

# PERFORMANCE IMPROVEMENTS IN SQL SERVER 2019

SQL Saturday Minnesota  
December 12, 2020



# ABOUT ME

In IT for 21 years, 13 of it working with SQL Server  
Senior Data Engineer at Concurrency, Inc.

Learning more about Azure every day

Lover of all things internal to SQL Server

When not working with the Microsoft Data Platform I love to read, volunteer at the Art Institute of Chicago, and hang out with my cat

Twitter - @skreebydba

Email - [skreebydba@gmail.com](mailto:skreebydba@gmail.com)

Blog - [www.skreebydba.com](http://www.skreebydba.com)



# WHAT WE WILL COVER

Transaction log

Current crash recovery process

Accelerated Database Recovery (ADR)

ADR crash recovery process

In-memory tempdb metadata

Persistent Memory



# TRANSACTIONS

Unit of work in the database

All transactions begin

Transactions can commit or rollback

Default behavior is auto-commit



# TRANSACTION LOG

Records all changes to the database

Changes written to the log buffer in memory

SQL Server uses write-ahead logging (WAL)

Log buffer is flushed to disk on COMMIT or when it fills up

Data pages associated with transactions can remain in memory



# TRANSACTION LOG ARCHITECTURE

Transaction log contains logical units called Virtual Log Files (VLF)

VLFs can be active or free

VLFs containing log records that may be needed must be active

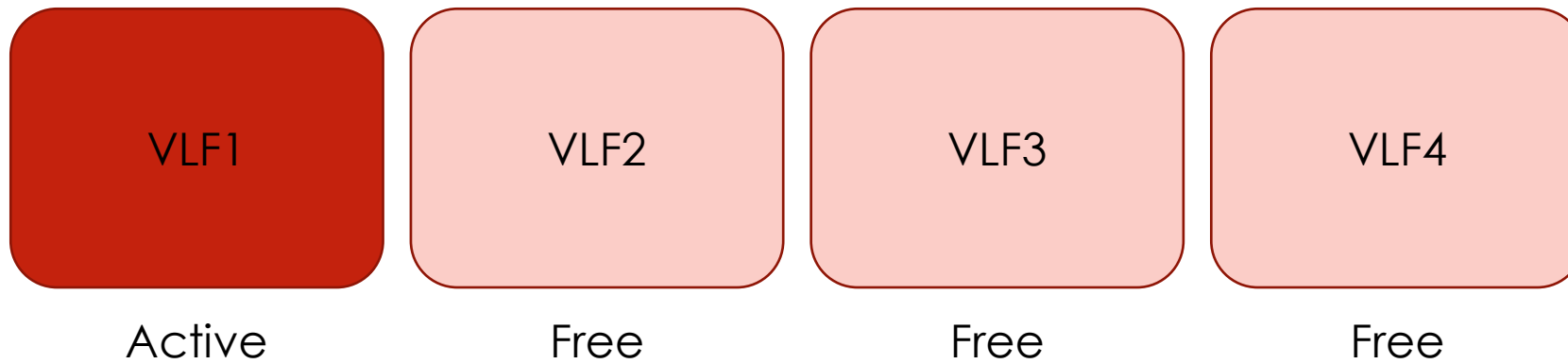
Always one active VLF

In SIMPLE recovery, VLFs freed by CHECKPOINT operation

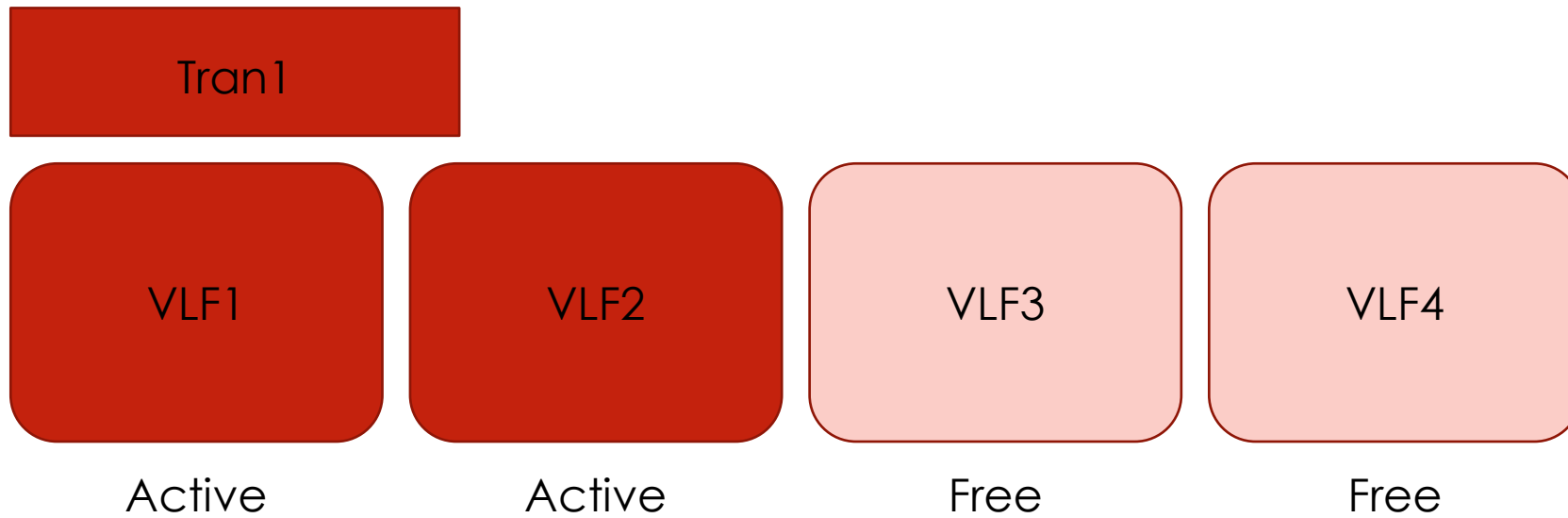
In FULL recovery, VLFs freed by LOG backup

