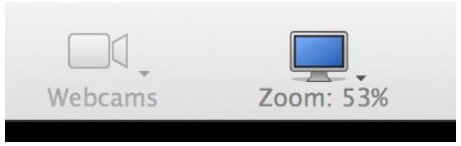
Advanced Common Table Expressions

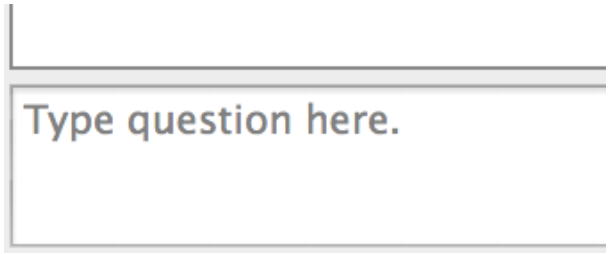
Much more than just data paging

Steve Stedman, Founder/Owner of Stedman Solutions, LLC.
Moderated by Andrea Allred

Technical Assistance



Maximize your screen with the zoom button on the top of the presentation window



Type your questions in the question pane on the right side

Thank You

Presenting Sponsors

Microsoft

Gain insights through familiar tools while balancing monitoring and managing user created content across structured and unstructured sources.

www.microsoft.com

CISCO

Unifying computer, storage, networking, and virtualization, Cisco UCS is the optimal database and business intelligence platform for SQL Server.

www.cisco.com

DELL Software

Solutions from Dell help you monitor, manage, protect and improve your SQL Server environment.

www.software.dell.com

Supporting Sponsors

SQL SENTRY

hp

solarwinds

PASS SUMMIT 2015

October 27-30 / Seattle, WA

Planning on attending PASS Summit 2015? Start saving today!

- The world's largest gathering of SQL Server & BI professionals
- Take your SQL Server skills to the next level by learning from the world's SQL Server experts, in over 190 technical sessions
- Over 5000 attendees, representing 2000 companies, from 52 countries, ready to network & learn

Contact your Local or Virtual Chapter for an additional $150 discount.

$1795

until July 12th, 2015

Steve Stedman

- Founder / Owner, Stedman Solutions, LLC
- PASS Chapter Leader
- Author of SQL Common Table Expressions
- SQL Saturday Speaker

http://stevestedman.com/

@SqlEmt

https://www.linkedin.com/in/stevestedman

Advanced Common Table Expressions

Much more than just data paging

Steve Stedman, Founder/Owner of Stedman Solutions, LLC.

When I am not working with SQL Server I like to go boating with my family.

Advanced Common Table Expressions

Agenda

- Recursive Queries
- Hierarchical Recursive Data
- Manipulating Data
- Common Use Cases
- CTE Performance Considerations
- Classic Recursive Algorithms

Recursive Queries

Considered recursive when the CTE references itself

Recursion stops

- When the recursive query produces no results
- Or specify MAXRECURSION

Uses

- Hierarchical listing of categories
- Recursive calculations
- Much, much more…

Recursion Overview

Sum the numbers from 1 to 10 without recursion

55 = 10 + 9 + 8 + 7 + 6 + 5 + 4 +3 + 2 + 1

Sum the numbers from 1 to 10 recursively

55 = 10 + (sum of numbers 1 to 9)

55 = 10 + (9 + (sum of numbers 1 to 8))

55 = 10 + (9 + (8 + (sum of numbers 1 to 7)))

Eventually we get to:

55 = 10 + (9 + (8 + (7 + (6 + (5 + (4 + (3 + (2 + 1)))))))

Recursive Terminology

Anchor Query

- Start the recursion
- One or more anchor queries

Recursive Query

- The part that repeats and references the CTE by name
- One or more recursive queries

MAXRECURSION

- The number of times to repeat the recursive query
- Default is 100
- MAXRECURSION of 0 implies no maximum

11

Demo: Recursive CTE

```
;WITH DepartmentCTE(DeptId, Department, Parent, Lvl)
AS (
```

Step 1. Declare the CTE and Columns

Demo: Recursive CTE

```
;WITH DepartmentCTE(DeptId, Department, Parent, Lvl)
AS (
  SELECT id AS DeptId, Department, parent, 0 AS Lvl
     FROM Departments
  WHERE parent IS NULL
```

Step 2 – Add the Anchor Query

Demo: Recursive CTE

```
;WITH DepartmentCTE(DeptId, Department, Parent, Lvl)
AS (
  SELECT id AS DeptId, Department, parent, 0 AS Lvl
     FROM Departments
  WHERE parent IS NULL
   UNION ALL -- and now for the recursive part
```

Step 3 – Add the UNION ALL to connect to the recursive query

Demo: Recursive CTE

```
;WITH DepartmentCTE(DeptId, Department, Parent, Lvl)
AS (
  SELECT id AS DeptId, Department, parent, 0 AS Lvl
    FROM Departments
  WHERE parent IS NULL
   UNION ALL -- and now for the recursive part
  SELECT d.id AS DeptId, d.Department, d.parent, DepartmentCTE.Lvl + 1 AS Lvl
    FROM Departments d
   INNER JOIN DepartmentCTE ON DepartmentCTE.DeptId = d.parent
)
```

Step 4 – Add the recursive Query

Demo: Recursive CTE

```
;WITH DepartmentCTE(DeptId, Department, Parent, Lvl)
AS (
  SELECT id AS DeptId, Department, parent, 0 AS Lvl
    FROM Departments
  WHERE parent IS NULL
   UNION ALL -- and now for the recursive part
  SELECT d.id AS DeptId, d.Department, d.parent, DepartmentCTE.Lvl + 1 AS Lvl
    FROM Departments d
    INNER JOIN DepartmentCTE ON DepartmentCTE.DeptId = d.parent
)
SELECT *
  FROM DepartmentCTE
 ORDER BY parent;
```

Recursive Demo

17

Recursive CTE Notes

Recursion stops

- When the recursive query produces no results
- Or specify MAXRECURSION

Using TSQL functions for recursion allows for 32 levels of recursion

Using CTE for recursion allows for 32767 levels of recursion in the MAXRECURSION option, but much more if you set MAXRECURSION to 0.

- I have confirmed up to 100,000,000 levels of recursion.

