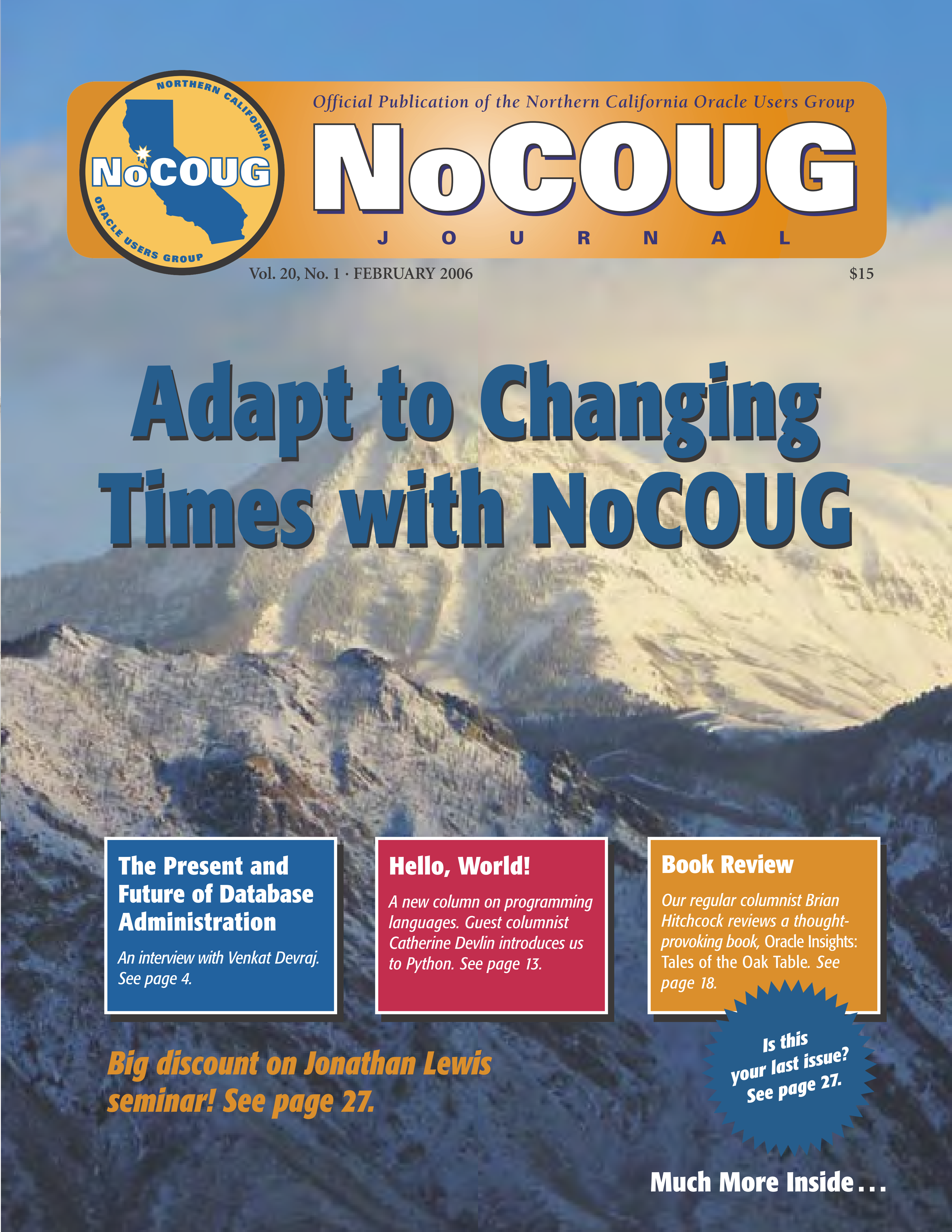

Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 20, No. 1 · FEBRUARY 2006 $15

Adapt to Changing Times with NoCOUG

The Present and Future of Database Administration

An interview with Venkat Devraj. See page 4.

Hello, World!

A new column on programming languages. Guest columnist Catherine Devlin introduces us to Python. See page 13.

Book Review

Our regular columnist Brian Hitchcock reviews a thought-provoking book, Oracle Insights: Tales of the Oak Table. *See page 18.*

Big discount on Jonathan Lewis seminar! See page 27.

Is this your last issue? See page 27.

Much More Inside . . .

