statistics
stats
AUTO
UPDATE
create
OPTIMIZATION
columns
updatestats
cardinality
histogram
ASYNC
sys
drop
sp
Filtered
createstats
CORRELATION
dbcc
autostats
date
NoRecompute
Resample
www.tagxedo.com

Introduction to Statistics in SQL Server

Andy Warren
www.sqlandy.com
@sqlandy
www.linkedin.com/in/sqlandy

Why Do We Need Statistics?

We can’t build a good plan to get the rows we need without having an idea of how many rows we’re going to get!

A Tale of Two Queries

Question: Will the two queries below use a similar query plan?

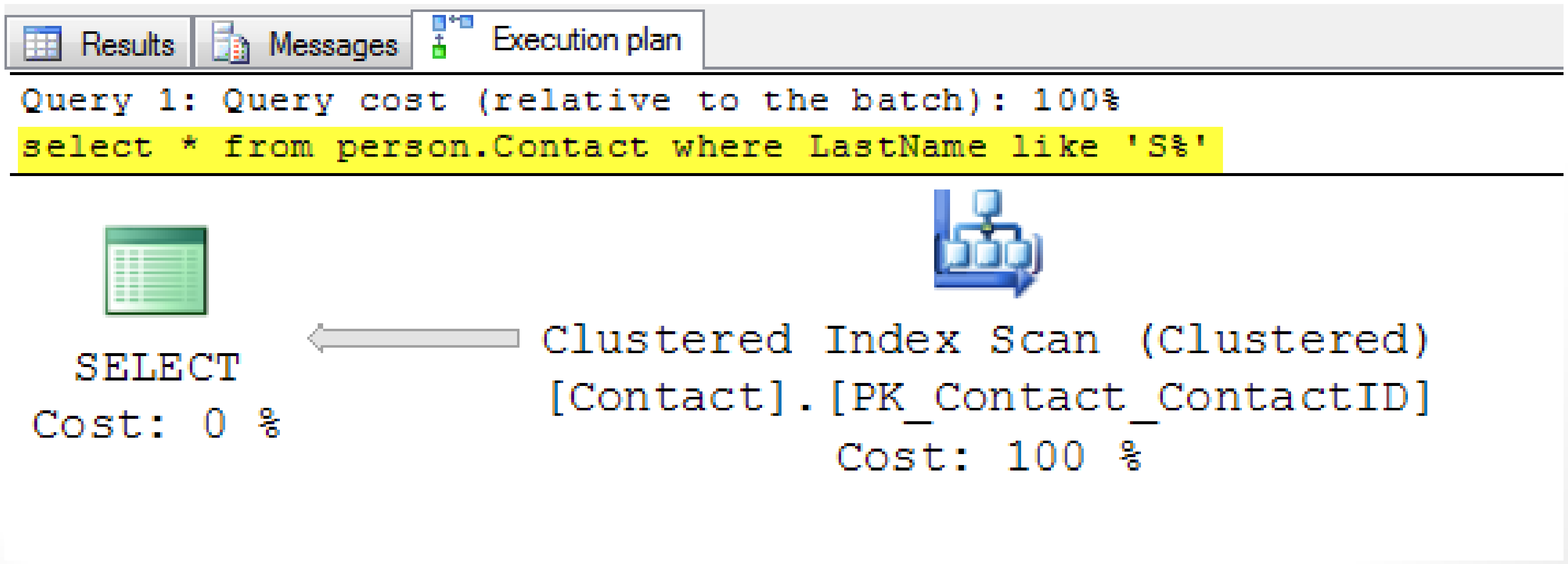
A) select * from person.Contact where LastName like 'S%‘
B) select * from person.Contact where LastName like 'SM%'

Not enough info you say? What if I told you that:

Query A returns 2694 rows
Query B return 669 rows

LastName Like S% = Scan

For Query A, we see that SQL has decided to do a table scan – brute force!

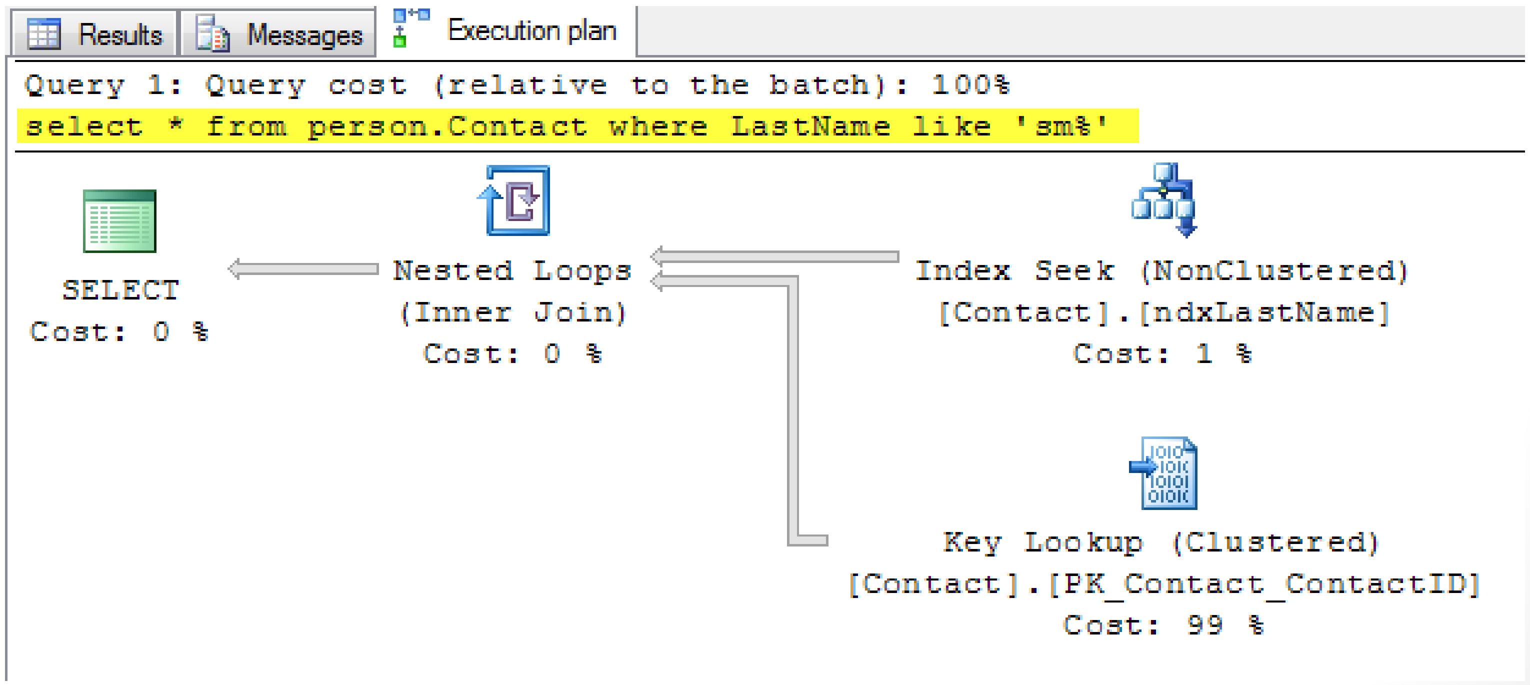

Results
Messages
Execution plan

```
Query 1: Query cost (relative to the batch): 100%
select * from person.Contact where LastName like 'S%'
```

SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[Contact].[PK_Contact_ContactID]
Cost: 100 %

LastName Like Sn% = Lookup

For Query B, we see that SQL has decided to take a lighter weight approach – a bookmark lookup



```
Query 1: Query cost (relative to the batch): 100%
select * from person.Contact where LastName like 'sm%'
```

Why Are the Plans Different?

SQL has a method that we can use to get an approximation of how many rows will be returned – that is our “statistics”. That in turn allows us to make smarter decisions about the plan we choose for the query.

High Level Overview of Stats

Created in various ways:

- Auto creation
- Based on indexes
- Manually

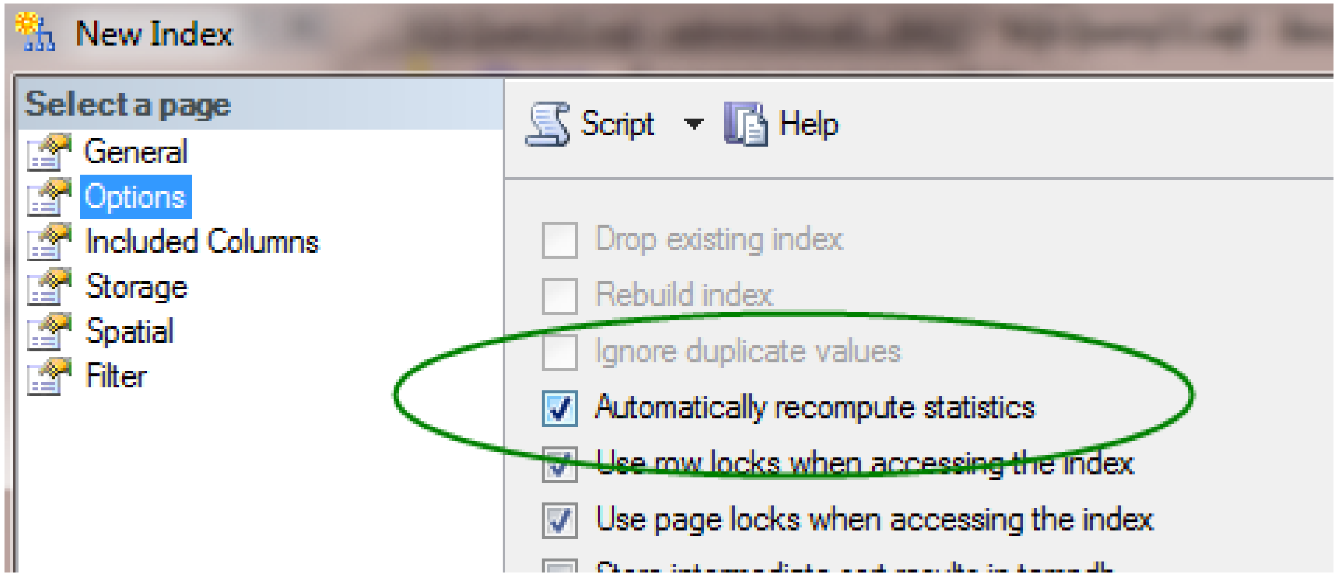
Maintained in various ways:

- Auto update based on thresholds
- Index rebuilds
- Maintenance plans
- Manual updates

