How do I compare statistics?

By Maria Colgan-Oracle on [May 18, 2012](https://blogs.oracle.com/optimizer/entry/how_do_i_compare_statistics)

This question came up recently when I was helping a customer migrate a large data warehouse to Oracle Database 11g. Prior to the upgrade, they were using an ESTIMATE\_PERCENT of 0.000001, the smallest possible sample size allowed, when they gathering statistics on their larger tables. When I asked why they picked such a tiny sample size they said it was because they needed statistics to be gathered extremely quickly after their daily load both at the partition and at the global level.

Since these large tables were partitioned, I thought they would be an excellent candidate for incremental statistics. However, in order to use incremental statistics gathering you have to let the ESTIMATE\_PERCENT parameter default to [AUTO\_SAMPLE\_SIZE](https://blogs.oracle.com/optimizer/entry/improvement_of_auto_sampling_statistics_gathering_feature_in_oracle_database_11g). Although the customer saw the benefit of using incremental statistics they were not keen on changing the value of ESTIMATE\_PERCENT. So I argued that with the new AUTO\_SAMPLE\_SIZE in Oracle Database 11g they would get statistics that were equivalent to a 100% sample but with the speed of a 10% sample.  And since we would be using incremental statistics we would only have to gather statistics on the freshly loaded partition and the global statistics (table level statistics) would be automatically aggregated correctly.

So with much skepticism, they tried incremental statistics in their test system and they were pleasantly surprised at the elapse time. However, they didn’t trust the statistics that were gathered and asked me, “how do I compare the statistics I got with AUTO\_SAMPLE\_SIZE  to the statistics I normally get with an ESTIMATE\_PERCENT of 0.000001?”

The answer to that was easy, ‘use DBMS\_STAT.DIFF\_TABLE\_STATS’. Although the answer was easy, it wasn’t an easy process to help them to work out how to use the DBMS\_STAT.DIFF\_STATS functions correctly. In this post I hope to share some of the gotchas you many encounter using DIFF\_STATS that are not so obvious from the description of the functions in our documentation or our whitepaper on ['Upgrading to Oracle Database 10g: What to Expect From The Optimizer'](http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/twp-bidw-optimizer-10gr2-0208-130973.pdf).  Let’s take the SALES table from the SH schema as an example. The SALES table has 28 partitions and 918843 rows.

### 13.2.1 Enabling and Disabling Automatic Optimizer Statistics Collection

The automated maintenance tasks infrastructure (known as **AutoTask**) schedules tasks to run automatically in Oracle Scheduler windows known as **maintenance windows**. By default, one window is scheduled for each day of the week. Automatic optimizer statistics collection runs as part of AutoTask and is enabled by default to run in all predefined maintenance windows.

If for some reason automatic optimizer statistics collection is disabled, then you can enable it using the ENABLE procedure in the DBMS\_AUTO\_TASK\_ADMIN package:

BEGIN

DBMS\_AUTO\_TASK\_ADMIN.ENABLE(

client\_name => 'auto optimizer stats collection',

operation => NULL,

window\_name => NULL);

END;

/

When you want to disable automatic optimizer statistics collection, you can disable it using the DISABLE procedure in the DBMS\_AUTO\_TASK\_ADMIN package:

BEGIN

DBMS\_AUTO\_TASK\_ADMIN.DISABLE(

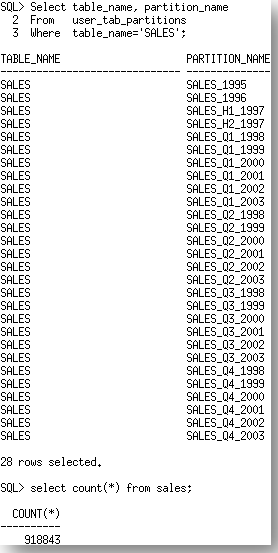
client\_name => 'auto optimizer stats collection',

operation => NULL,

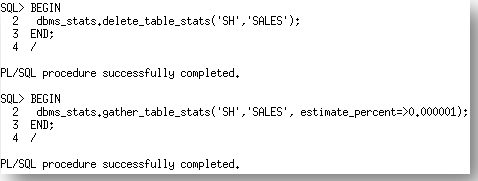
window\_name => NULL);

END;

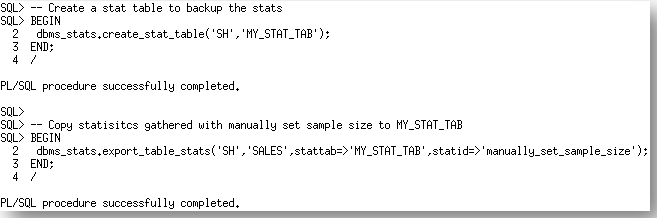
/



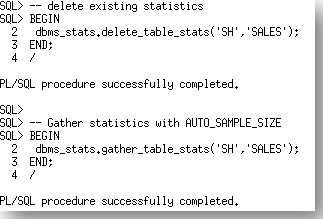
First we will gather statistics on the SALES table with their original setting for ESTIMATE\_PRECENT, 0.00001.



These statistics can now be backed up into a newly created stats table.



Now that we have an export of the statistics from the manually specified ESTIMATE\_PERCENT run, let’s re-gather statistics on the SALES table using the default, AUTO\_SAMPLE\_SIZE.



