Will my workload run faster with In-Memory OLTP?

Ned Otter | SQL Strategist

How to evaluate if In-Memory OLTP is right for your workload

How to evaluate if In-Memory OLTP is right for your workload

What is the *impact* of deploying In-Memory OLTP?

About me

MCSE Data Platform

Passionate about SQL Server

Obsessed with In-Memory

KEY TAKEAWAYS

Workload evaluation
Use cases
Architecture

Unsupported

Replication publisher/distributor
Partitioning
Event Notifications
DDL triggers

Unsupported

CDC
Compression
Containment
MERGE target
Database snapshots
TRUNCATE

Unsupported

Change tracking
DDL in transactions
Linked servers
Mirroring
Cross-database queries/transactions
FILESTREAM

Unsupported

Filtered indexes
“clustered” indexes
FULLTEXT
INCLUDE
Indexed views

FILESTREAM

Enabling FS is *NOT* required to use In-Memory!

FILESTREAM

File management provided by in-mem engine

FILESTREAM

File management provided by in-mem engine

FILESTREAM feature not supported for in-mem tbls

FILESTREAM

File management provided by in-mem engine

FILESTREAM feature not supported for in-mem tbls

FILESTREAM feature IS supported for on-disk tbls

Unsupported data types

datetimeoffset**	rowversion
geography	xml
geometry	sql_variant**
hierarchyid	UDT

SQL 2017

SQL 2017

- > 8 indexes per table

SQL 2017

- **> 8 indexes per table**
- **NC index rebuild speed improvement**

SQL 2017

- > 8 indexes per table
- NC index rebuild speed improvement
- Computed columns (plus indexes)

SQL 2017

- > 8 indexes per table
- NC index rebuild speed improvement
- Computed columns (plus indexes)
- sp_rename (tables and native modules)

SQL 2017

- > 8 indexes per table
- NC index rebuild speed improvement
- Computed columns (plus indexes)
- sp_rename (tables and native modules)
- sp_spaceused

SQL 2017

- > 8 indexes per table
- NC index rebuild speed improvement
- Computed columns (plus indexes)
- sp_rename (tables and native modules)
- sp_spaceused
- parallel REDO for AGs

SQL 2017 – native compilation

- CASE
- JSON
- CROSS APPLY

Environments

AZURE

LINUX

In-Mem improvements in SQL 2016

ALTER TABLE
Foreign keys
Collations
NO data limit per DB (editions*)
Per-container checkpoint thread
Query store

SEQUENCE
CHECK/DEFAULT
DML triggers
Row level security
Temporal tables
Nested NC procedures

In-Mem improvements in SQL 2016

Native inline TVFs
Scalar UDFs
Parallel plans for interop
TDE
MARS support

ALTER PROCEDURE
Parallel scan for HASH idx
UNIQUE indexes
Parallel plans for heaps

WHICH TABLES?

Migration

GOT A LOTTA LATCHES?

Migration

GOT A LOTTA LATCHES?
TOP WAITS = PAGELATCH_XX?

WAIT STATS

WRITELOG
IO_COMPLETION
ASYNC_NETWORK_IO
SOS_SCHEDULER_YIELD

WAIT STATS

PAGELATCH_EX
PAGELATCH_SH
WRITELOG

Evaluating tables to migrate

Data/index compression?

Deciphering Editions in SQL 2016/SP1+

Edition	Max In-Mem/CCI
Enterprise	OS limit/OS limit
Standard	32 GB/32 GB
Web	16 GB/16 GB
Express	352 MB/352 MB

Deciphering Editions in SQL 2016/SP1+

Edition	Max In-Mem/CCI
Enterprise	OS limit/OS limit
Standard	32 GB/32 GB
Web	16 GB/16 GB
Express	352 MB/352 MB

per db

per instance

Standard Edition (32 GB)

Feature	Per db	Per instance
In-Memory data		

Standard Edition (32 GB)

Feature	Per db	Per instance
In-Memory data	Yes	No

Standard Edition (32 GB)

Feature	Per db	Per instance
In-Memory data	Yes	No
Columnstore	???	???

Standard Edition (32 GB)

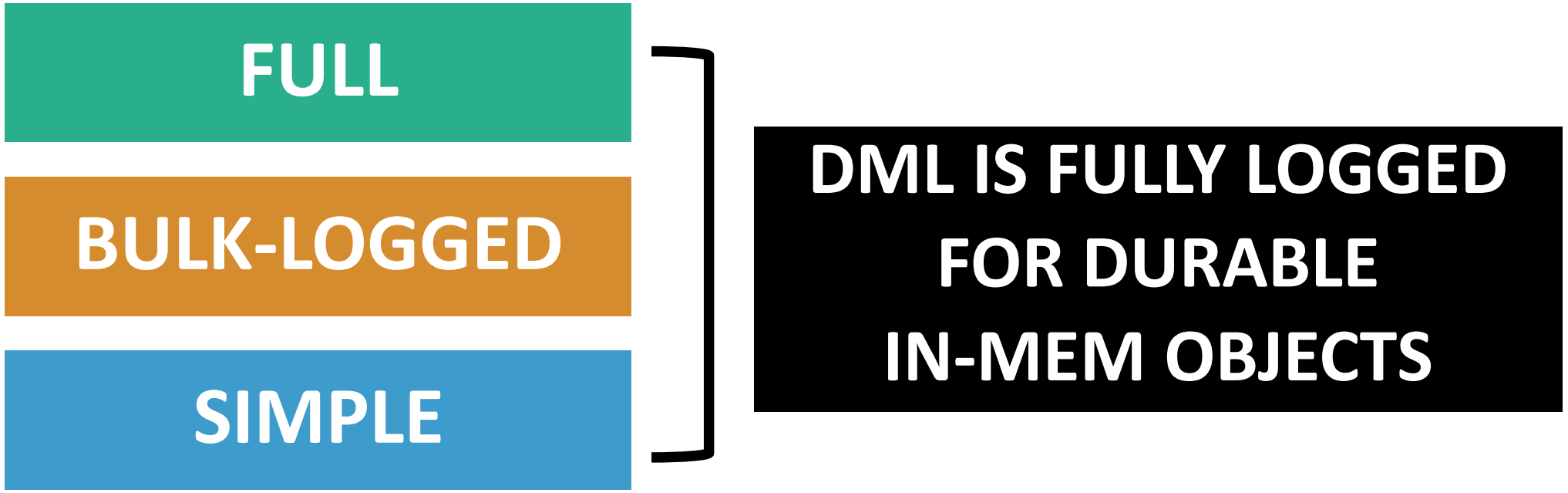
Feature	Per db	Per instance
In-Memory data	Yes	No
Columnstore/In-Mem tbl	Yes	No

Standard Edition (32 GB)

Feature	Per db	Per instance
In-Memory data	Yes	No
Columnstore/In-Mem tbl	Yes	No
Columnstore/HD tbl	No	Yes

Does the recovery model matter for durable DML? Why or why not?

Recovery model is . . . *irrelevant*



Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	Interpreted Compiled

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	Interpreted Compiled
Isolation	Locking	

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	Interpreted Compiled
Isolation	Locking	Validation

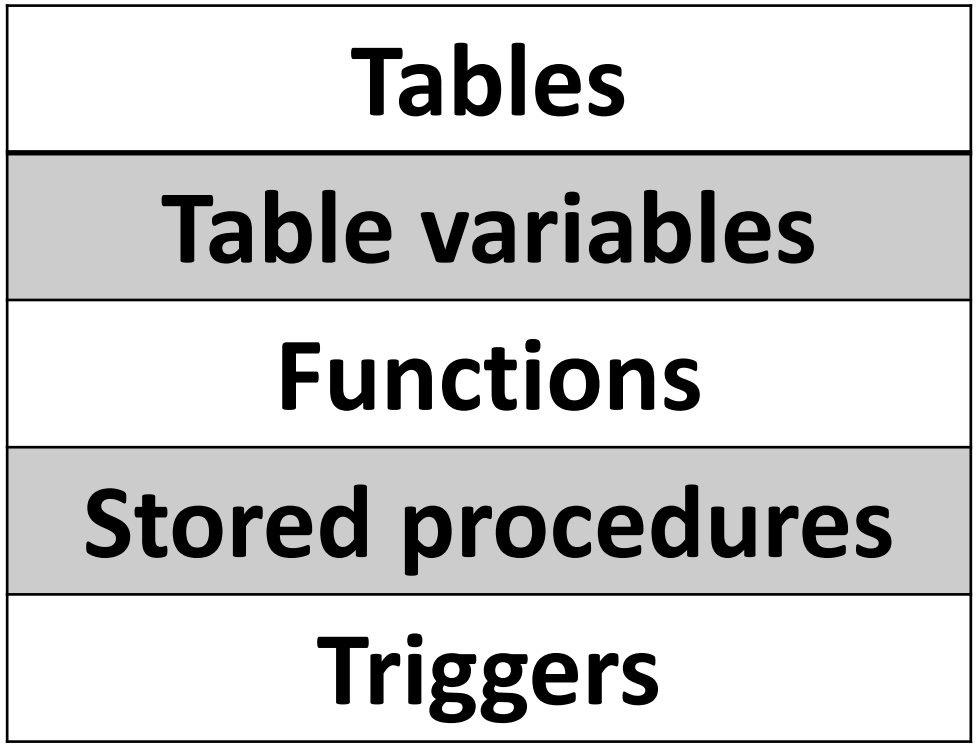
Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	Interpreted Compiled
Isolation	Locking	Validation
Deadlocks	Possible	

Comparison

Item	On-Disk	In-Mem
Durability	Mandatory	Optional
Storage	Pages/Extents	Streaming
TSQL	Interpreted	Interpreted Compiled
Isolation	Locking	Validation
Deadlocks	Possible	Impossible

Memory-optimized objects



Combining tables, On-Disk/In-Mem

```
SELECT *
FROM dbo.Disk
JOIN dbo.InMem ON Disk.Key = InMem.Key
```

Optimized for

OLTP

INGESTION

LOAD

TRANSIENT

Proof of concept

CORES

CONCURRENCY

COMPLEXITY

TOOLS

Desired workload characteristics

Item	Desired state
Concurrency	
Write level	
Latency/contention	
Hot data	
Business logic	
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	
Latency/contention	
Hot data	
Business logic	
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	Intensive
Latency/contention	
Hot data	
Business logic	
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	Intensive
Latency/contention	“Localized”
Hot data	
Business logic	
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	Intensive
Latency/contention	“Localized”
Hot data	Bounded
Business logic	
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	Intensive
Latency/contention	“Localized”
Hot data	Bounded
Business logic	Database
Temporary object use	

Desired workload characteristics

Item	Desired state
Concurrency	High
Write level	Intensive
Latency/contention	“Localized”
Hot data	Bounded
Business logic	Database
Temporary object use	High

Workload evaluation

Latency due to	In-Memory helps because
Engine	

Workload evaluation

Latency due to	In-Memory helps because
Engine	No locks/latches/spinlocks

Workload evaluation

Latency due to	In-Memory helps because
High CPU	

Workload evaluation

Latency due to	In-Memory helps because
High CPU	Native compilation: < CPU

Workload evaluation

Latency due to	In-Memory helps because
Logging	

Workload evaluation

Latency due to	In-Memory helps because
Logging	No write ahead logging

Workload evaluation

Latency due to	In-Memory helps because
Logging	No write ahead logging Logging efficiencies

Workload evaluation

Latency due to	In-Memory helps because
Logging	No write ahead logging Logging efficiencies Index changes not logged*

Workload evaluation

Latency due to	In-Memory helps because
TempDB	

Workload evaluation

Latency due to	In-Memory helps because
TempDB	Temp objects in memory

How In-Memory OLTP can help

You need better	In-Memory helps because
Performance	

How In-Memory OLTP can help

You need better	In-Memory helps because
Performance	Lower CPU (native) Less logging No logging (indexes) TempDB object replacement

How In-Memory OLTP can help

You need better	In-Memory helps because
Performance	Lower CPU (native) Less logging No logging (indexes) TempDB object replacement
Scalability	

How In-Memory OLTP can help

You need better	In-Memory helps because
Performance	Lower CPU (native) Less logging No logging (indexes) TempDB object replacement
Scalability	Latching Locking Spinlock *not possible*

Lock free

Tables

Indexes

Lock free != WAIT FREE

Lock free != WAIT FREE

Transaction log (DURABLE)

Lock free != WAIT FREE

Transaction log (DURABLE)

Dependent transactions

Use cases

- **High throughput, low latency**

Use cases

- **High throughput, low latency**
- **Caching/session state**

Use cases

- **High throughput, low latency**
- **Caching/session state**
- **Data ingestion (IoT)**

Use cases

- High throughput, low latency
- Caching/session state
- Data ingestion (IoT)
- TempDB object replacement

Use cases

- High throughput, low latency
- Caching/session state
- Data ingestion (IoT)
- TempDB object replacement
- ETL

CONCURRENCY

Concurrency model - disk

Pessimistic
Optional db setting for RCSI/SI (row versions) • TempDB • Solves R/W blocking • W/W still blocks

Concurrency model - memory

Optimistic – MVCC

Mandatory **row versions**

- **Memory**
- **Solves R/W and W/W blocking**
- **First COMMIT wins, other writers fail**

Potential workload issues

Excessive *conflicts*

Chatty applications/short procs (native)

