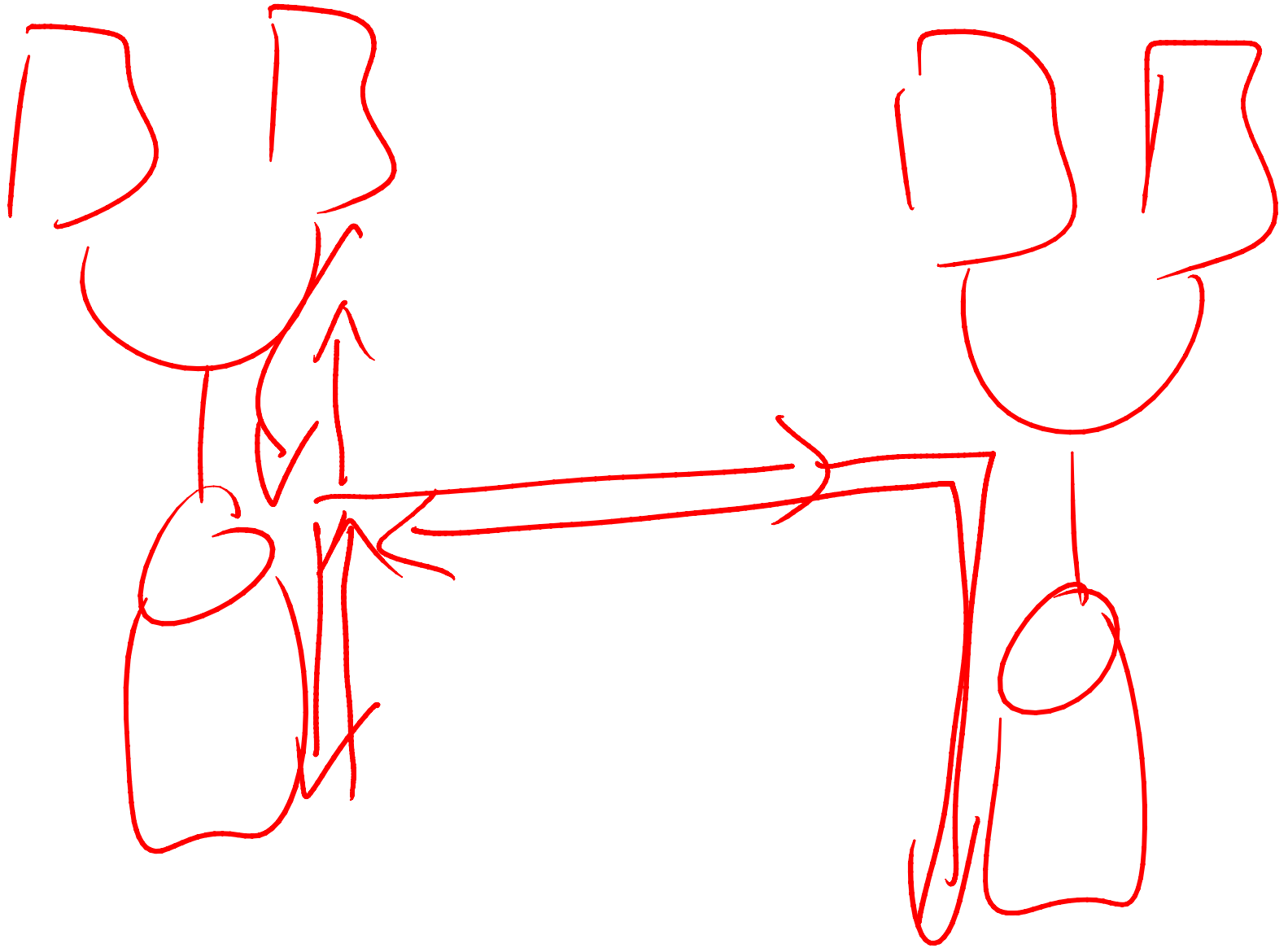
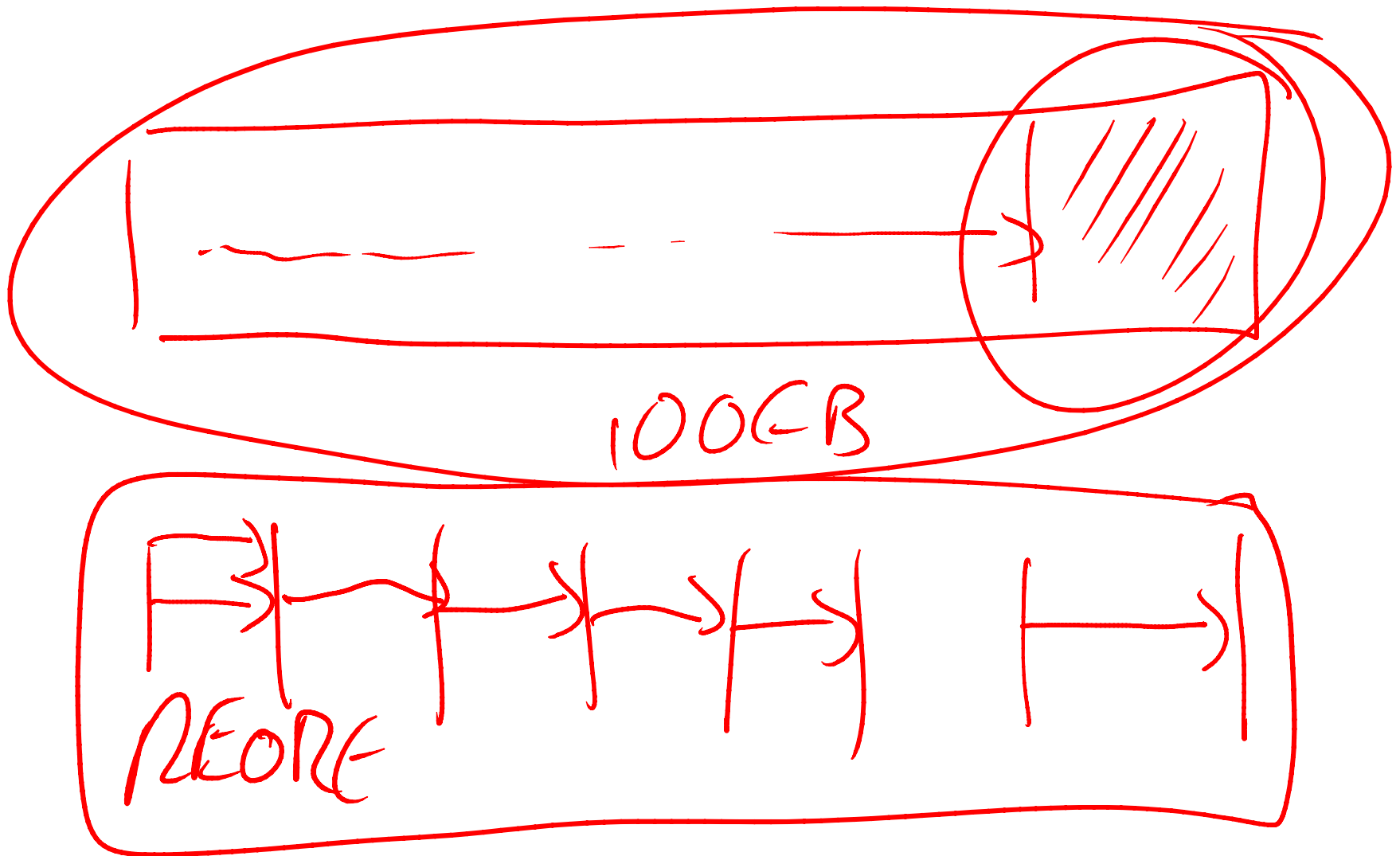




