Azure SQL Database Optimized Locking

Concurrency and performance at the next level!

Sergio Govoni

Slide and demo: bit.ly/DataSatPordenone2025-sgovoni-OL

Sponsors

CONSORZIO
UNIVERSITARIO
DI PORDENONE
MOLTIPLICATORE DI VALORE

About me

linkedin.com/in/sgovoni

github.com/segovoni

twitter.com/segovoni

Sergio Govoni

Microsoft® MVP Most Valuable Professional

DATA SATURDAYS

InnOva community

Agenda

- Introduction to optimized locking
  - Key components
  - Underlying technologies
- How optimized locking works
- Demo

DATA SATURDAYS

Introduction to Optimized Locking

Key components and foundational technologies

Introduction to optimized locking

- In the landscape of modern applications, scalability and concurrency are crucial
- Optimized locking is a new technology available in Azure SQL Database
  - It redefines how Azure SQL Database 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 locking** (TID) and **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 in Azure SQL Database: **Accelerated Database Recovery** (ADR) and **Read Committed Snapshot Isolation** level (RCSI)
- Accelerated database recovery is a **mandatory** requirement, 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)

- In Azure SQL Database, **ADR is enabled by default**
- 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)

- In Azure SQL Database, **RCSI is enabled by default**
- 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 READ_COMMITTED_SNAPSHOT is set to ON, 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

