Microsoft

SQL Server Statistics Structure

Module 5

Learning Units covered in this Module

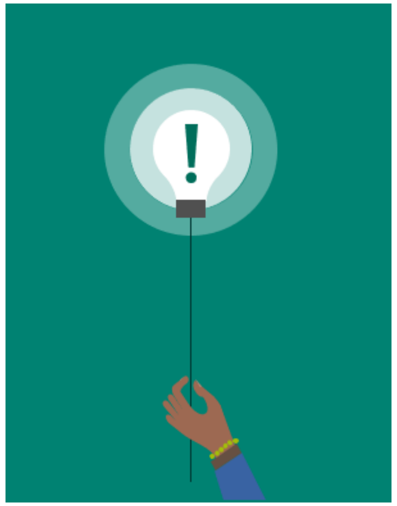
- Lesson 1: SQL Server Statistics Internals
- Lesson 2: SQL Server Statistics Maintenance
- Lesson 3: SQL Server Cardinality Estimation

Lesson 1: SQL Server Statistics Internals

Objectives

After completing this learning, you will be able to:

- Understand statistics, and to retrieve their contents.
- Review database options used to control statistics creation and update.
- Use different methods to read statistics information.



SQL Server statistics

What are the statistics?

Statistics contain statistical information about distribution of values in one or more columns of a table or index.

It is stored as binary large objects (BLOBs).

It is used by the Query Optimizer (QO) to estimate the *cardinality*, or number of rows, in the query result, and enable the creation of high-quality query plans.

Statistics are created:

- Intrinsically when indexes are created.
- Manually by using CREATE STATISTICS command.
- Automatically, to support WHERE clauses (if AUTO_CREATE_STATISTICS in ON).

Statistics components

Density Vector

Density is information about the number of duplicates in each column or combination of columns

•Used when query predicate contains variable
WHERE col = @variable
or when a stored procedure uses query on a modified parameter:
WHERE col = @local_variable

Histogram

A **histogram** measures the frequency of occurrence for each distinct value in a data set.

•Used when query predicate contains
WHERE col = 'literal'
or when a stored procedure uses a query on a parameter
WHERE col = @parameter

