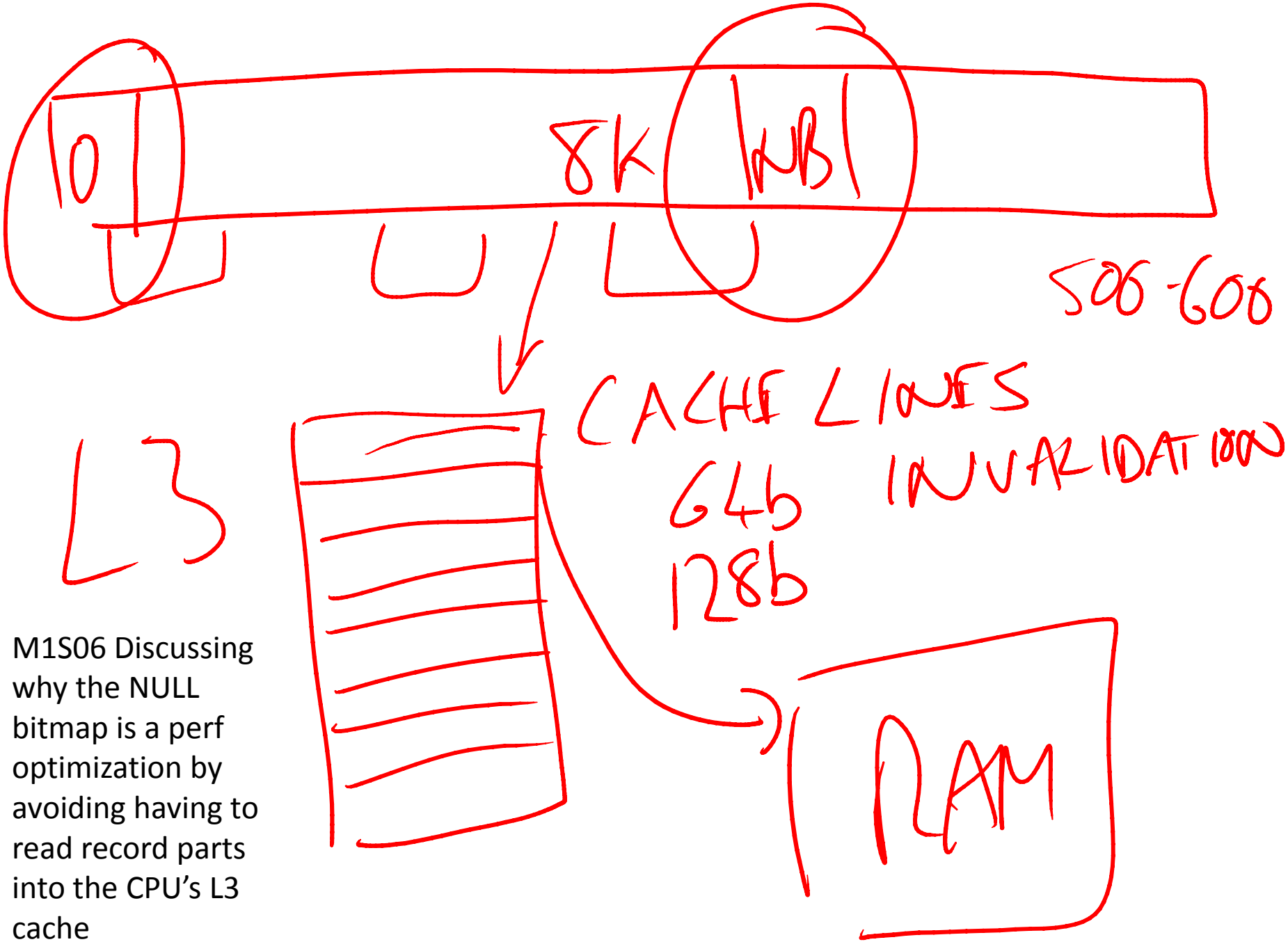
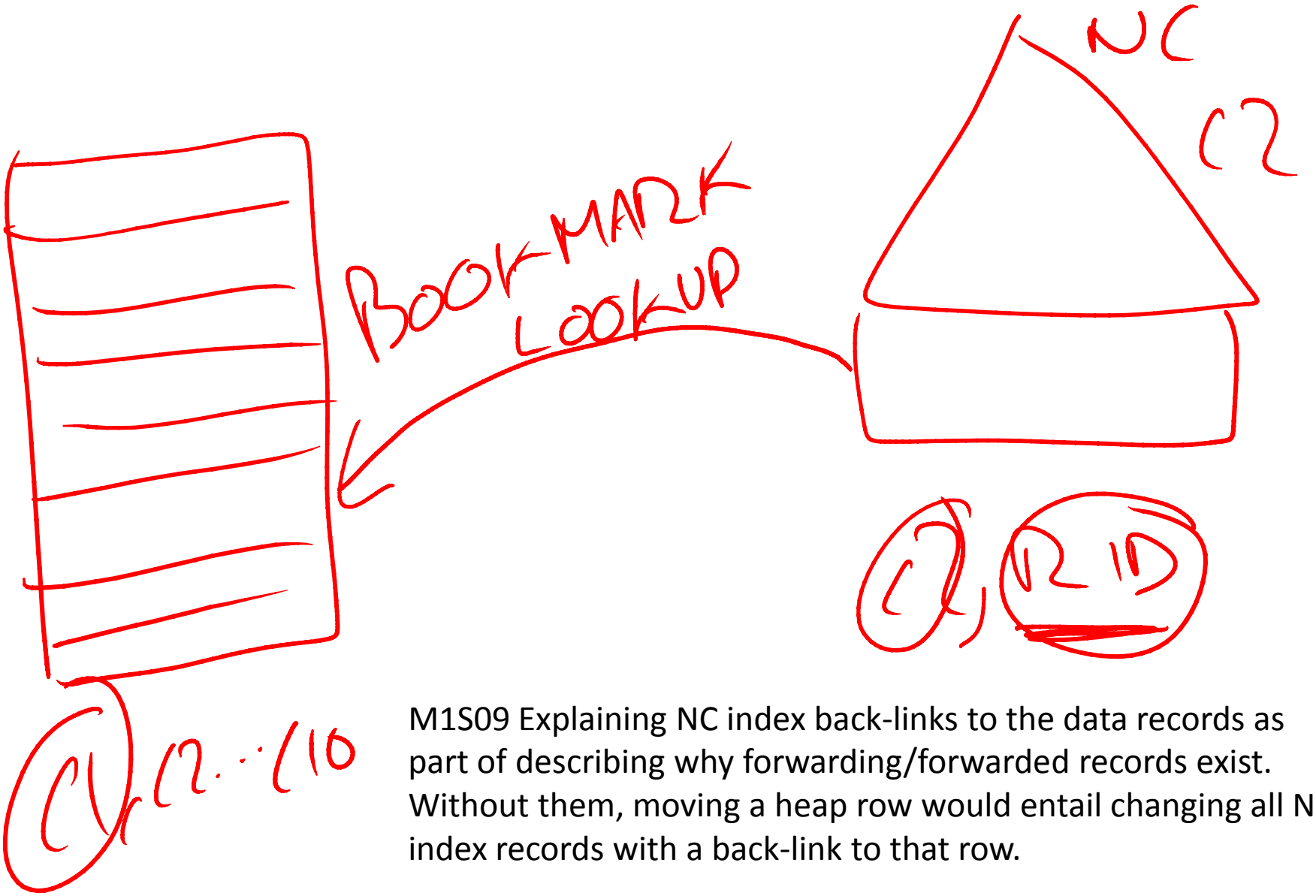
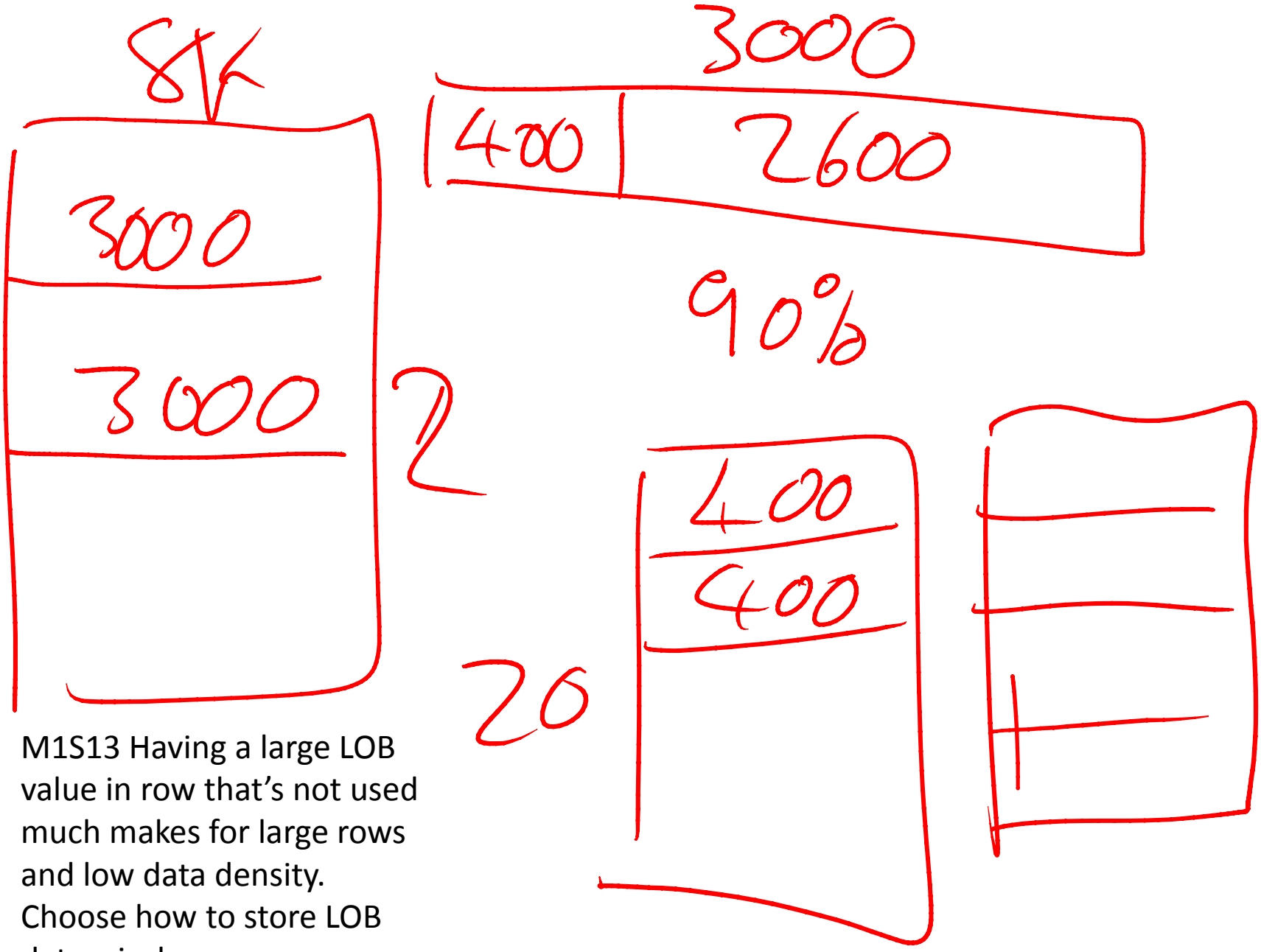
This is the hash tag to use on twitter to ask a question and get people like me and Kimberly responding with help.



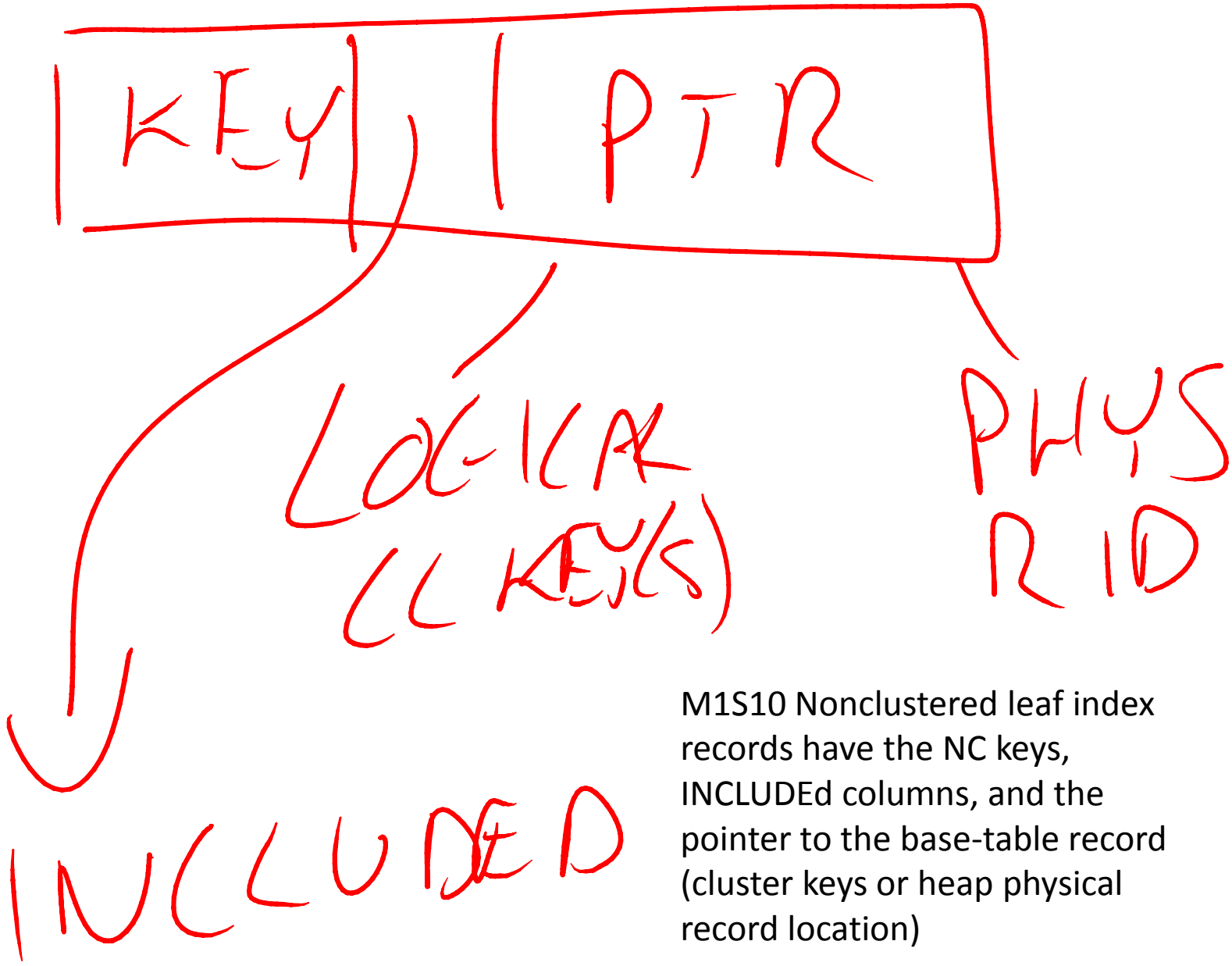
M1S06 Discussing why the NULL bitmap is a perf optimization by avoiding having to read record parts into the CPU's L3 cache



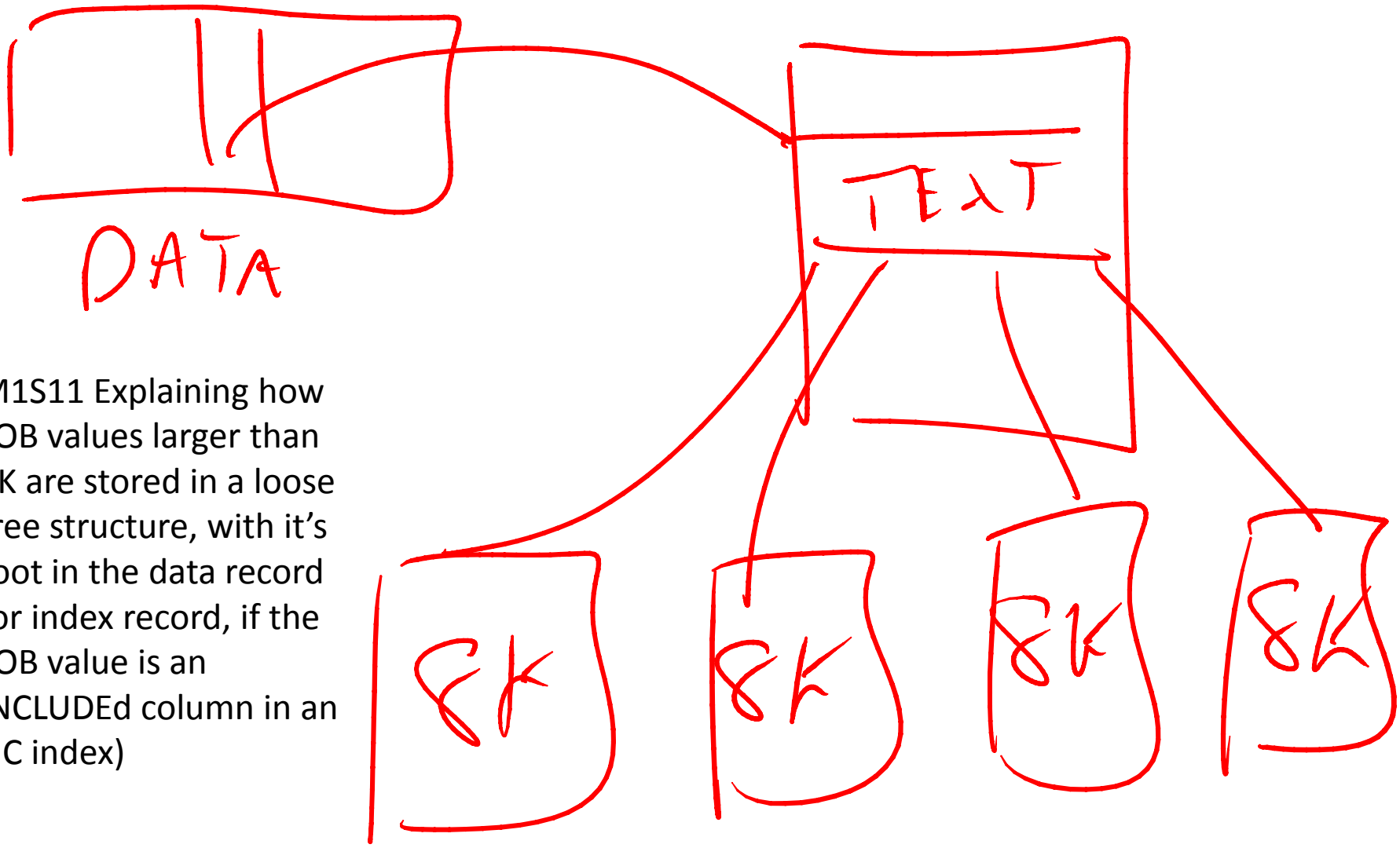
M1S09 Explaining NC index back-links to the data records as part of describing why forwarding/forwarded records exist. Without them, moving a heap row would entail changing all NC index records with a back-link to that row.



