

Whiteboard
annotations from
Sunday, April 7

SQLintersection

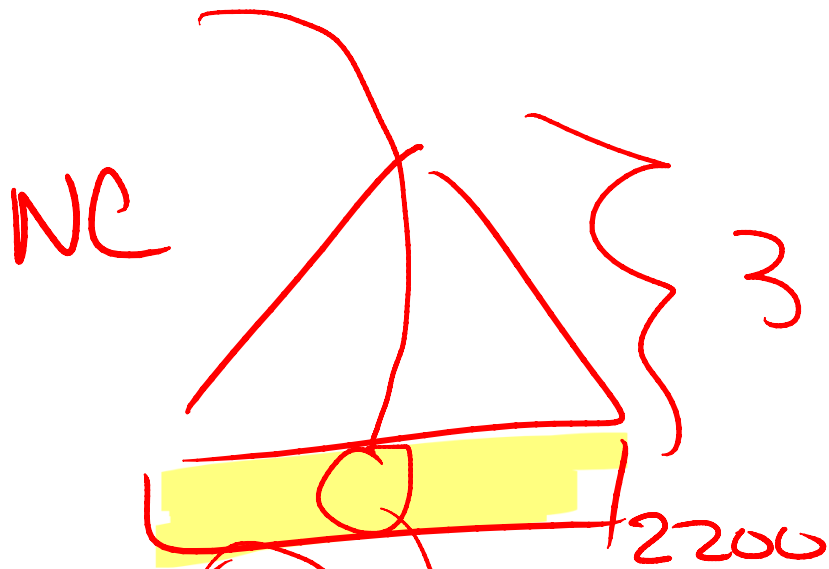
PRECON1

Indexing Strategies that Work:
Covering – Concepts, Concerns, Costs
(8:30 AM - 4:30 PM)

Kimberly L. Tripp

Kimberly@SQLskills.com



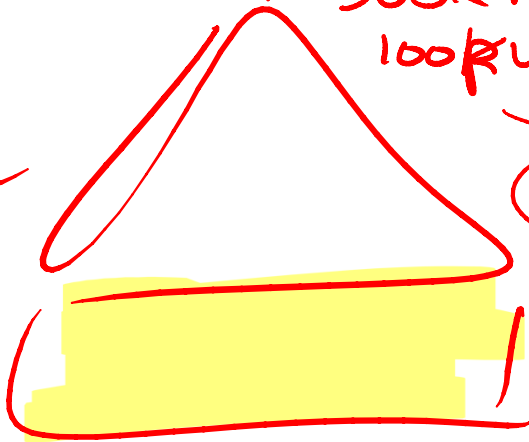


NC
int

CL
int

bookmark
lookup

CL



CL (int)

6

1.6
mill
~10K



NC
int

RID
8 bytes

2:4:2
f p slot



forwarding
ptrs

4-5

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. ☺

Time

Parse

Optimization Sample Exec

TSQL

↳ Normalized
Query
Tree

Missing Stats Event

If auto cr stats is on
WAIT Stats get cr.

