



**BRENT OZAR**  
UNLIMITED®

# CTEs, Temp Tables, and APPLY

What's the right tool for the job?

1.3 p1

You take pictures.





**Not every picture needs the same tool.**



Sometimes you just need good enough, quickly.



## Use cases we'll explore

1. “I have a few different queries that I want to merge into one.”
2. “I have a big query where the estimates go wrong halfway in, so I need to break it up into stages.”
3. “I want to run a subquery for every row, getting the top N rows, but I want it to go fast.”



1.3 p5

**“I have a big query with a lot of joins, and the estimates go wrong.”**

# Table variables and temp tables



# Users want us to tune this...

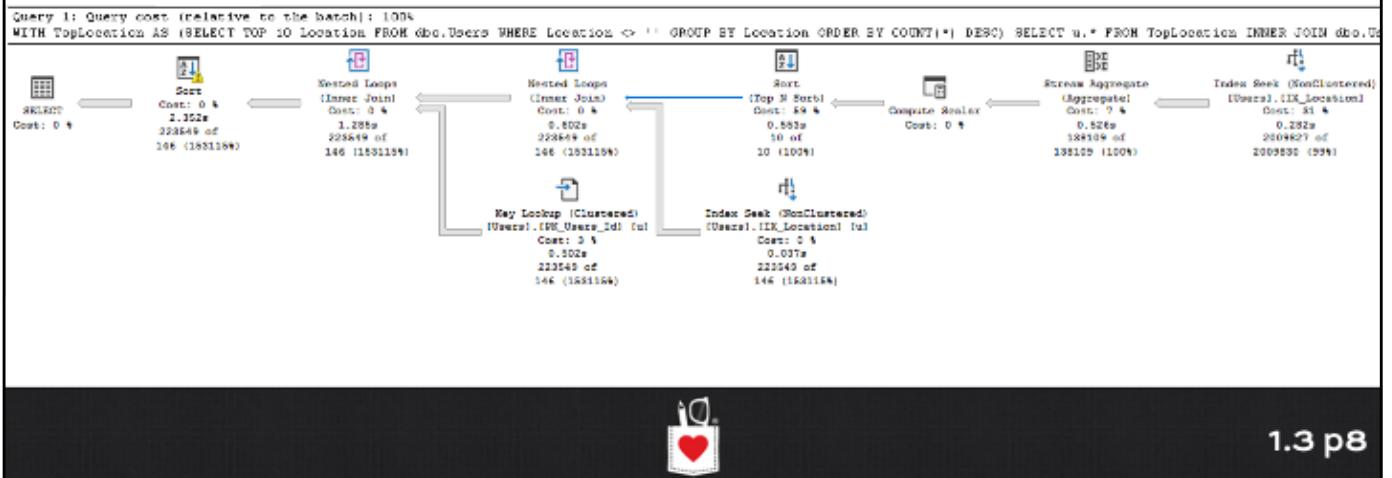
And we already added an index...

```
CREATE INDEX IX_Location ON dbo.Users (Location);
GO
CREATE OR ALTER PROC dbo.usp_UsersInTop10Locations AS
BEGIN
    WITH TopLocation AS (SELECT TOP 10 Location
        FROM dbo.Users
        WHERE Location <> ''
        GROUP BY Location
        ORDER BY COUNT(*) DESC)
    SELECT u.*
        FROM TopLocation
        INNER JOIN dbo.Users u ON TopLocation.Location = u.Location
        ORDER BY DisplayName;
END
```

Note: TEN locations, not one

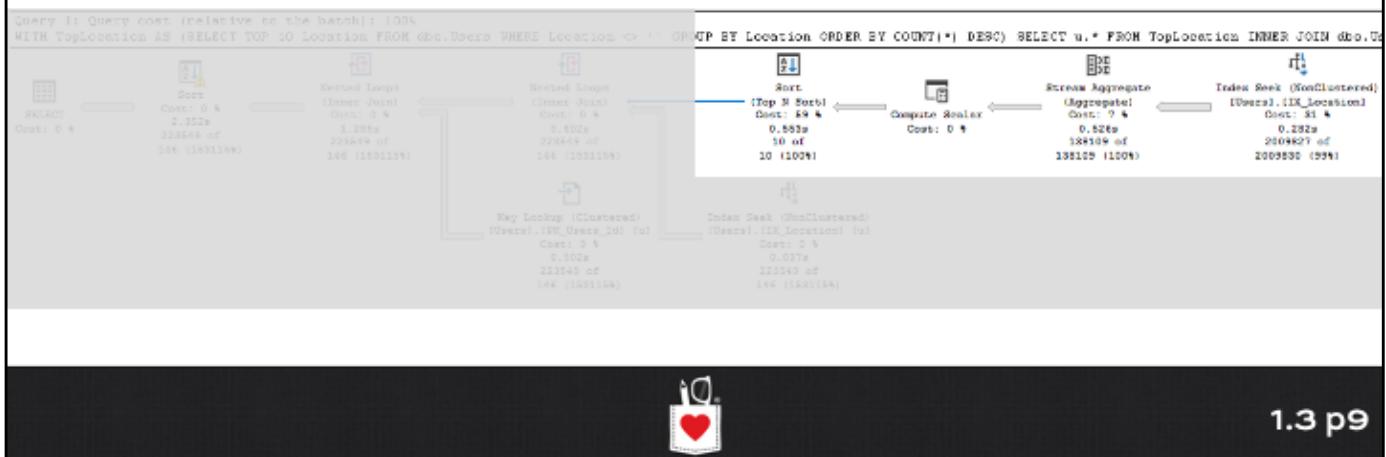
# The query plan: look familiar?

It's like our recent query where we got the top 1 location, but it's 10.