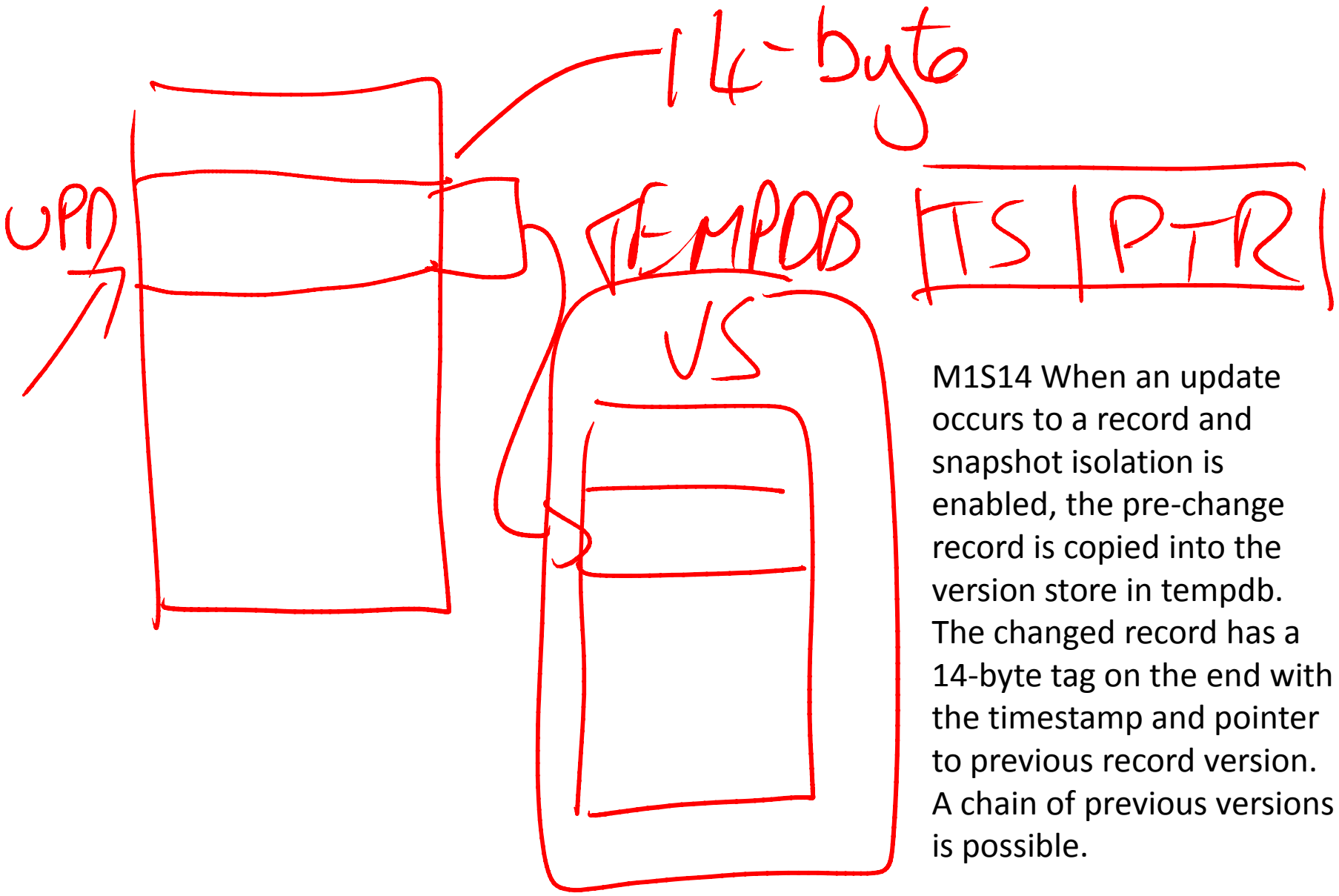
M1S13 Having a large LOB value in row that's not used much makes for large rows and low data density. Choose how to store LOB data wisely.



M1S10 Nonclustered leaf index records have the NC keys, INCLUDEd columns, and the pointer to the base-table record (cluster keys or heap physical record location)

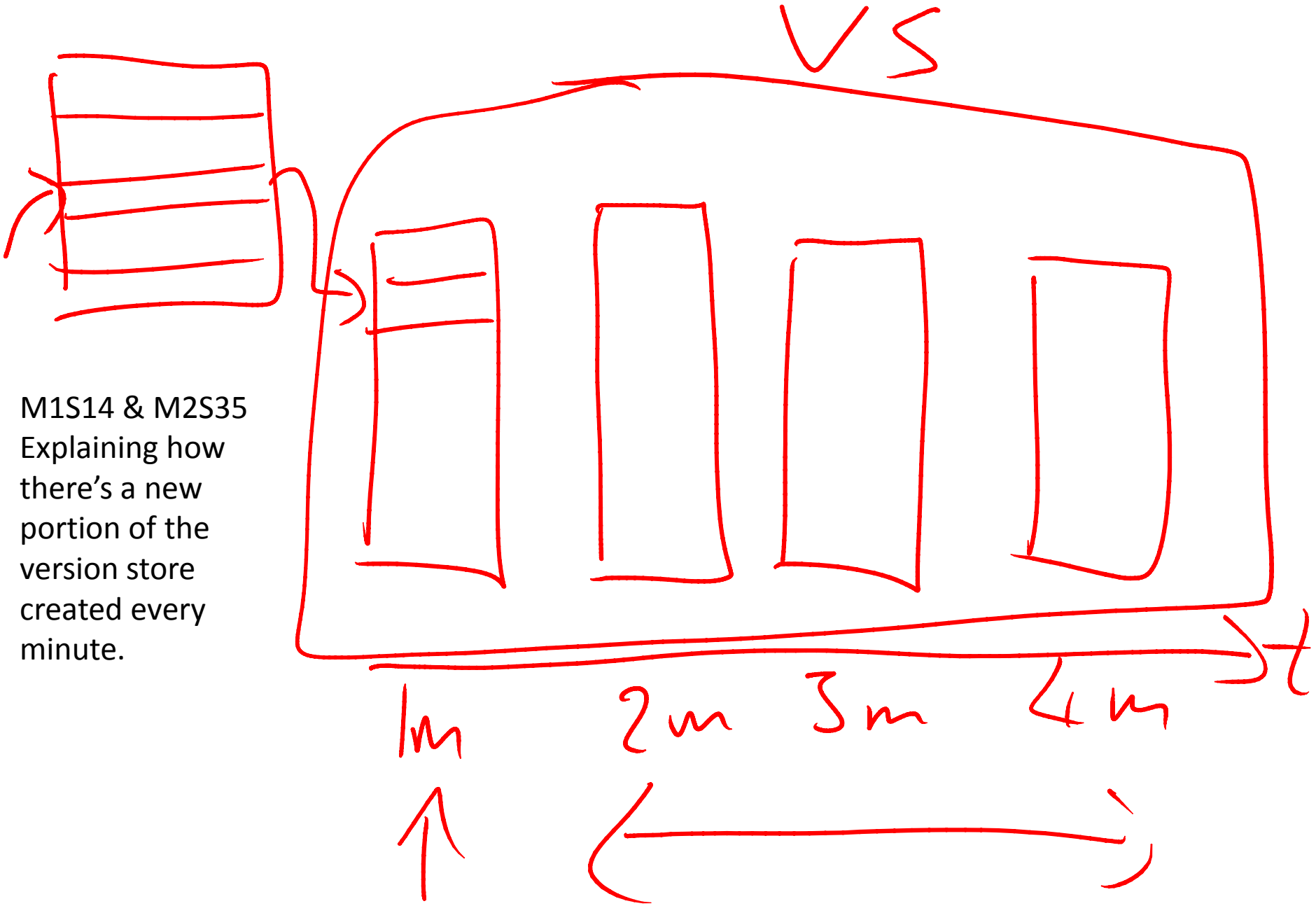


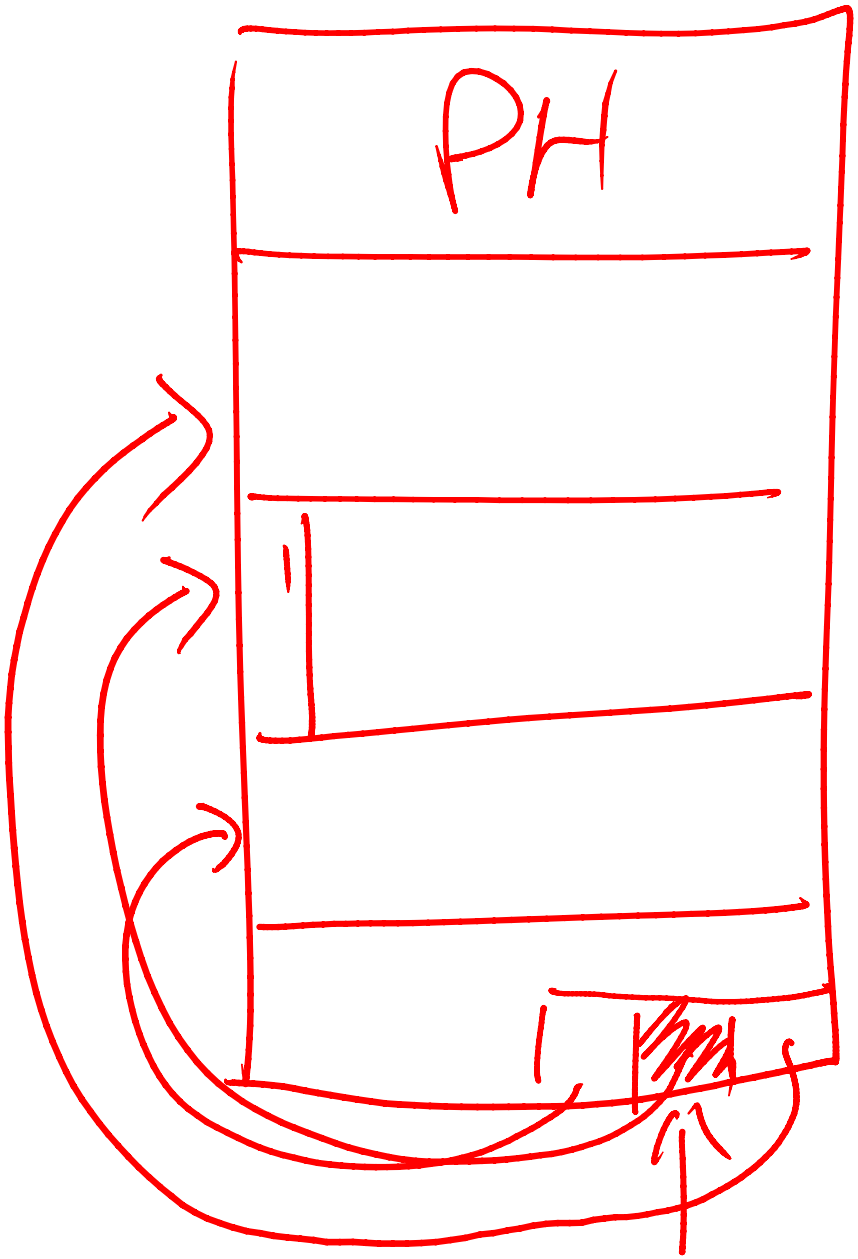
M1S11 Explaining how LOB values larger than 8K are stored in a loose tree structure, with it's root in the data record (or index record, if the LOB value is an INCLUDEd column in an NC index)



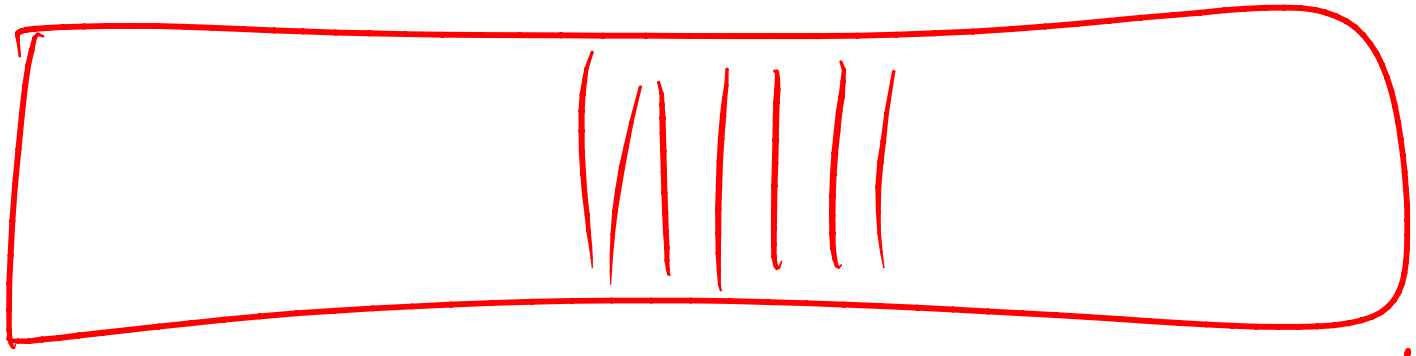
M1S14 When an update occurs to a record and snapshot isolation is enabled, the pre-change record is copied into the version store in tempdb. The changed record has a 14-byte tag on the end with the timestamp and pointer to previous record version. A chain of previous versions is possible.

M1S14 & M2S35
Explaining how there's a new portion of the version store created every minute.