Invalid Stats event

x if auto up a sync on
use invalid stat

Kick off up a sync
* if auto up is on

then wait

Statement Execution: Statistics Events

Optimization = Compilation

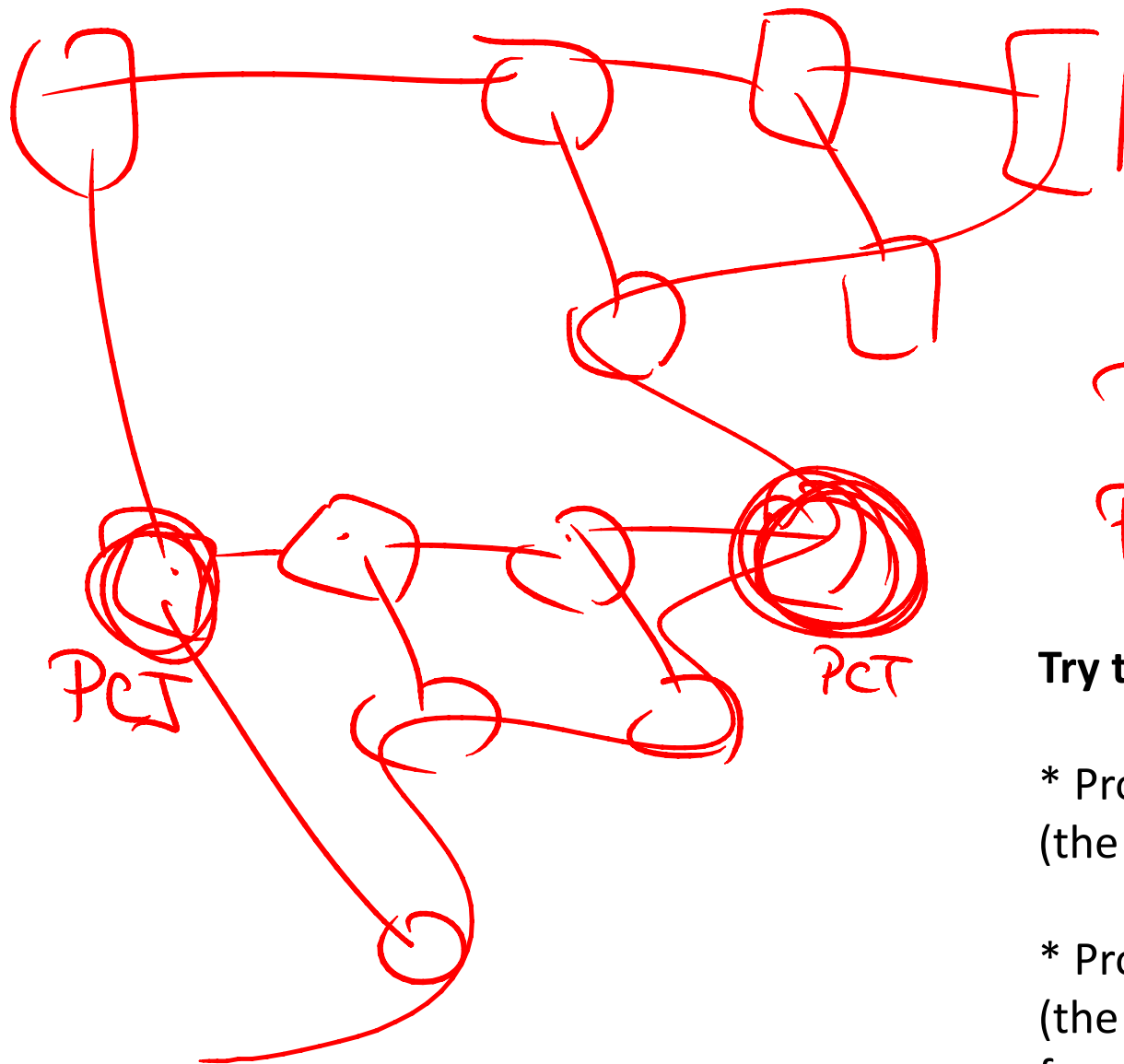


Missing Statistics Event

If **auto_create_stats** is enabled then SQL Server (the QO) will WAIT while statistics are created

Invalidated Statistics Event

- If **auto_update_stats_async** is disabled AND **auto_update_statistics** is enabled then SQL Server will WAIT while statistics are created
- If **auto_update_stats_async** is enabled then SQL Server will optimize based on the invalidated statistics AND kick off an update (which will be used by subsequent users)
- If both methods for updating statistics are disabled then the query will optimize using the invalidated statistic and a warning will be generated (visible in [xml] showplan)



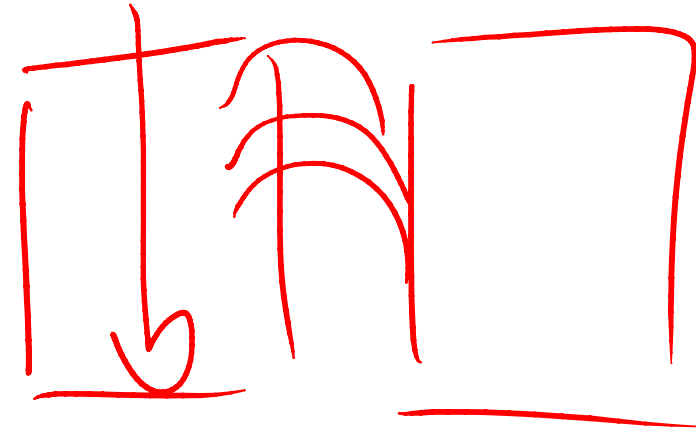
PCT
PCT
=

Try to find your:

- * Problem Child Table
(the table with the highest cost)
- * Problem Child Join
(the join – typically downstream from the table – with the highest cost)

Tuning a large complex join takes breaking it down into smaller chunks. The things that you consider are the costs of the tables (the outer most events) and the costs of the joins. And, typically, the most expensive join is downstream from the most expensive table.

