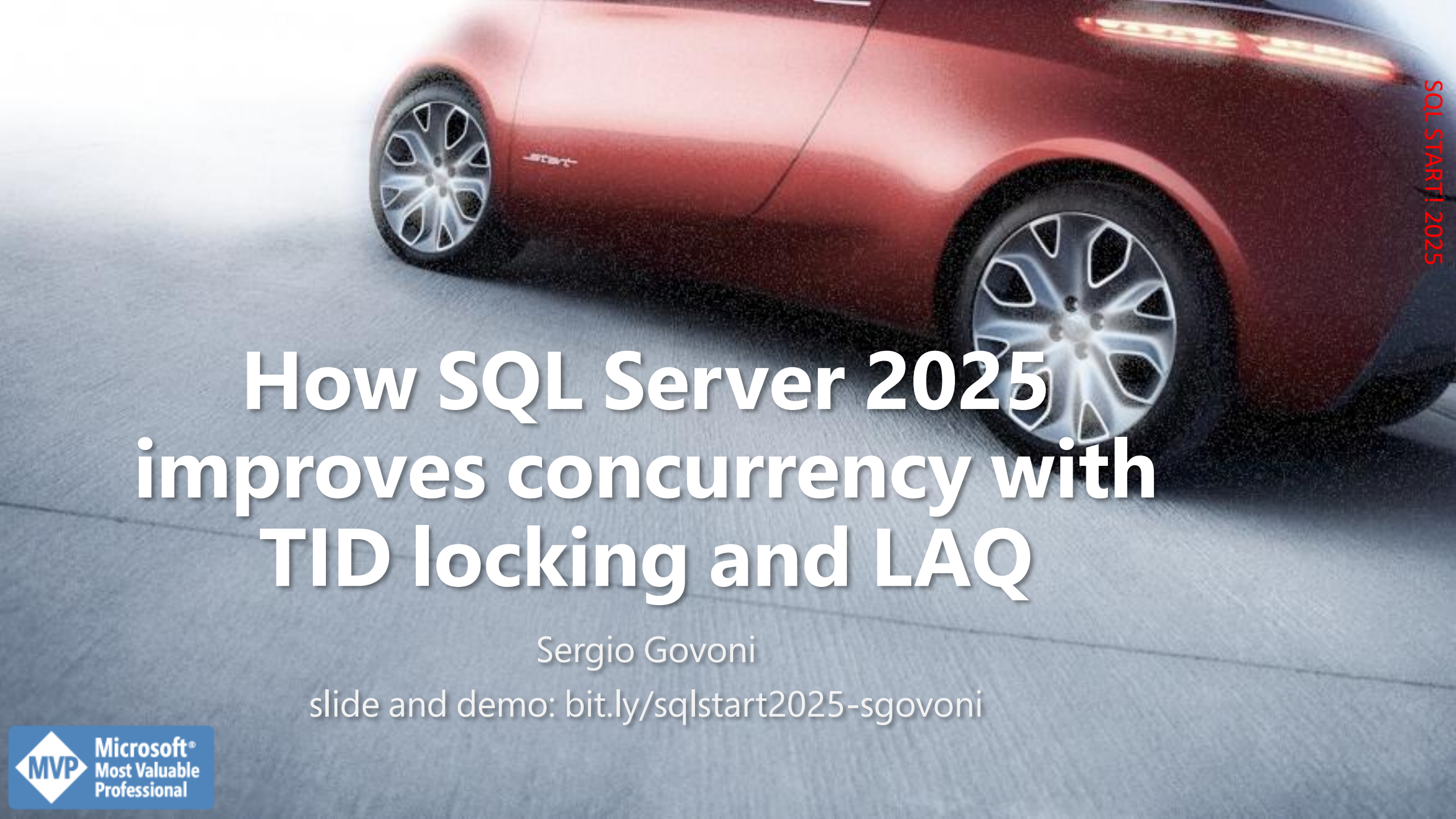


How SQL Server 2025 improves concurrency with TID locking and LAQ

Sergio Govoni

slide and demo: bit.ly/sqlstart2025-sgovoni

SQL START! 2025

Microsoft® Most Valuable Professional

Speaker bio

Sergio Govoni

Microsoft® MVP Most Valuable Professional

twitter.com/segovoni

github.com/segovoni

linkedin.com/in/sgovoni


α
SYS


APRA
a vargroup company


Bi Factory
DATA KNOWLEDGE ADVISOR


Lodestar
LEADING INNOVATION


PLUSERVICE.NET
myCICERO
Società del Gruppo
mooney


Dev>Marche


UGISS


UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Agenda

- SQL Server 2025 Optimized Locking
  - Key components
  - Underlying technologies
  - How it works
  - Demo

