AIDE-MÉMOIRE SQL SERVER / DÉPANNAGE / INTERMÉDIAIRE

Trouver les blocages dans SQL Server

Identifiez les sessions bloquées, les bloqueurs, les attentes de verrou et le bloqueur principal pour diagnostiquer les blocages SQL Server sans tuer la mauvaise session.



Mis à jour le 4 oct. 2026 Version en ligne →

PROBLÈME

Les utilisateurs signalent que SQL Server est lent ou « figé », mais le CPU n'est pas forcément élevé. Une ou plusieurs sessions attendent peut-être des verrous détenus par une autre session.

L'objectif est de répondre aux questions suivantes :

- Quelles sessions sont bloquées ?
- Quelle session les bloque ?
- Quel code SQL chaque partie exécute-t-elle ?
- Quelle ressource est attendue ?
- Existe-t-il une chaîne de blocage ?
- Quelle session est le bloqueur principal ?

La requête bloquée est souvent la **victime**, pas la cause racine.

RÉPONSE RAPIDE

Commencez par `sys.dm_exec_requests` et recherchez les requêtes actives pour lesquelles `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;
```

> Si cette requête renvoie des lignes, un blocage se produit **en ce moment même**.
>
> L'étape suivante n'est pas de tuer la session bloquée. Trouvez le bloqueur et déterminez s'il se trouve en tête de la chaîne.

Code / commandes

1. Afficher les requêtes bloquées et leurs bloqueurs

Utilisez cette requête comme première vérification des blocages.

```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;
```

Une valeur positive de `blocking_session_id` signifie qu'une autre session est identifiée comme le bloqueur.

N'utilisez pas `blocking_session_id <> 0` comme règle générale pour l'analyse courante des blocages. SQL Server peut renvoyer des valeurs négatives spéciales dans certains cas internes ou distribués.

2. Afficher côte à côte le SQL du bloqueur et celui de la session bloquée

C'est généralement la requête la plus utile en production, car elle donne le contexte des deux côtés.

```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;
```

Mise en garde importante : la session bloquante peut être **en veille** (sleeping) et n'avoir donc aucune ligne dans `sys.dm_exec_requests`.

Cela se produit souvent lorsqu'une application démarre une transaction, modifie des données, puis laisse la transaction ouverte sans requête active.

Dans ce cas, examinez `sys.dm_exec_sessions` et les DMV de transaction au lieu de supposer que le bloqueur a disparu.

---

3. Trouver les bloqueurs principaux probables

En pratique, un bloqueur principal est une session qui en bloque d'autres sans être elle-même actuellement bloquée par un autre ID de session positif.

```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;
```

Utilisez-la pour concentrer l'investigation sur la session située en tête d'une chaîne de blocage.

Il s'agit d'une requête de tri pratique, pas d'un graphe de verrous complet.

4. Afficher les tâches en attente des sessions bloquées

Utilisez `sys.dm_os_waiting_tasks` lorsque vous avez besoin de plus de détails sur l'attente en cours et la ressource concernée.

```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;
```

Cela peut révéler des attentes au niveau des tâches, plus détaillées que l'instantané au niveau des requêtes.

Une requête parallèle peut avoir plusieurs tâches en attente sur des ressources différentes : ne supposez donc pas qu'une ligne représente toujours la requête entière.

5. Mettre en correspondance les attentes bloquées et les informations de verrouillage

N'utilisez cette requête que lorsque vous avez besoin de détails au niveau des verrous.

```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;
```

Utilisez-la lorsque vous devez comprendre la ressource réellement verrouillée ou le mode de verrouillage en jeu.

`sys.dm_tran_locks` peut renvoyer un volume important de données sur une instance chargée : évitez donc de l'interroger de façon répétée sans raison.

---

6. Vérifier si un bloqueur détient une transaction ouverte

Une session en veille avec une transaction ouverte est une cause fréquente de blocage persistant.

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;
```

Si le bloqueur a une ancienne transaction ouverte et aucune requête active, examinez le cycle de vie des transactions de l'application avant d'agir.

