

Lock Escalation

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

Contents

- [Issue](#)
- [Investigation / Analysis](#)
- [Mitigation](#)
- [More Information - Lock escalation thresholds](#)
- [Public Doc reference](#)

Lock Escalation

Issue

Lock escalation is the process of converting many fine-grained locks such as row or page locks to table locks. SQL Server dynamically determines when to do lock escalation, considering the number of locks that are held on a particular scan, the number of locks that are held by the whole transaction, and the memory that's used for locks in the system as a whole.

SQL Server's default behavior causes lock escalation to occur only at those times when it would improve performance or when you must reduce excessive system lock memory to a more reasonable level. However, some application or query designs might trigger lock escalation at a time when this action not desirable, and the escalated table lock might block other users.

Lock escalation may occur under one of the following conditions:

- A memory threshold of 40 percent of lock memory is reached. When lock memory exceeds 24 percent of the buffer pool, a lock escalation can be triggered.
Lock memory is limited to 60 percent of the visible buffer pool. The lock escalation threshold is set at 40 percent of the lock memory. This is 40 percent of 60 percent of the buffer pool, or 24 percent. If lock memory exceeds the 60 percent limit (this is much more likely if lock escalation is disabled), all attempts to allocate additional locks fail, and 1204 errors are generated.
- A lock threshold is reached. After the memory threshold is checked, the number of locks acquired on the current table or index is assessed. If the number exceeds 5000, a lock escalation is triggered.

Investigation / Analysis

The first step in the investigation is to determine the blocking scenario. Use the steps in TSG [Blocking](#) to get an idea about the sessions that are involved in the blocking.

The guidance is:

- A lock escalation shows itself by the head blocker being a TAB lock (table-level lock) with a lock mode of S (shared) or X (exclusive). Lock escalation always converts to a TAB lock, never to a PAGE lock.

- If the lock that is blocking other users is anything other than a TAB lock that has a lock mode of S or X, then lock escalation is not the problem. In particular, if the TAB lock is an intent lock (such as a lock mode of IS, IU, or IX), this is not caused by lock escalation.

If your blocking problems are not caused by lock escalation, continue troubleshooting with the steps in TSG [Blocking](#) and the public article [Understand and resolve Azure SQL Database blocking problems](#).

Mitigation

The section [Prevent lock escalation](#) in article [Resolve blocking problems caused by lock escalation in SQL Server](#) discusses various options to avoid lock escalations:

- Break up large batch operations into several smaller operations. This will help with staying below the threshold of 5000 rows and reducing the memory allocation for the query.
- Reduce the query's lock footprint by making the query as efficient as possible. The article discusses this in further detail; it will be an exercise in performance troubleshooting.
- Have a separate connection that holds a lock that is incompatible to a TAB lock. Azure SQL Database might kill the separate, idle connection before the actual workload has completed.
- Eliminate lock escalation caused by lack of SARGability, by making sure that the query can use indexes for search predicates and join columns. Typical examples include implicit or explicit data type conversions, the ISNULL() system function, a user-defined function with the column passed as a parameter, or a computation on the column, such as `WHERE CONVERT(INT, column1) = @a OR WHERE Column1*Column2 = 5`.

Although it's possible to disable lock escalation at the table level, it is usually not a good idea to do so because of possible nasty side-effects. Instead, use the prevention strategies that are described above. If you still want to risk it, you can disable lock escalation by executing `ALTER TABLE ... SET (LOCK_ESCALATION = DISABLE)`.

More Information - Lock escalation thresholds

Lock escalation may occur under one of the following conditions:

- A memory threshold of 40 percent of lock memory is reached. When lock memory exceeds 24 percent of the buffer pool, a lock escalation can be triggered.
Lock memory is limited to 60 percent of the visible buffer pool. The lock escalation threshold is set at 40 percent of the lock memory. This is 40 percent of 60 percent of the buffer pool, or 24 percent. If lock memory exceeds the 60 percent limit (this is much more likely if lock escalation is disabled), all attempts to allocate additional locks fail, and 1204 errors are generated.
- A lock threshold is reached. After the memory threshold is checked, the number of locks acquired on the current table or index is assessed. If the number exceeds 5000, a lock escalation is triggered.



To check if memory the threshold is reached, you can query the [sys.dm_os_performance_counters](#) DMV.

```
select * from sys.dm_os_performance_counters
where counter_name = 'Lock Memory (KB)' or counter_name = 'SQL Cache Memory (KB)'
or counter_name = 'Target Server Memory (KB)' or counter_name = 'Total Server Memory (KB)'
```

From the result, $(\text{Lock Memory (KB)} / \text{Target Server Memory (KB)}) * 100$ is the memory used percentage by lock.

You can use the [sys.dm_tran_locks](#)  DMV to calculate the lock count.

Public Doc reference

- [Resolve blocking problems caused by lock escalation in SQL Server](#) 
- [Understand and resolve Azure SQL Database blocking problems](#) 

How good have you found this content?