Introduction to Optimized Locking

- In the landscape of modern applications, scalability and concurrency are crucial
- Optimized Locking is a new technology available in SQL Server 2025
  - It redefines how SQL Server Engine handles locks, improving concurrency and efficiency
  - It helps to reduce lock memory and avoids lock escalations

Introduction to Optimized Locking

- Optimized Locking is composed of two primary components:
  - Transaction ID (TID) Locking
  - Lock After Qualification (LAQ)
- Transaction ID Locking is designed to optimize memory usage in lock management
- Lock After Qualification eliminates the risk of lock escalation and enhances concurrency in DML operations

Introduction to Optimized Locking

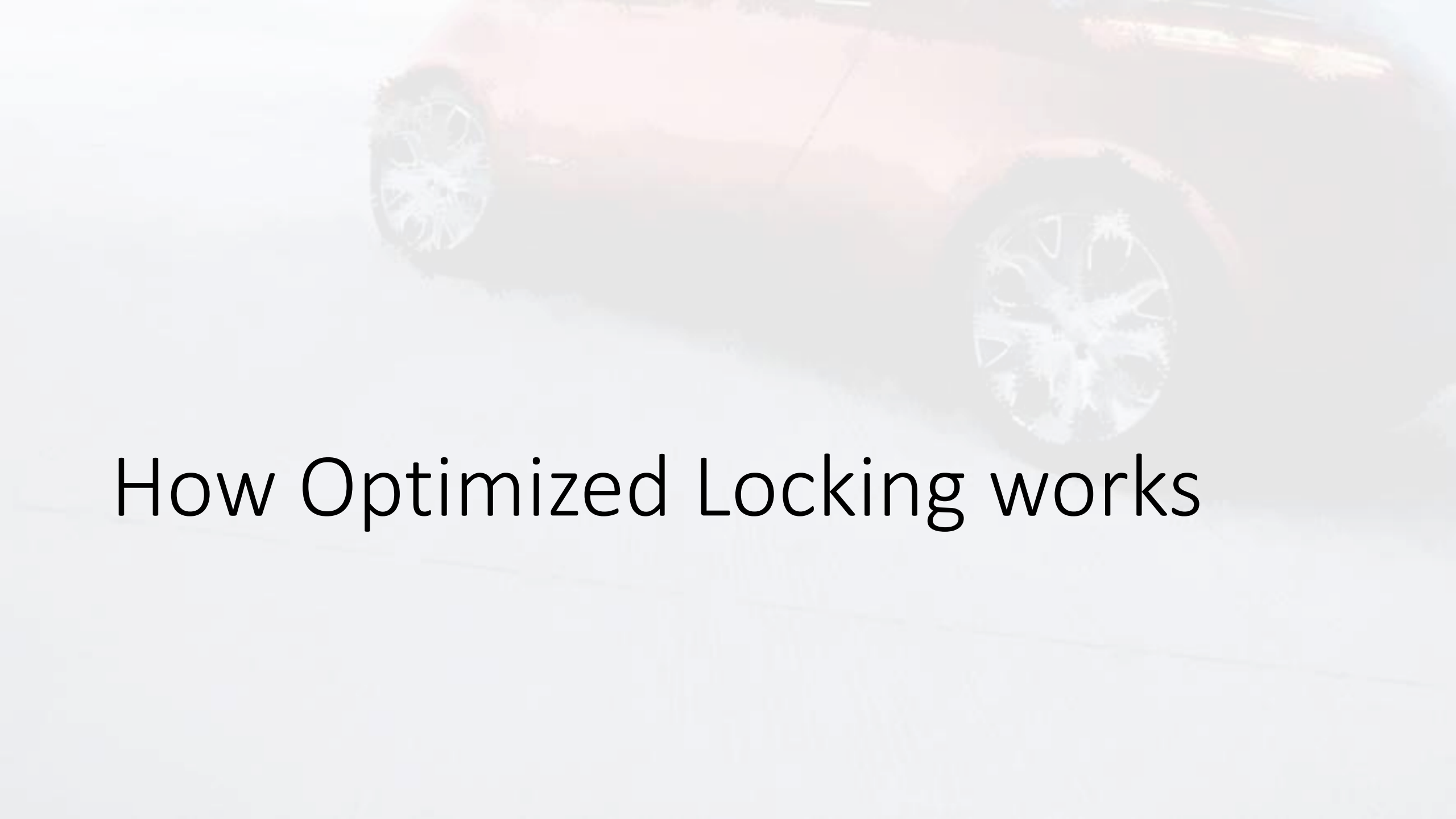
- Optimized Locking is built on two existing technologies
  - Accelerated Database Recovery (ADR)
  - Read Committed Snapshot Isolation level (RCSI)
- Accelerated Database Recovery is mandatory, it must be enabled at the database level
- Read Committed Snapshot Isolation level is not a strict requirement; it significantly enhances because LAQ is active only when READ_COMMITTED_SNAPSHOT option is enabled