---

7. Examiner la dernière commande soumise par un bloqueur en veille

Lorsqu'un bloqueur n'a aucune requête active, `sys.dm_exec_requests` ne peut pas vous fournir le texte SQL en cours.

Pour une investigation ciblée, `DBCC INPUTBUFFER` peut afficher la dernière instruction soumise par la session.

SQL

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

Utilisez l'ID de session réel du bloqueur.

Considérez le résultat comme un élément de contexte, et non comme la preuve que l'instruction affichée est exactement celle qui a acquis chacun des verrous encore détenus.

Interpréter le résultat

Champ	Pourquoi c'est important
`blocked_session_id`	Session actuellement en attente à cause d'une autre session.
`blocking_session_id`	Session que SQL Server identifie comme le bloqueur de la requête en cours.
`wait_type`	Attente en cours. Les attentes de verrou commencent généralement par `LCK_M_`.
`wait_time` / `wait_duration_ms`	Depuis combien de temps la requête ou la tâche en cours attend.
`wait_resource` / `resource_description`	Ressource concernée par l'attente. Utile pour une analyse approfondie des verrous.
`open_transaction_count`	Aide à identifier les sessions qui maintiennent des transactions ouvertes.
`program_name`	Application cliente déclarée par la connexion. Utile pour retrouver le service ou le job responsable.
`host_name`	Nom de la machine cliente déclaré par la connexion.
`request_mode`	Mode de verrouillage demandé ou détenu, par exemple les modes partagé ou exclusif.
`request_status`	État de la demande de verrou, par exemple `GRANTED`, `WAIT` ou `CONVERT`.

Points à vérifier

De nombreuses sessions bloquées par une seule session

C'est une chaîne de blocage classique.

Donnez la priorité au bloqueur commun plutôt que d'examiner chaque session bloquée séparément.

Types d'attente `LCK_M_*`

Ils indiquent des attentes de verrou.

Il peut s'agir par exemple d'attentes sur des verrous partagés, exclusifs, de mise à jour, de schéma ou d'intention.

Le type exact d'attente de verrou aide à caractériser la contention, mais la cause racine exige toujours le contexte de la transaction et de la charge de travail.

Bloqueur en veille avec des transactions ouvertes

Ce cas est particulièrement important.

Une session peut être inactive du point de vue de l'application tout en détenant encore des verrous, parce que sa transaction n'a été ni validée ni annulée.

Examinez :

- l'ancienneté de la transaction
- la gestion des transactions par l'application
- le comportement du pool de connexions
- les chemins d'erreur susceptibles d'ignorer le commit ou le rollback

Attentes longues avec un SQL bloquant court

Une instruction simple peut tout de même détenir des verrous longtemps si elle s'exécute dans une transaction volumineuse ou de longue durée.

Ne jugez pas l'impact d'un bloqueur uniquement sur le texte de l'instruction visible.

Blocages répétés provenant de la même application

Examinez :

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

Des blocages récurrents provenant d'une même application ou d'un même job appellent généralement une correction de l'application, de la requête ou de la transaction plutôt que des interventions manuelles répétées.

> **NOTES DE PRODUCTION**
>
> #### Le blocage est normal ; c'est le blocage prolongé qui pose problème
>
> Les verrous sont un élément central du contrôle de la concurrence dans SQL Server.
>
> Une session bloquée n'indique pas automatiquement un défaut. Concentrez-vous sur les blocages prolongés, étendus, inattendus ou qui affectent les utilisateurs.
>
> #### Les données de blocage sont éphémères
>
> `sys.dm_exec_requests` et `sys.dm_os_waiting_tasks` montrent l'activité en cours.
>
> Si le blocage se résorbe avant que vous ne le capturiez, les preuves disparaissent.
>
> Pour les incidents intermittents, capturez des instantanés des DMV dans la durée ou utilisez les Extended Events ou un outil de supervision.
>
> #### Les autorisations varient selon la version de SQL Server
>
> Pour SQL Server 2019 et les versions antérieures, voir toutes les sessions et requêtes nécessite généralement `VIEW SERVER STATE`.
>
> Pour SQL Server 2022 et les versions ultérieures, l'autorisation correspondante au niveau du serveur est `VIEW SERVER PERFORMANCE STATE`.
>
> Azure SQL Database applique des règles de portée et d'autorisation différentes.
>
> #### Évitez les analyses larges de `sys.dm_tran_locks` sauf si nécessaire
>
> Les diagnostics au niveau des verrous peuvent renvoyer un jeu de résultats volumineux sur un serveur chargé.
>
> Commencez par les requêtes et les tâches en attente, puis passez à `sys.dm_tran_locks` lorsque le détail au niveau des ressources est réellement utile.
>
> #### Ne tuez pas des sessions à la légère
>
> Mettre fin à un bloqueur peut :
>
> - annuler (rollback) une transaction volumineuse
> - provoquer des blocages supplémentaires pendant le rollback
> - entraîner des erreurs applicatives
> - interrompre des traitements métier
> - masquer la véritable cause racine
>
> Si l'arrêt de la session est opérationnellement nécessaire, commencez par comprendre la transaction et l'impact du rollback.

