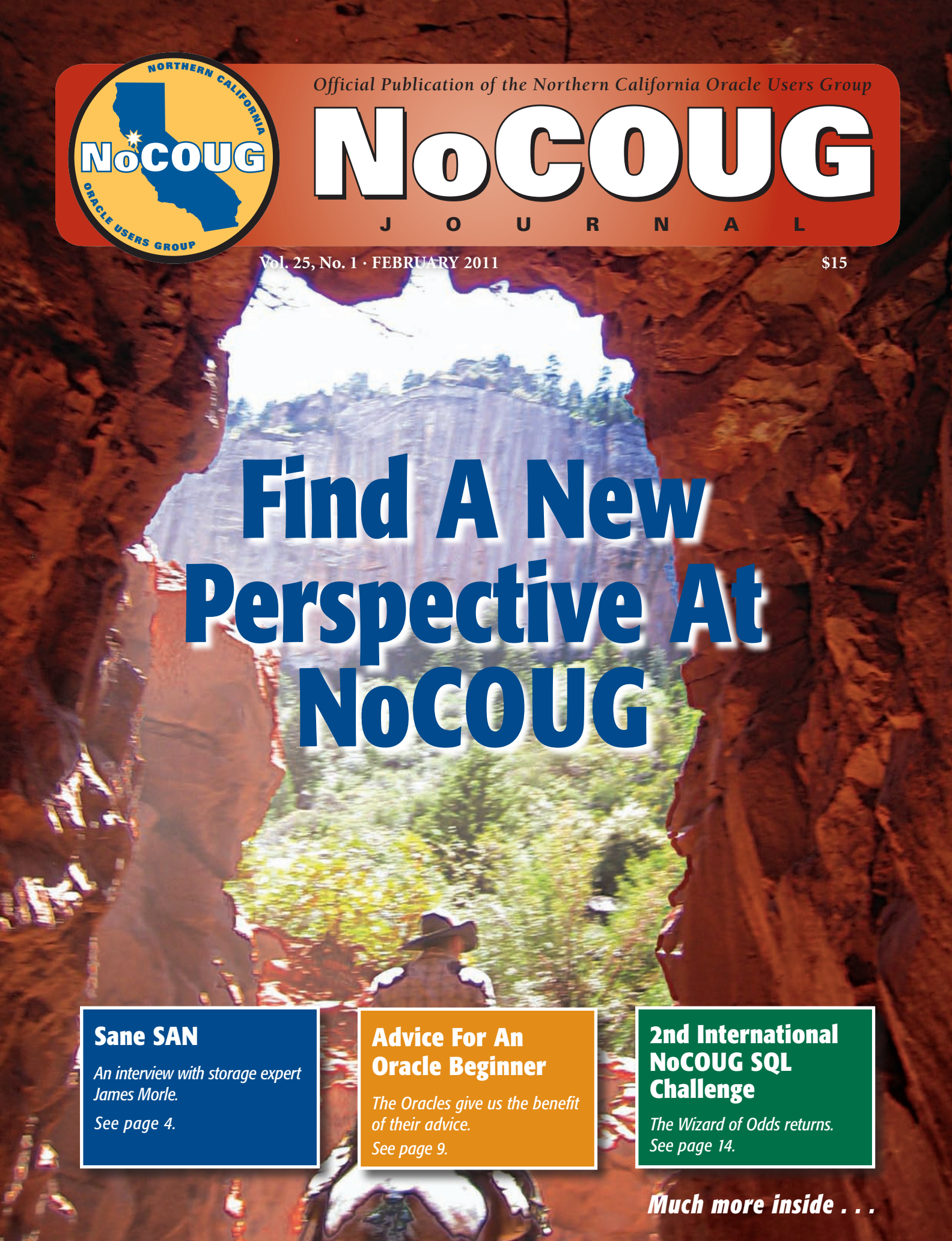
Official Publication of the Northern California Oracle Users Group

# NoCOUG JOURNAL

Vol. 25, No. 1 · FEBRUARY 2011 $15



# Find A New Perspective At NoCOUG

## Sane SAN

*An interview with storage expert James Morle.*

*See page 4.*

## Advice For An Oracle Beginner

*The Oracles give us the benefit of their advice.*

*See page 9.*

## 2nd International NoCOUG SQL Challenge

*The Wizard of Odds returns.*

*See page 14.*

***Much more inside . . .***

# Celebrating 25 Years

This year, the *NoCOUG Journal* is celebrating its 25th year of service to the Oracle community in Northern California. The *Journal* has been witness to the growth of Oracle from a fledgling startup into one of the mightiest software and hardware companies in the world. In its pages has been printed the wisdom of the brightest stars in the Oracle firmament, too many to list here. It would not have been possible without the strong support of the board of directors of NoCOUG, the selfless contributions of hundreds of authors, and the patience and outstanding work of NoCOUG staff and vendors, including Nora Rosingana, our trusty accountant; Karen Mead, our eagle-eyed copyeditor; and Ken Lockerbie, our layout artist extraordinaire.

Let's all wish the *NoCOUG Journal* a wonderful 25th year and 25 more wonderful years to come. ▲

—*NoCOUG Journal* Editor

# Table of Contents


President's Message ..... 3
Interview ..... 4
Seminar Announcement ..... 5
Ask the Oracles ..... 9
Book Review ..... 14
SQL Challenge ..... 18
SQL Corner ..... 20
Sponsorship Appreciation ..... 24
Session Descriptions ..... 25
Conference Schedule ..... 28


**ADVERTISERS**


Avior ..... 3
Delphix ..... 7
Rolta TUSC ..... 17
3PAR ..... 22
Confio Software ..... 23
Database Specialists, Inc. ..... 27


## Publication Notices and Submission Format

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

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

The submission deadline for the upcoming May 2011 issue is February 28, 2011. Article submissions should be made in Microsoft Word format via email.



## 2011 NoCOUG Board



## 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**

# Keeping Obsolescence Away For 25 Years

by Iggy Fernandez

*Iggy Fernandez*

This year, NoCOUG is celebrating its 25th year of service to the Oracle community in Northern California! It would not have been possible without the contributions of hundreds of speakers and volunteers, the outstanding work of NoCOUG staff and vendors, and—most of all—the loyal support of members like you.

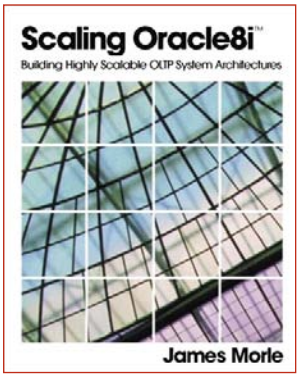
But there are still many thousands of Oracle database professionals in Northern California who have never heard of NoCOUG and have never attended a NoCOUG conference. If you believe in the career-building value provided by NoCOUG, won't you tell at least one colleague about NoCOUG today?

In this issue of the *NoCOUG Journal*, we asked the Oracle gurus to provide career advice to Oracle professionals. Oracle ACE Director Arup Nanda had this advice for us:

*"If what you do is deemed to be nonspecialized, that job may easily be passed on to another person in the middle of the bell curve (think outsourcing). If you don't differentiate yourself from all those in the middle of the bell curve, you are dissolving into obsolescence."*

What better place than a NoCOUG conference to find opportunities to differentiate yourself and make yourself more valuable to your employer? Our upcoming conference on Thursday, February 24, is a veritable smorgasbord of educational presentations on cutting-edge topics such as Virtualization, Cloud Computing, Business Intelligence (BI), Application Express (APEX), MySQL, and even NoSQL!

Who among us really wants to dissolve into technological obsolescence? I look forward to meeting you and your colleagues on February 24. ▲




# Sane SAN

## with James Morle

*James Morle*

*James Morle is the founder of Scale Abilities Ltd. and an Oracle and systems architecture consultant with 18 years' experience in high-end Oracle implementations and over 25 years in professional computing. He is the author of the critically acclaimed book* Scaling Oracle8*i, which is now available for free download from* **http://www.scaleabilities.co.uk**, *and one of the founders of the OakTable Network.*

***You've been giving a presentation titled "Sane SAN" for over 10 years now. Reading your blog, it looks like you mostly talk about the insanity that SAN can introduce into your database architecture. Is disk the main cause of poor performance? Did it get better or worse in the 10 years that you've been talking about this issue?***

I wrote Sane SAN because I frequently encounter poor-performing storage infrastructure when helping out my customers. I have tremendous sympathy for the embattled DBA trying to wring more performance from an underperforming SAN for many, many reasons. Some of those reasons are highlighted in my writing, but often the biggest hurdle of all is plain and simple politics. I'm not just talking about internal politics—I was at a customer site this week to kick off a storage performance workshop where there were 14 people in the room, including the sales guy from the vendor. That's a bad start. Disk is certainly not the main cause of poor performance: ignorance is. The problem with storage is that it is frequently presented as a black box, thus aggravating the ignorance problem. And on this aspect alone we are as badly off as we have ever been.

***Many businesses debate between using SAN and NAS. Which do you recommend? What are the differences in performance between the two systems? Is RAC on NAS safe and performant?***

I like to phrase this as "blocks vs. files" rather than "SAN vs. NAS." That is normally the inference in the question, rather than which is the best networking topology with which to attach the storage. On that subject, I do believe that 8 Gbps Fibre Channel will be the last revision we see, and even 8 Gpbs may not make the mainstream. 10 Gbps Ethernet and lower-latency software interfaces have killed Fibre Channel. The blocks (LUN-based presentation) vs. files (NFS-presentation) question is very much dependent upon the situation, though I am a great believer in the power of simplicity, and file-based presentation wins hands down in this respect. And that very much includes RAC, where NFS is both performant and safe.

***Is TPC relevant today? Why has no database vendor except MS SQL Server submitted a TPC-E benchmark? Why has no database vendor submitted a TPC-DS benchmark?***

I have no idea. I don't think the database vendors take them very seriously anymore, especially since the old database wars have mostly become irrelevant. I think that the old TPC-C benchmark is a useful measure of a given processor's ability to grind through database-based codepath. It's probably the most abused benchmark in history in terms of scaling and tuning, but that doesn't really take away the small degree of utility that remains in terms of "how good is this processor with large Oracle workloads?"

