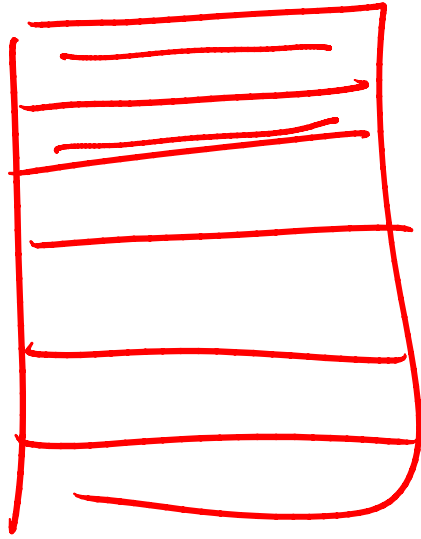


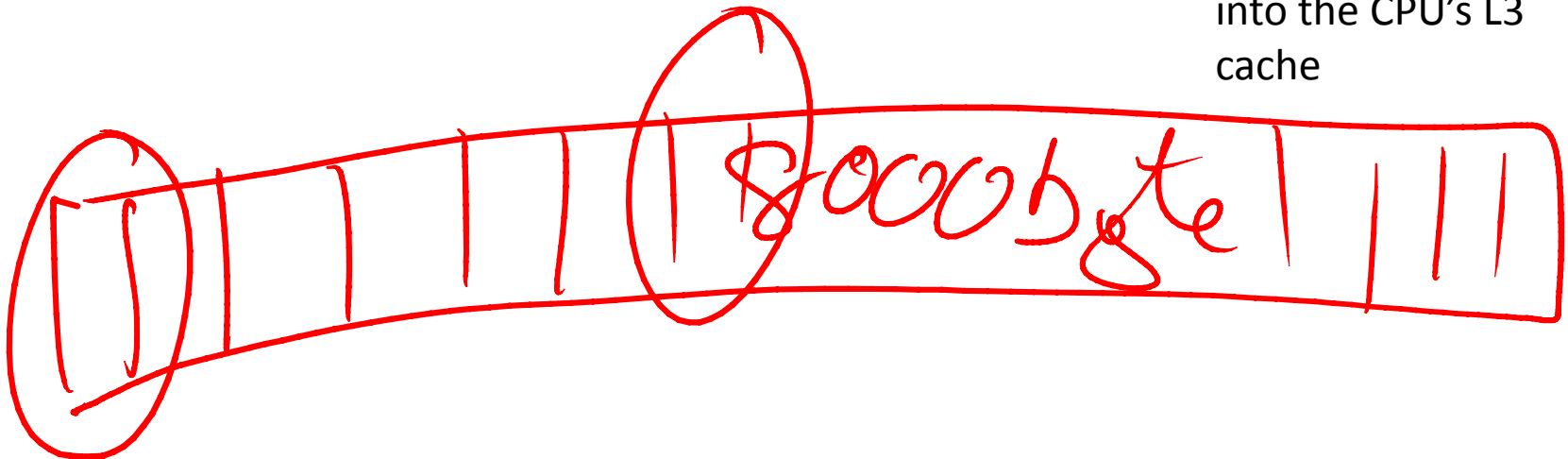
#Sqlhelp

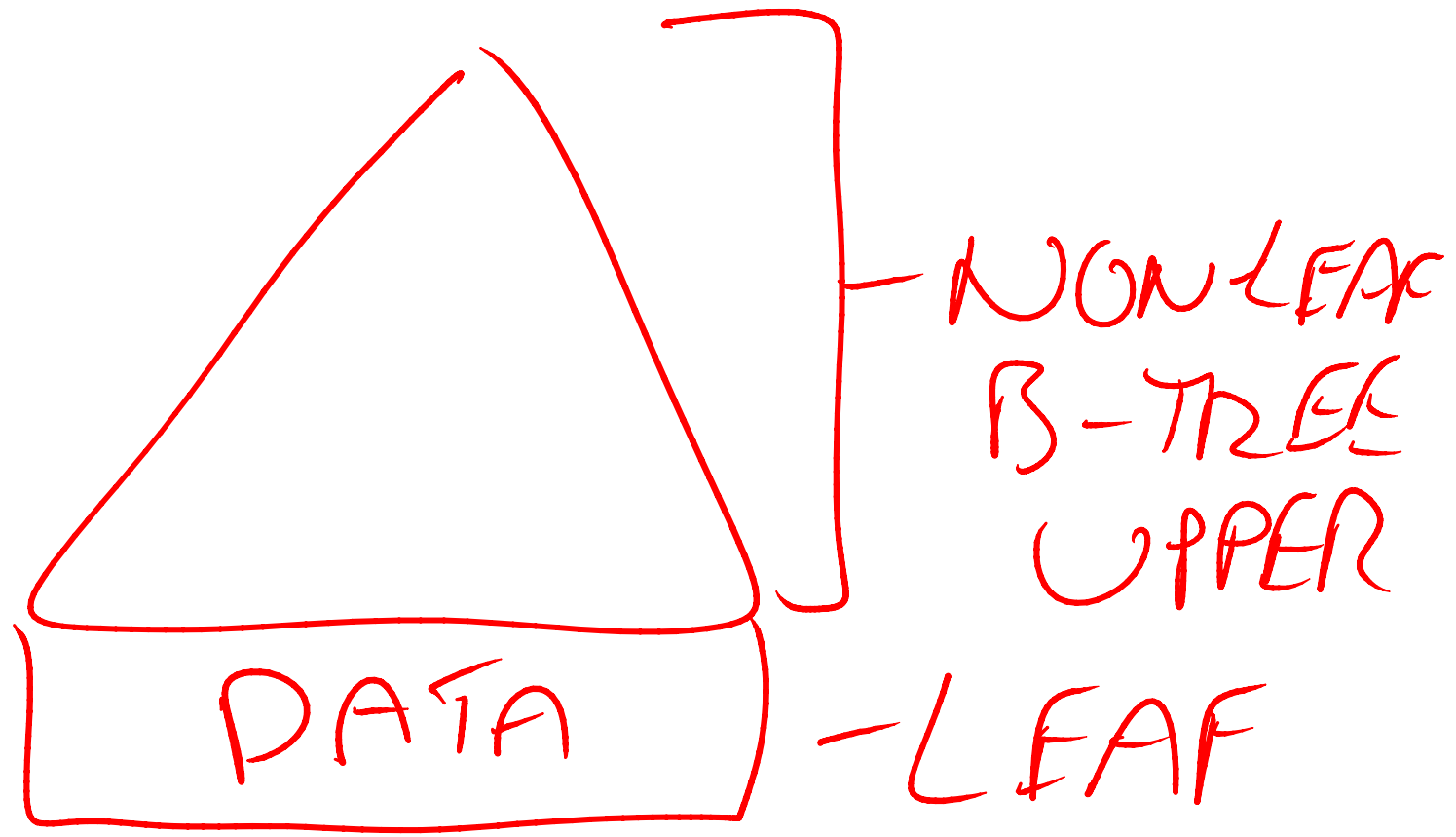
CACHE LINE INVALIDATION

CACHE LINES
64b / 128b



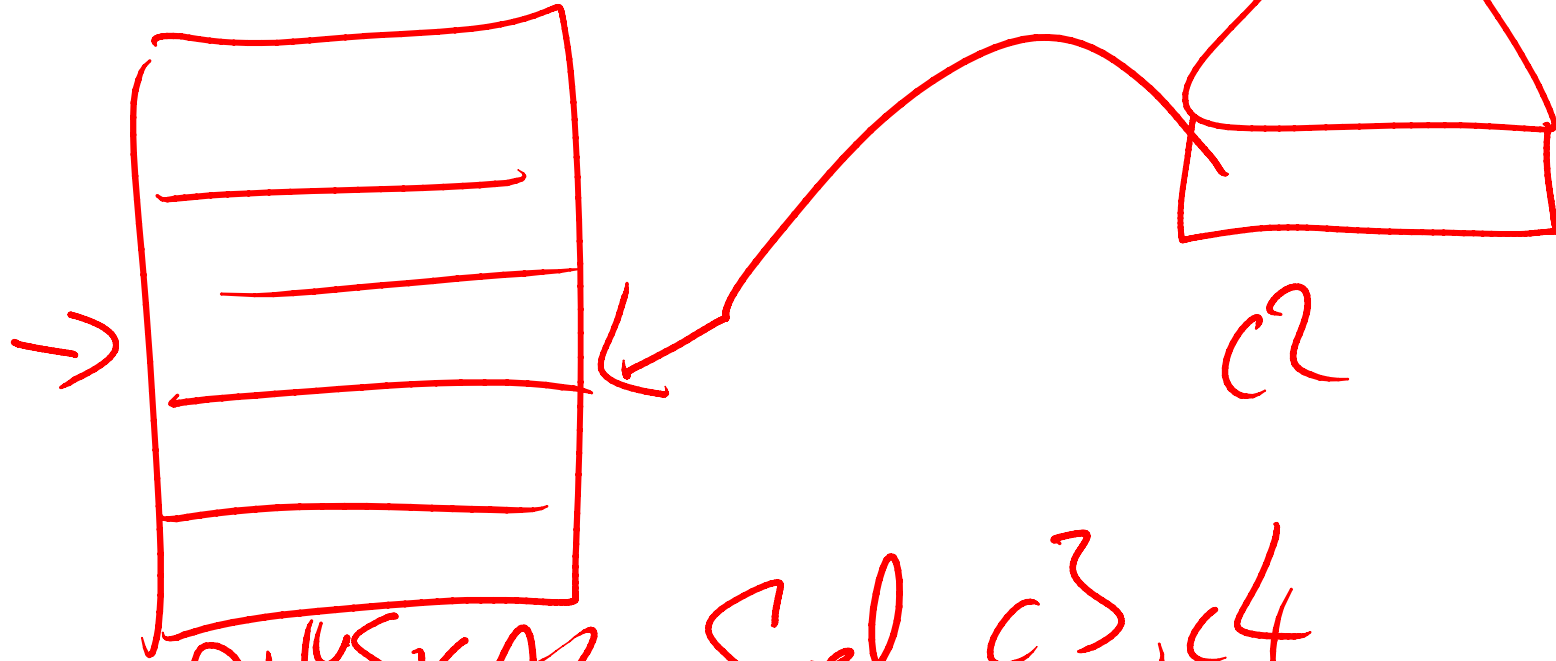
M1S06 Discussing
why the NULL
bitmap is a perf
optimization by
avoiding having to
read record parts
into the CPU's L3
cache





M1S8 Explaining the division of data records/pages and index records/pages in a clustered index.

$c_1, c_2, \dots, c_{10}$



HL: PHYSICAL
RID

CL: LOGICAL
RID

Set c_3, c_4

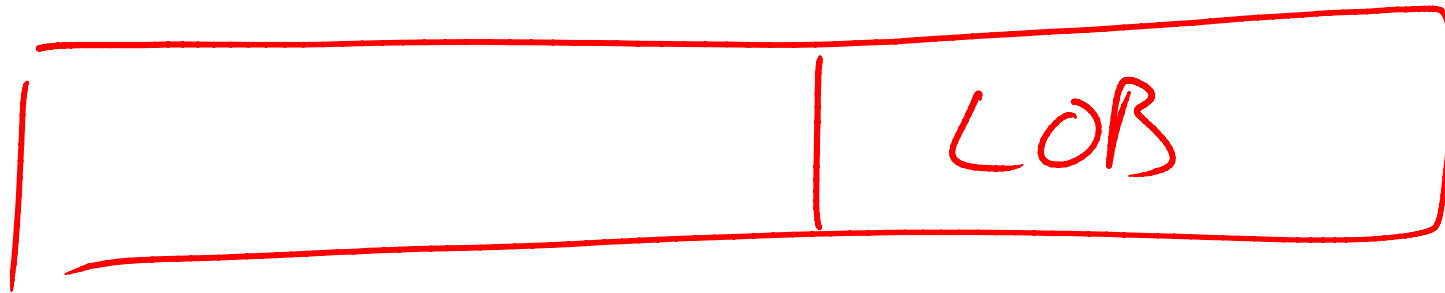
Where $c_2 = X$

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.

sys.dm_db_index_physical_stats

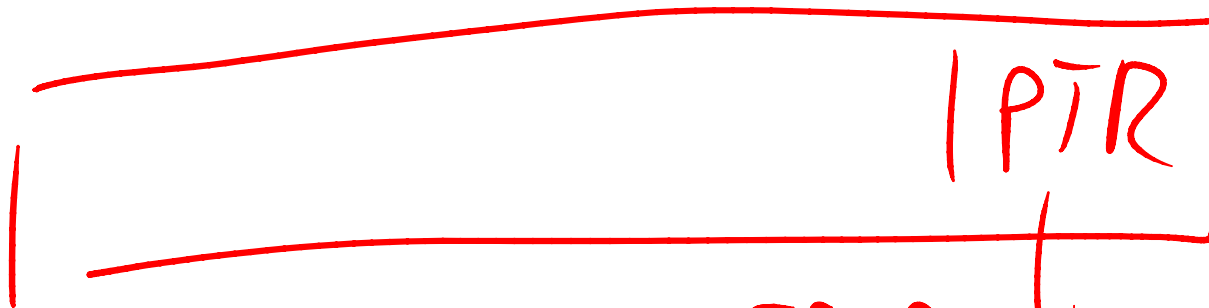
AccessMethods/Forwarded
Records/sec

M1S9 You can track forwarded records using the sys.dm_db_index_physical_stats DMV, and how often they're being used with the Forwarded Records/sec counter in the Access Methods perfmon object.

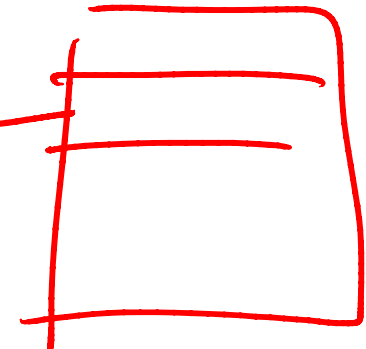


M1S11 A LOB value that is completely stored in the data row is called 'in-row'. If it's stored outside the data row, with a pointer to it stored in the data row, the LOB value is called 'off-row' or 'out-of-row'.

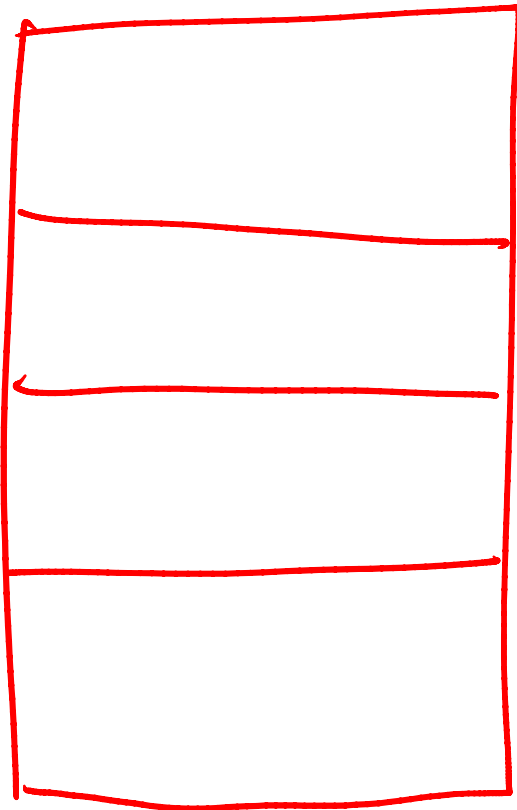
IN-Row



OFF-ROW

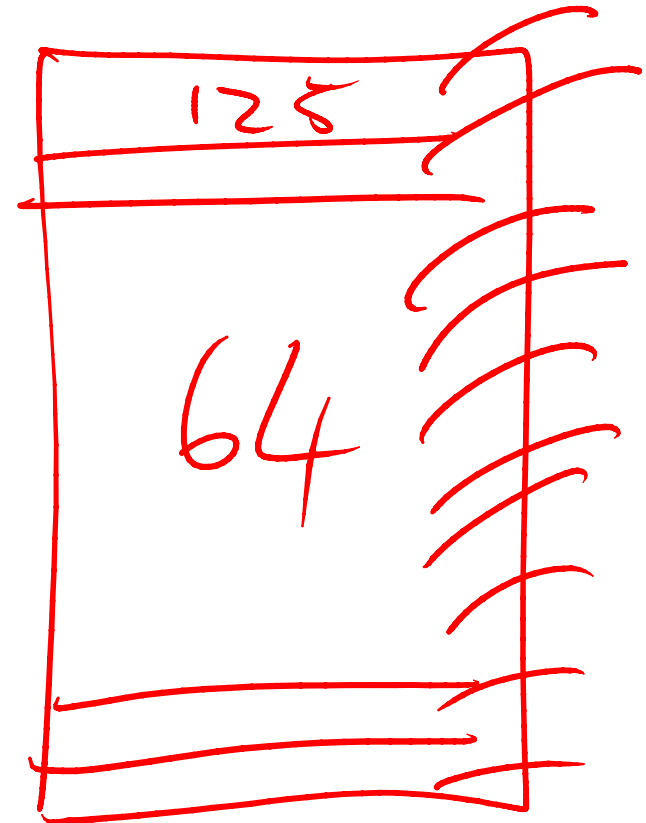


100	1900 byte
-----	-----------



4

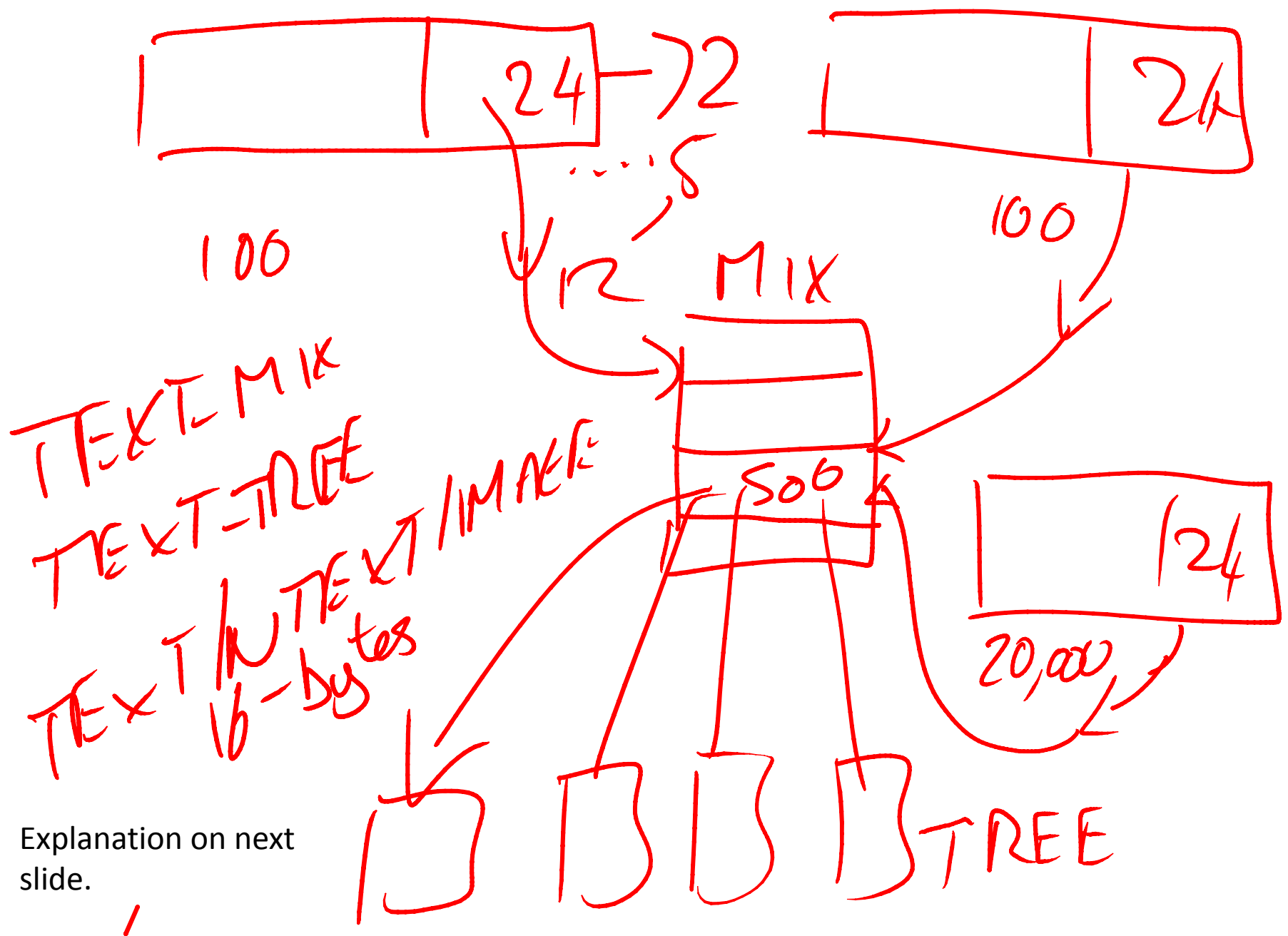
M1S12 Having a large LOB value in row that's not used much makes for large rows and low data density. Choose how to store LOB data wisely.



