SQL Server Blocking

Everything You Need to Know About SQL Server Blocking

Updated: June 29, 2020

Table of Contents:

What Is SQL Server Blocking?

Blocking occurs in lock-based concurrency systems when processes require access to resources. Depending on the type of access needed, a logical "lock" is granted to a process. This lock tells competing processes to wait. The process or processes that are waiting for the lock are "blocked."

SQL Server is a lock-based concurrency system. Therefore, SQL Server blocking is by design, integral to maintaining data integrity and process concurrency, and completely normal and expected.

More on SQL Server Locking

SQL Server Lock Modes

Locks in SQL Server have a sophisticated design. The design allows for greater process concurrency because not all types of lock will block all other types of lock. Locks designate a mode that tells a process the type of access it has to the resource. The lock mode determines how restrictive the lock is to processes and, ultimately, how sensitive the process might be to causing blocks.

Common lock modes are listed in the following table. (See Microsoft's documentation for a complete listing (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms175519%28v%3dsql.105%29).)

- **Exclusive**—An exclusive lock is granted for data modifications to make sure only one process can modify data at a time
- **Schema**—A schema lock is granted when a process must be able to rely on a consistent schema during its operations
- **Shared**—A shared lock is granted for read operations and contributes minimally to blocking
- **Update**—An update lock is granted for updates and tends to be used for updates that happen in multiple steps

Intent is a modifier lock mode. You will often see a lock mode of IX or IS, for example. A lock mode modified with "I" indicates an intent lock. This is a performance optimization allowing SQL Server to evaluate potential lock compatibility issues faster. A more detailed explanation of intent locks can be found in this blog post (https://www.sqlpassion.at/archive/2016/05/16/why-do-we-need-intent-locks-in-sql-server/).

SQL Server Lock Compatibility

Lock compatibility refers to whether one lock mode can be granted in relation to another lock mode that is already granted. Lock modes that are not compatible will result in blocking.

The table below can help you determine which lock types might block other lock types. It doesn't cover all lock modes, but it does include lock modes you will encounter regularly.

	Existing granted mode					
Requested mode	**IS**	**S**	**U**	**IX**	**SIX**	**X**
Intent shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No
Intent exclusive (IX)	Yes	No	No	Yes	No	No
Shared with intent exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

(https://www.sentryone.com/hs-fs/hubfs/51adab81-54a3-0126-5637-fca6772b7ceb.png) (https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver15)

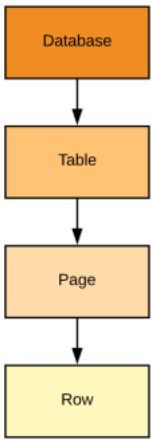
Credit: Microsoft SQL Server Documentation

Lock Escalation

Lock escalation is a mechanism used by SQL Server. It takes lots of low-level locks and moves them to fewer locks higher in the lock hierarchy. Lock escalation conserves memory used by the lock manager. It reduces memory requirements by tracking fewer locks on larger scope objects versus many locks smaller scope objects.

Lock Hierarchy

The lock hierarchy in SQL Server is fairly simple. At the highest level, we have database locks. At the lowest level, we have row locks. If you've heard of latches, put that out of your mind for the moment. Locks and latches serve a distantly similar purpose, but they are far from being the same.



(https://www.sentryone.com/hs-fs/hubfs/192993ec-3e4d-c88c-cc2b-01d6b15b2c86.png)

The lock hierarchy for SQL Server from top to bottom

SQL Server will default to low-level row or page locks. Once the lock manager has numerous locks (~5,000) on the same object, lock escalation consolidates the low-level locks to a table lock. Row locks will not escalate to page locks. Also, note that locks will escalate if they are exclusive (X) or intent exclusive (IX).