Loop Joins



filter
3
indexed
100 rows
10 pages

match
index
j.c
filter
100 rows
10 pages

Here we started to talk about how LOOP joins are an iterative process. The driver (the outer/first table) is typically chosen because it has the most selective set. Any of the tables that have a highly selective search argument are more likely to be chosen as the driver. An index that aids in efficiently finding those rows is REALLY helpful!

Cost can be calculated as:

Number of IOs required for first table +
Number of resulting rows in first table * the cost for each lookup (ideally with an index on the join condition)

- IF -

no good indexes
SQL HAD to
do loop

SCAN

$$10 + (100 \times 10)$$

In the worst case scenario the costing of this iterative process can be very high/expensive.

Instead of doing this, SQL Server is much more likely to do a loop join.

Merge Joins

Merge Joins

Merge joins leverage “suitably sorted sets.”

More specifically, merge leverages indexes whose leading keys are on the same column.

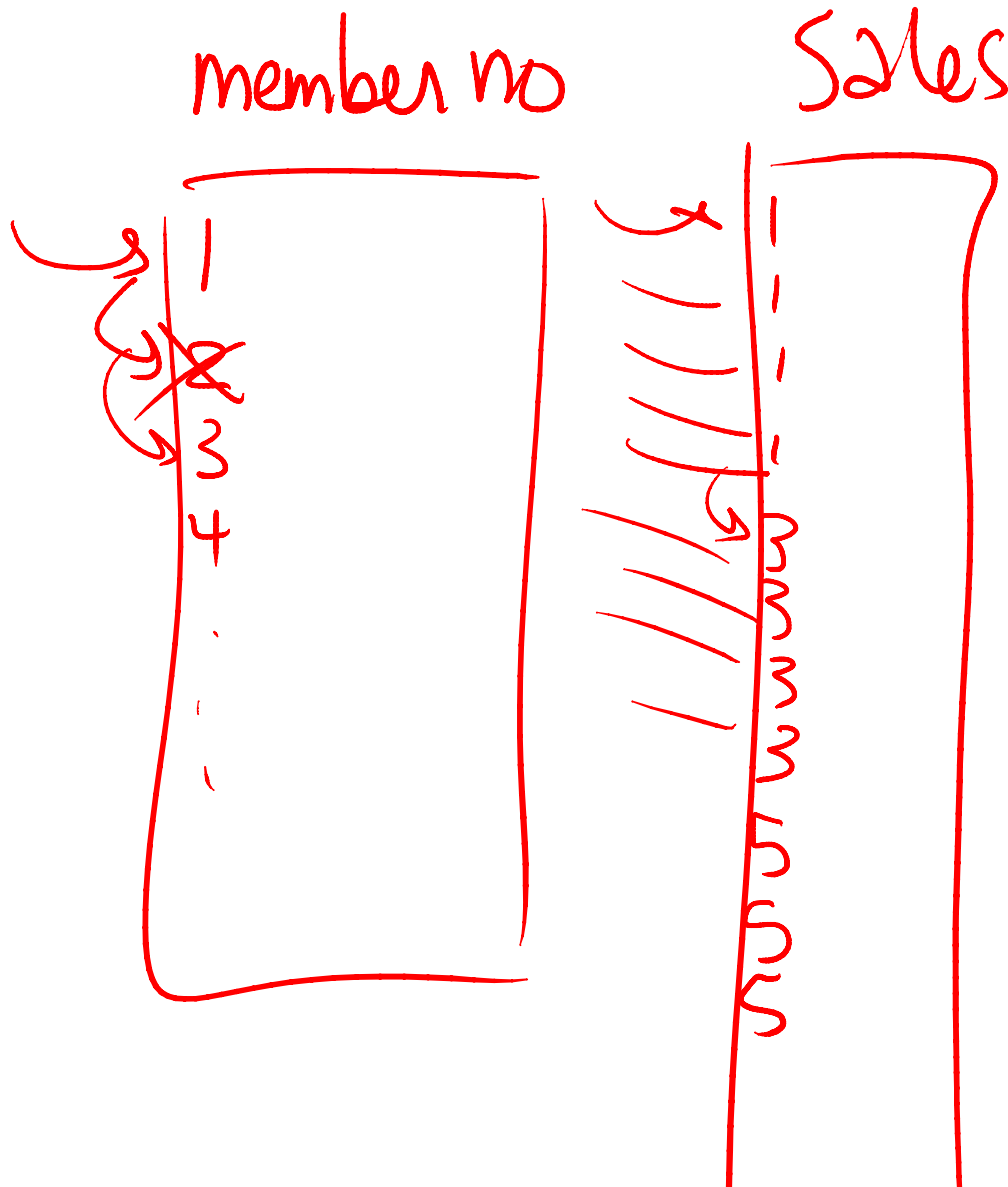
Key question:

What columns do these two tables have in common?

