SQL Day POLAND SPRING 2015

Strategic Sponsor

Microsoft

Gold Sponsors

PYRAMID ANALYTICS

KRAJOWY REJESTR DŁUGÓW www.krd.pl

devart Database Interaction Solutions

Silver Sponsors

Profisee®

SQL EXPERT.pl

CREDIT SUISSE

Technical Partners

Microsoft

PTI POLSKIE TOWARZYSTWO INFORMATYCZNE ODDZIAŁ GÓRNOŚLĄSKI

Academic Partners

Wyższa Szkoła Zarządzania i Bankowości w Krakowie

Wyższe Szkoły Bankowe

Media Partners

GEEKCLUB.pl wss.pl

ERP24.pl

4programmers.net

IT professional

IN FACT PR

itnews24

virtual-it.pl

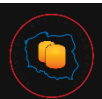
IT WORKS POLSKA

VirtualStudy

MAGAZYN IT PL

Made in Wro Do IT here!

ERP-view.pl Portal rozwiązań IT w biznesie

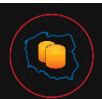


SQL Day

In-Memory OLTP Internals

Łukasz Grala

Architekt Platformy Danych i Rozwiązań Business Intelligence
MVP SQL Server



Łukasz Grala

MVP SQL Server | MCT | MCSE

- Architekt i trener - Data Platform & Business Intelligence Solutions
- Prelegent na licznych konferencjach
- Wykładowca i autor publikacji
- Doktorant Politechnika Poznańska – Wydział Informatyki (Bazy i hurtownie danych, Machine Learning, Mining of Massive Datasets)
- Posiada liczne certyfikaty
- Od 2010 roku co roku wyróżniany nagrodą MVP w kategorii SQL Server
- Lider PLSSUG Poznań (lukasz.grala@plssug.org.pl)

SQL EXPERT.pl

Agenda

- Co to jest In-Memory OLTP?
- Wersjonowanie, CAW w BW-Tree, Pamięć
- Checkpoint od środka
- Dziennik transakcji i RESTORE

SQL Server 2014 In-Memory Technology

In-Memory Technologies

In-Memory OLTP

- 5-30X performance gain for OLTP integrated into SQL Server

In-Memory DW

- 5-100X performance gain and high data compression
- Updatable and clustered

SSD Bufferpool Extension

- 4-10X of RAM and up to 3X performance gain transparently for apps

Applicable to

Transactional workloads:
Concurrent data entry, processing and retrieval

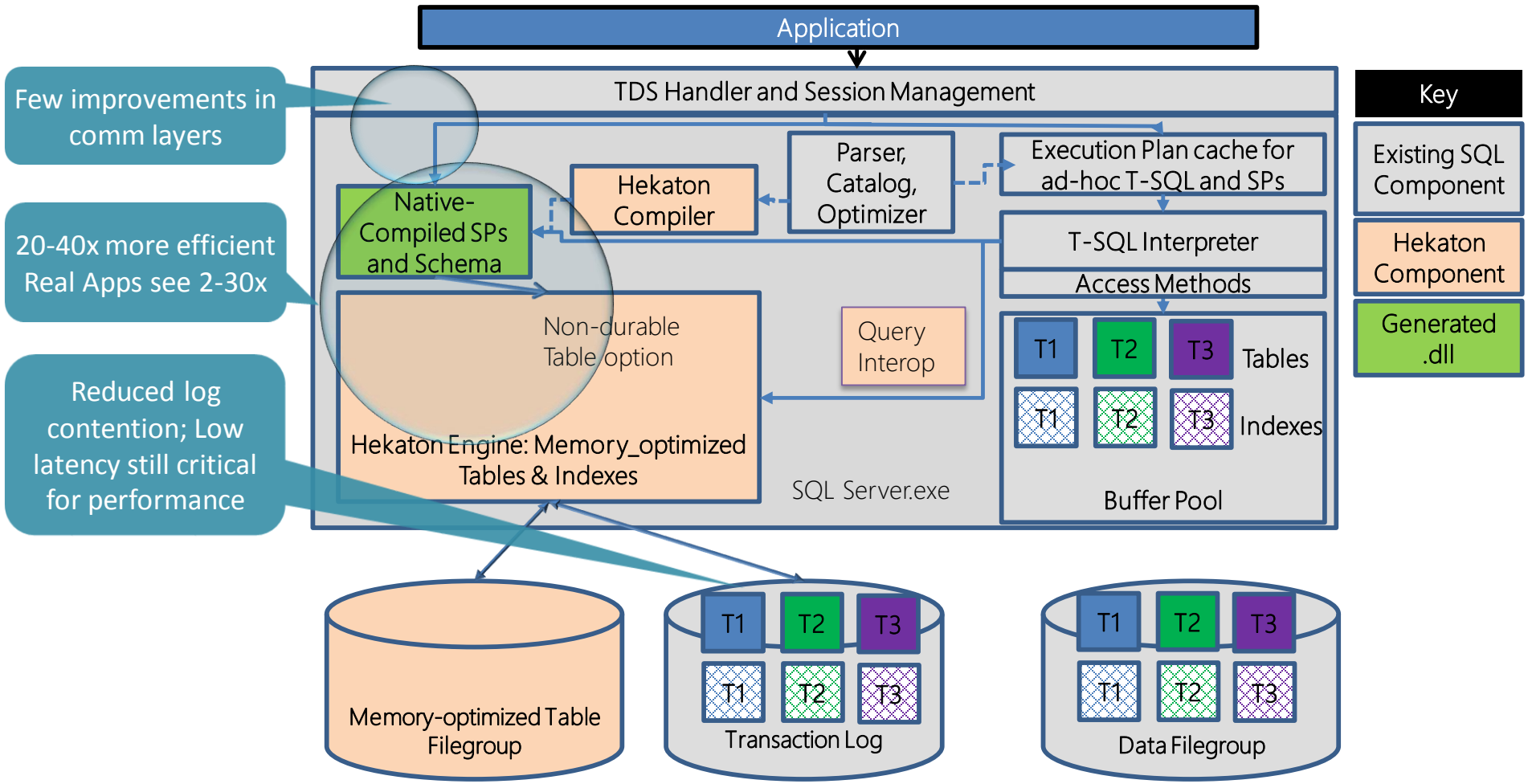
Applicable to

Decision support workloads:
Large scans and aggregates

Applicable to

Disk-based transactional workloads:
Large working (data)set

In-Memory OLTP Architecture



Create Table DDL

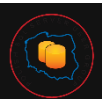
```
CREATE TABLE [Customer](
    [CustomerID] INT NOT NULL
            PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 1000000),
    [Name] NVARCHAR(250) NOT NULL
      INDEX [IName] HASH WITH (BUCKET_COUNT = 1000000),
    [CustomerSince] DATETIME NULL
)
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
```