M1 Explaining that I/O corruption doesn't necessarily get sent to the secondary side of a mirrored SAN if the corruption occurs after the point at which the I/O is sent to the mirror.



M1 Explaining staggered index maintenance with database mirroring. See <http://bit.ly/IS04W5> for more explanation

SPARSE ARRAY

int array(10 billion)

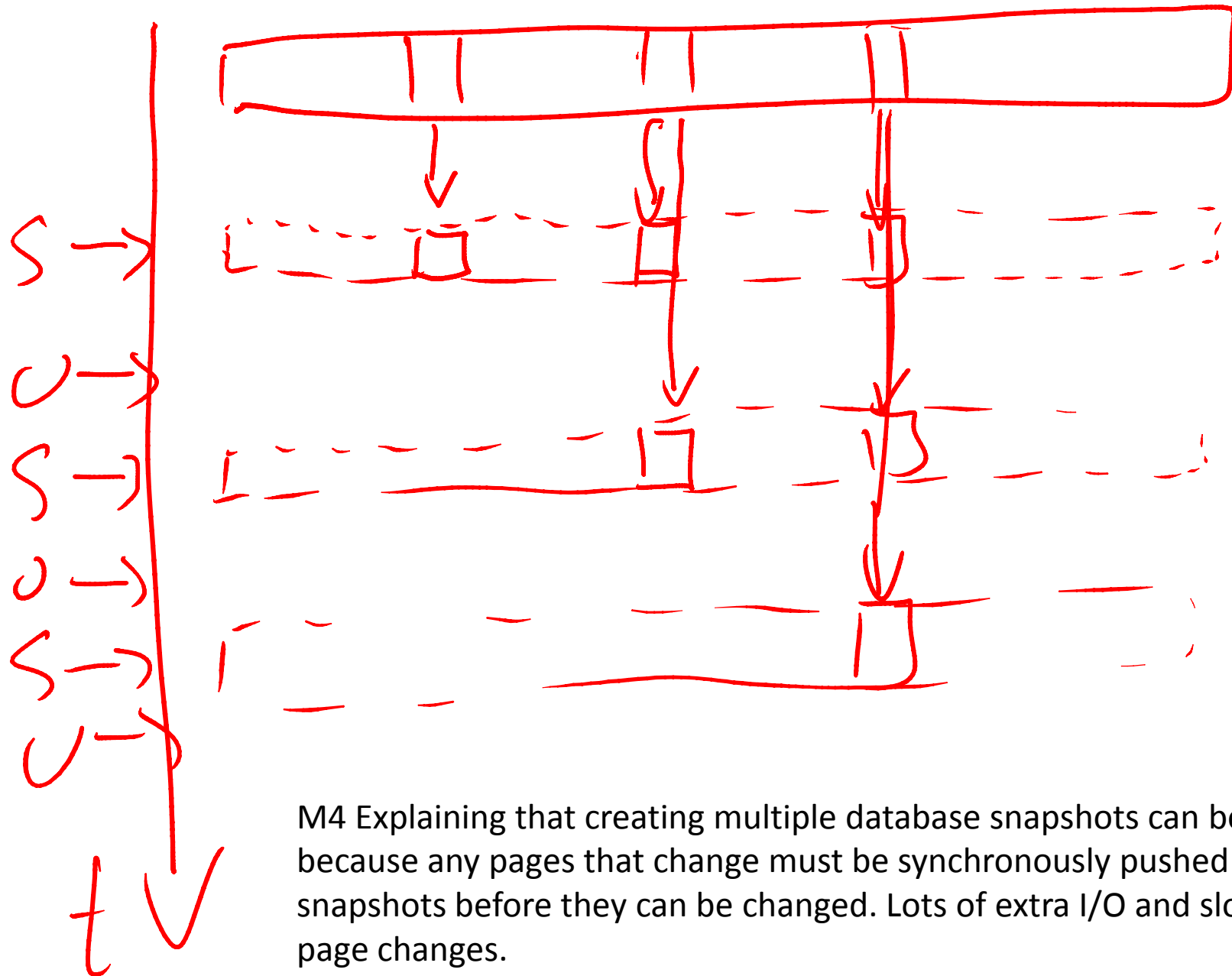
sparse

513111 10
↓ ↓

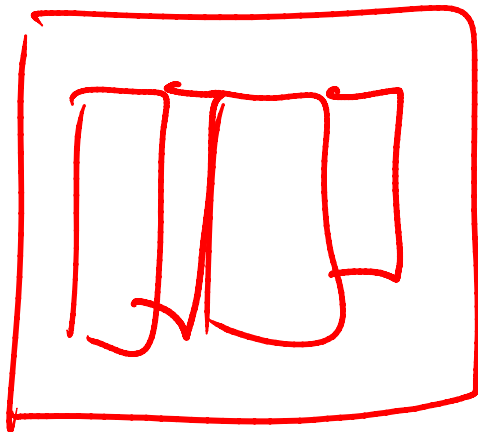
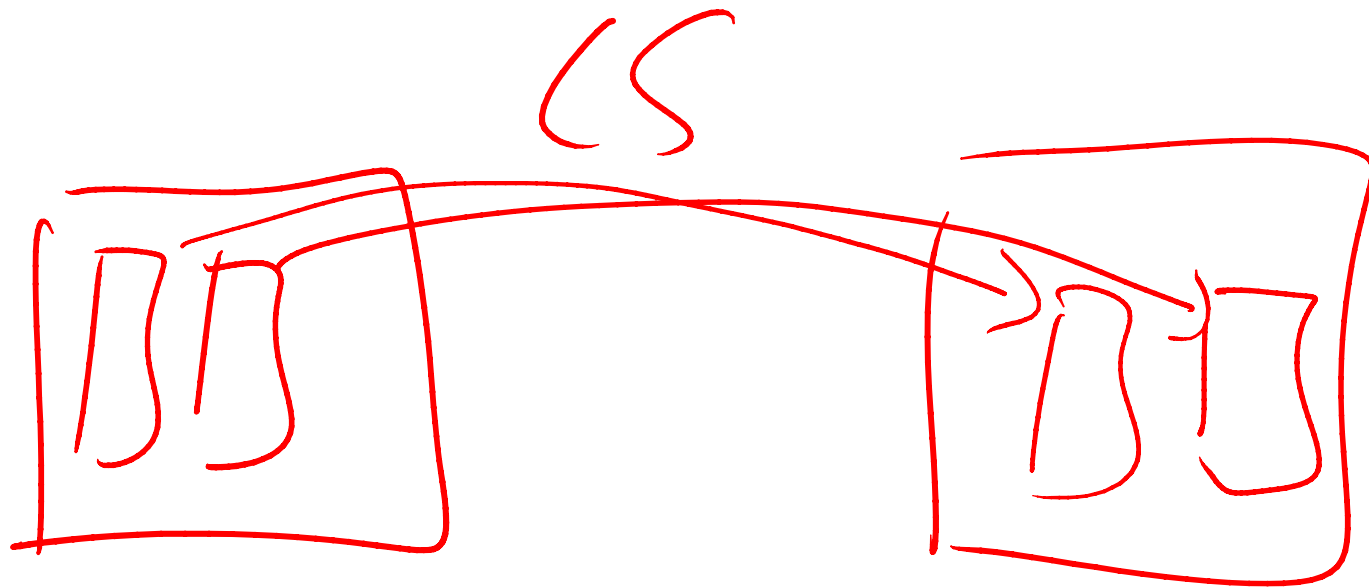


