Ask Anything Index

(3:30 PM - 4:45 PM)

SQLintersection Speaker Panel

Kendra Little, Kendra@BrentOzar.com

Joe Sack, Joe@SQLskills.com

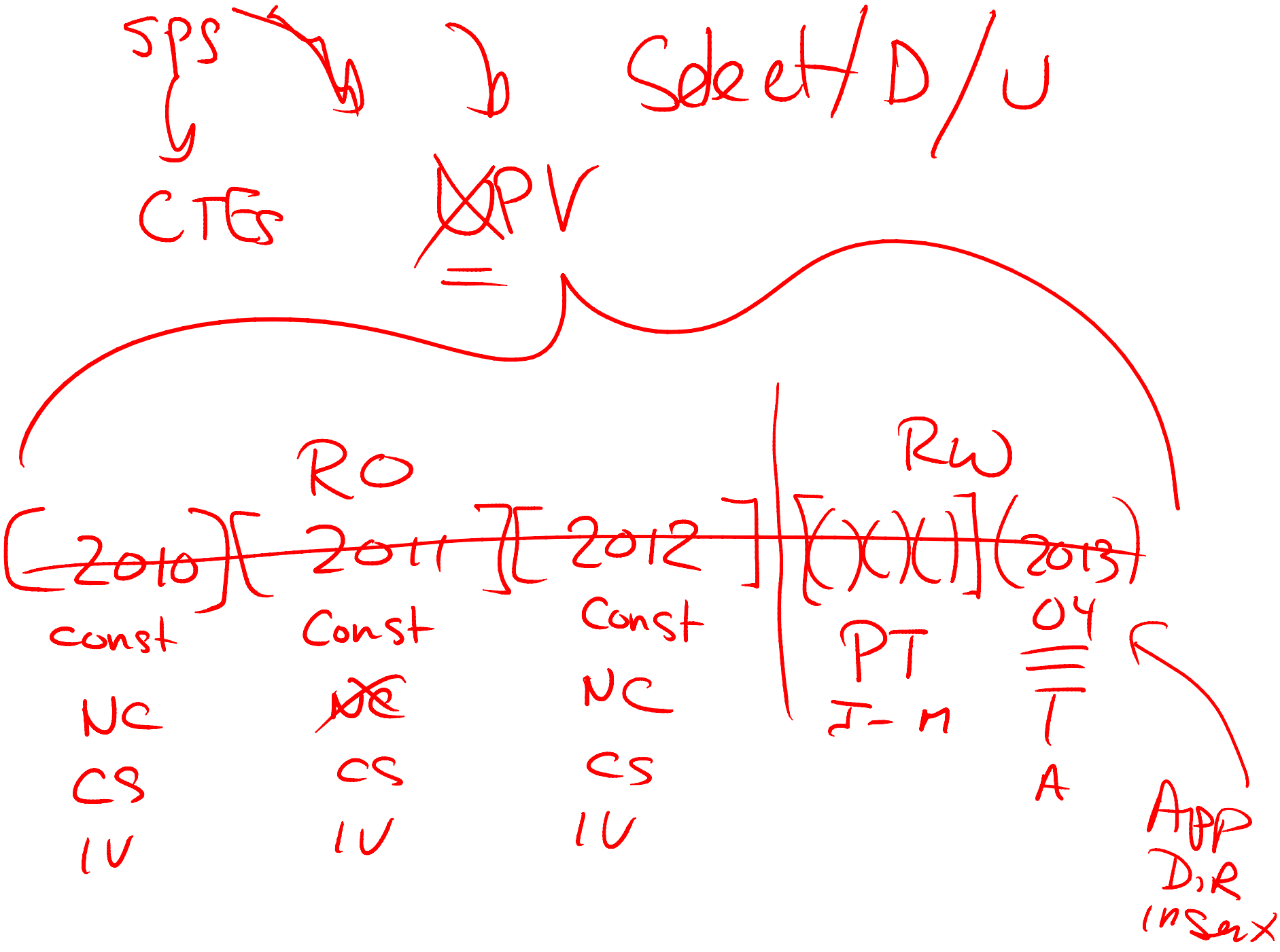
Kimberly L. Tripp, Kimberly@SQLskills.com