Lock escalation can be modified for a server using trace flag 1211. This disables lock escalation for the server. A better option for modifying lock escalation was provided for the ALTER TABLE statement:

```
ALTER TABLE <table_name> SET ( LOCK_ESCALATION = { AUTO | TABLE | DISABLE } )
```

Modifying lock escalation should be done cautiously and with explicit purpose. Disabling lock escalation could result in a lot of wasted memory, especially for high-throughput transactional applications.

Is SQL Server Blocking Bad?

Blocking is completely normal in SQL Server. You will typically be unaware of the many short blocks happening all the time. Sometimes, blocks take longer than expected to resolve. Blocks of longer duration can create chains, where a blocked process blocks additional processes and so on. This type of blocking scenario is problematic. So, the answer is yes and no. Blocking is expected, but it can also become a problem.

How Do I Detect Problematic Blocking?

The first step is determining what you consider to be a problematic block. Basing this on block duration is typical. If you are unsure, somewhere between 5 and 15 seconds is a good starting point. Going over 30 seconds is not recommended because the default command timeout for many client frameworks is 30 seconds. If the command times out before the block is detected, you might not know whether the block under it has impacted application users.

Querying Blocked Requests

Once you've decided on a duration filter, you can look for blocks with higher duration using SQL Server DMVs. Below is a simple query using the sys.dm_exec_requests DMV:

```
declare @durationInSeconds float = 5;

select

der.session_id,

der.blocking_session_id,

der.wait_type,

der.wait_time

from

sys.dm_exec_requests der

where

der.wait_time >= (@durationInSeconds * 1000)

and der.blocking_session_id != 0;
```

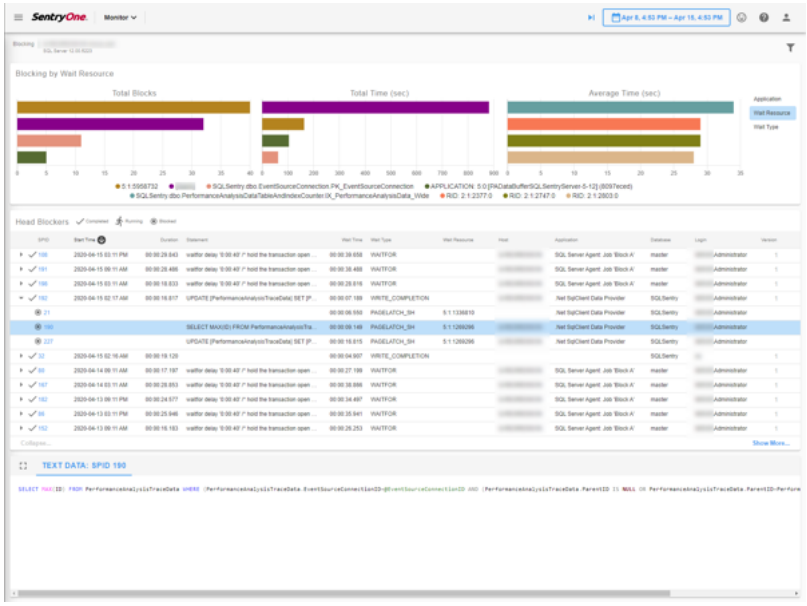
This query provides a list of blocked SPIDs, the SPID that is blocking, the current wait type, and the wait duration in MS. It filters results by a duration you set in the parameter defined at the beginning.

This query falls short in two major ways:

1. It shows a single moment in time, which won't help you understand blocks you might have missed when you weren't looking.
2. It only shows the request SPID and the blocking SPID. The root cause of the problem is at the head of the blocking chain. You'll have a hard time piecing the chain together from these results. The risk of overlooking the actual problematic process is high.

Monitoring Continuously for SQL Server Blocking

Solving the shortcomings of the DMV query would lead to continuous monitoring for blocks. This could take a simple form, where you schedule a job to save the results of a DMV query into a table. It could also be a third-party database performance monitoring solution, such as SentryOne SQL Sentry (https://www.sentryone.com/products/sentryone-platform/sql-sentry/sql-server-performance-monitoring). For instance, SQL Sentry automatically collects problematic blocking information, maintains ongoing history of blocking details, and presents blocking chains visually for analysis.