Hierarchical Recursive Data
Recursion Made To Look Good

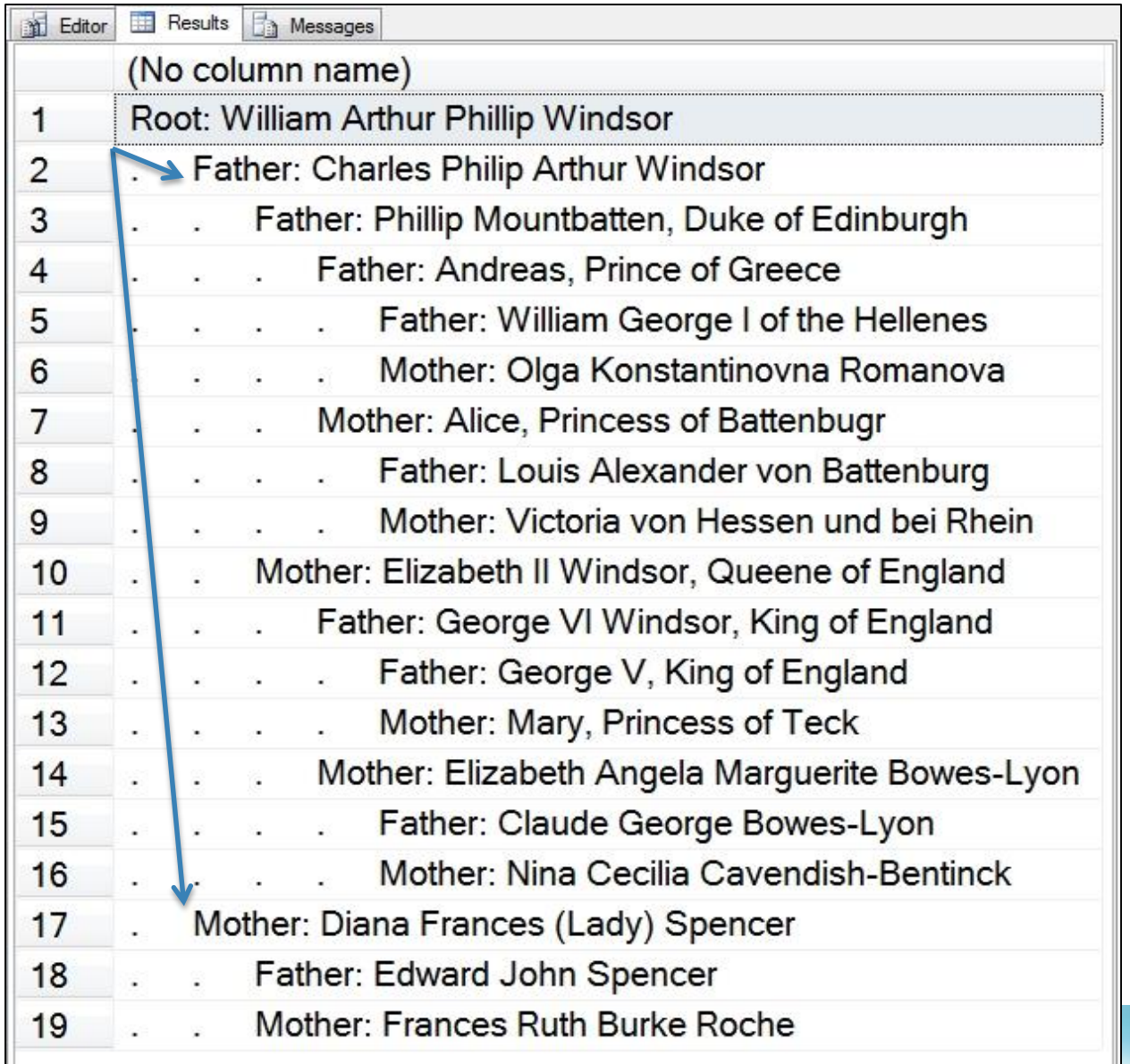
Editor | Results | Messages

	Department
1	Camping
2	. Backpacks
3	. Cooking
4	. Sleeping Bags
5	. Tents
6	. . 1 Person
7	. . 2 Person
8	. . . Backpacking
9	. . . Family Camping
10	. . . Mountaineering
11	. . . . Lightweight
12	. . . . Standard
13	. . . . Ultra-lightweight
14	. . 3 Person
15	. . 4 Person
16	Clearance

	id	Department	parent	lvl
1	1	Camping	NULL	1
2	2	Cycle	NULL	1
3	3	Snowsports	NULL	1
4	4	Fitness	NULL	1
5	28	Gifts	NULL	1
6	29	Clearance	NULL	1
7	15	Running	4	2
8	16	Swimming	4	2
9	17	Yoga	4	2
10	12	Ski	3	2
11	13	Snowboard	3	2
12	14	Snowshoe	3	2
13	9	Bikes	2	2
14	10	Helmets	2	2
15	11	Locks	2	2
16	5	Tents	1	2
17	6	Backpacks	1	2
18	7	Sleeping ...	1	2
19	8	Cooking	1	2
20	18	1 Person	5	3
21	19	2 Person	5	3
22	20	3 Person	5	3
23	21	4 Person	5	3

19

Hierarchy with multiple recursive queries is possible



Hierarchy with advanced formatting

	(No column name)
1	. . . . Father: William George I of the Hellenes
2	. . . Father: Andreas, Prince of Greece
3	. . . . Mother: Olga Konstantinovna Romanova
4	. . Father: Phillip Mountbatten, Duke of Edinburgh
5	. . . . Father: Louis Alexander von Battenburg
6	. . . Mother: Alice, Princess of Battenbugr
7	. . . . Mother: Victoria von Hessen und bei Rhein
8	. Father: Charles Philip Arthur Windsor
9	. . . . Father: George V, King of England
10	. . . Father: George VI Windsor, King of England
11	. . . . Mother: Mary, Princess of Teck
12	. . Mother: Elizabeth II Windsor, Queene of England
13	. . . . Father: Claude George Bowes-Lyon
14	. . . Mother: Elizabeth Angela Marguerite Bowes-Lyon
15	. . . . Mother: Nina Cecilia Cavendish-Bentinck
16	Top: William Arthur Phillip Windsor
17	. . Father: Edward John Spencer
18	. Mother: Diana Frances (Lady) Spencer
19	. . Mother: Frances Ruth Burke Roche

21

Recursive Heirarchy Demo

22

Manipulating Data

Insert

- Inserting data into a CTE.

Update

- Find out what happens if we update a CTE.

Delete

- Does deleting from a CTE delete from the referenced tables?

23

Deleting From a CTE

DELETE FROM CTE;

- Are the following SQL Statements valid?
- Can you delete from a CTE?
- What does that mean?

```
;WITH CustomerCTE AS
(
  SELECT *
    FROM Customer
   WHERE LastName like 'Williams'
)
DELETE FROM CustomerCTE;
```

```
;WITH CustomerCTE AS
(
 SELECT c.*
   FROM Customer AS c
  INNER JOIN SalesInvoice AS si
     ON si.CustomerID = c.CustomerID
  WHERE c.LastName like 'Williams'
)
DELETE FROM CustomerCTE;
```


24

Inserting To a CTE

INSERT INTO CTE;

1. Are the following SQL Statements valid?
2. Can you INSERT INTO a CTE?
3. What does that mean?

```
;WITH CustomerCTE AS
(
  SELECT *
    FROM Customer
   WHERE LastName like 'Stedman'
)
INSERT INTO CustomerCTE
   (CustomerID,FirstName,LastName)
VALUES (99999, 'Steve', 'Stedman');
```

```
;WITH CustomerCTE AS
(
 SELECT c.*
   FROM Customer AS c
  INNER JOIN SalesInvoice AS si
     ON si.CustomerID = c.CustomerID
  WHERE c.LastName like 'Stedman'
)
INSERT INTO CustomerCTE
   (CustomerID, FirstName, LastName)
VALUES (99999, 'Steve', 'Stedman');
```


25

Updating a CTE

UPDATE CTE;

1. Are the following SQL Statements valid?
2. Can you UPDATE a CTE?
3. What does that mean?

```
;WITH CustomerCTE AS
(
  SELECT c.*
    FROM Customer AS c
   WHERE c.LastName like 'Williams'
)
UPDATE CustomerCTE
   SET LastName = 'Willie';
```

```
;WITH CustomerCTE AS
(
 SELECT c.*, si.Comment
   FROM Customer AS c
  INNER JOIN SalesInvoice AS si
     ON si.CustomerID = c.CustomerID
  WHERE c.LastName like 'Williams'
)
UPDATE CustomerCTE
   SET Comment = 'Some Comment',
       CompanyName = 'Willies Toys';
```


26

INSERT, UPDATE, DELETE Notes

Insert, Update, and Delete all work against a CTE with only a single base table.

