Battle Against Any Guess

by Alex Gorbachev

During my experience with Oracle, I have become very engaged in the user community. I’ve been a frequent visitor on the Oracle Technology Network forums and the Oracle-L list and have become a regular participant and contributor at user group conferences and other events. My experience started with seeking help and gradually shifted towards helping others with their issues. My growth in Oracle has correlated with the booming popularity of the Internet, over which it becomes very easy to both seek and give advice.

While the Internet increases community participation, it also causes some dysfunction that can lower the quality of the information. Many times I have seen online discussions branch into controversial arguments in which the “combatants” are going by guesswork. It is surprising how few people will stop to test what really happens, and instead will battle endlessly over what might happen or what they believe ought to happen.

While my contributions to the community have been usually rather technical, this chapter is more generic and rather motivational reading. My first attempt at a motivational contribution was creating BattleAgainstAnyGuess.com, or the BAAG Party for short, in June 2007. This is where the title of the chapter comes from. The trigger to establish the BAAG Party was coming across yet another quest for guess-based solutions on the Internet; and I wanted something generic to refer to every time I see such symptoms. Thus, I want to start this chapter by showing some examples of guess-provoking questions.

Guess Hunting

The way you ask a question is crucial. A badly formed inquiry is almost guaranteed to attract guess-based solutions. Here is one example of seeking a quick solution from the Oracle-L list:

“I’m also having performance issues with 10g. Why would my dictionary queries take a long time to return? ... In 9i they used to take seconds, now they take minutes or they just never come back…”

When reading this question, it is difficult to divine precisely what the problem is that the writer is experiencing. Without a clear understanding of the problem, the proposed solutions were all over the map. Here are some of the suggestions that turned up quickly:

“You might need to analyze the system tables.”
“There are a few known bugs with DD queries in 10g. Few of them involved the CDEF$ table so you might want to do a search for that and/or for the particular views you’re having trouble with. The solution was to delete statistics from the involved tables and then lock the stats.”

“Remove any initialization parameters set for Oracle 9i.”

“Apply application vendor suggestions (like _optimizer_cost_based_transformation=false, NLS_LENGTH_SEMANTICS=CHAR, _gby_hash_aggregation_enabled=false).”

“Disable sub-query unnesting (_UNNEST_SUBQUERY = FALSE).”

“Don’t use FIRST_ROWS optimizer goal.”

All these might be absolutely valid solutions for different people’s own problems. One could very well be the solution the original poster needs, but we don’t know which one. A couple of these solutions actually contradict each other (collect vs. delete statistics). These recommendations are based on the previous experience (often quite extensive) of those who proposed them, and they might match well the symptom observed, that “dictionary queries take a long time to return.” However, there is one common problem to all of the proposed solutions: the analysis phase is missing. No one has done any analysis or testing to verify the problem, or to verify that their proposed solution even addresses the problem. Everyone is, in essence, guessing.

To show you the magnitude of guesswork and where it leads, here is another example, this time from the OTN Forums:

“My database running on AIX 5.3, oracle getting the version 9.2.0.5.0, after migration it is getting very slow. Kindly provide a solution to tune the database and increase performance.”

This is probably an extreme example of ignorance and/or laziness that pushes the author to search for a quick fix solution. Now let’s see how this plea for help is being followed up. First of all, a number of people asked for clarification on what is actually running slowly, and for more details about the environment—fair enough. However, there was also a shower of well-meaning advice. One well-intended bit of advice was:

“You can delete and re-gather dbms stats for your application schemas after the upgrade.”

And these were the results after the original poster tried implementing the advice:

“getting same problem
continuously database have lock and the
dbcache hit ratio is 60% only.
total sga size is 20GB
db_cache_size 13gb”

This next proposed solution is a bit better. It’s hinting towards actually analyzing the problem:

“From OEM you can view the performance and the SQL statements which are being fired every moment and then find out about missing indexes or tune the SQL.”