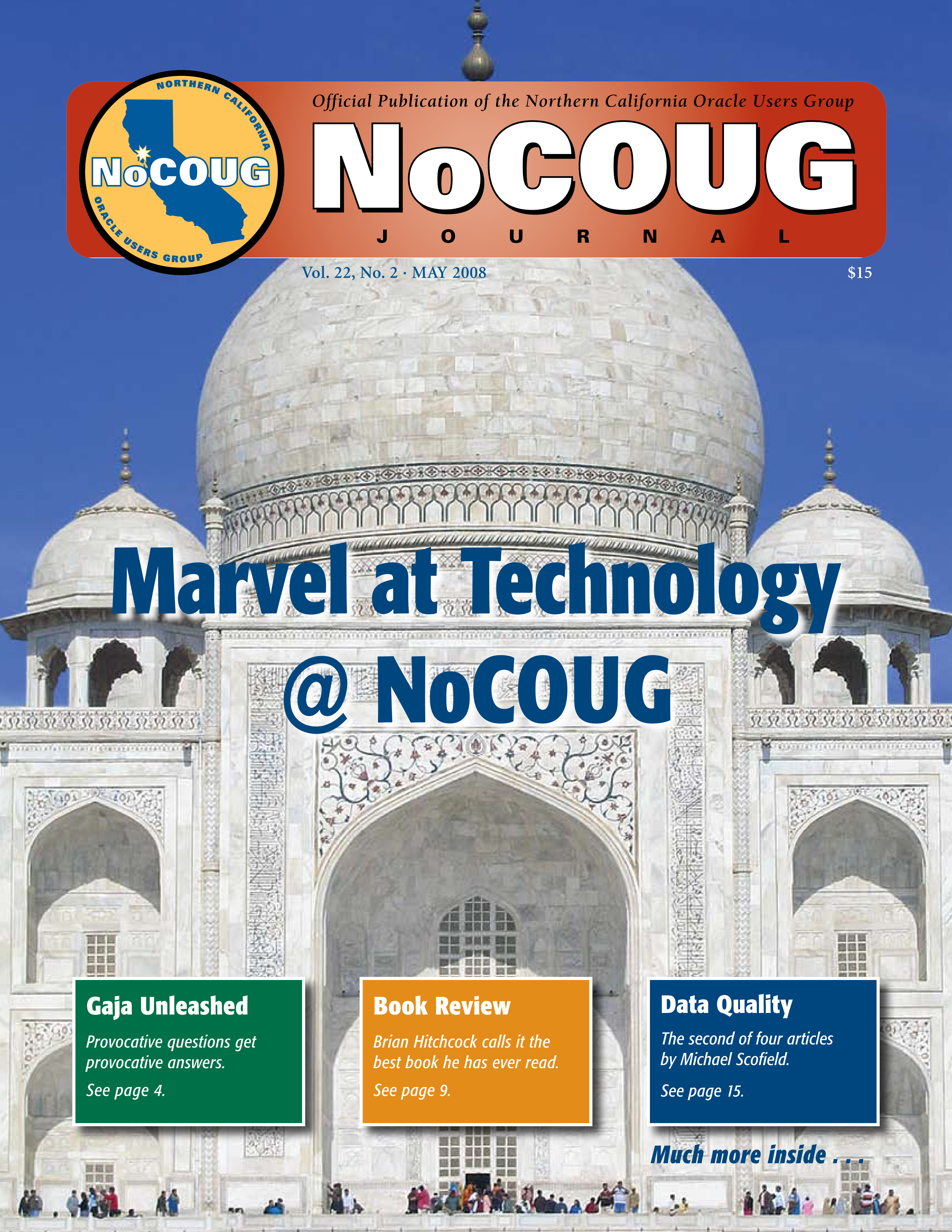
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 22, No. 2 · MAY 2008

$15



Marvel at Technology @ NoCOUG

Gaja Unleashed

Provocative questions get provocative answers.

See page 4.

Book Review

Brian Hitchcock calls it the best book he has ever read.

See page 9.

Data Quality

The second of four articles by Michael Scofield.

See page 15.

Much more inside . . .

Gaja Unleashed

Gaja Vaidyanatha

Gaja Krishna Vaidyanatha is a regular at NoCOUG; he's a frequent contributor to our journal and a frequent speaker at our conferences. He's a member of the Oak Table network and invented the term Compulsive Tuning Disorder *for the disease that afflicts so many of us.*

What's the diagnosis, doctor? I love to try out the latest ideas. My motto is "Nothing ventured, nothing gained." Do I have Compulsive Tuning Disorder?