Potential Topics

- **Clustered Index**
- **Clustering Key Choice**
- **Nonclustered Indexes**
- **Nonclustered Index Key Order**
- **Using INCLUDE**
- **Filtered Indexes**
- **Columnstore Indexes**
- **Partitioning and Indexes**
  - Aligned Indexes
  - Un-aligned Indexes
- **Index DMVs**
- **DML and indexes?**
- **Table design and Columnstore**
- **Partitioned Views**
  - Union ALL issues with Columnstore
- **Index Maintenance**
  - FILLFACTOR
  - PADINDEX
- **Duplicate Indexes**
- **Redundant/Similar Indexes**
- **Removing Indexes**
  - What could go wrong?

FYI

- The slides in this deck are from the whiteboard drawings that were made during the session.
- Don’t forget to check out the links at the end!


SPS
CTEs
PV
Select/D/U
RO
RW
2010
2011
2012
Const
Const
Const
NC
CS
IV
NC
CS
IV
PT
I-M
04
T
A
App
D,R
Index

This discussion was the combination of partitioned views and partitioned tables and the benefits that combining these technologies brings.

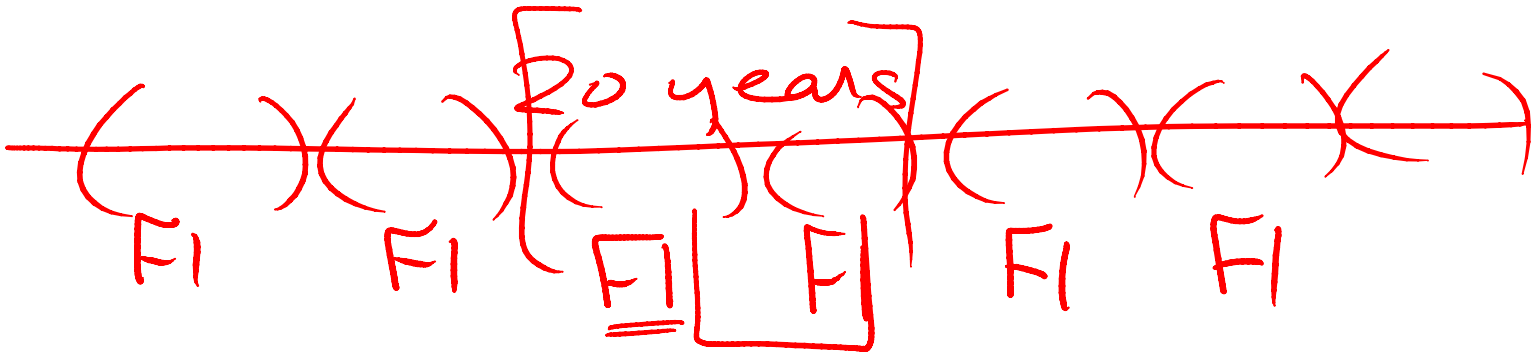
Partitioned views

- Any edition
- Lots of tables to administer
  - Must create/drop indexes on all base tables
    - Can have different indexes
    - Harder for the optimizer to optimize with so many indexes
  - Must verify business logic so that there are no gaps or overlapping values
  - Each table has [potentially] better statistics as the tables are smaller
- Can rebuild any of the tables ONLINE (if using EE)
- Can support multiple constraints on one or more columns