~~VARCHAR~~ VARCHAR(5000)
... (5000)
✓ (5000)

ROW_OVERFLOW

M1S11 If multiple (n)varchar (1-8000) columns exist, and their values are such that they can't all fit in-row, one or more will be pushed off row into ROW_OVERFLOW allocations.



Explanation on next slide.

From previous slide:

M1S11 Off-row LOB values are stored as loose trees. Values that are small enough from the same table will share TEXT_MIX text pages. Portions from large LOB values will have their own TEXT_TREE text pages. In-row pointers to off-row LOB values vary in size from 16 bytes (for legacy LOB types) to 24-72 bytes (for 2005+ LOB types).

TEXT INTERNALS

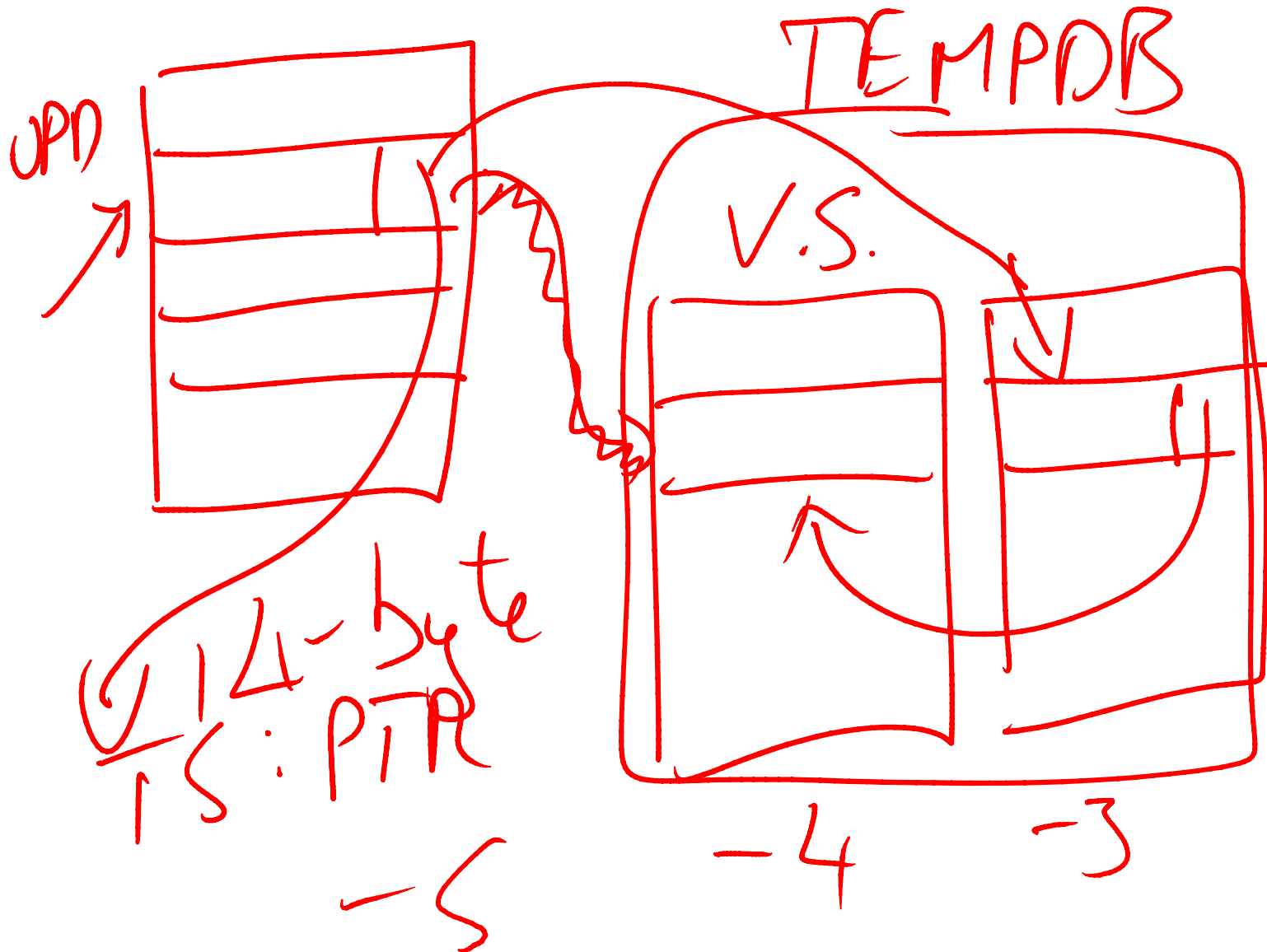
MARK

RASMUSSEN

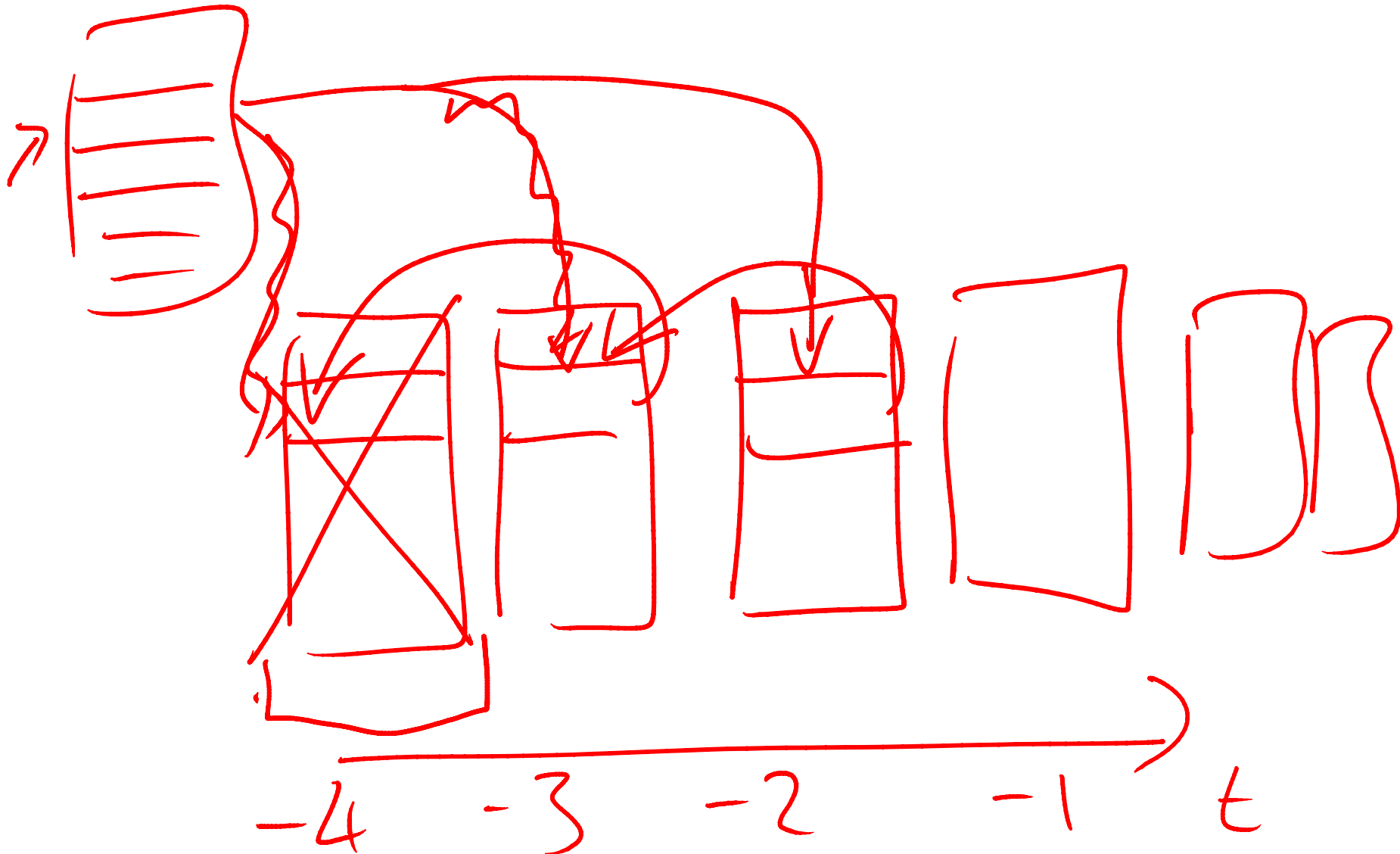
ORCAMDF

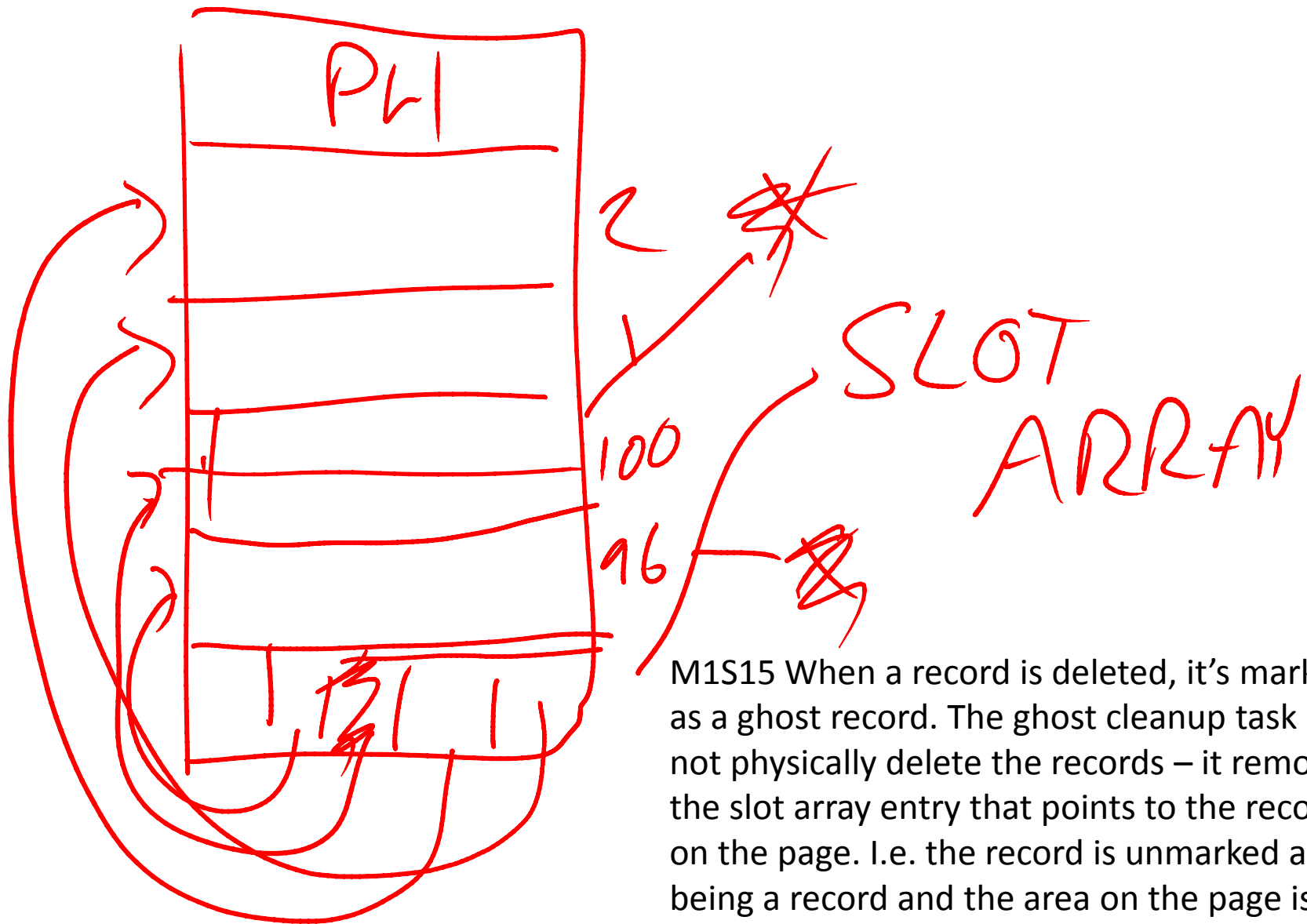
M1S11 Mark Rasmussen
has blogged about LOB
structure internals.

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

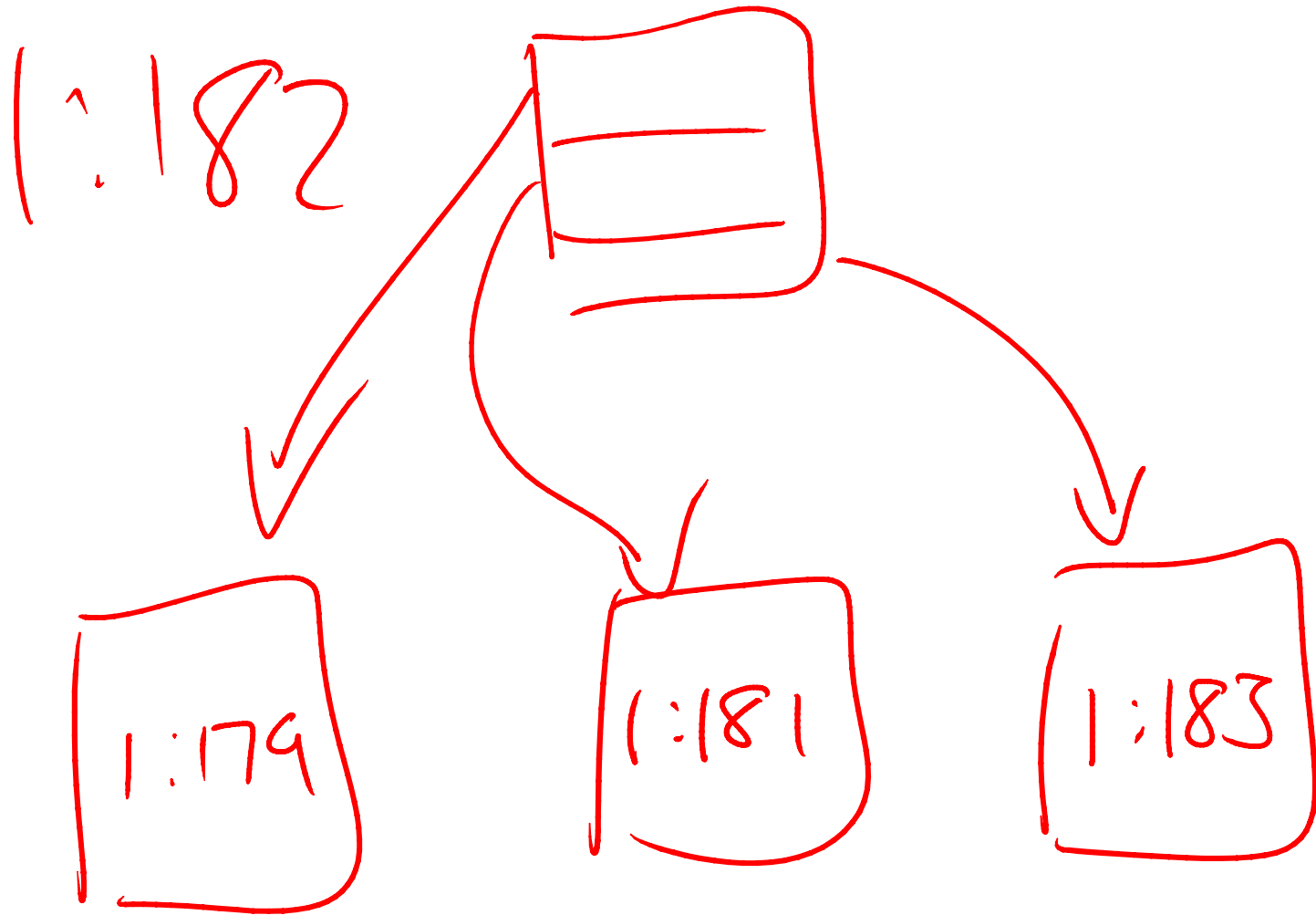


M1S14 & M2S35 Explaining how there's a new portion of the version store created every minute.

