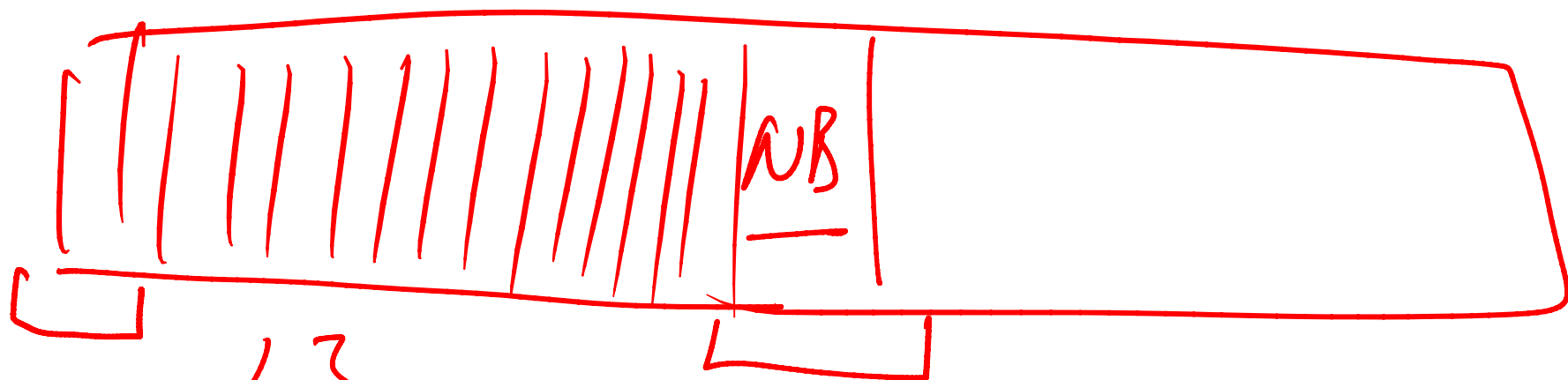




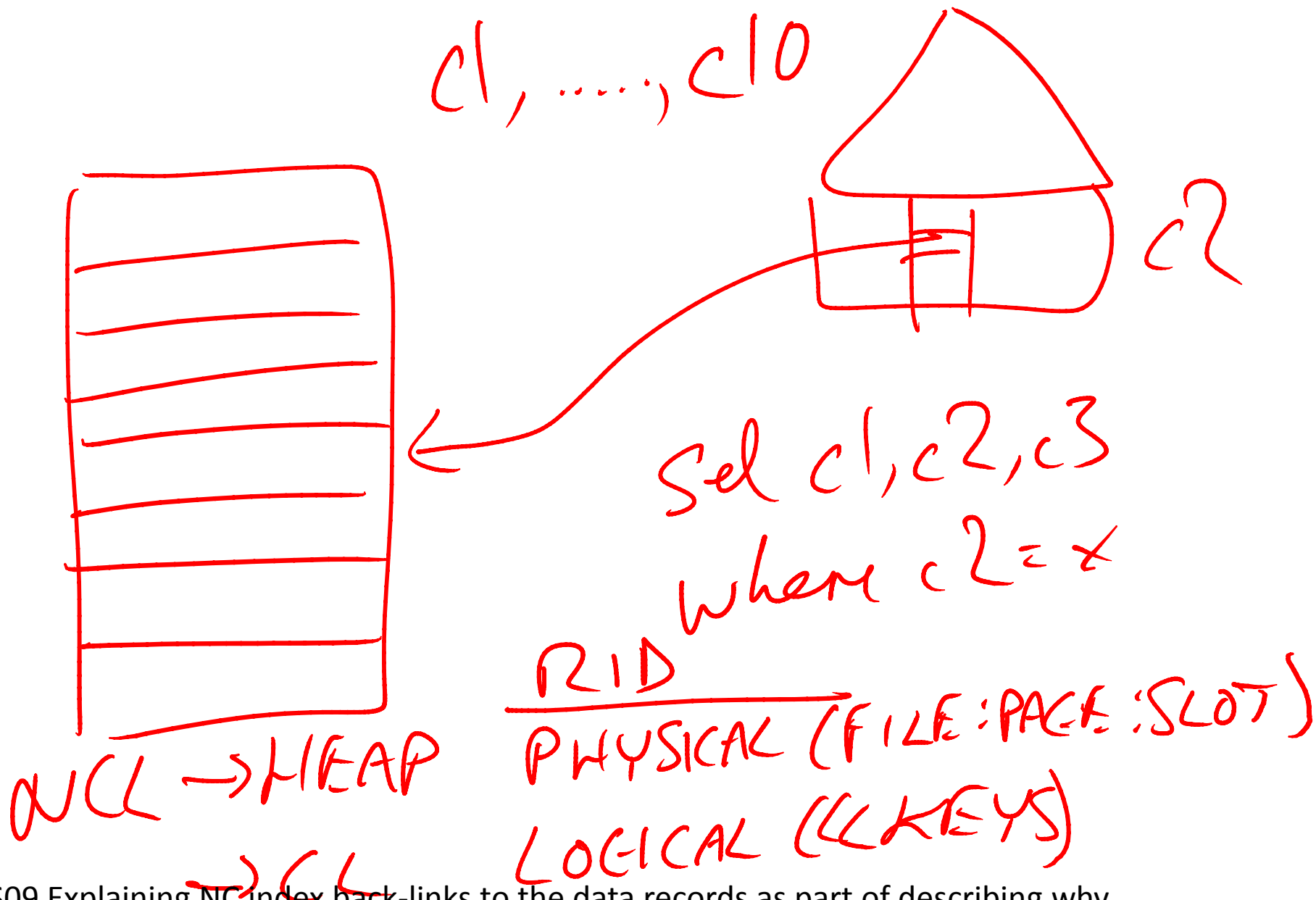
L3

CACHE LINES

64b
128b

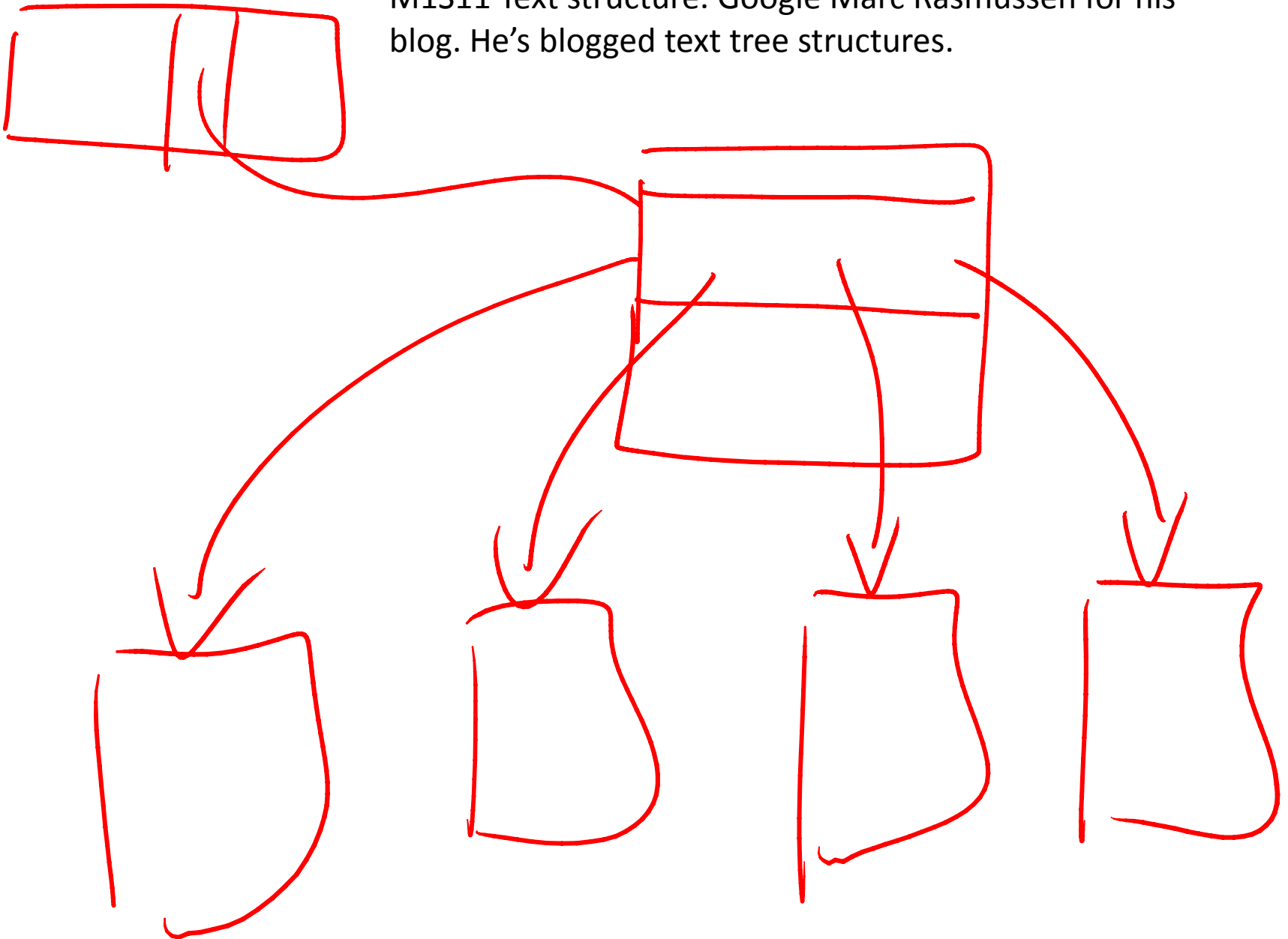
INVALIDATION

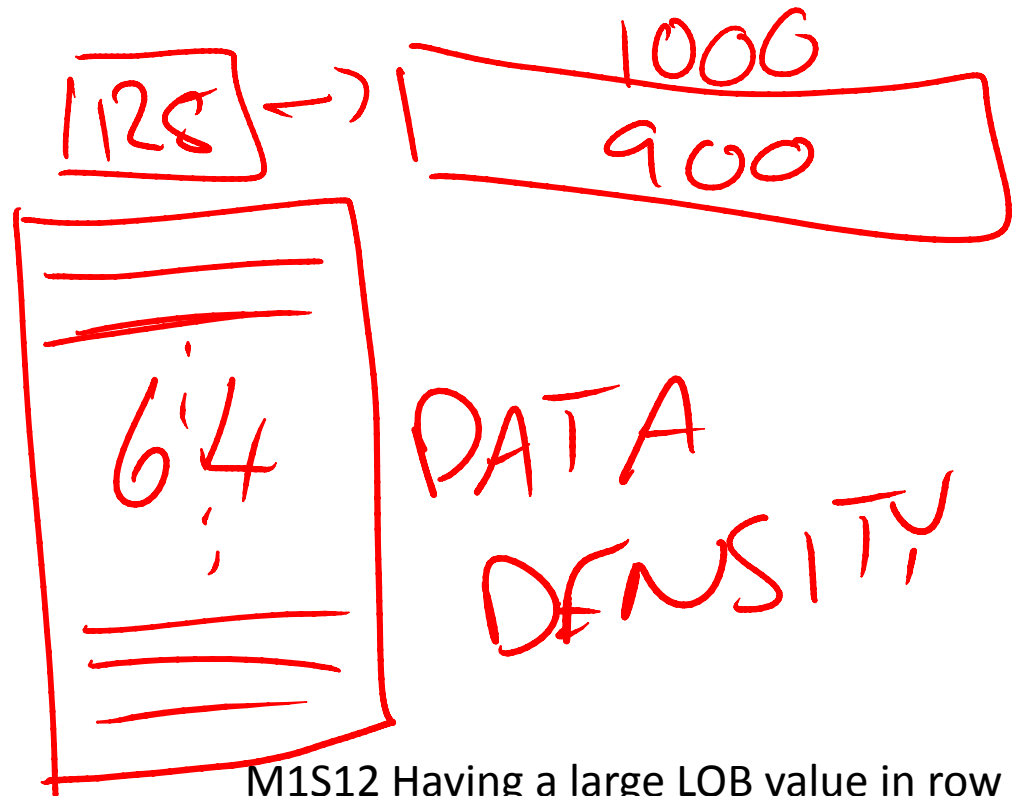
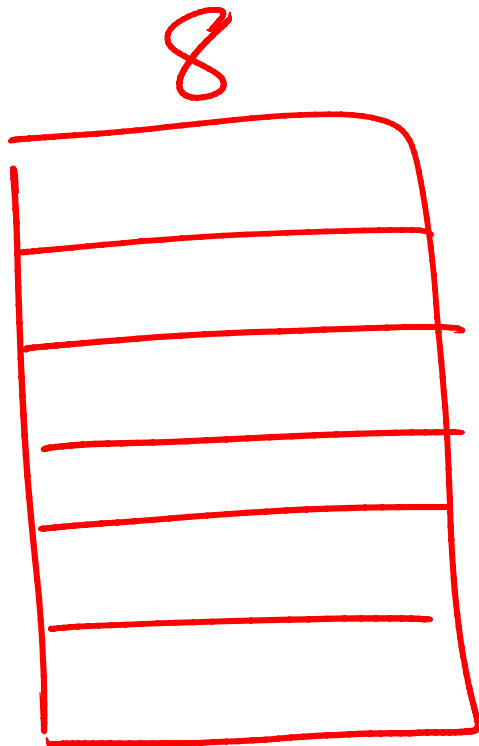
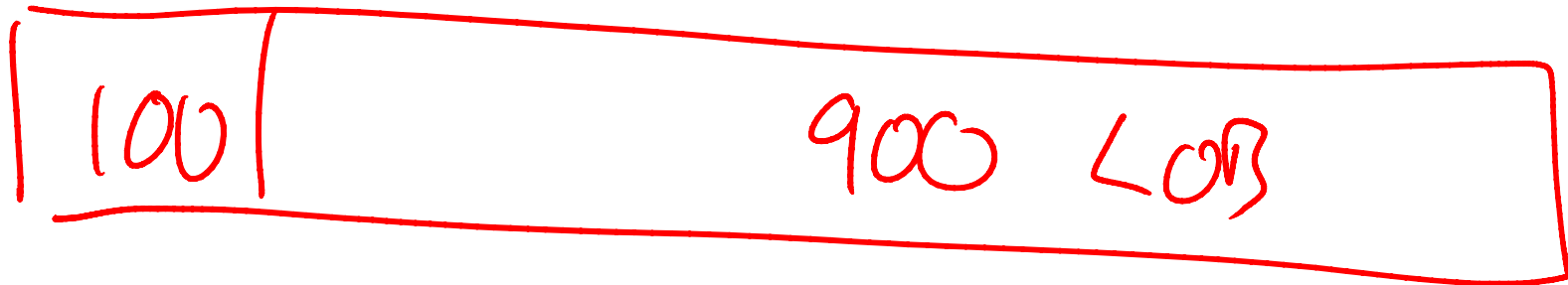
M1S06 Discussing
why the NULL
bitmap is a perf
optimization by
avoiding having to
read record parts
into the CPU's L3
cache



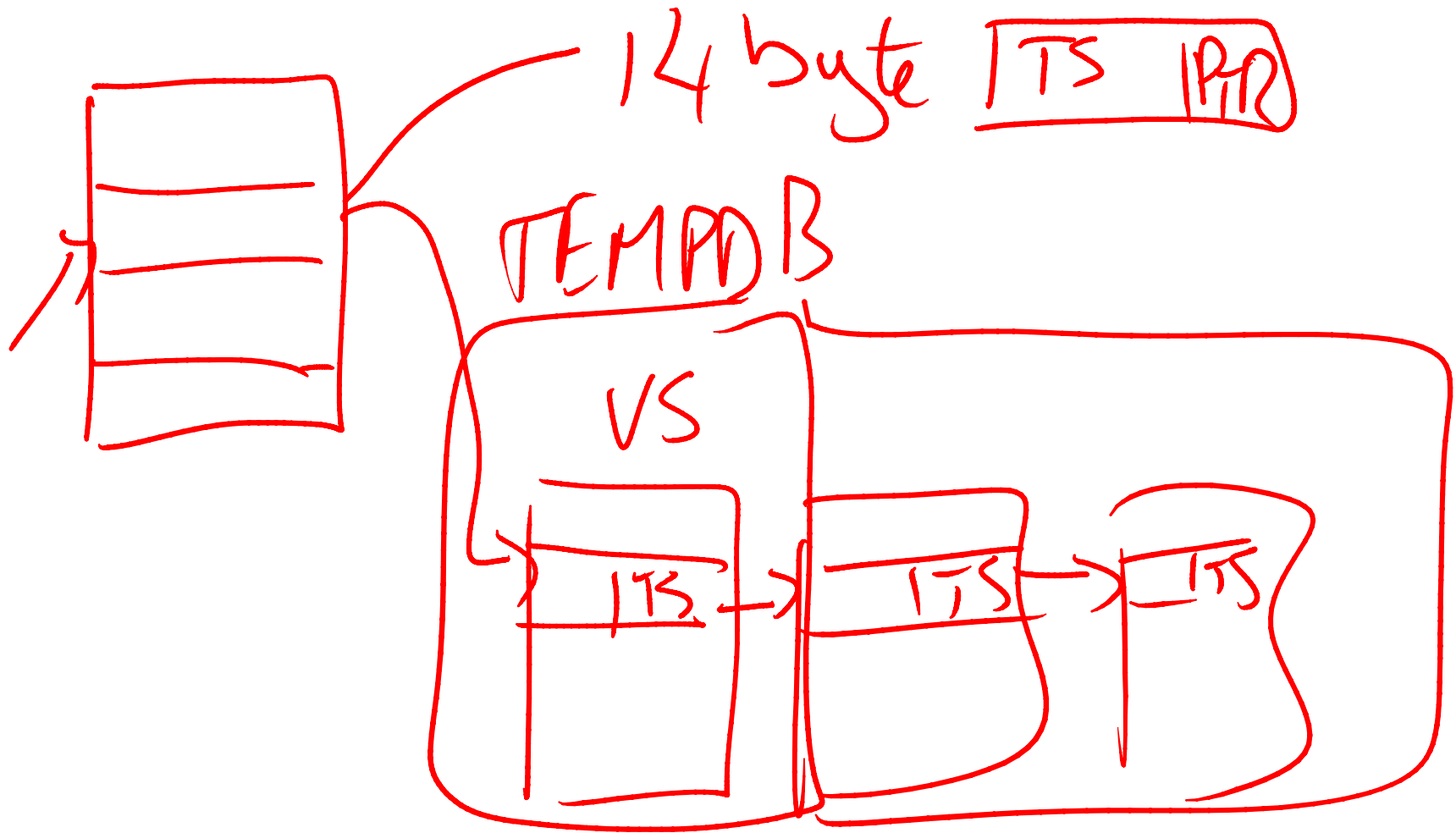
M1S09 Explaining NC index back-links to the data records as part of describing why forwarding/forwarded records exist. Without them, moving a heap row would entail changing all NC index records with a back-link to that row.

