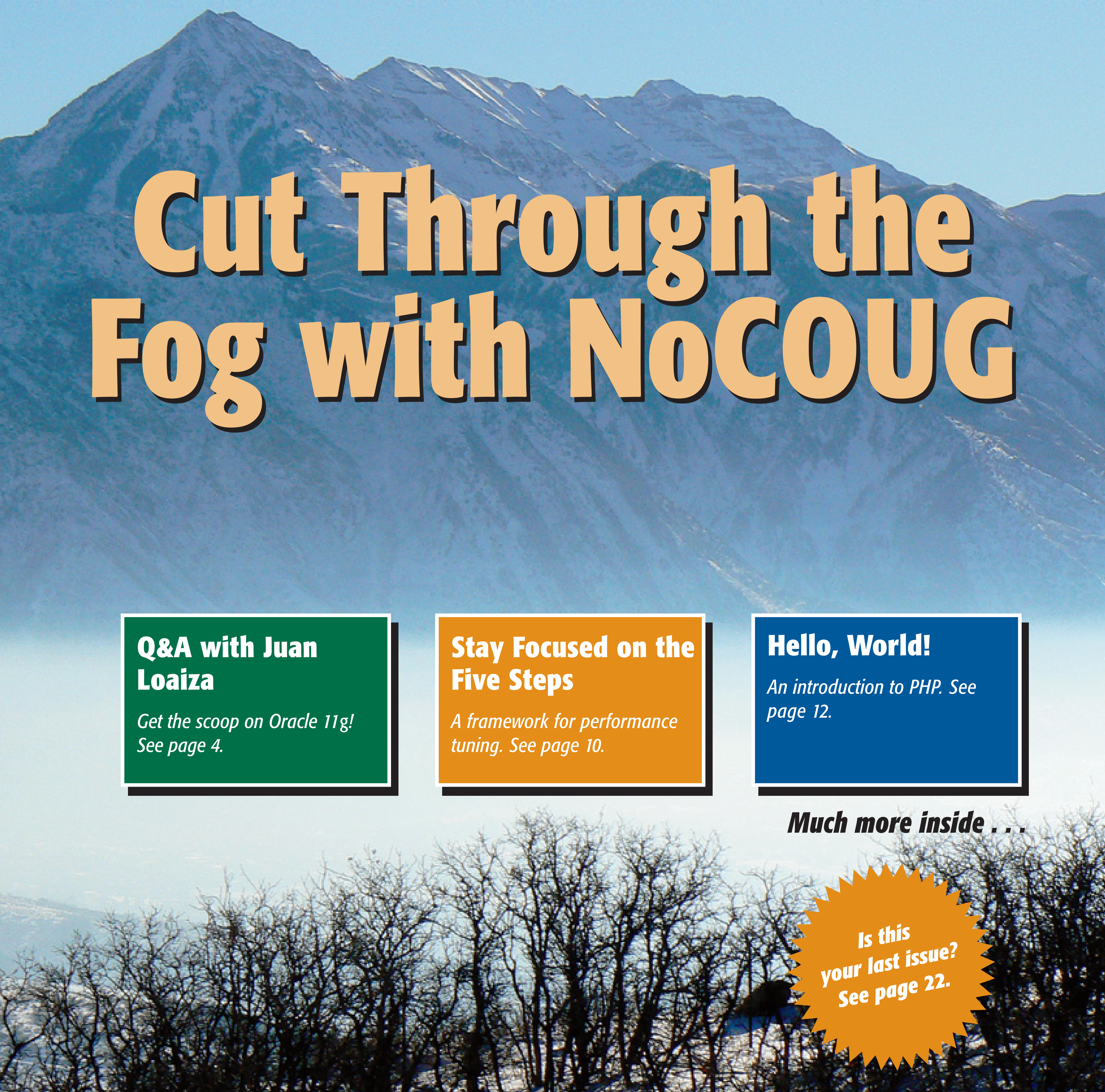
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 21, No. 1 · FEBRUARY 2007

$15



Cut Through the Fog with NoCOUG

Q&A with Juan Loaiza

Get the scoop on Oracle 11g! See page 4.

Stay Focused on the Five Steps

A framework for performance tuning. See page 10.

Hello, World!

An introduction to PHP. See page 12.

Much more inside . . .

Is this your last issue? See page 22.

How I Became the Editor of the *NoCOUG Journal*

The story so far: after introducing myself to then-NoCOUG President Roger Schrag at a NoCOUG conference in 2004, I was invited to the next few NoCOUG Board meetings as an observer.

I attended the next few Board meetings and also began writing articles for the *Journal* at the invitation of then-editor, Lisa Loper. When the time came to nominate a new Board, the nomination committee asked me if I would like to be NoCOUG's representative to the International Oracle Users' Group—my principal duty would be to attend the annual IOUG conference in sunny Florida. My company paid my way and off to Orlando I went.

A year of Board meetings had come and gone when, out of the blue, Lisa Loper decided that it was time to pass the baton. Being a regular *Journal* contributor and having a year of Board experience made me a natural candidate for the job.

And that's how I became the editor of the *NoCOUG Journal*.

—Iggy Fernandez
NoCOUG Journal editor

Table of Contents

–ADVERTISERS–

Publication 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 regional meetings.

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

The submission deadline for the upcoming May 2007 issue is March 1, 2007. Article submissions should be made in Microsoft Word format via email.

NoCOUG does not warrant the NoCOUG Journal *to be error-free.*

NoCOUG BOARD

President
Lisa Loper, Database Specialists, Inc.
lloper@dbspecialists.com

Vice President
Darrin Swan, Quest Software
darrin.swan@quest.com

Secretary/Treasurer
Jen Hong, Stanford University
hong_jen@yahoo.com

Director of Membership
Joel Rosingana, Independent Consultant
joelros@pacbell.net

Journal Editor
Iggy Fernandez, Verizon
iggy_fernandez@hotmail.com

Webmaster
Eric Hutchinson, Independent Consultant
erichutchinson@comcast.net

Vendor Coordinator/IOUG Rep
Diane Lee, Lockheed Martin
diane.c.lee@1mco.com

Director of Conference Programming
Roger Schrag, Database Specialists, Inc.
rschrag@dbspecialists.com

Director of Marketing
Naren Nagtode, Franklin Templeton
nagtode@yahoo.com

Training Day Coordinator
Hamid Minoui, Database Specialists, Inc.
hminoui@pacbell.net

Track Leader
Randy Samberg, PalmSource
rsamberg@sbcglobal.net

Track Leader
Hanan Hit, Skyrider, Inc.
hanan@skyrider.com

NoCOUG Staff
Nora Rosingana, Independent Consultant
noraros@pacbell.net

ADVERTISING RATES

The NoCOUG Journal *is published quarterly.*

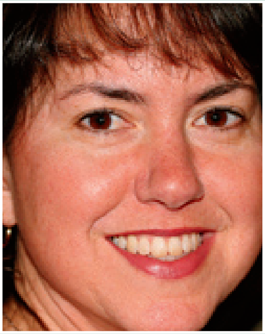
Size	Per Issue	Per Year
Quarter Page	$125	$400
Half Page	$250	$800
Full Page	$500	$1,600
Inside Cover	$750	$2,400

Personnel recruitment ads are not accepted.

journal@nocoug.org

What's Your Plan for 2007?

by Lisa Loper


Lisa Loper

As the newly elected president of NoCOUG, I'm busy making plans for the year ahead. If you want something to happen, make a plan, set some goals, and you are well on your way. Just be sure to wave as you pass the folks who never make plans or write down their goals.

So, what are your career plans for 2007? Have you included other people? Life isn't a solo sport, and neither is your career as an Oracle professional. It's just not possible to do it alone—and it probably wouldn't be any fun if you could.

In your Oracle career, you might do well to take on a concept that's popular in tennis (one of my favorite sports). It's often said that in order to improve your tennis game, you should play three matches a week—once with someone at your skill level, once with someone more advanced, and once with someone more junior. You get something worthwhile out of each of these experiences.

NoCOUG offers you a great way to accomplish this on a professional level. As you make your career plans for 2007, think about how NoCOUG can help you achieve your goals.

Time with Oracle professionals at your level: Set a goal to attend all four quarterly conferences included in your membership this year. This is a great opportunity to network with other professionals in your field. Meet people you can brainstorm with, problem solve together, and help each other grow in your respective careers. If you're not the type to "work the room" at conferences, consider volunteering and helping the board of directors in some capacity. It's a very welcoming group, and you'll make new friends quickly.

Time with Oracle professionals who are more advanced: One of the best ways to advance in almost anything is to learn from more experienced people. NoCOUG provides this educational opportunity with the outstanding Oracle professionals that we bring to our conferences and our special training days during the year. You'll be seeing nationally recognized names as well as many local folks who share their knowledge and help you learn from their experience.

Time with Oracle professionals who are more junior: It is often said that you don't really know something until you can thoroughly explain it to someone else. Coaching and helping someone more junior than you gives you that opportunity. There is much to be learned by acting as a "teacher" and encouraging others. With NoCOUG involvement, you can give presentations, write an article for the NoCOUG Journal, or just network and offer to be available to give advice to the more junior DBAs or developers you meet. Just remember, there was a time when you were less experienced and someone helped you. Now it's your turn to give back. Be generous with your time.

So, think about these three areas as you plan your involvement with NoCOUG in 2007.

We are sure to have a great year—full of education and opportunities—thanks to the dedicated NoCOUG volunteers and board members. Our past president, Darrin Swan, has done a wonderful job of leading NoCOUG for the past two years, and I plan to continue his efforts. I feel lucky to be associated with such a great, experienced, and dedicated group.

You'll notice a few changes to the composition of the NoCOUG Board, effective January 1, 2007. Diane Lee has taken over my previous role as Vendor Coordinator and will also take over IOUG Representative responsibilities from NoCOUG Webmaster Eric Hutchinson. Jen Hong has taken over Secretary and Treasurer responsibilities from Diane Lee and Darrin Swan will move into the role of Vice President. Hanan Hit's title has changed from Director At Large to Track Leader, while Director of Membership Joel Rosingana, NoCOUG Journal Editor Iggy Fernandez, Director of Conference Programming Roger Schrag, Director of Marketing Narendra Nagtode, Training Day Coordinator Hamid Minoui, Track Leader Randy Samberg, and NoCOUG Staff Member Nora Rosingana will continue to share their talents with us in 2007 in the same roles they performed in 2006.

Here's wishing you a happy and prosperous new year. Speaking member to member, I'm hoping that part of your plans will include expanding your participation in all that NoCOUG has to offer in 2007.

See you at the Winter Conference on Thursday, February 8, at Oracle in Redwood Shores! ▲

Is this your last issue? See page 22.

Q&A
with Juan Loaiza

Juan Loaiza

NoCOUG welcomes Juan Loaiza, an Oracle veteran best known to the Oracle community as the originator of the SAME concept—Stripe And Mirror Everything. He will be delivering the keynote address at NoCOUG's Winter Conference at Oracle Headquarters in Redwood Shores. We caught up with him and asked him to comment on the future direction of Oracle.

What do you do at Oracle? How are you involved in the design and development of Oracle 11g?

I am the manager in charge of the Systems Technology group at Oracle. The Oracle database development organization is divided into several areas including languages (SQL, PL/SQL, optimizer, etc), Security, XML, RAC, and Systems. I manage the Systems part, which includes most of the core Engine functions of the database.

The areas I manage include Tables, Indexes, Backup, Recovery, Transactions, Caching, Process Management, Replication, Data Guard, RMAN, etc.

I have been working at Oracle for 18 years and worked on every release since Oracle Version 6, so I have a lot of context on what has been done and why we did it.

What can you tell us about Oracle 11g? Main focus? Specifics? Personal favorites?

There are a lot of great improvements in 11*g*. Too many to mention in a short answer. One of my personal favorites is SecureFiles. In 11*g* we have completely revamped the way that unstructured data—which we traditionally call LOBs—are managed by the database.

Today applications that have both structured (relational) data and unstructured (file) data sometimes use the database to manage the structured data, and manage the unstructured data themselves in file systems. One of the reasons for this is that file systems are generally more performant at managing files. There is really no reason why this has to be true.

In 11*g* we took the best ideas of the file system community, and combined them with the best ideas of the database community to produce a new data type we call SecureFiles. We are able to store file data with comparable or better performance than file systems, while preserving the traditional benefits of databases like transactions, multi-versioning, SMP and Cluster scalability, consistent backup, partitioning, etc.

Another major innovation is in the area of testing. It is pretty much impossible today to truly test a complex production application at scale. The tests that people develop are generally orders of magnitude less complex than the production application. This leaves a lot of opportunity for errors to creep into production that are not caught in testing.

