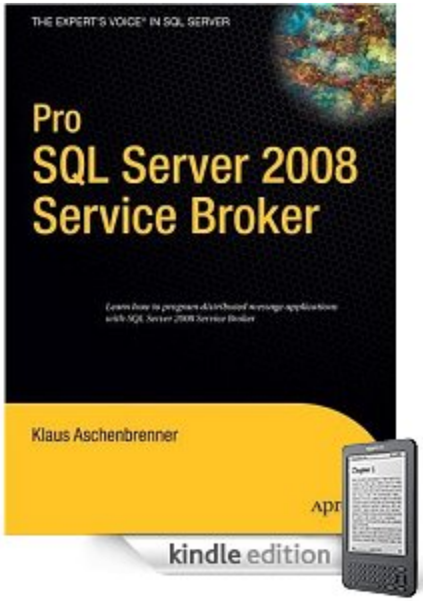
~~SQL Server TempDb~~
The public toilet of SQL Server

Klaus Aschenbrenner
Microsoft Certified Master SQL Server 2008
www.SQLpassion.at
Twitter: @Aschenbrenner

Bring your SQL Server installations to a new level of excellence!

About me

- Independent SQL Server Consultant
- International Speaker, Author
- SQL Server 2008 MCM
- „Pro SQL Server 2008 Service Broker“
- www.SQLpassion.at
- Twitter: @Aschenbrenner
- SQLpassion Academy
  - http://www.SQLpassion.at/academy



THE EXPERT'S VOICE® IN SQL SERVER

Pro
SQL Server 2008
Service Broker

Learn how to program distributed message applications with SQL Server 2008 Service Broker

Klaus Aschenbrenner

Apr

kindle edition

Agenda

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices

Agenda

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices

TempDb Overview

- Stores
  - User Objects
  - Internal Objects
  - Version Store (Row Versioning)
- One TempDb for the whole SQL Server Instance
  - Performance bottleneck by design
- Always recreated after restart
  - Built from model database
- Uses „Simple“ recovery model
  - No need for transaction log backups
- Only one filegroup – the PRIMARY filegroup

TempDb Internals

- Dropped and recreated when SQL Server is stopped and restarted
- Inherits all settings from the model database
  - MDF file of 8 MB
  - LDF file of 1 MB
  - Autogrowth is set to 10%
- Drop, Detach, Attach are not possbile

User Objects

- Local Temp Tables
  - Scoped to the session where you created it
  - Dropped after the session is closed
  - Prefix „#“
- Global Temp Tables
  - Scoped across all sessions
  - Dropped after the session is closed
  - Prefix „##“
- Table Variables
  - Caution: No statistics available!
- Tables returned in Table Valued Functions

Internal Objects

- Work tables for DBCC CHECKDB and DBCC CHECKTABLE
- Work tables for hash operations, such as joins and aggregations
- Work tables for processing cursors
- Work tables for processing Service Broker objects
- Work files needed for many GROUP BY, ORDER BY, UNION, SORT, and SELECT DISTINCT operations
- Work files for sorts that result from creating or rebuilding indexes (SORT_IN_TEMPDB)

Version Store

- Stores row-level versioning data
- 2 Types
  - Common Version Store
    - Triggers
    - Snapshot Isolation
    - Read Committed Snapshot Isolation
    - MARS (Multiple Active Result Sets)
  - Online Index Rebuild Version Store
    - Used for Online Index Rebuilds

Performance Counters

- Access Methods:Worktables Created/sec
  - Created for query spools, LOB variables, and cursors
- Access Methods:Workfiles Created/sec
  - Created by hashing operations
  - Store temporary results for hash and hash aggregates
- General Statistics:Temp Tables Creation Rate
  - Number of temp tables created/sec
- General Statistics:Temp Tables For Destruction
  - Number of temp tables waiting to be destroyed by the cleanup thread

Demo

Temp Tables/Table Variables

Agenda

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices

Version Store

- Needed for
  - Read Committed Snapshot Isolation Level
  - Snapshot Isolation Level
  - Triggers
  - Online Index Operations
  - MARS (Multiple Active Result Sets)
- sys.dm_tran_version_store
  - Returns all versions that must be currently stored to provide consistency across transactions

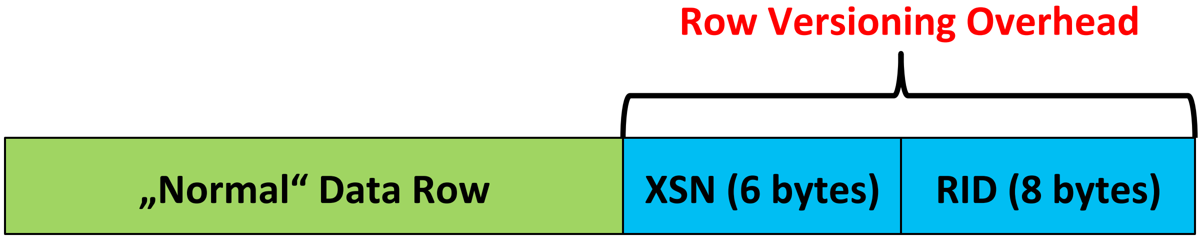
Append-Only Stores

- Common Version Store
  - Triggers
  - Snapshot Isolation
  - Read Committed Snapshot Isolation
  - MARS (Multiple Active Result Sets)
- Online Index Rebuild Version Store
  - Used for Online Index Rebuilds
- Each CPU scheduler has its own page(s) in the Version Store
  - Increases scalability