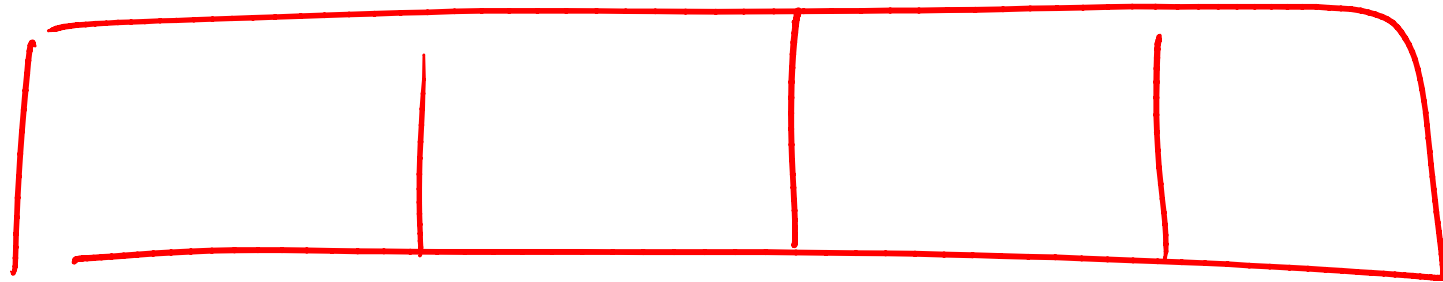




M1S15 When a record is deleted, it's marked as a ghost record. The ghost cleanup task does not physically delete the records – it removes the slot array entry that points to the record on the page. I.e. the record is unmarked as being a record and the area on the page is now free space – that just happens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



M1S22 Showing the clustered index structure generated by the page/record demo.

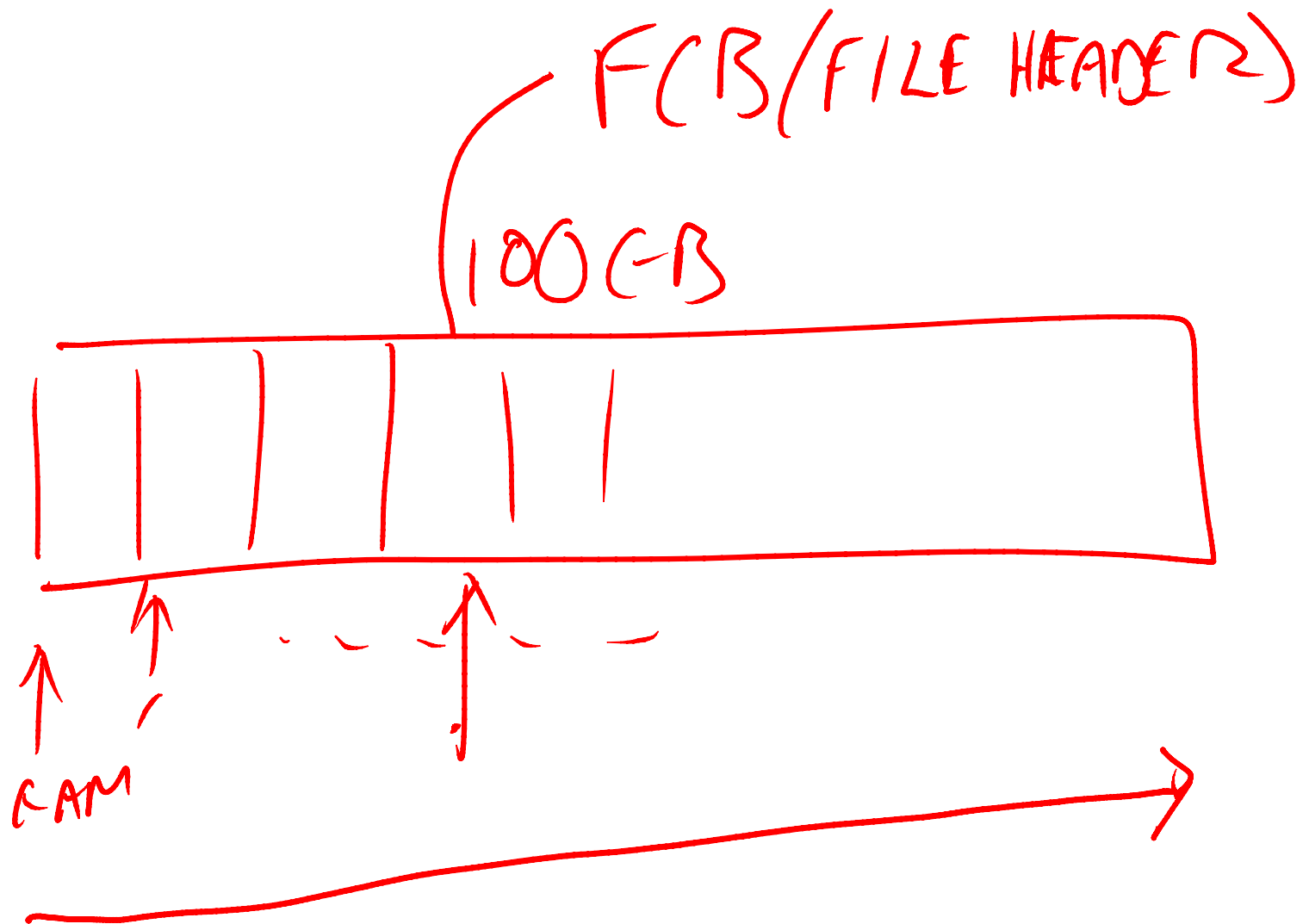


0 — 7 8 — 15 16 — 23

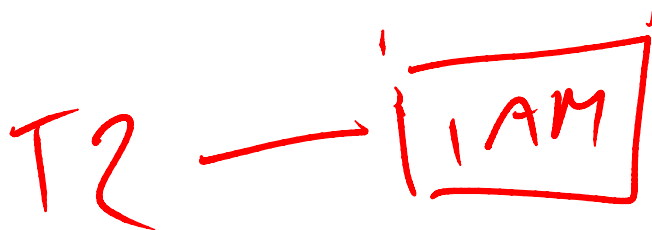


M1S24 Explaining the difference between mixed and dedicated extents. Mixed extent is one in which the 8 pages could be allocated to 8 different objects. Dedicated extent is one where all 8 pages are reserved for exclusive use of a single object.

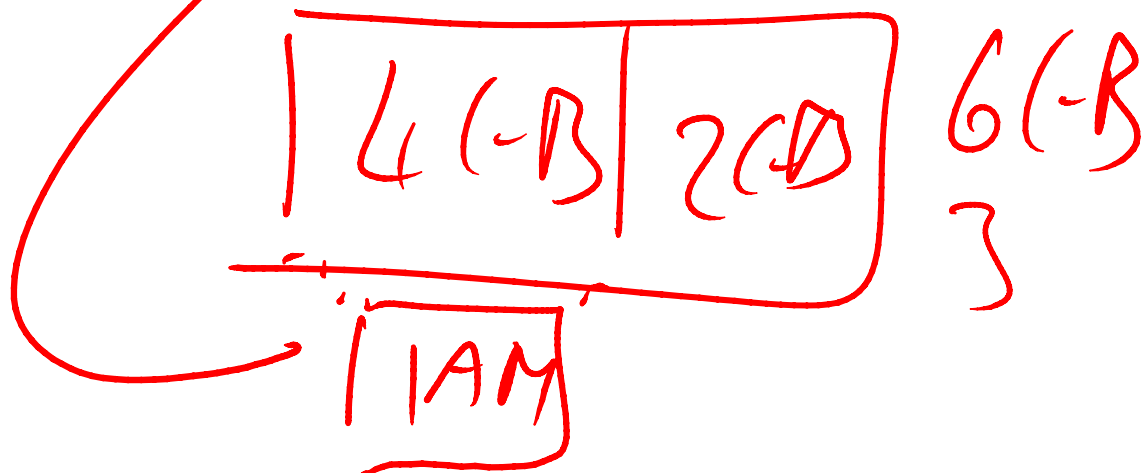
Later in the deck I refine 'object' to really be 'allocation unit'.



M1S30 When a new extent is allocated, if all extents up to the end of the file have not yet been allocated, the next extent to allocate is found from a field in the File Control Block instead of searching through the GAM pages.

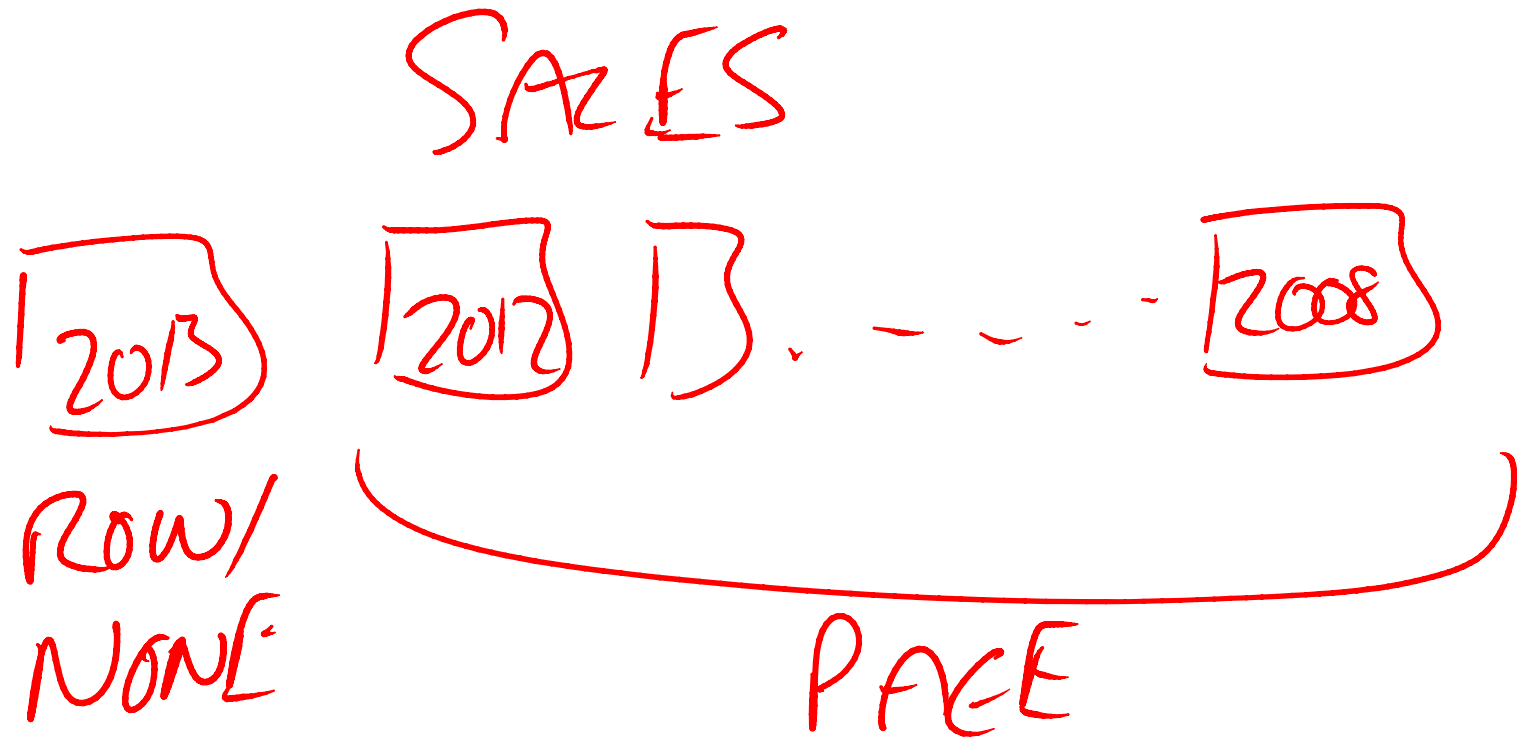


M1S36 Example of IAM chains.
Allocations from each 4GB
chunk of a data file require an
IAM page to track the
allocation. Multiple IAM pages
make an IAM chain.

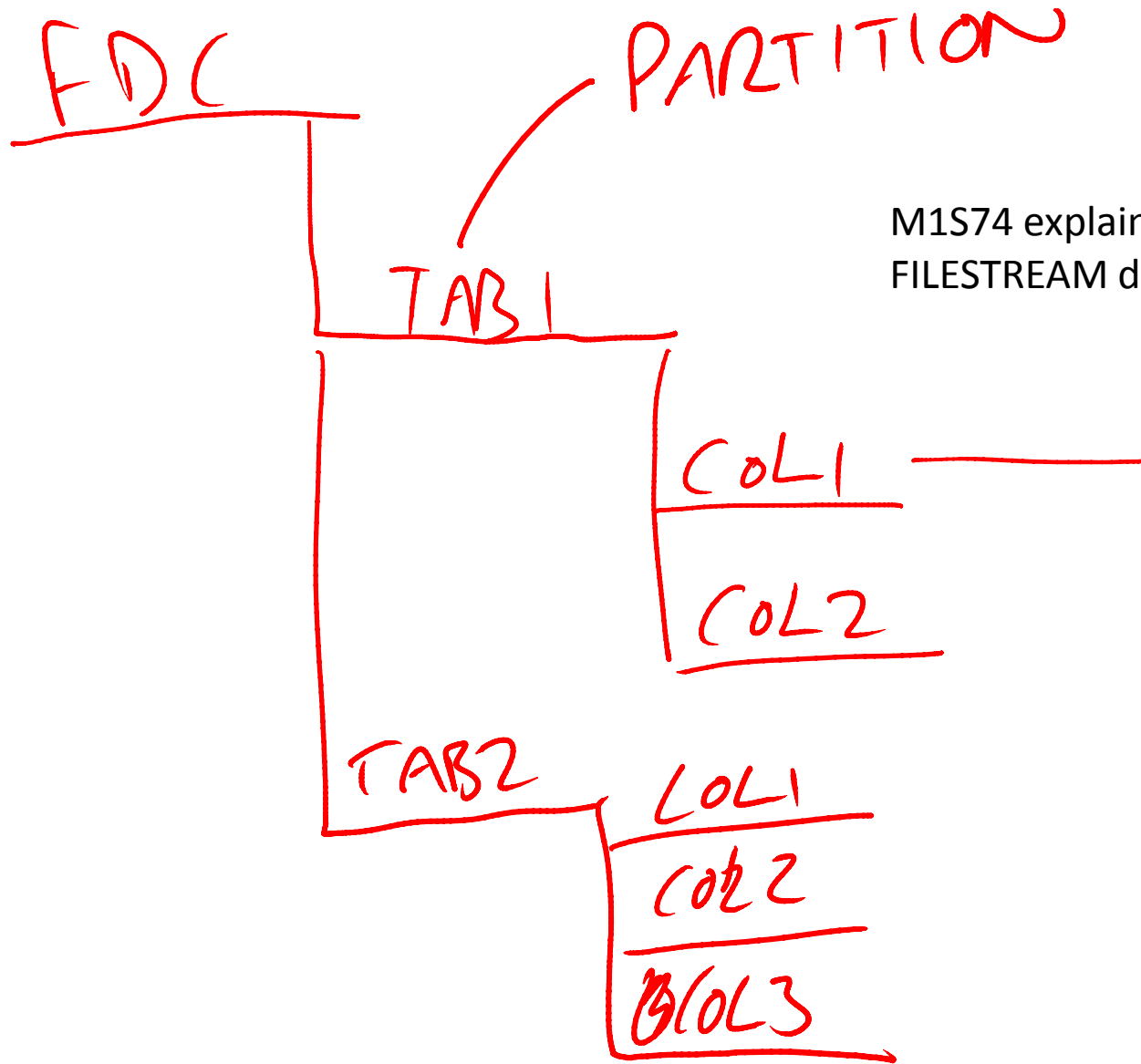


`%%LOCKRES%%`

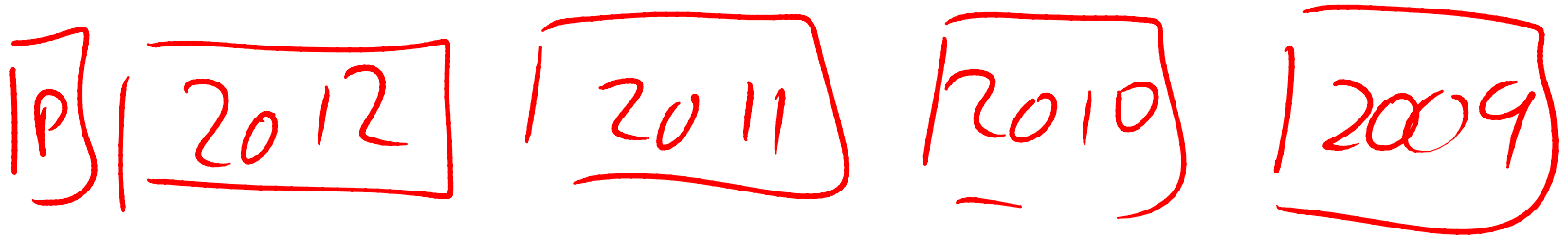
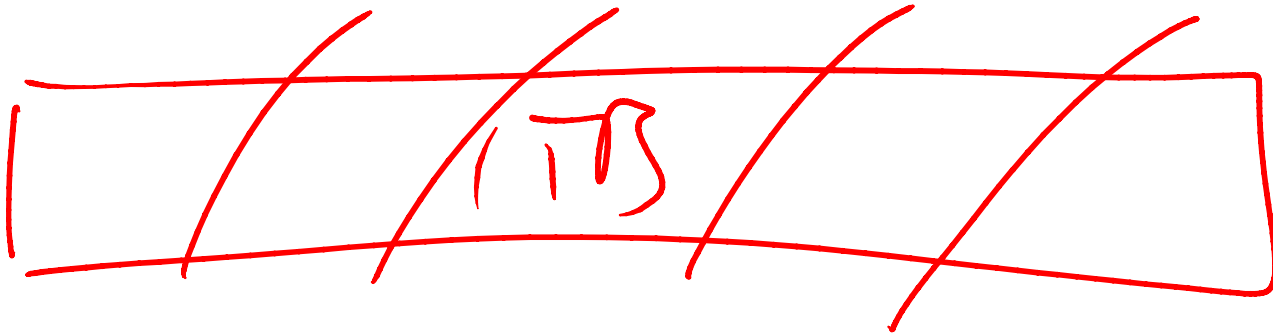
M1 If you want to see what the lock hash resource for records will be inside the lock manager, use the undocumented `%%LOCKRES%%` function.



M1S50 Discussing how different partitions in a table can have different compression settings.