Accelerated Database Recovery (ADR)

- It improves database availability, especially in the presence of long-running transactions, by redesigning the database engine recovery process
- When ADR is enabled, every row in the database internally contains a transaction ID (TID) that is persisted on disk

Read Committed Snapshot Isolation (RCSI)

- Read Committed Snapshot is not a separate isolation level, it is a modification of the read committed isolation level when the READ_COMMITTED_SNAPSHOT option is enabled
- When it is enabled, locks are not used to protect data from updates by other transactions, it allows reading the last committed version from the snapshot, reducing contention between reads and writes

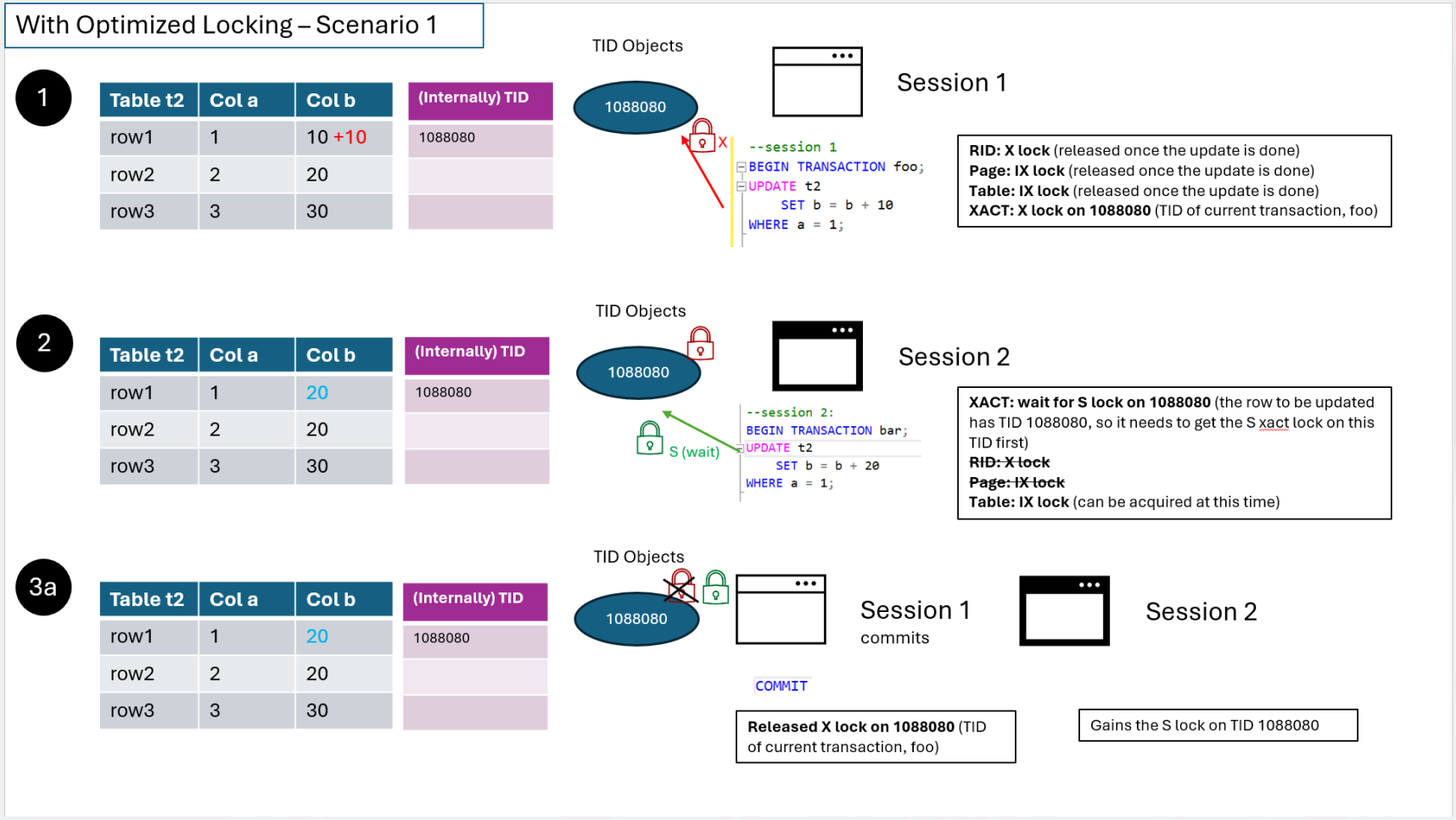


How Optimized Locking works

Transaction ID (TID) locking in action

- With TID locking
  - Each row in the database internally contains a TID
  - TID is persisted on disk, and every transaction modifying a row assigns its own TID to that row
  - Instead of acquiring a lock on the row's key, a lock is taken on the row's TID

Transaction ID (TID) locking in action



https://techcommunity.microsoft.com/blog/azuredbsupport/understanding-optimized-locking-in-azure-sql-database/4358636

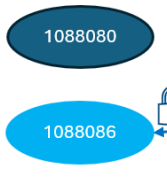
Transaction ID (TID) locking in action

With Optimized Locking – Scenario 1

3b