M1S11 Text structure. Google Marc Rasmussen for his blog. He's blogged text tree structures.



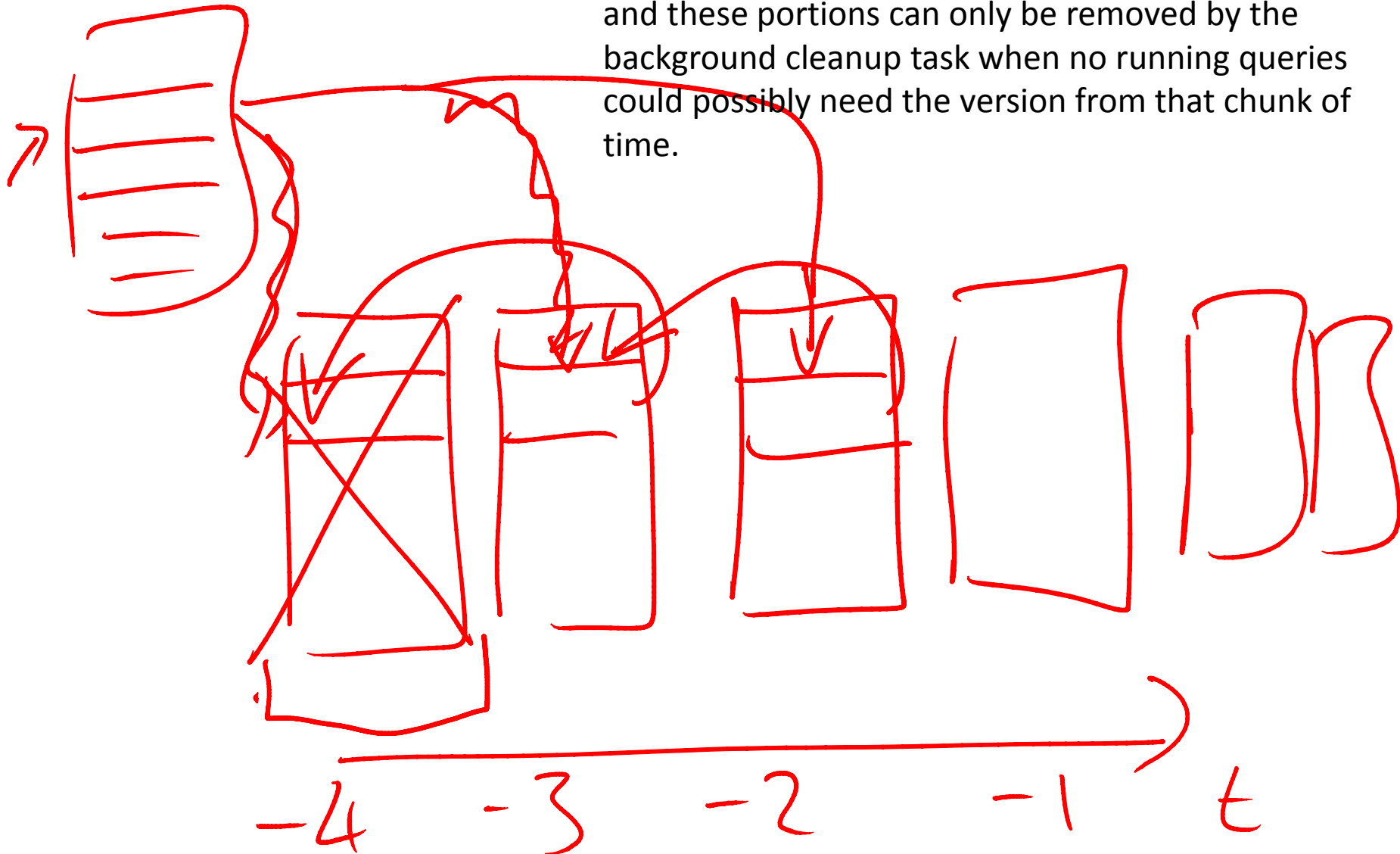


M1S12 Having a large LOB value in row that's not used much makes for large rows and low data density. Choose how to store LOB data wisely.



M1S14 When an update occurs to a record and snapshot isolation is enabled, the pre-change record is copied into the version store in tempdb. The changed record has a 14-byte tag on the end with the timestamp and pointer to previous record version. A chain of previous versions is possible.

M1S14 & M2S56 Explaining how there's a new portion of the version store created every minute and these portions can only be removed by the background cleanup task when no running queries could possibly need the version from that chunk of time.

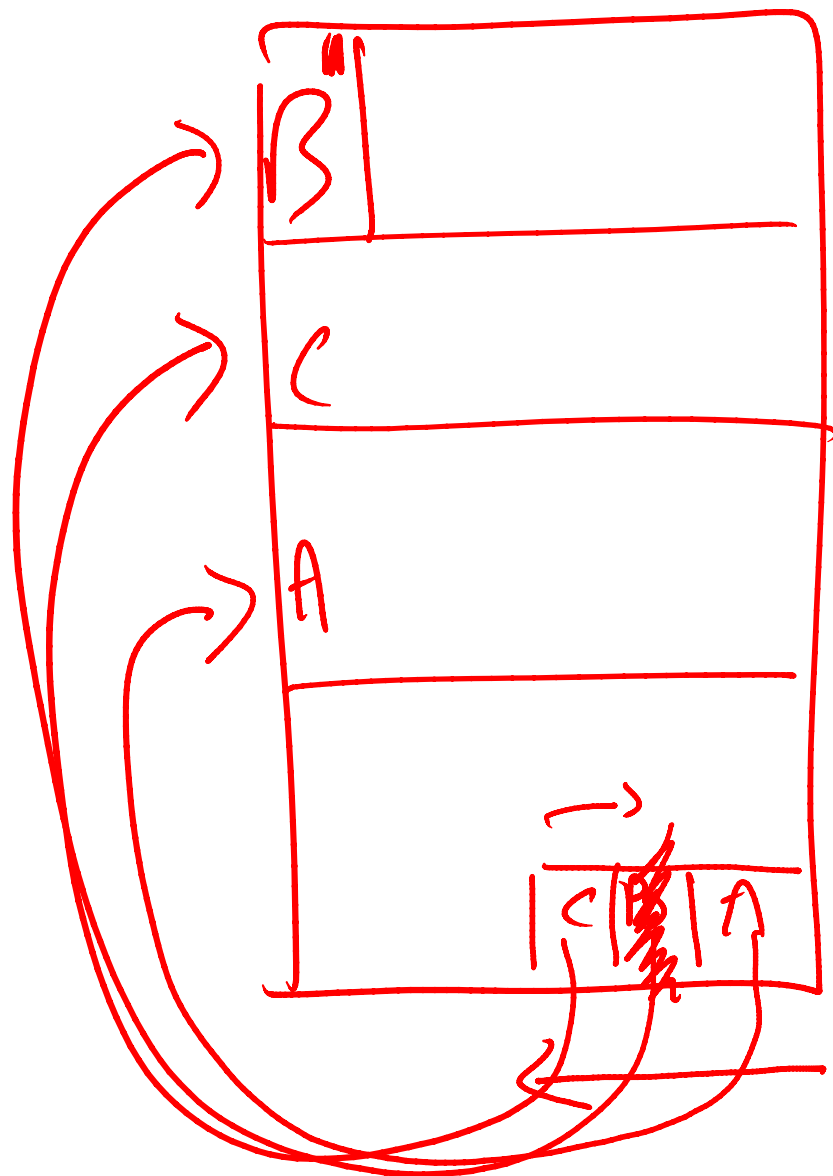


DBCC

TRACEON(662, -1)

3605

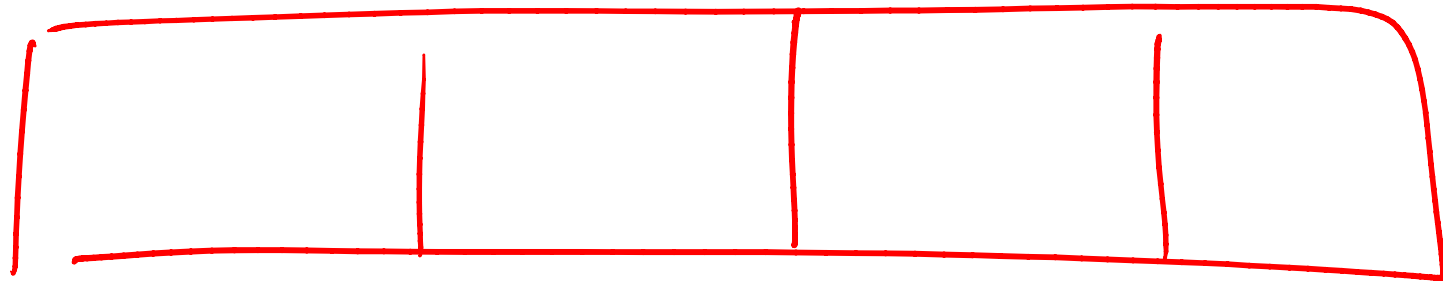
M1S15 How to enable the trace flag globally to watch the ghost cleanup task. You may also need to enable 3605 as well. That one allows debug traceprints to go to the error log.



LOP EXPUNCE -
CHOST

SLOT ARRAY

M1S15 When a record is deleted, it's marked as a ghost record. The ghost cleanup task does not physically delete the records – it removes the slot array entry that points to the record on the page. I.e. the record is unmarked as being a record and the area on the page is now free space – that just happens to have bits and bytes that used to be a valid record.



0 — 7 8 — 15 16 — 23

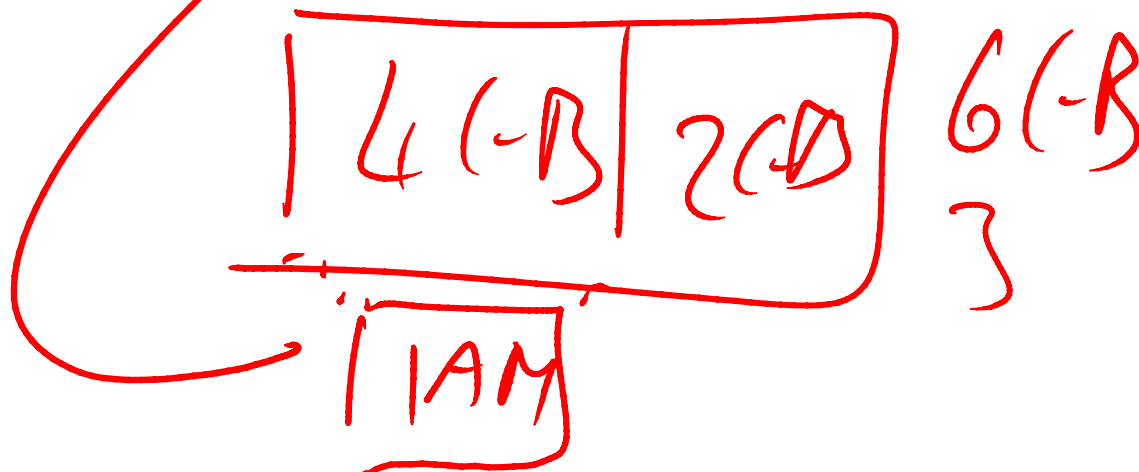


M1S22 Explaining the difference between mixed and dedicated extents. Mixed extent is one in which the 8 pages could be allocated to 8 different objects. Dedicated extent is one where all 8 pages are reserved for exclusive use of a single object.

Later in the deck I refine 'object' to really be 'allocation unit'.



M1S35 Example of IAM chains.
 Allocations from each 4GB
 chunk of a data file require an
 IAM page to track the
 allocation. Multiple IAM pages
 make an IAM chain.

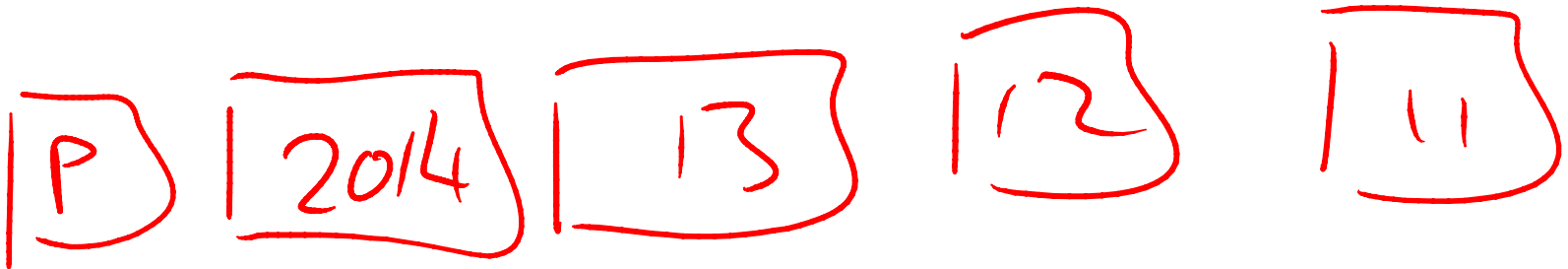


1 TB SALES

2011

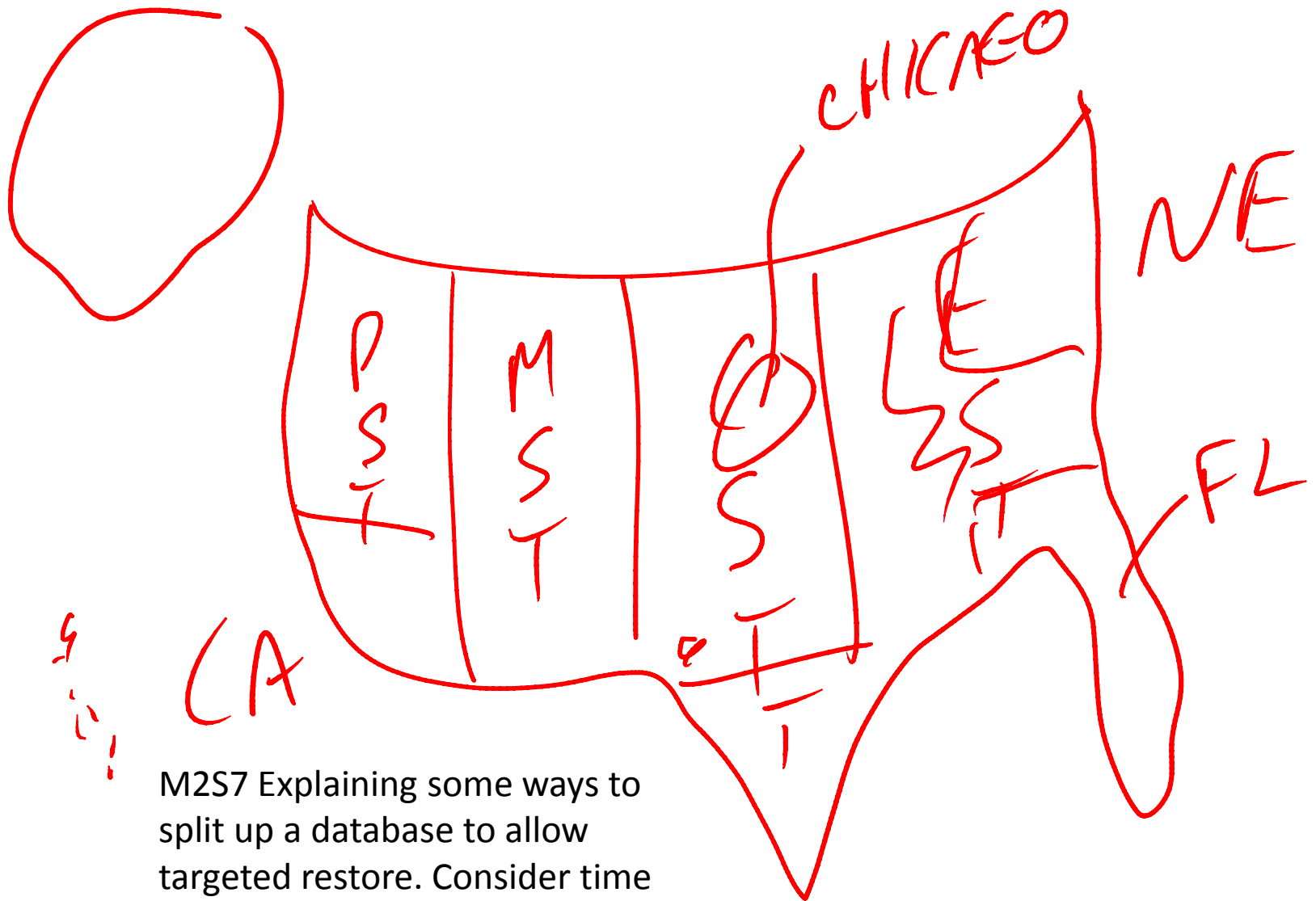
2014

M2S7 Splitting a large database up into multiple filegroups can enable you to do small, targeted restores or even a partial restore from scratch – in the event that the database is damaged or destroyed.