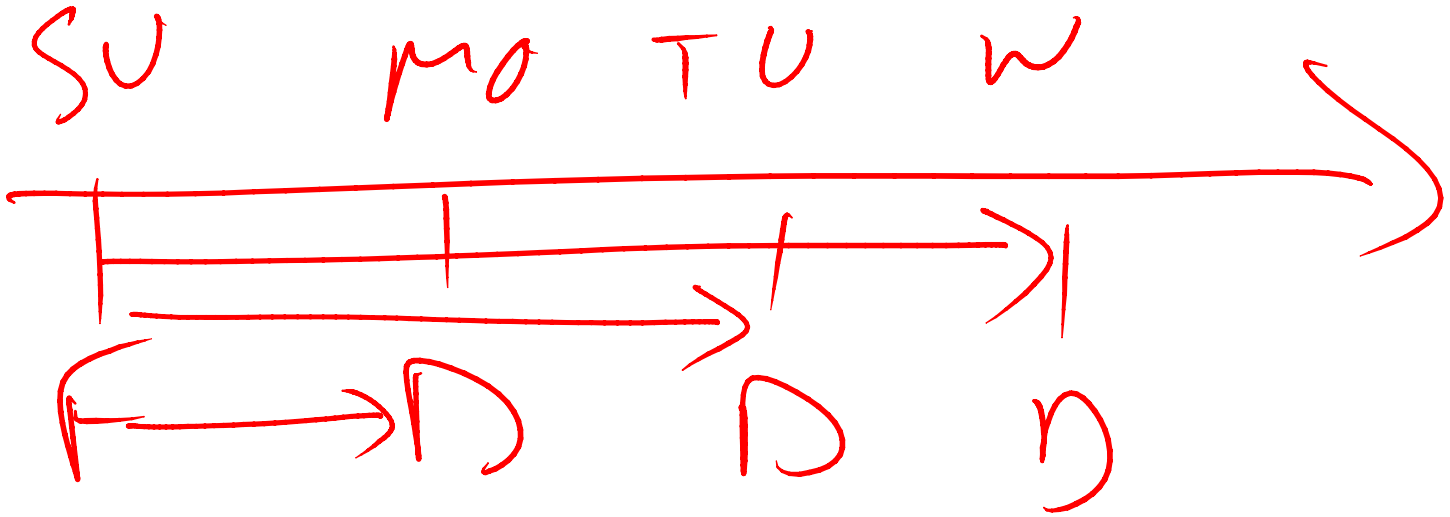
M1S15 When a record is deleted, it's marked as a ghost record. The ghost cleanup task does not physically delete the records – it removes the slot array entry that points to the record on the page. I.e. the record is unmarked as being a record and the area on the page is now free space – that just happens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



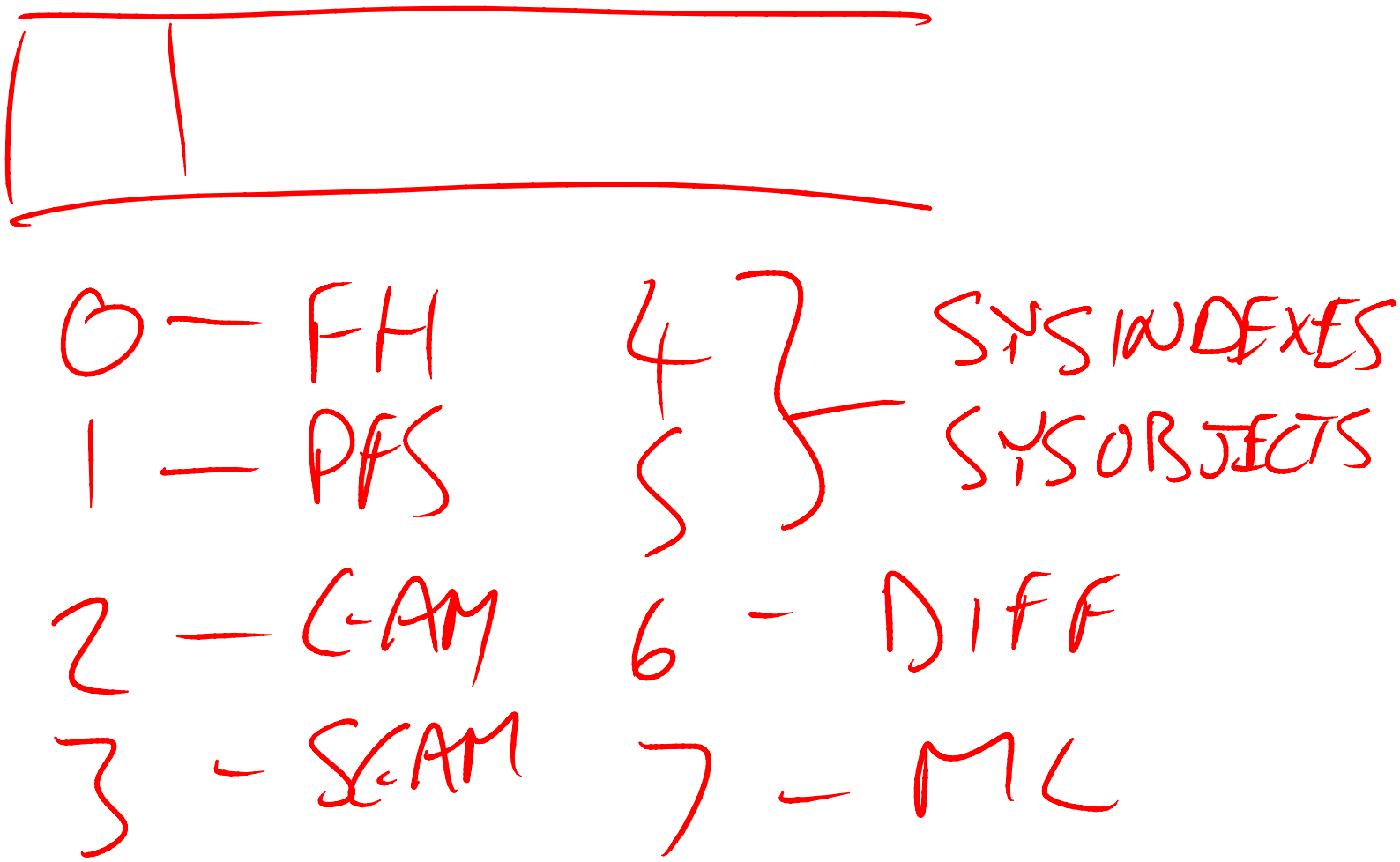
0x00FA → Reg col

0x80FA

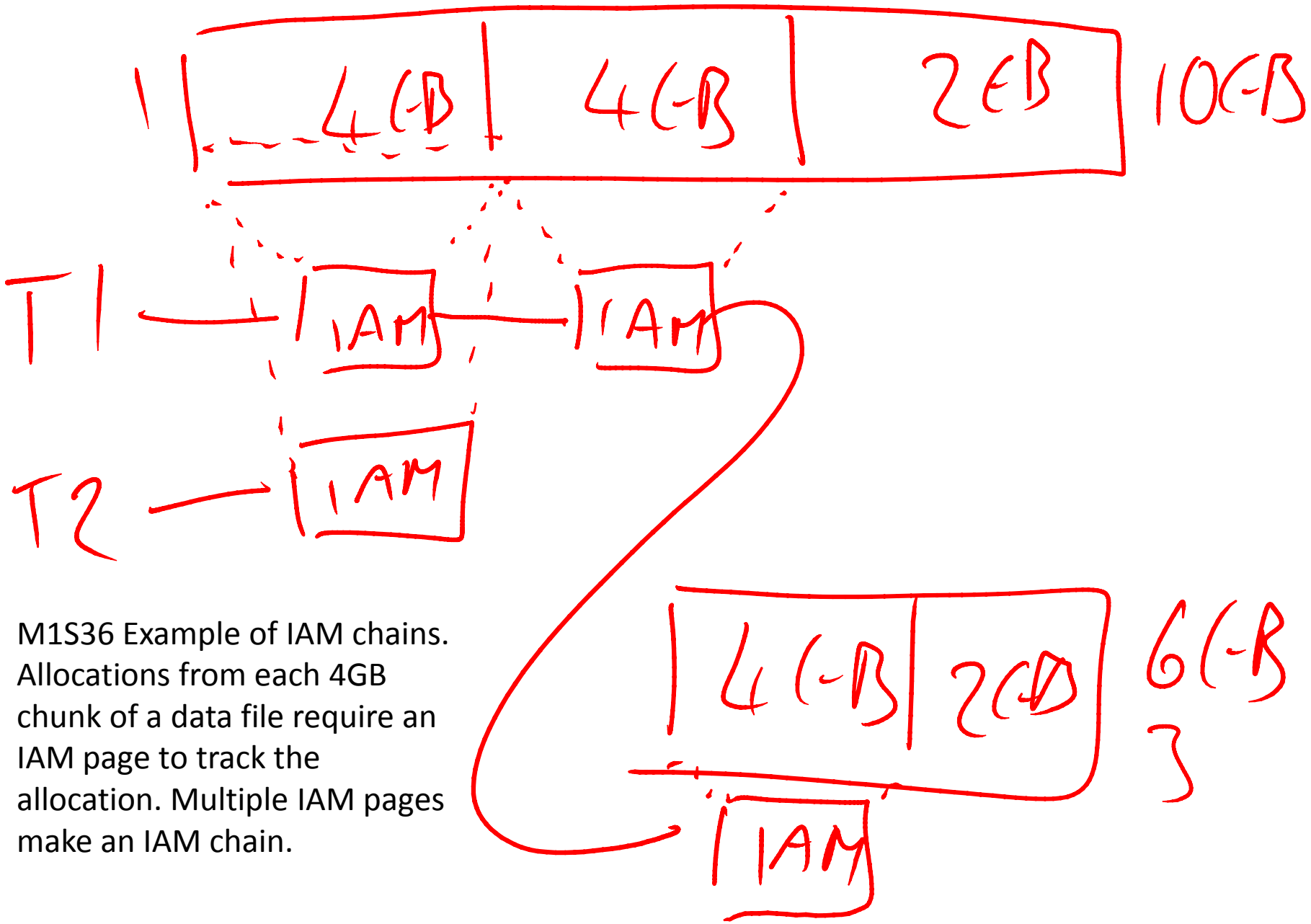
M1S22 In the variable-length column offset array, if the 0x8000 bit is set in the two-byte length, the column is a complex column that requires further cracking open.



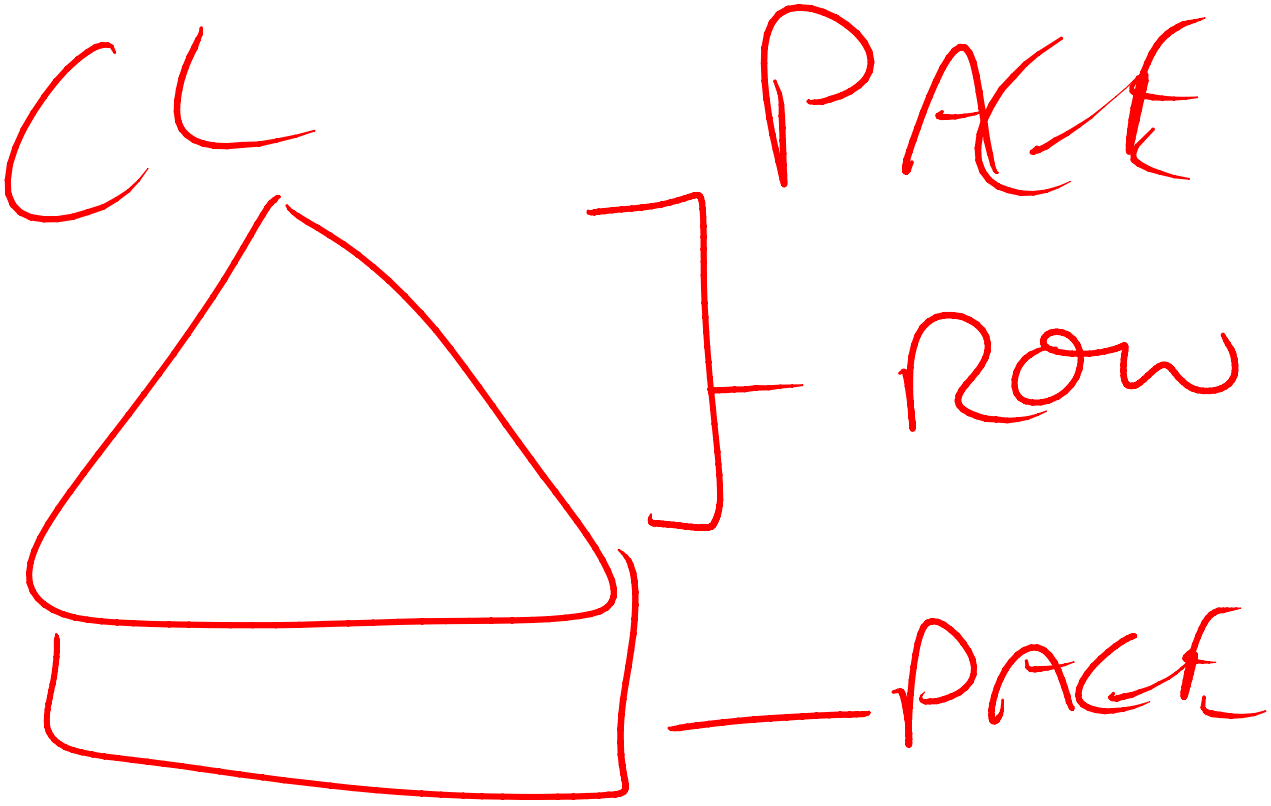
M1S32 Differential backups are cumulative. Successive differential backups after a full backup will get larger as more of the database changes.



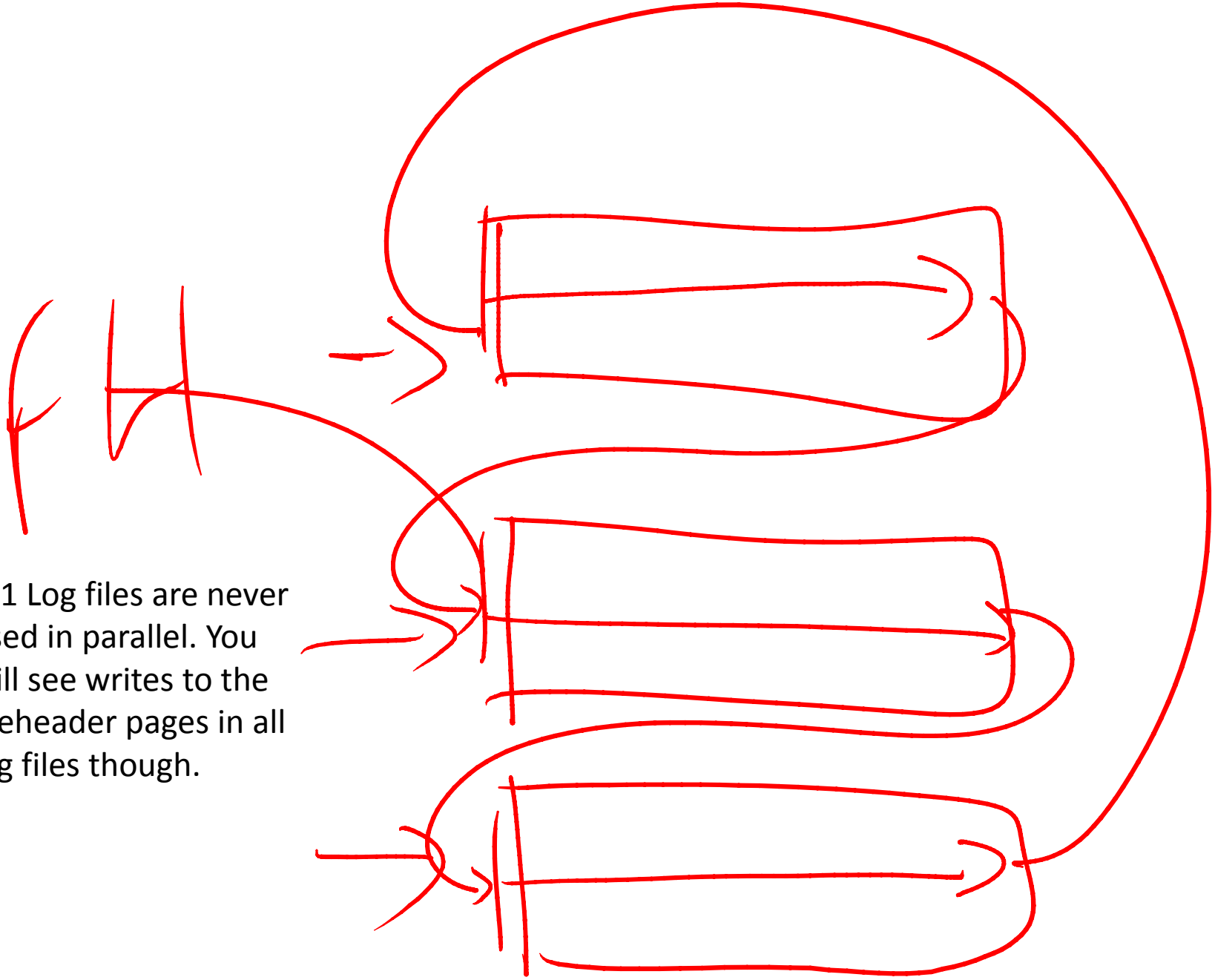
M1S32 Allocation bitmaps in the first extent of each data file. This extent repeats (except the file header page) every 511232 pages.



M1S36 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.



M1S61 If you enable page compression on an index, the non-leaf levels will only be row compressed. This is to save CPU cycles from having to page-decompress the non-leaf levels during index traversals from the root to the leaf.