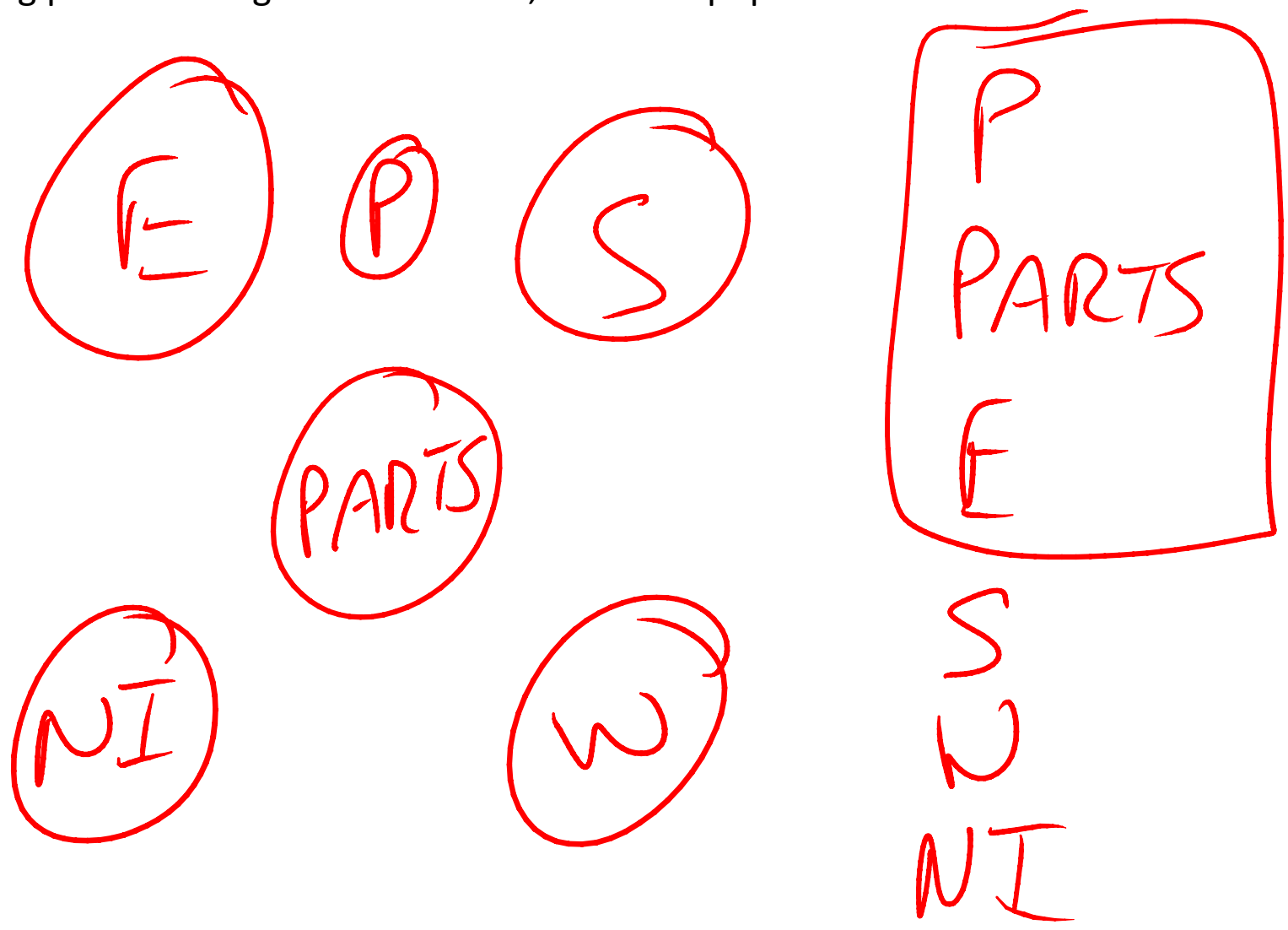


M1S74 explaining the NTFS structure of a FILESTREAM data container



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. Auto parts sales system with countries in the UK – makes sense to bring parts and England online first, based on population size.



2008 → SALES ← 2013

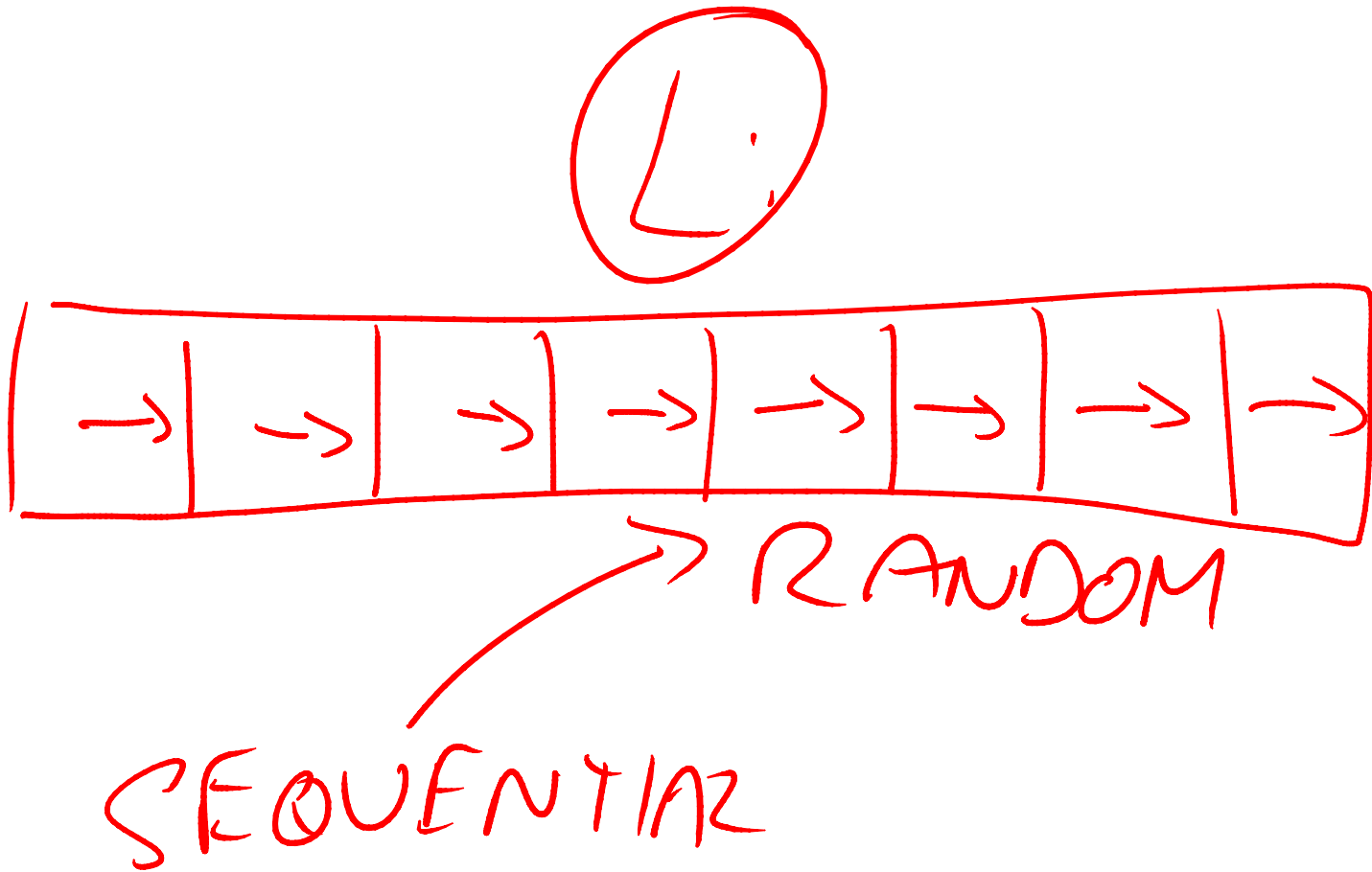
M2S7 Splitting a table into partitions
allowed targeted index maintenance
without operating on the whole table.

REBUILD

REORG

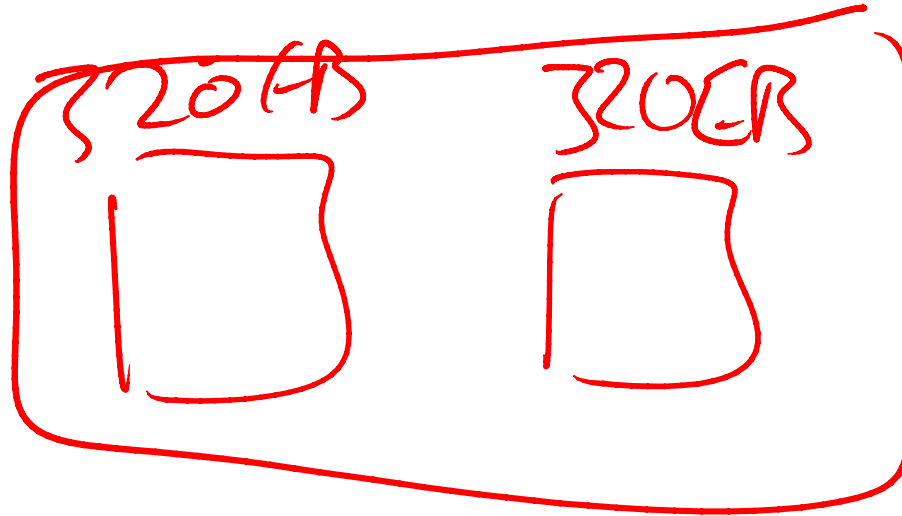
2013

2008

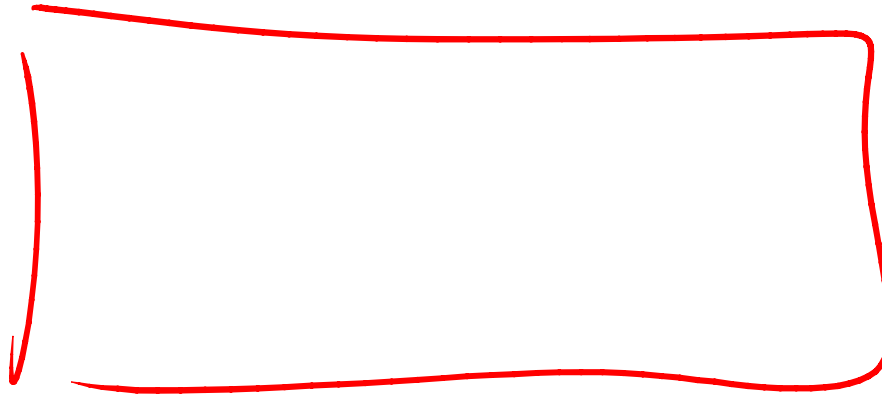


M2S12 Be careful about putting multiple log files, with sequential write workloads, one a single LUN, as that might make the overall I/O workload random, slowing it down.

Fusion-io 640GB



M2S12 If you're using SSDs, you need to have multiple of them to run in at least RAID-1. Some cards present multiple drives (e.g. Fusion-io) but they're on one card, so there's no redundancy.

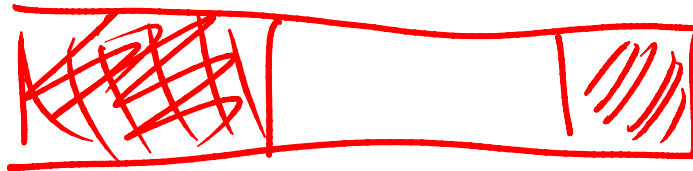


Sys.dm_io_virtual_file_stats

M2 Use this DMV to analyze I/O latencies within SQL Server. See

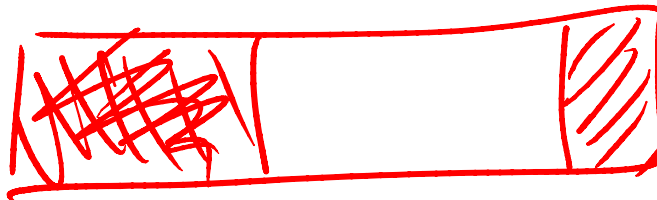
<http://www.sqlskills.com/blogs/paul/how-to-examine-io-subsystem-latencies-from-within-sql-server/>

RR

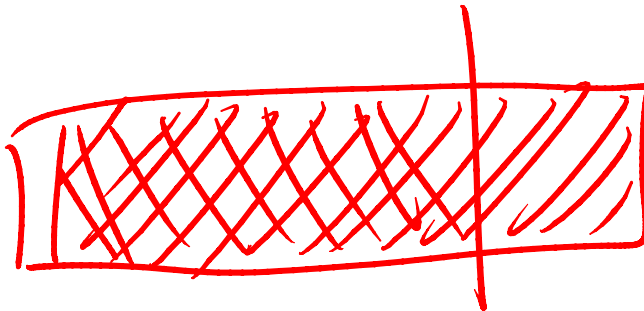


PROP FILL
WEIGHTING

~~8100~~
SO (8099)

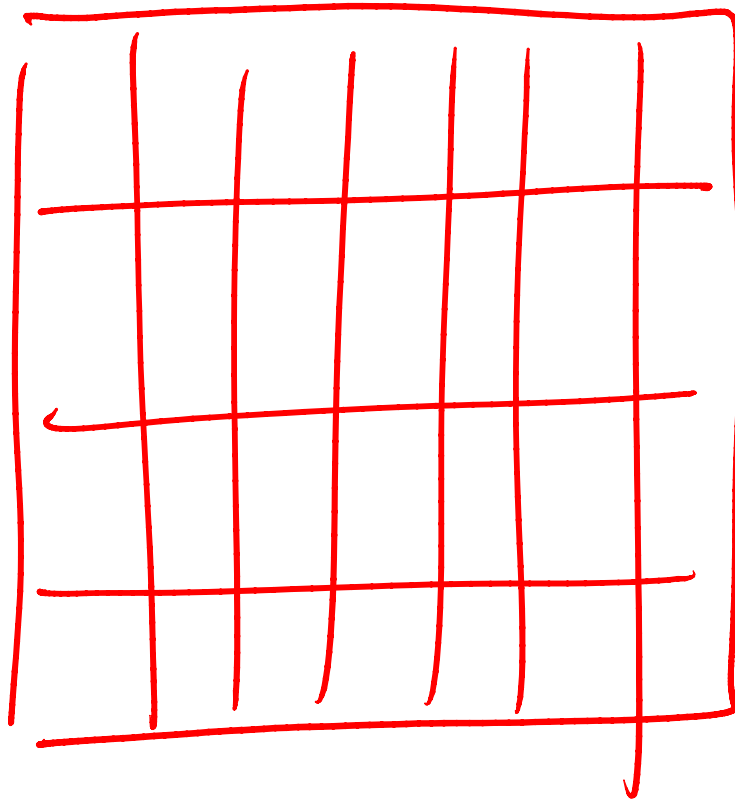


~~8100~~
SO (8099)

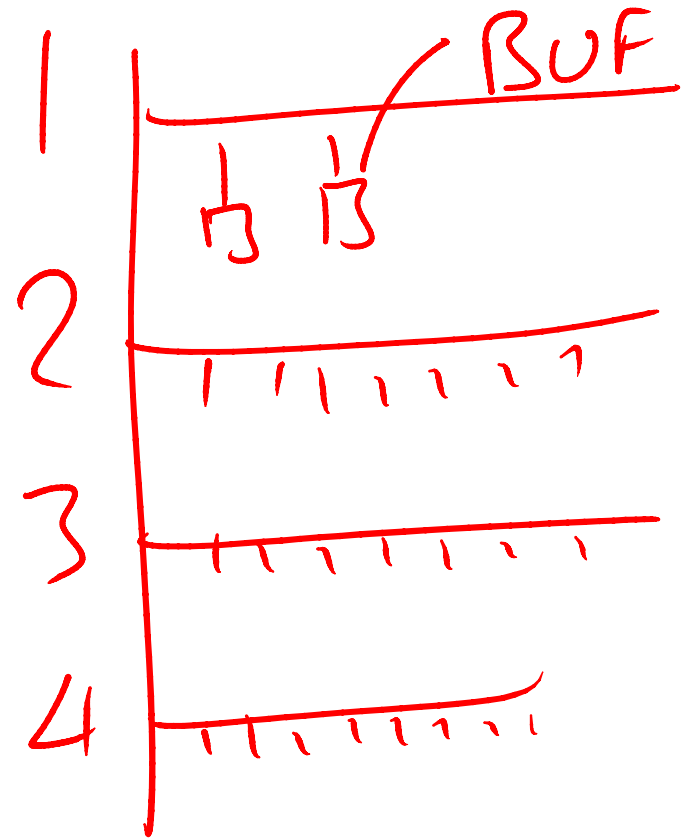


1
8100 ← ✓

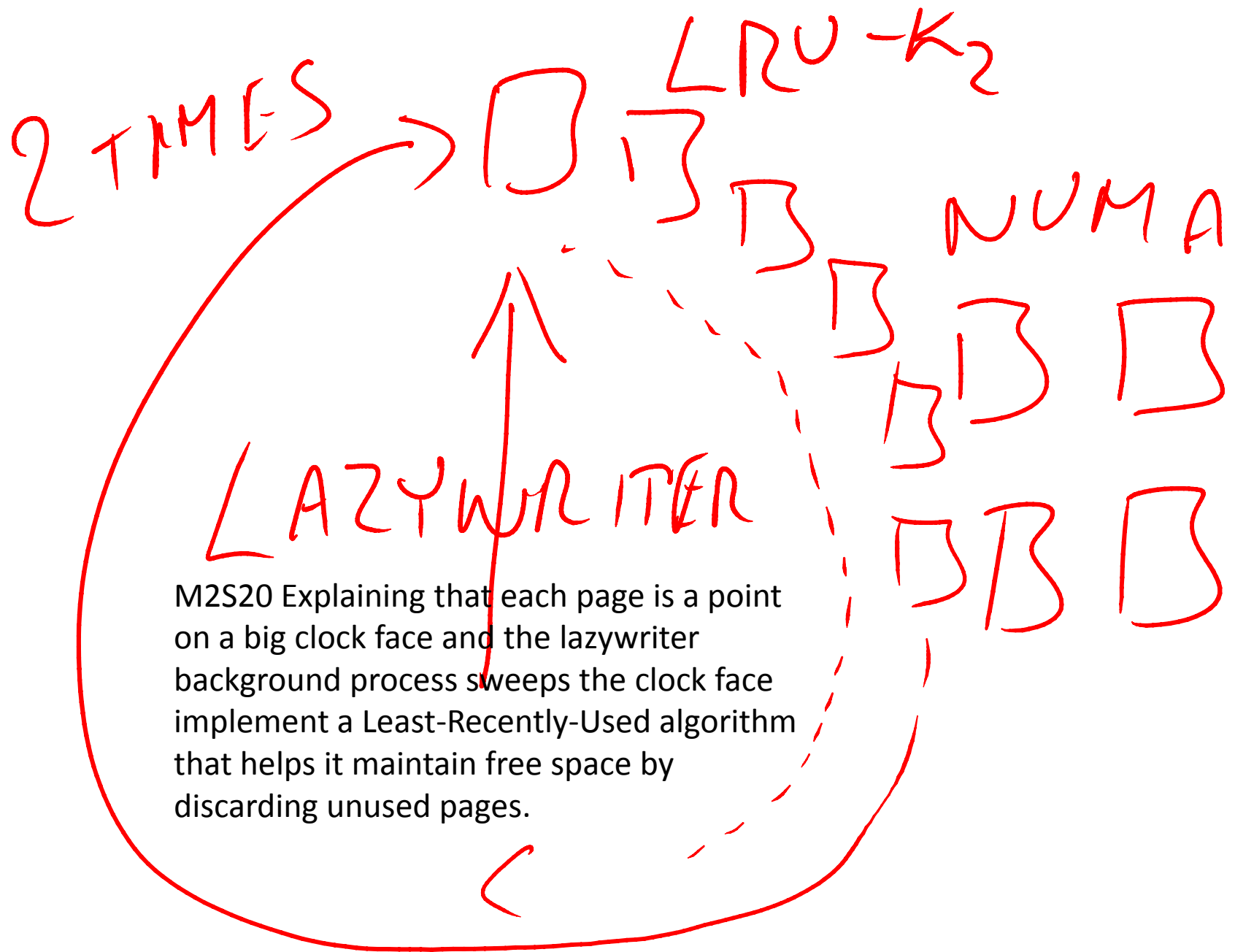
M2S19 Explaining round-robin
allocation and proportional fill