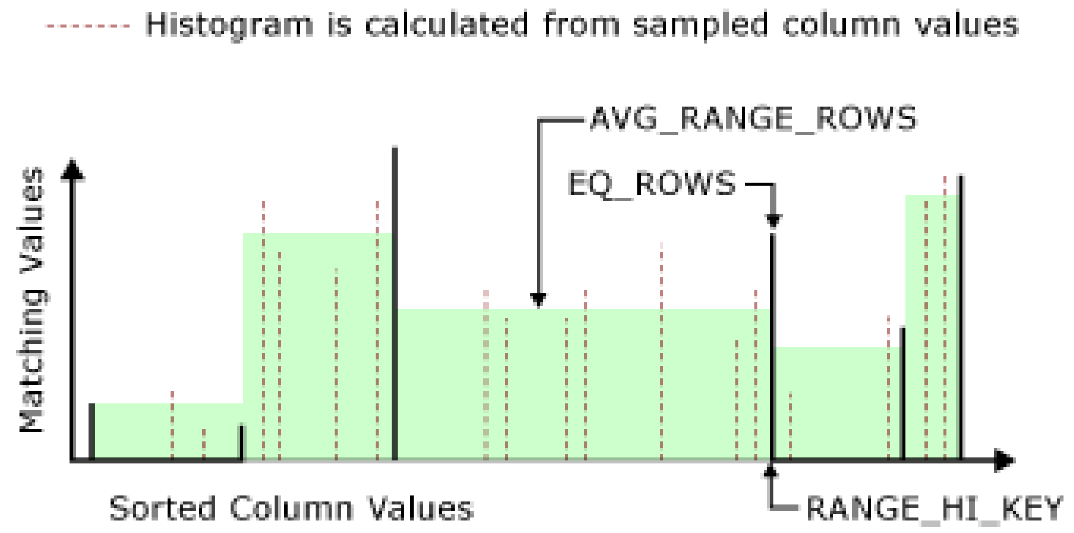
Statistics components

Histogram

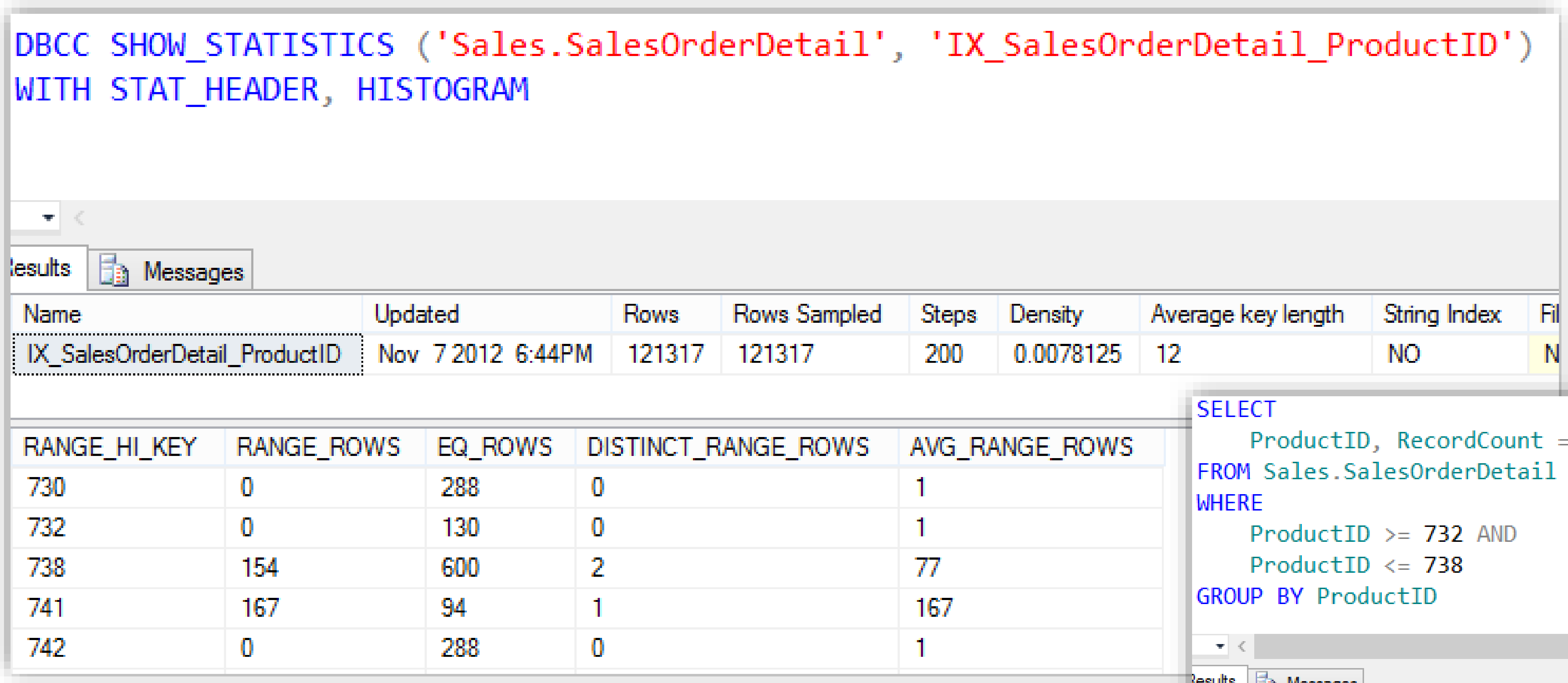
It is computed from the values in the leftmost key column of the statistics object.

Data is selected from table or view in one of two ways:

- Statistically sampling rows.
  Data shown contains estimates of rows and distinct values.
- Performing a full scan of all rows.
  Data shown contains actual numbers.



