



DelphiDay
italian conference

SQL SERVER 2025: What's new in the engine

Focus on Optimized Locking

Slide e demo: bit.ly/delphiday2025-sgovoni



SERGIO GOVONI

CENTRO SOFTWARE S.P.A.



segovoni.medium.com



bit.ly/sgovoni-MVP



twitter.com/segovoni



github.com/segovoni



linkedin.com/in/sgovoni



DelphiDay
italian conference

19-20 Giugno 2025
Piacenza



wintech
italia

OPEN-SOURCE PROJECTS

github.com/segovoni

SQL command-line utility

github.com/segovoni/sqlcmdcli

Alter column with dependencies

github.com/segovoni/sp_alter_column

Conference demos

github.com/segovoni/sql-server-demos



19-20 Giugno 2025
Piacenza





AGENDA

- SQL Server 2025: What's new in the engine
- Optimized locking
 - Key components
 - Underlying technologies
- How optimized locking works
- Demo



SQL Server 2025: What's new in the engine

1



SQL 2025: What's new in the engine

Improve Concurrency

Optimized Locking

Abort Query Hint

Tempdb resource governance

Accelerate Performance

IQP enhancements

Columnstore indexes

Query store on read replicas

Increase HADR

Reliable Failover for AGs

AG Tuning and diagnostics

Backup enhancements



Industry proven engine



40+ features inside the SQL Server engine



Security

- Security cache improvements
- OAEP support for encryption
- PBKDF password hashing
- Authentication using system-assigned managed identity
- Backup to URL with managed identity
- Managed identity support for EKM
- Entra logins with nonunique display names
- Custom password policy on Linux
- TDS 8.0/TLS 1.3 support for tools



Performance

- Optimized Locking
- Tempdb space resource governance
- ADR in tempdb
- Persisted stats for readable secondaries
- Change tracking cleanup
- Columnstore index maintenance
- CE feedback for expressions
- Optional parameter plans optimization
- DOP feedback on by default
- Optimized Halloween protection
- Query store for readable secondaries
- ABORT_QUERY_EXECUTION query hint
- Optimized sp_executesql
- Batch mode optimizations
- Remove In-Memory OLTP from a database
- tmpfs support for tempdb in Linux

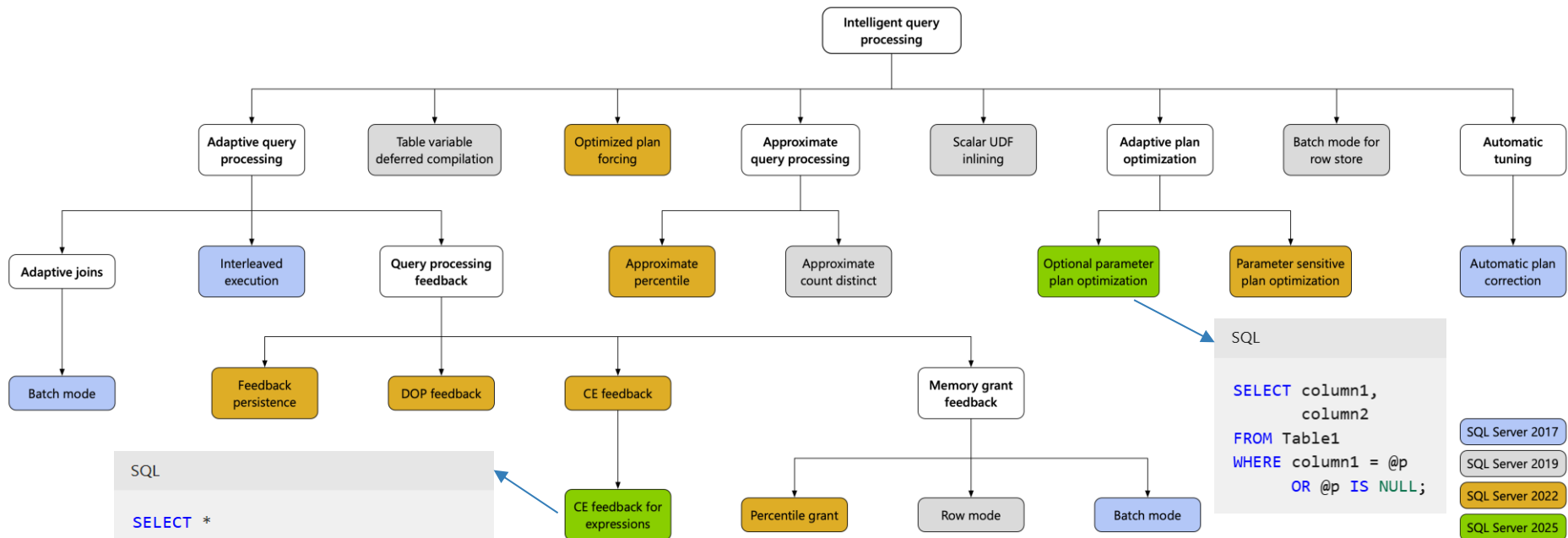


HADR

- Fast failover for persistent AG Health
- Async page request dispatching
- Improved health diagnostics
- Communication control flow tuning
- Switching to resolving state
- Remove listener IP address
- NONE for routing
- AG group commit waiting tuning
- Contained AG support for DAG
- DAG sync improvements
- Backups on secondary replicas
- ZSTD Backup compression



SQL 2025 Intelligent query processing



SQL Server 2017

SQL Server 2019

SQL Server 2022

SQL Server 2025



SQL 2025 Intelligent query processing

- Cardinality estimation feedback for expressions
- Optional parameter plan optimization (OPPO)
- DOP feedback enable by default
- Optimized Halloween protection
- Query Store for readable secondaries enable by default
- Block future execution of problematic queries
 - `ABORT_QUERY_EXECUTION`



Optimized Locking

2



Lock mode

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.



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. Please, verify your application before the activation of RCSI



How optimized locking works

3



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



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



Demo

3



SQL Server 2022 vs 2025

Results Messages

	spid	rt	LockCount
1	71	XACT	128
2	97	key/page	14165
3	71	key/page	18609
4	97	XACT	77

	spid	wait_type	WaitTime
1	97	PAGEIOLATCH_SH	5200
2	71	LCK_M_S_XACT_MODIFY	11299
3	97	PAGEIOLATCH_EX	26009
4	71	PAGEIOLATCH_EX	24360
5	71	PREEMPTIVE_HTTP_REQUEST	1500
6	71	PAGEIOLATCH_SH	4807

	counter_name	cntr_value	counter_time
1	Lock Memory (KB)	1024	2025-04-30 21:20:00.335530
2	Lock Memory (KB)	1120	2025-04-30 21:23:34.4508693

Results Messages

	spid	rt	LockCount
1	67	key/page	131460
2	64	key/page	74920

	spid	wait_type	WaitTime
1	64	PAGEIOLATCH_EX	5148
2	67	LCK_M_X	8559
3	67	PAGEIOLATCH_SH	1115
4	64	MEMORY_ALLOCATION_EXT	1071
5	67	PAGEIOLATCH_EX	10509
6	64	PAGEIOLATCH_SH	1918

	counter_name	cntr_value	counter_time
1	Lock Memory (KB)	880	2025-04-30 20:51:00.0316717
2	Lock Memory (KB)	3064	2025-04-30 20:53:52.1970706



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](#)



THANK YOU !