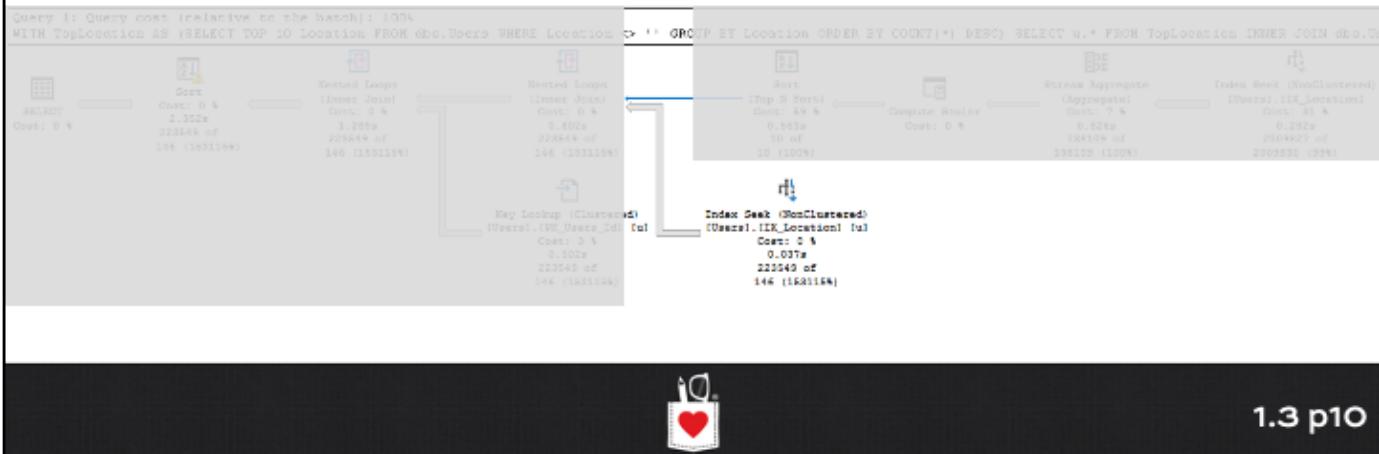
# In here, the estimates are right

SQL Server groups the locations together to find the top 10, and it knows that there will be 10 values...just not which locations they'll be.



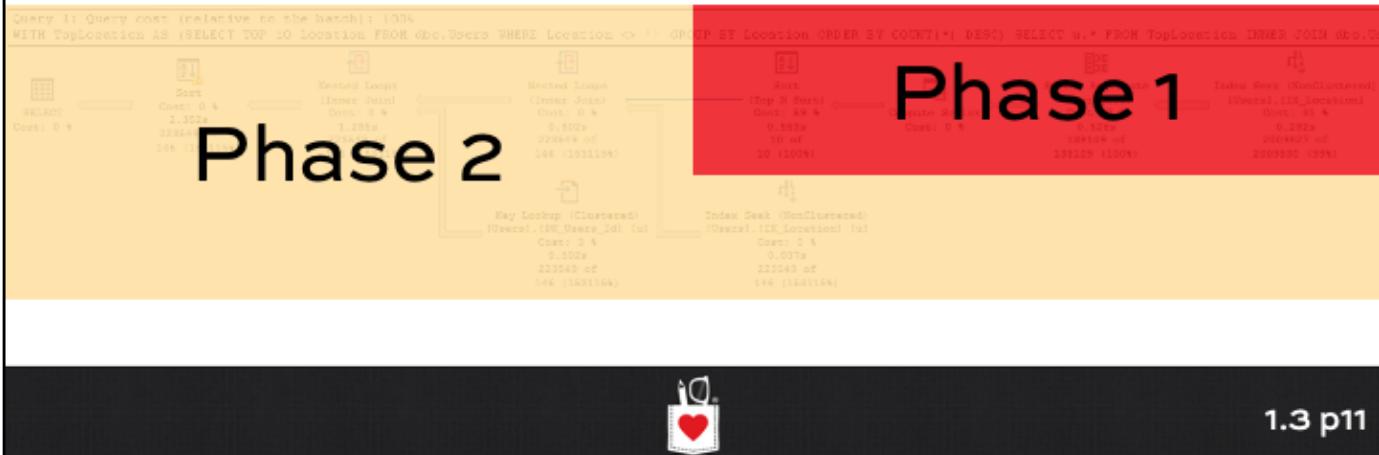
# Here, the estimates go wrong

Because SQL Server underestimates how many rows will come back for each location – it's using the average number of rows per average location.



# We need to separate these parts

Because SQL Server underestimates how many rows will come back for each location – it's using the average number of rows per average location.



## Before...

```
CREATE OR ALTER PROC dbo.usp_UsersInTop10Locations AS
BEGIN
    WITH TopLocation AS (SELECT TOP 10 Location
        FROM dbo.Users
        WHERE Location <> ''
        GROUP BY Location
        ORDER BY COUNT(*) DESC)
    SELECT u.*
        FROM TopLocation
        INNER JOIN dbo.Users u ON TopLocation.Location = u.Location
        ORDER BY DisplayName;
END
```