Showing Statistics

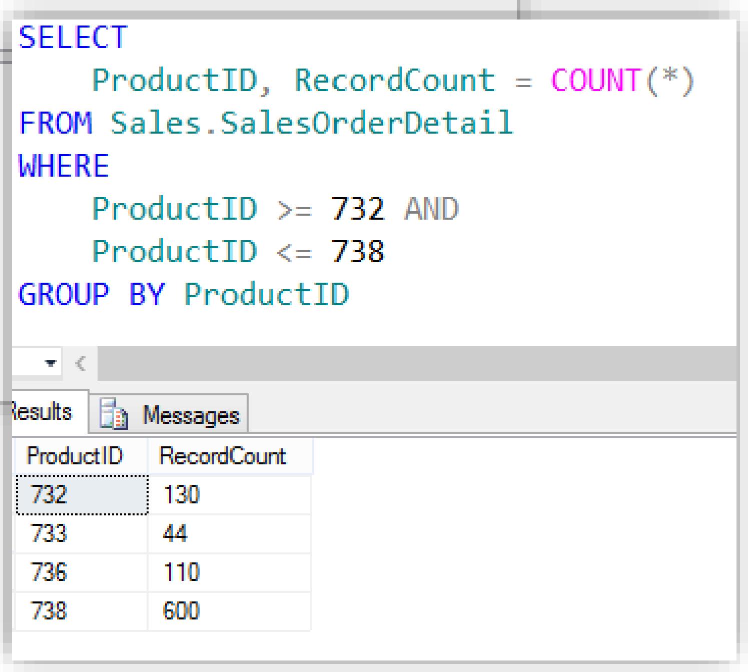


```
DBCC SHOW_STATISTICS ('Sales.SalesOrderDetail', 'IX_SalesOrderDetail_ProductID')
WITH STAT_HEADER, HISTOGRAM
```

Results | Messages

Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Fi
IX_SalesOrderDetail_ProductID	Nov 7 2012 6:44PM	121317	121317	200	0.0078125	12	NO	N

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
730	0	288	0	1
732	0	130	0	1
738	154	600	2	77
741	167	94	1	167
742	0	288	0	1

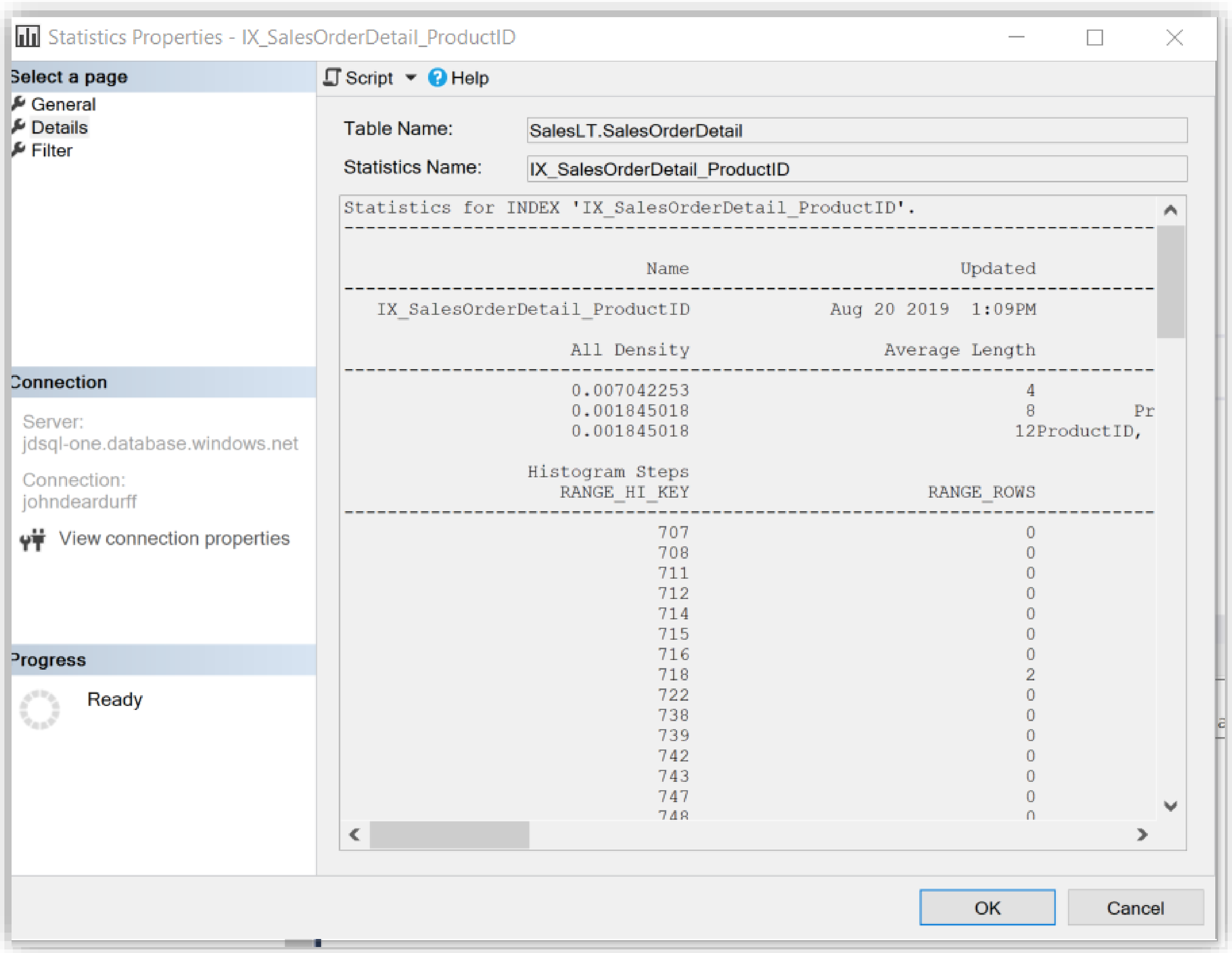


```
SELECT
    ProductID, RecordCount = COUNT(*)
FROM Sales.SalesOrderDetail
WHERE
    ProductID >= 732 AND
    ProductID <= 738
GROUP BY ProductID
```