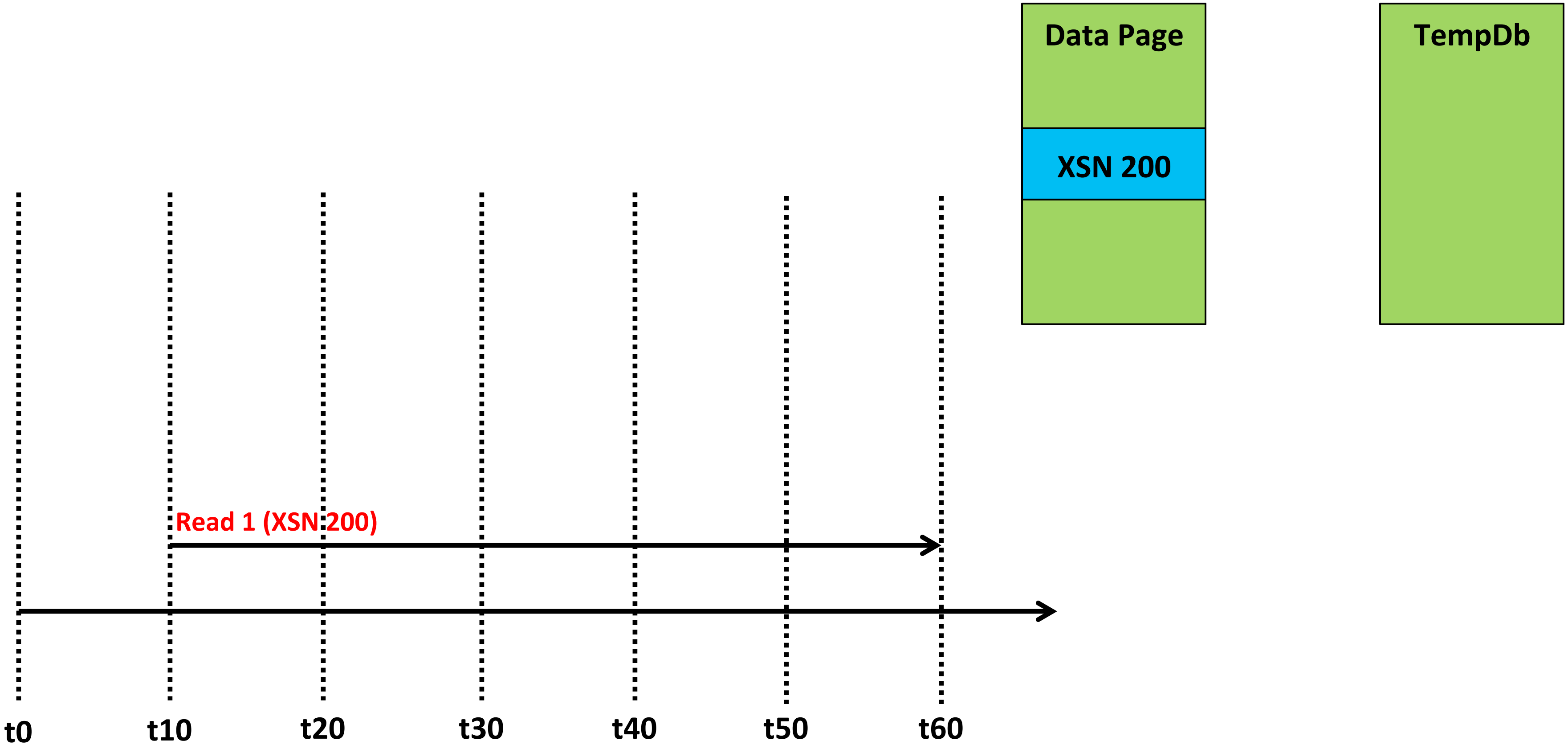
Version Store Overhead

- 14 bytes per row
- XSN
  - „Transaction Sequence Number“
  - 6 bytes
  - Used to chain multiple versions together
- RID
  - „Row Identifier“
  - 8 bytes
  - Locates the row version in TempDb
- Doesn‘t reduce the max row size of 8.060 bytes
  - Row is splitted across 2 pages