Are a point in time view of the data distribution

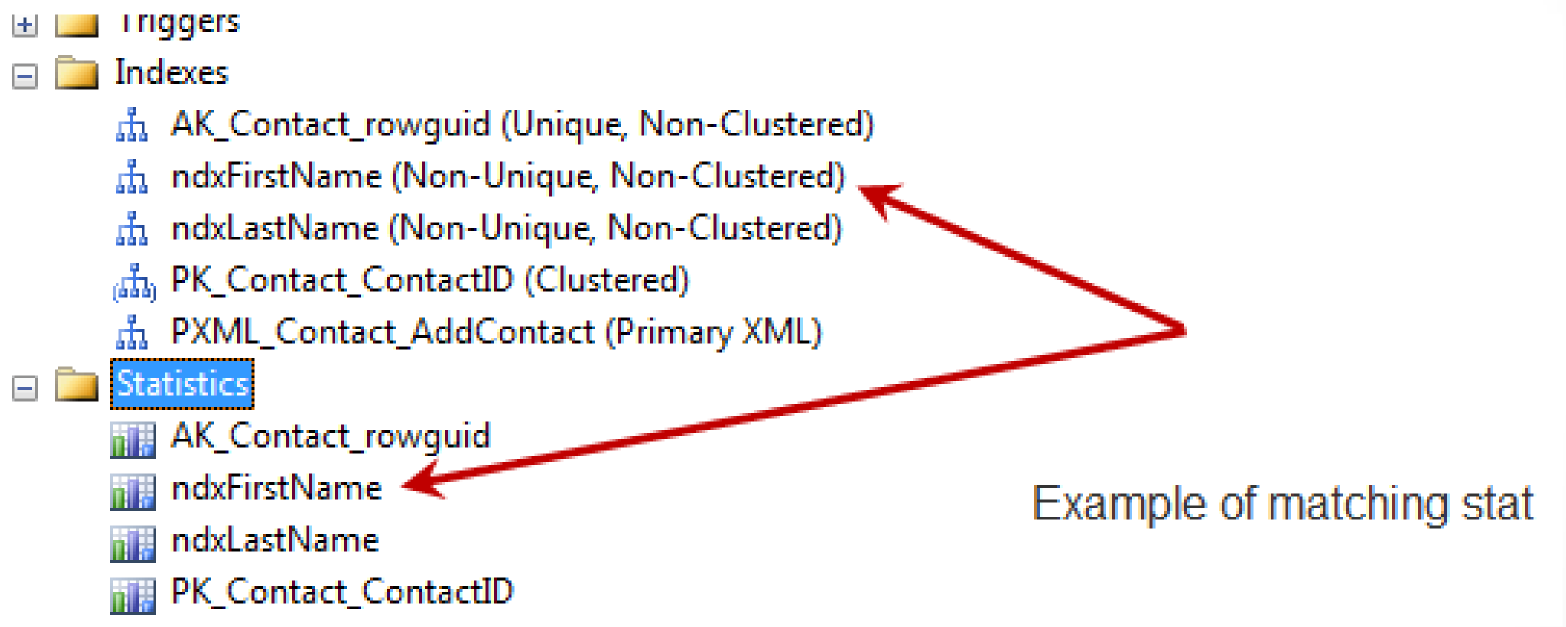
Creating Stats – Via Indexes

When an index is created a matching stat is created. This will handle 95% of your stats needs.



- [ ] Drop existing index
- [ ] Rebuild index
- [ ] Ignore duplicate values
- [✓] Automatically recompute statistics
- [✓] Use row locks when accessing the index
- [✓] Use page locks when accessing the index

Creating Stats – Via Indexes

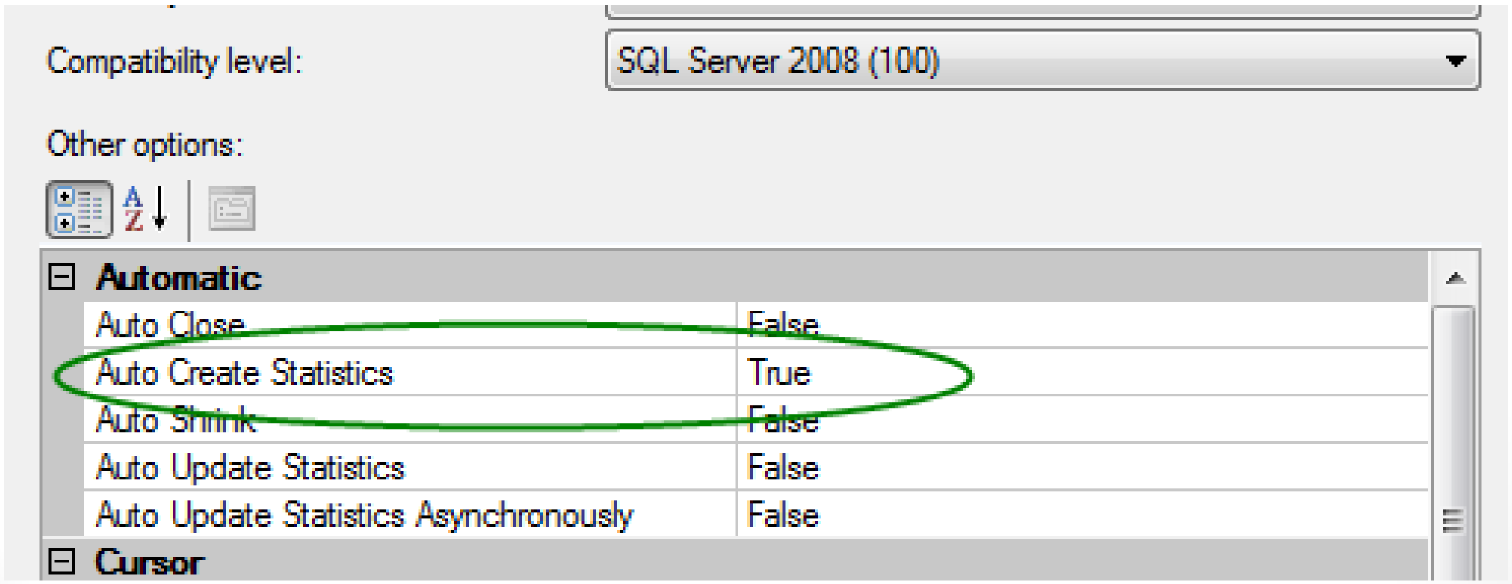


Triggers
Indexes
AK_Contact_rowguid (Unique, Non-Clustered)
ndxFirstName (Non-Unique, Non-Clustered)
ndxLastName (Non-Unique, Non-Clustered)
PK_Contact_ContactID (Clustered)
PXML_Contact_AddContact (Primary XML)
Statistics
AK_Contact_rowguid
ndxFirstName
ndxLastName
PK_Contact_ContactID

Example of matching stat

Creating Stats - Automatic

The default setting for each database is to have automatic creation of stats enabled, allowing SQL to create a new stat if a query uses a column in a where clause or join that doesn't have a stat. Think of this as a safety net for stats.



Compatibility level: SQL Server 2008 (100)

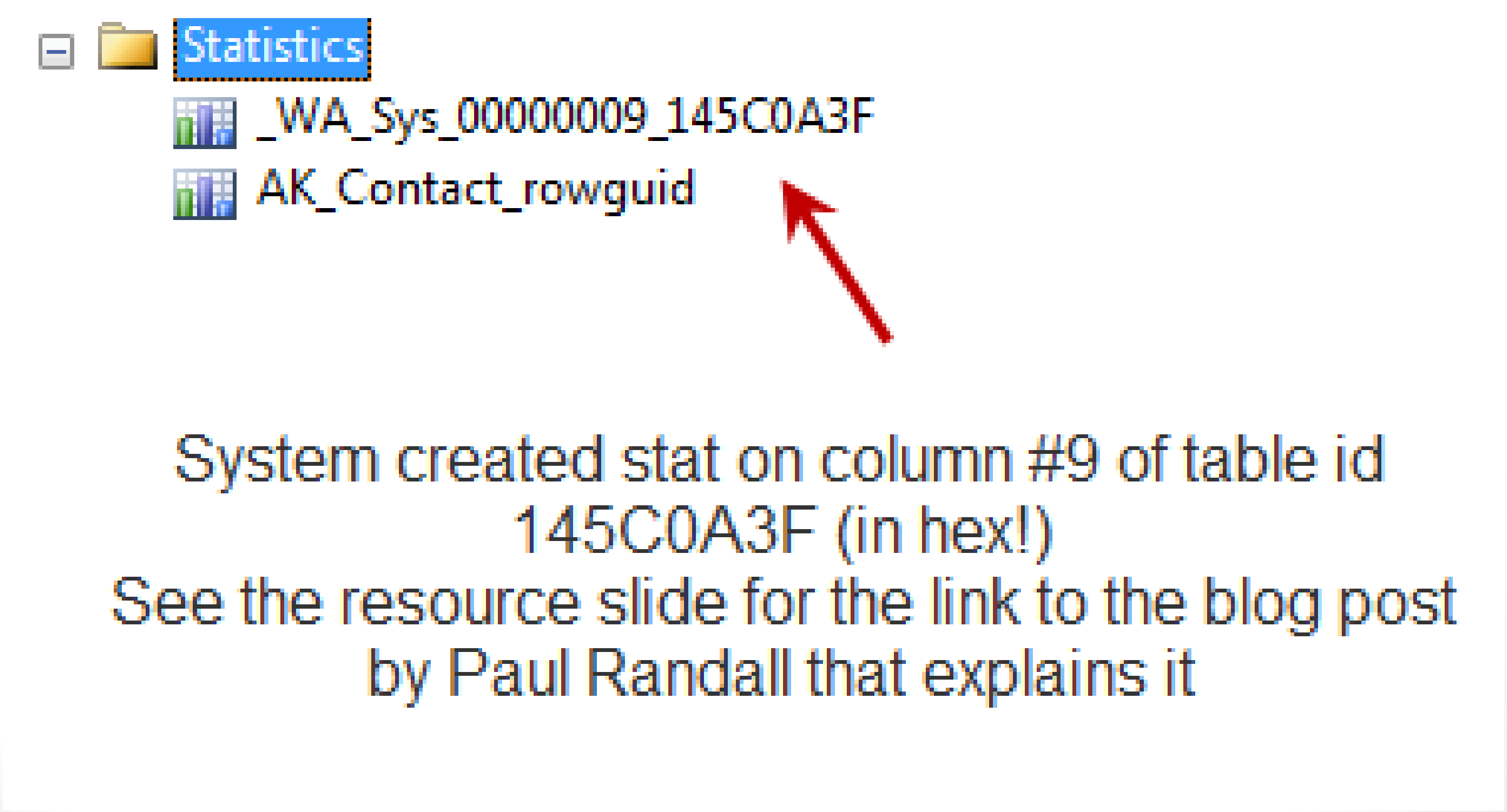
Other options:

Automatic	
Auto Close	False
Auto Create Statistics	True
Auto Shrink	False
Auto Update Statistics	False
Auto Update Statistics Asynchronously	False

Cursor

Creating Stats - Automatic

Example: System created statistic



_WA_Sys_00000009_145C0A3F

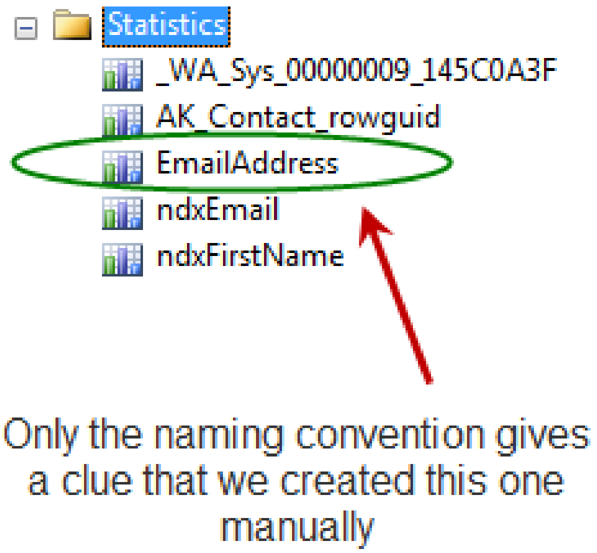
AK_Contact_rowguid

System created stat on column #9 of table id 145C0A3F (in hex!)
See the resource slide for the link to the blog post by Paul Randall that explains it

Creating Stats - Manually

It’s not common, but you might need to create a statistic manually. Here is an example:

```
create statistics EmailAddress on person.contact (EmailAddress)
```



Only the naming convention gives a clue that we created this one manually

How are Statistics Updated?

Unlike indexes, statistics are a batch operation. That decreases the load on the system, but it means that over time the accuracy of the stats can decrease as the distribution of the data changes from what it was at the time we built our statistic.

The fix is to periodically update our statistics:

- By association when we rebuild indexes
- Directly, either manually or via a job
- Based on thresholds if auto update enabled

Updating Stats Via Rebuild

- This only works for a true rebuild, not a defrag/reorg!
- This only works if they created the index with the default behavior to create/maintain stats (STATISTICS_NORECOMPUTE = OFF)
- This only works for index related stats. Stats created manually or auto created are not changed as part of an index rebuild *even if one of the columns is part of an index*

Updating Stats Directly

The most surgical approach to updating stats is to use UPDATE STATISTICS which allows us to:

- Update a single statistic, or all stats on a table
- Specify the sampling rate or reuse the previous sample rate
- Update index based stats, other stats, or both
- Disable automatic statistics update on a stat

If you need to update all the stats in a database, look at sp_updatestats or maintenance plans

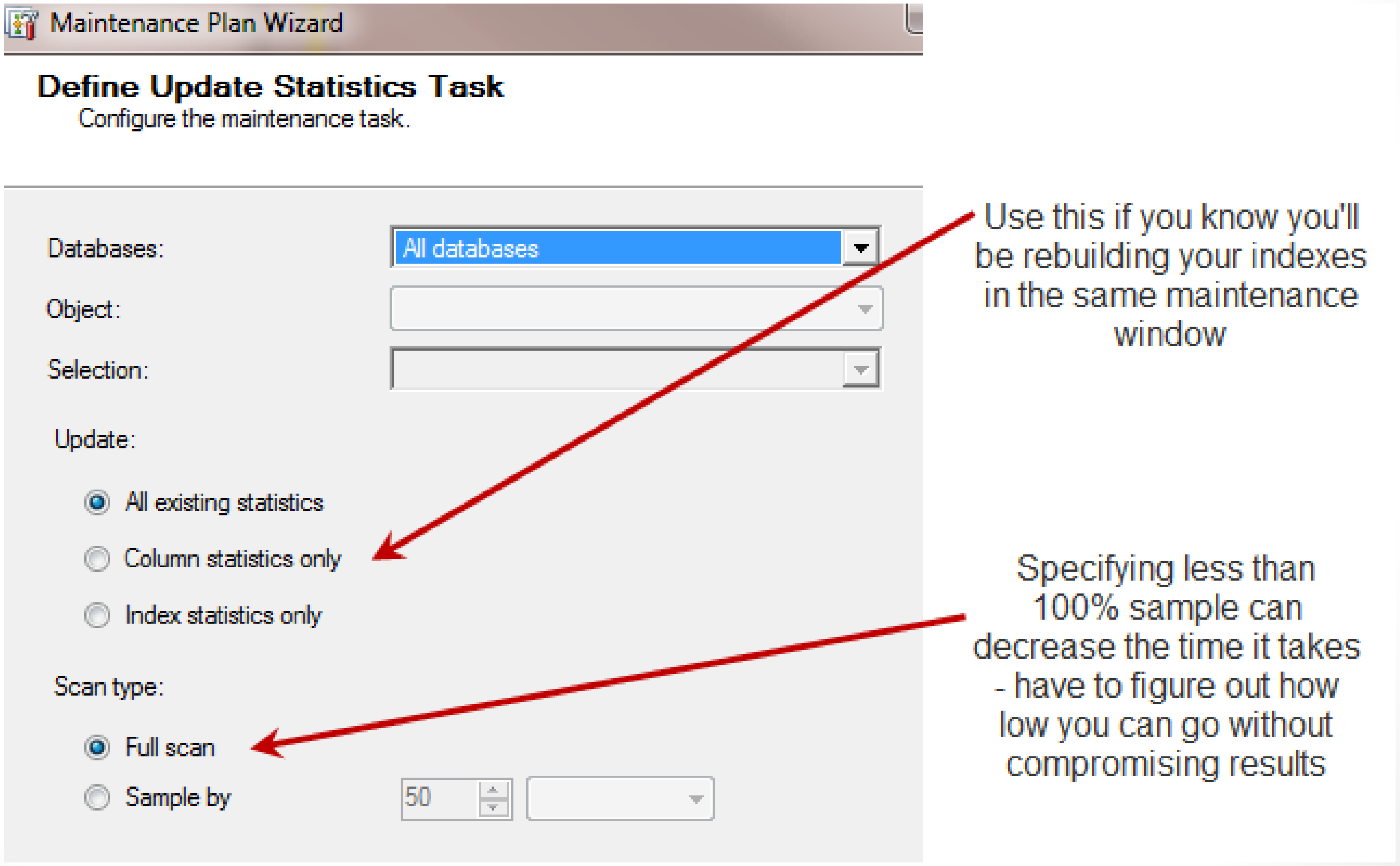
Updating Stats Directly

Examples:

- update statistics Person.contact(ndxemail) with fullscan
- update statistics Person.contact(ndxemail) with sample 50 PERCENT
- update statistics Person.contact with columns
- update statistics Person.contact with index

Note: If the table is less than 8 meg then you will get a 100% sample even if you request less.

Updating Stats Directly



Maintenance Plan Wizard

Define Update Statistics Task

Configure the maintenance task.

Databases: All databases

Object:

Selection:

Update:

- (●) All existing statistics
- ( ) Column statistics only
- ( ) Index statistics only

Scan type:

- (●) Full scan
- ( ) Sample by 50

Use this if you know you'll be rebuilding your indexes in the same maintenance window

Specifying less than 100% sample can decrease the time it takes - have to figure out how low you can go without compromising results

Updating Stats Directly

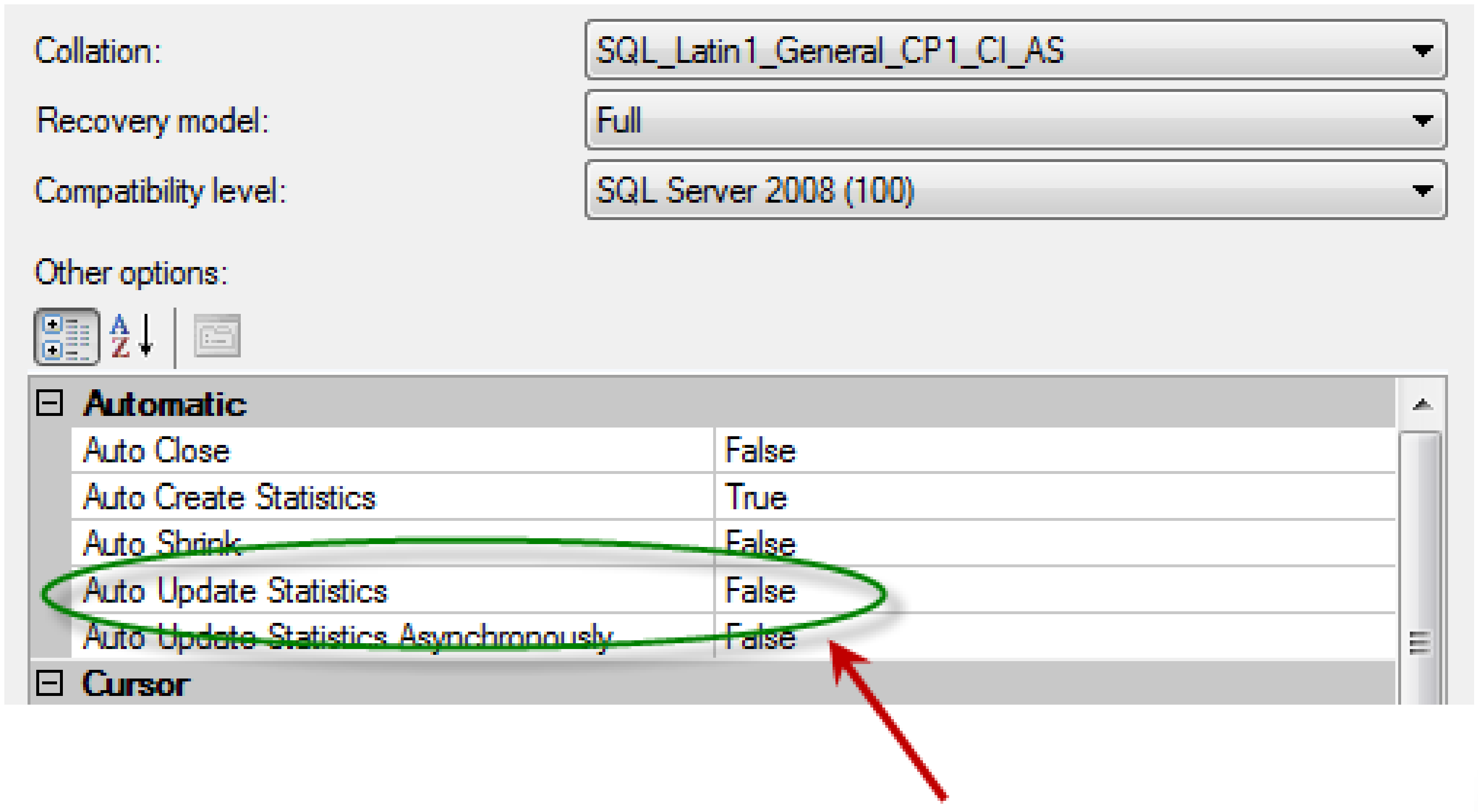
For routine maintenance you can also use sp_updatestats:

- Only updates stats that need updating (based on update thresholds we'll cover in a bit)
- Does rebuild stats for disabled non-clustered indexes
- By default will select a "default" sample rate, if you want to use the one you set, use 'resample'

Sp_updatestats

Sp_updatestats 'resample'

Updating Stats Automatically



Collation:	SQL_Latin1_General_CP1_CI_AS
Recovery model:	Full
Compatibility level:	SQL Server 2008 (100)

Other options:

Automatic	
Auto Close	False
Auto Create Statistics	True
Auto Shrink	False
Auto Update Statistics	False
Auto Update Statistics Asynchronously	False

Cursor

Default is True, and in most cases should be True!