Results | Messages

ProductID	RecordCount
732	130
733	44
736	110
738	600

Showing Statistics



Statistics Properties - IX_SalesOrderDetail_ProductID
Select a page
General
Details
Filter
Script
Help
Connection
Server:
jdsql-one.database.windows.net
Connection:
johndeardurff
View connection properties
Progress
Ready
OK
Cancel

Table Name: SalesLT.SalesOrderDetail

Statistics Name: IX_SalesOrderDetail_ProductID

```
Statistics for INDEX 'IX_SalesOrderDetail_ProductID'.
--------------------------------------------------------------------------


                        Name                           Updated
--------------------------------------------------------------------------
   IX_SalesOrderDetail_ProductID             Aug 20 2019   1:09PM

                  All Density                   Average Length
--------------------------------------------------------------------------
                  0.007042253                                4
                  0.001845018                                8          Pr
                  0.001845018                               12ProductID,

              Histogram Steps
                 RANGE_HI_KEY                        RANGE_ROWS
--------------------------------------------------------------------------
                          707                                0
                          708                                0
                          711                                0
                          712                                0
                          714                                0
                          715                                0
                          716                                0
                          718                                2
                          722                                0
                          738                                0
                          739                                0
                          742                                0
                          743                                0
                          747                                0
                          748                                0
```

Automatically created statistics

Database options that affects automatic statistics creation and update

AUTO_CREATE_STATISTICS

AUTO_UPDATE_STATISTICS

AUTO_UPDATE_STATISTICS_ASYNC

INCREMENTAL

Use the defaults unless you NEED to do otherwise.

Do not enable auto-create statistics on SharePoint content database.

The small sample rate of AUTO_UPDATE_STATISTICS can cause some workloads to choose sub-optimal execution plans.

Manually created Statistics

For most queries, the query optimizer generates necessary statistics for a high-quality query plan.

In a few cases, additional statistics is needed to improve query performance.

```
CREATE STATISTICS ContactPromotion1
ON Person.Person (BusinessEntityID, LastName, EmailPromotion)
```

Filtered Statistics

May help address statistics quality issues for large tables with uneven data distributions.

Update threshold on filtered statistics is based on overall table threshold and *not* the filter predicate.

Filtered Statistics will not be used when RECOMPILE hint is missing.

```sql
CREATE STATISTICS ContactPromotion1
ON Person.Person (BusinessEntityID, LastName, EmailPromotion)
WHERE EmailPromotion = 2
```

Incremental statistics

Introduced in SQL Server 2014.