Oracle Insights: Tales of the Oak Table

Review by Brian Hitchcock

Our regular columnist Brian Hitchcock takes an irreverent and thought-provoking look at a fascinating new book, Oracle Insights: Tales of the Oak Table, and highlights some of the equally irreverent and thought-provoking comments made in the book such as, "We have many world-renowned experts whose tuning methods lead us nowhere near the true problem" and "Sometimes the cleverest people made the most fundamental errors."

Title: *Oracle Insights: Tales of the Oak Table*

By: Mogens Nørgaard and 10 others

Published by: Apress, 2004

Summary

Overall review: Some of the material is worthwhile, but overall, I think your time and money could be better spent on other books.

Target audience: Oracle DBAs.

Would you recommend to others? Again, there is some good material, but not enough to pay for.

Who will get the most from this book? Readers who want to spend the time to learn more of the lore of Oracle.

Is this book platform specific? No.

Why did I obtain this book? I didn't; I was asked to review it and it was given to me.

Opening Remarks

I think it is important that you understand where I'm coming from before you read this review. While I don't own any oak furniture (all my furniture comes from Wal-Mart: it works, and it's a good value), these observations represent my perspective:

- You are the only one who knows your systems.
- You should use anything that works, including ratios, SQL trace data, OS level statistics, and whatever tools and techniques get you and your customers to a good outcome.
- Talking to people is the most important technique you have. Communicating with the people (and they are people) who support all the other parts of the overall application system does more to solve complex performance issues than any single expert can do.
- Proprietary has seen its best days. Proprietary Unix and proprietary RDBMS are on the way out. They won't be gone tomorrow, but the various open-source operating systems and RDBMS products are growing. The database itself is moving away from the center of the action to a supporting role. I think you should use the time you have to learn about open-source RDBMS and the many other components of modern application systems. It is these other components that are coming up more and more in performance engagements.

Overall Review

This book is a collection of material from its various authors. As such, the content and quality of the chapters vary widely. While there are pieces of information that are worth knowing, much of the material is widely known, commonsense, or both. There is little here that is new or unique. I don't think you will learn very much that you can immediately apply to your current projects.

This book reflects a lack of editing. You don't know how good your editor is until you see how other publications handle the task. There is no consistency between the chapters. One will be very high level, and the next will have very specific technical detail, with nothing tying them together.

Far too much space is taken up with the authors telling us how impressed they are with each other. Reading 32 pages of this is tedious. And do you really need to pay to read about these experts' bathroom and sleep habits?

Some of the material is very old. There are many contradictions from chapter to chapter. One expert explains just how bad something is, and in the next chapter another expert explains how that same thing is a valuable solution. Some of the material is repetitive. How many times do we need to be told that using "ratios" for performance tuning is a bad idea?

Given the vast differences in quality, tone, and theme of each chapter, I decided to review each chapter separately.

The chapters are:

1. A Brief History of Oracle—Dave Ensor
2. You Probably Don't Tune Right—Mogens Nørgaard
3. Waste Not, Want Not—Connor McDonald
4. Why I Invented YAPP—Anjo Kolk

5. Extended SQL Trace Data—Cary Millsap
6. Direct Memory Access—Kyle Hailey
7. Compulsive Tuning Disorder—Gaja Krishna Vaidyanatha
8. New Releases and Big Projects—James Morle
9. Testing and Risk Management—David Ruthven
10. Design Disasters—Jonathan Lewis
11. Bad CaRMa—Tim Gorman

Appendix: Join the BAARF Party (or Not)—James Morle and Mogens Nørgaard

Chapter 1: A Brief History of Oracle—Dave Ensor

I enjoyed the brief history, and it helps explain how some "features" came to be. It isn't always great science. I also recommend the section on Oracle Parallel Server (OPS) and Real Application Clusters (RAC). The author gives a review of OPS and RAC and offers the opinion that overall it isn't much better than a well-configured SMP machine. Once you really grasp the point that you need to "partition" your database to get good scalability, you realize that the less you really use RAC, the better it performs. I also found it interesting that the author refers to Flashback features as a "complete farce," and the point is made that most of the Flashback features aren't really useful for most systems.