That's awesome! My first thought to your motto is "*Give a man enough rope and he'll hang himself.*" Don't get me wrong, I'm all about implementing appropriate solutions for appropriate problems. But in the end, you have to ask yourself whether you possess a method to your madness. If you do, then venturing into the realm of new ideas may not be a bad thing, given the right context. But if you took every new feature in every new release of the software and tried implementing it, just because . . . then you are in for a serious increase in production bad hair days! If that is the case, you may have to pay a visit to your local Tuning Doctor in search of a cure to your chronic malady—CTD.

I want to upgrade my database from Oracle 9i to Oracle 10g, but my manager's motto is "If it isn't broken, don't fix it." The database runs beautifully right now but I don't want my brains to rot. How do I convince my manager?

I'm a big believer in changing the fewest things to make a positive impact when confronted with a problem. The issue here is that *there is NO problem!* Well almost . . . although there is no problem with your application running on Oracle 9*i*, there is the issue of support for your database. Given that Premier Support has officially ended in July 2007, is your management willing to dole out extra cash for *extended* support? Even if they are willing, do they realize the inherent limitations of such a setup, as it relates to bug fixes and backporting! Finally, do they realize that July 2010 will be here before they know it, and it will be in everyone's interests to have an upgrade plan tested and implemented before that? Living on the edge excites many people, but running a production Oracle database without support is not my idea of fun!

How about RAC then? It'll work wonders for my resume and I've got it working on my laptop already.

Really? Then you have something in common with my buddy James Morle, who in 2003 configured one of the world's largest laptop RACs at an OracleWorld Conference in Copenhagen, Denmark. I think at the time, he had up to 10 nodes in his cluster. I'm all for RAC when:

- ➤ Maintaining high availability of a database is non-negotiable, and providing application uptime during instance-level changes/upgrades is required.
- ➤ The largest SMP box money can buy is packed with the maximum number of CPUs, and you still need more CPU power to scale your application.
- ➤ Your application generates so much change in the database that the redo logs are the single biggest source of bottlenecks (waits) in your database.

"If you took every new feature in every new release of the software and tried implementing it, just because . . . then you are in for a serious increase in production bad hair days! If that is the case, you may have to pay a visit to your local Tuning Doctor in search of a cure to your chronic malady—CTD."

Gloves Off

Will there be any jobs left for me in five years? Should I move to Bangalore? Are the Indian shops any good? How should I adapt? One of my previous employers suffered wave after wave of layoffs. I personally know many IT professionals who have switched careers: a PeopleSoft engineer became a police officer, a project manager became an insurance agent. Should I plan on a career change? I've always wanted to be a beekeeper.

Hmmm . . . a beekeeper . . . you surely have given me food for thought about my own future! Of course there will be jobs left in five years. The Indian shops are as good as you manage them. The same goes with the other IT outsourcing nation. With the large time differences between the U.S. and those countries, and various other issues that one faces with outsourced entities, offshore application development and system management is a mixed bag with lots more than what meets the eye—cost savings. Now whether or not you move to Bangalore is your call, but if I were you, I would stay plugged into the projects that are going overseas and make sure that you play an integral part in their design and development. The role of a DBA and a developer has changed in the past few years. I don't believe there are fewer DBAs and developers today when compared to five years ago. So there is my one bit of advice for adapting to the new world: Survival of the fittest. It doesn't matter whether it's in the Americas, the Middle East, Europe, Australia, or Asia.

"I don't think Google will ever replace the experience that a speaker shares with his attendees. The most important aspect that attendees of any user group event look for is jobcentric relevant information during the conferences. That is, how is the content of this talk relevant to my day-to-day job?"

It costs us about $30,000 to produce the Journal and a little more than that to organize four conferences per year. We have about 500 members and a little less than 200 attendees at each conference. Our membership levels have been stagnant for a long time. Has Google made us obsolete? I begged my previous company to buy a corporate membership but, except for my manager, not a single person from that company showed up at our conferences. Should we close shop? What could we do better to attract more members?

"The role of a DBA and a developer has changed in the past few years. I don't believe there are fewer DBA's and developers today when compared to five years ago. So there is my one bit of advice for adapting to the new world: Survival of the fittest. It doesn't matter whether it's in the America's, the Middle East, Europe, Australia, or Asia."

