**MONDAY, OCTOBER 19, 2009**

**How to use histogram in Oracle**

I would like to write about Oracle Histogram today. Histogram is very nice feature to help cost based optimizer to make right decision.  
  
What is Histogram? Histograms are feature in CBO and it helps to optimizer to determine how data are skewed(distributed) with in the column. Histogram is good to create for the column which are included in the WHERE clause where the column is highly skewed. Histogram helps to optimizer to decide whether to use an index or full-table scan or help the optimizer determine the fastest table join order.  
  
What are the advantage of Histogram? Histograms are useful in two places.   
  
1. Histograms are useful for Oracle optimizer to choose the right access method in a table.  
  
2. It is also useful for optimizer to decide the correct table join order. When we join multiple tables, histogram helps to minimize the intermediate result set. Since the smaller size of the intermediate result set will improve the performance.  
  
Type of Histograms: Oracle uses two types of histograms for column statistics: height-balanced histograms and frequency histograms.   
  
1. Height - balanced Histograms : The column values are divided into bands so that each band contains approximately the same number of rows. For instances, we have 10 distinct values in the column and only five buckets. It will create height based(Height balanced) histograms and it will evenly spread values through the buckets. A height-based histogram is when there are more distinct values than the number of buckets and the histogram statistics shows a range of rows across the buckets   
  
2. Frequency Histograms : Each value of the column corresponds to a single bucket of the histogram. This is also called value based histogram. Each bucket contains the number of occurrences of that single value. Frequency histograms are automatically created instead of height-balanced histograms when the number of distinct values is less than or equal to the number of histogram buckets specified.   
  
Method\_opt Parameter: This is the parameter which tells about creating histogram while collecting the statistics. The default is FOR ALL COLUMNS SIZE AUTO in Oracle10g. But in oracle9i, the default is FOR ALL COLUMN SIZE 1 which will turn off the histogram collection.   
  
FOR ALL [INDEXED HIDDEN] COLUMNS [size\_clause]  
FOR COLUMNS [size clause] columnattribute [size\_clause] [,columnattribute [size\_clause]...]  
  
size\_clause is defined as size\_clause := SIZE {integer REPEAT AUTO SKEWONLY}  
  
integer : Number of histogram buckets. Must be in the range [1,254]  
  
REPEAT : Collects histograms only on the columns that already have histograms.  
  
AUTO : Oracle determines the columns to collect histograms based on data distribution and the workload of the columns. We have a table called sys.col\_usage$ that stores information about column usage. dbms\_stats use this information to determine whether histogram is required for the columns.  
  
SKEWONLY : Oracle determines the columns to collect histograms based on the data distribution of the columns.  
  
Let me demonstrate how optimizer works with and without histogram as below two scenario. We take the emp table for this demonstration. The table has around 3.6 million records. The table emp\_status column is highly skewed. It has two distinct values(Y,N). We have bitmap index on emp\_status column.  
  
Scenario 1 Let us generate the statistics without any histogram and see what kind of execution path optimizer is using. Without the histogram, oracle assume that, the data is evenly distributed and optimizer think that, we will have around 1.8 million record for emp\_status Y and around another 1.8 million records for emp\_status N.  
  
Connected to:  
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production  
With the Partitioning, OLAP and Data Mining options  
  
SQL> select count(\*),emp\_status from scott.emp  
2 group by emp\_status;  
  
COUNT(\*) E  
---------- -  
1 N  
3670016 Y  
  
SQL> execute DBMS\_STATS.GATHER\_TABLE\_STATS(OWNNAME => 'SCOTT', TABNAME => 'EMP',ESTIMATE\_PERCENT =>  
10, METHOD\_OPT => 'FOR ALL COLUMNS SIZE 1',CASCADE => TRUE);  
  
PL/SQL procedure successfully completed.  
  
SQL> select ename from scott.emp where emp\_status='Y';  
  
3670016 rows selected.

--------------------------------------------------------------------------  
Id Operation Name Rows Bytes Cost (%CPU) Time  
--------------------------------------------------------------------------  
0 SELECT STATEMENT 1832K 15M 5374 (5) 00:01:05  
\* 1 TABLE ACCESS FULL EMP 1832K 15M 5374 (5) 00:01:05  
--------------------------------------------------------------------------

SQL> select ename from scott.emp where emp\_status='N';

--------------------------------------------------------------------------  
Id Operation Name Rows Bytes Cost (%CPU) Time  
--------------------------------------------------------------------------  
0 SELECT STATEMENT 1832K 15M 5374 (5) 00:01:05  
\* 1 TABLE ACCESS FULL EMP 1832K 15M 5374 (5) 00:01:05  
--------------------------------------------------------------------------

Conclusion: Optimizer is using full table scan for the query which returns 3670016 records as well as it using full table scan for query which returns just only one record. This is obvisouly incorrect. This problem will be resolved by collecting histogram. Let us see in the next scenario.

Scenario 2 : Let us generate the statistics with histogram and see what kind of execution path optimizer is using. FOR COLUMN SIZE 2 EMP\_STATUS will create two bucket for column emp\_status. If we are not sure the distinct number of values in the column, then we can use AUTO option to collect histogram. With this histogram, oracle optimizer knows that, the column emp\_status is highly skewed and it has two bucket and one bucket has around 3.6 million records with emp\_status Y and another bucket has only one record with emp\_status N. Now depends upon the query, optimizer decides whether to use index or Full table scan.

SQL> execute DBMS\_STATS.GATHER\_TABLE\_STATS(OWNNAME => 'SCOTT', TABNAME => 'EMP',ESTIMATE\_PERCENT =>  
10, METHOD\_OPT => 'FOR COLUMNS SIZE 2 EMP\_STATUS',CASCADE => TRUE);  
  
PL/SQL procedure successfully completed.  
  
SQL> select ename from scott.emp where emp\_status='Y';  
  
3670016 rows selected.  
  
--------------------------------------------------------------------------  
Id Operation Name Rows Bytes Cost (%CPU) Time  
--------------------------------------------------------------------------  
0 SELECT STATEMENT 3681K 31M 5375 (5) 00:01:05  
\* 1 TABLE ACCESS FULL EMP 3681K 31M 5375 (5) 00:01:05  
--------------------------------------------------------------------------  
  
SQL> select ename from scott.emp where emp\_status='N';  
  
--------------------------------------------------------------------------  
Id Operation Name Rows Bytes Cost (%CPU) Time  
--------------------------------------------------------------------------  
0 SELECT STATEMENT 1 9 1 (0) 00:00:01  
1 TABLE ACCESS BY INDEX ROWID EMP 1 9 1 (0) 00:00:01  
2 BITMAP CONVERSION TO ROWIDS  
\* 3 BITMAP INDEX SINGLE VALUE IDX\_EMP  
--------------------------------------------------------------------------

Conclusion: Optimizer is using full table scan for the query which returns 3670016 records. At the same time, optimizer is using index scan when for other query which returns one record. This scenario, the optimizer choose the right execution plan based on the query WHERE clause.

Data dictionary objects for Histogram:   
user\_histograms  
user\_part\_histograms  
user\_subpart\_histograms  
user\_tab\_histograms  
user\_tab\_col\_statistics