Delete does not work for any CTE with multiple base tables referenced in the CTE query.

Insert and Update works against a CTE with multiple base tables as long as only one base table is being udpated.

CTE Common Use Case 1: Alternative to a Numbers Table

```
;WITH Numbers (N) AS
(
  SELECT 1
   UNION ALL
  SELECT 1 + N FROM Numbers
   WHERE N < 1000
)
SELECT *
  FROM Numbers
OPTION (MAXRECURSION 1000);
```

CTE Common Use Case 2: Finding Holes

```
;WITH Numbers (N) AS
(  SELECT 0
   UNION ALL
  SELECT 1 + N FROM Numbers WHERE N < 23
), OrderHours (HourOfDay, TheCount) AS
(  SELECT DATEPART(HOUR, OrderDate) as HourOfDay, COUNT(1) AS TheCount
     FROM SalesInvoice
    WHERE OrderDate < '02/01/2006'
    GROUP BY DATEPART(HOUR, OrderDate)
)
SELECT n.N AS HourOfDay, ISNULL(oh.TheCount, 0) OrderCount
  FROM OrderHours oh
 RIGHT JOIN Numbers n ON n.N = oh.HourOfDay
 ORDER BY TheCount ASC;
```

CTE Common Use Case 2: Scrubbing Duplicates

```
WITH CustomerCTE AS
(
  SELECT *,
  ROW_NUMBER() OVER(PARTITION BY LastName,
                    FirstName ORDER BY CustomerID) AS DupNum
  FROM Customer
)
DELETE
  FROM CustomerCTE
 WHERE DupNum > 1;
```

CTE Performance Considerations

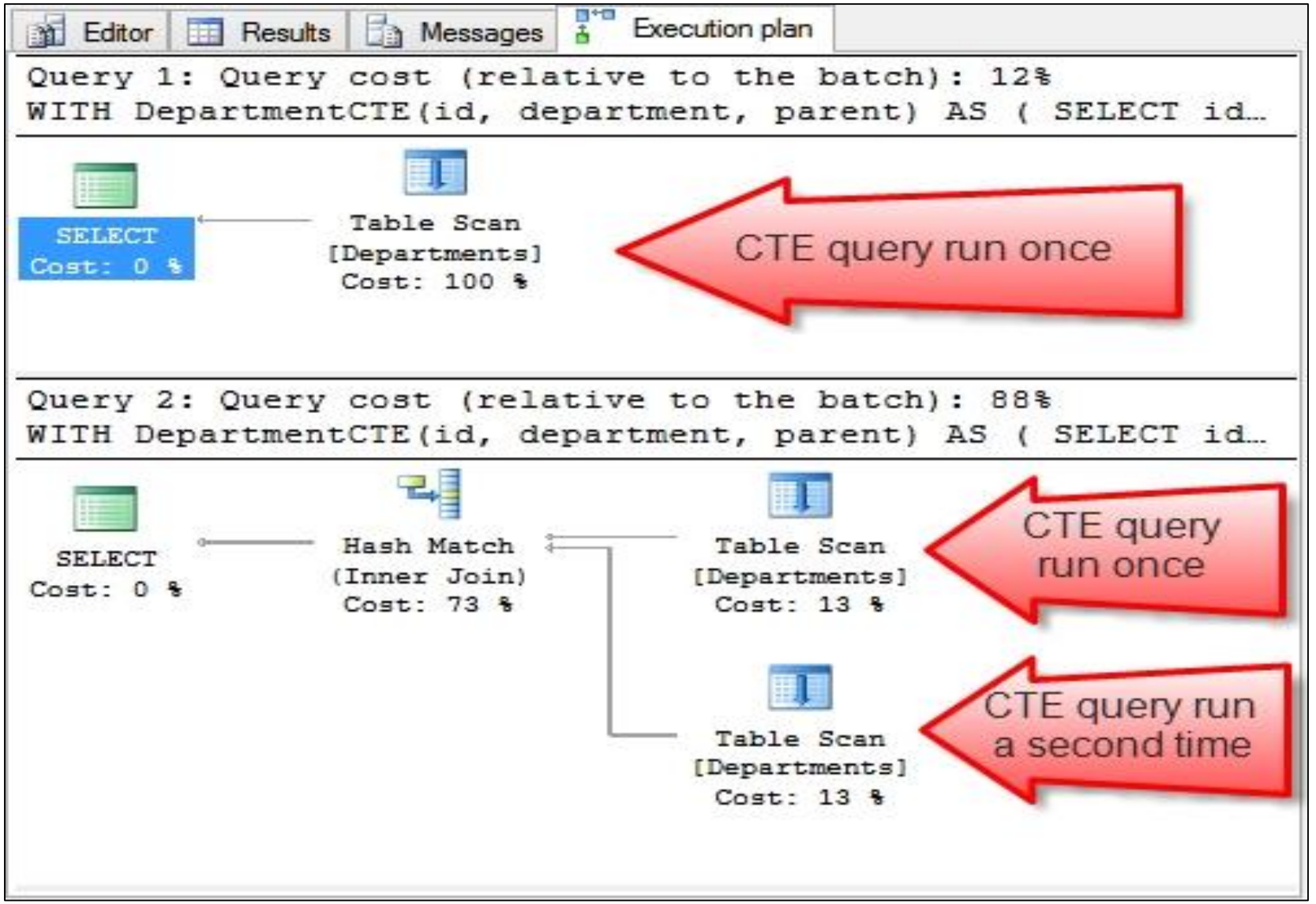
Non-Recursive Performance

- Multiple references to a single CTE
- CTEs vs. Derived Tables
- Multiple CTEs in a query

