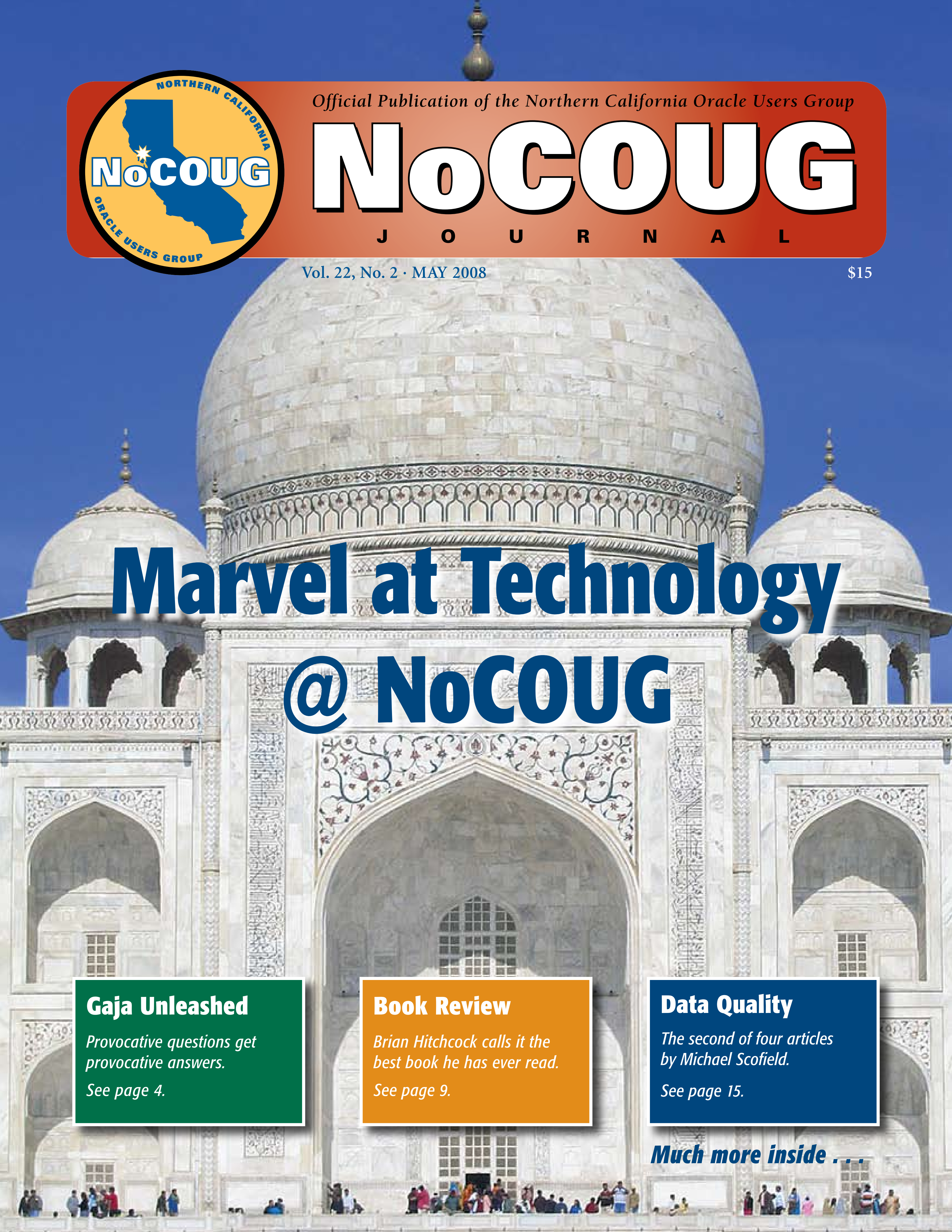
Official Publication of the Northern California Oracle Users Group

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 . . .

Marvel at Technology @ NoCOUG

The Taj Mahal is not only one of the most beautiful buildings in the world, it is also a marvel of building technology. It is built on a sandy river bank and its immense weight is supported by wells filled with rock. The four towers lean slightly outward so that if they ever fall in an earthquake, they will fall away from the main structure. The elaborate calligraphy around the immense archways provides a marvelous optical illusion: all the letters appear to be the same size to the observer at ground level.

Oracle Database 11*g* may not be a work of art but it is a marvel of software technology, and it is on display at NoCOUG's spring conference on May 15 at Crowne Plaza Hotel in Foster City. Richard Niemiec will deliver the keynote address; his subject is *How Oracle Came to Rule the Database World*. He will also make a technical presentation on the best new features of Oracle 11*g*.

In this issue of the *Journal*, we present the second in a series on data quality by Michael Scofield, a provocative interview with Gaja Krishna Vaidyanatha, and an insightful book review by Brian Hitchcock. I hope you enjoy it and I hope to see you on May 15. ▲

—Iggy Fernandez, *NoCOUG Journal* Editor

Table of Contents

Publication Notices and Submission Format

The *NoCOUG Journal* is published four times a year by the Northern California Oracle Users Group (NoCOUG) approximately two weeks prior to the quarterly educational conferences.

Please send your questions, feedback, and submissions to the *NoCOUG Journal* editor at **journal@nocoug.org**.

The submission deadline for the upcoming August 2008 issue is May 31, 2008. Article submissions should be made in Microsoft Word format via email.

ADVERTISING RATES

The *NoCOUG Journal* is published quarterly.

Size	Per Issue	Per Year
Quarter Page	$125	$400
Half Page	$250	$800
Full Page	$500	$1,600
Inside Cover	$750	$2,400

Personnel recruitment ads are not accepted.

journal@nocoug.org

The Well-Rounded Oracle Professional

by Roger Schrag

Roger Schrag

I visited the Taj Mahal ten years ago, and it is indeed something to marvel at. Photographs (like the one on the cover of this *NoCOUG Journal*) just can't convey the emotion you feel when surrounded by throngs of people—locals and tourists alike—awestruck by this 17th-century mausoleum built under Emperor Shah Jahan in memory of his wife, Mumtaz Mahal.

Thinking about the Taj takes me back to an interesting point in my Oracle career. By 1998, I had been working as an Oracle professional—first as an application developer and then as a database administrator—for about nine years, and I had started a consulting firm specializing in Oracle database management three years earlier. Looking back now, I think it was right about this time that I transitioned from being a "good Oracle DBA" to a "well-rounded Oracle professional."

After nine years of coding Oracle database applications and managing Oracle databases, I had gotten reasonably good at it. I had a lot of experience optimizing SQL, I knew some good questions to ask when designing a database schema, and so on. But I had come to realize that simply knowing how to harness Oracle technology was not enough.

I recently heard a story about an eager and able web developer who volunteered to rebuild the website for a nonprofit organization. He gave the new site a fresh look and he set up a Joomla! content management system so that the material on the site could be updated frequently with ease. Unfortunately, the new website never got put into production and the web developer moved on. Why? The nonprofit organization found that while the new website looked sharp and the content management system was sophisticated, it did not provide the functionality that the organization's customers required.

We've all heard stories of failed projects like this one. Projects can go amiss for lots of reasons, but poor communication is often a factor. Key players on a project working in a reactive mode instead of a proactive one can also contribute to the failure of a project.

I believe that the well-rounded Oracle professional not only has a strong command of Oracle technology, but also has superior communication skills, is a solid team player, and takes a proactive approach to everything he or she does. As a "good Oracle DBA," I liked to find slow queries and make them run faster. But as a "well-rounded Oracle professional," I understand that user impact is more important than number of logical reads, and thus many slow queries don't need to be made faster at all.

NoCOUG is a wonderful asset to Oracle DBAs and developers. With *NoCOUG Journal* articles and conference presentations on such topics as performance management, best practices, and Oracle Database 11*g* new features, it is easy to see that NoCOUG offers many ways for Oracle professionals to sharpen their technical skills. But what was becoming clear to me around the time of my trip to India ten years ago was the importance of the nontechnical skills in career development: communication, teamwork, and a proactive nature—so-called "soft" skills.

NoCOUG can help you develop your soft skills as well. Did you attend Jeremiah Wilton's keynote presentation at NoCOUG's 2007 Fall Conference? In the debate between empiricism and guesswork, Jeremiah demonstrated the importance of communication and following things through, and the limitations of pure technological prowess. And at the NoCOUG conference this past February, Bradley Brown gave a presentation on what he calls "The Arrow of Truth," a method for resolving conflict and effectively communicating with others. The most successful Oracle professionals have good technical skills but also excel in areas such as these.

I believe it was around the time I visited the Taj Mahal in 1998 that I shifted my career development from focusing purely on technical skills to both technical and soft skills. I went on to build up my consulting firm to more than a dozen employees while remaining a hands-on DBA who could mentor the technical team and explain our work to nontechnical managers as well.

Not everyone aspires to lead a company or even lead a team. But everyone who aspires to be the best Oracle DBA or developer they can ought to develop their communication skills, teamwork, and proactive approach while growing their technical skills. To me, these soft skills separate the well-rounded Oracle professionals from the good DBAs and developers.

I hope to see you at NoCOUG's Spring Conference in Foster City on May 15th! We'll be kicking the day off with a keynote and technical presentation by Rich Niemiec, past IOUG president and ever-popular presenter at Oracle conferences around the country. As always, we'll have a dozen technical sessions for you to choose from, book raffles, networking opportunities, and more. This will be an important day for the development of your Oracle career. Your mission, should you choose to accept it, is to find ways to develop your soft skills and your technical skills in tandem! ▲

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 back-porting! 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

"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!

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

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!!! ▲

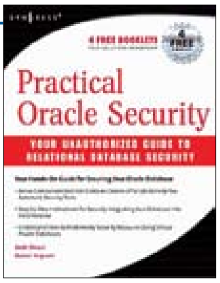
*Gaja Krishna Vaidyanatha has over 16 years of industry technical expertise working with Oracle systems. He is the Principal of DBPerfMan LLC (**www.dbperfman.com**), an independent consulting firm specializing in the area of Oracle Database Performance Diagnostics & Management for Fortune 500 corporations. 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."

Practical Oracle Security: Your Unauthorized Guide to Relational Database Security

A Book Review by Brian Hitchcock

Details



Authors: Aaron Ingram and Josh Shaul
ISBN: 978-1-59749-198-3
Pages: 432
Year of Publication: 2007
Edition: 1
Price: $27.95
Publisher: Syngress

Overall Review

This book is among the best Oracle books I've ever read. It was quick and I learned many things I can apply to my work immediately. It didn't waste my time going over the basics (what is a database?) and got to the point immediately. It doesn't pad the content to get to 600 pages, which seems to be required now for most technical books. The authors are very good at conveying useful information rather than just trying to impress you.

Summary

Overall review: Excellent, the most useful book I've ever read on Oracle technology.

Target audience: Almost anyone that supports or manages Oracle databases.

Would you recommend to others: Yes.

Who will get the most from this book: DBAs and others managing the security aspects of Oracle databases.