Hash index

Secondary indexes are specified inline

This table is memory optimized

This table is durable



Create Procedure DDL

```
CREATE PROCEDURE [dbo].[InsertOrder] @id INT, @date DATETIME
  WITH
    NATIVE_COMPILATION,
    SCHEMABINDING,
    EXECUTE AS OWNER
AS
BEGIN ATOMIC
  WITH
(TRANSACTION
   ISOLATION LEVEL = SNAPSHOT,
 LANGUAGE = 'us_english')

  -- insert T-SQL here
END
```

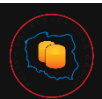
This proc is natively compiled

Native procs must be schema-bound

Execution context is required

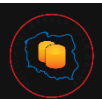
Atomic blocks
- Create a transaction if there is none
- Otherwise, create a savepoint

Session settings are fixed at create time

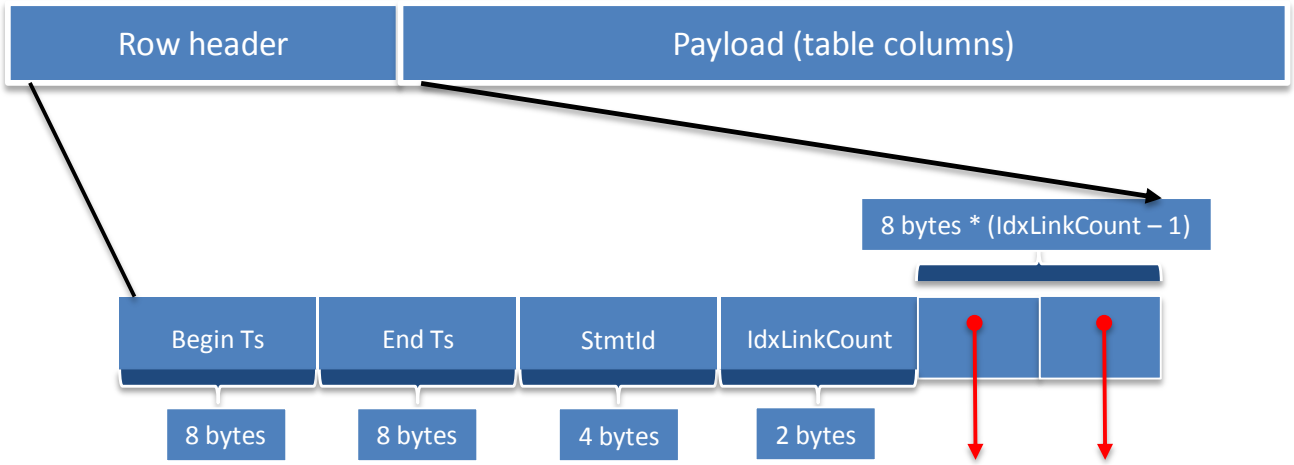


Overview In-Memory

DEMO 1



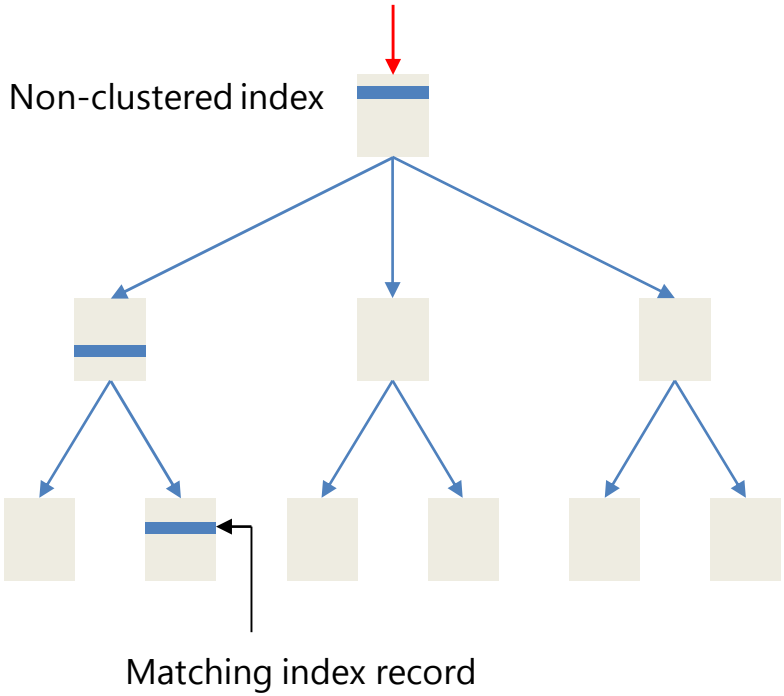
Memory-optimized table: Row format

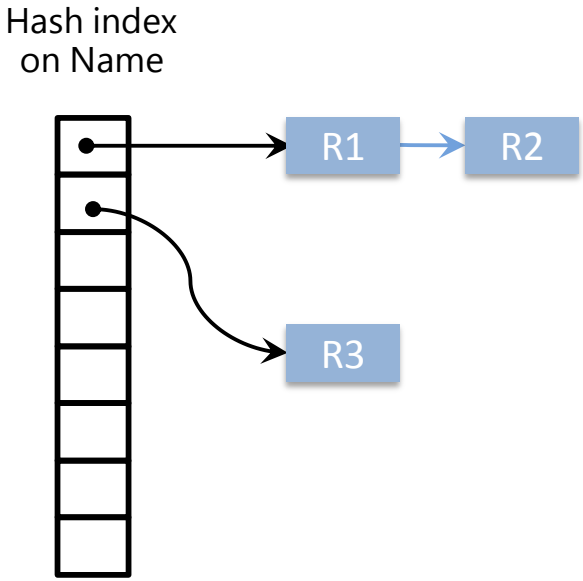


Key Points

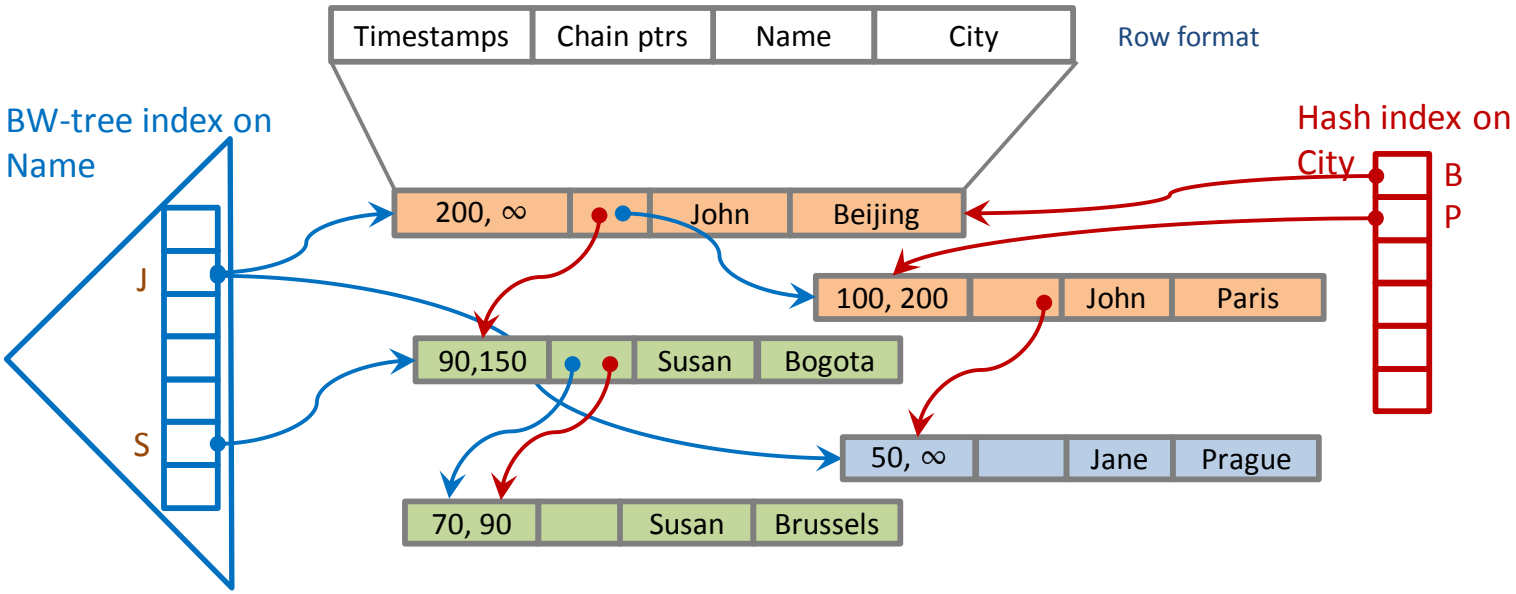
- Begin/End timestamp determines row's validity