In 11*g* we implemented a mechanism that can capture the full workload of a running production database, and replay the exact same workload with the exact same timing on a test system. I think this is going to dramatically improve the ability of database users to successfully and quickly roll out improvements and changes. It also has the potential to reduce the cost of testing, which is one of the biggest costs in production environments.

We also have major improvements coming in the areas of result caching, flashback, online application changes, database patching, partitioning, standby database, RAC performance, optimizer statistics and plan management, compression, connection pooling, encryption, backup, ease of recovery, etc.

There is a lot to look forward to. I am frankly amazed at the quality and quantity of improvements that are coming.

Noel Yuhanna from Forrester Research predicted at Openworld 2006 that databases would become more "adaptive." How do you see databases and database technology

> *"We also have major improvements coming in the areas of result caching, flashback, online application changes, database patching, partitioning, standby database, RAC performance, optimizer statistics and plan management, compression, connection pooling, encryption, backup, ease of recovery, etc."*

"Our competitors have pretty much given up on the claims that RAC is flawed and does not work. They have failed to deliver their much-hyped shared-nothing OLTP architectures. We have even heard that they have given up on shared nothing, and started projects to copy the RAC architecture."

evolving over the next decade or two? How is Oracle leading the pack?

I missed Noel's talk at Openworld so I don't know exactly how he is using the word "adaptive." I will give you my perspective on this. The community of database users is extremely diverse. It includes space scientists, telephone companies, governments, web sites, auto makers, financial firms, libraries, schools, etc. Every one of these communities has interesting and distinct uses for databases. One of the things that we are working on is making the database more "adaptive" to the special needs of each community.

For example, partitioning and bitmap indexing technologies adapt the capabilities of the database to better meet the needs of the data warehousing community.

Similarly Secure Files and XML processing technologies will adapt the database to better meet the needs the unstructured data communities.

The interesting thing about these technologies is that they are not really vertical in the traditional sense. For example, partitioning was driven by the needs of data warehousing, but it is widely used by many other communities.

Similarly our XML technologies are coming into widespread use. Almost every feature that we put in for a specific community finds use in many other communities.

We are also adding very sophisticated self-managing and self-tuning capabilities that will automatically adapt the database to better meet the needs of each individual customer within a community.

RAC is hot—everybody who's anybody is eyeing RAC in 2007. RAC has been quite a success story, hasn't it?

RAC has been extremely successful. When we first introduced RAC in 2001, we made some very strong claims about it. We said that with RAC you will be able to run off the shelf applications in a scale-out environment and simultaneously achieve better performance, higher availability, and lower cost—these had been contradictory goals up to that point.

At the time, our competitors said that our architecture was flawed, that our claims were untrue, and that they were going to roll out shared-nothing architectures that would surpass the capabilities of RAC.

Five years later we have thousands of customers running mission-critical applications using RAC. The applications they are running include the world's most complex and sophisticated OLTP applications—SAP and the Oracle E-Business Suite. Some of the largest data warehouses in the world are running using RAC. Some of the fastest-growing companies in the world have based their data architecture on RAC.

Our competitors have pretty much given up on the claims that RAC is flawed and does not work. They have failed to deliver their much-hyped shared-nothing OLTP architectures. We have even heard that they have given up on shared nothing and started projects to copy the RAC architecture. When your competitors endorse your approach, you know for sure that you are onto something.

Most Oracle professionals know you as the author of the SAME paper. Is SAME still alive and well today?

It's interesting that I am known for SAME these days. Years ago I was mostly known as the originator of SQL Trace and TKPROF.

SAME is based on the basic principle that the best way to optimize the performance of your storage is to spread your load evenly across all the disks. This is in contrast to the approach of optimizing the performance of each disk drive by laboriously laying out the storage of each table so that the I/Os issued by each query fall optimally on each disk and do not "interfere" with each other.

When I first published this approach, I got a lot of push back from storage experts that this approach was too simplistic and would not work in practice. I think over time, many people have come to understand that by micro-optimizing the performance of each table, you are de-optimizing the performance of the system as a whole. You are winning the battle and losing the war.

The really nice thing about the SAME approach is that it is quite simple to implement, and it keeps working well as the application evolves and the data set changes. In Oracle 10*g* we introduced our own volume manager called ASM that incorporates the principles of SAME and makes them trivial to implement. ASM also solves the biggest technical problem with implementing SAME, which was that once you stripe your data across your disks, it becomes difficult to add more disks without re-laying out the storage from scratch. ▲

Juan Loaiza will be delivering the keynote address at NoCOUG's Winter Conference on February 8 at Oracle Headquarters in Redwood Shores, and will also participate in a Q&A session with a panel of top Oracle developers. Please RSVP at www.nocoug.org/rsvp.html.

Leveraging Decision Automation in Database Administration

by Venkat S. Devraj

Venkat S. Devraj

Decision automation (DA) is the capability of an expert system to analyze metadata regarding the state of the runtime environment and make decisions in real time during the course of carrying out a task function. The analysis that leads to a decision can range from processing simple guidelines to executing complex yet crisp rules, all the way to steering through industrial-strength fuzzy logic. Further, an expert system can start out with a set of basic formulas and then be "trained" via observation in a sample target environment, effectively allowing it to develop its guiding reasoning and behavior.

DA has been highly leveraged in the embedded software and appliance markets; it can be found in applications as wide-ranging as your household washing machine all the way to the Mars-bound space rovers. DA is beginning to raise an eyebrow in data warehousing and RFID-based applications; however, its potential and implications are yet to be understood and applied to information technology (IT) operations—where significant value can be derived.

Leveraging DA Within DBA

Database administration (DBA) is an effective candidate for leveraging decision automation within IT. DBA is one of the most expensive IT line items—over $50B worldwide is spent on human capital, according to industry estimates, approximately five times the size of the DBMS market itself, as defined by Gartner. Yet the area suffers from lack of standards even within the same organization and team. Most work processes are retained by practitioners within their heads—the expediency and outcome of the task depends on who is performing it. There is sparse process documentation, and that too is often outdated. Most DBA tools are merely point solutions that are meant for implementing one-off decisions taken by humans, and not really capable of automating the decision-making process. As a result, quality of work across practitioners is highly inconsistent. As companies increasingly embrace off-shore IT support models, these inconsistencies further begin to stretch and rip the IT support fabric.

DBA is one of the most expensive IT line items—over $50B worldwide is spent on human capital, five times the size of the DBMS market.

DBA suffers from lack of standards . . . the outcome of the task depends on who is performing it . . . there is sparse or outdated documentation . . . Another challenge is complexity . . . complex environments are the most challenging to automate.

The other big challenge is the level of complexity found in most IT shops, especially those with multiple platforms, products and versions. This complexity is exacerbated by frequent changes brought on by user requests, change control requests and software releases occurring within the environment. Large and fast-paced environments tend to be the most complex and can benefit the most from automation. And understandably, these are the environments where implementing automation is most challenging.

There is also a prevailing tendency to equate IT automation efforts with "scripts," which severely lack any type of serious analytical capabilities or centralized deployment approach, and are difficult to maintain. Scripts tend to make certain assumptions in the target environment via the use of environment variables. Based on the values of these environment variables, the script can look at different options via a series of if-else or case statements and take appropriate action. However such logic amounts to evaluating a sequence of already-made decisions and not really making decisions on the fly. Given that environments change very often, the assumptions made by the programmer while building the script may no longer be applicable. Consequently, the decision options hard-coded in the script would not be relevant, resulting in failure. The lack of actual decision-making ability makes the script model inadequate for automating complex tasks.

At first glance, these challenges appear insurmountable, thereby making true lights-out automation a distant gleam in the eyes of most DBAs and IT managers. But these are precisely areas where DA can come to the rescue.

This is because even though database administration is complex, like other IT support areas, a large chunk can be automated provided such efforts are preceded by standard task plan definitions. DA can be leveraged to make such task plans more generic and reusable across a variety of environments. Doing so will free up human database administrators to focus on higher value tasks, reduce human errors in mundane areas, and allow documented procedures to be consistently applied, resulting in uniform output.

Case Study

Let's take an example to understand the use of DA in DBA automation—adding data files to a database. Oracle has a data-file autoextend feature wherein the underlying files that make up the database can grow without restriction to accommodate transactions. However many IT organizations have hard limits on how big their data-files can grow. These limits are placed for a variety of reasons, but mostly they are a part of IT requirements to facilitate backups to occur in the most effective manner and adhere to file system quotas established by systems adminstrators. Once these limits are enabled, when the file auto-extends and reaches that size, that part of the database (called a tablespace) fails to grow any further until a DBA adds another data-file to that tablespace—either manually upon receiving a space-bound alert or automatically via a script that is triggered in response to that alert.

We tend to equate automation with "scripts" . . . if-else logic amounts to evaluating a sequence of already-made decisions.

A typical script will check the mount-point(s) of the existing data-file(s) and other preapproved mount-points. If no free space is found in these locations, the script will exit with an error message. By leveraging DA, such inefficiencies can be a thing of the past. DA can bring about a layer of thinking and action based on available metadata and prior actions. With regard to our example, if no free space is found in the pre-approved mount-points, DA can trigger a discovery process to look for any newly allocated mount-points outside the pre-approved range; ensure they are usable (i.e., they are not NFS mounted, etc.); and in some cases, if none are found, provision additional space on the disk sub-system in real time and add the file in the best possible place. If provisioning the space requires special privileges, it can send out an email or ticket to the sys admin group, sleep in the background, check periodically for the space being available, and when it is finally available, add the file.

Via DA, the script can also decide that since it is adding the file in an unauthorized mount-point (i.e., a new mount-point that's not on the list of pre-approved names supplied to the script at run time), it can check the free-space threshold and if it is "significantly past the warning level," create a smaller than usual file to accommodate current growth in data and then email the DBA or sys admin about this event. They can then provision and add appropriate mount-points and then move the newly added file to the other mount-point. Thus, any application outages are averted by making space available in the short term as the database needs to grow and then giving the DBA/sys admins the buffer to clean up later.

The above paragraph references the words "significantly past the warning level." Fuzzy logic can be used to interpret available space and see if does meet the requirements for being classified as "significantly past the warning level." If say, 90% is the warning level and 99% is the critical level, how should 98.5% be classified? And how should 91% be classified when the current data volume growth rate is 300% higher than "normal?" A threshold check via traditional scripting tools would not trigger the "significantly past" threshold in the former case since it's 0.5% below the critical level. Also, traditional scripting tools wouldn't be able to accommodate external variables such as data growth rate in their decision making because the values are not readily accessible and are too time-consuming to compute on the fly.

As such, for DA to be truly valuable and allow lights-out automation, there's a need for the DA algorithms to be able to simulate rational human behavior when confronted with the simplest of problems (it doesn't get any simpler than adding space to a database) by being able to think, decide, and act rather than merely evaluating available choices. It needs to be able to build a new choice list and determine appropriate behavior related to each choice.[1]

The DA-Based Approach

The above need can be satisfied by leveraging an expert system. This system would handle the thinking portion and rely on a library of automated routines corresponding to task plans, to implement the most appropriate action—all based on the options available to make a decision. Each decision option is ranked based on how well it would satisfy the business requirement and accordingly, the one with the highest rank is implemented.

For the expert system to use DA techniques, it needs to know what the different variables are that need to be considered when making a decision. For instance, in the case of adding space, the primary variables are available disk space, required disk space, urgency level, and competitiveness of the objects in the tablespace. Each of these primary variables is usually dependent on several secondary variables. For instance, required disk space is a function of how much data is being added to that tablespace, how quickly it's growing compared to historical metrics, and whether this current growth is a temporary spike that will end before the tablespace completely runs out of space. These variables are often interdependent; for instance, the current growth curve could feed

(continued on page 19)

[1] Editorial comment: Examples include moving objects, compacting objects, deleting objects (e.g., unused indexes), and changing the growth characteristics of objects.

Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning

A Book Review by Mogens Nørgaard

Details

Authors: Richmond Shee, Kirtikumar Deshpande, and K. Gopalakrishnan

ISBN: 007222729X

Format: Softcover

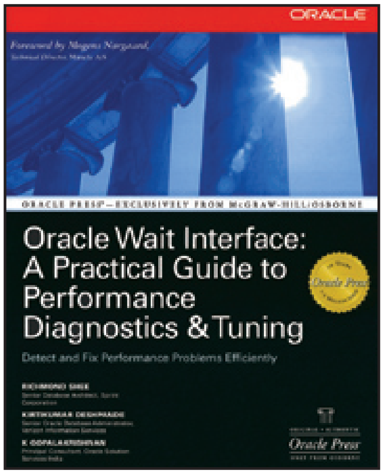
Pages: 384

Year of Publication: 2004

Edition: 1

Price: $49.99

Publisher: Oracle Press (McGraw-Hill/Osborne)


ORACLE
Foreword by Mogens Nørgaard
ORACLE PRESS™—EXCLUSIVELY FROM MCGRAW-HILL/OSBORNE
Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning
Detect and Fix Performance Problems Efficiently
RICHMOND SHEE
KIRTIKUMAR DESHPANDE
K GOPALAKRISHNAN
Oracle Press

Review*

I wish I had written this book. I don't think I could have done it as well and as thoroughly as K Gopalakrishnan (Gopal), Kirtikumar (Kirti), and Richmond, but I certainly had the opportunity to some years ago. With this book available, and with its contents being so thorough, I don't think there's a need for a second book on the Oracle Wait Interface—ever. At least I won't be writing one, that's for sure.

> *"With this book available, and with its contents being so thorough, I don't think there's a need for a second book on the Oracle Wait Interface—ever."*

(If you somehow have managed not to hear about, read about, or work with the Wait Interface, I suggest you start with Chapter 1, which is a good introduction to the topic.)

In the book *Oracle Insights: Tales of The Oak Table* (APress, 2004), there are some historic "fun facts" about how the Oracle Wait Interface came out of obscurity and into the mainstream, mainly told by Anjo Kolk, Cary Millsap, and me, so I'm not going to repeat that here.

Instead, I would like to take you on a very personal trip down Memory Lane . . .

Before I read the now-famous YAPP paper by Shari Yamaguchi and Anjo Kolk, a guy working out of Gold Support in Oracle France, Kyle Hailey, had told me in an intra-Oracle e-mail about the wait stuff and some clever tricks he was using it for, but I didn't really catch the full impact of it. (Kyle, by the way, is one of the technical reviewers of this book and is credited in one of the appendixes with his direct memory access code, and justly so.)

After reading the YAPP paper, which first championed the teaching of this method to the public, I started talking about it to everyone who cared to listen.

Whenever I had questions I would ask Anjo (we both worked at Oracle at that time), and he would usually respond within a few minutes, regardless of the time of day. In fact, I think our first contact happened a few hours after he and his family relocated from Japan to Oracle HQ, and his two sons were still suffering from jet lag.

Soon I was giving small presentations in Denmark. Then bigger presentations in Denmark. In Denmark, people will still refer to it as "Mogens' beloved Wait Interface," as if I had invented it.

Then came the presentations at various user group meetings and conferences. I couldn't talk about much else in that period of my life, and I still find the whole idea and method incredibly compelling.

I had the good fortune of being used by Lex de Haan to present Oracle Technical Seminars in many wonderful countries for a couple of years, and whatever the topic was (7.3 New Features or whatever), I always made sure the audience was also introduced to the YAPP method.

I think I presented technical seminars in close to 30 different countries, and so they heard about this wonder in Argentina, Taiwan, South Africa, Italy, and anywhere in between.

[*] Excerpted from the foreword to *Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning* by Richmond Shee et al., with permission from McGraw-Hill.

One memorable occasion was presenting at a symposium arranged by Cary Millsap in his then-role as head of Oracle's System Performance Group (SPG) in Las Vegas (in 1998 or 1999, I think). Eighty people attended my presentation (the maximum allowed) and several others wanted in—what a great feeling!

After the presentation, Cary strolled up to me and casually remarked, "I think that was the best presentation I ever saw." Imagine hearing that from one of your absolute heroes. I could have done anything in Las Vegas that night. But I didn't.

Fired up like that, it's hardly surprising that I kept talking about the Oracle Wait Interface whenever I had the chance.

Surprisingly, I was still one of the very few, if not the only one, to talk about the Wait Interface at the IOUG conference in Orlando in 2001. The maximum number of attendees allowed into the room was 400 and still 150 more wanted in.

That presentation went extremely well. Cary Millsap was taking care of my slides, the audience was in a very good mood, and the whole thing rocked. In fact, a nice guy in a wheelchair in the front of the room laughed so hard that at one point his glass of water fell to the floor.

> ***"Gopal, Kirti, and Richmond have made sure that their book contains essential details about the use of the wait information in Oracle 10g, while at the same time describing how to use it, what to look out for, and how to understand its output from Oracle 7.0.12 and up to the present."***

During the same year, Gaja Krishna Vaidyanatha and Kirti wrote a fine performance book, *Oracle Performance Tuning 101* (McGraw-Hill/Osborne, 2001), in which Gaja introduced the unforgettable phrase "Compulsive Tuning Disorder." He should also be credited with spreading the word about the Wait Interface with great enthusiasm.

I didn't participate at the recent IOUG conference in Toronto (2004), but there were a lot of presentations about the Wait Interface. The Oracle Wait Interface has truly entered the mainstream of Oracle knowledge, and I must find other things to talk about.

Cary Millsap and Jeff Holt of Hotsos have also written a very fine book, *Optimizing Oracle Performance* (O'Reilly, 2003), on the topic of Oracle performance methodology (including a chapter on queueing theory, which was invented in 1909 by a bored Dane, Agnar Erlang, who worked for the phone monopoly—one day I'll understand it).

So with *Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning* fully (and I mean fully!) documenting the Wait Interface, its pitfalls and joys, and many tips and tricks, I think the topic is covered.

This is why I have recently taken up talking to SQL Server audiences about the virtues of a Wait Interface. And you know what? They ask me whether there's a book available on the topic.

> ***"Until such time when we can all safely forget about performance problems, this book ought to be the preferred reference on this topic."***

The timing of this book is perfect. Any other time would also have been perfect if it wasn't for one important detail: With Oracle Database 10*g*, Oracle (really Graham Wood, the chief architect of the Manageability area in Oracle Development and another technical reviewer of this book) is finally taking full advantage of the Wait Interface. Oracle is recording the right things, storing them correctly in a repository, and using advisory services and other utilities to use the information to its fullest.

Gopal, Kirti, and Richmond have made sure that their book contains essential details about the usage of the wait information in Oracle Database 10*g*, while at the same time describing how to use it, what to look out for, and how to understand its output from Oracle 7.0.12 and up to the present. A very impressive piece of work.

The promise of all this is that optimization can be automated, but that's a bit out in the future. Until such time when we can all safely forget about performance problems, this book ought to be the preferred reference on this topic.

If it weren't for the fact that I get a copy for free, I would buy this book.

About the Authors

Richmond Shee is a senior database architect for Sprint. Kirtikumar Deshpande is a senior Oracle database administrator for Verizon and the co-author of *Oracle Performance Tuning 101*. K. Gopalakrishnan is a principal consultant for Oracle India and the author of *Oracle Database 10*g *Real Application Clusters Handbook*. ▲

*Mogens Nørgaard (***mno@miracleas.dk***) is the CEO of Miracle A/S, a database knowledge center and consulting/training company based in Denmark, and is the co-founder and "father figure" of the Oak Table network. He is a renowned speaker at Oracle conferences all over the world and organizes some highly respected events through Miracle A/S, including the annual Master Class and the Miracle Database Forum. He is also the co-founder of the Danish Oracle User Group (OUGKD) and was voted "Educator of the Year" in* Oracle Magazine*'s Editor's Choice Awards, 2003.*

Stay Focused on the Five Steps

by Chris Lawson

Chris Lawson

Sometimes performance problems seem overwhelming. Some designers invent SQL so complex that it's difficult to know where to start. Here's a time-tested suggestion: *When faced with SQL statements several pages long, keep focused on the five steps of the tuning process.*

Define the Problem. When we visit a doctor, the doctor soon asks, "What seems to be the trouble?" In medical clinics, this is called the "chief complaint." The DBA must do the same thing. We must ask the users, "What is the complaint?" Then we must follow up to get the details. For instance, we may ask the question, "How long is the query delay?" or "When does the problem occur?"

Investigate and Gather Facts. After defining the problem, the next step is investigation. Here, the DBA changes hats from physician to detective. The main objective in this step is to re-create and quantify the problem. Some questions that should be asked are "What is the elapsed time of the query?" or "How many disk/logical reads are performed?" Watch the end user as he executes the program. Get to know a little about how the application works. A side benefit of this is that the user will understand that you are serious about solving the problem and will appreciate your interest.

> ***"When faced with SQL statements several pages long, keep focused on the five steps of the tuning process."***

Find the Root Cause. Now that we have identified what the problem is and have quantified the problem, we are ready to find the root cause. In this role, the DBA wears the hat of a pathologist. We try to find the disease that is the primary cause of the performance problem. Note that there is no need to guess at the root cause. In fact, speculation not based on the facts should be discouraged. Instead, results from the previous steps will help focus attention on the key problem areas.

Devise a Solution. Instead of analyzing, we now synthesize. Now is the time to be creative and imagine the perfect solution that addresses the root cause in the simplest fashion possible. Is the solution just a matter of adding a missing index, or is a more complex solution required?

Implement and Confirm the Fix. This is a simple and fun step. All the hard work has been done, and we have a solution that truly addresses the performance bottleneck. It's a good idea to test the solution in a sandbox and also to clearly document the solution with "before" and "after" statistics.

To illustrate the value of following this logical process, we present here a case study involving a complex SQL statement having a few "twists." Let's walk through the process to solve this problem.

Step 1: Define the Problem

The users at a large engineering construction firm complained that a query for a single account was taking five minutes, even though the query always returned only a few rows. The users provided a trace file that confirmed the long runtime. A quick check confirmed that the query indeed took several minutes. The SQL in question retrieves a list of equipment transfers for a particular account, originating from two different equipment inventories. The SQL must also identify the earliest and latest transfers.

The original query was very intimidating, and encompassed several pages of code. Like most SQL, however, we can simplify it by temporarily omitting the actual select list. This simplified SQL is shown below:

```
Select [huge list], 6 “scalar subqueries”
FROM SCHEDULE S, EQUIP_INV1 INV1, EQUIP_INV2
INV2,
EQUIP_POOL EQ1, EQUIP_POOL EQ2, IV1, IV2
AND S.INV_ID = INV1.INV_ID
AND S.INV_ID = INV2.INV_ID
AND INV1.EQUIPMENT_ID = EQ1.EQUIPMENT_ID
AND INV2.EQUIPMENT_ID = EQ2.EQUIPMENT_ID
AND S.SCH_NUM = IV1.SCH_NUM(+)
AND S.SCH_NUM = IV2.SCH_NUM(+)
AND (EQ1.EQUIPMENT_ID = :b1 OR EQ2.EQUIPMENT_ID = :b1)
```

In the above SQL, recall that a scalar subquery is a subquery located in the Select list (sort of like an inline view, but part of the Select clause, rather than the From clause.) To keep the SQL readable, I don't list the details for the two inline views, IV1 and IV2, but just show them in the From clause. Within each inline view, analytical functions (not shown) query the huge EQUIPMENT_HISTORY table to find the earliest and latest transfers.

Step 2: Investigate and Gather Facts

This query is still pretty complicated, so let's simplify it further. The main part of the query simply retrieves all transfers for a piece of equipment. So let's query from just one of the equipment pools, and remove the two inline views and the "OR" condition in the very last line. Here's the new code:

```
Select *
FROM SCHEDULE S, EQUIP_INV1 INV1, EQUIP_POOL EQ1
AND S.INV_ID = INV1.INV_ID
AND INV1.EQUIPMENT_ID = EQ1.EQUIPMENT_ID
AND EQ1.EQUIPMENT_ID = [sample value]
```

This simplified query returns a few rows, but still takes way too long—30 seconds. The execution plan shows that Oracle performs a full scan of the leading table—the Equip_Pool table. Although this delay doesn't account for the full five-minute delay, we have already uncovered one significant problem—a missing index. After adding an index on Equip_Pool(Equipment_Id), the miniquery completes in less than one second, and only consumes 35 reads.

The results so far are excellent. If we can retrieve all relevant rows in less than one second, the final query should be similarly fast. Remember that the final query is simply various manipulations (e.g., calculate earlier and latest transfers) of the core result set.

