

ON TOUR | NEW YORK



# Optimized Locking: Improving SQL Server Transaction Concurrency

**Deborah Melkin**

she\her

Data Engineer

Advisor360



# Deborah Melkin

She/Her

Data Engineer  
Advisor360



- . 25 years as a DBA
- . Data Platform Women in Tech (WIT) Virtual UG, Co-leader
- . WITspiration, Co-founder
- . Redgate Community Ambassador
- . Microsoft MVP. Data Platform
- . In my spare time, I can usually be found doing something musical or something geeky with my husband, Andy, and our dog, Sebastian.



[@dgmelkin.bsky.social](https://dgmelkin.bsky.social)



[github.com/DebtheDBA](https://github.com/DebtheDBA)



[DebtheDBA.wordpress.com](https://DebtheDBA.wordpress.com)

# What's the problem we're trying to solve?

- Locking & Blocking!

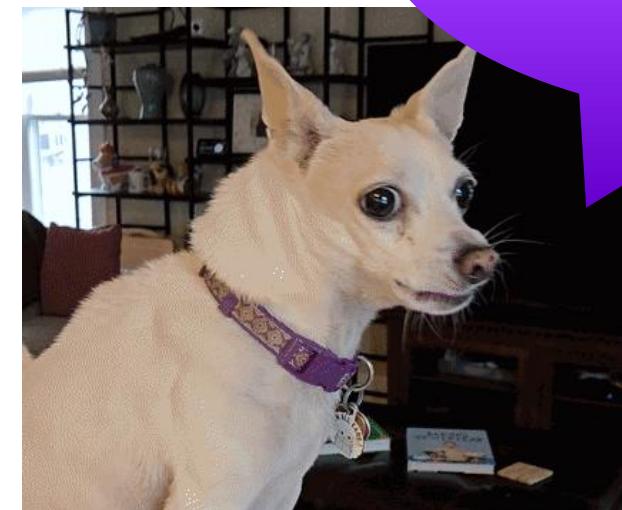


# Introducing Optimized Locking

- GA in Azure SQL DB in Feb 2023
- SQL Server 2025 Public Preview
- Writes not blocking Writes

# Agenda – How it works

- Components used by Optimized Locking
  - Read Committed Snapshot Isolation Level (RCSI)
  - Accelerated Database Recovery (ADR)
- Lock Escalation
- Optimized Locking
  - Transaction ID Locking
  - Lock After Qualification



**Warning:**  
**\* Internals Ahead \***  
**Level 300**

# Pessimistic vs Optimistic Locking

- **Pessimistic Locking** – Preventing users from modifying data in a way that affects other users
  - Read Uncommitted
  - Read Committed
  - Serializable
  - Repeatable Read

# Pessimistic vs Optimistic Locking

- **Optimistic Locking** – No locks when reading data but data is checked when modifying
  - Snapshot Isolation
  - Read Committed Snapshot Isolation
  - Optimized Locking

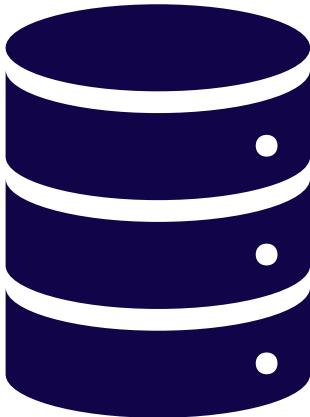
# Read Committed Snapshot Isolation Level (RCSI)

- Introduced with SQL Server 2005
- Writes and Reads don't block each other
- Two database options for setting this:
  - SET READ COMMITTED SNAPSHOT ISOLATION – sets default for the database
  - ALLOW SNAPSHOT ISOLATION LEVEL – allows individual sessions\queries to use SNAPSHOT isolation levels even if read committed snapshot isolation is not set.

