Agenda

What we're trying to do

A few ways we shouldn't do it, and why

The "right" way: sp_executesql

The drawbacks of the right way

Pro tips: troubleshooting and tuning

2.4 p2

Q&A site: you ask, other people do your job

Whole database is available under Creative Commons

Download it free: BrentOzar.com/go/querystack

We'll use the dbo.Users table

2.4 p3

```
46  /* Meet the Users table: */
47  SELECT TOP 100 * FROM dbo.Users;
48  GO
```

200 %

Results | Messages | Execution plan

	Id	AboutMe	DisplayName	Location	Reputation	Down
1	-1	Hi, I'm not really a person. I'm a backgrou...	Community	on the server farm	1	95671
2	1	<a href="http://www.codinghorror.com/blog/archi...	Jeff Atwood	El Cerrito, CA	43628	1309
3	2	Developer on the Stack Overflow team. Find me ...	Geoff Dalgas	Corvallis, OR	3431	88
4	3	<a href="http://blog.stackoverflow.com/2009/01/...	Jarrod Dixon	Raleigh, NC, United States	13323	100
5	4	I am: the co-founder and CEO of <a ...	Joel Spolsky	New York, NY	28403	96
6	5	Technical Evangelist at Microsoft, specializing in A...	Jon Galloway	San Diego, CA	38706	34
7	8	This is a puppet test account I use to validate "reg...	Eggs McLaren		942	9
8	9	Independent software engineer	Kevin Dente	Oakland, CA	14059	4
9	10	I'm not takin' my sneakers off! Actually, <b>...	Sneakers O'Toole	Lakehead, CA	101	0

2.4 p4

Our support team wants to search.

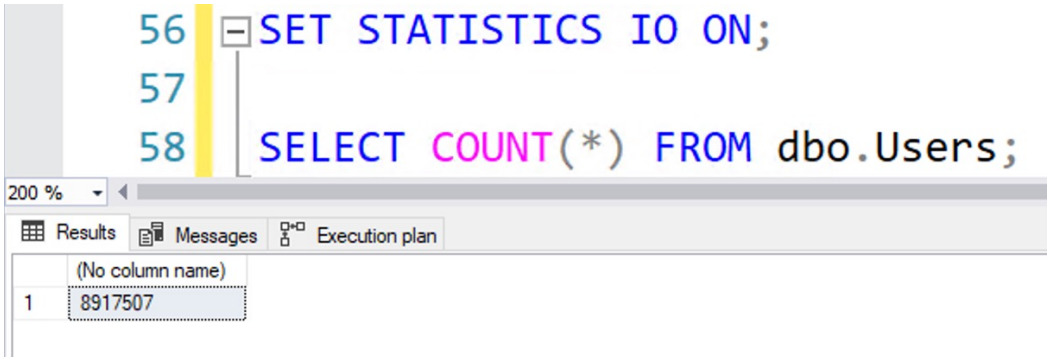
They want a single stored procedure that takes multiple parameters for searching:

```
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
GO
EXEC usp_SearchUsers @SearchReputation = 2;
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @SearchLocation = 'Las Vegas%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%', @SearchReputation = 107548;
GO
```

2.4 p5

They want it to be fast.

We have about 9 million users to deal with.



2.4 p6

They want it to be fast.

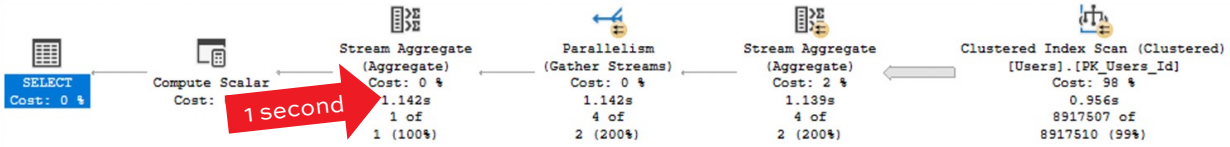
Worst case scenario: we scan the table.

```
56  SET STATISTICS IO ON;
57
58  SELECT COUNT(*) FROM dbo.Users;
```

200 %

Results | Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT COUNT(*) FROM dbo.Users



1 second

2.4 p7

Worst case scenario

```
56  SET STATISTICS IO ON;
57
58  SELECT COUNT(*) FROM dbo.Users;
```

200 % | Results | Messages | Execution plan

```
(1 row affected)
Table 'Users'. Scan count 5, logical reads 142155,
```

1 second

2.4 p8

We'll build supporting indexes.

```
CREATE INDEX DisplayName ON dbo.Users(DisplayName);
CREATE INDEX Location ON dbo.Users(Location);
CREATE INDEX Reputation ON dbo.Users(Reputation);
```

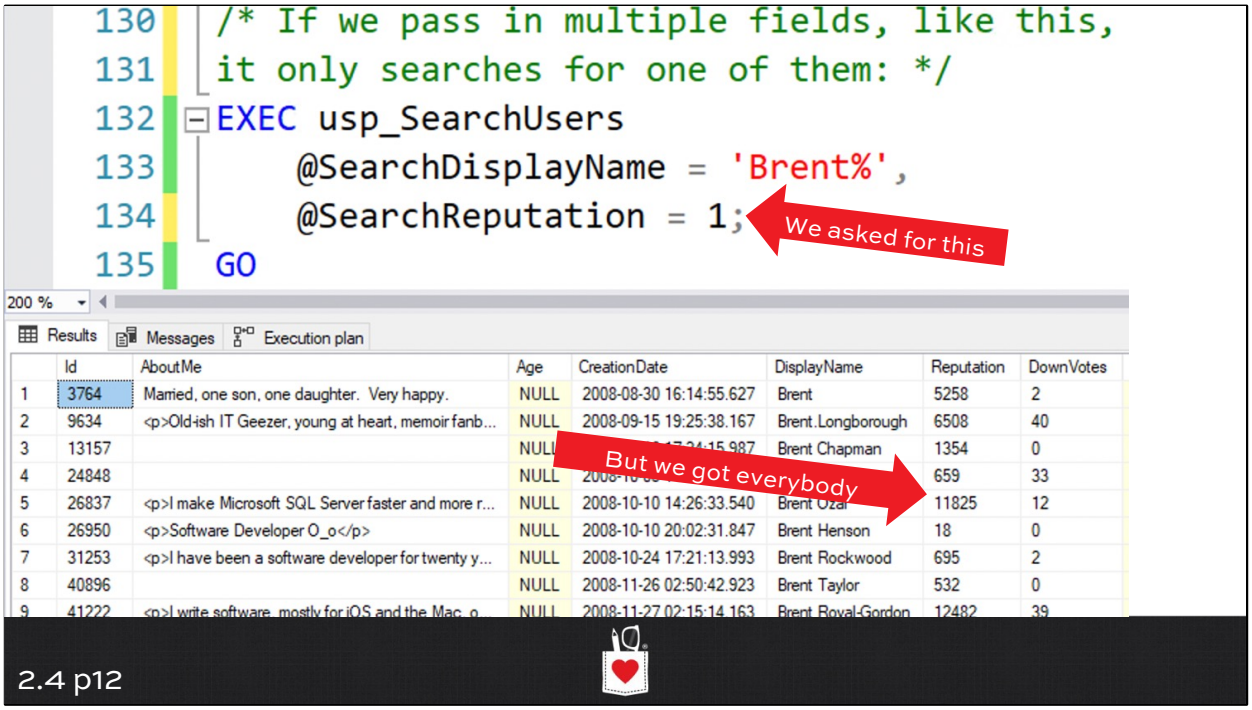
2.4 p9

Version 1:
IF Branches

2.4 p10

```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
IF @SearchDisplayName IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE DisplayName LIKE @SearchDisplayName
      ORDER BY CreationDate;
ELSE IF @SearchLocation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Location LIKE @SearchLocation
      ORDER BY CreationDate;
ELSE IF @SearchReputation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Reputation = @SearchReputation
      ORDER BY CreationDate;
END
```

Will this work?



	Id	AboutMe	Age	CreationDate	DisplayName	Reputation	DownVotes
1	3764	Married, one son, one daughter. Very happy.	NULL	2008-08-30 16:14:55.627	Brent	5258	2
2	9634	Old-ish IT Geezer, young at heart, memoir fanb...	NULL	2008-09-15 19:25:38.167	Brent.Longborough	6508	40
3	13157		NULL		Brent Chapman	1354	0
4	24848		NULL			659	33
5	26837	I make Microsoft SQL Server faster and more r...	NULL	2008-10-10 14:26:33.540	Brent Ozar	11825	12
6	26950	Software Developer O_o	NULL	2008-10-10 20:02:31.847	Brent Henson	18	0
7	31253	I have been a software developer for twenty y...	NULL	2008-10-24 17:21:13.993	Brent Rockwood	695	2
8	40896		NULL	2008-11-26 02:50:42.923	Brent Taylor	532	0
9	41222	I write software, mostly for iOS and the Mac, o...	NULL	2008-11-27 02:15:14.163	Brent Royal-Gordon	12482	39

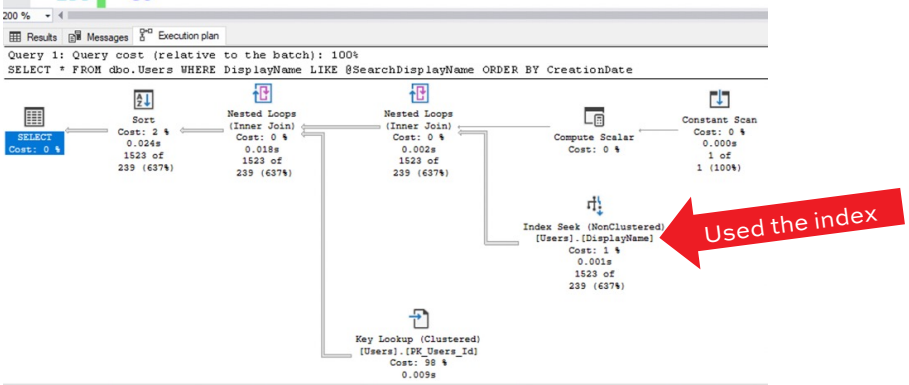
```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
IF @SearchDisplayName IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE DisplayName LIKE @SearchDisplayName
      ORDER BY CreationDate;
ELSE IF @SearchLocation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Location LIKE @SearchLocation
      ORDER BY CreationDate;
ELSE IF @SearchReputation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Reputation = @SearchReputation
      ORDER BY CreationDate;
END
```

Because it only went down one branch.