Is this book platform specific: No.

Why did I obtain this book: I've been working on security issues in Oracle databases and Oracle applications for some time and have to learn about and deal with security issues all the time.

Chapter 1—Oracle Security: The Big Picture

This overview of Oracle database security history and issues covers a lot of ground. An excellent explanation is given of why the "table or view does not exist" error message actually is a good thing. The section on privilege controls explains where the CONNECT, RESOURCE, and DBA roles came from and why they were created in the first place. This provides valuable perspective on just how much has changed from the early versions of Oracle to the present security concerns. Also explained is why the introduction of SQL*Net in version 5 was the beginning of many of the security issues we still see today. With 8*i*, the authors explain how Oracle hacking went mainstream.

Other sections cover the history of auditing features in Oracle and how Fine Grained Auditing (FGA) in 10*g* was improved to audit DML. The PASSWORD_VERIFY_FUNCTION is discussed. I have experience with this, because when we applied Oracle Applications patches to support this, it broke our Oracle Discoverer reporting environment for Oracle Applications. As with all things, when you patch to fix one thing (in this case to support the password verify function), you may be breaking other things, which will require more patching and testing.

I had never heard of the Bell-LaPadula security model, but the description is interesting and, if nothing else, provides good job -interview trivia. The explanation of Virtual Private Database is very good. I didn't really understand what this was until I read this. Similarly, I didn't know that Label Security could be applied at either the schema or individual table level, offering complete flexibility.

There is also a discussion of the tradeoffs that come with more security features. Oracle 10*g* has lots of security fea-

"This book is among the best Oracle books I've ever read. It was quick and I learned many things I can apply to my work immediately. It didn't waste my time going over the basics (what is a database?) and got to the point immediately. It doesn't pad the content to get to 600 pages, which seems to be required now for most technical books."

tures; this means complexity, which means DBA overhead. As the authors put it, "Oracle is likely the most secure and most vulnerable database in existence today."

I highly recommend the next two sections, "The Regulatory Environment," which provides very good info describing the major pieces of legislation driving many of the current security projects, and "Major Data Theft Incidents," which fills in a lot of details about what has been reported in the popular media but not really explained.

In the section "Appropriate Security for Each Class of Database System," I disagree with the authors when they discuss Development, QA, and test databases. These are often a complete copy of a critical production system and therefore are just as sensitive from a security perspective as the primary system. As the authors point out in other parts of the book, a hacker is just as happy—probably happier actually—to get the sensitive data from a non-production system where no one is watching. Further, even your backup tapes, which are a complete copy of your databases, should be viewed as just as sensitive as the critical systems that were the source of the backups.

In the Frequently Asked Questions section, the authors tell us that 70% of attacks involve an insider. What about consultants or a remote DBA that works on your critical systems for a brief time? Note that even if someone only has full access for a brief time, they can leave security holes that won't be apparent without careful scrutiny. Worse, what I've seen is that they may do things while they have access that open huge security holes just to get the job done on time. For example, granting SELECT ANY TABLE to give developers "read-only" access to the database. While the consultants are gone, they left behind a security issue that makes it easy for insiders to access sensitive data.

Chapter 2—File System

This chapter explains how database users can get access to the filesystem and why this is a security concern. The files in the filesystem are broken down into four categories: data, software, configuration, and logs; the chapter gives specific advice on securing each. A minor issue: the example showing how to list all datafiles doesn't include temp files supporting temporary tablespaces, yet a few pages later we're told how important it is to secure these files.

A good example of a seemingly innocent operation causing a security problem is given. When dropping a tablespace, Oracle doesn't remove the associated datafile(s) from the filesystem. This leaves the datafile(s) on disk where they could be accessed. While not part of the filesystems, the backups made of filesystems could give a hacker access to passwords, which then give access to live data. Backups need to be treated with same concern as the live data. Another related issue is that test machines often contain authentication info that is copied from production.

One security measure suggested is to compute and store hash values for all software and configuration files, so that any changes to these files could be detected. This may be a good idea, but who has the time and resources to implement and maintain this? Similarly, we are told to revoke permissions on Oracle software files, but how do we know the impact this may have on the operation of the software? Encrypting data is a good idea, but what happens if the encryption key is lost? There is a tradeoff between security and the cost of supporting a more secure environment.

"A hacker is just as happy—probably happier actually—to get the sensitive data from a non-production system where no one is watching. Further, even your backup tapes, which are a complete copy of your databases, should be viewed as just as sensitive as the critical systems that were the source of the backups."

Chapter 3—TNS Listener Security

The Oracle listener is the source of many security issues, and the authors tell us that while things are better than they were, there is still much to be concerned with. I didn't know about tnscmd and all the bad things you can do with it. 10*g* listener improvements are discussed but these improvements can all be turned off.

I found the following to be a surprise: If you remotely administer a database, the listener password is sent over the network in plain text, making it easy for anyone who can monitor the network to get the listener password. Further, before 10*g*, the listener password hash would be accepted by the listener as the password.

While not a direct attack on the database, the point is made that a denial of service (DoS) attack against the database listener is pretty much the same thing as a DoS attack against the database.

Given how many security issues are discussed with respect to the listener, the more general point is made that making software work is different from making it secure. This is another way of saying "we met the release date and anything else will get fixed later." Incomplete testing of applications is one of the main reasons why hackers can remain in business. This applies to both the Oracle software itself and the applications you build on top of it.

The authors suggest the best thing to do for listener security is upgrade to 10*g*. Perhaps this is a good idea, but is it really practical? Can most Oracle customers that are not on 10*g* now quickly upgrade to 10*g* just for the listener security improvements?

Another example of this book providing really useful information is the discussion of ADMIN_RESTRICTIONS. Oracle provided a good new security feature in a patch, but even after the patch is applied, these new features had to be enabled by the DBA. How many systems have this patch but don't have the benefit of these security improvements? Yes, the security improvements were patched into the system, but no one remembered to activate them. Again, better security requires more (and effective) resources.

A reference is given to a series of articles covering how to interpret the listener log file. I have read these articles and recommend them, although I haven't taken the SQL and used it on my own systems. The information in these articles is very worthwhile and addresses issues I've had for a long time. I've never been able to get much from the listener logs. Like the rest of this book, this is practical information that I can use on the job and that I haven't seen elsewhere.

Valid node checking is a lesser-known but more useful security feature of the listener. Note that Oracle thinks highly of this feature. When you install Oracle Applications 11*i*, by default this feature is configured so that the only host that can contact the database is the applications tier host. This feature should be used more often, as it prevents a lot of security issues by simply restricting the IP addresses that connection requests can come from.

On a practical note, while reviewing listener security issues and how to fix them, recall that the listener may use a range of ports. While the initial communication is done on the port assigned to the listener, after that the user connection to the database is handed off to one of a range of port numbers. This can cause issues if you have firewalls or other systems that only allow the listener to communicate on a specified port number.

Chapter 4—Managing Default Accounts

This chapter begins with some worrying observations. The success rates of breaking into a production Oracle database using default accounts and passwords are incredibly high. Default passwords are the most powerful and easiest to exploit vulnerability. Online lists of Oracle default accounts cover up to 600 combinations covering multiple versions of Oracle. 11*g* brings case-sensitive passwords as the default, which means case sensitivity was not the default for versions before 11*g*.

"The success rates of breaking into a production Oracle database using default accounts and passwords are incredibly high. Default passwords are the most powerful and easiest to exploit vulnerability."

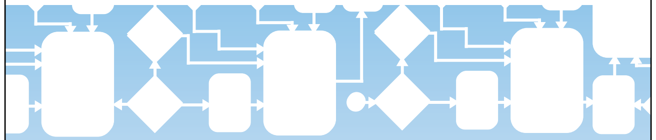
With 10*g* all default accounts are locked at install except for SYS and SYSTEM, which no longer have default passwords. Note that all the other default accounts still have default passwords but they are locked at install. However, if they are unlocked, the default passwords will still be a problem.

The chapter contains Default Accounts lists that contain many that I had never been aware of—for example, ADAMS. We are advised to remove many of these default accounts. While it is good advice, readers should be told how to determine the impact of removing an account and how to re-create that account if needed in the future.

The DBSNMP account is discussed in detail with respect to Oracle Enterprise Manager (OEM). The advice is to change the account that OEM uses and then lock down the DBSNMP account. I would like to see more details of all the steps involved. Oracle tells us to lock and expire default accounts that we don't need. While this makes sense, it's too easy for one or more of these accounts to be unlocked. If this happens and the

Announcing

Embarcadero® EA/Studio™



Join your processes and data with EA/Studio.

- Conceptual modeling
- Real-world process modeling
- Speak the same language as business users

Visit www.embarcadero.com/eastudio to download a free trial.

EMBARCADERO TECHNOLOGIES®

ENTERPRISE STORAGE SOLUTIONS FOR ORACLE.
SCALABLE STORAGE INFRASTRUCTURE.
SCALED DOWN STRESS.

PICK ANY THREE.

SETTLE FOR EVERYTHING WITH NETAPP.

Find out why we're trusted by the world's greatest companies that run Oracle.

Visit netapp.com/bizapps

NetApp® Simplifying Data Management

default password isn't changed, the issue of default accounts and default passwords comes up again.

The authors recommend password management tools to generate and manage passwords. This is needed because really secure passwords are very difficult to remember. For example, I won't be able to remember a password that is a string of random characters. The authors don't offer advice on how secure these password management tools are. How do we know that we aren't making things worse by putting all our passwords in one place? How do we balance the need for secure (hard-to-remember) passwords against practicality if we can't use a password management tool?

I found the explanation of what a hash is to be very useful.

The section defining impossible passwords is great. I had seen this described elsewhere before but didn't know what it meant. This book was worth reading for this information alone. I also think this should have been presented earlier when default accounts were first being discussed. For me, setting an impossible password is a better choice for most default accounts where I'm not sure what the impact would be of removing that account. In the interest of time, I think an impossible password would be a better choice than removing accounts.

We're told to run a default password scan monthly, but I don't know who has the resources for this. Note that while the authors cover commercial products, they also tell us about the free default password scanner available from Oracle.

Chapter 5—PUBLIC Privileges

I have never understood what PUBLIC was. Again, this book was worth reading for this information alone, and this is another example of the valuable information I learned. PUBLIC is neither a Role nor a User. This chapter offers a very good explanation of not only what PUBLIC is, but also the history of how it came to be and how it was never fully realized.

We are told to use roles instead of granting permissions to PUBLIC. The sensible use of PUBLIC is also described. PUBLIC is granted access to SELECT from any tables or views that are prefixed with USER or ALL. I didn't know that. It can be challenging to remove system privileges from PUBLIC. This can affect all users of the database. Do not grant roles to PUBLIC. A password for a role is described but I would like to have seen more detail on how to create this password.