DATA SATURDAYS

How optimized locking works

TID locking and LAQ in action

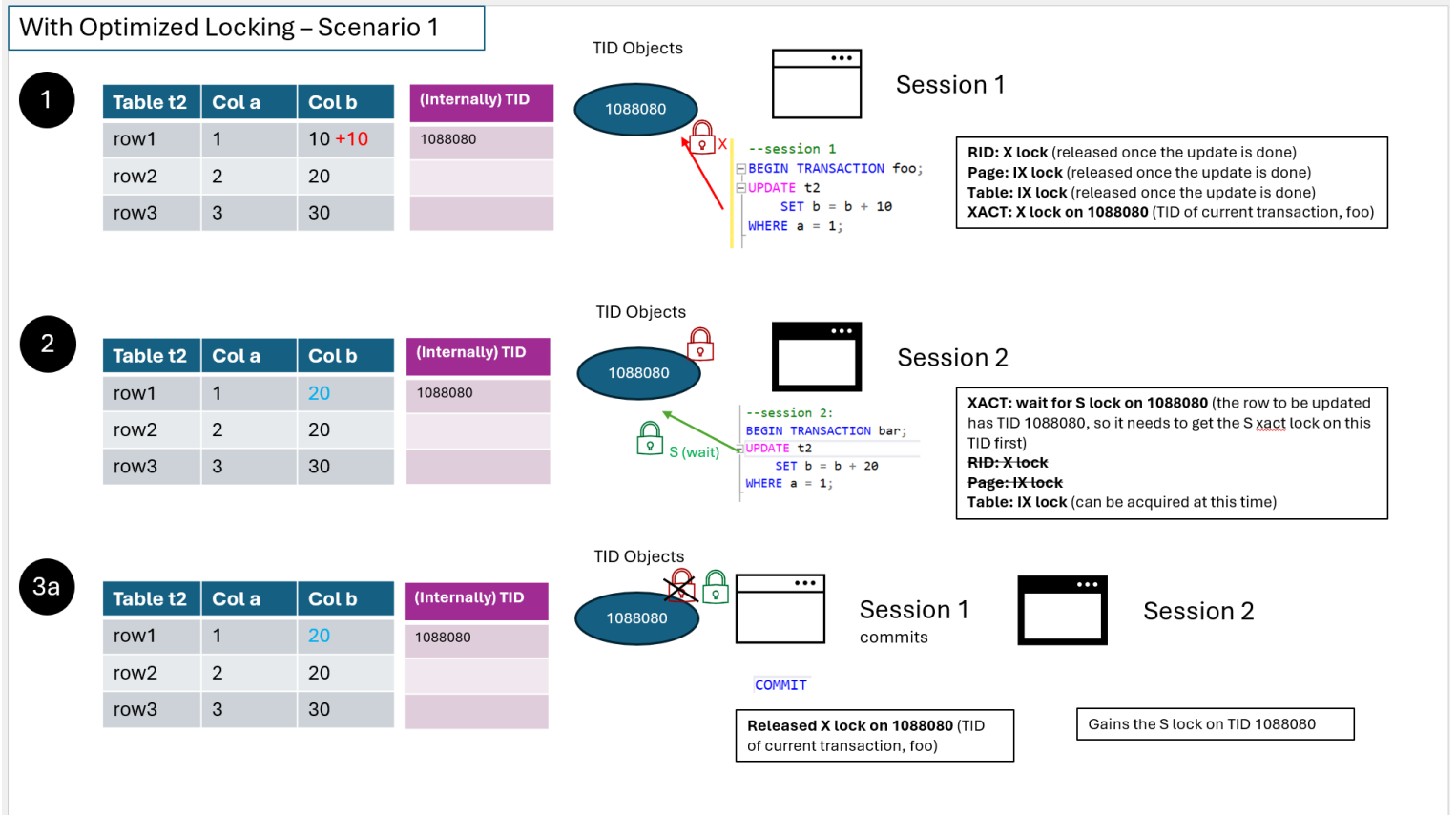
Transaction ID (TID) locking in action

- 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

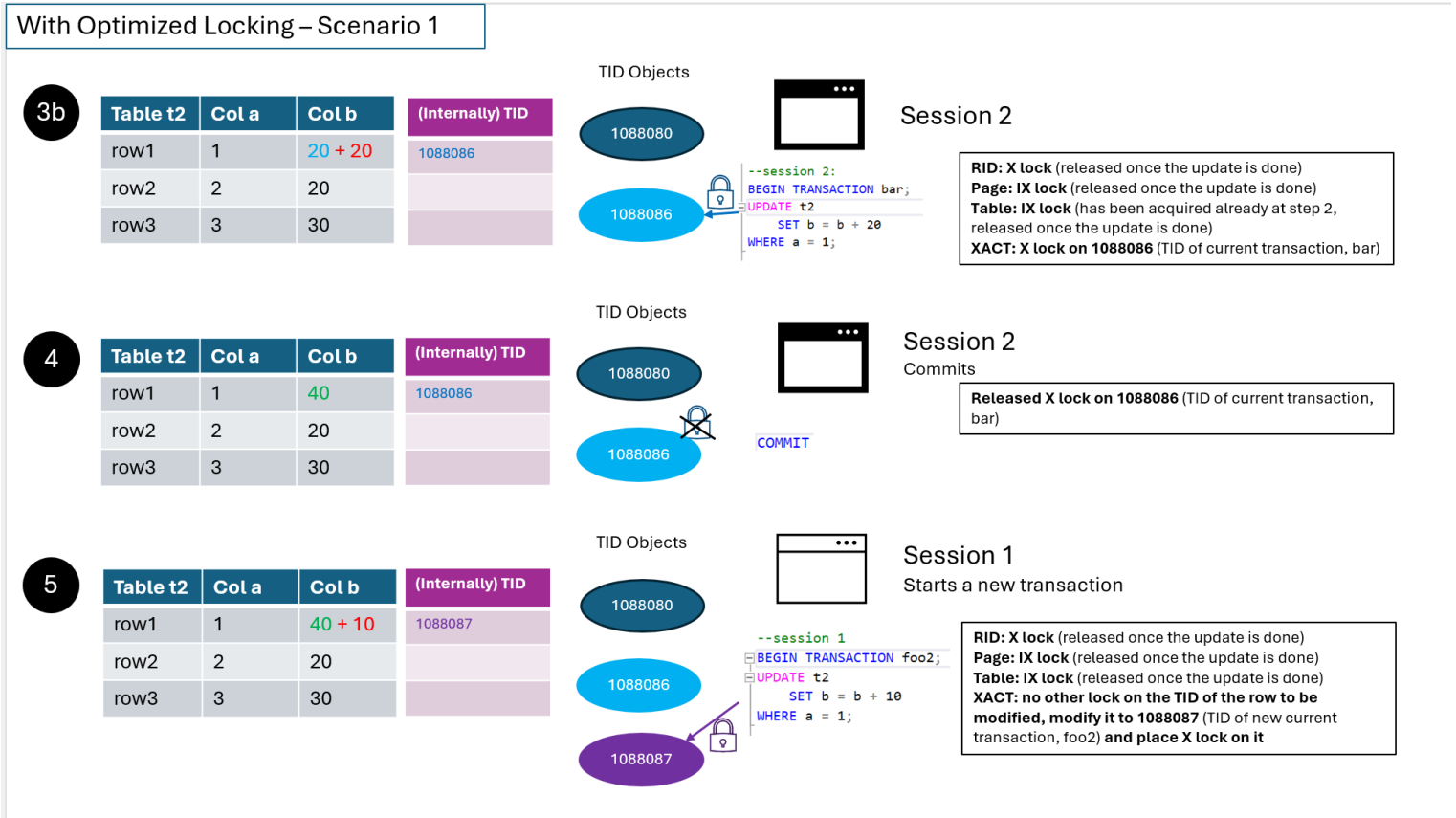
- With TID locking
  - Page and row locks are still taken during modifications, but each page and row lock is released immediately after modification
  - The only lock held until the end of the transaction is the single **X lock** on the TID resource, replacing multiple page and row (key) locks