array(513111) = 10

M4 Drawing an analogy between sparse files in NTFS and sparse arrays in various programming languages. You can define an array of arbitrary size but only the populated elements take up memory.

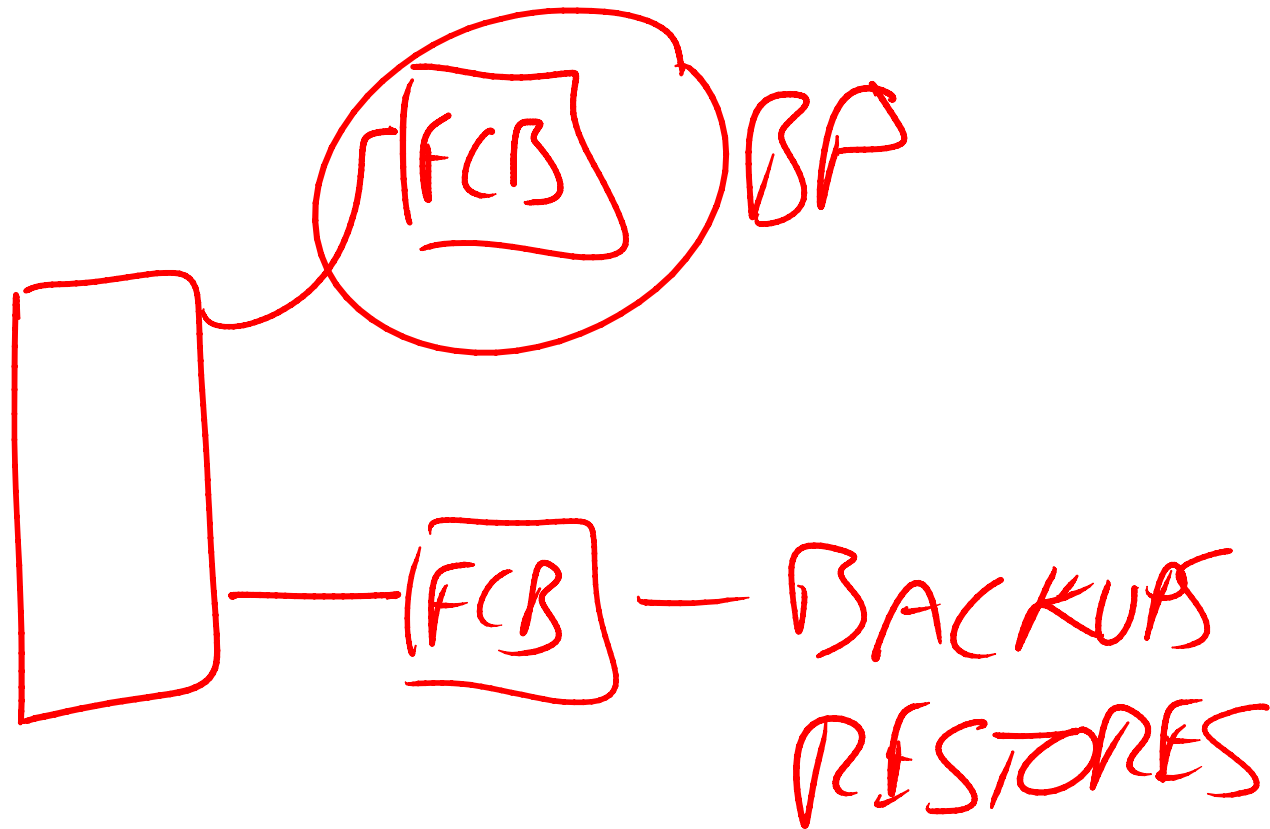


M4 Explaining that creating multiple database snapshots can be bad because any pages that change must be synchronously pushed into all snapshots before they can be changed. Lots of extra I/O and slows down page changes.