Chapter 2: You Probably Don't Tune Right—Mogens Nørgaard

This chapter has the best insights of the whole book, but it really makes the following chapters seem less worthwhile. The author explains how Oracle is becoming a commodity and that proprietary Unix is dying. These observations need to be heard and the implications considered. However, this leaves an important question unanswered. Having stated that Oracle is becoming a commodity and proprietary Unix is dying, the author doesn't answer the logical next question, namely, is there any reason proprietary RDBMS, such as Oracle, won't die as well? The implications of the end of the era of proprietary OS, RDBMS, and other pieces of our application environments need to be understood. If proprietary Unix is being replaced by Linux, why not include some insights on how Oracle is being replaced by open-source RDBMS?

The author also explains that you need fully instrumented code for all pieces of an application, and that without this you can't know what is happening to your app. This is a very important and reasonable observation. However, much of the rest of the book ignores this and covers nothing but the Oracle database. How many current business systems consist of only an Oracle database? I would like to know what insights the author could offer us about application systems that are more complex.

Chapter 3: Waste Not, Want Not—Connor McDonald

This chapter discusses very detailed solutions to very specific performance issues. While the author's description of these complicated tuning solutions is thorough, what isn't discussed is how this system performed over the long term. Once a system is highly tuned, it often needs even more tuning in the future, making complicated tuning solutions harder to maintain. Also, in Chapter 7 we are told that we shouldn't tune too much. The author doesn't discuss the issue of how much tuning is too much. The author suggests Flashback as a cure to one of his problems, but he doesn't address the issue that the author of Chapter 1 refers to Flashback as a "complete farce."

As with other chapters, there are interesting things here. The discussion of parsing issues and the Hidden DUAL Access interested me because it was something I had never thought of. Some of the material here covers Forms 6 on Oracle 8.1.6. Most readers would probably prefer hearing about the author's experience with something more current.

You need fully instrumented code for all pieces of an application, and that without this you can't know what is happening to your app.

Chapter 4: Why I Invented YAPP—Anjo Kolk

The most interesting thing in this chapter is the discussion of just how disorganized the implementation of the Wait Interface really was. It is popular to view the Wait Interface as something that was fully designed and covers everything. It was a one-off solution to a pressing problem and, to quote the author, "it wasn't really planned" and "it was kind of sneaked in at the last minute." Keep this in mind when you hear how the Wait Interface can solve all performance issues.

It is also pointed out that the Oracle software often doesn't work the way the manuals (and former Oracle employees) say it does. Keep this in mind when things just don't work the way you were told they would in the Oracle training class.

The author's main story covers the many problems he solved for a system using OPS, but the value of using OPS is not discussed. Didn't we read in Chapter 1 that OPS and RAC really aren't that useful despite all the marketing?

As with Chapter 2, the issue is brought out that you must have performance data for all the components of your application system before you can make any progress, but we are not offered any practical solutions to this issue. I think this is a very important insight that doesn't get much attention, because it goes against the status quo. Oracle experts don't want to spend too much time pointing out that you may not need an Oracle expert at all.

Chapter 5: Extended SQL Trace Data—Cary Millsap

You will not be surprised to find that this chapter is simply selling extended SQL tracing as the cure to all performance issues. When the dominant performance issue is in the Oracle database, SQL tracing information is very valuable. But this isn't always the case, and in my experience it isn't even *often* the case. This chapter contradicts what is stated in Chapters 2 and 4, and again in Chapter 7, that you must have performance data from all parts of your system, not just within the Oracle database. The author tells us that "performance problems cannot hide from Oracle's extended SQL trace data." If the performance issue is in the application server, it most certainly can and will hide from SQL trace data. Clearly, I don't agree that extended SQL tracing is all you need to solve real-world performance issues, so I found this chapter to be lacking in insight. I think it encourages people to waste time looking at very specific details of the Oracle database before they have any evidence that the issue is even database related. We are told that relying on the ratios (such as the much-maligned Buffer Cache Hit Ratio) is a very bad thing. Perhaps relying on SQL trace data is no better. There are times when either of these can help, and there are times when both are useless.

Chapter 6: Direct Memory Access—Kyle Hailey

The author describes his effort to write code that directly accesses the Oracle SGA and how this was used to help him resolve various problems. As with other chapters, this material describes an effort from 1994 using Oracle 7. Using direct access of the Oracle SGA can be dangerous, and even if we assume it was a good idea in 1994, the author doesn't provide any information on how this technique could or should be used on current systems. The author describes all the ways this can crash your database. Further, the author tells us that his code is "quite difficult to maintain."