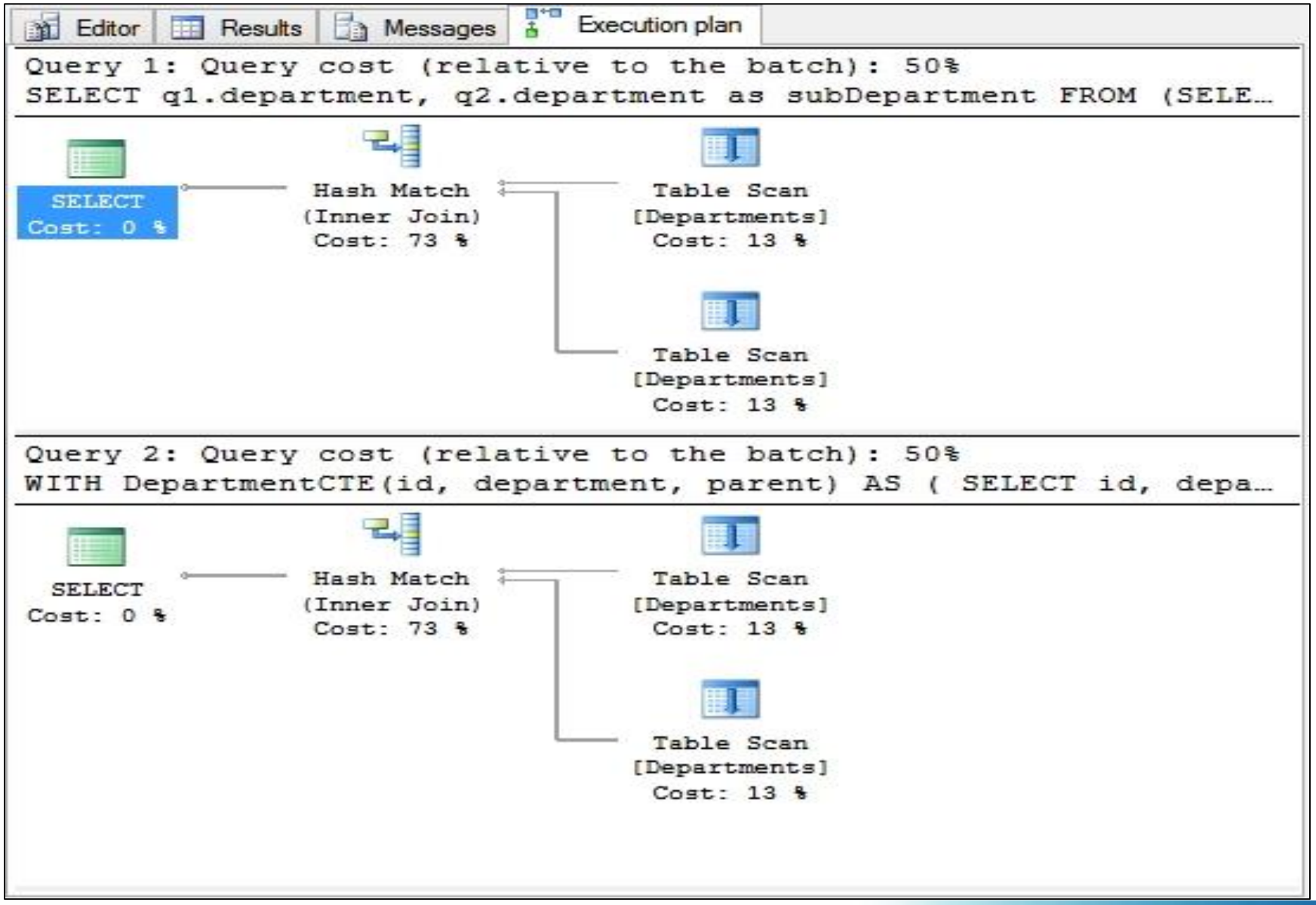
Recursive Performance

- Deep Recursion

Multiple references to a single CTE



CTEs vs. Derived Tables



Nested CTEs

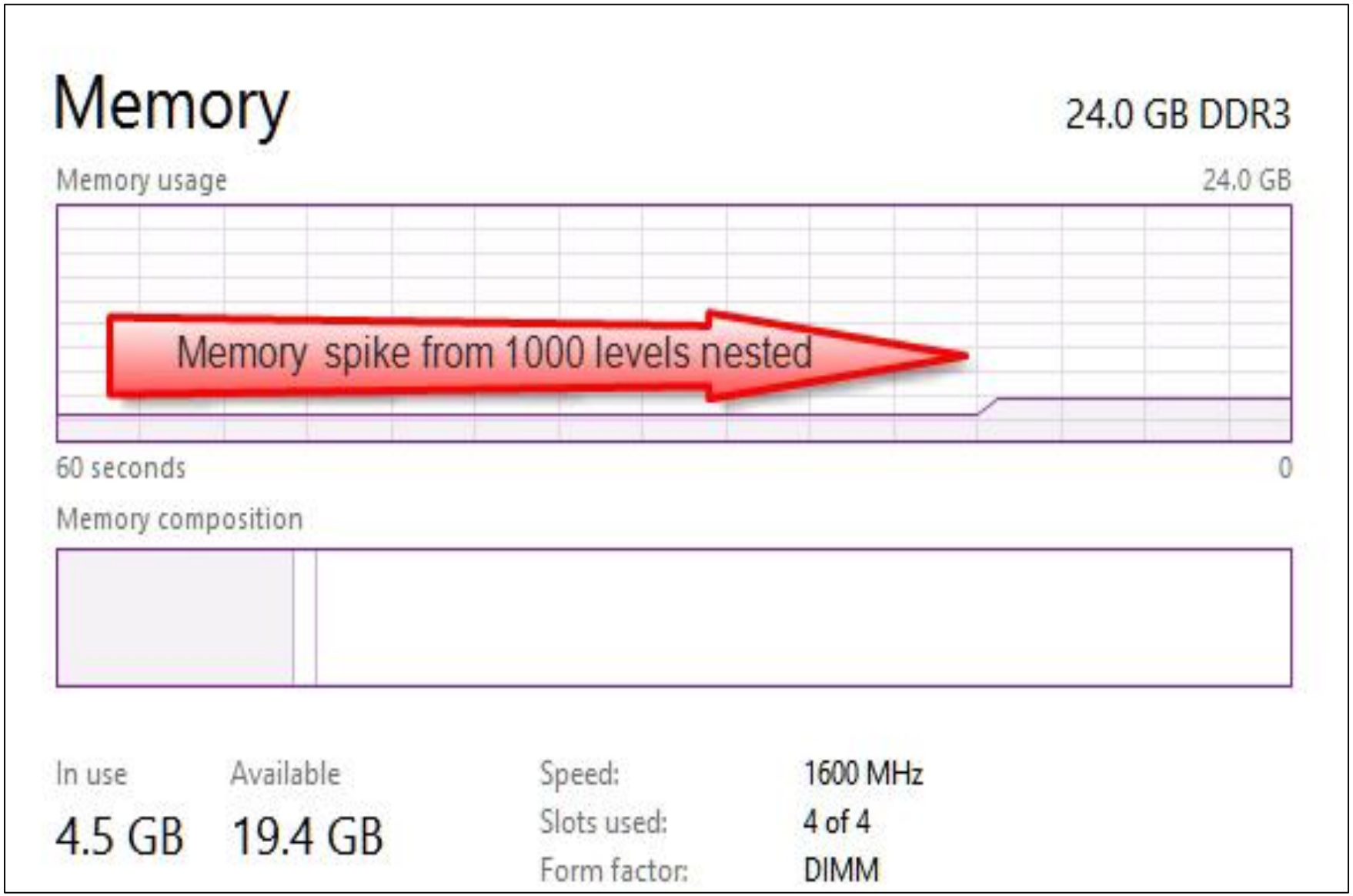
Take it to the extreme

```
with cte0 as
(  select 1 as num )
, cte1 AS (SELECT * FROM cte0)
, cte2 AS (SELECT * FROM cte1)
```