Resource issues: memory/storage/IOPS

New potential bottlenecks

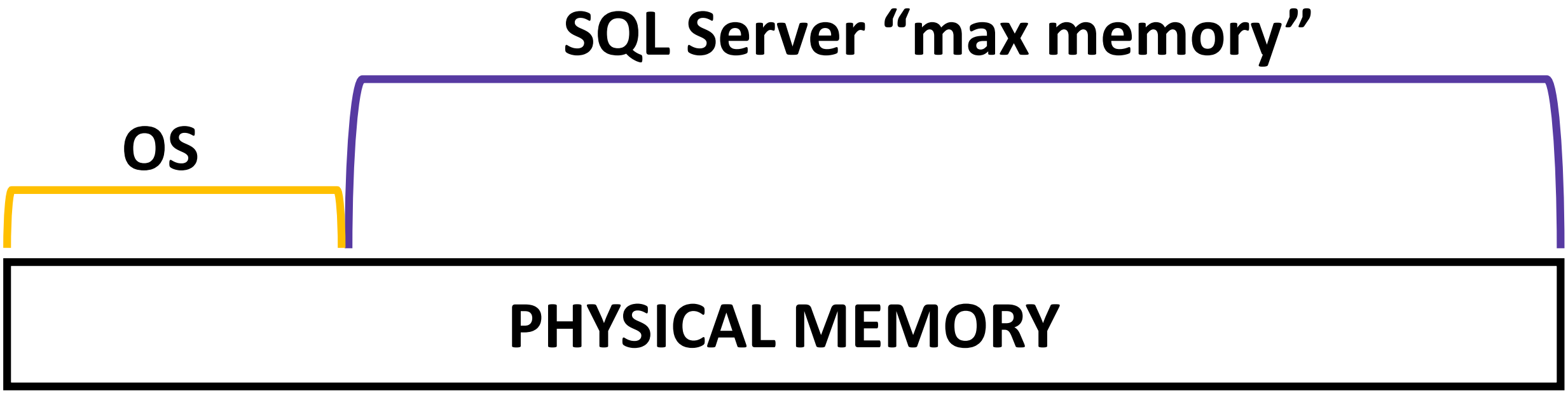
WRITELOG
IOPS

MEMORY

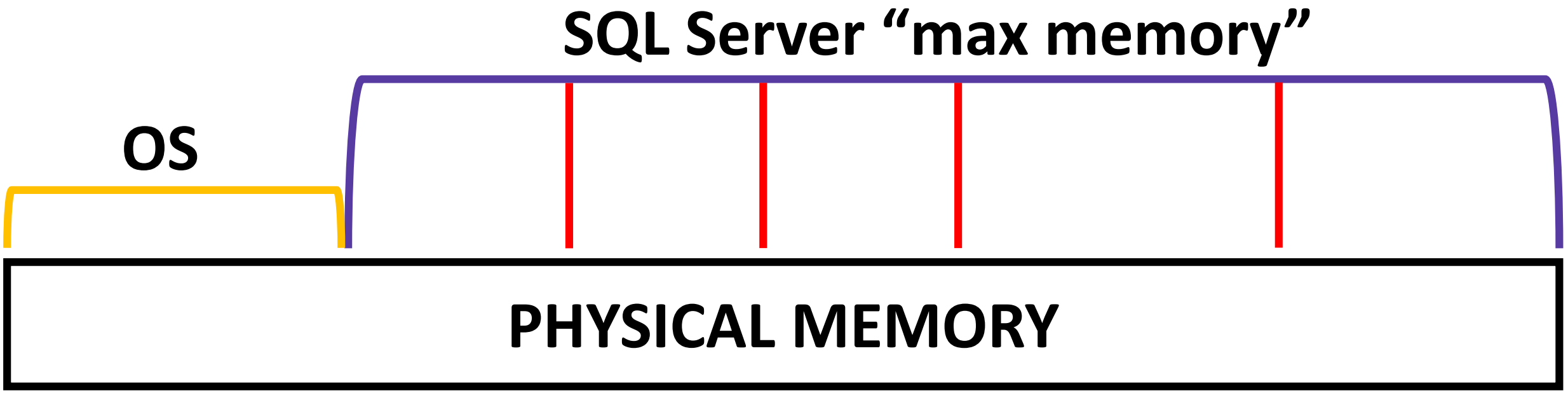
Memory allocation

Memory allocation

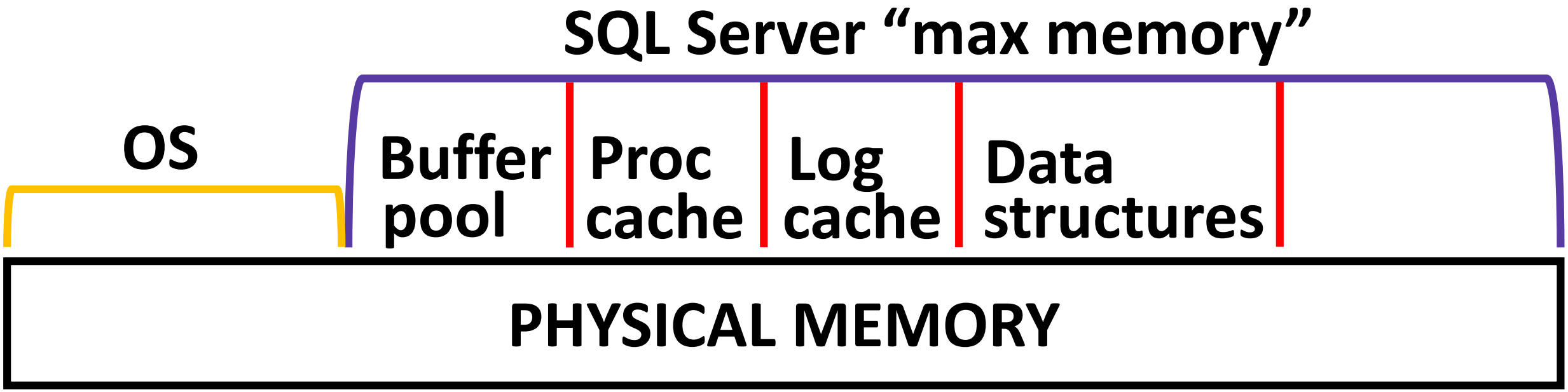
Memory allocation



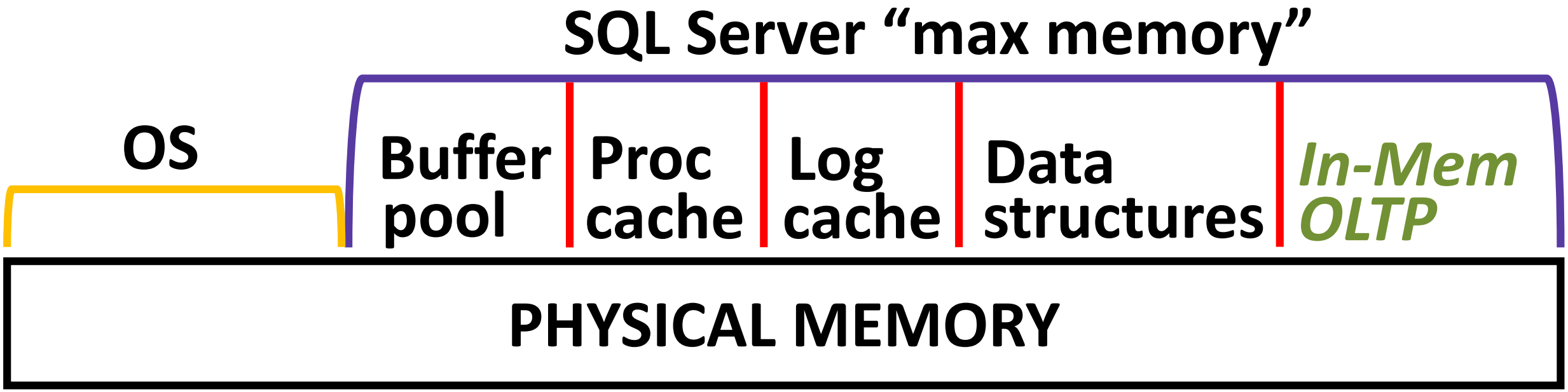
Memory allocation



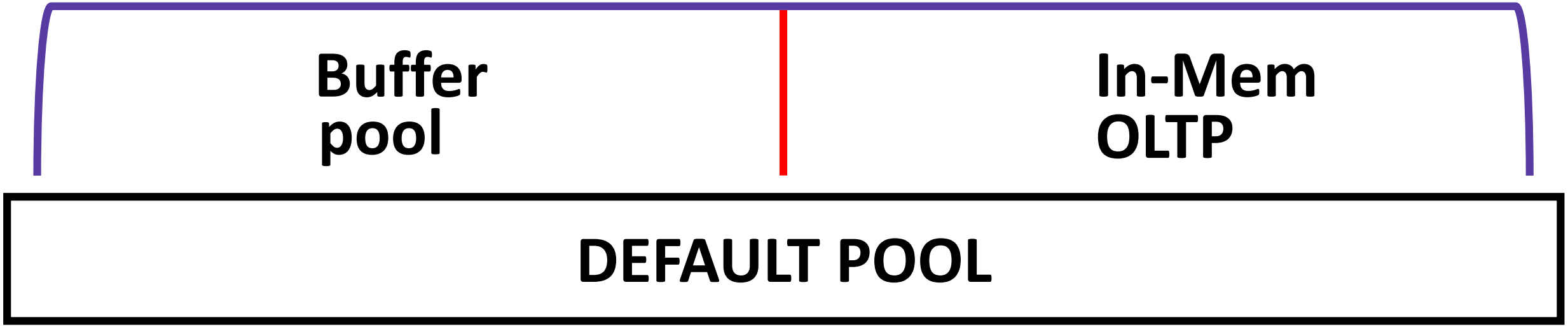
Memory allocation



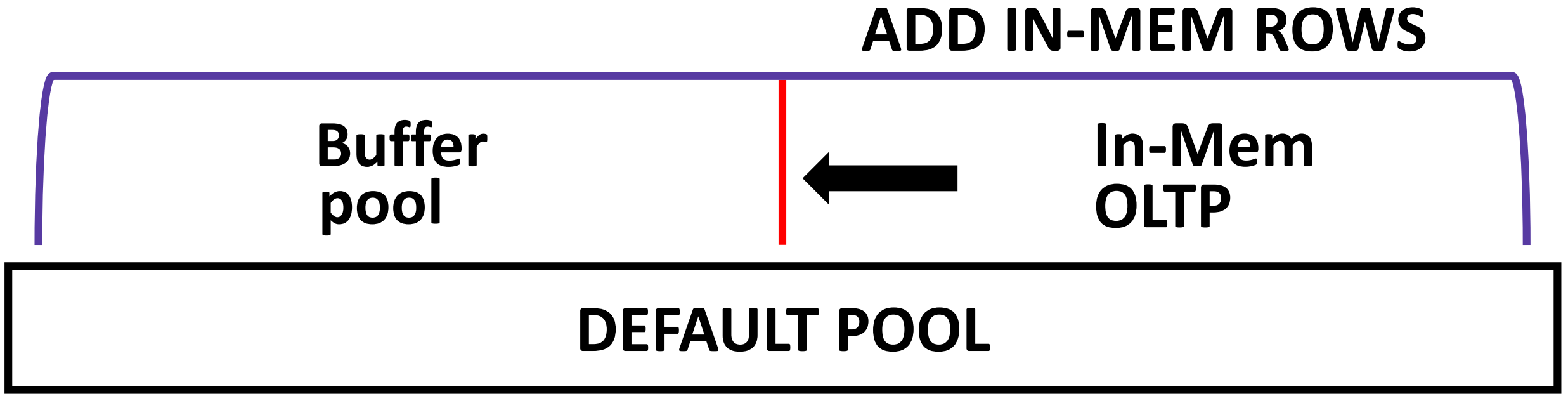
Memory allocation



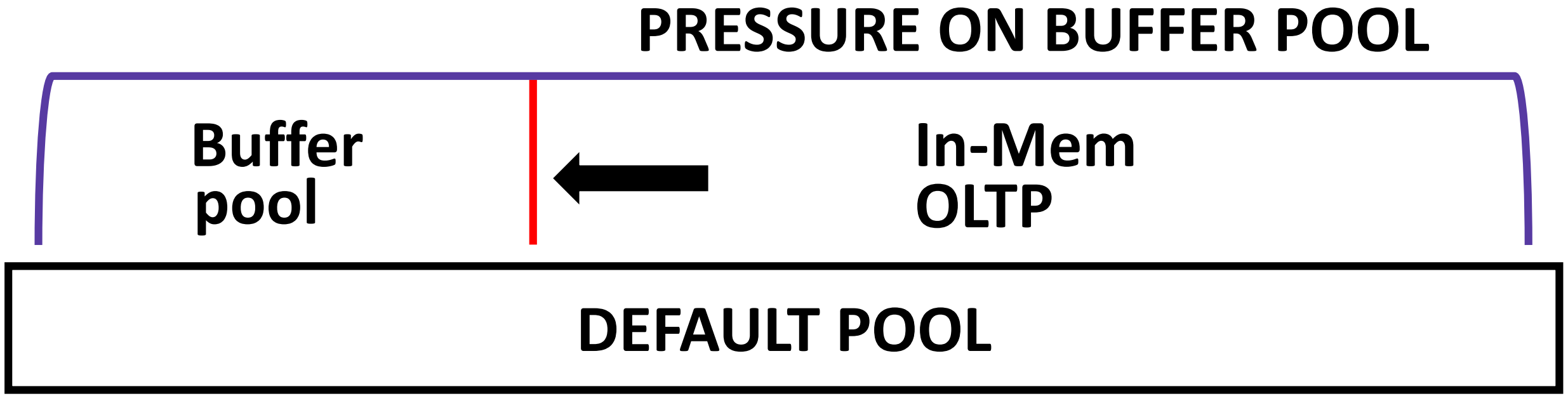
Memory dance



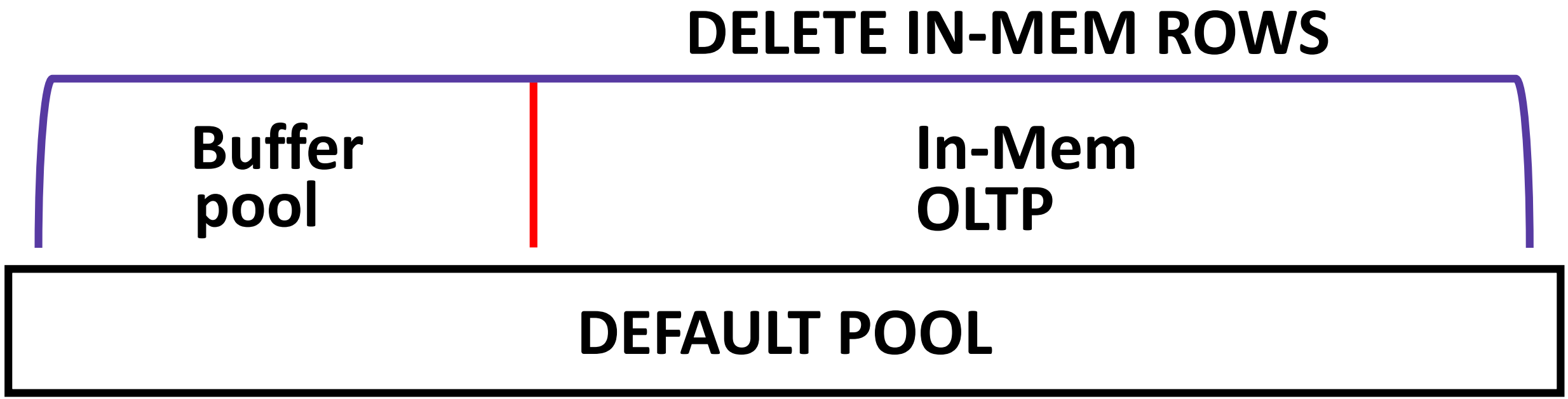
Memory dance



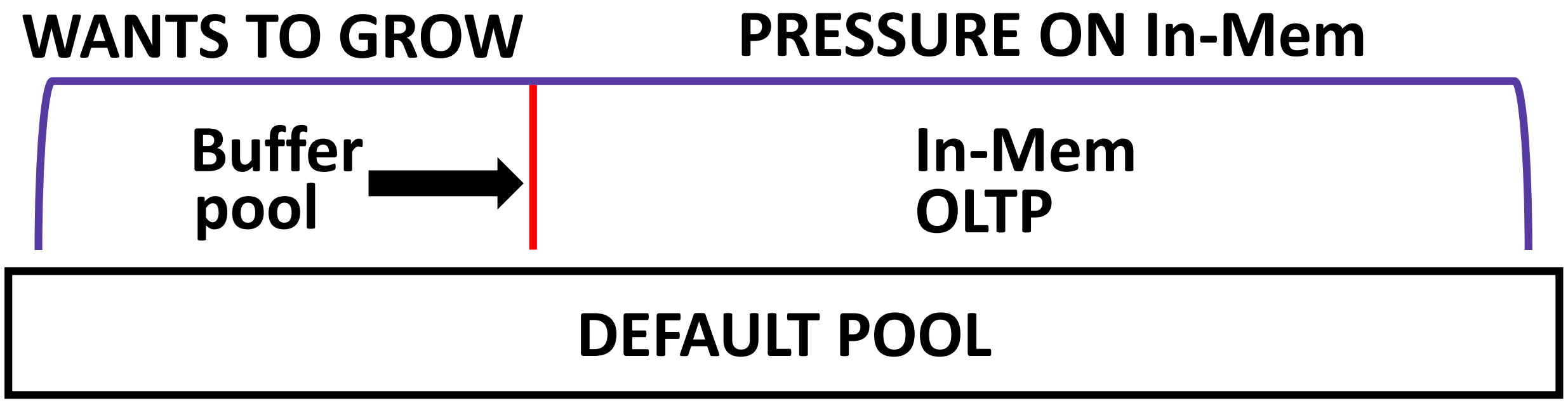
Memory dance



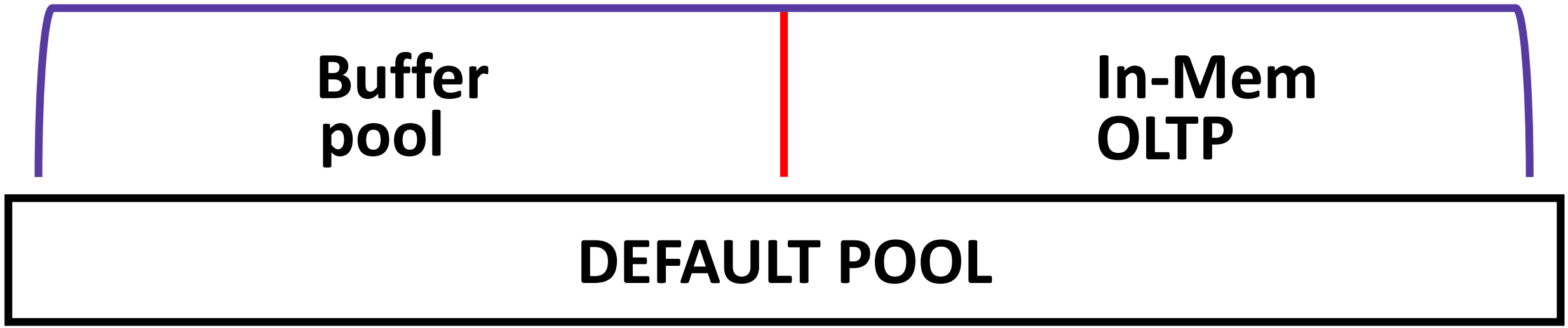
Memory dance



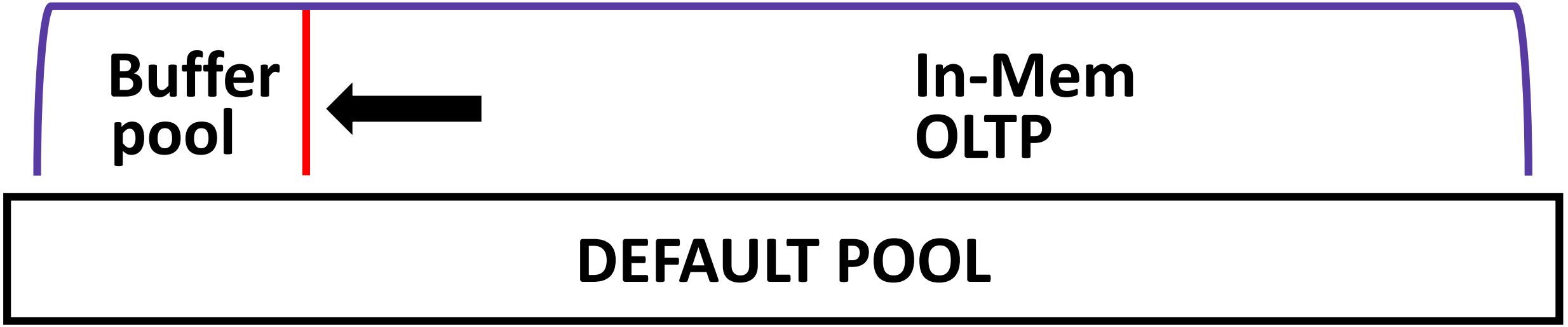
Memory dance



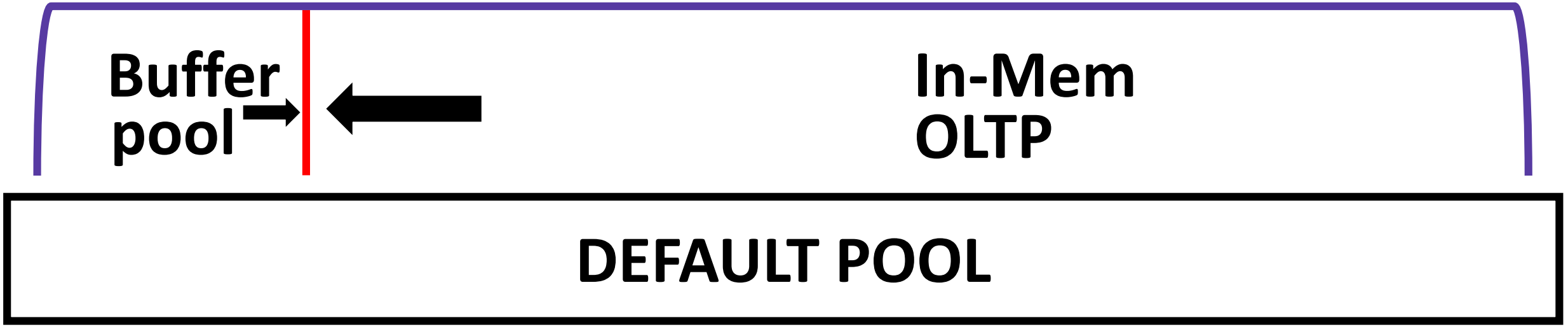
Memory dance



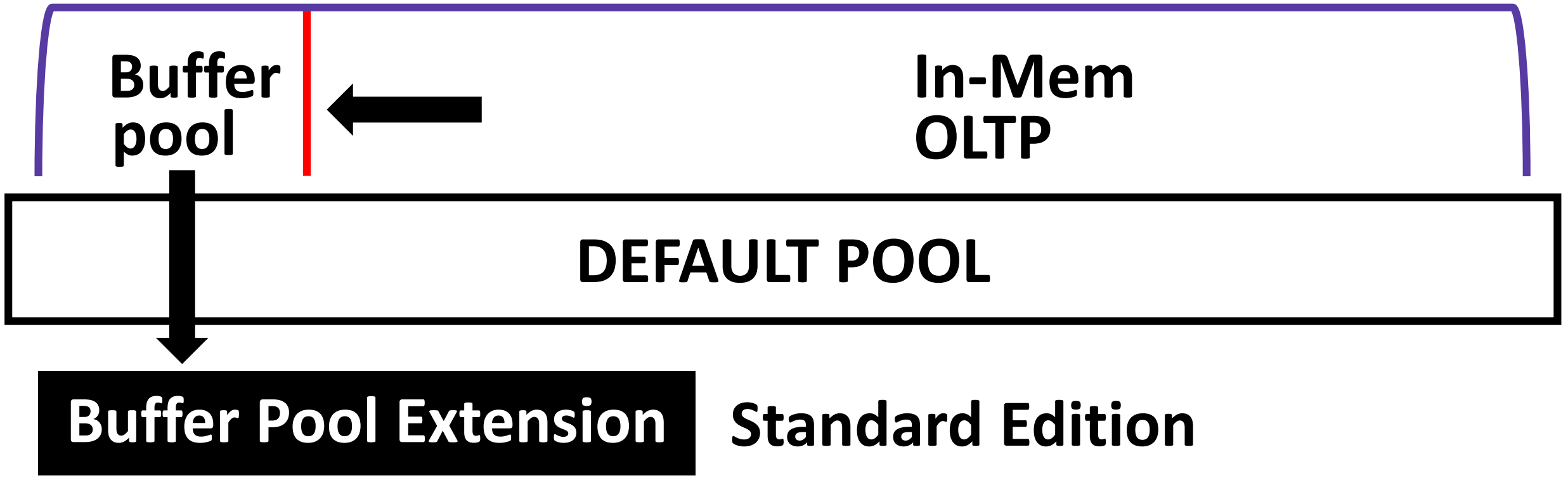
Memory dance



Memory dance



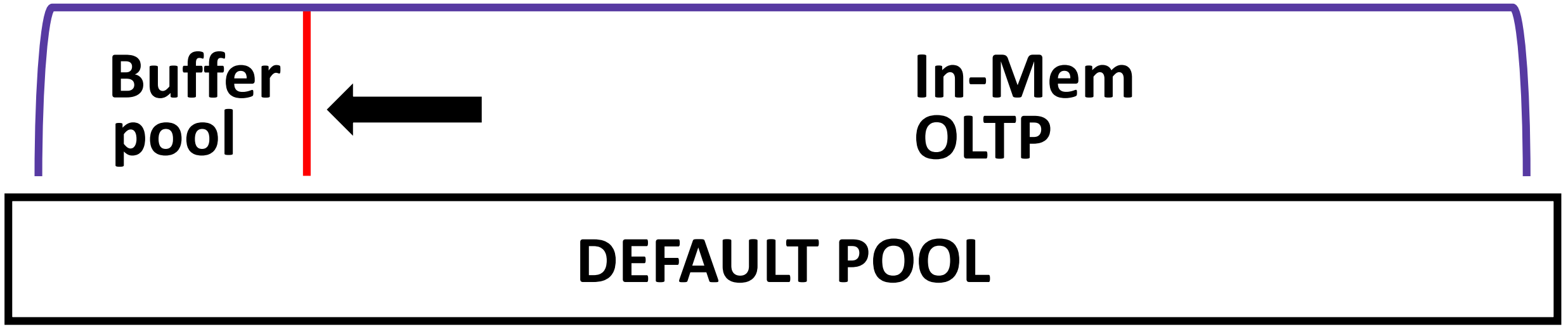
Memory dance



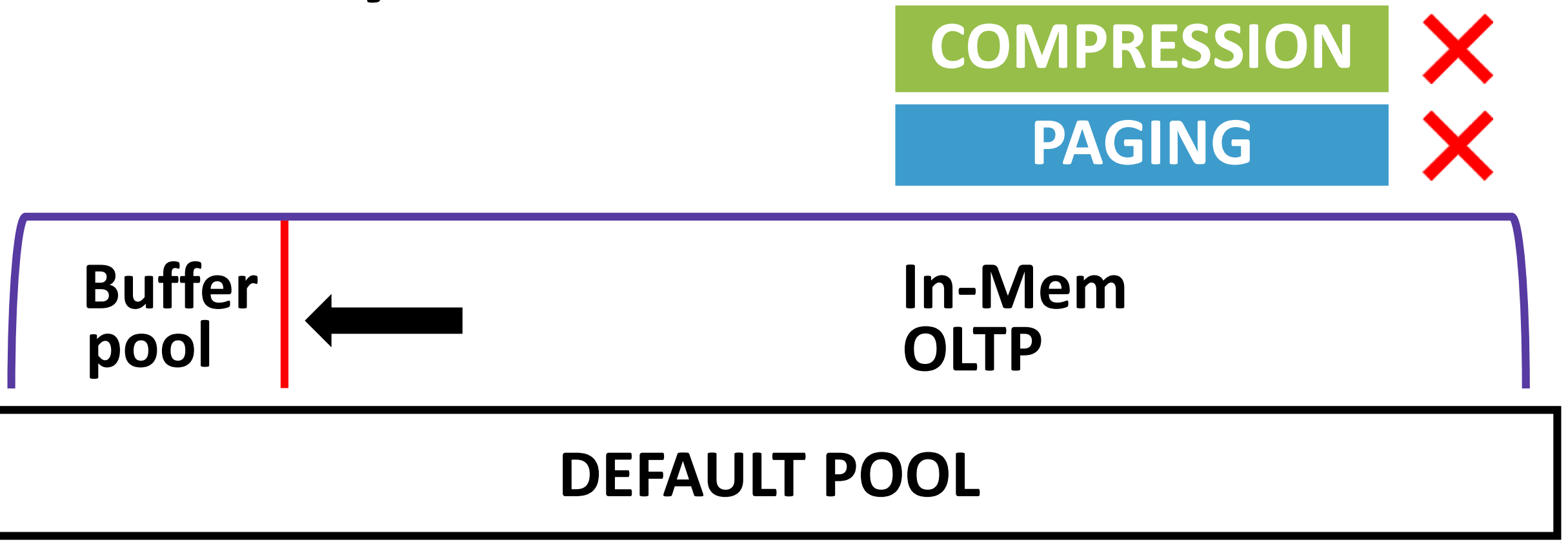
Memory dance

COMPRESSION

PAGING



Memory dance



Windows Server max memory

Windows 2012R2

Windows 2016/2019

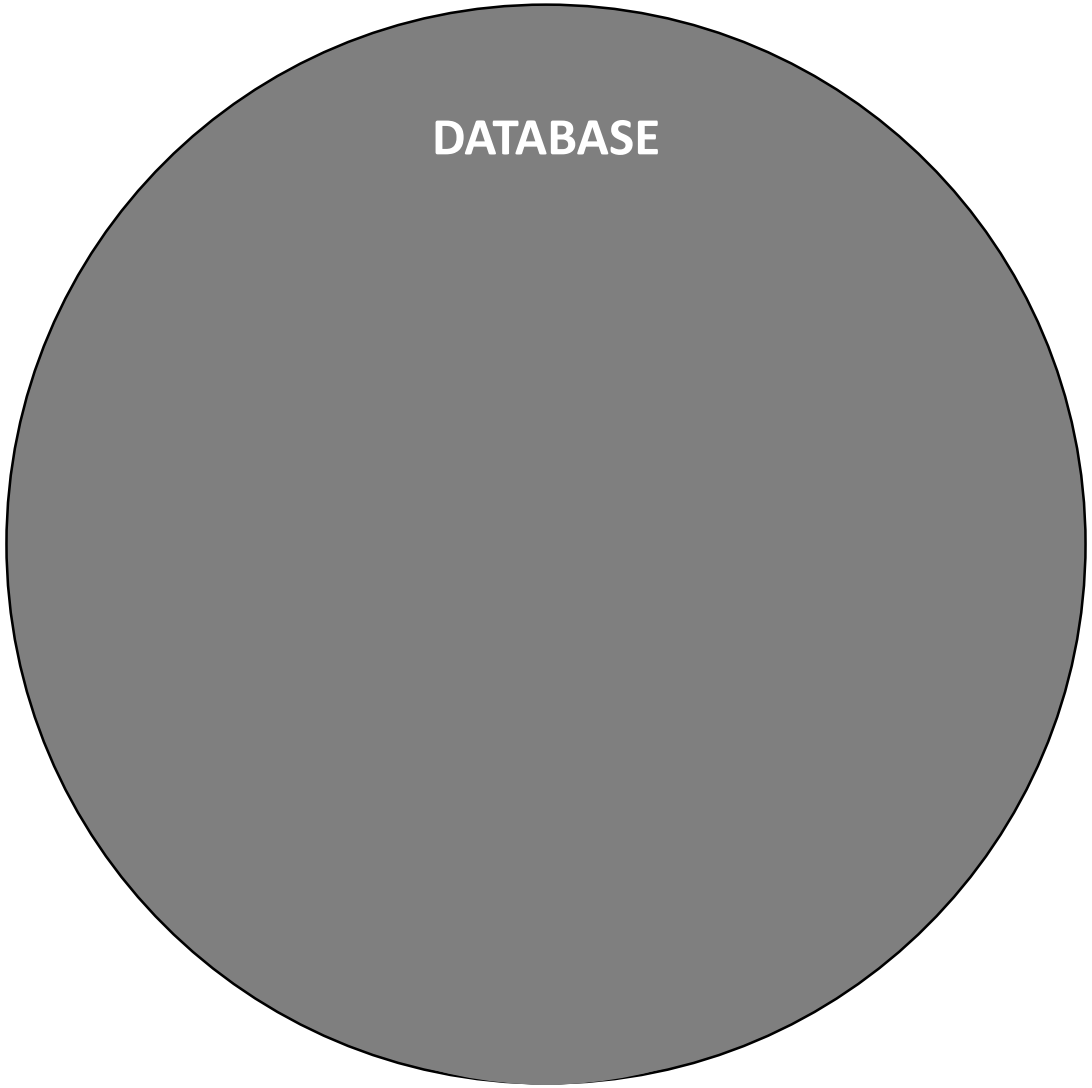
Windows Server max memory

Windows 2012R2 = 4TB

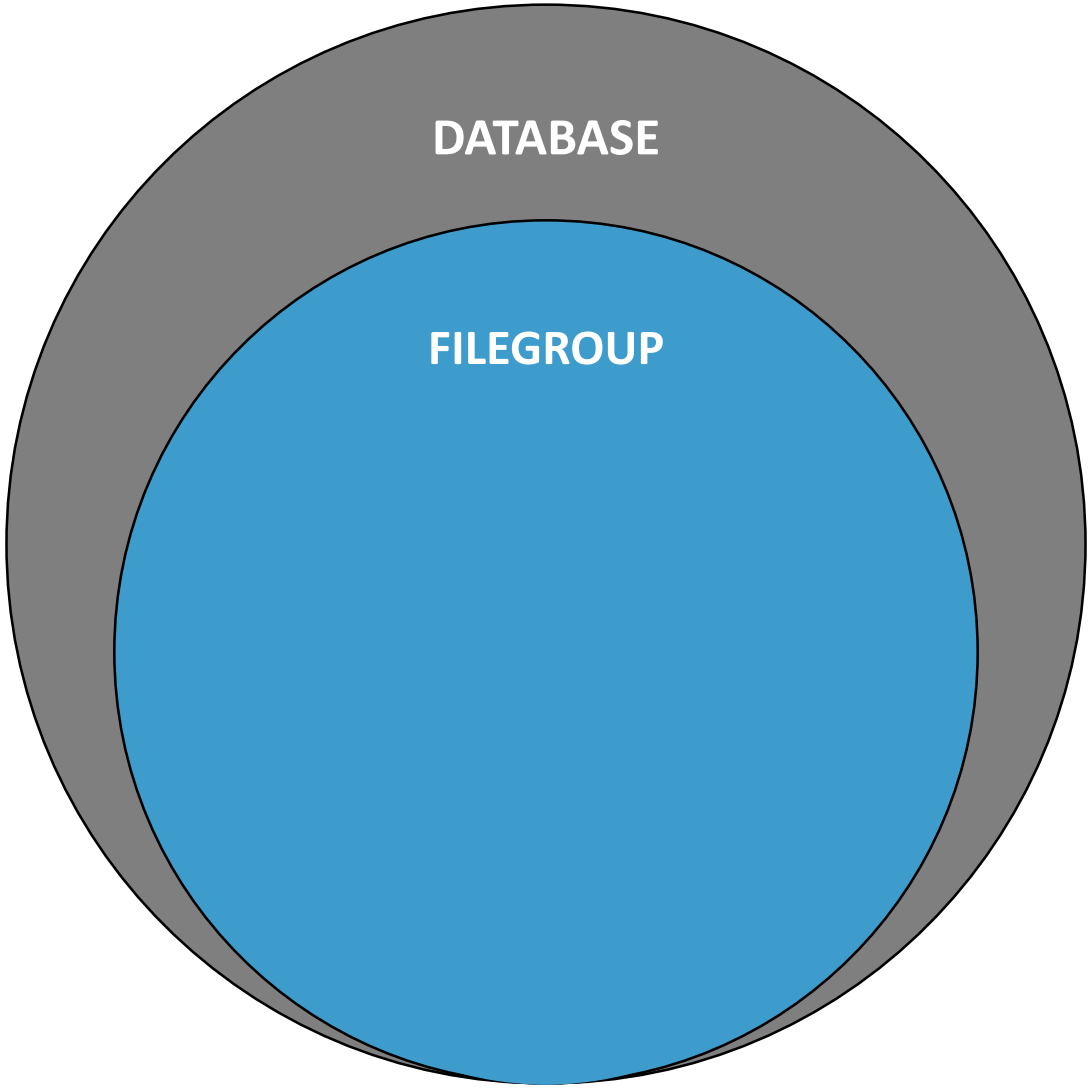
Windows 2016/2019 = 24TB

ARCHITECTURE

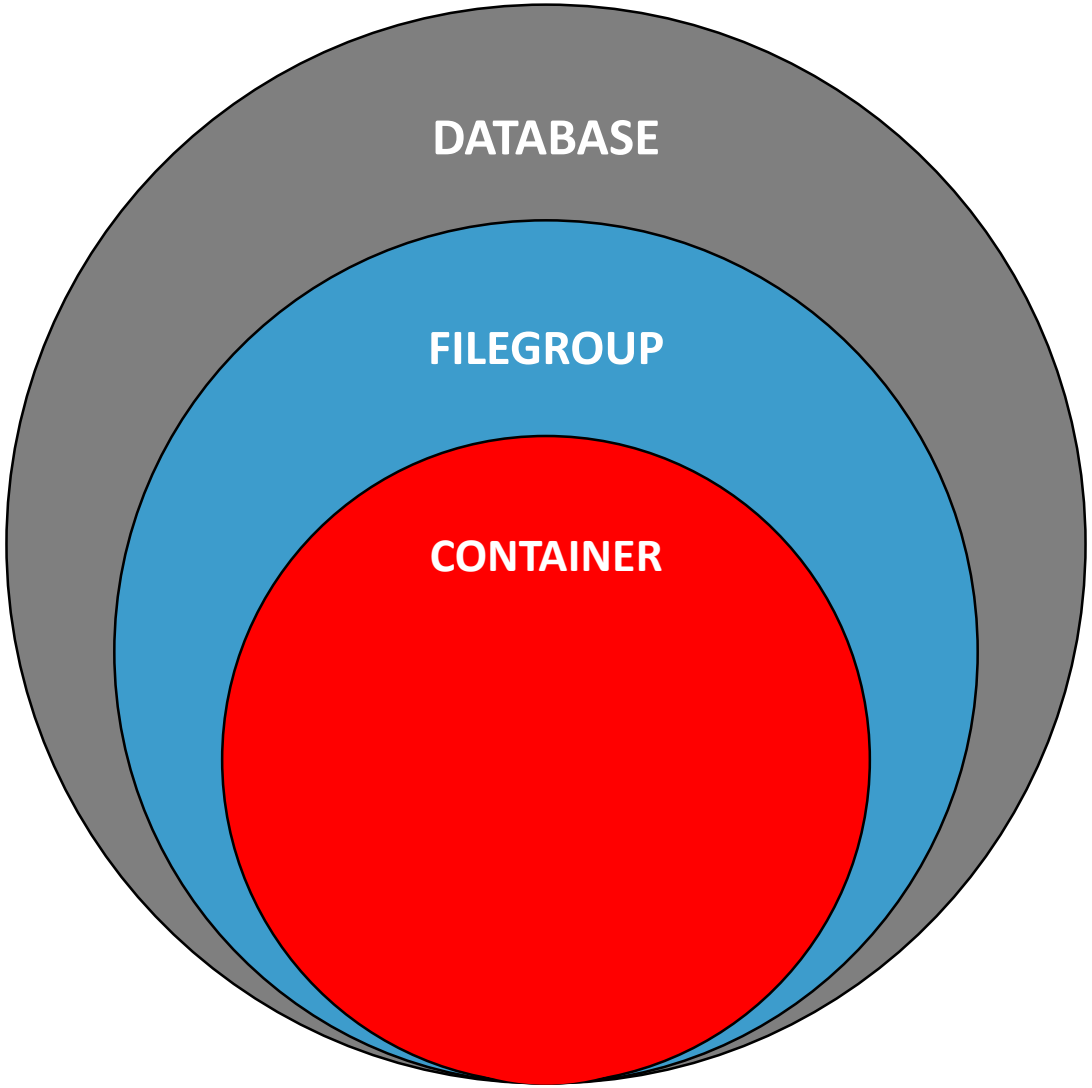
