

SPARSE FILES

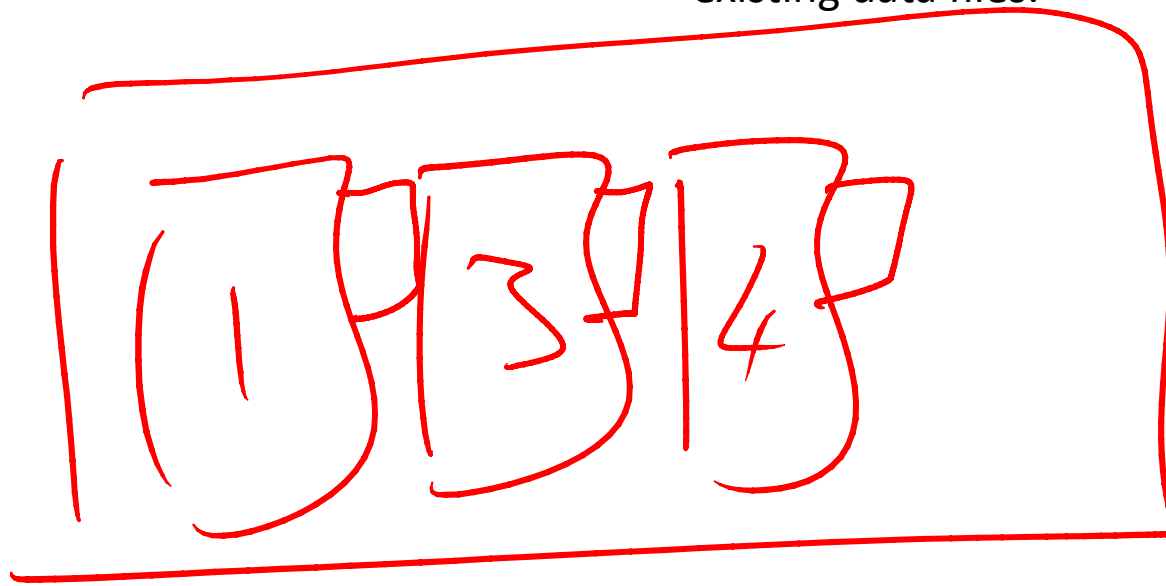
sparse array int [10000000]
array[44] = 6