Log records are also needed for rollback and high availability

# VIRTUAL LOG FILES

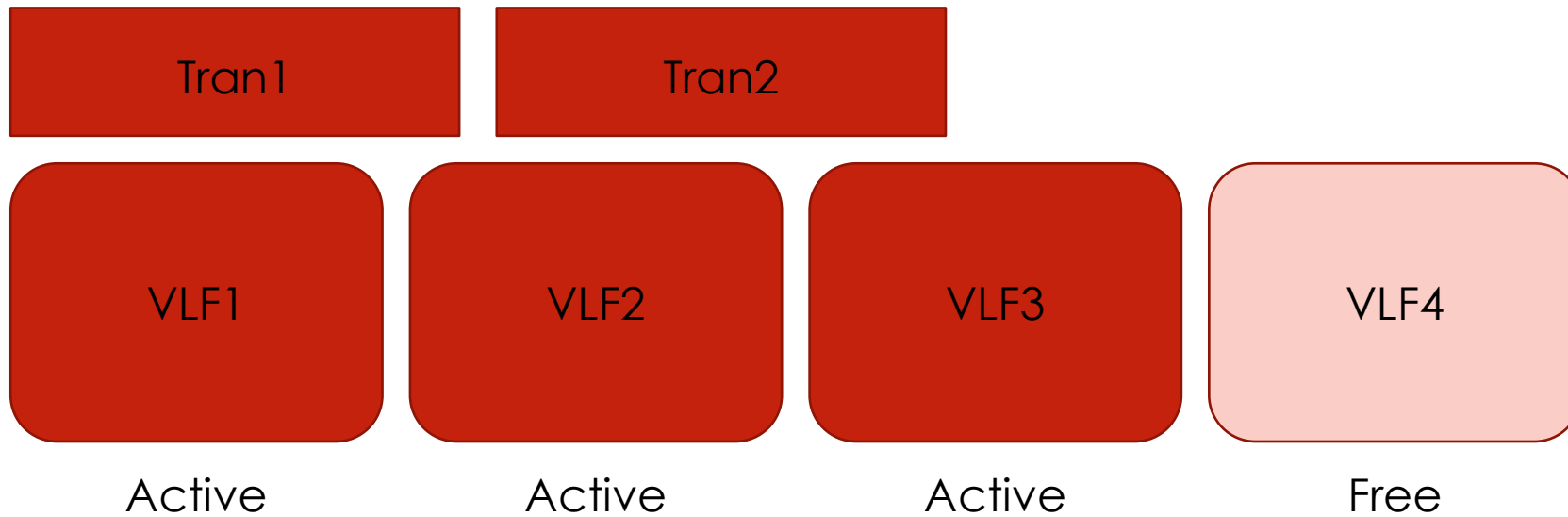


# TRANSACTION 1 COMMITS

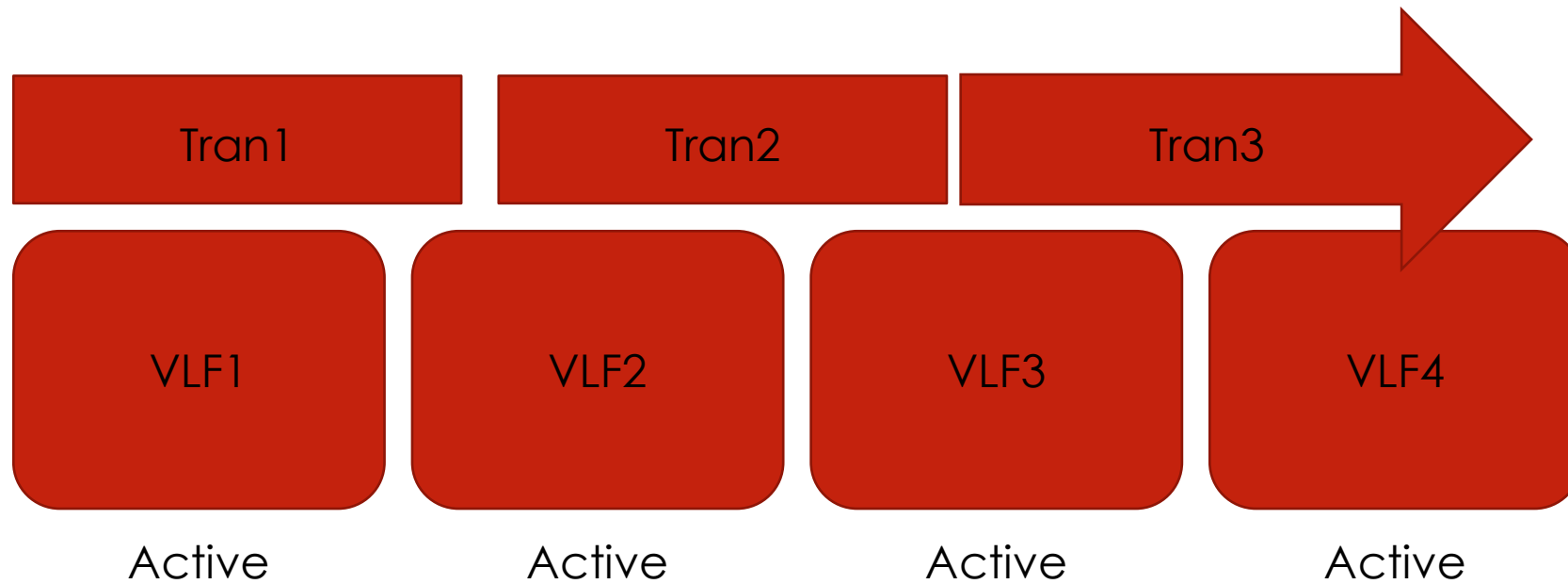




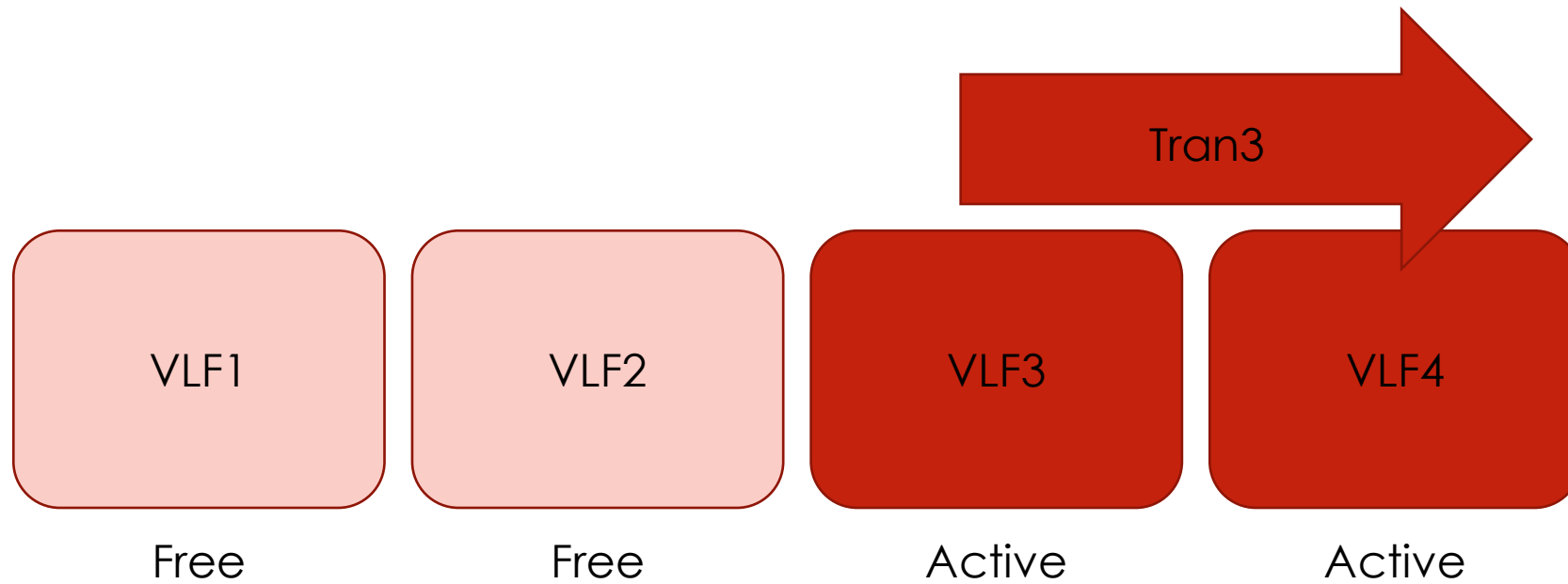
# TRANSACTION 2 COMMITS



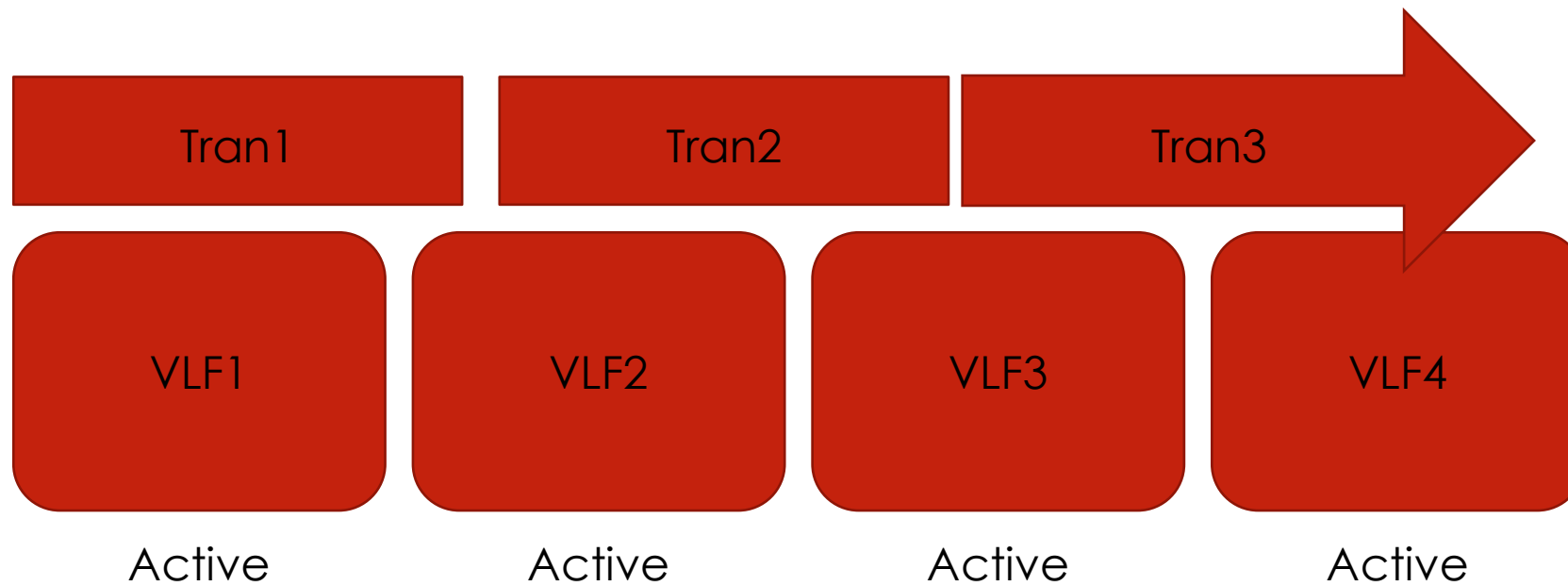
# TRANSACTION 3 BEGINS



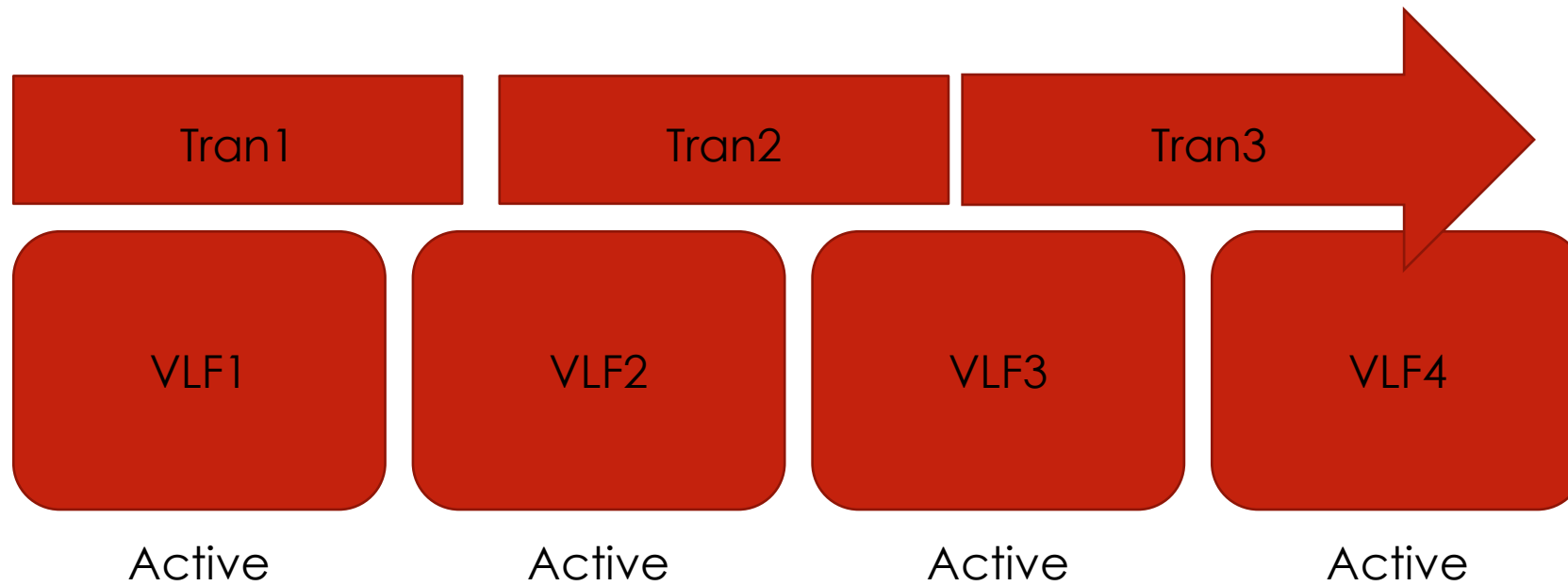
# LOG BACKUP OR CHECKPOINT




# NO LOG BACKUP OR CHECKPOINT



# WITHOUT AUTOGROW

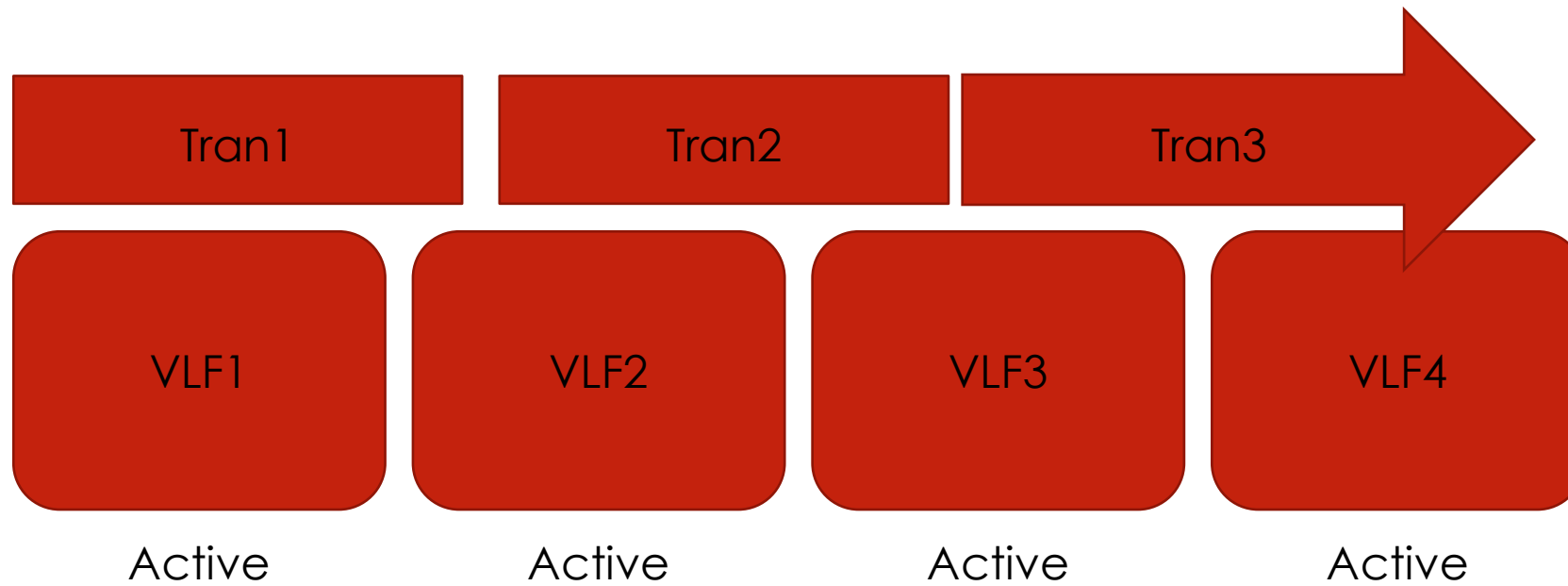




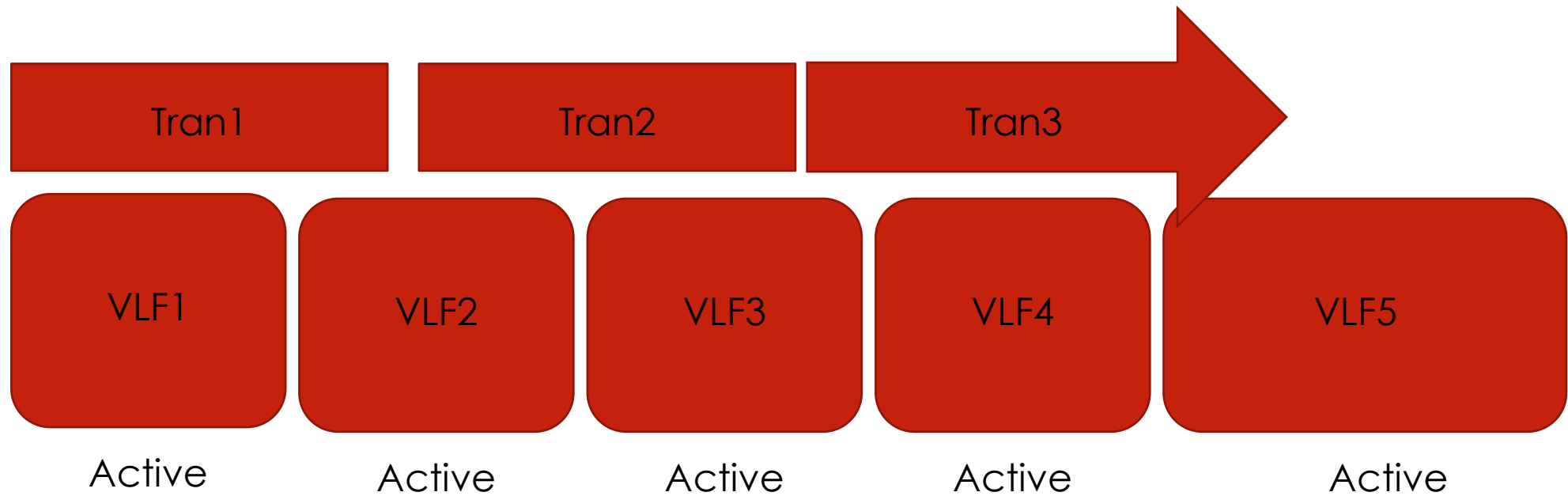
# COMPUTER MALFUNCTION

Image Source: <https://tarnmoor.files.wordpress.com/2018/12/PICComputerMalfunction2.jpg>

# WITH AUTOGROW



# AUTOGROW ADDS VLF(S)







# CRASH RECOVERY

SQL Server uses the transaction log to maintain consistency and durability  
After the SQL Server service restarts, each database transaction log is scanned

# CURRENT CRASH RECOVERY PROCESS

3 phases

Analysis – Scans log from last checkpoint searching for  
Transactions written to log file but not to data file  
Transactions not committed