Data/delta files



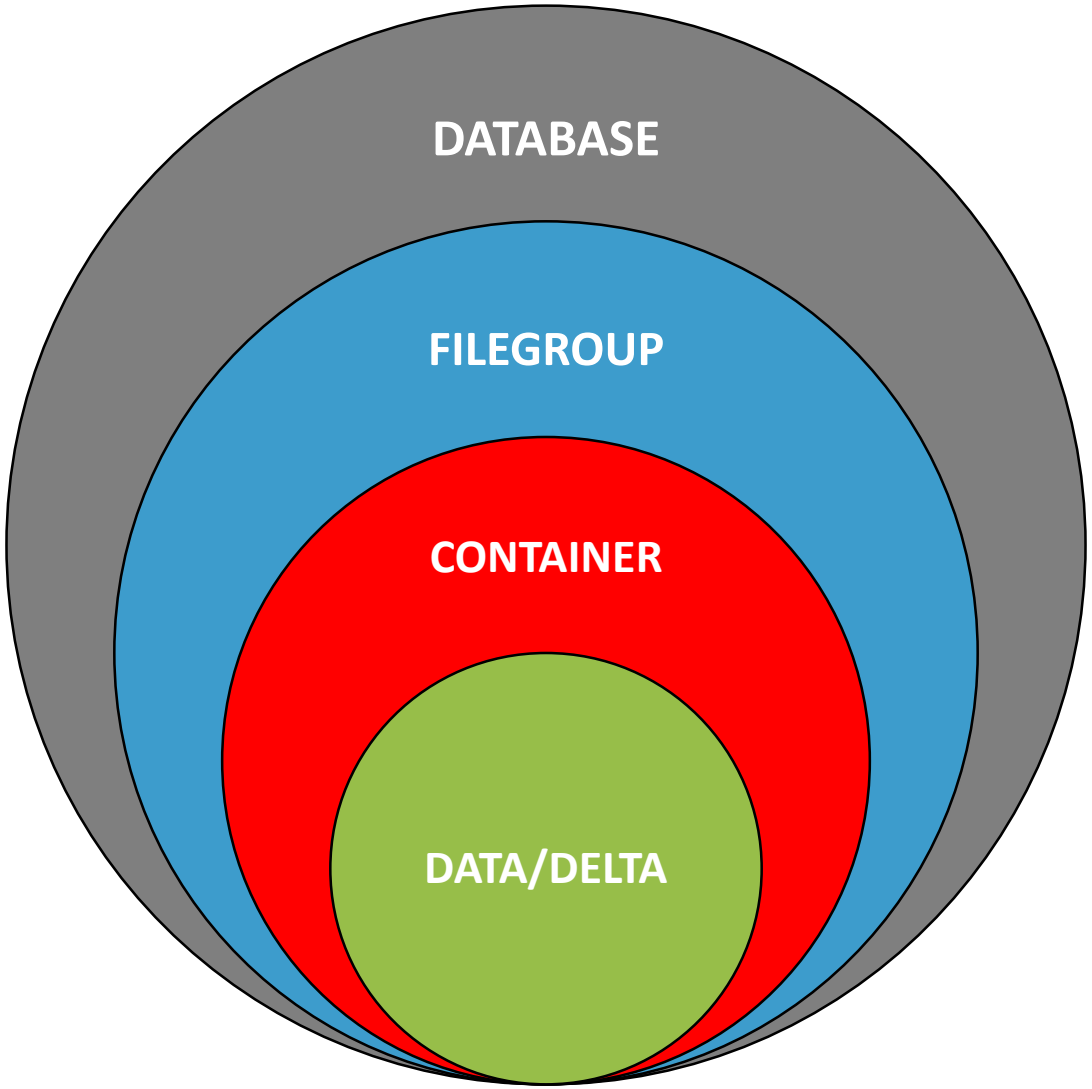
Data/delta files



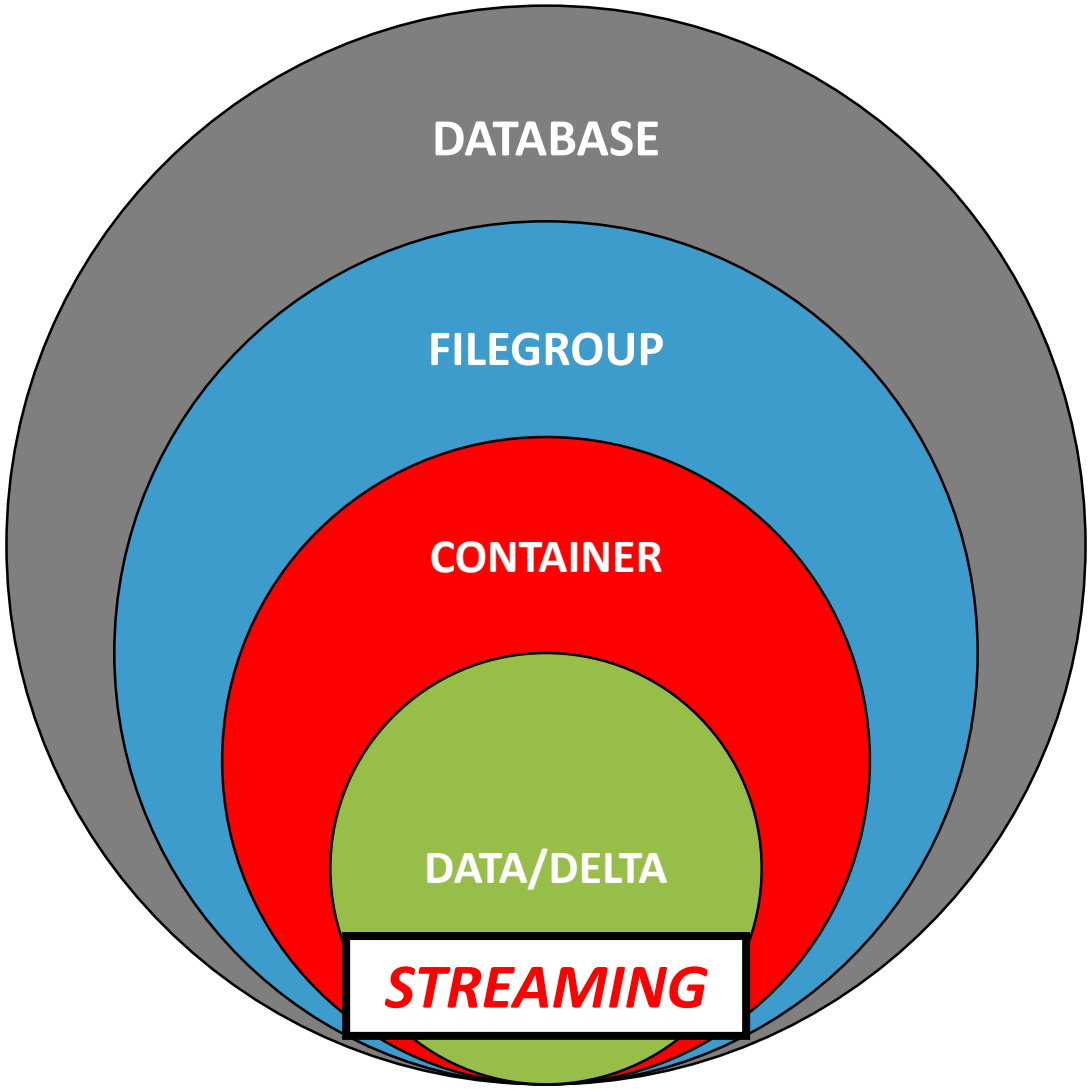
Data/delta files



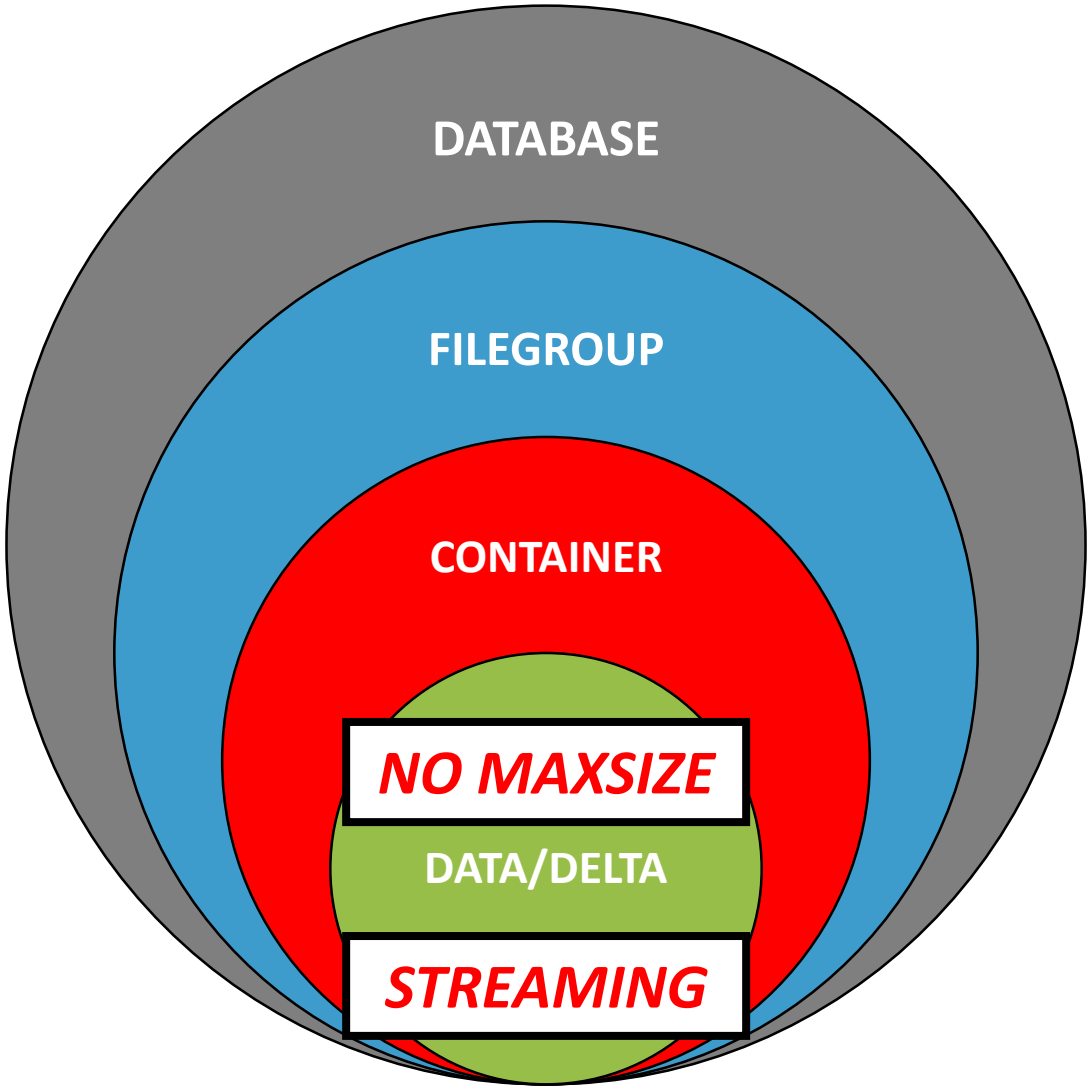
Data/delta files



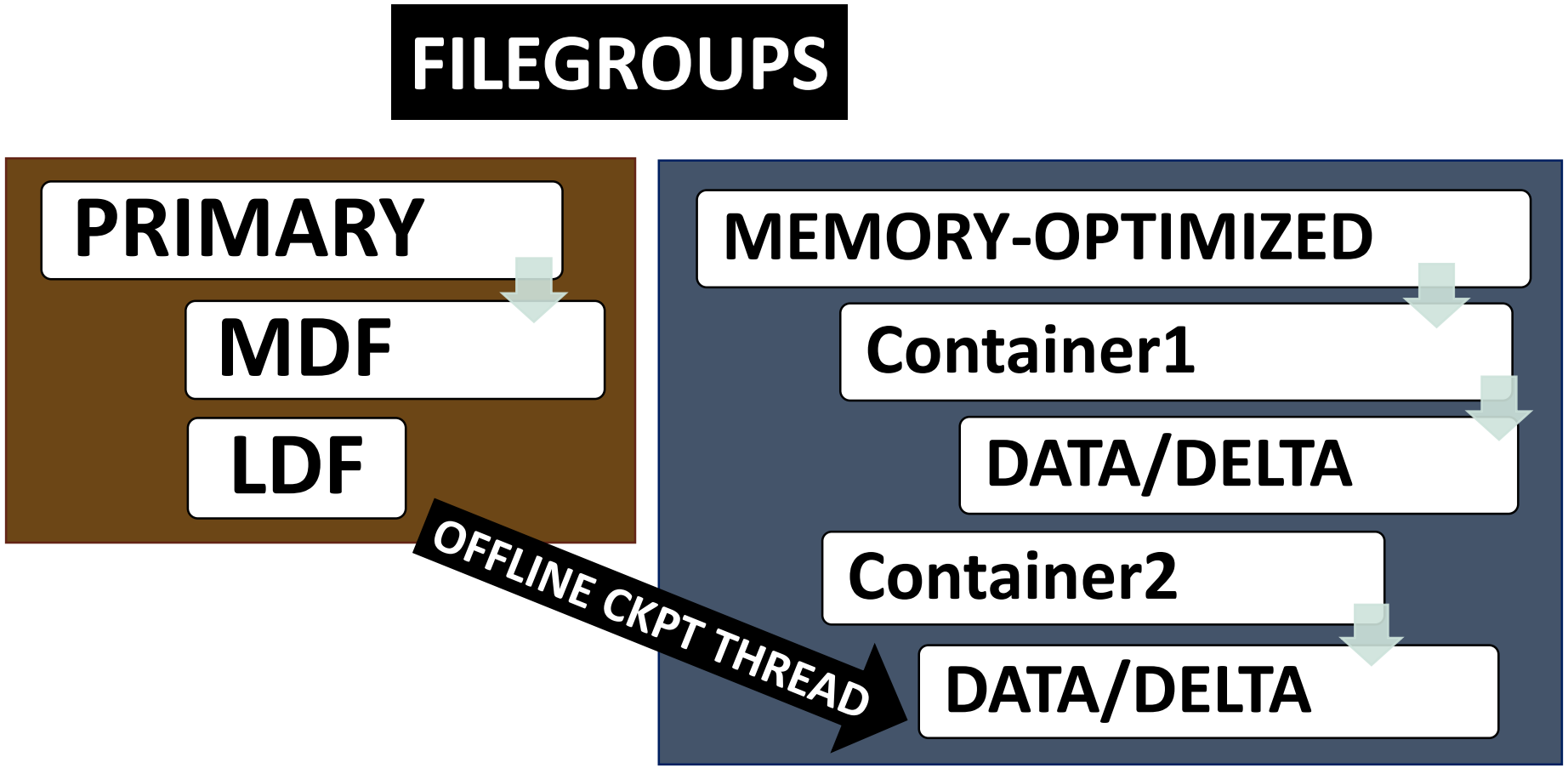
Data/delta files



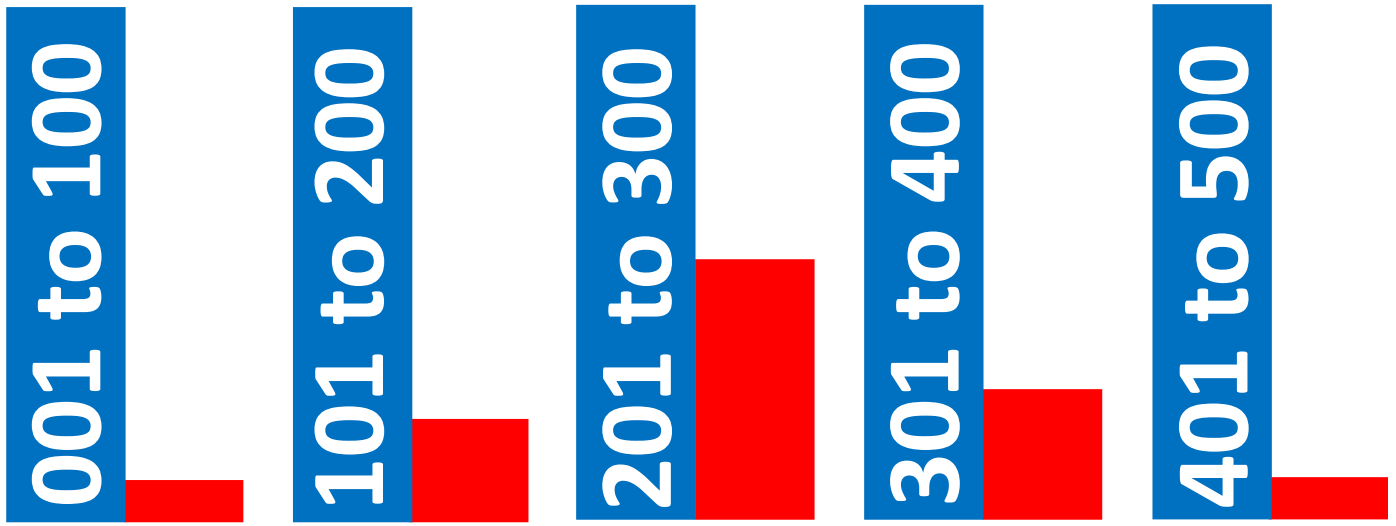
Data/delta files



Data/delta files



Data/delta files



DATA

"Checkpoint File Pairs"

DELTA

Data/delta files

```
Directory of H:\SQLDATA\InMemTestingMem1\$HKv2

06/10/2017  07:10 PM    <DIR>          .
06/10/2017  07:10 PM    <DIR>          ..
06/10/2017  07:07 PM         8,388,608 {356641CA-4910-4FE2-8AAD-7DC68FEA77EF}.hkckp
06/10/2017  07:07 PM         8,388,608 {41879355-81BD-4AA9-AFE3-CD049FFA88AF}.hkckp
06/10/2017  07:07 PM        67,108,864 {45B0CF15-8C13-4FBC-A724-50522314F1C2}.hkckp
06/10/2017  07:07 PM       134,217,728 {4631DDD7-1519-4C59-B136-1BCB5906DA24}.hkckp
06/10/2017  07:07 PM        67,108,864 {4A8CD6DD-F2D0-49BA-8C4F-E154546E54C6}.hkckp
06/10/2017  07:07 PM         8,388,608 {50FA7354-5D1B-4D4C-AACB-1989EBF24739}.hkckp
06/10/2017  07:07 PM        67,108,864 {88DC1881-D219-447B-8C4C-D2D4C1F33946}.hkckp
06/10/2017  07:07 PM        16,777,216 {8F248948-FD1A-43BD-B394-450E6D17434B}.hkckp
06/10/2017  07:07 PM         8,388,608 {98EB693A-5FB9-49BF-B651-323AC3AC5DD8}.hkckp
