SQL ServerCentral.com
SQL Provision NEW Provision databases efficiently and keep data safe TRY FREE
a community service from redgate
Try SQL Clone

Daily SQL Articles by email:
your@email.com
Sign up

Write For Us
Find out more

redgate Hub
Insights
Learning
Forums
Ideas
Discover the new Redgate Hub

Improve your SQL Server knowledge daily with more articles by email.
your@email.com
Sign up

Thank this author by sharing: G+ 0
Rate this | Join the discussion | Add to briefcase

Index Statistics in SQL Server: Stairway to SQL Server Indexes Level 14

By David Durant, 2012/03/16

The Series

This article is part of the Stairway Series: Stairway to SQL Server Indexes

Indexes are fundamental to database design, and tell the developer using the database a great deal about the intentions of the designer. Unfortunately indexes are too often added as an afterthought when performance issues appear. Here at last is a simple series of articles that should bring any database professional rapidly 'up to speed' with them

In Level 10, while explaining why an index needed a non-leaf portion as well as a leaf portion, we said: “SQL Server, however, has no intrinsic knowledge of English language last names, or of any other data.” Thus, SQL Server does not know whether the “Meyer, Helen” entries represent 50 percent, or .05 percent of the entries in the index.

But it needs to know something about your data’s distribution; because, as we have seen, the selectivity of the query is a major factor in deciding whether an index should be used or not. Therefore, for every index, SQL Server maintains a small amount of information about the data values contained in the index key columns. This information is called *index statistics*, or sometimes just *statistics*, and it is the subject of this Level.

Statistics can also be generated for columns that do not appear in an index. The structure of those statistics is identical to index statistics. Since this Stairway is about indexes, all discussion will refer to index statistics.

Index statistics are a bit like the engine in your car: It is nice to know how it works internally, but more important is scheduling regular preventive maintenance with someone who does know how an engine works. We will cover both aspects if index statistics in this Level; describing the internals of index statistics and explaining when and why your statistics will need maintenance.

Index statistics are divided into three parts: the header, density vector, and histogram. To view the complete statistics for an index, you can execute

Related Articles

FORUM index entry
2008/10/24
index entry

ARTICLE Index Statistics in SQL Server 7.0
2008/10/17
A look at Index Statistics in SQL Server 7.0 and how to maintain them.

ARTICLE SQL Server 6.5: Index Statistic
D
2
T
a

ARTICLE S
D
2
T
a

ARTICLE C
G
e
2
T
g
e

Fresh articles daily:
Get the SQL Server Central newsletter and get a new SQL Server article each day. Get Database Weekly for a roundup of all the biggest SQL news from around the web.
your@email.com
Sign up
No thanks

Tags
indexing
stairway series

Write For Us
Find out more

Copyright © 2002-2018 Redgate. All Rights Reserved. Privacy Policy. Terms of Use. Report Abuse.

```
DBCC SHOW_STATISTICS (tablename, indexname)
```

To view each part individually, execute:

```
DBCC SHOW_STATISTICS (tablename, indexname)
WITH STAT_HEADER
```

Or:

```
DBCC SHOW_STATISTICS (tablename, indexname)
WITH DENSITY_VECTOR
```

Or:

```
DBCC SHOW_STATISTICS (tablename, indexname)
WITH HISTOGRAM
```

But, before we can view statistics, we need a table and an index to use as an example. Our sample table will consist of three columns and two indexes, one index on the first two columns, and one on the third column, as shown in Listing 1.

```
USE AdventureWorks;
GO

IF EXISTS (SELECT *
             FROM sys.objects
            WHERE name = 'HistogramTest'
AND type = 'U')
BEGIN
    DROP TABLE dbo.HistogramTest
END
GO
CREATE TABLE dbo.HistogramTest
     (
       Col1 int not null
     , Col2 int not null
     , Col3 int not null
     )
GO

CREATE INDEX IX_HistogramTest
           ON dbo.HistogramTest
            ( Col1, Col2 )
CREATE INDEX IX_SingleValue
           ON dbo.HistogramTest
            ( Col3 )
GO
```