I don't think Google will ever replace the experience that a speaker shares with his attendees. The most important aspect that attendees of any user group event look for is job-centric relevant information during the conferences.

That is, how is the content of this talk relevant to my day-to-day job? This means that speakers need to provide meat to the topic they are discussing. This includes, among other things, details regarding any relevant methodology, design, configuration, and implementation of the topic discussed. If speakers engage in a bunch of hand-waving and techno-marketing that does not really provide nuts-and-bolts details, attendees are going to be less enthused about taking time away from work to listen to it.

This means that the method to select a presentation/paper by the conference committee has to be relatively foolproof. Apart from the keynote, which can be at a high level, all presentations need to provide applied details. If you don't ensure that, you are going to see the attendance dwindle over time. And please don't even think about closing shop . . . just improve the quality and technical detail content of the talks, and you should see folks coming back asking for more!

Ask the Oracle

Is 24x7 a myth?[1]

Lose weight instantly with this magical pil . . . no exercise or diet required! Transform your life completely within 12 minutes with this self-help video! Become a millionaire in less than 30 days without leaving your home . . . and more. Our day-to-day lives are filled with many such unbelievable claims. The media does a great job of propagating this.

If HA is defined by 99.999% uptime in any given year, the available downtime both for unscheduled and scheduled events is a paltry 5.2596 minutes. That is all the time that you have for all of your system maintenance, including routine database administration, and operating system and Oracle

[1] Originally printed in the "Ask the Oracles" column in the May 2006 issue of the *NoCOUG Journal.*

software patching. I don't think I have to even attempt to convince you that this uptime goal is impossible and downright crazy for most systems!

No matter how well you have automated your environment, using only 5¼ minutes of downtime a year to perform a plethora of required system maintenance and database administration activities in most systems (especially running Oracle databases) is a lofty goal. For most systems out there, there is a much higher probability that the DBAs supporting those systems will scale Mt. Everest without oxygen than there is of the system achieving 99.999% uptime! Remember the last time you applied an Oracle patch? How long did it take? Did everything go as planned? How long was your application (system) unavailable? Remember the last time a simple "zero risk" effort on your database caused downtime? Need I say more?

P.S. High availability in an Oracle environment is almost always equated to Oracle RAC. If you are thinking of achieving HA with Oracle RAC and you have not designed your application for a clustered environment . . . think again! Slapping Oracle RAC into your environment without expending the necessary time and effort to design your application and your environment is a guaranteed recipe for disaster . . . at least in the realm of performance management.

Does the optimizer need a hint?[2]

Let me start with a life-altering philosophical question to my male readers! When your wife—or significant other—gives you a hint, what does it really mean? If you are a smart man, you will answer, "A hint from her is a directive." If you have answered the above question correctly, you probably understand the Oracle Optimizer very well. A Hint in a SQL statement is—in no uncertain terms—a directive to the Optimizer.

"For most systems out there, there is a much higher probability that the DBAs supporting those systems that will scale Mt. Everest without oxygen than there is of the system achieving 99.999% uptime!"

But wait—do Hints always work? Let us add some real-life perspective. It is Super Bowl Sunday and you are ensconced in your comfy leather couch, watching the game on TV and eating out of a bag without consciously tasting its contents. You are witnessing the dying moments of the grand finale of the NFL season. Your beautiful wife, who is fixing dinner while keeping tabs on the game, gives you a

[2] Originally printed in the "Ask the Oracles" column in the August 2006 issue of the *NoCOUG Journal*.

"Does the Optimizer need a Hint? It sure does! Not always—just every now and then. Why? It is because the Opimizer and we do not live in a perfect world!"

hint: "Honey, the trash is full!" You hear the hint, but choose to ignore it. Given that the context was inappropriate, you firmly believe that the hint is invalid! Your significant other—to your absolute dismay—then plants herself in front of the television and asks, "Honey, did you hear what I just said?"

What your wife demonstrated was a real-life example of how to "Explain Plan." She gave you a hint, but followed up by asking you to "Explain Plan"! The moral of the story: if you introduce a Hint into your SQL, follow up by asking Oracle to "Explain Plan" and verify that the Optimizer is using it.

You may be interested in knowing that the Oracle Optimizer possesses "selective hearing" or "filtering," just as you and I do. There are many situations where the Optimizer registers the Hint that you give it, but chooses to ignore it even if the Hint is syntactically accurate. You can—and should—confirm this by asking it to "Explain Plan." To flash back to the final quarter of Super Bowl Sunday—taking the trash out during the dying moments of that game just did not seem to add up. It did not make sense. It was invalid! Thus you ignored it.