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



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
- Lock after qualification 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

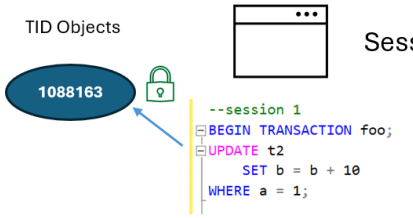
- With LAQ, predicates are evaluated on the latest committed row version without locks. If the condition is met, an X lock is acquired for the update and released immediately after
- This prevents blocking between concurrent queries modifying different rows

Lock After Qualification (LAQ) in action

With Optimized Locking – Scenario 2

1

Table t2	Col a	Col b	(Internally) TID
row1	1	10 +10	1088163
row2	2	20	
row3	3	30	

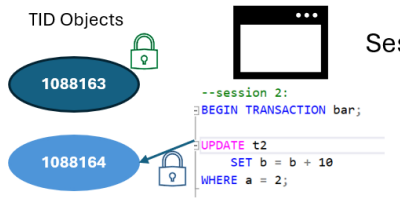


Session 1

RID: X lock (51960,1) (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: X lock on 1088163 (TID of current transaction, foo)

2

Table t2	Col a	Col b	(Internally) TID
row1	1	20	1088163
row2	2	20 +10	1088164
row3	3	30	

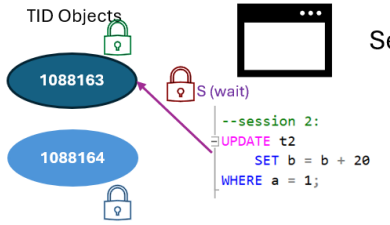


Session 2

RID: X lock (51960,65537) (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: X lock on 1088164 (TID of current transaction, bar)

3

Table t2	Col a	Col b	(Internally) TID
row1	1	20	1088163
row2	2	30	1088164
row3	3	30	