1.3 p12

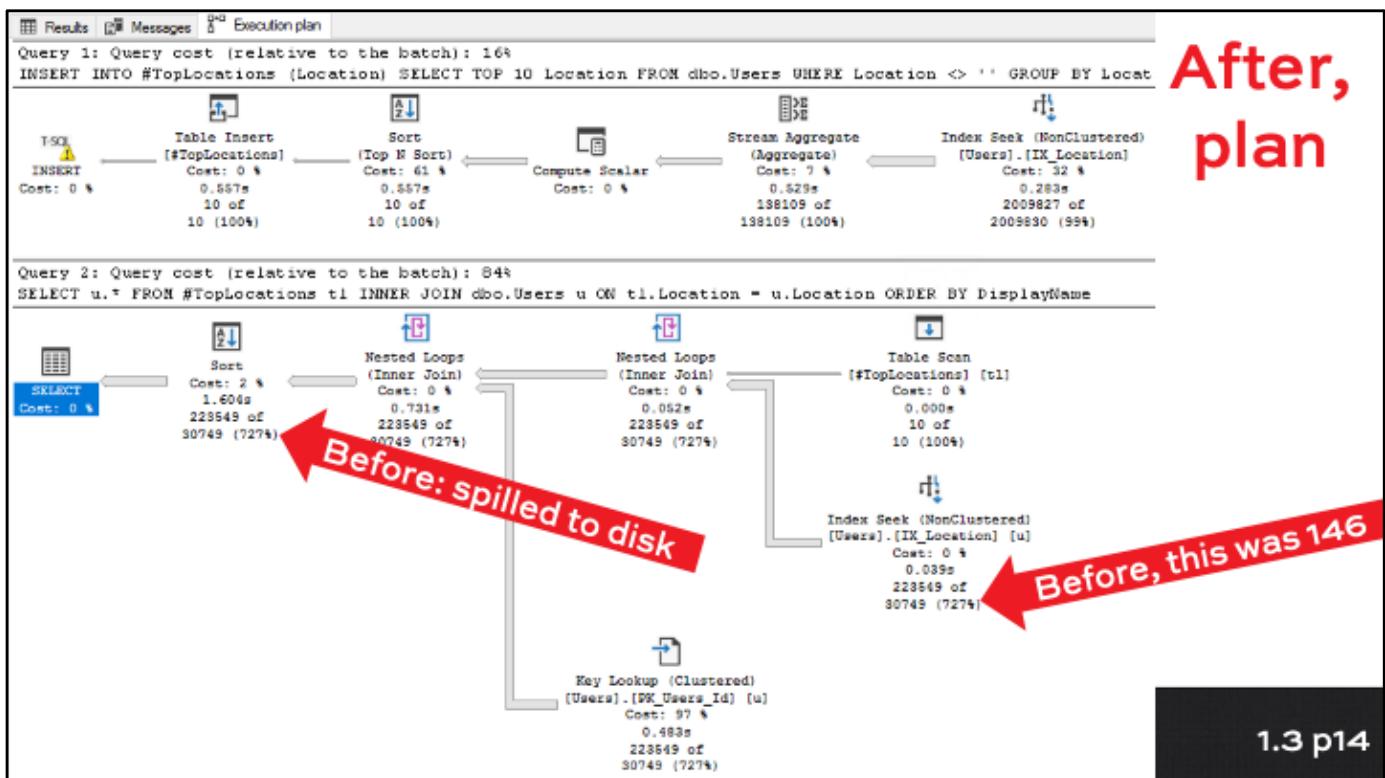
## After, with temp tables:

```
CREATE OR ALTER PROC dbo.usp_UsersInTop10Locations AS
BEGIN
    /* Phase 1 */
    CREATE TABLE #TopLocations (Location NVARCHAR(100));
    INSERT INTO #TopLocations (Location)
        SELECT TOP 10 Location
            FROM dbo.Users
            WHERE Location <> ''
            GROUP BY Location
            ORDER BY COUNT(*) DESC;

    /* Phase 2 */
    SELECT u.*
        FROM #TopLocations tl
        INNER JOIN dbo.Users u ON tl.Location = u.Location
        ORDER BY DisplayName;
END
GO
```

1.3 p13

After,  
plan



1.3 p14

## To see how, leave the temp table

```
/* Seeing the statistics */
/* Phase 1 */
CREATE TABLE #TopLocations (Location NVARCHAR(100));
INSERT INTO #TopLocations (Location)
SELECT TOP 10 Location
    FROM dbo.Users
    WHERE Location <> ''
    GROUP BY Location
    ORDER BY COUNT(*) DESC;

/* Phase 2 */
SELECT u.*
    FROM #TopLocations tl
    INNER JOIN dbo.Users u ON tl.Location = u.Location
    ORDER BY DisplayName;
GO
```

1.3 p15

# Get the temp table's name for sp\_BlitzIndex

The screenshot shows a SQL Server Management Studio (SSMS) window. The query pane contains the following T-SQL code:

```
USE tempdb;
GO
/* Get the temp table's full name: */
SELECT * FROM sys.all_objects WHERE name LIKE '%TopLocations%'

/* Then paste it into here: */
sp_BlitzIndex @TableName = '#TopLocations' _____
```

The results pane shows a single row of data:

name	object_id	type
#TopLocations	0000000000C93	-1590519843



1.3 p16

```

/* Then paste it into here: */
sp_BlitzIndex @TableName = '#TopLocations'

```

Results

Details: db_schema.table.index(indexid)	Definition: [Property] ColumnName (datatype maxbytes)	Secret Columns	Fillfactor	Usage Stats
1 [Database [tempdb] as of 2019-03-31 17:19 (xp_BlitzIndex(T...)]	[Property] ColumnName (datatype maxbytes)	From Your Community Volunteers	NULL	Server: SQL2017LAB1 Days Uptime: 0
2 dbo.#TopLocations	[HEAP]	[RID]	0	Reads: 1 (1 scan) Writes: 1

finding

Column Name	Found In	Type	Computed?	Length (max bytes)	Prec	Scale	Nullable?	Identity?	Replicated?	Sparse?	Filestream?	Collation
1 Location	0	nvarchar (100)		200	0	0	yes					SQL_Latin1_General_CI_AS

finding

1 No foreign keys.
--------------------

IT KNOWS!