In a perfect world, we and the Oracle Optimizer may not require Hints. This is because, in a perfect world, we will always possess accurate statistics and the perfect con-text. The point to note here is that the Optimizer has to make split-second decisions based on previously collected statistics. Nine out of ten times, when an Optimizer picks the "wrong plan," it does so because of insufficient or inappropriate statistics. And the single most relevant reason when the statistics go bad is in the cardinality of column values.

So you may ask, "What if I always computed statistics—using a sample size of 100%—on all of my objects and gave the Optimizer 100% accurate statistics? Will that be the perfect place to be?" The answer is no, not always. In fact, during a recent performance tuning engagement, we found that deleting the statistics on one of the tables and its associated indexes actually made a certain query run faster and consume fewer resources. The bottom line in that particular case was that the Optimizer chose an inefficient join method even when it had access to 100% accurate statistics, and that it did a significantly better job with default cardinality assumptions and dynamic sampling methods. Note that computing statistics using a sample size of 100% may not even be feasible if some objects are very large.

So then—does the Optimizer need a Hint? It sure does! Not always—just every now and then. Why? It is because the Optimizer and we do not live in a perfect world!

Oracle Best Practices and Worst Practices[3]

Wikipedia defines *best practice* as "a management idea which asserts that there is a technique, method, process, activity, incentive or reward that is more effective at delivering a particular outcome than any other technique, method, process, etc. The idea is that with proper processes, checks, and testing, a desired outcome can be delivered with fewer problems and unforeseen complications. Best practices can also be defined as the most efficient (least amount of effort) and effective (best results) way of accomplishing a task, based on repeatable procedures that have proven themselves over time for large numbers of people."

Those of you who have been there and done that with Oracle systems for many years know very well that it is impossible to predict and plan for every scenario that your application and database will present in production. This is especially true when you embark into upgrading any component of your system—especially when you are betting on a new feature in the software to bail your system out of a sticky situation! But you upgrade anyway, because inaction is worse than the current performance status-quo or corner-case bugs that you may encounter in the future. Best practices come in very handy in situations such as these.

In a recent performance management engagement at one of my customer sites, I was privy to a strange (yet seemingly common) situation. The Oracle database was upgraded from 9*i* to 10*g*, and even though all of the pre-upgrade load tests

"Chant and practice the response time mantra. At the end of the day, performance management is about user experience and application response times, not a slew of database health metrics. If you engage with a specific response time goal in mind, you will stop tuning when you achieve that goal. Remember, Compulsive Tuning Disorder is not a good thing!"

performed in stellar fashion, the database experienced sporadic hiccups in performance during the post-upgrade phase in production. And during these hiccups, the CPUs on the system were constantly and consistently maxed out. This was because the execution plans of the core SQL in the application had gone south. When a 32 CPU machine with an average idle percentage of 85% in Oracle 9*i* gets maxed out in Oracle 10*g*, you know there is a gremlin lurking out there. With adequate performance diagnostic data, it was determined that Oracle's "automatic feature" of "peeking for bind variable values" at hard-parse time was the cause for the pain. To ensure future

"Databases such as Oracle should not act only as "data stores," as they provide way more functionality for scalable performance. Not utilizing the inherent power of the RDBMS is like driving a Lamborghini with 95% of its power turned off."

stability and to guarantee performance capacity during the 2007 holidays, the feature was completely turned off.

How do you plan for a situation such as this, when it did not surface during pre-upgrade performance load tests? Clearly, in this case, the pain had to be experienced before remedial measures could be undertaken. But does that mean that you don't test? No, you still do. Nor does this mean that you take preemptive action and turn off every new feature or modify the default values of every Oracle initialization parameter that is different from the prior release—especially when it is an "underscore" parameter.