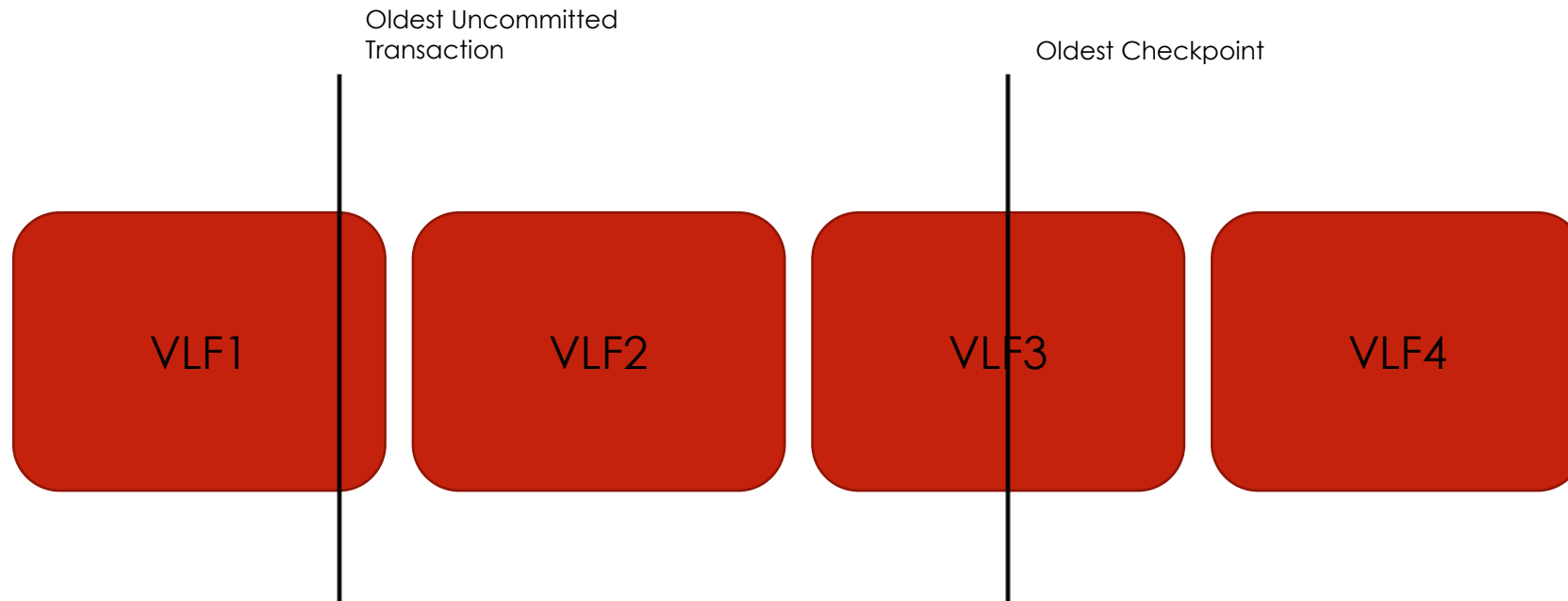
Redo

Committed transactions hardened to disk in data file

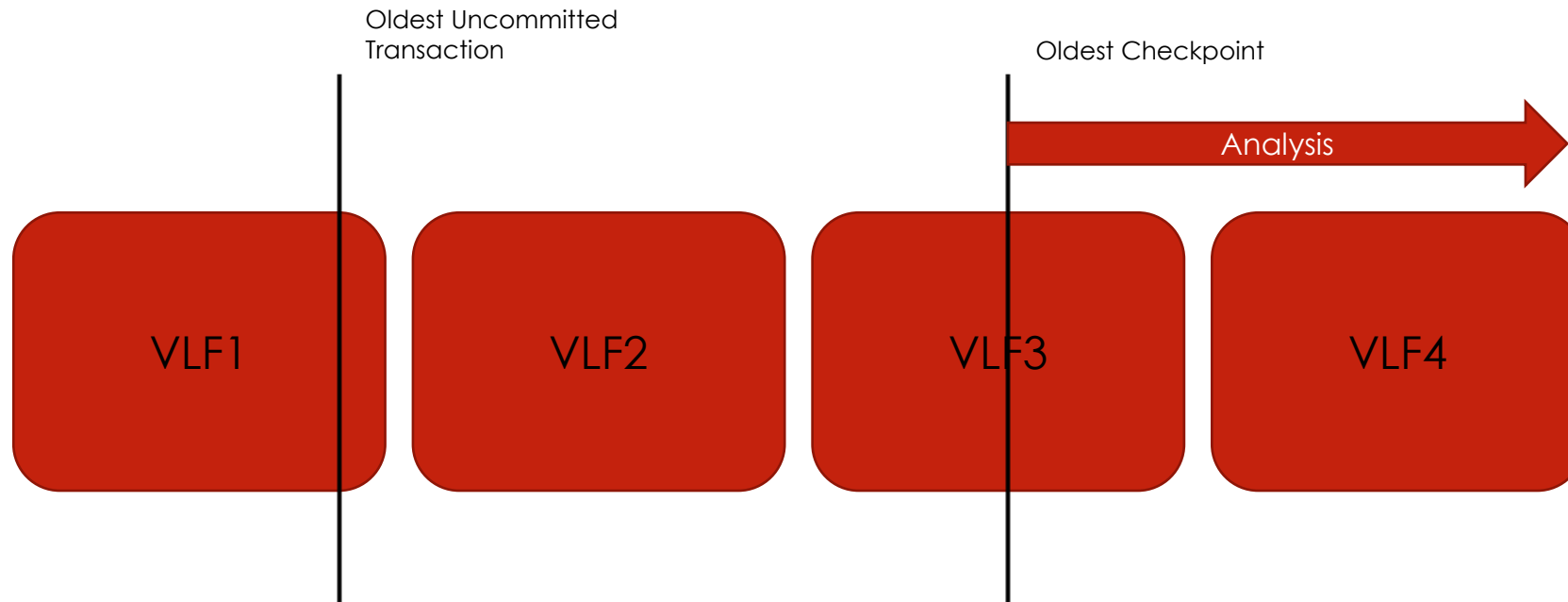
Undo

Uncommitted transactions rolled back

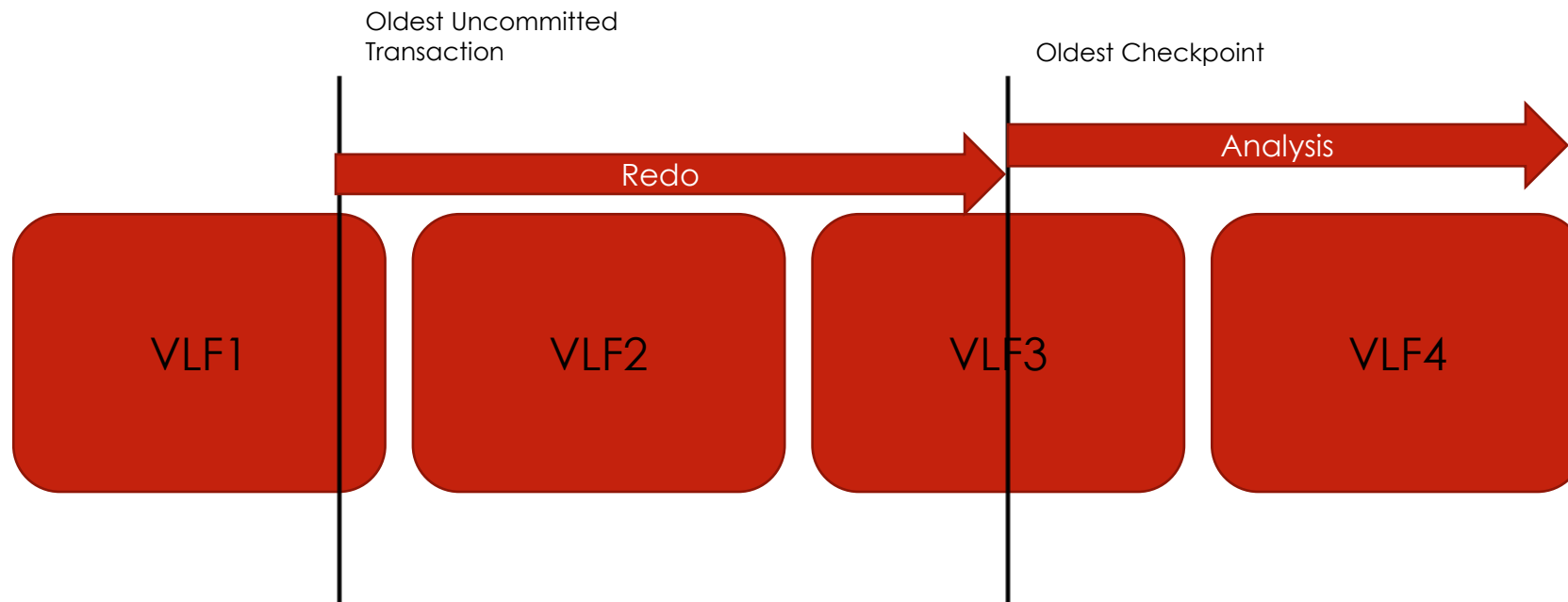
# CRASH RECOVERY



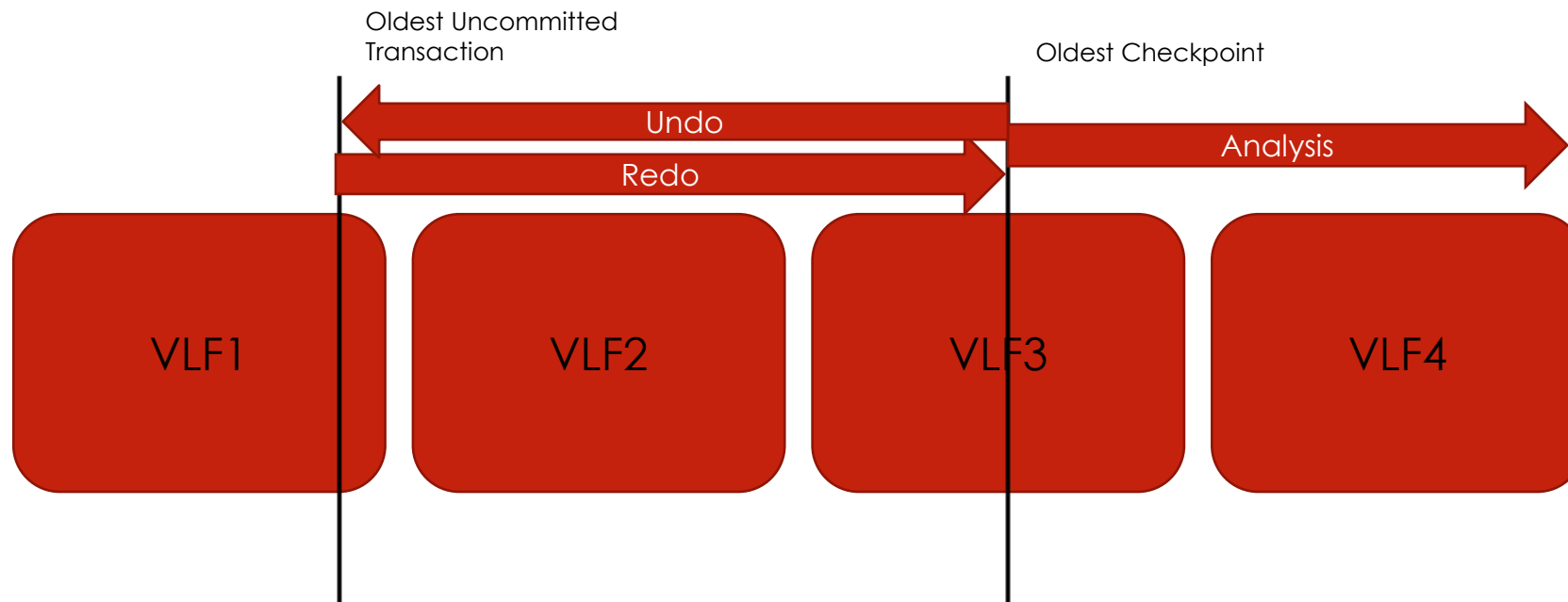
# ANALYSIS



# REDO



# UNDO



# ACCELERATED DATABASE RECOVERY – NEW CONCEPTS

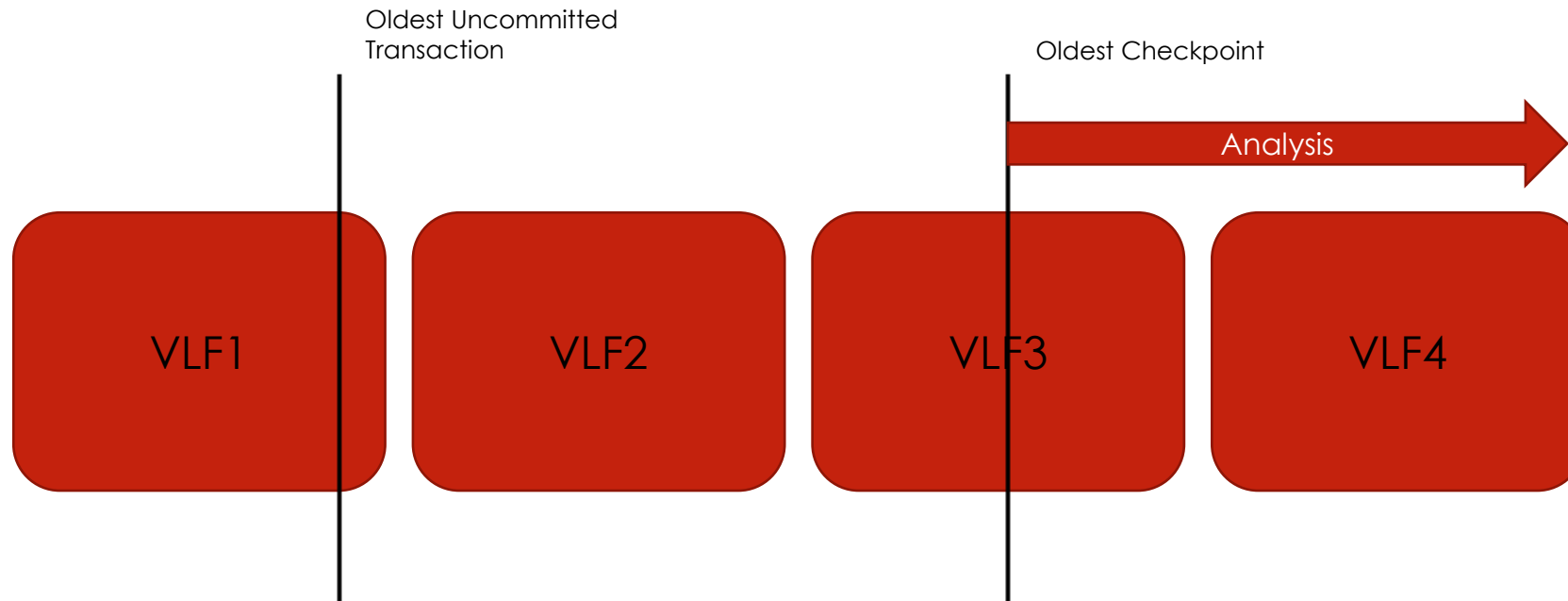
Persisted Version Store (PVS) – contains previous versions of modified rows, stored in the user database

Logical revert – On rollback, running transactions pull row version from the PVS

sLog – In-memory log stream that stores non-versioned activity (system metadata changes, locks for DDL, cache invalidation)