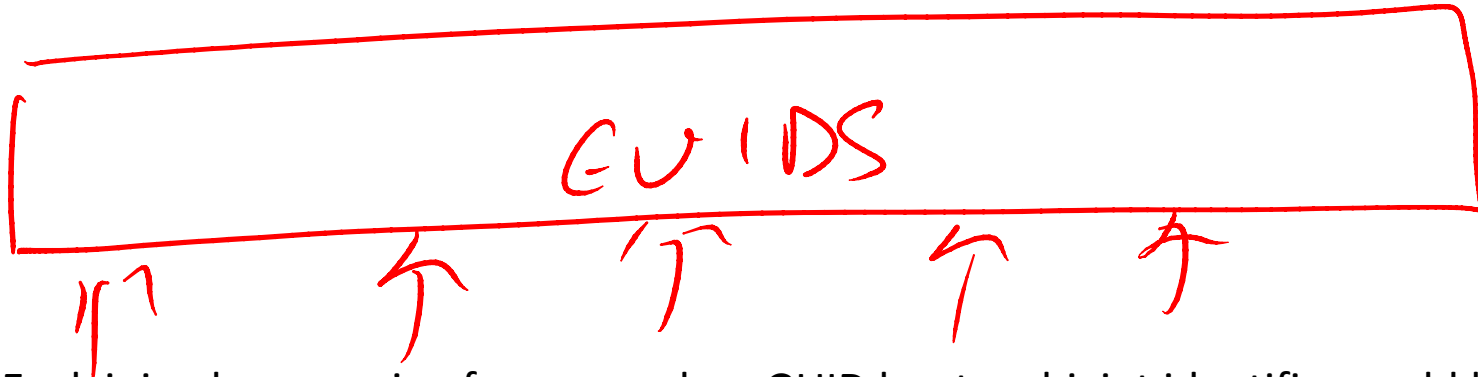




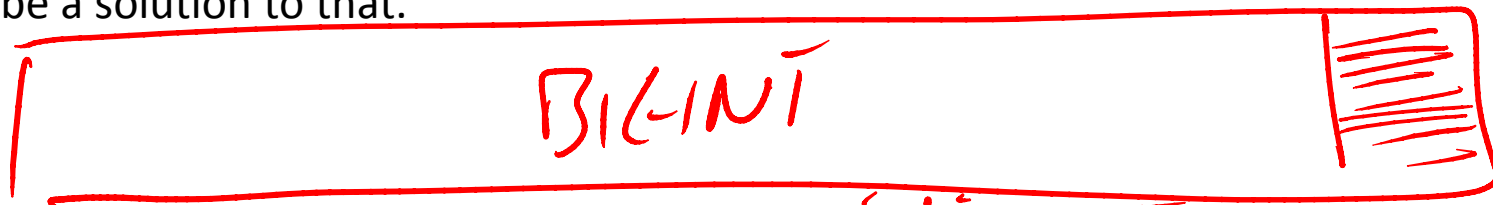
M2S7 Explaining some ways to split up a database to allow targeted restore. On the left, auto parts sales system – makes sense to bring parts and NSW online first.



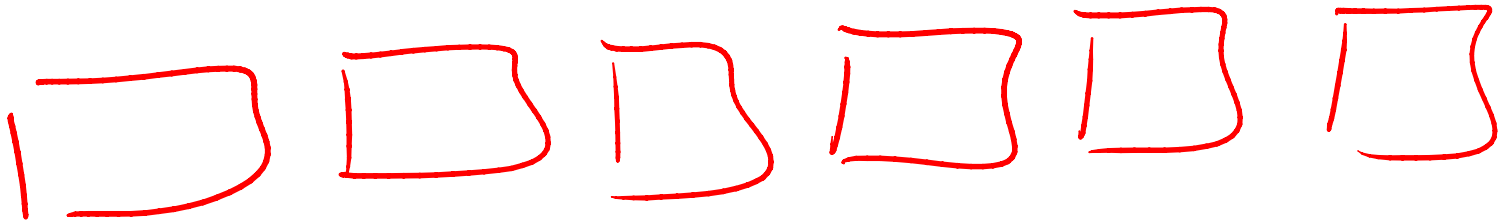
M2S7 Explaining some ways to split up a database to allow targeted restore. Consider time zone, population density, or other demographics.



M2S7 Explaining how moving from a random GUID key to a bigint identifier could lead to an insert hotspot, with small rows and many concurrent inserters. Artificial partitioning key could be a solution to that.

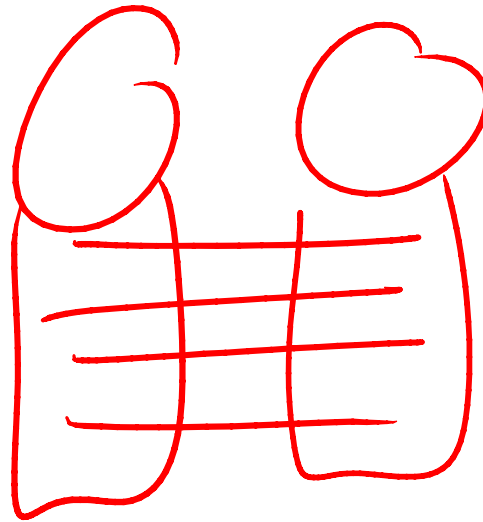


sys.dm_os_waiting_tasks
PAGELOCK_EX/UP



See M2S60/61 for details of the DMV and the PAGELATCH wait types

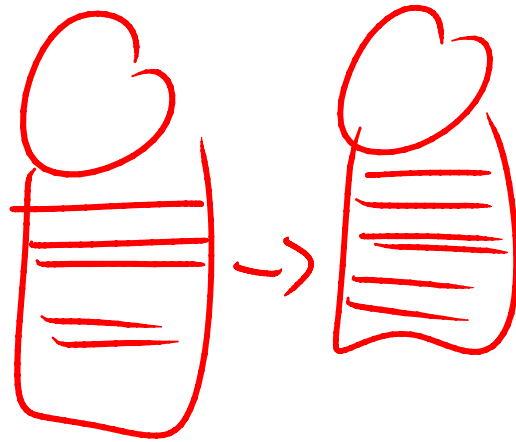
STRIPING



R0

http://en.wikipedia.org/wiki/Standard_RAID_levels

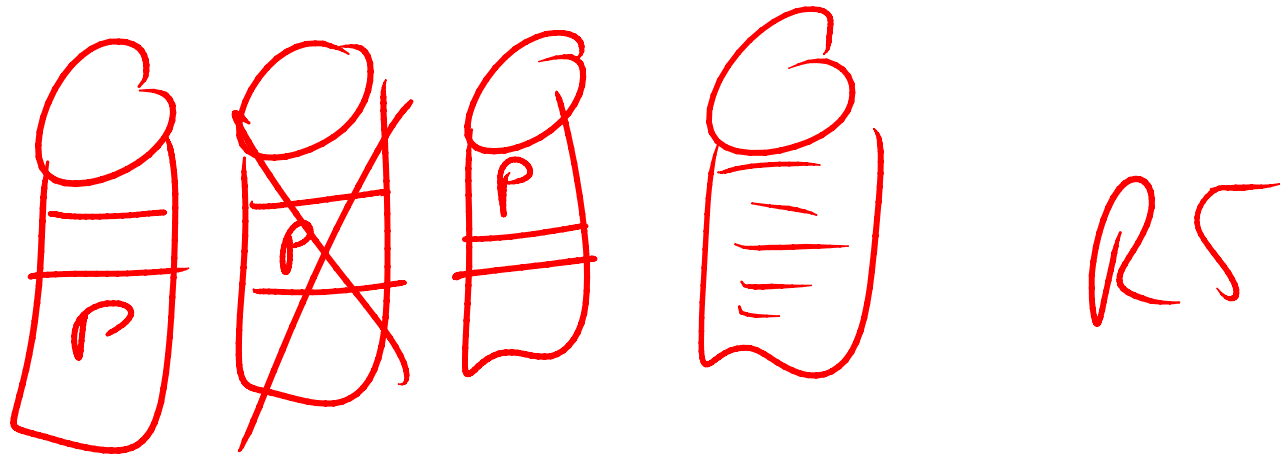
MIRRORING



RA1

http://en.wikipedia.org/wiki/Standard_RAID_levels

NOTATING PARITY

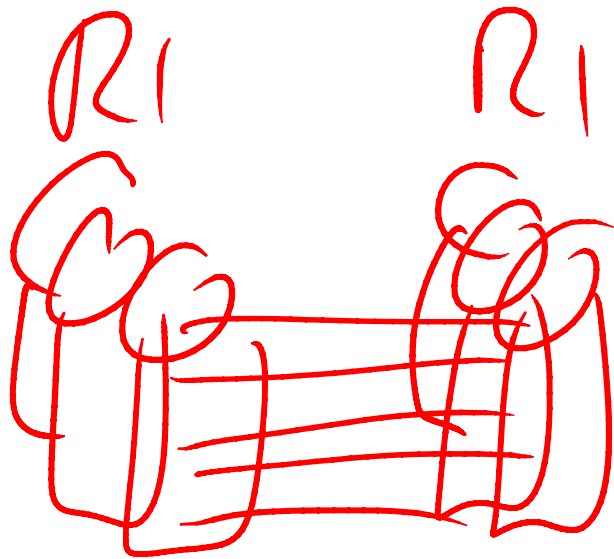


http://en.wikipedia.org/wiki/Standard_RAID_levels

$$R10 = R1 + G$$

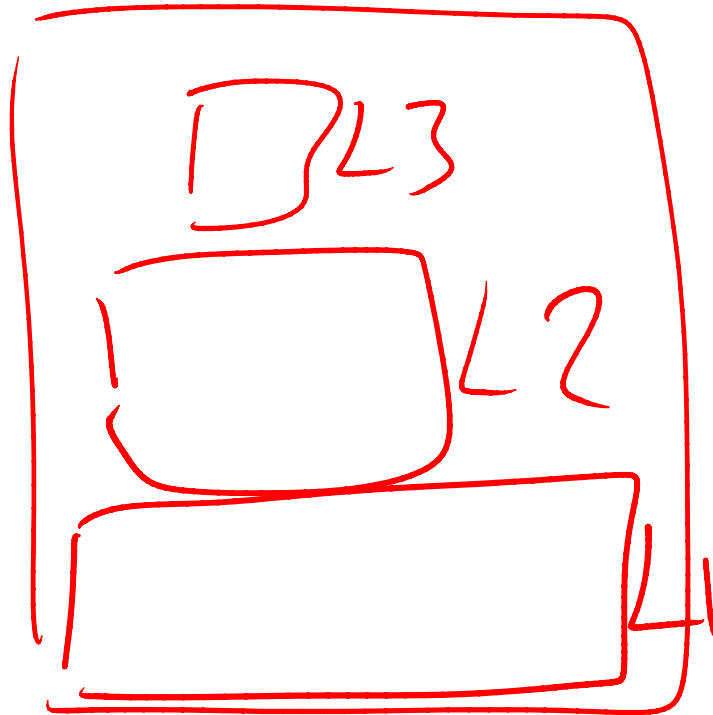
= MIRRORING

+ STRIPING

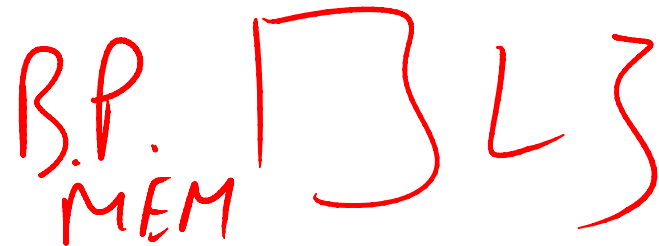


http://en.wikipedia.org/wiki/Standard_RAID_levels

CPU

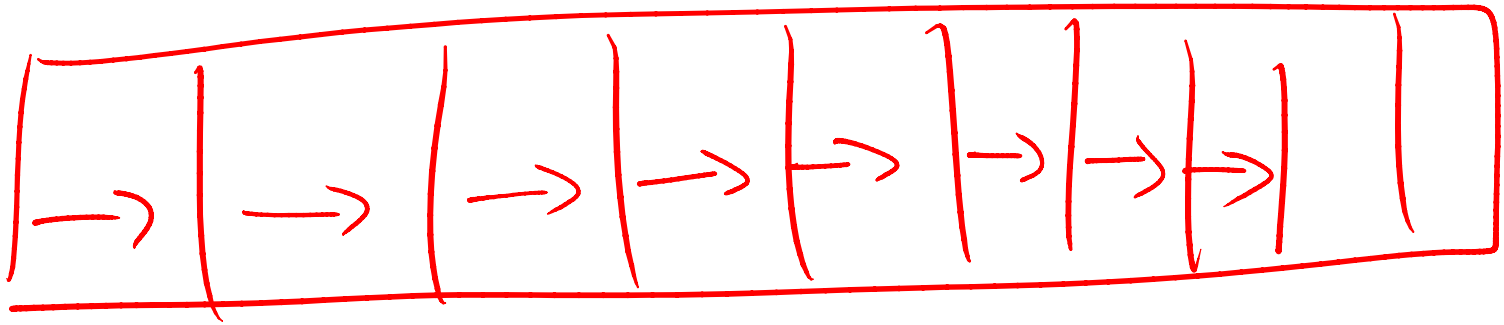


I/O LATENCY



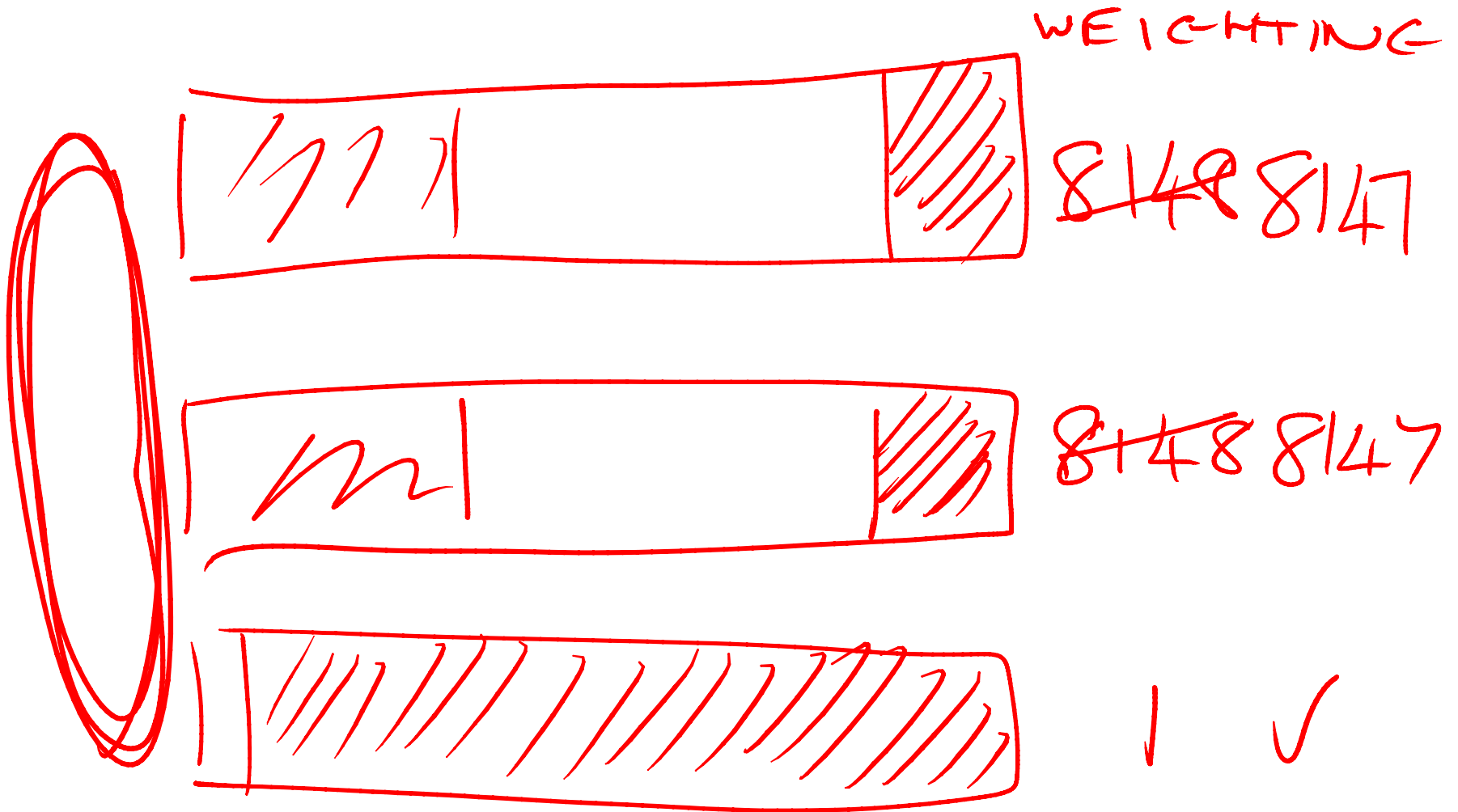
I/O SUBSYSTEM

M2S11 Explaining 2014 Buffer Pool Extension. It's like have L1-L3 caches for your data in memory. BPE sits in between memory and the I/O subsystem.