You need to review each database for any privileges that have been granted with the ADMIN option, since this allows a user to grant the privilege to other users. For system privileges, you need to check if any were granted with the GRANTABLE option, which means the grantee can grant the privilege to someone else. Note that DBA_TAB_PRIVS has a column TABLE_NAME, but this can refer to the name of a table as well as other objects such as views, stored procedures, and functions.

An explanation of SQL92_SECURITY is provided. I didn't know anything about this and it was very interesting and useful. It turns out that without this feature enabled, a user who doesn't have SELECT but does have UPDATE can see the data in the table as if they *did* have SELECT privilege.

"Don't ever grant a system privilege that contains ALL in the name . . . Granting SELECT ANY TABLE to database users allows users to select the password hash of any user and use external password cracking software to determine the password."

The example given is very good. The security impact of backwards compatibility is highlighted with the explanation of why 07_DICTIONARY_ACCESSIBILITY exists and what it does.

Very specific advice is given to remove specific permissions from sensitive packages that are granted by default. We are told that patching the database in the future may actually reenable these grants. You should, in a perfect world, check for these issues regularly.

The discussion of how to use UTL_FILE to break into a database is very good. And the fact that Oracle continues to grant privileges on UTL_FILE to PUBLIC is hard to believe, but true. We are told that using DIRECTORY objects is more secure than UTL_FILE, but we also must make sure that the CREATE ANY DIRECTORY is only granted to those who really need it.

The authors are very clear. Don't ever grant system privileges to PUBLIC. Similarly, don't ever grant a system privilege that contains ALL in the name. The explanation of exactly how this can cause big security issues is very good. A list of specific system privileges that PUBLIC must not have is listed. The description of how each privilege could be used to cause security issues is fascinating. My favorite is using DROP USER as a DoS attack.

Chapter 6—Software Updates

This chapter focuses on the Critical Patch Update (CPU) patches that Oracle releases quarterly. Previously Oracle released security alerts (patches) whenever needed to fix severe problems. Note that CPU patches don't address all known security issues. The patches generally fix the most serious issues that can be patched. This means that even if you are current for all CPU patches, your system(s) still may not be patched for all known security issues.

The discussion of the flaws in CPU patching is excellent. Also explained is the methodology used to rank security issues and how this ranking process can be gamed to make some security issues seem less serious than they may be. A detailed discussion of how software development is managed gives great insight into how security issues get started and perpetuated in commercial software. I had no idea.

An excellent example of an issue that appears to have been fixed by CPU patching is a privilege escalation attack where the OBJECT_TYPE can be used to execute code with SYS privileges. There are issues that get "fixed" over and over, indicating that the method of developing the fix is not very effective. A specific example is given where a CPU patch explained a new vulnerability and (in theory) fixed it, but in reality it was six months later when the effective patch was released. For those six months, those that applied the CPU patches on time were exposed to a vulnerability that they had every reason to think they were protected from.

As always, it is recommended to always be current on CPU patches. This is sound advice but it ignores the reality that not all organizations have the resources to assign one or more persons to the CPU patching effort. It is my opinion that to really be on top of all CPU patches and to be constantly planning, patching, testing and releasing into multiple environments would indeed take one or more full-time experienced Oracle DBAs.

In the discussion of planning the CPU patching task, we are told that over time, the planning process will become easier as you can reuse the plan from quarter to quarter. I disagree, at least partially. Patching can be very different from patch to patch. You can reuse the high-level plan but the detailed planning will be different each time. Further, deciding how much testing must be done after a patch can take considerable time as well, since different patches will patch different Oracle software components. This is especially true where you have Oracle Applications running in addition to stand-alone databases.

This chapter also includes a description of molecules and "napply" technology that allows applying a subset of a CPU patch set. This became available as of July 2007. I had not heard of this before.

While planning the patch process, the use of multiple test systems and clones are advised. This assumes you have lots and lots of hardware and people to run it all. It is also hard to have a long enough downtime where you have time to patch, test the patch, and completely roll back the patch before the environment is needed again. Another good suggestion that is hard to do in the real world is to patch everything on the server that relates to Oracle. For a big server running many applications for different organizations all with different security requirements, this can be very hard to accomplish.

Chapter 7—Passwords and Password Controls

The discussion of what makes a password weak covers how Oracle stores the password hash, which is good, but this also points out the dangers of granting SELECT ANY TABLE to database users. This allows users to select the password hash of any user and use external password cracking software to determine the password.

I was interested to learn that brute force password cracking isn't practical for passwords that are longer than 6 characters. This helps me understand why it is frequently required that passwords have a minimum of 8 characters. The point is made again that password management is just as important in non-production systems that are copies of critical systems. In reality, most organizations ignore this issue altogether.

We are told that there is no reason to disable account lockout in any Oracle database. Account lockout means that after a specified number of login failures the account is automatically locked. This sounds good, but you need to be aware that for an account used by an application, if the password gets changed, the application may get locked out if it is still trying to use the old password and this can affect all the users of the application.

Specific recommendations are given for each of the password controls such as FAILED_LOGIN_ATTEMPTS, PASSWORD_LIFETIME etc. The discussion of the issues with remote OS authentication is very good. Also valuable is knowing that the OS authentication prefix must not be NULL. I didn't know about either of these issues.

The authors point out that some security-related tasks may conflict. The example given is account lockout and password scanning software. The software will try to connect to various accounts using a series of passwords. This could result in all the accounts being locked out. You need to disable account lockout as this software runs.

Chapter 8—Database Activity Monitoring

This chapter discusses how to monitor your database(s) and tells us that some estimates show up to 70% of all database attacks are from insiders. This may all be true, but the discussion of monitoring database(s) assumes you have the resources to set up and maintain the monitoring software and to review and take action on all the logs that will be generated.

It is tough to know all the sources your users will connect from when those users can be on the company network, at home, on the road, connecting through their ISP, and so on. Similarly, we are told to define a SQL signature for what we expect the normal pattern of SQL access to be. I have no clue how to start this let alone maintain it. It sounds like a good idea. The description of the Sweeney attack is very good. I had no idea. This is yet another way that your data may not be as secure as you think.

The section discussing adhering to government and industry regulations is very good. It makes me wonder how anyone really does adhere to all of them for any length of time. The authors end the chapter by stating that the Sarbanes-Oxley (SOX) regulations are vague as far as technical requirements are concerned. Been there, done that, don't even have a T-shirt.

"The market needs more books like this—books that don't take months to read and don't go over basic information that the reader should already know . . . I hope this book is part of a larger trend in publishing towards specific and useful information and away from each book being a doorstop that almost never gets read."

Chapter 9—Implementation Guide

This final chapter states that the solutions presented so far are a lot to digest and taken together are a daunting task. I agree. While this chapter gives us a reasonable plan for implementing all the advice given in the preceding 8 chapters, it isn't clear who can afford the resources to make all this happen.

The point is repeated that CPU patches can reverse changes you made to improve security. We are told to remove all access to datafiles and redo logs from users other than Oracle. I don't know the impact of changing the permissions from those that were set up when the Oracle software was created. While it sounds like a good idea, without knowing the affect of these changes, how can I weigh the increased security against the possible downtime caused by these changes?

The point is made that making passwords so hard to guess and therefore virtually impossible to remember means users will write down the passwords. Have we really achieved better security or simply passed an audit?

Conclusion

The market needs more books like this—books that don't take months to read and don't go over basic information that the reader should already know. The authors have a point and they get to it quickly. This book targets a specific segment of the Oracle user base, namely those that already know about database security issues and need specific help working those issues. I hope this book is part of a larger trend in publishing towards specific and useful information and away from each book being a doorstop that almost never gets read. ▲

Brian Hitchcock has worked at Sun Microsystems in Newark, California, for the past 11 years. He is a member of a DBA team that supports 2400+ databases for many different applications at Sun. He frequently handles issues involving tuning, character sets, and Oracle applications. Other interests include Formula One racing, finishing his second Tiffany Wisteria lamp, Springbok puzzles, Märklin model trains, Corel Painter 8, and watching TV (TiVo rules!). Previous book reviews by Brian and his contact information are available at **www.brianhitchcock.net**.

- Analyze performance by workload/application
- Predict performance shortfalls in the next 12 months
- Understand the effects of change: new apps, 10g, more users, new hardware...in seconds

BEZ systems The Prediction People™

Real-World Experience

For Oracle database consulting and support, it makes sense to work with a company that has a proven track record. Since 1995 our clients have relied on us for:

- Performance tuning
- Migrations and upgrades
- Backup and recovery strategies
- Database security audits

Plus, we offer ongoing **remote DBA** support plans that are tailored to your business needs and budget.

Call Us Today!

(415) 344-0500 • (888) 648-0500

www.dbspecialists.com

Database Specialists

ORACLE CERTIFIED SOLUTION PARTNER

Fundamentals of Data Quality—Part II

By Michael Scofield

Michael Scofield

This is the second in a series of articles about data quality (DQ), including the techniques, politics, and tools involved in improving the quality of the data in an enterprise. We will not be pushing any tools or software. But we will be presenting general concepts and principles of managing data and its quality.

Data quality (the quality of data) is the result of what happens to data in production databases, and the business processes that capture data and pass that data along to the database. But DQ is also an issue for data residing outside of RDBMSs.

The quality of data can never be determined merely by reading the documentation or looking at a data model. (However, a data model that is severely out of sync with the business needs is probably going to contribute to collecting incorrect data.)

DBAs should be familiar with some basic characteristics of data quality (discussed in the previous installment of this series of articles) and some simple ways to test those characteristics in data. You don't need an expensive tool to do some simple tests of the quality of your data.

In a large, complex bureaucracy, one way to categorize data is either "at rest" (sitting in your DBMS overnight), or "in motion" (data flows from one place to another). Data flows include the importing of data from external sources. Imported data requires careful monitoring for data quality problems and unexpected changes in scope or changes in data architecture.

Data imported from external sources are often loaded to a data warehouse or even a production database—sometimes without being tested (gasp!). In either case, data surveillance for quality should be exercised prior to running any load utility or application. A thorough discussion of testing data flows is beyond the scope of this article, but many of the basic techniques are similar to those discussed below.

Your Existing Toolbox

Knowing how your production data is behaving requires a set of tests and reports, some of which are quite simple. Much of the data surveys you need to perform may be accomplished with query tools and reporting software that your shop probably already has. Even if you are contemplating the purchase of a specialized DQ tool, fourth-generation languages and off-the-shelf report writers may give you a head start in looking at your data while contemplating which DQ tool to purchase. Indeed, I recommend using existing tools for at least three months of intensive activity until you know what you expect data quality software to do.

Monitor Row Count Over Time

Perhaps the most fundamental measure of data behavior is at the table level—how many rows are there, and how does that tally change over time. While counting the rows in a table is a simple task, the current row count usually lacks meaning unless placed in some context—usually the history of similar data points back in time, or data from other tables that are logical peers (in the business sense).

Given the ability to store prior observations, a sudden increase or decrease in row count may warrant an investigation —particularly a sudden increase. Detecting such a "discontinuity" in this measure requires, of course, saving the historical data points.