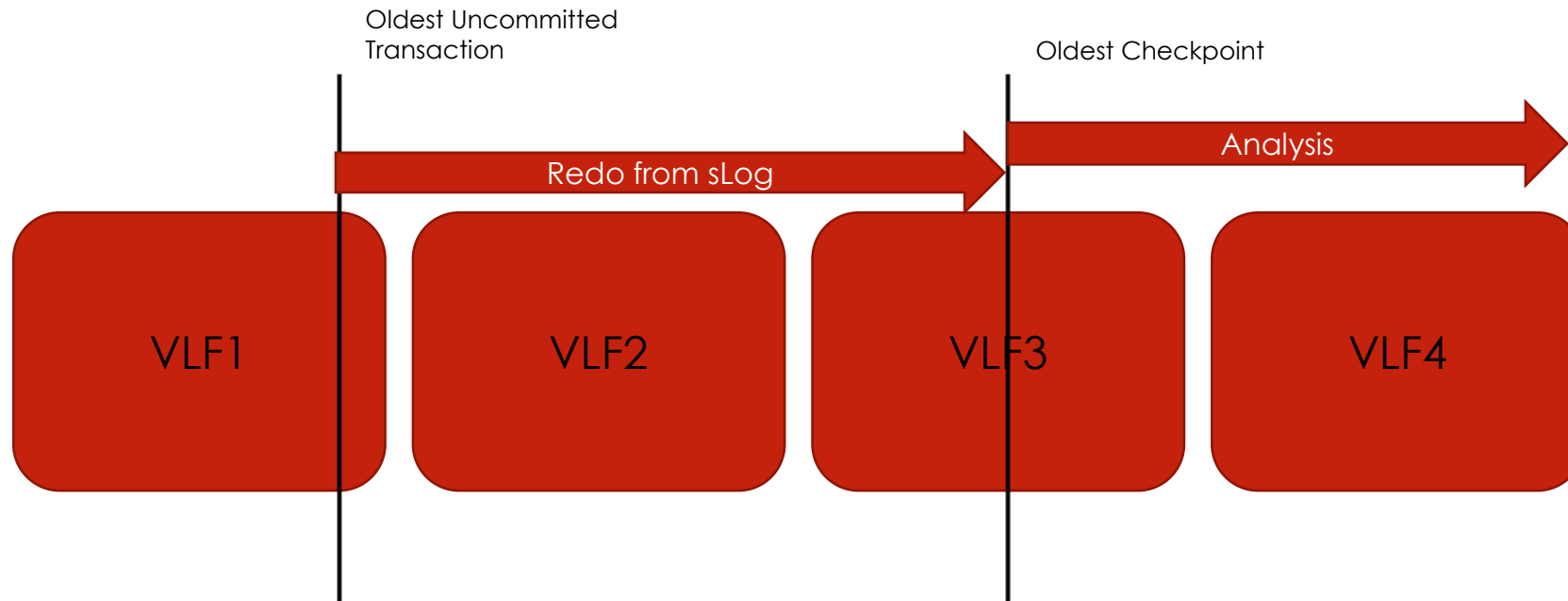
Cleaner – periodic process that cleans up unneeded row versions

# ADR ANALYSIS

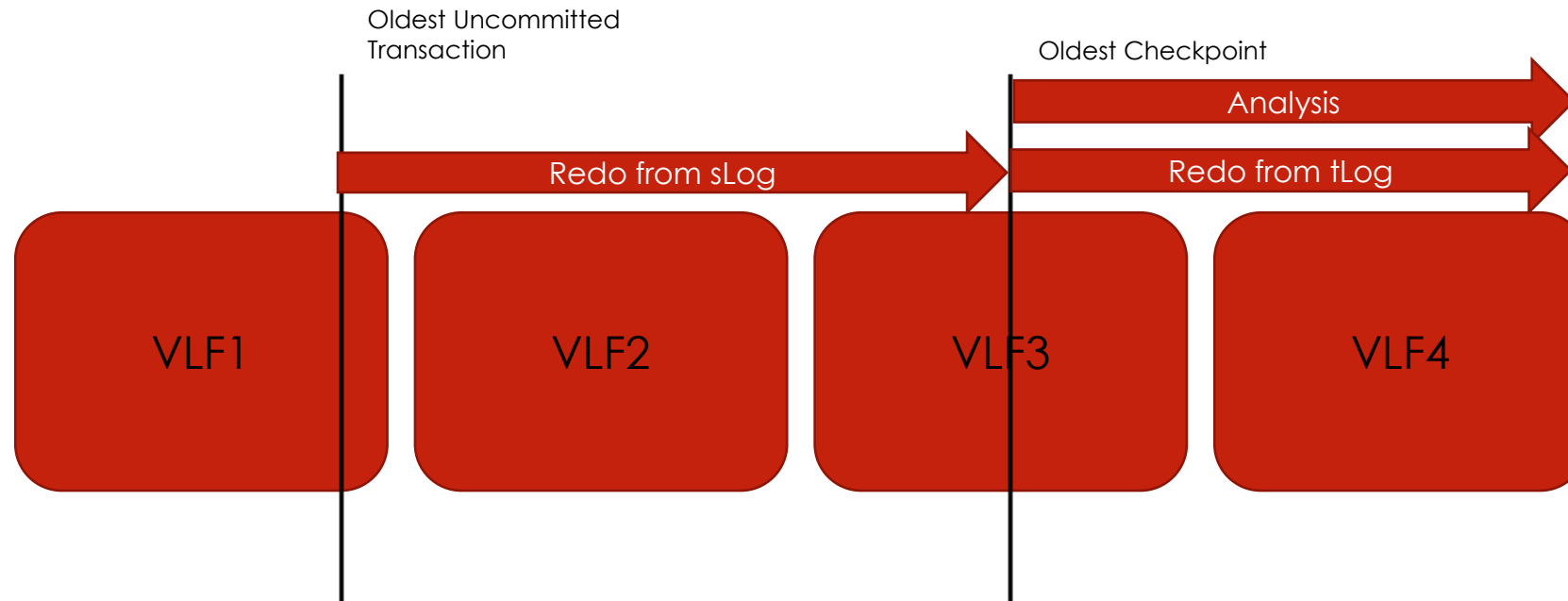




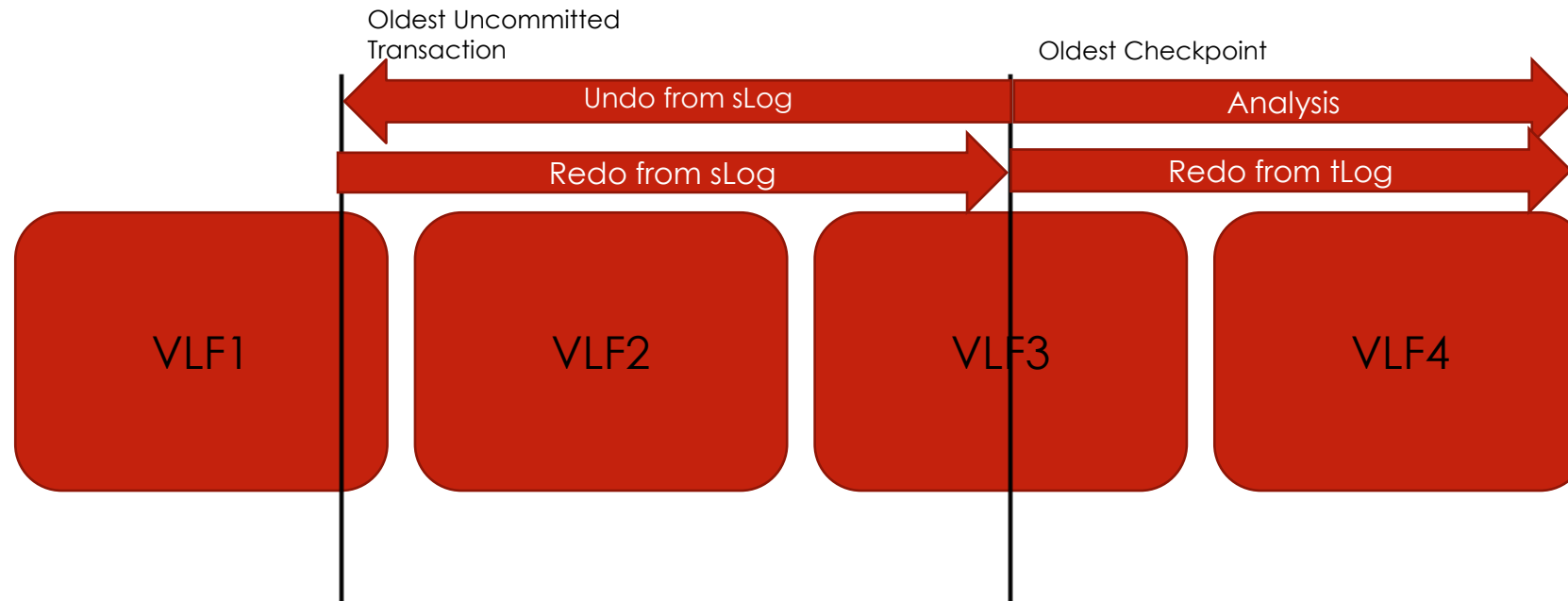
# REDO FROM SLOG



# REDO FROM TLOG



# REDO FROM TLOG





# BENEFITS OF ADR

Faster crash recovery

Faster AG failover

Faster rollback

Fast log truncation means smaller logs



# BENEFITS OF ADR

Faster crash recovery

Faster AG failover

Faster rollback

Fast log truncation means smaller logs

AND!



# BENEFITS OF ADR

Faster crash recovery

Faster AG failover

Faster rollback

Fast log truncation means smaller logs

AND!

It is available in Standard Edition!!!



Source: <http://www.quickmeme.com/meme/3q7ogg>



# ACCELERATED DATABASE RECOVERY DEMO





# IN-MEMORY TEMPDB METADATA TABLES



# WHAT IS TEMPDB?

tempdb is one of the system databases in SQL Server

Used to create temporary objects

Used for sort space

Used by everybody

Can cause contention



# ONE MAN'S OPINION

“TempDB – or as I call it.... SQL Server’s public toilet. You have no idea what other filthy, disgusting things people are doing in TempDB.” –Brent Ozar

Source: <https://ozar.me/2013/02/why-not-everybody-loves-my-sessions/>



# THE OLD CONTENTION PROBLEM

Each data file contains pages to manage page allocation in the database  
Because everyone uses tempdb, these pages can act as a bottleneck



# THE OLD CONTENTION SOLUTION

Add data files to tempdb

This increases the number of metadata pages

SQL Server can parallelize activity



# THE NEW CONTENTION PROBLEM

Each database contains system tables that store metadata about objects

Normally, this isn't a problem because objects are usually static

Because temp tables are constantly created and deleted, these tables can be bottlenecks

The problem is PAGELATCH waits



# THE NEW CONTENTION SOLUTION

Create the tempdb system tables in memory

Eliminates latch contention



# IN-MEMORY TEMPDB METADATA DEMO





# HOW LATCHING WORKS

# PAGE IS READ FROM DISK

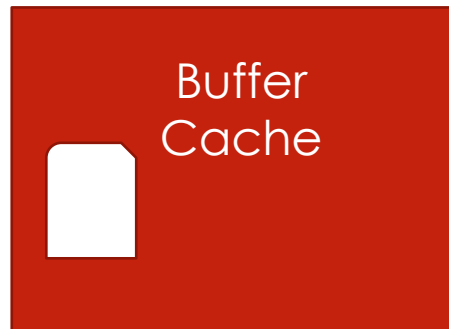
SQL Server  
executes an  
UPDATE to a  
page not in the  
buffer cache

Buffer  
Cache



# DATA IS READ

Page read into  
the buffer from  
disk



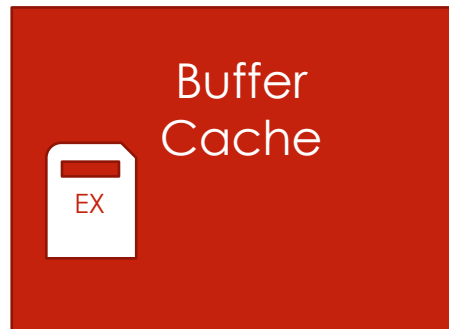
# PAGELATCH\_EX IS TAKEN

SQL Server takes an exclusive latch on the page to prevent collisions in-memory  
While the exclusive latch is held no other processes can access the page  
In the demo, all activity is updating the same row, so a single page is a bottleneck