array[34566] = 10

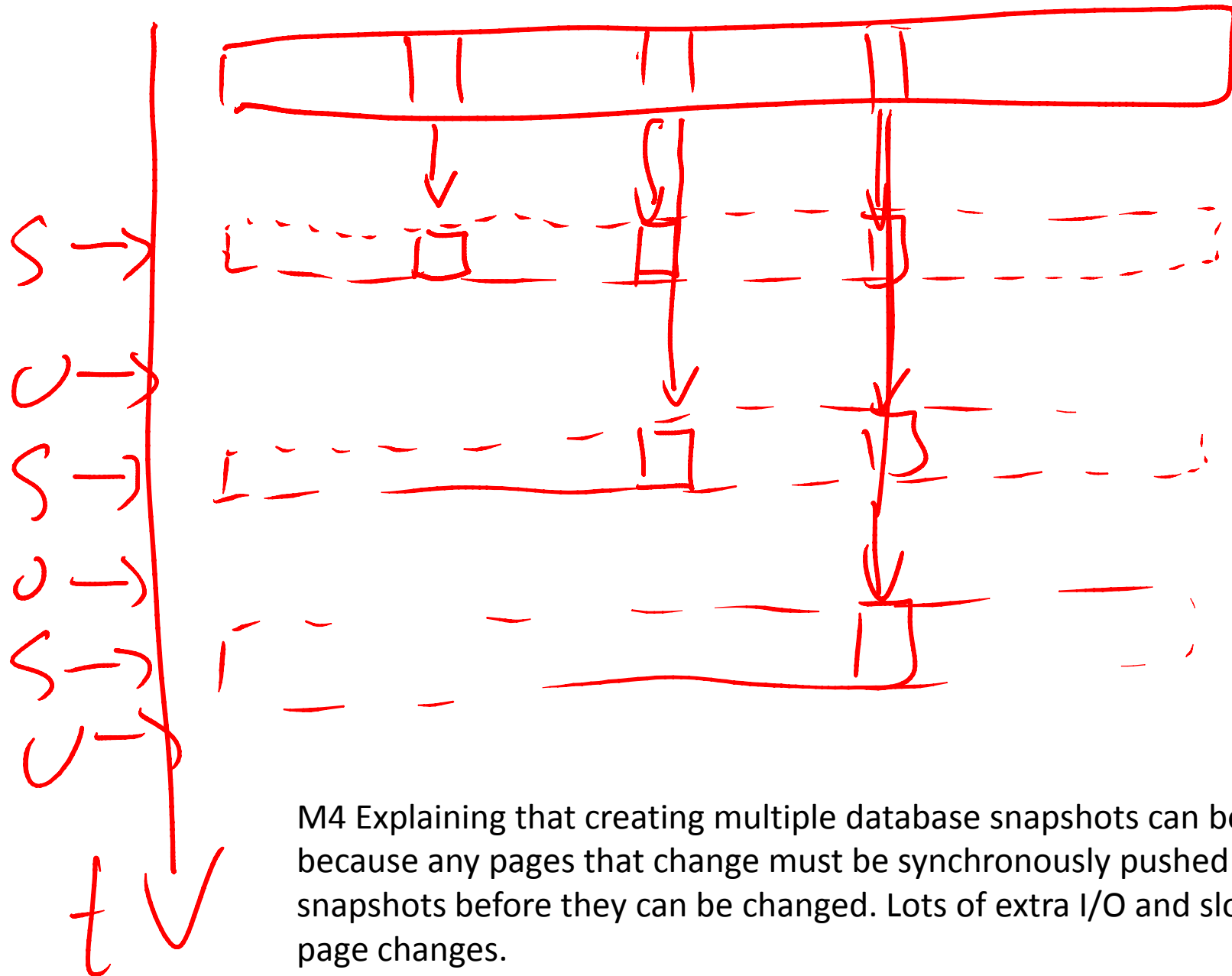
345	10	44	6
661			

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.

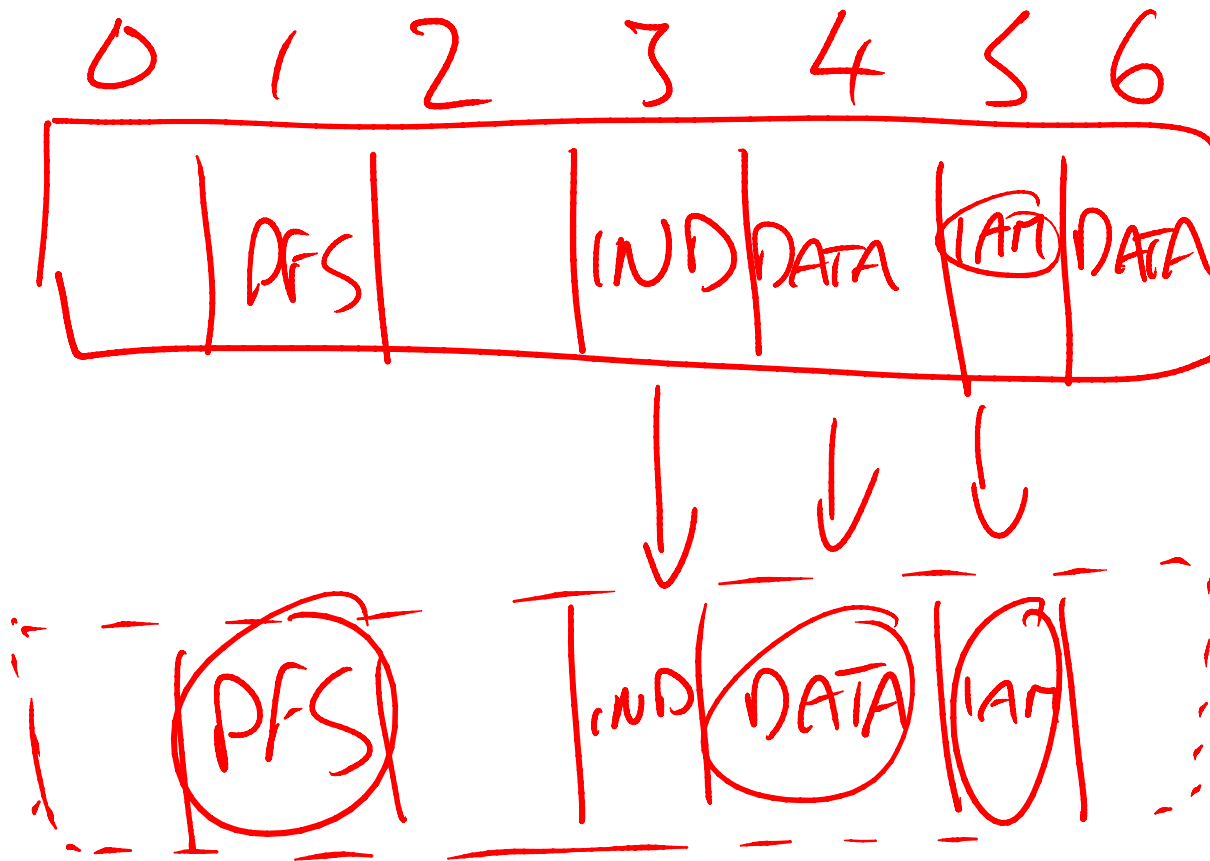
M4 Explaining the difference between location of a created snapshot vs. the hidden snapshot that CHECKDB creates using NTFS alternate streams on existing data files.