***Many companies turn to virtual machines to improve datacenter manageability. Can virtual machines deliver the performance we need?***

Yes and no. There are certain things that hypervisors can do really well and others they struggle with. I don't think that overcommitting memory is a viable option with databases—it never has been way back to the first Unix-based databases, and I recently tested this again using a leading hypervisor and many Oracle databases. It was still very unpredictable and, at best, lumpy in performance. That doesn't matter for databases that, well, don't matter, but it's not good for mission-critical workloads. The worst workloads of all for virtualization are those that are latency sensitive, such as those that do too many round-trips between the app-server and the database. A tiny uplift in network latency can result in a doubling of run-

***"Disk is certainly not the main cause of poor performance: ignorance is. The problem with storage is that it is frequently presented as a black box, thus aggravating the ignorance problem."***

time, for example, on very chatty batch jobs. Conversely, a workload that relatively infrequently performs I/O of any kind (network or disk), is not over-committed for memory, and spends the majority of its time in user-mode CPU—that is a workload that would perform very well when virtualized.

***Would you store BLOBS in the database? What if I have millions of BLOBS? What if they are very large? Would you recommend Database File System (DBFS) and SecureFiles?***

I'm not sure that I'm an expert in this area, but I certainly have opinions!

I think that BLOBS are, in general, a good thing. Why not? Not everything is relational in form. SecureFiles are a great enhancement to the traditional BLOB. However, they are not a

*"A workload that relatively infrequently performs I/O of any kind (network or disk), is not over-committed for memory, and spends the majority of its time in user-mode CPU—that is a workload that would perform very well when virtualized."*

high-performance feature by simple virtue of their arbitrary nature compared to predictable relational data. They carry much the same cross as LONGs in that respect, but they have a better programmatic and storage implementation. I remain unconvinced that a database is the right foundation for a filesystem—there is too much codepath and API complexity for storing files compared to a filesystem. That doesn't mean there

> ***"I am a great believer in the power of simplicity, and file-based presentation wins hands down in this respect. And that very much includes RAC, where NFS is both performant and safe."***

are not reasons to store files in the database—transactional integrity should not be broken just because there is a file involved, for example—but it's important to properly assess when it is time to store files in a database and when it is more appropriate to simply store them in a filesystem.

***I cannot forecast the future needs of my application. Developers frequently release new features with huge I/O impact without notifying anyone in advance. Web applications are sensitive to spammers who can make huge number of updates in a very short time. How do I ensure the scalability of the disk subsystem to meet future needs?***

I think it's important not to assume arbitrary growth when designing a storage system. It's rarely a requirement but rather an unlimited waste of money in that respect. If you truly have arbitrary (or at least massive) growth, it requires proper design. It requires design that precisely fits the needs of the application. It is the nature of all things storage that some portions of data will be hot and others will be not so hot—and the design must accommodate this. Databases applications are not Google Search—Google happens to have a data set and request profile that doesn't result in such pronounced hotspots as, for example, a trading application. Internet applications tend to have a combination of the two: areas of large growth but low read/write activity (think book reviews, spam comments, etc.) and areas of limited growth but massive read/write activity (think general ledger or shopping baskets). Those two areas should be designed for the profile they possess: horizontal scaling of low-cost disk for the former and focused, high-performance storage for the latter.

***Do disks degrade when they become full? Is there a rule of thumb for this?***

No they do not, but they tend to be used more when they are full of data and thus struggle to maintain the same level of performance as when there were fewer requests. I don't think there is really a rule of thumb for this except to design properly (as per the previous question) and to ensure that caches are configured properly at the various tiers so that frequently accessed data does not equate to frequently accessed disk. The exception to this question is Flash storage. Flash storage degrades significantly in performance when writes take place to used blocks (i.e., overwriting existing data). There are various workarounds to this in software, but the physical phenomenon remains.

***In your consulting business you probably solve some very interesting problems. We'd love to hear about a recent war story.***

I have many stories, of course, but I don't want to upset my customers by airing their dirty laundry in public! However, I will tell a short story about inherent stubbornness in storage decisions . . .

One of my customers was running a fairly large, mid-range SAN (block-based storage, presenting LUNs over Fibre Channel to the host) with Linux clusters running the Oracle instances. Having many years of experience with this customer I had a very good understanding that ease of use and labor-saving were probably of equal importance to them as getting the required performance. I accordingly recommended that they should look at a NAS solution, with full knowledge that their storage performance needs would be more than met by this solution and that they would gain significant usability advantages with a NAS-based solution. They simply did not believe me—NFS cannot deliver the performance of Fibre Channel, they told me. I analyzed all the performance data once again and presented a counter-argument. I had nothing to gain from them going NAS—perhaps the reverse, in fact, and I was not even on billable time for the recommendation I was making. Nope, the heels were dug in.

Eventually, I managed to convince them to run a trial, to run their critical batch processes on a loaner NAS and see what happened. The result was a 4x improvement in runtime with no other changes, but they were still unconvinced. Why? They were still worried that NFS was lower performance than Fibre Channel! The fairy-tale ending is that the customer did eventually go with the NAS solution, and they have been delighted with the performance and usability gains they have achieved. This is not an isolated case—storage is a terribly entrenched part of the stack, rich in personal agendas, brand preference, and voodoo. I plead with everyone reading this to stick to the science and believe in their own results!

***Storage administrators tend to differentiate between IOPS and throughput metrics for storage, but it looks like more IOPS would mean more throughput. Why do we need two different metrics?***

Good question. Let's use mathematics to explain this empirically, and then mock the whole thing with a provocative real-life case:

If I am doing 1024 IOPs, how much throughput do I have as a result? It depends, of course, on how large the request size is. If all those requests are 8 KB blocks, then we are getting 8 MB/s throughput. If those requests are 1 MB full-table scan requests, we are getting 1 GB/s throughput. 8 MB/s vs. 1 GB/s—that's why there are two metrics.

*(continued on page 8)*

*(continued from page 6)*

The real-life case is a bit different, though, and it's why storage performance is rarely understood by those that only look at such high-level, averaged scenarios. Consider this: if the DBA complains about slow storage, and the storage administrator looks at the array and sees 500 IOPs and 30 MB/s, what does he need to fix? He says that the database server is not driving the storage array hard enough . . . it is idle. He says that the average service time is very good—2 ms—and the DBA validates this from the Oracle viewpoint. Impasse—the query is still slow, but everything looks fine from both perspectives.

> ***"Transactional integrity should not be broken just because there is a file involved, for example—but it's important to properly assess when it is time to store files in a database and when it is more appropriate to simply store them in a filesystem."***

What both are missing is the type of the workload: it is a serial query. The DBA could run two of these queries concurrently, and the metrics on the array will show 1000 IOPs and 60 MB/s coming out the front end, but both queries will be just as slow as they ever were! The answer to this problem is to trace the query and plot all of the individual I/Os on a chart. It will then be evident that the 2 ms response time is, for example, made up of many very fast I/Os and some extremely slow I/Os, indicating a problem in the I/O layers. Serial I/O from a cached storage array should be very much faster than 2 ms, even though this would be a good overall average for mixed I/O.

***Are modern implementations of RAID-5 really all that bad? Is it just a philosophical argument or has anybody ever conclusively proved the superiority of RAID-10 with scientific testing? For what situations can we safely use RAID-5?***

RAID-5 is slow, specifically for "small" writes, for reasons of physics. It has nothing to do with "modern" versus "old," because those phrases imply that the problem was a processing-speed issue. It may well have been, long ago, but that went away a long time ago and RAID-5 still sucks for small writes.

What is a small write? Any write that is less than one strip in size. Vendors such as NetApp (technically they do not use RAID-5, but it is still a parity-based protection scheme) and software like ZFS mitigate the infamous parity-RAID write penalty by abstracting the location of writes using a copy-on-write file layout. This means that writes do not need to take place to a specific (physical) location and can therefore be grouped into units that, all being well, are a whole stripe in size.

Where can you safely use RAID-5? Read-mostly systems such as data warehouses and systems based on copy-on-write file layouts.

***Is* Scaling Oracle8*i*: Building Highly Scalable OLTP System Architecture *outdated or has it stood the test of time? It's where I learned that high load average does not always mean high CPU utilization. Which parts do you feel could use a rewrite?***

My advice to prospective authors would be to resist the pressure from the publisher to have an Oracle version in the title of their book! There are a bunch of parts that have become out of date, but much of it is very relevant even today. There are some obvious candidates for culling such as the OPS chapter, but even that has some valuable content that could go into a new RAC chapter.

***Thanks for spending so much time with us today. I hear that Arup Nanda is going to attend your NoCOUG training day, so it must be really good and I hope to attend. What topics will you cover? Are newbies welcome?***

It's great that Arup is attending; I am looking forward to seeing him. I have designed the course in such a way as to teach understanding rather than facts, and so there should be value for the newbie and experienced alike. I will be covering both the physical and logical side of storage and I/O, and even the importance of understanding a few computer science principles sufficiently to develop a well-trained "gut" for what is going on. We will be going over interconnects, physics, how storage arrays are built, how servers view I/O requests, and how Oracle does I/O, and we'll even dip into the implications of new technology to enter the mainstream such as SSD and new software APIs. I like to encourage an interactive session, time permitting, so it's a great chance to fill in those gaps of knowledge and prepare yourself for going back to the office to ask some intelligent questions! ▲

> ***"Where can you safely use RAID-5? Read-mostly systems such as data warehouses and systems based on copy-on-write file layouts."***

Interview conducted by Chen (Gwen) Shapira

## It's Registration Time Again!

For our 2010 members who have not yet registered for 2011, this issue of the *NoCOUG Journal* is complimentary. We know you plan to renew your NoCOUG membership. Please renew early to continue getting the *Journal* and streamline your conference registration.

—Noelle Stimely, NoCOUG Membership Director

# Advice for an Oracle Beginner

## *Ask the Oracles!*

**Tom Kyte:** When I first started out in IT, I had no experience, no training, no background whatsoever. I was a math major fresh out of college in the year 1987. I hadn't taken any computer courses beyond the initial "introduction to" type of classes. It wasn't until I got a job as a computer programmer—advertised as "no experience required"—that I started even really using computers.

So, given that I had no experience, no real formal training—how did I get started, how did I get to where I am today? I think it comes down to two simple words: mentorship and participation.

When I first started out as an entry-level programmer, I had an excellent mentor. This was probably the key difference between success and failure for me. My mentor—who back then was about the age I am now (that is, he was old)—took the time to teach me the ropes. He taught me the right way to do things—not the fast way, not the "shortcuts," not the checklist of things to do—but the right way. In many cases, the right way isn't the easy way, isn't the quickest way . . . but it is ultimately the best way. He taught me many things I myself teach these days. Simple things such as "make your subroutine fit on a screen, you have to see it all," "instrument your code to death," "write as little code as you can but as much as you have to," "code defensively; don't trust anyone else to just know what to do with your code," and "test, test, test, benchmark and test again." My mentor made me the programmer I am.