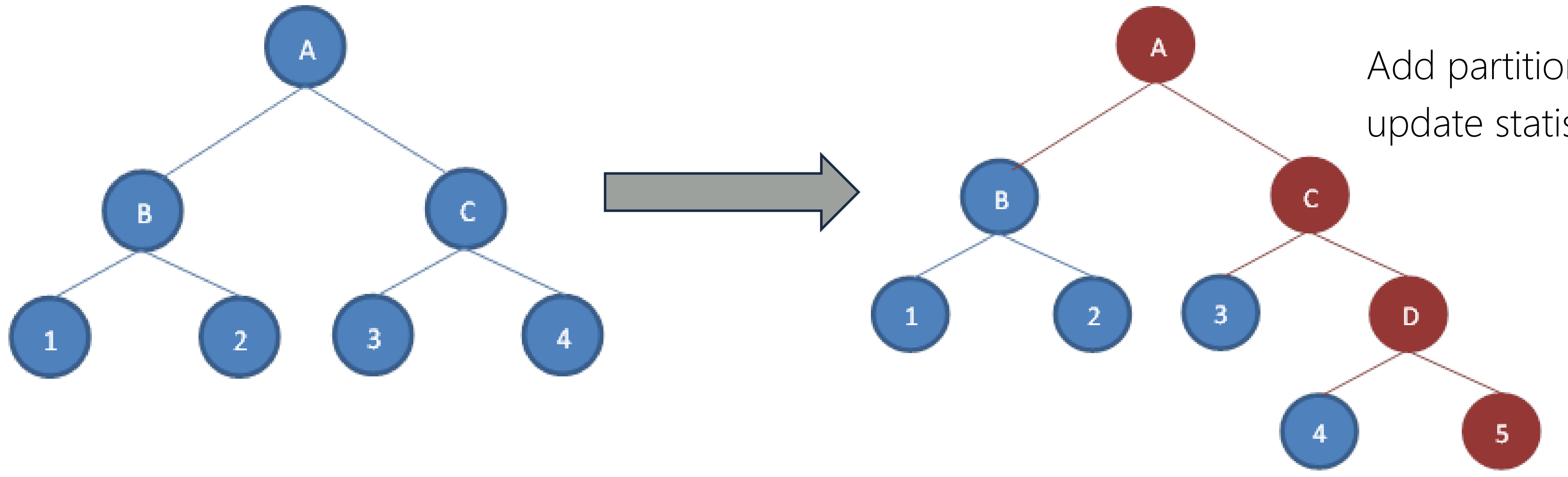
The incremental option creates and stores statistics on a per partition basis.

It allows to update statistics for a single partition. reducing maintenance times and eliminating the need to scan all partitions to get data to calculate statistics.

Partition level statistics are merged into a global statistic.

Per-Partition Statistics are not available for query optimization and the cardinality estimator still uses global table stats for query optimization.

Create Incremental Statistics on a four partition table



Add partition 5 and update statistics

Available tools to review statistics

	Metadata	Last Update Date	Sampling Rate	Row Mod Counter	Density Vector	Histogram
sys.stats	X					
STATS_DATE()		X				
sys.dm_db_stats_properties (object_id, stats_id)		X	X	X		
sys.dm_db_incremental_stats_properties (object_id, stats_id)		X	X	X		
sys.dm_db_stats_histogram (object_id, stats_id)						X
SQL Server Mgt Studio	X	X	X		X	X
DBCC SHOW_STATISTICS		X	X	X	X	X

Demonstration

Exploring statistics

Questions?

Knowledge Check

What is the density vector, and how does it differ from the histogram?

When would filtered statistics be a good option?

Explain incremental statistics?

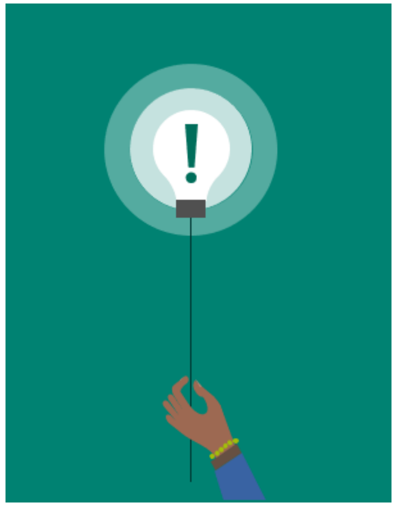
Is it possible to have an index with no statistics associated?

Lesson 2: SQL Server Statistics Maintenance

Objectives

After completing this learning, you will be able to:

- Describe the importance of updating statistics on a regular basis.
- Differentiate between manual and automatic update statistics methods.
- Understand when automatic statistics updates occur.



Why update statistics?

Up-to-date statistics are crucial for generating optimal query plans and to ensure excellent performance.

Data changes over time and a statistic created hours/days/weeks ago can not represent correctly data distribution.

Updating statistics ensures that queries compile with up-to-date statistics.