Searched by name only

But it DOES perform well...

```
132  EXEC usp_SearchUsers
133      @SearchDisplayName = 'Brent%',
134      @SearchReputation = 1;
135  GO
```



2.4 p14

And did few logical reads.

```
132  EXEC usp_SearchUsers
133       @SearchDisplayName = 'Brent%',
134       @SearchReputation = 1;
135  GO
```

200 %

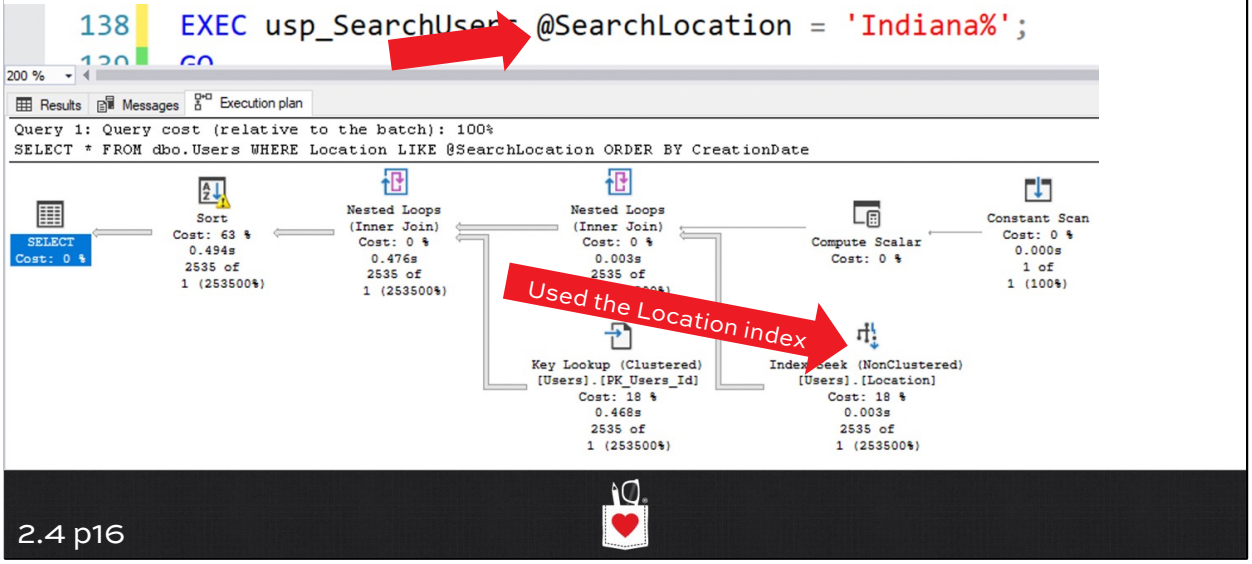
Results | Messages | Execution plan

```
(1523 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0,
Table 'Users'. Scan co          logical reads 4685,
```

Nice

2.4 p15

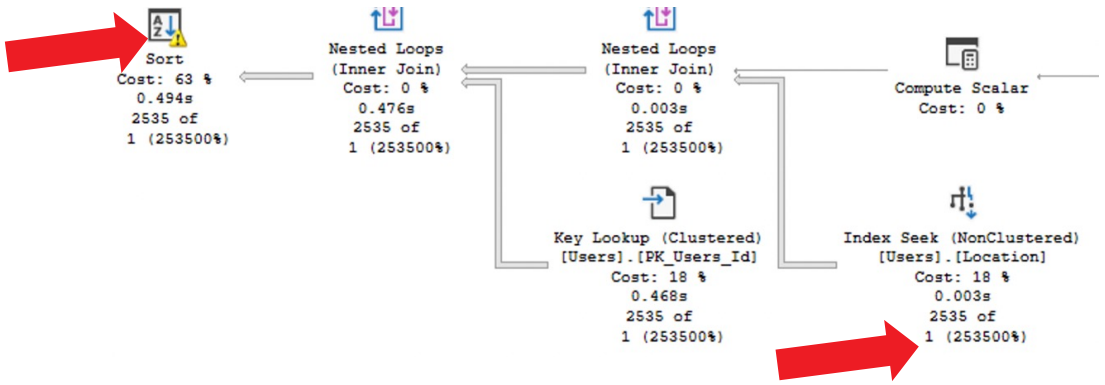
Try a different parameter...



2.4 p16

But there's a catch

There's a spill on the sort, caused by the low estimate.



```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
IF @SearchDisplayName IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE DisplayName LIKE @SearchDisplayName
      ORDER BY CreationDate;
ELSE IF @SearchLocation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Location LIKE @SearchLocation
      ORDER BY CreationDate;
ELSE IF @SearchReputation IS NOT NULL
    SELECT *
      FROM dbo.Users
      WHERE Reputation = @SearchReputation
      ORDER BY CreationDate;
END
```

SQL Server built the whole plan just once.

The first time we ran, this wasn't set

This design has big problems.

1. It's awful to write: you have to build so many branches to cover every scenario.
2. It produces incorrect results for param combos.
3. It produces inaccurate estimates.

2.4 p19

Version 2:
The Kitchen Sink

2.4 p20

```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
SELECT *
    FROM dbo.Users
    WHERE (DisplayName LIKE @SearchDisplayName OR @SearchDisplayName IS NULL)
      AND (Location LIKE @SearchLocation OR @SearchLocation IS NULL)
      AND (Reputation = @SearchReputation OR @SearchReputation IS NULL)
    ORDER BY CreationDate;
END
GO

EXEC usp_SearchUsers
    @SearchDisplayName = 'Brent%',
    @SearchReputation = 1;
```

2.4 p21

```
201  EXEC usp_SearchUsers
202       @SearchDisplayName = 'Brent%',
203       @SearchReputation = 1;
204  GO
```

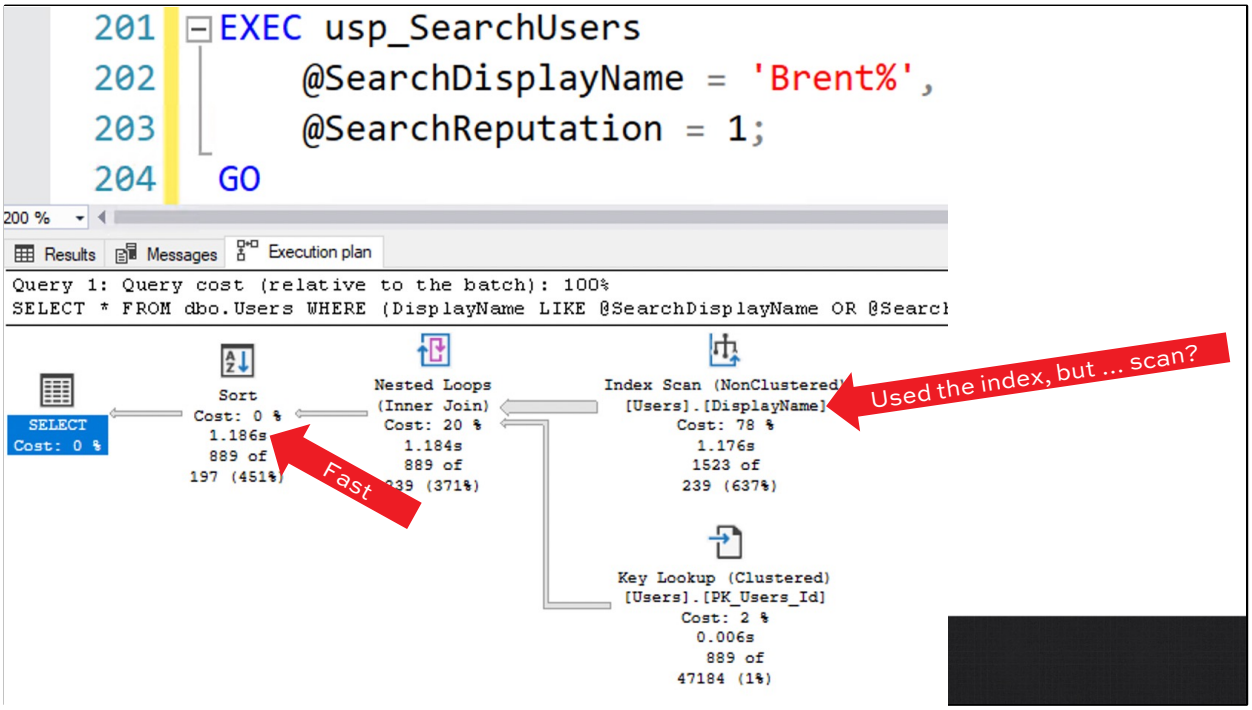
200 %

Yay, accurate results

Results | Messages | Execution plan

	Id	AboutMe	Age	CreationDate	DisplayName	Reputation	Location
1	211111		NULL	2009-11-14 18:38:51.167	Brent B	1	
2	212610	NULL	NULL	2009-11-17 05:26:08.937	Brent Scriver	1	NULL
3	222260	NULL	NULL	2009-12-01 16:47:46.843	Brent	1	NULL
4	245692	NULL	NULL	2010-01-07 16:21:07.397	Brent1970	1	NULL
5	262134	NULL	NULL	2010-01-29 20:55:21.453	Brent Ramsey	1	NULL
6	279829	NULL	NULL	2010-02-23 20:44:33.433	brent	1	NULL
7	356923	NULL	NULL	2010-06-02 22:08:31.497	Brent	1	Chicago,
8	359941	NULL	NULL	2010-06-06 23:08:15.983	Brent McConnell	1	NULL

2.4 p22



Better than a table scan, I guess

Worst case scenario = ~140000 logical reads

```
201  EXEC usp_SearchUsers
202       @SearchDisplayName = 'Brent%',
203       @SearchReputation = 1;
204  GO
```

