Bind Sensitive and Bind Aware Cursors

A new hybrid system was introduced by 11g called Adaptive Cursor Sharing (ACS), which is a little more subtle about when to introduce a new plan. The concepts of “bind sensitive” and “bind aware” were introduced. Bind sensitive is true when you have bind variables whose value *may* affect the execution plan. Bind sensitive means that the optimizer suspects that a new plan may be appropriate for some values, but it’s not sure. Think of bind sensitive as the first step in becoming aware that more execution plans are needed. You have to be sensitive to the bind variables before you become aware. If you run the SQL a number of times so that the number of buffer gets significantly changed for different bind variables, eventually the cursor will be marked “bind aware.” In other words, we went from bind sensitive (we have bind variables) to bind aware (these bind variables make a significant difference to the number of buffer gets). You can track the progress of the bind sensitivity and bind awareness values by looking at the SQLTXPLAIN report.

If the default value is set and dynamic sampling is used, then the optimizer will attempt to sample 64 blocks of data, unless the query is parallelized (see section below, “Dynamic Sampling and Parallel Statements”). This is not a percentage of the table or index size, it is a fixed number of blocks. The number of rows sampled is dependent on how many rows fit into a block. Remember the objective of dynamic sampling is to get some very basic statistics at the last moment just before the query is executed. To minimize this overhead the sampling size is set in blocks to, a clearly defined value that cannot expand or contract with the size of the table rows. Dynamic sampling was designed this way to stop the parsing process from consuming too many resources on large tables. If the dynamic sampling process takes place (and we can check in the SQLTXPLAIN report) the samples collected may help to make the execution plan better than it would otherwise be. The values for the parameter (as mentioned earlier) vary from 0 to 10, and control the operation of dynamic sampling. If optimizer\_dynamic\_sampling is set to

0: No dynamic sampling is used under any circumstances.

1: If there is at least 1 unanalyzed, unindexed, nonpartitioned table and this table is bigger than 32 blocks, then 32 blocks are sampled. This means that if the table is indexed or is partitioned or is smaller than 32 blocks no dynamic sampling will take place.

2: If at least one table has no statistics, whether it has been indexed or not, then 64 blocks are sampled. Partitioned and indexed tables are included in this. This will apply to all tables with no statistics unlike level 1, where some tables will be excluded.

3: If at least one table has no statistics and if there is a where clause with an expression, 64 blocks are sampled. This is trying to fix the problem of expressions on where clauses where it can be tricky to develop the right execution plan. This is more restrictive than level 2. This still applies to all tables just like level 2.

4: If at least one table has no statistics and if an OR or AND operator is used on predicates on the same table, then 64 blocks are sampled. This is attempting to deal with the problem of complex predicates. This is more restrictive than level 2. This level also applies to indexed and partitioned tables just like level 2.