06/10/2017  07:07 PM       134,217,728 {A68B10A6-E394-4CAC-BD9B-12D7A2686409}.hkckp
06/10/2017  07:07 PM        16,777,216 {B2F182B3-66D7-4BEB-9C8B-D565DFA5FBB8}.hkckp
06/10/2017  07:07 PM        16,777,216 {D37F53C2-264A-4A65-9068-4E8B266194F0}.hkckp
06/10/2017  07:07 PM       134,217,728 {D4C9351E-38B3-4EFA-9F64-09770E5FE12F}.hkckp
06/10/2017  07:07 PM       134,217,728 {EBC67990-4EAF-4FC9-8412-BEBE32ECAF2B}.hkckp
06/10/2017  07:07 PM        16,777,216 {F429C032-FD7C-463E-B925-FA4913D25EC2}.hkckp
06/10/2017  07:07 PM         8,388,608 {F5A305C3-F762-485B-A0FA-1E66DB21E778}.hkckp
06/10/2017  07:07 PM       134,217,728 {F624BEFB-D6C8-432C-89CF-B00C86A932D5}.hkckp
              17 File(s)    981,467,136 bytes
```

Durability work flow

UPDATE row in memory and COMMIT

Durability work flow

UPDATE row in memory and COMMIT

Write to transaction log

Durability work flow

UPDATE row in memory and COMMIT

Write to transaction log

Offline checkpoint thread harvests in-memory event from transaction log

Durability work flow

UPDATE row in memory and COMMIT

Write to transaction log

Offline checkpoint thread harvests in-memory event from transaction log

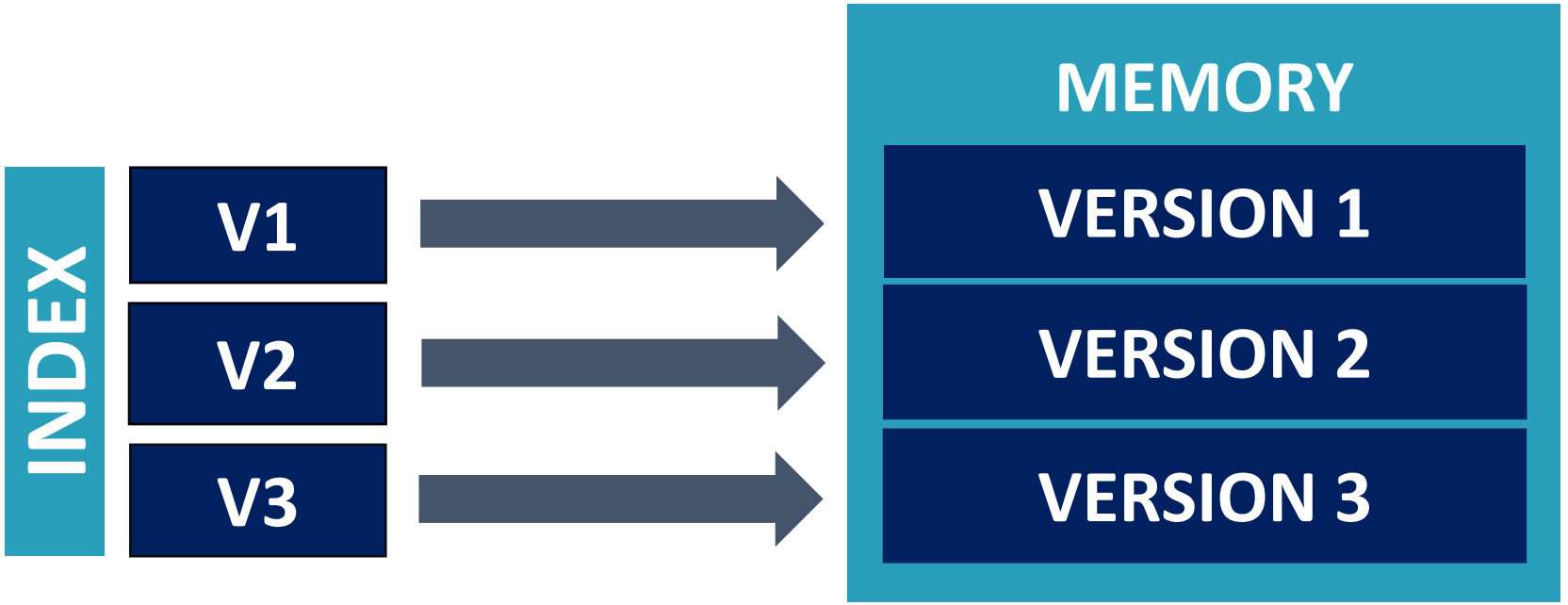
Insert to data/delta files

Data/delta files

ROW VERSIONS

Row structure (simplified)