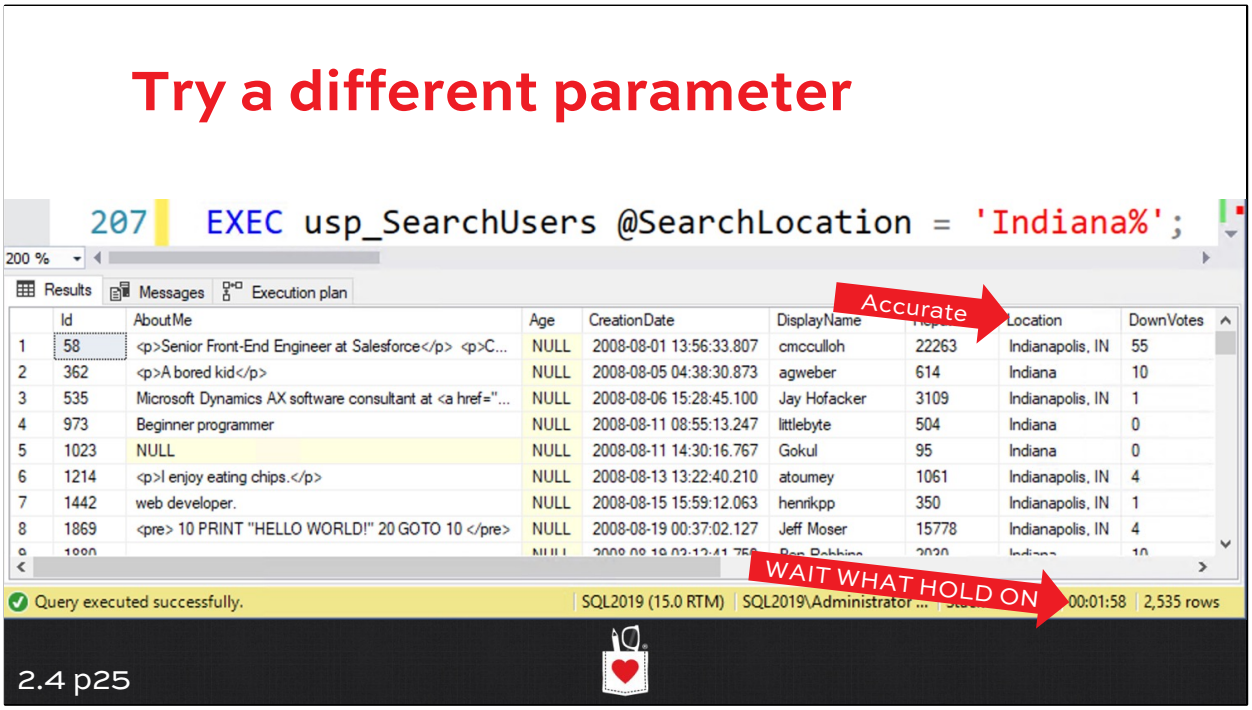
200 %

Results | Messages | Execution plan

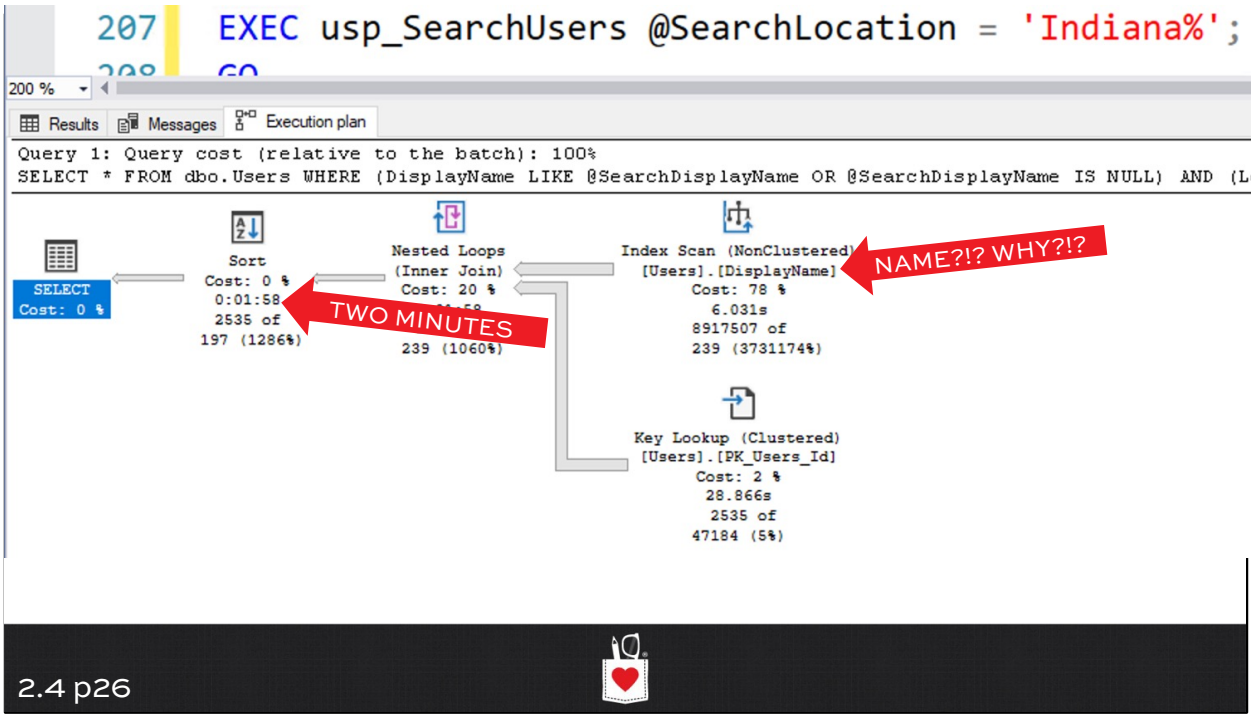
```
(889 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0
Table 'Users'. Scan count 1, logical reads 42635
```

2.4 p24

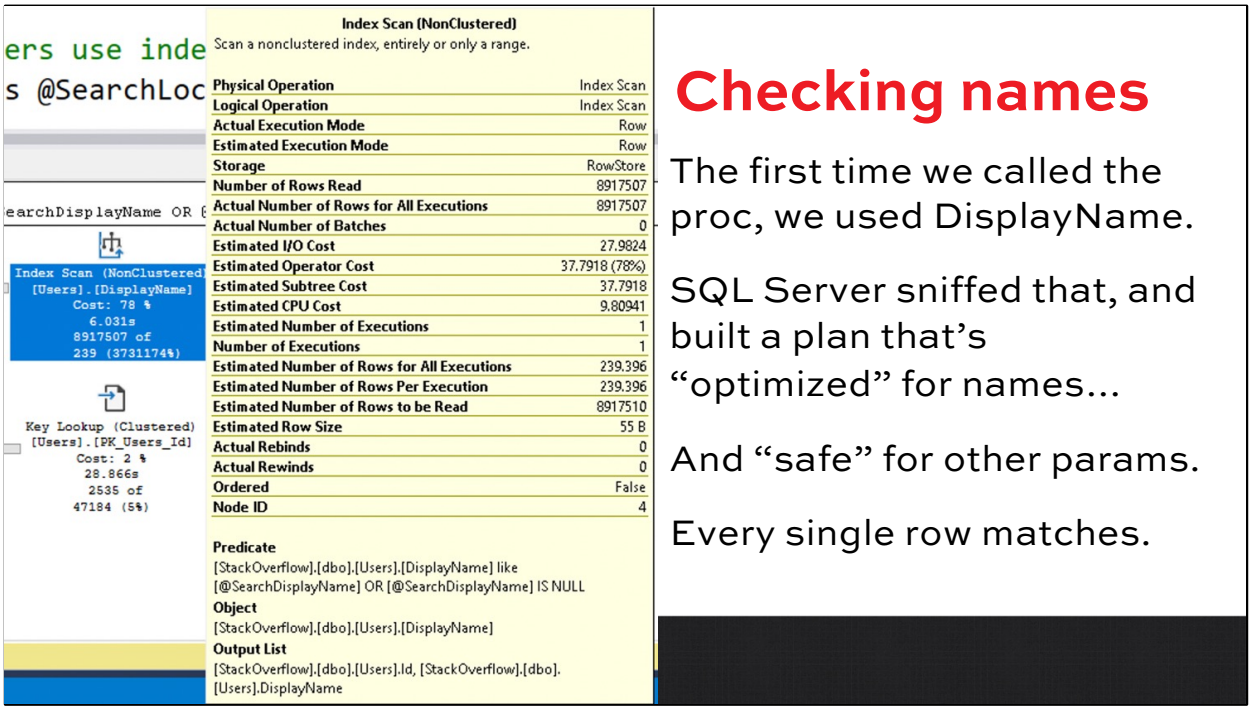
Try a different parameter



2.4 p25



2.4 p26



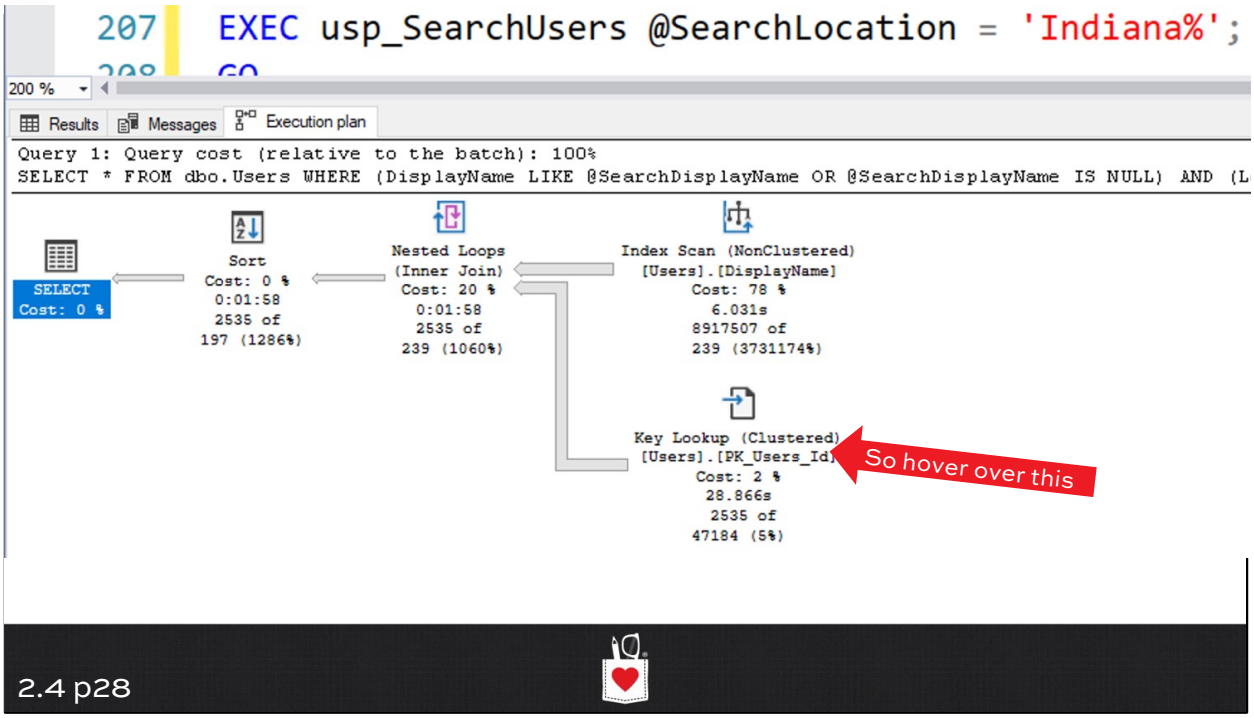
Checking names

The first time we called the proc, we used DisplayName.

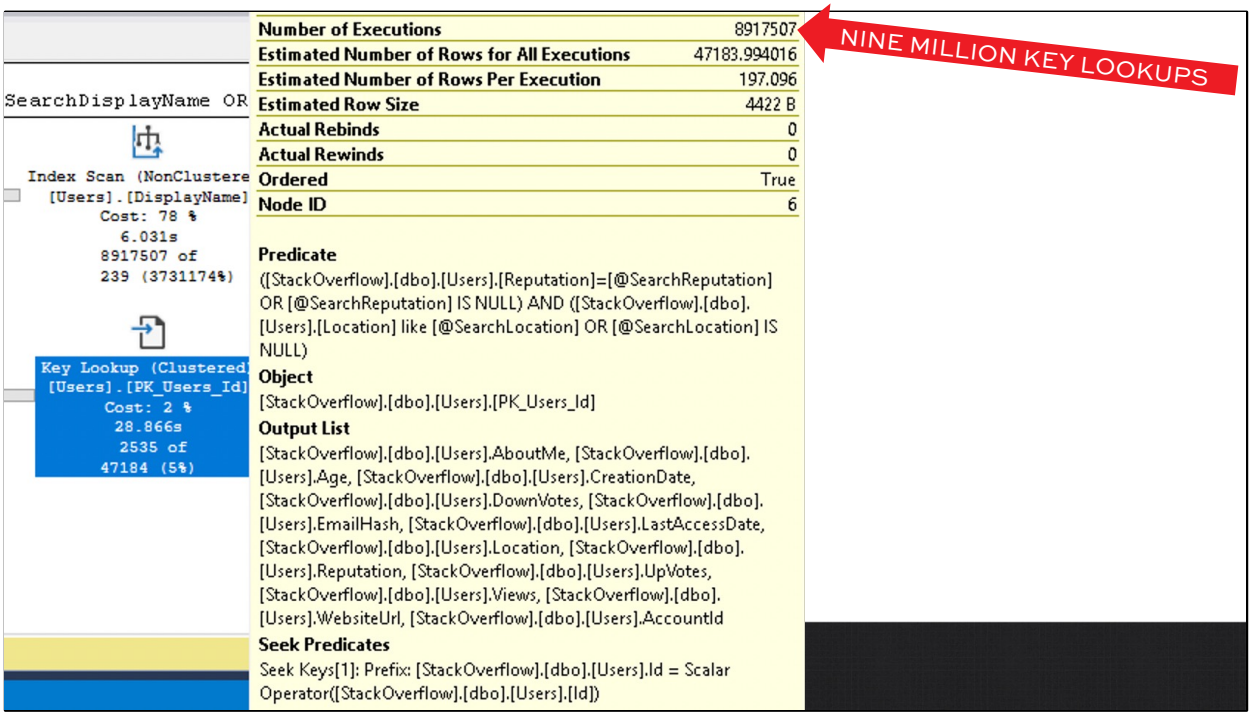
SQL Server sniffed that, and built a plan that's "optimized" for names...

