Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 20, No. 2 · MAY 2006 $15

Add Color to Your Career with NoCOUG

Down Memory Lane

An interview with Bill Schwimmer. *See page 4.*

Is 24x7 a Myth?

Six well-known Oracles answer our questions. *See page 7.*

Hello, World!

A new column on programming languages. Guest columnist Krishna Kakatur introduces us to SQLJ. *See page 19.*

Big discount on Steven Feuerstein seminar! ***See page 3.*** ***Much more inside . . .***

Em Dashes and En Dashes!

This is my second time as editor and I am slowly getting the hang of things. I now know the difference between a hyphen, an em dash, and an en dash! Learn about it at **www.getitwriteonline.com/archive/091502.htm**.

Thanks to all the contributors who pour their energies into writing for the journal. A new column called *Ask the Oracles* makes its debut in this issue. Is 24x7 a myth? Ask the Oracles! *Thanks once again to Tom Wagner for another beautiful picture of Mount Timpanogos in Utah.* Special thanks must go to Brian Hitchcock for another outstanding book review. Brian is one of the mainstays of the *Journal.* His reviews are unusually insightful, which reveals tremendous competence in all things Oracle, and they contain a trace of wit that really sets them apart. But the best part, for me, is that he finishes them three months in advance!

I would love to hear your comments about the *Journal* and your suggestions for improving it. Please drop off a note at **journal@nocoug.org** or let's talk at the next conference, on May 18, at Lockheed Martin in Sunnyvale.

—Iggy Fernandez, *NoCOUG Journal* Editor

Table of Contents

Publication and Submission Format

The *NoCOUG Journal* is published four times a year by the Northern California Oracle Users Group approximately two weeks prior to the quarterly regional meetings. Please send your questions, feedback, and submissions to the *NoCOUG Journal* editor at **journal@nocoug.org**.

The submission deadline for the upcoming August 2006 issue is June 1, 2006. Article submissions should be made in electronic format via email if possible. Word documents are preferred.

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

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

The Power and Value of NoCOUG

by Darrin Swan

Darrin Swan in Egypt.

My wife Stephanie and I just returned from a two-week vacation to Egypt and Jordan. It was the experience of a lifetime and one that helped me appreciate the opportunities that our society offers and the advancements in technology that we enjoy. Seeing firsthand the achievements of a society many thousands of years old put into perspective how significantly our work in the realm of Information Technology contributes to humankind. It made me wonder how our contributions might be looked at thousands of years from now.

As Oracle professionals, we are at the frontier of connecting disparate information—which binds cultures and societies—with modern forms of communications and ways of including everyone in the global union of common understanding. As I spoke with a few Egyptians about the state of technology in their country, a general theme of modernization surfaced. *Advancements happen one individual at a time and, collectively, the unimaginable is achieved.*

Since my primary career and personal interests are in the realm of Information Technology, during my time away from the daily grind, I thirsted for technical contact with an Egyptian with a like appreciation. On my second day of my Nile cruise and fifth day in Egypt, we had already maxed out our two CompactFlash cards totaling 5 gigabytes worth of images. My highest priority was to find options to transfer the images somewhere else in order to clear out the memory cards, which gave me the icebreaker to find a technically minded individual to converse with.

My first idea was to transfer the images to an online image gallery using the ship's PC. With only a 56k wireless Internet connection (Egypt's standard, because higher speeds interrupt military communications), uploading photos with an average image size of 3MB was clearly not an option. During one of our port stops, I did successfully run into an Internet café before we set sail again and successfully burned one CD of 700MB, but only scratched the surface of my task.

That night I discussed my little issue with the ship's captain and he put me in touch with the staff Egyptologist, who was regarded as the most technical person on the ship. After speaking with the Egyptologist, Hani, my storage challenge was solved and I made the technical human contact I was looking for.

I came to find that in addition to being an Egyptologist, Hani was involved with an Egyptian technical experts community, advising on building a better communications infrastructure for the growing Egyptian technology business sector. My discussion with the Egyptologist was very much like the exchanges I notice taking place during NoCOUG conferences. A technical challenge was presented to a like-experienced professional, options to solving the problem were discussed, and a solution to the problem was found. We reviewed the problem logically and discussed a variety of solutions until we found the optimal one.

The exchange and discussion of my technical challenge with a like-minded individual was invaluable to identifying a solution. The end result was to simply delete songs off my iPod and use it as a storage device. Successful results are mostly derived from sharing and brainstorming with someone who has experienced similar challenges and with someone who can equally appreciate the theater in which your challenge exists. This statement brings me to the power and value of NoCOUG.

