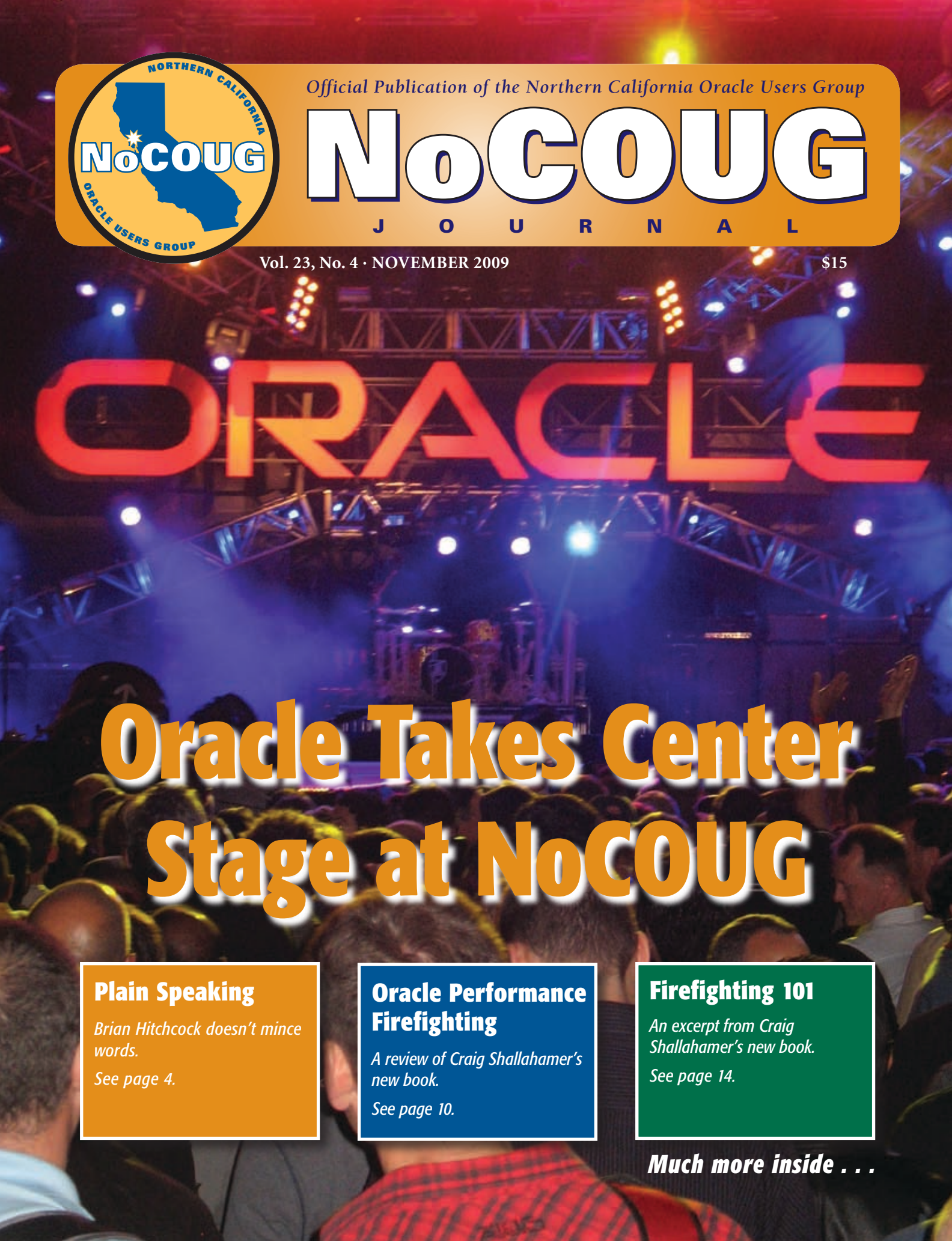
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 23, No. 4 · NOVEMBER 2009 $15



Oracle Takes Center Stage at NoCOUG

Plain Speaking

Brian Hitchcock doesn't mince words.

See page 4.

Oracle Performance Firefighting

A review of Craig Shallahamer's new book.

See page 10.

Firefighting 101

An excerpt from Craig Shallahamer's new book.

See page 14.

Much more inside . . .

Plain Speaking

An Interview with Brian Hitchcock

Brian Hitchcock needs no introduction to NoCOUG members; over the years, he's reviewed more than a dozen books for the NoCOUG Journal. *Brian's book reviews are very insightful; he uses them as an opportunity to reflect on the topic of the book and he doesn't mince words. For example, when reviewing* Database Hacker's Handbook *for the May 2006 issue of the* Journal, *Brian said: "Oracle is hard to secure because it is so big and so complex. A reasonable person would say that this argues for using simpler RDBMS software. Take MySQL for example. A simpler RDBMS to be sure, but what is the MySQL vendor doing? They're adding "features" as fast as they can, which adds to the size and complexity of their product. And why are they doing this? To meet market demands! Perhaps the enemy is us?"*

Hard to secure? The enemy is us? Them's fighting words.