Stat Name	Leading Column Name	Step Number	Range High Key	Range Rows	Equal Rows	Distinct Range Rows	Average Range Rows	Auto-Created
1 _WA_Sys_00000001_A13297DD	Location	1	Bangalore, Karnataka, India	0	1	0	1	1
2 _WA_Sys_00000001_A13297DD	Location	2	Chennai, Tamil Nadu, India	0	1	0	1	1
3 _WA_Sys_00000001_A13297DD	Location	3	China	0	1	0	1	1
4 _WA_Sys_00000001_A13297DD	Location	4	France	0	1	0	1	1
5 _WA_Sys_00000001_A13297DD	Location	5	Germany	0	1	0	1	1
6 _WA_Sys_00000001_A13297DD	Location	6	Hyderabad, Telengana, In...	0	1	0	1	1
7 _WA_Sys_00000001_A13297DD	Location	7	India	0	1	0	1	1
8 _WA_Sys_00000001_A13297DD	Location	8	London, United Kingdom	0	1	0	1	1
9 _WA_Sys_00000001_A13297DD	Location	9	Pune, Maharashtra, India	0	1	0	1	1
10 _WA_Sys_00000001_A13297DD	Location	10	United States	0	1	0	1	1

1.3 p17

## #tables have auto-created stats

Just like regular tables (although they're harder to see, since they drop)

Auto-updated just like regular tables after data is loaded

Just like real table stats, they can go bad

They can get REALLY strange due to plan reuse and statistics reuse:

<https://sqlperformance.com/2017/05/sql-performance/sql-server-temporary-object-caching>



1.3 p18

## When to use temp tables

If you need to refer to a result set multiple times in multiple queries

When you have a big query, and estimates go wrong partway in

If knowing what rows come out of Phase 1 will dramatically change Phase 2's query plan



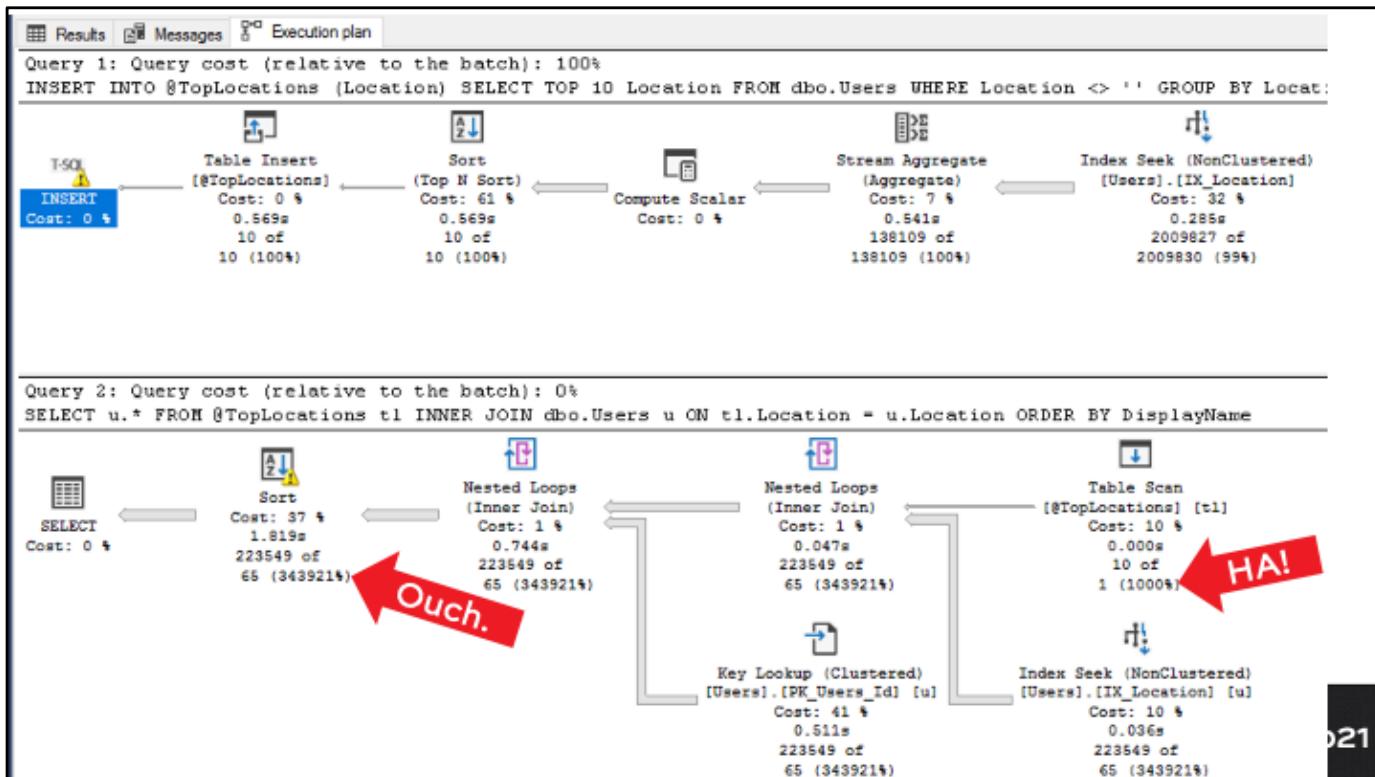
1.3 p19

## What about table variables?

```
CREATE OR ALTER PROC dbo.usp_UsersInTop10Locations AS
BEGIN
    /* Phase 1 */
    DECLARE @TopLocations TABLE (Location NVARCHAR(100));
    INSERT INTO @TopLocations (Location)
        SELECT TOP 10 Location
            FROM dbo.Users
            WHERE Location <> ''
            GROUP BY Location
            ORDER BY COUNT(*) DESC;

    /* Phase 2 */
    SELECT u.*
        FROM @TopLocations tl
        INNER JOIN dbo.Users u ON tl.Location = u.Location
        ORDER BY DisplayName;
END
GO
```

1.3 p20



## When to use table variables

Never because they don't have stats.



1.3 p22

# When to use table variables

~~Never because they don't have stats.~~

- When you purposely want an incorrect 1-row estimate.
- When you want an object that ignores transactions.
- When you want to reduce recompilations due to changing data.

SQL Server 2019:

- Gains deferred compilation for table variables
- But modifying table variable contents are still serial

More info in my Fundamentals of TempDB class.