As members of the elite community of NoCOUG, you have access to hundreds of peers who are dealing with, or have successfully solved, many of the challenges that you are faced with on a daily basis. Remember, advances in Information Technology happen one individual at a time and, collectively, the unimaginable is achieved.

Come find the power and value of NoCOUG! I'll see you on May 18 at Lockheed Martin in Sunnyvale. ▲

Big Discount on Steve Feuerstein seminar

NoCOUG and Speak-Tech join to bring you Steven Feuerstein's full-day seminar "21st Century PL/SQL" on Friday, August 18, 2006, at the San Ramon Valley Conference Center in San Ramon.

Steven Feuerstein is one of the world's leading experts on PL/SQL, having written nine books on the subject including *Oracle PL/SQL Programming* and *Oracle PL/SQL Best Practices*.

NoCOUG members get $50 off the regular price of $300, which includes meals. Go to **www.speak-tech.com/seminars/feuerstein_announce_aug06.html** for more details and to register. ▲

Down Memory Lane

An Interview with Bill Schwimmer

So it is the laughter
We will remember
Whenever we remember
The way we were

Bill Schwimmer

We welcome Bill Schwimmer, the first database administrator at Apple Computer and now a DBA manager at EDS. First he leads us down memory lane, then he peeks into his crystal ball and reads our futures.

Tell us something about yourself. Were you really one of the first database administrators at Apple Computer? How did your career progress?

I graduated from UC Berkeley, worked for the state of California as a developer, and was hired at Apple Computer in 1983 as a DBA. At that time Apple had one DBA who was hired about a month before me, and one right after me. But they both left and for a period of about five years, I was Apple's only DBA. In fact, when it came time for my sabbatical (every employee got a six-week paid vacation every five years), I had to hire and train a DBA to take over for me before I could go. Luckily, I found a top-notch DBA, Terry Couey, who was able to jump in and take over, and I still work with Terry—18 years later.

Apple's DBA function was chaotic and primitive in 1983, but it matured during the time I was at Apple, and when I left there were somewhere around 30 DBAs. Back in 1983 Apple had just installed and become an early adopter of Cullinet Applications Software—one of the first integrated ERP systems. It ran on IDMS—a network DBMS—while relational technology was still in its infancy. I worked at Apple for 15 years, progressing from DBA to DBA manager, and in 1996 Apple outsourced much of its IT to a Canadian company, Systemhouse. This was the period of acquisition and merger followed by the dot-com days, so things were changing fast. I worked for Systemhouse, MCI, WorldCom, and EDS during a period of five years—all due to buyouts and mergers. I'm still a DBA manager, but I've recently been managing teams of Oracle, Oracle E-Business Suite, and SQL Server DBAs for EDS, which has hundreds of DBAs around the globe supporting thousands of clients and tens of thousands of systems.

He called the entire IT department "bozos" for not moving to Unix.

There have been books and movies about the early days of Apple. You were there yourself. Tell us an interesting story about the early days.

I worked in the Apple II division when I first started. The Lisa, one of the first computers with a GUI, had just come out and the Macintosh was under development, not to be unveiled to the world until January 1984. All of the prestige and glamour was in the Mac division; they were flying a pirate flag over their building and covertly developing a product that Apple felt would change the world. They had all of the perks—a piano in the lobby, free massages, and free food. This created a bit of jealousy because the Apple II division was the cash cow that produced most of the revenue. One day Steve Jobs spoke in a meeting to the Apple II IT group, which was probably under 100 people at that time. Steve chastised the entire group, particularly management, for running the division's financial and manufacturing systems on IBM mainframes, IBM AS/400s, and DEC PDP 1170s. He was exasperated with IT and called the entire department "bozos" for not realizing that the world of IT—data processing, in those days—was moving to Unix and "C." Clearly he was way ahead of his time—in fact he was so far ahead that he neglected to realize that there were virtually no commercial software packages available at the time that ran on Unix—certainly no ERP systems—and of course it made no sense to build an ERP system from the ground up.

We like stories! Tell us another story!

