ASE 15 and the Optimizer Statistics – More Influential Than Ever

By Eric Miner

EXECUTIVE SUMMARY

ASE 15’s new optimizer produces significant performance improvements if it receives the most accurate information about your data. The statistics contain the information the optimizer needs to do its job quickly and efficiently. This paper discusses how to create and maintain accurate statistics to reach the highest level of performance for ASE 15.

ASE 15 introduces completely new Optimizer and Query Processing engines. Years of hard, diligent engineering work went into developing these two vital pieces of ASE 15. Many fundamental changes were incorporated to insure that ASE 15 is the best performing dataserver out there. With these improvements comes the requirement that you give the optimizer as much accurate information about your data and databases as you possibly can. All that the optimizer needs to know is found in the ‘statistics’.

This paper will not go in depth into the technical side of the optimizer statistics describing each and every facet of the subject and muddying the waters for you. Instead it will give you the information about the statistics that you need in order to insure that you can reach and maintain the highest level of performance for ASE 15.

Before we continue let’s quickly review exactly what statistics are available to the optimizer. There are two types of statistics – column level and object level. The column level statistics describe the distribution of values in the column; they consist of the column’s histogram and density values and are updated when an index is created or an ‘update statistics’ command is run. The object level statistics describe a table and its indexes and include values such as number of rows and pages in the table and/or index(es), the number of empty pages, and the cluster ratios among others. Some of the object level statistics are updated automatically by ASE, others when ‘update statistics’ is run.

WHY CREATING AND MAINTAINING ACCURATE STATISTICS IS IMPORTANT TO ASE 15

The optimizer has always been dependant on the statistics because it is ‘cost based’. That is, it makes its decisions about which query plan to use based on the estimated resource costs of executing it. Without them it’s flying blind and can only guess at which is the most efficient query plan.

ASE 15 is no exception to this. In fact the statistics have become even more critical to good performance. Why is this? Because many of the optimizer’s new features and functionality can be very I/O intensive, especially if an inefficient query plan is used. Some of the new features and functionality include new methods for handling groupings, unions and all query level sorting operations.

New and improved join processing that is sensitive to the accuracy of the statistics has also been added. Hash joins are new to ASE and Sort-Merge joins have been improved and are turned on by default in ASE 15. If there are no useful indexes or if the statistics tell the optimizer that an index would not be efficient to use then a worktable has to be created for the join. Joining values are moved into the worktable where they are sorted into the order required by the join and then merged with rows from the other joining table. All this, especially the sorting of the values in the worktable requires a great deal of I/O. Since both of these join methods can include sorting steps it is imperative that efficient query plans are chosen by the optimizer. The bottom-line is that large, unnecessary sorts are the bane of good query performance. It can’t be over emphasized how important it is to keep accurate statistics available for the optimizer to take advantage of when estimating the costs of joins. Even though ASE 15 is designed to avoid worktables that were often used in earlier versions, inaccurate statistics can lead to query plans that revert to using them.
One new piece of functionality added to ASE 15 in order to deal directly with a long standing join performance issue is Join Histograms.

Join Histograms

The Problem – Prior to ASE 15 the optimizer could only use a column’s density value, if it was available, to estimate the cost of joining one column to another. The density value is a ‘weighted average’ of the distribution of values in the column. If there was no density value the optimizer used a preset selectivity value, a sort of ‘magic number’ that was based on the join operator. Since the most common join is an equi-join the ‘magic number’ used was 0.10 (the optimizer believed that 10% of the rows in the column qualified for the join). As you might imagine most joins don’t qualify exactly 10% of either column.

When the column contained a fairly even distribution of values the density value was accurate. However, when the column contained any degree of data skew (many values occupying a small number of rows each and a few values occupying many rows each) the density value was not accurate. When the optimizer used a density value of a skewed column it would lead the optimizer to believe that a join of the skewed table would be more efficient than it actually was. This in turn resulted in some very poorly performing joins.

The ASE 15 Solution – The join histograms of ASE 15 always give the optimizer an accurate view of the table. For example if the where clause of a join statement disqualifies a highly duplicated value it is not included in the join histogram; why estimate the cost of retrieving the skewed values if they aren’t needed for the join? How do join histograms work? Very simply – if there are statistics on a column that is being joined and there is a search argument (SARG) in the query then a histogram will be built on the fly containing only the values that the SARG has qualified.

A quick example: customerID table contains 100 distinct customer ids, orderID table contains all the orders placed by the 100 customers, 100K rows. Let’s say that of the 100 customers 3 have placed 50% of all the orders and of the 3 all have placed close to the same number of orders; the orderID table will contain data skew while the customerID table will be evenly distributed.

