CHEAT SHEET SQL SERVER / TROUBLESHOOTING / INTERMEDIATE

Find Blocking in SQL Server

Identify blocked sessions, blockers, lock waits, and the head blocker so you can troubleshoot SQL Server blocking without killing the wrong session.



Updated Oct 4, 2026 Online version →

PROBLEM

Users report that SQL Server is slow or "hung," but CPU is not necessarily high. One or more sessions may be waiting on locks held by another session.

The goal is to answer:

- Which sessions are blocked?
- Which session is blocking them?
- What SQL is each side executing?
- What resource is being waited on?
- Is there a blocking chain?
- Which session is the head blocker?

The blocked query is often the **victim**, not the root cause.

QUICK ANSWER

Start with `sys.dm_exec_requests` and look for active requests where `blocking_session_id > 0`.

SQL

```sql
SELECT
    r.session_id AS blocked_session_id,
    r.blocking_session_id,
    DB_NAME(r.database_id) AS database_name,
    r.status,
    r.command,
    r.wait_type,
    r.wait_time AS wait_ms,
    r.wait_resource,
    CAST(r.total_elapsed_time / 1000.0 AS decimal(18,1)) AS elapsed_seconds,
    s.login_name,
    s.host_name,
    s.program_name,
    txt.text AS blocked_batch_text
FROM sys.dm_exec_requests AS r
INNER JOIN sys.dm_exec_sessions AS s
    ON s.session_id = r.session_id
OUTER APPLY sys.dm_exec_sql_text(r.sql_handle) AS txt
WHERE
    r.blocking_session_id > 0
ORDER BY
    r.wait_time DESC;
```

If this returns rows, blocking is happening **right now**.

> The next step is not to kill the blocked session. Find the blocker and determine whether it is the head of the chain.

Code / commands

1. Show blocked requests and their blockers

Use this as the first blocking check.

```sql
SELECT
    r.session_id AS blocked_session_id,
    r.blocking_session_id,
    DB_NAME(r.database_id) AS database_name,
    r.status,
    r.command,
    r.wait_type,
    r.wait_time AS wait_ms,
    r.wait_resource,
    CAST(r.total_elapsed_time / 1000.0 AS decimal(18,1)) AS elapsed_seconds,
    s.login_name,
    s.host_name,
    s.program_name,
    txt.text AS blocked_batch_text
FROM sys.dm_exec_requests AS r
INNER JOIN sys.dm_exec_sessions AS s
    ON s.session_id = r.session_id
OUTER APPLY sys.dm_exec_sql_text(r.sql_handle) AS txt
WHERE
    r.blocking_session_id > 0
ORDER BY
    r.wait_time DESC;
```

A positive `blocking_session_id` means another session is identified as the blocker.

Do not use `blocking_session_id <> 0` as a blanket rule for normal blocking analysis. SQL Server can return special negative values for some internal/distributed cases.

2. Show blocker and blocked SQL side by side

This is usually the most useful production query because it gives context for both sides.

```sql
SELECT
    blocked.session_id AS blocked_session_id,
    blocked.blocking_session_id,
    DB_NAME(blocked.database_id) AS database_name,
    blocked.status AS blocked_status,
    blocked.wait_type,
    blocked.wait_time AS blocked_wait_ms,
    blocked.wait_resource,
    blocked_sql.text AS blocked_batch_text,

    blocker.status AS blocker_status,
    blocker.command AS blocker_command,
    blocker.cpu_time AS blocker_cpu_ms,
    blocker.logical_reads AS blocker_logical_reads,
    blocker.start_time AS blocker_start_time,
    blocker_sql.text AS blocker_batch_text,
```

```
    blocker_session.login_name AS blocker_login_name,
    blocker_session.host_name AS blocker_host_name,
    blocker_session.program_name AS blocker_program_name,
    blocker_session.open_transaction_count AS blocker_open_transactions
FROM sys.dm_exec_requests AS blocked
LEFT JOIN sys.dm_exec_requests AS blocker
    ON blocker.session_id = blocked.blocking_session_id
LEFT JOIN sys.dm_exec_sessions AS blocker_session
    ON blocker_session.session_id = blocked.blocking_session_id
OUTER APPLY sys.dm_exec_sql_text(blocked.sql_handle) AS blocked_sql
OUTER APPLY sys.dm_exec_sql_text(blocker.sql_handle) AS blocker_sql
WHERE
    blocked.blocking_session_id > 0
ORDER BY
    blocked.wait_time DESC;
```

Important caveat: the blocking session may be **sleeping** and therefore have no row in `sys.dm_exec_requests`.

That often happens when an application starts a transaction, modifies data, then leaves the transaction open without an active request.

In that case, inspect `sys.dm_exec_sessions` and transaction DMVs instead of assuming the blocker disappeared.