Chapter 7: Compulsive Tuning Disorder—Gaja Krishna Vaidyanatha

"We have many world-renowned experts whose tuning methods lead us nowhere near the true problem."

In this chapter, the author covers the many aspects of too much tuning, wherein DBAs go off to tune things that don't matter, have no idea if they have made any real progress, and don't know when they should stop tuning. There are many things that I found interesting in this chapter. In the author's own words, "We have many world-renowned experts whose tuning methods lead us nowhere near the true problem." Keep this in mind whenever you read or hear anything from tuning experts.

We are told that "ratios are for losers." However, the same chapter asserts, "This is not to say that all ratios are completely useless." I've had situations in which a ratio did help resolve a performance issue. I conclude that this makes me a "not completely useless loser." My charming spouse finds this very amusing. I think she laughed a little too much, but I digress.

More seriously, the author brings up a very valuable topic, namely, the need to ask "what is the ROI of a tuning effort?" This is almost always ignored because it is hard and will often point us to something we don't want to work on, such as a performance issue outside the database. You can't know that SQL tracing, or a ratio, or anything else is worth looking at until you know why you are tuning and how you will know when you have tuned enough. This again points to the need to have performance data for all the pieces of your application. The author also tells us that performance tuning is "all about identifying the root cause of an application-level response time issue." Having made this statement, the author turns to discussing nothing but Oracle database performance issues. I would like to hear about the author's experiences in tuning application systems where the problems were more complex than a single database on a single machine. This is the type of performance issue I see every day.

Assuming that the performance issue is in the database, the author explains why we need both database- and OS-level performance data to "define the complete performance problem." In Chapter 5 we were told that SQL trace data is "usually the only data collection needed." Perhaps this depends on what you think "usually" means. Further, the author states that "any resource consumption and contention that is generated outside the realm of an Oracle database cannot be revealed by the Oracle Wait Interface." The same is true for SQL tracing. Sources of response time that are outside the database will not show up in a SQL trace.

The author tells us that every tuning effort must have a stated tuning requirement. Unfortunately, the real world doesn't work this way. I get many requests to see if we can make it faster and if we can add more databases to the consolidated database server. These are performance issues, but they don't have a stated numerical requirement.

I found the coverage of new features in 10g to be too detailed. I don't get much out of looking at listings of view definitions. On the other hand, the coverage of exactly how to get various OS-level metrics is very good. This is useful information for anyone tracking down performance issues on a single server.

Chapter 8: New Releases and Big Projects—James Morle

This chapter stands out for two reasons. First, it is simply the best writing in the whole book. This person can tell a story. Second, one of the insights left me speechless, but we'll get to that later. We are told (again) that OPS doesn't really scale as well as we have been told it will. We are also told that we need to have extensive testing at all stages of application development. This is a valid observation, but at the same time, does anyone really believe that most organizations do extensive testing at all stages of application development and maintenance? It simply costs too much, and the application itself changes too fast. The story involves the details of multiple performance issues for Oracle 7.0.13 and MTS. As with several other chapters, I'd prefer to hear about the author's experiences with more recent versions of Oracle.

I enjoyed the part of the story that covers character set issues, since I have been there and done that. If you are facing a character set conversion, this is a good story to read, and I think this issue is a lot more common than most DBAs realize. The discussion of the problems with RAID is very clear and makes the point very well, and the discussion of Functional Partitioning is very good. It provides detail on why you need to use RAC less to make it work better.

Now for the part that left me speechless: The author offers "Insight 4," where we are told that application systems should have manual processes in case the application isn't available. And we are also told that the "vast majority" of businesses can exist for short periods with manual procedures.

Since the specific performance problem that is described is from the early 1990s (when exactly was Oracle 7.0.13 the latest release?), perhaps a business could have a manual process for some part of its mission critical applications back then. The author states that his customer did have a manual process that could handle the "front office" aspects of the application when necessary. This may have been true at that time, but imagine Amazon or eBay having a manual process to display thousands of items to thousands of customers, let alone to gather auction bids, payment information, and so on. The author doesn't explain how the vast majority of modern businesses could even contemplate a manual process for their applications.

We are told the value of Flashback again, so we have two experts who like Flashback, and one who doesn't.