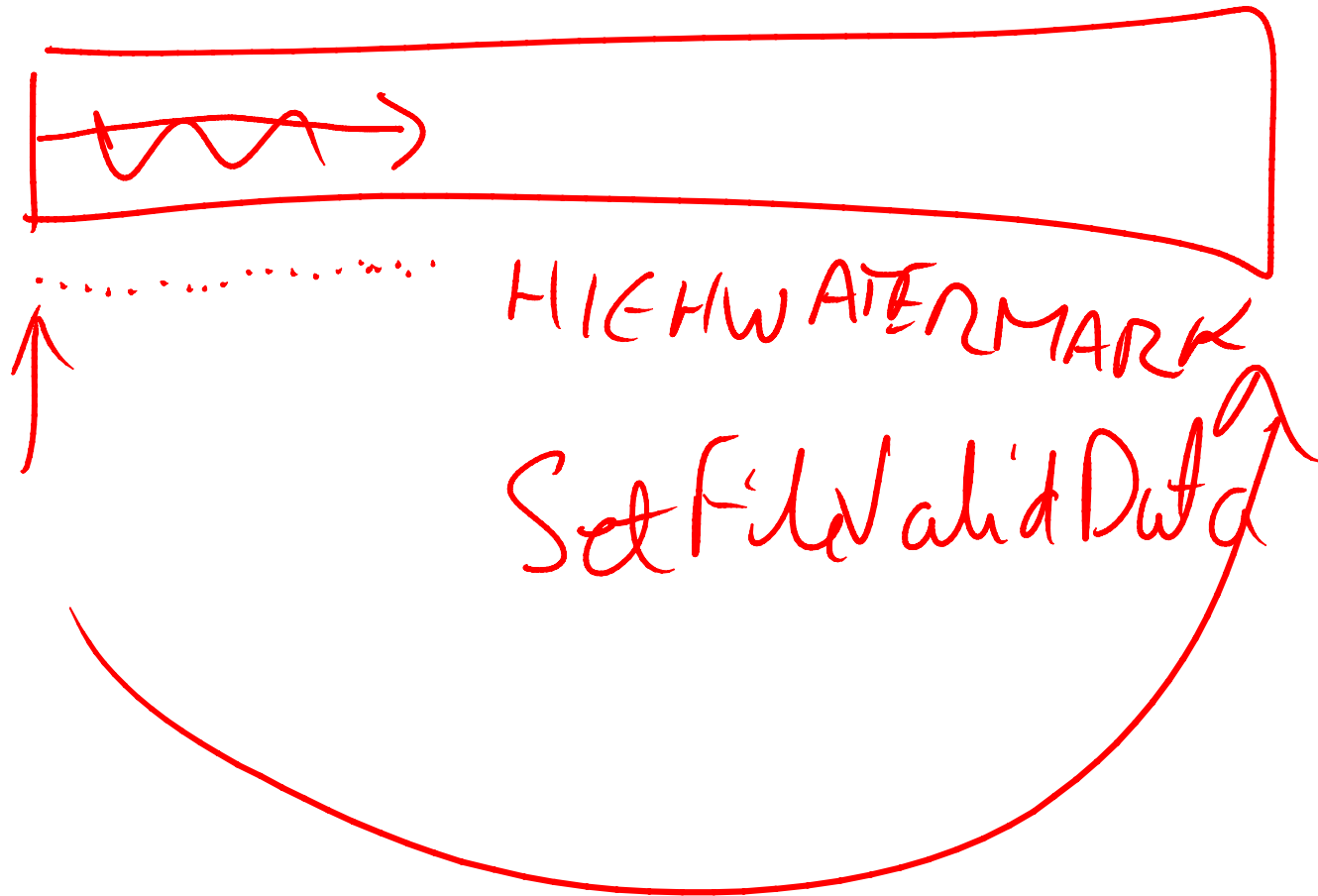


HASH LISTS



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



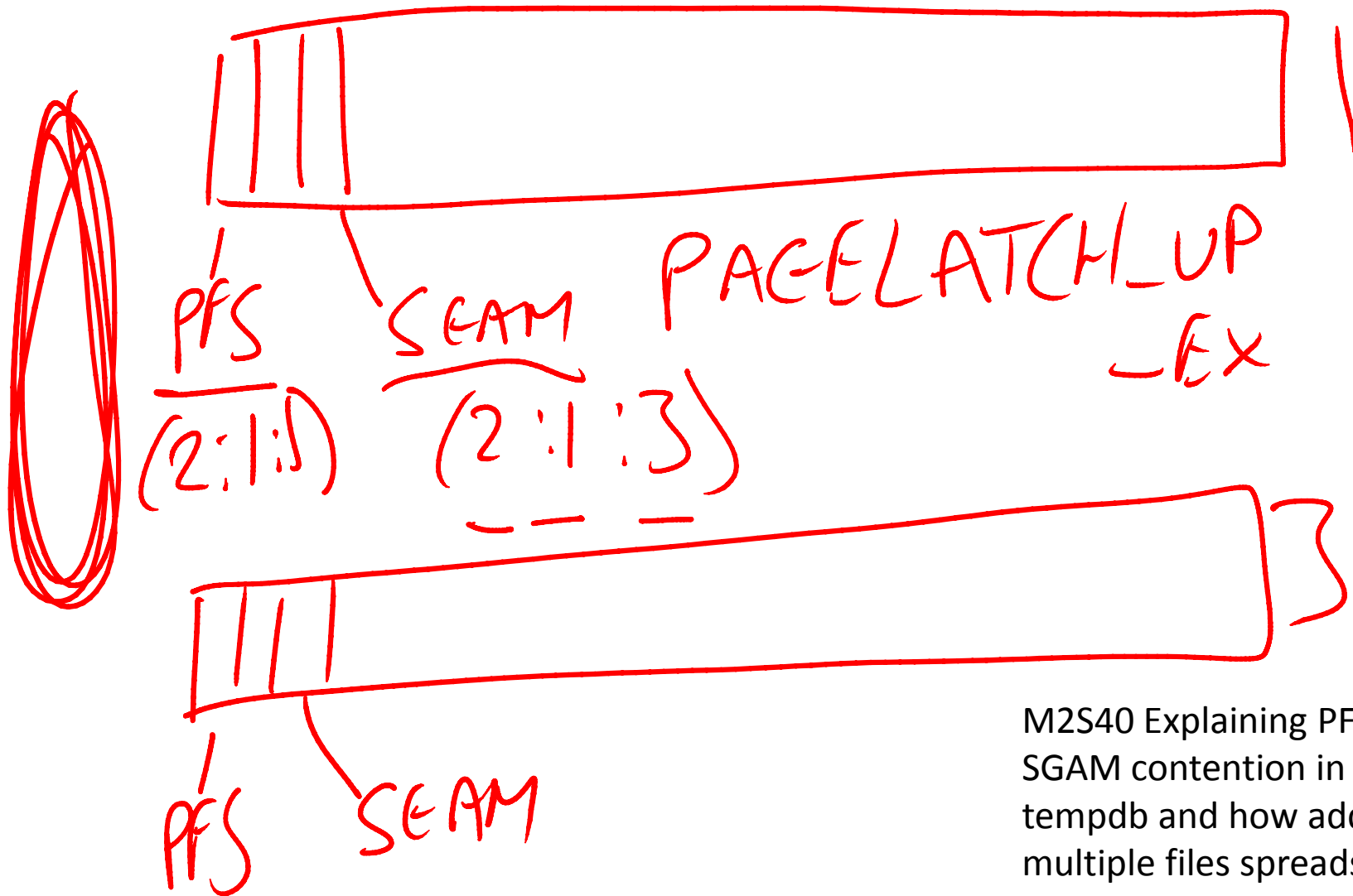


M2S22 Explaining zero initialization and how NTFS does not trust newly allocated space until the highwatermark of the file has increased, either through consecutive writes, or setting the HWM through a called to SetFileValidData.

SQL ~~SECURE FILES~~

DBCC PAGE (2)

M2S24 Explaining how the security flaw with instant initialization works. Secure files are deleted, SQL files expand and pick up the disk space without zeroing it out. SA using DBCC PAGE can look at the raw bytes of the disk space now included in the data file, potentially leading to information leakage.



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

2 CPU CPUs

12-CORES PHYSICAL CORES

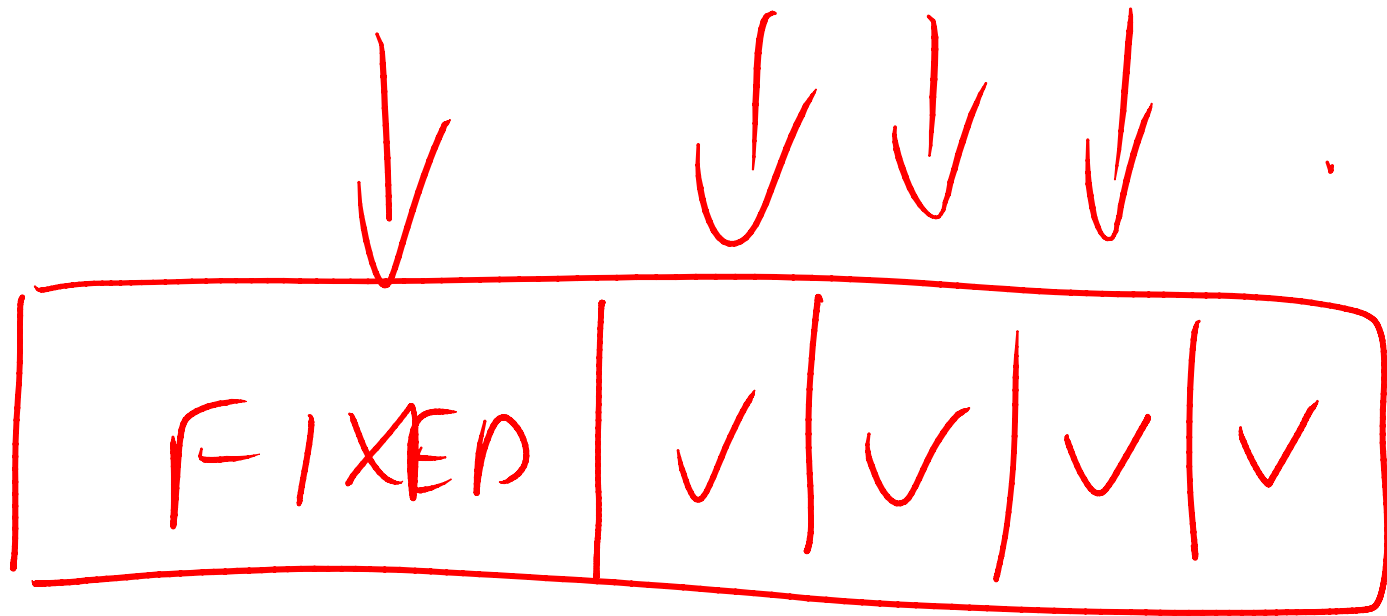
Explaining how a CPU contains physical cores. A server with two CPUs, each with 12 cores, will have 24 physical cores. If hyperthreading is enabled, that becomes 48 logical cores.

24-CORES

WIP → 48-CORES LOGICAL CORES

2537

M5S18 TF2537 allows fn_dblog to examine all possible transaction log, not just the active portion of the log.

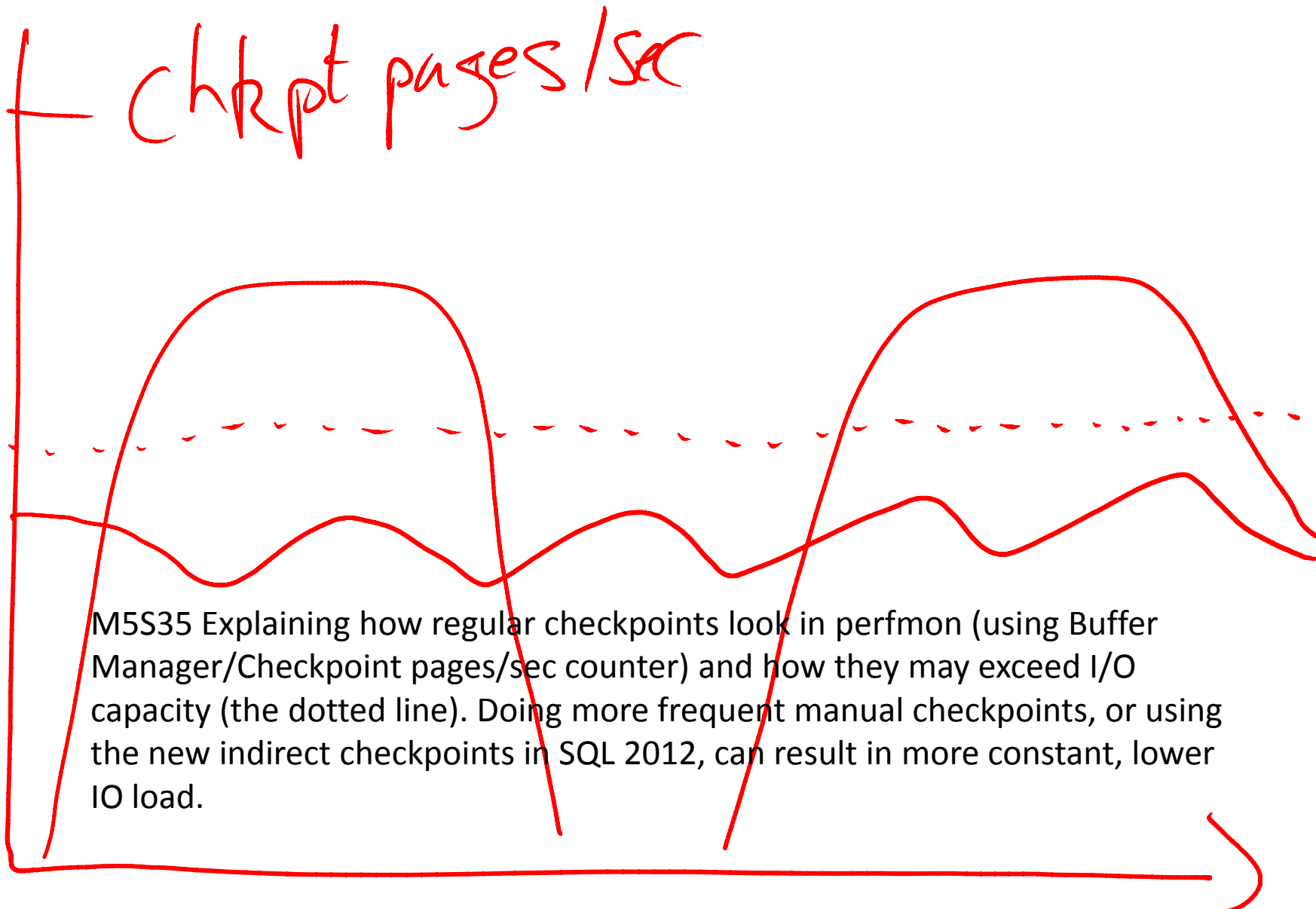


LOP_MODIFY_ROW

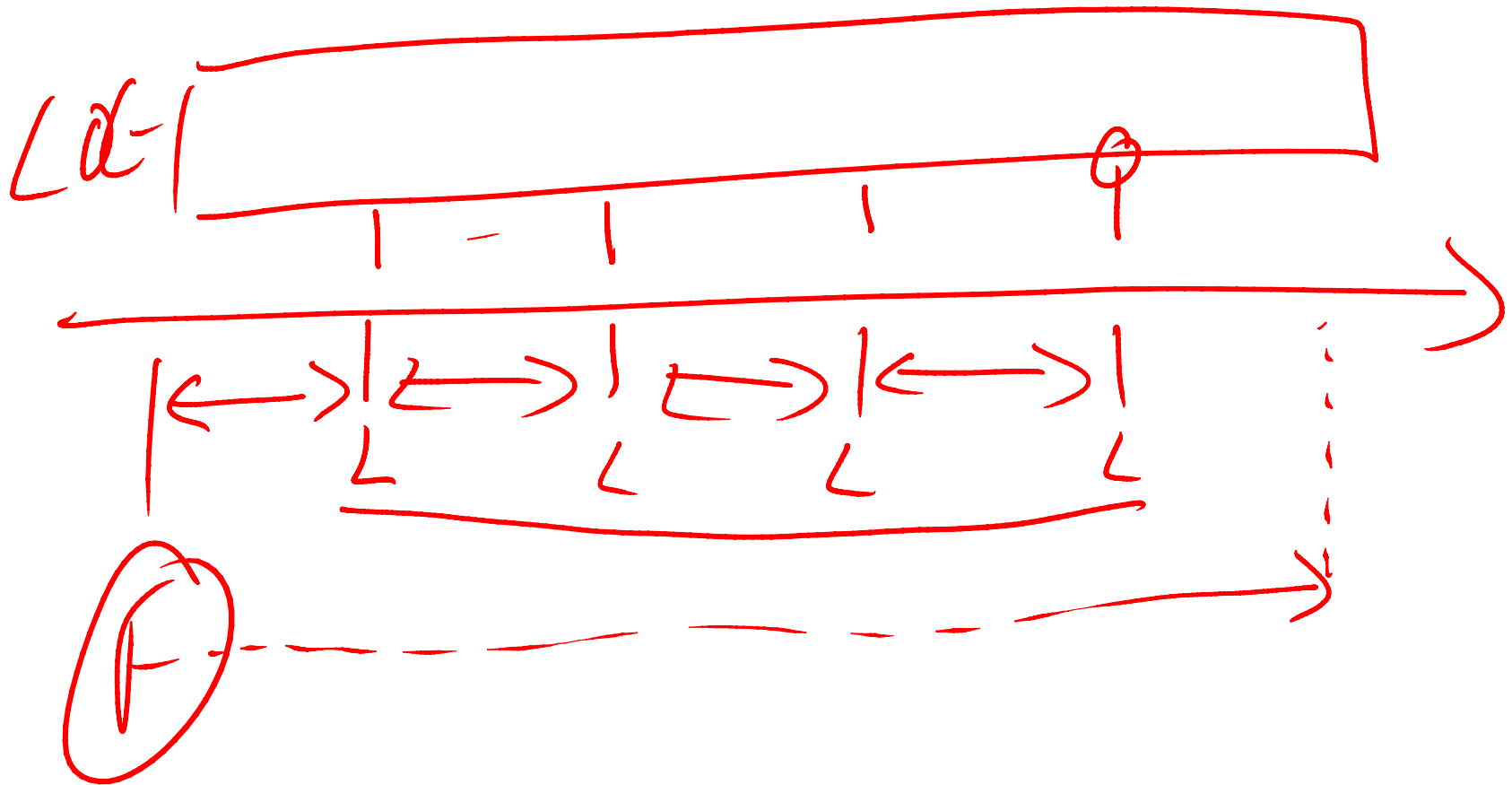
LOP_MODIFY_COLUMN

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)