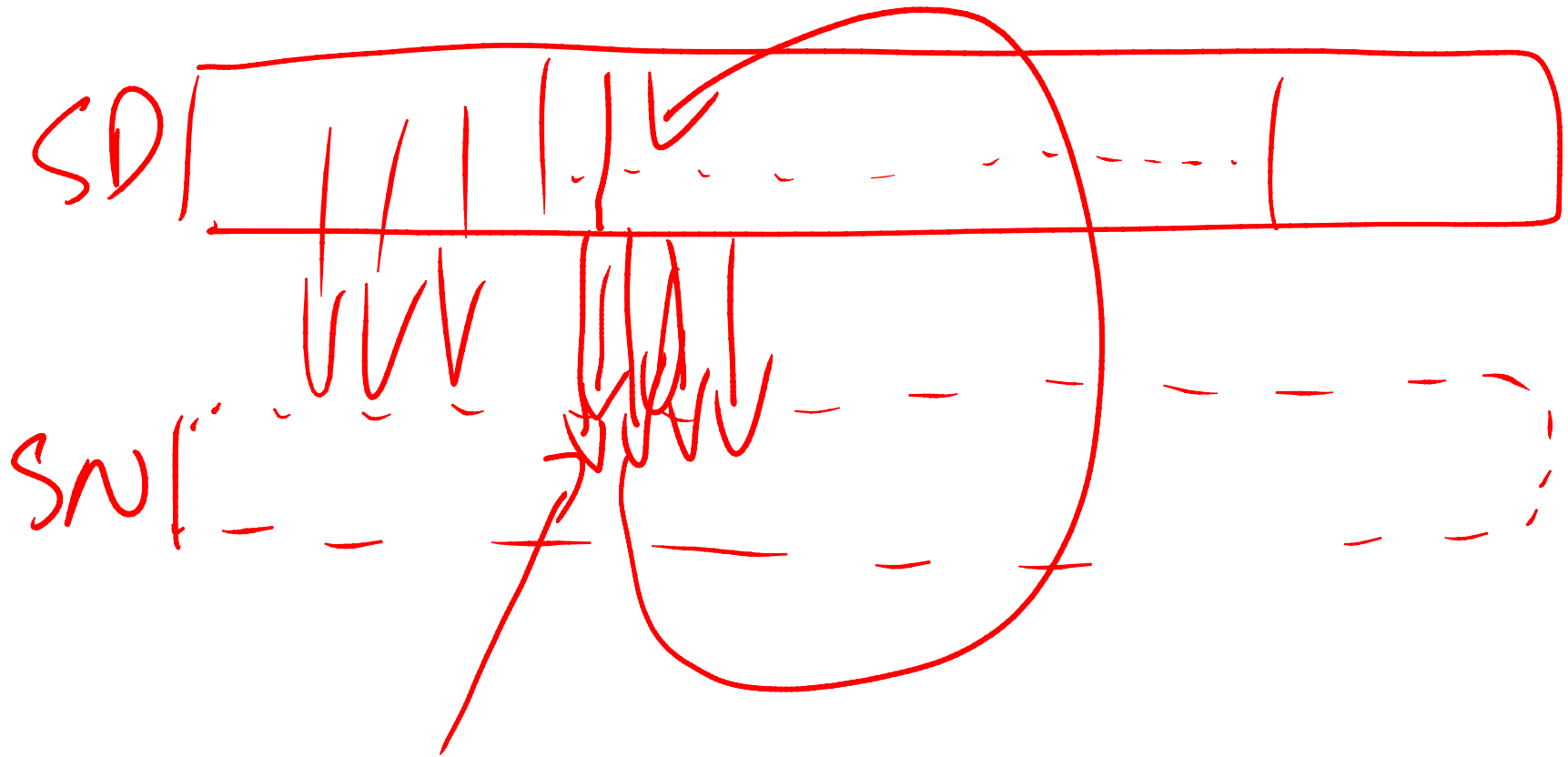


NTFS
ALTERNATE
STREAMS

M4 Explaining the difference between location of a created snapshot vs. the hidden snapshot that CHECKDB creates using NTFS alternate streams on existing data files.