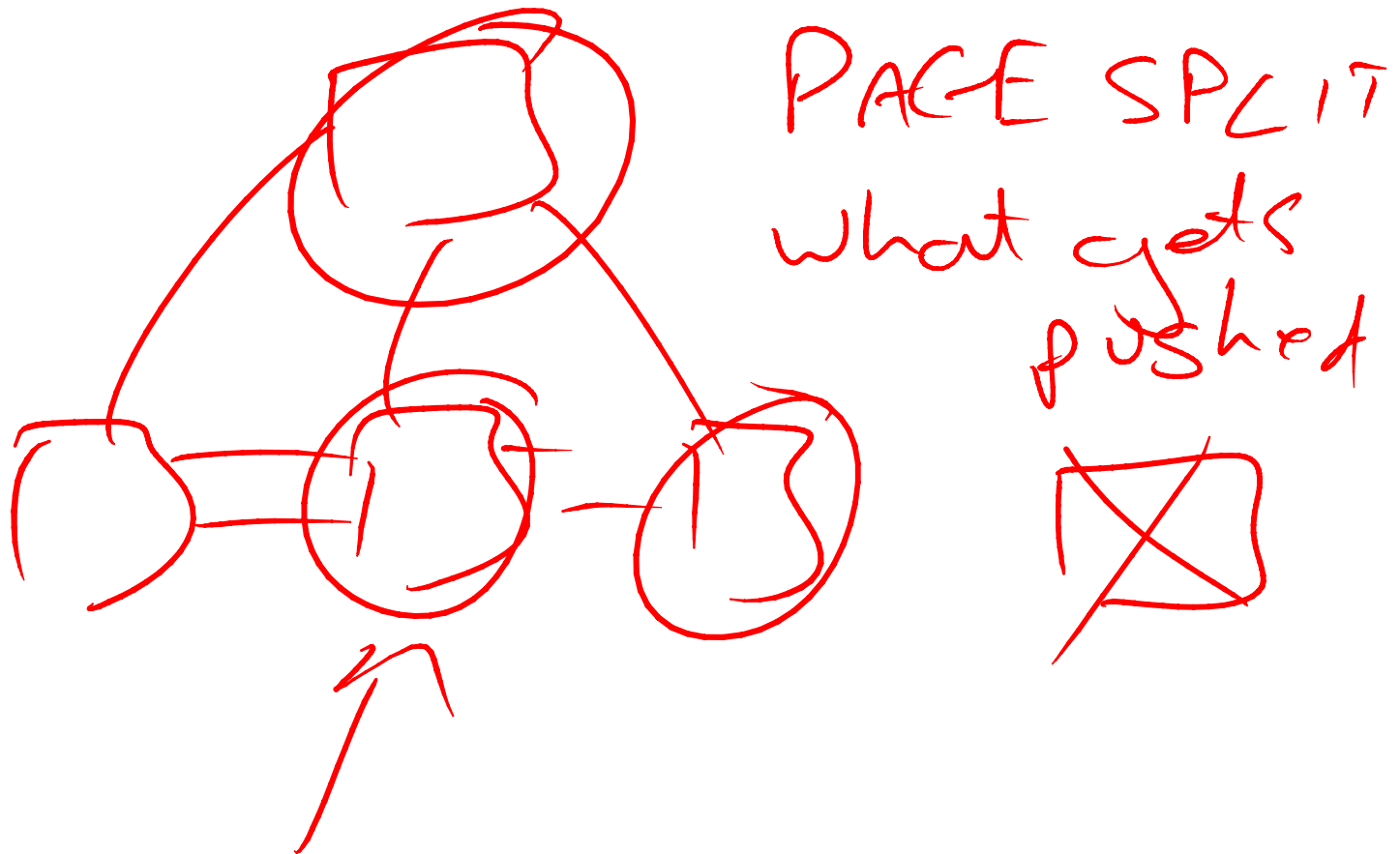
ALTERNATE
STREAMS
SPARSE



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.



M4 What gets pushed when a new page is added to a clustered index? Everything that changed that existed at the time the snapshot was created.