From an Oracle Performance Management perspective, the following are some key best practices I'd recommend:

- **System Performance Diagnostic Best Practice**—Chant and practice the response time mantra. At the end of the day, performance management is about user experience and application response times, not a slew of database health metrics. If you engage with a specific response time goal in mind, you will stop tuning when you achieve that goal. Remember, Compulsive Tuning Disorder (CTD) is not a good thing!
- **System Performance Diagnostic Best Practice**—Utilize mathematical data instead of expert opinions to arrive at all of your conclusions. Opinions are not facts and can be very easily protected by the expert with the YMMV (Your Mileage May Vary) camouflage. When you can categorically state that a query's elapsed time is 15 minutes, and 83.5% of its elapsed time is spent in performing single-block I/O requests, then that is a quantified problem that you can attempt to solve.
- **Application Performance Testing Best Practice**—Create a realistic test environment, so that major upgrades can be tested for feasibility, reasonability, and stability. Production life is hard enough with testing, and you truly do not want to be out there without testing. Your load tests should reflect reasonability and veracity when compared to your production environment. Testing should be done using realistic production data (or at least production Optimizer statistics), from both the database and application's perspective.
- **System Performance Environment Best Practice**—Engage in balanced maneuvers when building and up-

[3] Originally printed in the "Ask the Oracles" column in the February 2008 issue of the *NoCOUG Journal*.

grading systems. If you double the capacity of your CPUs, then take the time to review the capacity in your I/O sub-system to determine whether it too requires a similar upgrade. Throwing more or faster CPUs without reviewing the capacity of your I/O sub-system may result in real-time system imbalances. These imbalances will translate into long-term scalability and performance problems.

- **Application Performance Design Best Practice**—If a piece of application functionality is data intensive, build it with SQL first. If the code requires complex logic, then build it in the database (yes PL/SQL). Alternatively, if the functionality is processing intensive, then build it in the application tier. In either case, avoid unnecessary single-row processing roundtrips between the database and the application tier. Based on my experience in the field, I'd recommend that you steer away from the following worst practices:
- **Building "database agnostic applications"**—This is a commonly used buzzword line that does not make any sense. The term "agnostic" means "to neither believe nor disbelieve" and to limit one's belief to human experience. Are we saying that if we reinvent the wheel and build database constraints in the application tier, we are actually limiting our belief in Oracle based on our experience? Nonsense! Databases such as Oracle should not act only as "data stores," as they provide way more functionality for scalable performance. Not utilizing the inherent power of the RDBMS is like driving a Lamborghini with 95% of its power turned off.
- **Overusing dynamic SQL at the application tier**—If Oracle has to parse every single SQL statement that you present to it because of the dynamism that you have introduced in your application logic, it makes your system that much less scalable. There are application environments where CURSOR_SHARING cannot be set to FORCE and dynamic SQL (non-reusable SQL) will open up a slew of library cache and shared pool latch contention performance problems.
- **Designing without scalability**—If you have the luxury of custom designing any part of your system, not ensuring that it will work with large data sets is a worst practice. For example: if you write a query that works well with a one-thousand-row table, not ensuring that it will exhibit a reasonable response time when processing a one-million-row table is a recipe for disaster. The underlying theme here is "design for scalability."
- **Using database ratios to determine the true performance bottleneck**—Ratios may be indicators but definitely not "root causes" of performance problems. In all of my engagements with my customers, I have yet to find the need to review a single database ratio to solve a customer's performance problem. The 10046 trace files combined with STATSPACK and AWR reports and some OS commands pretty much define my tuning arsenal. Yes, I do skip the ratios in the aforementioned performance reports. And if I do find ratios to be useful in the future, you will be the first to know!
- **Drinking vendor Kool-Aid due to their stature in the marketplace**—Verify the durability and performance of any piece of software or hardware by putting it through the paces in your test environment. Again, remember Best Practice #2—Use mathematical data, not expert opinions. And don't fall prey to sales pitches!

So in the context of Oracle Performance Management, adhering to the best practices listed in this section is a good thing. And by no means is this a comprehensive list of best and worst practices; it is just a beginning. May I remind my readers: "Well begun is half done!" Cheers!

The Final Word

What's the ultimate answer to the great question of life, the universe, and everything?

42!!! ▲

*Gaja Krishna Vaidyanatha has over 16 years of industry technical expertise working with Oracle systems. He is the Principal of DBPerfMan LLC (**www.dbperfman.com**), an independent consulting firm specializing in the area of Oracle Database Performance Diagnostics & Management for Fortune 500 corporations. He is the primary author of* Oracle Performance Tuning 101*, published by Oracle Press, and one of the co-authors of* Oracle Insights: Tales of the Oak Table*, published by Apress. He can be reached at* **gaja@dbperfman.com**.

"If a piece of application functionality is data intensive, build it with SQL first. If the code requires complex logic, then build it in the database (yes PL/SQL). Alternatively, if the functionality is processing intensive, then build it in the application tier. In either case, avoid unnecessary single-row processing roundtrips between the database and the application tier."