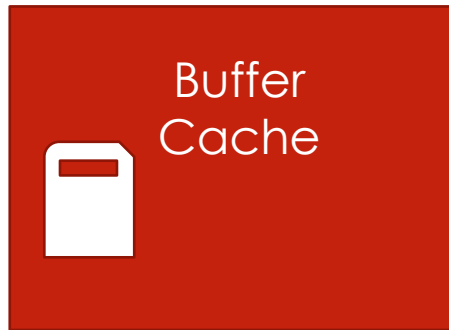
# DATA IS WRITTEN TO THE PAGE

SQL Server  
updates data on  
the page



# LATCH IS RELEASED

Latch is released  
and update  
completes





# TEMPDB LATCHING CONTENTION

tempdb metadata pages are accessed for each creation or deletion of a temp table

Each access requires a latch

Latch contention can occur

Negative impact to performance

# IN-MEMORY ROW STRUCTURE

Row structure

Row header

Row data



# ROW HEADER

Row structure

Row header

Inserted Row Data



Start TS

End TS

StatementID

idxLinkCount

# of Indexes

# IN-MEMORY INSERT

Row structure

Row header

Inserted Row Data



50

$\infty$

1

idxLinkCount

# of Indexes

# IN-MEMORY UPDATE

Row structure

Row header

Updated Row Data



50

90

1

idxLinkCount

# of Indexes

90

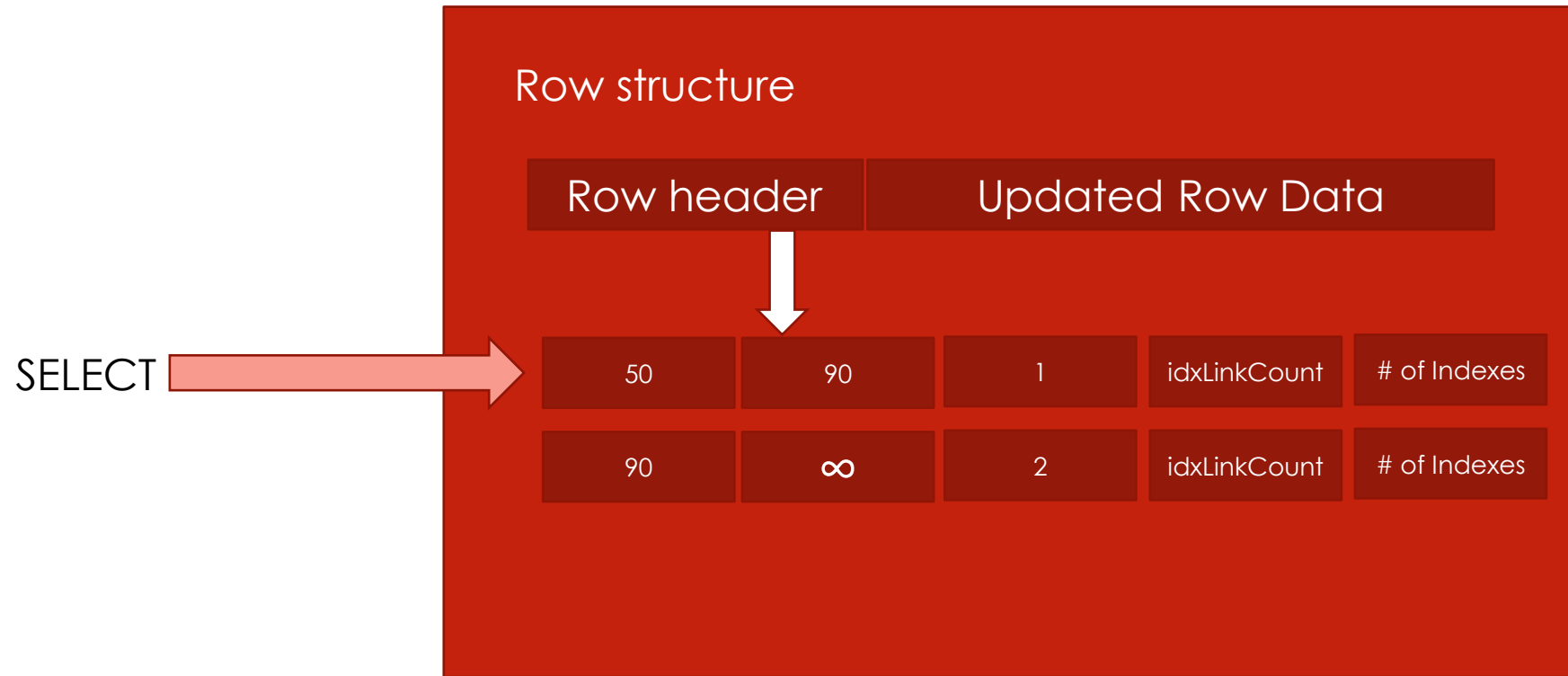
$\infty$

2

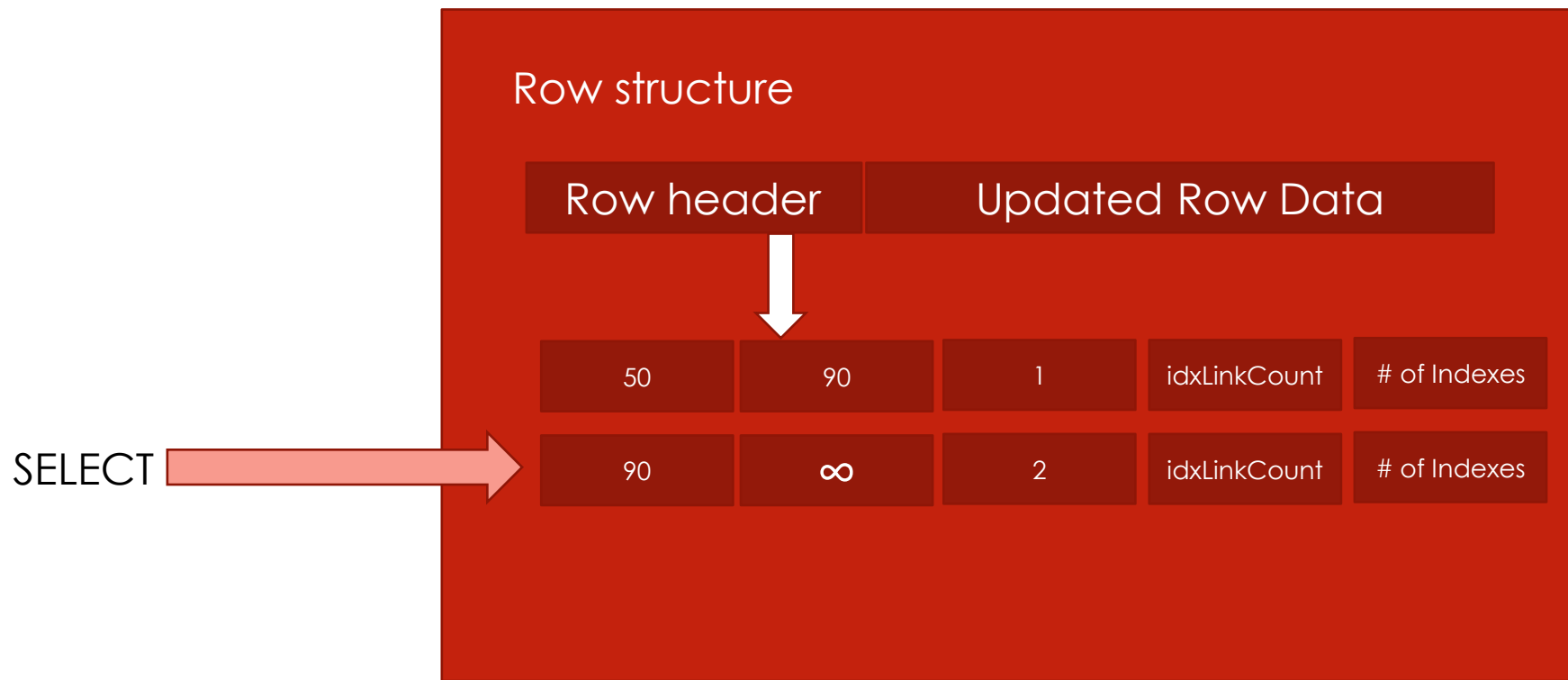
idxLinkCount

# of Indexes

# SELECT WITH TIMESTAMP 60



# SELECT WITH TIMESTAMP 120





# PERSISTENT MEMORY



# BEFORE PERSISTENT MEMORY

Historically, RAM has been transient

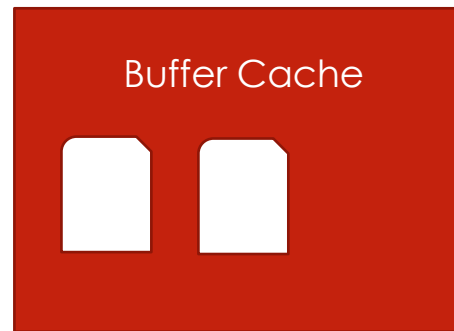
On shutdown, data in RAM is lost

As a result, the transaction log buffer is flushed to disk on COMMIT

SQL Server must wait for confirmation that the flush has completed

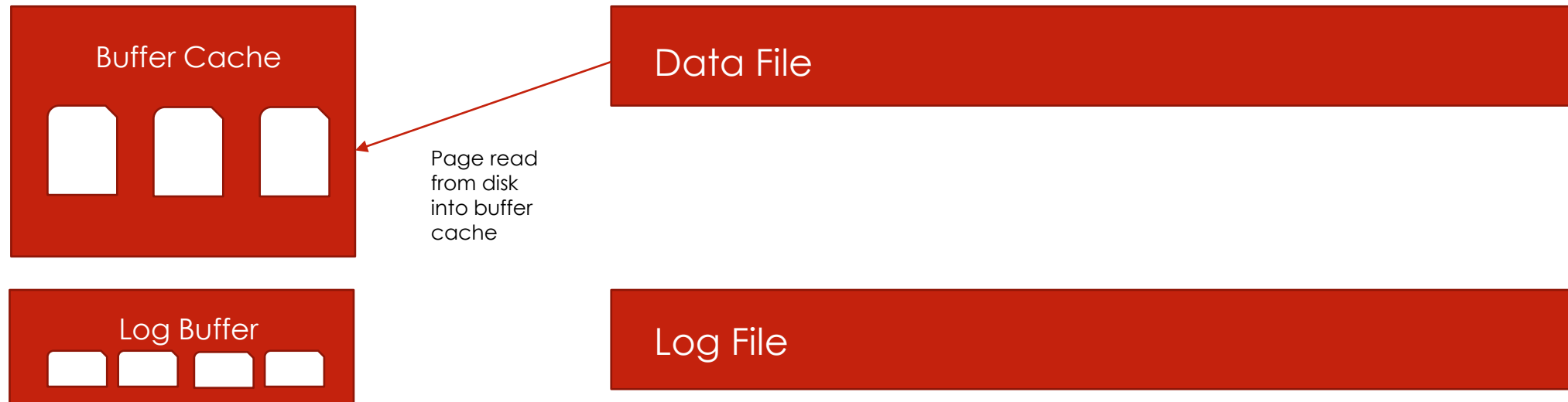
This allows redo and undo to take place in the event of a crash

