SQL Server Cardinality Estimation - Quick Notes, Part 1

Hi Friends,

SQL Server Cardinality Estimation is a complicated topic until you understand the internals of SQL Server Statistics. Here are some quick notes that will answer some common questions that you generally have:

How does SQL Server Query Optimizer estimate a default cardinality?

Let’s see and example (turn ON Actual Execution Plan or press Ctrl + M)

useAdventureWorks2012

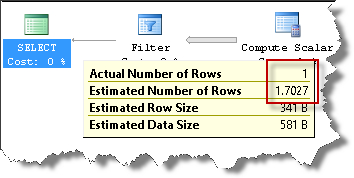
go

-- Turn On Actual Execution Plan

select\*fromsales.SalesOrderHeader

whereTotalDue=1457.3288

go



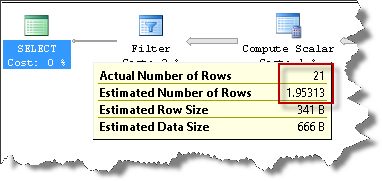
You can observe that optimizer estimated 1.7 rows (approx) vs 1 actual row returned.

Let us take another example:

select\*fromsales.SalesOrderHeader

whereTotalDue=472.3108

go



This time optimizer estimates 1.9 rows (approx) vs 21 actual row returned.

What’s going on? Answer is simple: SQL Server estimates the number of rows based on Statistics. Either statistics already exist on TotalDue column or SQL Server will automatically create one if AUTO\_CREATE\_STATS is turn ON. Either way, SQL Server uses this statistics object and more precisely, the histogram, to compute its estimation. You can read [this blog post](http://www.sqlmaestros.com/sql-server-statistics-histogram/) to learn more about histogram.

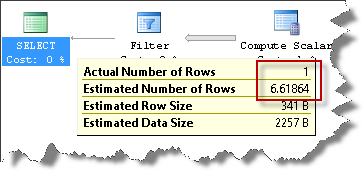
Now, let’s make some changes to our code:

declare@tdfloat

set@td=1457.3288

select\*fromsales.SalesOrderHeader

whereTotalDue=@td



Estimation: 6.61864 (observe that the same code, same predicate value, gave us an estimation of 1.7)

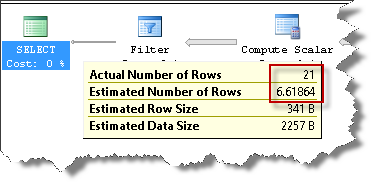
Let’s change the variable value and run the same code again:

declare@tdfloat

set@td=472.3108

select\*fromsales.SalesOrderHeader

whereTotalDue=@td



Estimation: 6.61864 (observe that the same code, same predicate value, gave us an estimation of 1.9)

Will the estimation be always 6.61864? What’s going on?

In the previous two examples, the optimizer was able to extract the value from the query to compute its estimation. In the next two example, since we are using local variables, the optimizer does not have the value until runtime. So will the estimation always be 6.61864 whenever we use local variable? Yes, if the table data does not change or more precisely, if the statistics behind the table does not change. When does statistics change, etc, etc is another blog post. But let’s come to the main question, where does 6.61864 come from? How does the optimizer compute this default value?

The formula for this default estimation is:

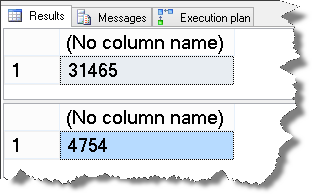
Total Rows \* Density (where Density is 1/number of distinct values)

SQL Server, apart from marinating the histogram also maintains density information for the columns.

So let us compute this:

selectcount(\*)fromsales.SalesOrderHeader

selectcount(distincttotaldue)fromsales.SalesOrderHeader



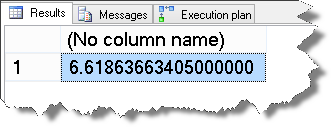
Density = 1/4754 = 2.103491796381994e-4

Total Rows \* Density = 31465 \* 2.103491796381994e-4 = 6.6186….

Or

selectcount(\*)\*cast((1.\*1/count(distincttotaldue))asdecimal(18,17))

fromsales.SalesOrderHeader



That’s how the default estimation was computed in this case. Hope the quick note was helpful.