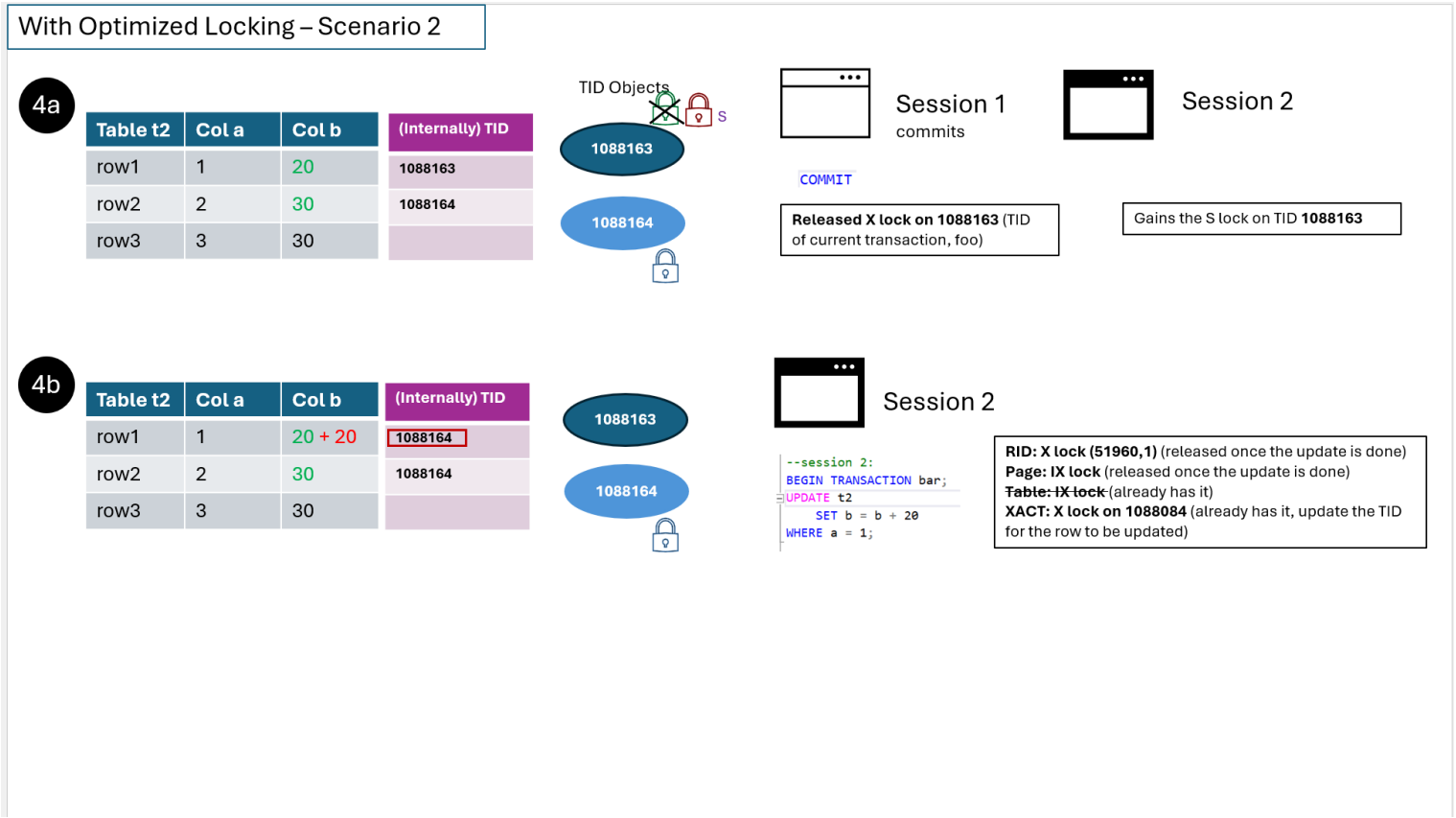


Session 2

XACT: wait for S lock on 1088163 (the row to be updated has TID **1088163**, so it needs to get the S xact lock on this TID first)
~~**RID: X lock**~~
~~**Page: IX lock**~~
Table: IX lock (already has it)

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

DATA SATURDAYS

Summary and resources

Summary

- Optimized locking in Azure SQL Database represents a significant evolution in concurrency management
- By using Transaction ID (TID) locking and Lock After Qualification (LAQ), optimized locking reduces memory consumption and eliminates the lock escalation, minimizing locks between concurrent transactions
- In Azure SQL Database, optimized locking is enabled by default

Resources

- [Optimized Locking in Azure SQL Database: Concurrency and performance at the next level]
- [Optimized Locking in Azure SQL Database: Concorrenza senza limiti]
- [Understanding Optimized Locking in Azure SQL Database]

DATA SATURDAYS

Q&A

DATA SATURDAYS

Thanks!