Update Thresholds

The auto update stats event will fire based on these rules:

- When table row count goes from zero to not zero
- Table had less than 500 rows and there have been more than 500 changes to the leading column of the stat since the last stat update
- Table had more than 500 rows and there have been at least 500 + 20% changes to the leading column in the stat since the last update
- For temp tables, the first update fires at *six* changes

Viewing Stats

As you might expect, there are a few different ways to view the statistics so you can examine the details:

- Management Studio (handy, no syntax to remember!)
- DBCC Show_Statistics

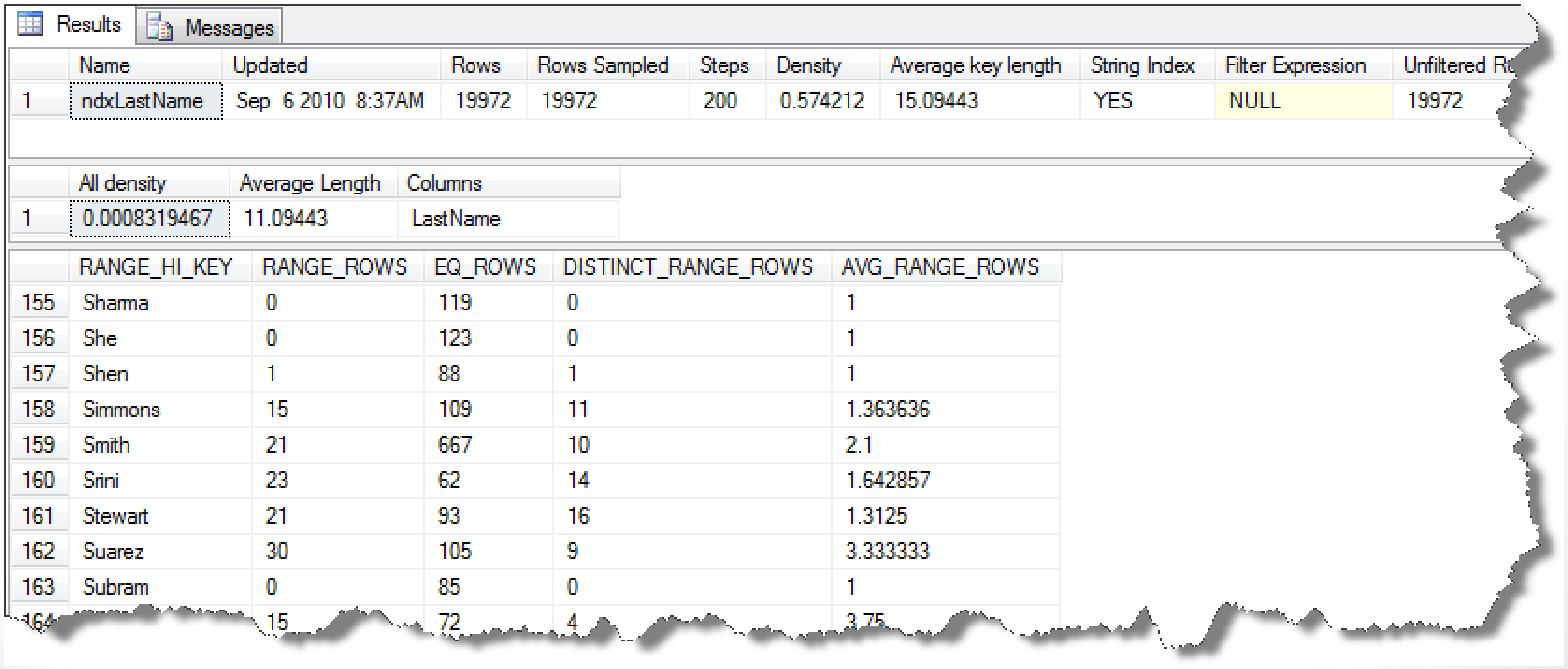
You can also get info about stats name and status by queryingL

- Sys.Stats
- Sys.Stats_Columns

We're going to focus on DBCC Show_Statistics

DBCC Show_Statistics

dbcc show_statistics ('person.contact', 'ndxlastname')



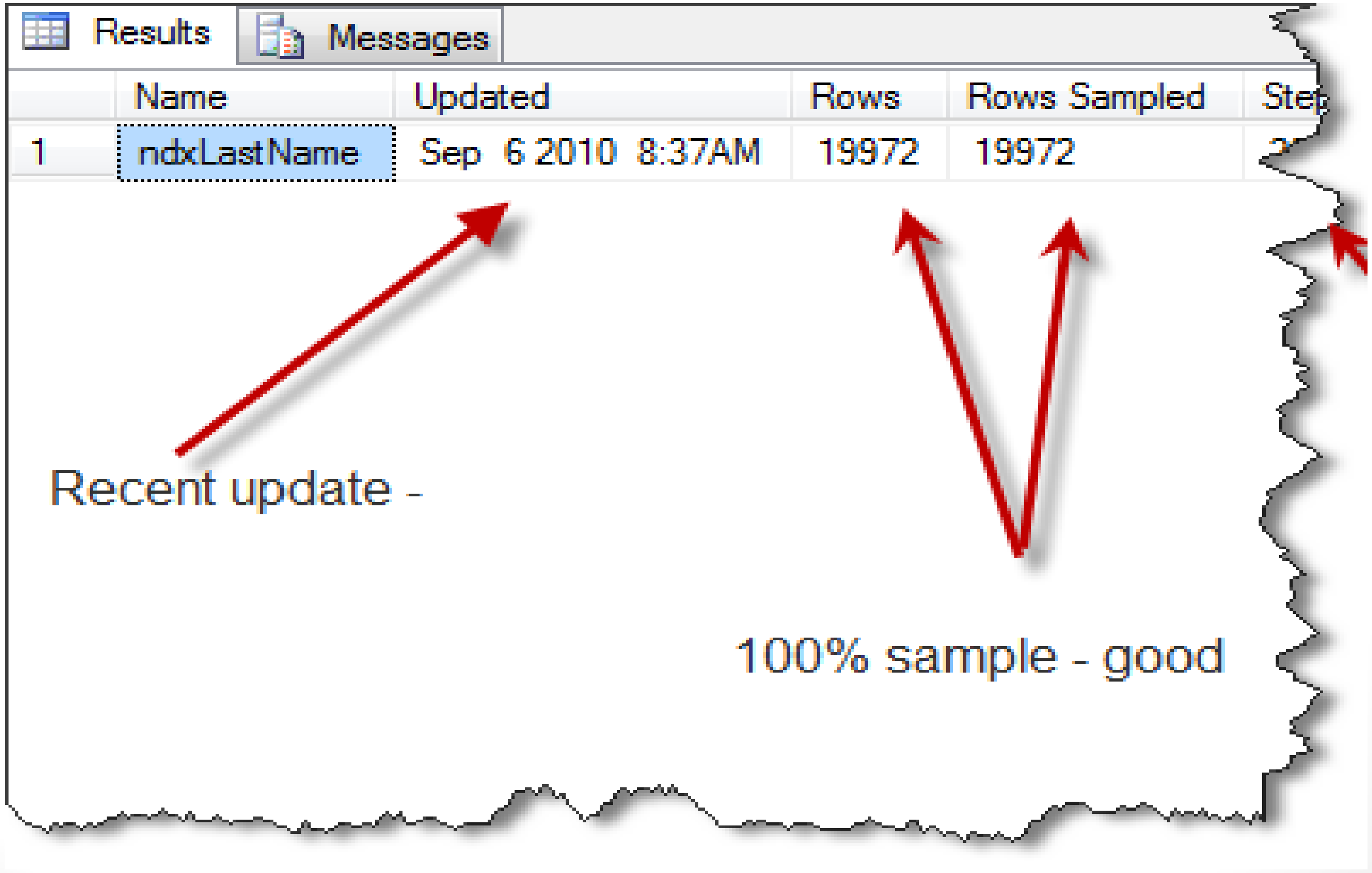
Results | Messages

	Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered R
1	ndxLastName	Sep 6 2010 8:37AM	19972	19972	200	0.574212	15.09443	YES	NULL	19972