In pre-ASE 15 the density value would be larger due to the highly duplicated values for the three big customers. There is a good chance that the optimizer would estimate that an index on the custID column in the orders table would be expensive to use and call for a table scan. In fact the index would be very selective for the query. In ASE 15 the join histogram would not include the highly duplicated values thus accurately making the index access look cheaper than a table scan.

Of course for join histograms to be accurate and useful, accurate statistics need to be in place for the joining columns.

Here’s another situation that missing or inaccurate statistics can cause—The optimizer having to examine more query plans than is necessary.

Timeout

The Problem – In ASE 15 the new functionality in the optimizer can lead to the optimizer having MANY more query plans to examine than in earlier versions. Add to this the large and complex queries common today and the optimizer is working much harder than ever before. The more query plans to estimate costs for, the more time it takes to find the best plan to execute. In fact there is a possibility that the optimizer can take more time optimizing the query than it takes to actually execute it and return the results.

The ASE 15 Solution – It is for this reason that ‘timeout’ was included in the ASE 15 optimizer. Put simply, once the optimizer has reached the timeout limit it will choose the cheapest (most efficient) query plan it has examined up to that point. This plan will be used to execute the query whether it’s the best or not. Accurate statistics can result in the optimizer having to examine far fewer query plans and finding the most efficient plan before the timeout limit is reached. This in turn will result in less proc cache usage for the optimizer’s search. When there are no statistics or the statistics are inaccurate the optimizer can overestimate the cost of what is actually the best query plan and go on to examine many more.

HOW TO CREATE AND MAINTAIN ACCURATE STATISTICS

Statistics are kept up to date by regularly running the update statistics command, or by dropping and recreating indexes, which can be very time and resource consuming. In the past there were a number of ‘rules of thumb’ floating around among DBAs about when to update statistics. Some of the more popular were –

“when 5-10% of the data has changed”
“once a week”
“every day”
“only when queries start performing badly”
“only when you have time to do it”
“I never run update statistics”, etc.

The honest answer to the question was, is, and will always be—“It all depends on your data and your queries”.

With the release of ASE 11.9 many more options on how to update, read and even directly write the statistics are offered. You can update statistics as you always have, on the leading column of an index. Or, you can also put statistics on all the columns of an index or on columns that are not part of an index. You can even put statistics on all the columns of a table; all this by issuing a variety of ‘update statistics’ commands. You can specify how many ‘cells’ will be in each column’s histogram, thus making the statistics more granular…choices, choices everywhere. And, each choice has a potential effect on performance; either for good or bad.
As we’ve seen, the ‘why’ of updating statistics is obvious and clear-cut; to provide the optimizer with the most accurate picture of your data as possible. The ‘how’ of maintaining accurate statistics offers you the opportunity to apply a little creativity to the process. By this point in your work with ASE you’ve likely developed your own statistics maintenance practices based on your system and experience. In most cases there shouldn’t be a need to make many changes to this for ASE 15. One change that you should consider very seriously is putting and maintaining statistics on all columns of all your composite indexes, and on all columns that are referenced in queries (especially joins) that do not belong to any index.

Let’s take a look at new ways in ASE 15 to make the statistics more accurate.

**Histogram Tuning Configuration Value**

This new configuration value was included in order to address an issue that appeared with the release of 11.9 – “How many steps should I use to make highly duplicated values visible to the optimizer?” and/or “How do I make sure data skew won’t unexpectedly affect my performance?” In past versions of ASE a DBA had to run problem queries against histograms with various numbers of steps before finding an optimal number to use with. Then the DBA had to make sure that all upstate statistics commands used the optimal requested steps for each column. Take a look at the Performance and Tuning Guide for a detailed discussion of the histogram steps or cells.

For now it’s important to understand that when a column contains some degree of data skew the histogram will be more accurate if it contains more steps. The greater the number of steps the more likely it is that the highly duplicated values will be represented by their own individual step or ‘cell’. This in turn gives the optimizer very accurate information about the column and especially about the amount of duplication. Histogram Tuning Factor makes this complex tuning step automatic. No need to run test after test to find that ‘sweet spot’ number of steps.

**Automatic Update Statistics**

The ability for update statistics to be run automatically has been a long standing feature request. ASE 15 now gives you that ability. The feature is very straightforward and easy to use. To make life a little easier there are a few ways to automate update statistics. You can use Job Scheduler to run the job. You can set up automated update statistics as part of the ‘self-management installation’ and you create your own scripts for the automation. All three of these methods use the new datachange() function described in the next section. Automatic Update Statistics is simple and easy to set up and use. But, and there always seems to be one, keep in mind that having update statistics running on multiple table/indexes during the regular work day could have a nasty effect on overall performance.