The second item—participation—is what propelled me beyond being just a programmer. In the early 1990s, I started participating in online forums hosted on Usenet. For those who never heard of it, "Usenet" was Twitter, blogging, Facebook—any social network goes here—before any of them were invented. Usenet consisted of a relatively small (by today's standards) group of individuals that would discuss topics of interest to them. My interest was, of course, all things Oracle, and discuss we did. I "met" in a virtual sense on those discussion forums many people I still correspond with and interact with face to face. I learned a lot from them—and they (hopefully!) learned a thing or two from me. It was on these forums that I found the answers to many of my questions—and formulated answers to questions from others. This give-and-take allowed my knowledge of Oracle, programming, and databases in general to expand and grow immeasurably. Participation in the Oracle community is what took me from being just another programmer to being "AskTom." It gave me the confidence to write my first book in the year 2000: Expert One on One Oracle. It also gave me the audience for such a book. Without the act of participating, I do not think I would be where I am today.

***"Given that I had no experience, no real formal training—how did I get started, how did I get to where I am today? I think it comes down to two simple words: mentorship and participation."***

So, in short, find a mentor. This is crucial. Find someone that you trust, that you respect, that others trust and respect. Learn from your mentor. Then, start participating. Participate in your local user group. Get up in front of an audience and present on some technical topic. Attend conferences. Get active in a discussion forum, such as those on **http://otn.oracle.com**. Don't be afraid to make mistakes (you will; I did), but make sure to learn from them. That would be my advice. ▲

---

*Tom Kyte is a Senior Technical Architect in Oracle's Server Technology Division. Before starting at Oracle, Tom worked as a systems integrator building large-scale heterogeneous databases and applications, mostly for military and government customers. Tom spends a great deal of time working with the Oracle database and, more specifically, working with people who are working with the Oracle database. In addition, he is the Tom behind the "AskTom" column in* Oracle Magazine*, answering people's questions about the Oracle database and its tools (***http://asktom.oracle.com***). Tom is also the author of* Expert Oracle Database Architecture *(Apress, 2005),* Expert One on One Oracle *(Wrox Press, 2001/ Apress 2004),* Beginning Oracle Programming *(Wrox Press, 2002/ Apress 2004), and* Effective Oracle by Design *(Oracle Press, 2003). These are books about the general use of the database and how to develop successful Oracle applications.*

# Advice for an Oracle Beginner

## *Ask Jonathan Lewis!*

**Jonathan Lewis:** The first piece of advice I'd give to a beginner would be in the form of a question: "Where do you want to be in five or ten years' time?"

"Oracle" is a big field, and there are plenty of different ways to "do Oracle." Do you see yourself at the developer end of the spectrum, either as a writer of application code that you hope to fire at an Oracle database or as an application designer trying to ensure that the applications you design make effective use of the available database functionality? Or do you see yourself at the DBA end of the spectrum, and if so, is this as a soloist in a small company responsible for every aspect of keeping the systems alive or as a member of a DBA team in a large company dealing with the issues of very large, busy databases teetering at the leading edge of Oracle's capability? Do you want to go one step further and find a niche where you can become an internationally recognized specialist (like Mark Rittman for BI, Pete Finnigan for security, or Cary Millsap for performance)?

Of course once I've asked you where you want to be, my advice on how to get there is the same irrespective of the destination. You need some appreciation of the core mechanisms of the database software, and these can probably be summed up as redo, undo, space management, memory management, and indexing. Your choice of future will dictate the depths to which you have to go in understanding these features and where you go to next, but everyone—including the basic application developer—needs some understanding of what work really goes on when you throw data at the database or try to pull data from the database.

To get started, read the *Concepts Guide* and (fairly lightly if you plan to be a developer) the basic *Database Administration Guide*. Get your hands on an Oracle database, run up an SQL*Plus session and start doing some experiments: write a couple of queries against the data dictionary, create a couple of tables and indexes, insert and delete data, see how quickly you can write a script to create a million rows of data, check the impact of creating several indexes on a table before inserting lots of data. Take notes on the problems you had and the mistakes you made and the errors that Oracle reported.

Next, get acquainted with the tools you need for simple analysis: the dynamic performance views—initially just v$sesstat (work done) and v$session_event (time spent not working)—and the tools for tracing your work, dbms_monitor and tkprof.

Then start looking at the public discussion sites (the OTN database forum, the **comp.databases.oracle.server** newsgroup, and the Oracle-L list server are good starting points) to see the questions that people ask about Oracle, and use those questions to help you practice how to learn. Think about how you could create tests to investigate other people's problems, and try to work out what Oracle is doing that they need to know; then make sure you write up what you've done, why you did it, what you observed, and what you concluded.

From this small, database-centric start you can learn the habits that can make you as much of an Oracle expert as you want to be. None of it is rocket science. Once you've learned how to investigate and know something about the core mechanisms in Oracle, you can take your career wherever you want it to go. ▲

***"Look at the public discussion sites to see the questions that people ask about Oracle, and use those questions to help you practice how to learn. Think about how you could create tests to investigate other people's problems, and try to work out what Oracle is doing that they need to know; then make sure you write up what you've done, why you did it, what you observed, and what you concluded."***

---

*Jonathan Lewis is well known to the Oracle community as a consultant, author, and speaker, with more than 22 years' experience in designing, optimizing, and troubleshooting on Oracle database systems. He has been working on the follow-up to his first volume on Cost-Based Oracle for the last five years—but in the meantime he has contributed a chapter on the problems of statistics collection to the latest Oak Table book,* Expert Oracle Practices: Oracle Database Administration *from the Oak Table.*

# Advice for an Oracle Beginner

## *Ask Wolfgang Breitling!*

**Wolfgang Breitling:** What advice to give to an Oracle beginner? Being a DBA myself I am assuming advice to a beginner Oracle DBA—which leads right to the first advice: never assume, never guess! Confirm and validate. Having said that, I shall immediately ignore my own advice and continue —else I don't have any advice to give.

As a DBA, you are the custodian and guardian of the data in your database(s). Everything else follows from this. A modern translation of the Hippocratic Oath taken by medical doctors reads, "*I will follow that system of regimen which, according to my ability and judgment, I consider for the benefit of my patients, and abstain from whatever is deleterious and mischievous.*" Replace "patients" with "databases," and you have what could serve as a code of conduct for DBAs. Everything else follows from there. Protect your databases from physical harm (backup and recovery) and accidental or malicious misuse (security and authorization), and ensure timely access to rightful uses (high availability and performance).

Start with the concepts, administrator, security, and backup and recovery guides. If you do not have a backup of a database or cannot recover it, you will not be a DBA for long. Augment the Oracle manuals with information from other sources: books, newsgroups forums, local user groups. Join online newsgroups (e.g., Oracle-L) and forums (e.g., forums.oracle.com). Even if you don't have anything to contribute (yet), just "listening" to the problems being posted and the advice given in response will advance your overall knowledge. Be careful, however: not every piece of advice is correct or good. Remember my first advice: confirm and validate. Build yourself a "sandbox" where you can try things out. You don't need a dedicated server for that. Get one of the virtual machine packages, download the Oracle database software from OTN (technet.oracle.com) and install it. There are plenty of step-by-step installation instructions for virtually any Oracle database version on the Internet. Just Google for it. Virtual machines have an advantage over real ones: you can take a snapshot of the machine state before embarking on something potentially destructive, and restore that snapshot if you get stuck and don't know how to recover. This is a great way to practice various recovery scenarios. A snapshot is also a great help to do a series of tests with the same starting point.

Never stop learning. That seems daft advice for a beginner. At the beginning all you appear to be doing is learning. But even once you become a senior DBA, do not stagnate and stop learning. It almost becomes harder then. In addition to learning what is new, you first have to unlearn what is old and no longer true. That is the hard part: recognizing that something you could have sworn by is no longer so.

Beyond becoming proficient in all things Oracle database, also learn about the peripherals: computer architecture in general and specifically the hardware and operating system your databases are hosted on; the storage and network architectures being used; and finally, the application being supported by the database. What language and framework is it written in? Get to know the developers, share your knowledge of the Oracle database engine with them and learn from them about the programming language and the challenges they face. You are not adversaries, even though at times it may seem so—to both sides! Developers may not be aware of the latest features of Oracle that could make their code easier, better, or faster. They are busy keeping up with the changes to their programming environment. Sometimes the programming framework or management directives can make it difficult, if not impossible, to use certain Oracle features. ▲

***"Once you become a senior DBA, do not stagnate and stop learning. In addition to learning what is new, you first have to unlearn what is old and no longer true. That is the hard part: recognizing that something you could have sworn by is no longer so."***

---

*Wolfgang Breitling is an independent consultant specializing in administering and tuning Peoplesoft on Oracle. The particular challenges in tuning Peoplesoft, often with no access to the SQL, motivated him to explore Oracle's cost-based optimizer in an effort to gain a better understanding of how it works and use that knowledge in tuning. He has shared the findings from this research in papers and presentations at IOUG, UKOUG, local Oracle user groups, and other conferences and newsgroups dedicated to Oracle performance topics.*

# Advice for an Oracle Beginner

## *Ask Arup Nanda!*

**Arup Nanda:** You just dipped your toe in the waters of DBAland and now you are regarding all those possibilities that lie ahead with awe and trepidation. If you had a choice, what would you eventually be—a general DBA, a specialist troubleshooter, an Oracle scientist (think Jonathan Lewis), a data architect, a DBA team lead, a manager of people, or even a CIO one day?

Some of these careers are obviously wide apart from each other. A successful CIO excels in people and contracts management, not a specific technology—not even Oracle—or, in other words, is a generalist. At the other end of the spectrum, a troubleshooter masters only a specific area of the database, e.g., Data Guard or performance tuning. A generalist DBA role lies somewhere in between, with expertise in a limited area (such as the Oracle database) but wide enough (such as both taking backups and setting up streams). On a slightly different career, data architects rely less on technology skills and master the organization's business processes, a skill that makes them somewhat less replaceable. Their knowledge may be transferable to the other organizations in the same industry, leading to a huge growth opportunity. A generalist DBA, who relies on technology, is seen as less attached to the organization but also as someone can make a switch to any other organization running Oracle database—another way to look at job security.

What about potential job opportunities? Think of a bell curve—the generalist DBA career comes in the middle, with the most number of openings. The numbers dwindle and competition rages on either side.

So, where do you see yourself? You don't have to answer that or make a decision right now, but it will be a serious mistake to brush this question off as idle contemplation. With everything you do or each career move you make, you always need to validate your choices against this question and learn from the events to make course corrections for your ultimate objective.

Look at the bell curve again. A smart choice may appear to be in the area of highest concentration of opportunities—right in the middle of the bell curve—with plenty of jobs, less friction, and a relatively cozy life. It may have been the smart choice at one time, but the winds are changing. The word plenty is usually associated with commodity. If what you do is deemed to be nonspecialized, that job may easily be passed on to another person in the middle of the bell curve (think outsourcing). If you don't differentiate yourself from all those in the middle of the bell curve, you are dissolving into obsolescence—your own obsolescence. Although DBAs do it, resetting passwords and adding datafiles to tablespaces are not really skilled jobs. You need to be a cut above the rest.