1.3 p23



# Common Table Expressions (CTEs)

## I'm going to use a dumb example.

You would never write a query this terribad:

1. Total up all the Votes by UserId
2. Show the top voters in a specific location

But bear with me. It'll tell the story.



1.3 p25

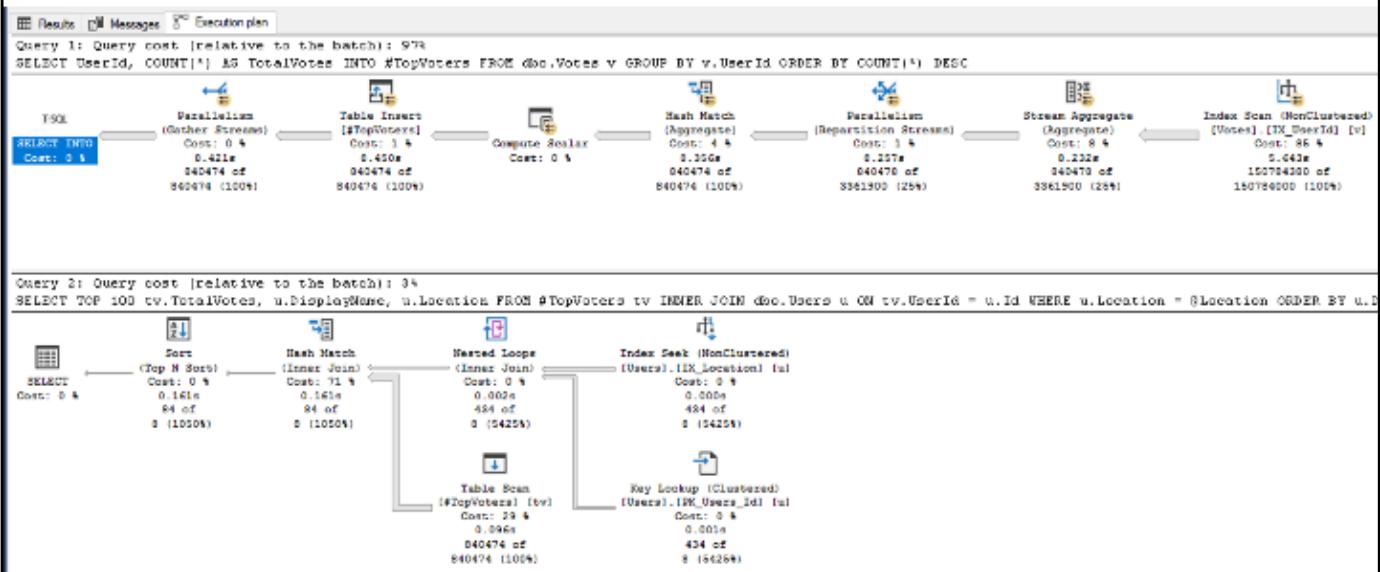
```
CREATE OR ALTER PROC dbo.usp_TopVotersInCity @Location NVARCHAR(40) AS
BEGIN
    /* Find the users who have left the most votes */
    SELECT UserId, COUNT(*) AS TotalVotes
        INTO #TopVoters
        FROM dbo.Votes v ← Scan all the votes
        GROUP BY v.UserId
        ORDER BY COUNT(*) DESC;

    SELECT TOP 100 tv.TotalVotes, u.DisplayName, u.Location
        FROM #TopVoters tv
        INNER JOIN dbo.Users u ON tv.UserId = u.Id
        WHERE u.Location = @Location ← Do some filtering
        ORDER BY u.DisplayName;
END
GO
```



1.3 p26

# Runs, but takes 8-10 seconds



# Common Table Expression (CTE)

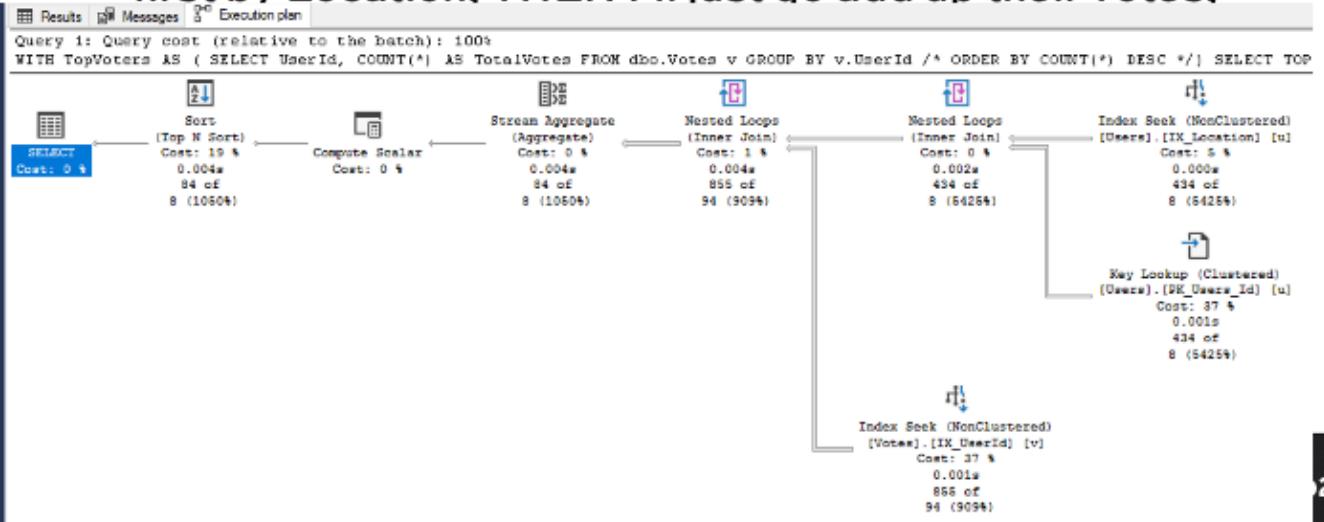
**WITH Alias AS:** lets SQL Server choose how to process this part

```
CREATE OR ALTER PROC dbo.usp_TopVotersInCity_CTE @Location NVARCHAR(40) AS
BEGIN
    WITH TopVoters AS (
        SELECT UserId, COUNT(*) AS TotalVotes
        FROM dbo.Votes v
        GROUP BY v.UserId
        /* ORDER BY COUNT(*) DESC */
    )
    SELECT TOP 100 tv.TotalVotes, u.DisplayName, u.Location
    FROM TopVoters tv
    INNER JOIN dbo.Users u ON tv.UserId = u.Id
    WHERE u.Location = @Location
    ORDER BY u.DisplayName;
END
```