	All density	Average Length	Columns
1	0.0008319467	11.09443	LastName

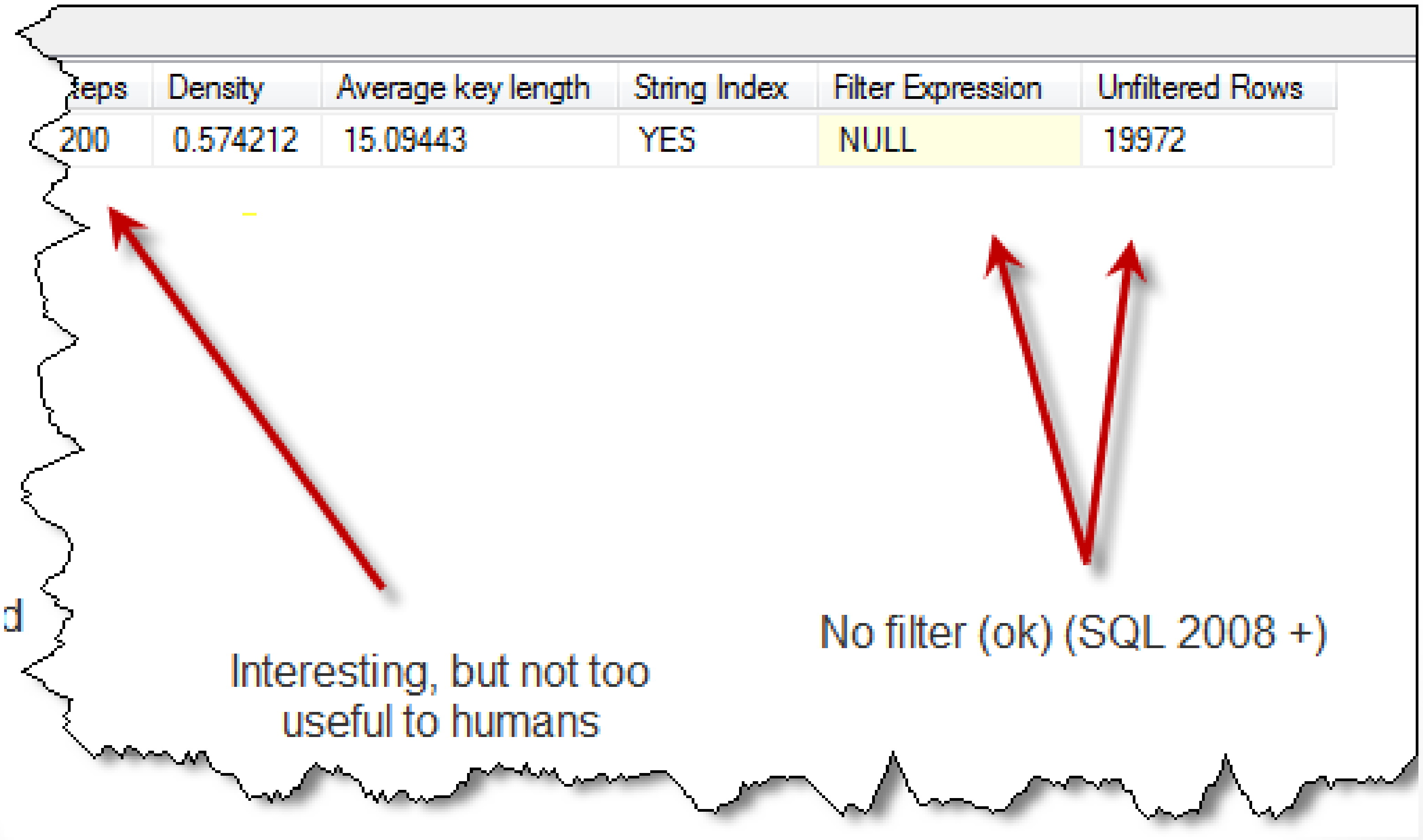
	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
155	Sharma	0	119	0	1
156	She	0	123	0	1
157	Shen	1	88	1	1
158	Simmons	15	109	11	1.363636
159	Smith	21	667	10	2.1
160	Srini	23	62	14	1.642857
161	Stewart	21	93	16	1.3125
162	Suarez	30	105	9	3.333333
163	Subram	0	85	0	1
164	[illegible]	15	72	4	3.75

Understanding the Header



	Name	Updated	Rows	Rows Sampled	Ste
1	ndxLastName	Sep 6 2010 8:37AM	19972	19972	2

Understanding The Header

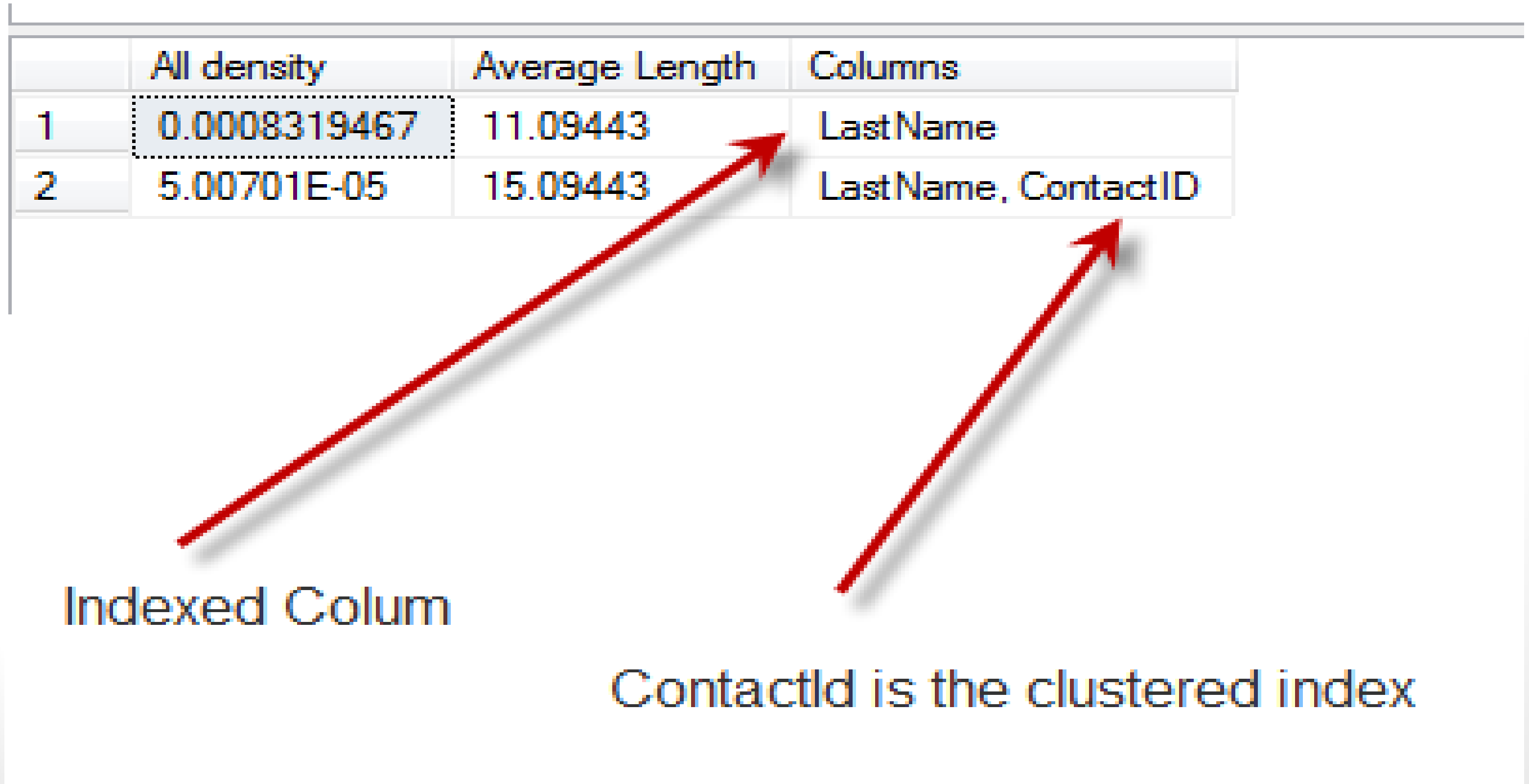


teps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows
200	0.574212	15.09443	YES	NULL	19972

d
Interesting, but not too useful to humans
No filter (ok) (SQL 2008 +)

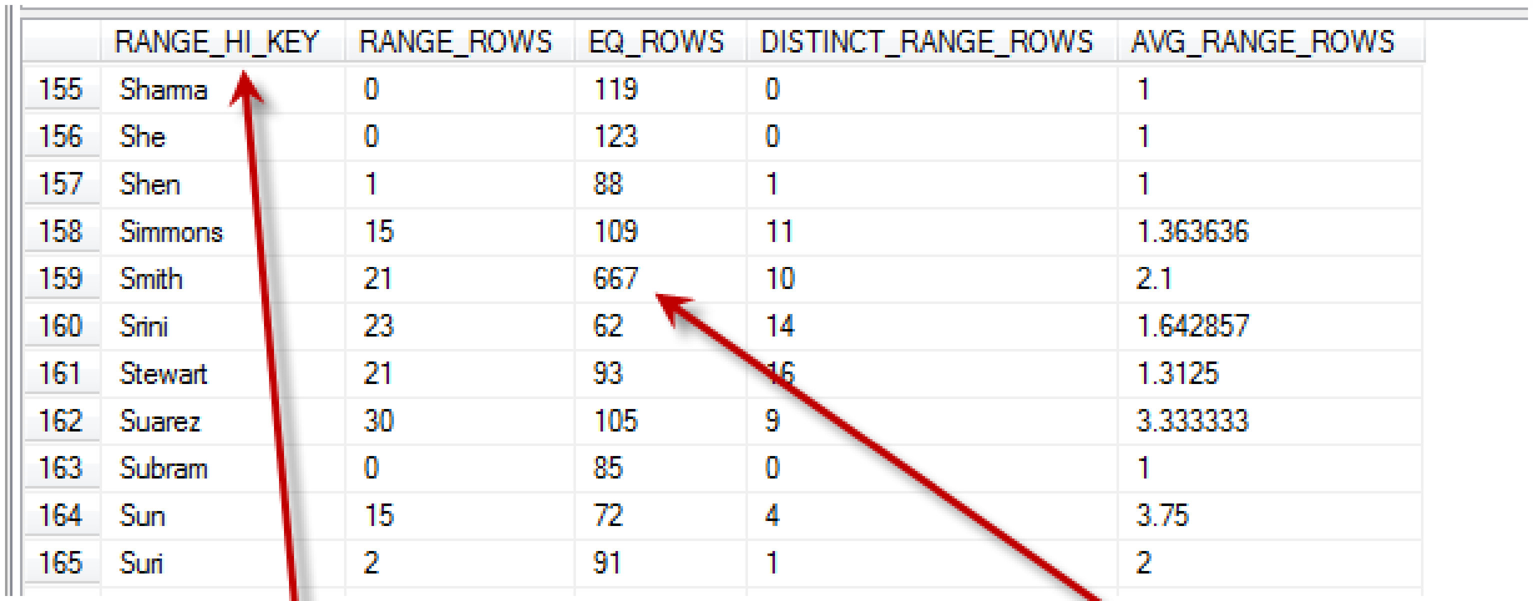
Column Densities

Not all that interesting, but sometimes can help you realize that you might gain from re-ordering columns.