---

3. Find likely head blockers

A practical head blocker is a session that is blocking others but is not itself currently blocked by another positive session ID.

SQL

```
WITH blocked AS
(
    SELECT
        r.session_id,
        r.blocking_session_id,
        r.wait_type,
        r.wait_time
    FROM sys.dm_exec_requests AS r
    WHERE r.blocking_session_id > 0
),
blockers AS
(
    SELECT DISTINCT
        blocking_session_id AS session_id
    FROM blocked
)
SELECT
    b.session_id AS head_blocker_session_id,
    s.status AS session_status,
    s.login_name,
    s.host_name,
    s.program_name,
    s.open_transaction_count,
    r.status AS request_status,
    r.command,
    r.wait_type,
    r.blocking_session_id,
    txt.text AS batch_text
FROM blockers AS b
INNER JOIN sys.dm_exec_sessions AS s
    ON s.session_id = b.session_id
```

```
LEFT JOIN sys.dm_exec_requests AS r
    ON r.session_id = b.session_id
OUTER APPLY sys.dm_exec_sql_text(r.sql_handle) AS txt
WHERE
    NOT EXISTS
    (
        SELECT 1
        FROM blocked AS x
        WHERE
            x.session_id = b.session_id
            AND x.blocking_session_id > 0
    )
ORDER BY
    b.session_id;
```

Use this to narrow the investigation to the session at the top of a blocking chain.

This is a practical triage query, not a complete lock graph.

4. Show waiting tasks for blocked sessions

Use `sys.dm_os_waiting_tasks` when you need more detail about the current wait and resource.

SQL

```
SELECT
    wt.session_id,
    wt.blocking_session_id,
    wt.wait_type,
    wt.wait_duration_ms,
    wt.resource_description
FROM sys.dm_os_waiting_tasks AS wt
WHERE
    wt.blocking_session_id IS NOT NULL
ORDER BY
    wt.wait_duration_ms DESC;
```

This can expose task-level waits that are more detailed than the request-level snapshot.

A parallel query can have multiple tasks waiting on different resources, so do not assume one row always represents the whole request.

5. Correlate blocked waits with lock information

Use this only when you need lock-level detail.

SQL

```
SELECT
    wt.session_id AS blocked_session_id,
    wt.blocking_session_id,
    wt.wait_type,
    wt.wait_duration_ms,
    wt.resource_description,

    l.resource_type,
    l.resource_database_id,
    DB_NAME(l.resource_database_id) AS database_name,
    l.resource_associated_entity_id,
    l.request_mode,
    l.request_status,
    l.request_session_id
```

```
FROM sys.dm_os_waiting_tasks AS wt
INNER JOIN sys.dm_tran_locks AS l
    ON l.lock_owner_address = wt.resource_address
WHERE
    wt.blocking_session_id IS NOT NULL
ORDER BY
    wt.wait_duration_ms DESC;
```

Use this when you need to understand the actual locked resource or lock mode involved.

`sys.dm_tran_locks` can return a large amount of data on a busy instance, so avoid repeatedly querying it without a reason.

6. Check whether a blocker owns an open transaction

A sleeping session with an open transaction is a common cause of persistent blocking.

SQL

```
DECLARE @session_id int = 57;

SELECT
    s.session_id,
    s.status,
    s.login_name,
    s.host_name,
    s.program_name,
    s.open_transaction_count,
    at.transaction_begin_time,
    at.transaction_type,
    at.transaction_state
FROM sys.dm_exec_sessions AS s
LEFT JOIN sys.dm_tran_session_transactions AS st
    ON st.session_id = s.session_id
LEFT JOIN sys.dm_tran_active_transactions AS at
    ON at.transaction_id = st.transaction_id
WHERE
    s.session_id = @session_id;
```

If the blocker has an old open transaction and no active request, investigate the application transaction lifecycle before taking action.

7. Inspect the last submitted command for a sleeping blocker

When a blocker has no active request, `sys.dm_exec_requests` cannot give you current SQL text.

For a targeted investigation, `DBCC INPUTBUFFER` can show the last statement submitted by the session.

SQL

```
DBCC INPUTBUFFER (57);
```

Use the actual blocker session ID.

Treat the result as context, not proof that the displayed statement is the exact statement that acquired every lock still being held.

What the result means

Field	Why it matters
`blocked_session_id`	Session currently waiting because of another session.
`blocking_session_id`	Session SQL Server identifies as the blocker for the current request.
`wait_type`	Current wait. Lock waits commonly begin with `LCK_M_`.
`wait_time` / `wait_duration_ms`	How long the current request/task has been waiting.
`wait_resource` / `resource_description`	Resource involved in the wait. Useful for deeper lock analysis.
`open_transaction_count`	Helps identify sessions holding transactions open.
`program_name`	Client application reported by the connection. Useful for finding the owning service/job.
`host_name`	Client machine name reported by the connection.
`request_mode`	Lock mode requested or held, such as shared or exclusive modes.
`request_status`	Lock request state, for example `GRANTED`, `WAIT`, or `CONVERT`.