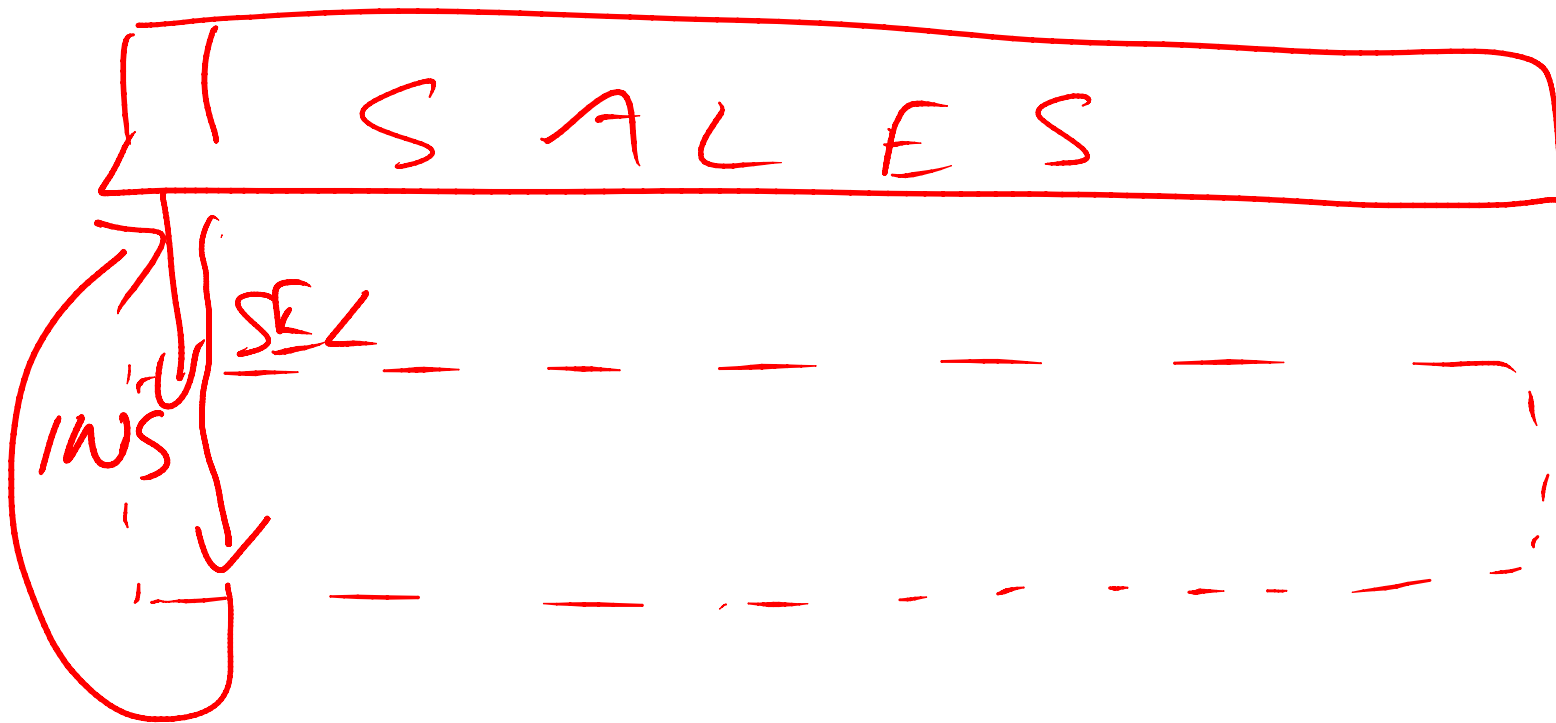


M4 Explaining what gets pushed into a snapshot when a page split occurs. The update, existing pages may get pushed (circled), but the newly allocated page will only get pushed if it was allocated at the time the snapshot was created (then deallocated, and re-used as the split page).