	All density	Average Length	Columns
1	0.0008319467	11.09443	LastName
2	5.00701E-05	15.09443	LastName, ContactID

The Good Stuff



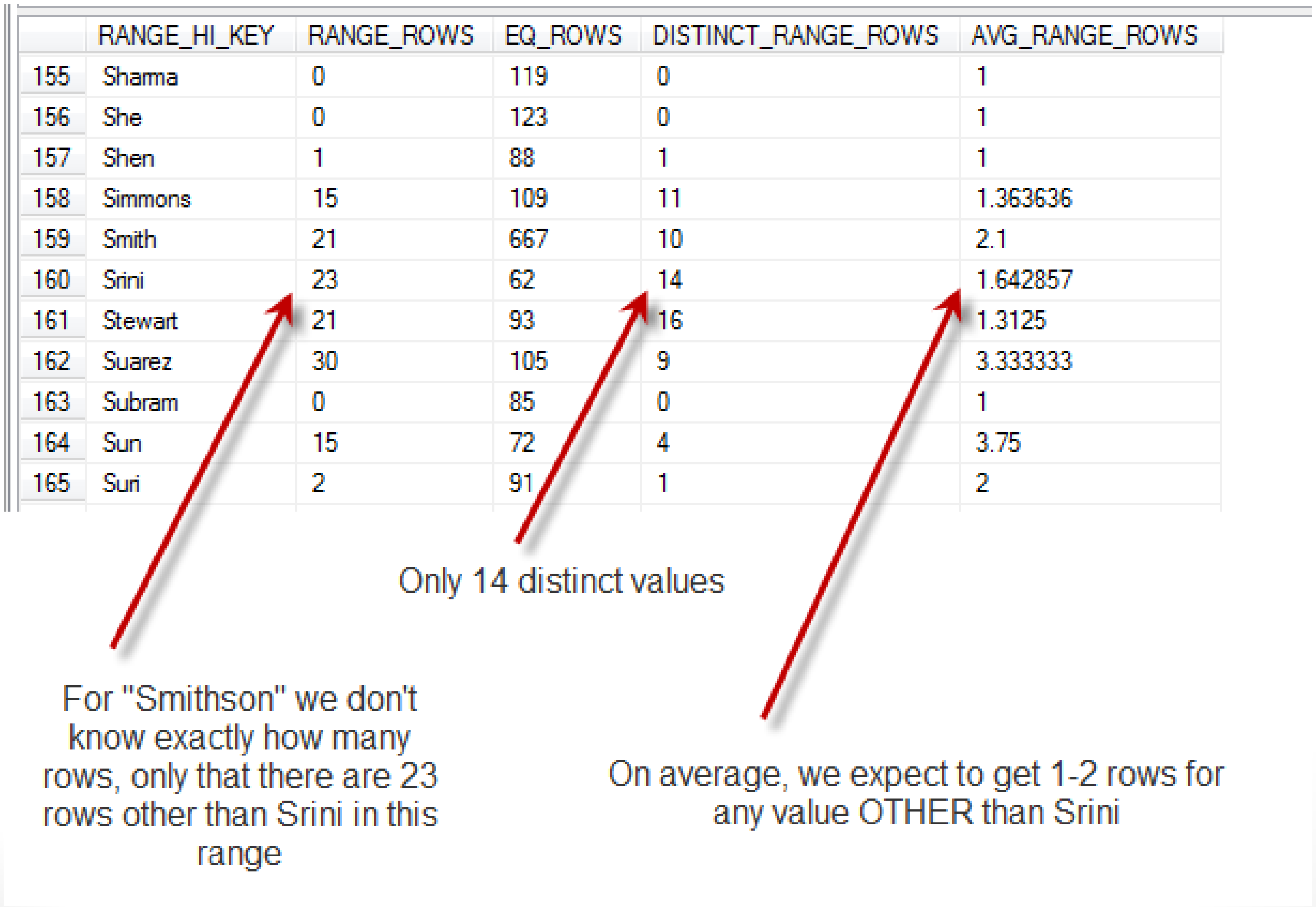
	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
155	Sharma	0	119	0	1
156	She	0	123	0	1
157	Shen	1	88	1	1
158	Simmons	15	109	11	1.363636
159	Smith	21	667	10	2.1
160	Srini	23	62	14	1.642857
161	Stewart	21	93	16	1.3125
162	Suarez	30	105	9	3.333333
163	Subram	0	85	0	1
164	Sun	15	72	4	3.75
165	Suri	2	91	1	2

For example, "Smithson" would fall into the buck on line 160.

> Smith and < = Srini

Lucky here, we know EXACTLY how many rows match Smith

More Good Stuff



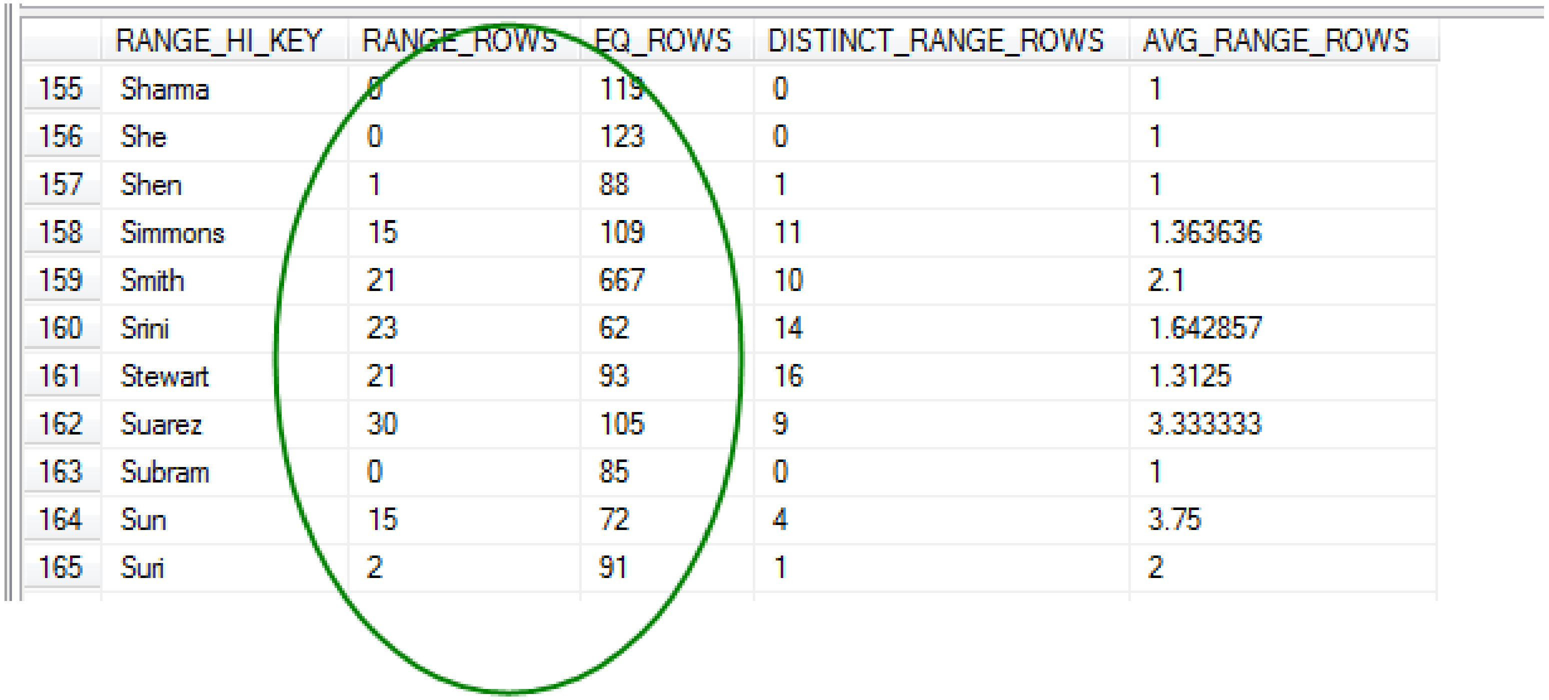
	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
155	Sharma	0	119	0	1
156	She	0	123	0	1
157	Shen	1	88	1	1
158	Simmons	15	109	11	1.363636
159	Smith	21	667	10	2.1
160	Srini	23	62	14	1.642857
161	Stewart	21	93	16	1.3125
162	Suarez	30	105	9	3.333333
163	Subram	0	85	0	1
164	Sun	15	72	4	3.75
165	Suri	2	91	1	2

Only 14 distinct values

For "Smithson" we don't know exactly how many rows, only that there are 23 rows other than Srini in this range

On average, we expect to get 1-2 rows for any value OTHER than Srini

Using Multiple Ranges



	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
155	Sharma	0	119	0	1
156	She	0	123	0	1
157	Shen	1	88	1	1
158	Simmons	15	109	11	1.363636
159	Smith	21	667	10	2.1
160	Srini	23	62	14	1.642857
161	Stewart	21	93	16	1.3125
162	Suarez	30	105	9	3.333333
163	Subram	0	85	0	1
164	Sun	15	72	4	3.75
165	Suri	2	91	1	2

For a wildcard like our 'S%' example, we can get a quick and close approximation of the total rows by summing range rows plus eq rows that start with S

If All Goes Well

With the necessary stats in place and appropriate updates, then we’ve got the information we need for SQL to make a pretty good guess on how many rows will match, and from there build a query plan that matches the expected load.

This happens most of the time.

But I bet you want to hear about how things can go awry!

And When Things Go Wrong

Typically stats related problems fall into a couple of categories:

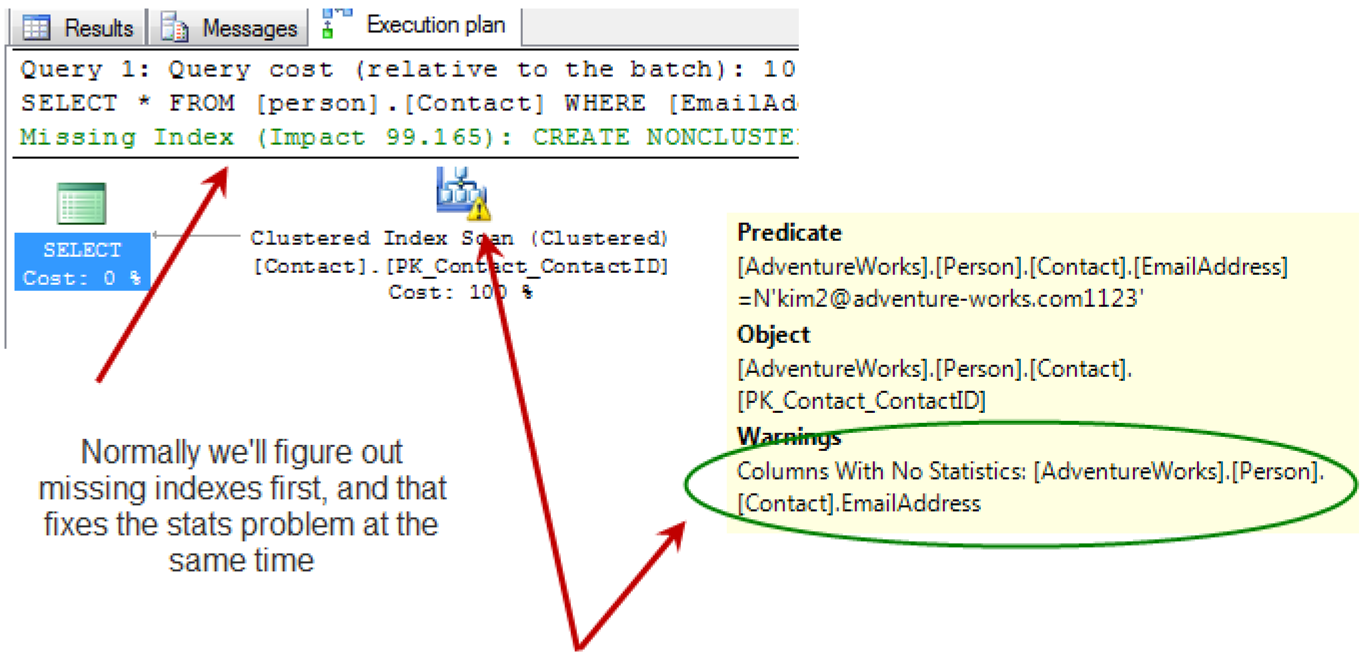
- No stats
- Out of date stats (let’s say “not updated lately”)

And one problem that can happen even with current stats:

- Uneven data distribution

No Stats = Guess = Bad!