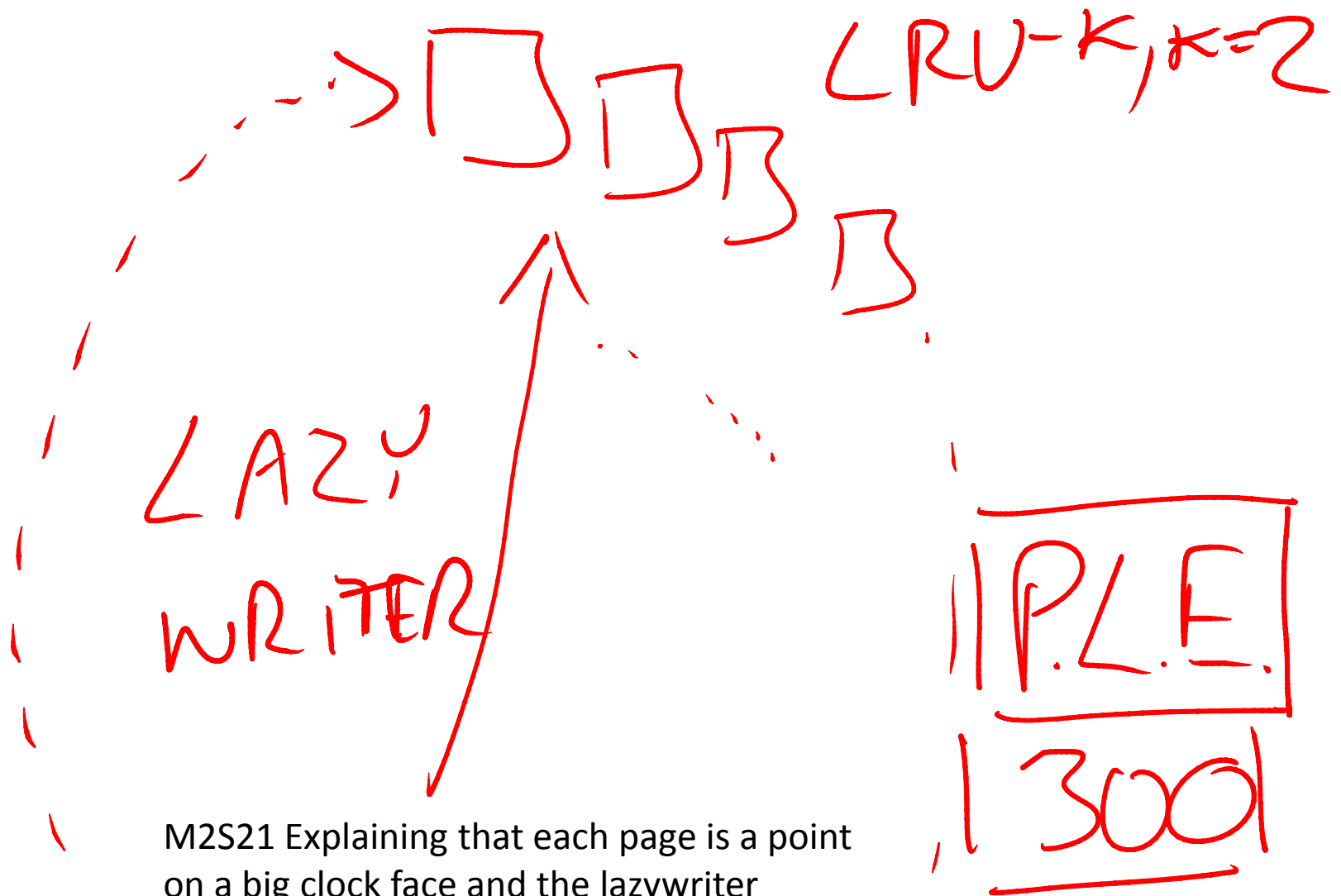


∠:

M2S14 If you put all your log files on one volume, its I/O characteristics become random

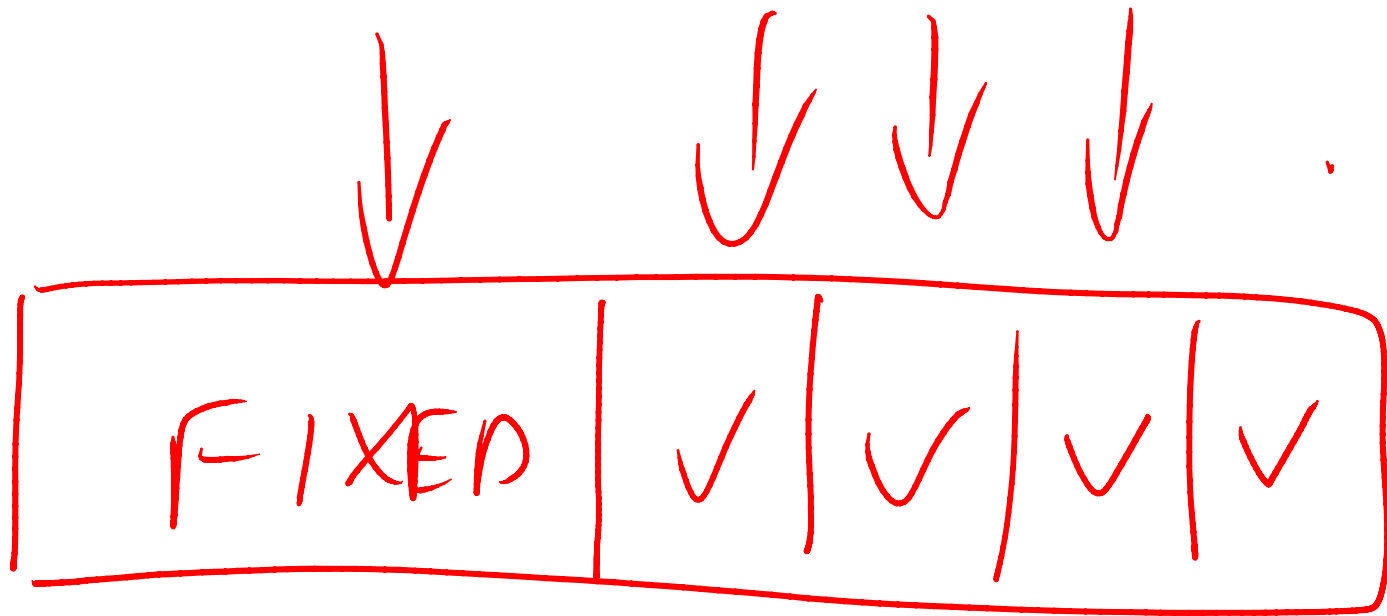


M2S20 Explaining round-robin allocation and proportional fill. Each file has a proportional fill weighting. Allocations happen proportionally more frequently from files with more free space.



M2S21 Explaining that each page is a point on a big clock face and the lazywriter background process sweeps the clock face implement a Least-Recently-Used algorithm that helps it maintain free space by discarding unused pages.

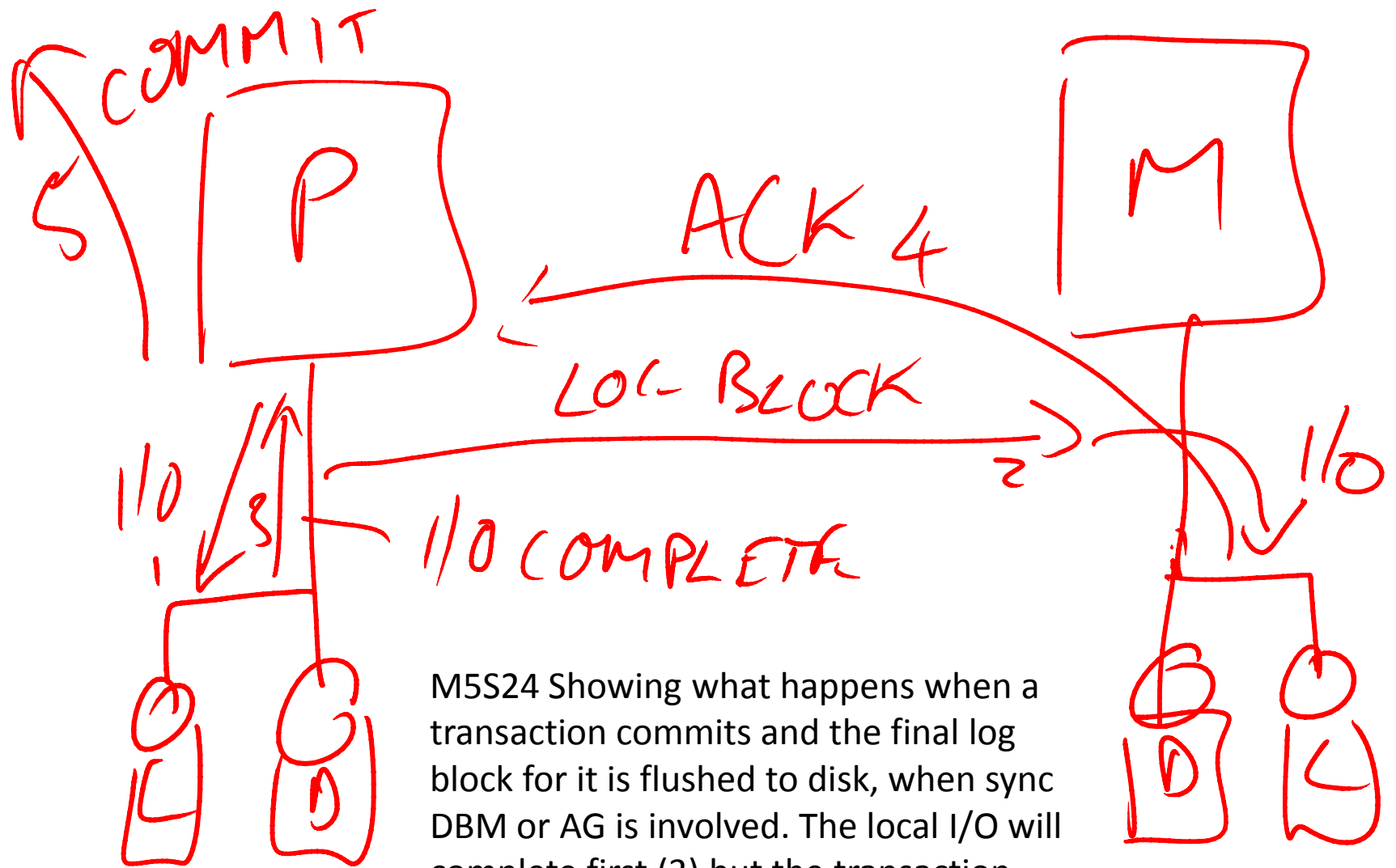
PLE = 300 is a nonsense threshold for tuning. See next slide.



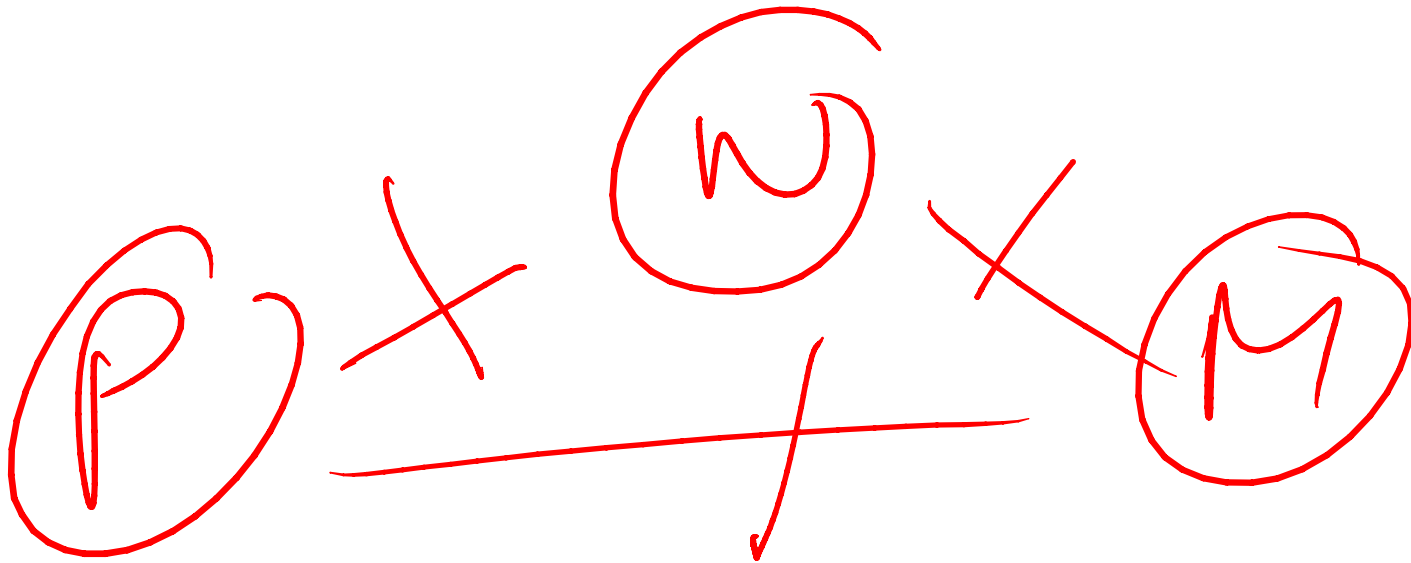
LOP_MODIFY_ROW

LOP_MODIFY_COLUMN

M5S19 Explaining how the Access Methods determines how to log UPDATES efficiently by splitting the logging by portion of the record. Each variable length column is a portion, as is the fixed-width portion (up to 16 bytes in each log record)

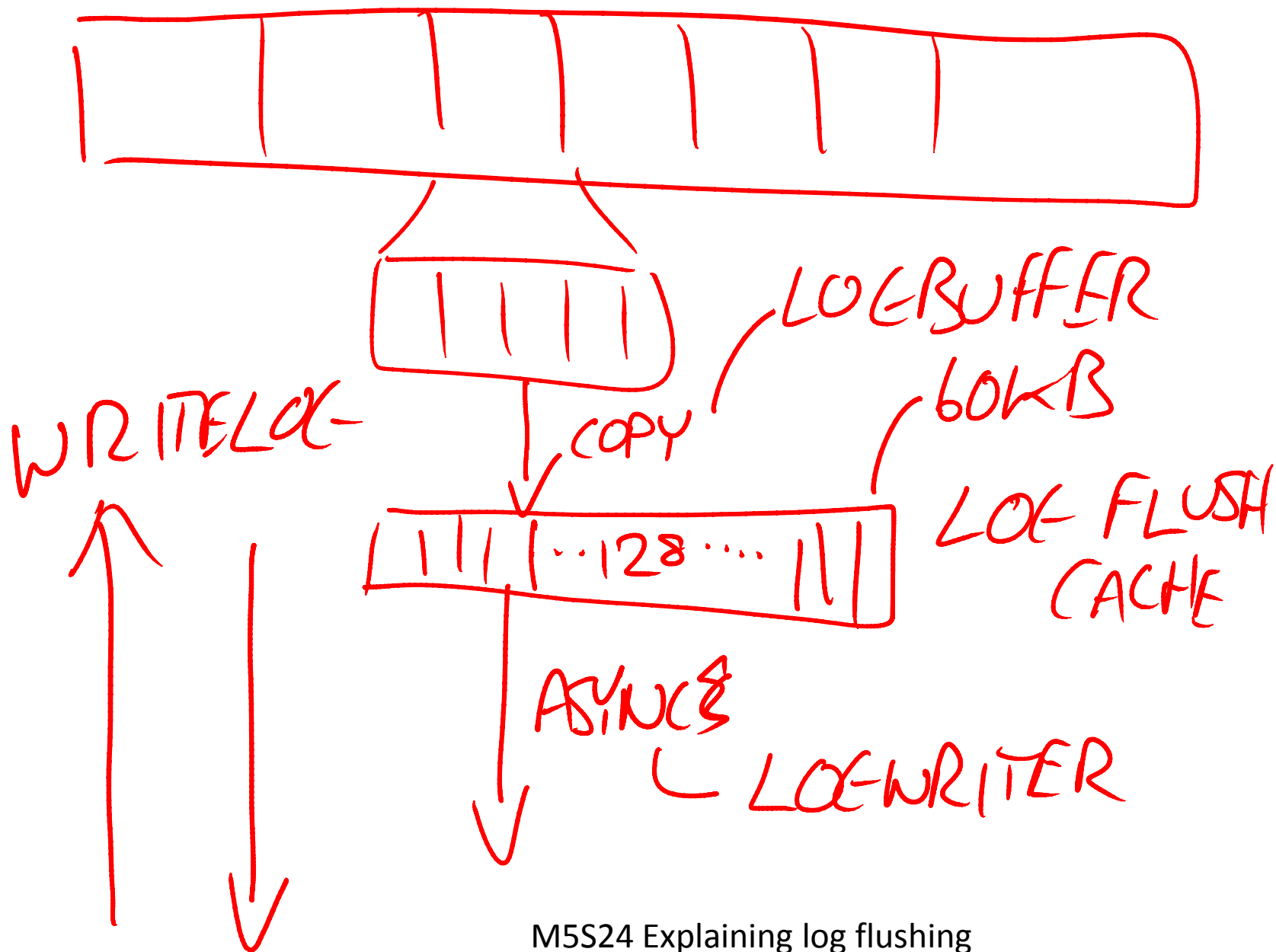


M5S24 Showing what happens when a transaction commits and the final log block for it is flushed to disk, when sync DBM or AG is involved. The local I/O will complete first (3) but the transaction commit cannot complete (5) until the mirror acknowledges the remote I/O (4).



