

Ask Anything Performance

(3:15 PM - 4:15 PM)

SQLintersection Speaker Panel

Brent Ozar, BrentO@BrentOzar.com

Paul Randal, Paul@SQLskills.com

Kimberly L. Tripp, Kimberly@SQLskills.com

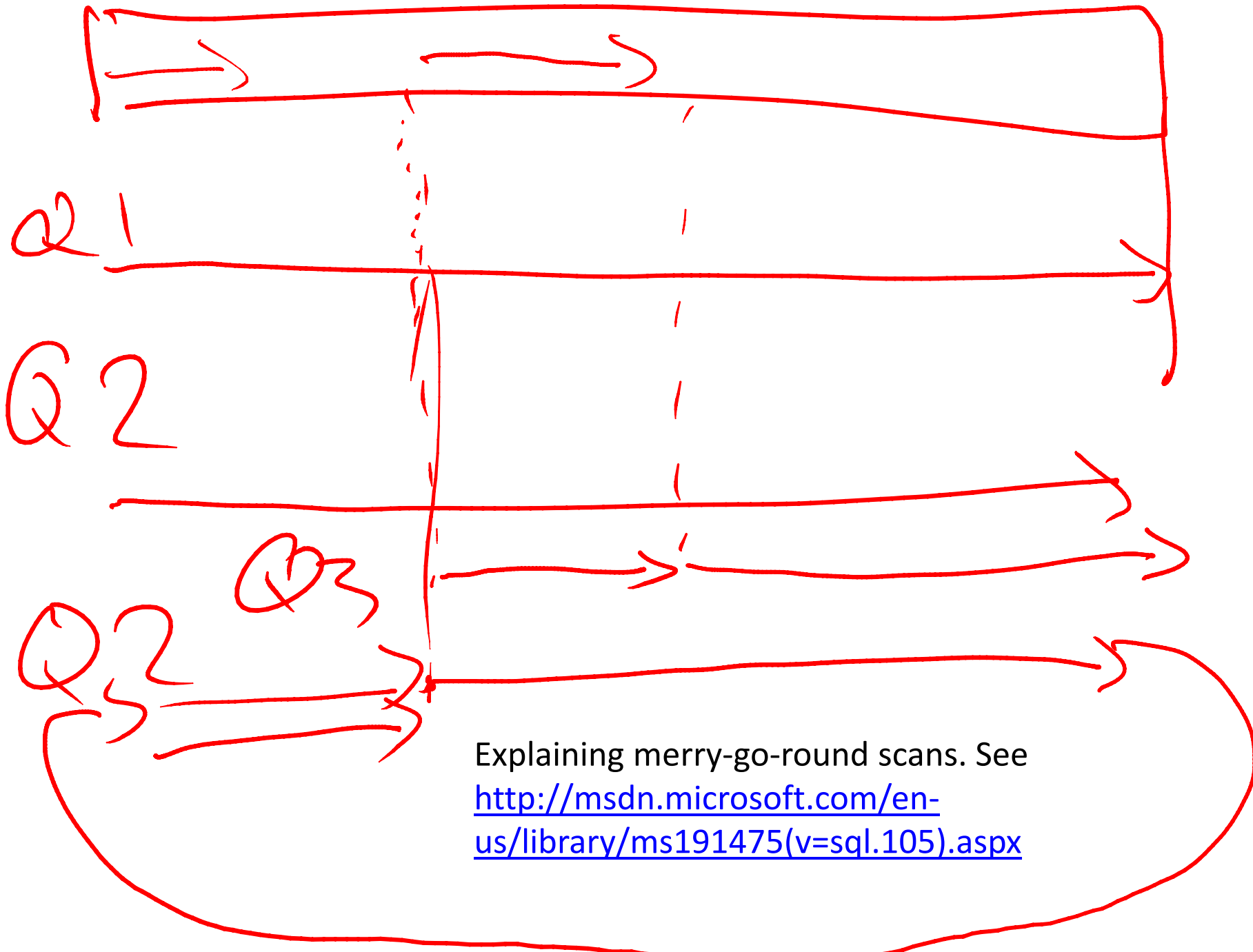


Potential Topics

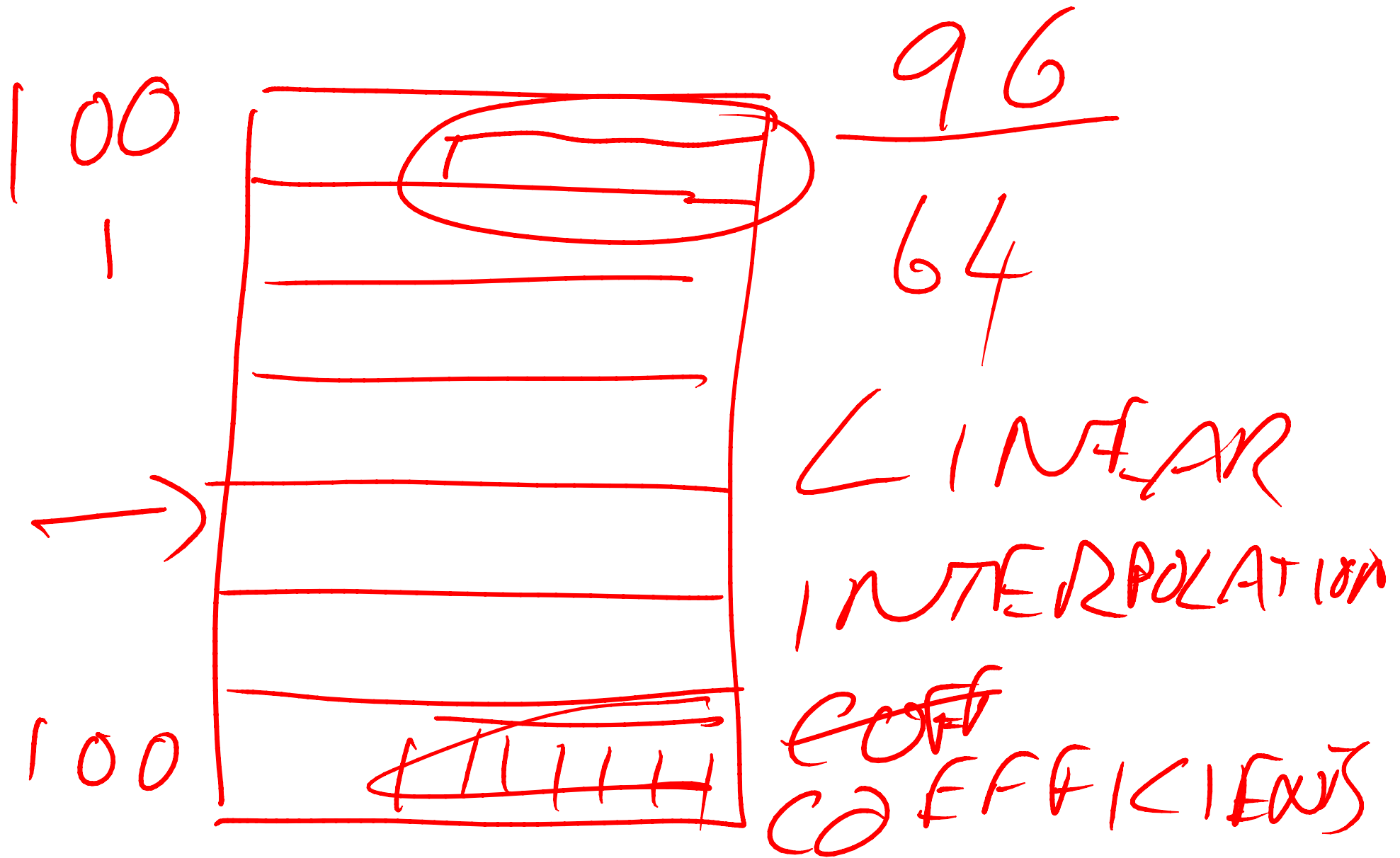
- Data/log file management
- Recovery model choice
- VLF fragmentation
- Logging and recovery
- Tempdb/temp tables
- Index strategy
- Fragmentation
- Index maintenance
- Query plans
- Plan caching
- DMVs
- Virtualization
- I/O subsystems
- Wait/latch/spinlock statistics
- Locking/blocking/deadlocks
- Statistics
- Database mirroring
- Database snapshots
- Partitioning
- Data compression
- Resource governor
- Extended events
- Performance tuning
- Instant initialization
- Backup compression
- Heaps vs. clustered indexes
- Data type choice
- Columnstore
- Scale-out
- Monitoring