Step 3: Find the Root Cause

Although the new index solved one problem, we still don't know why the original query required several minutes. Reviewing the trace file for the original query, we see something very suspicious. The step related to one of the inline views shows that millions of rows were processed, along with a full scan of the EQUIPMENT_HISTORY table. This suggests that the inline view is not just finding the latest transfer for the specified equipment—it must be finding the transfers for all of the equipment!

```
Rows       Row Source Operation
-------    ---------------------------------
4116400    VIEW (cr=108339 pr=81770 pw=64873
time=176454866 us)
4116400      WINDOW SORT (cr=108339 pr=81770
pw=64873 time=172338458 us)
4116400        TABLE ACCESS FULL EQUIPMENT_HISTORY
(cr=108339 pr=16896 pw=0 time=16465877 us)
```

To confirm that the inline views are really the root problem, I modify the miniquery, adding back the suspect inline view. As suspected, the miniquery degrades substantially. This confirms the root cause of the bottleneck. Now, what do we do about it?

Step 4: Devise a Solution

So far, to keep things simple, we haven't examined how the inline views are defined. Each inline view is actually a view within a view, with an analytical function thrown in just to make things interesting. We've already shown that the optimizer is sufficiently confused that it simply scans the whole Equipment_History table, finding the latest date for all equipment. Afterwards, it throws away 99.9% of the results.

Our fix, then, must allow the inline view to quickly find just the desired rows—that is, the rows that match the join condition SCH_NUM. So our solution must do two new things: first, find the applicable SCH_NUM; second, apply this value in the inline view of interest.

Assume for the moment that the desired SCH_NUM value will be held in a table called Prelim. Knowing this, we modify the inline view, joining this value to the huge history table. So the troublesome inline view will now look something like this:

```
Select * from Equipment_History T, Prelim P
Where T.SCH_NUM = P.SCH_NUM
```

Now, let's see how to find the desired SCH_NUM. To identify and save the value for SCH_NUM, we will need to prerun the essential SQL. We'll use a technique called "query subfactoring" (often called the "with syntax.") In the SQL below, the object "PRELIM" holds the SCH_NUM of interest. The main body of the query goes right after the "with" part:

```
WITH PRELIM AS (SELECT SCH_NUM
FROM SCHEDULE S, EQUIP_INV1 INV1, EQUIP_INV2
INV2,
EQUIP_POOL EQ1, EQUIP_POOL EQ2
AND S.INV_ID = INV1.INV_ID
AND S.INV_ID = INV2.INV_ID
AND INV1.EQUIPMENT_ID = EQ1.EQUIPMENT_ID
AND INV2.EQUIPMENT_ID = EQ2.EQUIPMENT_ID
AND (EQ1.EQUIPMENT_ID = :b1 OR EQ2.EQUIPMENT_ID = :b1)
--
[main query goes here]
```

Step 5: Implement and Confirm the Fix

Running the new, simplified query for a sample equipment confirms a 1.5 second runtime, with only 200 logical reads. By adding the restrictive condition directly inside the inline view, Oracle is able reduce its scope of work to just a single equipment (and schedule.) Admittedly, the query looks a bit more complicated due to the query subfactoring, but the 100x faster runtime makes this a good trade!

Concluding Remarks

I have found that few problems are efficiently resolved by relying solely on one approach. The best DBAs and performance specialists I know use a tool kit of different approaches. For any given assignment, these experts adapt their heuristics to fit the circumstances. If necessary, they develop new approaches. However our case study has illustrated a consistent, time-proven *framework* to solve complex performance issues. Instead of trying to guess at solutions, we emphasize identifying root cause first. By following this logical framework, you'll find the performance tuning process to be a lot less frustrating. You'll actually look forward to solving complex SQL statements that have baffled your colleagues! ▲

Chris Lawson is an Oracle DBA consultant in the San Francisco Bay Area, where he specializes in performance tuning of data warehouse and financial applications. He is a frequent speaker at NoCOUG, and has written for a number of publications such as Oracle Internals, Exploring Oracle, SELECT, Oracle Informant, *and* Intelligent Enterprise. *Chris has held a variety of positions in the IT field—ranging from systems engineer to department manager—and is an instructor for the University of Phoenix. He can be contacted via* **www.oraclemagician.com**.

Hello, World! An Introduction to PHP

by David Schleis

Life is really simple, but we insist on making it complicated! —Confucius

The use of the Web to deliver applications does not appear to be a fad that will be disappearing any time soon. In fact, with the marketing of software as a service, the importance of web-based solutions will probably only increase. There are a number of programming languages available to create Oracle-backed web-based applications, each with its own advantages and disadvantages that must be weighed when deciding which to use. Beyond the hardware and software costs that may be associated with the new technology, a critical consideration is the amount of time and effort it will take to add the necessary skills to become proficient and productive in the new environment. A language that consistently scores high when taking all of these factors into account is PHP.

Introduction to PHP

PHP is a powerful and versatile open-source programming language that provides "write once, run anywhere" capabilities for both web-based and desktop applications. As of late 2006, PHP was running on 40% of Apache web servers around the world, making it by far the most popular add-on module to the Internet's most popular web server.

PHP stands for "PHP Hypertext Preprocessor." However, since this recursive definition may not satisfy everybody, we should note that PHP was originally created by Rasmus Lerdorf in 1994 as a means of monitoring interest in his online resume. Lerdorf freely distributed this set of "Personal Home Page" tools, and from these humble beginnings PHP has evolved into a remarkably useful and user-friendly "server-side" language.

Why PHP?

The question as to why you should use PHP can be answered with one word. *Simple*. Simple installation, simple configuration, simple architecture, simple deployment, simple syntax, simple licensing, just plain *simple*. But if a one-word answer is not enough, how about *two* words: *availability* and *functionality*.

> *"The question as why you should use PHP can be answered with one word. Simple."*

PHP's first measure of availability is its cost, or lack thereof. PHP, along with its extensive manual, available in 24 languages, is free for the downloading. In addition to the manual, there is a great deal of support freely available from the online community in the form of lists, newsgroups, tutorials, how-to articles and more usable code than you could ever actually use.

Another aspect of the availability of PHP is that the odds are that, whatever operating system or web server you are currently running, you can run PHP today.

The final characteristic defining its availability is that the language is easy to learn and use. While offering many advanced features for the experienced programmer, PHP has a very short learning curve, allowing you to see results immediately and be productive sooner.

To the question "What can PHP do?" the manual immodestly answers "Anything." Admittedly, that is a bit vague. I will try to be more specific, but in the space provided I will only be able to hint at the capabilities of this language. It should be noted that although primarily used to create web-based applications, PHP can also be used in a command-line mode functioning like AWK or Perl, or to write cross-platform GUI applications using the PHP-GTK extension.

> *"To the question 'What can PHP do?' the manual immodestly answers 'Anything.'"*

PHP has native support for about 20 databases as well as ODBC. Additionally, the database abstraction library, ADOdb, is available to provide your application with a more agnostic view of the data world. PHP programs can provide as much or as little access to operating system commands and the file system as you desire. The language also provides useful text-processing features including regular expression pattern matching, reading and writing of compressed files, and the creation of PDF files. There is also extensive support for XML and its alphabet soup of related technologies. One noteworthy example of this support is the SimpleXML functions that provide an easy and efficient means of processing XML documents, allowing them to be treated much like an array.

Communication with other services is supported via COM, HTTP, IMAP, LDAP, POP3, and others. And if the needed communication protocol is not supported, a raw network socket can be opened to allow interaction with any other protocol. You can also call Java and .Net objects from within your PHP code using the PHP/Java Bridge.

There are, of course, many other aspects of its functionality that I have not mentioned in this limited space. Please visit the references in the Further Reading section to find out more.

The PHP Architecture

Like other aspects of the language, PHP's architecture is extremely flexible. To start with, it allows applications to be written following an object-oriented (OO) or procedural model, or even a combination of both. Sometimes OO is overkill. A simple problem begs for a simple solution, and PHP gives you the flexibility to decide the best approach. When creating a web application, PHP can be used in several ways. The simplest of these is akin to a Microsoft Active Server Page (ASP) or Java Server Page (JSP), which is basically an HTML page with server-side programming commands enclosed within a set of special tags. This embedded code model is far more intuitive than generating markup from within an application, and easier for those familiar with HTML to use to create applications. PHP can also be used to support a model-view-controller (MVC) architecture. In this situation, the view could be in the form of Savant or Smarty templates. The model is encapsulated in PHP objects that use either ADOdb or specific database calls, and the controller would be contained in PHP objects. These are just two examples of the continuum of architecture options available. Other choices are defined in the many frameworks available for assisting you in constructing your PHP applications.

> ***"The simplest [way to use PHP] is...an HTML page with server-side programming commands enclosed within a set of special tags."***

Getting Started with PHP

The first step to using PHP is obviously to get it installed on your machine. If you intend to connect to an Oracle database, you will also need the Oracle Instant Client. Installation instructions for both of these products, for Windows and Linux, can be found at the Oracle Technology Network PHP Developer Center (see entry in Further Reading). The installation on either platform is quite painless, but a gotcha I found when doing my last install is that, if you are using PHP 5.1.2 or above, only the Oracle 10*g* client libraries will work, no matter which version of the database you are connecting to.

There is no specific Integrated Development Environment (IDE) required for PHP development, but there are some available. In continuing with the availability theme, one IDE is PHPeclipse; a plug-in for the popular Eclipse open-source development platform. There are, of course, others available to make the job of writing code easier by providing features such as context-sensitive help and debugging, but development could be done using any text editor.

PHP originated as an embedded scripting language, meaning that the parser will simply pass along any text not enclosed within the tags "<?php" and "?>." The simple script:

```
<HTML><body><?php print “Hello, World!”;?></body></HTML>
```

is transformed to read

```
<HTML><body>Hello, World!</body></HTML>
```

after parsing. Although this is a trivial example, it hints at the ease of incorporating PHP with HTML.

The coding syntax of PHP is similar to AWK, C, or Unix shell scripting. Like C or Java, instructions must be terminated with a semicolon. Comments can be added to the scripts using Java/C++ (//, /* . . . */) or Unix shell (#) style comment indicators.

Variable names take the form of a dollar sign followed by a series of letters, numbers, and underscores. The variable names are case sensitive and cannot begin with a number. PHP variables do not have to, and in fact cannot, be declared as a specific type. A variable will change type based on the context in which it is being used. Assign "3" to a variable, and its type is string. Assign 3, and its type is integer. PHP provides functions to determine the type of a variable at any given instant, and also to set the type of a variable to a given type.

Functions in PHP are defined using the syntax:

```
function fn ($arg1, &$byRefArg, $optArg=”anything”)
{
 // any valid PHP code
 return $something; // or not
}
```

Data may be passed to the function via an argument list that consists of comma-separated variables or constants. By default, arguments are passed to the function by value, but they can also be passed by reference. Also, optional arguments can be indicated by providing a default value. One of the bothersome historical aspects of the language is that function names, unlike variable names are not case sensitive.

While the nearly 80 built-in array functions are not specific to Oracle, they provide the database developer with incredible flexibility. The array data type maps a value of any data type, including another array, to a key that can be either a positive integer or a string. This allows for the implementation of a wide variety of data structures.

This brief description of some of the basic aspects of the PHP language should help in your understanding of the next section.

The First Program

The following program is designed to run on a web server, and will display "Hello, World!" in the browser. Additional notes about the code follow the listing.

```
<html> <body>
<?php
 function showError($Stmt=FALSE) {

  /* put the error details into an array.
    Call the function differently based on
    the existence of a statement object */
  $e = ($Stmt ? oci_error($Stmt) : oci_error());


  print htmlentities($e[‘sqltext’]);
 }

 // create a connection to an oracle instance
```

(continued on page 20)

DBArtisan from Embarcadero

by Scott Walz

Introduction by Iggy Fernandez

A workman is only as good as his tools, so goes the old trueism. Alton Brown could not bake muffins half as fast without his trusty KitchenAid. Robin Hood would not be half as merry without his trusty bow. Sir Winston and his brave band of Brits would not have been able to finish the job if we hadn't given them the right tools.

Certainly, I'm sure it's possible to be a crackerjack database administrator or software developer with lousy tools. Our Neanderthal ancestors used pointy sticks and stones to make meatloaf out of the mighty mastodon, and the Egyptians built their amazing pyramids with little more than elbow grease and spit. But in today's fast-paced world, you'd better have a good bike if you want to keep pace with the peloton.

What tools do I want for Christmas? For starters, I'd like some really good documentation on the myriad databases that I need to work on—where are those MetaLink credentials when I need them? I'd like a repository of performance data and some decent graphing and reporting tools that I can use to poke under the covers. But, most of all, I'd like a really good GUI tool to aid in database administration tasks.

When I first started out as a DBA many winters ago, I had to keep track of less than half a dozen databases, all the same make and model, and none more than 100 megabytes in size. These days, I may be called on to work on any one of hundreds of databases, from the antediluvian to the futuristic—Oracle 8i, Oracle 9i, Oracle 10g, Dataguard, RAC, ASM—and the smallest of them is 100 gigabytes in size. To make matters worse, I'm also expected to work on SQL Server and DB2 databases when the need arises. O tempora, o mores!