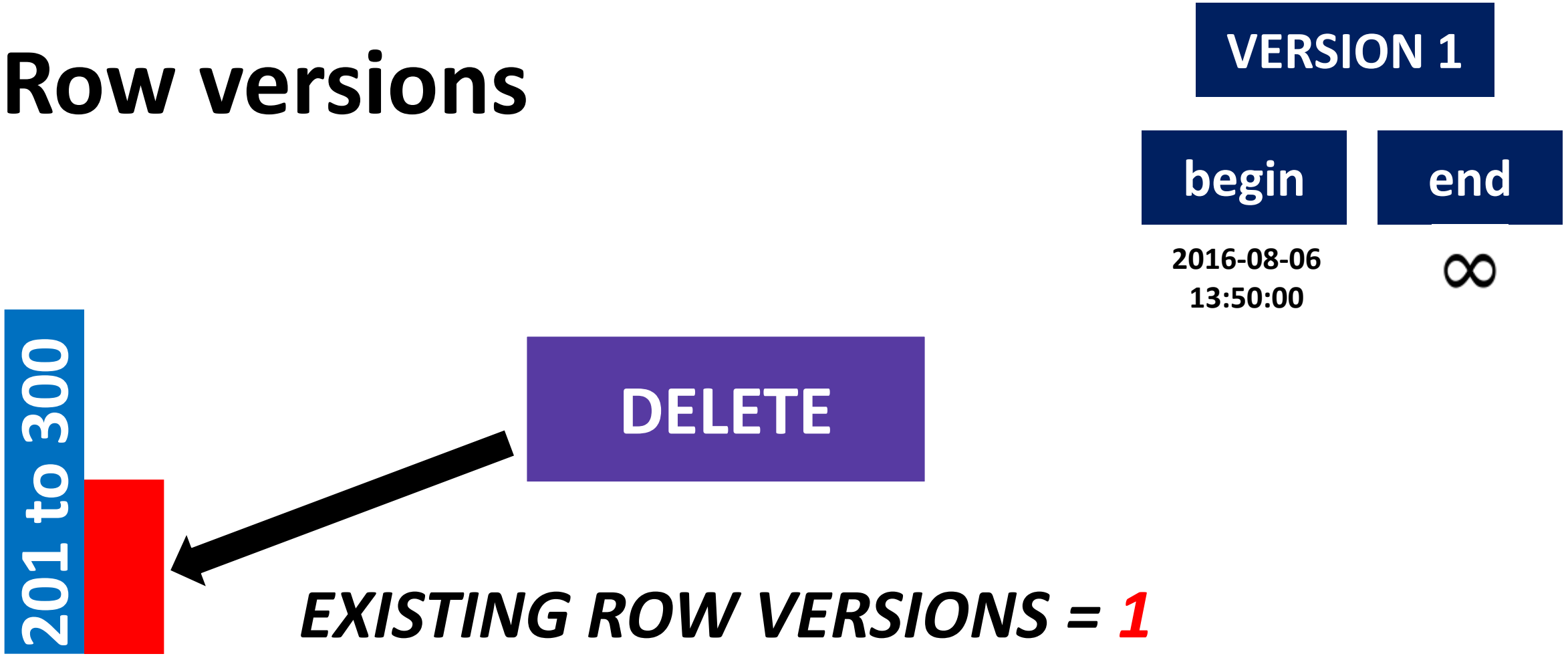
Row versions



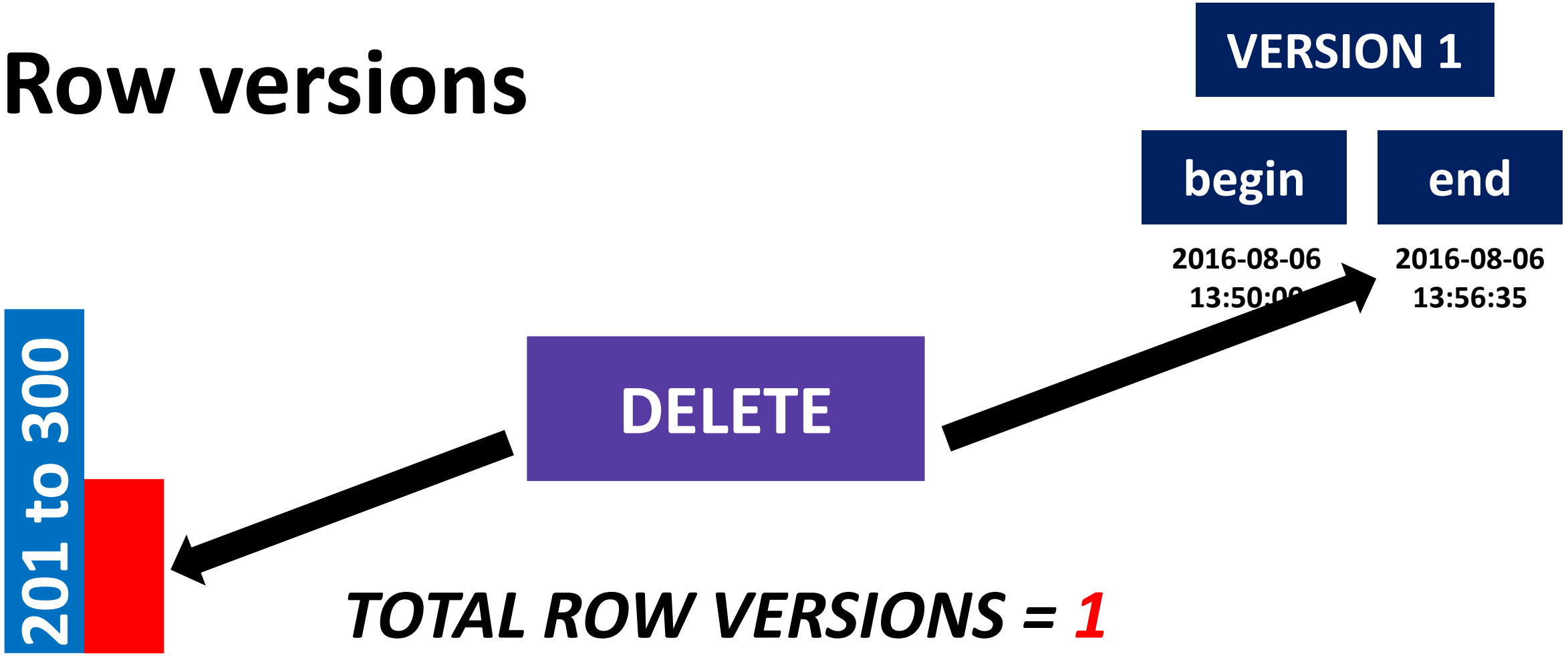
Row versions

DML

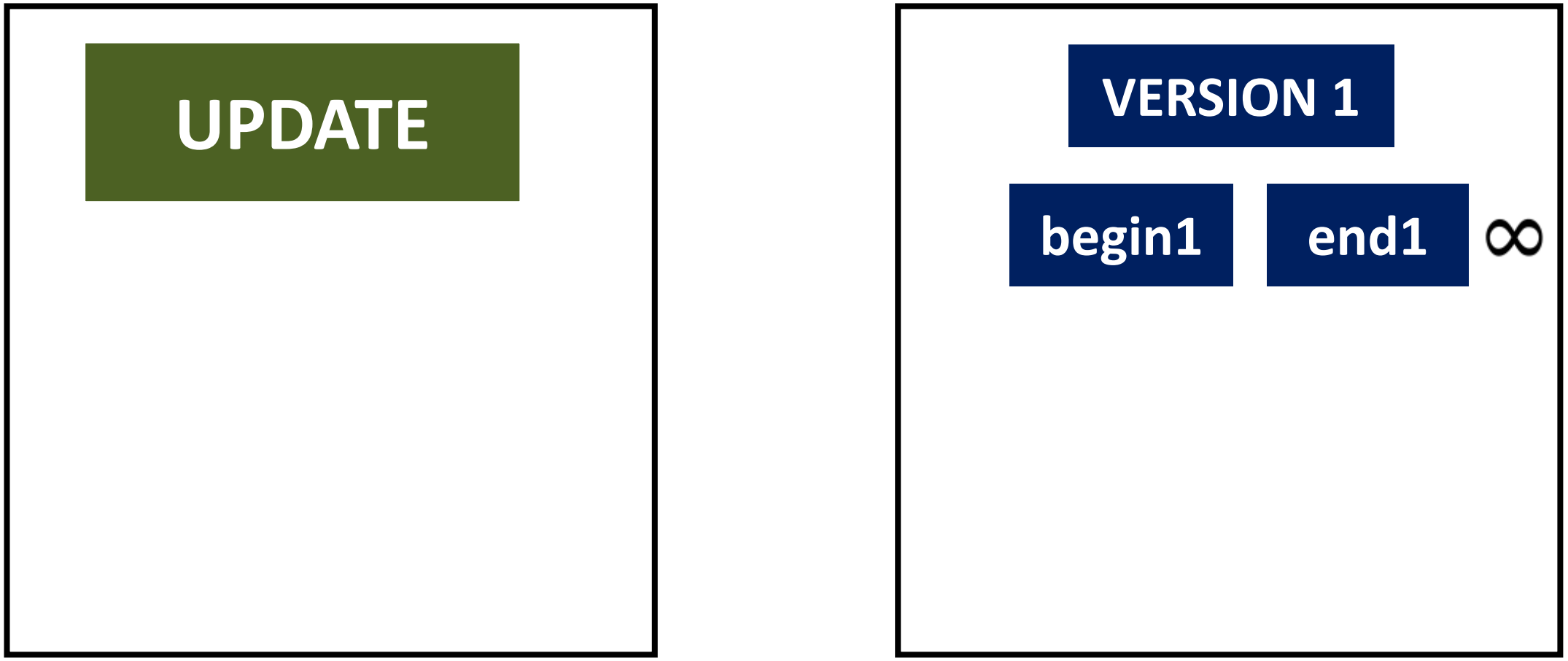
Row versions



Row versions

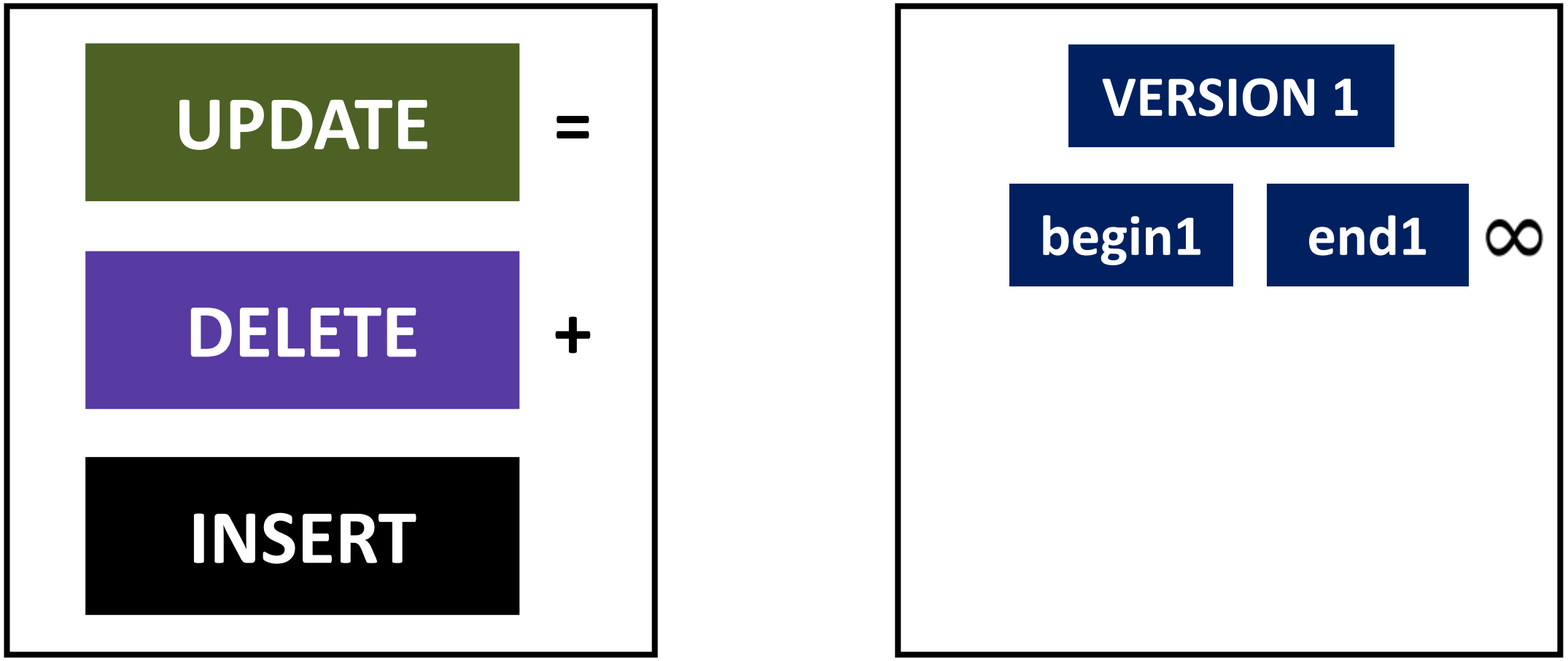


Row versions



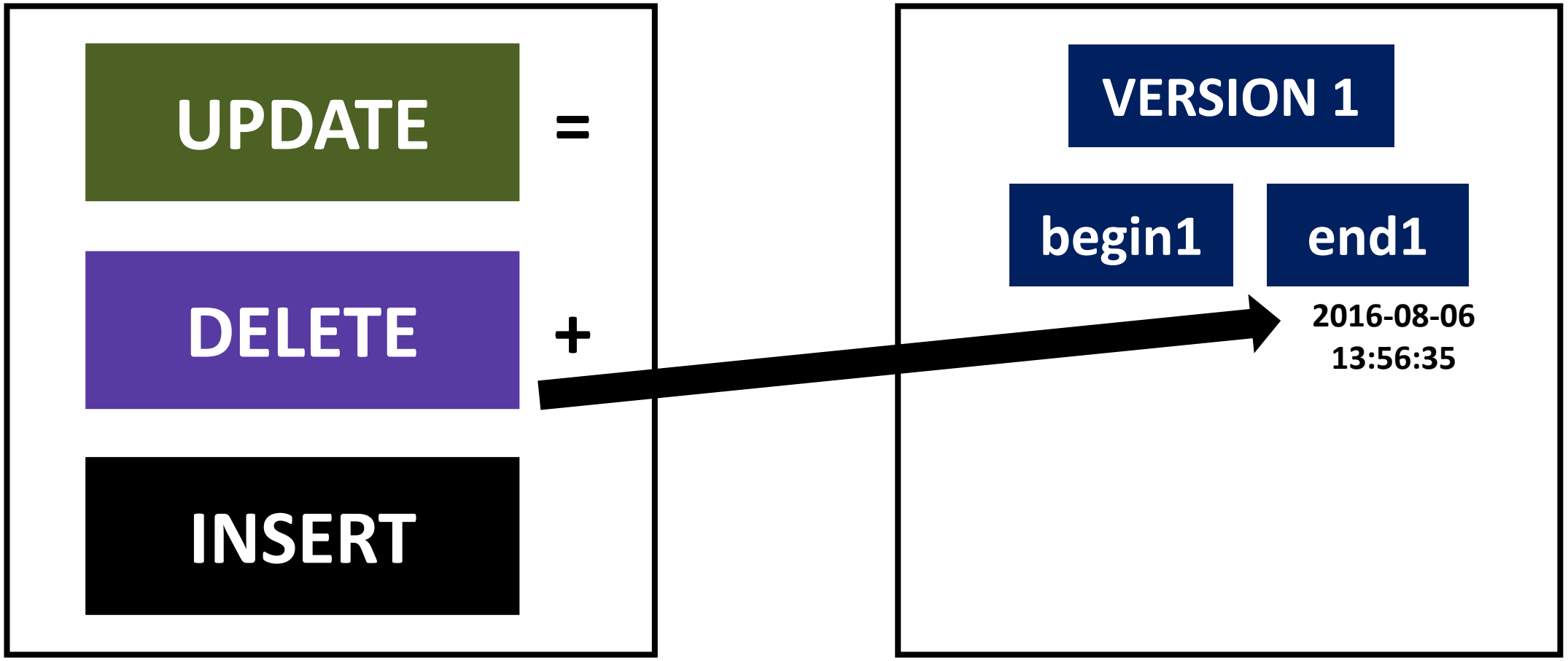
EXISTING ROW VERSIONS = 1

Row versions



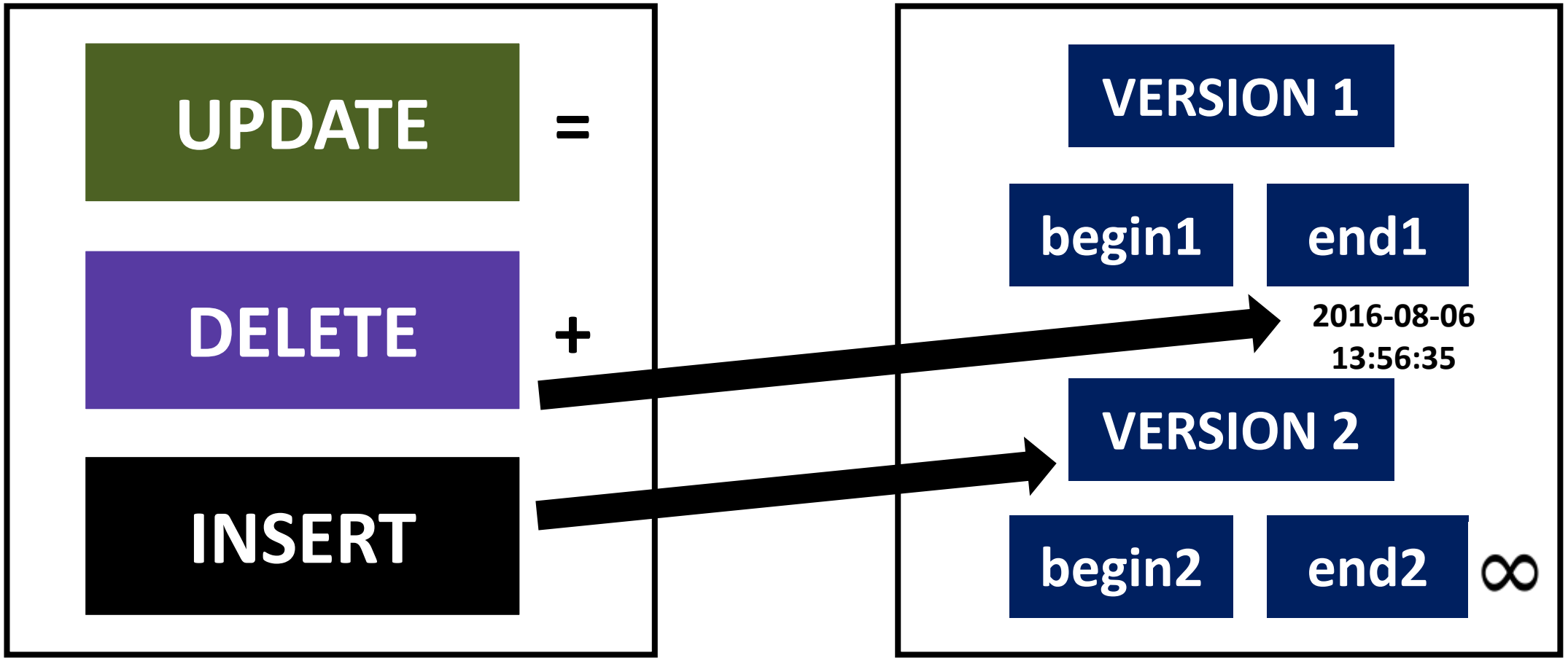
EXISTING ROW VERSIONS = 1

Row versions



EXISTING ROW VERSIONS = 1

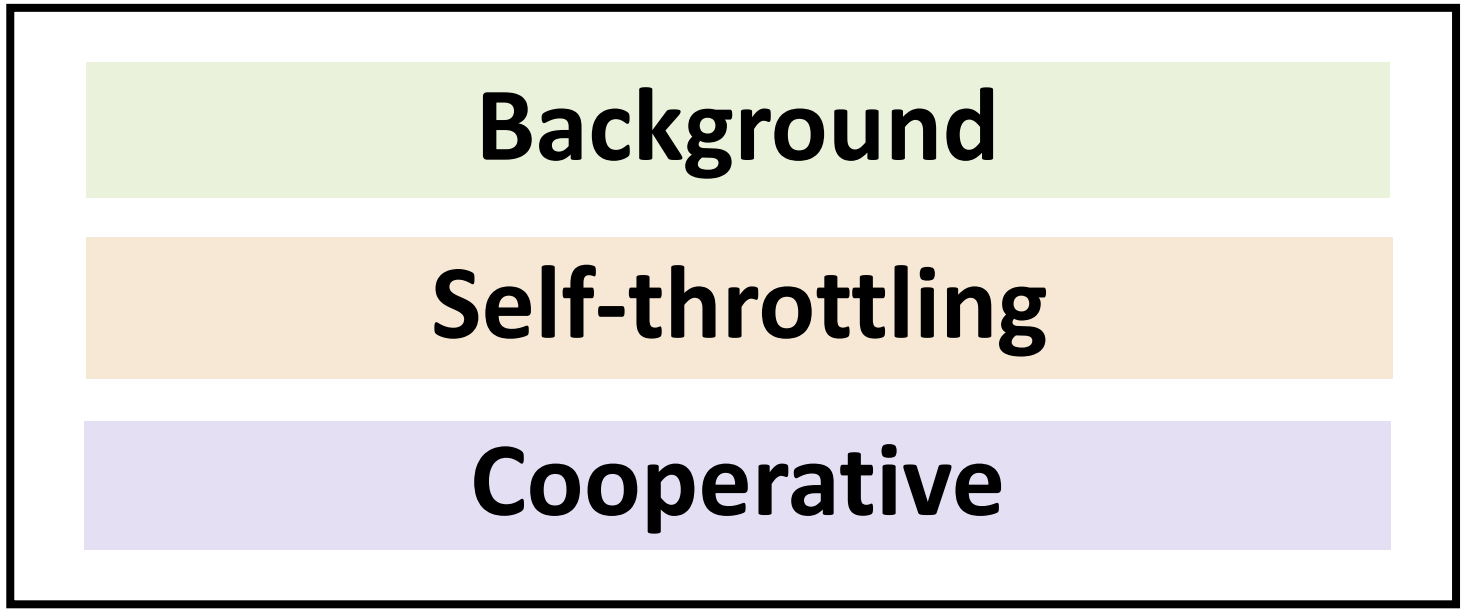
Row versions



TOTAL ROW VERSIONS = 2

GARBAGE COLLECTION

Garbage Collection attributes



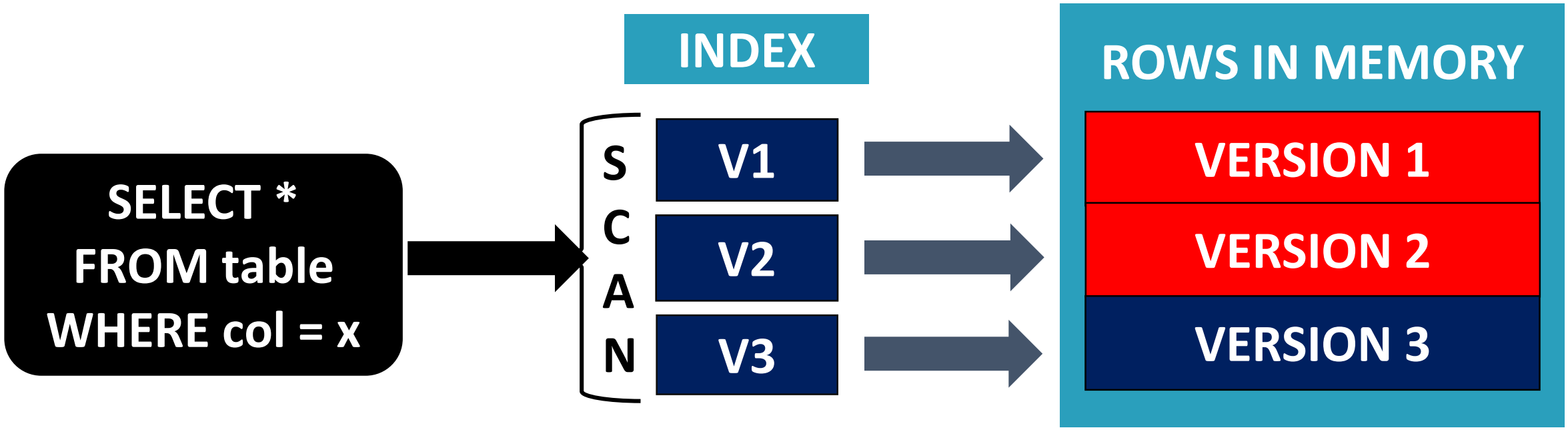
Garbage Collection process

IDENTIFY

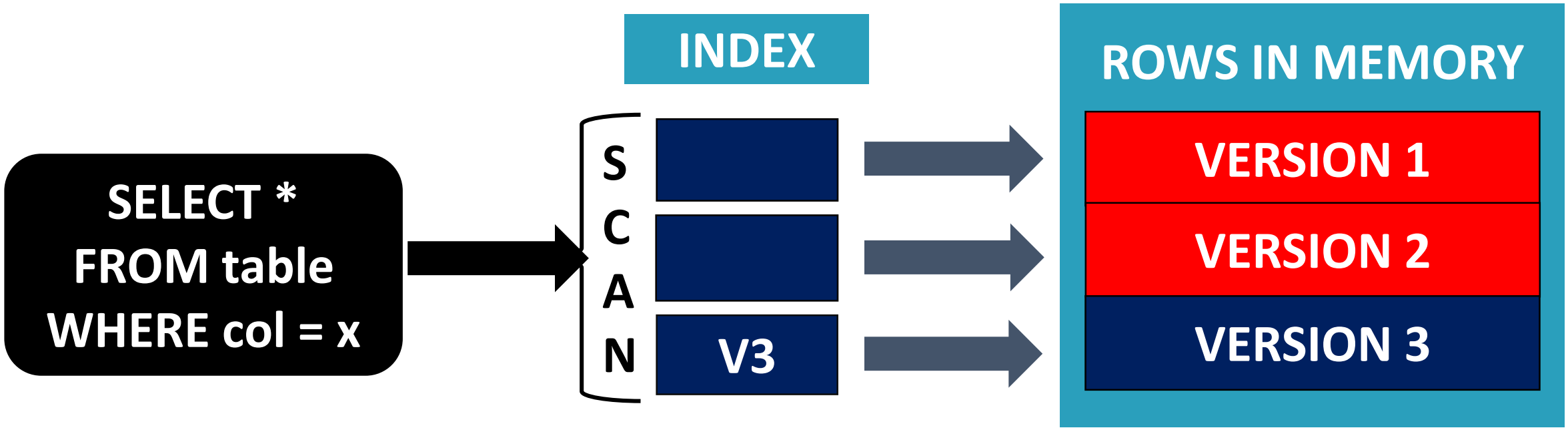
UNLINK

SCHEDULE

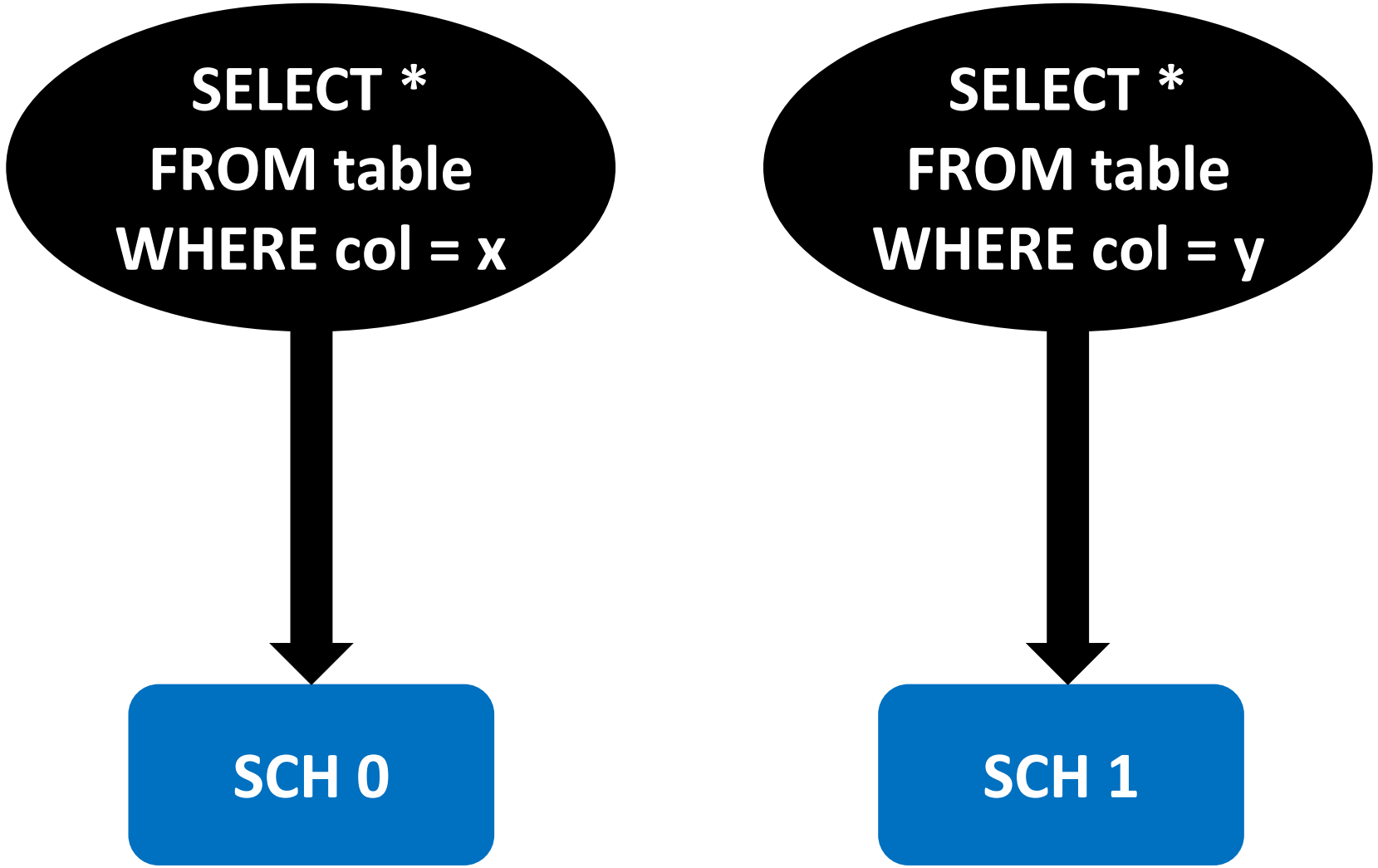
Garbage Collection: Identify



Garbage Collection: Unlink



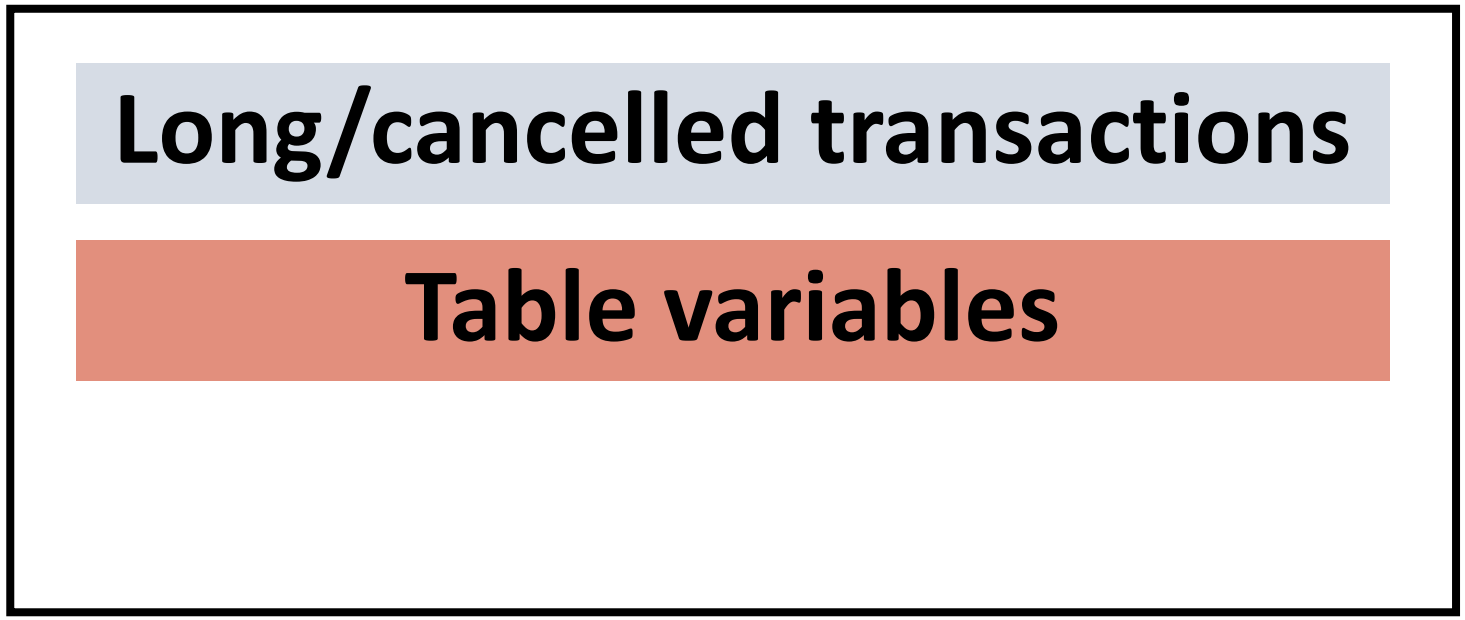
Garbage Collection: Schedule



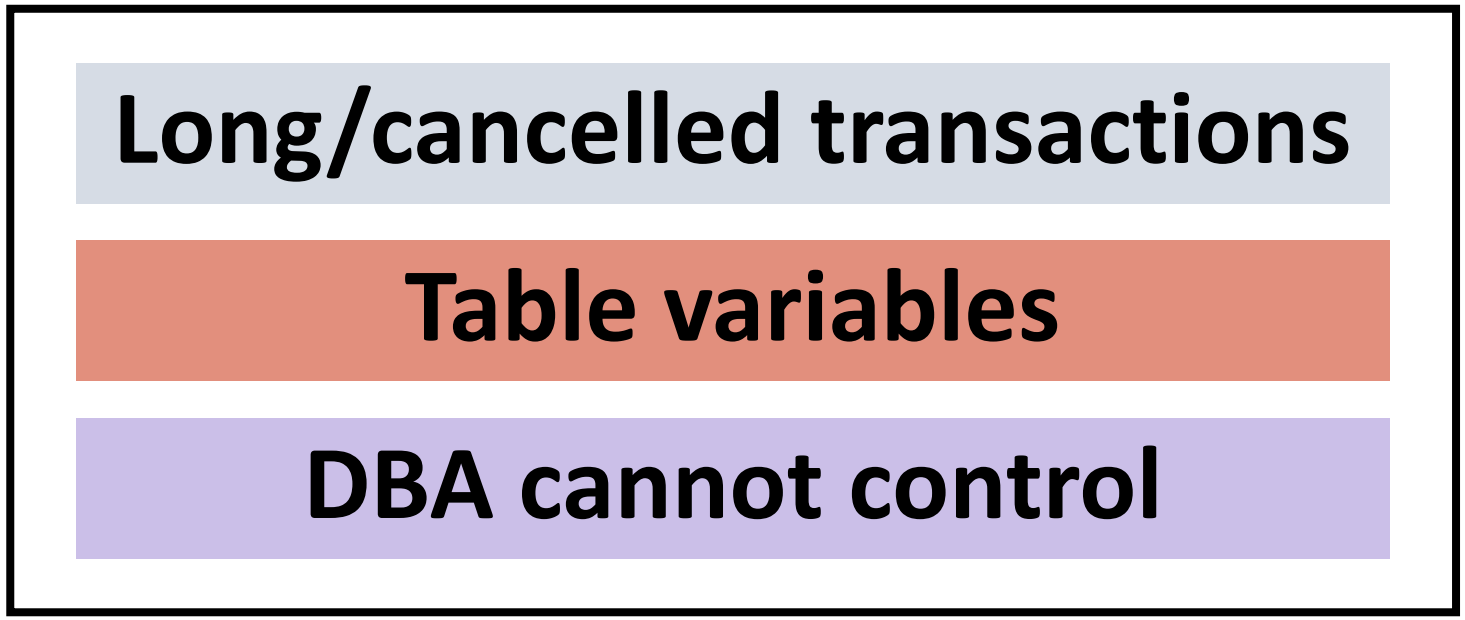
Garbage Collection – issues

Long/cancelled transactions

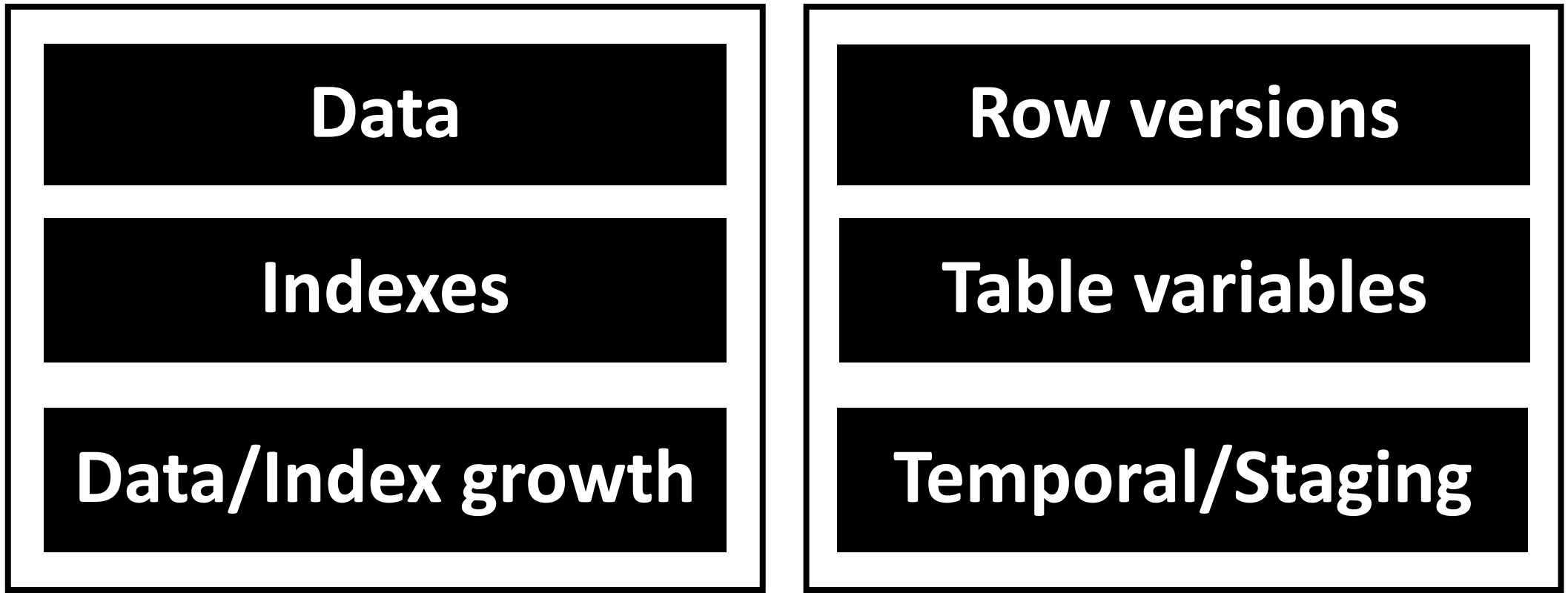