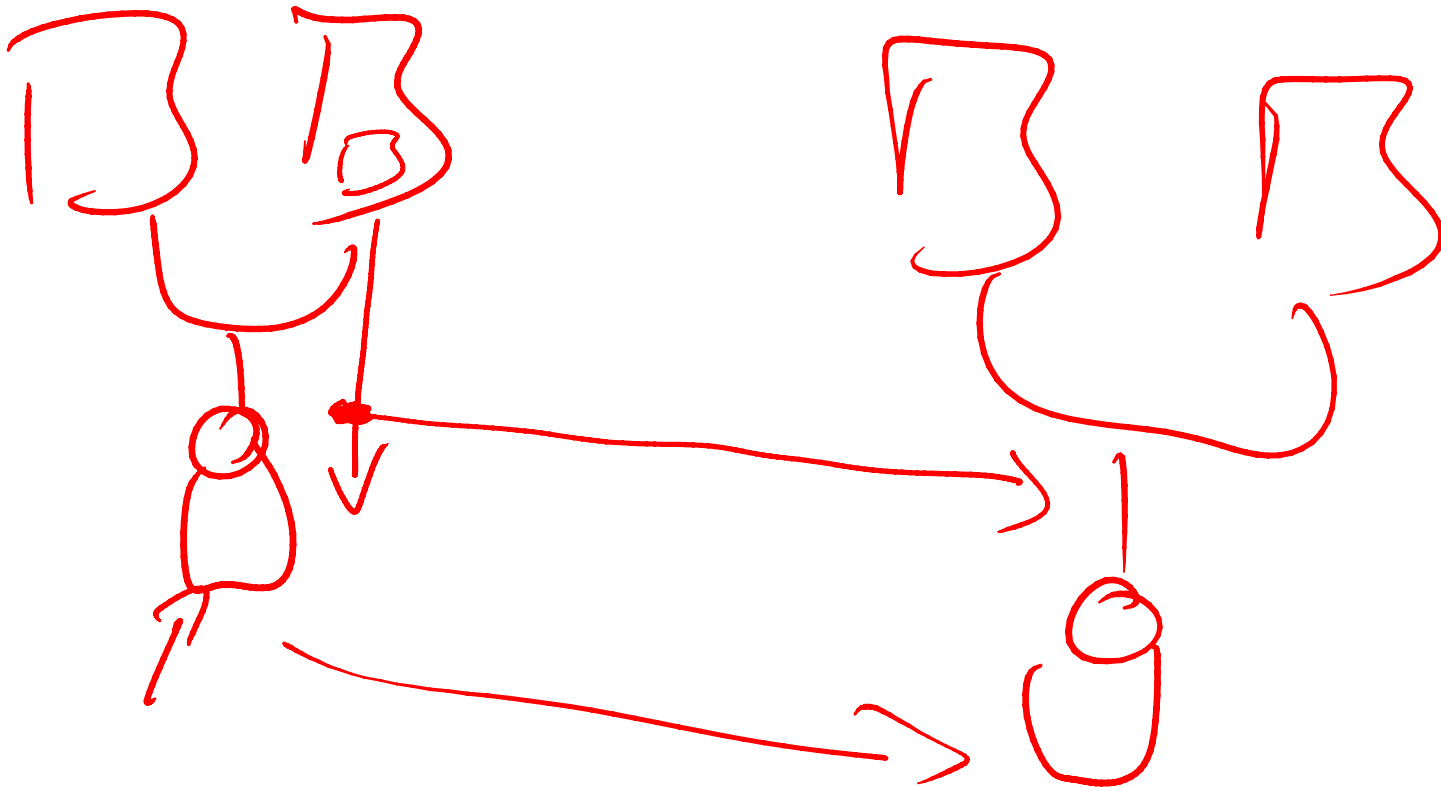


Rebuild

M4 Explaining that when rebuilding an index into newly allocated space in a file, the new data/index pages are not pushed into the snapshot, as they were not allocated at the time the snapshot was created and so their page images do not need to be preserved.



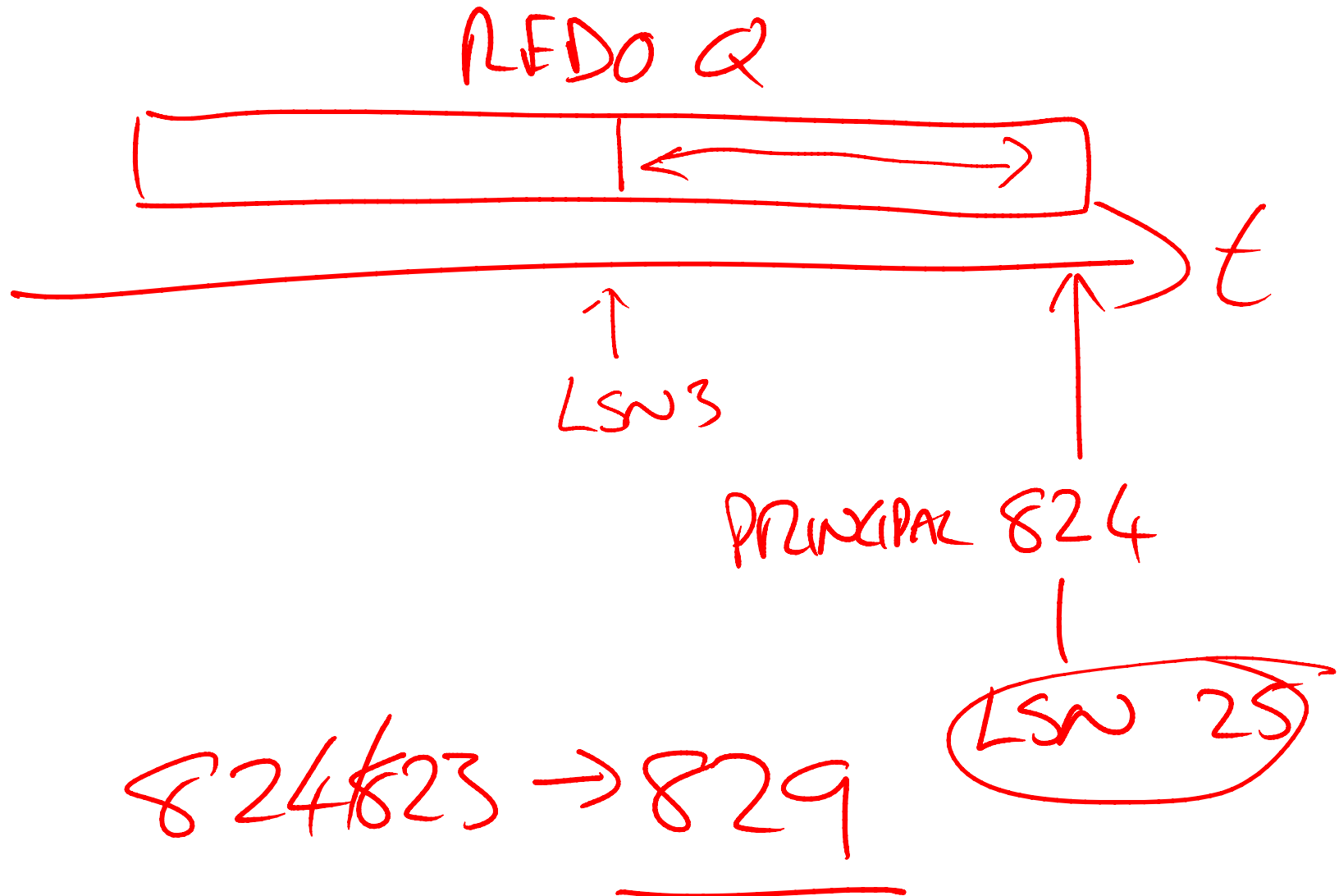
M4 Explaining the demo on recovering data from a snapshot. After dropping the table, the snapshot doesn't grow because the pages from the dropped table haven't been changed yet. As we repopulate the newly-created table though, it's pulling unchanged pages from the source database through the context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