And "safe" for other params.

Every single row matches.



2.4 p28



Worse than our worst case

Worst case scenario = ~140000 logical reads

```
207   EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
208   GO
```

200 % | Results | Messages | Execution plan

```
(2535 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical rea
Table 'Users'. Scan count 1, logical rea  36993604, physical
```

WOW

2.4 p30

Kitchen sink = terribad.

You can get this to perform if:

- The table doesn't have any indexes at all
- You use OPTION RECOMPILE

ISNULL, COALESCE solutions
suffer from the same problems.

We're gonna need something better.

2.4 p31

Version 3: Dynamic SQL

2.4 p32

```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    SET @StringToExecute = N'SELECT * FROM dbo.Users WHERE 1 = 1 ';

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @SearchDisplayName ';

    IF @SearchLocation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Location LIKE @SearchLocation ';

    IF @SearchReputation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Reputation = @SearchReputation ';

    SET @StringToExecute = @StringToExecute + N' ORDER BY CreationDate; ';

    EXEC sp_executesql @StringToExecute,
        N'@SearchDisplayName NVARCHAR(100), @SearchLocation NVARCHAR(100), @SearchReputation INT',
        @SearchDisplayName, @SearchLocation, @SearchReputation;
END
```

```
342  EXEC usp_SearchUsers
343       @SearchDisplayName = 'Brent%',
344       @SearchReputation = 1;
345  GO
```

200 %

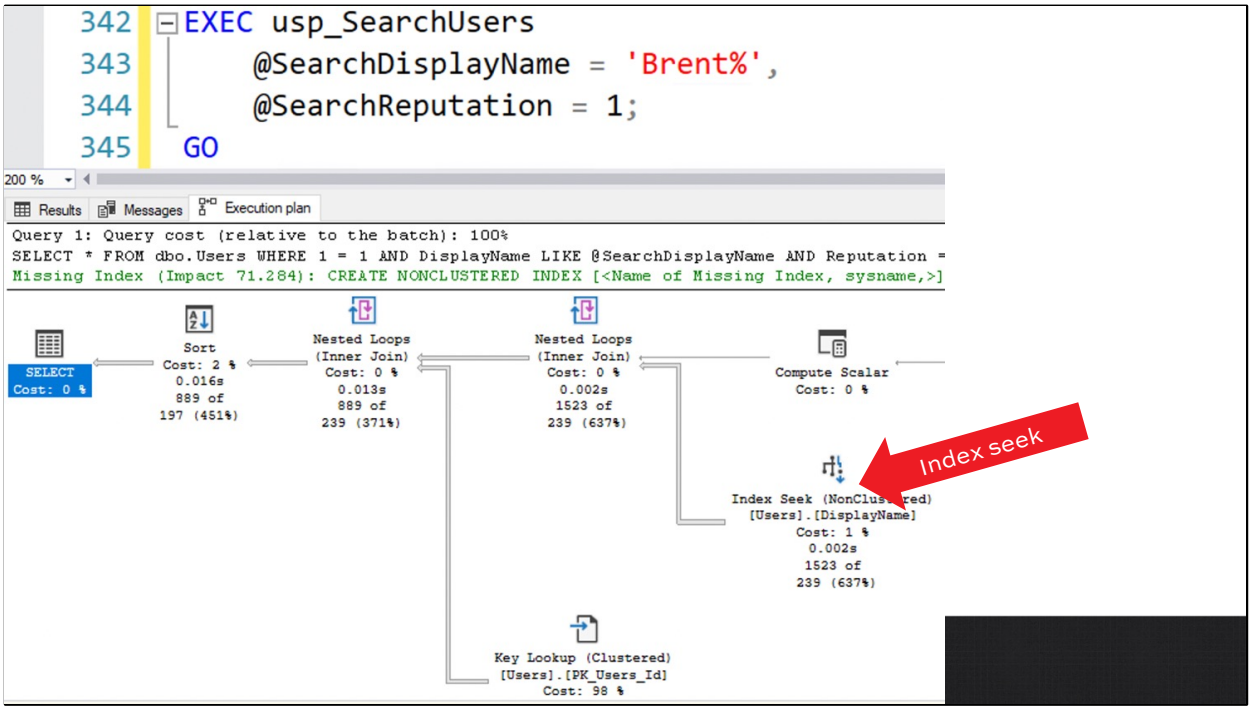
Results | Messages | Execution plan

Accurate

Accurate

	Id	AboutMe	Age	CreationDate	DisplayName	Location	Reputation
1	211111		NULL	2009-11-14 18:38:51.167	Brent B		1
2	212610	NULL	NULL	2009-11-17 05:26:08.937	Brent Scriver	NULL	1
3	222260	NULL	NULL	2009-12-01 16:47:46.843	Brent	NULL	1
4	245692	NULL	NULL	2010-01-07 16:21:07.397	Brent1970	NULL	1
5	262134	NULL	NULL	2010-01-29 20:55:21.453	Brent Ramsey	NULL	1
6	279829	NULL	NULL	2010-02-23 20:44:33.433	brent	NULL	1
7	356923	NULL	NULL	2010-06-02 22:08:31.497	Brent	Chicago, IL	1
8	359941	NULL	NULL	2010-06-06 23:08:15.983	Brent McConnell	NULL	1
9	360183	NULL	NULL	2010-06-07 08:22:42.310	Brent	NULL	1
10	410120	NULL	NULL	2010-08-03 20:41:47.303	Brent	NULL	1
11	432231	NULL	NULL	2010-08-26 18:52:12.507	Brent	NULL	1

2.4 p34



Great logical reads

Worst case was a table scan at ~140K logical reads

```
342  EXEC usp_SearchUsers
343       @SearchDisplayName = 'Brent%',
344       @SearchReputation = 1;
345  GO
```

200 % Results Messages Execution plan

```
(889 rows affected)
Table 'Worktable'. Scan count 0, logical reads (
Table 'Users'. Scan count 1, logical reads 4685,
```

2.4 p36

Try another parameter

```
348  EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
349  GO
```

200 %

Results | Messages | Execution plan

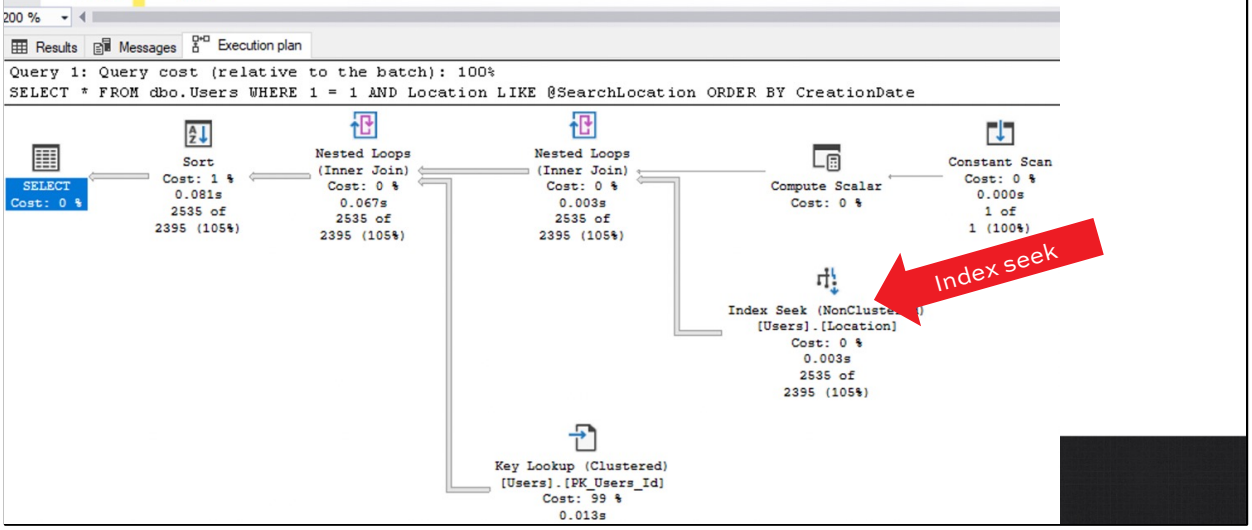
Accurate

	Id	AboutMe	Age	CreationDate	DisplayName	Location	Reputation	DownVote
1	58	Senior Front-End Engineer at Salesforce C...	NULL	2008-08-01 13:56:33.807	cmcculloh	Indianapolis, IN	22263	55
2	362	A bored kid	NULL	2008-08-05 04:38:30.873	agweber	Indiana	614	10
3	535	Microsoft Dynamics AX software consultant at <a href="...	NULL	2008-08-06 15:28:45.100	Jay Hofacker	Indianapolis, IN	3109	1
4	973	Beginner programmer	NULL	2008-08-11 08:55:13.247	littlebyte	Indiana	504	0
5	1023	NULL	NULL	2008-08-11 14:30:16.767	Gokul	Indiana	95	0
6	1214	I enjoy eating chips.	NULL	2008-08-13 13:22:40.210	atoumey	Indianapolis, IN	1061	4
7	1442	web developer.	NULL	2008-08-15 15:59:12.063	henrikpp	Indianapolis, IN	350	1
8	1869	<pre> 10 PRINT "HELLO WORLD!" 20 GOTO 10 </pre>	NULL	2008-08-19 00:37:02.127	Jeff Moser	Indianapolis, IN	15778	4
9	1880		NULL	2008-08-19 03:12:41.750	Ben Robbins	Indiana	2030	10
10	1942		NULL	2008-08-19 14:45:43.723	kemiller2002	Indianapolis, IN	90588	4

2.4 p37

Gets an index seek on Location

```
348  EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
349  GO
```



Low logical reads, too.

```
348  EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
349  GO
```

200 % ▾

Results | Messages | Execution plan

```
(2535 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical read
Table 'Users'. Scan count 1, logical reads 11529, physical read
```

2.4 p39

Dynamic SQL is great!

Each set of parameters gets its own cached plan.

- @DisplayName gets a cached plan
- @Location gets a cached plan
- @Reputation gets a cached plan
- @DisplayName, @Location gets a cached plan
- @Location, @Reputation gets a cached plan

And so on.

Proving it

```
DBCC FREEPROCCACHE
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
GO
EXEC usp_SearchUsers @SearchReputation = 2;
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @SearchLocation = 'Las Vegas%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%', @SearchReputation = 89836;
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @SearchReputation = 12145;
GO
sp_BlitzCache
```

2.4 p41

We bloat the plan cache a LITTLE.

This proc now has several plans in the cache.