However, when updating statistics consider that:

- It causes queries to recompile.
- It can use TempDB to sort the sample of rows for building statistics.

Manual statistics update

UPDATE STATISTICS

sp_updatestats

Automatic statistics update

Using ALTER INDEX

- When an index is rebuilt, all data is read, and index statistics are updated with a sample of 100%.
- Statistics are not created or updated by scanning all the rows in the table for partitioned indexes. Instead, the default sampling algorithm is used.
- Reorganizing an index does not update statistics.

Automatic statistics update

AUTO_UPDATE_STATISTICS database option

AUTO_UPDATE_STATISTICS is ON by default on all new databases.

SQL Server fires statistics updates if a query is executed and it considers the statics might be out of date considering the tracked changes on column.

AUTO_UPDATE_STATISTICS_ASYNC is OFF	AUTO_UPDATE_STATISTICS_ASYNC is ON
The query executing is suspended until statistics are updated and a new plan is created. This can create performance issues as the process to update statistics can take a long time on big indexes.	The query is executed using existing statistics and statistics are updated as asynchronous operation. It can be useful in scenarios where synchronous statistics update take a long time.

Automatic statistics update

AUTO_UPDATE_STATISTICS threshold calculation

SQL Server 2014 or databases with compatibility level 120 or lower:

	Threshold calculation
Permanent Table	If n <= 500, RecompilationThreshold(RT) = 500 If n > 500, RT = 500 + 0.20 * n
Temp Table	If n < 6, RT = 6 If 6 <= n <= 500, RT = 500 If n > 500, RT = 500 + 0.20 * n
Table Variable	No statistics, unless Trace flag 2453 enabled Trace flag 2453 allows a table variable to trigger recompile when enough number of rows are changed
Incremental statistics	To update global stat: 500 + 20% of average partition size (non-empty) To update per-partition stat: 20% data change based on average partition size

The previous formula is a linear function, so the biggest the table, the lower the probability that statistics are automatically updated, leading to potential performance problems.

Automatic statistics threshold

In SQL Server 2016 and later the following formula is used:

RT = SQRT (1000 * #rows)

Rows	Old formula	New Formula
1,000	**700**	1,000
10,000	**2,500**	3,162
20,000	4,500	4,472
30,000	6,500	**5,477**
50,000	10,500	**7,071**
100,000	20,500	**10,000**

The new formula can be enabled in SQL 2008 R2 SP1 and later by using Trace Flag 2371.

TF23711 can be used in SQL Server 2016+ for databases with compatibility level 120 or lower.

Automatic statistics update

Disabling automatic updates (AUTO_UPDATE_STATISTICS) for specific indexes

To control when statistics are updated on a table basis.

Auto update statistics can be disabled in three ways:

- Using the option NORECOMPUTE in the CREATE STATISTICS and UPDATE STATISTICS commands.
- Using the option (STATISTICS_NORECOMPUTE = ON) in the CREATE INDEX and ALTER INDEX … REBUILD commands.
- Executing the stored procedure sp_autostats.

Automatic statistics updates can be reenabled, using the sp_autostats or by executing UPDATE STATISTICS without the NORECOMPUTE option.

Performance Monitor Counters

Execution Statistics

SQL Errors\Errors/sec

- Error types must be investigated and possibly resolved.