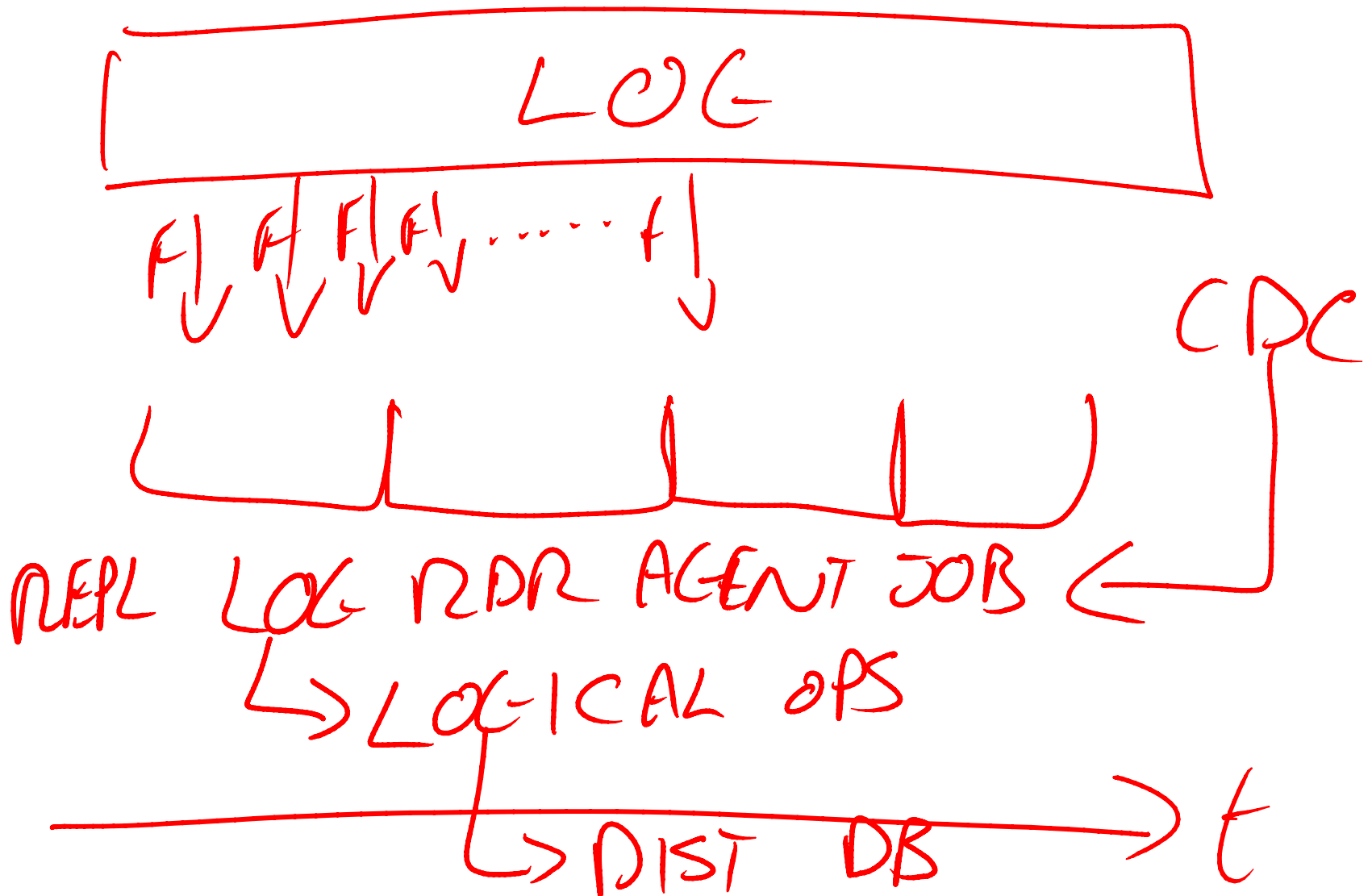
Then we got into a tangent about mirroring states/failovers. If you're running in high-availability configuration (with a witness), if the principal loses connectivity with the witness and with the mirror, it will shut down as it assumes the mirror has failed over.

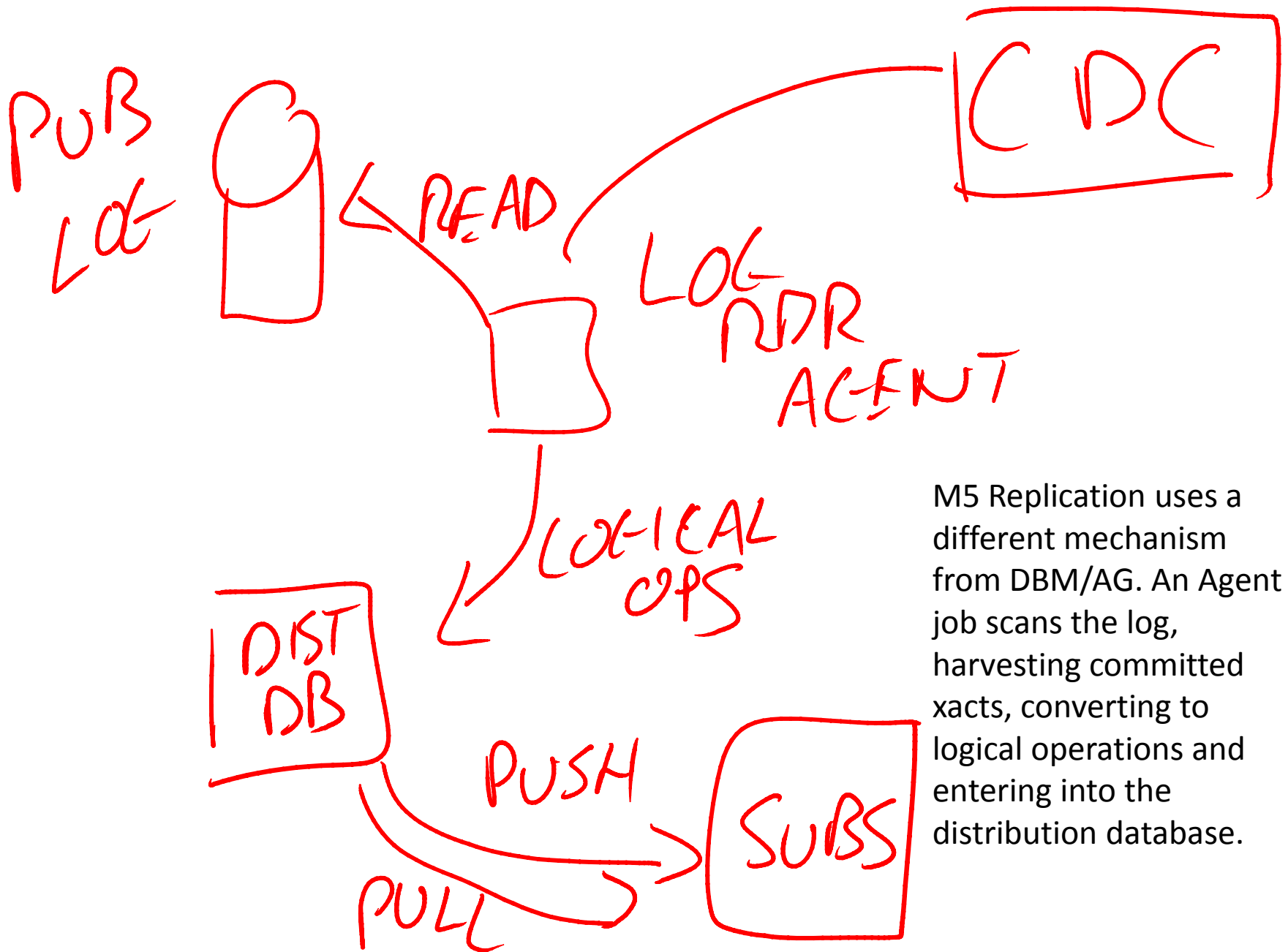
To prevent network failures causing spurious shutdowns and failovers, put the witness in the same DC as the mirror, so principal can always see the mirror.



M5S24 Explaining log flushing

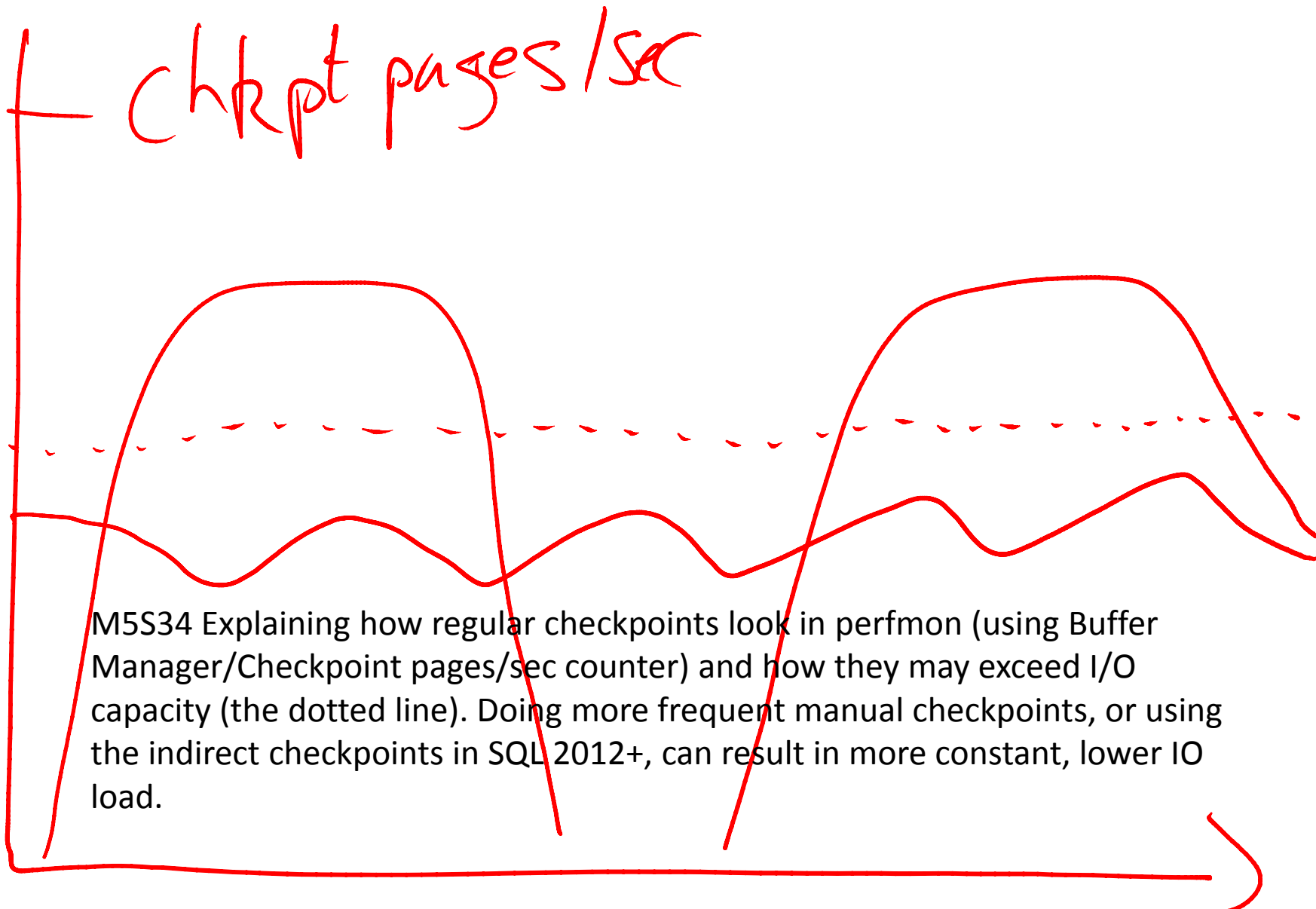
M5 When does repl/CDC pick up transactions from the log? It's done async by the log reader agent job.



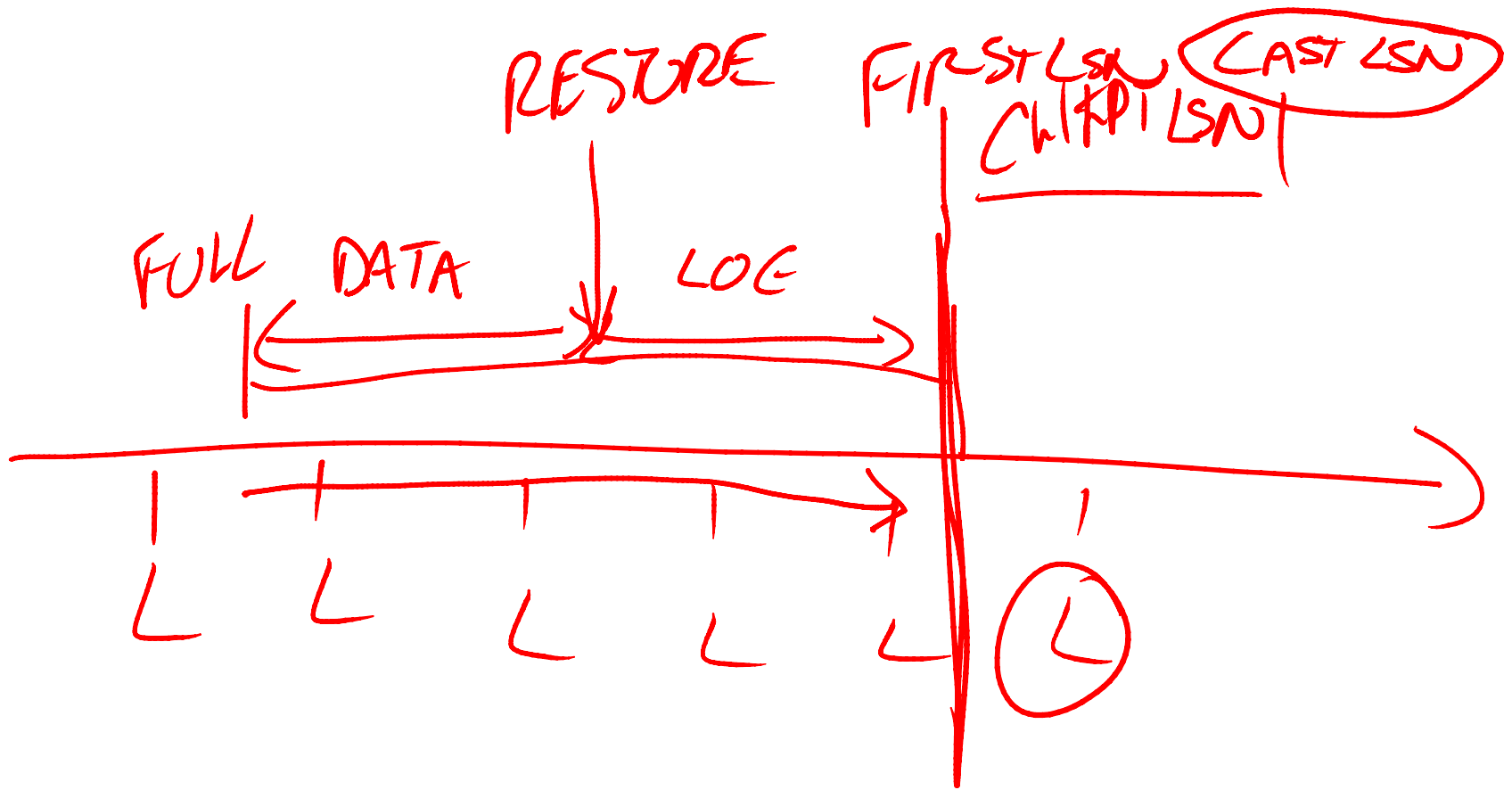


M5 Replication uses a different mechanism from DBM/AG. An Agent job scans the log, harvesting committed xacts, converting to logical operations and entering into the distribution database.

chkpt pages/sec

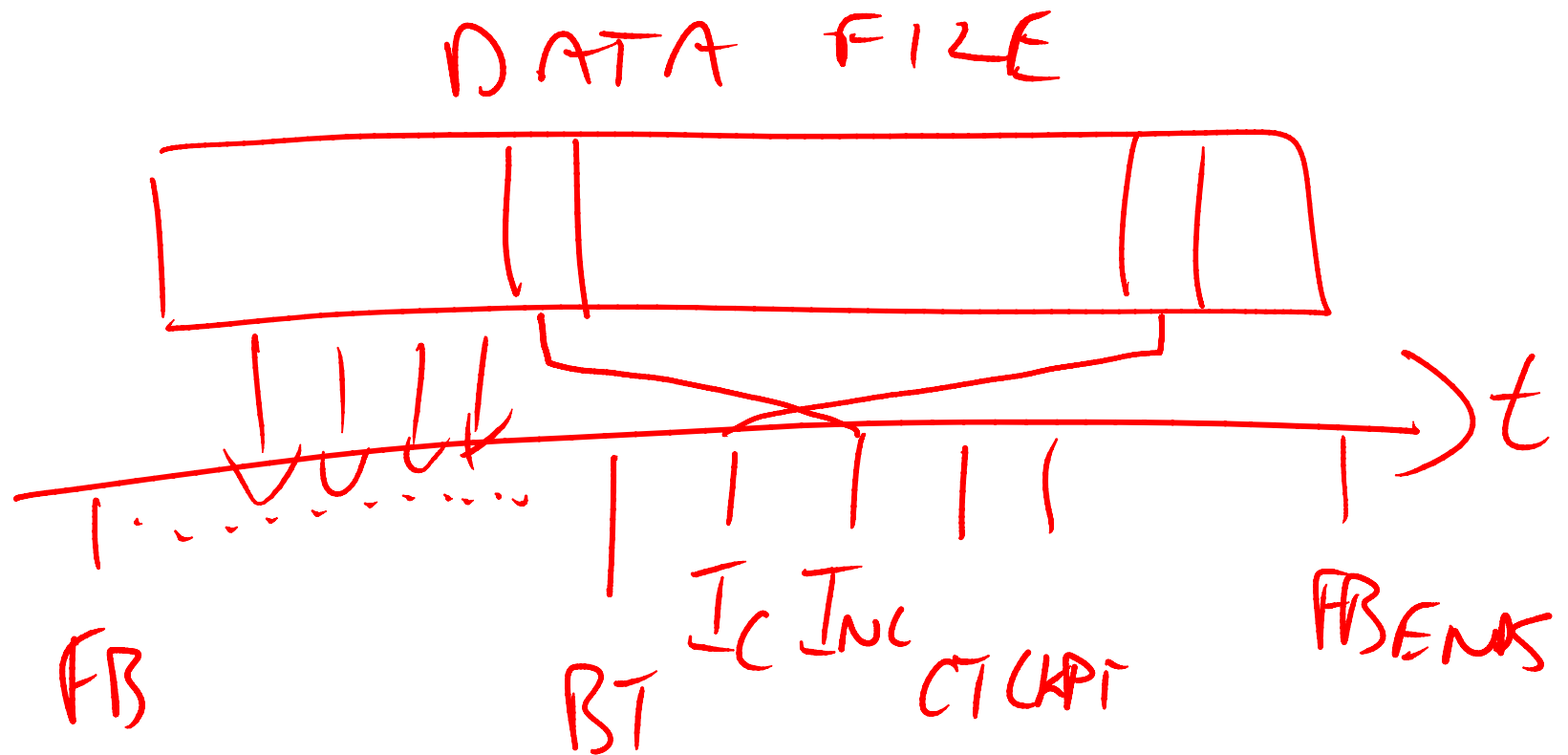


M5S34 Explaining how regular checkpoints look in perfmon (using Buffer Manager/Checkpoint pages/sec counter) and how they may exceed I/O capacity (the dotted line). Doing more frequent manual checkpoints, or using the indirect checkpoints in SQL 2012+, can result in more constant, lower IO load.