Listing 1: A Sample Table with Two Indexes

We also need some sample data; data with specific patterns to help illustrate index statistics. We generate 5050 rows of this sample data with the code shown in Listing 2.

```
SET NOCOUNT ON;
GO
DECLARE @maxLeftColumnValue int = 100;
DECLARE @leftColumnValue int = 0;
DECLARE @middleColumnValue int = 0;
DECLARE @rightColumnValue int = 0;
WHILE @leftColumnValue <
@maxLeftColumnValue
BEGIN
    SET @leftColumnValue += 1
    SET @middleColumnValue = 0
    WHILE @middleColumnValue <
@leftColumnValue
    BEGIN
```

```
            SET @middleColumnValue += 1
            SET @rightColumnValue += 1
            INSERT dbo.HistogramTest VALUES
( @leftColumnValue

                  , @middleColumnValue

, @rightColumnValue )
    END
END
GO
```

Listing 2: The Sample Data Generator

The resulting sample data will have the following characteristics:

- Each value that appears in *Col1* will appear in as many rows as the value. Thus, 1 will appear in one row, 2 will appear in two rows, 3 will appear in three rows, and so on.
- For any row, the combination of the *Col1* - *Col2* values is unique within the table. That is, *IX_HistogramTest* could have been specified as a unique index.
- The values appearing in *Col3* are also unique, starting at 1 and incrementing by 1 for each row thereafter. Thus, *IX_SingleValue* also could have been a unique index.

For demonstration purposes, we create the table but not the indexes; load the table; then create the indexes. This gives the best load performance and ensures that our index statistics are up-to-date.

Once we have loaded the table and created the indexes, we run the queryshown in Listing 3, to verify that our sample data has the characteristics mentioned above. The results, shown in Figure 1, verify that our data has the specified three characteristics.

And, because the results shown in Figure 1 are in *Col1* - *Col2* sequence, *Figure 1*, minus *Col3*, also represents a sample portion of the *IX_HistogramTest* index.

```
SELECT TOP 20  Col1, Col2, Col3
  FROM dbo.HistogramTest
  ORDER BY Col1, Col2;
```

Listing 3: Query the Sample Data

Col1	Col2	Col3
1	1	1
2	1	2
2	2	3
3	1	4
3	2	5
3	3	6
4	1	7
4	2	8
4	3	9
4	4	10
5	1	11
5	2	12
5	3	13
5	4	14
5	5	15
6	1	16
6	2	17
6	3	18
6	4	19
6	5	20

Figure 1: A subset of our sample data

One common term that you encounter when dealing with statistics is *density*. Density is a measure of uniqueness of values, and is directly related to *selectivity*, a term that we have used in earlier levels. For example, an index key density of 0.01 means that are 1 / 0.01 = 100 different values in the index key. To put it another way: each value occurs, on average, in 1 out of every 100 entries. Density can be measured on individual columns or on a composite of columns, such as *Col1 – Col2*.

If Figure 1 were the entire table, rather than a subset of the rows, the density of *Col1* would be 1 / 6 = 0.1667, while the density of *Col3* would be 1 / 20 = 0.05.

The smaller the density, the fewer the rows that will match an equality comparison, and the more selective a WHERE clause such as WHERE *Col3* = 17 will be.

When we execute:

```
DBCC SHOW_STATISTICS('dbo.HistogramTest',
'IX_HistogramTest')

        WITH STAT_HEADER
```

We receive the index statistics header shown in Figure 2.

Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows
IX_HistogramTest	Apr 18 2011 2:16PM	5050	5050	59	0.01694915	8	NO	NULL	5050

Zoom in | Open in new window

Figure 2: The Index Statistics Header

The header's fields tell us the following:

Name: The name of the index is *IX_HistogramTest*.

Updated: The statistics were last updated April 18th at 2:16pm. (In our case, they were initially generated at that

time, they have never been updated.)

Rows: The number of entries in the index is 5050. This number is always the number of entries in the index, not the number of rows in the table. See the *Unfiltered Rows* definition below.

Rows Sampled: The number of entries that were sampled to generate these statistics is 5050. (More on sampling later.)

Steps: The number of steps in the histogram portion of the statistics is 59. (More on *steps* later.)

Density: This value is not used by SQL Server 2008 and exists solely for backward compatability.

Average Key Length: The average length of the index key values is 8. In our case, all the keys are fixed width of size 8 bytes; consisting of two 4 byte integers, *Col1* and *Col2*.

String Index: These statistics do not include string summary statistics. (More on *string summary statistics* later.)

Filter Expression: This index was not created with a *FILTER* clause specified.

Unfiltered Rows: The number of rows in the underlying table is 5050. If this were a filtered index, this number could be greater than the *Rows* value.

When we execute:

```
DBCC SHOW_STATISTICS('dbo.HistogramTest',
'IX_HistogramTest')

    WITH DENSITY_VECTOR
```

We receive the index density vector shown in Figure 3.