If we have no stats for a column, we force the query optimizer to guess – not good



Results
Messages
Execution plan

```
Query 1: Query cost (relative to the batch): 10
SELECT * FROM [person].[Contact] WHERE [EmailAd
Missing Index (Impact 99.165): CREATE NONCLUSTE
```

SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[Contact].[PK_Contact_ContactID]
Cost: 100 %

Predicate
[AdventureWorks].[Person].[Contact].[EmailAddress]
=N'kim2@adventure-works.com1123'
Object
[AdventureWorks].[Person].[Contact].
[PK_Contact_ContactID]
Warnings
Columns With No Statistics: [AdventureWorks].[Person].
[Contact].EmailAddress

Normally we'll figure out missing indexes first, and that fixes the stats problem at the same time

But it is possible to have an index with no stats, or just a case where the optimizer wants stats it doesn't have - watch for the "!" on the plan operator and investigate when you see it

No Stats - Should Be Rare

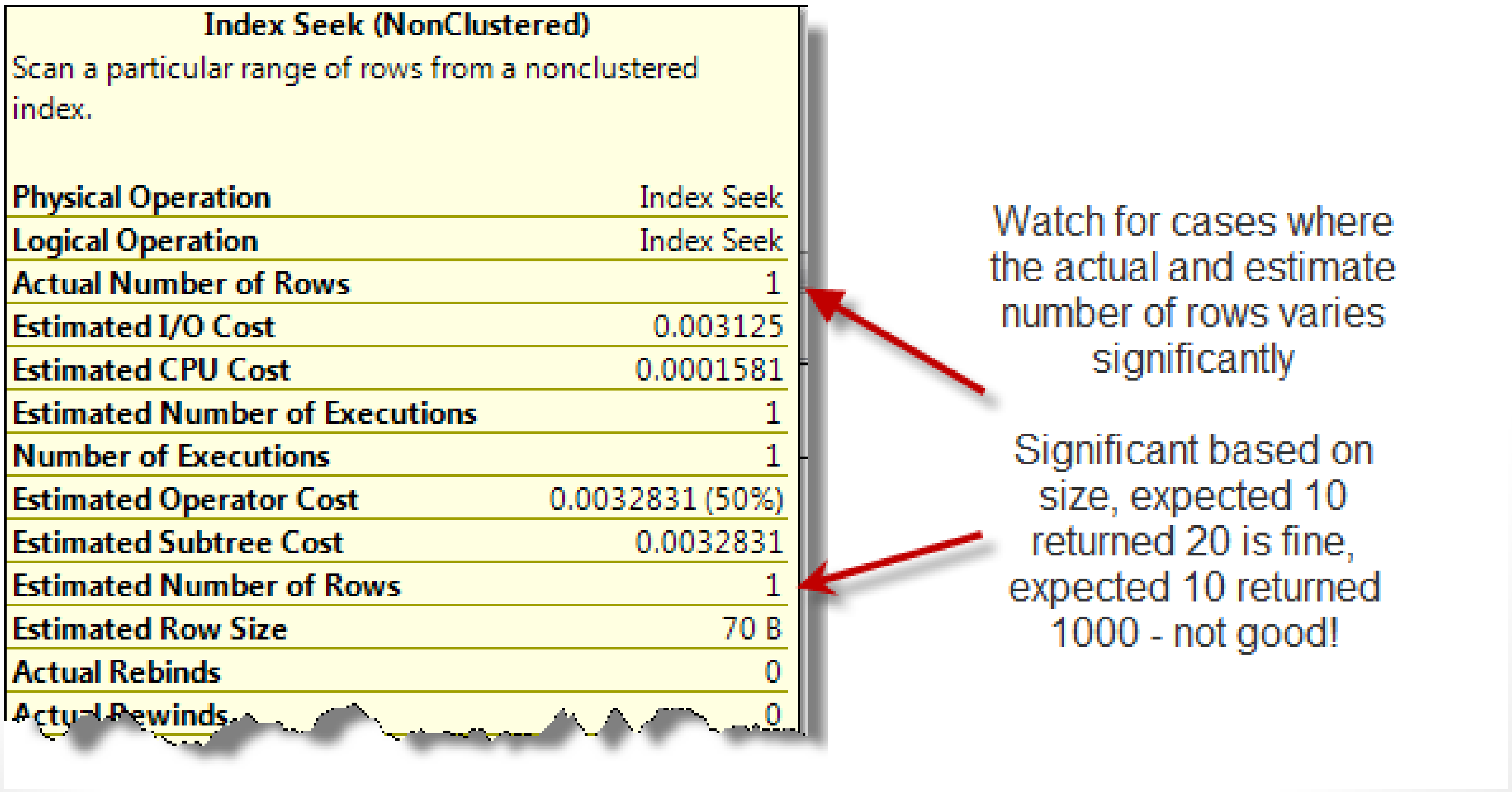
If you keep the default behaviors enabled you'll always have stats. Well, almost always. There are a few edge cases where things don't behave quite as expected:

- No stats on table variables
- No stats on table valued functions
- No stats on CLR columns unless binary ordering

Otherwise, if you find you're missing stats, get that fixed and then keep it fixed!

Good Stats Gone Bad

Having a significant mismatch in actual vs estimated often indicates stale stats



Index Seek (NonClustered)

Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Number of Rows	1
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Number of Executions	1
Estimated Operator Cost	0.0032831 (50%)
Estimated Subtree Cost	0.0032831
Estimated Number of Rows	1
Estimated Row Size	70 B
Actual Rebinds	0
Actual Rewinds	0

Watch for cases where the actual and estimate number of rows varies significantly

Significant based on size, expected 10 returned 20 is fine, expected 10 returned 1000 - not good!

Advanced Techniques

- DB Setting: Update Statistics Async prevents delays when a stats update is triggered by allowing the query to use the existing plan until the new stats are ready
- Query Hint: OPTION (KEEP PLAN) changes the threshold for recompile on temp tables to match that of permanent tables (rarely used)
- Query Hint: OPTION (KEEPFIXED PLAN) will prevent recompiles based on changes to stats (rarely used)

Best Practices

- Enable auto create, auto update
- Update stats as often as you rebuild indexes, or more so
- Update only column statistics if you've already rebuilt your indexes in the same session
- Watch for stats related issues by checking estimated vs. actual rows in the query plan

Resources

- 2005 Stats Whitepaper
- 2008 Stats Whitepaper
- Paul Randall on Auto Created Stats
- Kim Tripp on Filtered Stats
- Glenn Berry on Out of Date Stats
- Recompilation Whitepaper
- Kendal Van Dyke on Identifying Overlapping Stats

Thanks for Attending!

Please connect with me

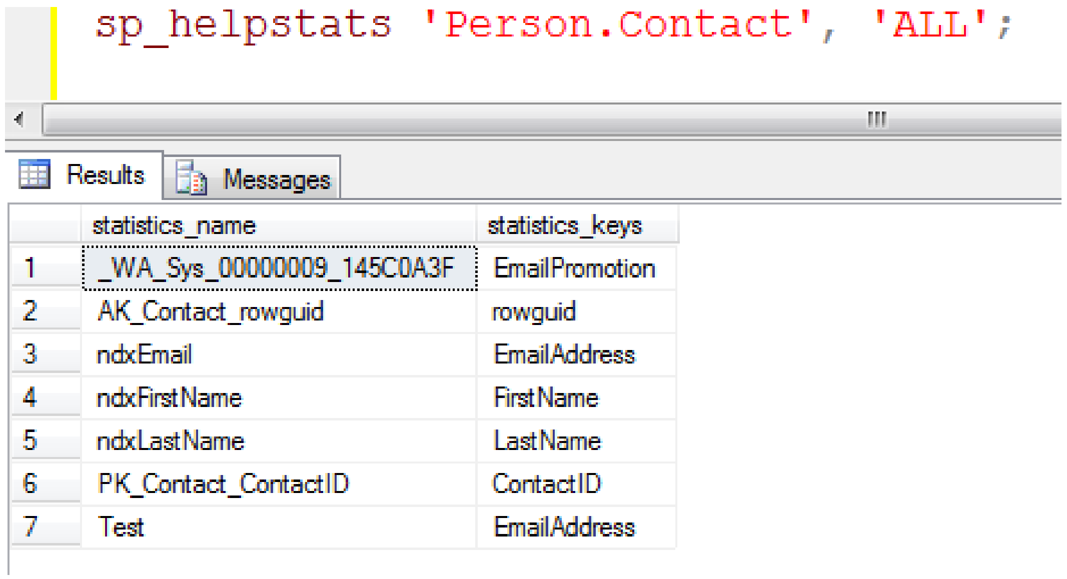
www.sqlandy.com

@sqlandy

www.linkedin.com/in/sqlandy

SP_HelpStats - Deprecated

Sp_helpstats is a quick way to return stats information about a table, but it has been **deprecated**. Instead, use the sys.stats and sys.stats_columns tables to get the same info