But it's NOT a plan for EACH parameter VALUE.

```
367  sp_BlitzCache
368  GO
```

200 %

Results | Messages

	Database	Cost	Query Text	Query Type	# Executions
1	StackOverflow	0	CREATE PROC dbo.usp_SearchUsers @SearchDisplayName NVARCHAR(100) = NULL, @Searc...	Procedure or Function: [dbo].[usp_SearchUsers]	6
2	StackOverflow	74.5458	SELECT * FROM dbo.Users WHERE 1 = 1 AND Reputation = @SearchReputation ORDER BY Creati...	Statement	1
3	StackOverflow	7.87265	SELECT * FROM dbo.Users WHERE 1 = 1 AND Location LIKE @SearchLocation ORDER BY Creatio...	Statement	1
4	StackOverflow	0.802025	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName ORDER B...	Statement	1
5	StackOverflow	0.0553486	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName AND Loca...	Statement	1
6	StackOverflow	0.0184266	SELECT * FROM dbo.Users WHERE 1 = 1 AND Location LIKE @SearchLocation AND Reputation = ...	Statement	1
7	StackOverflow	0.0184266	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName AND Repu...	Statement	1

2.4 p42

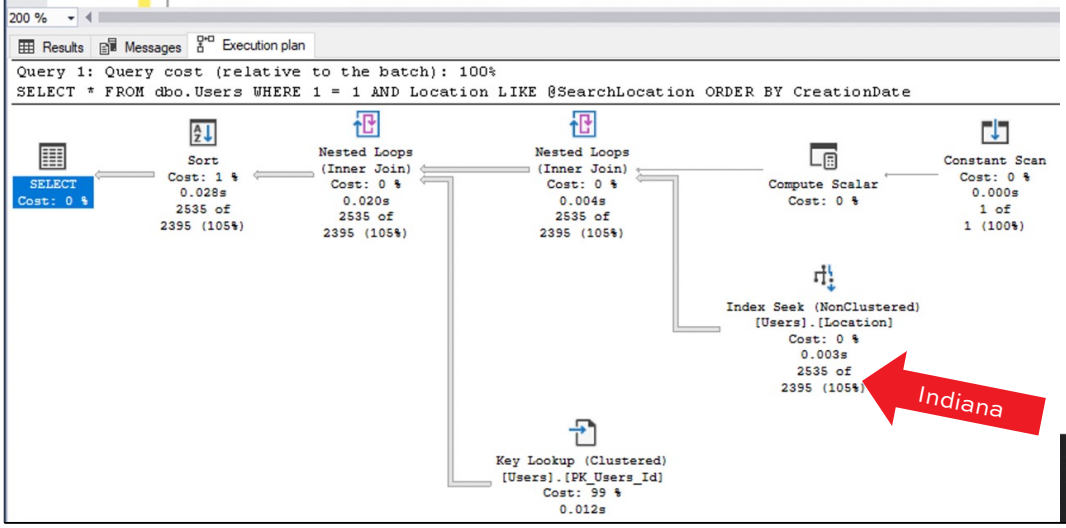
```
DBCC FREEPROCCACHE
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
GO
EXEC usp_SearchUsers @SearchReputation = 2;
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @SearchLocation = 'Las Vegas%';
GO
EXEC usp_SearchUsers @SearchLocation = 'Indiana%', @SearchReputation = 89836;
GO
EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @SearchReputation = 12145;
GO
sp_BlitzCache
```

The first params we used

2.4 p43

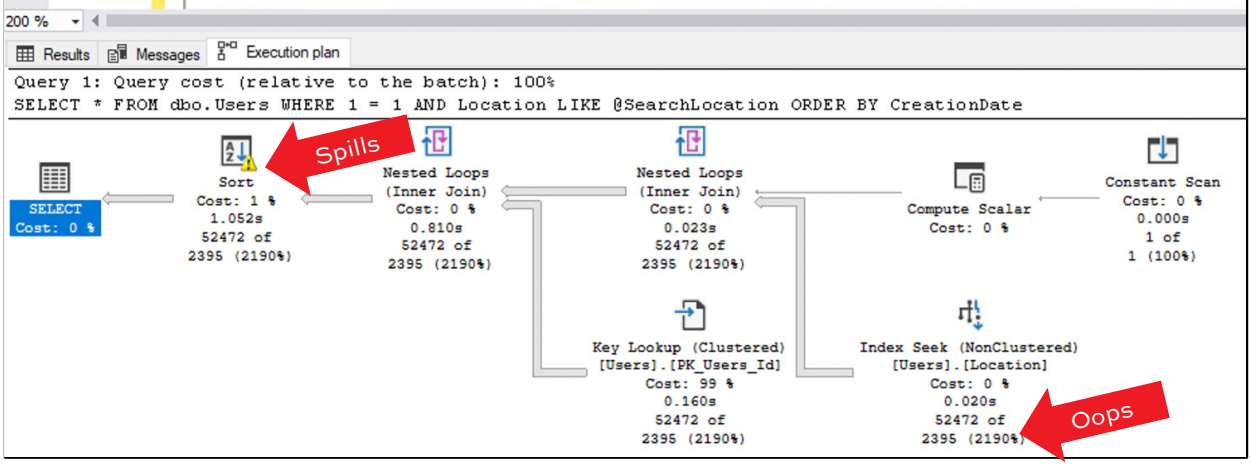
When I ran it for Indiana...

```
386 EXEC usp_SearchUsers @SearchLocation = 'Indiana%';
387
```



But if I run it for INDIA...

```
388  EXEC usp_SearchUsers @SearchLocation = 'India%';
389
```



Dynamic SQL doesn't ELIMINATE parameter sniffing.

It just reduces the blast radius.

Each parameter has its own query plan.

Sniffing effects are limited to just that one plan.

	Database	Cost	Query Text	Query Type	# Executions
1	StackOverflow	0	CREATE PROC dbo.usp_SearchUsers @SearchDisplayName NVARCHAR(100) = NULL, @Searc...	Procedure or Function: [dbo].[usp_SearchUsers]	6
2	StackOverflow	74.5458	SELECT * FROM dbo.Users WHERE 1 = 1 AND Reputation = @SearchReputation ORDER BY Creati...	Statement	1
3	StackOverflow	7.87265	SELECT * FROM dbo.Users WHERE 1 = 1 AND Location LIKE @SearchLocation ORDER BY Creatio...	Statement	1
4	StackOverflow	0.802025	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName ORDER B...	Statement	1
5	StackOverflow	0.0553486	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName AND Loca...	Statement	1
6	StackOverflow	0.0184266	SELECT * FROM dbo.Users WHERE 1 = 1 AND Location LIKE @SearchLocation AND Reputation = ...	Statement	1
7	StackOverflow	0.0184266	SELECT * FROM dbo.Users WHERE 1 = 1 AND DisplayName LIKE @SearchDisplayName AND Repu...	Statement	1

2.4 p46

Questions so far?

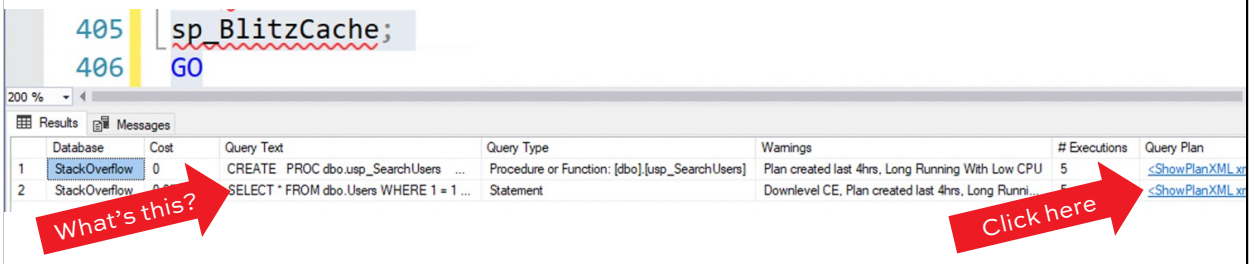
2.4 p47

Now let's get to the tips & tricks.

2.4 p48

When you're tuning the plan cache

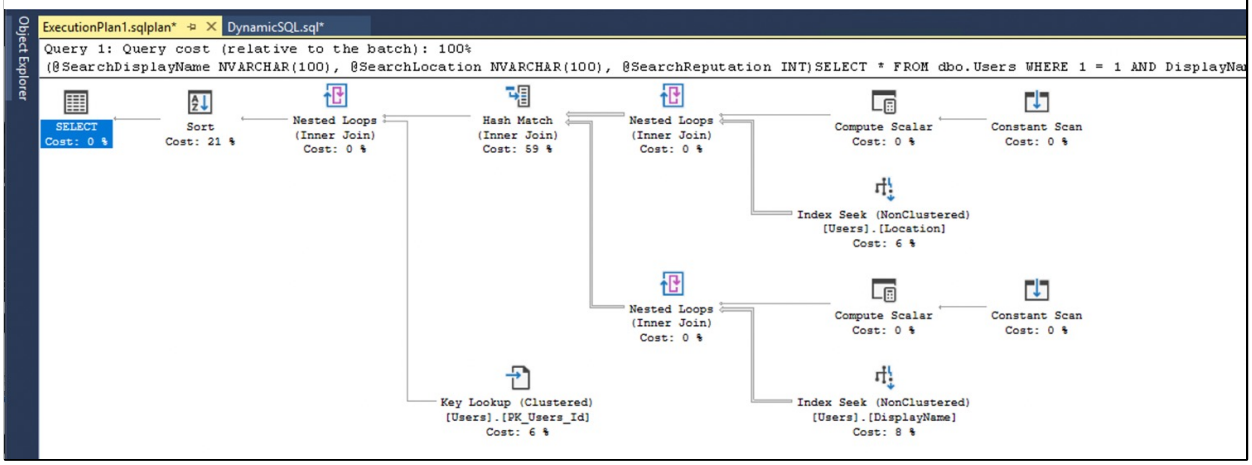
You're going to see queries created by dynamic SQL and you're going to want to click on the plan to see where they're from and how to fix 'em:



It’s just a query.

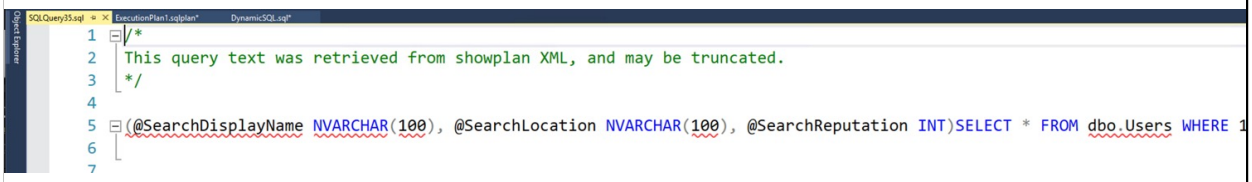
There’s no hint of what built it.

You can right-click on the query, click Edit Query Text



No clues here either.

You're going to wanna know what built this.

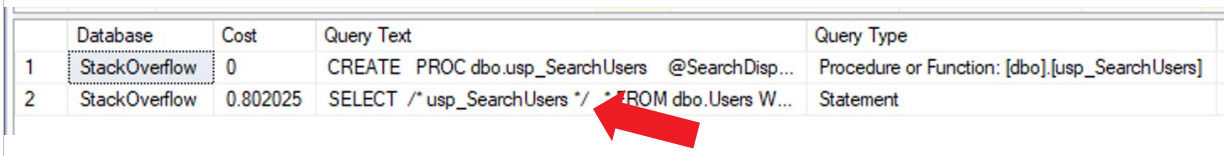


Well, YOU control query text, so when you build it...

```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    SET @StringToExecute = N'SELECT  /* usp_SearchUsers */  * FROM dbo.Users WHERE 1 = 1 ';

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @SearchDisplayName ';

    IF @SearchLocation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Location LIKE @SearchLocation ';

    IF @SearchReputation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Reputation = @SearchReputation ';

    SET @StringToExecute = @StringToExecute + N' ORDER BY CreationDate; ';

    EXEC sp_executesql @StringToExecute,
        N'@SearchDisplayName NVARCHAR(100), @SearchLocation NVARCHAR(100), @SearchReputation INT',
        @SearchDisplayName, @SearchLocation, @SearchReputation;
END
```

So when you view the plan cache

The generator name is obvious:



	Database	Cost	Query Text	Query Type
1	StackOverflow	0	CREATE PROC dbo.usp_SearchUsers @SearchDisp...	Procedure or Function: [dbo].[usp_SearchUsers]
2	StackOverflow	0.802025	SELECT /* usp_SearchUsers */ * FROM dbo.Users W...	Statement

Use this same trick wherever you generate T-SQL: including C#, Java, Perl, etc.

