

SQL Server Deadlocks

Techniques to identify and resolve them!

Sergio Govoni



Slide and demo: bit.ly/DataSatPordenone2025-SQL-Deadlocks



Sponsors



About me



linkedin.com/in/sgovoni



github.com/segovoni



twitter.com/segovoni



Sergio Govoni



- Transactions and concurrency models
- Lock vs. deadlock
- Deadlock
- Best practices and prevention

Transactions and concurrency models

Transactions and concurrency models

- A-C-I-D: The four properties of a transaction
- Pessimistic concurrency model
 - Uses blocking to avoid conflicts
 - Readers can block writers, and writers can block readers
- Optimistic concurrency model
 - Use row versioning
 - Readers cannot block writers, and writers cannot block readers, but the writer can block another writer

Pessimistic concurrency models

- Four isolation levels
 - Read Uncommitted
 - Allows the dirty read problem
 - Read Committed
 - We can only read committed data
 - Repeatable Reads
 - Acquires shared-lock until the end of the transaction
 - Serializable
 - Any transaction is waiting until the current transaction completes

Optimistic concurrency models

- Two isolation levels(*) based on row versioning
 - Snapshot
 - Read Committed Snapshot

Lock modes

Lock mode	Description
Shared (s)	Used for read operations that do not change or update data, such as a <code>SELECT</code> statement.
Update (u)	Used on resources that can be updated. Prevents a common form of deadlock that occurs when multiple sessions are reading, locking, and potentially updating resources later.
Exclusive (x)	Used for data-modification operations, such as <code>INSERT</code> , <code>UPDATE</code> , or <code>DELETE</code> . Ensures that multiple updates cannot be made to the same resource at the same time.

https://learn.microsoft.com/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide#lock_modes

Lock vs. deadlock

Difference between lock and deadlock

- Deadlocking is often confused with normal blocking
- When a transaction requests a lock on a resource locked by another transaction, the requesting transaction waits until the lock is released
 - The requesting transaction is blocked, not deadlocked
- By default, transactions don't time out, unless LOCK_TIMEOUT is set

Difference between lock and deadlock

- Deadlocks are resolved almost immediately
- Blocking can, in theory, persist indefinitely
- [Lock granularity and hierarchies](#)

Resource locked by another transaction



Misconceptions about deadlocks

- Deadlocks in SQL Server are bugs
- Deadlocks cannot be prevented
- Using NOLOCK on all SELECTs will prevent deadlocks
- Adding covering indexes for every query prevents deadlocks
- Only very experienced SQL Server developers or administrators can troubleshoot deadlocks

Deadlock

What is a deadlock?

A deadlock is a specific type of lock that occurs when two or more tasks permanently block each other, creating a **cyclic dependency** situation!

Cyclic dependency scenario



Cyclic dependency scenario



Deadlock discovery

- The lock monitor thread is tasked with deadlock discovery, conducting periodic scans to identify and resolve them
- During deadlock discovery, the lock monitor identifies blocked tasks and recursively searches the tasks to trace the blocking resource owners, uncovering cyclic blocking that forms a deadlock
- Each time a deadlock is discovered, the search interval reduces from a default of 5 seconds to as fast as 100 milliseconds, depending on the deadlock frequency on the server

Deadlock discovery

- If the monitor detects a cyclic dependency, it chooses one of the tasks as a **victim** and terminates its transaction with an error
 - This allows the other task to complete its transaction
- How is a deadlock victim chosen?

Deadlock priority

- Any user can set the DEADLOCK_PRIORITY session option to influence deadlock resolution behavior
 - It is impossible to prevent a user from setting DEADLOCK_PRIORITY, even with Resource Governor
- Assigning a higher DEADLOCK_PRIORITY to important transactions ensures they are not chosen as the deadlock victim when a deadlock occurs with a lower priority session

Deadlock victim selection

- When a deadlock is detected, deadlock victim is chosen
 - The deadlock victim is killed, rolling back its transaction
 - The client receives a 1205 error
- The deadlock victim is selected based on the following criteria:
 - The DEADLOCK_PRIORITY, lowest priority is the victim
 - If both session have the same DEADLOCK_PRIORITY, the least expensive transaction to rollback, is selected as the victim (default)