# **Read Committed Snapshot Isolation Level (RCSI)**

- Every row in the database has a Transaction ID (TID)
- Keeps copies of the previously committed versions of the records in the Version Store in TempDB
- Only reading the committed version of the data

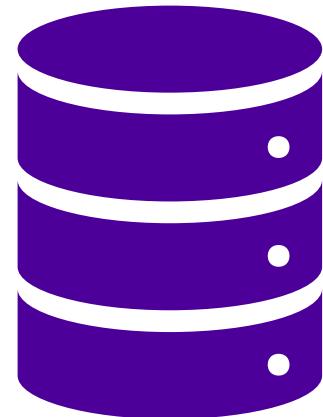
User DB



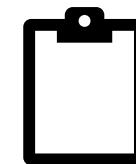
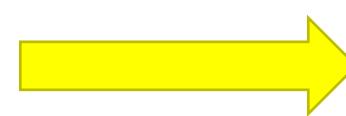
BEGIN TRAN 1A

```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Sebastian'
```

tempdb



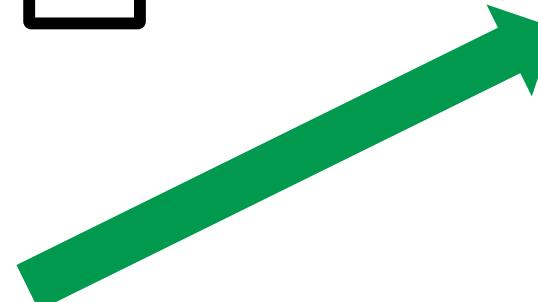
|           |    |     |           |
|-----------|----|-----|-----------|
| Sebastian | MA | Key | TID<br>1A |
|-----------|----|-----|-----------|



| Version Store |    |           |   |
|---------------|----|-----------|---|
| Sebastian     | AZ | TID<br>1A | 1 |

TID  
Transaction Seq No.

```
SELECT FirstName,  
LastName,  
State  
FROM Customer  
WHERE FirstName = 'Sebastian'
```



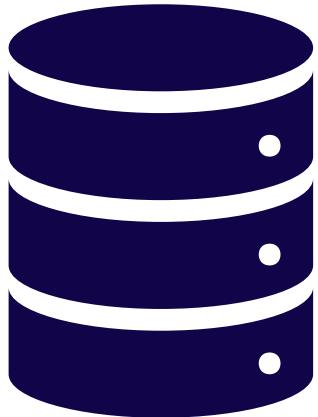
Read Committed  
Snapshot Isolation

# Accelerated Database Recovery (ADR)

- Introduced in SQL Server 2019
- Default for Azure SQL DB
  - Cannot be turned off
- Changes the way the transactions logs are read to be able to recover from long running transactions or just restore faster
- Uses a Persistent Version Store in Database rather than the Version Store in TempDB

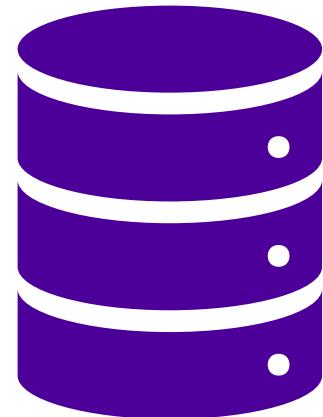
BEGIN TRAN 1A

User DB

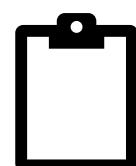


```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Sebastian'
```

tempdb



|           |    |     |    |
|-----------|----|-----|----|
| Sebastian | MA | Key | 1A |
|-----------|----|-----|----|



Persistent Version Store

|           |    |    |   |
|-----------|----|----|---|
| Sebastian | AZ | 1A | 1 |
|-----------|----|----|---|

```
SELECT FirstName,  
LastName,  
State
```

```
FROM Customer  
WHERE FirstName = 'Sebastian'
```

Version Store

Accelerated Database  
Recovery

# Lock Escalation

- Threshold at which SQL Server will convert lower granularity locks to a higher level in order to manage a small number of locks
  - 5,000 on a table or index for a single statement
  - Each additional 1,250 locks taken in a transaction