1 TB

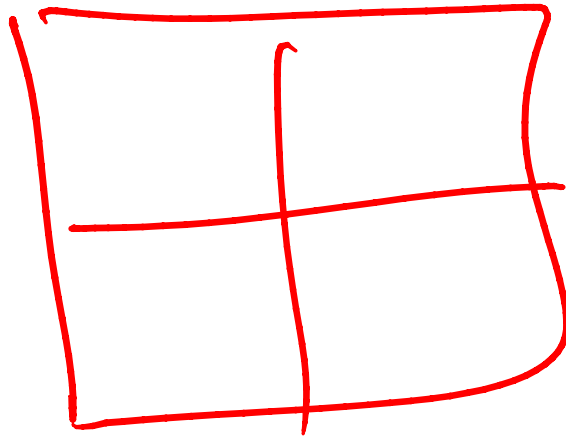
100 GB



Explaining merry-go-round scans. See [http://msdn.microsoft.com/en-us/library/ms191475\(v=sql.105\).aspx](http://msdn.microsoft.com/en-us/library/ms191475(v=sql.105).aspx)

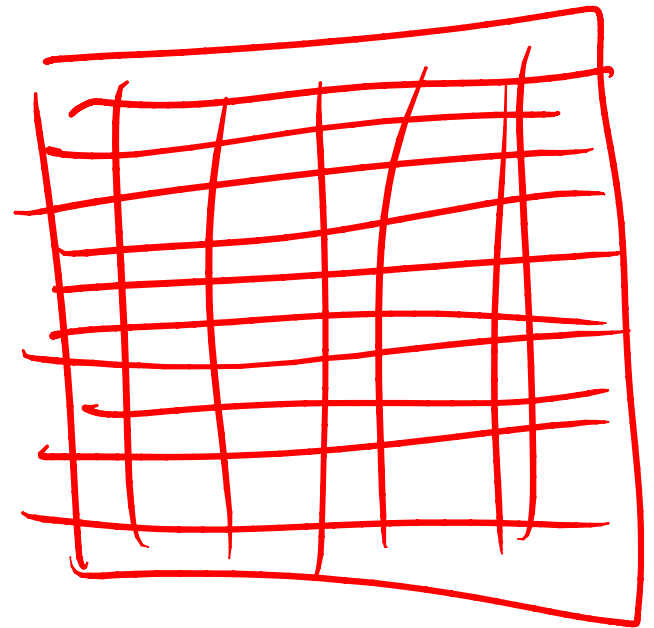


Explaining linear-interpolation for fixed size rows with monotonically increasing keys. A benchmark special put in for TPCH.