ERREURS FRÉQUENTES

1. Tuer la session bloquée

La session bloquée attend généralement parce qu'une autre session détient un verrou incompatible.

Tuer la victime ne règle pas le problème du bloqueur.

2. Supposer que la session dont la requête est la plus longue est le bloqueur

La durée et le blocage sont des signaux différents.

Suivez la chaîne de `blocking_session_id`.

3. Ignorer les sessions en veille

Une session en veille peut toujours détenir des verrous par le biais d'une transaction ouverte.

Examinez toujours les sessions et l'état des transactions lorsqu'un bloqueur identifié n'a aucune requête active.

4. Traiter chaque `blocking_session_id` non nul comme un SPID normal

SQL Server peut renvoyer des valeurs négatives spéciales dans certains scénarios de verrouillage internes ou distribués.

Utilisez les ID positifs pour l'analyse courante des blocages de session à session.

5. Supposer que toute attente `LCK_M_*` signifie « ajouter un index »

Un index peut réduire la durée des verrous pour certaines charges de travail, mais les blocages peuvent aussi provenir :

- de transactions longues
- de la portée des transactions dans l'application
- de modifications par lots volumineux
- du comportement du niveau d'isolation
- de modèles d'accès conflictuels

Diagnostiquez la transaction et le plan d'exécution avant de modifier les index.

6. Ignorer les limites des transactions

Une requête peut se terminer rapidement alors que la transaction reste ouverte.

Demandez-vous toujours si l'application valide ou annule rapidement ses transactions.

ÉTAPE SUIVANTE

Utilisez le profil de blocage pour décider de ce qu'il faut examiner ensuite.

- **Une session en bloque beaucoup d'autres** → examinez d'abord le bloqueur principal.
- **Le bloqueur principal est actif** → examinez son SQL, son plan d'exécution, la portée de sa transaction, ses lectures, ses écritures et sa durée.
- **Le bloqueur principal est en veille** → examinez les transactions ouvertes et la façon dont l'application gère le commit et le rollback.
- **Les attentes `LCK_M_*` dominent** → examinez le mode de verrouillage, la ressource verrouillée, la taille de la transaction et l'ordre d'accès.
- **Les blocages proviennent régulièrement d'un même job ou d'une même application** → passez de la réponse à incident à la correction de l'application ou de la requête.
- **Le blocage disparaît avant la capture** → ajoutez des Extended Events ou une collecte périodique des DMV.
- **Aucun blocage n'est présent** → revenez à la démarche plus large de diagnostic des performances SQL Server.

Pour aller plus loin

- Aide-mémoire associé : Voir ce qui s'exécute dans SQL Server
- Aide-mémoire associé : Inspecter une session SQL Server
- Aide-mémoire associé : Trouver les transactions ouvertes
- Aide-mémoire associé : Types d'attente SQL Server
- Aide-mémoire associé : Démarche de diagnostic des performances SQL Server
- Article associé : Diagnostiquer les blocages dans SQL Server
- Article associé : Obtenir l'ID de session SQL Server et voir ce qu'elle fait
- Article associé : SQL Server est lent : une méthode pratique de diagnostic