- No data or index page; just rows
- Row size limited to 8060 bytes to allow data to be moved to disk-based table
- Not every SQL table schema is supported

Key lookup: B-tree vs. memory-optimized table



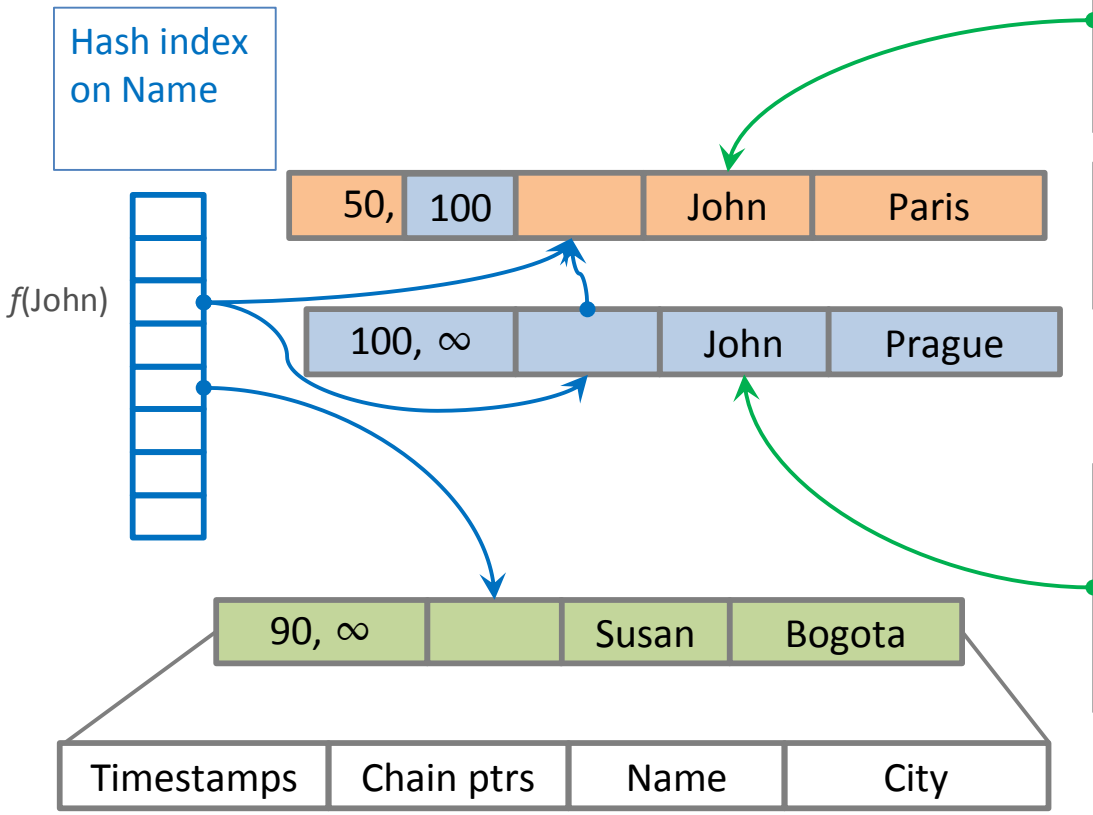


Memory-Optimized Tables



- Rows are multi-versioned, each has a valid time range indicated by two timestamps
- A version is visible if transaction read time falls within the version's valid time
- Row can be part of multiple indexes, but there is only a single copy of the row
  - Hash indexes accelerate point-lookup
  - BW-tree is a lock-free b-tree (range index) invented by MSR
  - Indexes are not stored on disk

Direct Access, Multi-Version, Lock-Free Transactions



Transaction **99**: Running compiled query

SELECT City WHERE Name = 'John'

Simple hash lookup returns direct pointer to 'John' row

Background operation will unlink and deallocate the old 'John' row after transaction 99 completes.