Garbage Collection – issues



Garbage Collection – issues

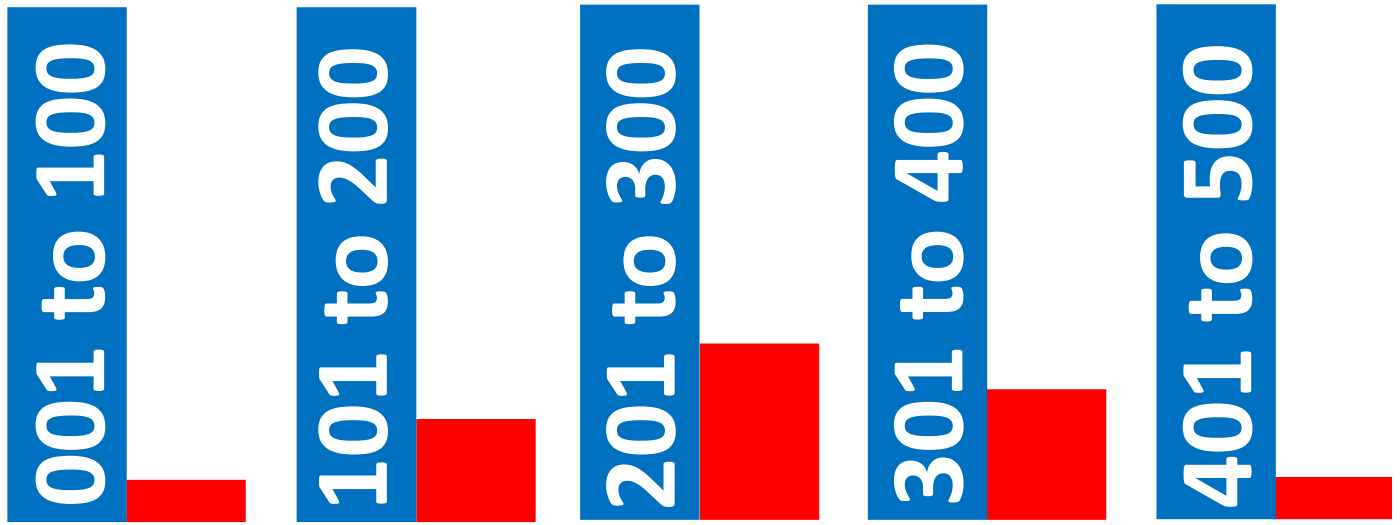


Capacity planning, Memory



FILE MERGE

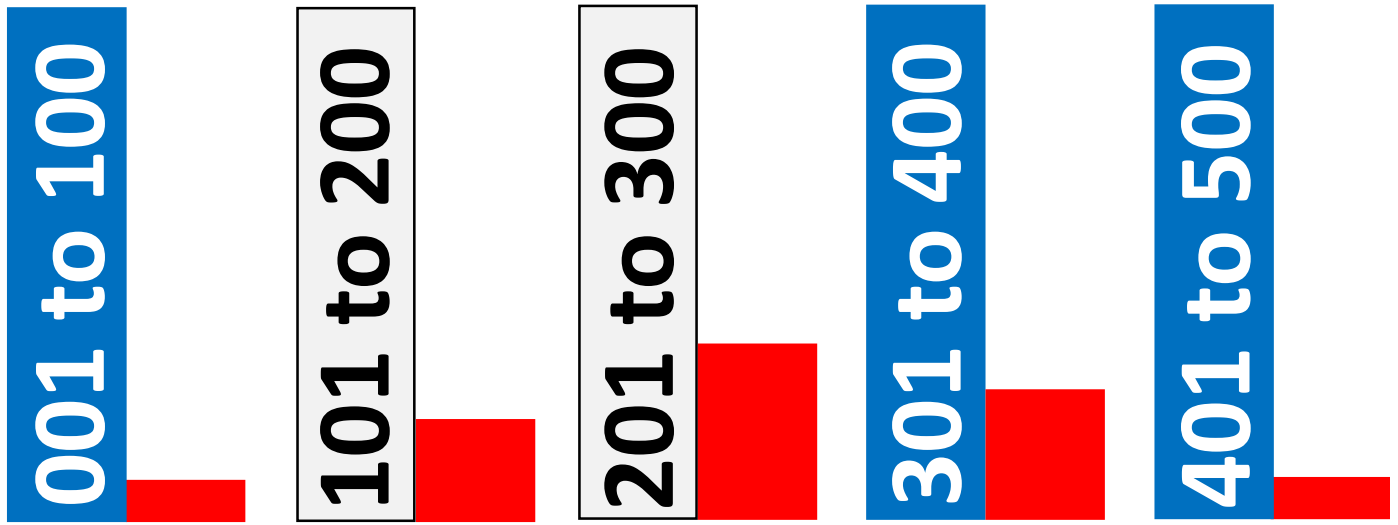
Garbage Collection – File Merge



DATA

DELTA

Garbage Collection – File Merge



DATA

DELTA

Garbage Collection – File Merge

DATA

DELTA

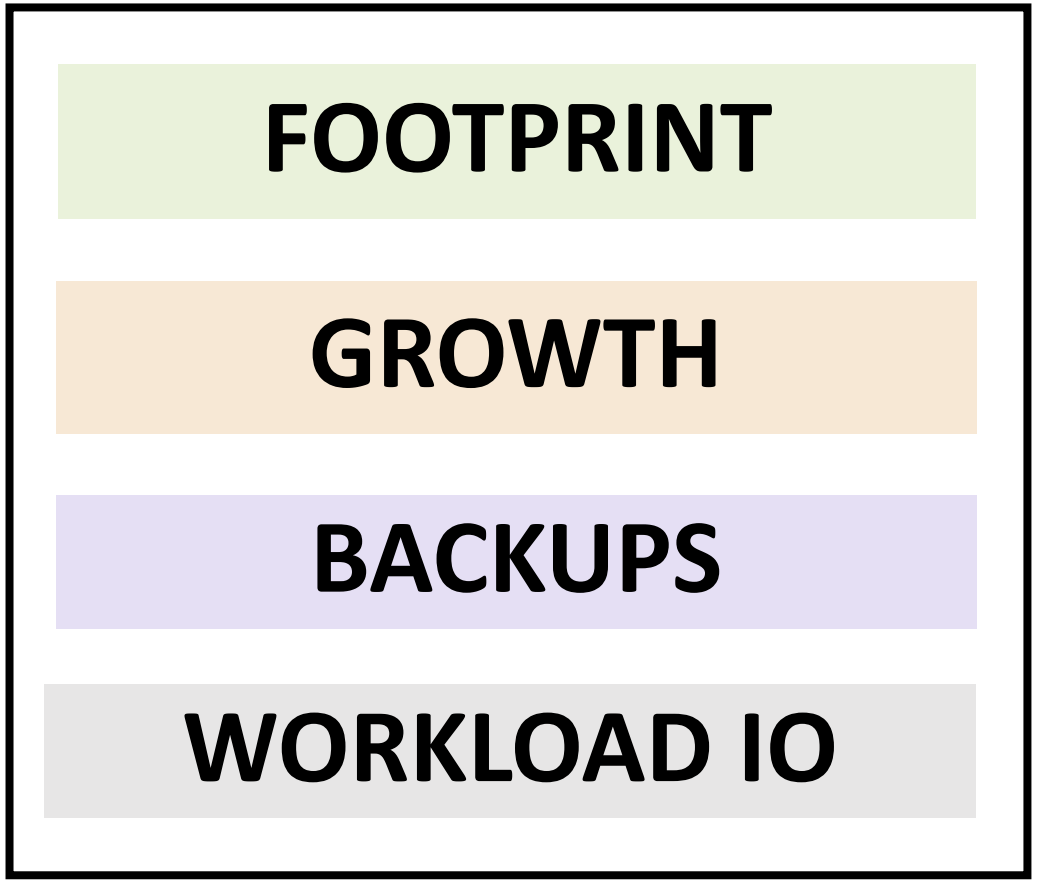
Garbage Collection – File Merge

sys.sp_xtp_checkpoint_force_garbage_collection

Capacity planning

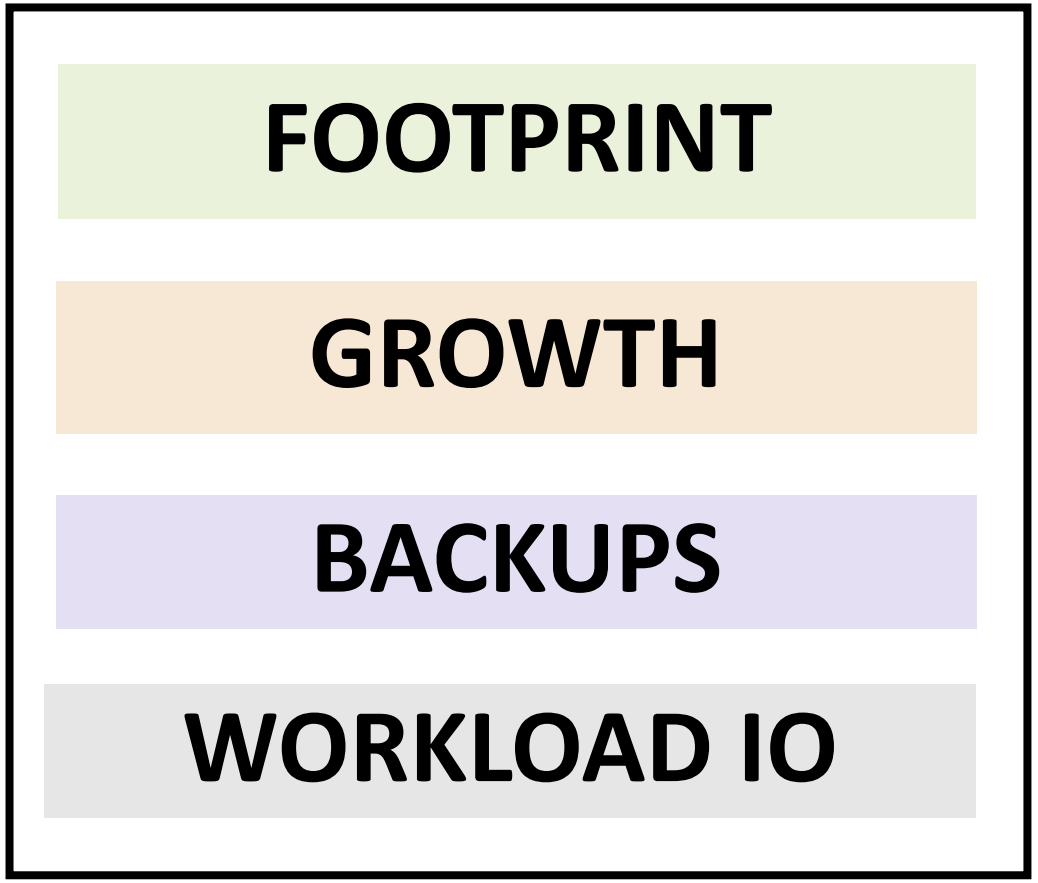
STORAGE

Capacity planning, Storage/IO



4x durable in-mem size

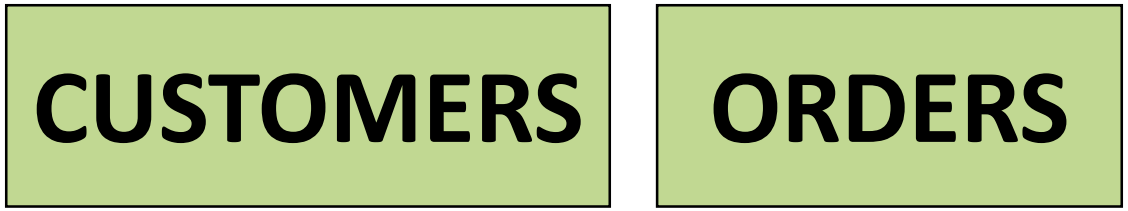
Capacity planning, Storage/IO



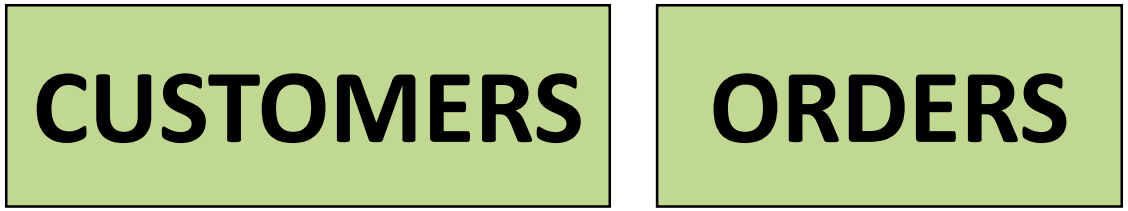
4x durable in-mem size

FAST/SEQUENTIAL

Sample migration

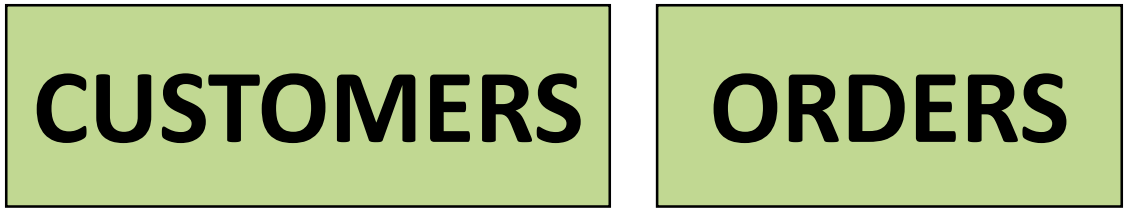


Sample migration



hot orders in-memory
cold orders on-disk

Sample migration



no orphaned child records
must validate Parent ID

Sample migration

CUSTOMERS

ORDERS

Customers table on-disk?

Customers table in-memory?