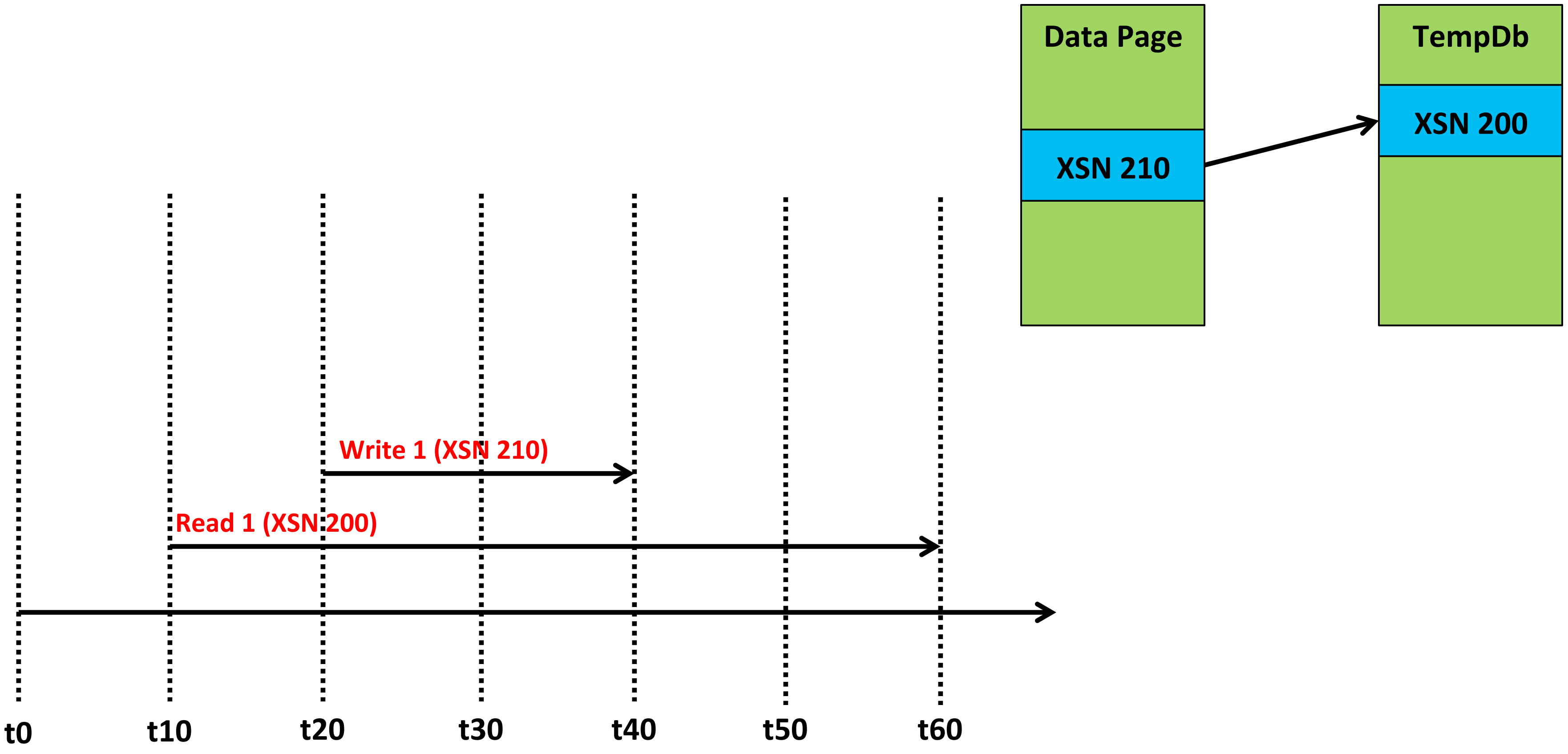
Row Version Overhead



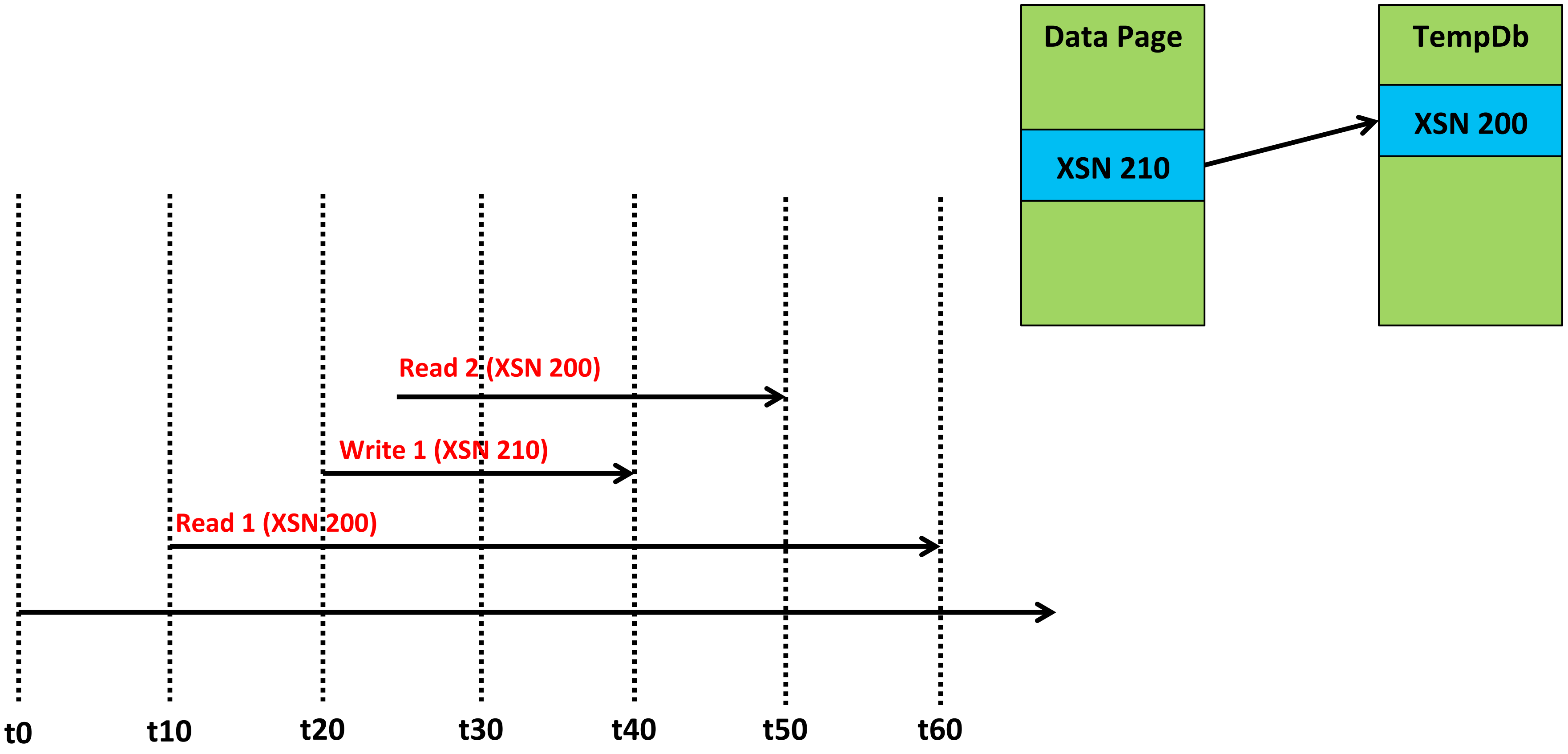
Version Store Internals