Sometimes a significant increase in row count represents an expansion of the scope of the business—such as a merger with a peer company (although the DBA should be aware of such activities far enough in advance to plan for an increased need for space).

Graphing the periodic row-count observations certainly helps to understand what is going on. If we see the following chart, we have every reason to ask what happened in April.

A similar principle applies to importing data from external sources. Count the rows. If you consistently receive between 4,800 and 5,200 records, month to month, and suddenly encounter 8,900 records, you probably want to hold onto that data and talk to the source to find out why the row count went up so suddenly.

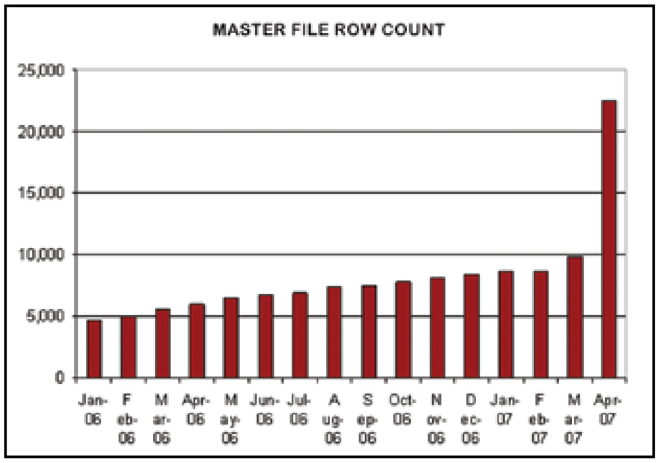


Figure 1—Monitoring the Row Count

Know What Values Are Found in Each Column, Independent of Context

It is amazing what simply counting the records by the value in any particular field will tell you. For example, if you have a field called STATUS_CODE in a customer master table, a simple tally of the records by that field may be quite revealing.

```
STATUS_CODE       RECCNT
-----------  -----------
A                  3,212
B                  4,115
C                  2,948
E                      2
K                      1
-----------    ---------
TOTAL             10,278
```

In this simple report, we see three codes (A, B, and C) that appear with high frequency—the result of normal business behavior. But we see two values (E and K) which appear so seldom that I would, at first, describe them as anomalies.

And of course, we should compare these values to what is in the data documentation (if it exists). If “E” and “K” are not mentioned in the documentation, then these might be either keying errors, or new, legitimate values that have not been mentioned in the (now obsolete) documentation. Or they may be entrepreneurial activity (on either the part of business people or “cowboy” programmer analysts, using the field for some other purpose).

This kind of field is what I call “low cardinality”—few values.

High-Cardinality Domain Study

Many columns contain too many distinct values to permit using the simple approach shown above. In such cases, we want to look at the value extremes. Some DQ tools provide the maximum and minimum amounts observed. If the field is an amount (numeric), they may also provide the mean or median. Those are often useful measures. But sometimes the extremes of the collating sequence reveal anomalies that are worth investigating. This is true for identifiers (codes) and text, as well as amount fields.

For example, if the largest prices observed on a price file are . . .

```
       PRICE   RECCNT
  ----------   ------
   4,010,000   1
       1,500   1
       1,477   1
       1,450   1
       1,425   3
       1,400   5
       1,375   2
```

. . . that largest amount is obviously an “outlier.” It has no neighbors in the value spectrum. It is at minimum an anomaly, and probably a keying error. The trailing zeros contribute to that suspicion of keying error.

Now, in such a case, you need to know what record contains this value. And at this time, it is useful to have some report-generator code ready to create a formatted dump. The code could be customized to show only the rows that are of interest—in this case, the anomaly and perhaps some “peer” records, or perhaps its nearest neighbors. (In this case we could select those records where PRICE is over a certain threshold amount.)

```
ID      DESCRIPTION          PRICE   CREATED   UPDATED   STATUS
------  -------------    ---------  --------  --------   ------
400127  1/4 CM WIDGET    4,010,000  02/12/04  02/12/04   A
401883  3/4 CM BOLT          1,500  11/22/07  11/23/07   G
397026  2.5 CM FLANGE        1,477  05/02/03  05/02/03   A
```

This formatted dump does not show every field on the record. But it does show the ID of the record of interest, a textual description that is meaningful to the business reader, the field and value being challenged, and the record create and record update dates. The latter two fields are particularly important in chasing down any “audit trail,” in case the value is deemed to be woefully unreasonable (an understatement bordering on a “known error”).

The same technique (counting records by value) can be employed for examining the sort extremes of text fields. For example, at the low end of the sort sequence for a customer name field, we see the following values:

```
CUST_NAME                         RECCNT
-----------------------------    ------
    CARSON ELECTRIC, INC.             1
 PENDERGAST & SONS                    1
AABEY RENTS - DOWNTOWN                1
AABEY AWNINGS COMPANY                 1
ABERJANIAN FOODS                      1
ACE DEVELOPMENT CORP.                 1
```

Here we see something which many readers will recognize as leading blanks. Probably accidentally keyed into some screen. This data error is not catastrophic. When used in a mailing label, the postal carrier can still read it. But if one is searching for it, one may not think to enter the leading spaces.

Sometimes the data anomalies occur in the most frequently occurring values. To find these, simply count the records by value and order (descending) by record count. Below, we see the high-end distribution a U.S. Social Security number collected on a customer master file.

```
SSAN            COUNT
-------------  ------
000000000       5,241
987654321         747
999999999          74
718453821          51
777777777          47
532784421          15
957264142          14
444444444          13
```

In looking at these numbers, it helps to understand the business (not just the technology). Social Security numbers have limited ranges. First of all, all zeros is not a valid SSN. It probably means missing data. One must understand how the “null” concept is expressed and recorded in this data table.

There can be many business reasons (good or bad) for no SSN being collected. Some customers may not be humans (but rather businesses with no SSN). Or customers may truly not have a SSN (although that is unusual in the U.S.). So the causes of those 5,241 records deserves more exploration.

Now, “987654321” is a curious number. Indeed, we should be suspicious. First of all, it is outside of the valid range of SSNs (as defined by the policy of the Social Security Administration). But the sequence suggests that it was keyed in by an eager customer service representative (CSR) to get to the next screen in some application. In other words, the customer didn’t want to provide a SSN, but an SSN is required for the application to proceed, so the CSR invents something. This sequence of numbers keys quickly.

The first "in-range" valid SSN is 718453821, and that occurs 51 times. This is not a natural behavior. It is probably a heavily shared number, perhaps many of the residents of a tenement share the same Social Security card when applying for jobs.

Sometimes, the least frequently occurring values are anomalies. We saw this in the STATUS_CODE example above. But for "high-cardinality" fields (like phone number area codes), we may still find anomalies in the rare values. By looking not at the *most* frequently used values but at the *least* frequently used values, we may find anomalies or keying errors.

```
AREA_CODE           COUNT
-------------     -------
213               248,492
415               138,803
lines omitted
682                 2,342
740                 1,492
733                     2
583                     1
466                     1
221                     1
759                     1
```

I sometimes call these "onezies and twozies." Research will show that the final five area codes listed in the above report (733 and below) are not valid nor currently used. So they must be data errors, probably keying errors.

Now, what to do about these errors is a larger business question. The fixes may require changes outside of IT. There may be some politics involved. But fixing processes, while important, is beyond the scope of this immediate article.

However it is important to see that simple reports like these do give visibility to anomalies that may reveal data quality problems. The first step in solving a problem is to know what the problem is.

Duplicate Keys

One of the cultural values of good data quality analysts is to be reluctant to assume that any aspect of the data is OK. An example is keys and unique identifiers. Is a key really unique? Many would not think to ask that.

Oh, how naïve! Test that assumption! In a flat file, the key or unique identifier is often at the left end of the layout, with a name that includes "ID" or "key." Count the number of records per key value, and order the results by the record count. If you get results like this, we know something is wrong. This key is not unique!

```
REC_KEY     COUNT
-------     -----
44203           5
04932           4
20483           3
39212           2
49932           2
00001           1
00002           1
00004           1
```

We see that five values each occur in multiple records. That could be a problem, or it may not be. It depends upon the business rules, and business expectations of how that field and table behaves.

Just as an aside, the gap between 00002 and 00004 says something about the data behavior also. If the IDs were issued sequentially, then records have probably been deleted since their first issuance. This is part of the table behavior that may be of interest to the data quality analyst.

Evaluating Inter-Column Logical Consistency

Sometimes data errors may be found through the context of logically adjacent data fields. There are often many data elements in a table that may constrain each other. A common example is city and zip code. Some large cities have multiple zip codes. Some zip codes can describe more than one community. So there is a many-to-many relationship, but anomalies can still be detected.

Below, in a customer master file we see several city names that are served by a common post office (with a single zip code).

```
ZIP    CITY NAME                   RECCNT
-----  ---------------------       ------
93940  DEL REY OAKS                     3
93940  MONTEREY                       932
93940  PRESIDIO OF MONTEREY            18
```

Here are several different place names sharing a common zip code. But anyone familiar with the Monterey area will realize that they are different communities (or facilities) within the same general area (and probably served by the same post office). There are many examples of this. Greenwich Village is really a neighborhood of New York City. Hollywood is a neighborhood of the city of Los Angeles.

But if we see the following distribution, we may have reason to be suspicious:

```
ZIP    CITY NAME          RECCNT
-----  --------------     ------
93901  SALINAS               473
93902  SALINAS             1,285
93902  SACRAMENTO              1
93905  SALINAS               482
93907  SALINAS                92
```

Sacramento is nowhere near Salinas. They are too far apart to have any shared digits (even the first three digits). And the record count of 1 suggests to me that it is at minimum, an anomaly, and more likely a keying error.

Cost-to-Find vs. Cost-to-Overlook Trade-Off

This article has looked at only a few of the many kinds of easy-to-create tests for data quality. Which test is appropriate for any particular field depends on the format and business meaning of the field.

People with experience in data quality can imagine lots of obscure kinds of errors to test for. But testing takes time and effort. Testing is a cost. And most data managers or DBAs do not have unlimited free time. So one must consider the cost of devising and running a test against the likelihood of the problem occurring, along with the cost of a data error not being discovered and repaired. It's a trade-off.

Coming Up Next

In the next installment, we will look at some of the functionalities of data quality tools on the market, and how to clearly define what requirements you have before going out to select an off-the-shelf DQ product. ▲

Michael Scofield is manager of data asset development at ESRI, Inc., in Redlands, Calif., and holds a faculty appointment—assistant professor of health information management—at Loma Linda University. He can be reached at **mscofield@esri.com**.

Enterprise Java Beans Redux

by Joel Thompson

Joel Thompson

This article first talks about the benefits of EJB3 technology, taking you beyond the marketing hype to explain in some detail why they can help your development effort. I'll also explain some of the problems with EJB2 and how it is simplified with EJB3, hopefully helping you to consider EJB technology. Then in the second part, I'll give you an example of how to get started with EJB3 in JDeveloper. The example that I work through is standard EJB3 and JSP page that will run in any container.