1.3 p28

# Runs instantly

Because SQL Server says, lemme find the right Users first by Location, THEN I'll just go add up their votes.



29

## CTEs are great when...

You have multi-step queries

You want to break them up to be easier to read  
(easier than subqueries in some cases)

You're not sure which parts of the processing should be done first

When the row estimates from each phase may not matter  
(or when SQL Server can do a pretty good job at estimating them)

You don't need to refer to the results over and over in multiple steps  
(because they disappear – they're not materialized)

Advanced use case: recursive hierarchies



1.3 p30

Getting into specialist territory

# APPLY



# Stack Overflow example

Questions & answers (Posts) can be edited

History stored in the PostHistory table, so you can see edits over time:

The screenshot shows the Object Explorer on the left with 'dbo.PostHistory' selected. The 'Columns' node is expanded, showing columns: Id, PostHistoryTypeId, PostId, CreationDate, UserId, Comment, and Text. The 'Keys' node is also visible. On the right, a results grid displays the history for PostId 4673436. The first few rows show edits like 'fixed bug pointed out by fserb' and 'Added optimization proposed by @mikeycglo'. The last row shows a comment from '1413158' about checking if it's implemented yet.

Id	PostHistoryTypeId	PostId	CreationDate	UserId	Comment	Text
1	9883984	2	2011-01-12 20:02:34.957	2733	NULL	Build
2	10806723	5	2011-02-22 14:56:08.980	2733	fixed bug pointed out by fserb	Build
3	16445908	5	2011-09-16 16:55:36.770	2733	Added optimization proposed by @mikeycglo	Build
4	35251132	24	2013-01-30 20:25:25.950	NULL	Proposed by 1413158 approved by 664108, 454533, ...	NUL
5	35251131	5	2013-01-30 20:25:25.950	1413158	Added checking if it's implemented yet	Build
6	35306341	5	2013-01-31 19:22:33.530	2733	2 spaces	Build
7	36067672	5	2013-02-15 14:48:46.227	2733	fixed formatting	Build
8	45933641	24	2013-08-14 14:04:26.517	NULL	Pinned by 2295405 announced by 794877, 1765868	NULL

PostHistoryTypeId: title changed, post closed, protected, tweeted, etc.



1.3 p32

```
SELECT CreationDate, Text
FROM dbo.PostHistory
WHERE PostId = 41948428
AND PostHistoryTypeId IN (1, 4, 7)
ORDER BY CreationDate;
```

150 %

Results Messages

	CreationDate	Text
1	2017-01-31 01:41:47.690	How to make a 3D velocity field to look more professional?
2	2017-01-31 02:26:35.760	How to make a 3D velocity field to look more professional and informative?
3	2017-01-31 02:49:59.600	How to make a 3D velocity field plot to look more professional and informative?
4	2017-01-31 04:11:12.470	How to analyze a 3D velocity vector field and make the plots more informative and professional?
5	2017-01-31 11:46:37.467	How to plot a 3D velocity field more professionally?
6	2017-01-31 15:18:10.533	How to plot a 3D velocity field and its stream ribbons?
7	2017-01-31 17:41:39.843	How to plot a 3D velocity field and its stream ribbons plot?
8	2017-02-01 04:24:48.710	How to visualize a 3D velocity field in order to see vorticity?
9	2017-02-01 21:33:50.400	How to plot the curl of a 3D velocity field?
10	2017-02-03 00:23:53.370	How to plot the curl of a 3D velocity vector field?
11	2017-02-03 02:59:48.860	Plot the curl of a 3D velocity vector field?
12	2017-02-03 03:27:53.270	How to plot the curl of a 3D velocity vector field?
13	2017-02-05 01:25:00.533	How to calculate and plot the properties of a 3D velocity vector field?
14	2017-02-05 01:56:53.117	How to analyze and visualize a 3D velocity field / streams?

1.3 p33

# Our query requirements, v1

For a user, show their top 10 questions by score.

```
/* Go get a user's top 10 questions by score */

SELECT TOP 10 q.Id AS QuestionId, q.Title AS CurrentTitle, q.Score
FROM dbo.Posts q
WHERE q.OwnerUserId = 1031417
AND q.PostTypeId = 1 /* Question */
ORDER BY q.Score DESC;
```

150 %

Results Messages

QuestionId	CurrentTitle	Score
1	9999727	584
2	8450472	65
3	13785498	40
4	15065010	39
5	14751973	37
6	13468286	35
7	8393424	21
8	15779185	17
9	15553493	16
10	10085273	15

1.3 p34

## Our query requirements, v2

For a user, show their top 10 questions by score...

And add rows for the first 5 edited titles (PostHistoryTypeId = 4)

Our original query:

```
/* Go get a user's top 10 questions by score */

SELECT TOP 10 q.Id AS QuestionId, q.Title AS CurrentTitle, q.Score
    FROM dbo.Posts q
   WHERE q.OwnerUserId = 1031417
     AND q.PostTypeId = 1 /* Question */
   ORDER BY q.Score DESC;
```

But now we need to add joins to PostHistory,  
but we only want to show the first 5 rows.