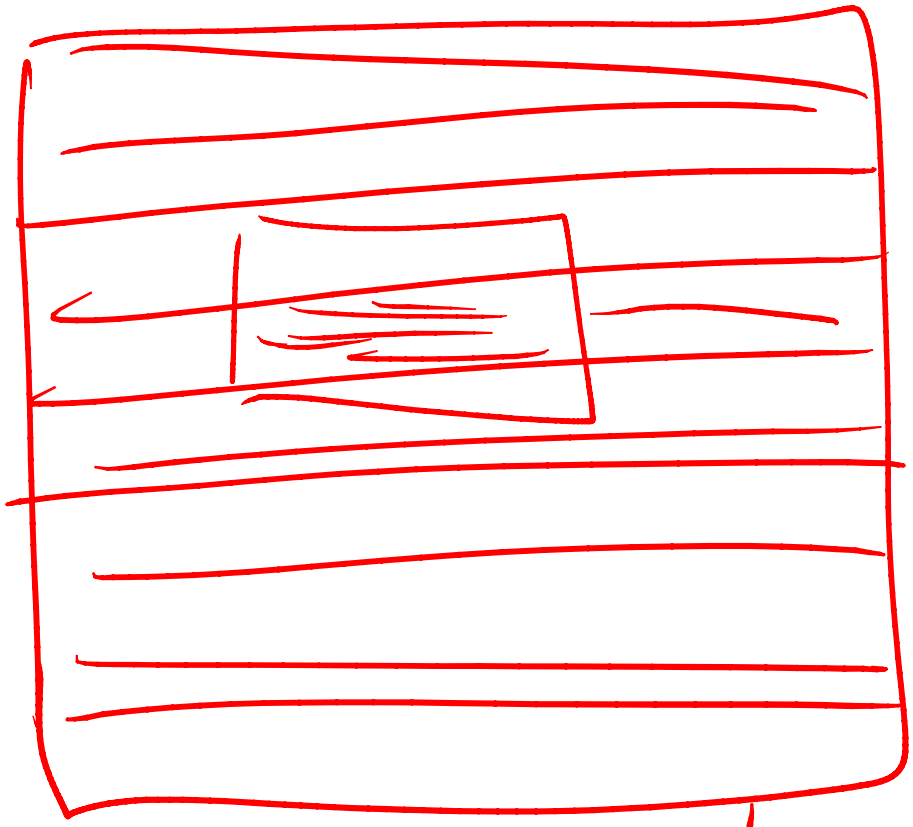
M1 Log files are never used in parallel. You will see writes to the fileheader pages in all log files though.

JONATHAN KEHAYIAS

M2: You can store SQL files on SMB shares now in 2012. Prior to that, use TF1807.

2008 R2 NETWORK SHARE KB 304261

TF 1807

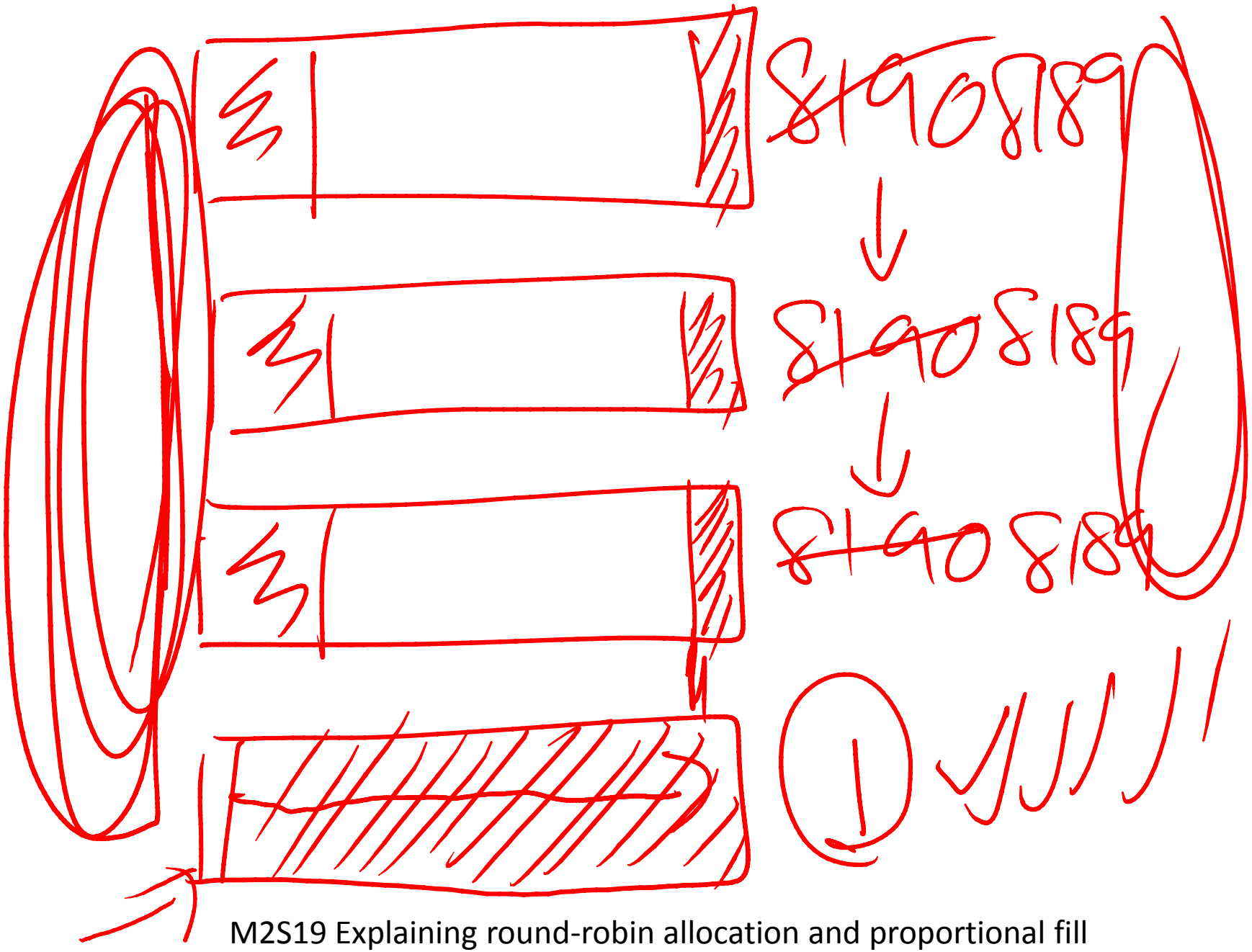


M2S17 Explaining how writes updating a file (small rectangle) on an SSD will actually be spread across SSD surface to amortize problem of SLC memory cells wearing out.

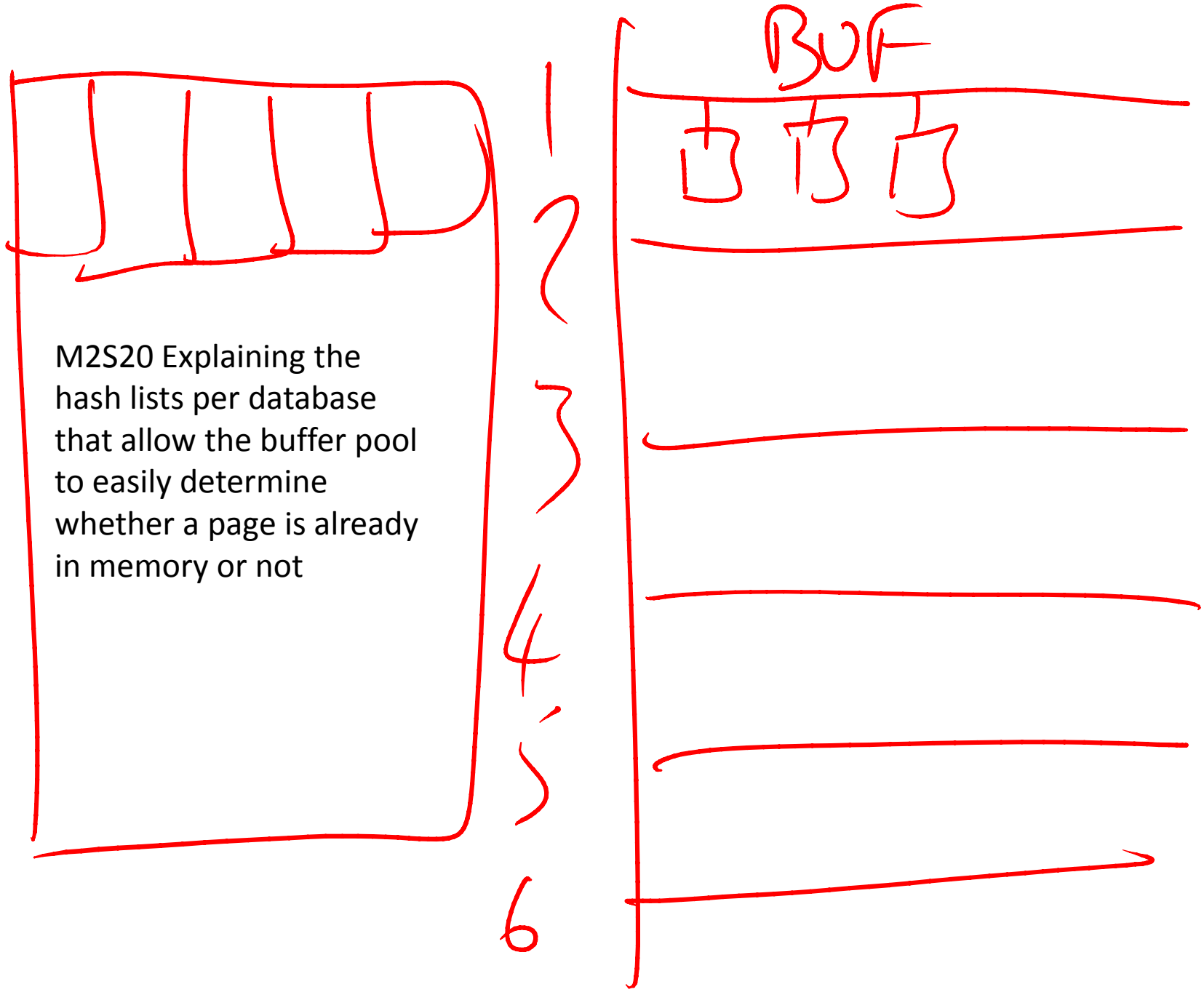
Monitor Physical Disk/Avg. Disk Secs/read and /Write to see disk latencies. Correlate with the virtual file stats DMV.

Phys Disk/Avg Disk Secs/Read /Write

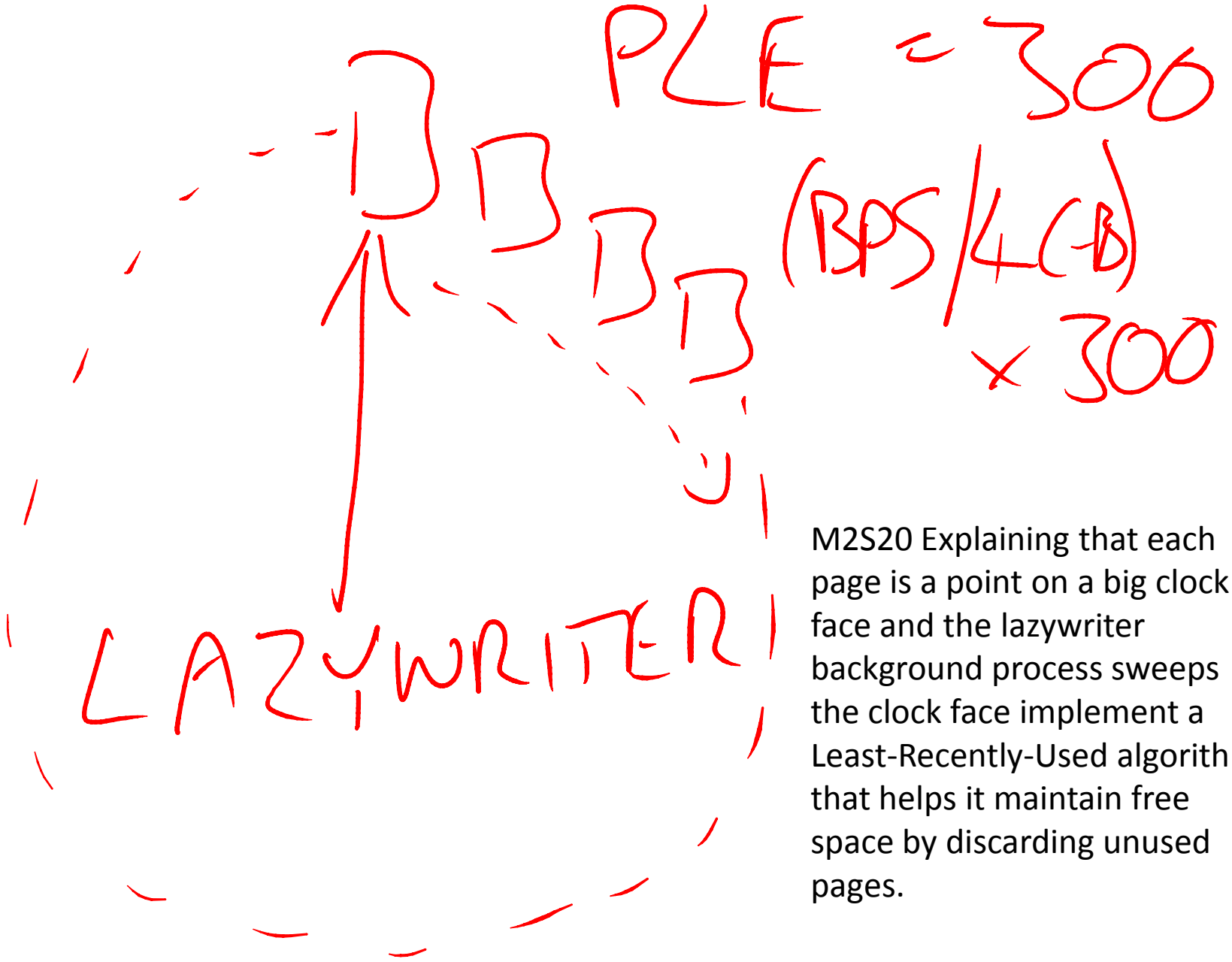
sys.dm_io_virtual_file_stats



M2S19 Explaining round-robin allocation and proportional fill

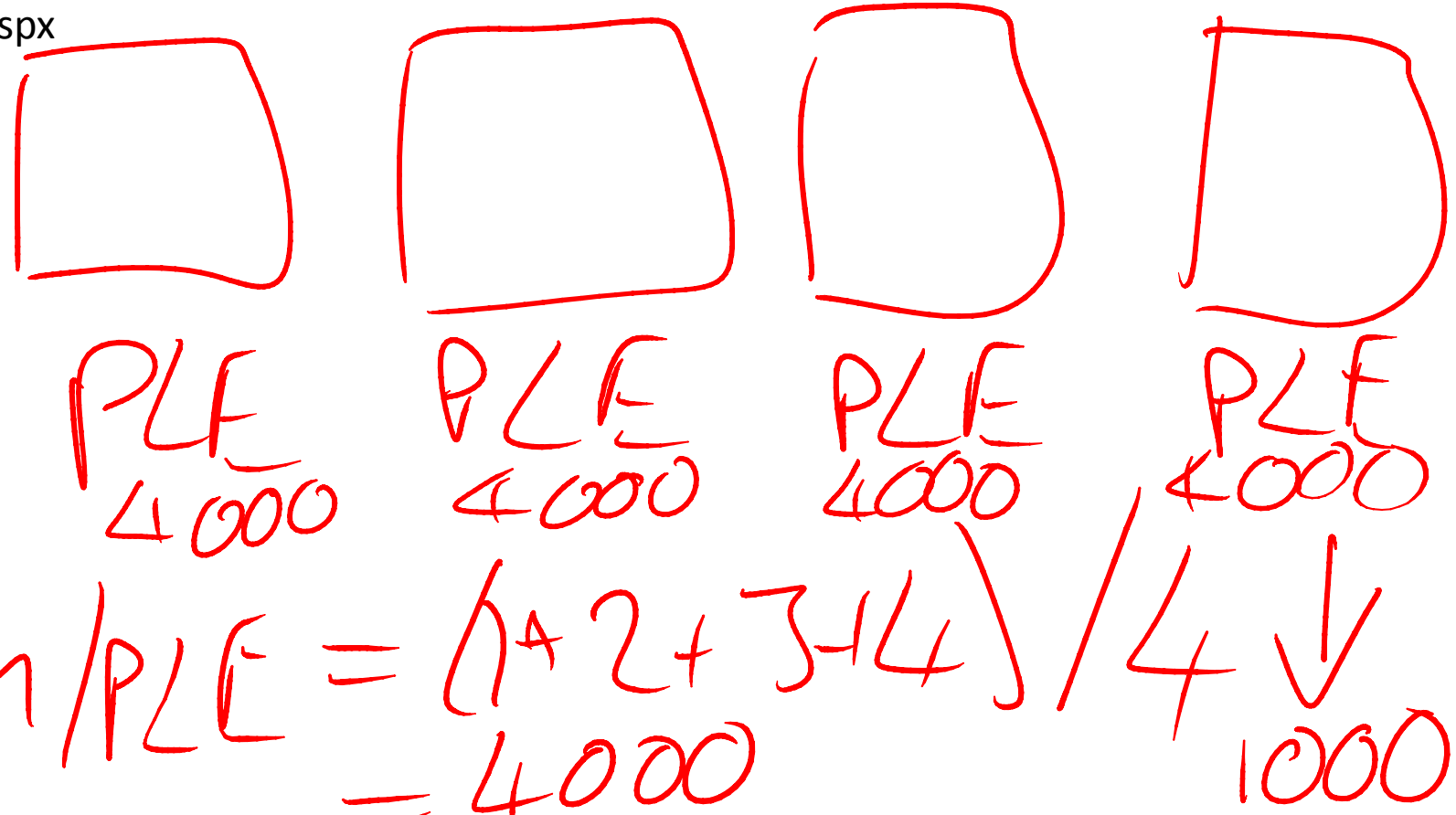


M2S20 Explaining the hash lists per database that allow the buffer pool to easily determine whether a page is already in memory or not