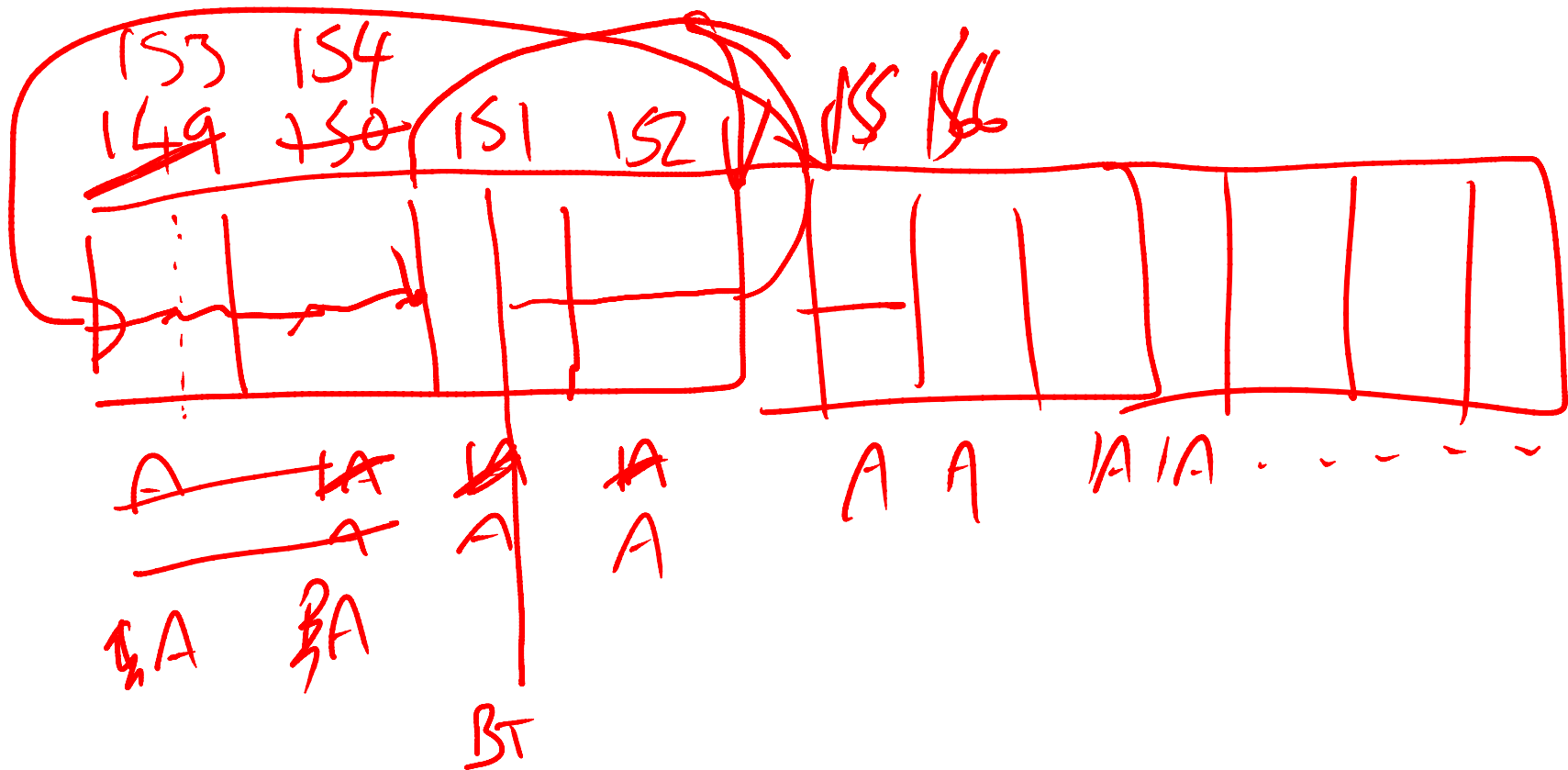
chkpt pages/sec



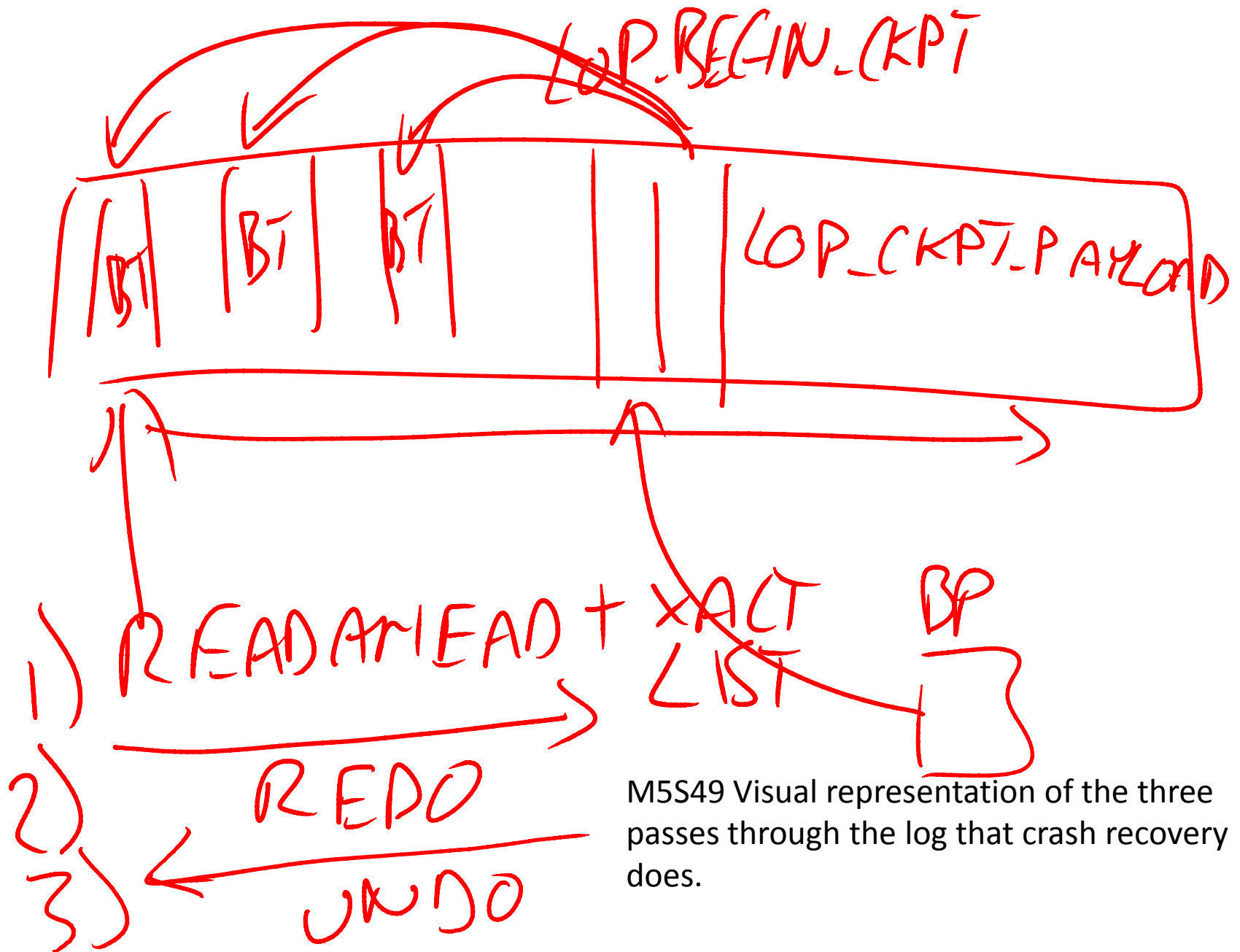
M5S35 Explaining how regular checkpoints look in perfmon (using Buffer Manager/Checkpoint pages/sec counter) and how they may exceed I/O capacity (the dotted line). Doing more frequent manual checkpoints, or using the new indirect checkpoints in SQL 2012, can result in more constant, lower IO load.



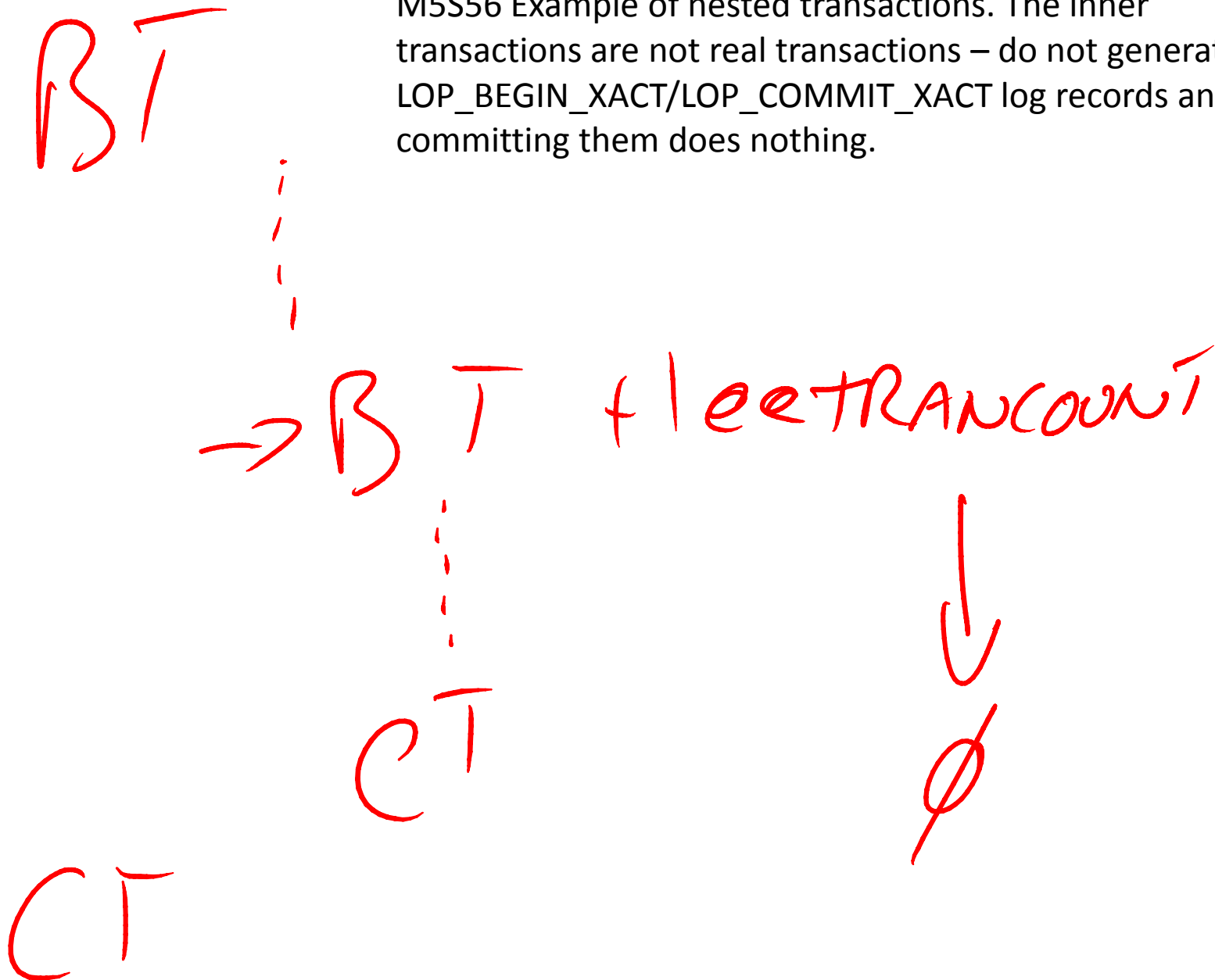
M5S42 Explaining that if log backups happen concurrently with a data backup, log clearing cannot occur. When the data backup finishes, the log can be cleared up to the point of the most recent concurrent log backup – called deferred log clearing. It appears that the data backup cleared the log, but it didn't.



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.



M5556 Example of nested transactions. The inner transactions are not real transactions – do not generate LOP_BEGIN_XACT/LOP_COMMIT_XACT log records and committing them does nothing.



100,000

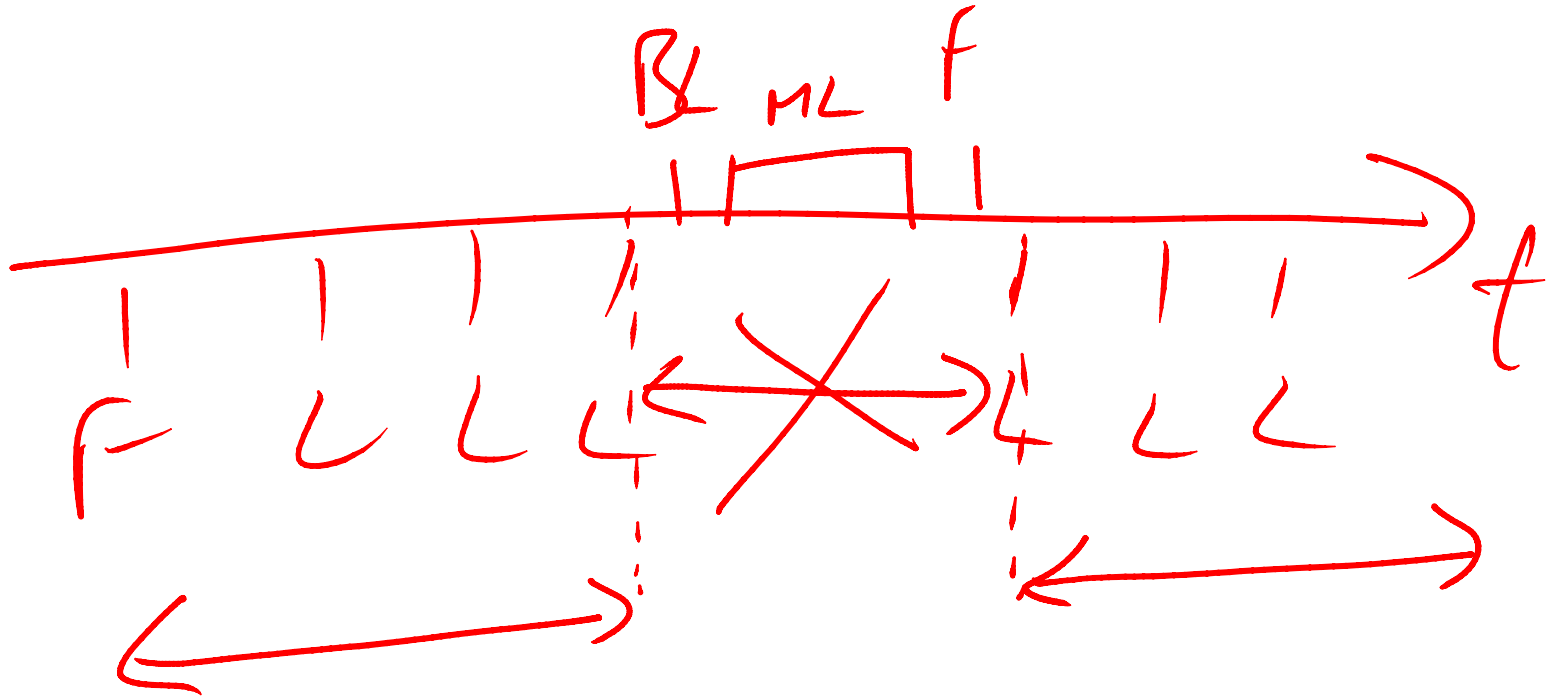
10

M5S60 Explaining that full recovery doesn't mean there's always one log record per operation. An index rebuild in full recovery will log full LOP_FORMAT_PAGE log records instead of multiple LOP_INSERT_ROWS log records. It's more efficient, but is still fully logged.

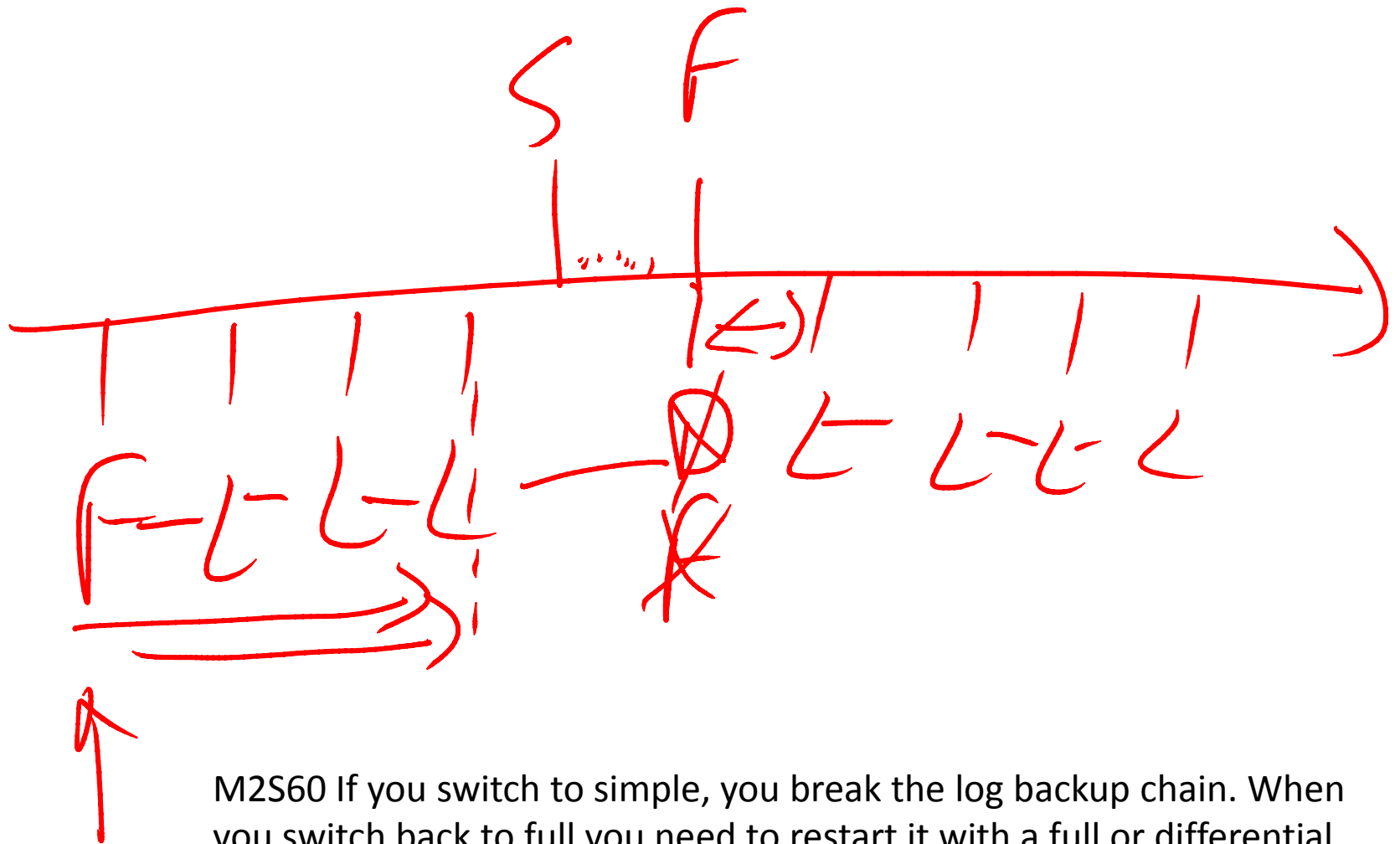
~~INS
INS
INS~~

F - PAGE
F - PAGE

10



M5S60 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.