See the ASE 15 Performance and Tuning Series. Query Processing and Abstract Plans’ document in the manuals area of the Sybase web site for instructions on how to set up each option for Automatic Update Statistics.

**WHEN TO ADD OR UPDATE STATISTICS**

If everything is humming along just fine and your overall performance is high then there’s no need for you to look for statistics to add or change. A well running ASE server is a good sign that your current statistics maintenance schedule is working well. However in ASE 15, if you do notice a negative change in performance you’ll need to start looking for the cause quickly. It’s recommended that the first thing to look at is the statistics.

**The New datachange() Function**

As your data changes the statistics need to be updated to stay accurate. Sometimes it’s useful to know exactly to what degree the data has changed. This is where the datachange() function comes in. It will give you the percentage of change that has occurred in a table or column. It’s up to you to decide exactly what percentage of change is necessary to trigger a run of update statistics. That specific percentage of change is then set as the threshold for the datachange() function. On the other hand you can use it to determine if you are unnecessarily updating statistics on a table and/or column that has seen little or no changes.

**Using set option show_missing_stats**

If your queries are joining on, or your SARGs are searching on, columns that are part of an index but not the leading column of an index you should run ‘update_index_statistics’. This will add statistics to all columns of the specified index or on all columns of all indexes on the specified table. By adding statistics to inner columns of composite indexes, the optimizer has a much better view of an index and thus is more likely to use it. Also if your queries are referencing columns that are not part of any index it is highly recommended that you put statistics on them with ‘update statistics’, or better still add them to an index.

In some cases it won’t be easy to determine which columns queries reference that may need statistics; this is when ‘set option show_missing_stats’ demonstrates its value.
Prior to ASE 15 there were few methods that could be used to find columns that were missing statistics. The most common was to pour through optimizer related trace outputs looking for missing statistics messages. This could become very time consuming to say the least. Another time intensive method is to go through all your queries and all your table schemas matching queries to columns. Of course you could always use a shotgun approach by running `update index statistics` on all your tables. This will put statistics on all columns in all indexes, but it will miss columns that are referenced by queries but not a part of an index. In that case, you could use the ‘nuclear option’ by running `update all statistics` on all tables. Keep in mind that if you use this option it is likely to take days to complete and will be a major nightmare to maintain all the unnecessary statistics you’ll create. But in ASE 15 you can perform this chore quickly, easily and in a more focused way by using the new set option “show_missing_stats”. Here’s how it works – Identify a query that is causing a performance problem and get its text. Then in ISQL or a script:

```
set option show_missing_stats on
go
dbcc traceon (3604)
go
set noexec on
go
your problem query here
go

NO STATS on column tableA.column1
NO STATS on column tableA.column4
NO STATS on column tableB.column2
NO STATS on column tableB.column5
```

(This will be followed by your result set)
You can then easily compare this output to the table’s schema to determine if you need to run `update index statistics` and/or `update statistics column_name` to put statistics on the columns where they’re missing.

CONCLUSION
ASE 15’s optimizer incorporates state of the art technology. A great deal of highly skilled and dedicated work went into the huge project of redesigning and rewriting it. It will produce amazing performance levels as long as it gets the most accurate information about your data as possible. The statistics contain everything the optimizer needs to do its job quickly and efficiently. It’s up to you to make sure that the statistics fully represent your data. It’s a job you’ve been doing for years, but in ASE 15 it should have a high priority.

This short paper is designed to give you a basic overview of how important the statistics are to the performance of ASE 15. Check both the Sybase and International Sybase User Group’s (ISUG) websites for much more in-depth information on numerous aspects of ASE 15.

---

Eric Miner is an independent consultant who has worked on database management systems for over 15 years. He was with Sybase from 1992 until 2004 working in the ASE Support and Engineering organizations. While in Engineering he was a member of the ASE Optimizer team. Eric served as the ISUG (International Sybase User Group) ASE Enhancements Director and received the organization’s ‘Outstanding Achievement’ award in 2002. He has written several articles for the ISUG Technical Journal, co-authored the book “ASE 12.5 Performance and Tuning”; delivered sessions at TechWave conferences and has participated in a number of ISUG international tours and user education sessions. Eric is a longtime amateur photographer and has won awards in international competitions. He lives in Charlotte, North Carolina with his wife Laurie and son Liam.