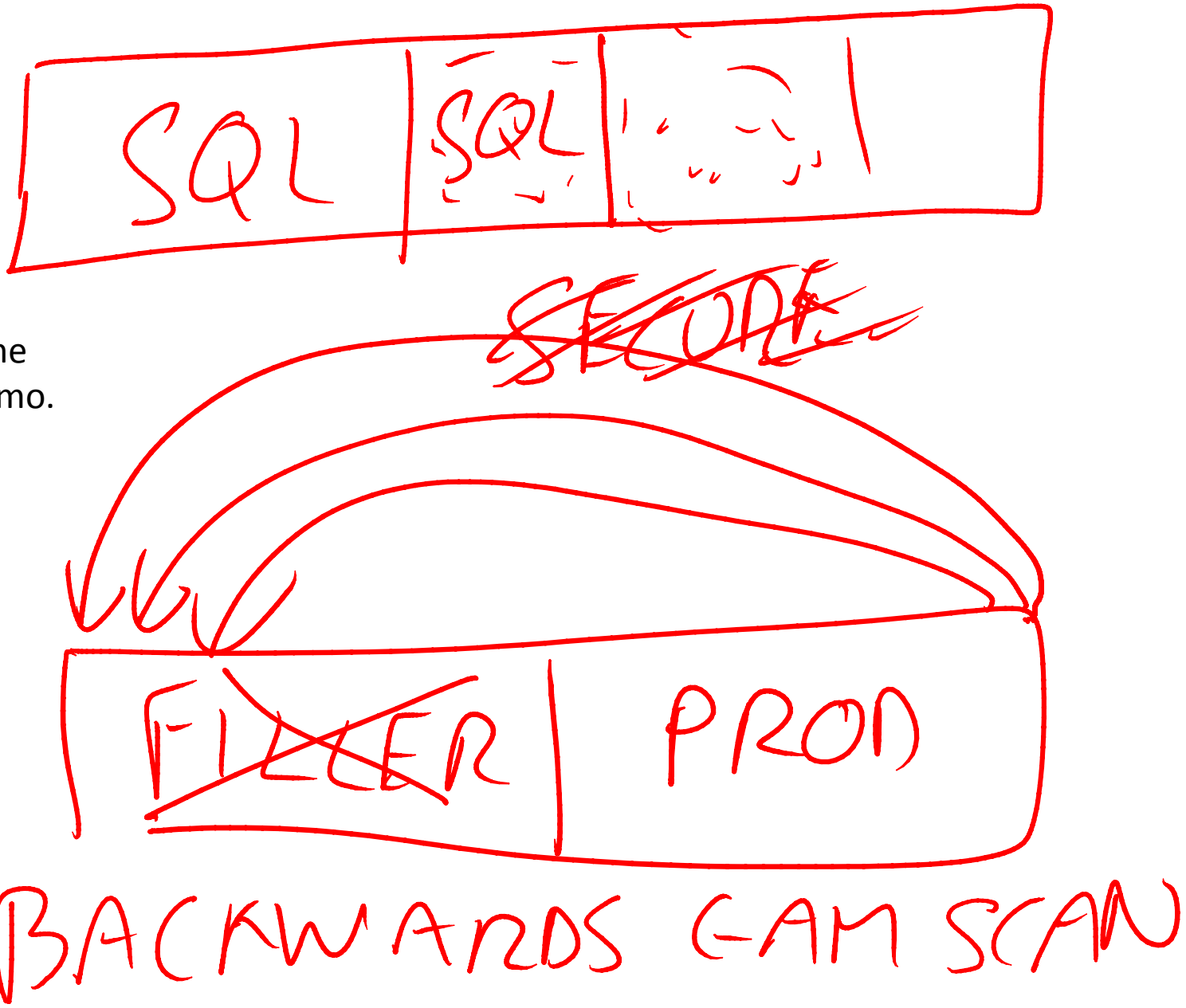
M2S20 Explaining that each page is a point on a big clock face and the lazywriter background process sweeps the clock face implement a Least-Recently-Used algorithm that helps it maintain free space by discarding unused pages.

See http://www.sqlskills.com/BLOGS/PAUL/post/Page-Life-Expectancy-isnt-what-you-think.aspx



This should be Buffer Node instead of Buffer Partition – my mistake.

M2S28 The shrink demo.

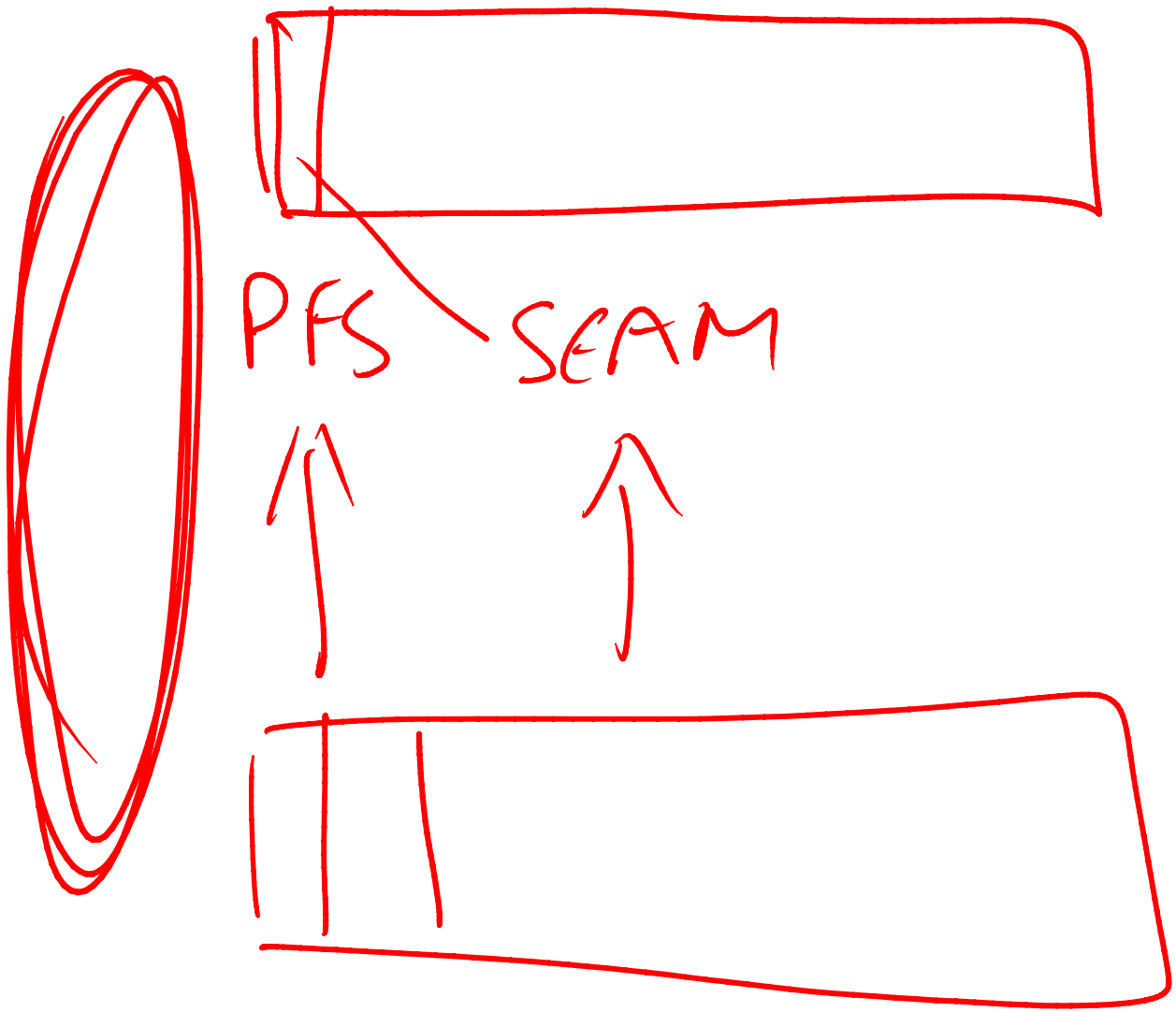


sys.dm_db_task_space_usage

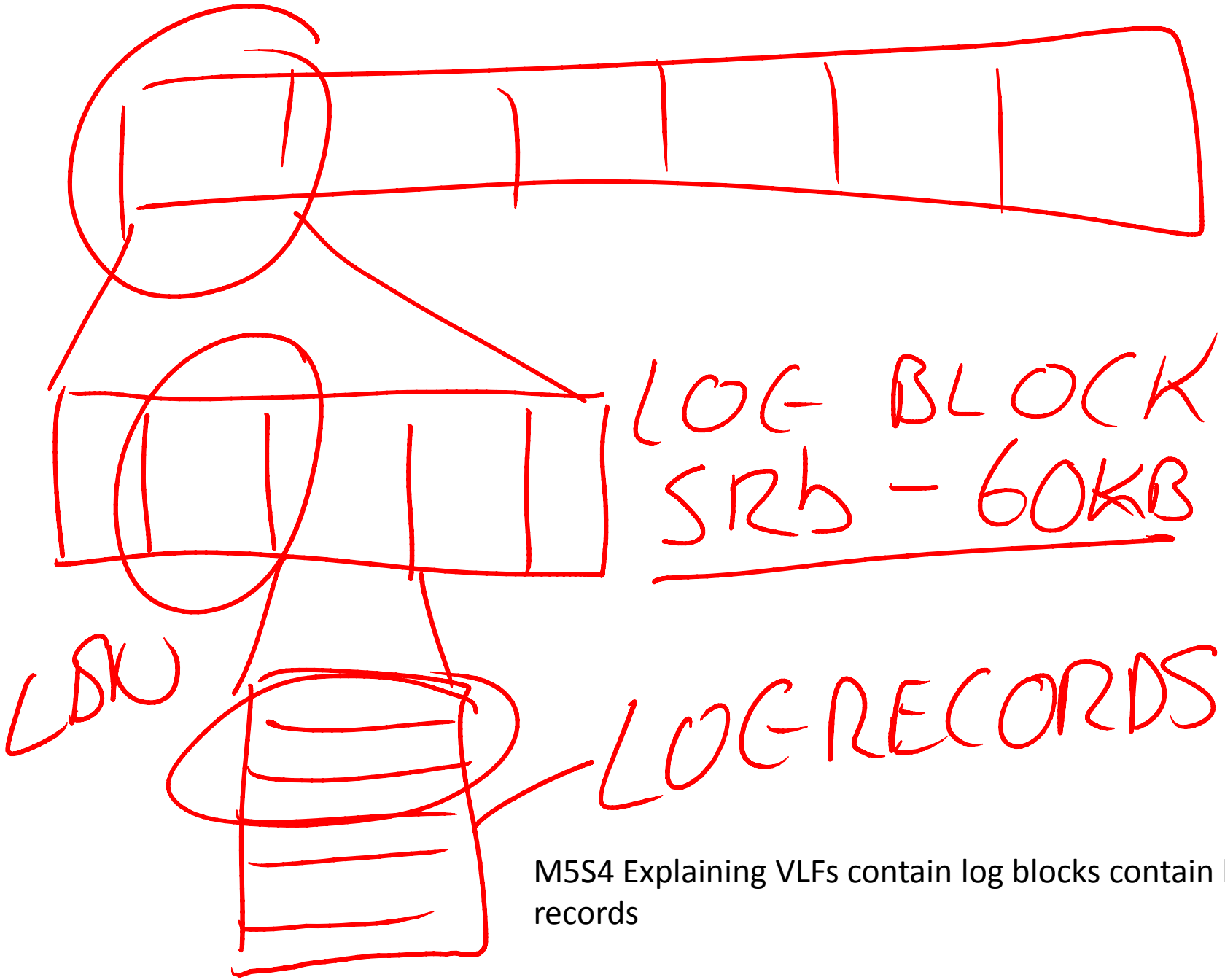
sp_whoisactive

Adam Machanic

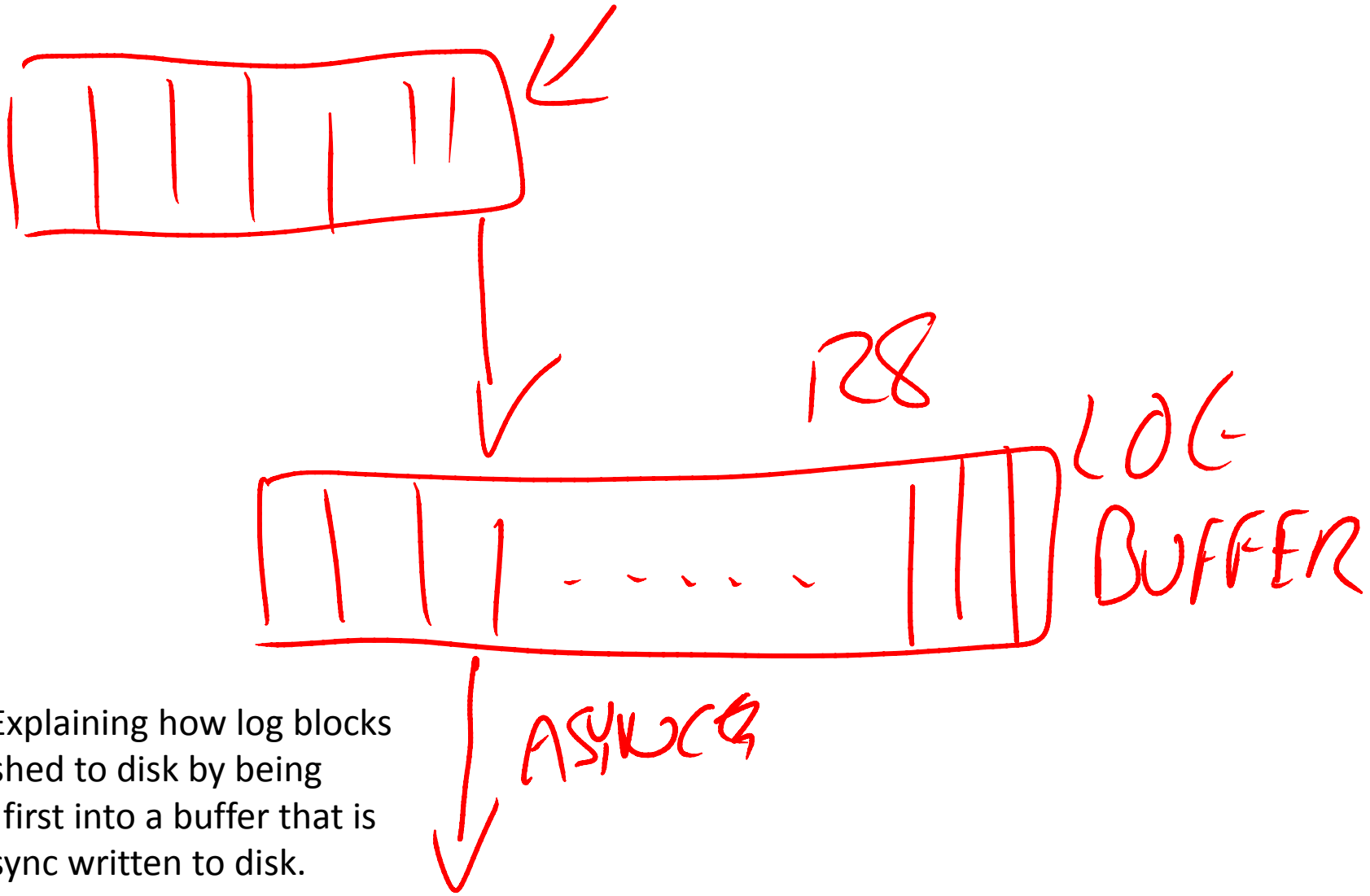
M2: Monitoring tempdb space issues.