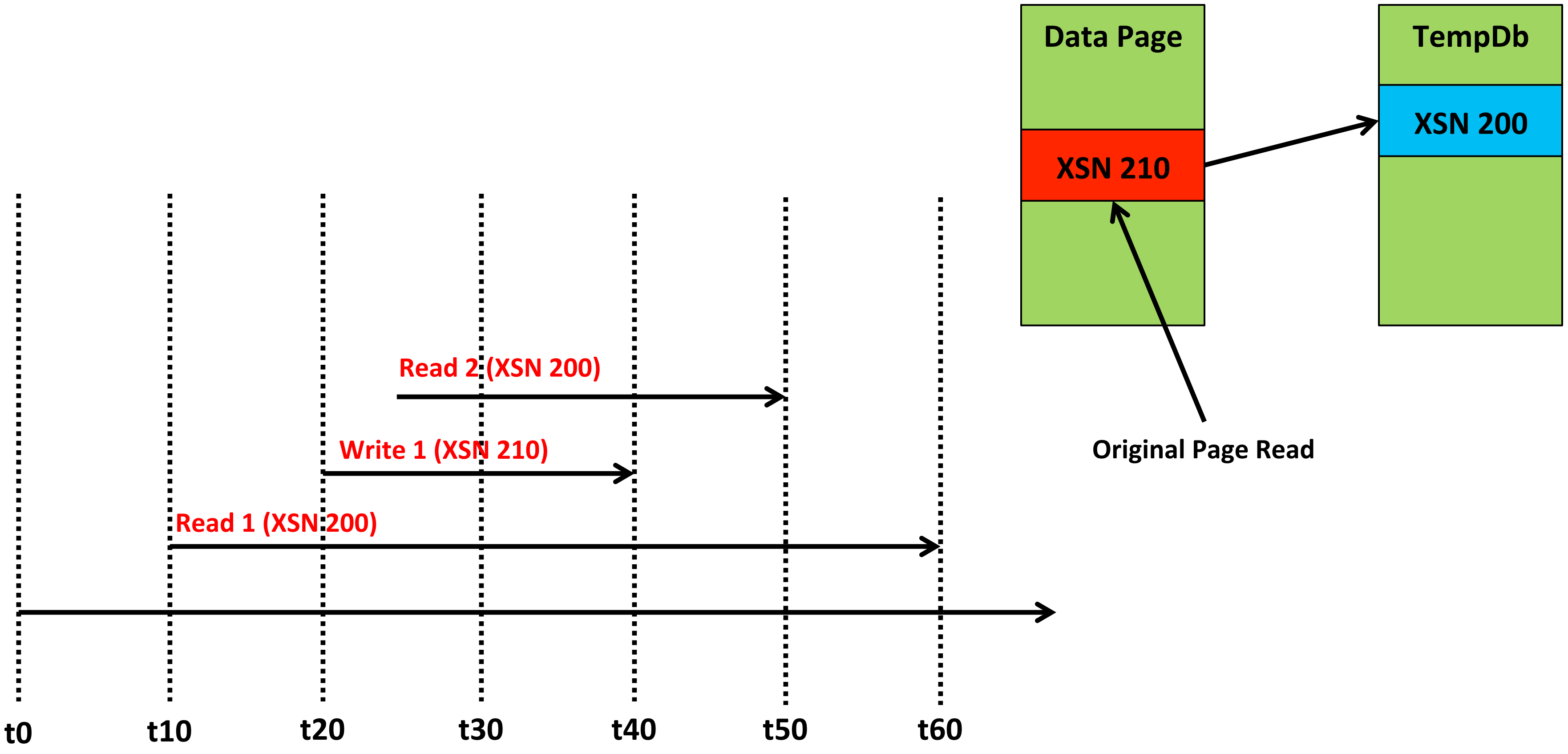
Version Store Internals



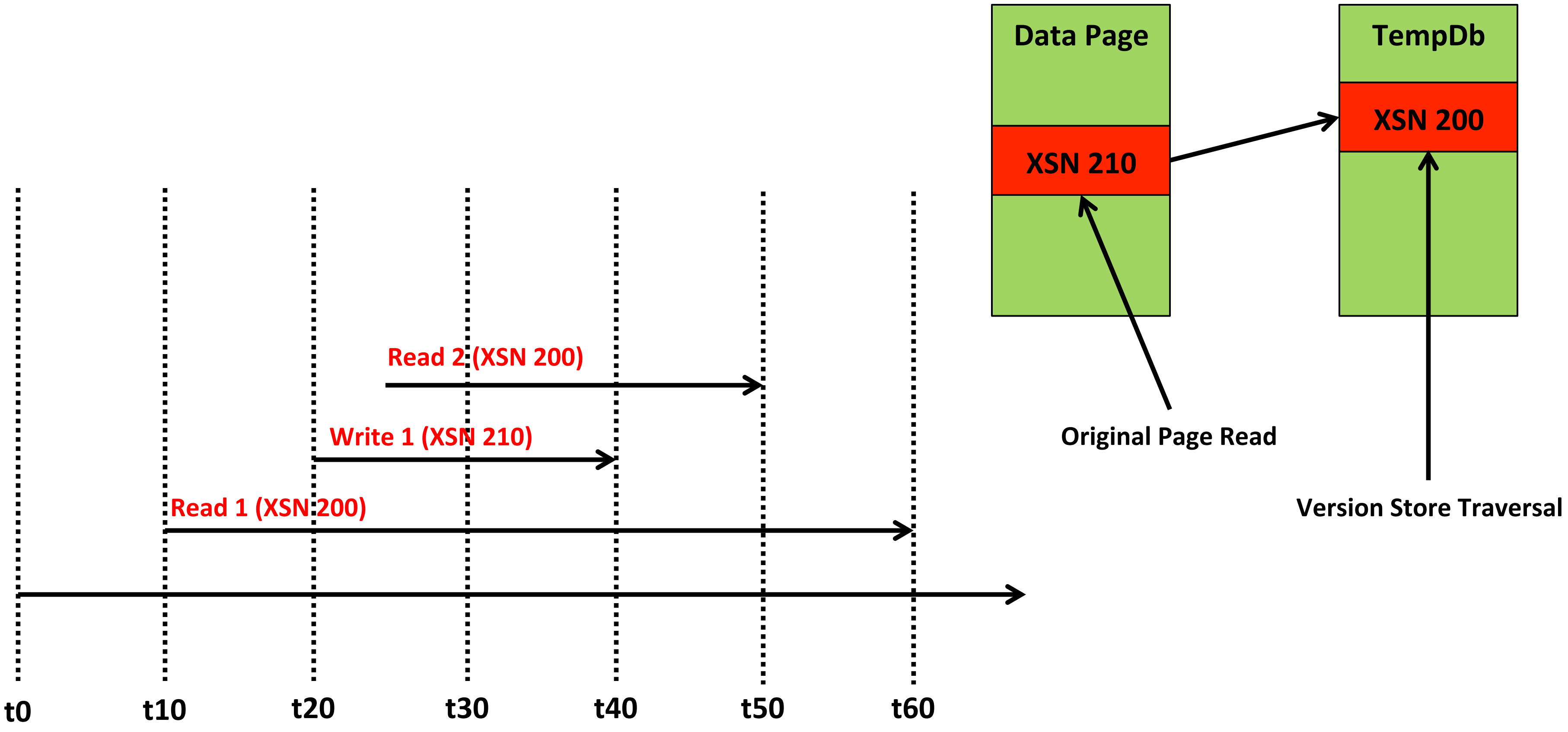
Version Store Internals



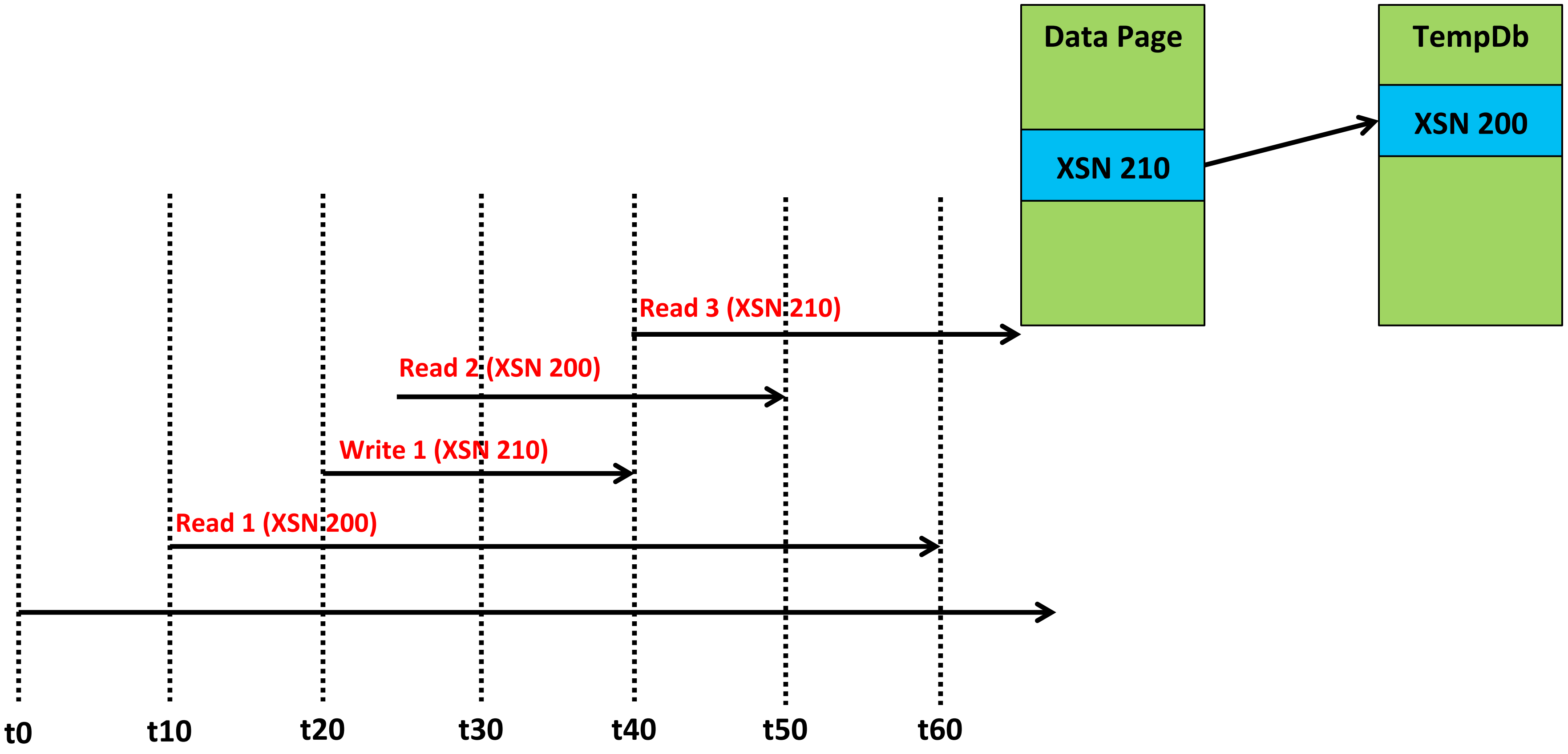