It has to be AFTER the SELECT.

Like this:

```
SET @StringToExecute = N'SELECT  /* usp_SearchUsers */   * FROM
```

NOT like this:

```
SET @StringToExecute = N'/* usp_SearchUsers */ SELECT *
```

Because SQL Server clips anything before the query.

2.4 p54

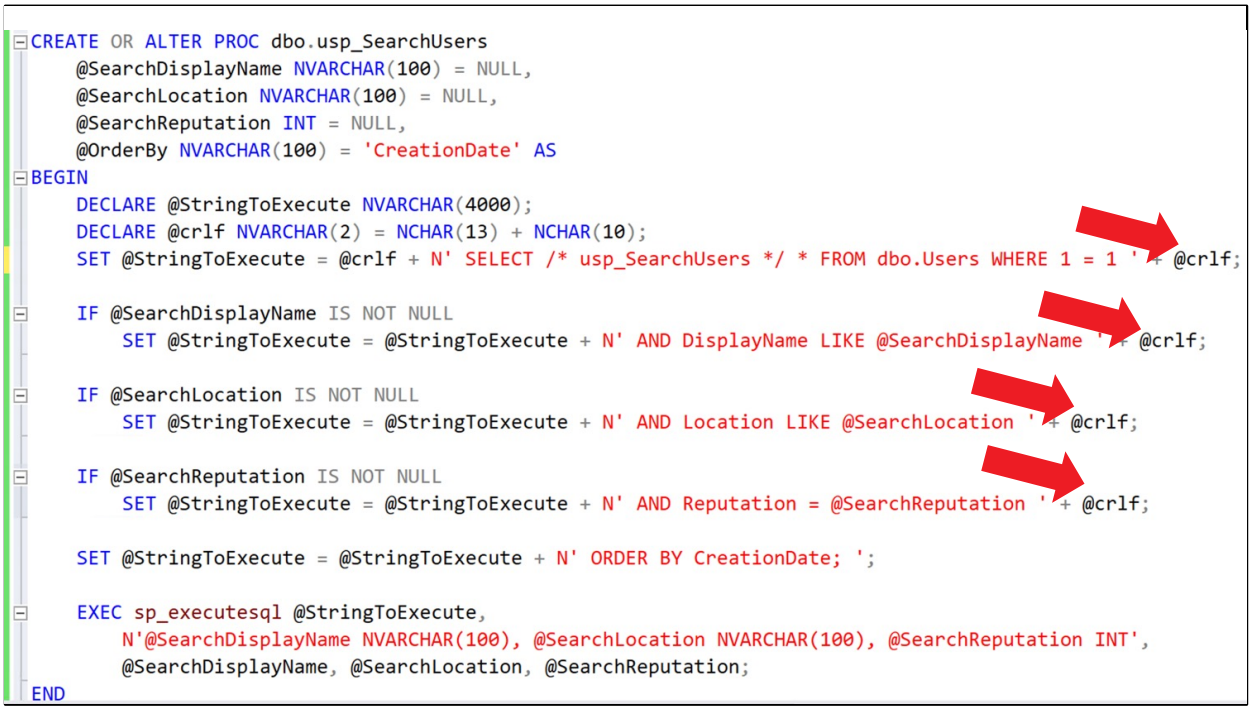
Why not make it easier to read?

As long as we're making debugging easier, create a carriage-return-line-feed variable.

Throw it into the SQL you're building whenever you want a line break.

```
DECLARE @StringToExecute NVARCHAR(4000);
DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
SET @StringToExecute = @crlf + N' SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1 ' + @crlf;
```

2.4 p55



```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL,
    @OrderBy NVARCHAR(100) = 'CreationDate' AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @StringToExecute = @crlf + N' SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1 ' + @crlf;

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @SearchDisplayName ' + @crlf;

    IF @SearchLocation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Location LIKE @SearchLocation ' + @crlf;

    IF @SearchReputation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Reputation = @SearchReputation ' + @crlf;

    SET @StringToExecute = @StringToExecute + N' ORDER BY CreationDate; ';

    EXEC sp_executesql @StringToExecute,
        N'@SearchDisplayName NVARCHAR(100), @SearchLocation NVARCHAR(100), @SearchReputation INT',
        @SearchDisplayName, @SearchLocation, @SearchReputation;
END
```

So when you're viewing a plan

And you right-click the query, click Edit Query Text...

```
1  /*
2  This query text was retrieved from showplan XML, and may be truncated.
3  */
4
5  (@SearchDisplayName NVARCHAR(100), @SearchLocation NVARCHAR(100), @SearchReputation INT)
6   SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1
7    AND DisplayName LIKE @SearchDisplayName
8    ORDER BY CreationDate
9
10
```

2.4 p57

Do NOT use dynamic comments.

You're going to be tempted to put in user names, client names, server names, variables, etc.

Here's an example with date/time:

```
DECLARE @StringToExecute NVARCHAR(4000);
DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
SET @StringToExecute = @crlf + N'SELECT /* usp_SearchUsers at date/time: '
    + CONVERT(NVARCHAR(100), GETDATE(), 21) + '*/' + @crlf
    + N' * FROM dbo.Users WHERE 1 = 1 ' + @crlf;
```

2.4 p58

The problem: plan cache bloat.

Every different string gets its own cached plan.

```
544  sp_BlitzCache;
545  GO
```

200 %

Results | Messages

	Database	Cost	Query Text
1	StackOverflow	0	CREATE PROC dbo.usp_Sea… @SearchDisplayName NVARCHAR(100) = NULL,
2	StackOverflow	0.802025	SELECT /* usp_SearchUsers at date/time: 2022-09-15 01:17:50.920*/ * FROM dbo.Users W
3	StackOverflow	0.802025	SELECT /* usp_SearchUsers at date/time: 2022-09-15 01:17:52.200*/ * FROM dbo.Users W
4	StackOverflow	0.802025	SELECT /* usp_SearchUsers at date/time: 2022-09-15 01:17:53.407*/ * FROM dbo.Users W

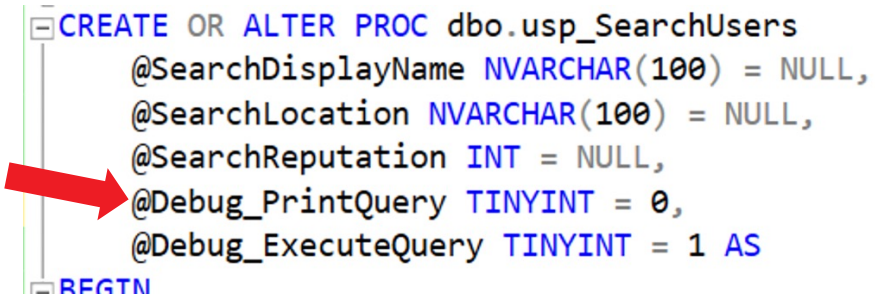
2.4 p59

Next tip:
easier debugging.

2.4 p60

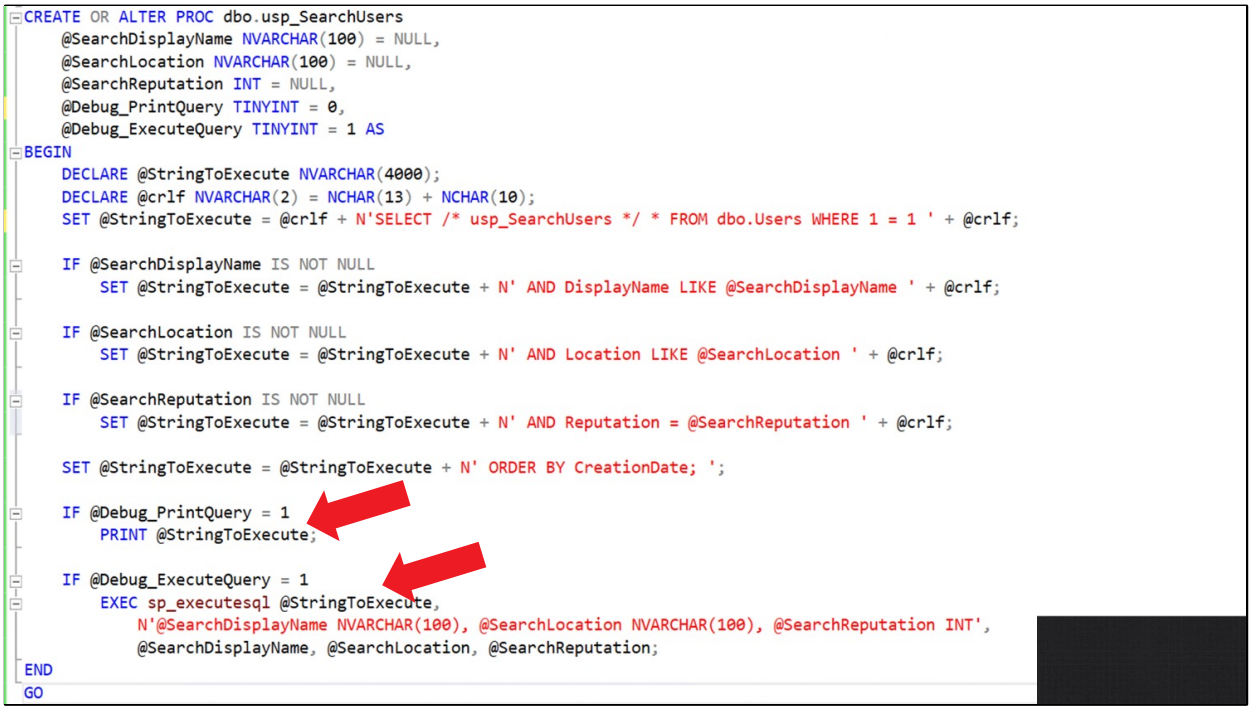
Every dynamic SQL proc needs...

A @Debug_PrintQuery & @Debug_ExecuteQuery parameter:



```
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL,
    @Debug_PrintQuery TINYINT = 0,
    @Debug_ExecuteQuery TINYINT = 1 AS
BEGIN
```

2.4 p61



```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @SearchLocation NVARCHAR(100) = NULL,
    @SearchReputation INT = NULL,
    @Debug_PrintQuery TINYINT = 0,
    @Debug_ExecuteQuery TINYINT = 1 AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @StringToExecute = @crlf + N'SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1 ' + @crlf;

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @SearchDisplayName ' + @crlf;

    IF @SearchLocation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Location LIKE @SearchLocation ' + @crlf;

    IF @SearchReputation IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND Reputation = @SearchReputation ' + @crlf;

    SET @StringToExecute = @StringToExecute + N' ORDER BY CreationDate; ';

    IF @Debug_PrintQuery = 1
        PRINT @StringToExecute;

    IF @Debug_ExecuteQuery = 1
        EXEC sp_executesql @StringToExecute,
            N'@SearchDisplayName NVARCHAR(100), @SearchLocation NVARCHAR(100), @SearchReputation INT',
            @SearchDisplayName, @SearchLocation, @SearchReputation;
END
GO
```

Useful for dangerous debugging

Helps you avoid running queries that:

- Modify data
- Delete files
- Configure SQL Server itself

Use it with:

- @Debug_PrintQuery = 1
- @Debug_ExecuteQuery = 0

2.4 p63

PRINT is limited to 4,000 chars.

For longer, use Helper_LongPrint:

https://www.codeproject.com/Articles/18881/SQL-String-Printing

2.4 p64

Before we go on...

2.4 p65

I'm going to simplify it.