What to look for

Many sessions blocked by one session

This is a classic blocking chain.

Prioritize the common blocker rather than examining every blocked session independently.

`LCK_M_*` wait types

These indicate lock waits.

Examples include shared, exclusive, update, schema, and intent-lock waits.

The exact lock wait type helps characterize the contention, but the root cause still requires transaction and workload context.

Sleeping blocker with open transactions

This is especially important.

A session can be idle from the application's point of view while still holding locks because its transaction has not committed or rolled back.

Investigate:

- transaction age
- application transaction handling
- connection pooling behavior
- error paths that may skip commit/rollback

Long waits with short blocker SQL

A simple statement can still hold locks for a long time if it is inside a large or long-lived transaction.

Do not judge blocker impact only by the visible statement text.

Repeated blocking from the same application

Look at:

- `login_name`
- `host_name`
- `program_name`

Recurring blocking from one application or job usually calls for an application/query/transaction fix rather than repeated manual intervention.

PRODUCTION NOTES

Blocking is normal; prolonged blocking is the problem

Locks are a core part of SQL Server concurrency control.

A blocked session does not automatically indicate a defect. Focus on blocking that is prolonged, widespread, unexpected, or user-impacting.

Blocking data is transient

`sys.dm_exec_requests` and `sys.dm_os_waiting_tasks` show current activity.

If blocking clears before you capture it, the evidence disappears.

For intermittent incidents, capture DMV snapshots over time or use Extended Events / monitoring.

Permissions vary by SQL Server version

For SQL Server 2019 and earlier, seeing all sessions/requests generally requires `VIEW SERVER STATE`.

For SQL Server 2022 and later, the corresponding server-level permission is `VIEW SERVER PERFORMANCE STATE`.

Azure SQL Database has different scope and permission rules.

Avoid broad `sys.dm_tran_locks` scans unless needed

Lock-level diagnostics can return a large result set on a busy server.

Start with requests and waiting tasks, then move to `sys.dm_tran_locks` when the resource-level detail is actually useful.

Do not kill sessions casually

Terminating a blocker can:

- roll back a large transaction
- create additional blocking during rollback
- cause application errors
- interrupt business processing
- hide the actual root cause

If termination is operationally necessary, understand the transaction and rollback impact first.

COMMON MISTAKES

1. Killing the blocked session

The blocked session is usually waiting because another session owns an incompatible lock.

Killing the victim does not fix the blocker.

2. Assuming the session with the longest query is the blocker

Duration and blocking are different signals.

Follow the `blocking_session_id` chain.

3. Ignoring sleeping sessions

A sleeping session can still hold locks through an open transaction.

Always inspect sessions and transaction state when an identified blocker has no active request.

4. Treating every nonzero `blocking_session_id` as a normal SPID

SQL Server can return special negative values for some internal/distributed locking scenarios.

Use positive IDs for normal session-to-session blocker analysis.

5. Assuming every `LCK_M_*` wait means "add an index"

An index may reduce lock duration in some workloads, but blocking can also come from:

- long transactions
- application transaction scope
- large batch modifications
- isolation behavior
- conflicting access patterns

Diagnose the transaction and execution plan before changing indexes.

6. Ignoring transaction boundaries

A query may finish quickly while the transaction remains open.

Always ask whether the application commits or rolls back promptly.

NEXT STEP

Use the blocking pattern to decide what to inspect next.

- **One session blocks many others** → investigate the head blocker first.
- **Head blocker is active** → inspect its SQL, execution plan, transaction scope, reads, writes, and duration.
- **Head blocker is sleeping** → inspect open transactions and the application's commit/rollback behavior.
- `LCK_M_*` **dominates** → inspect lock mode, locked resource, transaction size, and access order.
- **Blocking repeatedly comes from one job/application** → move from incident response to application/query remediation.
- **Blocking disappears before capture** → add Extended Events or periodic DMV collection.
- **No blocking is present** → return to the broader SQL Server performance troubleshooting workflow.

Related deep dives

- Related cheat sheet: Find What Is Running in SQL Server
- Related cheat sheet: Inspect a SQL Server Session
- Related cheat sheet: Find Open Transactions
- Related cheat sheet: SQL Server Wait Types
- Related cheat sheet: SQL Server Performance Troubleshooting Flow
- Related article: How to Find Blocking in SQL Server
- Related article: How to Get a SQL Server Session ID and See What It Is Doing
- Related article: SQL Server Is Slow: A Practical Troubleshooting Workflow

Javi Builds Online version More cheat sheets javibuilds.com