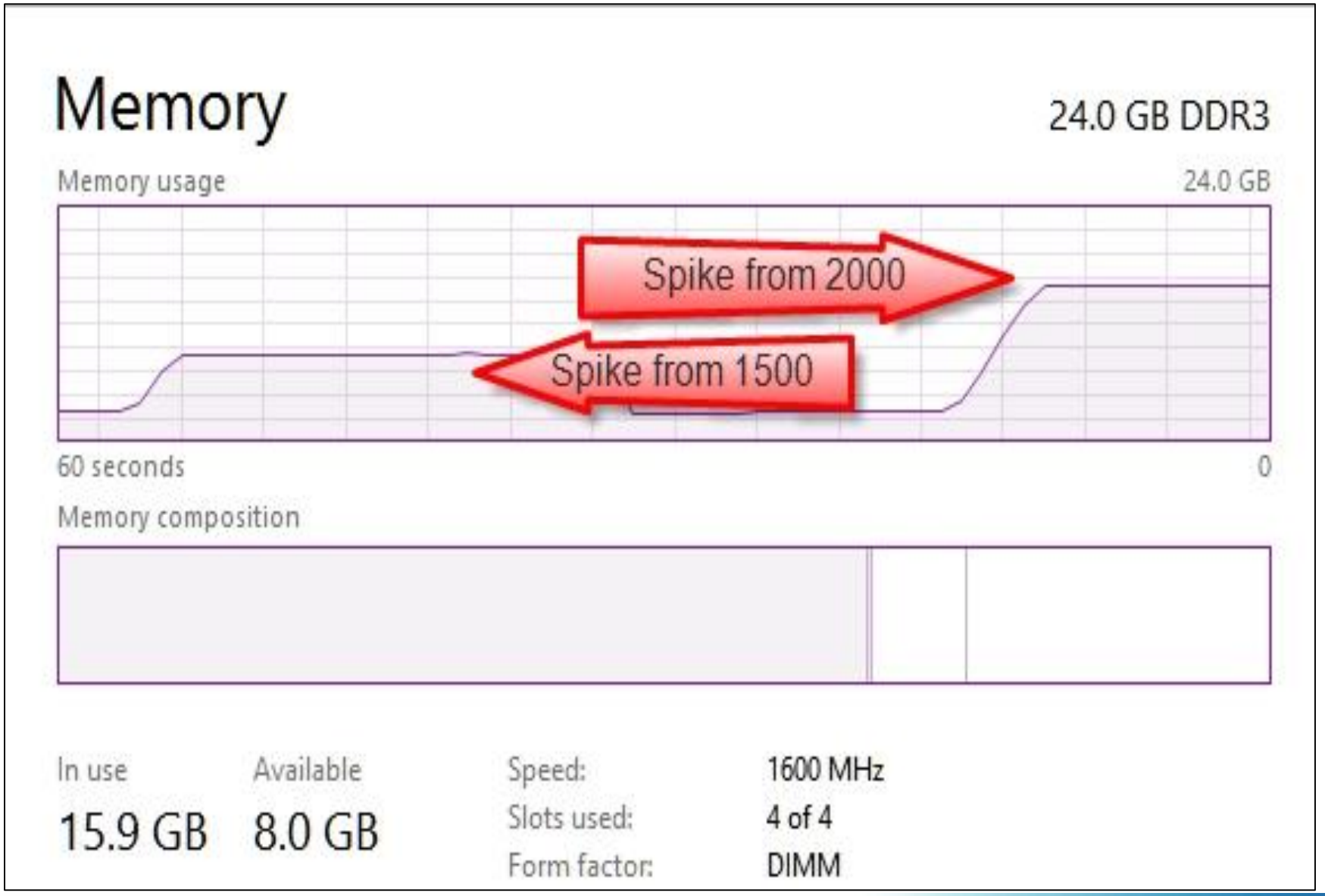
Repeated from 2 to 254.

```
, cte255 AS (SELECT * FROM cte254)
, cte256 AS (SELECT * FROM cte255)
, cte257 AS (SELECT * FROM cte256)
select * from cte257;
```