Partitioned tables

- Enterprise Edition only
- Only 1 table to administer
  - Only 1 table to create/drop indexes
    - All partitions have same indexes (which is easier for the optimizer to optimize)
    - Can create different indexes with filtered indexes
  - No possibility of errors (or gaps or overlapping values)
  - Table-level statistics can be less accurate for very large tables when there's a lot of skew to the data
- Partition-level rebuilds are offline but can rebuild the ENTIRE table online (not desirable)
- Can only support partitioning over a single column

F1 only NC indexes



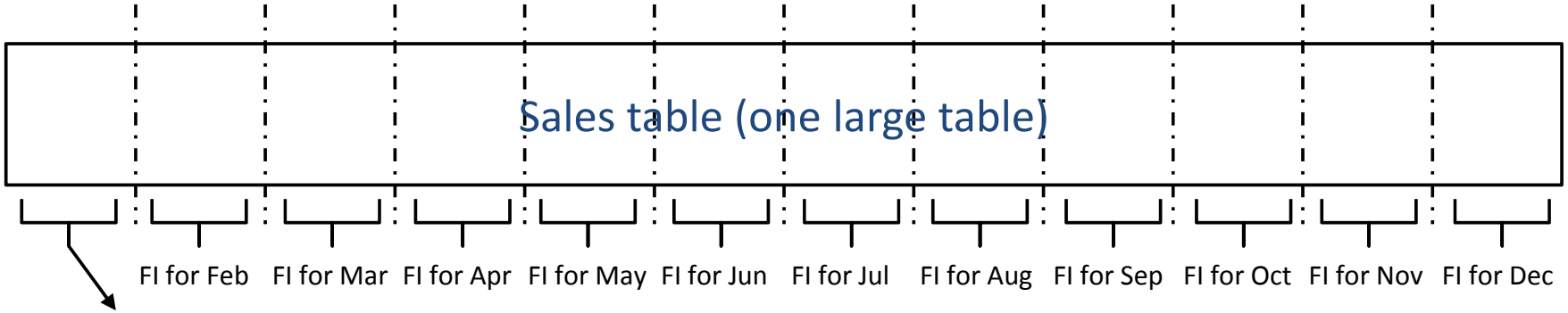
Internal 20111 2

subsumption problems

Session Settings (CR, S, I/D/U)

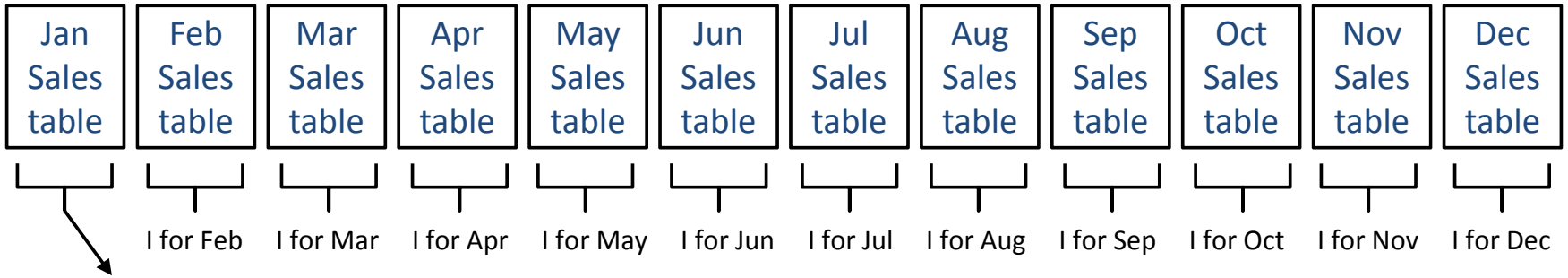
Stored procs need OPTION(recompile)

DBs that have FORCED PARAM



Filtered Index for January

```
CREATE INDEX JanuaryItems
ON Sales (col1, col2)
WHERE SalesDate >= '20120101'
        AND SalesDate < '20120201'
```



Non-filtered index for January

```
CREATE INDEX JanuaryItems
ON Sales (col1, col2)
```