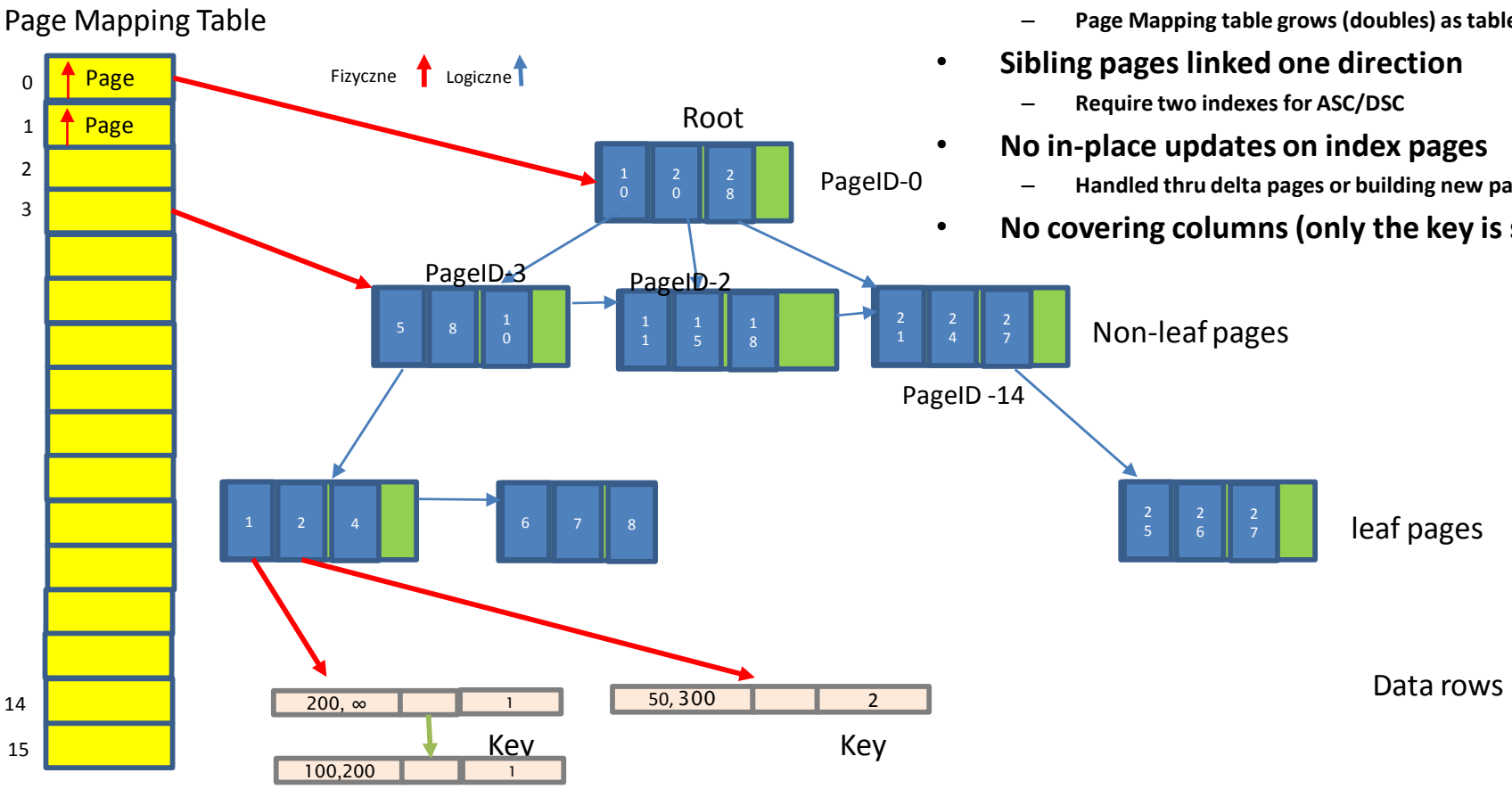
Transaction **100**:

UPDATE City = 'Prague' where Name = 'John'

No locks of any kind, no interference with transaction 99

BW Tree

- **Page size- up to 8K. Sized to the row**
- **Logical pointers**
  - **Indirect physical pointers through Page Mapping table**
  - **Page Mapping table grows (doubles) as table grows**
- **Sibling pages linked one direction**
  - **Require two indexes for ASC/DSC**
- **No in-place updates on index pages**
  - **Handled thru delta pages or building new pages**
- **No covering columns (only the key is stored)**

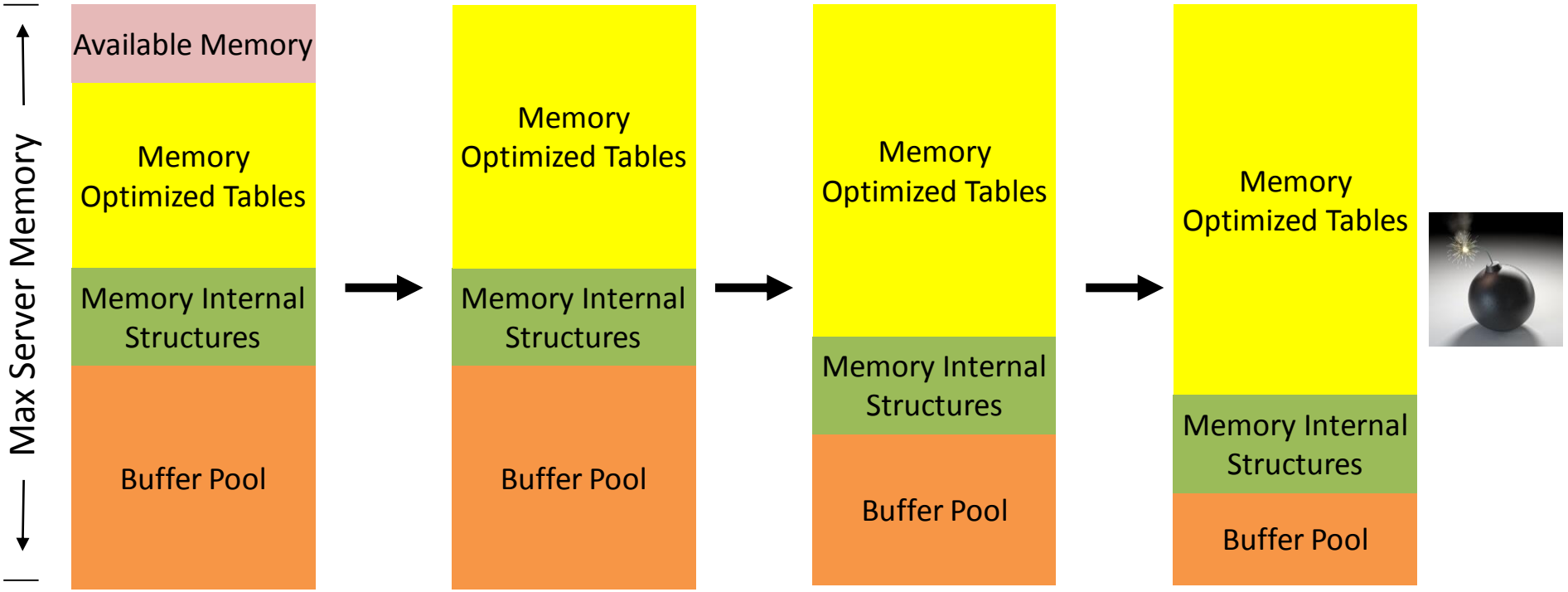


Example: Write conflict

Time	Transaction T1 (SNAPSHOT)	Transaction T2 (SNAPSHOT)
1	BEGIN	
2		BEGIN
3		UPDATE t SET c1='bla' WHERE c2=123
4	UPDATE t SET c1='bla' WHERE c2=123 (write conflict)	