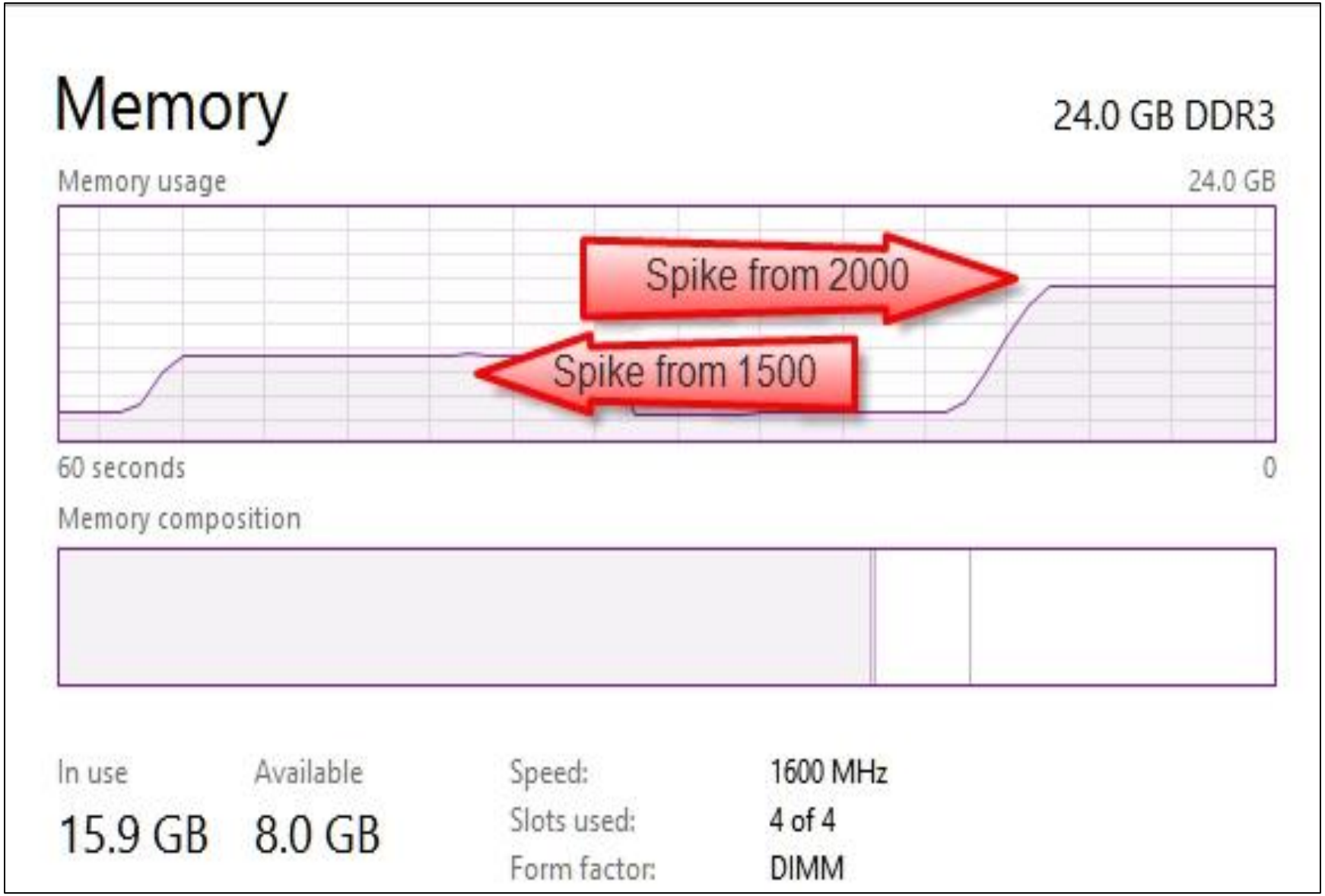
Nested CTEs


Memory
24.0 GB DDR3
Memory usage
24.0 GB
Memory spike from 1000 levels nested
60 seconds
0
Memory composition
In use
4.5 GB
Available
19.4 GB
Speed:
1600 MHz
Slots used:
4 of 4
Form factor:
DIMM

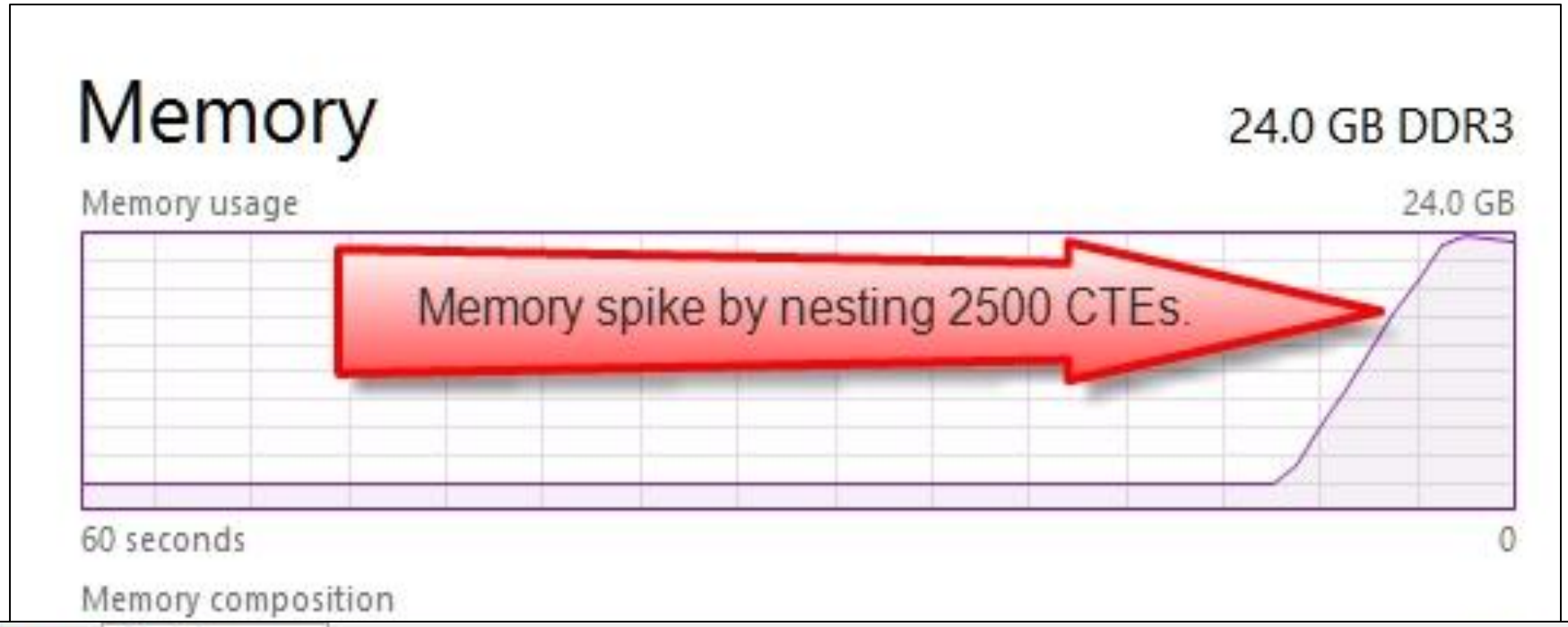
Nested CTEs


Memory
24.0 GB DDR3
Memory usage
24.0 GB
Spike from 2000
Spike from 1500
60 seconds
0
Memory composition
In use
15.9 GB
Available
8.0 GB
Speed:
1600 MHz
Slots used:
4 of 4
Form factor:
DIMM

Nested CTEs


Memory
24.0 GB DDR3
Memory usage
24.0 GB
Spike from 2000
Spike from 1500
60 seconds
0
Memory composition
In use
15.9 GB
Available
8.0 GB
Speed:
1600 MHz
Slots used:
4 of 4
Form factor:
DIMM