As a consultant, I work at a wide variety of businesses, from "mom and pop" to Enterprise Fortune 500-type businesses. Some of these businesses adopt EJB and others have rejected it. There is no across-the-board acceptance of EJB technology. The businesses that are not inclined to use EJBs are typically using a servlet-based approach such as Servlets, JSP, Spring, Struts, or JSF. Granted, they could be using EJB as their backend services layer, but they typically are not. They think their software development will be "simpler" with a simple web tier approach. Some businesses have a good sense of Model 2 (aka MVC) architecture, meaning that they separate out all the Java/SQL from their front-end presentation layer, which is a good thing. If not, then they'd likely have a good case of spaghetti code, and nightmare bugs to confront. However, this discipline is at the cost of comprehensive documentation and heavy-handed monitoring to make sure the programmers are writing code according to these proprietary standards. When asked why they haven't adopted EJB, the most common answer is that they don't want to change. After all, "if it ain't broke, don't fix it." Another reason given is that EJBs are slow or difficult to manage. One of my favorites is, "they are too heavyweight for our solution." Some honestly say that they don't want to adopt EJBs due to lack of EJB knowledge. The lack of EJB knowledge may be what gives people the "heavyweight" opinion as well.

I'll briefly explain the latest EJB3 (versus EJB 2) technology in an attempt to dispel some of these concerns, and give you some good reasons to adopt EJB3 into your software solution (and even re-engineer your existing code). Let's start with listing and briefly explaining some of the benefits:

- MVC design
- Component architecture
- Web services (SOA)
- Different clients
- Service layer/business logic
- Integration with legacy systems
- Caching
- Open systems
- High performance
- Framework
- Portability
- Distributed computing
- Clustering
- Pooling/multithreading
- Transaction management
- Security
- Failover
- Messaging

I won't cover all these topics in this article; however, it is worthwhile noting that most people would not want to develop all these features into a "simple" dynamic-driven website. We'd rather focus on business logic and presentation. If you're developing a web solution that is beyond the bare minimum, then it is most likely that you'll need to consider at least a few of these features, and even more of them as your website grows.

That's exactly why I promote using the EJB framework from the start, even for simple database-driven websites. So what about that argument that EJB is too heavyweight? I'll assume they are talking about two things when they say "heavyweight": 1) It is too difficult to configure (XML file) and deploy EJB, and 2) we don't need all these EJB services. The configuration concern is true for EJB2, but with EJB3's annotations, EJB development is a breeze. You simply create the class and put in a simple annotation or two, and you have defined your EJB. That is a huge difference from EJB2, where you had to create and configure complex XML files. Near the end of this article, you'll see firsthand how simple it is to create a stateless session bean and invoke it. The second aspect of the heavyweight issue can be mitigated by considering two key benefits: 1) You may need these services someday and you would create a better architecture if you consider them now, rather than patching them in later and possibly disrupting your code base; and 2) the EJB framework will provide for self-regulating and self-documenting aspects of your code. This is explained next, as I discuss the practical side of software development and code management using EJB versus in-house standards.

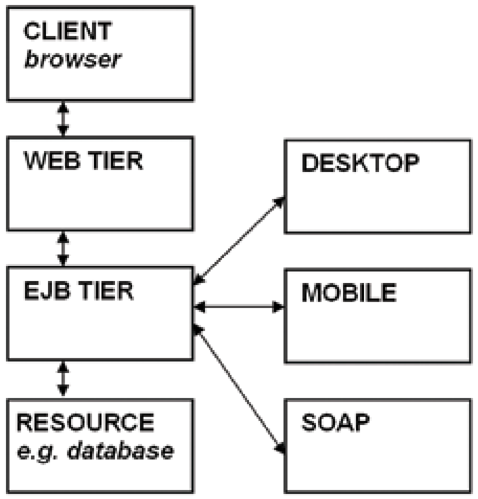


Figure 1—Many clients benefit from your EJB tier.

No matter how you look at your software development, you should still document a well-thought-out design with coding standards, considering issues like flow, sequences, and classes and function naming conventions (the signatures of the method). It is best to use UML notation for this. However, when using EJB architecture a lot of the basic design, like layering your software, is determined by the fact that you are using this architecture. You see, EJB is a "component architecture" and this forces the developer to layer the application using the Model View Controller (MVC) pattern, such that the presentation (or web tier) is clearly separate from the business logic or services layer, where the components live. This is not so of a servlet-type approach. You would have to define this in your design documentation, and then educate/monitor/manage your programmers to make sure they conform. This is important because: 1) Your developers may be here today and gone tomorrow—or worse, you may be offshoring your software development—and when you receive all the code to maintain yourself, you have to figure out the basic architecture of these "home-brewed" systems; 2) If the code is forced to adhere to the EJB standards, it will have fewer bugs related to "interpretation" of the semantics of your classes. As an example, if you call a stateless session bean (SLSB), you know simply because it is an SLSB that the bean will not save any state between method calls, and that all service functionality of this SLSB must be encapsulated in the method's arguments and returned object. Additionally, you also know that SLSB returned objects (i.e., a Person class), will not have any tightly coupled resources from the EJB side. This is due to the requirement that all arguments and return types must be serializable. For example, if you are returning a JPA-managed object (Java's new Persistence API), then you'll quickly discover that related objects modeled as attributes of your returned object may not be accessible, and you're likely to get a "Lazy Initialization Exception."

If you architect your solution with EJB from the beginning, then issues like high performance, clustering, and caching would be a natural progression as the site's demands increase. For instance, you would want to cache most if not all of your business data in stateful session beans and none in your web tier (except reference data). You would also want to keep your stateless session beans as lean as possible and handle the bulk of your incoming requests for things like queries. This way if you decided to cluster at the EJB tier by adding another EJB server, EJB would take care of all the synchronization that you'd need. If you had to write this from scratch, you'd probably spend a significant amount of time on this system-level detail, and it would not be as robust and stable as commercial products like OC4J.

There are a lot of other benefits to using EJB, such as reusing your components to service other clients (besides the web tier) like a mobile device or a desktop program. You can also provide web services end points, cluster your servers such that failover or load distribution is enabled, and use built-in transaction management, security, and distributed programming. It is worth mentioning that you can define an EJB service layer and then have it work with Oracle's ADF components on the front end. One last thing before we move on to the example: entity beans are no longer a part of EJB. JPA is the new persistence tier, which gives us persistence in or out of an EJB environment. For instance, you can use JPA in your Swing desktop program. I'd like to add that the performance issues often globally attributed to the EJB environment are mainly a result of improper use of entity beans in EJB2. With the new JPA persistence, you have the ability to swap out persistence providers like Toplink, Hibernate, and JDO, and you have better control over your entities, which addresses the performance issues; of course, you still need to consider caching and proper entity/query design.

Create the Model Layer and Define an EJB

Let's see an example of how easy it is to use the new EJB3 technology. Start by firing up JDeveloper 10.1.3 to create an application and a project. I'll name my project "Model" by Oracle's normal convention. In order to include the proper libraries for EJB, you must right-click the project, select "Properties," and choose EJB as a technology, as shown in Figure 2a.

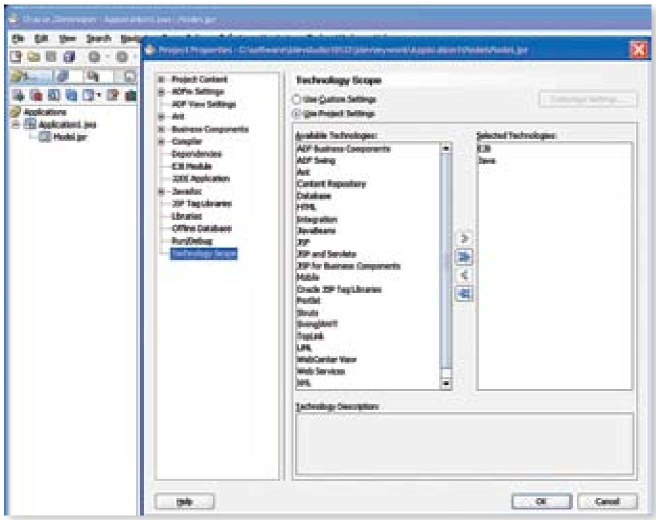
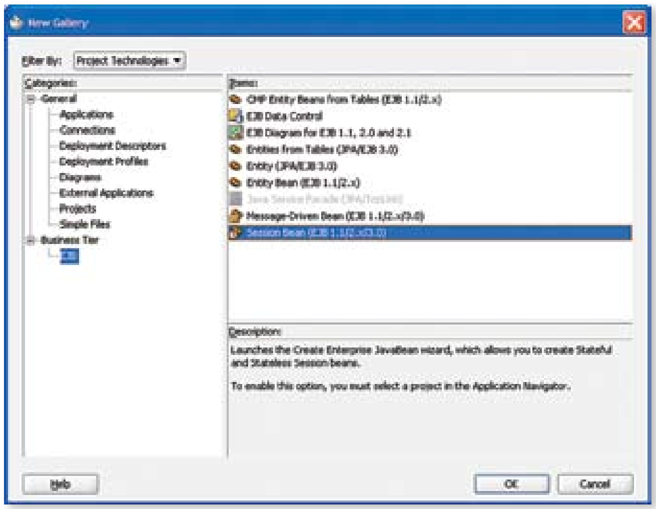


Figure 2a—Create the Model Layer.

Next you need to right click your model and select "New . . . ," and then select EJB as session bean, as shown in Figure 2b.


New Gallery
Filter By: Project Technologies
Categories:
General
Applications
Connections
Deployment Descriptors
Deployment Profiles
Diagrams
External Applications
Projects
Simple Files
Business Tier
EJB
Items:
CMP Entity Beans from Tables (EJB 1.1/2.x)
EJB Data Control
EJB Diagram for EJB 1.1, 2.0 and 2.1
Entities from Tables (JPA/EJB 3.0)
Entity (JPA/EJB 3.0)
Entity Bean (EJB 1.1/2.x)
Java Service Facade (JPA/TopLink)
Message-Driven Bean (EJB 1.1/2.x/3.0)
Session Bean (EJB 1.1/2.x/3.0)
Description:
Help
OK
Cancel

Launches the Create Enterprise JavaBean wizard, which allows you to create Stateful and Stateless Session beans.

To enable this option, you must select a project in the Application Navigator.

Figure 2b—Create the Model Layer.

Walk through the wizard, keeping all defaults (Name: Manager, Session Type: Stateless, Transaction Type: Container). We won't check box the Generate Session Façade Methods, since we don't have any entities in our model. If we previously defined JPA entities to this "model" project, then checking this option would define some methods that expose create, delete, and query operations for the EJB service. Check box the local and remote interfaces but not the "Include Web Service Endpoint Interface." By checking "Local" and "Remote," this wizard will create these two interfaces. These will be important if you want to expose your service locally, i.e., within the same EJB/web servlet container (which is typical), or you'd like to expose this service remotely to other Java containers, like a Swing desktop program. The key difference is that the client Java container is the same for local (or remote), and different for remote.