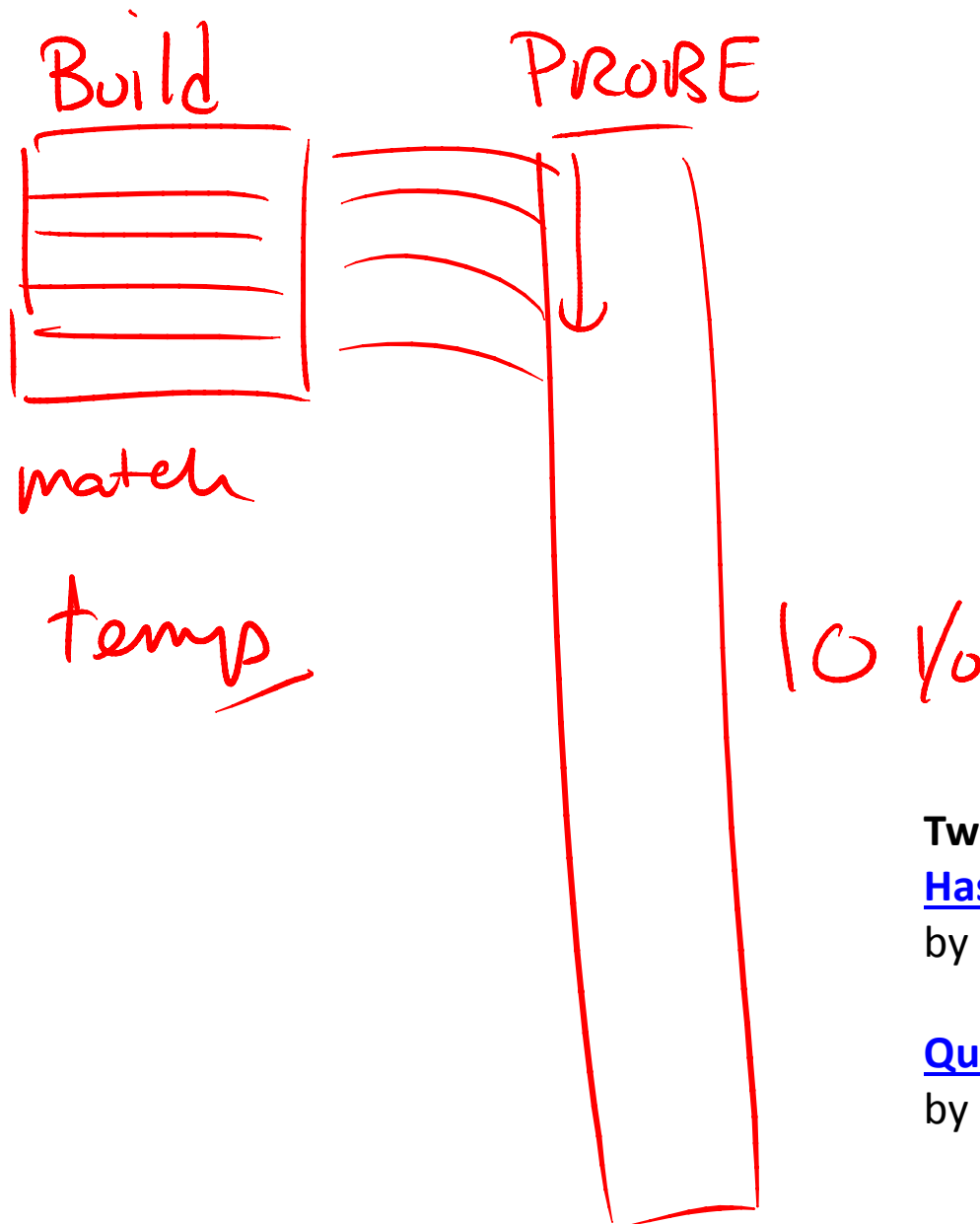
The join column...

Corollary question:

Do you have indexes on EACH of the columns (in each table)? The one that’s often missing: the join column that’s the foreign key.



Hash Joins



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.