SQL Statistics\Batch Requests/sec

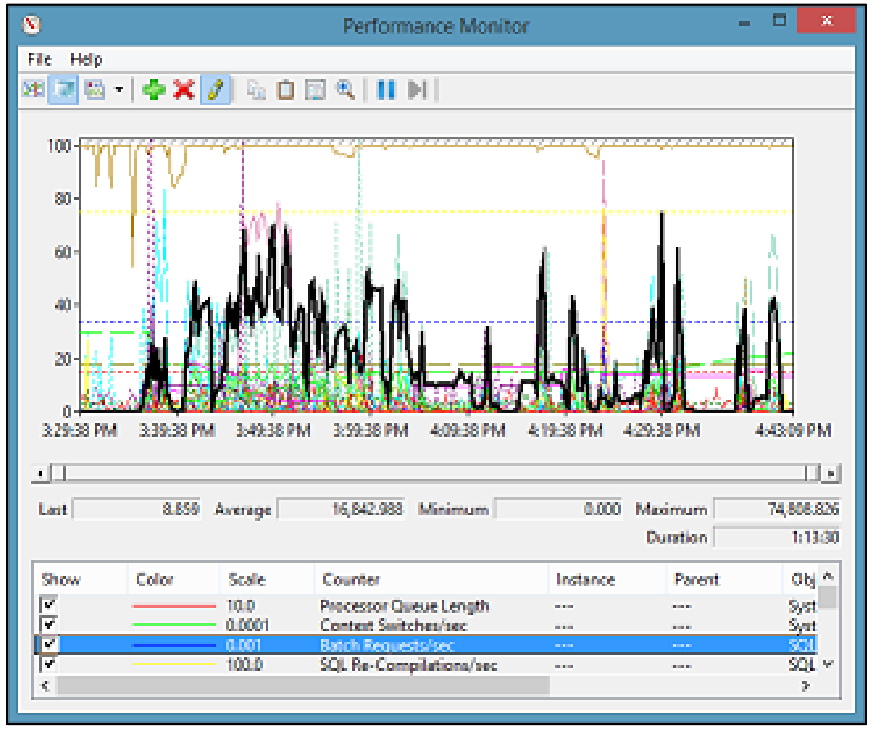
- Batch Requests > 1000 indicates busy server.

SQL Statistics\SQL Compilations/sec

- A high number can be an indicator of ad hoc queries, this must be cross referenced with ad hoc plans in the plan cache.

SQL Statistics\SQL Recompilations/sec

- If high determine recompilation reason with Xevent session. Usually stale statistics, Temp table usage and option WITH Recompile.



Last 8.859 Average 16,842.988 Minimum 0.000 Maximum 74,808.826
Duration 1:13:30

Show	Color	Scale	Counter	Instance	Parent	Obj
✓	🟥	10.0	Processor Queue Length	---	---	Syst
✓	🟩	0.00001	Context Switches/sec	---	---	Syst
✓	🟦	0.001	Batch Requests/sec	---	---	SQL
✓	🟨	100.0	SQL Re-Compilations/sec	---	---	SQL

Statistics Update

- Observing Automatic statistics update
- Updating Statistics by executing ALTER INDEX

LAB

Questions?

Knowledge Check

Does an INDEX REORG update its statistics?

What is the % of data sampled used to update statistics when ALTER INDEX … REBUILD is executed?

Is it possible to disable auto updates for a particular statistic?

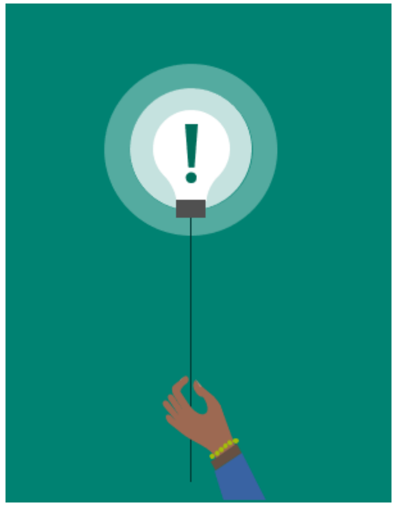
What is the formula for the auto update statistics threshold in SQL Server 2106+ (compatibility 130+)?

Lesson 3: SQL Server Cardinality Estimation

Objectives

After completing this learning, you will be able to:

- Understand what the Cardinality Estimator does.
- Describe the different versions of the Cardinality Estimator.
- Understand the impact it has on query plan creation.
- Decide which Cardinality Estimator version to use when compiling a query.



SQL Server Cardinality Estimation

Overview

The Query Optimizer uses a cost-based algorithm to select query plans having lowest estimated processing cost to execute.

The cost of the query plan is based on the total number of rows processed at each level of a query plan (cardinality of the plan) and the cost of the operators used in the query.

Improved cardinality leads to better estimated costs and, in turn, faster execution plans.

The Cardinality estimator uses information from statistics to calculate estimates.

SQL Server Cardinality Estimation

Challenges

Some constructs do not allow SQL Server to accurately calculate cardinalities, causing inaccurate cost calculations that may lead to suboptimal query plans. Avoid:

- Predicates that use comparison operators between different columns of the same table.
- Using built-in functions or a scalar-valued, user-defined function whose argument is not a constant value.
- Joining columns through arithmetic or string concatenation operators.
- Comparing variables whose values are not known when the query is compiled and optimized.
- Predicates that use operators and there are no statistics on the columns involved on either side of the operators.
- Predicates that use operators and the distribution of values in the statistics is not uniform, but the query seeks a highly selective value set.
- Predicates that uses the not equal to (!=) comparison operator or the NOT logical operator.