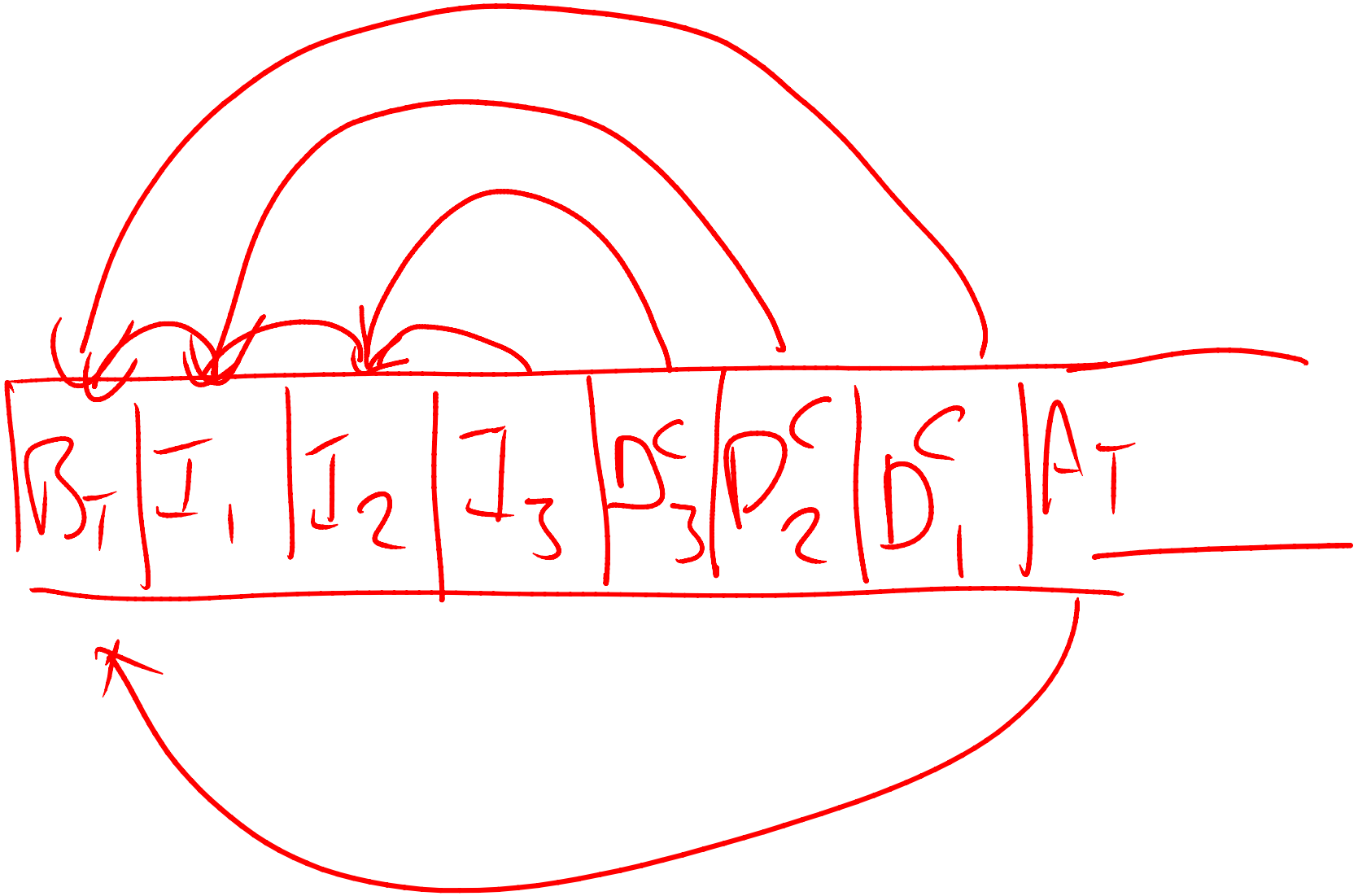
M2S40 Explaining PFS and SGAM contention in tempdb and how adding multiple files spreads the contention over multiple files, thus reducing it.



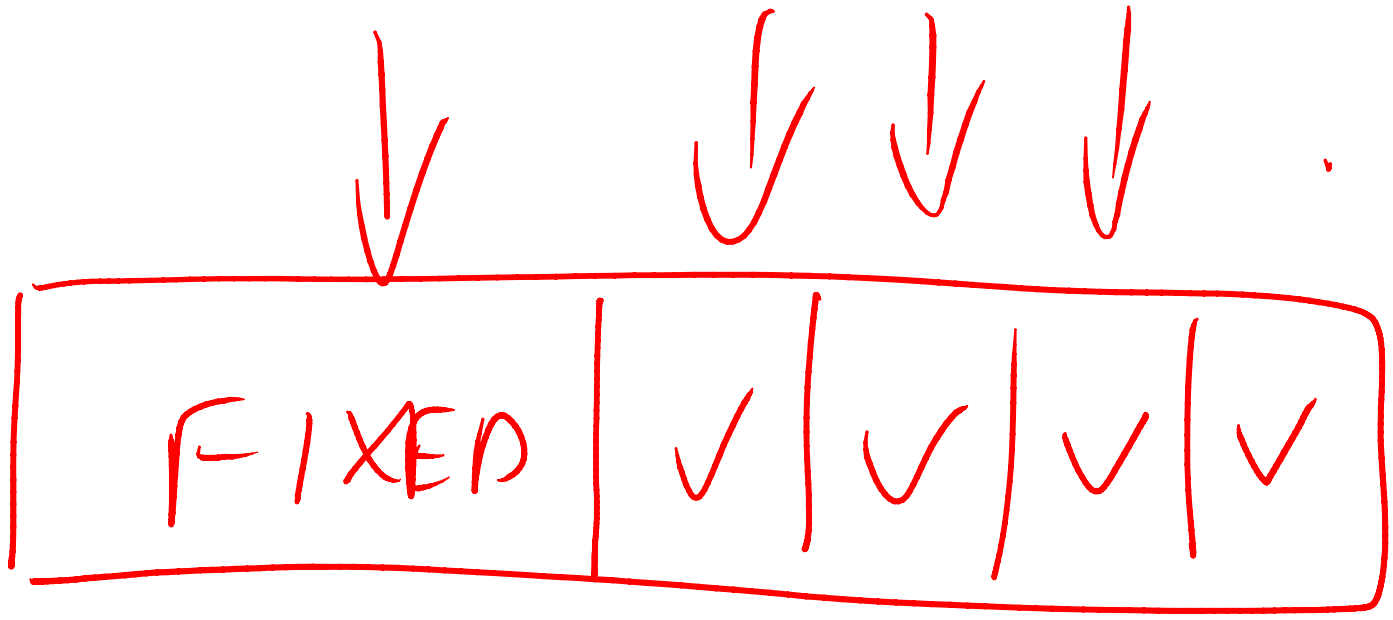
M5S4 Explaining VLFs contain log blocks contain log records



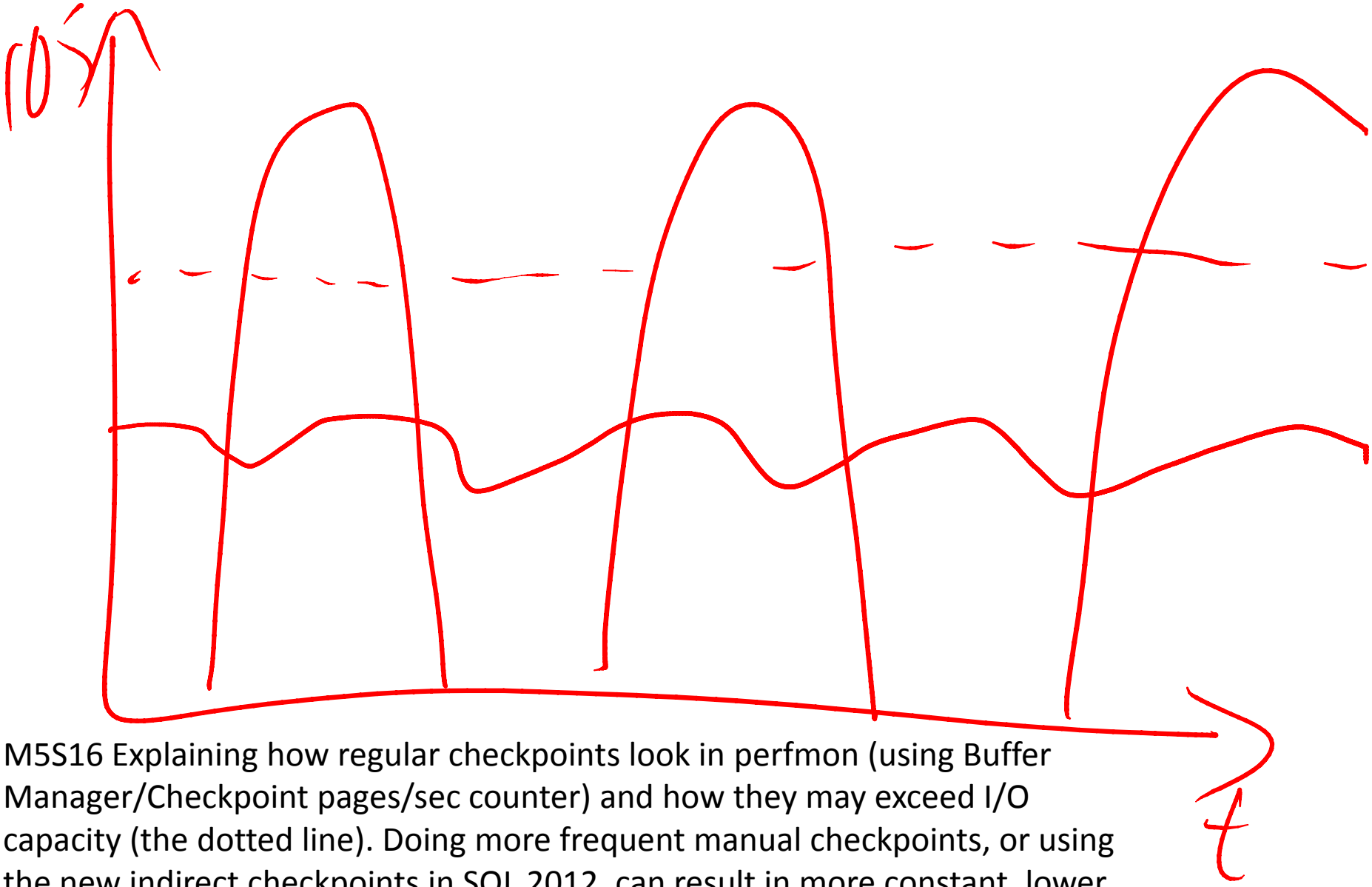
M5S4 Explaining how log blocks are flushed to disk by being copied first into a buffer that is then async written to disk.



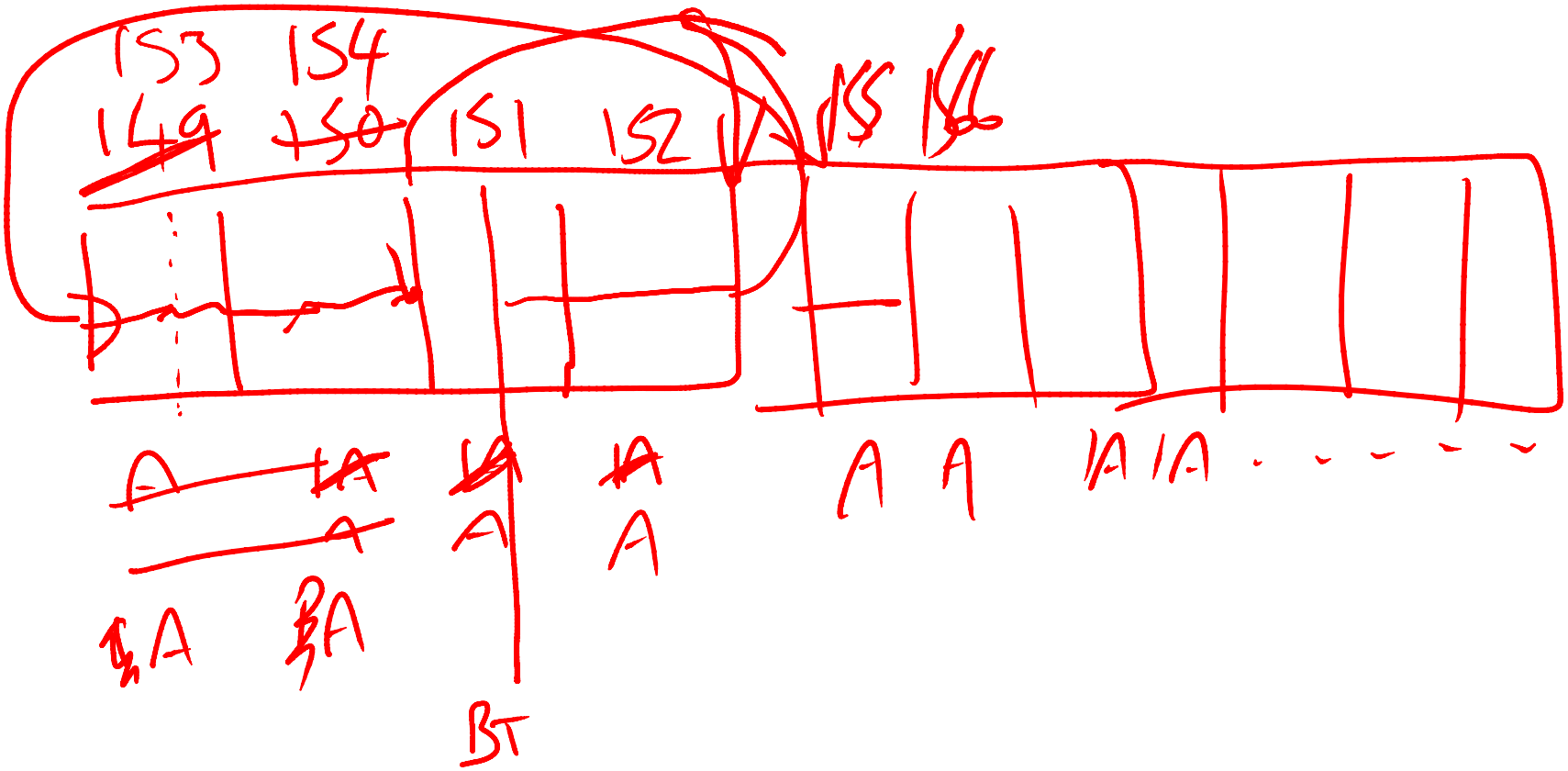
M5S11 Explaining how log records link back to the previous log record in the transaction to allow the transaction to be rolled back by generating compensating actions in reverse order. A compensation log record will link over the log record it is compensating for.



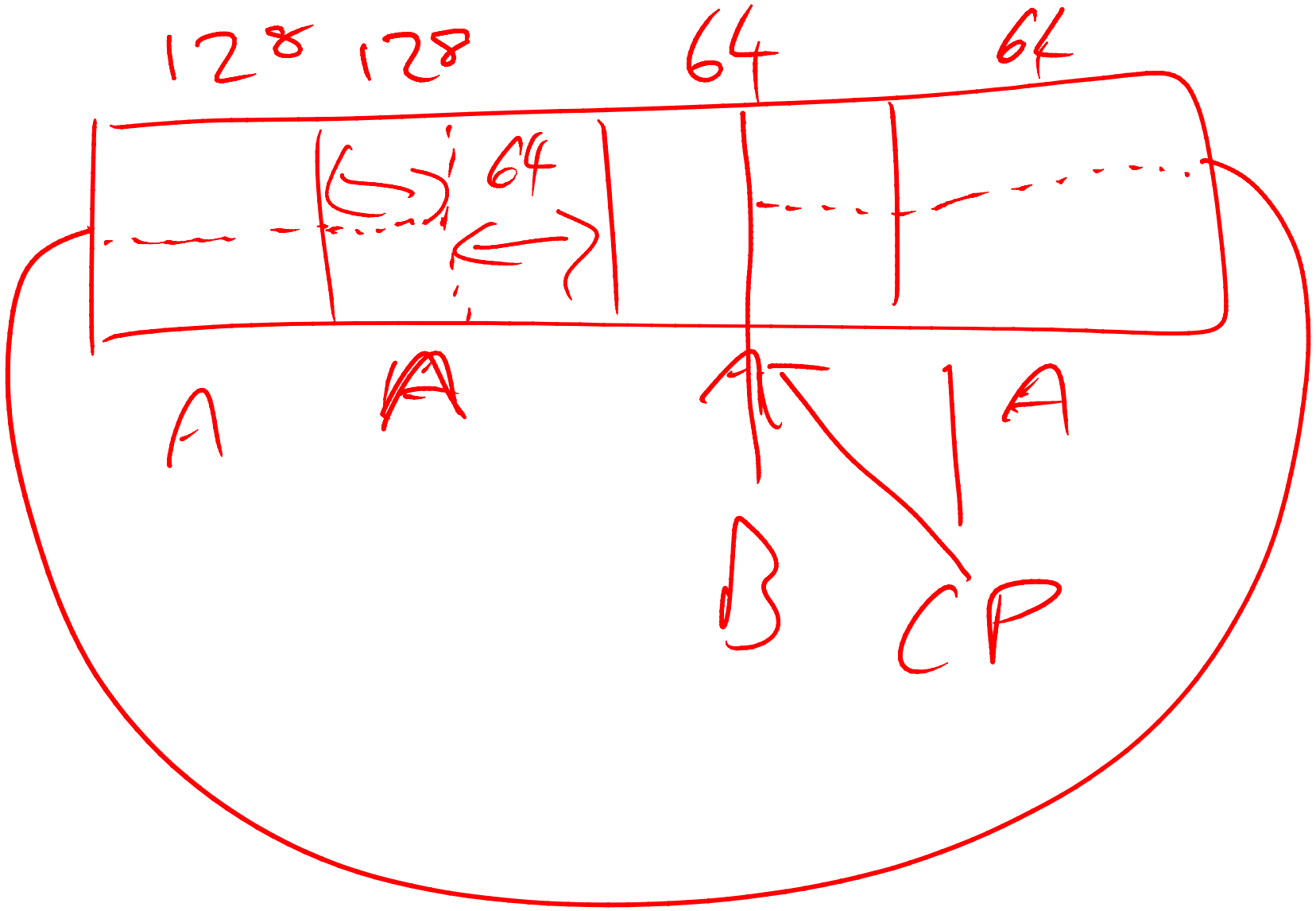
M5S11 Explaining how the Access Methods determines how to log UPDATEs efficiently by splitting the logging by portion of the record. Each variable length column is a portion, as is the fixed-width portion (up to 16 bytes in each log record)