Well, you'd think Apple would have been on the cutting edge of database technology, but in the early '80s Apple was not interested in relational. Few companies were at that time. As Apple's DBAs, we took it upon ourselves to evangelize relational technology—now it's called "thought leadership"—to try to move the company to an SQL-based DBMS as the corporate standard replacing the older technology. We promoted relational by bringing in Chris Date,

the father of relational (along with Ted Codd) to speak to Apple's engineers and managers about the relational model, SQL, and how they could provide the foundation for Apple's corporate systems. Speaking to a techie crowd in Cupertino, Chris's academic yet lighthearted speaking style and his technical credentials (Oxford U and IBM R&D) helped sway the movers and shakers to try relational technology—at least as a pilot project. But the commercial vendors were fairly new, the DBMS shakeout had not yet occurred, and no one knew which platform provided the best technology and ROI or which companies would survive. So chaos ensued: First we standardized on DB2 since IBM seemed to be a safe bet—even though it was a bit embarrassing for Apple; then we switched to Sybase, then Ingres, compromised with a dual standard of DB2 and Ingres, and then ultimately converted all of Apple's databases to Oracle/Unix. So, Steve Jobs's declaration of his preferred platform—Unix—over a decade earlier finally came to pass. And Apple was way ahead of most companies, standardizing on Oracle/Unix in the early '90s.

> *Certification is important but it's not enough.*

Whenever I go for an interview, the manager says, "Tell us about a crisis and how you handled it." May I turn the tables on you? You've been a manager for many years and have seen your share of crises and disasters. Tell us about a really big crisis that you and your team handled. What was the moral of the story?

I've had my share of sleepless nights through the years, but I try to block out bad memories of disasters and crises. You conduct a postmortem, learn the lessons, make the necessary changes to avoid future problems, and move on. Don't dwell on the mistakes but don't repeat them, either.

Has the practice of database administration changed over the years or has it remained essentially the same? Is there a qualitative difference or simply a quantitative difference (more and bigger databases)?

DBA work has changed quite a bit. In the pre-relational days, DBAs had to do a lot more low-level work—they were closer to the machine, so to speak. Experienced DBAs learned to read hexadecimal dumps of the internal DBMS code to debug problems. Some knew assembler language; nearly all had been programmers. I don't think there's much dump reading going on these days. As DBMS technology has matured, the work of the DBA hasn't gotten any easier, but the work has definitely changed so that DBAs are much more productive in terms of designing and building databases as well as maintaining and troubleshooting them. And as the technology has improved with sophisticated tools and self-healing features, I've seen the number of wake-up calls for problems decline and databases per DBA increase dramatically. And tools like Oracle Designer, Oracle Warehouse Builder, and OEM have had a tremendous impact on DBA productivity in the design and tuning stages.

How would you define the role of database administrator? Is there room for application administration, data administration, and software development in your definition?

There's no universally accepted definition of the role of the DBA—the boundaries are different depending on the size of the enterprise and its organizational structure. In the largest companies, the tendency seems to be toward more specialization. At EDS, we've split the DBA responsibilities between application/development DBAs and system/infrastructure DBAs. This forces a more disciplined and consistent approach because DBAs focus on their areas of expertise and a different set of DBAs implements changes and installs software. Sarbanes-Oxley Act compliance has been a major driver in splitting responsibilities. There are certainly pros and cons to both approaches—having DBAs do it all or splitting the role.

What qualities do you look for in a database administrator? Is certification important? I applied for a government job once and was shocked that the job description did not reference specific technologies. Government seems to work differently than the private sector. Would you hire somebody with many years of database administration experience but none with Oracle specifically? Would you hire somebody to work on Oracle 10g if they only have Oracle 8i experience? Couldn't you train them on the job?

> *High-availability solutions do work as advertised once set up properly, but they also raise the bar of complexity to a new level.*

Yes, certification is important but it's not enough. I look for years of experience in the technology that I'm hiring for. I also look for an indication that the DBA can work well within a team. You can have all of the experience and an impeccable track record while working alone on projects, but if you haven't proven that you can collaborate in a team, I'm not interested. I don't think that it's necessary to have expertise in the latest version because that can be picked up on the job, and obviously when a new version is first released, no one has experience with it.

What are some of the challenges that you have experienced with high-availability solutions? Do they really work as advertised? Is the extra complexity compensated by the increased availability?

They do work as advertised once set up properly, but they also raise the bar of complexity to a new level. My experience has been that RAC has been challenging for many DBAs and it has been slow to be adopted by Oracle's customers, but now it seems to be taking off—especially with 10*g*. You can't put a price tag on availability—the business has to determine what the cost of downtime is and the ROI of the various HA options.

I'll advise my kids to choose a career that requires them to touch something onshore.