When you're finished with the wizard, you should get three files: Manager.java, ManagerBean.java, and ManagerLocal.java.

Let's create a simple function called "businessLogic1" that takes a person's name and returns a new string to say hello.

```
package model;

import javax.ejb.Stateless;

//"Manager" is the name of this EJB, that you'll use to
//lookup from the client side.
@Stateless(name="Manager")
public class ManagerBean implements Manager, ManagerLocal {
    public ManagerBean() {
    }

    /*
    * defined your business service layer, than can be
    * used by many different clients, like
    * Swing desktop programs
    * and web clients.  Same code servicing two different
    * architectures.
    */
    public String businessLogic1(String name) {
        return "hello " + name;
    }
}
```

You need to add the method's signature to the local and remote interfaces. The local and remote interfaces work within the same container. You'll need a remote interface if you want to make calls from a client program to another EJB container.

```
package model;
import javax.ejb.Remote;

@Remote
public interface Manager {
    String businessLogic1(String name);
}

package model;
import javax.ejb.Local;

@Local
public interface ManagerLocal {
    String businessLogic1(String name);
}
```

Define a Client (Web Project)

Select the application, and right-click to create a new project (select "New . . . " and then "Web Project"). Arbitrarily name this project "Viewcontroller," keeping with Oracle's naming conventions. Walk through the wizard selecting the defaults. Notice: "Servlet 2.4\JSP 2.0" and the Context Root "Application1-Viewcontroller-context-root." You want to know the version of your container, so you can avoid putting in features that won't run with this version, and the Context Root defines part of the URL to access your web page.

You need to configure your Viewcontroller to depend on the EJB layer. Right-click your Viewcontroller project and select "Properties." Then select "Dependencies," and check box the Model. This will set up the Model to be in the classpath of the Viewcontroller, such that we can access the EJBs.

Create a simple JSP page, and invoke the Manager EJB's method. Right-click your Viewcontroller project, and select "New . . . " and then "JSP page." Name this page "page1.jsp," and select the defaults for the rest of the wizard options.

Select the Source tab at the bottom of the page1.jsp, and then type this code to call your EJB:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1252"%>
<%@ page import="javax.naming.*" %>
<%@ page import="model.*" %>
<html>
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=windows-1252"/>
    <title>page1</title>
  </head>
  <body>
<%
    InitialContext ctx=null;
    String result="";
    try {
        ctx = new InitialContext();
        Manager  mgr;
        //this is the remote interface
        String mgrslsb="Manager";

        mgr = (Manager)ctx.lookup(mgrslsb);
        result=mgr.businessLogic1("Joel");