Version Store Internals



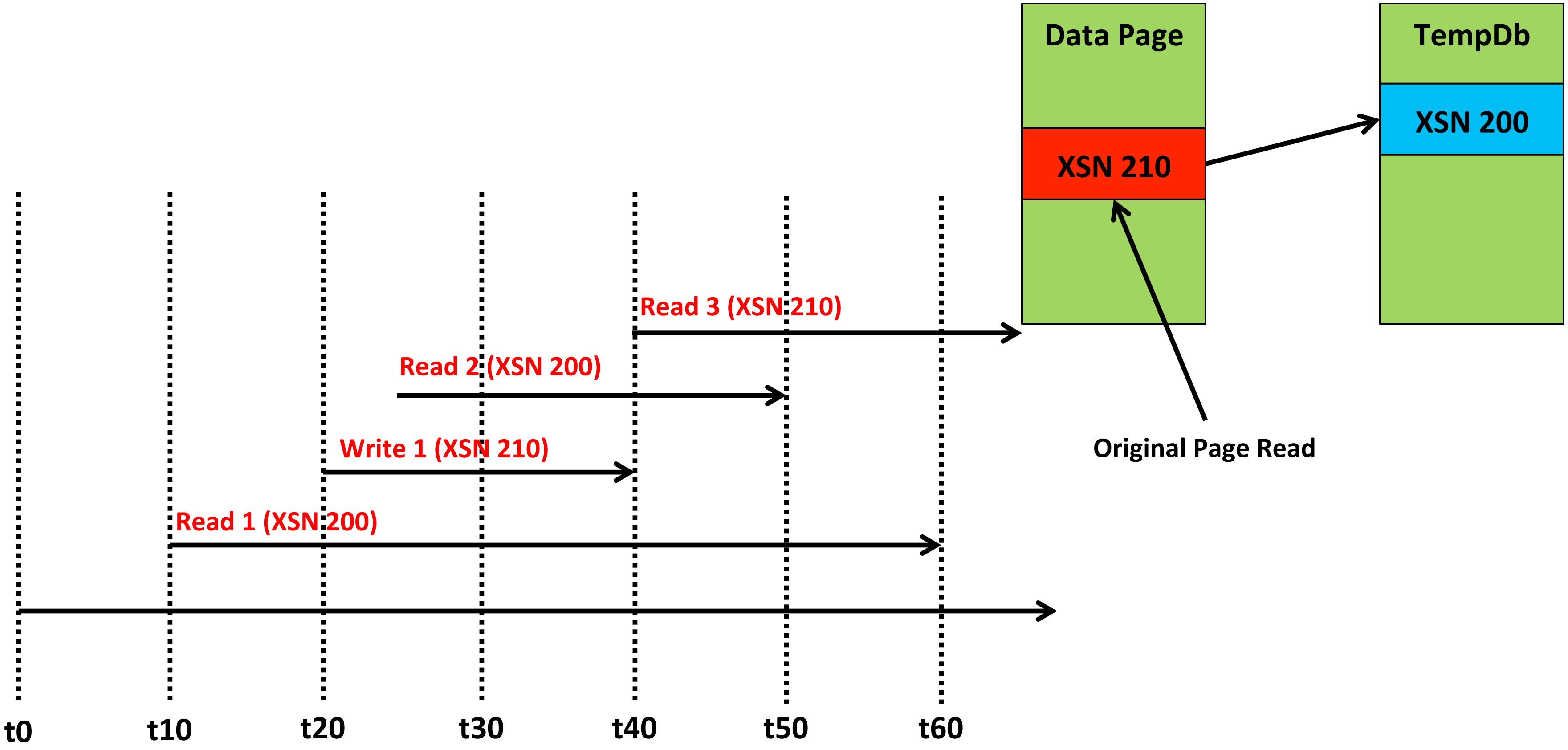
Version Store Internals



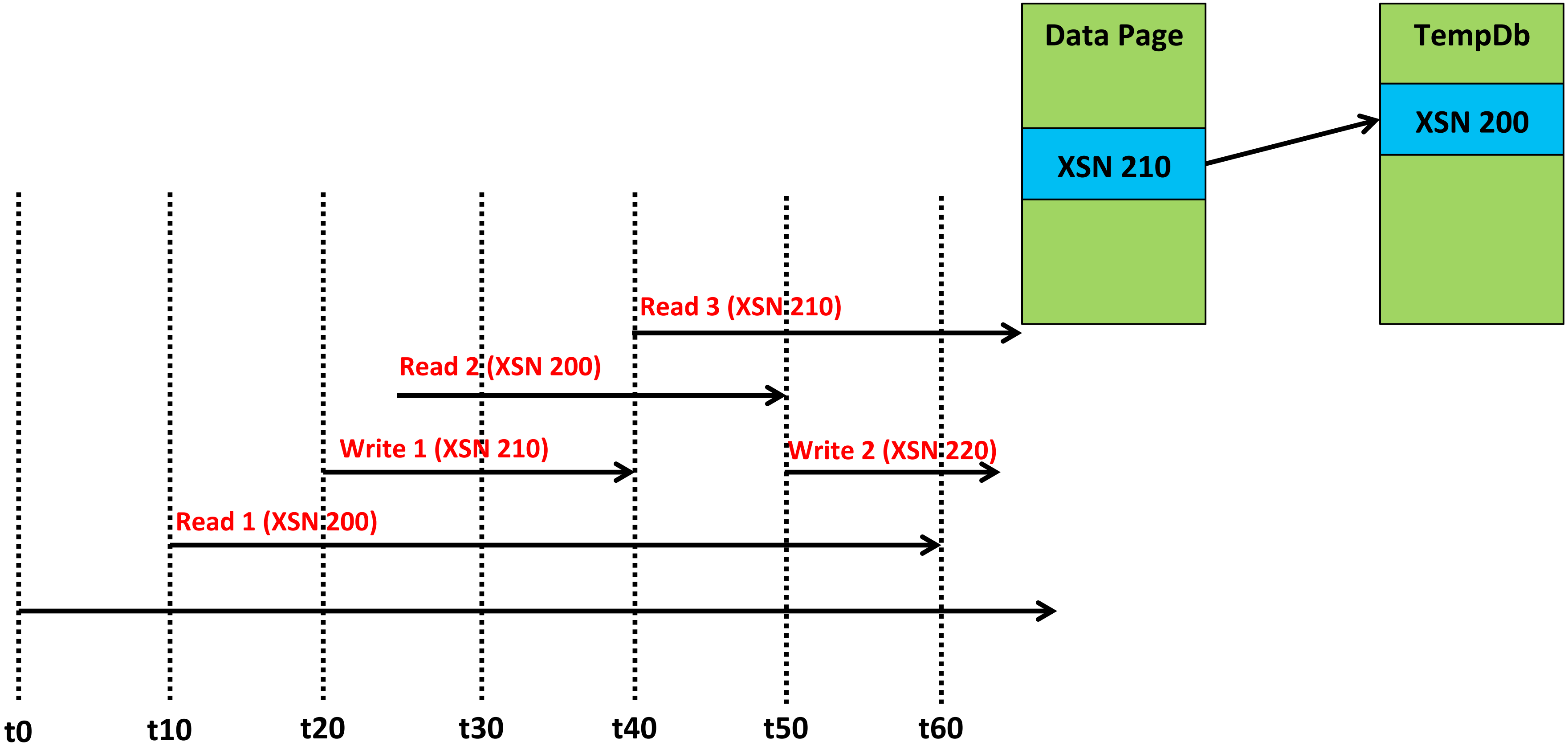