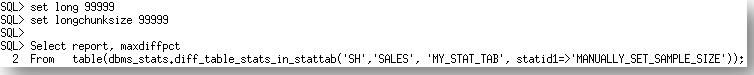
So, we are now ready to compare the two sets of the statistics for the SALES table using theDBMS\_STAT.DIFF\_TABLE\_STATS function. There are actually three version of this function depending on where the statistics being compared are located;

* DBMS\_STAT.DIFF\_TABLE\_STATS\_IN\_HISTORY
* DBMS\_STAT.DIFF\_TABLE\_STATS\_IN\_PENDING
* DBMS\_STAT.DIFF\_TABLE\_STATS\_IN\_STATTAB

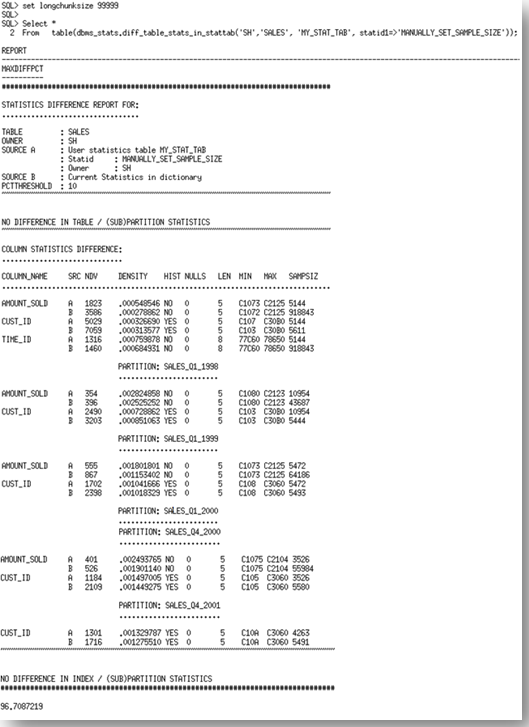
 In this case we will be using the DBMS\_STAT.DIFF\_TABLE\_STATS\_IN\_STATTAB  function. The functions also compare the statistics of the dependent objects (indexes, columns, partitions) and will only displays statistics for the object(s) from both sources if the difference between the statistics exceeds a certain threshold (%). The threshold can be specified as an argument to the function; the default value is 10%. The statistics corresponding to the first source will be used as the basis for computing the differential percentage.

The DBMS\_STAT.DIFF\_TABLE\_STATS functions are table functions so you must use the key word table when you are selecting from them, otherwise you will receive an error saying the object does not exist.

https://blogs.oracle.com/optimizer/resource/error_no_table_word.png  
The table function returns a report (clob datatype) and maxdiffpct (number). In order to display the report correctly you must use the set long command to define the width of a long so the report can be displayed properly.

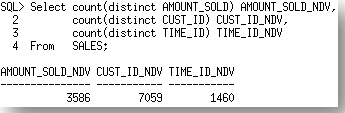


How that we know how to generate the report, let’s look at what it says,



The report has three sections. It begins with a comparison of the basic table statistics. In this case the table statistics (number of rows, number of blocks etc) are the same. The results so far are to be expected since we can accurately extrapolate the table statistics from a very small sample. The second section of the report examines column statistics.

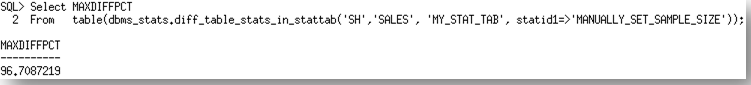
Each of the columns where the statistics vary is listed (AMOUNT\_SOLD, CUST\_ID, TIME\_ID) along with a copy of the statistics values from each source. Source A is the STATTAB, which in our case is the ESTIMATE\_PERCENTof 0.000001. Source B is the statistics currently in the dictionary, which in our case is the AUTO\_SAMPLE\_SIZE set. You will notice quite a significant difference in the statistics, especially in the NDV (number of distinct values) and the minimum and maximum values for each of the columns. If we compare these results with the actual number of distinct values for these (below), we see that the statistics reported by source B, the AUTO\_SAMPLE\_SIZE are the most accurate.



The report then goes on to list the column statistics differences for each of the partitions. In this section you will see that the problems occur only in the AMOUNT\_SOLD, CUST\_ID columns. The SALES table is range partitioned onTIME\_ID, so there is a limited number of TIME\_IDs in each partition, thus the percentage difference between the results on this level is not enough to meet the threshold of 10%.

Finally the report looks at the index statistics.In this case the index statistics were different but they were not greater than 10% different so the report doesn't show them.

Along with the report the function returns the MAXDIFFPCT. This is the maximum percentage difference between the statistics. These differences can come from the table, column, or index statistics. In this case the MAXDIFFPCT was 96%!



So the statistics were significantly different with [AUTO\_SAMPLE\_SIZE](https://blogs.oracle.com/optimizer/entry/improvement_of_auto_sampling_statistics_gathering_feature_in_oracle_database_11g) but the proof is in the pudding. We put the new statistics to the test and found that the Optimizer chose much better execution plan with the new statistics. So much so they were able to remove a ton of hints from their code that had been necessary previously due to poor statistics. You can't ask for anything better than that!