Table t2	Col a	Col b	(Internally) TID
row1	1	20 + 20	1088086
row2	2	20	
row3	3	30	

TID Objects



Session 2

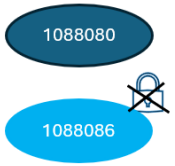
```
--session 2:
BEGIN TRANSACTION bar;
UPDATE t2
    SET b = b + 20
WHERE a = 1;
```

RID: X lock (released once the update is done)
Page: IX lock (released once the update is done)
Table: IX lock (has been acquired already at step 2, released once the update is done)
XACT: X lock on 1088086 (TID of current transaction, bar)

4

Table t2	Col a	Col b	(Internally) TID
row1	1	40	1088086
row2	2	20	
row3	3	30	

TID Objects



Session 2

Commits

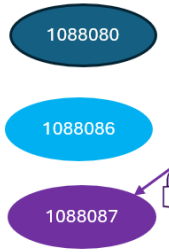
```
COMMIT
```

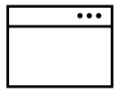
Released X lock on 1088086 (TID of current transaction, bar)

5

Table t2	Col a	Col b	(Internally) TID
row1	1	40 + 10	1088087
row2	2	20	
row3	3	30	

TID Objects





Session 1

Starts a new transaction

```
--session 1
BEGIN TRANSACTION foo2;
UPDATE t2
    SET b = b + 10
WHERE a = 1;
```

RID: X lock (released once the update is done)
Page: IX lock (released once the update is done)
Table: IX lock (released once the update is done)
XACT: no other lock on the TID of the row to be modified, modify it to 1088087 (TID of new current transaction, foo2) **and place X lock on it**

https://techcommunity.microsoft.com/blog/azuredbsupport/understanding-optimized-locking-in-azure-sql-database/4358636