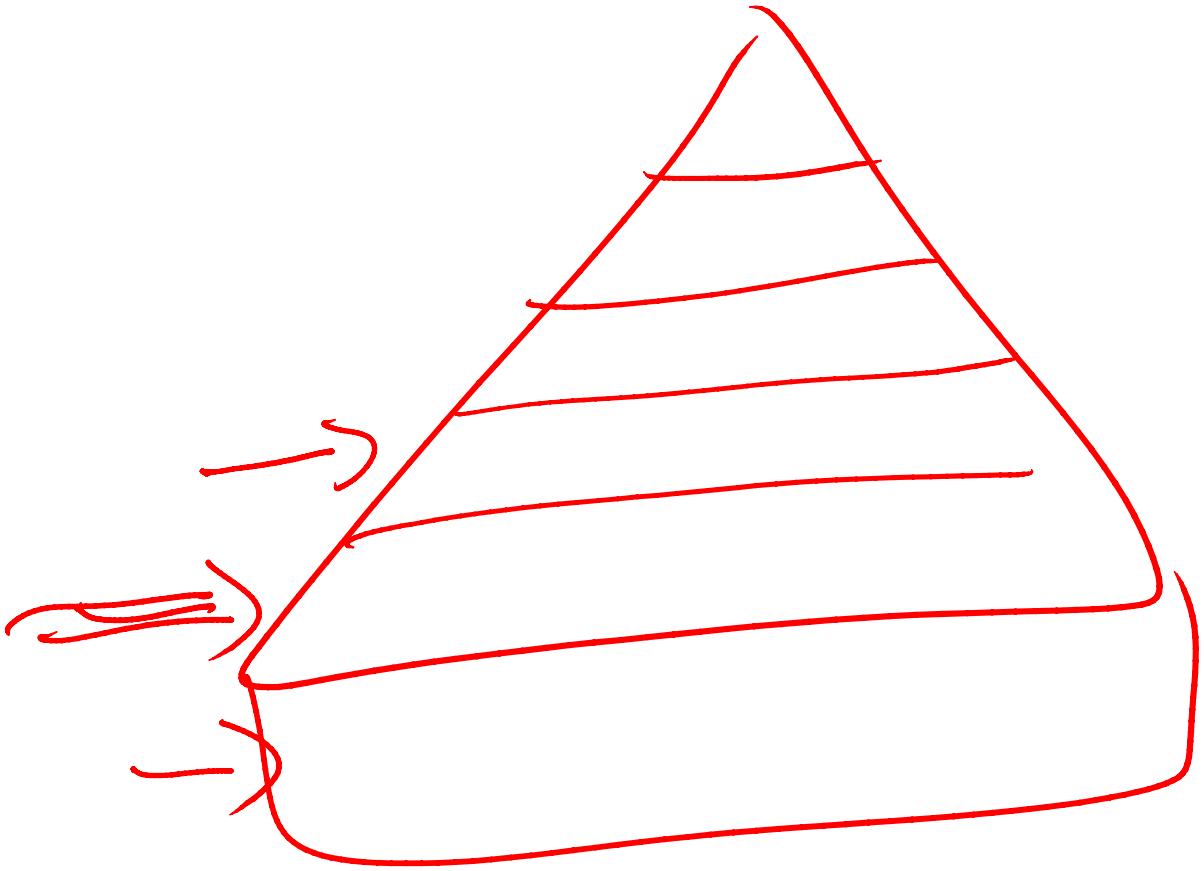
M5S16 Explaining how regular checkpoints look in perfmon (using Buffer Manager/Checkpoint pages/sec counter) and how they may exceed I/O capacity (the dotted line). Doing more frequent manual checkpoints, or using the new indirect checkpoints in SQL 2012, can result in more constant, lower IO load.



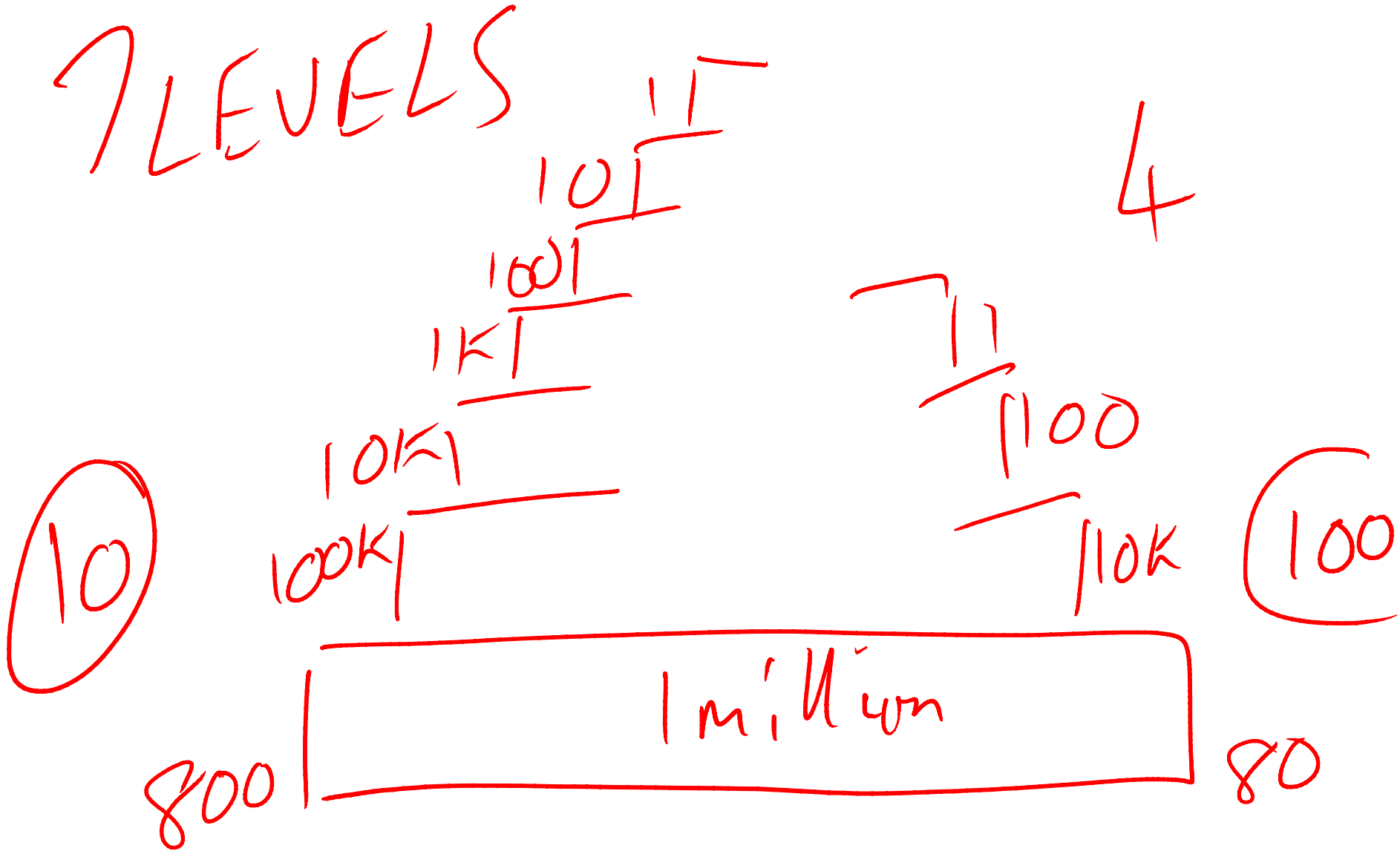
M5S28 Explaining the circular log demo. We discussed log space reservation that happens to ensure the active transactions can roll back without having to grow the log – this accounts for extra log growths. Space reserved in the log does not make a VLF become active – as there are no log records written out, just empty space reserved.



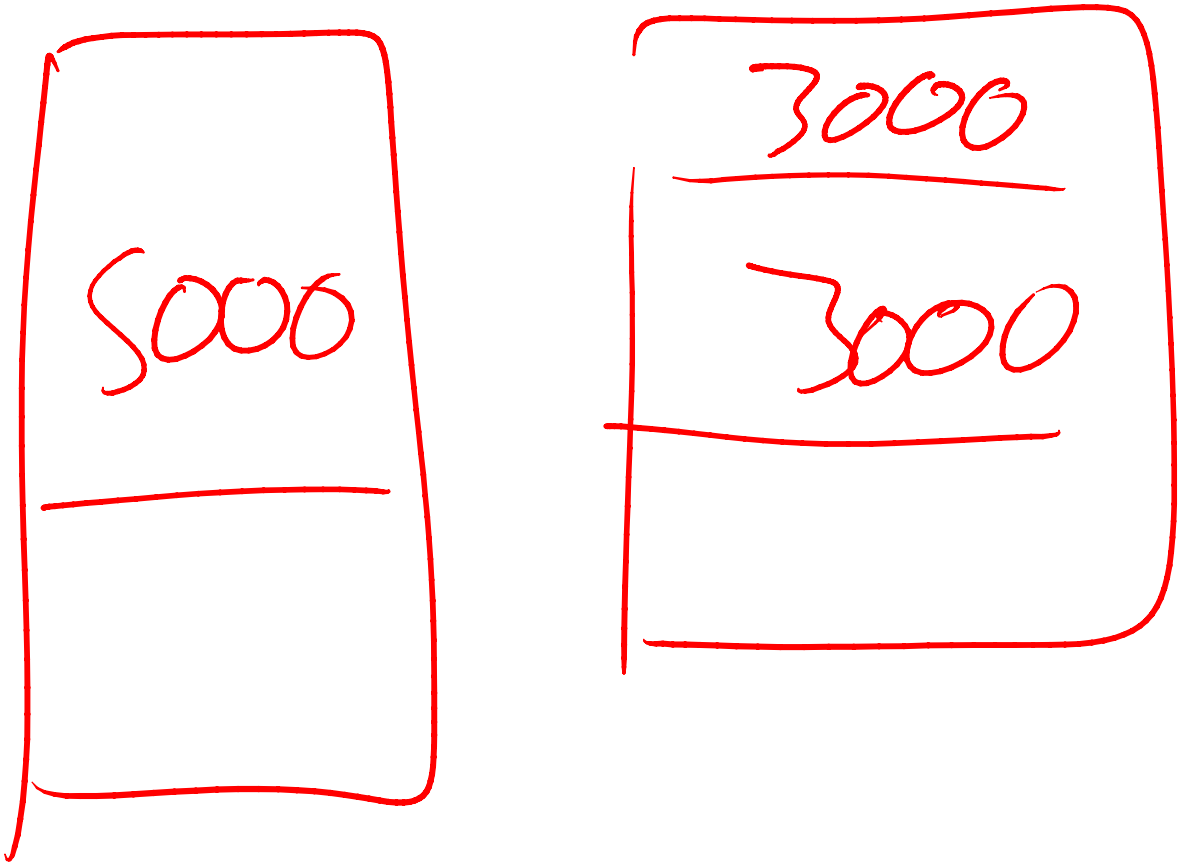
M5S29 Explaining the use of parity bits to help SQL Server determine where the end of the log is during crash recovery.



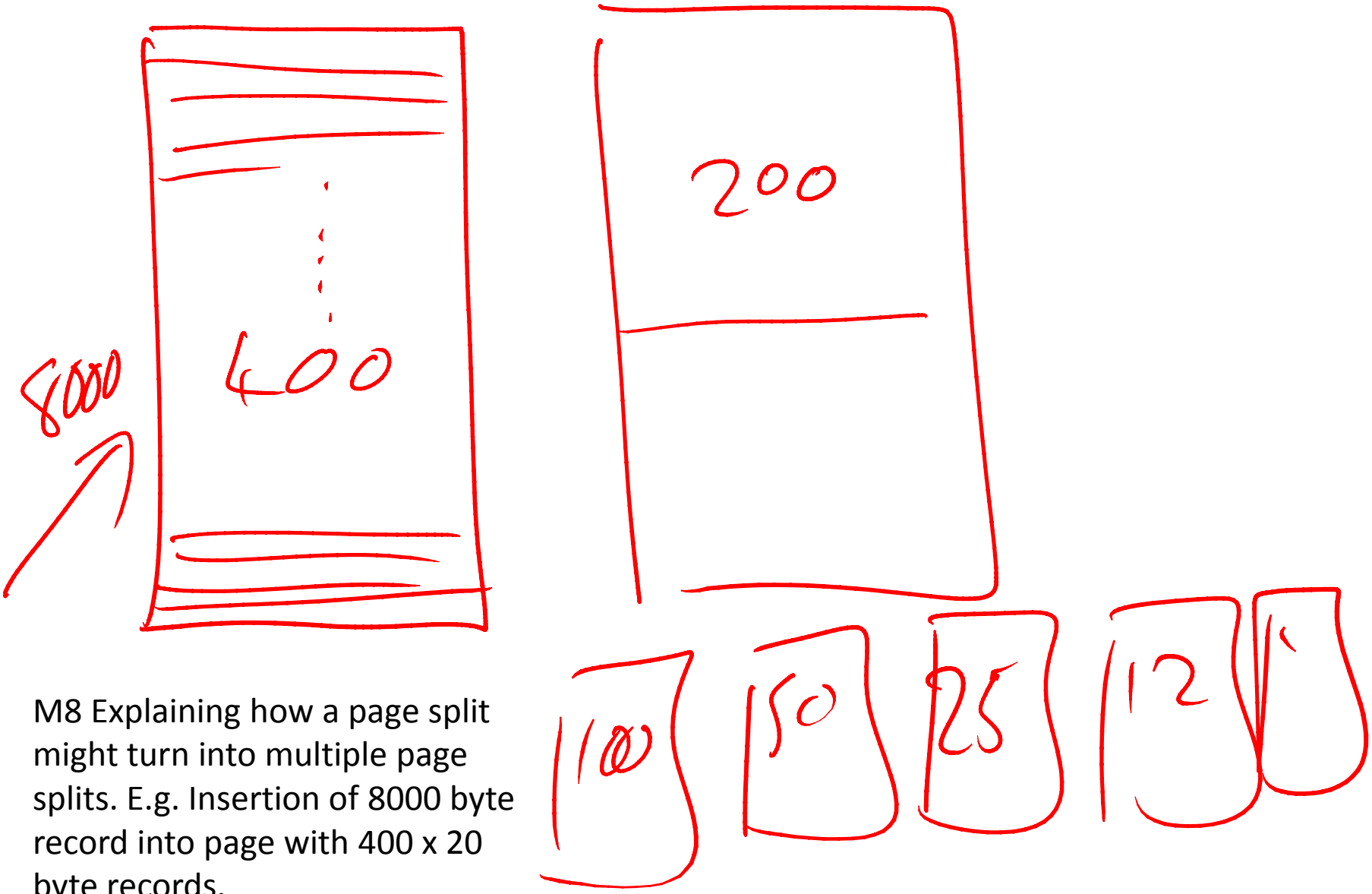
M8S10 Explaining that sometimes there will be readahead using level 2 to help drive the scan on level 1 that’s doing readahead on level 0.



M8 Explaining ‘fanout’. Larger index key size means less rows per page on each index level, meaning more index levels and more CPU for index traversal from root to leaf.