The technology vendors are always creating new patches and upgrades. Is it feasible or even desirable to keep systems fully patched? The Sarbanes-Oxley auditors certainly seem to think that it is a good idea to keep systems fully patched.

Absolutely. It's very important to keep mission-critical systems patched to the latest levels and Oracle has made this simpler with their quarterly "critical patch updates." We've been notifying all of our business partners that they need to give us the time to apply the patches or accept the risk of not doing so, and we've been successful in most cases in keeping the systems up-to-date. And you're right—SOX auditors will not be at all happy with patching strategies that are lacking. In the old days I subscribed to the "ain't broke, don't fix" school of thought and advocated upgrades and patches only when necessary, but that is no longer a sound strategy given the myriad security vulnerabilities.

I think offshoring will continue and probably accelerate.

Let's pick on the whole Sarbanes-Oxley thing. The legislation does not dictate specific dos and don'ts for IT. The auditors seem to be making it up as they go along; e.g., they want systems to be fully patched and they want us to stop using generic accounts such as oracle, SYS, and SYSTEM. What value do you see in Sarbanes-Oxley audits? How can we make the auditors see the light?

I think it's irrelevant whether or not we see value. And have you ever tried to make an auditor "see the light"? Not gonna happen . . . that's why they wear those dark green eye shades. Regardless of whether there is value or not, SOX is the law and we will be required to meet the auditors' criteria or fail the audit. You're right; they're making it up as they go because Sarbanes-Oxley doesn't specify what IT must do to comply. One of the main goals of it is to improve companies' internal control over financial reporting. It's pretty apparent that processes supporting IT need to be controlled so that applications are not exposed to failures from events like human error, sabotage, and changes. So having IT processes under control helps us to be ready for a SOX audit, but SOX compliance should be built into normal day-to-day operations as a standard best practice, not something done in addition to regular work just to comply with an audit.

For a long time, Larry Ellison has been predicting that database administrators will become extinct. His words seem to be finally coming true, at least in the case of America, because all the jobs are going to India and Elbonia. Is there any hope that the rate will slow down? What has been your experience working with offshore personnel? What are the challenges? What are the advantages?

You hear a few stories about "backshoring" [bringing work back to the U.S.] but I don't think it will become a real trend unless U.S. tax laws change and it becomes disadvantageous for American corporations to go offshore for lower-cost labor. I think offshoring will continue and probably accelerate. The skill level of the talent pool of offshore DBAs is definitely improving and it is growing exponentially in numbers. And as salaries in India increase, U.S. companies will look to China and Eastern Europe to find even cheaper labor. There's no end to it as far as I can see. We've worked in collaborative teams with offshore DBAs for years and, although there were initially problems in communication and connectivity, now they're considered just another part of the team. Clearly the biggest advantage to American DBAs is that in a scenario like I described earlier with a team approach, DBAs on both sides of the globe get to sleep through the night. The disadvantage—job loss—is obvious.

Some say that Apple's technology is far superior to that from Microsoft. Yet Microsoft has a commanding market share. How does Oracle's technology compare with others? What is the reason for its commanding market share?

A combination of great marketing and great technology.

The book* Who Moved My Cheese? *is a compelling story about people who cannot handle change. With all the jobs going to India, we can either pound on the walls or put on our running shoes and go looking for new opportunities. I know veteran IT professionals who own health food stores, sell teddy bears on the Web, sell insurance, and patrol the streets of Berkeley on bicycles as cops. What's your advice for IT professionals? Should we give up and find other professions?

Tough question. Depending on what survey you believe, U.S. IT jobs may be either expanding or contracting due to offshoring. I believe that as long as we don't see a recession in the U.S., there will be an expansion of IT needs, and I think that highly trained IT professionals—especially data engineering types—are probably going to be in demand in the U.S. for years to come. But to be safe, I think I'd advise my kids to choose a career that requires them to physically touch something onshore! ▲

Bill Schwimmer can be reached at **bill.schwimmer@ eds.com** *or* **bill.schwimmer@gmail.com**.

Is 24x7 a Myth?

Ask the Oracles!

Gaja Krishna Vaidyanatha: Lose weight instantly with this magical pill . . . no exercise or diet required! Transform your life completely within 12 minutes with this "self-help" video! Become a millionaire in less than 30 days without leaving your home . . . and more. Our day-to-day lives are filled with many such unbelievable claims. The media does a great job of propagating this.