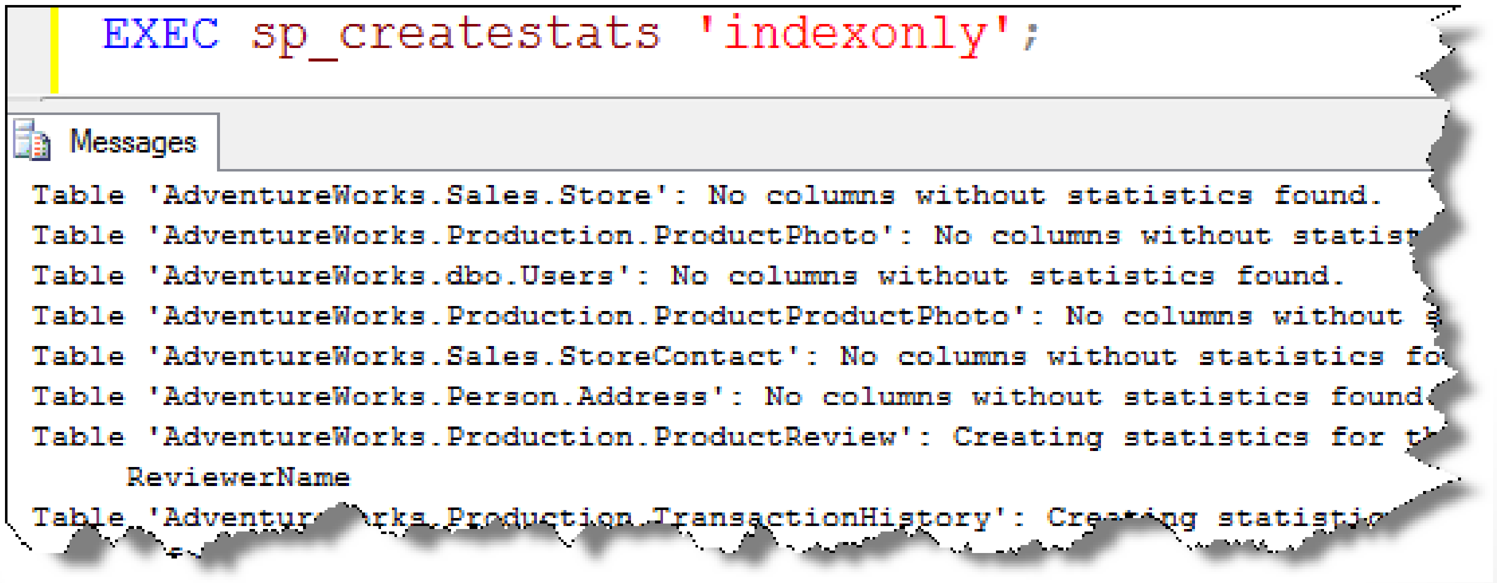
```
sp_helpstats 'Person.Contact', 'ALL';
```

Results | Messages

	statistics_name	statistics_keys
1	_WA_Sys_00000009_145C0A3F	EmailPromotion
2	AK_Contact_rowguid	rowguid
3	ndxEmail	EmailAddress
4	ndxFirstName	FirstName
5	ndxLastName	LastName
6	PK_Contact_ContactID	ContactID
7	Test	EmailAddress

SP_CreateStats

Creates single column stats for any column that isn't the leading column in an existing statistic.



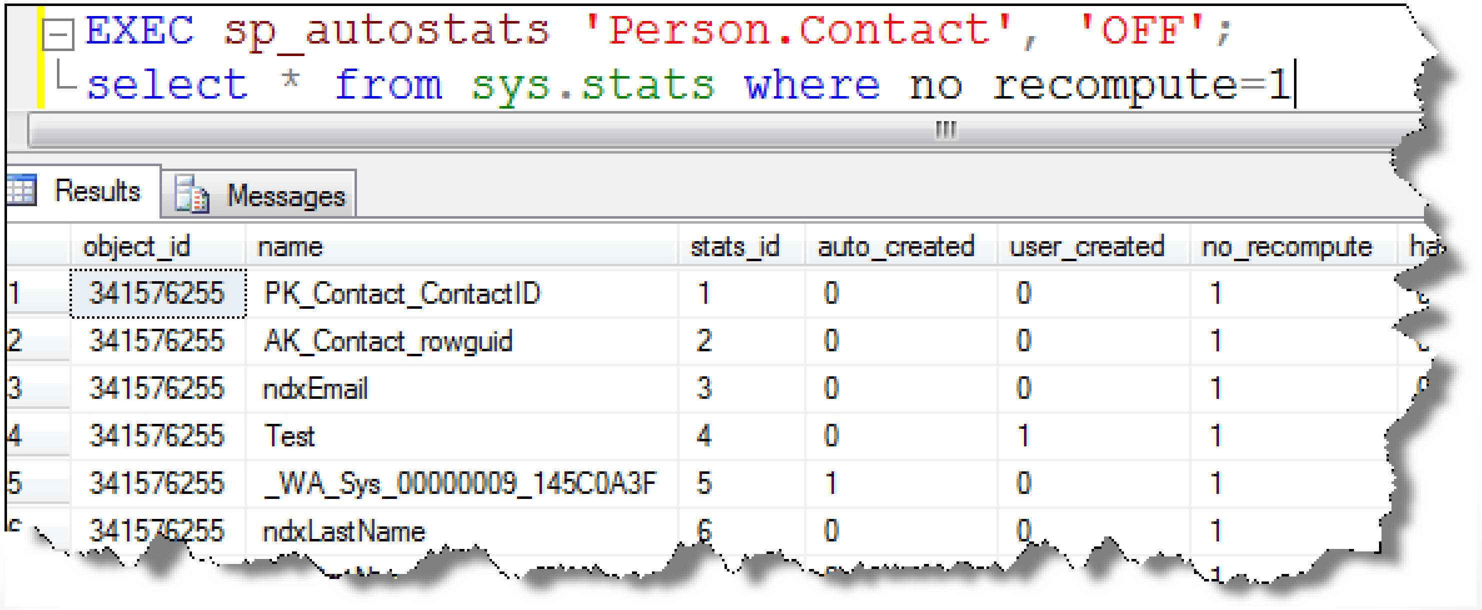
```
EXEC sp_createstats 'indexonly';
```

Messages

```
Table 'AdventureWorks.Sales.Store': No columns without statistics found.
Table 'AdventureWorks.Production.ProductPhoto': No columns without statist
Table 'AdventureWorks.dbo.Users': No columns without statistics found.
Table 'AdventureWorks.Production.ProductProductPhoto': No columns without s
Table 'AdventureWorks.Sales.StoreContact': No columns without statistics fo
Table 'AdventureWorks.Person.Address': No columns without statistics found
Table 'AdventureWorks.Production.ProductReview': Creating statistics for th
     ReviewerName
Table 'Adventur  rks.Production.TransactionHistory': Cre  ing statistic
```

SP_AutoStats

Used to change the NO_RECOMPUTE setting for all statistics on a table or index. The NO_RECOMPUTE flag is stored at the stat level in sys.stats.



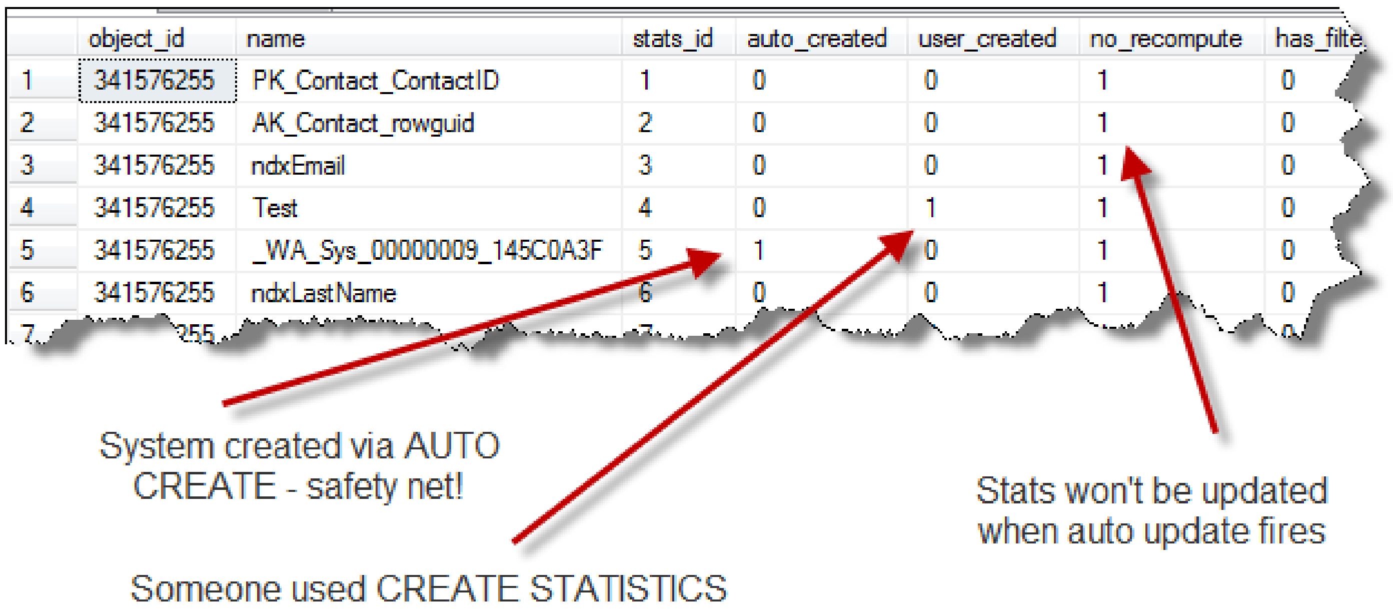
```
EXEC sp_autostats 'Person.Contact', 'OFF';
select * from sys.stats where no_recompute=1
```

Results | Messages

	object_id	name	stats_id	auto_created	user_created	no_recompute	ha
1	341576255	PK_Contact_ContactID	1	0	0	1	
2	341576255	AK_Contact_rowguid	2	0	0	1	
3	341576255	ndxEmail	3	0	0	1	
4	341576255	Test	4	0	1	1	
5	341576255	_WA_Sys_00000009_145C0A3F	5	1	0	1	
6	341576255	ndxLastName	6	0	0	1	

Sys.Stats

Sys.Stats and Sys.Stats_Columns let you see all the available statistics. For example, we can use this to see which stats have NO_RECOMPUTE enabled.



	object_id	name	stats_id	auto_created	user_created	no_recompute	has_filte
1	341576255	PK_Contact_ContactID	1	0	0	1	0
2	341576255	AK_Contact_rowguid	2	0	0	1	0
3	341576255	ndxEmail	3	0	0	1	0
4	341576255	Test	4	0	1	1	0
5	341576255	_WA_Sys_00000009_145C0A3F	5	1	0	1	0
6	341576255	ndxLastName	6	0	0	1	0
7	255		7				0

System created via AUTO CREATE - safety net!

Someone used CREATE STATISTICS

Stats won't be updated when auto update fires