Yes, we are the problem. We have created the problem and we continue to demand that the problem be made worse. Of course, as in all real stories, everyone has an agenda that is being served. The more complex Oracle becomes, the better for our job security.

We could, in a perfect world, demand that Oracle (and other vendors) stop adding features to their products and delay shipment of anything they sell until it was perfect. We could, hypothetically, demand that no software be sold to us until it was perfectly secure, performed flawlessly, and didn't need administration or support. But we don't. Why? Because all the complexity—which causes many of the security issues—is what we want and what we demand. Oracle knows what sells and that isn't always what we need.

We think we can get some sort of advantage over our competition if we just have somemore features in our business systems, and we assume that these business advantages will always outweigh the costs of actually implementing and supporting these new features in our systems. Consider the ERP system. How many stories have you heard where a business started to implement a large, complex ERP system because it would make them more profitable, and later on that same business has lost huge amounts of money due to project delays, bugs, etc.?

Because of our belief (we want to believe) that new and more complex is better, we buy shiny new stuff. Whether that stuff really works is a support detail left to those unlucky persons such as myself. Yes, it is new and it is shiny but it won't run on our servers. No problem, time for new servers! But what about the existing servers? Well, you must support all of them! No problem! And now we need to move all this into a cloud—wait, is that a private or a public cloud? Will any mobile device that our users find at Best Buy be fully interoperable with the new software? And we wonder why security is so hard.

We complain about it, but we really get what we really want—more interesting and more secure jobs. And for all the complaining, we will now spend lots of dollars traveling to the city by the bay to spend a week wallowing in all the new shiny stuff that will create lots of new problems—and maybe give our businesses some real competitive advantage.

This generates an IT system arms race—my company must keep up with our competition. Do any of our businesses really need a fully virtualized, cloud-based, mobile-enabled order-fulfillment system? I'm not sure, but we absolutely must not fall behind our competitor down the street (or in Redmond). The arms race supports itself. Instead of limiting the complexity to simplify our businesses, we seek out new and shiny and try to bolt it onto what we have. More complexity breeds more complexity, which needs more of us to support. Is everybody happy?

Actually, when you look at the big picture, we have a pretty sweet deal. We make our own mess and then try to clean it up, which creates a bigger mess . . . and we get paid all along the way.

My pointy-haired manager wants me to patch the database every quarter, but he's never heard of "attack surface area" and "surface area configuration." I want to blurt out that patching the database is like recaulking all the doors and windows but forgetting to close them. Are patches guaranteed to be free of regressions? Do I dare inject them into our mission-critical e-commerce databases without full-blown regression tests? Can we afford to patch? Can we afford not to patch? Do you patch?

First, my manager has the most wonderful hair I've ever seen. He made me say that. Really! I live in fear. He requires that we call him "Mr. Big." But his hair is just wonderful!

Let's start with the easier parts of the question. I support Oracle Applications environments and patching is a full-time job for an Oracle Applications DBA, so yes, I patch. But that doesn't really address the issue. Oracle Applications is just a more extreme case of an Oracle database environment. There are always new bugs coming to light with patches to fix them. And, there are new features that can be added by patching. So, back to the question, do you patch? Yes, but only with a really good reason. So what is a really good reason?

This is not a simple question. There are so many Oracle products with so many patches that I don't know that it is theoretically possible to be completely patched at any given point in time. And even if you were, it wouldn't last long. I think it is clear that no one is fully patched. Therefore, the real question is how do you decide what to patch and when. I think you should patch only when there is a clearly defined reason for taking the risk of patching. Examples: your system won't run without a patch—pretty easy, everyone from IT to Sales will agree you need that patch. Next, you must have a patch to have a feature that allows you to sell more stuff, again, pretty easy. How about patches because they are recommended by support? Not so easy. Again, I would want a specific reason to apply these patches. The phrase I really don't respond well to is "this patch fixes a lot of bugs." This means I am patching lots of stuff I may not be using and could break stuff I am using.

I don't see this as a good enough reason to patch. But reality plays a part in this. In the Oracle Applications world, there are many times when you have to apply a huge rollup (or family pack) patch just to fix the one bug you are hitting. You don't have a choice,

Oracle doesn't (and realistically they can't) offer all the bug fixes as individual patches. When looking at a patch I ask the following: What will I tell the business if this patch brings down production? If I have a clear answer (our system was down already, we wanted to sell more stuff . . .), then I am confident that the need to patch outweighs the risk.