NOTE: Each table has a constraint

```
ALTER TABLE JanSalesTable
ADD CONSTRAINT JanuaryItems
CHECK (SalesDate >= '20120101'
        AND SalesDate < '20120201')
```

Session Settings

- See BOL topic: *Set Options that Affect Results*
- Session settings control behavior – and the result of some computations
- Data in these persisted structures must be consistent
- Session settings that must be on:
  - ☐ ANSI_NULLS
  - ☐ ANSI_WARNING
  - ☐ QUOTED_IDENTIFIER
  - ☐ CONCAT_NULL_YIELDS_NULL
  - ☐ ANSI_PADDING
  - ☐ ARITHABORT
- Session setting that must be off:
  - ☐ NUMERIC_ROUNDABORT

Msg 1934, Level 16, State 1, Line 1

CREATE INDEX failed because the following SET options have incorrect settings: 'QUOTED_IDENTIFIER'. Verify that SET options are correct for use with indexed views and/or indexes on computed columns and/or filtered indexes and/or query notifications and/or XML data type methods and/or spatial index operations.

Client Consistency

- **Consistency with table(s), view and the clustered index (on the view) creation**

  *All tables on which the index/view is based, the view itself and the index, must be **created** with the correct session settings set or the index cannot be created*

- **Consistency with base table access**

  *All INSERT, UPDATE and DELETE statements must be **executed** with correct session settings or the insert, update or delete will fail*

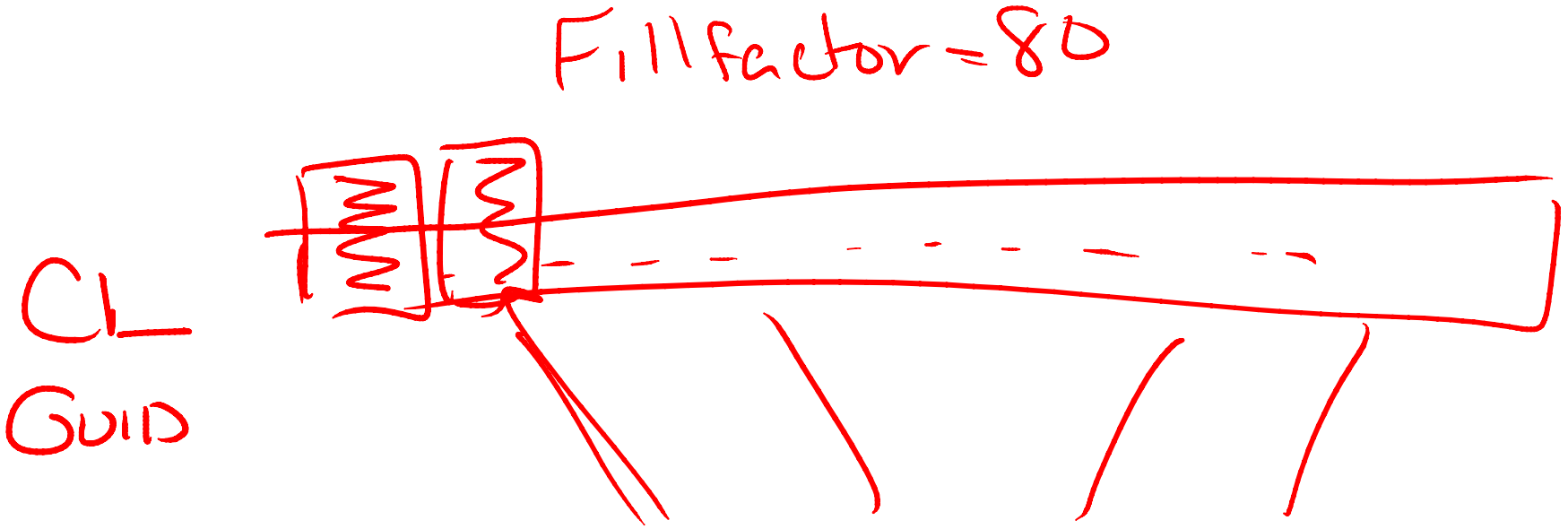
- **Consistency with view access**

  *All queries that **SELECT** against views with indexes must access them with the correct session settings set otherwise the view's data will need to be recalculated, rejoined or recomputed*