1.3 p35

# Enter APPLY.

```
/* Add rows for the first 5 titles per post... */

SELECT q.Id AS QuestionId, q.Title AS CurrentTitle, q.Score,
       qTitles.CreationDate AS TitleEditDate, qTitles.Text AS TitleText
  FROM dbo.Posts q

  CROSS APPLY (SELECT TOP 5 ph.CreationDate, ph.Text
               FROM dbo.PostHistory ph
              WHERE ph.PostId = q.Id
                AND ph.PostHistoryTypeId = 4
              ORDER BY ph.CreationDate) qTitles

 WHERE q.OwnerUserId = 1031417
   AND q.PostTypeId = 1 /* Question */
 ORDER BY q.Score DESC, q.Id, qTitles.CreationDate;
```



1.3 p36

# Like subquery or join, but different.

```
CROSS APPLY (SELECT TOP 5 ph.CreationDate, ph.Text  
    FROM dbo.PostHistory ph  
    WHERE ph.PostId = q.Id  
        AND ph.PostHistoryTypeId = 4  
    ORDER BY ph.CreationDate) qTitles
```

Subqueries can't return multiple fields – this can.

Joins require you to define relationships and take either one, or all of the rows – this lets you pick and choose.

Subqueries can struggle with correlation performance – this doesn't.\*

\* If you don't have indexes to support this, though, it'll be slow.



1.3 p37

```

/* Add rows for the first 5 titles per post... */

SELECT q.Id AS QuestionId, q.Title AS CurrentTitle, q.Score,
       qTitles.CreationDate AS TitleEditDate, qTitles.Text AS TitleText
  FROM dbo.Posts q

  CROSS APPLY (SELECT TOP 5 ph.CreationDate, ph.Text
               FROM dbo.PostHistory ph
              WHERE ph.PostId = q.Id
                AND ph.PostHistoryTypeId = 4
              ORDER BY ph.CreationDate) qTitles

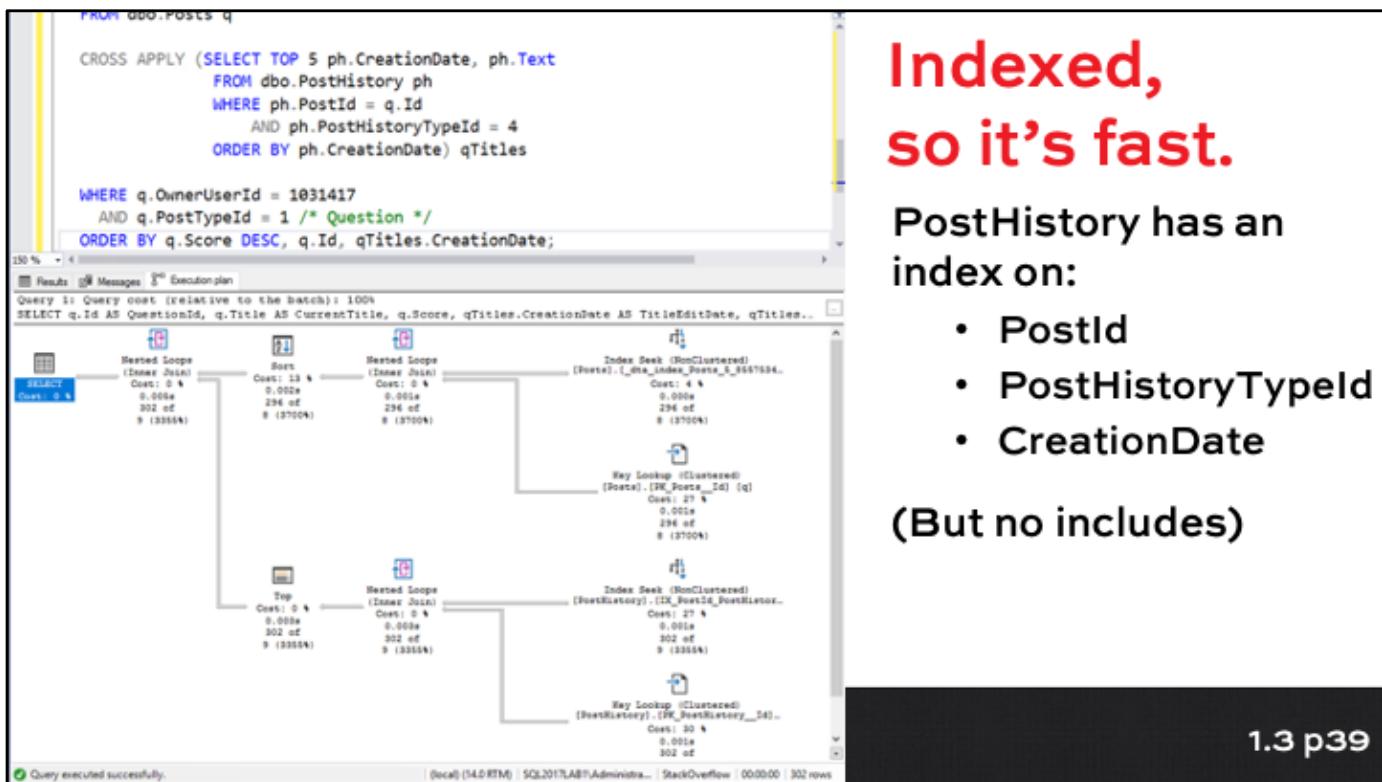
 WHERE q.OwnerUserId = 1031417
   AND q.PostTypeId = 1 /* Question */
 ORDER BY q.Score DESC, q.Id, qTitles.CreationDate;