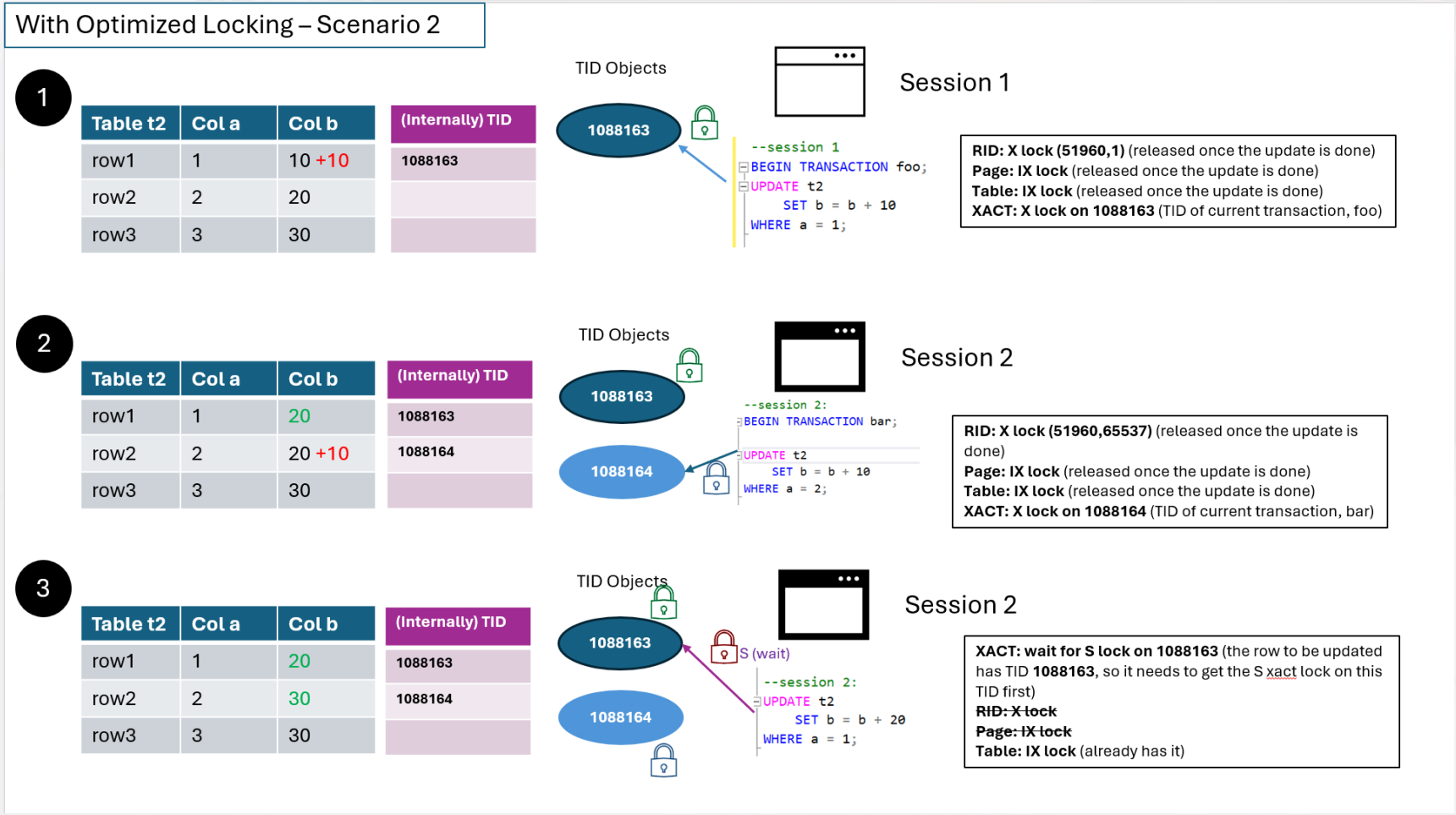
Lock After Qualification (LAQ) in action

- One major cause of DML slowdowns is acquiring locks while searching for qualifying rows. LAQ modifies the way DML statements acquire locks
- Without optimized locking, queries evaluate predicates row by row, first acquiring a U lock, which is upgraded to an X lock if the row meets the condition. The X lock remains until the transaction ends