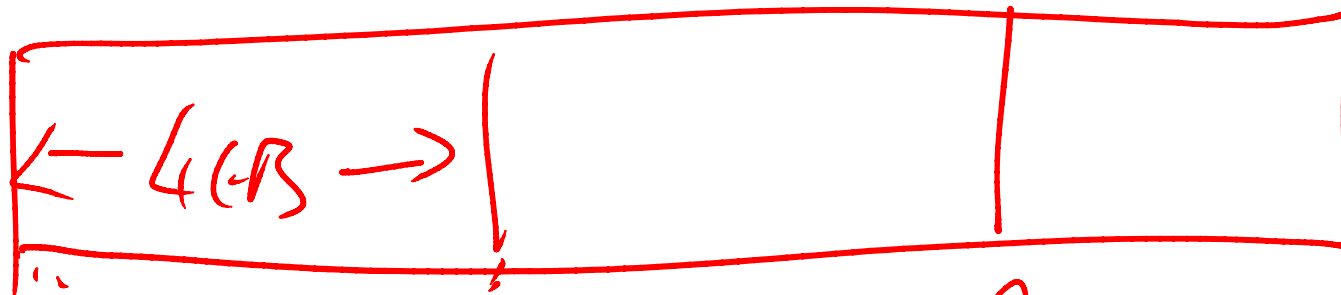
M10 Explaining that I/O corruption doesn't necessarily get sent to the secondary side of a mirrored SAN if the corruption occurs after the point at which the I/O is sent to the mirror.

M10S11 Explaining torn-page detection





M10S15 Explaining how automatic page repair in DBM may not be instant, because the mirror REDO queue needs to get to the LSN at which the principal asked for the damaged page, to avoid sending an older page image to the principal



