource_type IN ('DATABASE', 'FILE', 'METADATA') THEN DTL.resource_type
       WHEN DTL.resource_type = 'OBJECT' THEN OBJECT_NAME(DTL.resource_associated_entity_id)
       WHEN DTL.resource_type IN ('KEY', 'PAGE', 'RID') THEN
           (
           SELECT OBJECT_NAME([object_id])
           FROM sys.partitions
           WHERE sys.partitions.hobt_id =
           DTL.resource_associated_entity_id
           )
       ELSE 'Unidentified'
   END AS requested_object_name, DTL.request_mode, DTL.request_status,
   DOWT.wait_duration_ms, DOWT.wait_type, DOWT.session_id AS [blocked_session_id],
   sp_blocked.[loginame] AS [blocked_user], DEST_blocked.[text] AS [blocked_command],
   DOWT.blocking_session_id, sp_blocking.[loginame] AS [blocking_user],
   DEST_blocking.[text] AS [blocking_command], DOWT.resource_description
FROM sys.dm_tran_locks DTL
   INNER JOIN sys.dm_os_waiting_tasks DOWT
       ON DTL.lock_owner_address = DOWT.resource_address
```

```
    INNER JOIN sys.sysprocesses sp_blocked
        ON DOWT.[session_id] = sp_blocked.[spid]
    INNER JOIN sys.sysprocesses sp_blocking
        ON DOWT.[blocking_session_id] = sp_blocking.[spid]
    CROSS APPLY sys.[dm_exec_sql_text](sp_blocked.[sql_handle]) AS DEST_blocked
    CROSS APPLY sys.[dm_exec_sql_text](sp_blocking.[sql_handle]) AS DEST_blocking
WHERE DTL.[resource_database_id] = DB_ID()
```

The script output is conveniently divided into two sections (and I've split the screen shots of the output along those lines): information pertaining to the **blocked** request and information about the **blocking** request.

	resource_type	requested_object_name	request_mode	request_status	wait_duration_ms	wait_type	blocked_session_id	blocked_user	blocked_command
1	RID	Categories	U	WAIT	708800	LCK_M_U	54	Sauron\Timothy	(@1 varchar(8000),@2 varchar(8000))UPDATE

	blocking_session_id	blocking_user	blocking_command	resource_description
1	53	Sauron\Timothy	BEGIN TRAN INSERT_FOO INSERT INTO dbo.Categori...	ridlock fileid=1 pageid=153 dbid=8 id=lock52310...

Metadata about the blocked request includes the name of the object incurring the block, the lock attempting to be granted to the blocked request (in this case an U or Update lock), the status of the request (WAIT), the type of wait being encountered and its duration, as well as the session_id, command text, and login for the blocked request. Likewise, the session_id, login_id, command text, and resource_description of the blocking transaction is provided. All items are fairly self-explanatory, with the exception of resource_description. Its value is comprised of the various IDs associated with the objects involved in the lock at various granularities.

Let's look at the full value of this column for the query we've executed:

resource_description
ridlock fileid=1 pageid=153 dbid=8 id=lock5231000 mode=X associatedObjectId=72057594038779904

The value of this field denotes that file 1 (the primary data file for this database) which is identified by the database_id (or dbid) of 8 is affected. Furthermore, the lock id is provided as is the object_id that corresponds to the object in question for the lock. In the case of an RID lock, this is the partition_id of the table in question.

If I was to rollback my INSERT_FOO transaction or commit it you'll see the state of blocking for the active database I've been working with is greatly diminished; matter-of-fact. There is none to report.

Next Steps

- More tips from the author are available via this link.
- Learn about other methods for identifying current activity in your databases at MSSQLTips.com
- Learn about CROSS APPLY in this recent tip.
- Review the previous tip in this series on locking Understanding SQL Server Locking
- The author blogs about SQL Server topics at his personal blog: thesqlagentman.com.
- Read these additional tips about Locking and Blocking

Last Update: 2010-04-06

You're invited
to our next free
SQL Server webcast!

click to learn more

Next Tip

Click Here

About the author

Tim Ford is a Senior Database Administrator and SQL Server MVP. He has been working with SQL Server since 1999.

View all my tips

Related Resources

- Identify the cause of SQL Server blocking...
- How to identify blocking in SQL Server...
- Understanding SQL Server Blocking...
- Different techniques to identify blocking in SQL S...
- More SQL Server DBA Tips...

Post a comment or let the author know this tip helped.

All comments are reviewed, so stay on subject or we may delete your comment. Note: your email address is not published. Required fields are marked with an asterisk (*).

*Name *Email Notify for updates

*** NOTE *** - If you want to include code from SQL Server Management Studio (SSMS) in your post, please copy the code from SSMS and paste the code into a text editor like NotePad before copying the code below to remove the SSMS formatting.

p

Send me SQL tips:

I'm not a robot

reCAPTCHA
Privacy - Terms

Save Comment

Get Free SQL Tips

enter business email Join

SQL Monitor
Get your free
ROI calculation
Start now

SQL Server
Performance
Tuning Tutorial

Get Free SQL
Server Tips

Today's Tip

Copyright (c) 2006-2018 Edgewood Solutions, LLC All rights reserved

Some names and products listed are the registered trademarks of their respective owners.