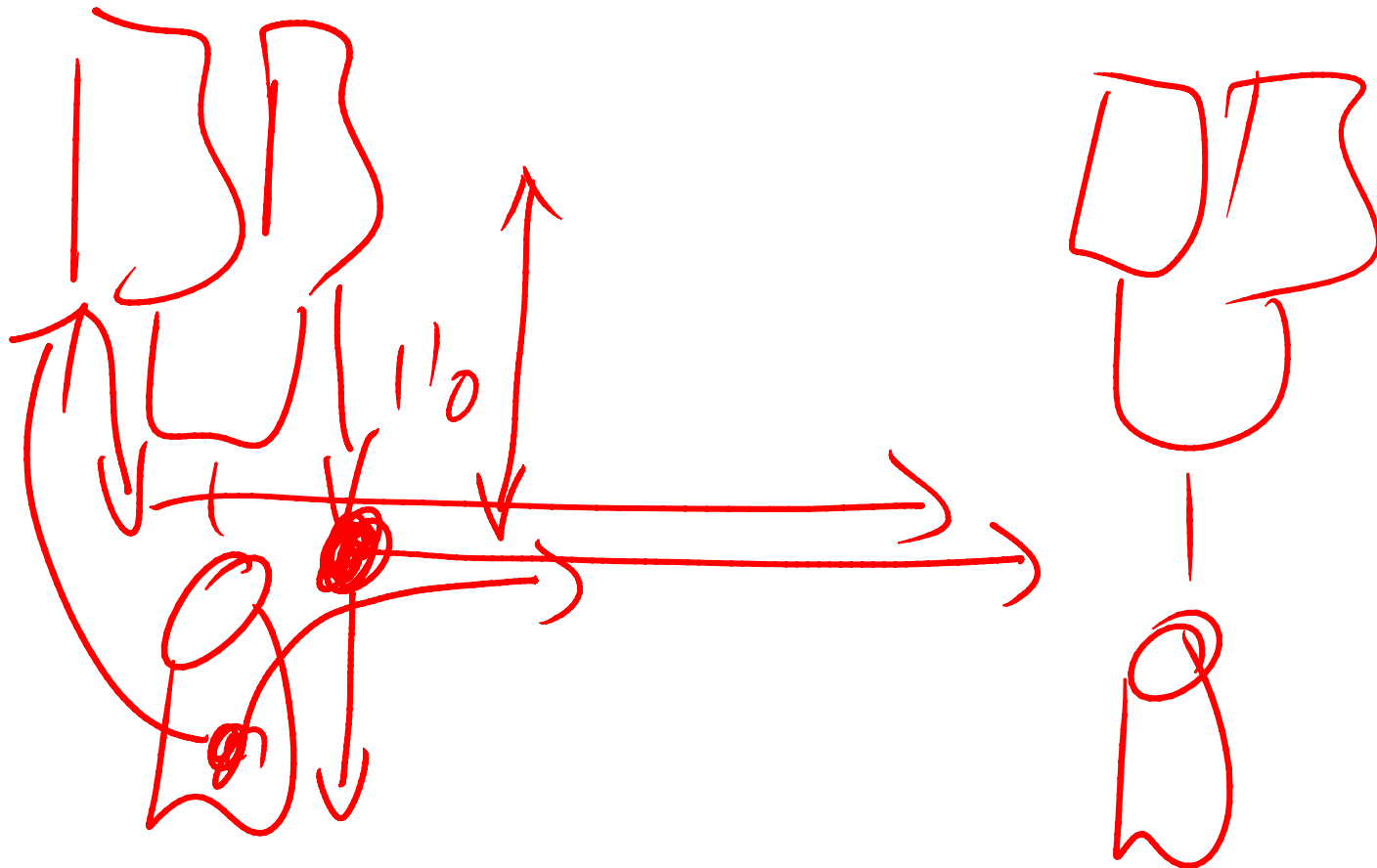


M4 Explaining how a backup opens it's own channels to the data files (called an FCB inside SQL Server – File Control Block) and reads outside the context of the buffer pool.



M4 Explaining the demo on recovering data from a snapshot. After dropping the table, the snapshot doesn't grow because the pages from the dropped table haven't been changed yet. As we repopulate the newly-created table though, it's pulling unchanged pages from the source database through the context of the snapshot to populate the new table, thus changing pages and pushing them into the snapshot. A bit of a mind-bender.

M10 Explaining that I/O corruption doesn't necessarily get sent to the secondary side of a mirrored SAN if the corruption occurs after the point at which the I/O is sent to the mirror.



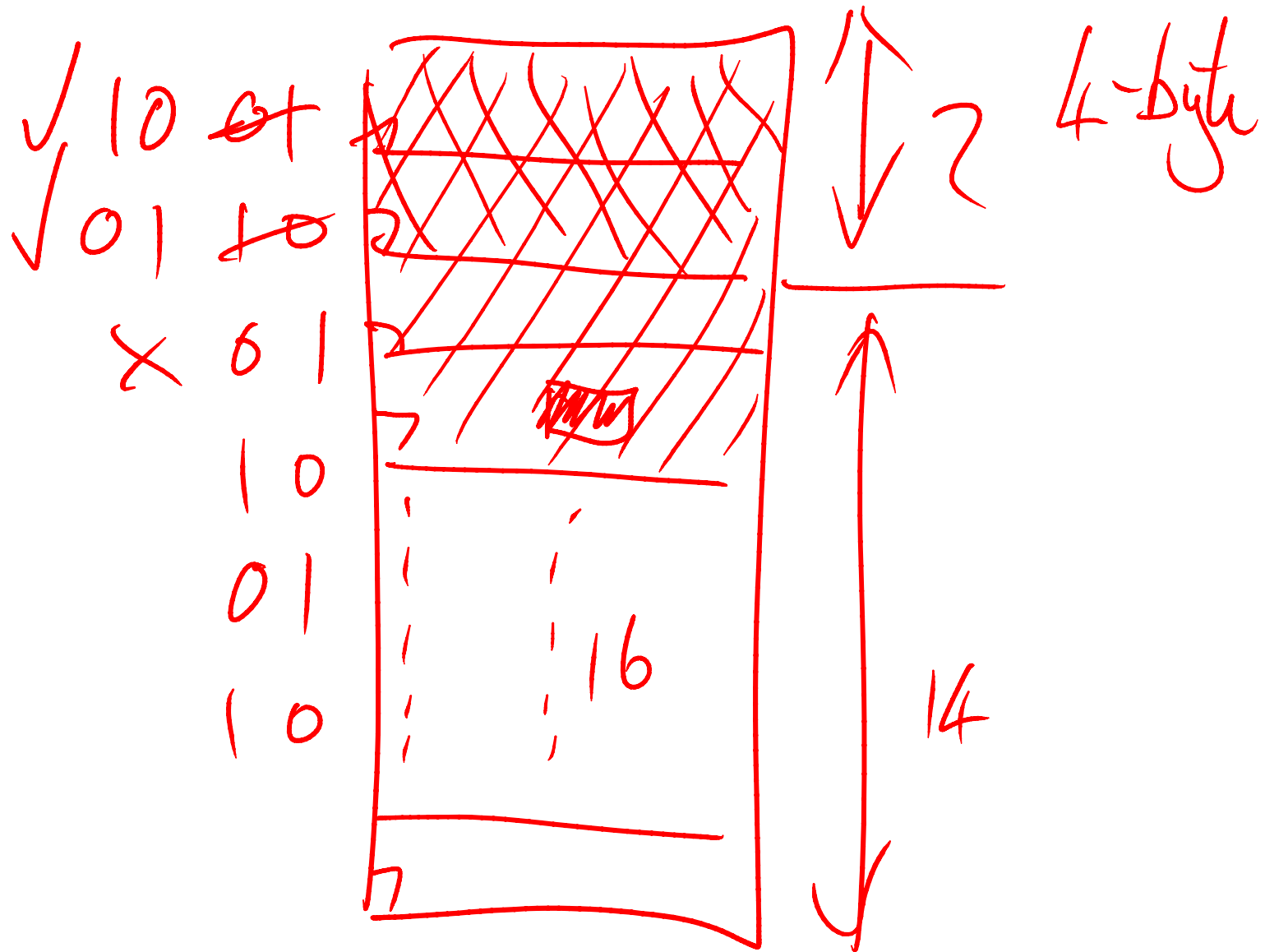
M10S7 Explaining the various page corruption errors. 832 is in-memory corruption of a page that has not changed since it was last read from disk. See <http://bit.ly/pxMKv> for more details.