Second, consider carefully if you would rather become a manager of people at some point or if you would rather remain technically strong throughout your career. The skills required to become a successful manager are very different from those for a technical person. It's easier to make that transition early in your career rather than later. The transition from elite technological Special Forces operations to management is tough. Some have done that, but many have failed. On the flip side, if you do want to remain strongly technical, remember that time is counted in technology years, which are even shorter than dog years.

Summing up, my advice to a green DBA is as follows: Don't decide on what you want to be right now, but set a goal from the beginning that you want to be the top dog in your chosen field as soon as possible, and commit yourself to the effort that this goal demands. If you want to be a commodity DBA, look for a second career. In the U.S., that role is going to be as extinct as the pterodactyl. If you wish to be a data architect, work toward it immediately and establish a foothold in the organization's business process. If you would rather be a manager, cultivate and demonstrate those traits of your personality as soon as possible. Remember, your skills as a technocrat will have little bearing on your attractiveness as a manager. To excel, train yourself hard, mentor someone, present a topic at a user group meeting, write an article or blog, and participate in challenging issues. Remember, it's no fun to be in a stressful situation, but it's like immunization: you are the biggest beneficiary. Above all, always try to be the best, not the rest. ▲

---

*Arup Nanda has been an Oracle DBA for more than 14 years, working on all aspects of the Oracle database from modeling, performance tuning, backup, and disaster recovery to security. He has co-authored four books (*RMAN Recipes for Oracle Database 11*g, 2007, Apress; Oracle Database 11*g *New Features Guide, 2007, Oracle Press;* Oracle PL/SQL for DBAs, *2005, O'Reilly; and* Oracle Privacy Security Auditing, *2003, Rampant Techpress), written more than 300 articles, and presented some 100 times at various Oracle technology conferences. He is an OTN ACE Director and a member of the OakTable Network.*

# Advice for an Oracle Beginner

## *Ask Brad Brown!*

**Brad Brown:** I think the real secret to learning a new topic depends on your own learning style. I know how I learn best, but I have no idea how you will best learn a new topic . . . only you can know this. Some people are visual learners (i.e., they learn best from pictures), some like to be led (i.e., they follow along in a classroom, read a book, etc.), and some like to lead (i.e., they offer to teach a class, write a paper, write a book, etc.). I tend to fall into the latter category.

When I want to learn a new technology, I tend to put my neck out there on the topic. I'll offer to teach a class on a subject…or I'll set a personal goal of writing a blog entry about a new topic. This style of learning requires that I study the topic from top to bottom. If I'm going to teach a class at a user's group, I will have a hard deadline. I'll often go out and buy a few books on the topic (if any are available); I'll Google for articles and read them; and then—most important to me—I'll challenge myself to solve specific problems with the technology. I typically try to solve problems I've previously solved in another technology.

For example, when I first heard about Marvel, which later became Oracle HTML DB and is now Oracle Application Express or ApEx, oddly enough, my first thought was "this might be book material." At that time, no books were available on ApEx, so I thought that maybe I could write one of the first books on the topic. I ended up co-authoring with Larry Linnemeyer on this book. I downloaded the product, installed it on my laptop, and started trying to write a simple application. That was easy enough, so I challenged Larry to re-write an application that I had previously written using Oracle's Web Toolkit (which was the hard way of writing Oracle-based Web apps—PL/SQL code actually wrote the HTML dynamically). I would have dug in and written the app myself, but Larry had some bench time, so he and I worked on the app together. I did the design work (since I was very familiar with our prior version of the application—I had written it in Access) and Larry did the development.

SQL and then PL/SQL are obvious places to start in the Oracle world. You can write queries and get data out of an Oracle database—that sounds simple enough, right? Sure, for the simple stuff, it's simple, but when things get complex, writing a query is a bit more difficult. Beyond queries, you can manipulate the data too—this is called Data Manipulation Language or DML. How do you write a query to remove duplicate customers? How do you extract a spreadsheet from the database? How do you summarize the data? What about more complex structures like materialized views (previously called snapshots)? What is a materialized view log table? How about creating new tables and object types? This is called Data Definition Language or DDL. The topics for SQL alone go on endlessly. Oracle also has its own extensions to SQL for complex operations not supported by SQL92. Every database has slightly different SQL syntaxes that it supports. With my learning style, my goal is to challenge myself with different queries, and then write them—of course documenting what I learn mentally or on paper, or whatever works for you.

SQL returns a result set of data. In other words, it gives you a list of customers in the state of Colorado. That's nice to know. Yes, you put that information in a spreadsheet or you can print it on a report, but what if you want to do something with each and every row in the result set? That's where PL/SQL comes into play. The PL stands for Procedural Language— in other words, it's a 3GL for SQL. PL/SQL is another topic that you can study for years—and many have! So challenge yourself with PL/SQL. How can you write a block of PL/SQL code that takes a value as its input, looks up certain records, updates records, deletes some log records, and so on? The answer is by using PL/SQL, of course.

Once you've mastered SQL and PL/SQL, I'd recommend that you build an application using Oracle Application Express. This is where Oracle will get exciting for you. You can quickly write SQL that will display on a web page and can be exported to Excel. Using PL/SQL, you can add logic that will log each time a user clicks on a button (i.e., inserts data into the database).

As I suggested above, take the time to understand your learning style and then do what you need to do to learn the technologies. If you learn best by doing . . . then do! If you learn best by reading…buy the materials and read! If you learn best by teaching . . . then teach! If you learn best under pressure . . . put yourself under pressure—but if not, don't! ▲

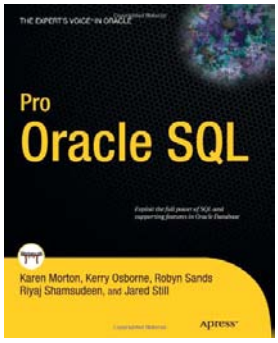
---

*Brad is a TUSC founder and is currently the chief technology officer (CTO) for Rolta TUSC. His vast experience and expertise have earned him roles as acting chief information officer of several companies over more than 22 years at TUSC. Brad has also served on numerous company boards. Brad is the author of several best-selling Oracle Press books. Oracle awarded Brad the honorary title of Oracle ACE Director for Fusion Middleware. Brad taught "New Venture Creation" at the University of Denver's Daniels College of Business for two years. In 2009, his alma mater, Illinois State University, put Brad into their first Hall of Fame for the College of Applied Science and Technology. He blogs at* **http://bradleydbrown.blogspot.com**.

# Pro Oracle SQL

## A Book Review by Dave Abercrombie

### Details




**Author:** Karen Morton, Kerry Osborne, Robyn Sands, Riyaj Shamsudeen, Jared Still

**ISBN-13:** 978-1-4302-3228-5

**ISBN-10:** 1-4302-3228-5

**Pages:** 600

**Publication Date:** December 2010

**Edition:** 1

**Price:** $49.99 (list)

**Publisher:** Apress, New York

### Summary

**Overall review:** An excellent resource for the Oracle SQL developer. It covers the basics well; topics such as access and join methods, execution plans, index design, plan stability, etc., are approached in a very practical way. However, what makes this book indispensable is its coverage of topics such as analytic functions, the MODEL clause, subquery factoring, and advanced DML. Helpful code listings and examples guide any reader toward greater proficiency with Oracle SQL.

**Target Audience**: SQL writers with some experience.

**Would you recommend this to others?** Yes.

**Who will get the most from this book?** An Oracle SQL developer and tuner.

**Is this book platform specific?** Mostly 10*g* and later.

**Why did I obtain this book?** The Oak Table Network endorses this book, and that alone was enough to inspire me to read it. Also, I really enjoy collaborative book efforts; the mix of style and perspective adds interest.

### Overall Review

This is a lengthy book, with a wide scope and real depth. Its target audience of Oracle SQL developers is well served by its advice. It does not get bogged down in unnecessary detail. Its conversational tone helps with readability, yet it is not too informal. Much of the advice, such as subquery factoring, can be used right away. Other topics, such as the MODEL clause or advance grouping, may not be relevant to you this week, but when the time comes to use those features, you must have this book on your shelf. Its biggest drawback is its lack of a bibliography, and its internal cross-referencing could be improved. But these are minor blemishes, and overall I'd rate this book as essential for any Oracle SQL developer.

### Chapters 1 and 2 – Core SQL and SQL Execution

Karen Morton starts this book with an introduction to Oracle's SQL implementation. The experienced reader might be tempted to skip over this chapter. We already know about the five core SQL statements SELECT, INSERT, UPDATE, DELETE, and MERGE, right? However, her treatment of these basic topics sets a nice tone for this book: it is clear and concise, with appropriate levels of detail and relevant code listings.

Karen quickly moves us into more advanced SQL topics. She leads with an analogy that contrasts a casual automobile driver with a professional mechanic. We all know how to "start the engine, shift gears, . . . use turn signals, and so on." But Karen promises to teach us "how the engine works, how to change the oil," to become more than just an "everyday driver."

Of course, we were all taught at first that SQL is a declarative language, and that we should not worry too much about the procedures used by Oracle to satisfy our queries. Obviously, this is not completely true. Knowledge of how Oracle works turns out to be essential for developing efficient, production quality data structures and queries.

Karen illustrates the high costs of hard parsing dramatically with a table summarizing the 59 queries against 17 system objects issued by the optimizer in order to parse a single SQL query. She clearly demonstrates the use of bind variables to reduce parsing costs.

Karen summarizes the high cost of library cache latching by returning to the automobile analogy. While trying to obtain a latch, an Oracle session is like "a kid in the backseat of a car that asks 'are we there yet?' over and over and over." This "spinning" consumes CPU resources and can lead to growth of the CPU run queue. She points out that this can be "time-consuming," and that "effects on performance and scalability can be dramatic."

Karen goes into good depth about the optimizer's query transformation. She wants to make sure we know that "it is the query transformer's main objective to determine if changing the way the query is written will provide a better query plan." She provides insightful examples of view merging, subquery unnesting, predicate pushing, and query rewrite with materialized views. Her discussion of conditions under which transformation will be used or ignored is very helpful and deserves careful study.

Discussion of the optimizer would not be complete without touching on table, column, and index statistics gathered by Oracle. Karen illustrates this topic with an example that uses her st-all.sql script. I am switching over to this script in my own work, due to its completeness and relevant format. She only briefly dips into issues of cardinality, selectivity, and effective cost estimates. Curiously, she does not refer the reader to Jonathan Lewis's important book on this topic, which is essential for mastering this topic.