Migration scenario 1

PARENT ID VALID: HOT

PARENT ID VALID: COLD

NO ORPHANS

MEMORY

CUSTOMERS

HOT: ORDERS

DISK

COLD: ORDERS

Migration scenario 1

PARENT ID VALID: HOT

PARENT ID VALID: COLD

NO ORPHANS

MEMORY

DISK

COLD: ORDERS

Migration scenario 1

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD

NO ORPHANS

MEMORY

CUSTOMERS ← FK — HOT: ORDERS

DISK

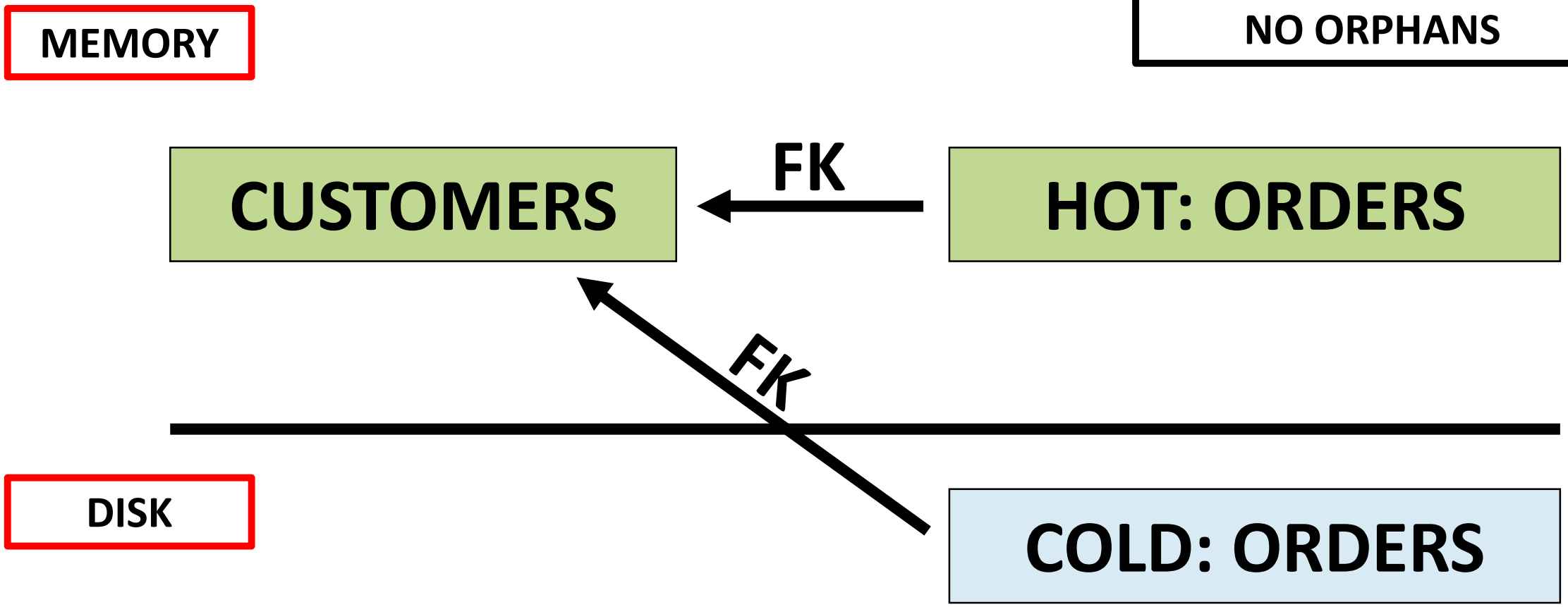
COLD: ORDERS

Migration scenario 1

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD

NO ORPHANS

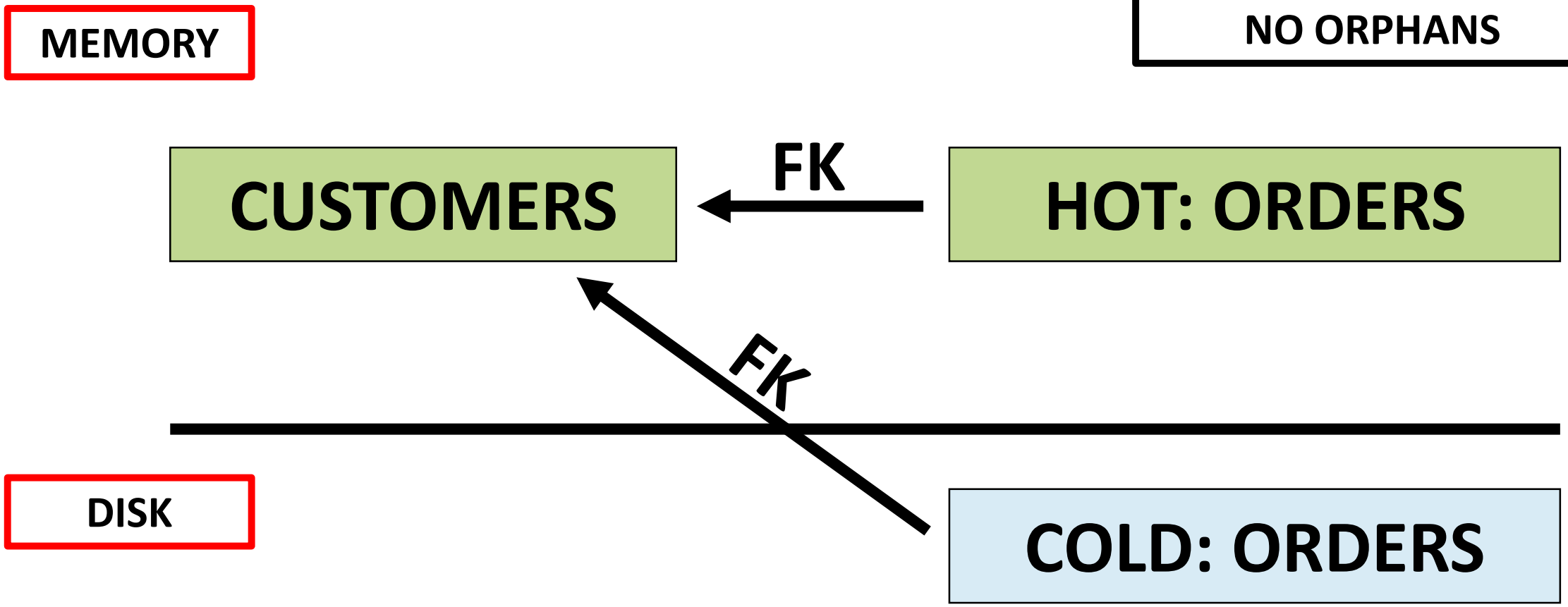


Migration scenario 1

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD

NO ORPHANS ✗



Migration scenario 2

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD

NO ORPHANS

MEMORY

CUSTOMERS ← FK — HOT: ORDERS

DISK

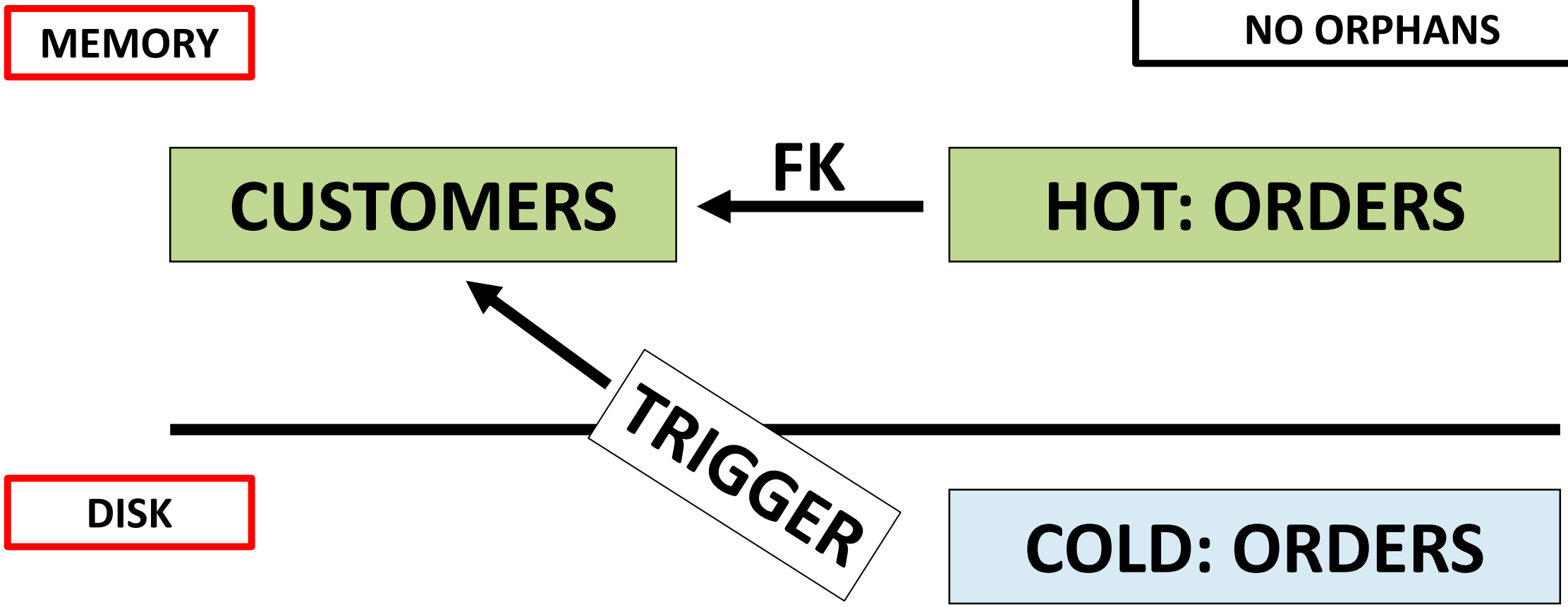
COLD: ORDERS

Migration scenario 2

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD

NO ORPHANS

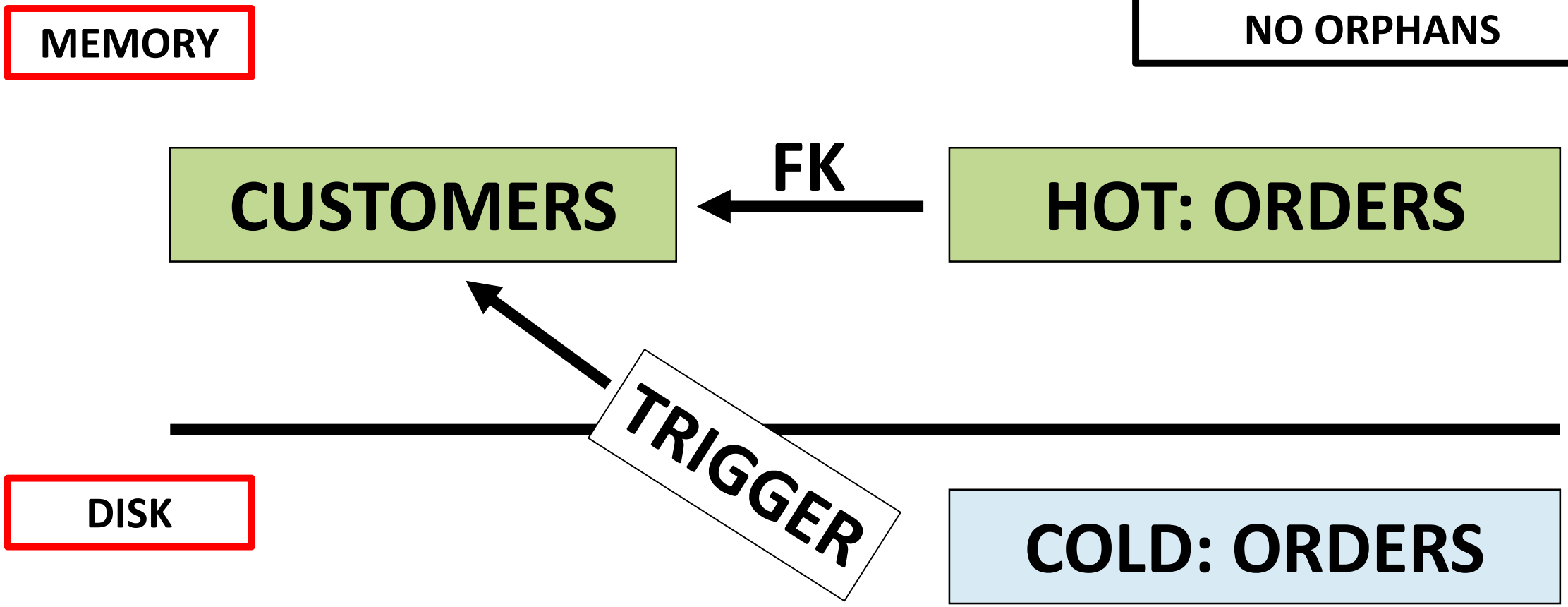


Migration scenario 2

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD ✓

NO ORPHANS

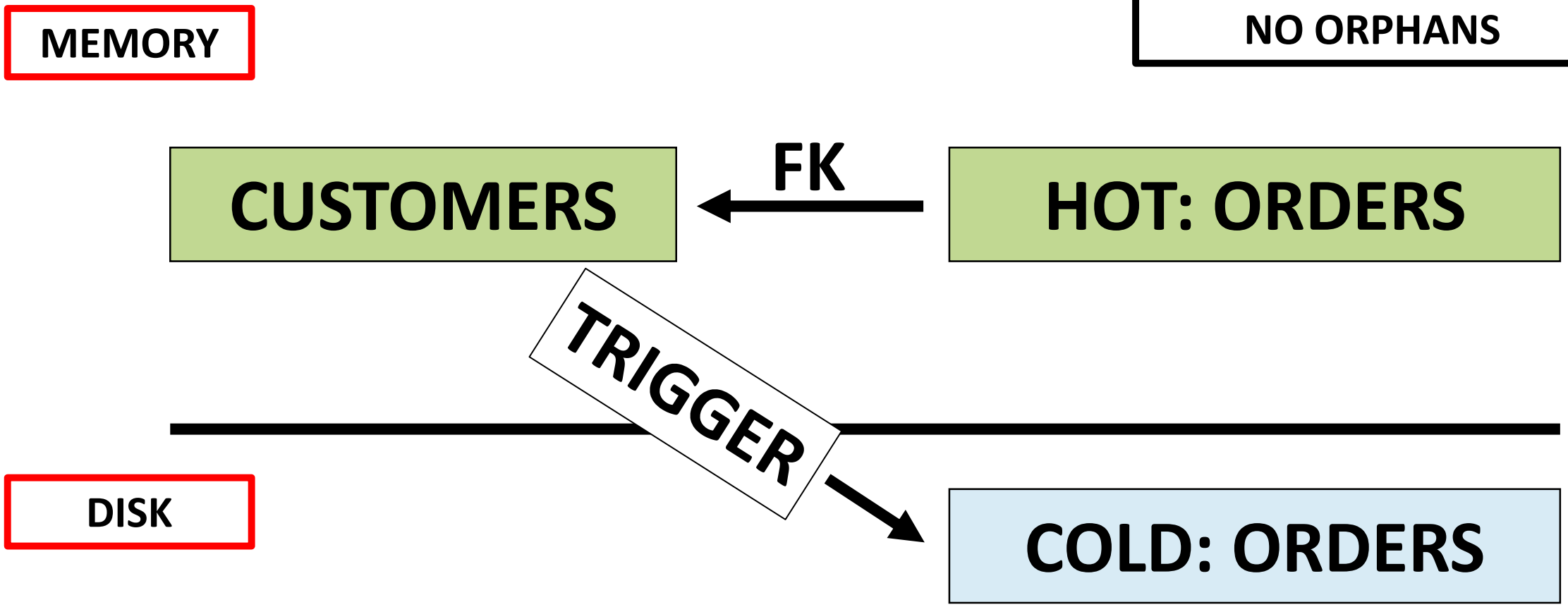


Migration scenario 2

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD ✓

NO ORPHANS

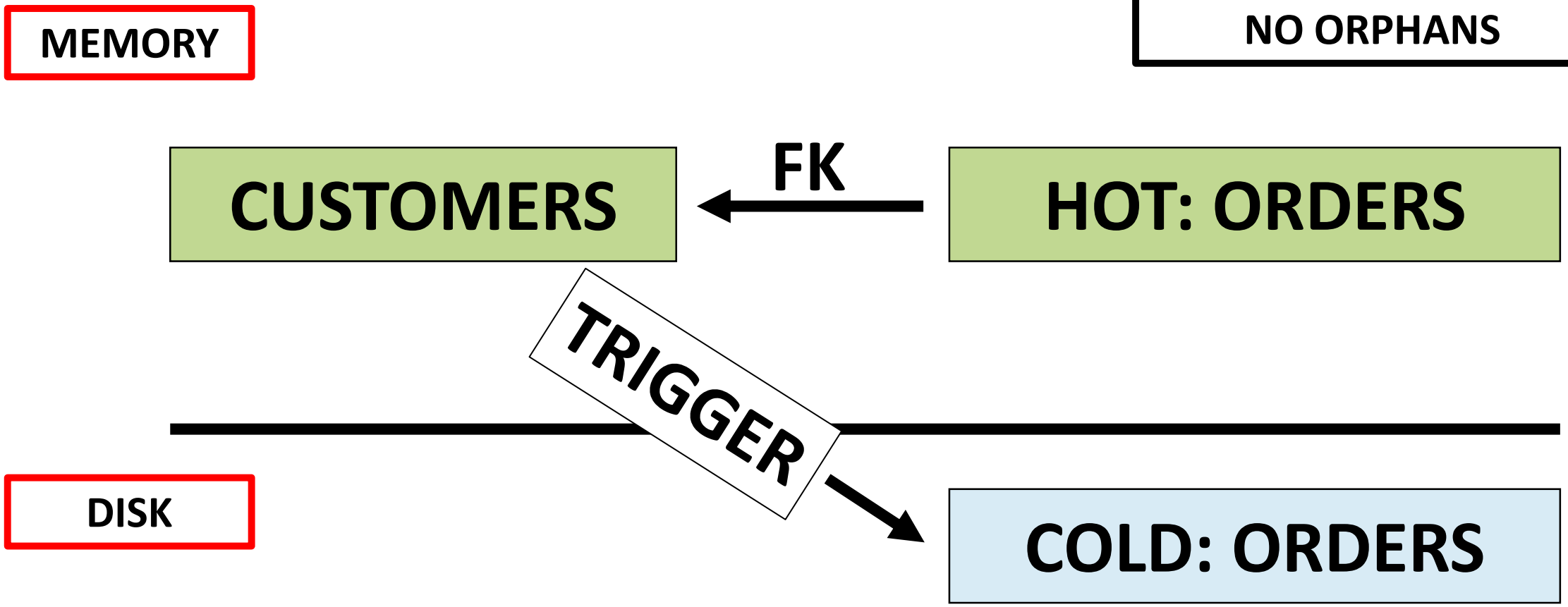


Migration scenario 2

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD ✗

NO ORPHANS



Migration scenario 3

PARENT ID VALID: HOT

PARENT ID VALID: COLD ✓

NO ORPHANS

MEMORY

HOT: ORDERS

DISK

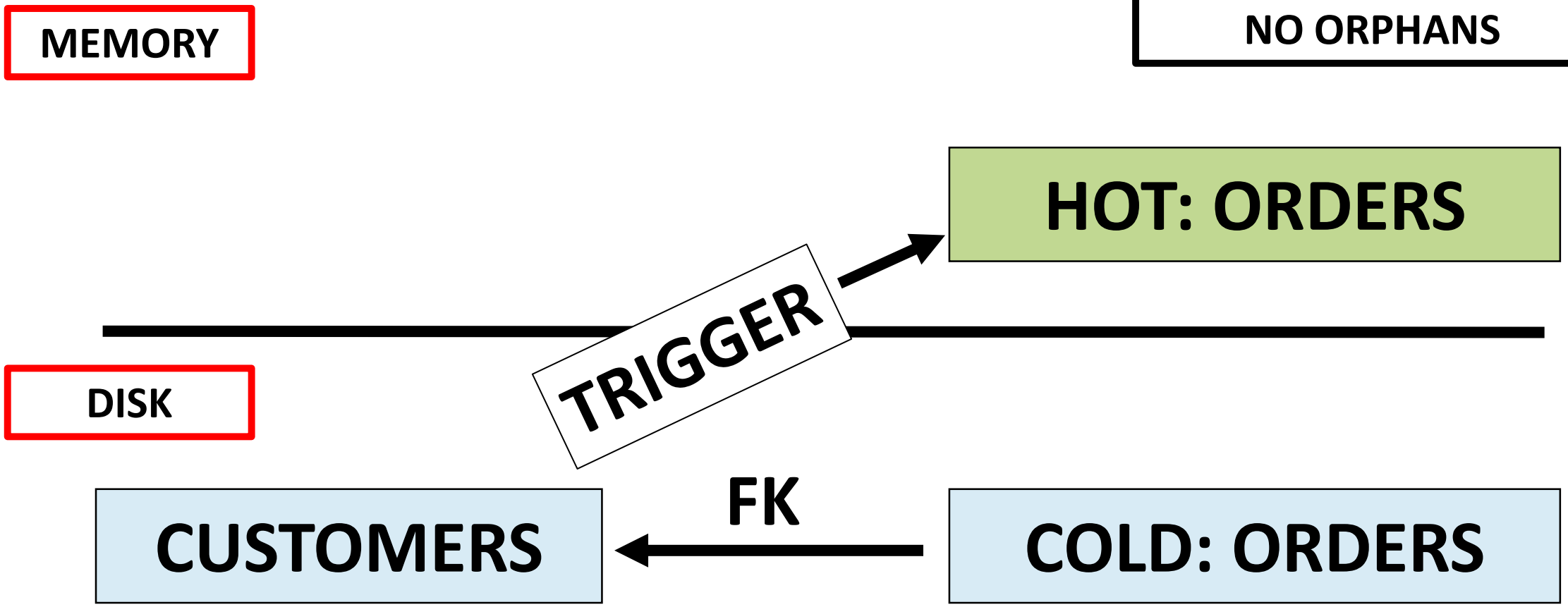
CUSTOMERS ← FK — COLD: ORDERS

Migration scenario 3

PARENT ID VALID: HOT

PARENT ID VALID: COLD ✓

NO ORPHANS

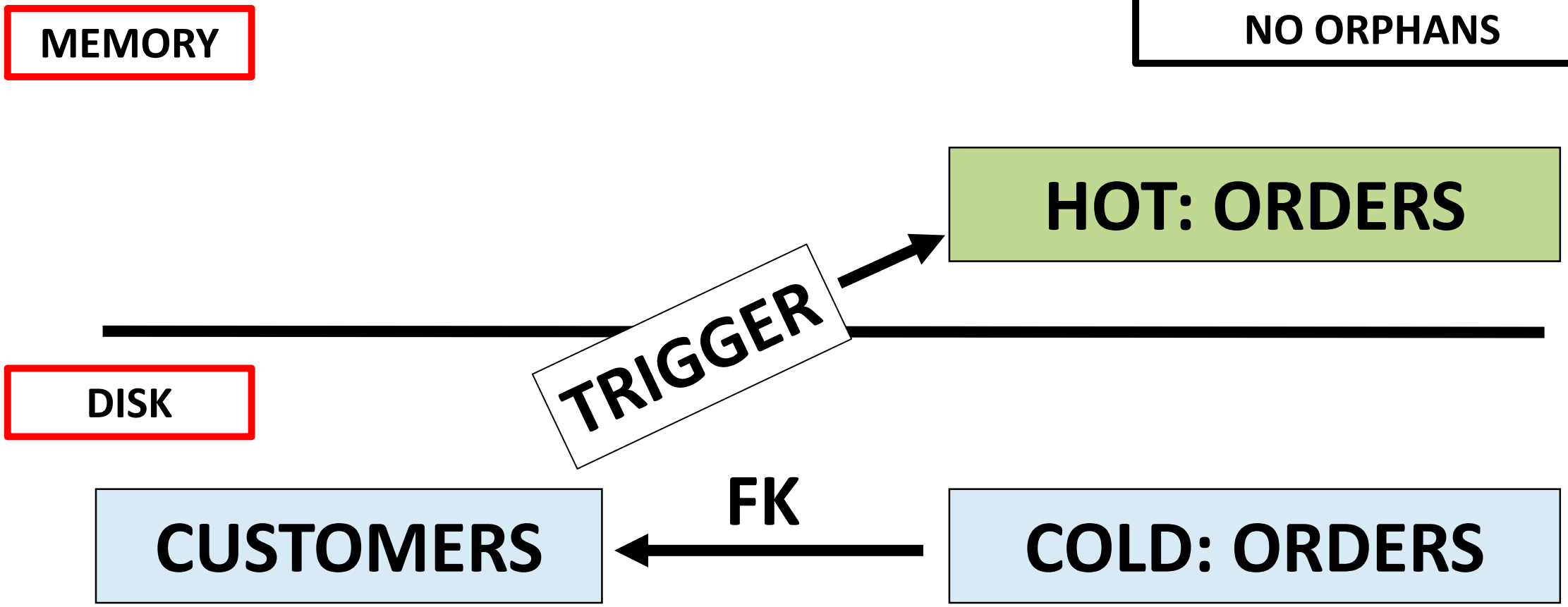


Migration scenario 3

PARENT ID VALID: HOT

PARENT ID VALID: COLD ✓

NO ORPHANS ✓

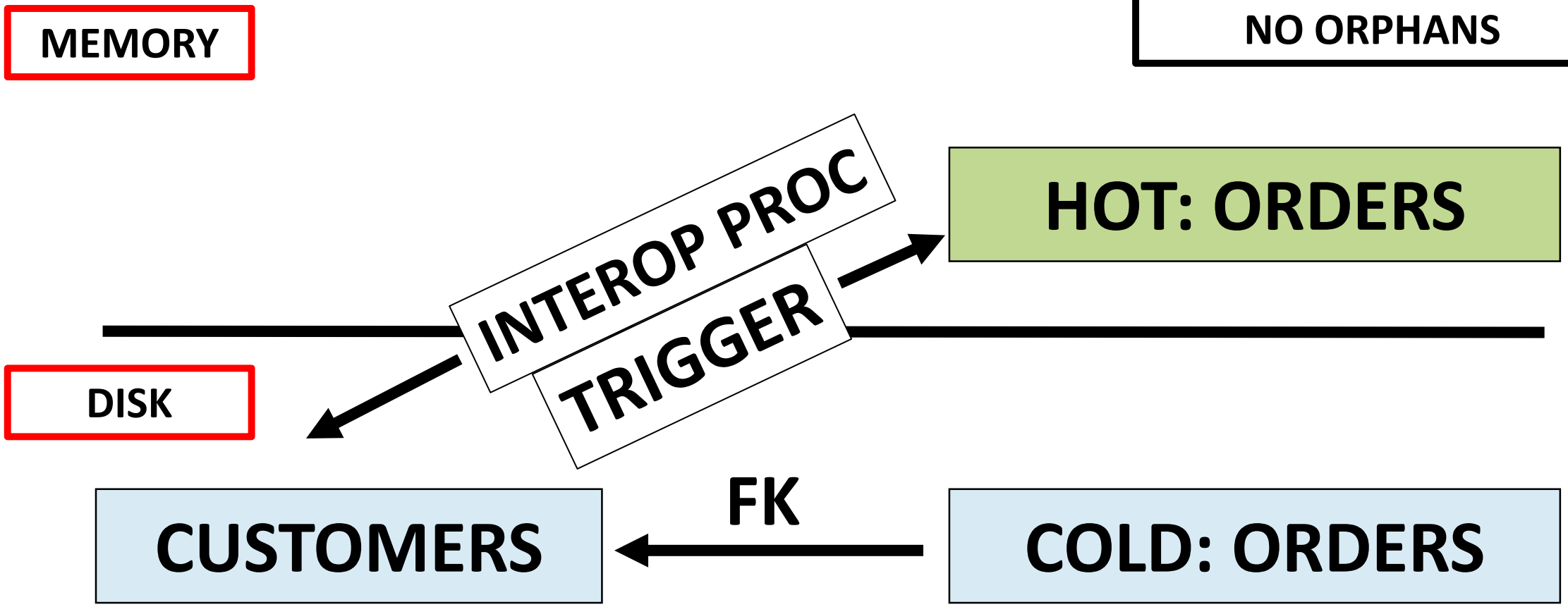


Migration scenario 3

PARENT ID VALID: HOT

PARENT ID VALID: COLD ✓

NO ORPHANS ✓