823 }
824 } I/O errors

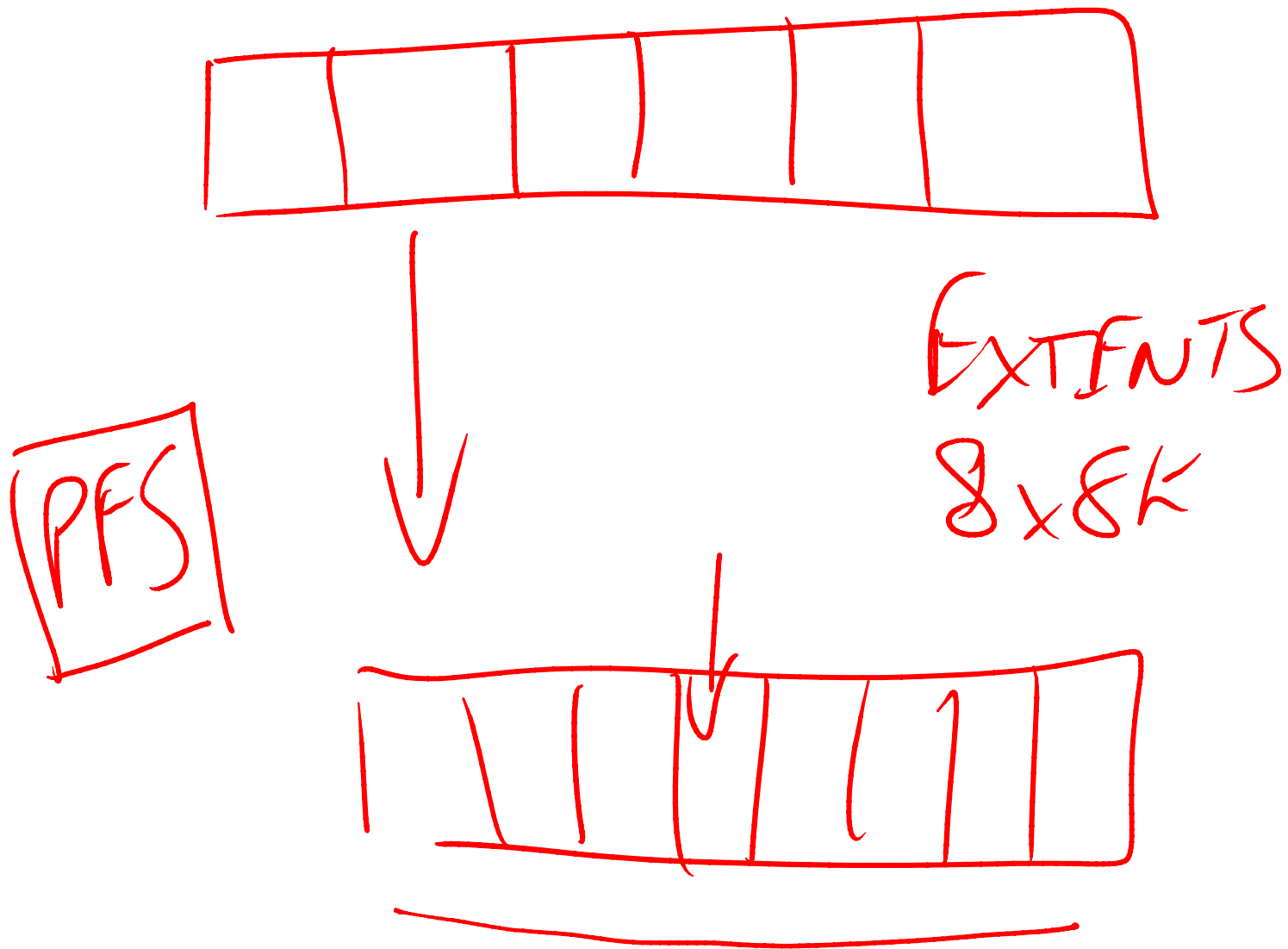
825 — Read Retry

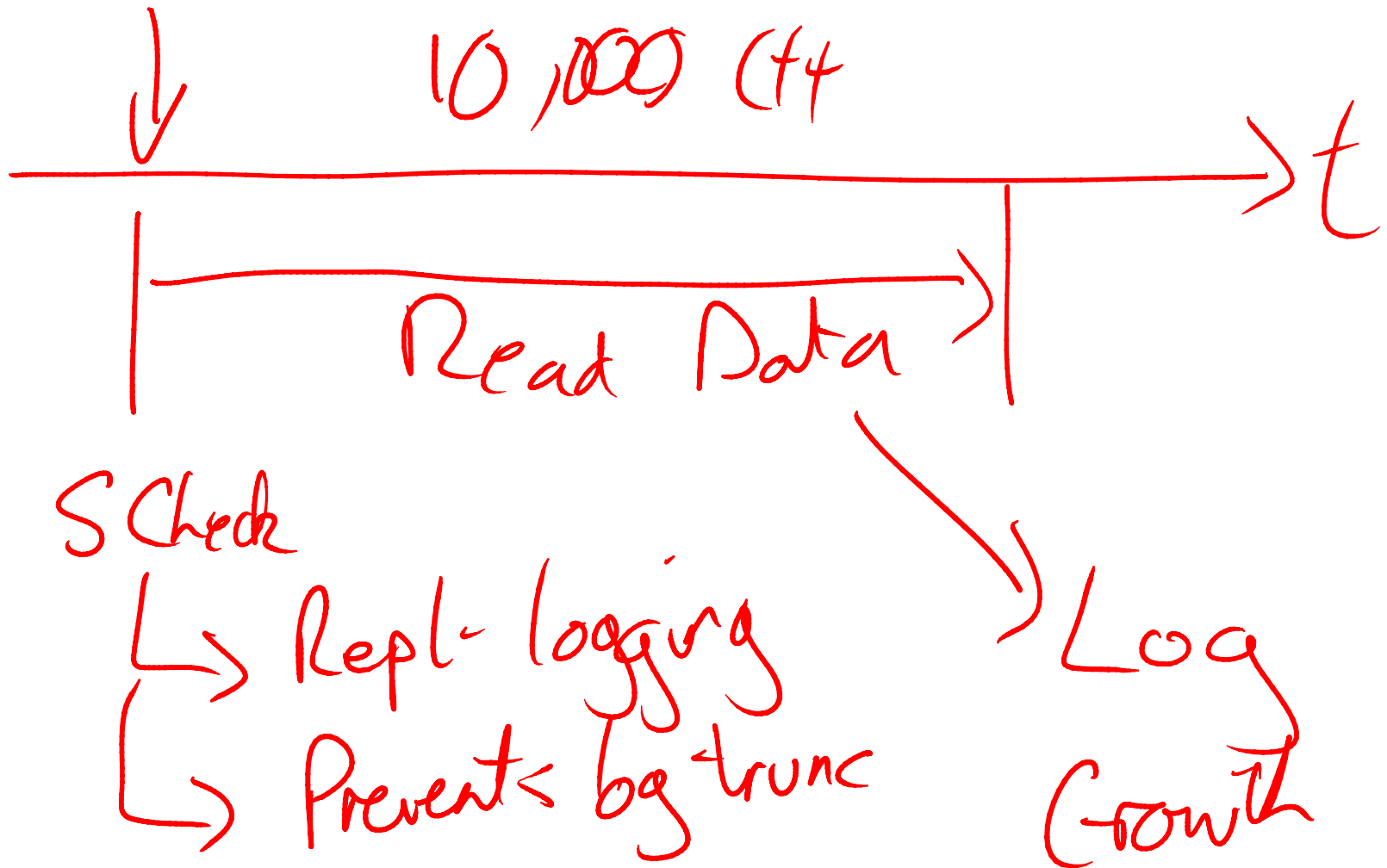
832 — Memory Corruption

M10S11 Explaining torn-page detection

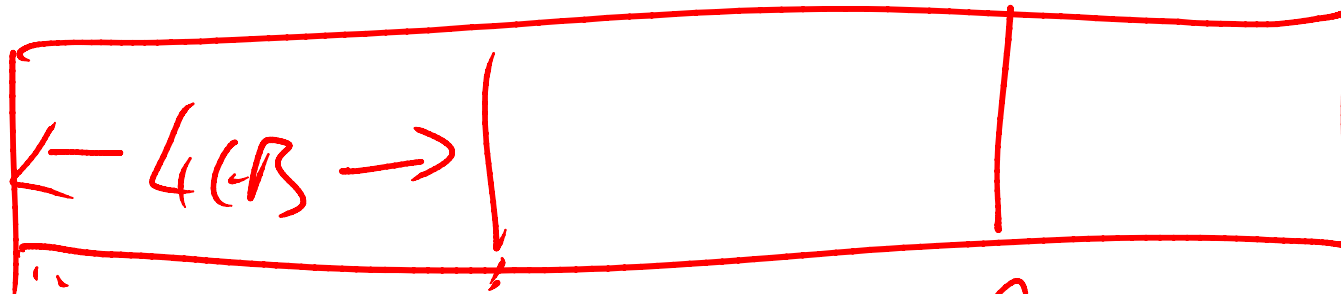


M10S13 Explaining how a backup will read an entire extent, even if only, say, 1 page is allocated from the extent. It uses the PFS allocation byte-map to tell which pages are allocated and need their page checksums checked.





M10S23 Explaining how CHECKDB in 2000 was online, using log analysis to perform crash recovery inside CHECKDB. Log truncation was prevented, leading to log growth during long CHECKDB runs. I had fun ripping out the 10,000 lines of code it took for this during 2005 dev when I switched to using a database snapshot.