Lock After Qualification (LAQ) in action

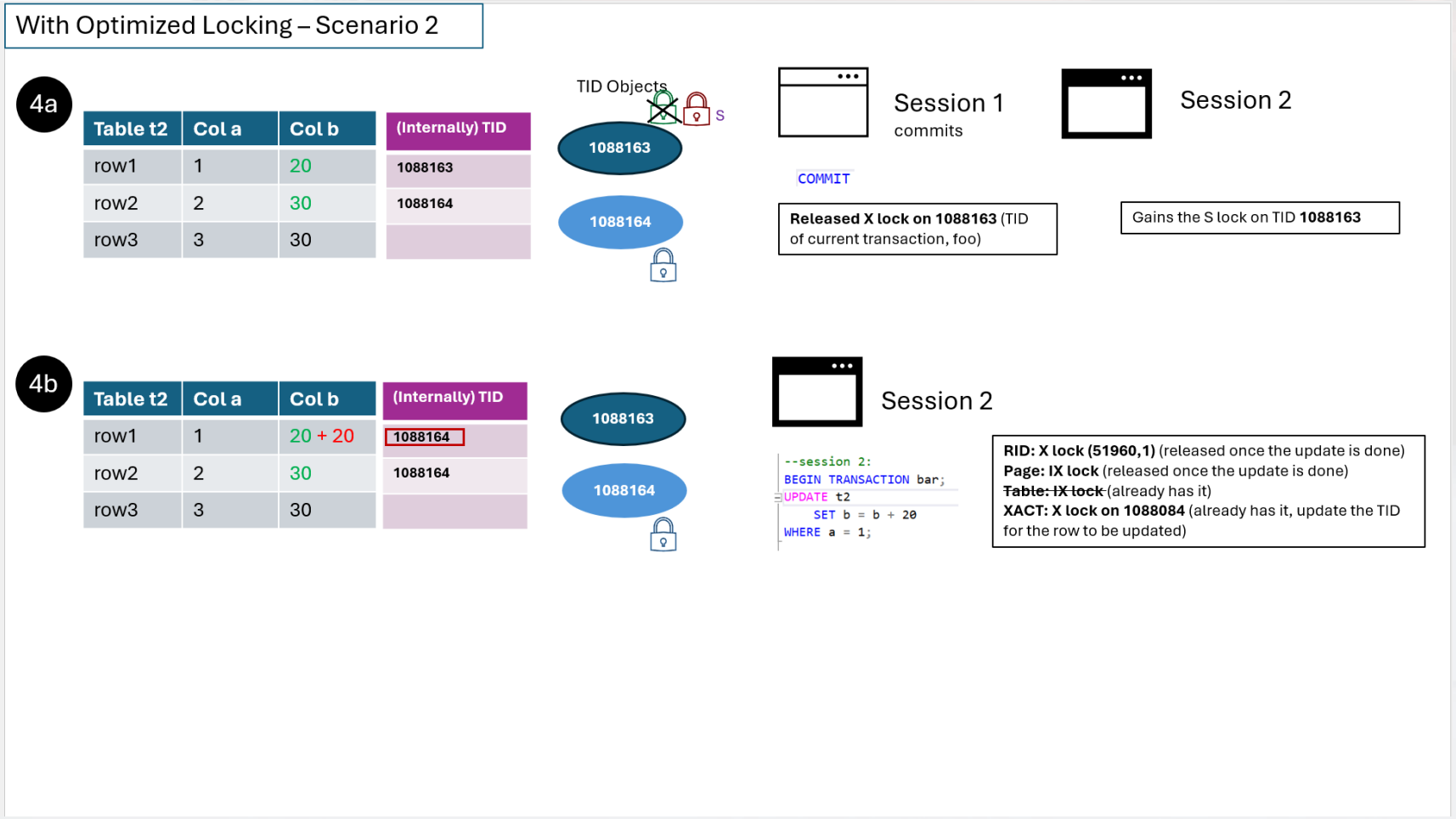
- With LAQ, predicates are evaluated on the latest committed row version without locks. If the condition is met, an X lock is acquired for the update and released immediately after
- This prevents blocking between concurrent queries modifying different rows