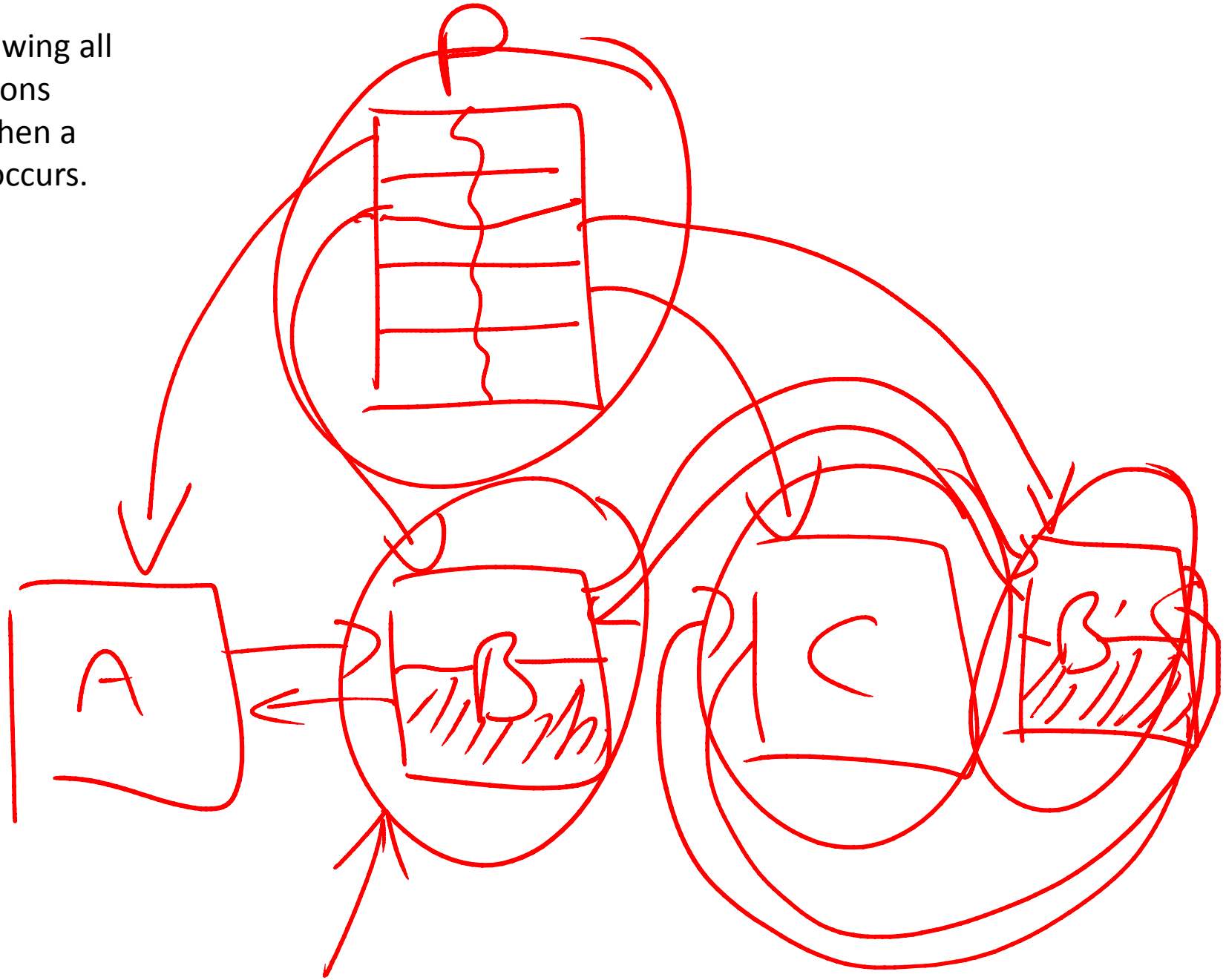


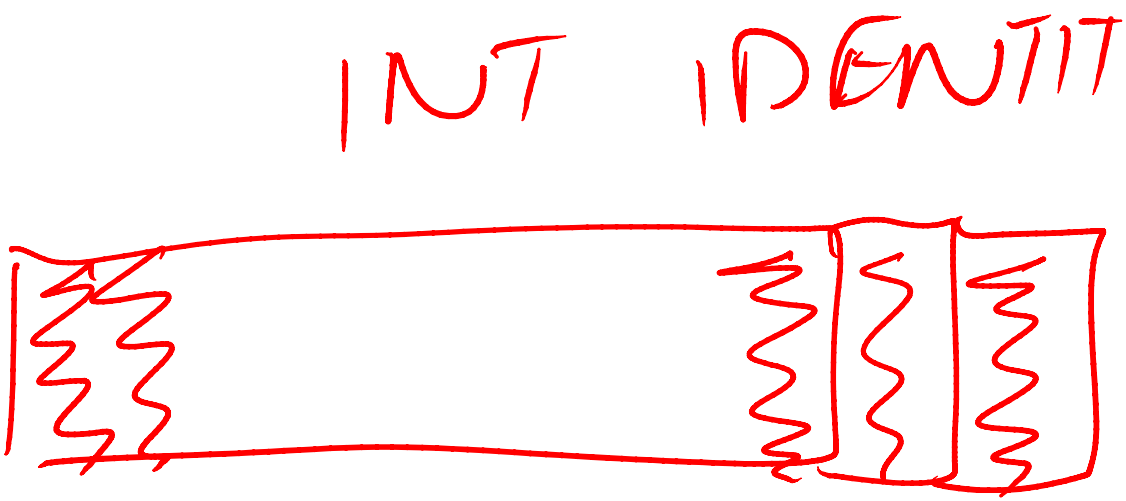
M8S13 Explaining how fixed-size large rows leads to low page density and wasted space.



M8 Explaining how a page split might turn into multiple page splits. E.g. Insertion of 8000 byte record into page with 400 x 20 byte records.

M8S15 Showing all the operations required when a page split occurs.

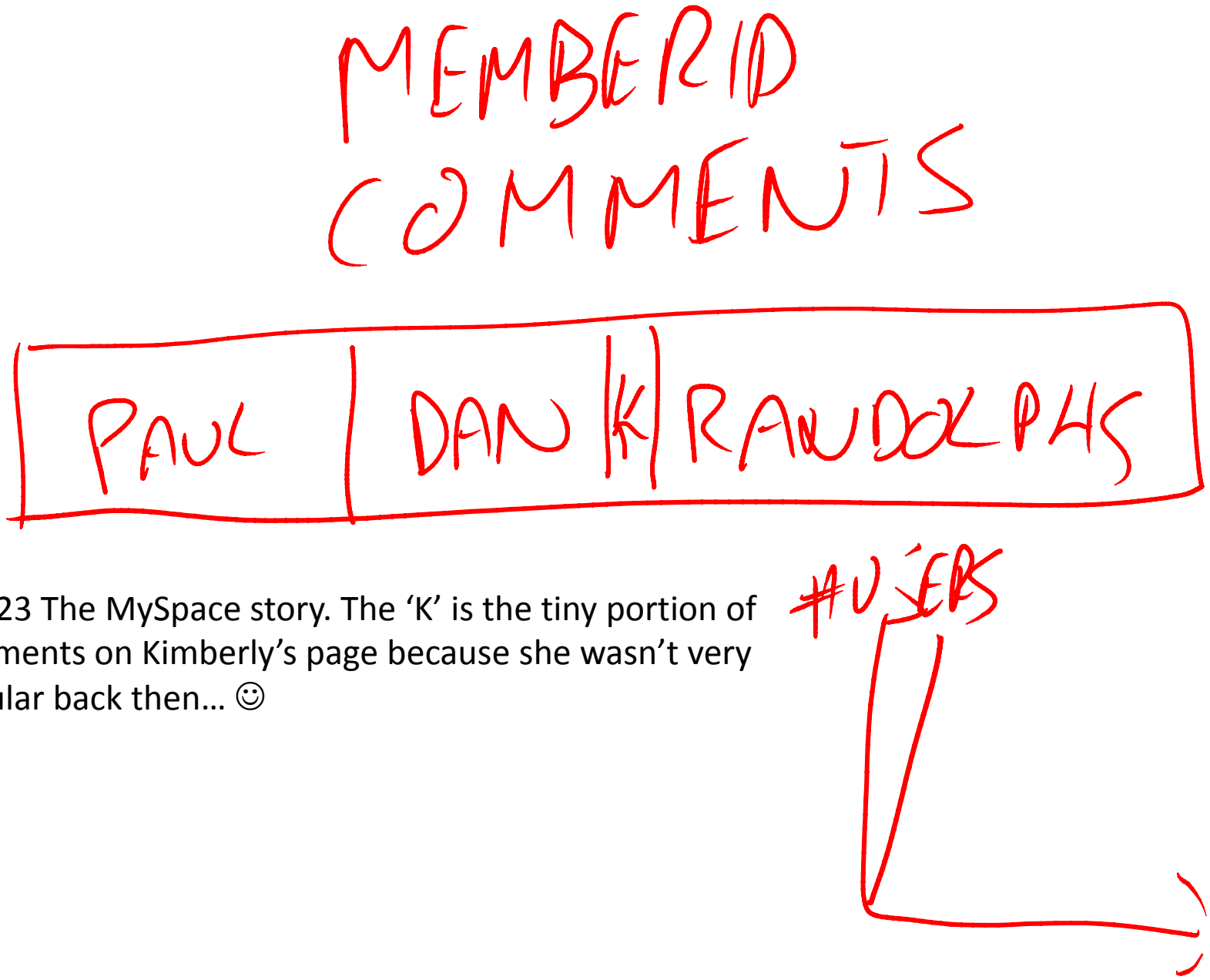




M8S18 Append-only insert pattern fills pages on right-hand side of index. New page allocations in this case are called page splits too – making all methods of tracking page splits largely useless.

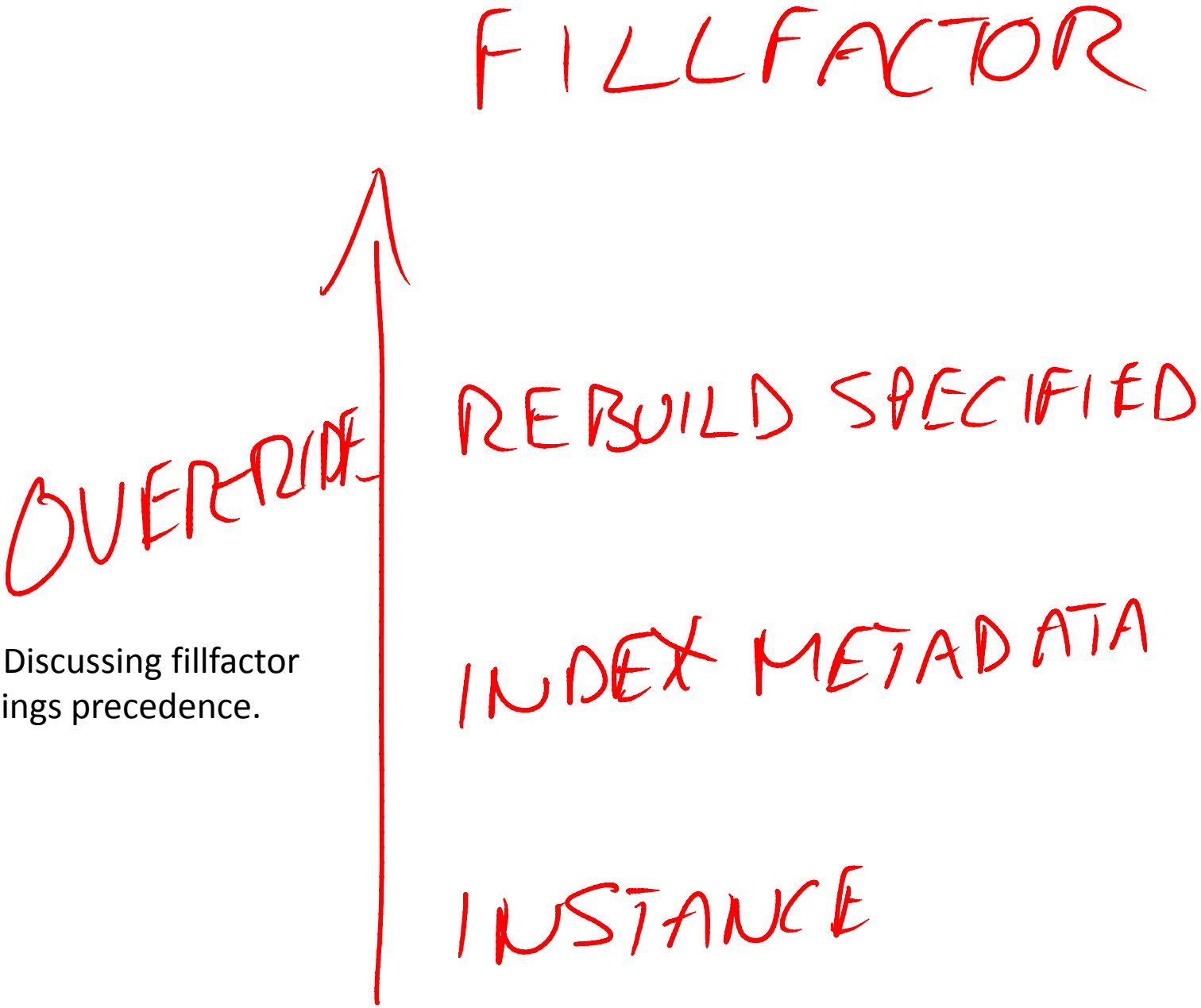
Row Size	No Split	Split	How Bad
1060	1244	6772	x5
100	344	5964	x18
10	256	10364	x45

M8S22 The numbers from the page split logging cost demo.

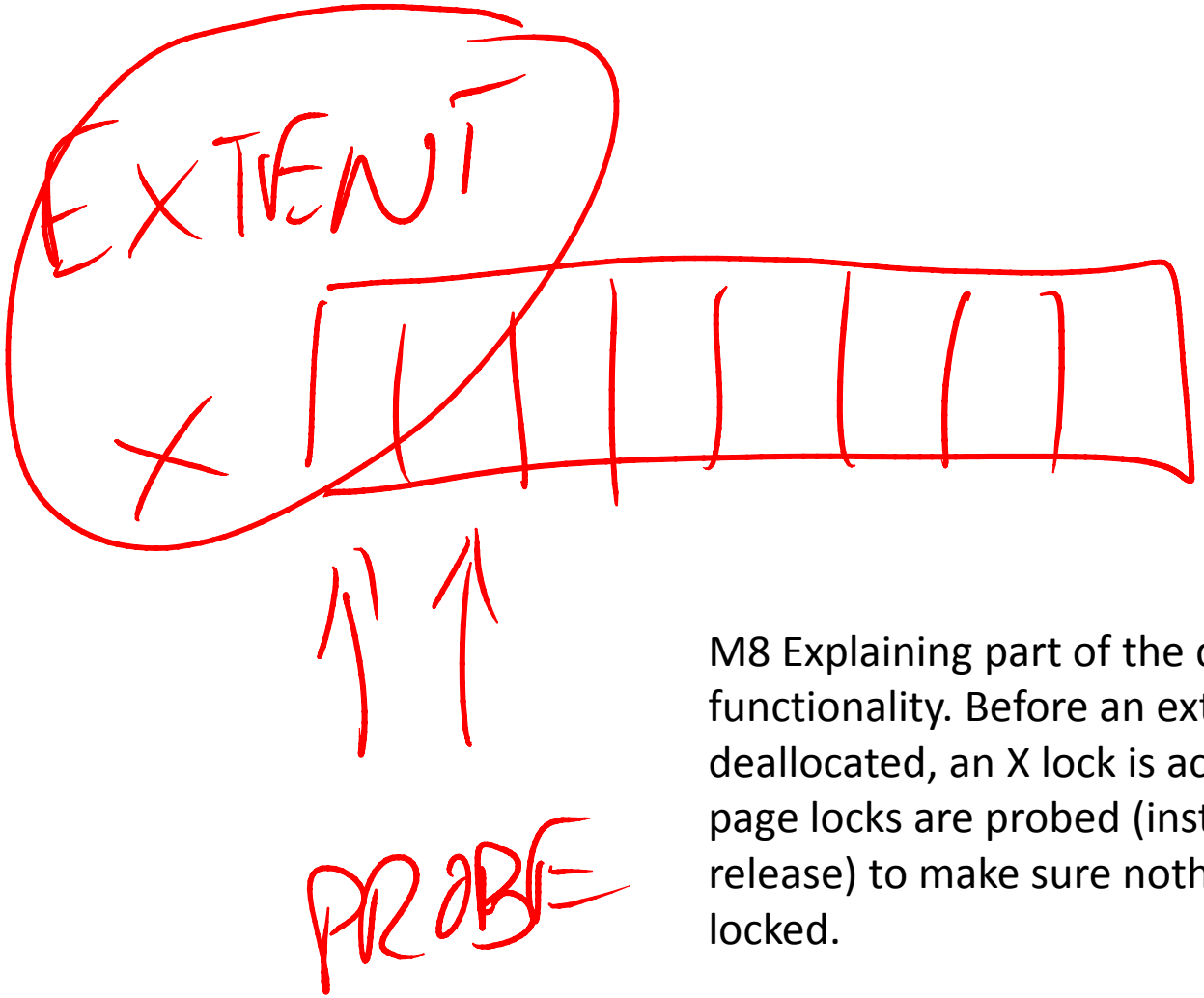


M8S23 The MySpace story. The ‘K’ is the tiny portion of comments on Kimberly’s page because she wasn’t very popular back then… ☺

M8S37 Explaining staggered index maintenance with database mirroring. See http://bit.ly/IS04W5 for more explanation



M8 Discussing fillfactor settings precedence.



M8 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.