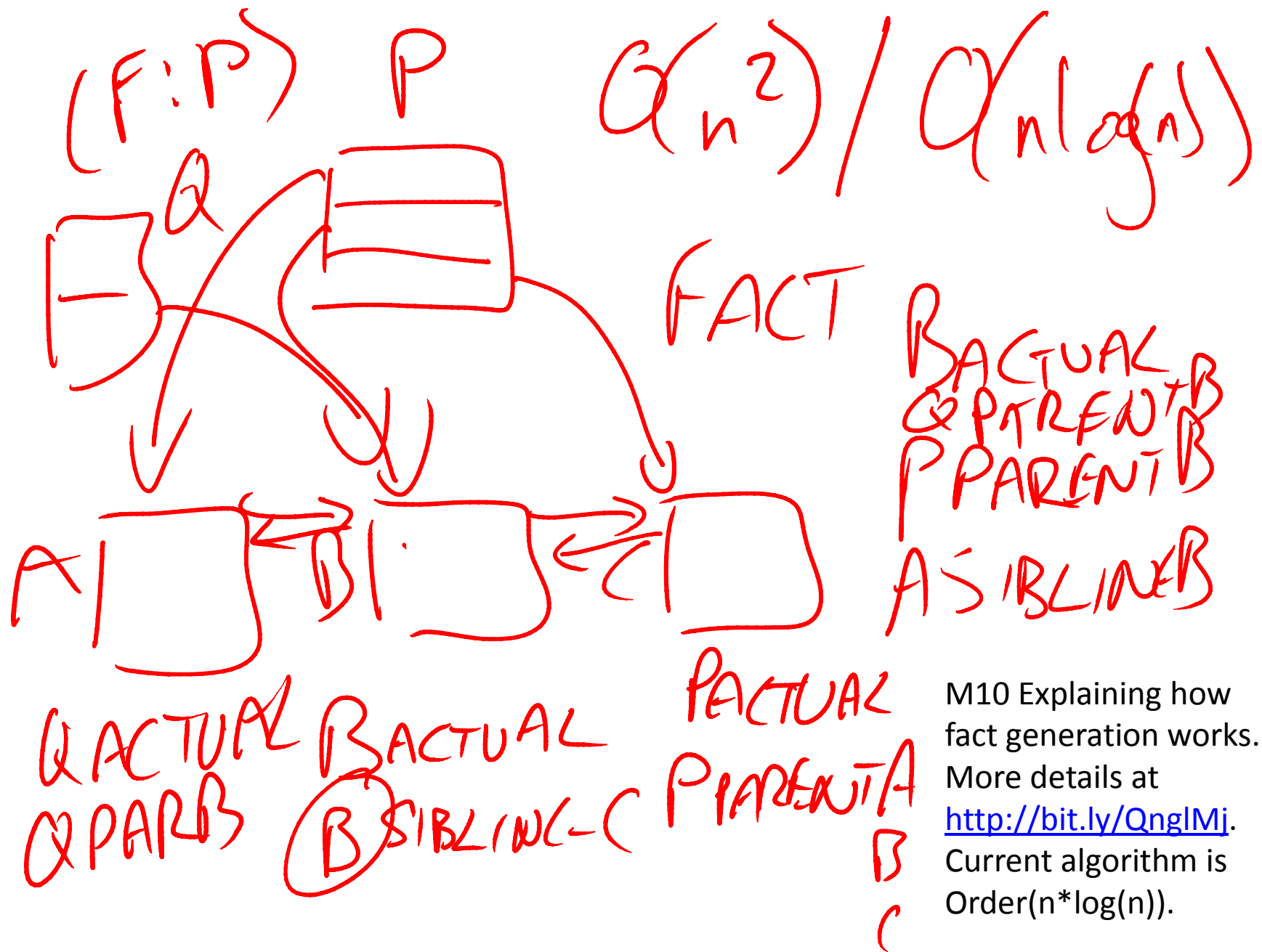
GAM INTERVAL
4GB

T1 ... IAM

T2 ... IAM

M10 Allocation checks will XOR all the IAM pages for a single 4GB GAM interval together to make sure no two tables have the same extents allocated to them.

XOR = $\emptyset$

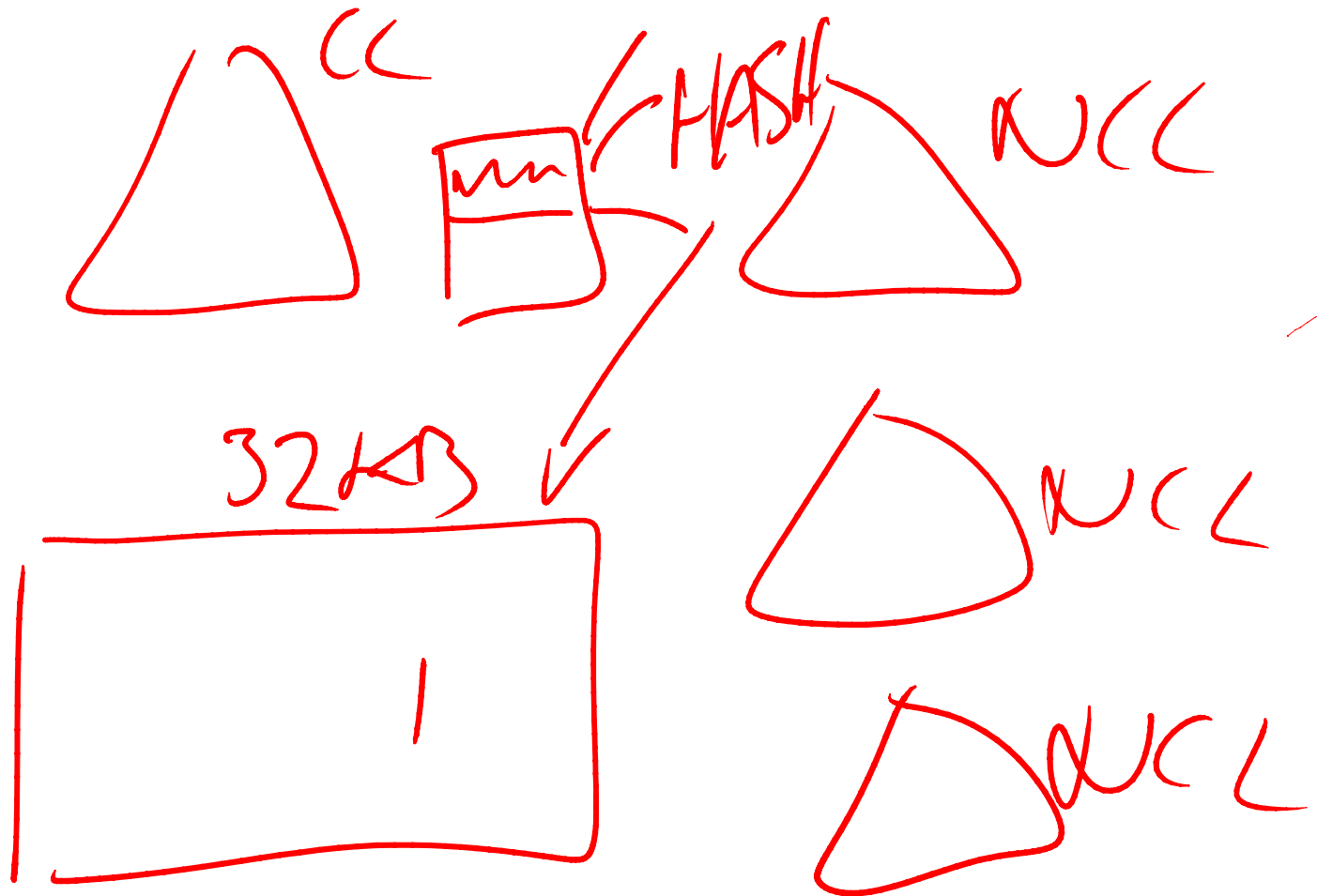


M10 How to find how much tempdb space CHECKDB will use. Details at <http://bit.ly/QnglMj>

WITH
ESTIMATEONLY

30KB

M10 Explaining how the nonclustered index checking algorithm works. Each data record is used to construct all NC index records, then each is hashed to a value and a bit is flipped in a bitmap. Hashing the actual NC index records should map to the same bits, flipping the bits back again. Lossy algorithm, so any bits set at end mean a deep-dive to figure out which records are incorrect.



M10 Two ways to graphically view the data file internals.



ORCA MDF

INTERNALS VIEWER