TF834

TRANSLATION LOOKASIDE BUFFER

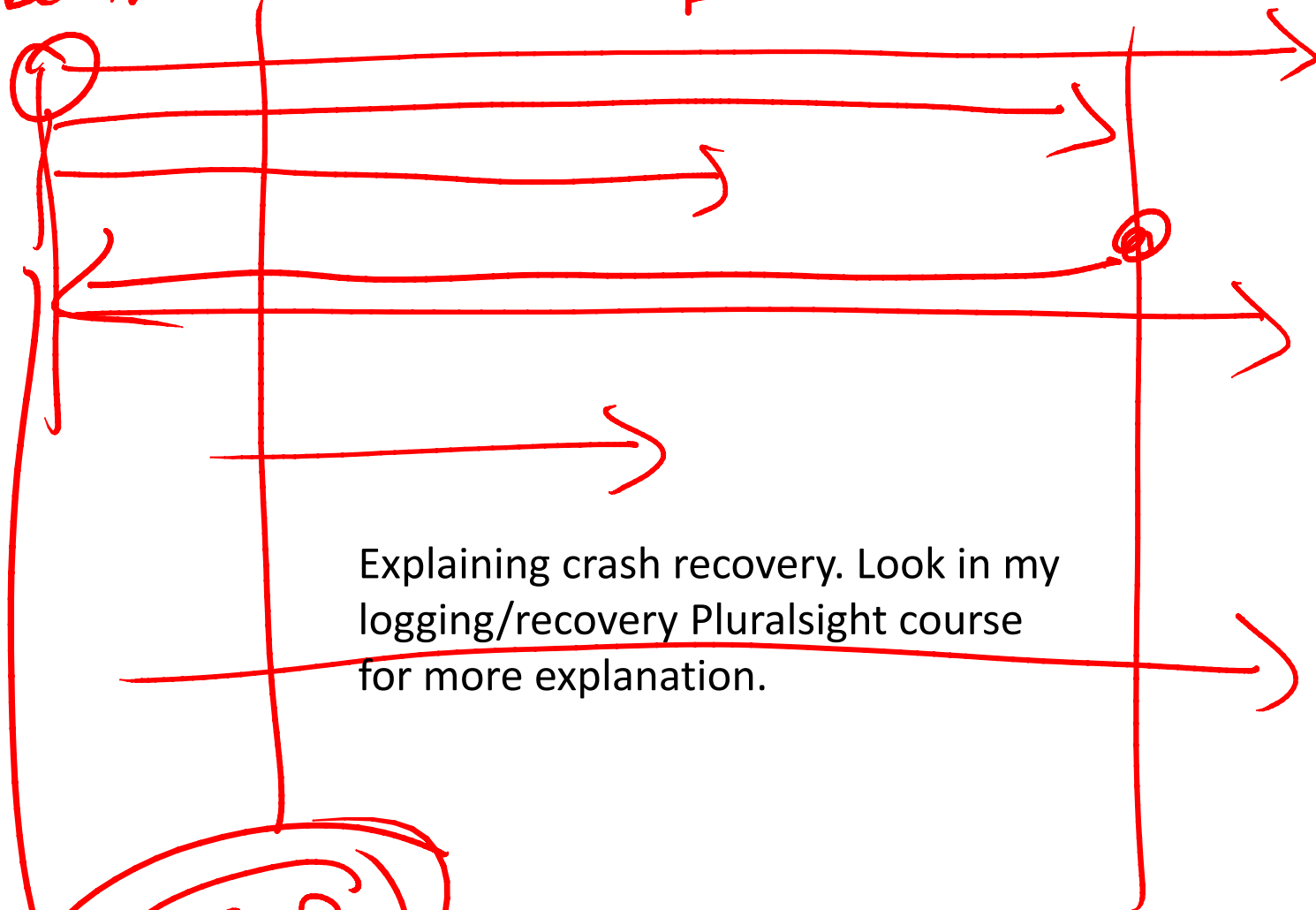


Explaining large pages using TF834 that reduces the requirements for memory translation lookaside buffers. Speeds up memory access for SQL Server.

BEGIN

END

3



Explaining crash recovery. Look in my
logging/recovery Pluralsight course
for more explanation.