GAM INTERVAL
4GB

T1 ... IAM

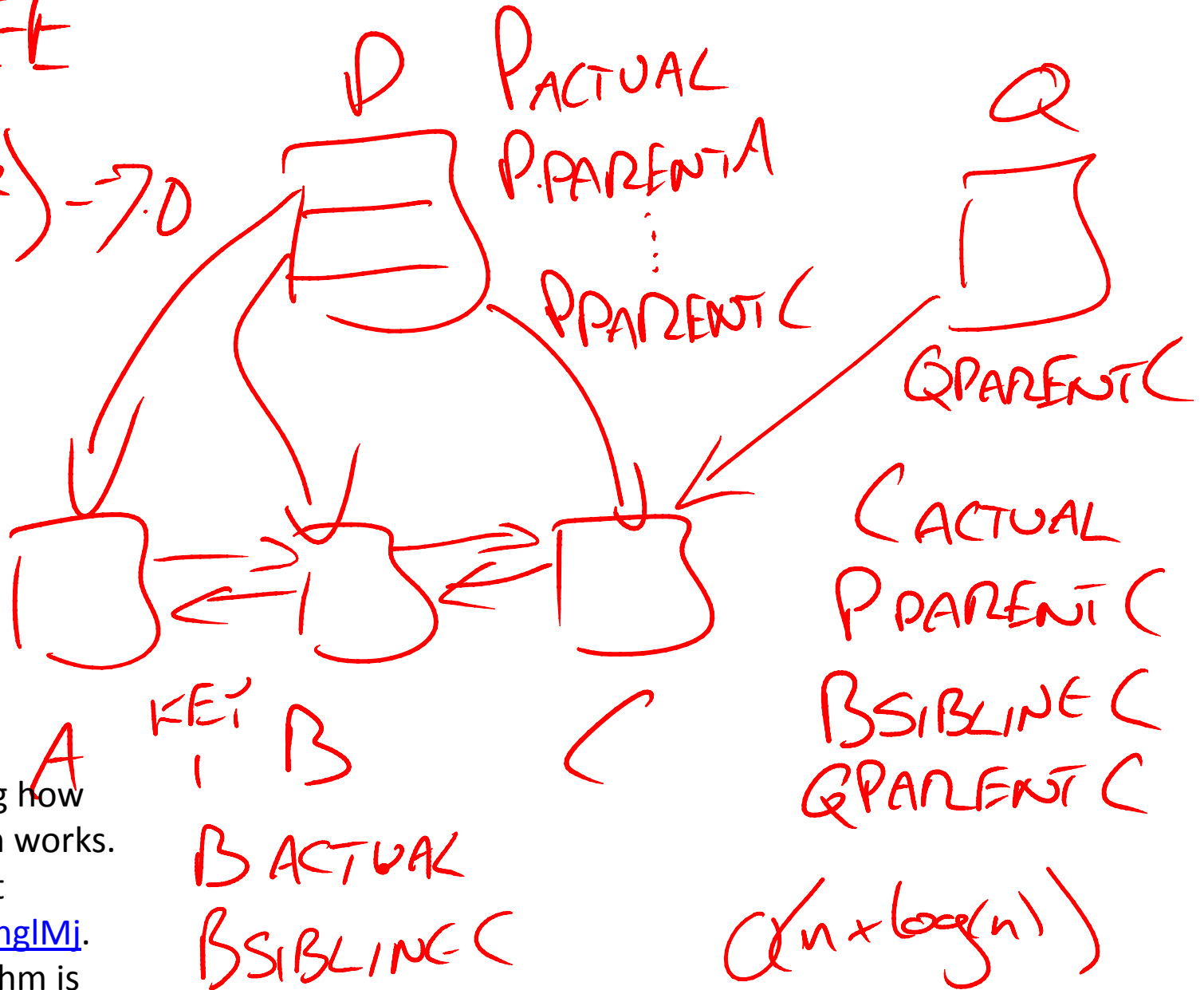
T2 ... IAM

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 = $\emptyset$

BSTREE

$O(n^2) \rightarrow 0$



M10 Explaining how
fact generation works.
More details at
<http://bit.ly/QnglMj>.
Current algorithm is
 $\text{Order}(n \cdot \log(n))$.

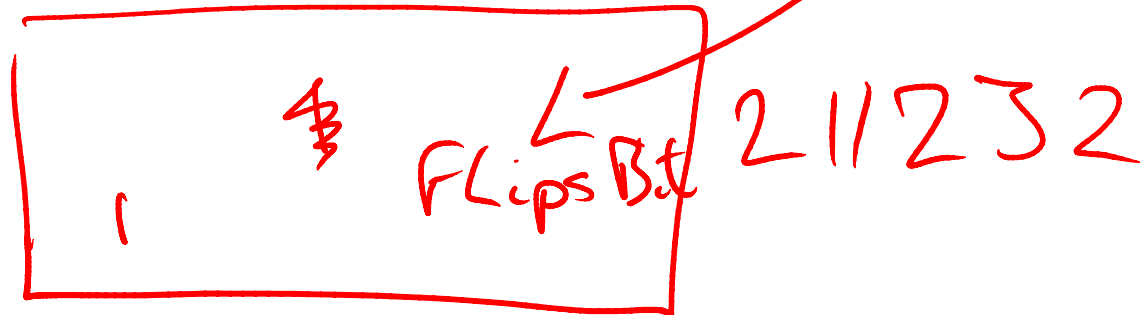
TABLE

[c1, c2, c3]

[NC 1 c1
NC 2 c2]

→ gen NC 1 record

→ HASH → 4 byte #



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.