When I was one and twenty, I heard a wise man say that a workman was only as good as his tools, but I wouldn't complain if the only tools they let me have were called "svrmgrl" and "sqlplus." I was one and twenty—no use to talk to me.

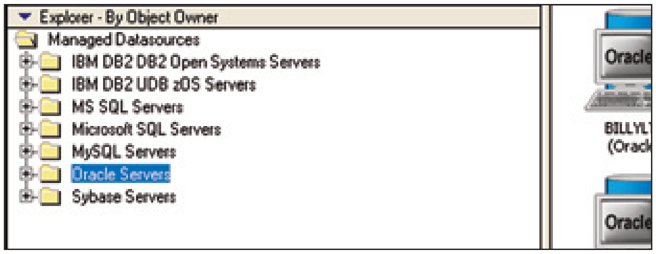
I'm older and wiser now and I realize that to consistently perform the Herculean feats my customers want to see, I need good tools. Certainly, they cost crowns and pounds and guineas, but it ought to be really easy to measure the return on investment—simply multiply the expected productivity gain by the labor cost and compare with the price of the tools. Or think of it as a little insurance against costly mistakes.

This year I have a special request to Santa, not for a "Motorola C139 with texting, games and graphics," but for DBArtisan from Embarcadero. From a single GUI console, I want to be able to reach all my databases—Oracle and DB2, SQL Server and MySQL—and I want to be able to perform routine tasks without having to type clumsy commands. What's that magic spell to change a user's password in a SQL Server database—my neurons are short-circuiting from information overload. Help!

Santa better be listening.

Cross-Platform Database Administration—The DBArtisan Way

While standing on the floor of a trade show recently, I was asked a question regarding the administration of cross-platform databases. A person new to the industry asked, "How much different can the other platforms be to administer?" I responded with an off-the-cuff "Well, they both have tables full of data. I can't imagine it being that different." As we talked more, we got into the differences between the platforms that can make the job of a database administrator challenging—challenging, but not impossible.



Most Oracle DBAs spend most of their time working with OEM. But, if you get one of the other databases platforms "thrown over the wall" for you to manage, DBArtisan just might become your new best friend.

Under the Same Roof

You've heard the expression, "one-stop shopping" or "under the same roof." Those very expressions come into play when discussing DBArtisan. As mentioned above, the database vendors provide their own tools, but that requires different applications being open and navigating between them. With a tool that provides all administration tasks from one client, you reduce the need to have those other tools, not to mention learning the different nuances of each.

A benefit from working within one tool is the ability to use your experience on Oracle and transition over to the new platform. For example, though each database platform has specific parameters when creating an object, cross-platform

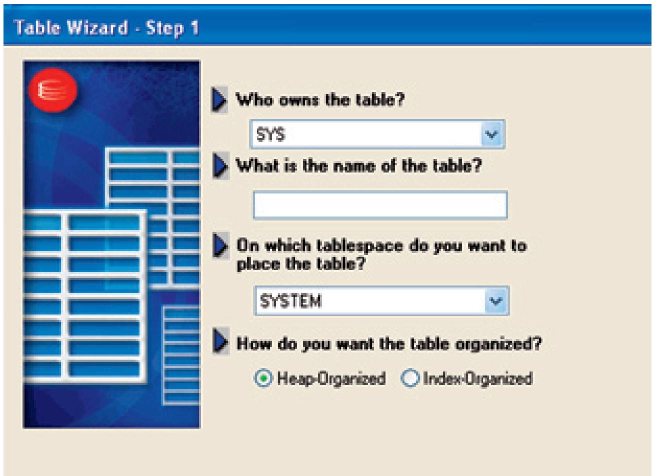


Table Wizard - Step 1

Who owns the table?

SYS

What is the name of the table?

On which tablespace do you want to place the table?

SYSTEM

How do you want the table organized?

● Heap-Organized ○ Index-Organized

tools provide the ability to step through wizards and present the options to you, giving you a breather from having to have that knowledge right out of the gate. The wizard might have eight steps for Oracle but only four steps for Sybase—the wizard handles the complexity for you.

DBArtisan also allows you to keep all of your scripts in a central repository and execute them from a single console. It also has the ability to interface with a job scheduler, making the steps from creation to scheduling to verifying as simple as 1-2-3. And, by relying on its cross-platform power, you'll get assistance in creating your jobs. If it's simply an updating of statistics for a table or an advanced online reorganization, relying on DBArtisan removes the guesswork.

Rolling Up Your Sleeves

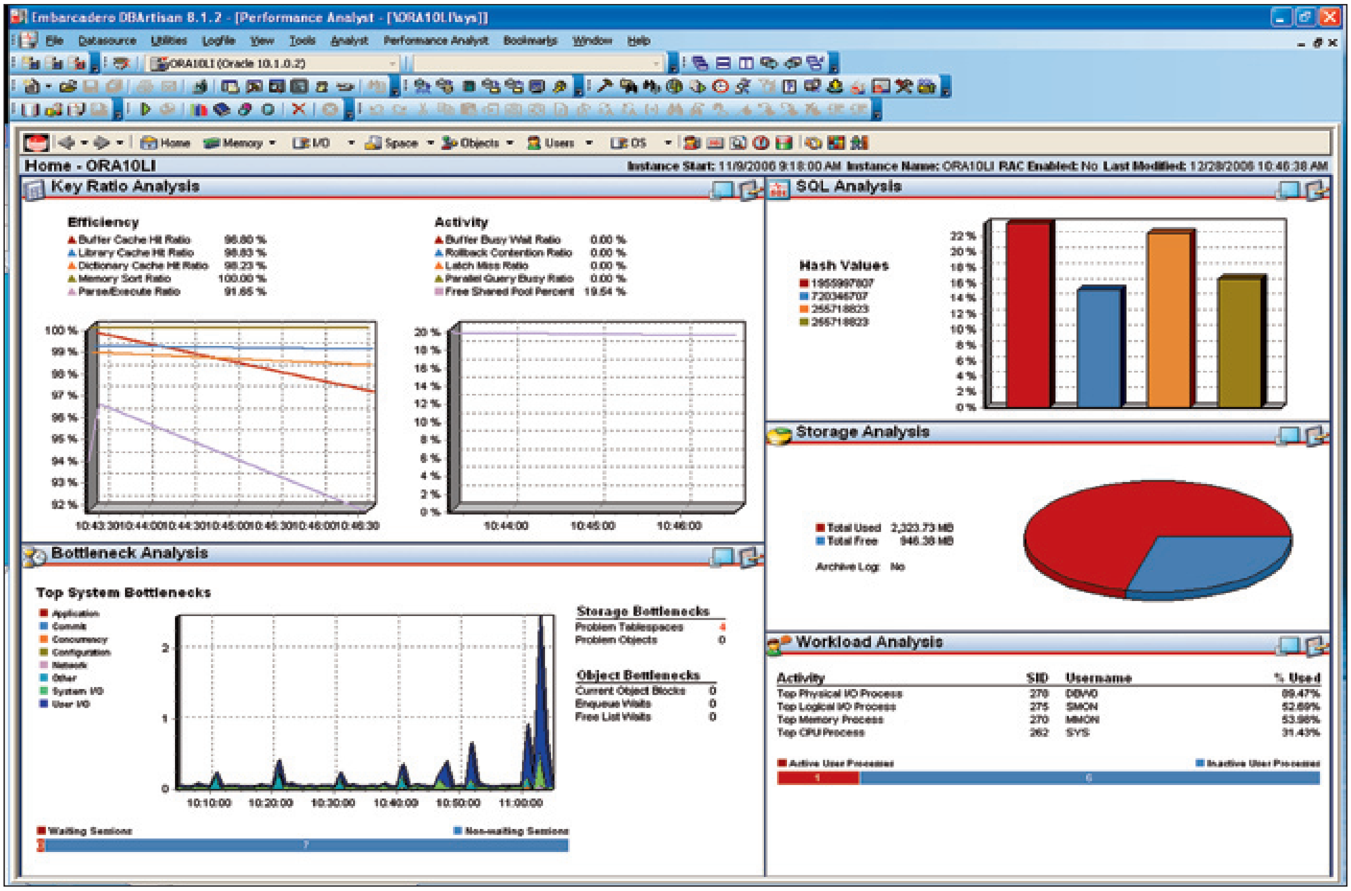
Though daily administration tasks are important, it's usually only a matter of time before you have to roll up your sleeves and really get down into the database. It might be second nature to identify performance bottlenecks on your native platform, but what about the new one? What system table or view do you need to access to see the information you need? How about space issues? Are the devices large enough to handle your data growth? These are all questions that DBArtisan can answer for you, taking the guesswork out of determining where the issues might reside. Say good-bye to spending time thumbing through manuals or researching on the Web, DBArtisan puts the information you need at your fingertips, along with the ability to set alarms and thresholds to alert you outside of the application via email or pager or to just change the color of the data to make it easy to pick up on the screen.

Though identifying the issues is the first part of the equation, resolving them is what your job depends on. DBArtisan takes you from problem identification to resolution in just a mouse click. By accessing the context menu, you can navigate into the appropriate object editor to correct the problem. DBArtisan's Workbench series includes Performance Analyst, Space Analyst, and Capacity Analyst, for those deep dives into the database.

Summary

As you continue to be asked to do more with less, you'll find that incorporating DBArtisan into your job will not only benefit you, but your company as well. ▲

Scott Walz is a senior product manager at Embarcadero Technologies, with primary responsibility for database availability tools. For more information about DBArtisan, visit **www.embarcadero.com/products/dbartisan**.



Embarcadero DBArtisan 8.1.2 - [Performance Analyst - [\ORA10LI\sys]]
File Datasource Utilities Logfile View Tools Analyst Performance Analyst Bookmarks Window Help
ORA10LI (Oracle 10.1.0.2)
Home Memory I/O Space Objects Users OS

Home - ORA10LI Instance Start: 11/9/2006 9:18:00 AM Instance Name: ORA10LI RAC Enabled: No Last Modified: 12/28/2006 10:46:38 AM

Key Ratio Analysis

Efficiency	
Buffer Cache Hit Ratio	96.80 %
Library Cache Hit Ratio	98.83 %
Dictionary Cache Hit Ratio	98.23 %
Memory Sort Ratio	100.00 %
Parse/Execute Ratio	91.65 %

Activity	
Buffer Busy Wait Ratio	0.00 %
Rollback Contention Ratio	0.00 %
Latch Miss Ratio	0.00 %
Parallel Query Busy Ratio	0.00 %
Free Shared Pool Percent	19.54 %

SQL Analysis

Storage Analysis

Total Used	2,323.73 MB
Total Free	946.38 MB

Archive Log: No

Bottleneck Analysis

Top System Bottlenecks

Storage Bottlenecks	
Problem Tablespaces	4
Problem Objects	0

Object Bottlenecks	
Current Object Blocks	0
Enqueue Waits	0
Free List Waits	0

Workload Analysis

Activity	SID	Username	% Used
Top Physical I/O Process	270	DBW0	89.47%
Top Logical I/O Process	275	SMON	52.69%
Top Memory Process	270	MMON	53.98%
Top CPU Process	262	SYS	31.43%

Lies—Damned Lies—and SQL!

by Iggy Fernandez

Iggy Fernandez

After all, facts are facts, and although we may quote one to another with a chuckle the words of the Wise Statesman, "Lies—damned lies—and statistics," still there are some easy figures the simplest must understand, and the astutest cannot wriggle out of.—Leonard Henry Courtenay

Recap

In the May issue of the *NoCOUG Journal*, we offered a prize of a SanDisk Sansa M240 1 GB MP3 Player for the best solution to a puzzle sent to us by Sumit Sengupta from Columbus, OH. The not-so-good news is that we were underwhelmed by the number of replies. The good news is that the few replies we received were extremely creative, proving, once again, that "there are more things in heaven and earth than are dreamt of in our philosophy." *The prize goes to Roger Schrag for his creative solution based on stale materialized views.*

The Puzzle

Describe a combination of circumstances in which the COUNT function could produce the seemingly anomalous results seen in the example below.

```
SQL> DESCRIBE employees;

Name             Null?     Type
---------------- --------- ----------------
NAME             NOT NULL  VARCHAR2(25)
SALARY           NOT NULL  NUMBER(8,2)
COMMISSION_PCT   NOT NULL  NUMBER(4,2)

SQL> SELECT COUNT(name) FROM employees;

COUNT(NAME)
-----------
          3

SQL> SELECT COUNT(salary) FROM employees;

COUNT(SALARY)
-------------
            2

SQL> SELECT COUNT(commission_pct) FROM employees;

COUNT(COMMISSION_PCT)
---------------------
                    1

SQL> SELECT COUNT(1) FROM employees;

  COUNT(1)
----------
         0
```

The First Solution—The Magician's Nephew

Chris Lawson, NoCOUG member and author of *The Art and Science of Oracle Performance Tuning*, gave me an object lesson in Oracle's transaction management principles by simply observing that, since readers and writers do not block each other (by acquiring locks on the data) as they do in other database architectures, other transactions could conceivably be in the process of deleting the same rows that we are reading. The result of each of our queries depends on the "isolation level" that we have selected. If we use the *default* isolation level of "READ COMMITTED," the result of any of our queries will reflect the state of our database at the precise moment that query started, and the results are therefore *not* guaranteed to be repeatable for the duration of our transaction. If we choose the stricter isolation level of "SERIALIZABLE," the result of each of our queries will reflect the state of our database at the precise moment our *transaction* started, and the results are guaranteed to be repeatable for the duration of our transaction. Chris observed that the seemingly anomalous results produced by our sequence of queries could be the consequence of operating at the default READ COMMITTED isolation level and finding that the "magician's nephew" had deleted a row (and committed his work) every time we queried the database.