# LOG BUFFER AND TRANSACTIONS

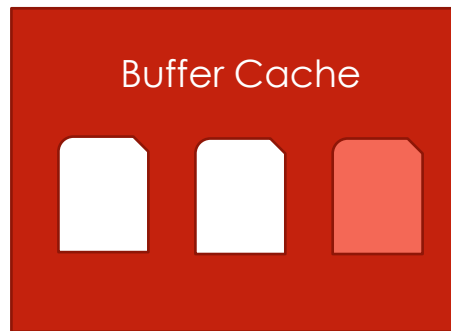




# READ PAGE INTO BUFFER CACHE



# CHANGE PAGE IN MEMORY



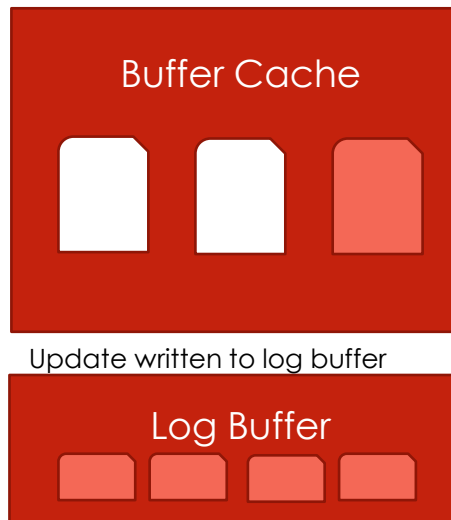
Page updated



Data File

Log File

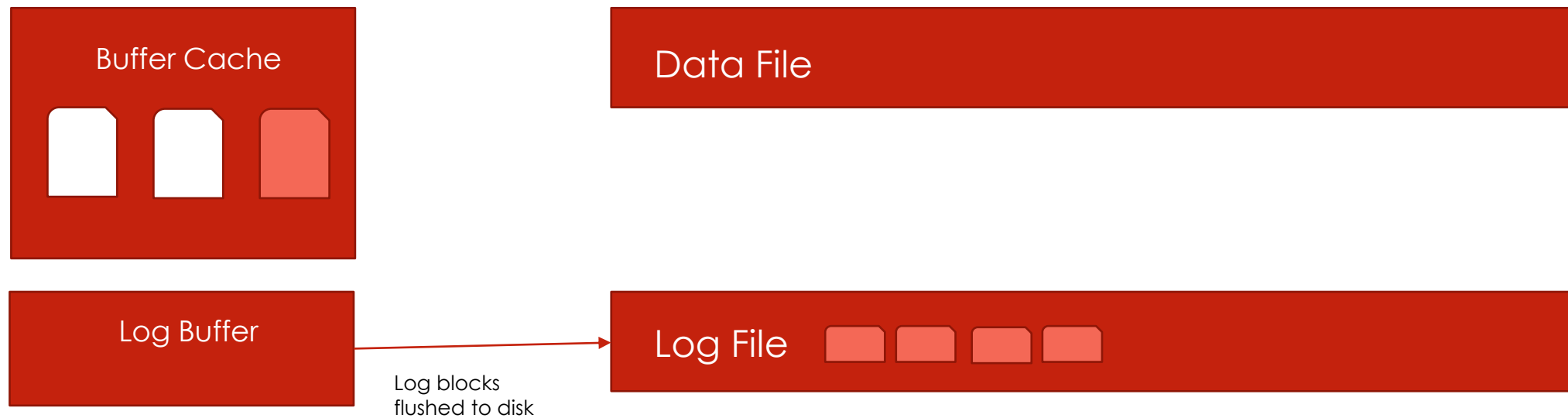
# WRITE LOG RECORDS TO LOG CACHE



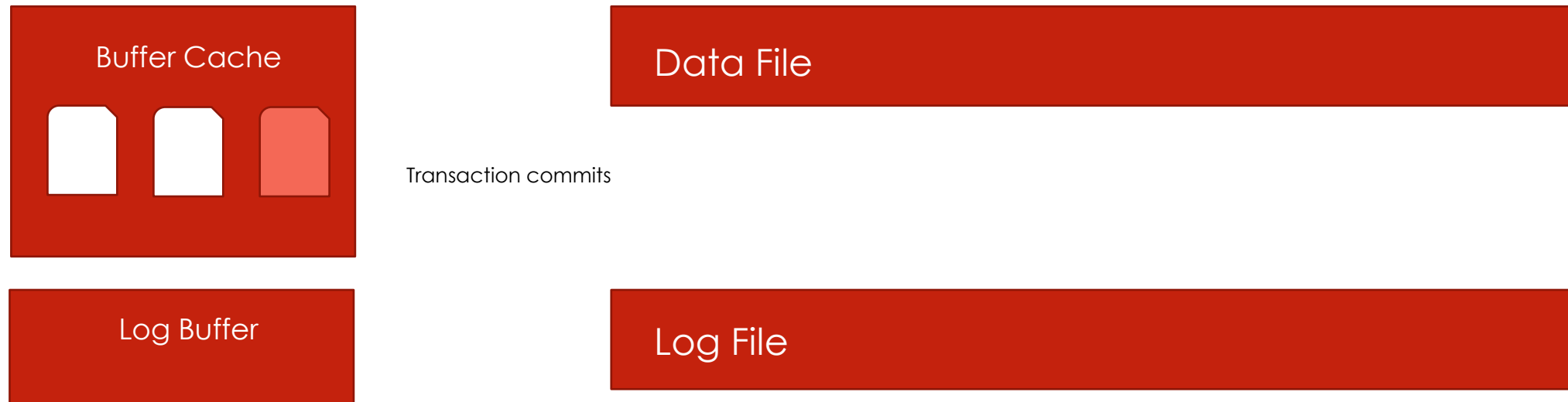
Data File

Log File

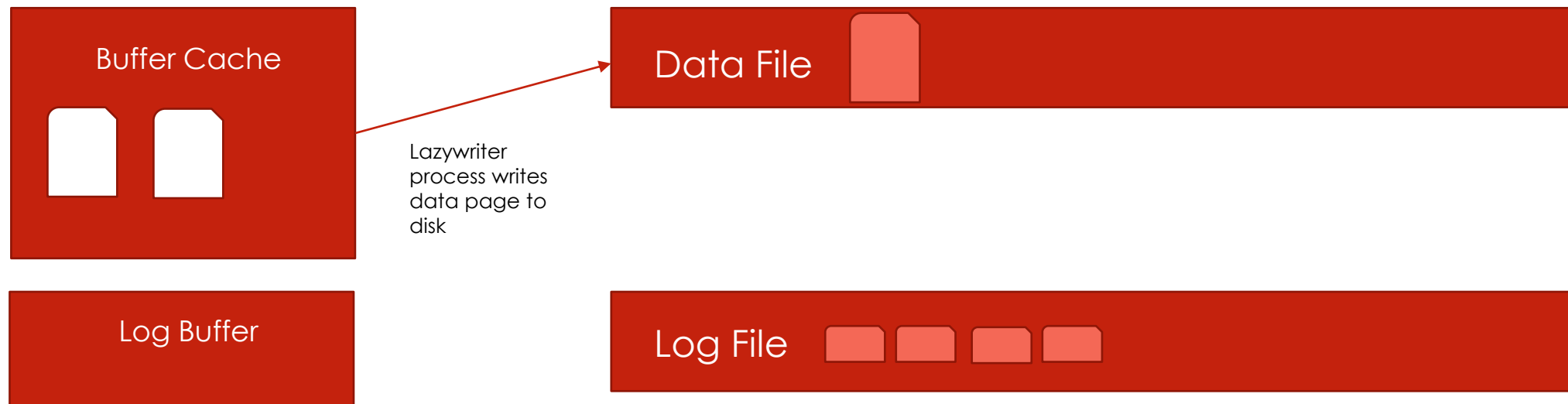
# FLUSH LOG CACHE TO DISK



# WRITE LOG RECORDS TO LOG CACHE



# DATA PAGES FLUSHED ASYNCHRONOUSLY





# PERSISTENT MEMORY

New development in hardware

RAM with a battery

Data stored in RAM can survive a restart

This provides several opportunities for performance enhancements



# PERSISTENT LOG BUFFER CACHE

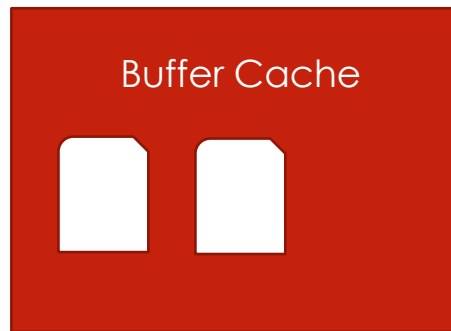
Log flush is no longer necessary on COMMIT

Log flush can happen in the background

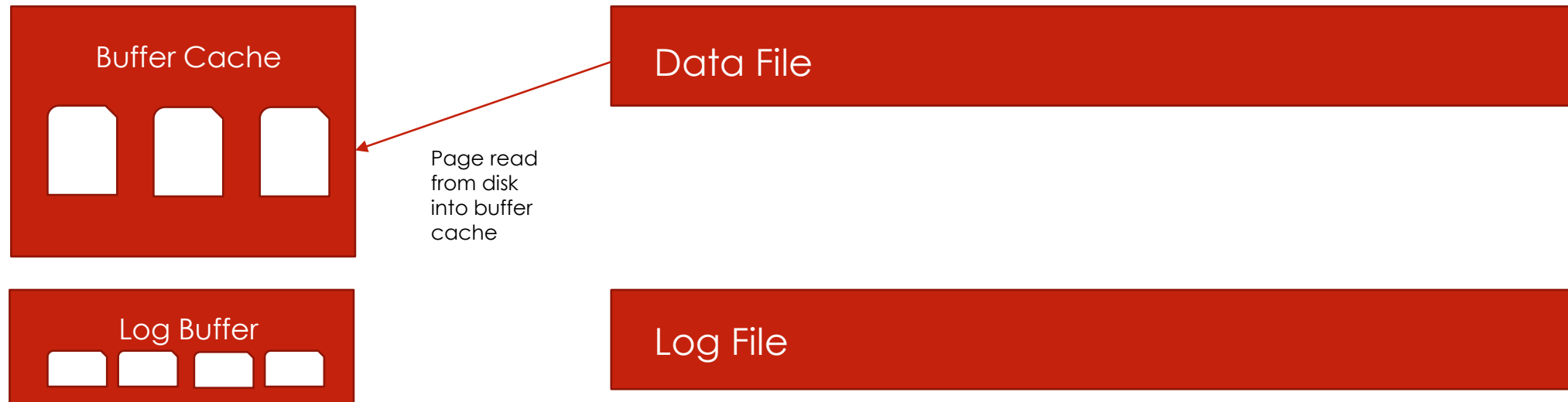
Persistent buffer processed with the log file on restart