# Lock Granularity (from Microsoft)

| Resource        | Description   |
|-----------------|---|
| RID             | A row identifier used to lock a single row within a heap.   |
| KEY             | A row lock to lock a single row in a B-tree index.  |
| PAGE            | An 8 kilobyte (KB) page in a database, such as data or index pages.   |
| EXTENT          | A contiguous group of eight pages, such as data or index pages.   |
| HoBT            | A heap or B-tree. A lock protecting a B-tree (index) or the heap data pages in a table that doesn't have a clustered index. |
| TABLE           | The entire table, including all data and indexes.   |
| FILE            | A database file.  |
| APPLICATION     | An application-specified resource.  |
| METADATA        | Metadata locks.   |
| ALLOCATION_UNIT | An allocation unit.   |
| DATABASE        | The entire database.  |
| XACT            | Transaction ID (TID) lock used in Optimized Locking   |

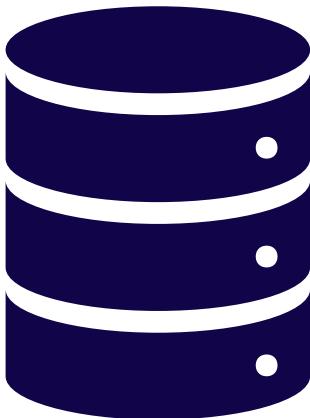
# Lock Escalation (cont'd)

- Row and key locks escalate to table locks, not page locks
- The query determines whether lock escalation is needed and not the isolation level (Erik Darling)

# Optimized Locking

- Changes to the locking mechanisms
  - Locks on the Transaction ID
  - Releases locks on other objects quickly
  - Minimizes lock escalation
- Accelerated Database Recovery is Required

User DB

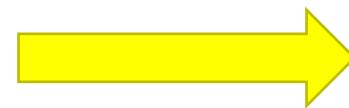
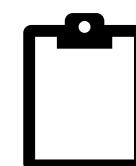
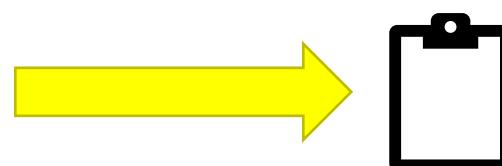


BEGIN TRAN 1A

```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Sebastian'
```



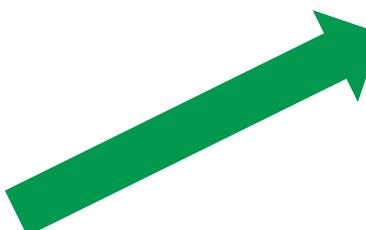
|           |    |     |         |
|-----------|----|-----|---------|
| Sebastian | MA | Key | Padlock |
|-----------|----|-----|---------|



Persistent Version Store

|           |    |    |   |
|-----------|----|----|---|
| Sebastian | AZ | 1A | 1 |
|-----------|----|----|---|

```
SELECT FirstName,  
LastName,  
State  
FROM Customer  
WHERE FirstName = 'Sebastian'
```



Optimized Locking -  
Reads

**Yeah, but that was just  
a SELECT statement...**

**Regular RCSI stuff.**

**Now what about writes?**



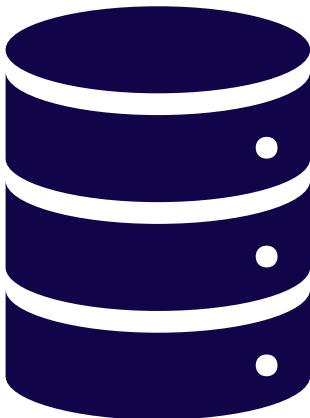
ON TOUR | NEW YORK

 **APASS**

# Transaction ID Locking

- As the lock is held on the Transaction ID, the next transaction can see when rows aren't affected and update them without waiting.