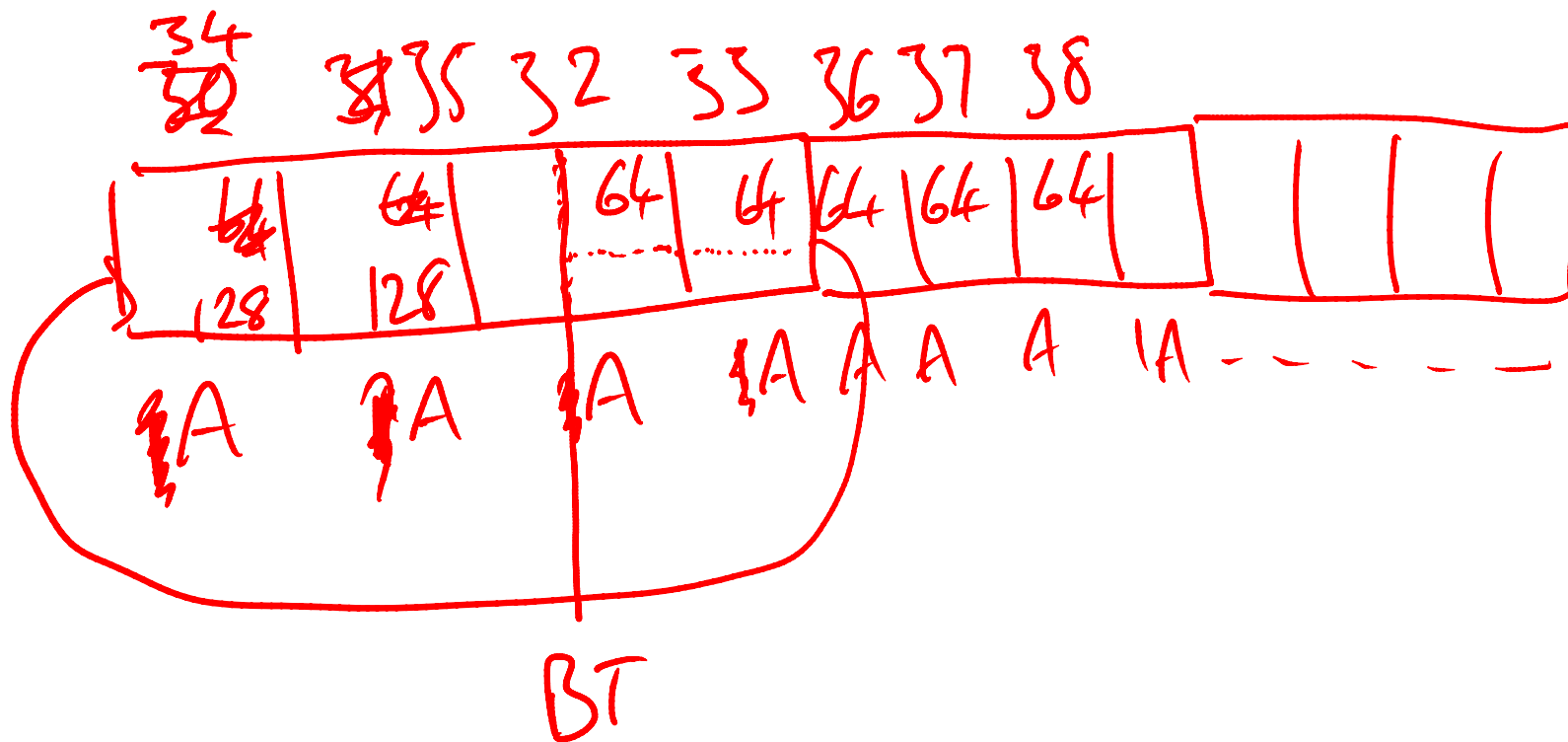
M5S46 It's possible for concurrent log backups to occur with a data backup (full or differential). Log clearing cannot occur though, as the data backup will back up the log. When the data backup finishes, deferred log clearing takes place.

The point in time you get when you restore a data backup (and don't restore any more) is the point when the data reading portion of the backup ended (marked RESTORE in the diagram).

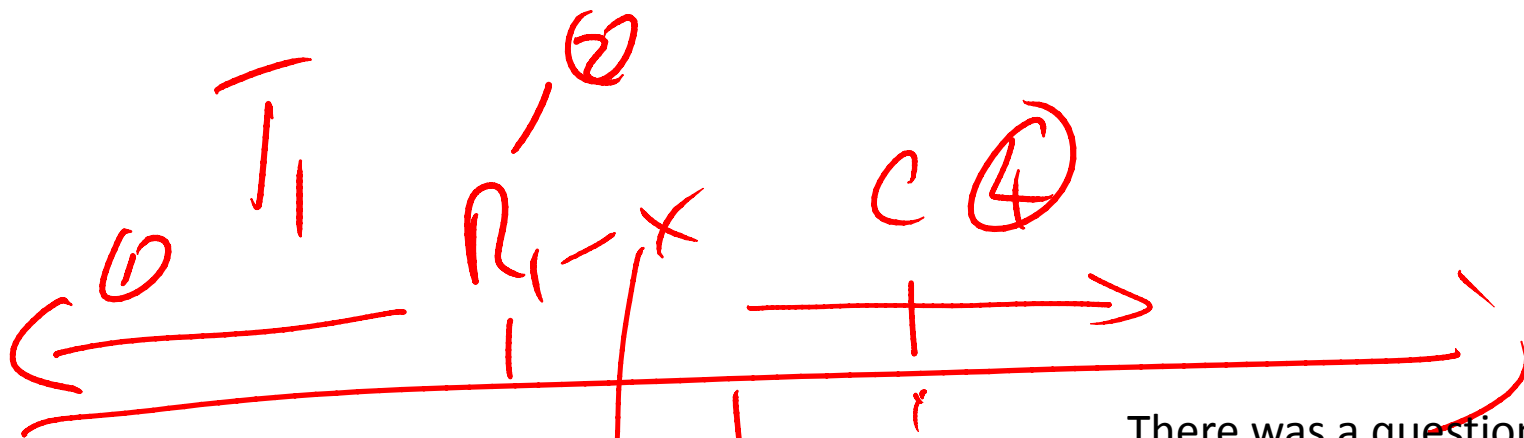


M5S46 Why does a full/diff backup contain log? It's to make sure that the restored database is transactionally consistent.

Scenario: full backup starts and reads nonclustered index page before insert. Transaction starts and inserts a row into clustered index page (IC) and nonclustered index page (INC). Checkpoint flushes out both pages. Backup then reads clustered index page with insert. If no log in the backup, restored database would be corrupt – missing nonclustered index record.



M5S52 Explaining the circular log demo. We discussed log space reservation that happens to ensure the active transactions can roll back without having to grow the log – this accounts for extra log growths. Space reserved in the log does not make a VLF become active – as there are no log records written out, just empty space reserved.



There was a question about lock waits. Each lock has a queue of threads waiting to acquire the lock. Current owner releases the lock and notifies head of queue that they now own the lock.

M5S57 Explaining how even if two transactions change the same page, with one committed and one needing rolled back during crash recovery, they cannot both be changing the same row (because of locking) and so crash recovery works fine.

1, 2, 3, 4 were crash scenarios based on the LSN on the page compared to R1 or R2.

PENDING Q

BT

1

M5S60 Nested transactions don't exist.
Nested begins don't cause log records, nor
do nested commits. Rollback at any time
rolls everything back.

BT

SS



BT



BT - nested 2



BT 3



CT 2

RT

System transactions (e.g.
implementing a page
split) DO commit before
your transaction does.
They do not roll back if
your transaction rolls
back.



CT

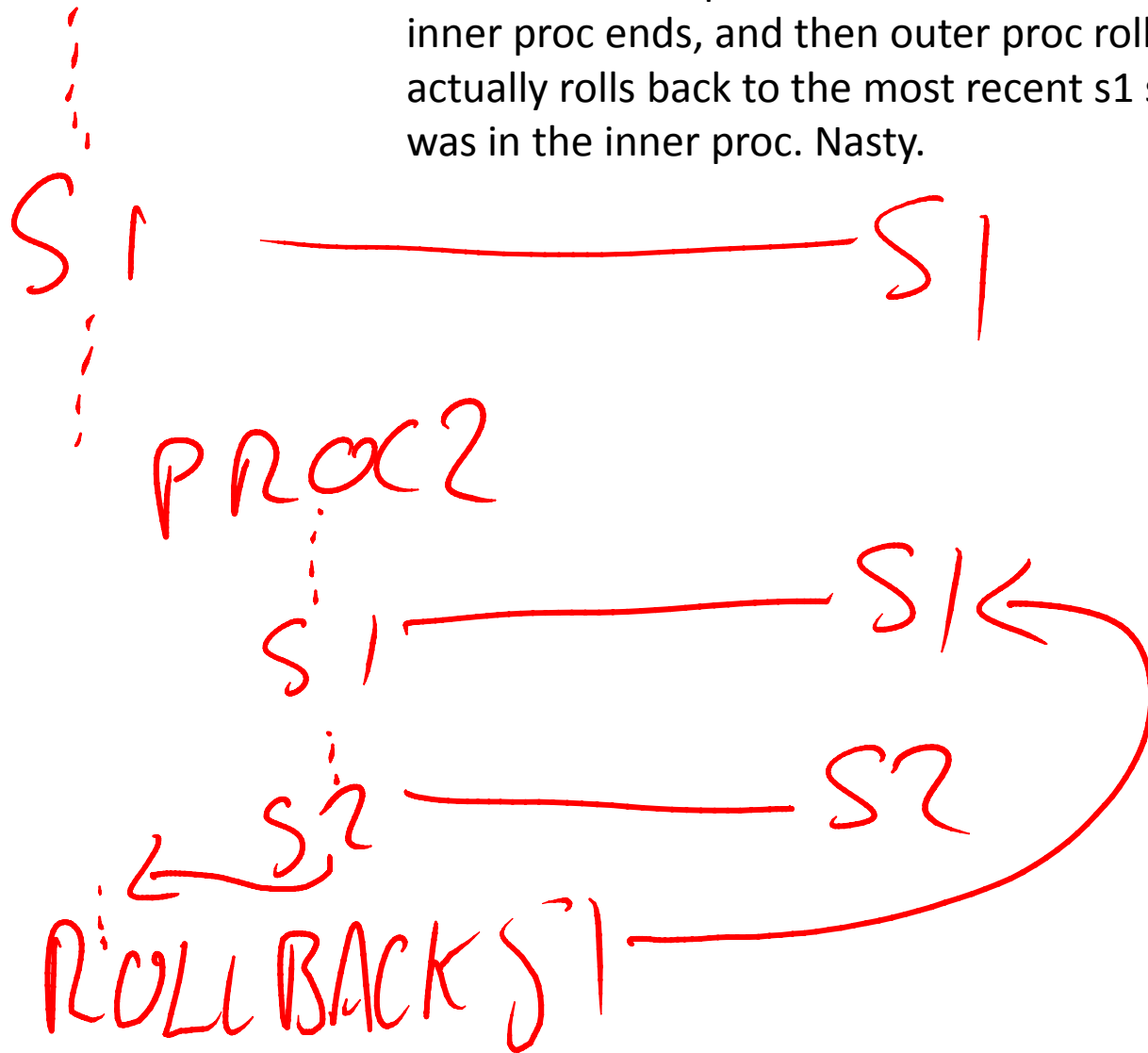


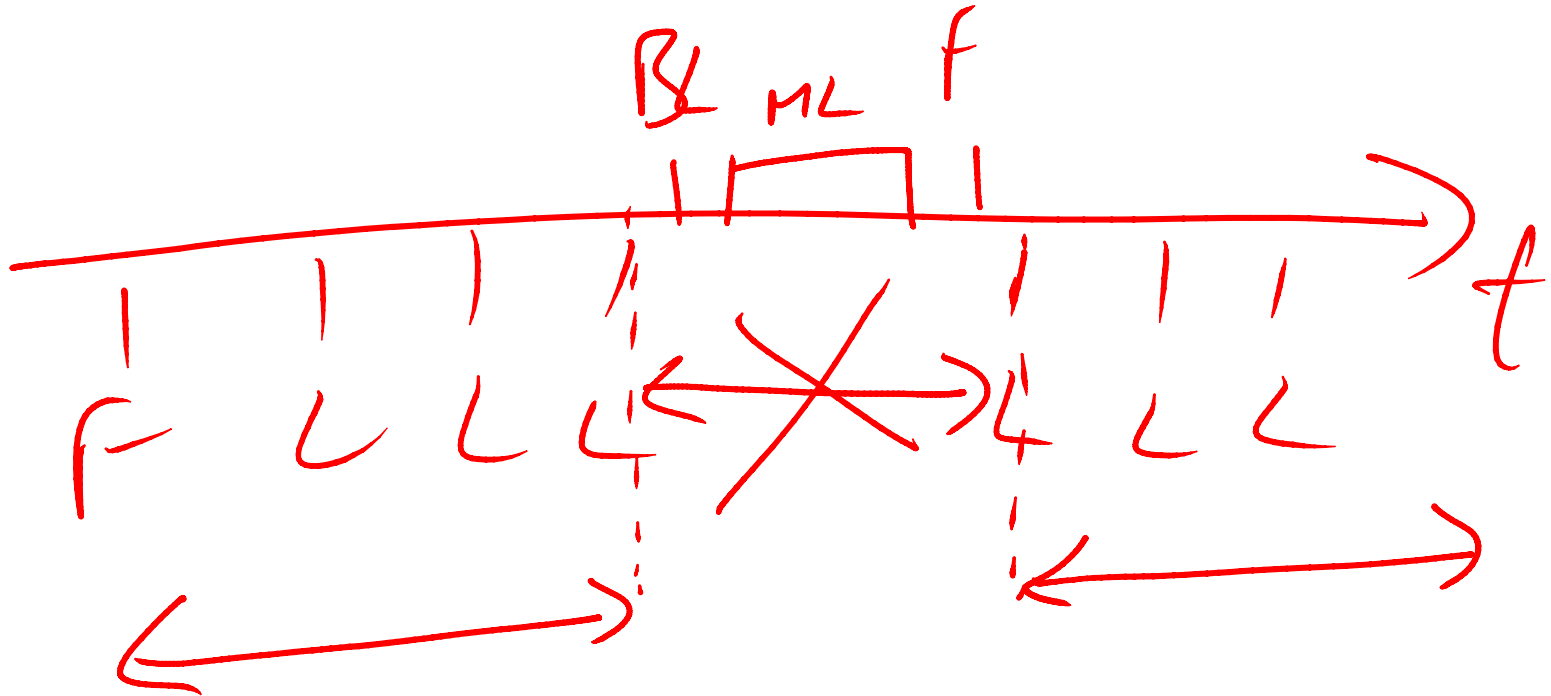
WHILE
 @@TRANCOUNT > 0
 COMMIT TRAN;

M5S60 Seen code like this? Example of how devs may close out unbounded transaction nesting in their code.

