WHEN PERFORMANCE TUNING IS A MATTER OF LIFE OR *DEATH*

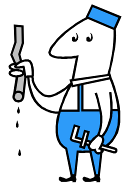
Performance tuning requires creativity and innovation. It's not at all just a routine, drab job—it requires versatility and stepping into different roles. That's why I like it so much.

Those who read this column know that I call the overall performance tuning process "Physician to Magician." This means that we play a different role at different times. Sometimes the analyst needs to gather facts, like a detective. At other times we simply listen to our patient, like a kindly physician. Later, when we need to synthesize a solution, we are like an artist, creating a work of art that our customer will appreciate.

Now, I admit that not all of performance tuning is exciting. There's plenty of cases that are solved by adding an index, fixing the statistics, etc. These cases require more scientific competence than creativity. In general, however, performance tuning is not a mechanical process of "turning the crank" to spit-out a solution to a performance problem. Designing a fix that meets your customer's needs is often not so trivial.

A MISUNDERSTANDING OF THE PROCESS

Those not truly experienced in tuning tend to trivialize the process, frequently using trite phrases such as "add missing indexes" or "increase buffer cache." This misunderstanding helps explain why relatively few DBAs become really good at performance tuning—their view of the task is simply too short-sighted. If your solution set comprises only trivial solutions like new indexes, you're likely not going to excel in this field.



Cranking the handle?

Here's a sampling of various root causes of performance logjams:

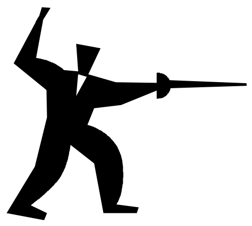
- Overly strict business requirement
- Optimizer incorrectly handling new Oracle functionality
- Optimizer predicting one execution plan, but really running another
- Program calling custom function millions of times

Note that in each case above, a good solution would not be forthcoming by a "turn the crank" approach. The solution for each case required understanding of the particular environment. Sometimes, however, even with all facts in hand, a solution is evasive, or very expensive for our client. Allow me to suggest a novel alternative—one that I have found very helpful.

A MATTER OF "LIFE AND DEATH!"

Here's a tactic I use when I'm faced with an apparently unsolvable performance problem. I imagine that the problem is so critical that people will actually *die* if I don't resolve the difficulty. I imagine that the lives of people are actually in my hands.

WHEN PERFORMANCE TUNING IS A MATTER OF LIFE OR *DEATH*



The deadly performance problem

Yes, I admit this is a bizarre idea, but it actually works. If you can temporarily pretend that there are no limits, you'll be amazed at the variety of solutions you can invent. Here's an actual case where I "saved" a life.

FIX PERFORMANCE OR SOMEONE DIES!

A major retailer had a problem with a long-running batch job. We were able to identify several bottlenecks; unfortunately, the fix required a program redesign; however, this was not acceptable. Along with QA testing and all the paperwork, the ensuing delay would be at least several weeks, if not a month or more.

In cases like this, it's tempting to throw the problem back at the developers. Of course, this isn't too helpful to the client. So just for fun, I *pretended* that my performance problem was life-threatening. I asked myself, "Is it *really* true that I can do nothing to speedup the job? Even if someone dies I can't do anything?"

THE PERFORMANCE "BOOST"

Given my new motivation, all limits were off. I considered all kinds of crazy ideas—server changes, init.ora changes, disk changes, etc. Finally, I thought of one very simple, but admittedly *weaselly* idea. Since the bottleneck was mostly due to disk i/o, any improvement in disk access would directly affect the problem job. Since we knew the exact Sql, we could pre-run Sql that duplicated the "real" Sql that would shortly follow. That is, we would *pre-cache* many of the blocks that would shortly be needed. Caching would take place within the SAN unit, as well as at the Oracle database cache. The entire performance improvement would take place with absolutely no change to the batch program!

To speed-up the pre-caching program, we used multi-threading to run about 20 simultaneous database sessions (we had the spare CPU capacity). This was really simple, and made use of Oracle's parallel query option. (For details on how to cause Oracle parallelism to perform multithreading, see OracleMagician.com/magic12.pdf.)

We successfully ran the "Boost" program just prior to the problem batch job, reducing the runtime by several hours. This satisfied the short-term needs of the client. (Incidentally, I have used this "boost" idea several other times, where an urgent, short-term fix was required.)

WRAP-UP

Okay, I admit my "life and death" scheme is an unusual way to approach performance tuning. While our fix would certainly not be considered a long-term solution, it did give the client some breathing room, so that they could fix the program design. That's all the customer needed.

The concept discussed here is really more about the art of human creativity and motivation than about databases. One could argue it's really just another way to "think outside the box." Here's the point:

WHEN PERFORMANCE TUNING IS A MATTER OF LIFE OR *DEATH*

You don't really what you can do unless you're pushed really hard. So, the next time you face a seemingly impossible problem, don't be too quick to throw in the towel. Instead, pretend, just for a minute, that you absolutely have to fix it—or else!

Chris Lawson is an Oracle Performance Consultant and Oracle "Ace" who lives with his family in Dublin, California. He is the author of *The Art & Science of Oracle Performance Tuning*, available on Amazon. Chris' web site is: www.Oraclemagician.Com. In his spare time, Chris enjoys golf, choral singing (bass), and throwing frisbees to Morgan (the border collie).