A detailed treatment of transaction isolation levels can be found in Chapter 13 of the *Oracle 10*g *Concepts Manual*, available at **www.oracle.com/technology/documentation**.

The Second Solution—A Series of Unfortunate Events

While Chris Lawson should receive high marks for his eminently pedagogical response, his solution did not require the use of four different queries—we could simply have repeated the first query four times and produced four different results. It also requires the assistance of a willing coconspirator. NoCOUG member, Roger Schrag, invented a solution that makes use of stale materialized views and does not require a secret assistant hiding behind the looking glass. "Curiouser and curiouser!" I cried when I received Roger's solution (I was so surprised, that for the moment I quite forgot how to speak good English).

A *materialized* view is a way of improving the performance of joins and aggregations in SQL queries. Unlike a regular view, which exists simply as a *definition* stored in the Oracle data dictionary and which helps with data security and "logical data independence," a *materialized* view actually contains real data and occupies storage space. One can retrieve data directly from a materialized view, but Oracle is capable of transparently rewriting queries to take advantage of any materialized views that have been created!

A materialized view can be kept current using the "REFRESH ON COMMIT" option, which forces the materialized view to be updated every time the constituent data tables are updated. However, not all materialized views can be kept current in this way; for details, refer to the *Oracle 10*g *Data Warehousing Guide*, available at **www.oracle.com/technology/documentation**. Further, Oracle will not always rewrite queries to take advantage of materialized views. In particular, stale materialized views will be used by Oracle only if the query is willing to tolerate the staleness in the interest of efficient performance.

A very good overview of Materialized View technology can be found in a white paper by Dr. Lilian Hobbs, available at **www.oracle.com/technology/products/oracle9i/pdf/o9i_mv.pdf**. The following extract is a nice summary.

> *Today, organizations using their own summaries waste a significant amount of time manually creating summaries, identifying which ones to create, indexing the summaries, updating them and advising their users on which ones to use.*
>
> *Now the DBA will only have to initially create the materialized view, it can then be automatically updated whenever changes occur to its data source. There is also a Summary Advisor component which will recommend to the DBA which materialized views to create, delete and retain.*
>
> *One of the biggest benefits of using materialized views will be seen by the users of the data warehouse or database. No longer will they have to be told by the DBA which materialized views exist. Instead, they can write their query against the tables or views in the database. Then the query rewrite mechanism in the Oracle server will automatically re-write the SQL query to use the materialized views.*

With that short introduction, here is Roger's solution. He uses materialized views that are eligible for automatic refresh but purposely chooses not to take advantage of it. He does not require a secret assistant during the performance (as is the case with Chris Lawson's solution) but the setup does require a "series of unfortunate events."

```
SQL> ALTER SESSION SET optimizer_goal = FIRST_ROWS;

SQL> ALTER SESSION SET query_rewrite_enabled = TRUE;

SQL> ALTER SESSION SET query_rewrite_integrity =
STALE_TOLERATED;

SQL> CREATE TABLE employees (
  2          name VARCHAR2(25) NOT NULL,
  3          salary NUMBER(8,2) NOT NULL,
  4          commission_pct NUMBER(4,2) NOT NULL
  5  );

SQL> INSERT INTO employees VALUES (‘Roger’, 1.02,
0.00);

SQL> CREATE MATERIALIZED VIEW employees_mv1 ENABLE
QUERY REWRITE AS
  2  SELECT COUNT (commission_pct) FROM employees;

SQL> INSERT INTO employees VALUES (‘Darrin’,  2.01,
0.00);

SQL> CREATE MATERIALIZED VIEW employees_mv2
ENABLE QUERY REWRITE AS
  2  SELECT COUNT (salary) FROM employees;

SQL> INSERT INTO employees VALUES (‘Iggy’,
100000, 10.00);

SQL> CREATE MATERIALIZED VIEW employees_mv3
ENABLE QUERY REWRITE AS
  2  SELECT COUNT (name) FROM employees;

SQL> DELETE FROM employees;

SQL> COMMIT;
```

The Third Solution—Blue Smoke and Mirrors

Roger Schrag's solution was very creative but, like Chris Lawson's solution, it requires that the data be physically deleted. The original solution sent to us by the puzzle's creator illustrates the use of "fine grained access control" (often referred to as Virtual Private Database or VPD) to *hide* sensitive data from unprivileged users. Specifically, the solution uses techniques made available in Oracle 10*g*—a different restriction can be attached to each *column* of a table. Oracle automatically and silently appends these restrictions to all queries involving the columns in question. For more details, refer to the *Oracle 10*g *Security Guide*, available at **www.oracle.com/technology/documentation**. "COUNT(1)"—which is equivalent to "COUNT(*)"—will cause all of the column-based restrictions to be appended to your query. The following quote puts things in a nutshell.

> *Column-level VPD policies provides [sic] more fine-grained access controls on data. With column-level VPD, security policies can be applied only where a particular column or columns are accessed in the user's query. This means that when a user has rights to access the object itself, VPD can limit the individual rows returned only if the columns the user accesses contain sensitive information, such as salaries, or national identity numbers.*
>
> *The default behavior of column-level VPD restricts the number of rows returned when a query addresses columns containing sensitive data. In contrast, column masking behavior allows all rows to be returned for a query against data protected by column-level VPD, but the columns that contain sensitive information are returned as NULL values. With column masking, users see all the data they are supposed to see, but privacy is not compromised.*

A good introduction to VPD and the Oracle 10*g* enhancements in particular can be found in Part 14 of *Oracle Magazine*'s 2003 DBA of the Year Arup Nanda's survey of the new features of Oracle 10*g*, available at download at time of writing from **www.oracle.com/technology/pub/articles/10gdba/week14_10gdba.html**.

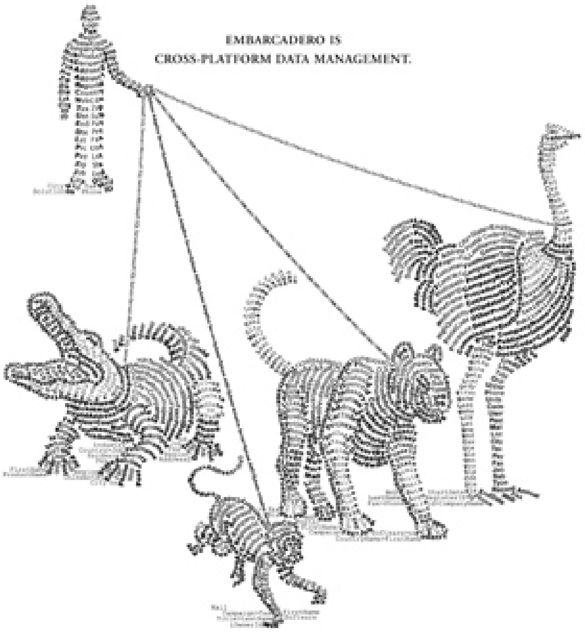
Here now is the original solution provided by Sumit. There are exactly three rows of data in the Employees table *throughout the exercise* but the combination of fine-grained access control policies results in the seemingly anomalous results described in the puzzle.

```
SQL> CREATE TABLE employees (
 2            name VARCHAR2(25) NOT NULL,
 3            salary NUMBER(8,2) NOT NULL,
 4            commission_pct NUMBER(4,2) NOT NULL
 5 );

SQL> INSERT INTO employees VALUES ('WINKEN', 1000, 30);

SQL> INSERT INTO employees VALUES ('BLINKEN', 2000, 20);

SQL> INSERT INTO employees VALUES ('NOD', 3000, 10);

SQL> GRANT SELECT ON employees TO PUBLIC;

SQL> CREATE OR REPLACE FUNCTION employees_policy_fn_1 (
 2          p_schema_name VARCHAR2,
 3          p_object_name VARCHAR2
 4  )
 5  RETURN VARCHAR2 AS
 6          l_return_val VARCHAR2(2000);
 7  BEGIN
 8          l_return_val := 'salary <= 2000';
 9          RETURN l_return_val;
10  END;
11  /

SQL> CREATE OR REPLACE FUNCTION employees_policy_fn_2 (
 2          p_schema_name VARCHAR2,
 3          p_object_name VARCHAR2
 4          )
 5  RETURN VARCHAR2 AS
 6          l_return_val VARCHAR2(2000);
 7  BEGIN
 8          l_return_val := 'commission_pct <= 10';
 9          RETURN l_return_val;
10  END;
11  /

SQL> EXEC DBMS_RLS.ADD_POLICY ( -
>          OBJECT_NAME => 'EMPLOYEES', -
>          POLICY_NAME => 'EMPLOYEES_POLICY_1',
-
>          POLICY_FUNCTION => 'EMPLOYEES_POLICY_FN_1', -
>          STATEMENT_TYPES => 'SELECT', -
>          SEC_RELEVANT_COLS => 'SALARY' -
> );

SQL> EXEC DBMS_RLS.ADD_POLICY ( -
>          OBJECT_NAME => 'EMPLOYEES', -
>          POLICY_NAME => 'EMPLOYEES_POLICY_2',
-
>          POLICY_FUNCTION => 'EMPLOYEES_POLICY_FN_2', -
>          STATEMENT_TYPES => 'SELECT', -
>          SEC_RELEVANT_COLS => 'COMMISSION_PCT'
-
> );
```

Concluding Remarks

For extra credit, explain why Roger needed to set OPTIMIZER_MODE to ALL_ROWS in his solution. Also review the quote about column-level VPD policies from the *Oracle 10g Security Guide* and guess the result of the query "SELECT COUNT(name), COUNT(salary), COUNT(commission_pct), COUNT(1) from employees" when using Sumit's setup. ▲

The author can be reached at **iggy_fernandez@ hotmail.com.**



(continued from page 7)

into the urgency level.

Since many of these pieces of information are time-series based, they have to be computed well in advance so the expert system can access this information during crucial junctures and make its decision whether "100MB will suffice to accommodate growth over the next 30 days and the best place to add this file would be on /u06 since there are no tablespaces with competing objects there."

Decision automation simulates rational human behavior when confronted with problems by being able to think, decide, and act.

DA Components

A decent DA-based system takes time to build (years, versus the hours or days for a script). While examining how to build one is outside the scope of this article, a DA-based system needs to have access to at least the following:

Historical metadata describing the target environments, especially in pre-summarized manner to facilitate fast decision making based on metrics.

Environmental heuristics such as "do not use NFS-mounted file systems for any database files" or "do not extend any data-file beyond 8GB," etc. A good DA system may ship with a basic set of crisp and fuzzy rules, which can be edited/customized by the user as required or adjusted by the system itself based on observation.

A library of automation routines, each allowing a certain specific action to be carried out. A single routine can work across platforms, or one can have one routine per platform, or if required, one routine per version of the platform.

Task units or cartridges that feed domain context directly into the DA system to analyze and perform tasks that it previously had no awareness of. This allows the brain of the expert system to be expanded to accommodate newer task methods and functionality, especially as newer versions of database products hit the market.