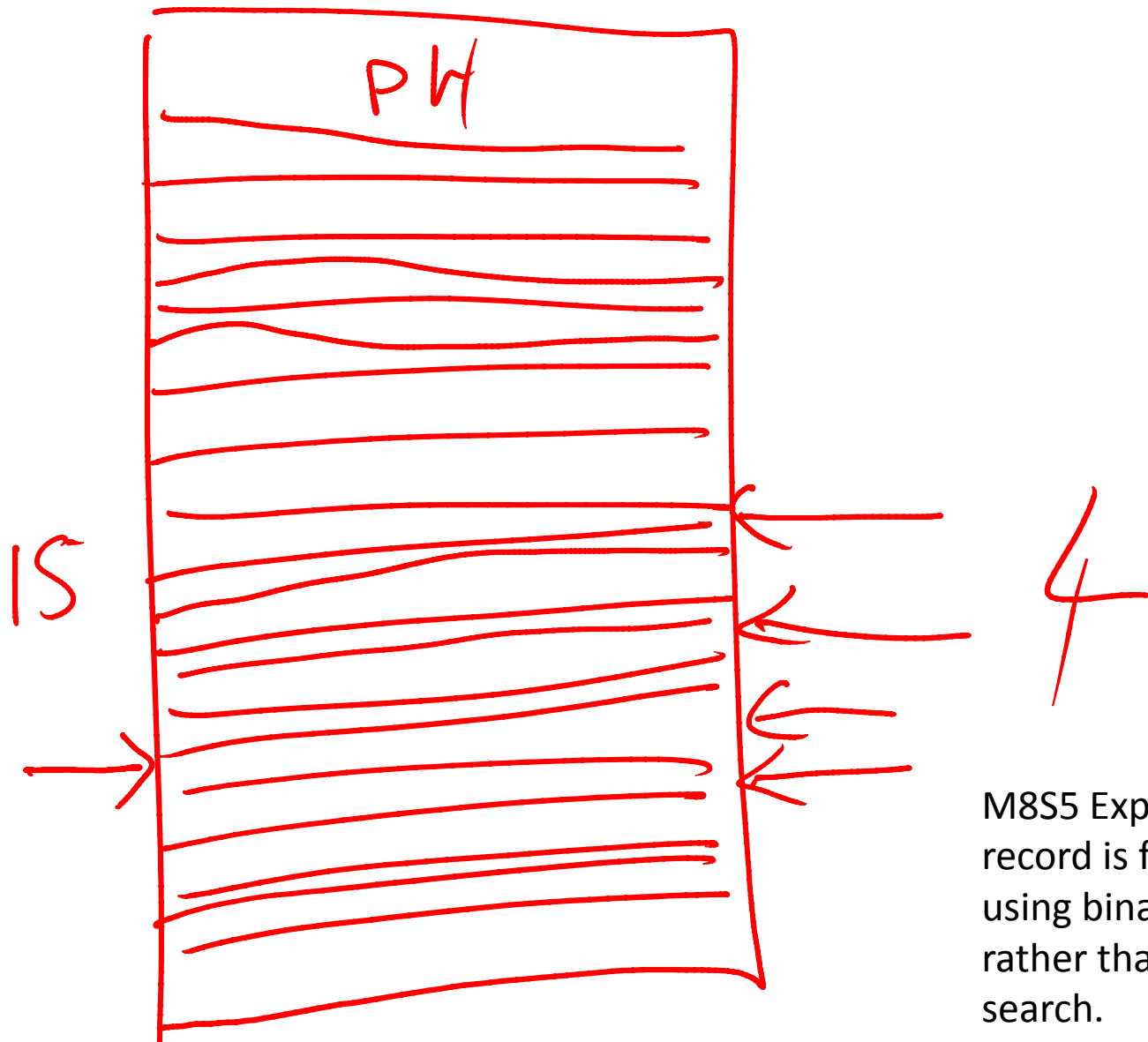


M2S60 If you switch to simple, you break the log backup chain. When you switch back to full you need to restart it with a full or differential backup.



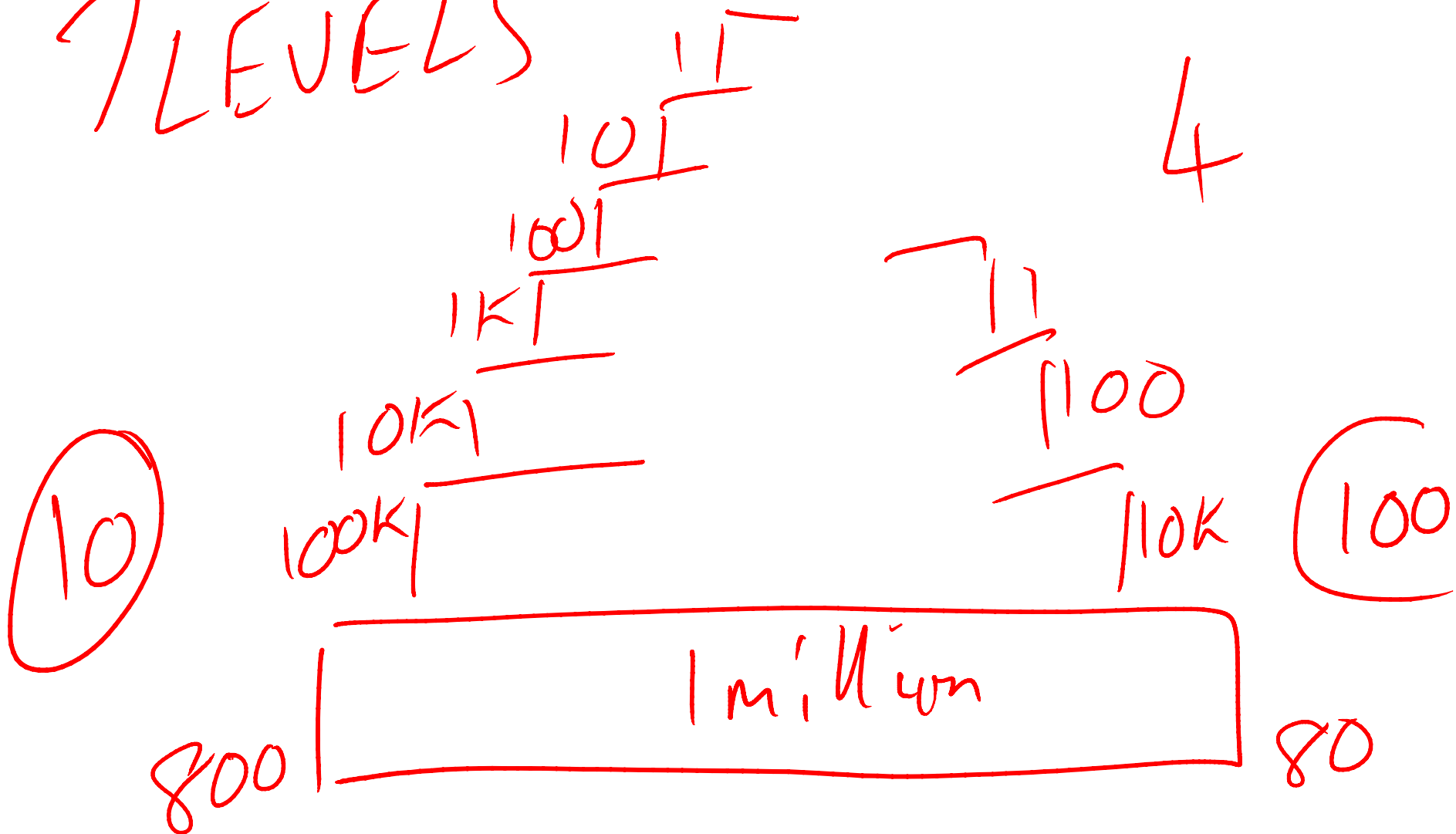
M5S25 VLF fragmentation is bad because of having to search through the VLFs for a particular VLF sequence number. See the KB article link for more info.

LSN LSN
VLF

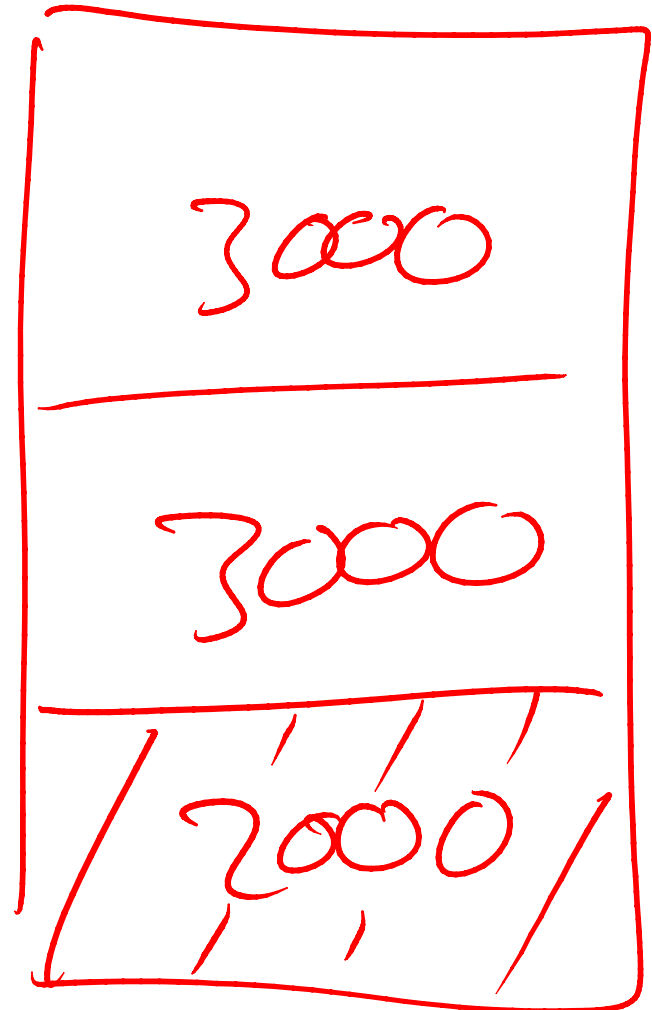
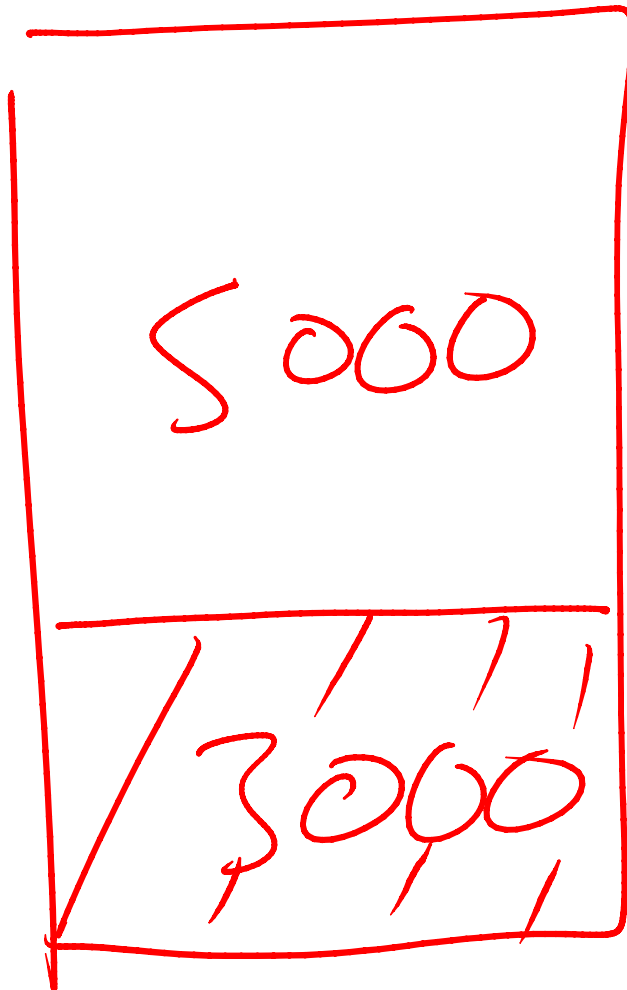


M8S5 Explaining how a record is found on a page using binary search rather than brute force search.

7 LEVELS



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.

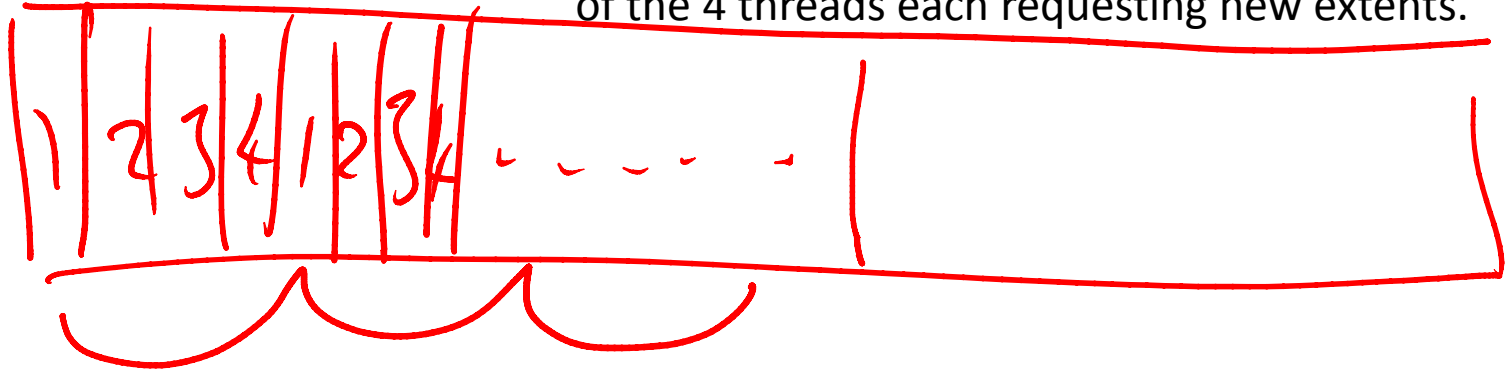


M8S13 Explaining how fixed-size large rows leads to low page density and wasted space.

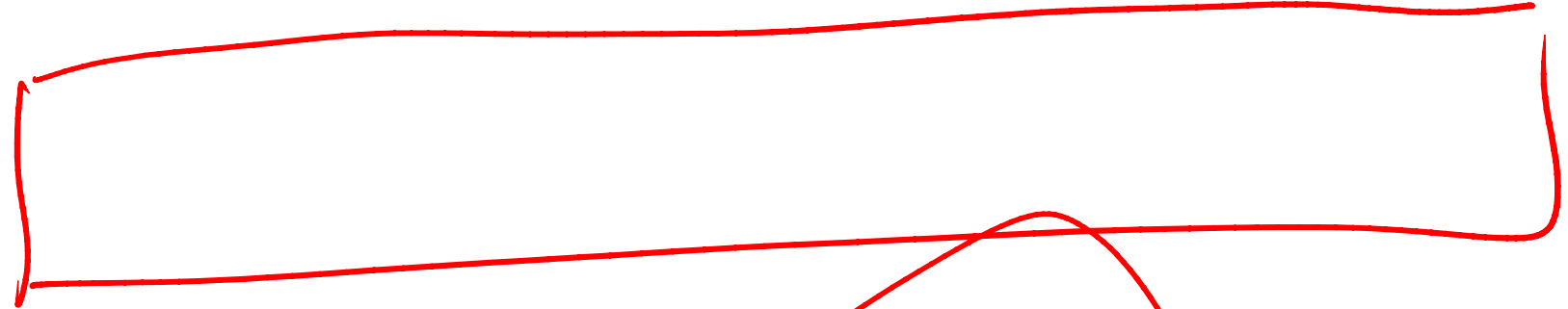
-E

64

M8S14 Explaining index rebuild DOP 4 after using -E will not lead to large contiguous index ranges because of the 4 threads each requesting new extents.