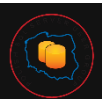
First writer wins

Memory Challenge

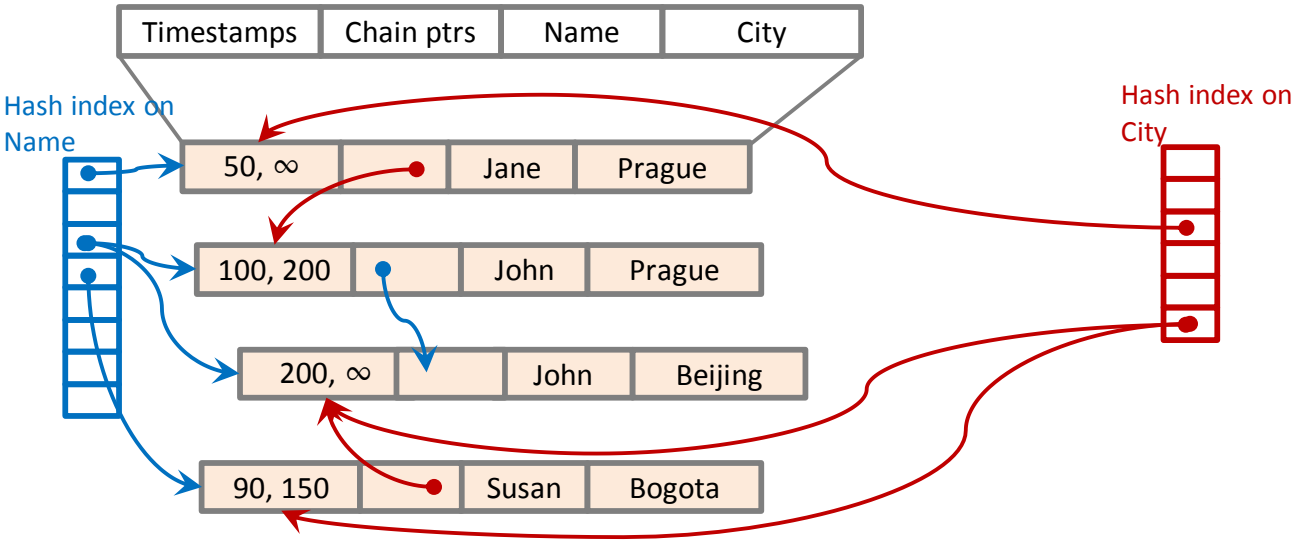


Memory Challenge and Resolution

- Memory Pressure can lead to
  - Other SQL workloads slow down to unacceptable performance
  - DML Transactions on memory-optimized tables can fail due to OOM
- Possible Solutions
  - Allow Paging - Rejected. SQL Server does not support it.
  - Provision and Manage Memory appropriately - Recommended
- Steps
  - Estimate Memory needed for Memory Optimized Tables - guidance provided
  - Limit memory consumption by binding database to Resource Pool
  - Tools to monitor memory consumption for preemptive strike



Memory Optimized Tables: Garbage Cleanup



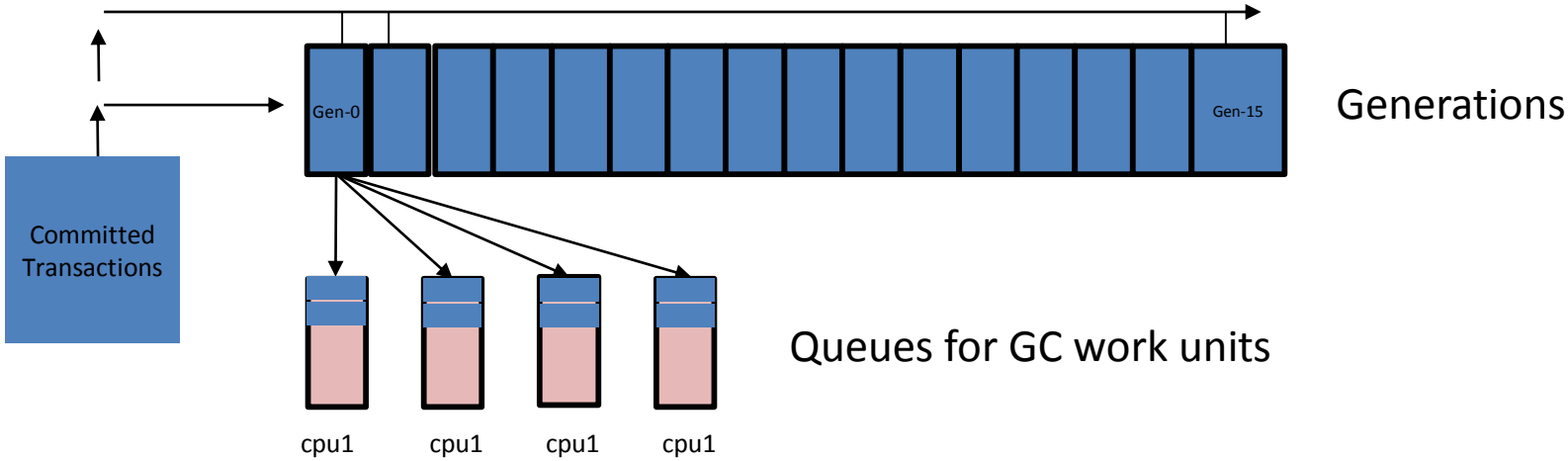
T250: lowest Active Transaction Timestamp

Select * from <table> where name = 'John'

Automatic Garbage Collection kicks in

Memory Optimized Tables: Garbage Cleanup

- Design
  - Non-blocking, Cooperative, Efficient, Responsive, Scalable
  - Active transactions work cooperatively and pick up parts of GC work
  - A dedicated system thread for GC



Garbage collection

Stale Row Versions

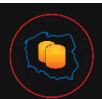
- Updates, deletes, and aborted insert operations create row versions that (eventually) are no longer visible to any transaction
- Slow down scans of index structures
- Create unused memory that needs to be reclaimed (i.e. Garbage Collected)

Garbage Collection (GC)

- Analogous to version store cleanup task for disk-based tables to support Read Committed Snapshot (RCSI)
- System maintains ‘oldest active transaction’ hint

GC Design

- Non-blocking, Cooperative, Efficient, Responsive, Scalable
- A dedicated system thread for GC
- Active transactions work cooperatively and pick up parts of GC work

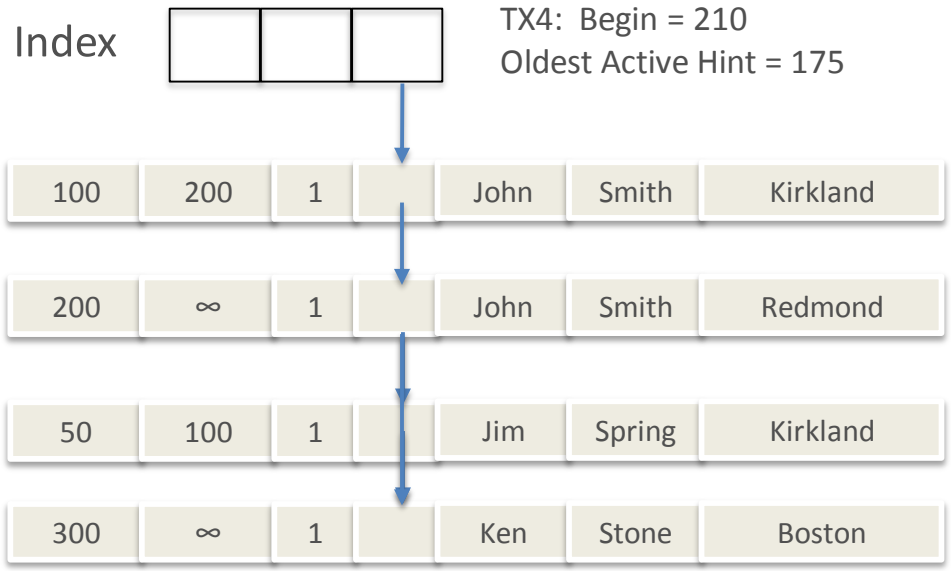


Cooperative garbage collection

Key Points

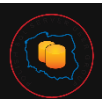
- Scanners can remove expired rows when found
- Offloads work from GC thread
- Ensures that frequently visited areas of the index are cleaned regularly