(https://www.sentryone.com/hs-fs/hubfs/f39861b8-a667-9485-5f20-ea1513de4fd5.png)

How Do I Troubleshoot SQL Server Blocking?

There are a few common reasons often found at the root of problematic SQL Server blocking.

1. A query that runs slowly and locks aggressively while many other processes read or modify the same data.
2. An application uses a complex transaction that performs several statements or batches before committing.
3. A transaction is left open accidentally and isn't releasing locks.

To address the first two reasons, you need to start by discovering the queries at the head of the blocking chain. You'll get this from your monitoring platform or analyzing DMVs. Once you have the queries involved, you can improve the blocking situation by performance tuning those queries. Performance tuning queries is outside the scope of this article, but SentryOne Plan Explorer (https://www.sentryone.com/plan-explorer) is a good place to start.

For abandoned open transactions, you'll need to discover the SPID holding the transaction open. Querying the DMV sys.dm_tran_active_transactions (https://docs.microsoft.com/e
us/sql/relational-databases/system-dynamic-management-views/sys-dm-tran-active-transactions-transact-sql?view=sql-server-ver15) can help with that. From there, your options a
If you are able to track down and take control of the query, you can issue a rollback. If not, you will likely need to use the T-SQL kill command to end the process.

Monitoring SQL Server Blocking with SQL Sentry

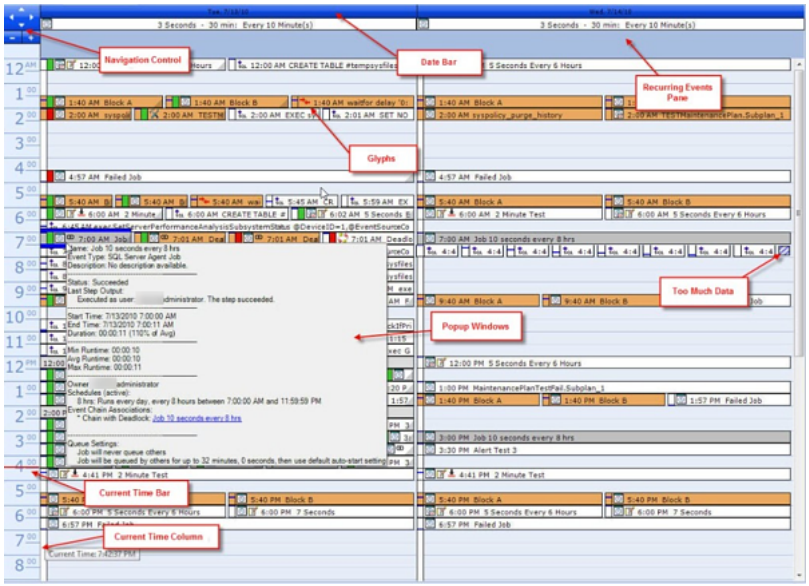
SQL Server blocking doesn't have to be difficult to track down and resolve. SQL Sentry offers block analysis that captures all SQL blocking details (https://docs.sentryone.com/help/blocking-sql#blocking-sql-metrics) based on the configured blocking duration thresholds. You can view blocking chains either in real-time or historically. Information such as the executed statement, login, host, and database are provided for every process involved. You can also see wait types as well as wait resources. View blocks right when they occur and kill blocking processes directly from the SQL Sentry client with the click of a button.

SQL Sentry maintains history for Top SQL (https://www.sentryone.com/C:/products/Features/top-sql), blocking SQL, and deadlock (/sql-server/sql-deadlock) data. Accessing readily available historic performance data makes troubleshooting blocking easier by painting a clear picture of what has occurred. You can even configure actions to respond to various blocking SQL conditions. When problematic blocking occurs, you are alerted to the situation, and a specified action is taken based on your preferences.