Version Store Internals



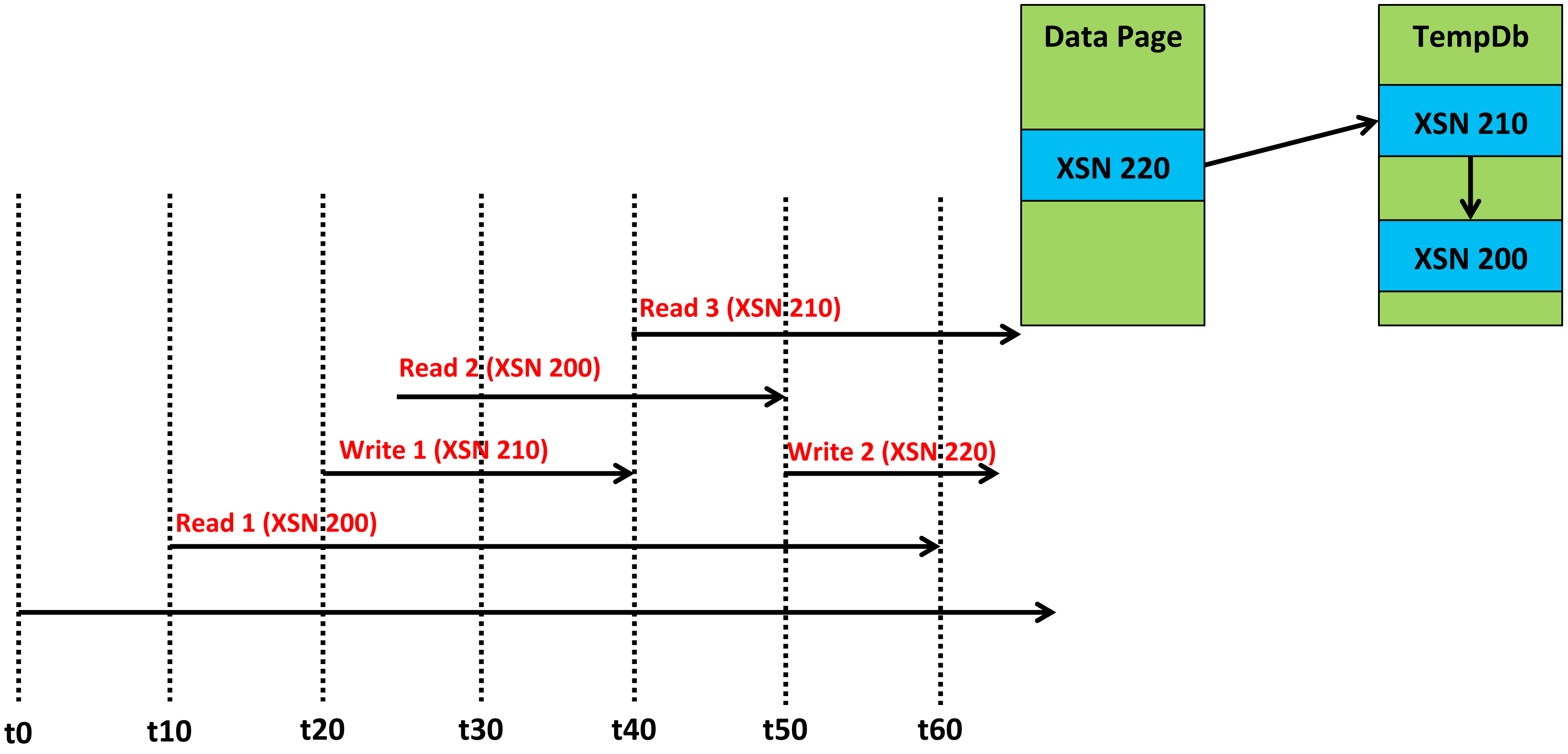
Version Store Internals



Version Store Internals



Version Store Internals



Row Version Pointers

Page	File	Slot	XSN
00000000	0000	0000	0d0100000000
88090000	0100	0000	0e0100000000
88090000	0100	0100	110100000000
88090000	0100	0200	120100000000