    } catch (Exception e) {
        System.out.println("Exception" +
        e.getMessage());
        e.printStackTrace(System.out);
    }
%>
  Results:  <%=result%>
  </body>
</html>
```

Next, in order to test this page, right-click your "page1.jsp" file within JDeveloper and select "Debug." JDeveloper will start up an embedded OC4J server and automatically bundle and deploy this application. JDeveloper will also invoke your default browser and pass the URL to the browser.

It can be tricky configuring Oracle's JNDI naming for standalone OC4J servers and remote clients. However, once you have your OC4J server up and running and have deployed your EJB's, you can access them from any Java client, which is a big plus in your Enterprise solution. If you get stuck, go to otn.oracle.com and check out the forums or refer to the documentation listed below.

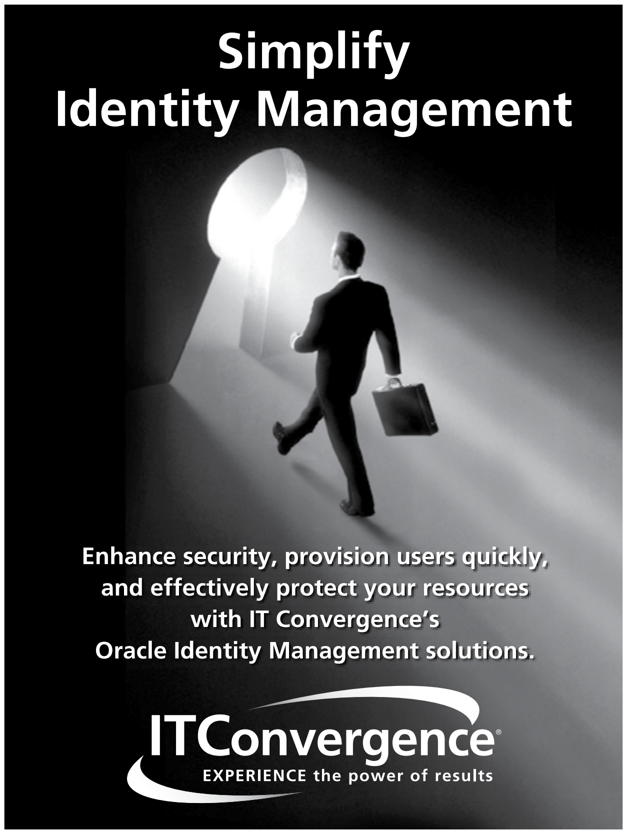
It is worth noting that this sample application can deploy in JBoss as well. All you need to do is create a JAR of the EJBs, WAR up your Viewcontroller project, then package everything together in an EAR and copy the EAR file to JBoss's deployment directory.

One thing about JBoss is that you'd have to change the mgrslsb variable to be mgrslsb="earfile/Manager/remote", where "earfile" is the name of the ear file that you are deploying.

Concluding Remarks

In conclusion, we covered some reasons why EJBs are important to your software development: basically, it forces you to implement an MVC design pattern, it opens up the implementation of your business logic to other clients besides web servers, and with EJB3 it is a lot easier to create EJBs than it was with its EJB2 predecessor. With EJB3 annotations, all you have to do is to use a few annotation tags and then bundle and deploy your application. There is no more need for complex XML configuration of the EJBs. Additionally, EJB EntityBeans no longer exist and are now separate layer/API, called JPA, which you can swap out with your favorite persistence providers such as Toplink, Hibernate, or JDO. I hope you found the example helpful, and you can see for yourself that building EJBs into your software solution from the beginning will help in development and with future growth of your software needs. If you need any help in walking through the above example or if you have any questions, please feel free to email me and I'll do my best to help. You can also find more information at **www.oracle.com/technology/documentation/appserver.html**. ▲

Joel Thompson is president of RHINO Systems Inc., a software consulting firm serving the greater Sacramento and Bay Area. He can be reached at www.rhinosystems.com, and via email at **joel@rhinosystems.com**. *He has programmed in Java/J2EE since 1997, and prior to that worked as senior programmer and development manager at Oracle Corporation since 1989. He resides in Auburn, CA, with his wife and three children, and enjoys snowboarding, jogging, tennis, and many other athletic activities.*


Simplify
Identity Management
Enhance security, provision users quickly, and effectively protect your resources with IT Convergence's Oracle Identity Management solutions.
ITConvergence
EXPERIENCE the power of results

Contact us today: **www.itconvergence.com**

—SAVE THE DATES—

Summer conference:
August 21, 2008 at Chevron in San Ramon

Fall conference:
November 13, 2008 at CarrAmerica Conference Center in Pleasanton

Win Every Time with the Right Tools.

Whatever the job, there's a Toad® that fits.

Whether your team needs to quickly build great code, make database administration a snap or easily access data wherever it resides, Quest makes sure the deck's always stacked in your favor.

QUEST SOFTWARE®

Find the Toad that's right for your team.
Visit www.quest.com/cards

Many Thanks to Our Sponsors

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low-cost memberships. If your company is able to offer sponsorship at any level, please contact NoCOUG's president, Roger Schrag, at **www.nocoug.org/contact_us.html?recipient=roger**. ▲

Long-term event sponsorship:

- LOCKHEED MARTIN
- CHEVRON
- ORACLE CORP.
- PG&E

Thank you! Year 2008 Gold Vendors:

- ➤ BEZ Systems
- ➤ Confio Software
- ➤ Database Specialists, Inc.
- ➤ Embarcadero Technologies
- ➤ IT Convergence
- ➤ Network Appliance
- ➤ Princeton Softech
- ➤ Quest Software
- ➤ Roundstone Systems

For information about our Gold Vendor Program, contact the NoCOUG vendor coordinator via email at: **vendor_coordinator@nocoug.org**.

$ TREASURER'S REPORT

Jen Hong, *Treasurer*

Beginning Balance		
January 1, 2008		**$ 42,548.10**
Revenue		
Membership Dues	16,836.82	
Meeting Fees	200.00	
Vendor Receipts	2,500.00	
Advertising Fee	400.00	
Training Day	–	
Sponsorship	–	
Interest	59.89	
Total Revenue		**$ 19,996.71**
Expenses		
Regional Meeting	9,071.14	
Journal	6,078.98	
Membership	237.08	
Administration	–	
Website	–	
Board Meeting	586.44	
Marketing	365.11	
Insurance	–	
Vendors	14.80	
P.O. Box	–	
Training Day	205.00	
Miscellaneous	20.00	
Total Expenses		**$ 16,558.55**
Ending Balance		
March 31, 2008		**$ 45,986.26**

NoCOUG Spring Conference

Session Descriptions

For more detailed descriptions and up-to-date information, see **www.nocoug.org**.

—Keynote—

How Oracle Came to Rule the Database World

Rich Niemiec, *TUSC*. 9:30–10:30

This is a break from the technical presentations, but not too much of a break. Sit back and listen to the history of the relational database. Find out the crucial moves that Oracle made at critical junctures of its history. See what drove the product from inception, over the rocky road, and eventually to the top of the mountain. Learn what made Oracle, the product, a success, but also find out the attributes that made Oracle, the company, a font of technological wizardry. This talk will reveal several seldom-heard facts and some unknown secrets of Oracle's success.

I. The paper that started it all—E. F. Codd
II. System R and Ingres
III. Oracle is founded as SDL
IV. Version 1 to Version 10*g*
V. Why did Oracle win?
VI. Future market direction
VII. Summary

—Columbus—

The Best Oracle Database 11*g* New Features

Rich Niemiec, *TUSC*. 11:00–12:00

This presentation will look at which 11*g* new features should be investigated for use. Most of the features that will be covered will be related to the DBA, but there will also be a few outside that realm. There will be simple examples to show the basic functionality of the following new features:

➤ Memory Target
➤ Partition Advisor
➤ Security Enhancements
➤ DDL Lock Timeout
➤ The Invisible Index
➤ Automatic Diagnostics Repository
➤ SQL Plan Management
➤ Workload Capture and Replay
➤ SQL Repair Advisor
➤ ADDM Enhancements
➤ Interval Partitioning
➤ Optimizer Enhancements

Richard J. Niemiec is chief executive officer of TUSC and author of Oracle 9*i* Performance Tuning Tips & Techniques. *In 2001, Rich was named by Oracle Corp. as an Oracle Certified Master—one of the first six so recognized around the world. Rich is former president of the International Oracle Users Group and currently serves as vice president for its Real Application Clusters special interest group. In addition, Rich is the current president of the Midwest Oracle Users Group.*

Oracle Database Security in a Nutshell

Daniel Liu, *First American Real Estate Solutions* 1:00–2:00

In this seminar, students will learn how they can use Oracle database features to meet the security and compliance requirements of their organization. The current regulatory environment of the Sarbanes-Oxley Act, HIPAA, the UK Data Protection Act, and others requires better security at the database level. The session provides suggested architectures for common problems. It covers security features of the database, including auditing, column and file encryption, virtual private database, label security, enterprise user security, and more.

Daniel Liu is a senior technical manager at First American Real Estate Solutions in Santa Ana, California, and co-author of Oracle Database 10*g* New Features: Oracle10*g* Reference for Advanced Tuning & Administration, *from Rampant TechPress. His expertise includes Oracle database administration, performance tuning, Oracle networking, and Oracle Application Server. Daniel has contributed to DBAzine,* Oracle Internals, *Oracle Technology Network, and* SELECT Journal. *He received the* SELECT *Editorial Award for Best Article in 2001 and was named Architect of the Week by the Oracle Technology Network in 2004. He has also given presentations at IOUG-A Live, LAOUG, OCOUG, NoCOUG, TOUG, OOUG, Wilshire MetaData conference, and Oracle OpenWorld. Daniel has served as a panelist on Oracles of Oracle at Oracle World and IOUG-Live.*

Oracle Archiving—Best Practices

Dave Moore, *Neon Enterprise Software*. 2:30–3:30

The subject of data archiving is never addressed until the data retention problem is serious. Requirements for archiving range from legal obligations for data retention (Sarbanes Oxley, HIPAA, and so on) to operational performance to internal business requirements. What data should you archive? How often? How should you do it? And what needs to be done inside of Oracle to regain operational efficiency? Whatever needs are driving your business archive initiatives, join Dave Moore to determine archiving best practices. This informative session will describe how you can meet your Oracle database archiving requirements while simultaneously optimizing database performance.

Dave Moore is the author of Oracle Utilities, *from Rampant TechPress. He was the solution architect at BMC for Oracle and SQL Server performance products, including DBXray, Space Expert, SQL Explorer, and SmartDBA Cockpit. Dave has presented numerous times at IOUG and Oracle OpenWorld, as well as at many local user groups, and is a frequent contributor to* Oracle Professional *and DBAZine.com.*

How Independent Software Companies are Leveraging Oracle Embedded Products in Their Software Applications

Gabe Stanek, *Oracle Corporation*. 4:00–5:00

The market for embedded database software continues to experience significant growth, based on a highly diverse range of use cases, including consumer and mobile devices, desktop and enterprise software, large-scale networking, and storage equipment and appliances. Oracle offers the industry's broadest portfolio of world-class embeddable database products, which ranges from Oracle TimesTen for real-time, in-memory relational data management and caching to Oracle Berkeley DB for high-performance, non-relational data management, and Oracle Database Lite for online/offline mobile data management. Requiring virtually no human administration, these products are ideal for developers in industries such as telecommunications and high technology, which have demanding requirements for intelligent edge devices and services.

Skip Morehead is a team lead sales consultant at Oracle Corp.

—Drake I—

Building the Technology Stack for Modern Applications

Caleb Small, *Camosun College*. 11:00–12:00

This session offers an overview of building clustered Oracle Application Servers, clustered RAC database servers, clustered NetApp storage arrays, and clustered load balancers to deploy Java or forms and reports applications in a high-availability environment.

Caleb Small is the primary instructor for the Oracle Workforce Development program at Camosun College in Victoria, British Columbia, and a private consultant with years of experience implementing and teaching Oracle throughout North America. He has over 30 years' experience in the IT industry in numerous business areas, including both the public and private sectors. Caleb is also active as a director of the Victoria and Puget Sound Oracle User Groups, and has delivered numerous lectures, presentations, live demos, and articles to other groups. He has authored and led many of the PSOUG workshops, including RAC, ASM, RMAN and Data Guard.