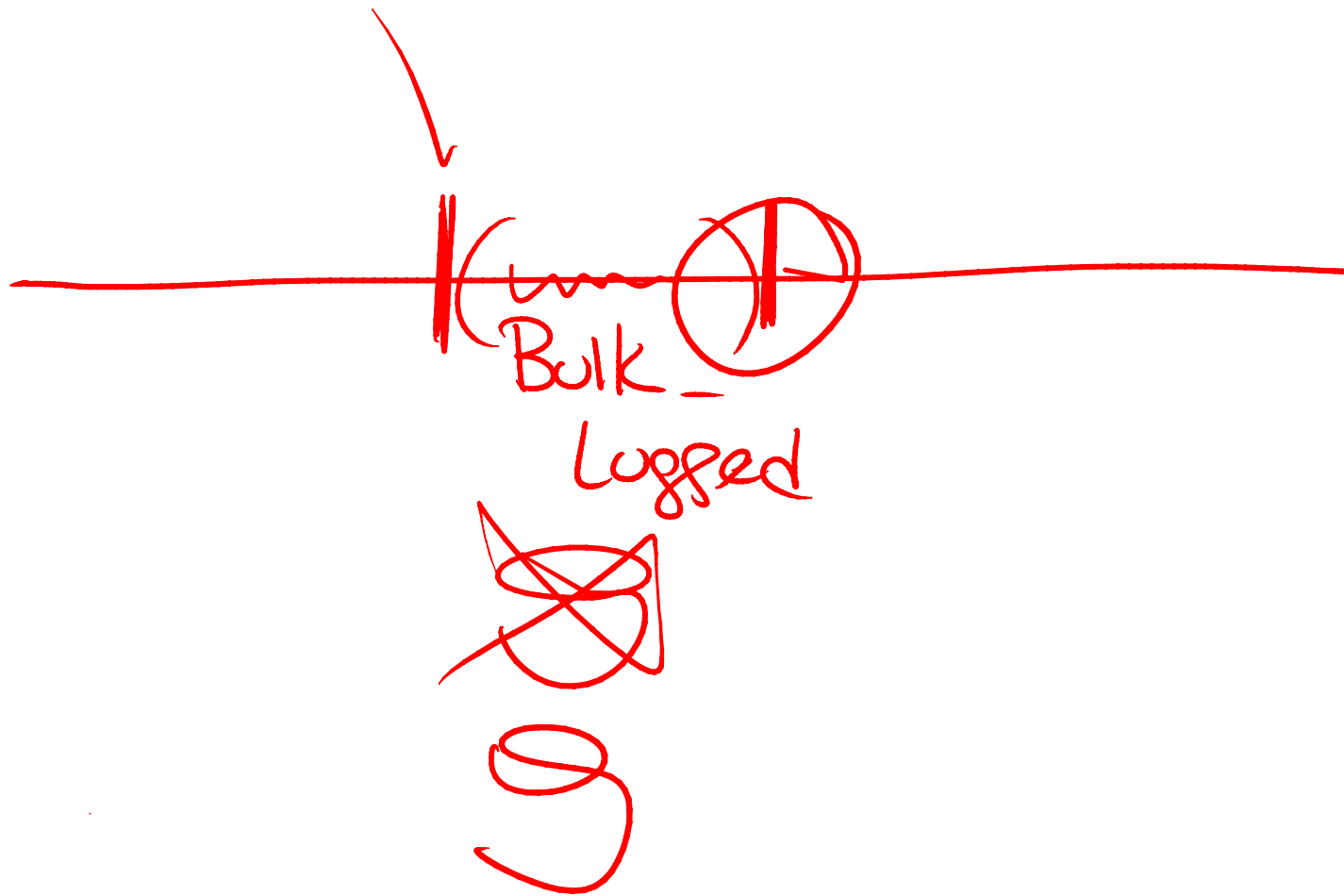
X

X

CP

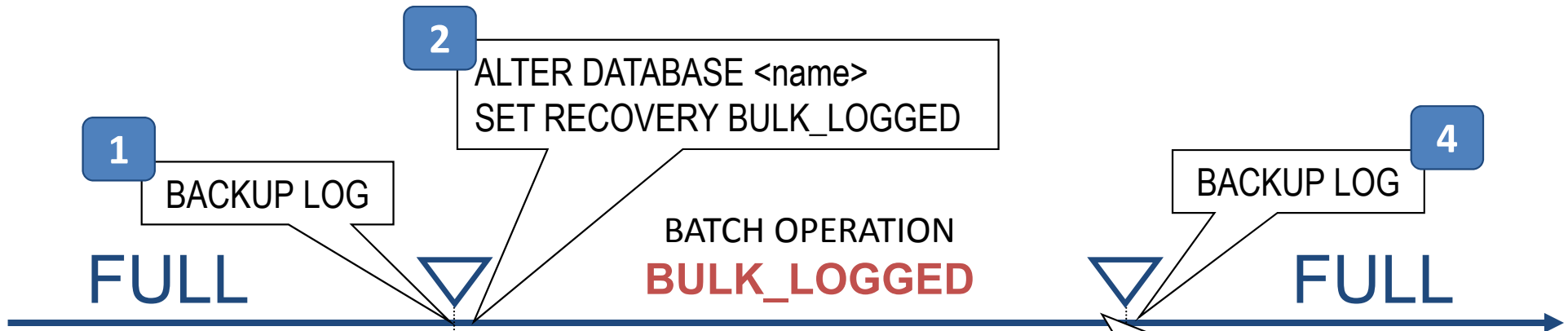
LSA

CRASH



This was my simple discussion on using the bulk_logged recovery model. Check out the next slide to see a bit more detailed content for this.