- A row needs to be removed from all indexes before memory can be freed
- Garbage collection is most efficient if all indexes are frequently accesses



GC

DEMO 2



On-disk storage

Filestream is the underlying storage mechanism

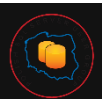
Checksums and single-bit correcting ECC on files

Data files

- ~128MB in size, write 256KB chunks at a time
- Stores only the inserted rows (i.e. table content)
- Chronologically organized streams of row versions

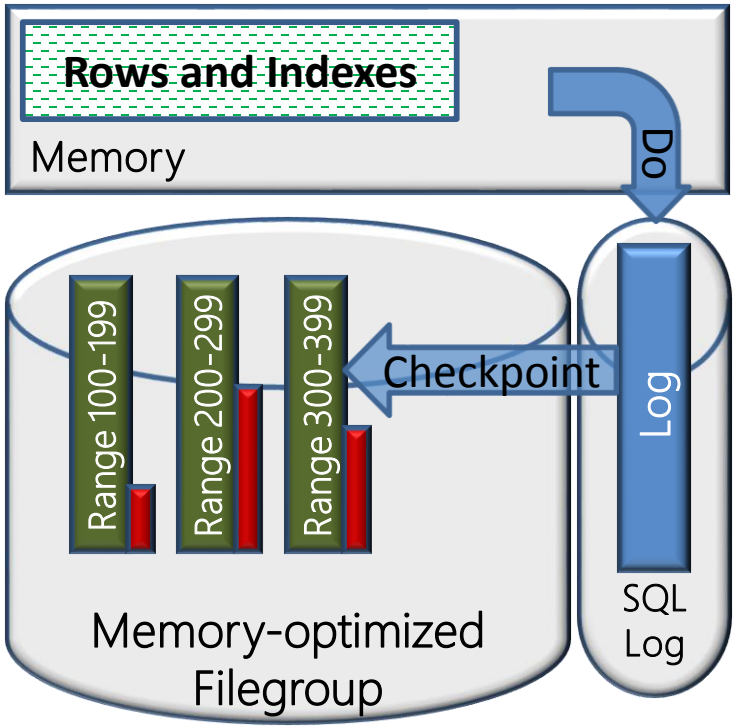
Delta files

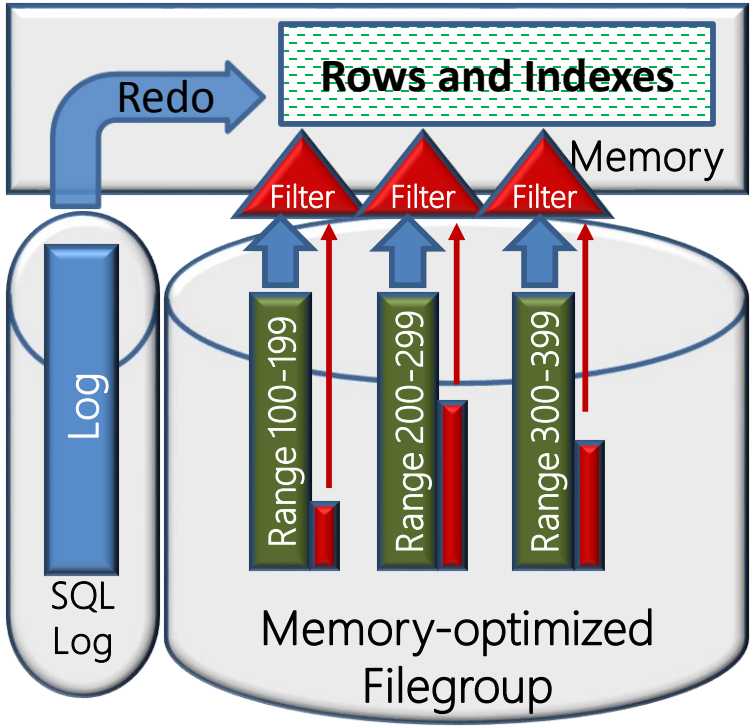
- File size is not constant, write 4KB chunks at a time.
- Stores IDs of deleted rows



Logging, Checkpoint and Recovery

Checkpoint





Checkpoints: Organized for fast, parallel recovery, require only sequential IO

Background process merges "depleted" checkpoint files to reduce storage needs

Key | Data file with rows inserted in timestamp range a-b | Delta file with IDs of deleted rows

Durability (all IM OLTP related IOs are sequential)

Memory-optimized tables can be durable or non-durable

- Default is 'durable'
- Non-durable tables are useful for transient data ("relational" cache scenario)

Committed transactions are logged

- A common system bottleneck; SSD or Fusion-IO recommended
- Delayed durability support – define at the SP/Transaction level

Durable tables are persisted in memory-optimized filegroup

- Storage used for memory-optimized has a different access pattern (Sequential IO) than for disk tables
- Recommend SSD or fast (15K e.g.) SAS drives
- Filegroup can have multiple containers (volumes) to aid in parallel recovery; recovery typically happens at the speed of IO

Storage in Memory-Optimized Filegroup

Filestream is the underlying storage mechanism

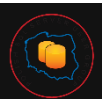
- Checksums and single-bit correcting ECC on files

Data files

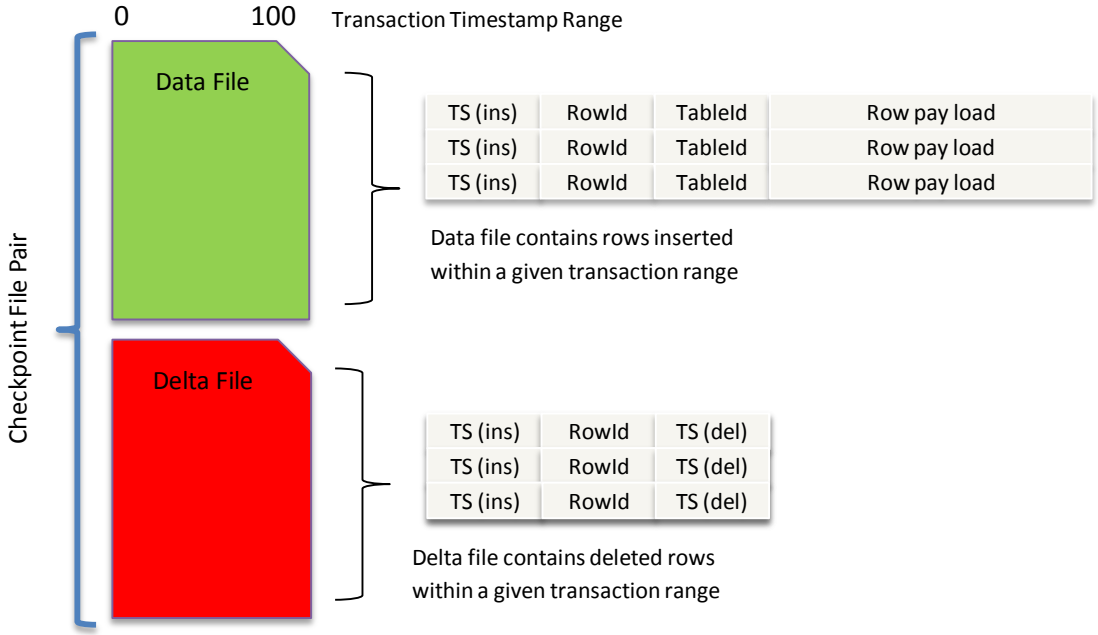
- For machines <=16GB memory – size 16MB. For larger machines – 128MB
- Writes in 256KB chunks at a time
- Stores only the inserted rows (i.e. table content)
- Chronologically organized streams of row versions