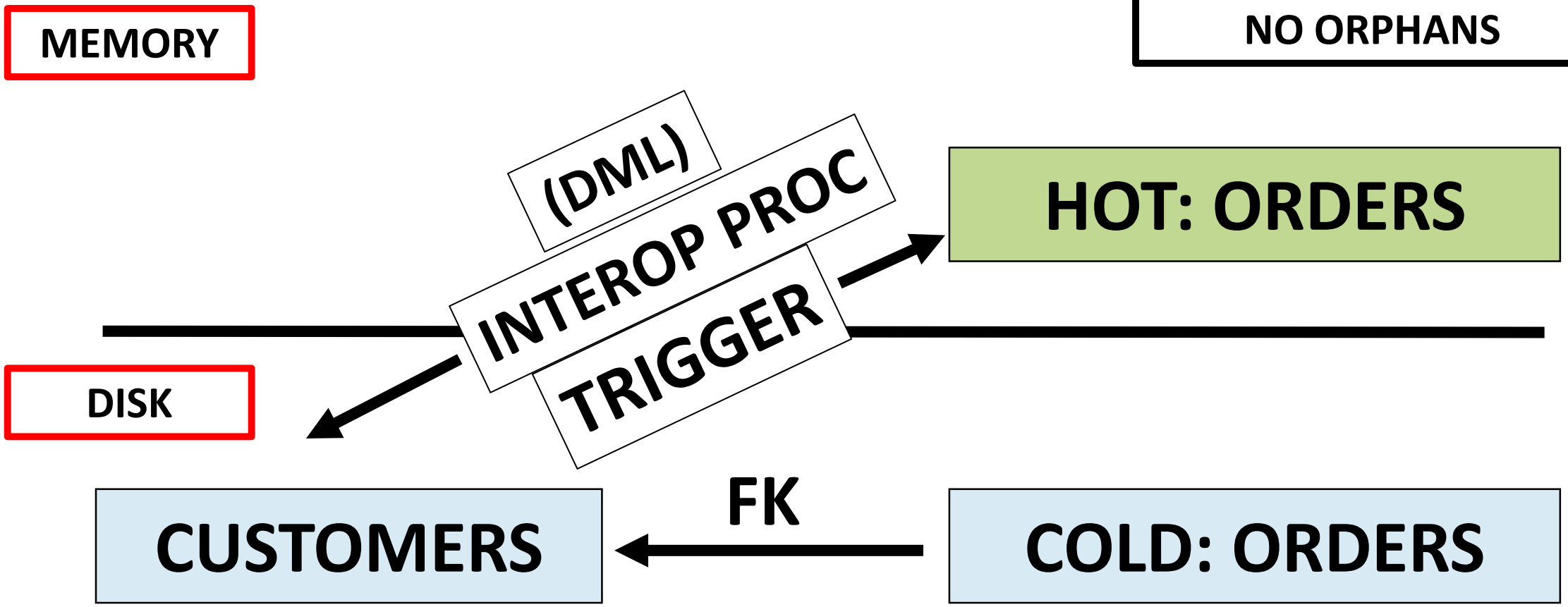


Migration scenario 3

PARENT ID VALID: HOT ✓

PARENT ID VALID: COLD ✓

NO ORPHANS ✓



Migration analysis

(workload)

Planning

1. Identify bottleneck(s)

Planning

1. Identify bottleneck(s)
2. Determine if In-Memory might help

Planning

1. Identify bottleneck(s)
2. Determine if In-Memory might help
3. POC

Capacity planning

- Memory = 2x in-mem size
- Total IOPS = 3x workload IOPS
- Storage = 4x durable in-mem size

White paper review

White paper review

- Indexes are never stored on disk

White paper review

- ~~Indexes are never stored on disk~~

White paper review

- ~~Indexes are never stored on disk~~
- DELETE and UPDATE operations will generate row versions

White paper review

- ~~Indexes are never stored on disk~~
- ~~DELETE and~~ UPDATE operations will generate row versions

White paper review

- ~~Indexes are never stored on disk~~
- ~~DELETE and~~ UPDATE operations will generate row versions
- Data is on disk to be used when restarting your SQL Server

White paper review

- ~~Indexes are never stored on disk~~
- ~~DELETE and~~ UPDATE operations will generate row versions
- Data is on disk to be used when restarting your ~~SQL Server~~ database

White paper review

- File creation

White paper review

- ~~File creation~~

White paper review

- ~~File creation~~
- 2TB limit

White paper review

- ~~File creation~~
- ~~2TB limit~~

White paper review

- ~~File creation~~
- ~~2TB limit~~
- Parallelism

White paper review

- ~~File creation~~
- ~~2TB limit~~
- ~~Parallelism~~

Parallelism

ALL INDEX TYPES

TABLES/VARIABLES

SAMPLED STATS

DML SOURCE

Parallelism

NATIVELY COMPILED ✗

DML TARGET ✗

KEY TAKEAWAYS

Workload evaluation
Use cases
Architecture

t: @NedOtter e: ned@nedotter.com
www.nedotter.com

Will my workload run faster with In-Memory OLTP?

Ned Otter | SQL Strategist