SQL Sentry displays blocks in a hierarchical view that shows the relationships of all blocking and blocked SPIDs in a SQL Server blocking chain. Color-coded nodes displayed with the blocking statement let you know if a statement is still running, has completed, or if it is blocked.

(https://www.sentryone.com/hs-fs/hubfs/9906c63b-2fb5-1eaa-14a3-e7447e9d2a8b.png)

You can also view blocks in the SentryOne Event Calendar (https://www.sentryone.com/C:/products/features/event-calendar) side-by-side with other events. SentryOne offers incredible functionality to help resolve SQL Server blocking issues and can help you get to the root of problems quickly and efficiently.



(https://www.sentryone.com/hs-fs/hubfs/0f09e062-8f41-27df-a664-a84ca81dd102.png)
The SQL Sentry Calendar view provides analysis for many types of events, including SQL Server blocking.

Find out what our customers are saying on TrustRadius

What our customers are saying on TrustRadius

SQL Sentry - Best of Breed (https://www.trustr…

★★★★½ May 1, 2020

❝ A report was added to two locations in our application. That blew out the CPU. SQL Sentry was able to identify the times when heavy usage ov...

Derek Knutsen
Database Administrator
American Health Network
Hospital & Health Care | 1001-5000 emp…

Read full review (https://www.trustradius.com/reviews/sql-sentry-2017-01-13-09-09-12?rk=1&utm_campaign=tqw&utm_medium=widget&utm_source=www.sentryone.com&trtid=f43114c9-165d-47bf-a270-47dc6c53be04)

SentryOne is an Awesome Product (https://w…

★★★★½ Oct 3, 2018

❝ Being able to query the SentryOne database for management reports is awesome (Deadlocks and Blocking)

Louis Fritz
Sr SQL DBA
Computer Software | 501-1000 employees

Read full review (https://www.trustradius.com/reviews/sql-sentry-2018-10-03-11-38-18?rk=1&utm_campaign=tqw&utm_medium=widget&utm_source=www.sentryone.com&trtid=f43114c9-165d-47bf-a270-47dc6c53be04)

First impressions of SQL Sentry Performance …

★★★★☆ Apr 30, 2016

❝ Easier to be aware of the blocking and deadlocks which leads to resolving them quicker.

❝ SQL Sentry Performance Advisor is being used for monitoring production and staging SQL Servers. The main tabs used are Top SQL, Blocking, an...

ARTHUR A.
Employee
Health, Wellness and Fitness | 501-100…

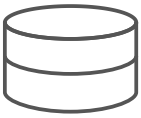
Read full review (https://www.trustradius.com/reviews/sql-sentry-performance-advisor-2016-04-29-11-16-31?rk=1&utm_campaign=tqw&utm_medium=widget&utm_source=www.sentryone.com&trtid=f43114c9-165d-47bf-a270-47dc6c53be04)

Read all reviews (https://www.trustradius.com/products/sql-sentry/reviews?rk=1&utm_campaign=tqw&utm_medium=widget&utm_source=www.sentryone.com&trtid=f43114c9-165d-47bf-a270-47dc6c53be04)

Learn more with these free resources:

- SQL Deadlocks - Detect & Resolve (/sql-server/sql-deadlock)
- Free Webinar: Introduction to Query Tuning on Microsoft SQL Server (https://info.sentryone.com/introduction-to-query-tuning-on-microsoft-sql-server)
- Advanced SQL Server Indexing for DBAs (https://info.sentryone.com/webinar-advanced-sql-server-indexing-for-dbas)
- SQL Sentry: Performance Analysis Blocking SQL (https://docs.sentryone.com/help/blocking-sql)

Ready to start a trial?



Download SQL Sentry Trial (https://www.sentryone.com/cs/c/?cta_guid=b247339a-875a-4a58-8649-258ea0c54acd&signature=AAH58kHgej4SB_eI4Bx4SbV1-ynHQ1KTTg&pageId=5373819617&placement_guid

© 2021 SQL Sentry, LLC. All rights reserved.