Chapter 9: Testing and Risk Management—David Ruthven

This chapter could have been left out with no impact. The author covers, in great detail, all of the aspects required for an ideal application development process. Topics covered include the need for software reliability, designing for scalability, the benefits of accurate software design, the fact that no parameters should be hard coded, the need for instrumentation, and the many kinds of testing that you should perform. This is all good information, but how many organizations can really follow this idealized approach? Further, there is no mention of how this applies to Oracle or applications that use Oracle software. This chapter may well be very good . . . for a book on software design and the testing needed during software development. I'd like to hear how the author has achieved this ideal development process for a complex, real-world application.

The author does provide a good story about customer support, though. A customer was very upset, and instead of creating a scene, the customer took the author to dinner. This made the author feel so bad, he made sure the customer's problem got fixed very quickly.

Chapter 10: Design Disasters—Jonathan Lewis

This is another chapter of very detailed database performance tuning stories. The discussion of how the optimizer gets it wrong is good. We are told the value of Flashback again, so we have two experts who like Flashback, and one who doesn't, and yes, I'm keeping score.

Chapter 11: Bad CaRMa—Tim Gorman

Included here are more detailed stories about performance issues with Oracle version 7. Perhaps this book

(continued on page 26)

Session Descriptions

(continued from page 25)

front-end code to the database. Finally, the presentation discusses the type of development shop that will benefit the most from ADF (and ADF Faces) and whether it really is possible to achieve the productivity of an Oracle Forms environment by using this release of JDeveloper.

Raptor: Introducing Oracle's New Graphical Database Development Tool
Kris Rice, *Oracle Corporation*

As a DBA or Database Developer, do you use a variety of tools to browse database objects, create and run SQL, and edit and debug PL/SQL? Raptor is Oracle's newest graphical alternative to SQL*Plus, allowing the Database Developer a convenient way to perform basic tasks. With Raptor you can manage objects in Oracle databases. You can browse, create, edit, and delete (drop) database objects; create, edit, and debug SQL statements and PL/SQL code; manipulate data; export and import database objects; and create reports. You can connect to any target schema using standard Oracle database authentication. In this session we introduce Raptor and demonstrate the latest functionality.

SQL Tuning in Oracle Database 10g: The Do's and Don'ts
Mughees A. Minhas, *Oracle Corporation*

As part of Oracle's ongoing strategy to make the database more self-managing, Oracle 10g has introduced two new solutions, SQL Tuning Advisor and SQL Access Advisor, which together can eliminate the need for manual tuning. Users can now let the Oracle database tune their enterprise applications for them. This paper discusses the two new advisors, describing the internals of how they work, best practices for their usage, their limitations, and their benefits.

Performance Diagnosis Usage Model in Oracle Database 10g
Prabhaker Gongloor, *Oracle Corporation*

Oracle Database 10g has a rich set of capabilities that make the identification of the root cause of performance problems quick and easy. This presentation discusses the usage model, various diagnostic solutions that are available, when to use which solution, and their most effective use. It covers functionality introduced in Oracle 10.2 that allows DBAs to analyze transient performance problems, and compare current and historical performance to baselines (preserved snapshot sets) to detect changes in configuration and workload. ▲

Book Review

(continued from page 21)

should be called "Oracle 7 Insights"? The higher-level story of the "data-driven" application is good because it points out the problems with trying to write a completely generic application. The author offers the following observation: "Sometimes the cleverest people made the most fundamental errors." This concept has come up in several chapters, and it bears repeating. Be careful when dealing with experts. Sometimes you really are better off with solutions that are less exciting and have been proven over time. The story related in this chapter is a good example of an organization buying into the vision of an expert and paying an incredible price for it.

Appendix: Join the BAARF Party (or Not)—James Morle and Mogens Nørgaard

Since this issue has been discussed for many years, I would be surprised if the conclusions presented are news to anyone.

Conclusion

While there is interesting material presented, the real value of this collection of authors was missed. If the various topics and themes had been identified, and if each of the authors had discussed each topic and either had come to a consensus or had explained in detail their divergent views, I think the book would have been much more valuable and would have come a lot closer to the stated goal of providing "Oracle insights." As examples, I'd really like to hear the various authors explain how they do or don't need OS-level statistics for resolving performance issues, whether OPS/RAC really is or isn't mainly marketing hype, and whether Flashback is useless or not.

I think this book had great potential, but by isolating the authors in their own private worlds, the potential was missed. This is a collection of material that lacks the power that would have come from interaction among these authors.

About the Authors

The book cover says that the authors are eleven leading authorities on Oracle's database engine who have written this book to share unique insights on how best to exploit the software. ▲