SQL Server Cardinality Estimator versions

Legacy CE

In 1998, a major update of the CE was part of SQL Server 7.0, for which the compatibility level was 70.

The CE version 70 (now referred as Legacy CE) is set on four basic assumptions:

Independence

Uniformity

Containment (Simple)

Inclusion

SQL Server Cardinality Estimator versions

CE updates in SQL Server 2014

The CE updates for levels 120 and above incorporate updated assumptions and algorithms that work well on modern data warehousing and on OLTP workloads.

Independence becomes **Correlation**

The combination of the different column values are not necessarily independent.

This may resemble more real-life data querying.

Simple Containment becomes **Base Containment**

Users might query for data that does not exist.

For example, for an equality join between two tables, base tables histograms are used to estimate the join selectivity, and then factor in the predicate's selectivity.

SQL Server Cardinality Estimator versions

CE 120 (and above) improvements: Correlation

The table Cars has 1000 rows: 200 Hondas and 50 Civics.

Executing a query with multiple predicates combined with “and”:

```sql
Select *
From Cars
Where Make='Honda' AND Model = 'Civic'
```

These are the estimates

Old CE (Independence)	New CE (Exponential Backoff)
0.05 * 0.2 * 1000 = 10	0.05 * sqrt(0.2) * 1000 = 22.36

New CE takes a more conservative approach, and it gets closer to reality.

SQL Server Cardinality Estimator versions

CE 120 (and above) improvements: Ascending Key problem

Statistics for table Sales were updated January 31st, 2020.

Executing a query that filter values after the date of the last stats update.

```sql
SELECT s.item, s.category, s.amount
FROM dbo.Sales AS s
WHERE Date = ' 20201004'
```

Result into these estimates:

CE 70

- Cardinality estimation is always 1 row.
- Trace Flag 2389 can help as leading statistics column is marked as ascending, then the histogram used to estimate cardinality will be adjusted at query compile time.

CE 120 (and above)

- Estimate cardinality by using average frequency.
- Trace flag 2389 does not apply.

Choosing the Cardinality Estimator version

Use the old CE version

Use Trace Flag 9481 as service parameter to use old CE version in all databases.

To use the old version by default on all queries on a database there are two options:

- set compatibility level to 110 or lower
- set compatibility level to 120 or higher and set the database scope configuration LEGACY_CARDINALITY_ESTIMATION to ON

To use the old version for a particular query when database compatibility level is 120 or higher and LEGACY_CARDINALITY_ESTIMATION is OFF, use hint 'FORCE_DEFAULT_CARDINALITY_ESTIMATION in SQL Server 2016 SP1 +

```
SELECT * FROM FactCurrencyRate
WHERE DateKey = 20101201
OPTION(USE HINT('FORCE_LEGACY_CARDINALITY_ESTIMATION'));
```

Choosing the Cardinality Estimator version

Use the new CE version

To use the new version on all queries by default, set compatibility level to 120 or higher and set the database scope configuration LEGACY_CARDINALITY_ESTIMATION to OFF (default).

If compatibility level is 120 and higher and the database scope configuration LEGACY_CARDINALITY_ESTIMATION is set to ON, or Trace Flag 9481 is used, use the hint 'FORCE_DEFAULT_CARDINALITY_ESTIMATION in SQL Server 2016 SP1 +

```
SELECT * FROM FactCurrencyRate
WHERE DateKey = 20101201
OPTION(USE HINT('FORCE_DEFAULT_CARDINALITY_ESTIMATION'));
```

Demonstration

Cardinality Estimation

- Query Optimizer estimated number of rows calculation
- Using Filtered Statistics to improve cardinality estimation

Questions?

Knowledge Check

Why is the Cardinality Estimator so important for query performance?

After SQL Server 2000, in which SQL Server version did the Cardinality Estimator suffer the greatest change?

Can SQL Server 2014 or higher use the older cardinality estimator?

What can be the reason to use cardinality estimator version 70 in a database running on SQL Server 2014 or higher?

Microsoft