Further, there is always risk in patching. Even if a patch fixes whatever the problem was that made you want to patch in the first place, you don't know, for sure, that the patch didn't mess up something else. This becomes complicated. Imagine all the products Oracle offers. We all have different combinations of those products and we all have a slightly different set of versions, release levels, patch levels, and options installed for all of those products. And they interact with the OS and the hardware and the network and who knows what else you have added over the years. How can Oracle test any patch to be sure it doesn't break something else? They can't. I'm not criticizing Oracle (full disclosure, I want to work for Oracle, anytime soon would be good!). But we have to keep in mind that Oracle can only do so much in the way of regression testing. When you patch, you take on risk. A good example of this is the dreaded "one-off" patch. The README files for these patches include a disclaimer that the patch hasn't been regression tested.

Truly bad science, but it is what we do. So, to answer another part of the question, no, patches are not guaranteed to be free of regression issues.

Which brings us to one of my favorite patching topics, CPU patching. These are bundles of patches to fix bundles of bugs (if there are gaggles of geese, what is a whole lot of bugs?) related to known and recently discovered security issues with all Oracle products.

From above, I think we need a really good reason to patch, but for CPU patches, what exactly is that good reason? Well, the obvious good reason is that we want to have a secure system. We all agree that is what we want, but do CPU patches really get us there?

If you read the CPU patching docs carefully, you will find that they don't do what you think they do. A reasonable person would assume that if you apply the quarterly CPU patches regularly and you keep up to date, your environment is secure from the known security threats as of the time of the most recent CPU patch. Not exactly. Each CPU patch represents Oracle's best effort to date to fix all of the known security issues. That doesn't mean the fix for any given security issue in a given CPU patch really fixes the problem. It may contain the best that Oracle has at the time, but more bugs are often found that get fixed in the next CPU patch. And you don't really know how Oracle decides which security fixes get into each CPU patch. Are you sure Oracle puts all the most important security patches into the latest CPU patch, or, do they put in the things they can fix before the next quarterly CPU patch release date? You *don't* know—there may well be a very serious, well-known security problem that didn't get fixed before the latest CPU patch was released. You think you are secure because you are up to date on CPU patches, but you may or may not be. Look at this from another perspective: How do you know you need any or all of the fixes in a CPU patch? Oracle is trying to fix as much as they can as fast as they can, but that doesn't mean they are fixing anything that affects you. For example, if your system is behind a firewall and does not interact with the Internet, do you need CPU patches at all? I don't know. You have to make all of these impossible decisions with imperfect data.

Given this, and given the risk of patching, how do we assess the CPU patching question? I think this is the same as any patch. You need to understand why Oracle does what it does, and you need to merge that agenda with that of your business. If your business would be harmed by press reports that you weren't up to date with all the vendor security patches, then you'd better be CPU patched all the time and learn to live with it. Always have a good reason to apply any patch. And I'm the first to say that political reasons are just fine. In all my experience, the political issues always trump the technical issues and technical persons do much better once they embrace this reality.

On to the next part of the question, can you "inject" them into a mission-critical database without full regression testing? I think you have to. You can't do full regression testing, it would take forever. And your business can't afford it. Clearly you don't "inject" patches into your production environment; you test them in dev or alpha and beta systems, etc. The problem with this is that only production has the most current set of data. Beta may be a month behind, and the data could be different enough that the same patch works in beta but not in alpha, depending on what the patch is changing. All in all, we take some level of risk to move forward. Want to be on the latest version of Oracle? Cool—then you take the risk of having the latest bugs. Want to avoid patching by staying on the old version of Oracle you have been running for years? No problem—you take on the risk of very old bugs that won't be patched because your version isn't supported anymore.

What would a reasonable person do? First, a reasonable person wouldn't become a DBA to begin with, but that is for another interview. You and your business will make some decisions. Most shops that I have been involved with went one of two ways: *all patching all the time* or *none at all unless abso-*

lutely required. Both of these approaches have their problems and benefits. I was asked if you can afford to patch, can you afford not to? My answer is "both." You can't support Oracle for very long and not apply some sort of patch somewhere. CPU patching alone requires quarterly patching at a minimum. Or, you could decide to do CPU patching once a year, but then you are exposed (in theory) to some of the known security issues for most of a year.

Finally, some examples from my experience, which may or may not have anything to do with anything.