I need to show you other features, so I'm going to simplify this to just have one search: DisplayName.

```
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @OrderBy NVARCHAR(100) = 'CreationDate',
    @Debug_PrintQuery TINYINT = 0 AS
BEGIN
```

And let's implement a dynamic OrderBy.

```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @OrderBy NVARCHAR(100) = 'CreationDate',
    @Debug_PrintQuery TINYINT = 0 AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @StringToExecute = @crlf + N'SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1 ' + @crlf;

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @SearchDisplayName ' + @crlf;

    IF @OrderBy IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' ORDER BY ' + @OrderBy;

    IF @Debug_PrintQuery = 1
        PRINT @StringToExecute;

    EXEC sp_executesql @StringToExecute,
        N'@SearchDisplayName NVARCHAR(100)',
        @SearchDisplayName;
END
```

2.4 p68

```
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @OrderBy NVARCHAR(100) = 'CreationDate',
    @Debug_PrintQuery TINYINT = 0 AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000);
    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @StringToExecute = @crlf + N'SELECT /* usp_SearchUsers */ * FROM dbo.

    IF @SearchDisplayName IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' AND DisplayName LIKE @Se

    IF @OrderBy IS NOT NULL
        SET @StringToExecute = @StringToExecute + N' ORDER BY ' + @OrderBy;
```

2.4 p69

Sort by reputation...

```
652  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%',
653      @OrderBy = 'Reputation', @Debug_PrintQuery = 1;
```

200 % Results Messages Execution plan

```
SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1
 AND DisplayName LIKE @SearchDisplayName
 ORDER BY Reputation
```

2.4 p70

Descending works too...

```
656  /* And try descending order: */
657  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%',
658  @OrderBy = 'Reputation DESC', @Debug_PrintQuery = 1;
659  GO
```

200 % Results Messages Execution plan

```
SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1
 AND DisplayName LIKE @SearchDisplayName
 ORDER BY Reputation DESC
```

2.4 p71

I have some work to do...

```
656  /* And try descending order: */
657  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%',
658  @OrderBy = 'Reputation DESC', @Debug_PrintQuery = 1;
659  GO
```

200 %

Results | Messages | Execution plan

	Id	AboutMe	Age	CreationDate	DisplayName	Reputation	DownVotes	EmailHash	LastA
1	284758	Brent is an .net tech-stack expert. He engages ...	NULL	2010-03-02 20:52:18.357	Brent Arias	12880	6	NULL	2018
2	41222	I write software, mostly for iOS and the Mac, occ...	NULL	2008-11-27 02:15:14.163	Brent Royal-Gordon	12482	39	NULL	2018
3	26837	I make Microsoft SQL Server faster and more reli...	NULL	2008-10-10 14:26:33.540	Brent Ozar	11825	12	NULL	2018
4	151269	Tech Lead for Enterprise Accounts group for sto...	NULL	2009-08-05 18:32:59.130	Brent Baisley	11705	9	NULL	2018
5	159658		NULL	2009-08-19 23:19:05.970	Brent Nash	11638	18	NULL	2018
6	432910	Owner and chief developer for BNR Branding So...	NULL	2010-08-27 12:34:28.643	Brent Friar	10449	26	NULL	2018
7	584846	Web and database specialist with strong empha...	NULL	2011-01-21 18:07:30.787	Brent Washburne	8714	84	NULL	2018
8	633150	SOreadytohelp	NULL	2011-02-24 21:30:58.370	Brent Worden	7932	31	NULL	2018

2.4 p72

The dark side

```
662  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%',
663  @OrderBy = 'Reputation DESC; SELECT * FROM sys.databases;',
664  @Debug_PrintQuery = 1;
665  GO
```

200 %

Results | Messages | Execution plan

	Id	AboutMe	Age	CreationDate	DisplayName	DownVotes	EmailHash	LastAccessDate	Location
1	284758	Brent is an .net tech-stack expert. He engages ...	NULL	2010-03-02 20:52:18.357	Brent Arias	6	NULL	2018-05-31 20:31:03.773	Phoenix, AZ
2	41222	I write software, mostly for iOS and the Mac, occ...	NULL	2008-11-27 02:15:14.163	Brent Royal-Gordon	39	NULL	2018-05-29 03:34:38.563	San Clemente, CA
3	26837	I make Microsoft SQL Server faster and more reli...	NULL	2008-10-10 14:26:33.540	Brent Ozar	12	NULL	2018-06-02 10:45:38.307	Chicago, IL
4	151269	Tech Lead for Enterprise Accounts group for sto...	NULL	2009-08-05 18:32:59.130	Brent Baisley	9	NULL	2018-05-31 02:14:27.147	10004
5	159658		NULL	2009-08-19 23:19:05.970	Brent Nash	18	NULL	2018-06-02 05:05:32.350	NULL
6	432910	Owner and chief developer for BNR Branding So...	NULL	2010-08-27 12:34:28.643	Brent Friar	26	NULL	2018-05-31 20:51:41.857	Huntersville, NC
7	584846	Web and database specialist with strong empha...	NULL	2011-01-21 18:07:30.787	Brent Washburne	84	NULL	2018-06-02 22:22:56.843	Portland, OR
8	633150	SOreadytohelp	NULL	2011-02-24 21:30:58.370	Brent Worden	31	NULL	2018-06-01 00:13:00.200	Minnesota
9	925897	I like doing things manually. But I'm finally picking...	NULL	2011-09-02 19:33:58.020	brentonstrine	43	NULL	2018-05-23 14:00:56.960	Atlanta, GA

	name	database_id	source_database_id	owner_sid	create_date	compatibility_level	collation_name
1	master	1	NULL	0x01	2003-04-08 09:13:36.390	150	SQL_Latin1_General_CP1_CI_AS
2	tempdb	2	NULL	0x01	2022-09-14 00:02:45.230	150	SQL_Latin1_General_CP1_CI_AS
3	model	3	NULL	0x01	2003-04-08 09:13:36.390	150	SQL_Latin1_General_CP1_CI_AS
4	msdb	4	NULL	0x01	2019-09-24 14:21:42.270	150	SQL_Latin1_General_CP1_CI_AS
5	DBAtools	5	NULL	0x0105000000000005150000000D67423C5FC18FA4DBE2C6E...	2019-11-28 13:46:30.503	150	SQL_Latin1_General_CP1_CI_AS
6	StackO...	6	NULL	0x0106000000000005500000000DCA88F14B79FD47A992A3D...	2022-08-24 10:19:06.213	140	SQL_Latin1_General_CP1_CI_AS
7	StackO...	8	NULL	0x0106000000000005500000000DCA88F14B79FD47A992A3D...	2022-08-24 10:39:12.713	100	SQL_Latin1_General_CP1_CI_AS

The query it built

```
662  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%',
663    @OrderBy = 'Reputation DESC; SELECT * FROM sys.databases;',
664    @Debug_PrintQuery = 1;
665  GO
```

200 %

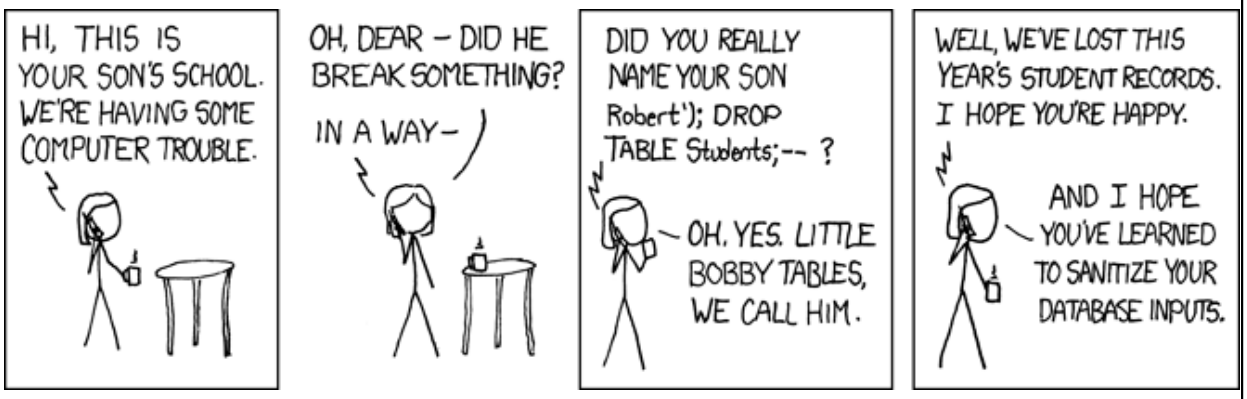
Results | Messages | Execution plan

```
SELECT /* usp_SearchUsers */ * FROM dbo.Users WHERE 1 = 1
 AND DisplayName LIKE @SearchDisplayName
 ORDER BY Reputation DESC; SELECT * FROM sys.databases;
```

Uh oh

2.4 p74

https://xkcd.com/327/



2.4 p75

SQL injection resources

Bert Wagner's posts and videos:
https://bertwagner.com/posts/4-sql-injection-techniques-for-stealing-data/

Automatic SQL injection script kits:
https://www.kitploit.com/p/sql-injection-tools.html

The safer way

Listen to the user's input, but build your own string:

```
IF @OrderBy IS NOT NULL
    BEGIN
    SET @StringToExecute = @StringToExecute + N' ORDER BY ';

    SET @StringToExecute = @StringToExecute +
        CASE WHEN @OrderBy LIKE 'CreationDate%' THEN N' u.CreationDate '
             WHEN @OrderBy LIKE 'DisplayName%' THEN N' u.DisplayName '
             WHEN @OrderBy LIKE 'Location%' THEN N' u.Location '
             WHEN @OrderBy LIKE 'Reputation%' THEN N' u.Reputation '
             ELSE N' u.Id '   /* Or whatever default ordering you want, ideally to make SQL's life easier */
        END;

    IF @OrderBy LIKE '% DESC'
        SET @StringToExecute = @StringToExecute + N' DESC ';
```

2.4 p77

That only sorts by one column

In the demo scripts, I have a multi-column solution.

It's going to use something you don't like.

Normally, it's a bad thing – but here it's fine.

2.4 p78

```
IF @OrderBy IS NOT NULL
    BEGIN
    SET @StringToExecute = @StringToExecute + N' ORDER BY ';

    /* Split the ordering string */
    DECLARE cursor_OrderBy CURSOR
      FOR SELECT ItemNumber, RTRIM(LTRIM(Item))
      FROM dbo.STRING_SPLIT(@OrderBy, ',')
      ORDER BY ItemNumber;

    OPEN cursor_OrderBy;
    FETCH NEXT FROM cursor_OrderBy INTO @OrderByItemNumber, @OrderByItem;
    WHILE @@FETCH_STATUS = 0
        BEGIN

        IF @OrderByItemNumber > 1
            SET @StringToExecute = @StringToExecute + ' , ';

        SET @StringToExecute = @StringToExecute +
            CASE WHEN @OrderByItem LIKE 'CreationDate%' THEN N' u.CreationDate '
                 WHEN @OrderByItem LIKE 'DisplayName%' THEN N' u.DisplayName '
                 WHEN @OrderByItem LIKE 'Location%' THEN N' u.Location '
                 WHEN @OrderByItem LIKE 'Reputation%' THEN N' u.Reputation '
                 ELSE N' u.Id '   /* Or whatever default ordering you want, ideally to make SQL's life easier */
            END;

        IF @OrderByItem LIKE '% DESC'
            SET @StringToExecute = @StringToExecute + N' DESC ';

        FETCH NEXT FROM cursor_OrderBy INTO @OrderByItemNumber, @OrderByItem;
        END /* Cursor */
    CLOSE cursor_OrderBy;
    DEALLOCATE cursor_OrderBy;
```

Next tip:
use CASES to make debugging easier.