Nested CTEs

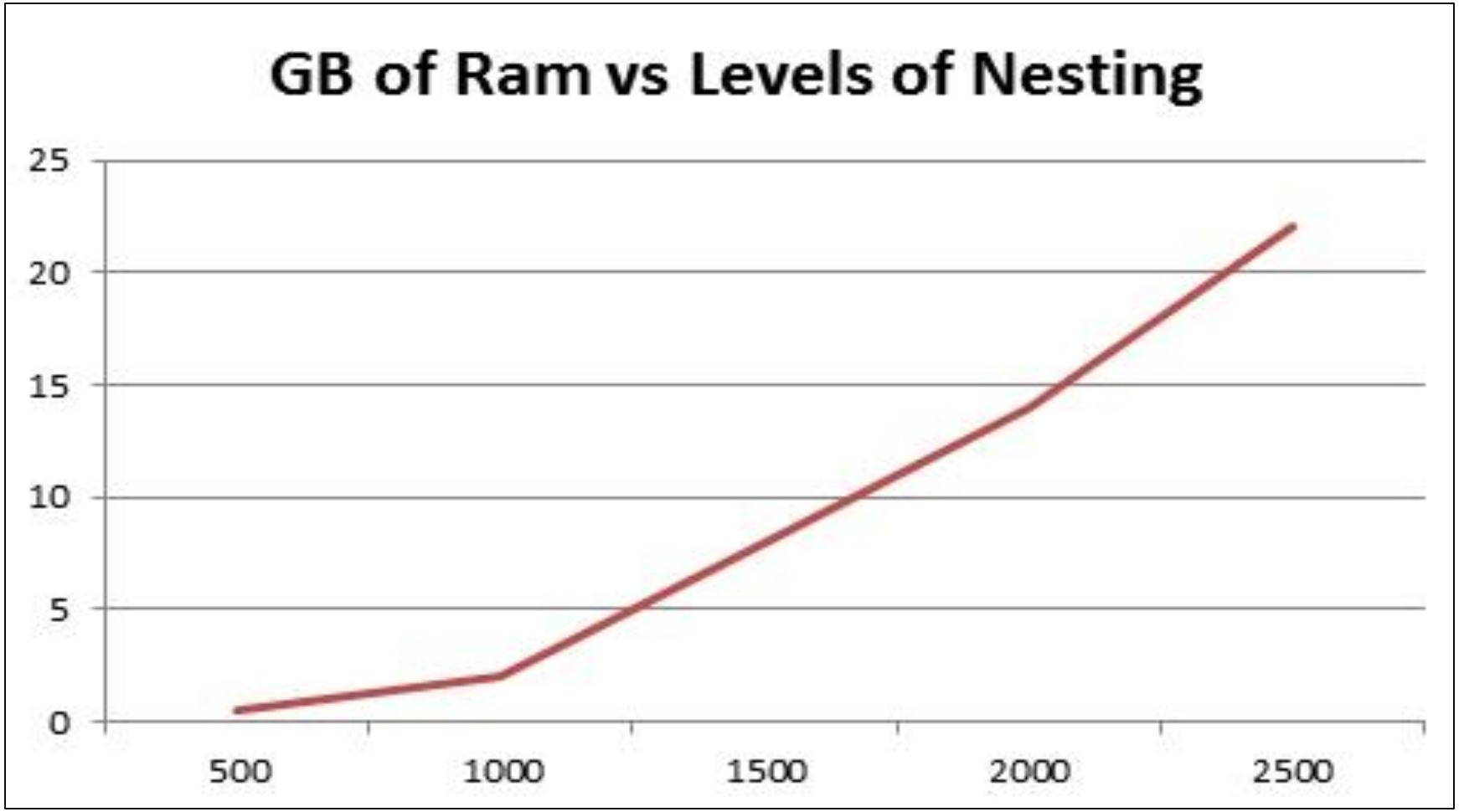


Editor | Messages

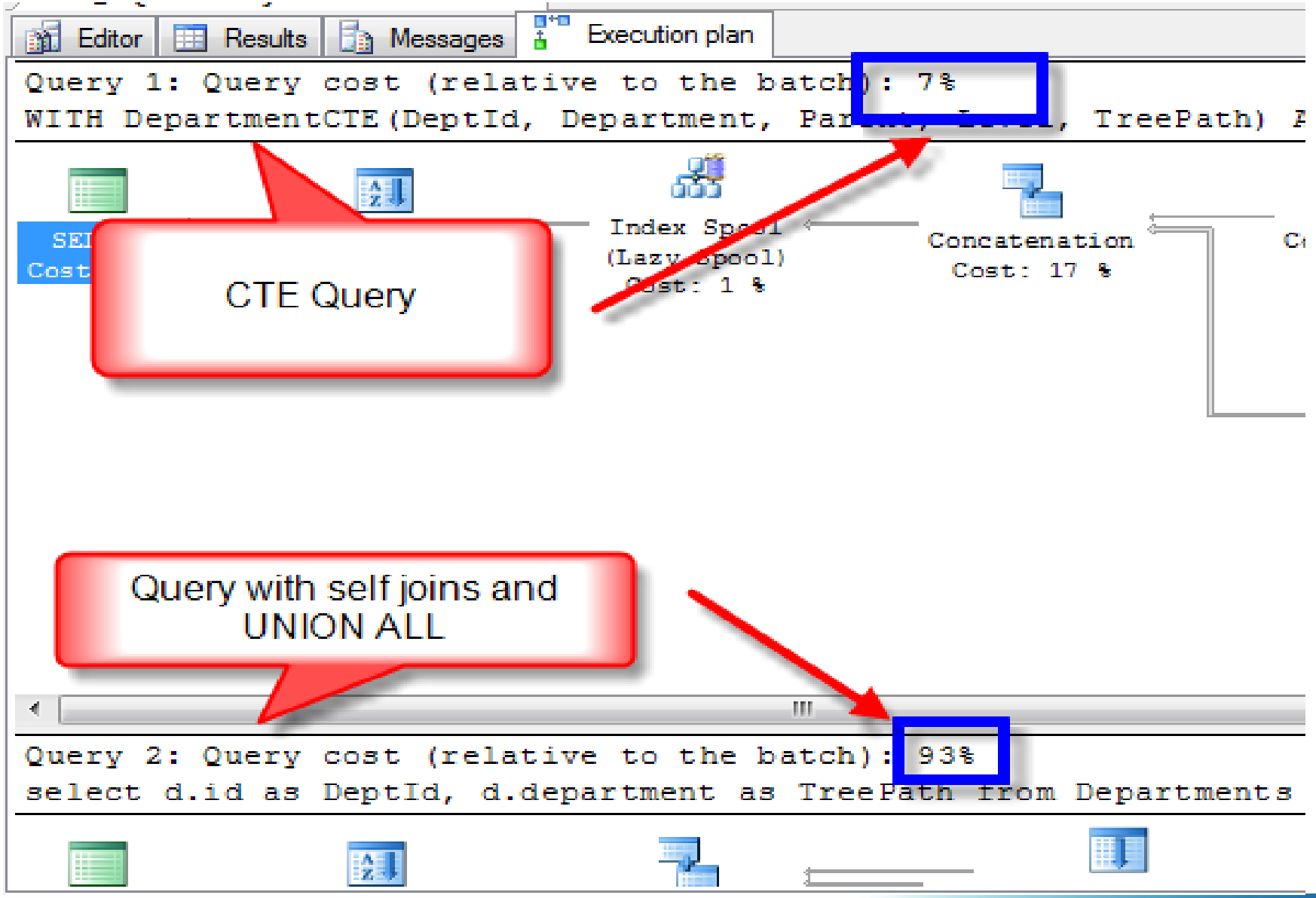
```
Msg 701, Level 17, State 123, Line 2
There is insufficient system memory in resource pool 'default' to run this query.
```