If HA is defined by 99.999% uptime on any given year, the "available downtime" both for unscheduled and scheduled events is a paltry 5.2596 minutes. That is all the time that you have for all of your system maintenance including routine database administration, and operating system and Oracle software patching. I don't think I have to even attempt to convince you that this uptime goal is impossible and downright crazy for most systems!

There is a much higher probability that the DBAs supporting these systems will scale Mt. Everest without oxygen.

No matter how well you have automated your environment, using only 5¼ minutes of downtime a year to perform a plethora of required system maintenance and database administration activities in most systems (especially running Oracle databases) is a lofty goal. For most systems out there, there is a much higher probability that the DBAs supporting those systems will scale Mt. Everest without oxygen than there is of the system achieving 99.999% uptime! Remember the last time you applied an Oracle patch? How long did it take? Did everything go as planned? How long was your application (system) unavailable? Remember the last time a simple "zero risk" effort on your database caused downtime? Need I say more?

P.S. High availability in an Oracle environment is almost always equated to Oracle RAC. If you are thinking of achieving HA with Oracle RAC and you have not designed your application for a clustered environment . . . think again! Slapping Oracle RAC into your environment without expending the necessary time and effort to design your application is a guaranteed recipe for disaster . . . at least in the realm of performance management.

Gaja Krishna Vaidyanatha has over 13 years of industry technical expertise working with Oracle systems and providing strategic and technical direction to companies such as Veritas Corporation, Oracle Corporation, and Quest Software. He is the primary author of Oracle Performance Tuning 101, *published by Oracle Press, and one of the co-authors of* Oracle Insights: Tales of the Oak Table. *Gaja is a member of the Oak Table Network and can be reached at* **gaja@dbperfman.com**.

Mogens Norgaard: I have known of many systems running VMS or Linux or Unix with Oracle on them that would run for hundreds of days without interference and without problems.

But when I look back, I think those systems that just keep running day in and day out are either very simple or very, very complex (guess what happened to costs, too).

The very simple systems, where you just have a stable box with some standard Oracle installed, and where you don't fiddle all the time with parameters, setup, etc., are very often the ones that will keep running for close to a year at a time—and in rare cases even longer.

The very simple systems are very often the ones that will keep running for close to a year at a time.

The very complex ones where you have mirrored SANs, Data Guard, RAC, and iAS installations with all high-availability features enabled (in other words Oracle's High Availability Architecture and more), and redundant power supplies and all that, can also be made to work for prolonged

periods, but chances are smaller due to the higher stack and the sheer complexity of all those products and options interacting with each other.

Then there are the occasional bugs, for instance, the one that caused Oracle 8 to shut down after 248 days due to an internal counter. Refer to Metalink note 118228.1.

Do most businesses need 24x7? Probably not. When we had a huge power outage on the island of Zealand last year, with most of Copenhagen and its surroundings blacked out, no businesses had to close as far as I know.

But when the MBA boys from the huge consultancies tell you that a few minutes downtime will cost you $42 billion, the conclusion is obvious: You must invest heavily (really, you cannot invest too much!) in so-called Business Continuity, as they call high availability these days.

Nah, I don't think so.

Buy a standard box with standard CPUs in it. Make sure it's produced in the same factory as your home computer, so you know it has zero defects. Then take a good backup and test it now and then.

Use the Disposable Computing Architecture (James Morle invented that term), which means you should have another standard box standing ready (powered off to save energy) in the corner.

Invest in a couple of NAS boxes, or even better, attach disks directly, and you're ready and open and highly available with simple and cheap (and therefore efficient and easy-to-understand) technology.

---

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

Jeremiah Wilton: *For the individual database, yes. For a service sufficiently abstracted from the database, no.*

High availability is no myth. But while you can take many steps to create a stable database, true availability is measured from the end-user's perspective, several layers removed from the database. High-availability solutions at the database level, such as RAC and Data Guard, while potentially important tools in creating stability for a given database, are not the starting point of any larger software-based high-availability solution.

The best availability in the industry comes from application software that is predicated upon a surprising assumption: *The databases upon which the software relies will inevitably fail.* The better the software's ability to continue operating in such a situation, the higher the overall service's availability will be.

But isn't Oracle unbreakable? At the database level, regardless of the measures taken to improve availability, outages will occur from time to time. An outage may be from a required upgrade or a bug. Knowing this, if you engineer application software to handle this eventuality, then a database outage will have less or no impact on end users.