Lock After Qualification (LAQ) in action

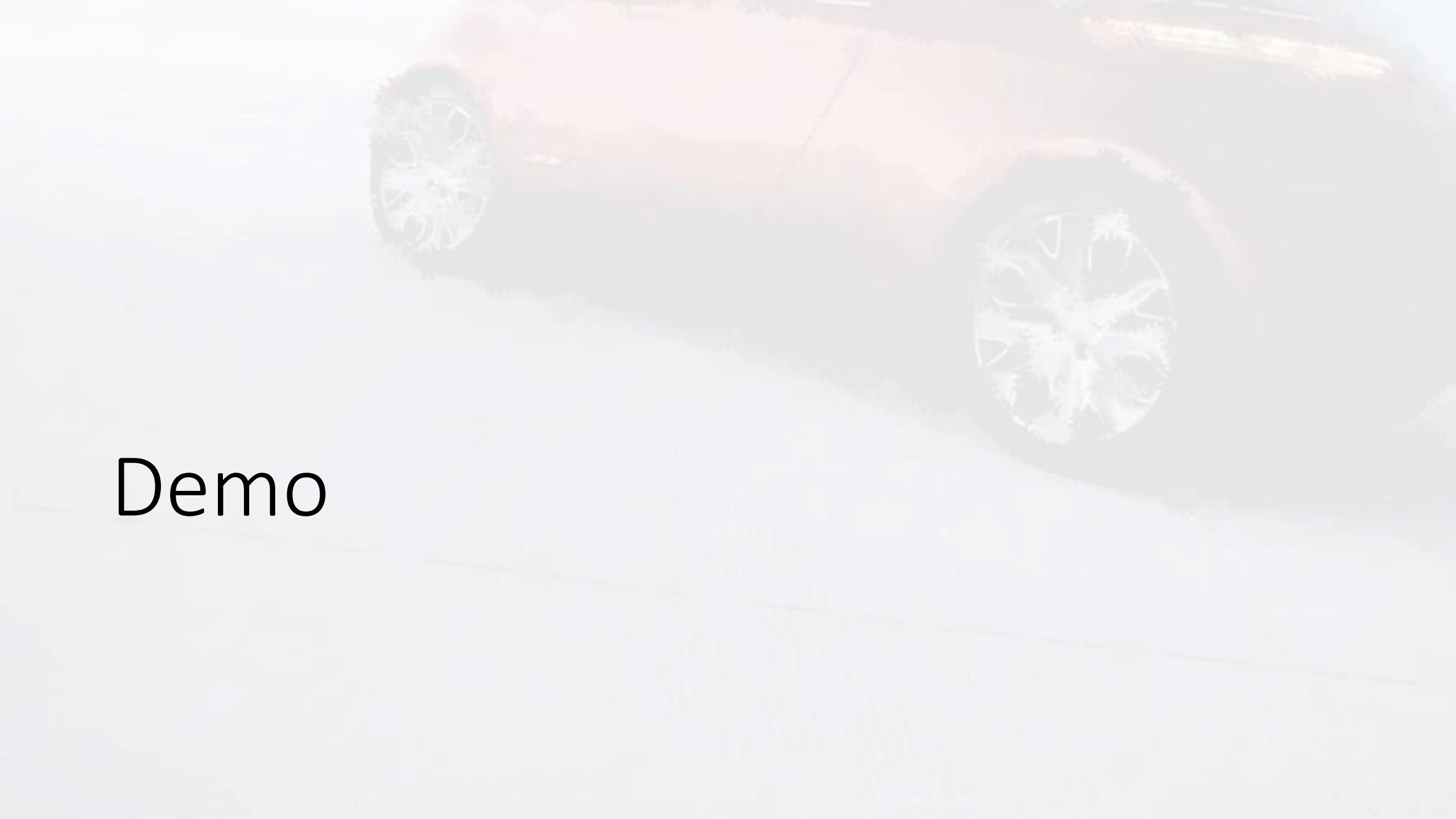


https://techcommunity.microsoft.com/blog/azuredbsupport/understanding-optimized-locking-in-azure-sql-database/4358636

Lock After Qualification (LAQ) in action



https://techcommunity.microsoft.com/blog/azuredbsupport/understanding-optimized-locking-in-azure-sql-database/4358636



Demo

Summary

- Optimized Locking represents a significant evolution in concurrency management; it redefines how SQL Server Engine handles locks
- By using TID locking and LAQ, optimized locking reduces memory consumption and eliminates the lock escalation
- In Azure SQL Database, optimized locking is enabled by default

Resources

- Download SQL Server 2025 today
- SQL Server 2025 documentation
- Upgrade to the new SSMS 21 and Copilot
- Optimized Locking in Azure SQL Database: Concurrency and performance at the next level
- Understanding Optimized Locking in Azure SQL Database

Thanks

Questions?

 github.com/segovoni

 twitter.com/segovoni

 linkedin.com/in/sgovoni