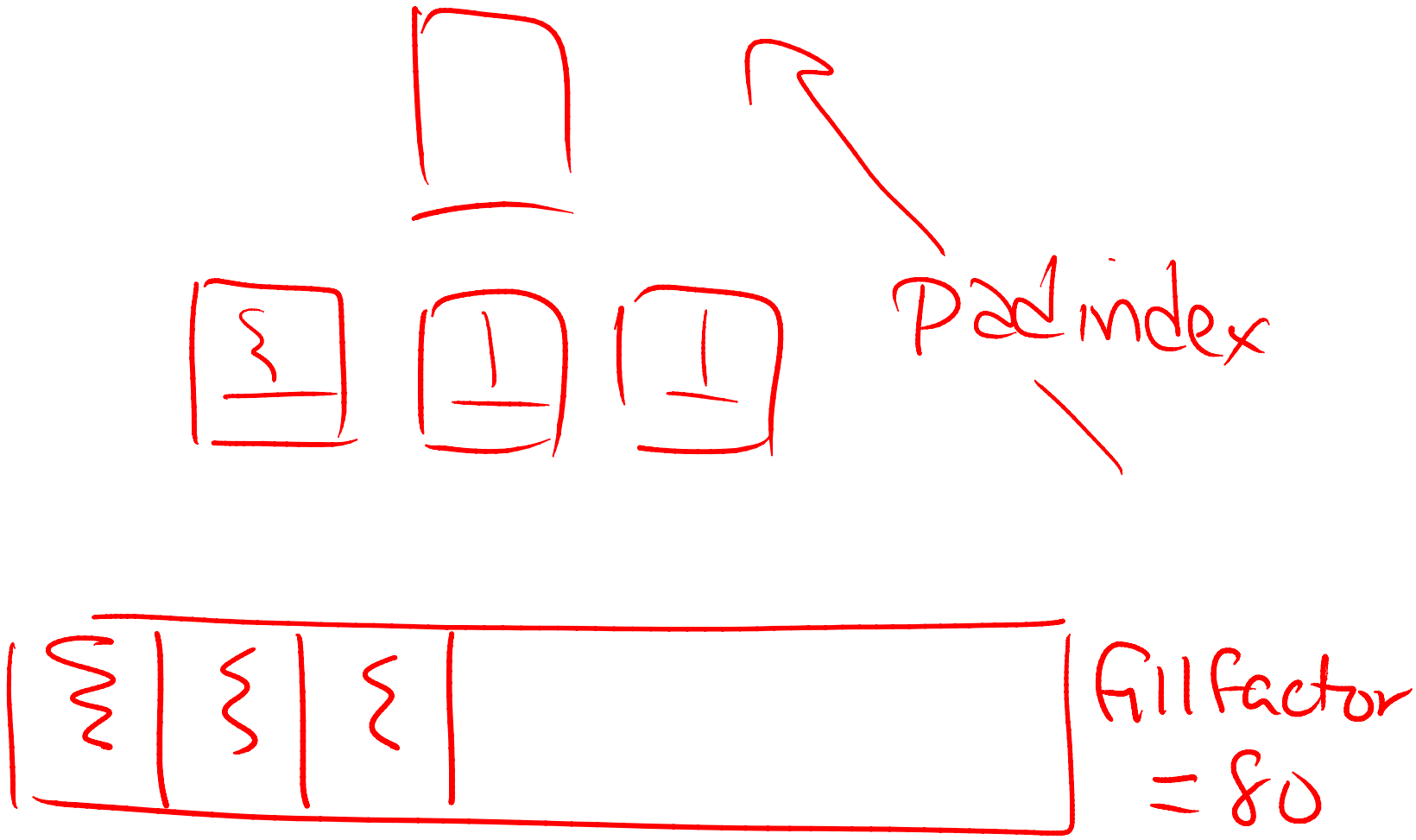
Fillfactor

only applies at CR\Rebuild\Reorg

defines fullness of leaf level

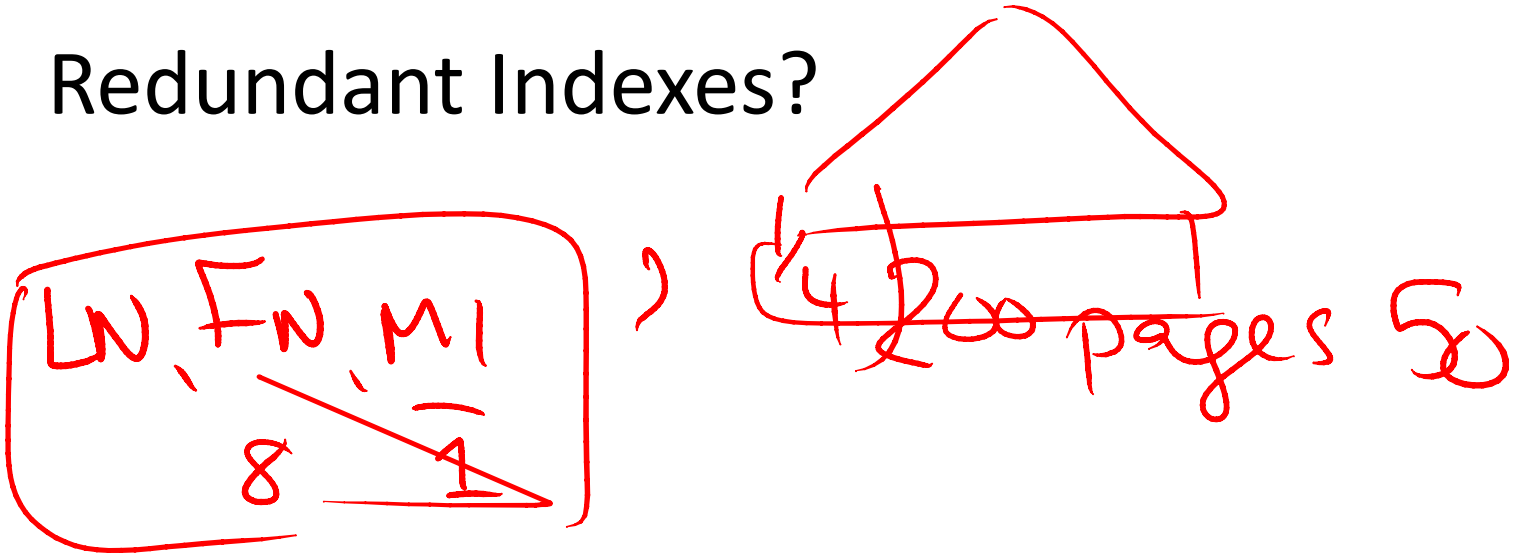


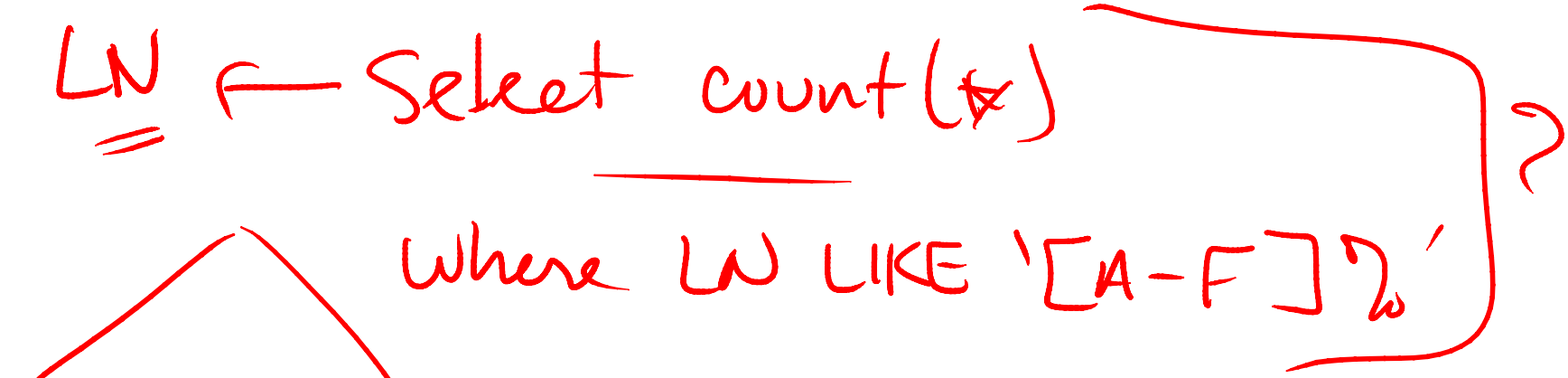
Fillfactor affects the fullness of the leaf level of the index to which you've applied it.



PADINDEX does not allow a specific setting and can only be used IF you've set FILLFACTOR. The fillfactor value is pushed up the tree. If you've set a good fillfactor for the leaf level then it's unlikely that you'd need to use PADINDEX.

Redundant Indexes?





The question was Could you drop an index if it's a left-based subset of anther…
Absolutely – you *could*. But, do you want to?
You really need to know the workload. If the smaller index isn't used often or the queries that use it aren't critical then you could drop it. But you definitely need to know what's using it FIRST.

Left-based subsets and ORDER

Can the index on FN, LN be dropped and the index on LN, FN, MI be used instead?
Maybe?
If LN & FN are *always* supplied and they're always equality-based then YES
But, if you do lookups where you sometimes specify only LN then the index on LN, FN, MI would need to be used. If you do lookups where you sometimes specify only FN then you need that second index.