## Chapters 3 and 4 – Access and Join Methods, Sets

To a SQL developer, the most important aspects of Oracle's internals are its access and join methods. Karen Morton provides a concise overview of these methods, at a level of detail appropriate for her target audience. Karen starts with table full scans, covering relevant issues such as multi-block reads and the highwater mark. She then summarizes Oracle's many methods for reading indexes. This section is only a couple-dozen pages long and about half code listings, so it does not include comprehensive details. In my experience, comprehension of these topics is greatly enhanced by getting quantitative—actually attempting to re-create the optimizer's cost estimates. For that greater level of detail, I refer the advanced reader to Jonathan Lewis's work on the cost-based optimizer.

Karen moves to an overview of Oracle's join methods: nested loops, hash joins, sort-merge, and Cartesian. She provides a basic level of detail, sufficient for a basic understanding of execution plans. I learned that the optimizer's decision to use hash join "depends not just on the number of rows, but the size of the rows as well, since the entire row must be stored in the hash table" (emphasis added).

In the next chapter, Karen addresses the need to avoid thinking procedurally and to think instead in terms of sets. Her example problems are deceptively simple: "Produce a list of all employees who spent the same number of years in each job they held," and "Compute an average number of days between orders for a customer." Karen concludes this chapter with an overview of Oracle's set operations: UNION, UNION ALL, MINUS, and INTERSECT.

## Chapters 5 – Questions

Karen Morton provides a good discussion of one of the most important issues for a SQL developer. Surprisingly, this is not a technical issue but one of communication and collaboration. I'll bet that you've been asked many times to tune or fix a query, and you are provided only with the broken SQL. Annoying, isn't it? How do you know what the query is actually supposed to do? In many cases, the purpose of the query is undocumented, and perhaps the original author is no longer with the company. Perhaps the best fix is to completely rewrite the query, but this could be dangerous unless you fully understand the necessary semantics.

Karen provides a fascinating taxonomy of questions. She compares "open" questions to "closed" ones, and contrasts "problem-solving" questions with "objective" ones. Karen further breaks down closed questions into three types: "identification," "selection," and "yes/no." I'll read this discussion again to prepare for my next meeting with product managers and users!

Karen shifts from people-based questions about the problem to questions about the data. She starts by advising us to "think like the optimizer would." Anyway, she rightly emphasizes the need for production-like data and statistics. Indeed, in my experience, nearly all really difficult performance problems can be traced to data distribution quirks and non-uniformities. You are unlikely to solve a difficult performance problem without understanding production data distributions.

## Chapter 6 – Execution plans

Karen Morton starts her discussion with a very welcome distinction between "explain plans" and "execution plans." The former are only approximate: easy to generate yet created using a different code path than the actual optimizer. They are often wrong, and Karen provides a useful, repeatable example. In contrast, "execution plans" are the actual plans used by Oracle and contain valuable runtime statistics. However, "execution plans" require actually running the query, which is not always feasible. In addition, in order to gather the valuable runtime statistics, the query must be run in a special mode that gathers all statistics.

Mastery of Oracle's DBMS_XPLAN supplied package is essential for effective SQL tuning, and Karen provides a useful tutorial on its use. Karen advises you to compare row counts estimated by the optimizer to row counts actually observed at runtime. This can help you to understand where the optimizer went astray.

However, Karen repeats a common misconception about reading plans. She states "the operation that is most indented is actually the first operation that will be executed." Consider a nested loop where the outer, driving table was small and full scanned, and the inner table was accessed via primary key. The most indented operation will be the INDEX UNIQUE SCAN of the primary key. But this can happen only after the less-indented full scan of the outer table has started. In my experience, the best advice for reading plans is provided in Chapter 6 of Christian Antognini's book *Troubleshooting Oracle Performance*.

## Chapter 7 – Advanced Grouping

We have all used the GROUP BY clause in our SQL. Most of us understand the HAVING clause, knowing that it resembles the WHERE clause. But few of us are comfortable with the more advanced grouping clauses like CUBE, GROUPING, GROUPING_ID, and ROLLUP. Jared Still demystifies these clauses in this detailed, example-based tutorial.

This function has some very unusual characteristics. Jared explains that "when used with the GROUP BY clause, it will cause all possible combinations of the elements included in the arguments to CUBE to be considered for each row. This operation will generate more rows than actually exist in the table." It gets stranger. Jared continues "when generating the combinations of all columns included in the arguments to CUBE, a NULL value will be generated n-1 times for each column, where n is the number of columns in the list." Jared clarifies this with an example using ". . . cube(cust_income_level,age_range) . . ." where he explains that "you can expect to see a NULL value for CUST_INCOME_LEVEL generated once for each distinct value of AGE_RANGE." Jared points out that "these null values can be a problem if there are rows in the data that have NULL values for either of these columns. How do you discern between NULLs in the data and NULLs inserted by the CUBE extension?" This is where the GROUPING() function comes in: "the value returned will be a 1 if the current row is a superaggregate row generated by CUBE and a 0 for all other cases." The GROUPING_ID() function generalizes this. It "evaluates an expression, determines which, if any, of the

columns in its arguments are being used to generate a superaggregate row, creates a bit vector, and returns that value as an integer."

Confused? So was I. Jared warns us that "it takes a little effort to get started, as the benefits of the newer functionality are not always clear until you spend some time learning to use them." Jared's helpful tutorial walks us through these features one tiny step at a time, allowing us to make sense of them. Even so, they are so unusual that you will need to sit down with SQL*Plus and work through these examples at the keyboard in order to see how they can benefit you.

*"This book is a must-have for any Oracle SQL developer, except perhaps for the brand-new beginner. Advice on topics such as subquery factoring can and should be used right away. More complicated topics, such as the MODEL clause or advanced grouping, will require more effort and practice to understand, but this book will get you up to speed quickly when you need to tackle them. This book belongs in your hands."*

### Chapters 8 and 9 – Analytic Functions and the MODEL Clause

Riyaj Shamsudeen provides wonderful explanations and clear examples for using analytic functions. Have you ever needed to refer to combined data from multiple rows of the same table? For example, say you have a table containing a time series of observations of a changing numeric value, such as daily stock closing prices. Let's say you needed to calculate the change in price for each day. A common solution is to do a self-join, joining one day's row with the preceding day's so that you could do the subtraction. This self-join approach can be cumbersome and inefficient: for example, how would you deal with data gaps over the weekend? An elegant solution is provided by the analytic function LAG(). You simply sort the rows by date, and then use the LAG() function to refer to the price value from the preceding day. No self-join is required, and the SQL is efficient and easy to read and maintain.

Riyaj teaches us that "analytic functions have three basic components: partitioning-clause, order-by-clause, and the windowing-clause." If you have tried using these functions, you might have been put off by the somewhat confusing Oracle documentation. Riyaj gently guides us through these concepts so that they become obvious, natural, and compelling. In addition to the explanations and examples, Riyaj describes their practical use cases, and compares and contrasts them.

I'll bet that most of you have never used Oracle's MODEL clause. It allows you to do calculations much as you would in a spreadsheet. Database programmers are encouraged to think in terms of sets (e.g., Chapter 4 of this book), so the notion of using spreadsheet-style relative cell addressing seems quite foreign. So Riyaj's step-by-step tutorial, based on excellent examples, is especially welcome.

Sure, for small datasets a spreadsheet works fine. But unlike a spreadsheet, Oracle's "MODEL clause is highly scalable . . . multi-dimensional, multi-variate calculations over millions of rows, if not billions of rows, can be implemented easily . . . and database features such as object partitioning and parallel execution further improve scalability."

How does it work? Riyaj explains that you define "three groups of columns: partitioning columns, dimension columns, and measures columns. Partitioning columns are analogous to a sheet in a spreadsheet. Dimension columns are analogous to row rags (A,B,C..) and column tags (1,2,3..). The measures columns are analogous to [spreadsheet] cells." Finally, the MODEL clause "rules" are analogous to the formulas used in spreadsheets. Additional features of the MODEL clause include using positional or symbolic notation, FOR loops, iteration, and aggregation. I know that this all sounds confusing, but Riyaj's excellent tutorial makes it clear and accessible.

Riyaj's coverage of analytic functions and the MODEL clause is comprehensive. He goes way beyond the Oracle documentation to cover practical performance issues, execution plan interpretation, and index design. He even warns us that "the hardest thing about analytic functions is the organizational resistance to change. [We] are comfortable writing SQL statements using conventional syntax. Using analytic syntax will not come easy." I guarantee that you will be inspired to start using analytic functions and the MODEL clause after reading these chapters.

### Chapter 10 – Subquery factoring

Jared Still provides a complete tutorial on the use of the WITH clause. This allows you to break down a complex query into its component parts. Each part is becomes easier to understand and maintain, and might even become reusable. The concept is similar to inline views, but it encourages even more readability and modularity. This feature is not as well known as it should be, Jared points out that "Prior to the release of Oracle 11*g*R2, the official Oracle documentation barely mentions it, providing just a brief synopsis of its use, a couple of restrictions, and a single example." The ANSI standard calls it "common table expressions" but apparently, Oracle Corporation "has chosen to obfuscate it." Jared's discussion is thorough and inspiring.

### Chapter 11 – Semi-Joins and Anti-Joins

Kerry Osbourne provides a useful discussion of semi-joins (WHERE EXISTS or WHERE IN) and anti-joins (WHERE NOT EXISTS or WHERE NOT IN). He carefully contrasts

these to regular joins, dispelling myths and highlighting gotchas. The main advantage to these join operations is that they can "cut short the processing of normal hash, merge, or nested loop joins," often providing "dramatic performance improvements." Particularly helpful is his discussion of nulls and anti-joins, and 10*g*'s use of the LNNVL function that disables potential index paths.

### Chapter 12 – Indexes

Riyaj Shamsudeen provides a succinct overview of indexes and Oracle's physical implementations. This overview is geared toward practical use by the SQL developer. For example, he does not go into great detail about bitmap index data structures, but instead focuses on their practical benefits and limitations. His emphasis is on useful information, and it is well organized and accessible.

### Chapter 13 – DML

So far, this book has focused on SELECT statements. But if course, the data has to get into the database somehow, and so Kerry Osbourne provides a nice overview of DML operations. He delves in depth into performance issues such as direct path inserts, fully describing their advantages and disadvantages. He demonstrates useful features you might not be aware of, such as multi-table inserts and conditional inserts. He shows how these features can be very useful when loading data from external systems, since you can avoid procedural code, and they can be "applied directly to the loading process, rather than after they have been staged in an Oracle table." I particularly enjoyed the tutorial on DML error logging that can divert into another table those rows that would otherwise cause the operation to fail.