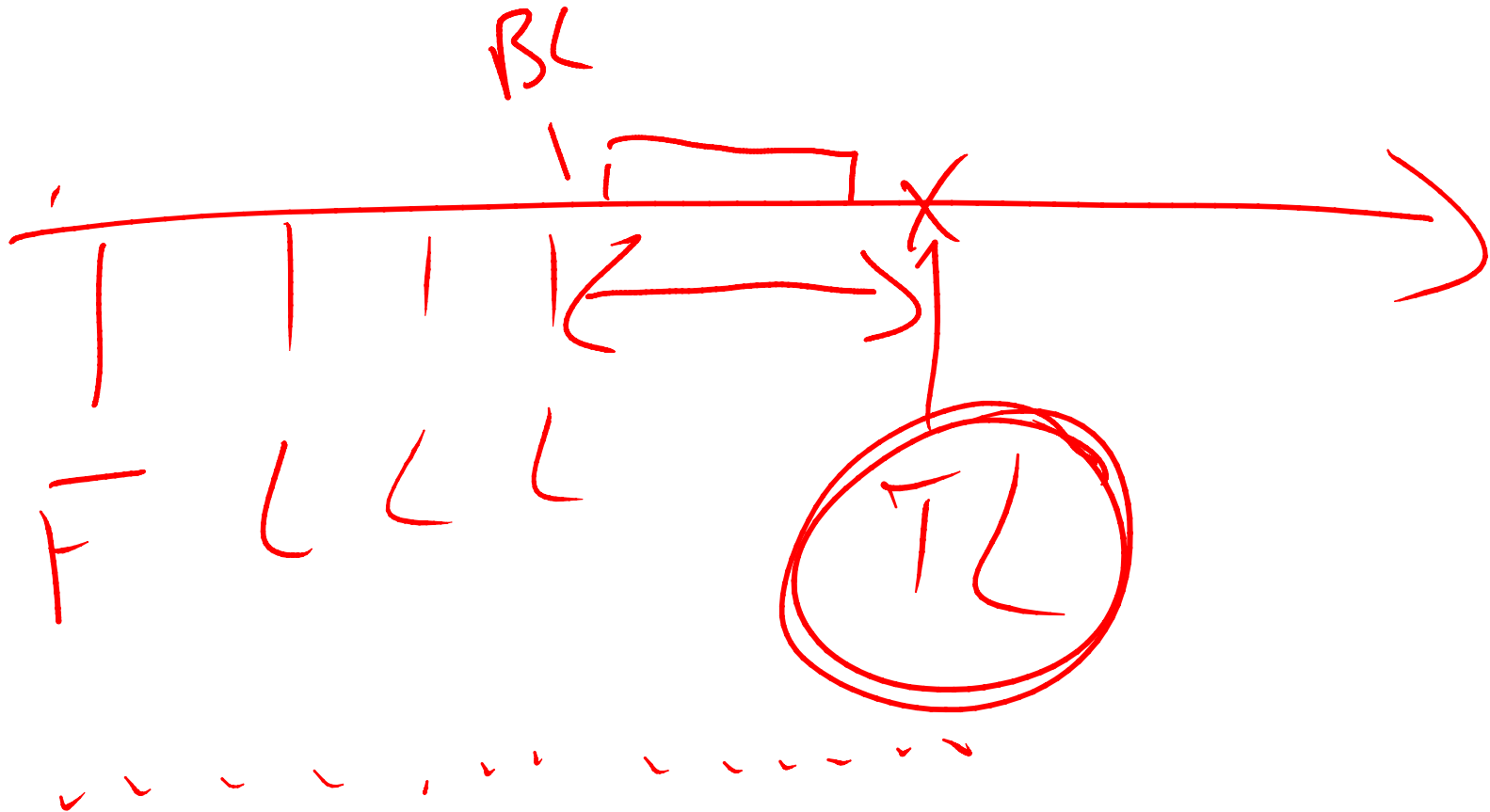
PROC 1

M5S60 You can use savepoints to create rollback points within a transaction. Beware, the stack of savepoint names doesn't get pruned until your transaction commits. E.g. outer and inner proc both use s1 as a save point name. If inner proc ends, and then outer proc rolls back to s1, it actually rolls back to the most recent s1 save point, which was in the inner proc. Nasty.





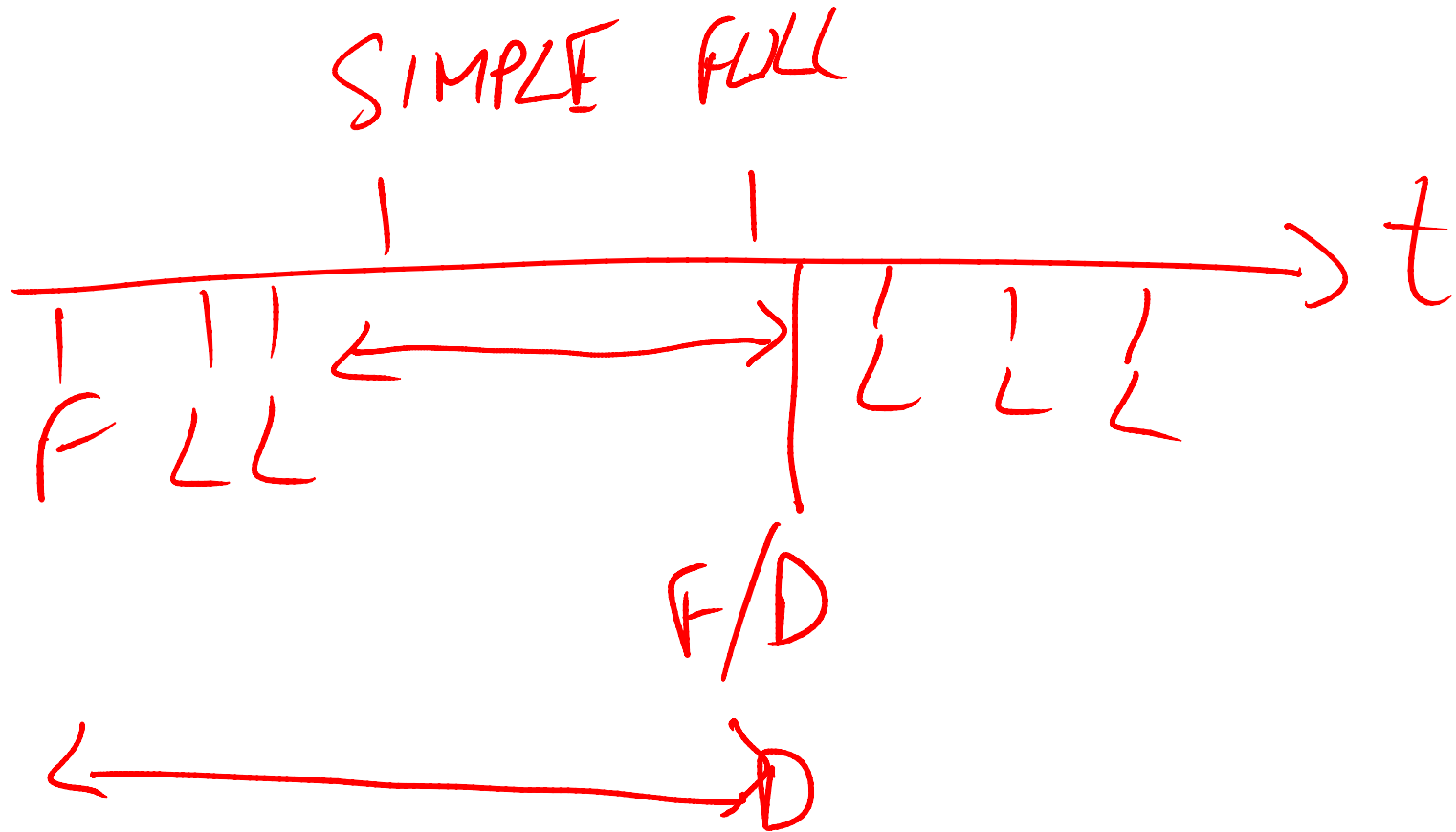
M5S64 If a log backup contains a minimally-logged operation, it cannot be used to do a point-in-time restore using STOPAT for any point in time between the beginning and end of the backup. It can be used as part of a restore sequence as normal though.

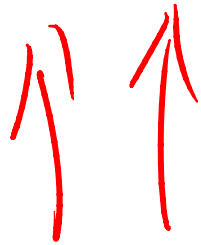
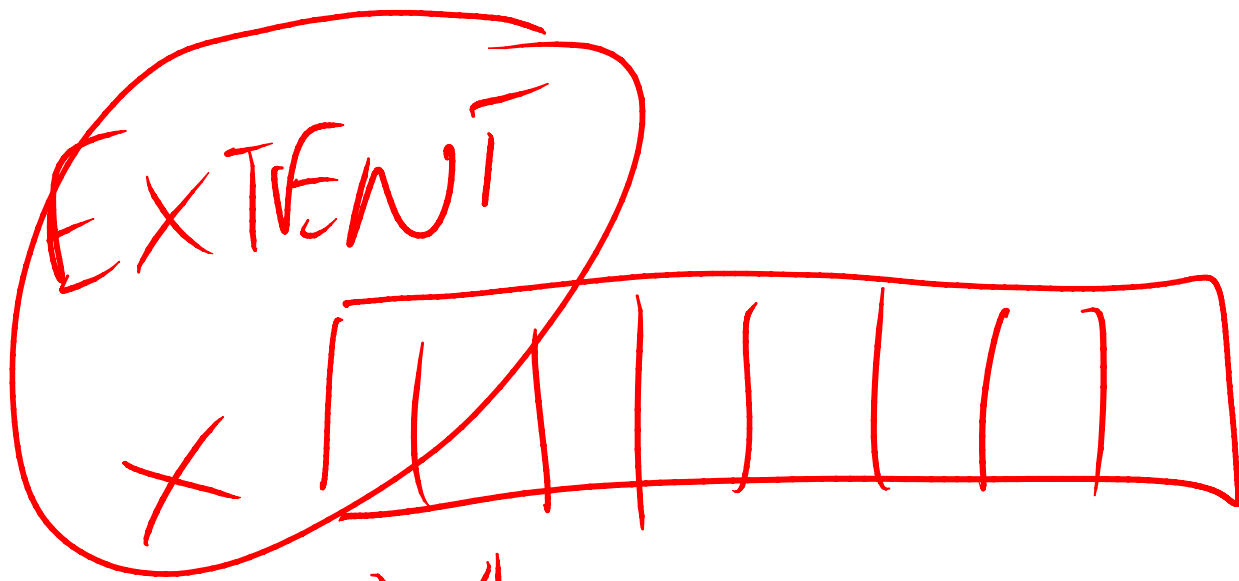


M5S64 Explaining that if a minimally-logged operation is present in a log backup, you cannot do a point-in-time restore to any time covered by that backup (except the start or end of that time period). You also cannot do a tail-of-the-log backup after a crash that destroys the data files if there has been a minimally-logged operation since the last log backup. Well in 2008 R2+, you can, but the log backup is invalid.

M5S61 If you switch to simple and then back to full, you can use a full or differential backup to reestablish the log-backup chain.

Don't switch to simple if you can avoid it as it dangerously reduces your restore options down to whatever is your most recent full backup.





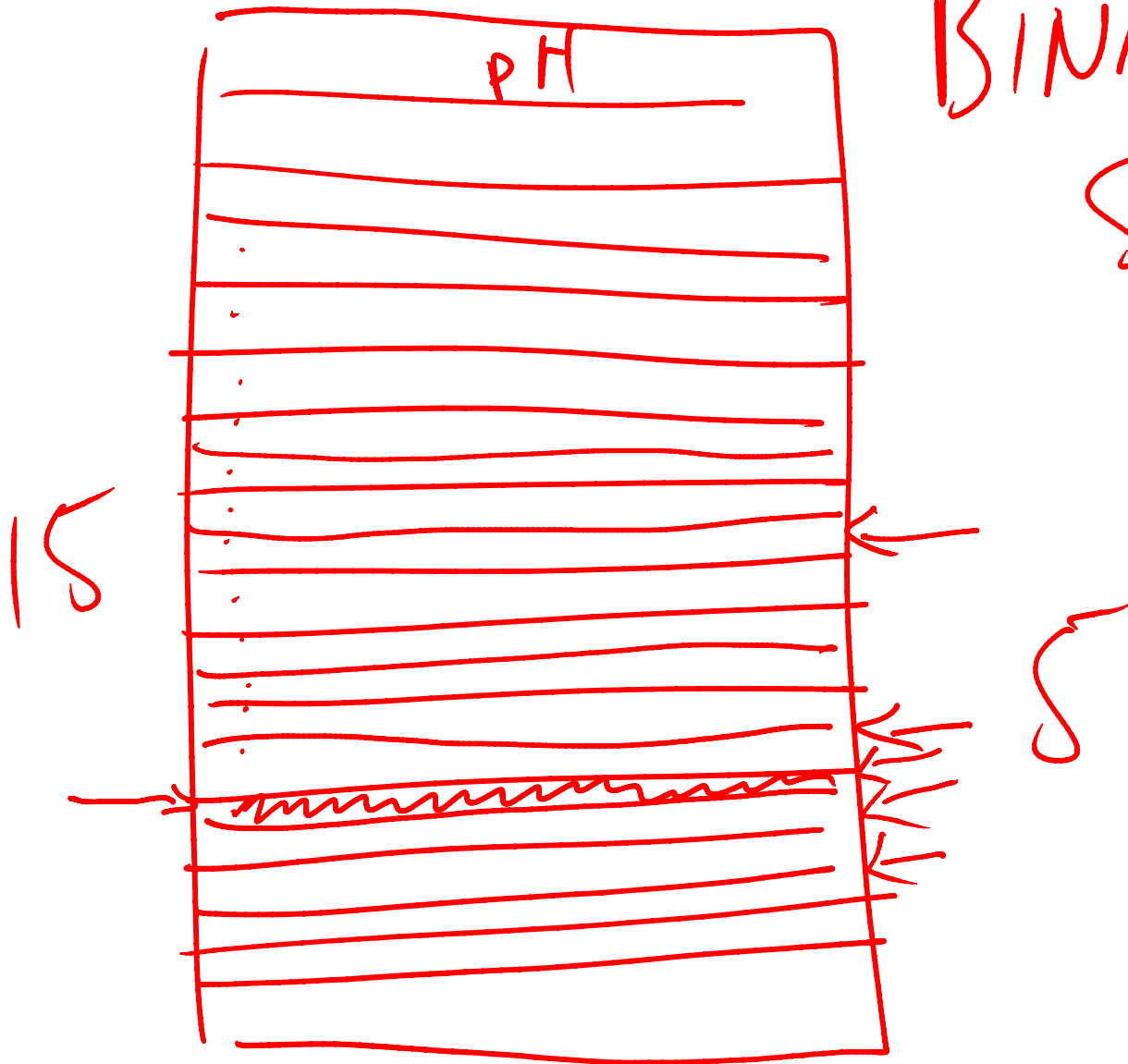
PROBE

M5S66 Explaining part of the deferred-drop functionality. Before an extent can be deallocated, an X lock is acquired on it and all 8 page locks are probed (instant acquire X lock and release) to make sure nothing else has them locked.



MSS67 VLF fragmentation is bad because of having to search through the VLFs for a particular VLF sequence number. See the KB article link for more info.

BINARY SEARCH



M7S5 Explaining how a record is found on a page using binary search rather than brute force search.

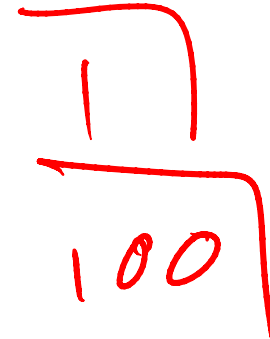
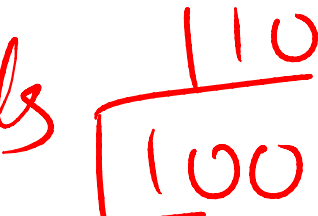
FANOUT

1 million pages

80

800 = 7 levels

FANOUT = 10

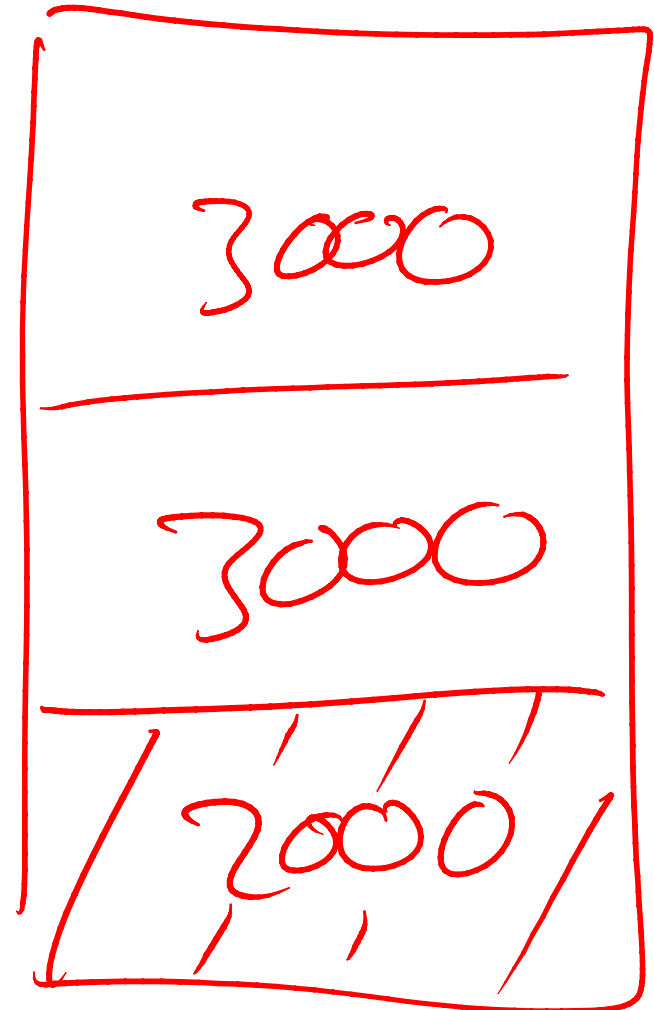
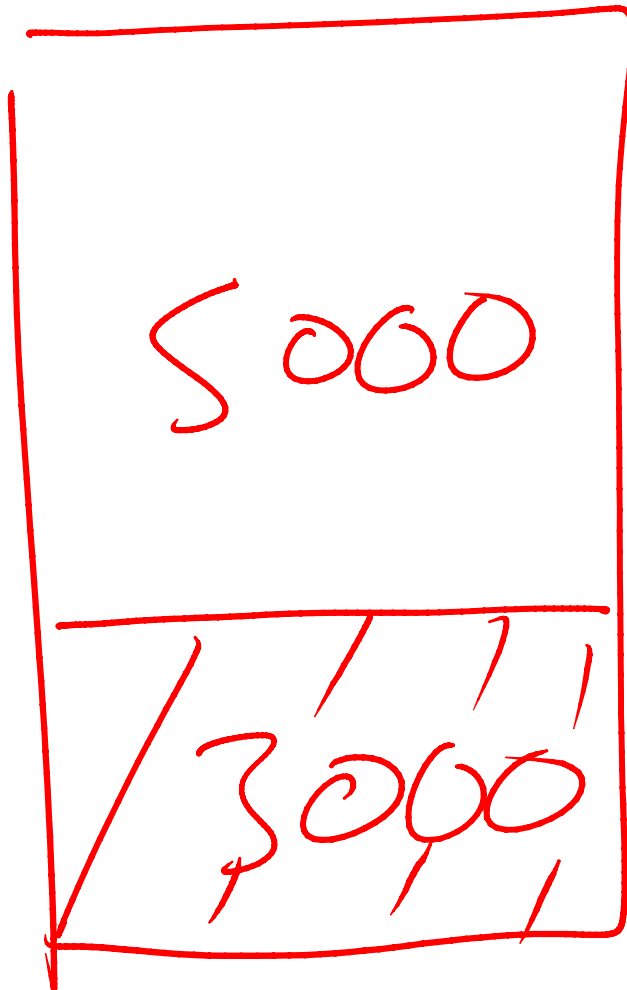