Poor Man's Auditing with Oracle LogMiner

Caleb Small, *Camosun College*. 1:00–2:00

The need for database auditing is a topic that most DBAs have heard loud and clear. Oracle offers many auditing solutions, and the specific requirements of each individual application must be assessed before choosing. One of the simplest, yet most powerful solutions is the time-tested Oracle LogMiner. LogMiner allows reconstruction of past SQL statements from the online and archived redo logs, provided that some simple configuration requirements are met. While there is some additional overhead imposed on the database, for the most part all the necessary mechanisms are already in place and no additional audit trails or log files need to be maintained. This presentation steps through the decision criteria for selecting this option, along with the actual database implementation steps, based on an actual production system.

Caleb Small is the primary instructor for the Oracle Workforce Development program at Camosun College in Victoria, British Columbia, and a private consultant with years of experience implementing and teaching Oracle throughout North America. He has over 30 years' experience in the IT industry in numerous business areas, including both the public and private sectors.

ROUNDSTONE

ROCK SOLID RELATIONSHIPS

Roundstone Systems is Northern California's HP Gold Business Partner authorized in HP's Oracle Solution Network Program.

HP, **Oracle** and **Roundstone Systems** have joined forces to help you reap the benefits of IT consolidation.

HP BladeSystem Servers:

- Increase agility - get products to market faster
- Reduce costs - attain higher system utilization, share resources, reduce management and maintenance
- Increase the overall effectiveness of computing systems and networks

510-764-2080

Gold Business Partner

www.roundstonesystems.com

Enterprise data. Aligned.

You dedicate huge resources to enterprise applications like Oracle®, PeopleSoft®, JD Edwards®, Siebel®. And every year they grow in size and complexity — increasing maintenance challenges. Now you can take back control with Princeton Softech Optim™. Optim gives you the ability to align application data management to performance goals. Optimizing results, mitigating risk and controlling the cost of every IT investment — servers, software, storage, networks and people. Is this heaven? No, it's the power of Enterprise Data Management.

Learn more at princetonsoftech.com.

princeton softech

The Power of Enterprise Data Management

Caleb is also active as a director of the Victoria and Puget Sound Oracle User Groups, and has delivered numerous lectures, presentations, live demos, and articles to other groups. He has authored and led many of the PSOUG workshops, including RAC, ASM, RMAN, and Data Guard.

Web 2.0 Ajax-Based User Interfaces Development Made Simple

Shay Shmeltzer, *Oracle Corporation* 2:30–3:30

The Web 2.0 generation of applications has brought a new user experience to web-based applications. Ajax-based UIs are much more dynamic and offer better usability. This session explains how the new Oracle Application Development Framework (Oracle ADF) Faces rich-client components drastically simplify the development of such advanced UIs. Learn about the new components, how they work, and how they can influence the design of your user interface.

Shay Shmeltzer is a principal product manager in the Oracle JDeveloper group. He has been working with Oracle's development tools since 1990, using everything from Forms 2.0 and Oracle 5.0 to the latest things you can get from OTN. He has been with Oracle Corporation since 1993 in various roles spanning development, sales, consulting, and even marketing.

Hacking and Defending Databases

Todd DeSantis, Sentrigo. 4:00–5:00

This session will present a new angle on a popular attack vector on databases: SQL injection. We will describe types and techniques of SQL injection attacks on both Oracle-based web applications and built-in database stored program units, and show how simple SQL injection can be used to own the database server through the means of privilege escalation. We will also list ways of preventing SQL injection attacks ranging from secure coding practices to various external tools that will alert and prevent SQL injection attempts, and demonstrate how hacker techniques of evasion can be used to subvert them. Finally, we will introduce new deep inspection tools for Oracle 9*i*/10*g* that can prevent SQL injection, even in zero-day scenarios.

Take away points:

- ➤ How SQL injection attacks work
- ➤ Secure coding practices
- ➤ Existing tools for SQL injection prevention and techniques to evade them
- ➤ New resilient technologies used to solve SQL injections entirely, even those exploiting zero-day vulnerabilities.

Todd DeSantis is a technical pre-sales consultant at Sentrigo.

—Drake II—

Natural Data Clustering: Why Nested Loops Win So Often

Dan Tow, *Singing SQL* 11:00–12:00

This session will present a mix of empirical data and theoretical explanation to demonstrate why optimizers tend to favor hash joins, while experienced SQL tuners so often find they can do better by forcing nested-loops joins. The key is co-clustering—the tendency of well-clustered rows in one table to join to well-clustered rows in another table, a tendency that favors nested-loops joins.

Dan Tow has 18 years of experience fully focused on performance and tuning, beginning at Oracle Corporation from 1989 to 1998, followed by TenFold Corporation from 1998 to late 2002 and success as an independent consultant under his SingingSQL banner. He invented a systematic, patented method (U.S. Patent #5761654) to tune any query efficiently, which he has taught (or at least introduced) directly to over 1600 people, and this method is extended and elaborated in a course he teaches and in his book, SQL Tuning, *published by O'Reilly. Dan lives in Palo Alto, Calif., and is reachable at* **dan-tow@singingsql.com**. *You can view his website at www.singingsql.com, where you can also find his complete resume.*

Getting SQL Right the First Try

Editor's Pick

Dan Tow, *Singing SQL* 1:00–2:00

Most SQL tuning material addresses the question of how to fix performance of slow SQL. However, there are little-known rules that enable developers to avoid most tuning problems proactively, while also avoiding common functional mistakes and writing SQL that is far easier to understand and to maintain when it does need later work. This presentation proposes rules and processes to write correct and fast SQL from the beginning.

Dan Tow has 18 years of experience fully focused on performance and tuning, beginning at Oracle Corporation from 1989 to 1998, followed by TenFold Corporation from 1998 to late 2002 and success as an independent consultant under his SingingSQL banner. He invented a systematic, patented method (U.S. Patent #5761654) to tune any query efficiently, which he has taught (or at least introduced) directly to over 1600 people, and this method is extended and elaborated in a course he teaches and in his book, SQL Tuning, *published by O'Reilly. Dan lives in Palo Alto, Calif., and is reachable at* **dan-tow@singingsql.com**. *You can view his website at www.singingsql.com, where you can also find his complete resume.*

RAC 11g, Virtualization and More

Vijay Ragunathan 2:30–3:30

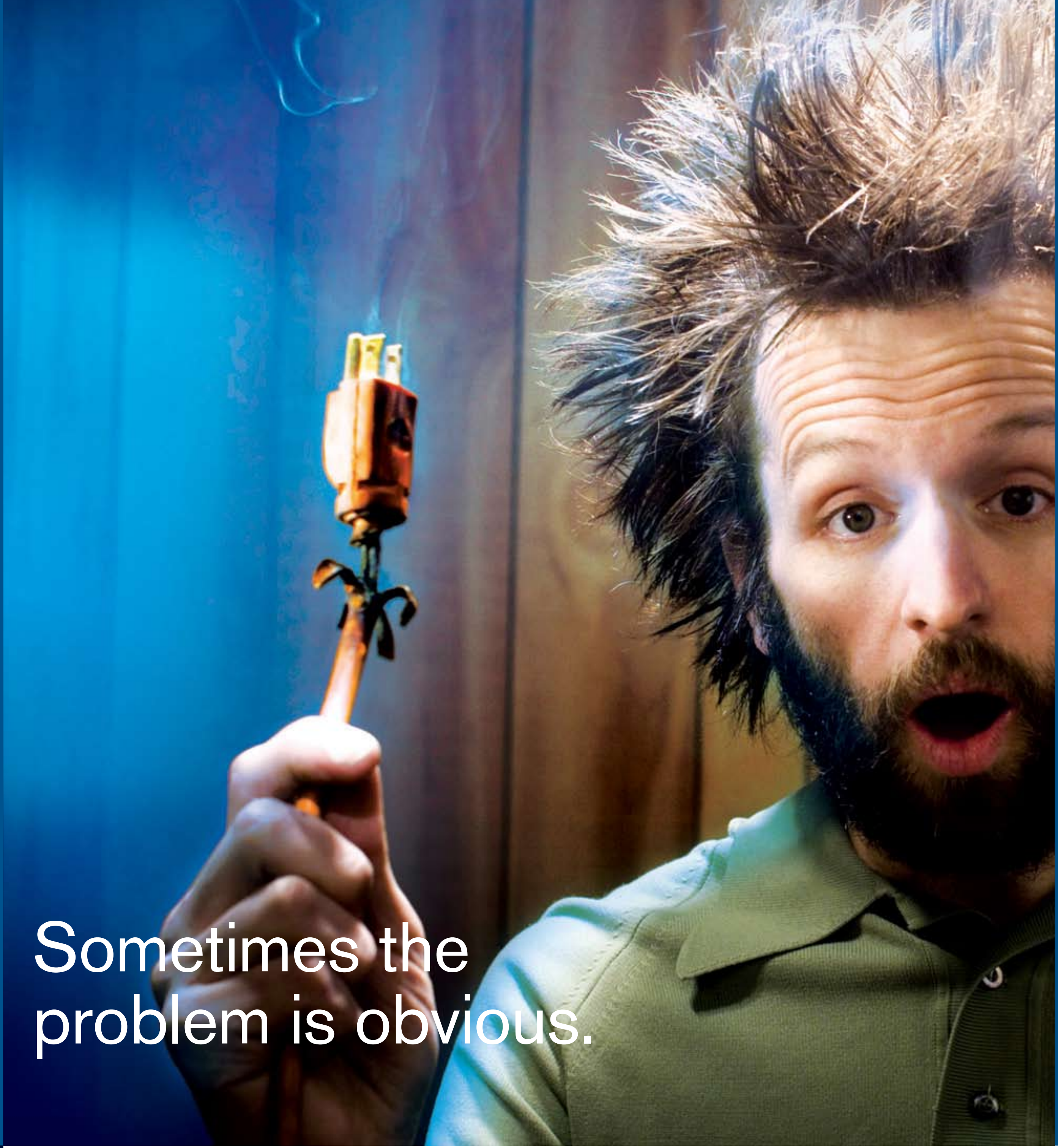
Virtualization software like VMware and Xen are changing the commodity hardware world. One of the benefits of this virtualization software is to be able to run Oracle RAC with more than one node with limited cost for development and testing environments. We will talk about on how to set up RAC 11*g* with this software. In addition, we will also talk about some 11*g* features related to RAC, ASM, and AWR/ADDM and cache fusion tracing.

The Evolving Web UI

Avrom Roy-Faderman, *Oracle Corporation* 4:00–5:00

Applications on the Web have come full circle—from highly interactive applets, through request/response-cycle based JSP applications, through richer applications through asynchronous server calls, and back to highly interactive Web 2.0 applications. This talk covers the current state of the art in web applications, with a focus on developing these applications in Oracle JDeveloper 11g.

Avrom Roy-Faderman is a principal technical writer in the Application Development Tools division at Oracle Corporation. He is the co-author, with Paul Dorsey and Peter Koletzke, of two books: Oracle9i JDeveloper Handbook *and, more recently,* Oracle JDeveloper 10g Handbook, *both from Osborne/McGraw-Hill and Oracle Press.*



Usually, it's harder to pinpoint.

Amazing what you can accomplish once you have the information you need.

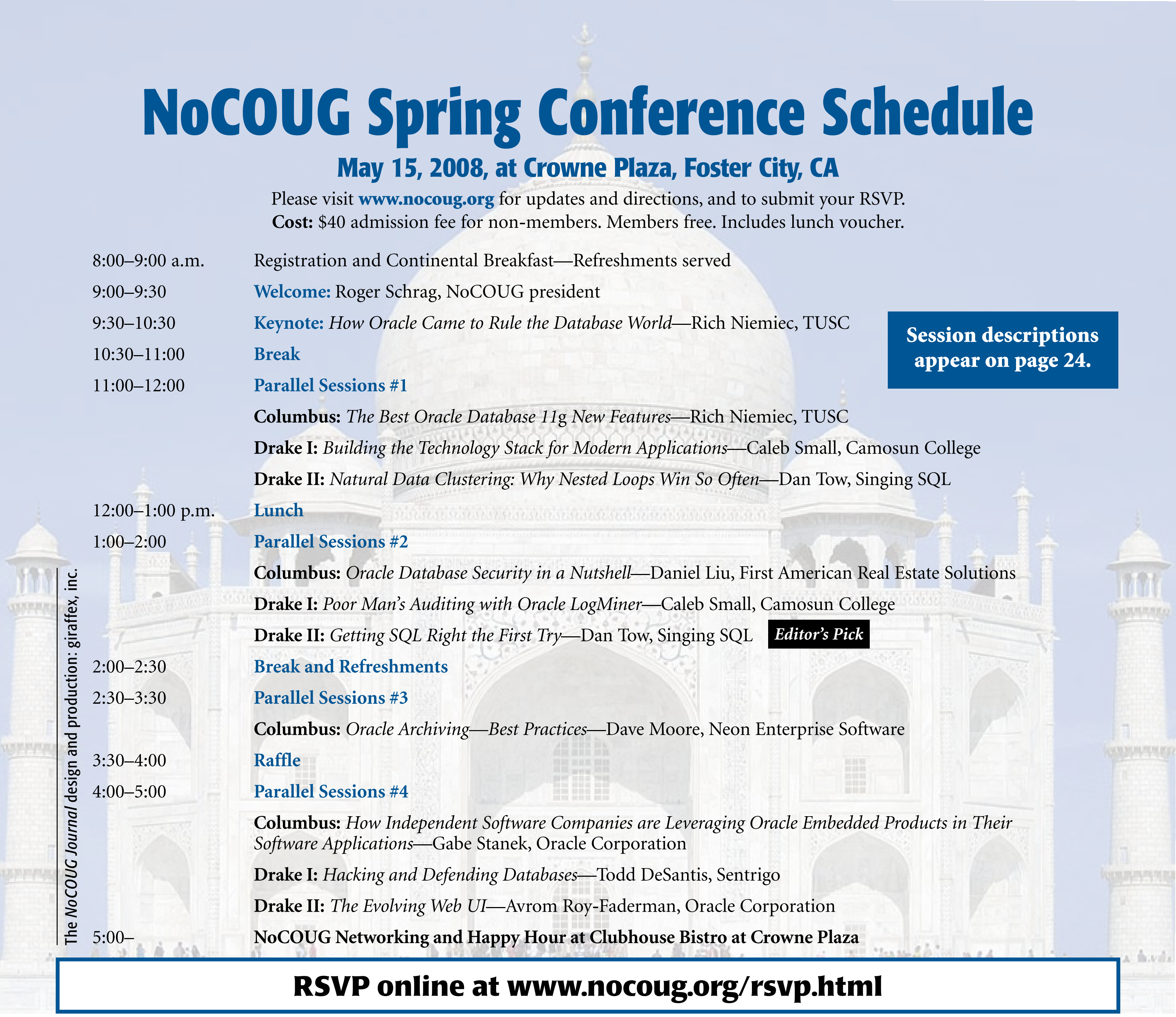
When the source of a database-driven application slowdown isn't immediately obvious, try a tool that can get you up to speed. One that pinpoints database bottlenecks and calculates application wait time ***at each step***. Confio lets you unravel slowdowns at the database level with no installed agents. And solving problems where they exist costs a ***tenth*** of working around it by adding new server CPU's. Now that's a vision that can take you places.

A smarter solution makes everyone look brilliant.

CONFIO SOFTWARE

Download your FREE trial of Confio Ignite™ at www.confio.com/obvious

Download our FREE whitepaper by visiting www.oraclewhitepapers.com/listc/confio



NoCOUG Spring Conference Schedule

May 15, 2008, at Crowne Plaza, Foster City, CA

Please visit **www.nocoug.org** for updates and directions, and to submit your RSVP.
Cost: $40 admission fee for non-members. Members free. Includes lunch voucher.

Session descriptions appear on page 24.

8:00–9:00 a.m.	Registration and Continental Breakfast—Refreshments served
9:00–9:30	**Welcome:** Roger Schrag, NoCOUG president
9:30–10:30	**Keynote:** *How Oracle Came to Rule the Database World*—Rich Niemiec, TUSC
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Columbus: *The Best Oracle Database 11g New Features*—Rich Niemiec, TUSC
	Drake I: *Building the Technology Stack for Modern Applications*—Caleb Small, Camosun College
	Drake II: *Natural Data Clustering: Why Nested Loops Win So Often*—Dan Tow, Singing SQL
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Columbus: *Oracle Database Security in a Nutshell*—Daniel Liu, First American Real Estate Solutions
	Drake I: *Poor Man's Auditing with Oracle LogMiner*—Caleb Small, Camosun College
	Drake II: *Getting SQL Right the First Try*—Dan Tow, Singing SQL ***Editor's Pick***
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Columbus: *Oracle Archiving—Best Practices*—Dave Moore, Neon Enterprise Software
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Columbus: *How Independent Software Companies are Leveraging Oracle Embedded Products in Their Software Applications*—Gabe Stanek, Oracle Corporation
	Drake I: *Hacking and Defending Databases*—Todd DeSantis, Sentrigo
	Drake II: *The Evolving Web UI*—Avrom Roy-Faderman, Oracle Corporation
5:00–	**NoCOUG Networking and Happy Hour at Clubhouse Bistro at Crowne Plaza**

The *NoCOUG Journal* design and production: giraffex, inc.

RSVP online at www.nocoug.org/rsvp.html

NoCOUG
P.O. Box 3282
Danville, CA 94526

RETURN SERVICE REQUESTED