Switching Recovery Models



Advantages:

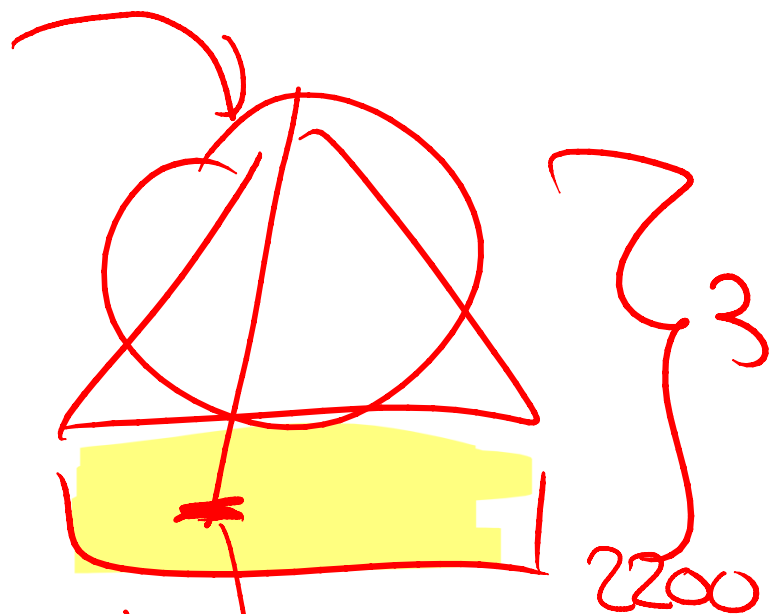
- Up-to-the-minute recovery
- Point in time recovery
- Access to "tail" of the log

Advantages:

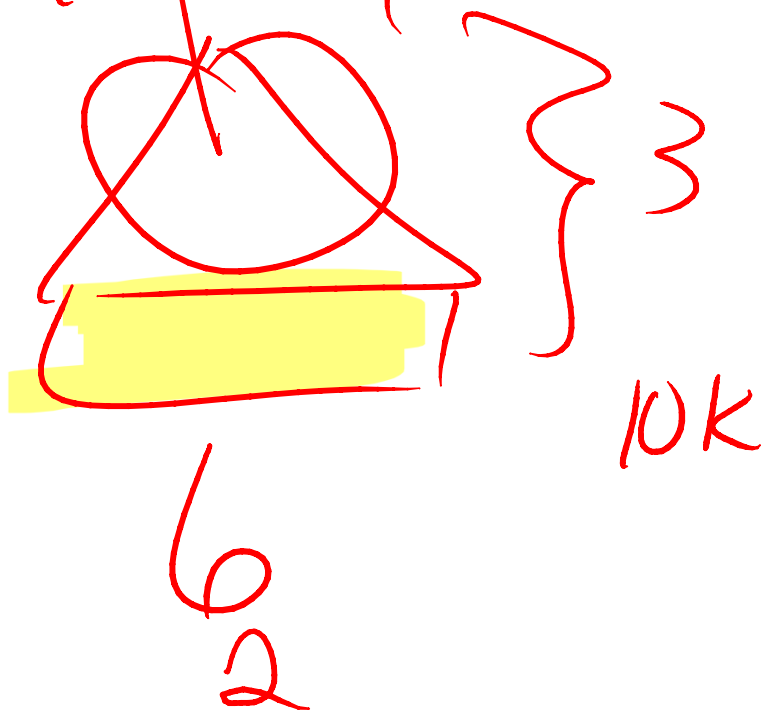
- Minimal logging
- Faster bulk operation