All density	Average Length	Columns
0.01	4	Col1
0.0001980198	8	Col1, Col2

Zoom in | Open in new window

Figure 3: The Index Statistics Density Vector

The density vector values shown in Figure 3 tell us:

Col1 averages 4 bytes in size, and contains 1 / 0.01 = 100 distinct values; the number that we would expect, given the sample data generation code shown in Listing 2.

The composite of *Col1* - *Col2* is 8 bytes in size and contains 1 / 0.0001980198 = 5050 distinct values; also a number that we expect, for we said earlier that this value would be unique across the index.

When we execute:

```
DBCC SHOW_STATISTICS('dbo.HistogramTest',
'IX_HistogramTest')

    WITH HISTOGRAM
```

We receive an index histogram, whose first 25 steps, out of 59, are shown in Figure 4.

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	0	1	0	1
2	0	2	0	1
3	0	3	0	1
4	0	4	0	1
5	0	5	0	1
6	0	6	0	1
7	0	7	0	1
8	0	8	0	1
9	0	9	0	1
10	0	10	0	1
11	0	11	0	1
12	0	12	0	1
13	0	13	0	1
14	0	14	0	1
15	0	15	0	1
16	0	16	0	1
17	0	17	0	1
18	0	18	0	1
20	19	20	1	19
22	21	22	1	21
24	23	24	1	23
26	25	26	1	25
28	27	28	1	27
30	29	30	1	29
32	31	32	1	31

Zoom in | Open in new window

Figure 4: The first 25 steps of our Index Statistics Histogram

The histogram looks like a table with five columns. However, the correct term for each item is step, not row. All the values in a histogram are derived from values the left most column of the index key. In the case of IX_HistogramTest, this column is Col1. All other columns are ignored in the generation of a histogram.

Each step of a histogram spans, or represents, a consecutive set (a range) of the index's entries. A histogram will never have more than 200 steps, regardless of the size of the table. Different steps may span a different number of entries, perhaps with one step being derived from 20 entries and the next step representing the next 45 entries.

To help explain the individual values of a histogram, we define the following four terms here, and illustrate them in Figure 5:

> The left value of an index entry is the value contained in the entry's left most column. Thus, in our sample, this is the value of Col1.
>
> A step set is the set of consecutive index entries represented by a step. All the values in a step's five columns are derived from the left values contained in the step's step set.
>
> The upper subset of a step set is those entries whose left value is equal to the highest left value found in the step set.
>
> The lower subset of a step set is those entries whose left value is less than the highest left value found in the step set.

For example, the index step set shown in Figure 5 generated the Figure 4 step whose RANGE-HI-KEY value is 20.

```
Col1          Col2
----          ----
        :
        :                End of Previous
18            17            Step Set
18            18     ^^^
```

```
19          7                                |
19          8                                |
19          9        Sub                     |
19          10                               |
19          11                               |
19          12                               |
19          13       Set                     |
19          14                               |
19          15                               |
19          16                               |
19          17                               |
19          18                               |
19          19   ------------                |
20          1    ------------                |
20          2                                |
20          3
20          4                                |
20          5        Upper                   |
20          6                                |
20          7                                |
20          8                                |
20          9                                |
20          10       Sub                     |
20          11                               |
20          12                               |
20          13                               |
20          14                               |
20          15       Set                     |
20          16                               |
20          17                               |
20          18                               |
20          19                               |
20          20   ------------      -------------

21          1   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
21          2           Start of Next
       :                Step Set
       :
```

Figure 5: A Step Set within an Index

Because this, our first sample, has only 100 different left values, each step spans two or less left values. Later in this Level, when we increase the sample's row count by a multiple of 100, we will see steps that span a much large range of left values.

The values in the five columns of a histogram, such as the one shown in Figure 4, are calculated as follows:

Note: Since the first and third columns of the histogram are derived from the step's *upper subset*, we define them first. The remaining three columns, derived from the step's *lower subset*, are defined last.

RANGE_HI_KEY:

> The highest left value in the step's step set. The *RANGE_HI_KEY* value identifies the step; thus, two steps will never have the same *RANGE_HI_KEY* value.
>
> For the first step, *RANGE_HI_KEY* is the lowest left value in the index. Thus, the first step always has an empty *lower subset*. For the last step, *RANGE_HI_KEY* is the highest left value in the index.
>
> Steps are maintained, and displayed, in *RANGE_HI_KEY* sequence.