Concluding Remarks

Finally, the most powerful thing about a DA-based system is that, in spite of all the effort that goes into deploying one, unlike scripts, it is meant to handle a variety of areas requiring analysis and decision making and not just a single task. For instance, the same algorithms that help with database space allocation can also be used to figure out usable space on a target server during a database refresh or a clone. The resultant productivity gains with DA, especially in large and complex environments, are well worth the effort it takes to learn and deploy it. ▲

Venkat S. Devraj is the author of Oracle 24x7 Tips & Techniques *from McGraw-Hill. He is the co-founder of StrataVia Corporation (***www.StrataVia.com***) and the primary architect of the patent-pending automation suite Data Palette™. He blogs at* **vdevraj.blogspot.com** *and can be reached at* **dbafeedback@StrataVia.com**.

(continued from page 13)

```
 $conn = oci_connect(‘scott’, ‘tiger’, ‘orcl’);

 // if the connection failed...
 if (!$conn) {
   showError();
  exit; // leave the program
 }

 // define the sql statement that is to be run
 $sql = “SELECT ‘Hello, World!’ FROM dual
  WHERE DUMMY=:bndvar”;

 // parse the SQL statement
 $stmt = oci_parse ($conn, $sql);

 // set the bind variable and then bind it
 $bindVar = ‘X’;
 oci_bind_by_name($stmt, ‘:bndvar’, $bindVar);

 // execute the SQL statement
 if ( oci_execute ($stmt)) {
  // if successful . . .
  while ($row = oci_fetch_row ($stmt)) {
   print $row[0];
  }
 } else { // if Unsuccessful . . .
  showError($stmt);
 }
 // close the cursor
 oci_cancel ($stmt);
 // disconnect from the database
 oci_close ($conn);
?>
</body> </html>
```

We defined the showError() function because there is more than one place in the code where it might be needed. Within this function, the statement

```
$e = ($Stmt ? oci_error($Stmt) : oci_error());
```

is an example of the ternary operator. The expression *((expr1) ? (expr2) : (expr3))* will return expr2 if expr1 is TRUE and expr3 if expr1 is FALSE.

The oci_error() function returns the last Oracle error as an *associative* array with the elements *code* (the “ORA” error code), *message* (the standard error message), *sqltext* (the statement that caused the error), and *offset* (the position within sqltext where the error occurred). The function must be passed no arguments for connection errors, and with a statement object for execution errors.

Either single or double quotes can be used to designate strings. When included in a single-quoted string, variables will not be expanded. Using double quotes to specify a string causes non-escaped variable names to be expanded to their value, and also allows the use of special escape characters such as “\t” for a tab and “\n” for a linefeed. The period “.” is the string concatenation operator. The function htmlentities() converts special characters to their HTML equivalents. For example, “<” would be converted to “&lt;” and is used if you want to display these characters on a web page.

After a connection is established, SQL statements need to be parsed before execution[1]. Parsing does not validate the SQL, but creates a PHP statement object that is used for execution or parameter binding. Executed statements are committed by default, but of course, oci_execute() takes an optional parameter that will allow the statement to execute without commit, or to not fully execute, but gather metadata about the results of the select statement.

Once a statement has been executed, its result set is available and can be accessed by several functions. The oci_fetch_row() function used above returns the next row of data as a numeric array, while the oci_fetch_array() function allows you

to use column names as the array keys. Additionally, the oci_fetch_all() function places all of the rows of a result set into an array. The optional arguments of skip and maxrows can be used to easily provide paging functionality to applications.

Compiling, Linking, and Executing

Because PHP is an interpreted language, there is no compiling required. Lack of compilation however does not equate to a lack of speed. PHP is remarkably fast out of the box, but there are a number of compilation programs available that make it even faster, or allow it to run in a .Net environment.

Different elements of a program may be linked by the use of include files. These included files may contain class objects or simply supporting code of another type. The linking is performed upon execution of the program. Again, one would instinctively assume that this run-time linking would lead to performance issues, but PHP continues to impress. Executing a PHP web application is simply a matter of entering its URL in a browser, and voila—"Hello, World!"

> ***"Executing a PHP web application is simply a matter of entering its URL in a browser, and voila—'Hello, World!'"***

Further Reading

- The Official PHP Site: **www.php.net**
- Oracle and PHP: **www.oracle.com/technologies/php/index.html**
- Oracle PHP Developer Center: **www.oracle.com/technology/tech/php/index.html** ▲

David Schleis has worked for the Wisconsin State Laboratory of Hygiene for over 20 years. Now working exclusively with PHP and Oracle, he continues to build the "ultimate" Laboratory Information Management System (LIMS) for the Occupational Health Lab. If you want to talk to Dave about PHP or anything else, you can email him at **dave@mail.slh.wisc.edu**.

Endnote

[1] There are three functions provided to connect to the database, depending on whether the requirements call for a standard connection, a persisted connection to increase efficiency, or a new connection to ensure transaction autonomy. Something to consider when contemplating the use of persisted connections on a multi-process web server is that connections cannot be shared among different processes. I have found that using persisted connections leads to a buildup of idle sessions, and attempting to clean up idle sessions in Oracle by setting RESOURCE_LIMIT=TRUE causes connection errors in the PHP applications. I have since switched to standard connections, and there has been no noticeable increase in page load times.

```
// standard connection
$cnn = oci_connect ($User, $Pass, $Inst);
// persisted connection
$pcnn = oci_pconnect ($User, $Pass, $Inst);
// new connection
$ncnn = oci_new_connect ($User, $Pass, $Inst);
```

My problems were with IIS 5 and PHP 4; things may be different now, or perhaps I did not find the correct combination of settings. I have not revisited the issue, as scaling to very high volumes is not as yet a vital concern to my organization.

It’s Registration Time Again!

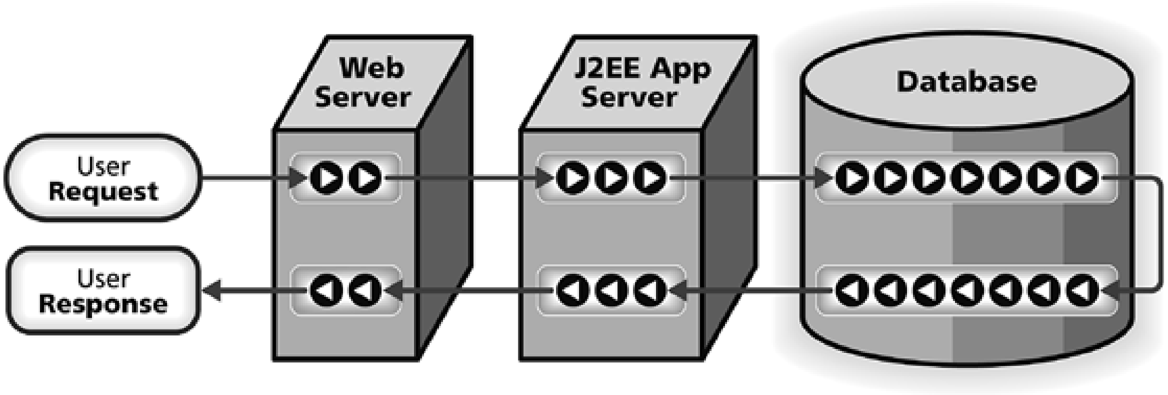
I know the year just zoomed by. I’m sure it’s due to the great 2006 NoCOUG program. The board has another stellar year planned for you. Please register early to continue getting this award-winning *Journal* and streamline your conference registration. For our 2006 members who have not yet registered for 2007, this issue of the *NoCOUG Journal* is complimentary. We know you plan to register.

Check the address label on the back of the *Journal*. If there is an asterisk (*) affixed to your name, you are already registered for 2007. Keep in mind that the mailing list was pulled on 1/10/07. The (*) status indicator will only be used on this issue.

All 2006 members should have received an email or U.S. mail reminder by now. That will include your NoCOUG ID for online registration at **www.nocoug.org**. Contact me with any questions at **membership@nocoug.org**.

Hope to see you at the coming conferences!

—Joel Rosingana
NoCOUG Membership Director



Bag ‘o Tricks

by Danny Chow

Danny Chow

Most DBAs will know how to find out what foreign keys are defined on a table, and if there are other tables in the database having foreign keys referencing the particular table. This information is available in the *all_constraints* and *all_cons_columns* data dictionary views in every Oracle database. In a complex schema without data model documentation or any development tools such as TOAD around, sorting out foreign key relationships can be a bit challenging.

It is not difficult to construct a SQL script to find this information, and indeed there are many such scripts available on the Internet. A handy script will definitely help to keep the DBA in good health, so we'll present one here. There is no software to install on your PC to use this script—just a quick login to the Oracle database with SQL*Plus; run the script shown below, and a "quick question" will be a responded to by a quick answer. This script works with all releases of Oracle since Oracle 8.

```
SET LINESIZE 132

SET FEEDBACK OFF

SET VERIFY OFF

COLUMN con_name FORMAT a30 HEADING “Foreign Key”

COLUMN con_col FORMAT a30 HEADING “Local Column”

COLUMN r_con FORMAT a65 WRAPPED HEADING “Refers To”

COLUMN ref_tab FORMAT a30 HEADING “Referenced By”

ACCEPT 1 PROMPT ‘Input table owner: ‘

ACCEPT 2 PROMPT ‘Input table name: ‘

SET HEADING OFF

SELECT global_name || ‘ at ‘ || TO_CHAR (SYSDATE,
‘DD-MON-YY HH24:MI:SS’)
FROM global_name;

SELECT ‘Foreign Key Constraints on ‘ || UPPER (‘&1’)
|| ‘.’ || UPPER (‘&2’) || ‘:’
FROM dual;

SET HEADING ON

BREAK ON CON_NAME

SELECT A.constraint_name con_name, B.column_name
con_col, C.table_name || ‘.’ || C.constraint_name ||’
(‘ || C.constraint_type || ‘)’ r_con
FROM all_constraints A, all_cons_columns B, all_
constraints C
WHERE A.owner = UPPER (‘&1’)
AND A.table_name = UPPER (‘&2’)
AND A.constraint_type = ‘R’
AND B.owner = A.owner
AND B.table_name = A.table_name
AND B.constraint_name = A.constraint_name
AND C.owner = A.r_owner (+)
AND C.constraint_name = A.r_constraint_name (+)
ORDER BY con_name, B.position;

SET HEADING OFF

SELECT ‘Table ‘ || UPPER (‘&1’) || ‘.’ || UPPER
(‘&2’) || ‘ is Referenced by:’
FROM dual;

SET HEADING ON

BREAK ON con_col SKIP 1

SELECT B.column_name con_col, C.table_name ref_tab,
C.constraint_name con_name
FROM all_constraints A, all_cons_columns B, all_
constraints C
WHERE A.owner = UPPER (‘&1’)
AND A.table_name = UPPER (‘&2’)
AND A.constraint_type IN (‘P’, ‘U’)
AND B.owner = A.owner
AND B.constraint_name = A.constraint_name
AND A.owner = C.r_owner (+)
AND A.constraint_name = C.r_constraint_name (+)
AND C.constraint_type = ‘R’
ORDER BY B.column_name, C.table_name;
```

Sample output from this script is as follows.

```
SQL> @table_fkey.sql
Input table owner: test
Input table name: invoices

TESTDB.DBSPECIALISTS.COM at 01-DEC-06 15:39:41

Foreign Key Constraints on TEST.INVOICES:

Foreign Key  Local Column   Refers To
------------ ------------   -------------------------
INVOICES_FK1 USER_ID        USERS.USERS_PK (P)
INVOICES_FK2 PAY_TERMS      PAY_TERMS.PAY_TERMS_PK (P)

Table TEST.INVOICES is Referenced by:

Local Column Referenced By           Foreign Key
------------ -------------           ----------------

INVOICE_ID INVOICE_LINES             INVOICE_LINES_FK1
```

I hope that you find this script worthy of addition to your toolbox. ▲

Danny Chow is a senior DBA with Database Specialists, a nationwide provider of consulting and Oracle-managed services for companies with mission-critical Oracle database systems. His email address is **dchow@dbspecialists.com***.*