Delta files

- For machines <=16GB memory – size 1MB. For larger machines – 8MB
- Writes in 4KB chunks at a time.
- Stores IDs of deleted rows

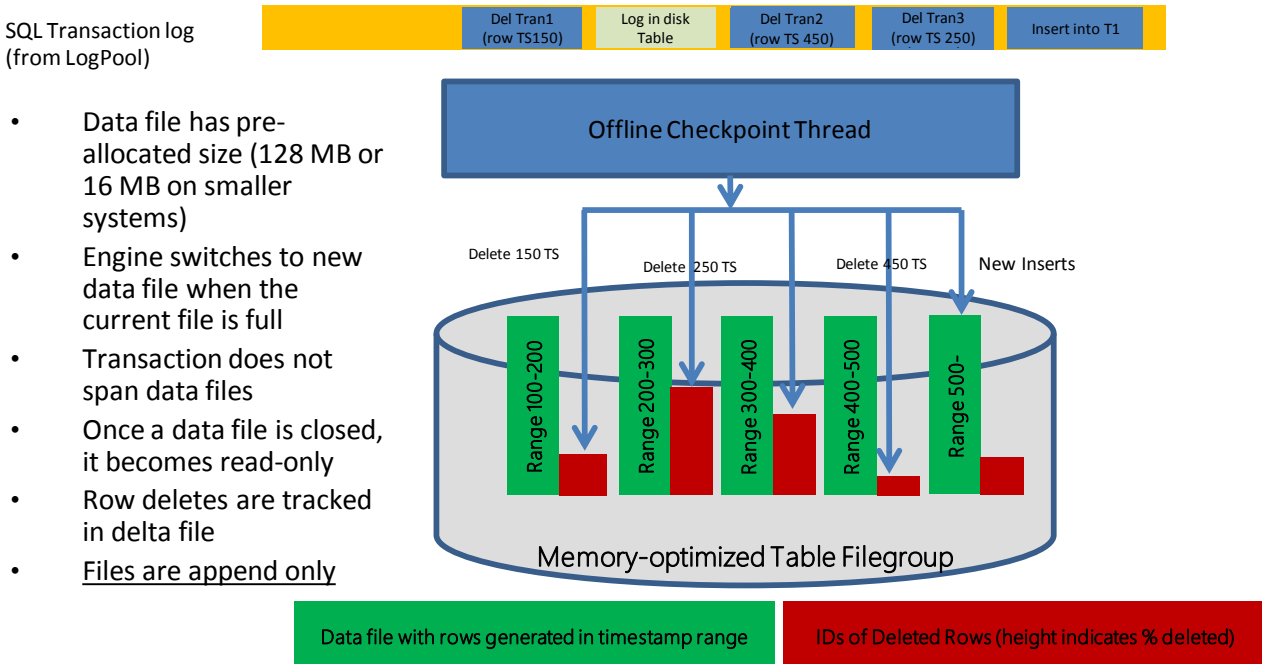


Storage: Data and Delta Files



Populating Data/Delta files

- Data file has pre-allocated size (128 MB or 16 MB on smaller systems)
- Engine switches to new data file when the current file is full
- Transaction does not span data files
- Once a data file is closed, it becomes read-only
- Row deletes are tracked in delta file
- Files are append only



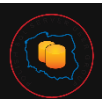
Merge Operations

What is a Merge Operation?

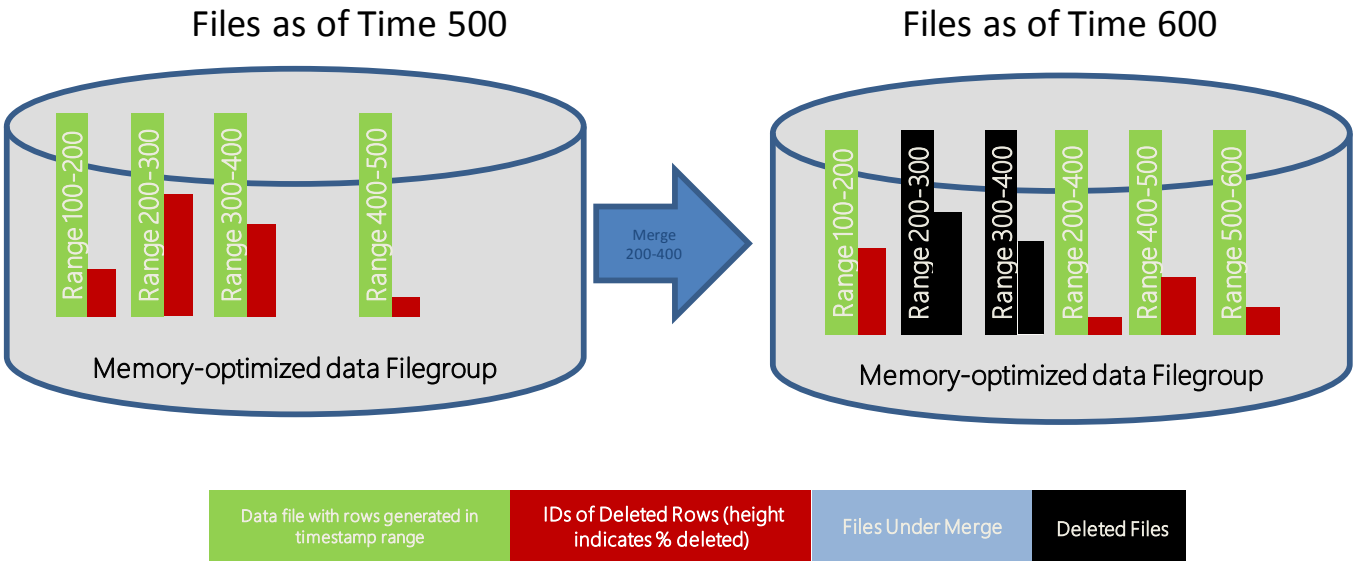
- Merges one or more adjacent data/delta files pairs into 1 pair
- Need for Merge
- Deleting rows causes data files to have stale rows
- DMV: *sys.dm_xtp_checkpoint_files* can be used to find inserted/deleted rows and freespace
- Benefits of Merge