### Chapter 14 and 15 –Transactions, Testing

Robyn Sands teaches us about transactions. She reminds us about ACID compliance, and how Oracle achieves it. She provides a useful overview of issues such as transaction isolation levels and multi-version read consistency. These issues can be complicated and abstract, but Robyn helps us by focusing on practical, useful examples. Her next chapter is on testing, something that is too easy to let slide. She describes approaches for using both unit tests and regression tests in a database environment. I agree with her claim that "the data model is the single most important indicator of a database application's potential to perform." She tackles the difficult problem of refining your data model while maintaining code quality. Finally, Robyn advocates for the use of instrumentation based on Oracle's DBMS_APPLICATION_INFO supplied package. She recommends the Instrumentation Library for Oracle (ILO) that is written by Method-R.

### Chapter 16 – Plan Stability

Kerry Osbourne closes the book with a discussion of plan stability. He discusses its causes, such as changes to statistics and execution environment, as well as bind variable peeking. He describes how to find instability, and I especially liked his suggestion to use statistical variance, a technique that I wish was more widely used and appreciated. He provides comprehensive step-by-step tutorials on stabilizing plans, using hints, control of statistics, database parameters, indexes, outlines, and profiles.

### Conclusion

This book is a must-have for any Oracle SQL developer, except perhaps for the brand-new beginner. Advice on topics such as subquery factoring can and should be used right away. More complicated topics, such as the MODEL clause or advanced grouping, will require more effort and practice to understand, but this book will get you up to speed quickly when you need to tackle them. This book belongs in your hands. ▲

---

*Dave Abercrombie has worked at Convio (with a "v," not an "f") for about ten years, having helped to found GetActive Software before its merger with Convio. This company's business model is rather like a distributed denial of service attack against itself. Its customers are non-profit membership organizations who want to use the Web to engage and activate their members. So each day, Convio sends tens of millions of emails to these members, and then tracks the ensuing member transactions and activities, such as donations, advocacy, and click-throughs. Dave has honed his troubleshooting and scalability skills by keeping these very busy databases happy. He has presented at Hotsos and is becoming a regular presenter at NoCOUG. He can be reached at* **dabercrombie@convio.com**.

## Second International

# NoCOUG SQL Challenge

**BE IT KNOWN BY THESE PRESENTS** that the great Wizard of Odds at Hogwash School of Es-Cue-El needs your help in solving the riddle of the ancient manuscript found in the secret chamber of mystery. A great tournament has been organized, and all practitioners of the ancient arts of Es-Cue-El have been invited to demonstrate their prowess.

# Unsolvable Riddle

An ancient manuscript titled "Love Your Data" has been discovered in the secret chamber of mystery at Hogwash School of Es-Cue-El. The manuscript was covered with mysterious words such as those shown below; the complete list has been published in the *Wizarding Journal* of the great Oracles of Northern California. The great Wizard of Odds implores you to create an Es-Cue-El spell that reveals the secret message.

| | | |
|---|---|---|
| | A | |
| COMPREHENSION | ABILITY | OLD |
| | ABOUT | |
| | ALWAYS | |
| SCIENCE | AND | PHYSICS |
| | ANY | |
| | AS | |
| SO | ASK | ABILITY |

# Big Prizes

The *August Order of the Wooden Pretzel* will be conferred on the winner, in keeping with the celebrated pronouncement of another great wizard that *"some people can perform seeming miracles with straight Es-Cue-El, but the statements end up looking like pretzels created by somebody who is experimenting with hallucinogens."* As if that singular honor were not enough, a marvelous collection of Oracular tomes will be bestowed upon the champion. May the best wizard win!

**RULES**: The winner will receive a Kindle and his or her choice of six books from the Apress book catalog. Due to shipping costs and limitations for certain parts of the world, electronic copies may be substituted. Prizes may be awarded to runners-up at the discretion of the organizers. Submissions should be emailed to **SQLchallenge@nocoug.org**. Contestants may use any database technology at their disposal, but the submitted solutions should be compatible with at least one of the following database technologies: Oracle 11*g* for Windows, SQL Server 2008, DB2 9.5 for Windows, and MySQL 5.1 for Windows. The competition will be judged by Jonathan Gennick, Assistant Editorial Director at Apress, and Iggy Fernandez, author of *Beginning Oracle Database 11*g *Administration* published by Apress. Judging criteria include correctness, originality, efficiency, portability, and readability. The judges' decisions are final. The competition will close at a time determined by the organizers. The judges and organizers reserve the right to publish and comment on any of the submissions with due credit to the originators. More information about the problem and additional rules can be found at **http://www.nocoug.org**.

# Love Your Data

| | | |
|---|---|---|
| | A | |
| COMPREHENSION | ABILITY | OLD |
| | ABOUT | |
| | ALWAYS | |
| SCIENCE | AND | PHYSICS |
| | ANY | |
| | AS | |
| SO | ASK | ABILITY |
| | BE | |
| POSSIBLY | BEEN | MARILYN |
| GRAY | BETWEEN | EARS |
| | BEYOND | |
| | BUT | |
| | CANNOT | |
| SCIENTISTS | CARE | STRUCTURE |
| | CHALLENGES | |
| READING | COMPREHENSION | WRITING |
| WILL | DICTIONARIES | THESAURI |
| HUNDRED | DISTINCT | WORDS |
| WHAT | DO | YOU |
| THREE | DONE | |
| | DONT | |
| YOUR | EARS | SERIOUSLY |
| ASK | ELBOW | WANT |
| | ELEVEN | |
| EVEN | ENCYCLOPEDIAS | BUT |
| | EVEN | |
| | FASHIONED | |
| ELBOW | FIVE | DONE |
| | FOUR | |
| | GOOD | |
| THAT | GRAY | MATTER |
| | GREASE | |
| PREFERENCE | HAS | ALWAYS |
| RESULTS | HEY | LOOK |
| | HUNDRED | |
| IN | I | THOSE |
| | IF | |
| ONE | IN | IS |
| SINGLE | IS | TOUGH |
| | LEFT | |
| | LETTERS | |
| | LOOK | |
| WITH | MANY | LETTERS |
| BETWEEN | MARILYN | THIS |
| SOME | MATH | FIVE |
| | MATTER | |
| | MAYBE | |
| ELEVEN | MORE | LEFT |
| ENCYCLOPEDIAS | MY | HAS |

| | | |
|---|---|---|
| REPEAT | OF | THEN |
| GOOD | OLD | FASHIONED |
| TO | ONE | DISTINCT |
| HEY | ONLY | MORE |
| | OR | |
| | PARAGRAPH | |
| | PHYSICS | |
| DICTIONARIES | POSSIBLY | MY |
| | PREFERENCE | |
| | PROBABLY | |
| | READERS | |
| | READING | |
| | REALLY | |
| CANNOT | REPEAT | ANY |
| | REQUIRE | |
| | RESULTS | |
| | SCIENCE | |
| GREASE | SCIENTISTS | DONT |
| | SERIOUSLY | |
| MANY | SHOULD | USED |
| A | SINGLE | PARAGRAPH |
| REQUIRE | SKILLS | BEYOND |
| AND | SO | DO |
| I | SOME | BEEN |
| ABOUT | STRUCTURE | THEY |
| CHALLENGES | SUCH | AS |
| | THAT | |
| | THEM | |
| THEM | THEN | PROBABLY |
| THESAURUSES | THESAURI | OR |
| | THESAURUSES | |
| | THEY | |
| SUCH | THIS | SKILLS |
| OF | THOSE | SHOULD |
| FOUR | THREE | TWO |
| TRYING | TO | TYPE |
| REALLY | TOUGH | IF |
| | TRYING | |
| | TWO | |
| | TYPE | |
| BE | USED | MAYBE |
| | UTILIZE | |
| CARE | WANT | ONLY |
| | WHAT | |
| READERS | WILL | UTILIZE |
| | WITH | |
| | WORDS | |
| | WRITING | |
| | YOU | |
| | YOUR | |

Pythian

love your data

# SQL 101: Which Query Is Better?—Part III

by Iggy Fernandez

In 1988, a SQL researcher named Fabian Pascal wrote an article (**http://www.dbdebunk.com/page/page/1317920.htm**) for *Database Programming and Design* in which he quoted Chris Date as follows:

*"SQL is an extremely redundant language. By this I mean that all but the most trivial of problems can be expressed in SQL in a variety of different ways. Of course, the differences would not be important if all formulations worked equally well but that is unlikely. As a result, users are forced to spend time and effort trying to find the "best" formulation (that is, the version that performs best)—which is exactly one of the things the relational model was trying to avoid in the first place."*

Pascal then went on to test seven equivalent queries with five different database engines. Only one out of the five database engines came anywhere near to acing the test; it appeared to use the same execution plan for six of the queries but did not support the seventh query. The other engines used a range of query plans with different execution times. Pascal then predicted that

*"Eventually, all SQL DBMSs, for competitive reasons, will have to equalize the performance of redundant SQL expressions and to document their execution plans. Forcing users to maximize performance through query formulation is not only unproductive, but simply a lost cause, especially if there is no guidance from the system. The more users understand the relational model and its productivity intentions, the more they will demand equalized performance and documented execution plans from vendors, instead of doggedly attempting to undertake unnecessary and futile burdens."*

Let's find out if Pascal's 20-year-old prediction of equalized performance came true. First, let's create tables similar to those that Pascal used. We need a table called Personnel containing employee details and a table called Payroll containing salary payments. The Payroll table is linked to the Personnel table by the Employee ID. The tests were conducted using Oracle Database 11*g* Enterprise Edition Release 11.2.0.1.0 on 32-bit Windows XP Professional Version 2002 Service Pack 3.

```
CREATE TABLE personnel
(
  empid CHAR(9) NOT NULL,
  lname CHAR(15) NOT NULL,
  fname CHAR(12) NOT NULL,
  address CHAR(20) NOT NULL,
  city CHAR(20) NOT NULL,
  state CHAR(2) NOT NULL,
  ZIP CHAR(5) NOT NULL
);

CREATE TABLE payroll
(
  empid CHAR(9) NOT NULL,
  bonus INTEGER NOT NULL,
  salary INTEGER NOT NULL
);

INSERT INTO personnel
SELECT
  TO_CHAR(LEVEL, '09999') AS empid,
  DBMS_RANDOM.STRING('U', 15) AS lname,
  DBMS_RANDOM.STRING('U', 12) AS fname,
  '500 ORACLE PARKWAY' AS address,
  'REDWOOD SHORES' AS city,
  'CA' AS state,
  '94065' AS zip
FROM
  dual
CONNECT BY LEVEL <= 9900;

INSERT INTO payroll(
  empid,
  bonus,
  salary
)
SELECT
  empid,
  0 AS bonus,
  99170 + 10000 * CEIL(DBMS_RANDOM.VALUE * 10)
    AS salary
FROM
  personnel;

CREATE UNIQUE INDEX personnel_pk
  ON personnel(empid);

CREATE UNIQUE INDEX payroll_pk
  ON payroll(empid);

ALTER TABLE personnel
  ADD CONSTRAINT personnel_pk
  PRIMARY KEY (empid);

ALTER TABLE payroll
  ADD CONSTRAINT payroll_pk
```

```
  PRIMARY KEY (empid);

ALTER TABLE payroll
  ADD CONSTRAINT payroll_fk1
  FOREIGN KEY (empid)
  REFERENCES personnel(empid);

EXEC DBMS_STATS.GATHER_TABLE_STATS(
  ownname=>'HR',
  tabname=>'PERSONNEL'
);

EXEC DBMS_STATS.GATHER_TABLE_STATS(
  ownname=>'HR',
  tabname=>'PAYROLL'
);
```

Observe that we gathered optimizer statistics for the newly created tables, although they are not strictly necessary for the tests that follow. A database engine can generate query plans even in the absence of statistics and should generate the same plan for all semantically equivalent queries whether or not statistics are available.