```

Results Messages

QuestionId	CurrentTitle	Score	TitleEditDate	TitleText
1	__proto__ VS. prototype in JavaScript	584	2012-04-28 17:24:38.603	What is the difference between __proto__ and prototype?
2	__proto__ VS. prototype in JavaScript	584	2012-07-30 00:21:32.177	Java Script - What is the difference between __proto__ and prototype?
3	__proto__ VS. prototype in JavaScript	584	2013-04-10 16:11:05.233	What is the difference between __proto__ and prototype?
4	__proto__ VS. prototype in JavaScript	584	2013-04-10 16:29:24.743	What is the difference between __proto__ and prototype in JavaScript?
5	__proto__ VS. prototype in JavaScript	584	2013-06-13 13:08:50.883	What is the difference between __proto__ and prototype in JavaScript?
6	How to format print output into fixed width?	65	2012-10-30 21:23:01.470	How to format print to fixed width of string in python?
7	How to format print output into fixed width?	65	2012-11-30 07:20:31.887	python - How to format the print call into a fixed width output?
8	How to format print output into fixed width?	65	2013-04-19 12:51:58.850	python - How to format print output into a fixed width output?
9	How to format print output into fixed width?	65	2013-04-20 21:36:28.567	python - How to format print output into fixed width?
10	How to format print output into fixed width?	65	2013-05-04 16:09:40.837	python 2.7 - How to format print output into fixed width?
11	What is the difference between 'sh' and 'source'?	40	2015-04-09 15:03:19.427	What is the difference in Linux between 'source' and 'sh'?
12	What is the difference between 'sh' and 'source'?	40	2015-12-25 23:37:30.307	What is the difference in a Unix shell between 'source' and 'sh'?

1.3 p38



# Indexed, so it's fast.

PostHistory has an index on:

- PostId
- PostHistoryTypeId
- CreationDate

(But no includes)

1.3 p39

## This beats a subquery when you need >1 col

```
SELECT u.DisplayName,
      (
        SELECT TOP 3 *
        FROM dbo.Posts AS p
        WHERE p.OwnerUserId = u.Id
      ) AS TopThreePosts
  FROM dbo.Users AS u;
```

131 % ▾ Messages

Msg 116, Level 16, State 1, Line 10  
Only one expression can be specified in the select list  
when the subquery is not introduced with EXISTS.



1.3 p40

## It's not all magic and fairies

When writing queries using Apply, this stuff generally works better

- The base table should be the smaller table
- If possible, use a Primary Key or other Unique index to correlate

You still need indexes, pal

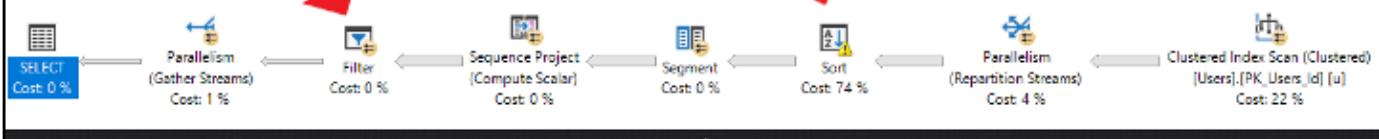
- The WHERE clause is obvious
- Support your ORDER BY
- Cover any select list columns



1.3 p41

# CTEs also need indexes

```
WITH slow_cte
AS
(
    SELECT u.Id,
           u.DisplayName,
           ROW_NUMBER() OVER (PARTITION BY u.Reputation
                               ORDER BY u.LastAccessDate DESC) AS rn
    FROM dbo.Users AS u
)
SELECT *
FROM slow_cte
WHERE slow_cte.rn <= 3;
```



1.3 p42

## P.O.C. – No, not like your car

**Partition:** key columns should lead with Partition by columns

**Order:** next should come any columns you order by

**Cover:** where sane and reasonable, cover selected columns

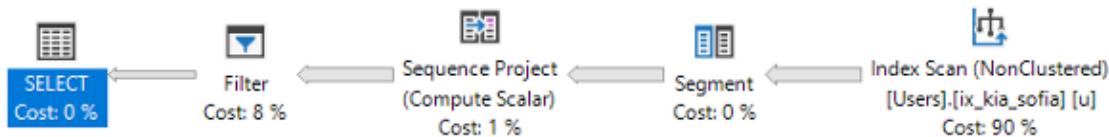
```
WITH slow_cte
AS
(
    SELECT u.Id,
           u.DisplayName,
           ROW_NUMBER() OVER ( PARTITION BY u.Reputation
                                ORDER BY u.LastAccessDate DESC ) AS rn
    FROM   dbo.Users AS u
```

```
CREATE INDEX ix_kia_sofia
ON dbo.Users ( Reputation, LastAccessDate DESC )
INCLUDE ( DisplayName );
```

1.3 p43

# Happy little plan

This plan does a whole lot less work



No need to sort data – all we have to do is assign row numbers as they come out of the index, and then filter



1.3 p44

# Lens cap Recap



# Our cameras: temp tables, CTEs, APPLY.

- Help to separate units of work (especially SELECT \* soon)
- Make queries easier to read
- Avoid (or embrace) repeated processing

CTEs are also great for

- Recursing over hierarchies
- Paging queries

APPLY is also great for

- Top N per group
- Working with XML



1.3 p46

	Temp tables & table variables	CTEs & APPLY
Scope (lasts for)	Your session, with gotchas	Just one statement
Materialized to disk*	Yes	No
Executed	Just once, when you insert	Can be multiple times even in one query
Has statistics (which can be good or bad)	Temp tables: yes Table variables: no, til 2019	No

\* Not every temp object gets written to disk, and CTE/APPLY can cause TempDB spills just like any other query operation



1.3 p47

## These are more advanced tools

Like everything in SQL Server, they aren't fast or efficient by themselves – it's up to you to design queries and indexes to make them fast and efficient

These tools aren't always the best in every situation, either – but now that you know about them, it's also up to you to test them to see where they fit into tuning efforts

In the end, the optimizer may just laugh at your hard work



1.3 p48