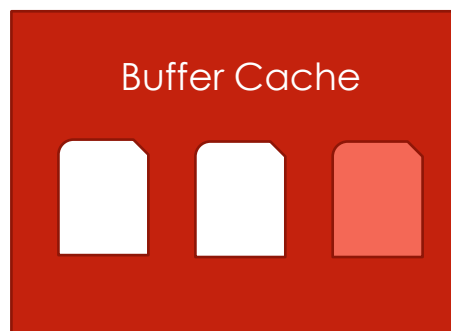
# PERSISTENT LOG BUFFER CACHE



# READ PAGE INTO BUFFER CACHE



# CHANGE PAGE IN MEMORY



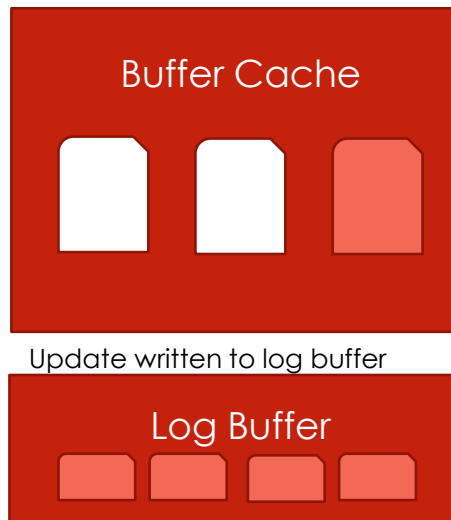
Page updated



Data File

Log File

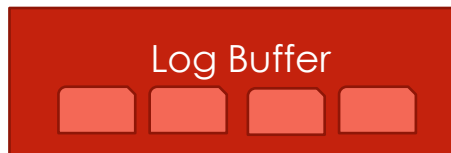
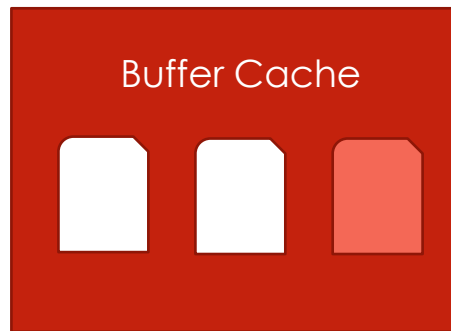
# WRITE LOG RECORDS TO LOG CACHE



Data File

Log File

# TRANSACTION COMMITS

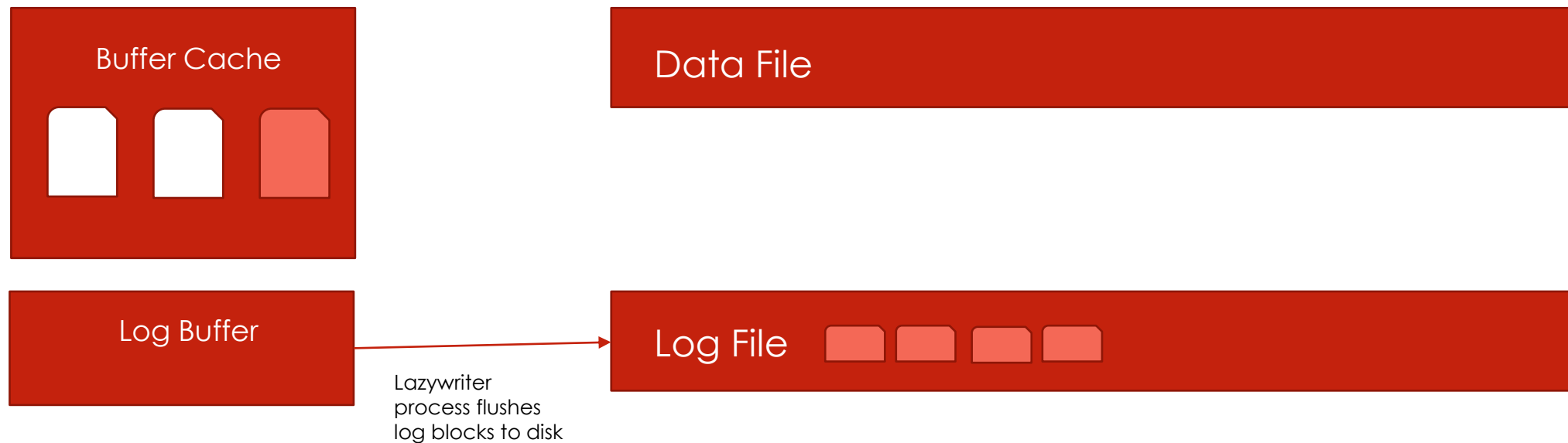


Transaction commits

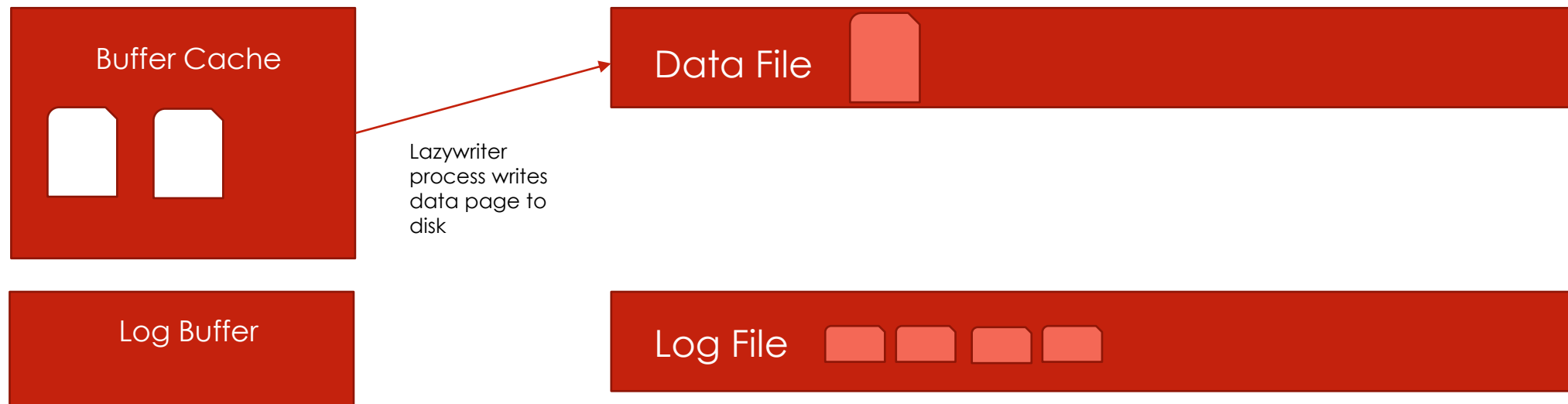
Data File

Log File

# LOG CACHE FLUSHED BY BACKGROUND PROCESS



# DATA PAGES FLUSHED ASYNCHRONOUSLY





# ADVANTAGES OF PERSISTENT LOG BUFFER CACHE

Log buffers not flushed to disk on commit

Speeds transactional processing

Reduces LOGWRITE waits





# HYBRID BUFFER POOL

Enhancement of Buffer Pool Extension (BPE)

BPE extended the buffer pool onto fast SSD disk

Hybrid Buffer Pool extends the buffer pool onto PMEM

BPE without the IO overhead



# ENLIGHTENED IO

Linux-only feature

Data and log files can be placed on PMEM

Allows the file system and storage stack to be bypassed

Note – PMEM is not as fast as traditional RAM

If your database fits in memory, this is not the solution for you




# NO PERSISTENT MEMORY DEMO

# WHY?

[Home](#) > [Data Storage Products](#) > [Hard Drives](#) > [Solid State Drives \(SSDs\)](#)



 Share

 Save as Favorites



## Intel Optane DC Persistent – DDR-T – module – 512 GB – DIMM 288-pin

Mfg.Part: UCS-MP-512GS-A0= | CDW Part: 6020774 | UNSPSC: 43201402

**Availability:** • 4-6 Weeks

Orders placed today will ship within 4-6 weeks by a CDW partner.

 Advertised Price

**Claim up to a 5% Discount**  
[Create an account](#) to get My CDW Advantage today.

1

Add to Cart

[Lease Option](#) (\$562.19/month)

### Product Details

- DDR-T
- module
- 512 GB
- DIMM 288-pin
- 2666 MHz / PC4-21300
- 1.2 V
- for UCS C220 M5

[View Full Product Details](#)

Source: [Intel Optane DC Persistent - DDR-T - module - 512 GB - DIMM 288-pin - UCS-MP-512GS-A0= - Hard Drives - CDW.com](#)

# THAT'S WHY

[Home](#) > [Data Storage Products](#) > [Hard Drives](#) > [Solid State Drives \(SSDs\)](#)



Share



Save as Favorites



## Intel Optane DC Persistent – DDR-T – module – 512 GB – DIMM 288-pin

Mfg.Part: UCS-MP-512GS-A0= | CDW Part: 6020774 | UNSPSC: 43201402

Availability: • **4-6 Weeks**

Orders placed today will ship within 4-6 weeks by a CDW partner.

**\$19,206.99** Advertised Price

**Claim up to a 5% Discount**

[Create an account](#) to get My CDW Advantage today.

1



Add to Cart

[Lease Option](#) (\$562.19/month)

### Product Details

- DDR-T
- module
- 512 GB
- DIMM 288-pin
- 2666 MHz / PC4-21300
- 1.2 V
- for UCS C220 M5

[View Full Product Details](#)



# WHAT WE'VE COVERED

Accelerated Database Recovery

In-memory tempdb metadata

Persistent Memory

- Persistent Log Buffer Cache

- Hybrid Buffer Pool

- Enlightened IO

# RESOURCES

Slides and Scripts– <https://github.com/skreebydba/MinnesotaPresentation>

Tiger Team Materials – <https://microsoft.github.io/sqlworkshops/>

Hybrid Buffer Pool –

<https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/hybrid-buffer-pool?view=sql-server-2017>

# RESOURCES

Accelerated Database Recovery –

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-accelerated-database-recovery>

Constant Time Recovery in Azure SQL Database (White paper that gets deep into the internals of ADR, also known as CTR) –

<https://www.microsoft.com/en-us/research/publication/constant-time-recovery-in-azure-sql-database/>

In-memory tempdb Metadata –

<https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15#memory-optimized-tempdb-metadata>