The SQL exercise is to list the employees who were paid $199,170. Here are five different ways in which a programmer might formulate the required query, three of which are from Pascal's article. Focus on the plan hash values in the query plans below. We see that Oracle uses four different query plans, meaning that Pascal's prediction of equalized performance for equivalent queries is yet to be fulfilled.

The first formulation uses an unnecessary join when only a semi-join is strictly necessary. However, in this particular case, a semi-join is equivalent to a join because the Payroll table is linked to the Personnel table by the Employee ID. Oracle uses a HASH JOIN in the EXPLAIN PLAN.

```
SELECT lname
FROM personnel, payroll
WHERE personnel.empid = payroll.empid
AND salary = 199170;

Plan hash value: 2476600843
```

| Id | Operation | Name |
|---|---|---|
| 0 | SELECT STATEMENT | |
| 1* | HASH JOIN | |
| 2* | TABLE ACCESS FULL | PAYROLL |
| 3 | TABLE ACCESS FULL | PERSONNEL |

```
1 - access("PERSONNEL"."EMPID"="PAYROLL"."EMPID")
2 - filter("SALARY"=199170)
```

The second formulation implements a semi-join using an uncorrelated subquery. Oracle neatly converts this into a join and once again uses a HASH JOIN in the EXPLAIN PLAN. The query plan is identical to the previous one.

```
SELECT lname
FROM personnel
WHERE empid IN (
  SELECT empid
  FROM payroll
  WHERE salary = 199170
);
```

***"SQL is an extremely redundant language. By this I mean that all but the most trivial of problems can be expressed in SQL in a variety of different ways. Of course, the differences would not be important if all formulations worked equally well but that is unlikely. As a result, users are forced to spend time and effort trying to find the 'best' formulation (that is, the version that performs best)—which is exactly one of the things the relational model was trying to avoid in the first place."***

```
Plan hash value: 2476600843
```

| Id | Operation | Name |
|---|---|---|
| 0 | SELECT STATEMENT | |
| 1* | HASH JOIN | |
| 2* | TABLE ACCESS FULL | PAYROLL |
| 3 | TABLE ACCESS FULL | PERSONNEL |

```
1 - access("EMPID"="EMPID")
2 - filter("SALARY"=199170)
```

The third formulation implements a semi-join using a correlated subquery. This time Oracle declines to convert it into a regular join.

```
SELECT lname
FROM personnel
WHERE EXISTS (
  SELECT *
  FROM payroll
  WHERE personnel.empid = payroll.empid
  AND salary = 199170
);

Plan hash value: 1844477241
```

| Id | Operation | Name |
|---|---|---|
| 0 | SELECT STATEMENT | |
| 1* | HASH JOIN RIGHT SEMI | |
| 2* | TABLE ACCESS FULL | PAYROLL |
| 3 | TABLE ACCESS FULL | PERSONNEL |

```
1 - access("PERSONNEL"."EMPID"="PAYROLL"."EMPID")
2 - filter("SALARY"=199170)
```

The fourth formulation uses a scalar subquery in the WHERE clause. Oracle uses a different query plan.

```
SELECT lname
FROM personnel
WHERE (
  SELECT salary
  FROM payroll
  WHERE personnel.empid = payroll.empid
) = 199170;

Plan hash value: 1680572657
```

| Id | Operation | Name |
|---|---|---|
| 0 | SELECT STATEMENT | |
| 1 | FILTER | |
| 2* | TABLE ACCESS FULL | PERSONNEL |
| 3* | TABLE ACCESS BY INDEX ROWID | PAYROLL |
| 4 | INDEX UNIQUE SCAN | PAYROLL_PK |

```
1 - filter(=199170)
4 - access("PAYROLL"."EMPID"=:B1)
```

The fifth formulation uses a scalar subquery in the SELECT clause. Oracle uses yet another query plan.

```
SELECT
  (
    SELECT lname
    FROM personnel
    WHERE personnel.empid = payroll.empid
  )
FROM payroll
WHERE salary = 199170;

Plan hash value: 1891291052
```

| Id | Operation | Name |
|---|---|---|
| 0 | SELECT STATEMENT | |
| 1 | TABLE ACCESS BY INDEX ROWID | PERSONNEL |
| 2* | INDEX UNIQUE SCAN | PERSONNEL_PK |
| 3* | TABLE ACCESS FULL | PAYROLL |

```
2 - access("PERSONNEL"."EMPID"=:B1)
3 - filter("SALARY"=199170)
```

Christopher Charles laments the need to tune SQL manually in a well-written piece titled "My Brilliant Career Tuning SQL" (**http://christophercharles.blogspot.com/2003/07/databases-my-brilliant-career-tuning.html**). Why was Ingres able to achieve equalized performance in 1988? Because back in 1988, Ingres internally converted SQL queries into a QUEL representation. As Pascal explains (**http://searchoracle.techtarget.com/tip/When-will-they-ever-learn**), *"QUEL was a different relational data language, one property of which was lack of sub-queries, meaning that however you expressed a query in SQL, it would map to only one QUEL execution (Chris Date designed the mapping), hence the ease of optimization and consistency."* ▲


## Usually, it's harder to pinpoint.

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

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

***A smarter solution makes everyone look brilliant.***

CONFIO SOFTWARE

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

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

# 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, Iggy Fernandez, at **iggy_fernandez@hotmail.com**. ▲

*Long-term event sponsorship:*

CHEVRON

ORACLE CORP.

## Thank you! Year 2011 Gold Vendors:

- 3PAR
- Confio Software
- Database Specialists, Inc.
- Delphix
- Rolta TUSC

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

## TREASURER'S REPORT

Naren Nagtode, *Treasurer*

| | | |
|---|---|---|
| ***Beginning Balance*** | | |
| October 1, 2010 | | **$ 49,417.69** |
| **Revenue** | | |
| Membership Dues | 3,605.00 | |
| Meeting Fees | 450.00 | |
| Vendor Receipts | 500.00 | |
| Advertising Fee | 1,600.00 | |
| Training Day | 1,200.00 | |
| Sponsorship | – | |
| Interest | 3.22 | |
| Paypal balance | – | |
| **Total Revenue** | | **$ 7,358.22** |
| **Expenses** | | |
| Regional Meeting | 6,343.12 | |
| Journal | 3,751.00 | |
| Membership | – | |
| Administration | 1,481.36 | |
| Website | – | |
| Board Meeting | 630.01 | |
| Marketing | 100.00 | |
| Insurance | 527.00 | |
| Vendors | – | |
| Tax | – | |
| Training Day | 6,792.62 | |
| IOUG-rep | – | |
| Miscellaneous | 309.42 | |
| **Total Expenses** | | **$ 19,934.53** |
| ***Ending Balance*** | | |
| December 31, 2010 | | **$ 36,841.38** |

# NoCOUG Winter Conference

## Session Descriptions

*For the most up-to-date information, please visit* **http://www.nocoug.org**.

### –Keynote–

**Optimizing**
*Cary Millsap, Method R* . . . . . **9:30–10:30**

*Cary Millsap is the founder and president of Method R Corporation (***http://www.method-r.com***), a company devoted to genuinely satisfying software performance. He is widely known in the Oracle community as a speaker, educator, consultant, and writer. He is the author (with Jeff Holt) of* Optimizing Oracle Performance, *for which he and Jeff were named* Oracle Magazine*'s 2004 Authors of the Year. He is also a contributor to* Oracle Insights: Tales of the Oak Table. *Cary is the former vice president of Oracle's System Performance Group, and a co-founder of Hotsos. Cary is also an Oracle ACE Director and a founding partner of the Oak Table Network, an informal association of Oracle scientists that are well known throughout the Oracle community.*

### –Auditorium–

**Thinking Clearly About Performance**
*Cary Millsap, Method R* . . . . . 11:00–12:00

Creating high performance as an attribute of complex software is an extremely difficult achievement for developers, technology administrators, architects, system analysts, project managers, and their users. However, by understanding some fundamental principles, performance problem solving and prevention can be made far simpler and more reliable. This paper describes those principles, linking them together in a coherent journey covering the goals, terms, tools, and decisions that you need to maximize your application's chance of having a long, productive, high-performance life. Examples in this paper touch upon Oracle experiences, but the scope of the paper is not restricted to Oracle products.

**Thinking Clearly About Performance** *(continued)*
*Cary Millsap, Method R* . . . . . 1:00–2:00

**Maximize Your Oracle Investment with Enabling Technologies from EMC and VMware**
*Jason Kotsaftis, EMC* . . . . . 2:30–3:30

Oracle applications and databases are delivering mission-critical data and drive critical decision making in small, medium, and global enterprises. Increasing performance, efficiency, and management of these environments to deliver new value back to the business is a driving force in 2011 and beyond. With the advent of advanced virtualization, cloud computing, and data de-duplication, many customers are re-architecting their server, storage, backup, and network architectures. Technology innovations from EMC, VMware, Intel, Cisco, and others are delivering customers 45% performance improvements with Virtual RAC environments, consolidating by 80%, and streamlining management.

Come learn from technology experts as they share their hands-on knowledge of developing Integrated, proven solutions based on actual environments that showcase real savings in cloud computing. The topics will cover Oracle virtualization benefits, how to establish optimum backup and recovery processes, and how to plan for the virtual data center.