Winter Conference Abstracts

	Auditorium	Room 102	Room 103
11:00 a.m. to Noon	**Oracle Performance Management: Real-World Case Studies—Gaja Krishna Vaidyanatha, DBPerfMan** We share the core principles utilized to determine, investigate, and implement solutions to real Oracle tuning issues. Every case study discussed is a real-world performance problem that was solved using a repeatable and reliable method. Bottom line—Oracle performance is a mathematical science that requires us to back everything with mathematical evidence.	**Thinking Out of the Box: Redefining Database Storage Management with ASM—Ara Shakian, Oracle Corporation** ASM redefines the traditional methods for database volume and file management, empowering the DBA and simplifying storage administrator tasks. This presentation discusses the new database storage management paradigm, best practices, and storage capacity and performance monitoring.	**Oracle 11*g* Secure Files—Amit Ganesh, Oracle Corporation** Abstract unavailable.
1:00 p.m. to 2:00 p.m.	**Oracle's ASM Reduces Cost of Deploying VLDB—Hanan Hit and Lina Shabelsky, SkyRider** ASM accesses the I/O subsystem through the kernel without the overhead of the host operating system. With ASM, Oracle 10*g* changes the rules of the game of cost vs. scalability and availability for building large database systems. In this presentation we describe how we built a terabyte-sized decision support system using low-cost NetApp storage hardware.	**RMAN in the Trenches: To Go Forward, We Must Back Up—Philip Rice, UC Santa Cruz** Backups are not glamorous, but essential. RMAN has many good features, but that also means there are subtleties and plenty to learn. This presentation is not a 30,000-foot overview—it is a sharing of personal experiences and lessons learned. We'll look at implications of infrastructure and feature choices and sample the good, the bad, and the ugly.	**The Tie That Binds: An Introduction to ADF Bindings—Peter Koletzke, Quovera** This presentation briefly reviews where ADF Model (ADFm) fits and how one of its components, ADF Bindings, allows you to quickly connect components from any user interface library such as ADF Faces and JSF Reference Implementation to business services such as ADF Business Components.
2:30 p.m. to 3:30 p.m.	**Comparing SQL Trace and ASH—Graham Wood, Oracle Corporation** Abstract unavailable.	**EDITOR'S PICK** **Fixing Broken SQL—Dan Tow, Singing SQL** Frequently, developers must fix SQL without knowing its business purpose. Surprisingly, there are patterns in most incorrect SQL that you can learn to recognize and repair even without knowing the purpose of the SQL. This presentation reviews a breakdown of the most common recognizable patterns in broken SQL, and teaches how to find and fix what's broken.	**Oracle Database 11*g*: Preserve Order amid Change with Database Replay and SQL Replay—Mughees Minhas and Leng Tan, Oracle Corporation** Oracle 11*g* introduces two new features, Database Replay and SQL Replay, that are designed to help manage and cope with changes by significantly reducing the time, cost, and risk associated with them. We explain how they work and provide a live demonstration.
4:00 p.m. to 5:00 p.m.	**Real-World Performance Panel and Q&A Session—Andrew Holdsworth, Juan Loaiza, and other panelists, Oracle Corporation** A panel of Oracle technology leaders including Juan Loaiza and Graham Wood. The panel will discuss real-world performance topics and will field questions from the audience.	No session.	No session.

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, Lisa Loper, at **lloper@dbspecialists.com**. ▲

Long-term event sponsorship:

LOCKHEED MARTIN

CHEVRON

ORACLE CORP.

PG&E

Thank you! Year 2007 Gold Vendors:

- BEZ Systems
- Confio Software
- Data Domain
- Database Specialists, Inc.
- Embarcadero Technologies
- GoldenGate Software, Inc.
- IT Convergence
- Quest Software
- Quovera, Inc.

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

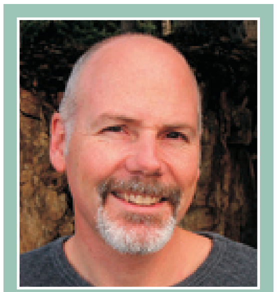
$ TREASURER'S REPORT

Diane Lee, *Treasurer*

Beginning Balance		
October 1, 2006		**$ 47,818.16**
Revenue		
Membership Dues	2,796.65	
Meeting Fees	620.00	
Vendor Receipts	4,400.00	
Miscellaneous	20.00	
Interest	72.12	
Total Revenue		**$ 7,908.77**
Expenses		
Regional Meeting	2,523.24	
Journal	7,746.89	
Membership	83.05	
Administration	910.00	
Website	–	
Board Meeting	744.17	
Marketing	1,200.91	
Insurance	500.00	
Vendors	–	
P.O. Box	74.00	
Total Expenses		**$ 13,782.26**
Ending Balance		
December 31, 2006		**$ 41,944.67**

Why Every Performance Tool Sucks–Including Mine!

by Craig Shallahamer


Craig Shallahamer

I'm surrounded by performance tools that suck—including mine! You may be wondering why someone who makes a living planning and optimizing Oracle-based systems thinks every performance tool sucks. I'll tell you why!

The problem is the industry. With all its millions of market research, it still has not captured the true need and the true heart and soul of a DBA. There is so much information oozing out of performance tools, someone who can read *The Matrix* language still could not understand this stuff. I know the information is there in some tools. But, I'm also equally certain the information *presentation* is not. And so, we enter the world where over 100 performance tool vendors compete for your IT budget.

With all the tools on the market today, you would think a problem's root cause could be firmly established within four mouse clicks. But most software wants you to look at tons of information and sift through all the sludge. It's like using an email filtering program—you still have to double-check all the filtered emails just in case—worthless. How about a *lot* of intelligence built into the product? I'm talking about automatic, self-aware, and root cause identification. But I don't think there is anything on the market yet. Vendors will argue this point, but I'm not convinced. It still takes a skilled professional to sift through the data and a highly skilled salesperson to make you believe it's automatic and self-aware.

And don't tell me that Response Time Analysis (RTA) is the answer. I published the first RTA paper, and I'll tell you it's just a piece of the pie. A significant piece, but all the same, just a piece. It's an answer, not *the* answer. Some professional RTA tools want you to do absurd things to your system (which I won't mention) so you can diagnose the problem. Nope—not going to do it. There has got to be something better.

Something like graphics that actually mean something. You've been there too. You're standing around a really slick graphic depiction of a system and after a few hours someone has the guts to say, "What's it mean?" Another hour or two later someone else says, "I don't know, but it sure is cool looking." Cool is not good enough for me. I'm looking for graphics that grab my soul and bring me into the system, while I still maintain and even flourish in my humanity.

I know our industry's companies can do better but they just stick with what "works." Honestly, take a close look at those stacks of binders from really expensive software that is supposed to make your job easier and more fun—they never get used.

The software is not used because the solution is not elegantly simple. I mean *really* simple and *really* elegant. The first user interface for my rightsizing product, HoriZone™ was very thorough, but it took people too long to figure out how to use the system. So hardly anyone did and I wondered why. Then I ran across a book that asked the question, "How would Leonardo da Vinci use a computer?" and "What would his computer be like?" After reading the book it dawned on me why I had this love/hate relationship with Windows, why people weren't flocking to HoriZone™, and why the best performance tools on the market today do, in fact, really suck.

It's because our tools don't do what we want them to do in an elegant fashion. Most tools try to conform us to them, not empower us. Ask yourself, when was the last time your computer really empowered you to create something beautiful—something that you're really proud of? For most people, it doesn't happen very often, if ever. And that's sad. We all have the innate capability to create beauty. But the tools we use just help us get by, not fly.

I know I'm whining. But as an experienced performance specialist, and especially as a consumer, that is my right. I will always be looking for the DBA tool that will expand my capabilities and allow me to leave work feeling good, not tired.

Someday I'll see that tool—I hope. I haven't yet. But when I do, I'll post it on my web-site, use it on all my performance engagements, and sell it with a passion. ▲

Craig Shallahamer helps companies improve Oracle IT service delivery. He brings his unique experiences to thousands as a speaker, teacher, researcher, writer, and author of the upcoming book Forecasting Oracle Performance. *He co-founded Oracle's Core Technologies and System Performance Group and left Oracle in 1998 to start OraPub, Inc. He offers his classes worldtries on five continents. He is also the architect of HoriZone, OraPub's service-level management solution.*

Teamwork at NoCOUG

(DIANE LEE)

Back row, left to right: Jen Hong—Secretary and Treasurer, Lisa Loper—President, Roger Schrag—Director of Conference Programming, Eric Hutchinson—Webmaster, Randy Samberg—Track Leader, Hamid Minoui—Training Day Coordinator, Naren Nagtode—Director of Marketing, Joel Rosingana—Director of Membership, Darrin Swan—Vice President; Front row, left to right: Iggy Fernandez—Journal Editor, Nora Rosingana—NoCOUG Staff Member. Not pictured: Hanan Hit—Track Leader, Diane Lee—Vendor Coordinator and IOUG Representative.

NoCOUG is a successful organization with more than 500 members, and there's no way it could run without teamwork. We have a full and active Board of Directors, plus other volunteers who contribute regularly. All the people on the NoCOUG team contribute in both big and small ways, depending on what they have time for. And it's all of us working together as a team that makes for the great conferences, training days, and other benefits.

But volunteering your time is far from without rewards. In fact, volunteering with NoCOUG offers opportunities to meet and talk with speakers, authors, and other professionals in the Oracle field, as well as other activities. In fact, if your day-to-day job has become routine or doesn't offer you the chance to use some of your other skills—interacting with people, writing, organizing events, etc.—volunteering is a great way to utilize those skills. It's surprisingly fun once you get started. You'll find we are a welcoming bunch of people, and most volunteers say their favorite aspect of volunteering is the people they meet. So, if you would like to get involved but don't know where to start, here are some quick things you can do that don't take much time:

- Contribute an article to the *NoCOUG Journal*
- Volunteer to speak and share your knowledge at a conference
- Recruit a knowledgeable Oracle colleague to speak at a conference or contribute an article
- Help with registration on the day of our quarterly conference
- Assist with marketing our conferences and training days

And, there are plenty of other opportunities to help out. Remember, it takes a lot of teamwork to keep our successful organization growing and providing value to its members. So, if you want to be part of a great team, just send an email to **board@nocoug.org** and let us know how you want to get involved.

What are you waiting for. Join the NoCOUG Team! ▲

"I have met many wonderful people during my years of volunteering. Some long-term friendships have developed. Every quarter brings new friends. This is what I enjoy about NoCOUG volunteering.

"A big reason for volunteering is the give-back factor. I have received a great deal of valuable experience as a NoCOUG member. I feel it's important to give something back to the organization."

—Joel Rosingana
NoCOUG Membership Director
and Past President
Independent Consultant

"The friendships and professional relationships I have enjoyed during the last six years serving as a Board member ranks #1 on my list of reasons why I volunteer for NoCOUG. Every member of our organization from each Board member, to you our Oracle Users membership, makes NoCOUG a great organization to be a part of. The teamwork that goes into planning each event has become a well-oiled machine as we work diligently to deliver valuable Oracle content to our members. I have watched many of you stay loyal to NoCOUG during the challenging times and watched as you found new opportunities, myself included. Thank you for making NoCOUG an exciting family to be a part of. When it comes down to it, the heartbeat of NoCOUG is you."

—Darrin Swan
NoCOUG Vice President
Quest Software

NoCOUG Winter Conference Schedule

February 8, 2007—Oracle, Redwood Shores, CA

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

8:00A.M.–9:00	**Registration and Continental Breakfast**—Refreshments served
9:00–9:30	**General Session and Welcome**—Lisa Loper, NoCOUG President
9:30–10:30	**Keynote:** *The Future of Database and Information Technology*—Juan Loaiza, Oracle
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Auditorium: *Oracle Performance Management: Real-World Case Studies*—Gaja Krishna Vaidyanatha, DBPerfman
	Room 102: *Thinking Out of the Box: Redefining Database Storage Management with ASM*—Ara Shakian, Oracle Corporation
	Room 103: *Oracle 11g Secure Files*—Amit Ganesh, Oracle Corporation
12:00–1:00 P.M.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Auditorium: *Oracle's ASM Reduces Cost of Deploying VLDB*—Hanan Hit and Lina Shabelsky, SkyRider
	Room 102: *RMAN in the Trenches: To Go Forward, We Must Back Up*—Philip Rice, UC Santa Cruz
	Room 103: *The Tie That Binds: An Introduction to ADF Bindings*—Peter Koletzke, Quovera
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Auditorium: *Comparing SQL Trace and ASH*—Graham Wood, Oracle Corporation
	Room 102: *Fixing Broken SQL*—Dan Tow, Singing SQL
	Room 103: *Oracle Database 11g: Preserve Order amid Change with Database Replay and SQL Reply*—Mughees Minhas and Leng Tan, Oracle Corporation
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Auditorium: *Real-World Performance Panel and Q&A Session*—Andrew Holdsworth, Juan Loaiza, and other panelists, Oracle Corporation
5:00–??	**NoCOUG Networking and Happy Hour at TBD**

Session descriptions appear on page 24.

Cover photos © Tom Wagner. Mount Timpanogos, Utah.
The *NoCOUG Journal* design and production: giraffex/sf.

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

NoCOUG
P.O. Box 3282
Danville, CA 94526

RETURN SERVICE REQUESTED

Is this your last issue?
See page 22.