Deadlock information tools

- There are several tools that can be used to identify deadlocks:
 - Extended event (recommended)
 - SQL Profiler
 - Trace flags 1204, 1222
 - Avoid using trace flags 1204 and 1222 on workload-intensive systems that are experiencing deadlocks
 - [How It Works: SQL Server Deadlock Trace Flag 1222 Output](#)
- Event notification



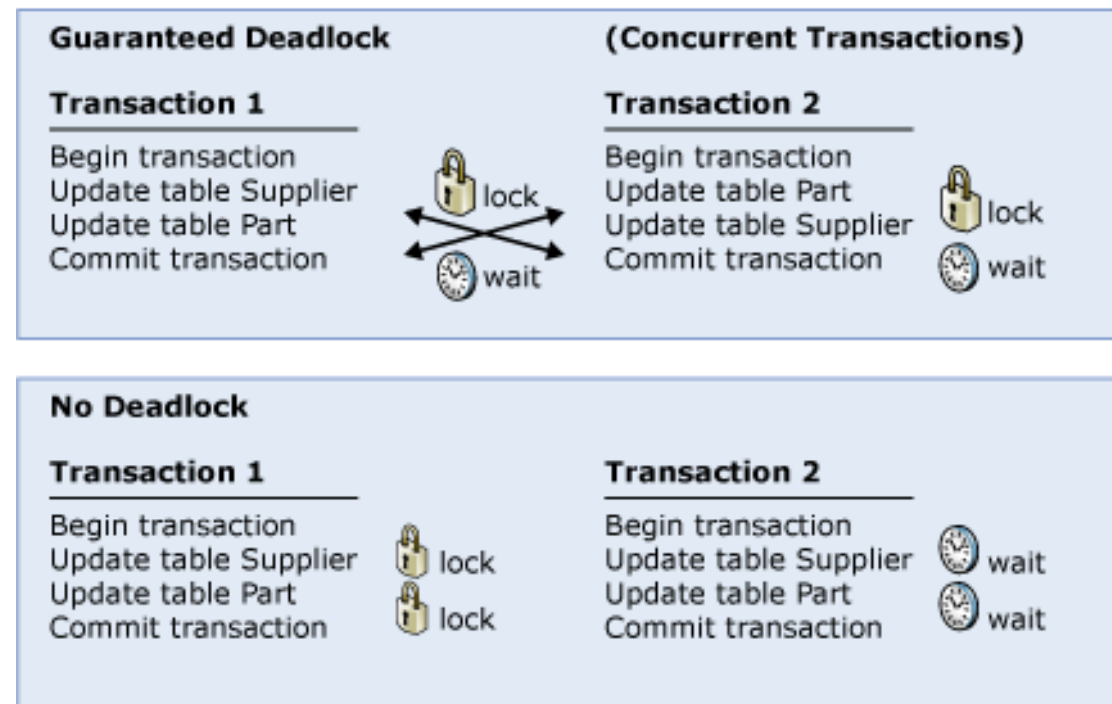
Best practices and prevention

Best practices and prevention

- Deadlocks cannot be completely avoided; anyway, these coding conventions can minimize the chance of generating a deadlock:
 - Access objects in the same order
 - Avoid user interaction in transactions
 - Keep transactions short and in one batch
 - Use a lower isolation level
 - Use a row versioning-based isolation level
 - Use bound connections

Access objects in the same order

- If all concurrent transactions access objects in the same order, deadlocks are less likely to occur



Summary and resources

- Deadlocks in SQL Server aren't bugs but an outcomes of resource contention in multi-user environments
- To prevent deadlocks, focus on designing transactions carefully, using the same order when accessing resources, and use a row versioning-based isolation level (when it is possible)
- SQL Server tools like extended events, SQL Profile, Trace Flags and First Responder Kit can help you to identify and monitor deadlocks

- [Deadlocks guide](#)
- [Analyze Deadlocks with SQL Server Profiler](#)
- [How It Works: SQL Server Deadlock Trace Flag 1222 Output](#)
- [Hands-On-Lab: SQL Server Deadlock Types](#)
- [SQL Server First Responder Kit](#) by BrentOzar
 - sp_Blitz *

Q&A

Thanks!