User DB

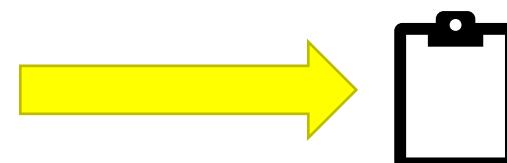


BEGIN TRAN 1A

```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Sebastian'
```



|           |    |                 |                 |
|-----------|----|-----------------|-----------------|
| Sebastian | MA | Locked (Orange) | Locked (Orange) |
| Deborah   | TN | Locked (Green)  | Locked (Green)  |

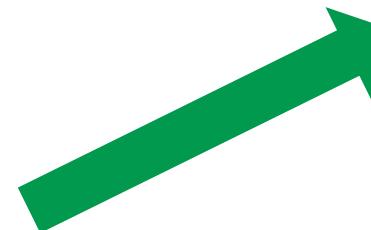
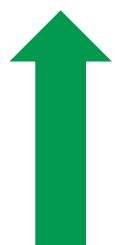


Persistent Version Store

|           |    |    |   |
|-----------|----|----|---|
| Sebastian | AZ | 1A | 1 |
| Deborah   | TN | 2A | 1 |

BEGIN TRAN 2A

```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Deborah'
```

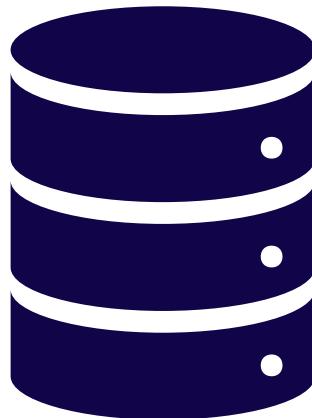


Optimized Locking -  
TID Locking

# Lock After Qualification (LAQ)

- RCSI is a requirement for this to work
- Looks to see if the previously committed version of the rows with TID locks would also be affected by the current transaction
  - If yes, transaction ***will wait*** for previous transaction to finish and include those columns as part of the change
  - If no, the transaction will update the rows that do match

User DB



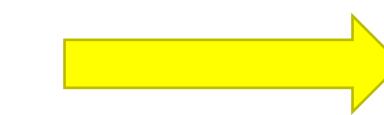
BEGIN TRAN 1A

```
UPDATE Customer  
SET State = 'GA'  
WHERE FirstName = 'Sebastian'
```



COMMIT TRAN 1A

|           |    |     |        |
|-----------|----|-----|--------|
| Sebastian | MA | Key | Locked |
|-----------|----|-----|--------|



BEGIN TRAN 2A

```
UPDATE Customer  
SET State = 'MA'  
WHERE FirstName = 'Sebastian'
```



| Persistent Version Store |       |         |         |       |
|--------------------------|-------|---------|---------|-------|
| FirstName                | State | Version | TransID | Order |
| Sebastian                | AZ    | 1A      | 1       |       |
| Sebastian                | GA    | 2A      | 1       |       |

Optimized Locking -  
LAQ

# Demos!



ON TOUR | NEW YORK

APASS

# Be aware of

- Version Store
  - Make sure there is enough space available
  - Asynchronous Cleaner
    - Data remains until it's no longer needed
  - When full, only reads can happen – no writes
  - Keep Transactions Fast

# Be aware of

- Lock After Qualification
  - Blocking and waits can still occur
  - Be aware of multiple transactions that may change data for the same set of records

# Recap



- Understand how RCSI & ADR are used for optimized locking
- Minimizes blocking but does not solve ALL situations
- Reduces the number of locks being held and the length they are held for
- Writes don't block Reads OR Writes

# Resources – Optimized Locking

- [SQL Server Transaction Locking and Row Versioning Guide](#)
- [Optimized Locking](#)
- [Bob Ward's Github - Optimized Locking Demos](#)
- [Isolation Level Locking \(Erik Darling\)](#)

# Resources - Accelerated Database Recovery

- [Soaring to New Heights with Accelerated Database Recovery \(John Morehouse\)](#)
- [Monitor and Troubleshoot Accelerated Database Recovery](#)
- [Why to Use Accelerated Database Recovery in SQL Server \(Luis Lema\)](#)
- [Constant Time Recovery in Azure SQL Database \(whitepaper\)](#)

# Thank you

Any Questions?

**Deborah Melkin**



[@dgmelkin.bsky.social](https://bsky.social/@dgmelkin)



[github.com/DebtheDBA](https://github.com/DebtheDBA)



[DebtheDBA.wordpress.com](https://debthedba.wordpress.com)