100 %

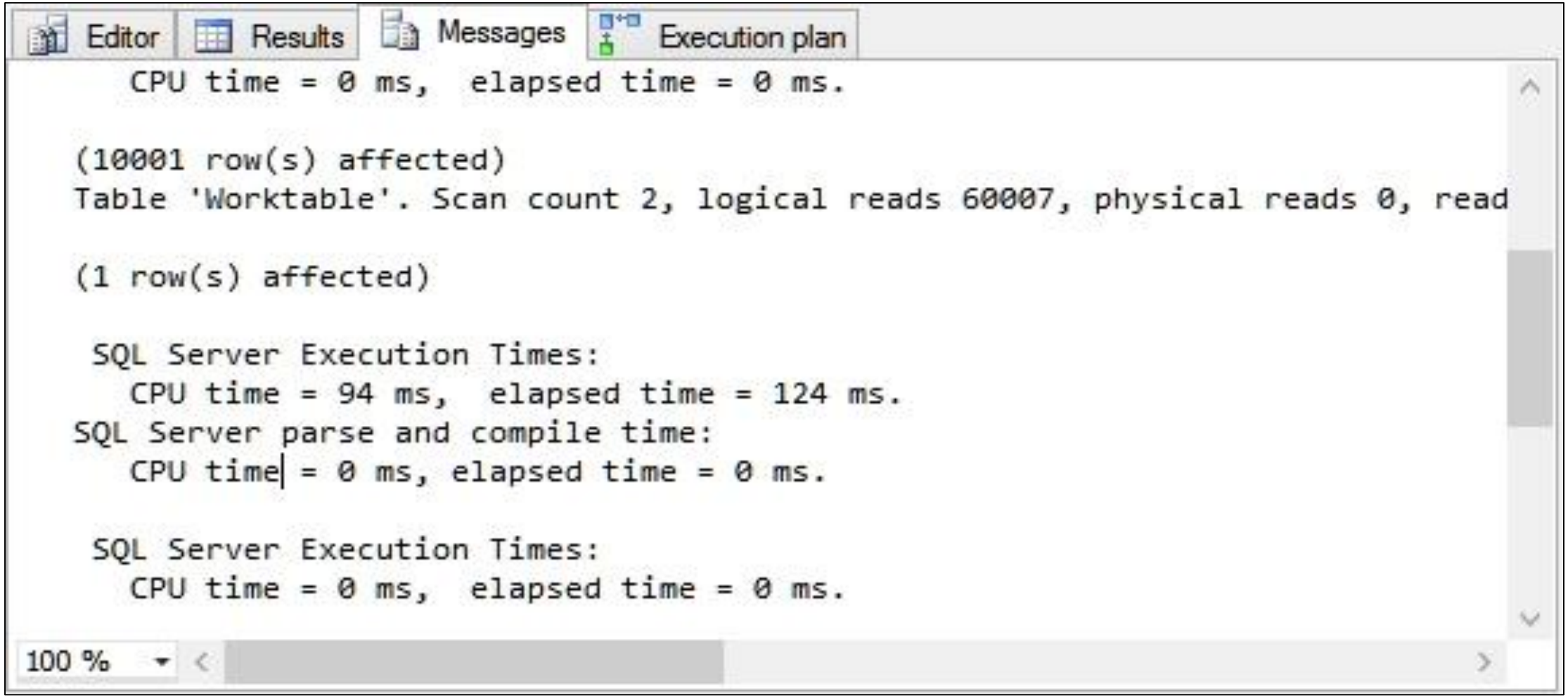
Recursive Performance



Recursive Performance



Deep Recursion



Classic Recursive Algorithms

Fibonacci

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

Factorial

The product of an integer and all the integers below it; e.g., factorial four (4!) is equal to 24.

42

Advanced Common Table Expressions

Summary

- Recursive Queries
- Hierarchical Recursive Data
- Manipulating Data
- Common Use Cases
- CTE Performance Considerations
- Classic Recursive Algorithms

More Details

- All samples shown are available on my website http://SteveStedman.com

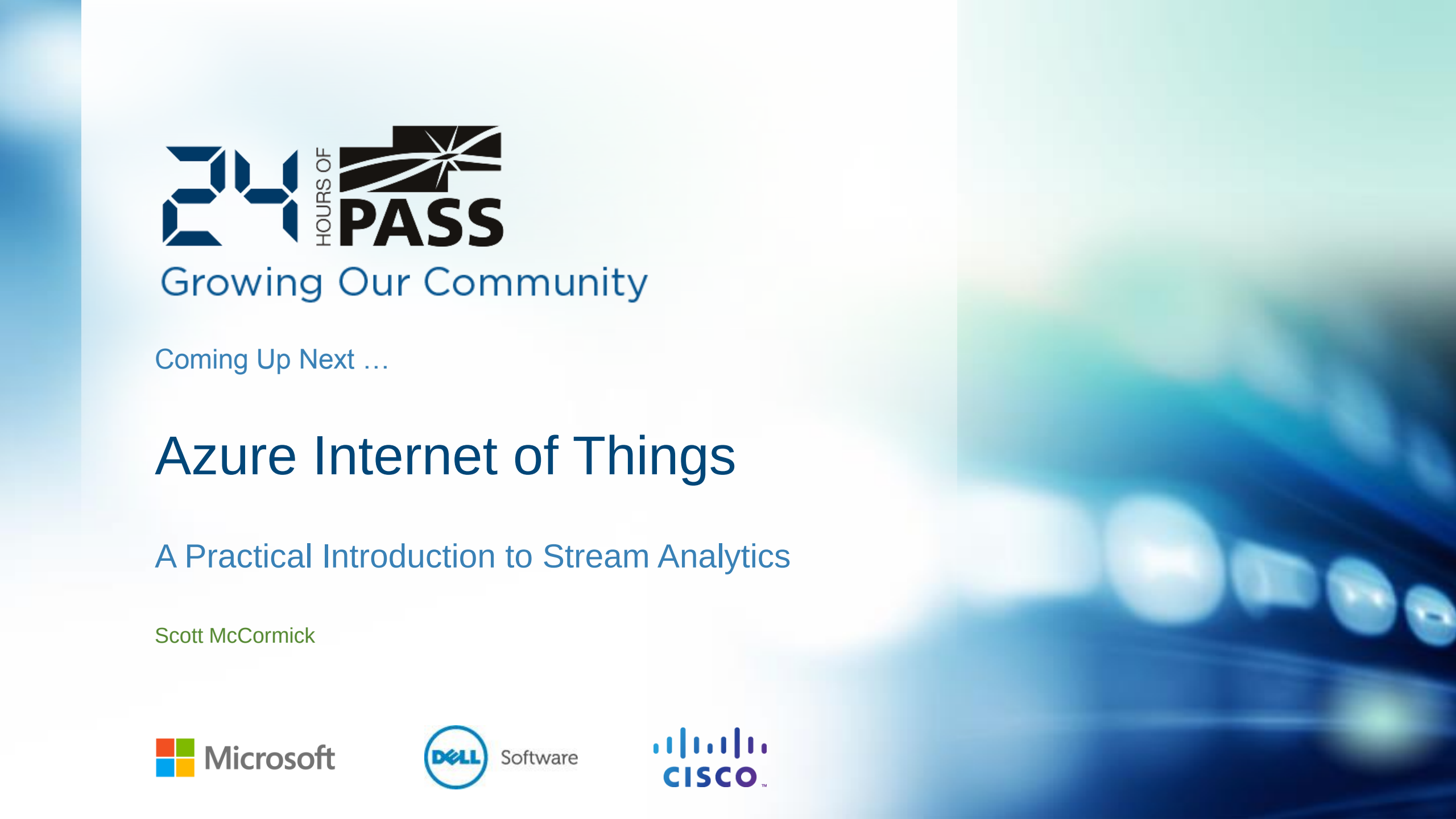
Questions?

Growing Our Community

Thank You for Attending

Follow @pass24hop

Share your thoughts with hashtags
#pass24hop & **#sqlpass**



Coming Up Next …

Azure Internet of Things

A Practical Introduction to Stream Analytics

Scott McCormick

Microsoft

DELL Software

CISCO