Current Version (Data Page)

Row Version Pointers

Page	File	Slot	XSN
0	0	0	269
2440	1	0	270
2440	1	1	273
2440	1	2	274

Current Version
(Data Page)

Demo

Version Store

Agenda

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices

Troubleshooting TempDb

- Latch Contention
- IO Performance Issues
- Space Issues
- Transaction Log Issues

Latch Contention

- Latch
  - Short term synchronization object used to protect physical pages
  - Every time when a page is accessed, a latch is needed
  - Other threads can‘t access the page in the mean time
- Big amount of creation and destruction of many objects
  - Temp Tables
  - Table Variables
- Can lead to Latch Contention
  - PFS Page
  - GAM/SGAM Page

Creating a temp table means...

1. Reading the SGAM page (2:1:3) to find a mixed extent with free space
   - SQL Server uses an exclusive latch on the SGAM page while updating the page
2. Reading the PFS page (2:1:1) to find a free page within the extent
   - SQL Server also uses an exclusive latch on the PFS page while updating the page
3. SQL Server will report a PAGELATCH wait type with the appropriate resource description
   - 2:1:3 for SGAM page
   - 2:1:1 for PFS page

Resolving Latch Contention 1/3

- Multiple TempDB data files
  - ¼ to ½ data files of the CPU cores you have (HT cores should be included in the calculation!)
  - Allocation of new objects is done round-robin between the data files
  - All data files must have the same size to be effective
- Don‘t use the Microsoft recommendation – 1:1 mapping between data files and CPU cores!
  - http://www.sqlskills.com/BLOGS/PAUL/post/A-SQL-Server-DBA-myth-a-day-(1230)-tempdb-should-always-have-one-data-file-per-processor-core.aspx

Demo

Resolving Latch Contention 1/3

Resolving Latch Contention 2/3

- Temporary Object Reuse
  - SQL Server can cache temporary objects instead of recreating them again and again
  - 1 IAM page and 1 Extent are cached
- Caching is possible when
  - Named constraints are not created
  - DDL statements are not used that effect the object like
    - CREATE INDEX
    - CREATE STATISTICS
  - Object is not created dynamically, e.g. through sp_executesql
  - Object is created inside another object
    - Stored Procedure, Trigger, UDF

Demo

Resolving Latch Contention 2/3

Resolving Latch Contention 3/3

- Trace Flag 1118
  - Introduced with SQL Server 2000
  - Not really needed in SQL Server 2008, because of already improved algorithm when allocating space in Mixed Extents
- Disables all Mixed Extent allocations in TempDb
- Every object that is created, is created in an Uniform Extent
  - Every temp table therefore needs at least 64kb storage
- Should be only enabled when you have contention on the SGAM page (2:1:3)

IO Performance Issues

- RAID 10 and/or SSD drives preferred
- PerfMon counters
  - Avg. Disk sec/Transfer
  - Avg. Disk sec/Read
  - Avg. Disk sec/Write
- SQL Server DMVs
  - sys.dm_io_virtual_file_stats

Space Issues

- sys.dm_db_file_space_usage
  - unallocated_extent_page_count
  - version_store_reserved_page_count
  - user_object_reserved_page_count
  - internal_object_reserved_page_count
- sys.dm_db_session_space_usage
  - TempDb usage for the current active sessions
- sys.dm_db_task_space_usage
  - TempDb usage for the current running tasks

Transaction Log Issues

- TempDb uses „Simple“ recovery model
  - When checkpoint occurs, log records from committed transactions are marked for reuse
  - No Transaction Log backups needed
- Be aware of the Transaction Log when you have long-running transactions
  - Checkpoint process can‘t mark them for reuse
  - Transaction Log grows
- Checkpoint process
  - Occurs about every minute
  - „Checkpoints“ user databases first
  - And Finally „checkpoints“ system databases
  - There could be some timing issues!

CHECKPOINT Issues

- Only occurs when Transaction Log is 70% full
  - Not influenced by the Recovery Interval setting
- Dirty pages are NOT flushed to disk
  - Crash Recovery is NOT run for TempDb
- Only Lazywriter flushes dirty pages from TempDb to disk
  - SSDs doesn‘t make too much sense for TempDb

Agenda

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices

Minimize TempDb Usage

- Sort data only if needed
  - Sort data on the client
- Don‘t use UNION or SELECT DISTINCT
- Keep transactions short
- Use proper indexing
- Consider creating a permanent work table
- Avoid aggregating excessive amounts of data
- Avoid Hash Joins
- Avoid SORT_IN_TEMPDB when creating or rebuilding an index

File Placement

- Put TempDb on its own physical drive
  - No compete with other IO activity
  - No other databases!
- Put it onto the fastest IO subsystem
  - Avoid RAID 5, because of poor write performance
  - SSD drives preferred
  - RAID 10

Initial Sizing and Autogrowth

- Standard (3 MB) is too less
- Autogrowth is at 10%
  - Contributes to physical file fragmentation
  - Maybe timeouts because queries have to wait until Autogrowth completes
- Use Instant File Initialization
- Don‘t shrink TempDb
- Divide TempDb into multiple files
  - By default: one physical file for MDF and LDF
  - Will reduce Latch Contention
  - Each physical file must have the same identical size

Summary

- TempDb Introduction
- Version Store
- Troubleshooting TempDb
- Best Practices