2.4 p80

```
978  CREATE OR ALTER PROC dbo.usp_SearchUsers
979      @SearchDisplayName NVARCHAR(100) = NULL,
980      @ShowTotalVotes BIT = 0,
981      @Debug_PrintQuery TINYINT = 0 AS
982  BEGIN
983      DECLARE @StringToExecute NVARCHAR(4000), @Select NVARCHAR(4000),
984              @From NVARCHAR(4000), @Where NVARCHAR(4000);
985
986      DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
987      SET @Select = @crlf + N'SELECT /* usp_SearchUsers */ u.Id, u.DisplayName, u.Location ' + @crlf;
988      SET @From = N' FROM dbo.Users u ' + @crlf;
989      SET @Where = N' WHERE ' + @crlf;
990
991      IF @SearchDisplayName IS NOT NULL
992          SET @Where = @Where + N' AND DisplayName LIKE @SearchDisplayName ' + @crlf;
993
994      IF @ShowTotalVotes = 1
995          BEGIN
996          SET @Select = @Select + N' , VotesCast = (SELECT SUM(1) FROM dbo.Votes v WHERE u.Id = v.UserId) ' + @crlf;
997          END
998
999      /* Autobots, unite! */
1000     SET @StringToExecute = @Select + @From + @Where;
1001
1002     IF @Debug_PrintQuery = 1
1003         PRINT @StringToExecute;
1004
1005     EXEC sp_executesql @StringToExecute,
1006         N'@SearchDisplayName NVARCHAR(100)',
1007         @SearchDisplayName;
1008 END
```

Find the bug.

Debugging dynamic SQL is awful.

```
1011  /* Test it: */
1012  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @ShowTotalVotes = 0
1013  GO
```

150 %

Messages

```
Msg 156, Level 15, State 1, Line 1016
Incorrect syntax near the keyword 'AND'.
```

The line number is meaningless.

Is the error in our normal SQL, or our dynamic SQL?

2.4 p82

Make this message more useful.

```
Msg 156, Level 15, State 1, Line 1016
Incorrect syntax near the keyword 'AND'.
```

My normal SQL:

```
SELECT * FROM dbo.MyTable WHERE Id = @MyId
```

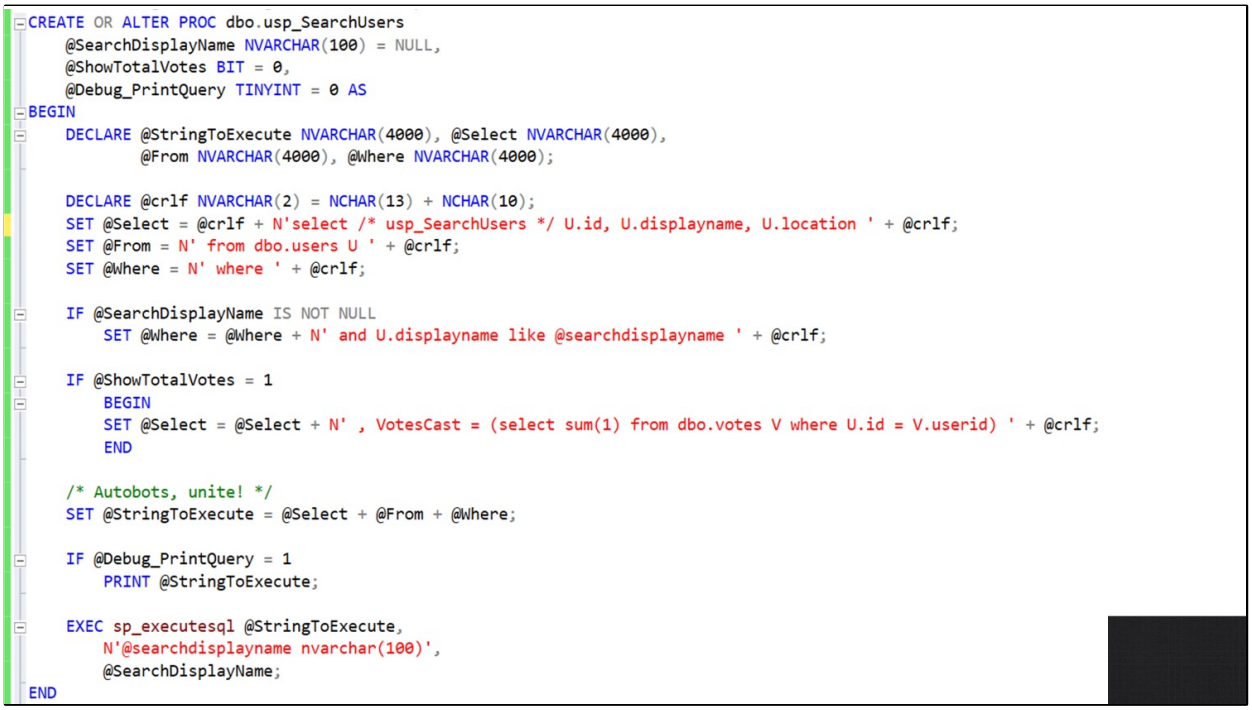
My dynamic SQL:

```
select * from dbo.MyTable WHERE Id = @myid
```

2.4 p83

```
Regular query:
*/
DECLARE @MyVariable INT = 1;
SELECT DisplayName FROM dbo.Users u WHERE Id = @MyVariable AND 1 = 1;
GO
/* Inside my dynamic SQL:  */
declare @myvariable int = 1;
select displayname from dbo.users U where id = @myvariable and 1 = 1;
GO
```

2.4 p84



```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @ShowTotalVotes BIT = 0,
    @Debug_PrintQuery TINYINT = 0 AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000), @Select NVARCHAR(4000),
            @From NVARCHAR(4000), @Where NVARCHAR(4000);

    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @Select = @crlf + N'select /* usp_SearchUsers */ U.id, U.displayname, U.location ' + @crlf;
    SET @From = N' from dbo.users U ' + @crlf;
    SET @Where = N' where ' + @crlf;

    IF @SearchDisplayName IS NOT NULL
        SET @Where = @Where + N' and U.displayname like @searchdisplayname ' + @crlf;

    IF @ShowTotalVotes = 1
        BEGIN
        SET @Select = @Select + N' , VotesCast = (select sum(1) from dbo.votes V where U.id = V.userid) ' + @crlf;
        END

    /* Autobots, unite! */
    SET @StringToExecute = @Select + @From + @Where;

    IF @Debug_PrintQuery = 1
        PRINT @StringToExecute;

    EXEC sp_executesql @StringToExecute,
        N'@searchdisplayname nvarchar(100)',
        @SearchDisplayName;
END
```

Ah, the error is in dynamic SQL.

```
1074  EXEC usp_SearchUsers @SearchDisplayName = 'Brent%', @ShowTotalVotes = 0;
1075  GO
1076
```

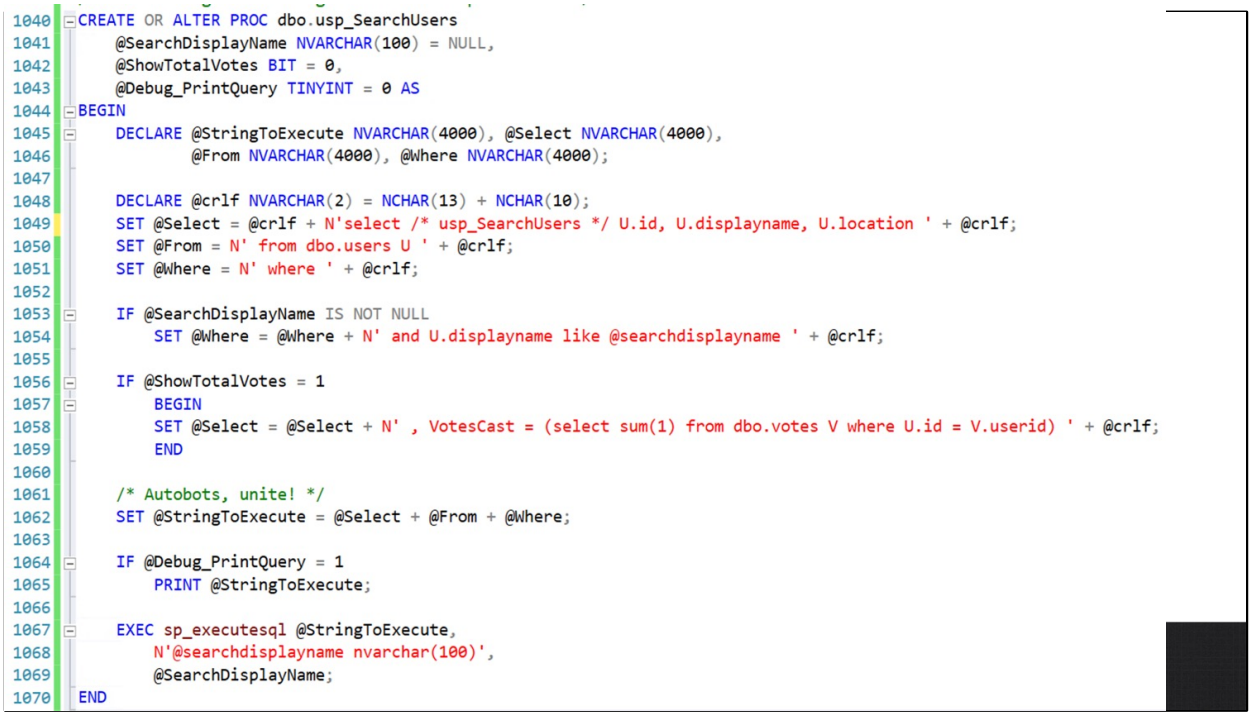
150 %

Messages

```
Msg 156, Level 15, State 1, Line 1078
Incorrect syntax near the keyword 'and'.
```

Lower case

2.4 p86



```sql
CREATE OR ALTER PROC dbo.usp_SearchUsers
    @SearchDisplayName NVARCHAR(100) = NULL,
    @ShowTotalVotes BIT = 0,
    @Debug_PrintQuery TINYINT = 0 AS
BEGIN
    DECLARE @StringToExecute NVARCHAR(4000), @Select NVARCHAR(4000),
            @From NVARCHAR(4000), @Where NVARCHAR(4000);

    DECLARE @crlf NVARCHAR(2) = NCHAR(13) + NCHAR(10);
    SET @Select = @crlf + N'select /* usp_SearchUsers */ U.id, U.displayname, U.location ' + @crlf;
    SET @From = N' from dbo.users U ' + @crlf;
    SET @Where = N' where ' + @crlf;

    IF @SearchDisplayName IS NOT NULL
        SET @Where = @Where + N' and U.displayname like @searchdisplayname ' + @crlf;

    IF @ShowTotalVotes = 1
        BEGIN
        SET @Select = @Select + N' , VotesCast = (select sum(1) from dbo.votes V where U.id = V.userid) ' + @crlf;
        END

    /* Autobots, unite! */
    SET @StringToExecute = @Select + @From + @Where;

    IF @Debug_PrintQuery = 1
        PRINT @StringToExecute;

    EXEC sp_executesql @StringToExecute,
        N'@searchdisplayname nvarchar(100)',
        @SearchDisplayName;
END
```

There's so much more to learn.

2.4 p88

Resources

The demo script continues, plus:

https://www.brentozar.com/dynamicsql

Erland Sommarskog's posts:

https://www.sommarskog.se/dynamic_sql.html

https://www.sommarskog.se/dyn-search.html