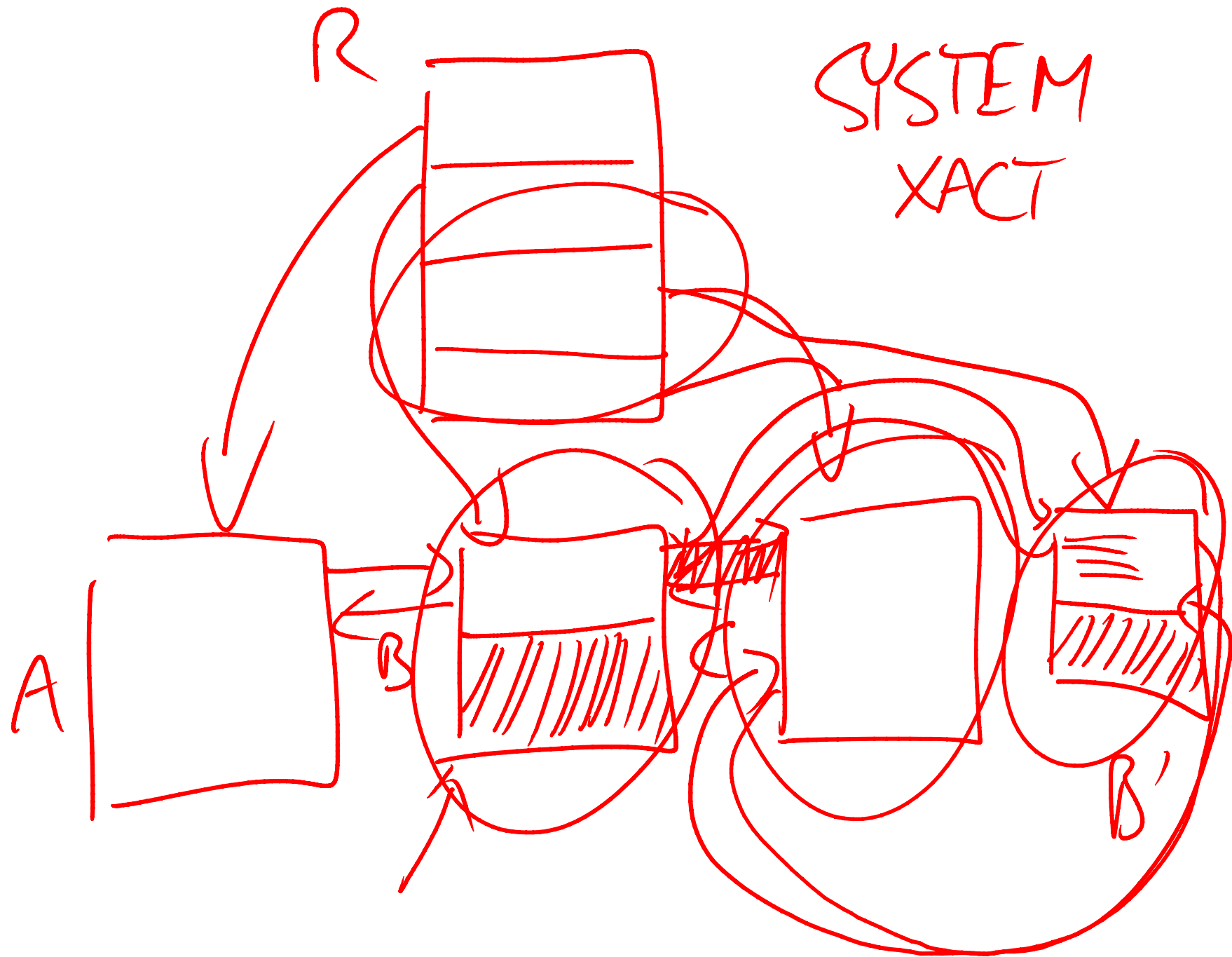


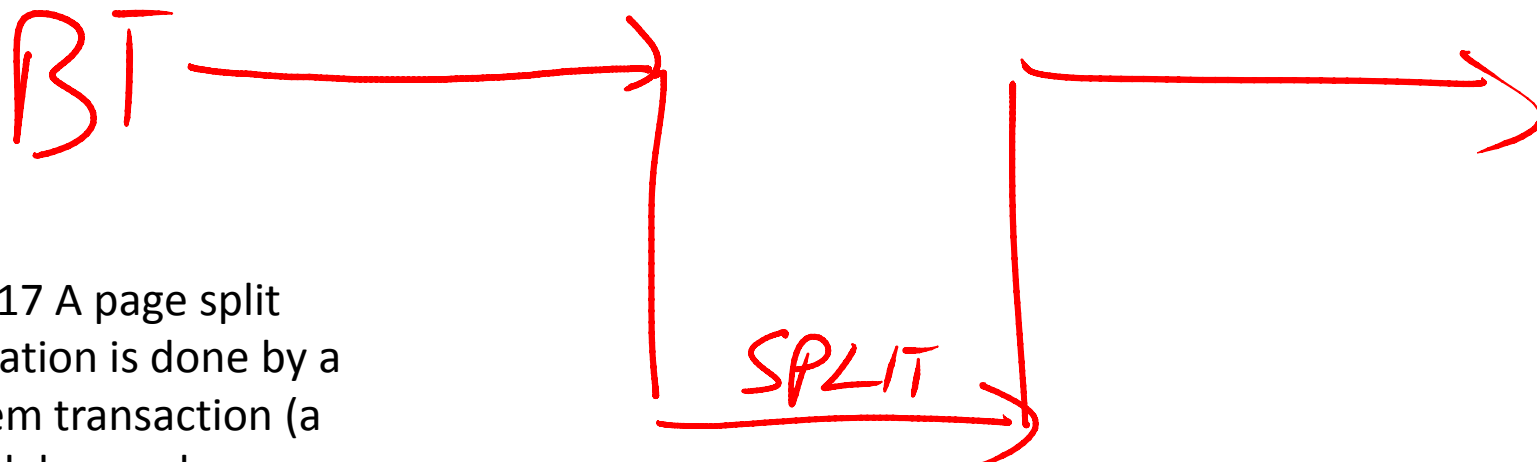
→



DOP = 4

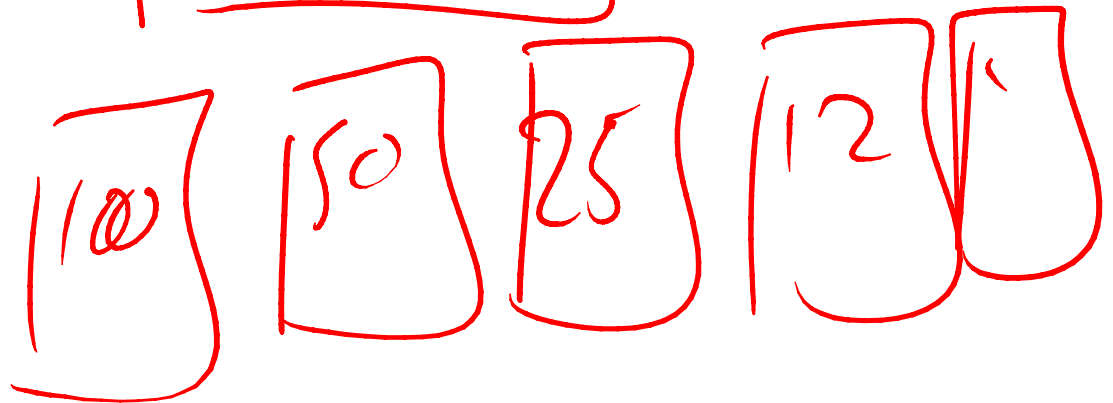
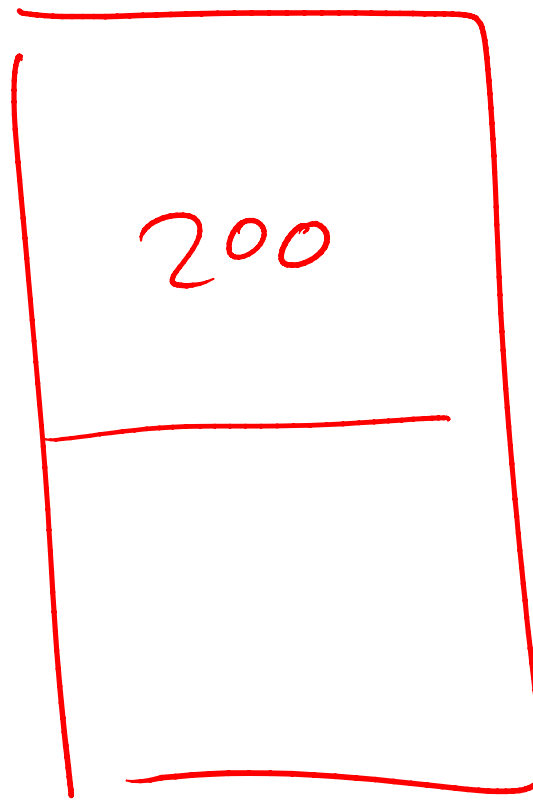
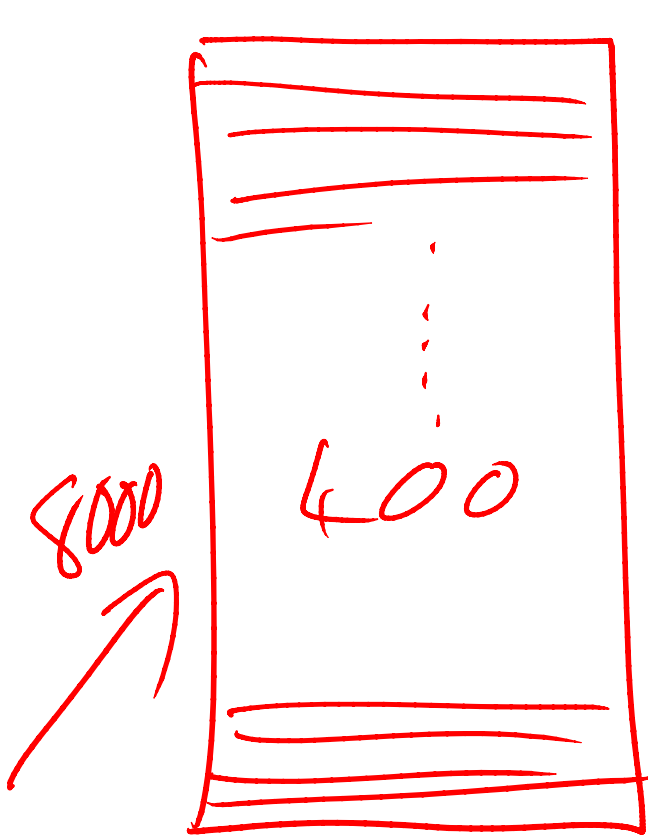
Four overlapping circles, each containing a number 4, representing the Degree of Parallelism (DOP) for the index rebuild.





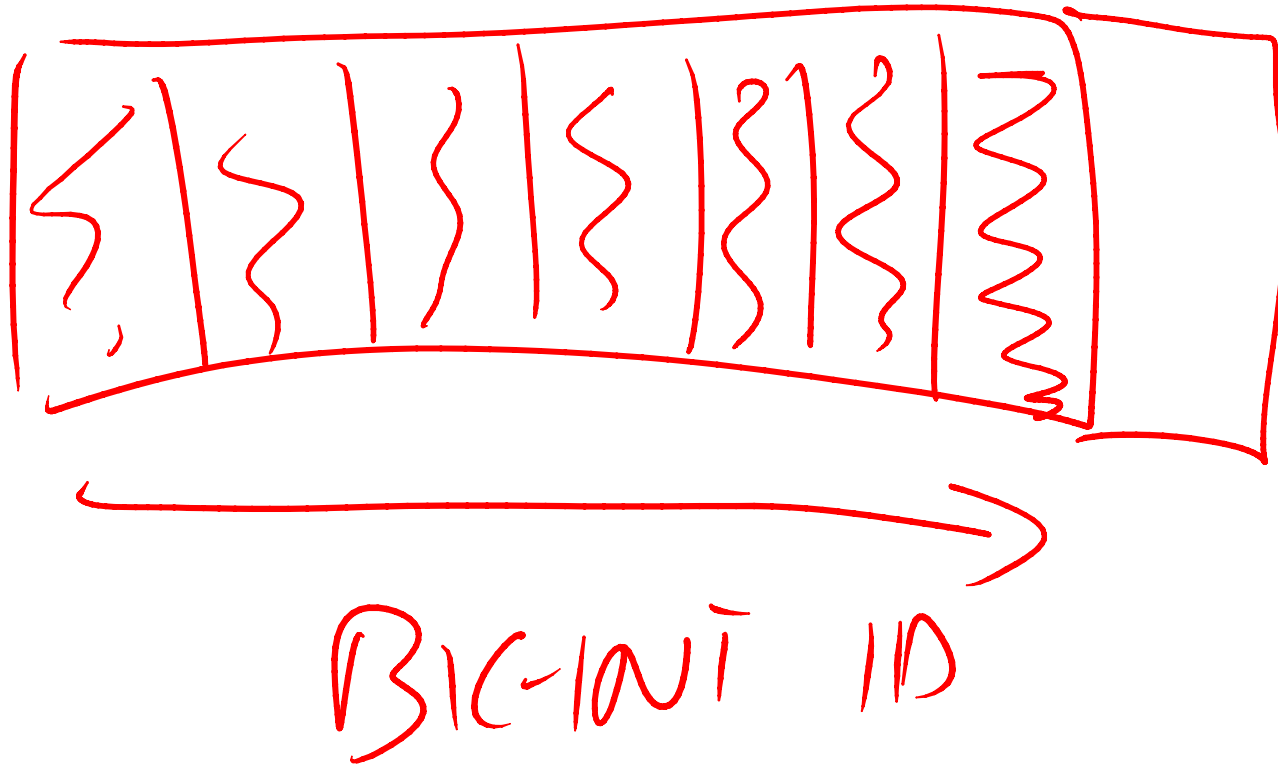
M8S17 A page split operation is done by a system transaction (a standalone sub-transaction from your transaction). It is committed automatically and will not roll back if your transaction rolls back.

BT
✓ SYSTEM
XACT



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

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.

M8S22 The numbers from the page split logging cost demo.

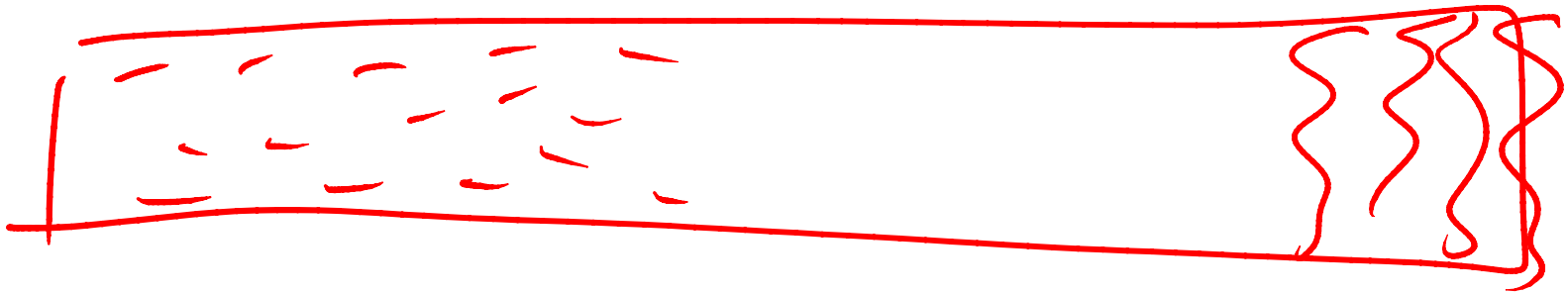
	NO SPLIT	SPLIT	
1000	1244	6772	<u>x5</u>
100	344	5964	<u><u>x18</u></u>
10	256	10364	<u><u>x45</u></u>

COMMENTS

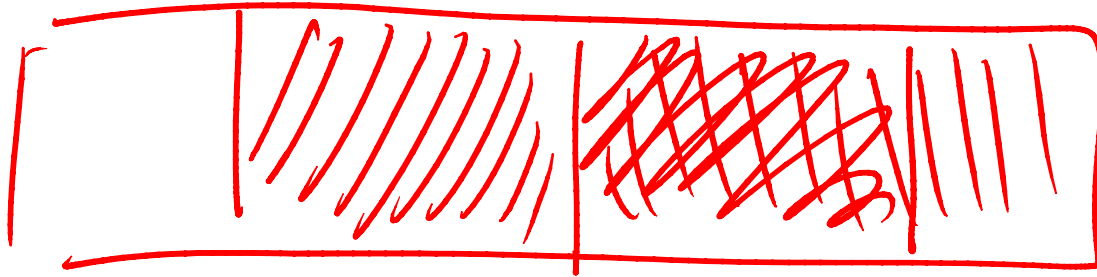
MEMBERID

PAUL	NILS		MALCOLM
	↑	↑K	↑

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

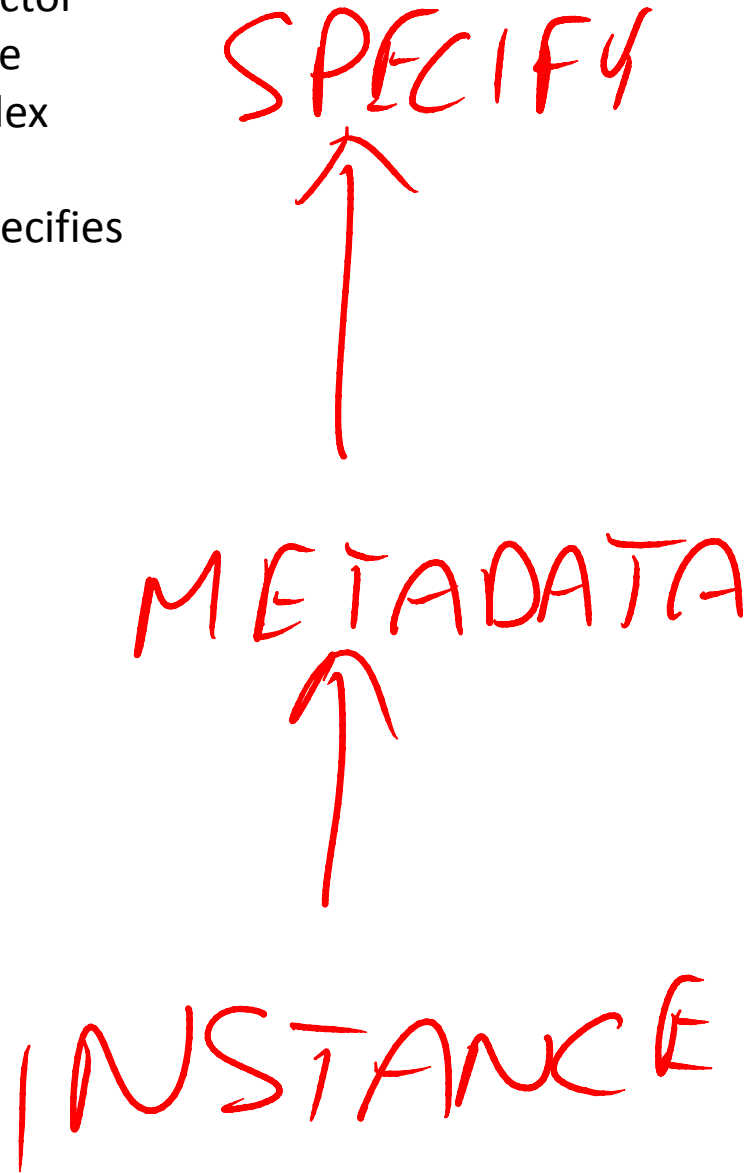


M8S24 Deletes can caused page density 'fragmentation' with an append-only insert pattern. The space created by the deletes will never be reused by new records, creating empty space and low page density.



M8 If you rebuild an index, you're not guaranteed to improve fragmentation. If the free space in the filegroup is very fragmented, the rebuild will use that free space, creating a fragmented index again.

M8S39 The instance fillfactor will be overridden by the fillfactor stored in the index metadata, which will be overridden if a rebuild specifies a fillfactor.



ITB



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