4 levels

FO = 100

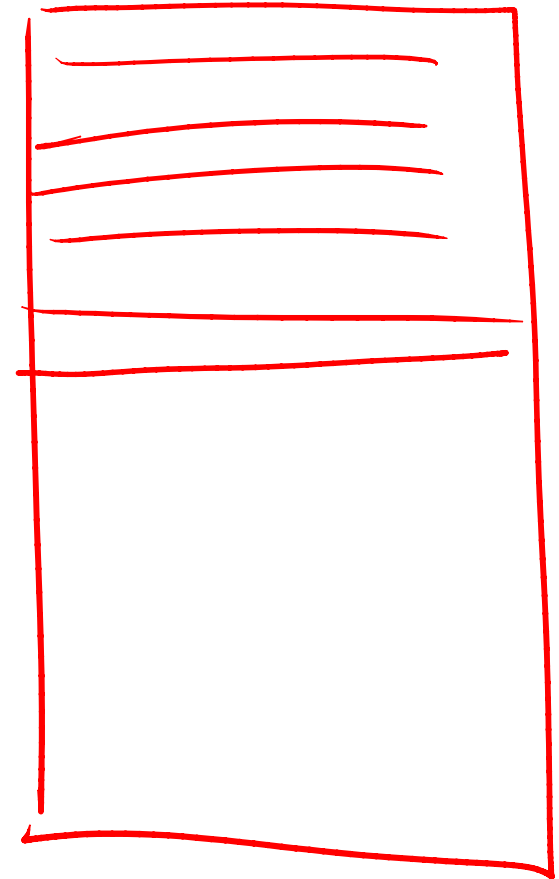
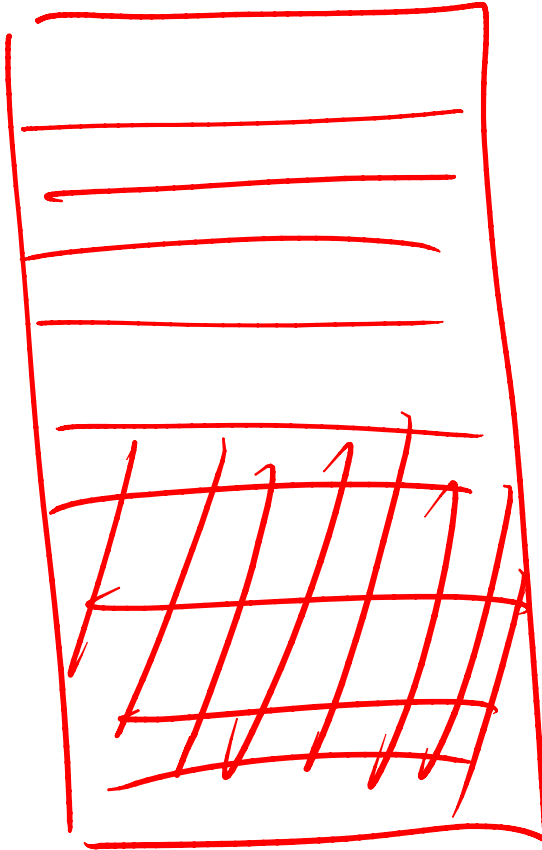
1 million

M7S5 The larger the index key, the smaller the fanout and the deeper the index -> more CPU to traverse.

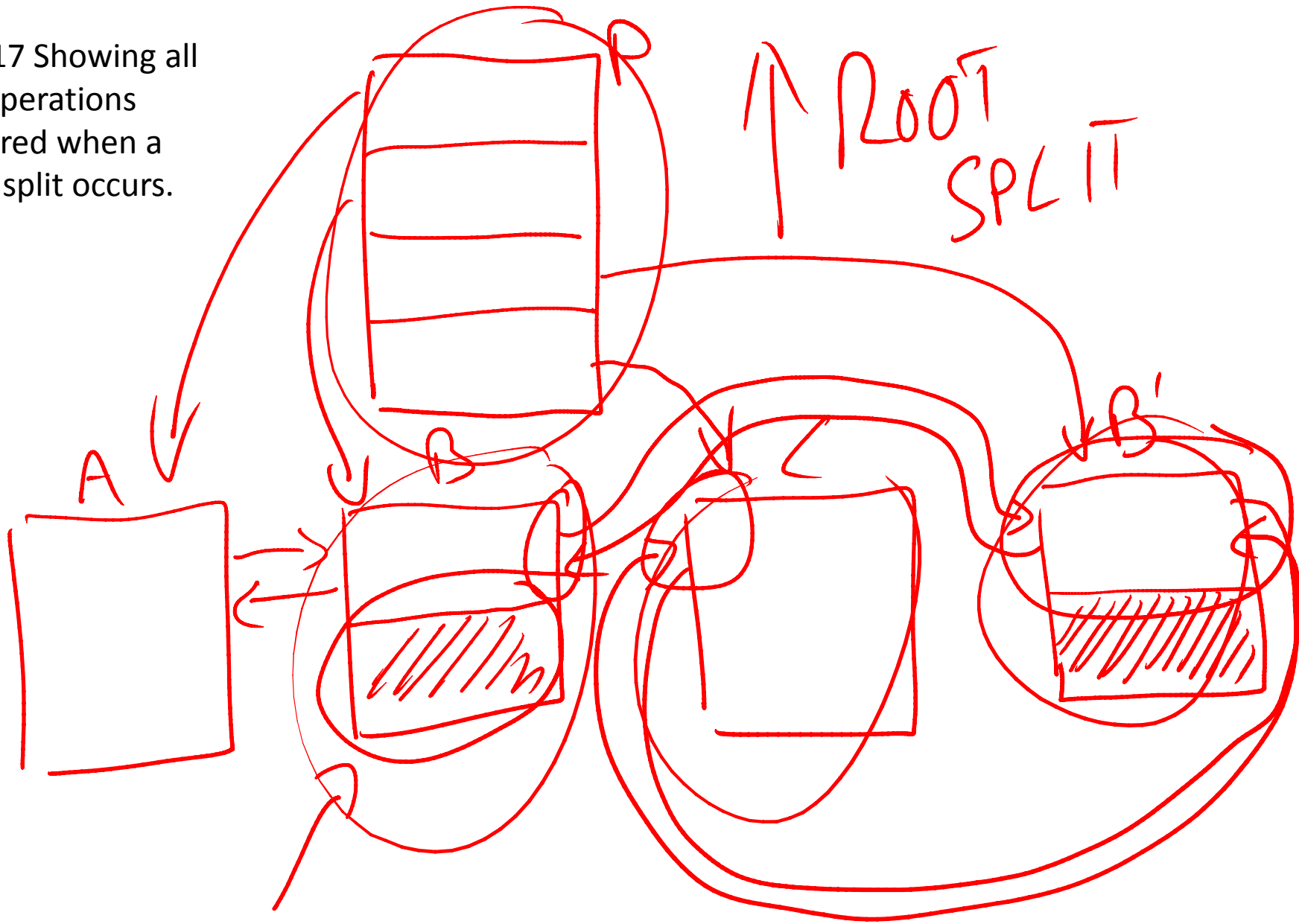


M7S14 Explaining how fixed-size large rows leads to low page density and wasted space.

M7S16 When a page split occurs, it's usually 50/50 and creates 50% free space on each page.



M7S17 Showing all
the operations
required when a
page split occurs.



BT

MS

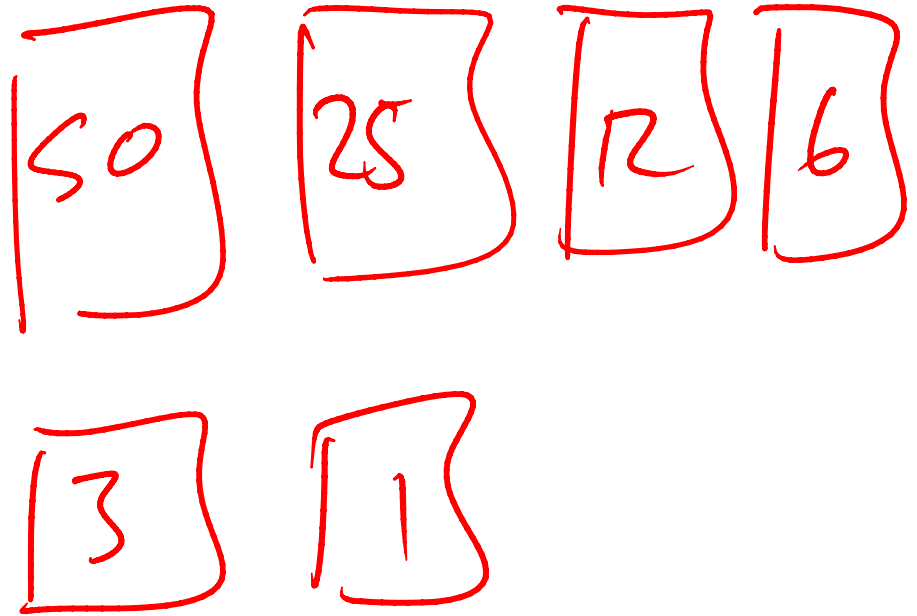
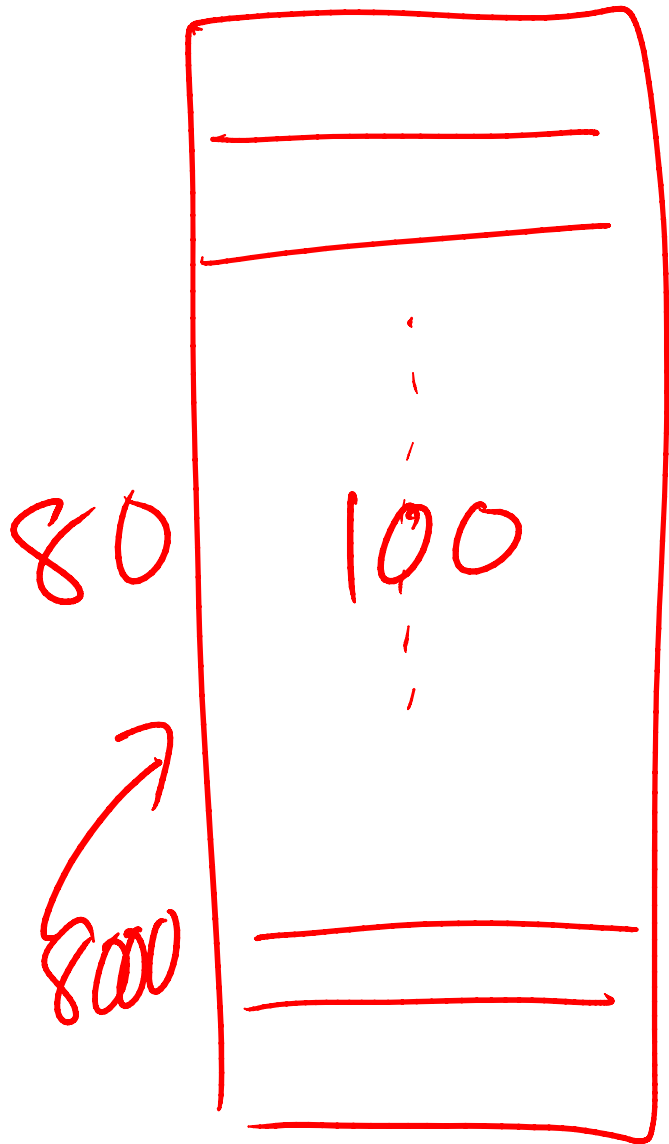
BT

~

CT

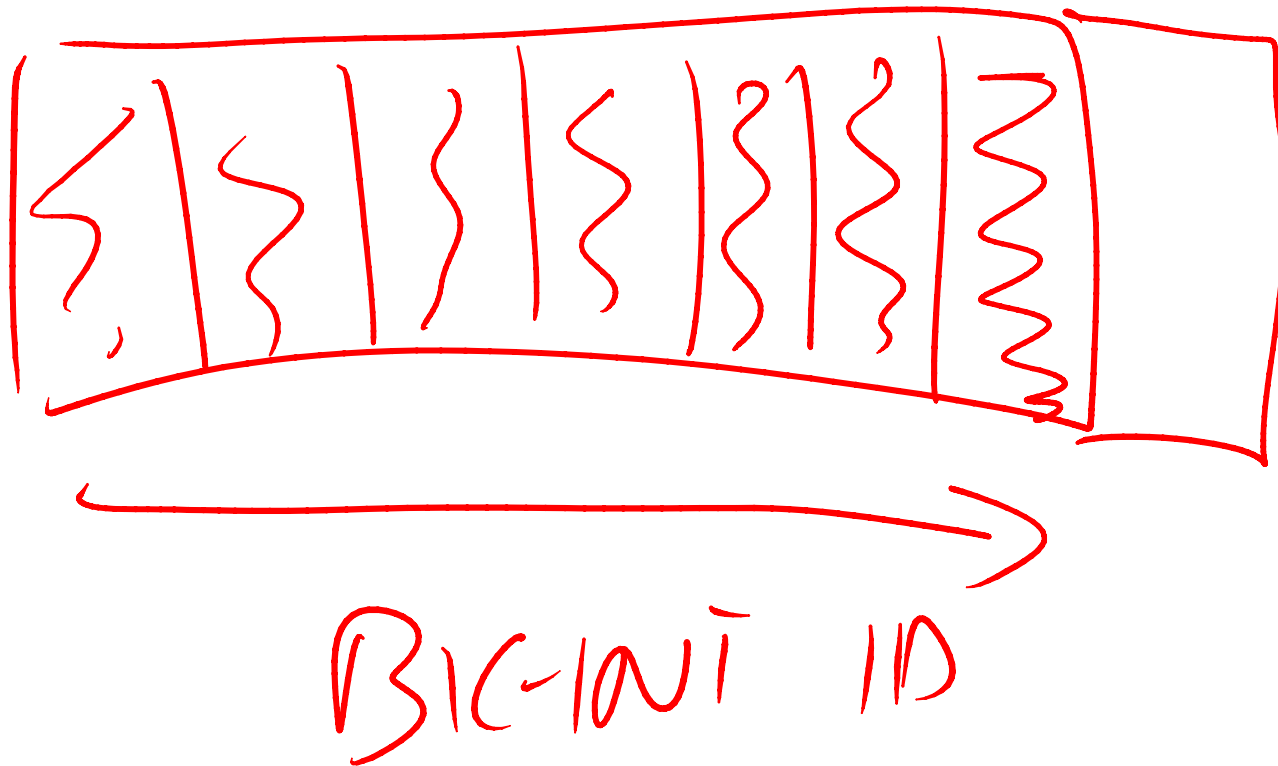
SYSTEM XACT

M7S17 Explaining how the page split is done under the covers as a system transaction, owned by your transaction. Once committed, the system transaction will not roll back if your transaction does – so the page split is permanent.



M7S18 Explaining how the Storage Engine is not smart sometimes. Imagine a page full of 80-byte rows. Inserting an 8000-byte row there will lead to 6 splits before enough space is created.

LOP_DELETE_SPLIT



M8S18 Append-only insert pattern fills pages on right-hand side of index. New page allocations in this case are called page splits too – making all methods of tracking page splits largely useless (until 2012 with Xevents). You can look for LOP_DELETE_SPLIT log records to see splits occurring.

	NO SPLIT	SPLIT	
1000	1256	7580	x5
100	356	6772	x18
10	268	11172	x45

M7S23 The numbers from the page split logging cost demo.

COMMENTS

PAUL	JON	ERIN	ELENA
MEMBER_ID			K

M7S24 The MySpace story. The 'K' is the tiny portion of comments on Kimberly's page because she wasn't very popular back then...



ITB



M7S34 Explaining staggered index maintenance with database mirroring. See <http://bit.ly/IS04W5> for more explanation