Two very good resources:

[Hash joins and hash teams in Microsoft SQL Server](#)

by Goetz Graefe, Ross Bunker, Shaun Shaun Cooper

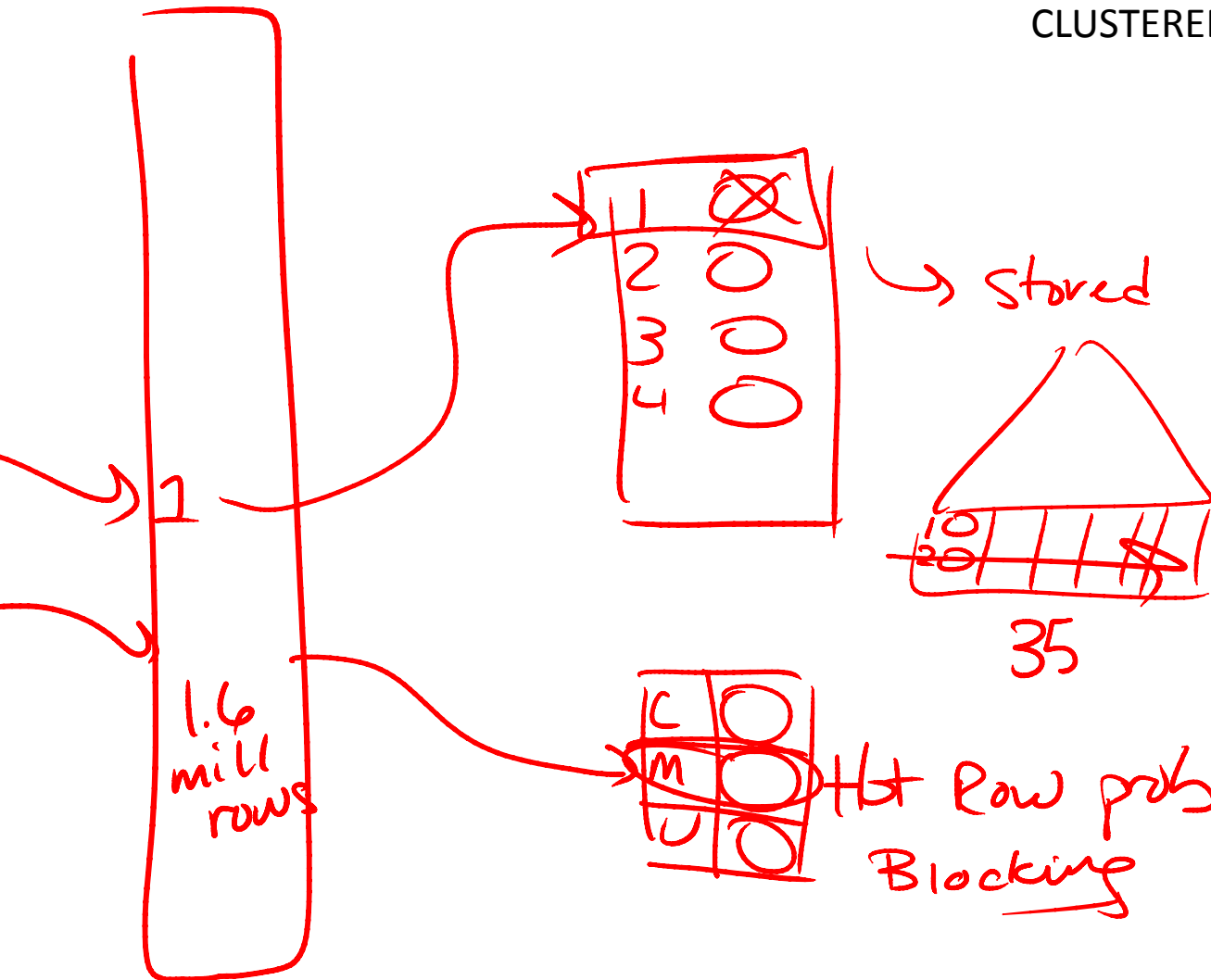
[Query Evaluation Techniques for Large Databases](#)

by Goetz Graefe

Indexed Views

Indexed Views

Are results sets defined by a view and materialized into the leaf level of the UNIQUE CLUSTERED INDEX that's defined on the view.



Hot Row

If your aggregate is too small then you can have a HOT ROW problem where all modifications are blocked trying to write to the aggregate. You'll serialize your inserts by country here... CAN, MEX, USA are the only countries with whom you do business – all US rows will have to wait as each updates the sum. This will become a terrible bottleneck.

Additional References

- **Poor Performance of Inserts on a Heap**

- <http://support.microsoft.com/kb/297861>
- Note: The article says it only applies to 2000 but really it's 2000 and higher. And, there's still a lot more "it depends" to it but, it's still a reference that I mentioned.