**Database Performance Made Easy Through Graphics**
*Kyle Hailey, Delphix* . . . . . 4:00–5:00

In his presentation covering Average Active Sessions (AAS) Formulas, Interpreting Load Charts, and Visual SQL Tuning (VST), Kyle begins with a discussion of the use of sample data to create Active Sessions History (ASH). But ASH is dense, multidimensional data, which makes it difficult to analyze. Fortunately, ASH can be simplified using a metric called Average Active Sessions (AAS). AAS shows the load on the database and allows you to drill down with precision and clarity into the cause of each bottleneck. Most often, bottlenecks are caused by poor-performing SQL statements. For analyzing suboptimal SQL statements, Kyle will show you how to lay out the query graphically in a Visual SQL Tuning (VST) diagram. Through the VST, you'll learn how to quickly visualize any coding errors in the query; discover flaws in the underlying database schema; and most importantly, find the best execution path through the query.

### –Tassajara–

**MySQL in Production—A War Story** **Editor's Pick**
*Chris Schneider, Ning* . . . . . 11:00–12:00

If you have been using Oracle or other RBDMS solutions in production and don't believe that MySQL can be run in production with great success, this talk is for you. Over the years MySQL has proven to be a free and reliable data store for some of the largest companies in the world, such as Facebook, Go Daddy, and Google. During this presentation I will share some of the experiences I've had over the years along with the different MySQL setups used.

During this presentation we will dive into the following:

- Hardware needs
- Scaling and sharding

*(continued on page 26)*

(continued from page 25)

- MySQL configurations
- Tips, tricks, and gotchas

### Best Practices for Upgrading to Oracle Database 11*g*

*Mark Townsend, Oracle* . . . . . 1:00–2:00

If you are planning to upgrade to Oracle Database 11*g* from Oracle 9*i* Database or Oracle Database 10*g*, this session is a rare opportunity to learn firsthand from an Oracle expert. In this session, we'll cover all the basics you need to know about upgrading to Oracle Database 11*g*, including all the required preparatory steps, which upgrade strategies best fit your environment, and how to address post-upgrade performance analysis—along with helpful tips and common pitfalls to watch out for.

### Extreme Performance Data Warehousing

*Hermann Baer, Oracle*. . . . . 2:30–3:30

Extreme query performance and in-database analytics are key requirements for data-driven organizations looking to uncover revenue opportunities and take action on trends faster. However, the need to ask more complex questions and analyze greater volumes of information can conflict with these requirements.

In this presentation, we'll examine how intelligent optimizations in Oracle Exadata and Oracle Database 11*g* can improve data warehouse performance by a factor of 10x, reduce storage footprint by 10x, and meet all of your data-warehousing requirements, from simple ad-hoc queries to complex predictive analytics.

### Don't Be in a Funk—Use Analytic Functions

*Philip Rice, UCSC* . . . . . 4:00–5:00

Analytics are similar to the more traditional functions such as sum, count, and avg, but they can offer much more power. Analytics are most helpful in a reporting environment, but all developers and DBAs can gain by seeing the flexibility and performance benefits. We'll start with basics, including syntax and a comparison with the traditional count function. We'll move into real-world examples in use at the UC Santa Cruz campus. The session is intended to raise awareness about alternatives in using SQL. The slides were already used for several presentations within UCSC, and NoCOUG members can use the slides as the basis for internal sessions at your own site.

## –Diablo–

### Get Proactive with DBA Task Automation

*Jagan Athreya, Oracle* . . . . . 11:00–12:00

Learn about Oracle's unique database management capabilities for automating many of the mundane and manual tasks facing DBAs. You'll discover how Oracle Database's intelligent self-managing infrastructure automates routine tasks that control configuration drift and provide rapid database patching and provisioning, as well as enforce security policies and compliance requirements. Whether you're an experienced DBA or someone looking to learn new skills, you won't want to miss this session.

### NoSQL Deep Dive

*Gwen Shapira, Pythian* . . . . . 1:00–2:00

NoSQL is a general name for a new type of database—one that throws away the relational model in favor of better scalability. In this presentation, Gwen Shapira will discuss the new model, which problems it was designed to solve, and when it makes sense to use it. The presentation will include details about the specific algorithms and design patterns used in NoSQL databases; an overview of the new data models; and most importantly, everything you need to check when picking a NoSQL solution to make sure it matches your requirement and will allow you to success-fully adapt it for your applications.

### How DBAs Can Help The Success of a BI Project

*Sunil Ranka, Trianz*. . . . . 2:30–3:30

Often, most of the success of a BI project is attributed to functionality; features; scalability; and most importantly, data. DBAs play a significant role in the success of any BI project. Most BI tools are smart SQL generators, but having an insufficient or misconfigured database can take a toll on performance as well as the success of a project. During this conversation we will touch upon the following topics:

- The role of data in a BI project
- The need to smart DBAs for the success of any BI project
- How DBAs can help
- The things DBAs need to know for any BI project.

### Oracle Application Express 4.0

*Raj Mattamal, Niantic Systems* . . . . . 4:00–5:00

This session will introduce all the exciting new features in APEX 4.0, the increasingly popular feature of Oracle Database 11*g* that enables you to quickly develop web-based applications that are fast, secure, and scalable. You will learn how Websheets, a new feature of APEX 4.0, makes application development even easier for nontechnical business users. This session will also cover other new features such as dynamic actions, plugins, charts, improved application builder, tabular forms, and REST Web Services. ▲

# Database Specialists: DBA Pro Service

## DBA PRO BENEFITS

- *Cost-effective and flexible extension of your IT team*
- *Proactive database maintenance and quick resolution of problems by Oracle experts*
- *Increased database uptime*
- *Improved database performance*
- *Constant database monitoring with Database Rx*
- *Onsite and offsite flexibility*
- *Reliable support from a stable team of DBAs familiar with your databases*

## CUSTOMIZABLE SERVICE PLANS FOR ORACLE SYSTEMS

Keeping your Oracle database systems highly available takes knowledge, skill, and experience. It also takes knowing that each environment is different. From large companies that need additional DBA support and specialized expertise to small companies that don't require a full-time onsite DBA, flexibility is the key. That's why Database Specialists offers a flexible service called DBA Pro. With DBA Pro, we work with you to configure a program that best suits your needs and helps you deal with any Oracle issues that arise. You receive cost-effective basic services for development systems and more comprehensive plans for production and mission-critical Oracle systems.

### DBA Pro's mix and match service components

**Access to experienced senior Oracle expertise when you need it**
We work as an extension of your team to set up and manage your Oracle databases to maintain reliability, scalability, and peak performance. When you become a DBA Pro client, you are assigned a primary and secondary Database Specialists DBA. They'll become intimately familiar with your systems. When you need us, just call our toll-free number or send email for assistance from an experienced DBA during regular business hours. If you need a fuller range of coverage with guaranteed response times, you may choose our 24 x 7 option.

**24 x 7 availability with guaranteed response time**
For managing mission-critical systems, no service is more valuable than being able to call on a team of experts to solve a database problem quickly and efficiently. You may call in an emergency request for help at any time, knowing your call will be answered by a Database Specialists DBA within a guaranteed response time.

**Daily review and recommendations for database care**
A Database Specialists DBA will perform a daily review of activity and alerts on your Oracle database. This aids in a proactive approach to managing your database systems. After each review, you receive personalized recommendations, comments, and action items via email. This information is stored in the Database Rx Performance Portal for future reference.

**Monthly review and report**
Looking at trends and focusing on performance, availability, and stability are critical over time. Each month, a Database Specialists DBA will review activity and alerts on your Oracle database and prepare a comprehensive report for you.

**Proactive maintenance**
When you want Database Specialists to handle ongoing proactive maintenance, we can automatically access your database remotely and address issues directly — if the maintenance procedure is one you have pre-authorized us to perform. You can rest assured knowing your Oracle systems are in good hands.

**Onsite and offsite flexibility**
You may choose to have Database Specialists consultants work onsite so they can work closely with your own DBA staff, or you may bring us onsite only for specific projects. Or you may choose to save money on travel time and infrastructure setup by having work done remotely. With DBA Pro we provide the most appropriate service program for you.

Database Specialists

CALL 1-888-648-0500 TO DISCUSS A SERVICE PLAN

**NoCOUG**
P.O. Box 3282
Danville, CA 94526


# NoCOUG Winter Conference Schedule

## February 24, 2011—CarrAmerica Conference Center, Pleasanton, CA

Please visit **http://www.nocoug.org** for updates and directions, and to submit your RSVP.
**Cost:** $50 admission fee for non-members. Members free. Includes lunch voucher.

| | |
|---|---|
| 8:00 a.m.–9:00 | Registration and Continental Breakfast—Refreshments served |
| 9:00–9:30 | **Welcome:** Iggy Fernandez, NoCOUG president |
| 9:30–10:30 | **Keynote:** *Optimizing*—Cary Millsap, Method R |
| 10:30–11:00 | **Break** |
| 11:00–12:00 | **Parallel Sessions #1** |
| | **Auditorium:** *Thinking Clearly About Performance*—Cary Millsap, Method R |
| | **Tassajara:** *MySQL in Production—A War Story*—Chris Schneider, Ning **Editor's Pick** |
| | **Diablo:** *Get Proactive with DBA Task Automation*— Jagan Athreya, Oracle |
| 12:00–1:00 p.m. | **Lunch** |
| 1:00–2:00 | **Parallel Sessions #2** |
| | **Auditorium:** *Thinking Clearly About Performance (Contd.)*—Cary Millsap, Method R |
| | **Tassajara:** *Best Practices for Upgrading to Oracle Database 11*g—Mark Townsend, Oracle |
| | **Diablo:** *NoSQL Deep Dive*—Gwen Shapira, Pythian |
| 2:00–2:30 | **Break and Refreshments** |
| 2:30–3:30 | **Parallel Sessions #3** |
| | **Auditorium:** *Maximize Your Oracle Investment with Enabling Technologies from EMC and VMware*—Jason Kotsaftis, EMC |
| | **Tassajara:** *Extreme Performance Data Warehousing*—Hermann Baer, Oracle |
| | **Diablo:** *How DBAs Can Help Success of a BI Project*—Sunil Ranka, Trianz |
| 3:30–4:00 | **Raffle** |
| 4:00–5:00 | **Parallel Sessions #4** |
| | **Auditorium:** *Database Performance Made Easy Through Graphics*—Kyle Hailey, Delphix |
| | **Tassajara:** *Don't Be in a Funk—Use Analytic Functions*—Philip Rice, UCSC |
| | **Diablo:** *Oracle Application Express 4.0*—Raj Mattamal, Niantic Systems |
| 5:00– | **Networking and No-Host Happy Hour at Faz Restaurant, 5121 Hopyard Road, Pleasanton, CA** |

**RSVP online at http://www.nocoug.org/rsvp.html**

The *NoCOUG Journal* design and production: Giraffex, Inc., S.F.
Front cover photo: Grand Canyon Mule Ride by Alan Kyte.