Disadvantages:

- No STOPAT
- Data loss possible if media failure...
- If database becomes suspect, cannot back up tail of log.



NCKey, CLKey
4 4



10K



2900

NCKey, RID
4 8

Fixed
Physical



Heap
4-5
2-3 phys.

Question/Discussion

This diagram was around the discussion of costing (of IOs) for lookups – between a heap and clustered table.

It **looks** like the CL index is worse:

Using a nonclustered to do a lookup (= 3 IOs), then – using the CL to lookup the data row = 3 more IOs for a total of 6 IOs

Whereas the heap seems to require fewer IOs. Using the nonclustered is about the same (=3 IOs), then – using the Heap to access the data row is 1 (possibly 2 if there's been record relocation) for a total of 4-5 IOs.

The long story short is that 4-5 IOs is less than 6 IOs. That *seems* better. Yes, the number is lower but the IOs are potentially more expensive. The yellow highlighting shows where the more expensive IOs are going to be performed (which is predominantly in the leaf structures).

As a result, a bookmark lookup from a NC to a clustered has 2 potentially physical IOs. The lookup from a NC to a heap has potentially 2-3 physical IOs.

While many lookups might be the same – there are still OTHER reasons for why heaps are not ideal. This is just yet-another-one. ☺

Clustering Key

DTA/missing

exp. CL key

Is it an issue that tools like the green hint and DTA explicitly add the clustering key columns in nonclustered indexes?

No, not an issue. SQL Server will not add a CL key column more than once in an index.

CL c6, c5, c2

UNC Key (c4, c5)

Where do the clustering key columns end up in a nonclustered index?

In a **non-unique** nonclustered index the clustering key column(s) must go all the way up the tree.

nonunique

B-tree c4, c5

leaf

c4, c5, c6, c2

c4, c5, c6, c2

c4, c5, c6, c2

Where do the clustering key columns end up in a nonclustered index?

In a **unique** nonclustered index the clustering key column(s) are only in the leaf level.

Clustering Key Columns WHERE?

Where do they go within nonclustered indexes?

- What if:

```
CREATE UNIQUE CLUSTERED INDEX IXCL  
ON tname (c6, c8, c2)
```

```
CREATE NONCLUSTERED INDEX IXNC1  
ON tname (c5, c2, c4)
```

- Leaf level: c5, c2, c4, c6, c8
- KEY/btree: c5, c2, c4, c6, c8

```
CREATE UNIQUE NONCLUSTERED INDEX IXNC1  
ON tname (c5, c2, c4)
```

- Leaf level: c5, c2, c4, c6, c8
- KEY/btree: c5, c2, c4

- Key points:

- Clustering key columns are added only ONCE to your nonclustered indexes
- Where they are added (leaf only or all the way up the tree) is based on whether or not the nonclustered is nonunique. When nonunique, the CL key goes up the tree.

Resources

- [Understanding the value of the Enterprise Edition, one feature at a time](#)
 - Blog series: Nacho Alonso Portillo (believe anything he says!)
- **Understanding Logging and Recovery in SQL Server**
 - Paul Randal, TechNet article
- **See the links in the zip for these resources**

Questions?

**Don't forget to enter your evaluation
of this session using EventBoard!**

Thank you!