CPU patching is complicated enough for the Oracle database, but for Oracle Applications, it is even more fun (and/or more job security, depending on how you look at it). For Oracle applications, CPU patching starts with the database but then moves on to the Oracle Application Server, the web and forms servers, various developer libraries, etc. And the best part? For Oracle Applications, the CPU patches are not cumulative. Think about that. Yes, CPU patches are cumulative for the database. No matter how far behind you are on CPU patching, you can apply the latest CPU patch to your database and be done. Not so for Oracle Applications. If you haven't applied any CPU patches ever, and you are running Oracle Applications 11.5.10.2, this means you have to go back 4 years and apply all of the CPU patches between then and now. No kidding. Once upon a time, I was asked to plan the CPU patching for such an environment, and after about a month of planning I offered this option: we are so far behind and the patch plan is so complex (300+ patches plus all their pre-req patches), it would be less effort to simply upgrade to Release 12 of Oracle Applications. And, with Release 12, the CPU patches for Oracle Applications are cumulative. Patching is a crazy business. But it pays well.

What about Oracle Database 11gR2 then? Should I upgrade this Oracle 7.3.4 database, which has been running without a hitch on Windows NT SP2 for years and years?

Many questions are posed in a way that masks their inherent assumptions. You are not running without a hitch. That is an illusion.

When you can't find any hardware that supports NT SP2 what will you do? Will that be a "hitch"? When you need to do anything to this system and you try to find people that can help you, will they be available and what will they cost? Will that be a "hitch"? There is no free lunch. Yes, you perceive that this database has been running without a hitch, but that isn't true. You have put this system into a deep freeze where time stands still, as if it were traveling to Mars. As long as everything is just perfect (not likely in the real world), yes, this system will continue to run. When something does happen, the options available to fix this system and the cost of these options will be very unpleasant. The longer you wait, the more painful the transition will be.

And by keeping this system frozen in the past, you add to the complexity of your environment. And that makes it more expensive to support, and so it goes. The logic is self fulfilling. If you wait long enough, your decision to keep this system in the past appears to make sense. You will be right in that it will be really expensive to bring this system into the present, but this is the illusion. You have to keep telling yourself that this situation is good because you need to justify your leaving this problem unresolved for so long. I get it, I really do. I support databases that really should be upgraded to a current version. When asked, the people who make these decisions say they "can't" upgrade these databases. The real reason is that they don't want to spend the money now, and that means they will spend lots more money later. Overall, this is good for me.

Thanks for taking the time. I wish I could ask you about medical insurance reform but let's not go there. But, before you go, I've just got to ask: what's with the Tiffany lamp on your Facebook and LinkedIn profiles?

That is precisely the problem with our health care system: no one wants to "go there." This ties it all up nicely. Just like the 7.3.4 Oracle database running on Windows NT SP2 mentioned before, there is a perception in our country that the health care system has been "running without a hitch." This isn't reality. And the painful changes needed to make our health care system viable for the long-term (that is, longer than the Boomers will be around) will only get more painful the longer we wait.

I don't claim to have all the answers, but I am very sure of this. I want everyone to have some level of health care because I want everyone working hard so they can pay my Social Security benefits. We are all in this together. Unfortunately, not all of us get that, and that is the real issue.

The Tiffany lamp was a seven-year project. I first took stained glass classes in high school and eventually even sold some work and did some installations. This lamp is my attempt at a Tiffany-style leaded-glass lamp. Tiffany produced many lamps using glass that his company produced to create the complex colors. This is the smaller version of the Wisteria lamp shade and is 10" in diameter. It is called the Pony Wisteria and has 873 pieces of glass, which are wrapped in strips of copper foil. The foil-wrapped pieces are placed on a mold to form the shape of the shade and soldered together. The full-size Wisteria lamp shade is 18" in diameter and has closer to 2000 pieces of glass.

An excellent book was published in 2007 discussing the woman who actually did the design work for many of Tiffany's most famous lamps, Clara Driscoll. The book is *A New Light on Tiffany, Clara Driscoll and the Tiffany Girls*. It describes the working environment in the late 1800s and relates many issues of the time that resonate today. Established workers threatened by new low-cost labor, management cost cutting, etc. Anyone interested in Tiffany lamps would find this book worthwhile.

Why did it take me seven years to complete? First, I had never done a lamp with such small, complex pieces of glass before, and I had to learn as I went. This meant I would stop for months while I figured out how to cut and grind such small pieces of glass. Similar delays occurred while I learned how to solder on a curved surface, and I have some scars where gravity and molten solder converged on my hands. Those that make Tiffany reproductions professionally could produce such a lamp in much less time, but then, this was my first and so far only. I want to do the larger version someday, but my job and NoCOUG have so far prevented that. I have all the materials but I haven't made the time so far. ▲

Interview conducted by Iggy Fernandez

All of Brian's book reviews are available at **www.brianhitchcock.net**.