Reduces storage (i.e. fewer data/delta files) required to store active data rows

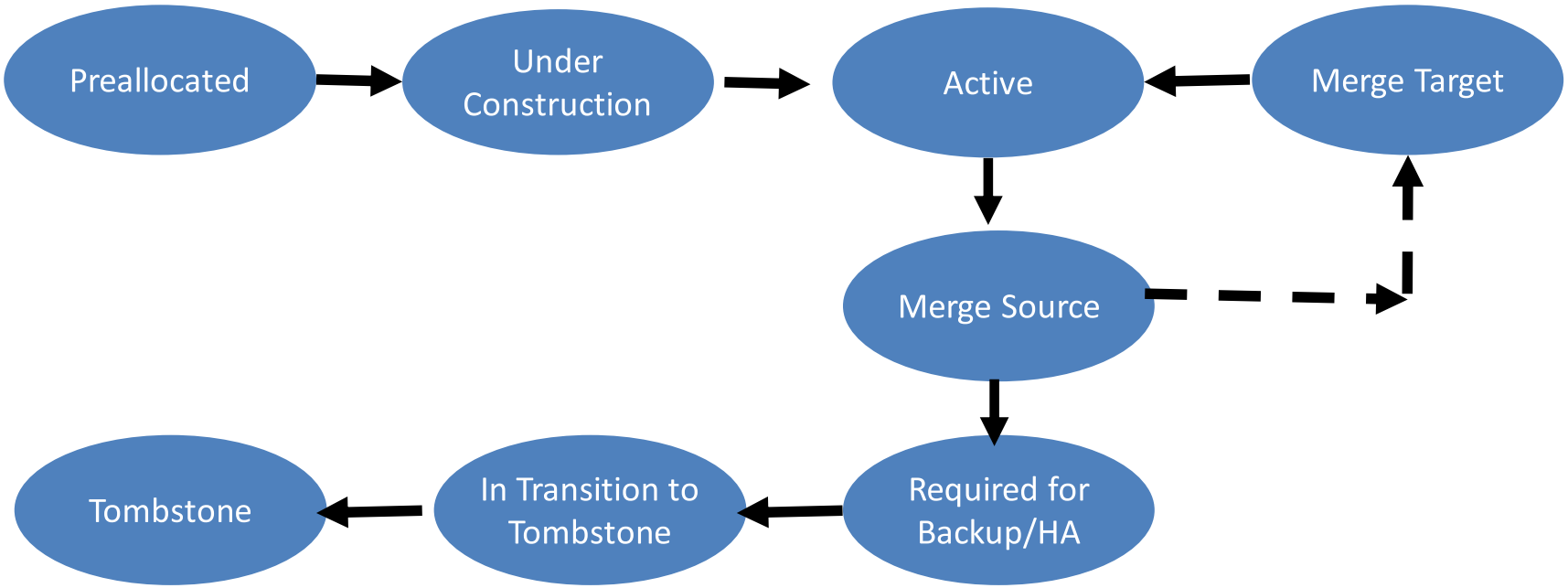
- Improves the recovery time as there will be fewer files to load
- Merge is a (non-blocking) background operation
- Merge does not block concurrent deletes in the affected file pairs



Merge Operation

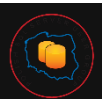


Data/Delta File State Transitions



Checkpoint

DEMO 3



Logging for memory-optimized tables

Uses SQL transaction log to store content

Each HK log record contains a log record header followed by opaque memory optimized-specific log content

All logging for memory-optimized tables is logical

- No log records for physical structure modifications.
- No index-specific / index-maintenance log records.
- No UNDO information is logged

Recovery Models

All three recovery models are supported

Logging for Memory-Optimized Tables

Uses SQL transaction log to store content

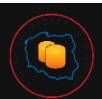
- Each HK log record contains a log record header followed by opaque memory optimized-specific log content.

All logging for memory-optimized tables is logical

- No log records for physical structure modifications.
- No index-specific / index-maintenance log records.
- No UNDO information is logged

  - Recovery Models

- All three recovery models (Simple, Full, Bulk) are supported



Backup for memory-optimized tables

Integrated with SQL Database Backup

- Memory-Optimized file group is backed up as part SQL Server database backup
- Existing backup scripts work with minimal or no changes
- Transaction log backup includes memory-optimized log records

Not supported

Differential backup

Backup for Memory-Optimized Tables

Integrated with SQL Database Backup

- Memory-Optimized file group is backed up as part SQL database backup
- Differential backup supported – only new data files since last full backup
- Piecemeal backups supported

Existing backup scripts work with minimal or no changes

Transaction log backup includes memory-optimized log records transparently

Recovery for Memory-Optimized Tables

Analysis Phase

- Finds the last completed checkpoint

Data Load

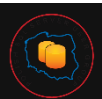
- Load from set of data/delta files from the last completed checkpoint
- Parallel Load by reading data files using 1 thread / file and 1 thread/container for delta files

Redo phase to apply tail of the log

- Apply the transaction log from last checkpoint
- Concurrent with REDO on disk-based tables

No UNDO phase for mem-opt tables

- Since only committed transactions are logged



Backup–With Memory-Optimized Tables

Data/Delta File	SQL 2014
Precreated	Empty file
Under Construction	Empty file
Active	Used bytes
Merge Source	Used bytes
Merge Target	Used bytes
Required for Backup/HA	Used bytes
In transition to Tombstone	Empty file
Tombstone	Skipped

Log

DEMO 4

Pytania?

Kontakt:

lukasz@grala.biz

lukasz.grala@plssug.org.pl

lukasz@sqlexpert.pl

Blogi:

blog.sqlexpert.pl (en/pl)

SQLResearch.com (pl)

SQL EXPERT.pl

SQL Day POLAND SPRING 2015

Strategic Sponsor

Microsoft

Gold Sponsors

PYRAMID ANALYTICS

KRAJOWY REJESTR DŁUGÓW www.krd.pl

devart Database Interaction Solutions

Silver Sponsors

Profisee®

SQL EXPERT.pl

CREDIT SUISSE

Technical Partners

Microsoft

PTI POLSKIE TOWARZYSTWO INFORMATYCZNE ODDZIAŁ GÓRNOŚLĄSKI

Academic Partners

Wyższa Szkoła Zarządzania i Bankowości w Krakowie

Wyższe Szkoły Bankowe

Media Partners

GEEKCLUB.pl wss.pl

ERP24.pl

4programmers.net

IT professional

IN FACT PR

itnews24

virtual-it.pl

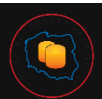
IT WORKS POLSKA

VirtualStudy

MAGAZYN IT PL

Made in Wro Do IT here!

ERP-view.pl Portal rozwiązań IT w biznesie



SQL Day