Index Analysis

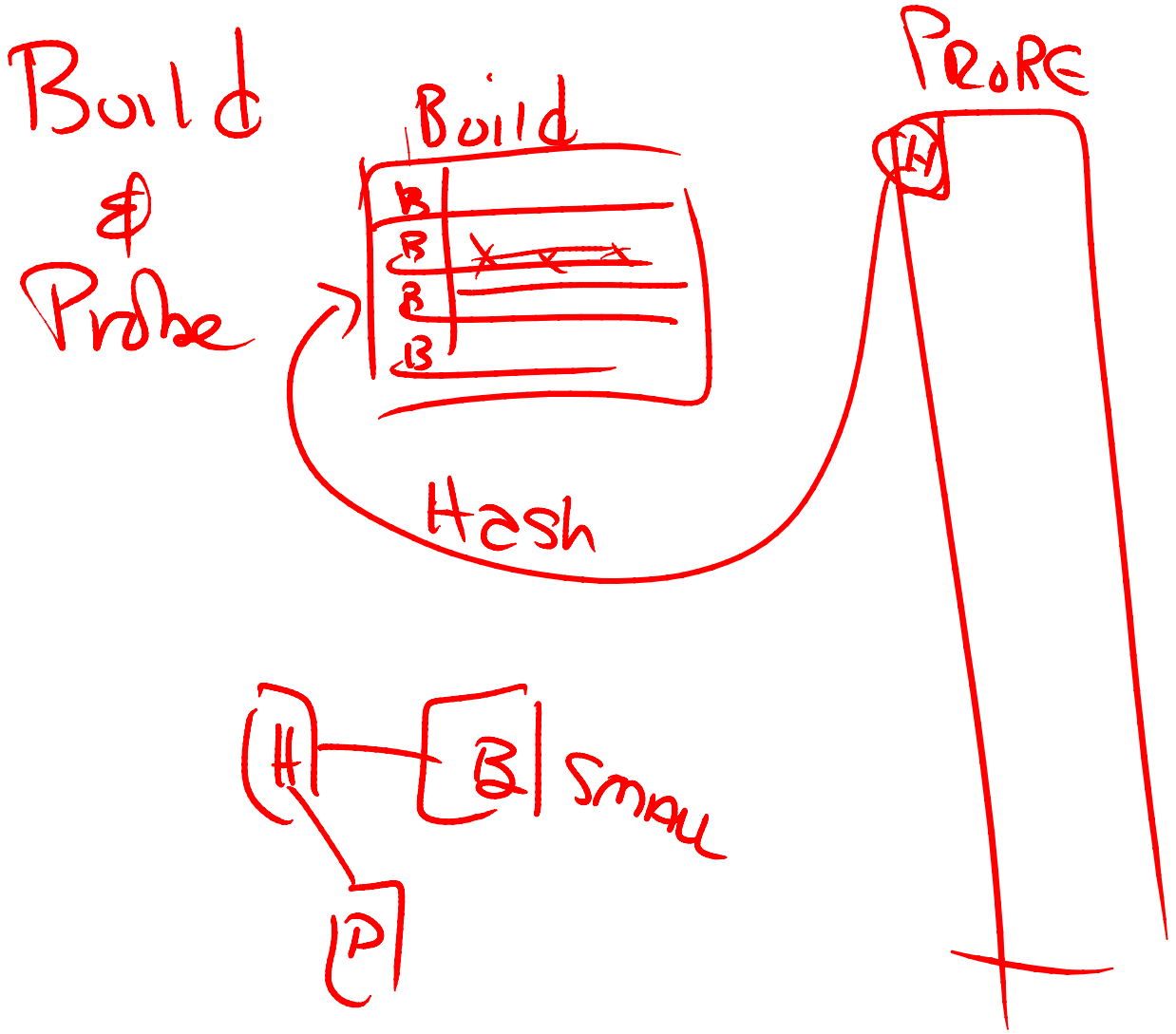
0) weird patterns (Heaps) Look

1) Unused Dupes \ Consol. \ Left-based subsets

2) Health FF

↳ state?

3) Missing Indexes



Hash Joins

Hash are a little more complicated. There are multiple hash types available in SQL Server and each provide different benefits. The general purpose of a hash join is to significantly reduce the number of rows that have to be processed.

More specifically, there are two phases:

BUILD phase
PROBE phase

The build phase is used to create a small structure into which the larger set can probe to determine if there's the possibility of a matching row.

UPDATE STATS (FAST) Sampling
Fullscan (parallel)

[Rebuild
Fullscan

Analyze other stats

Stats before Index Builds?

For a large table whose index rebuild could be parallelized – updating the stats first might help the optimizer better understand the data so that it might do a better job at the rebuild. However, updating stats is also improved by having rebuilt the index.

So, the end result – update stats sparingly (and just use sampling) to get the data closer to current, quickly.

Then, update statistics for everything else that remains – after rebuilding indexes – so that the process of updating statistics is faster too.

Resources Discussed

- **Data Loading Performance Guide**
  - by Thomas Kejser, Peter Carlin and Stuart Ozer
- **Hash joins and hash teams in Microsoft SQL Server**
  - by Goetz Graefe, Ross Bunker, Shaun Shaun Cooper
- **Query Evaluation Techniques for Large Databases**
  - by Goetz Graefe

- **See the links in the zip for these whitepapers**

Questions?

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

Thank you!

SQL intersection