One way of engineering applications to maximize availability in a down-database situation is through availability testing. Imagine an ordering system, with two databases—one for shopping carts and one for orders. If the application software were to time out or encounter a fatal error accessing the database, it would write to a common piece of local state, such as a shared memory segment, with a flag indicating a problem on the target database. Then other threads of the application, before attempting to make a call against the database, would check this state object and forego the database call in order to avoid encountering the same error.

The best availability in the industry comes from application software that is predicated upon a surprising assumption.

In such an example, failure of the ordering database would not cause hanging or termination of the application thread, and normal use of the shopping cart could continue. Conversely, if the shopping cart database were to become unavailable, then those users already in the order pipeline would be able to complete their orders without the application failing.

Another way to avoid failure-prone database access is to employ caching. An application should not keep going back to the database for the same data over and over again. For instance, a well-engineered application might only query a currency exchange rate once an hour. It would then maintain that data in memory and use it to service many requests, instead of repeatedly and inefficiently querying the database. Beyond improving availability, caching has the added benefit of reducing load on the database.

In summary, there are many ways to improve a single database's availability. But the highest availability comes from thoughtful engineering of the entire application architecture.

Jeremiah Wilton was the first DBA at Amazon.com. He now runs ORA-600 Consulting in Seattle, where he specializes in high availability, complicated recoveries, and seminars. Jeremiah's full white paper on high availability is available at **www.ora-600.net**.

Tim Gorman: 24x7 (a.k.a. 100% uptime) is not a myth. There is a production system running Oracle Rdb on DEC/Compaq/HP OpenVMS at the HP hosting center in Colorado Springs, which has been available continuously since 1992 or so, through hardware and software upgrades. So, 100% uptime is not widespread, but one or two examples do exist on this planet.

24x7 might be considered a myth for the Oracle RDBMS, at least at the present time. Oracle RAC is primarily designed to be a scalability solution that (of necessity) incorporates certain features that are construed (and marketed) to be "high availability," such as automatic instance failure detection and recovery. But the presence of some important redundancies does not overcome the fact that there remain important single points of failure within RAC, so it is not a high-availability solution by design. Oracle Data Guard is a solution truly designed for high availability and the elimination of single points of failure, but I don't know if the product has been around long enough to bear out the designation of 24x7 over any appreciable period of time. This is something that time will tell.

> ***There remain important single points of failure within RAC.***

*Tim Gorman has worked with databases since 1984 and with the Oracle RDBMS since 1990. He is currently with SageLogix (***www.sagelogix.com***), a small group of consultants based near Denver, Colorado. He is an active member and past president of the Rocky Mountain Oracle Users Group (***www.rmoug.org***). Tim can be reached at* **tim@sagelogix.com**.

James Koopman: Is a 24x7 database system a myth? Let me just say for the record, YES it is. At the root of the question is the definition of availability, which has the connotations of being ready for use, at hand, and accessible. With 24x7, that means 100% availability and completely accessible.

Now get the notion of a full database up and running out of your head. You see, a database is composed of many pieces. There are the Oracle processes, networking components such as TNS, tables, indexes, and procedural code. If any of these pieces is unavailable, does that mean our database system is unavailable? I tend to think YES. We have all administered databases that required us to fix some piece of data, rebuild an index, or restart our TNS listeners. Is this downtime? I tend to think YES. Downtime by definition is a window of opportunity taken by administrators in order to fix something, and directly affects availability.

You see, 24x7 is not just a hardware issue but in reality is a user issue. Even the best DBAs, if honest, will tell you countless stories about how they have snuck in some form of maintenance that they are proud of because it went undetected by an end user. Outages will always occur in database systems. It might not be today, it might not be tomorrow, someone may notice it, and most may not. But the fact is, it is impossible for us to have a truly 24x7 system when we take a look at availability from individual usage patterns, administrative tasks, and the end-user perspective.

> ***We have all administered databases that required us to fix some piece of data, rebuild an index, or restart TNS listeners.***

*James Koopman is founder of Pine Horse, Inc. (***www.pinehorse.com***), and creator of* **www.dbcorral.com**. *Over the years James has worked with a variety of database-centric software and tools vendors as strategist, architect, DBA, and performance expert. He is an accomplished author with articles appearing in publications including* Oracle Professional, Database Trends and Applications, DatabaseJournal.com, *and various vendor newsletters. James can be contacted at* **jkoopmann@pinehorse.com**.

Chris Lawson: A goal of 24x7 is fine for critical information systems, but I believe many businesses are fooling themselves when they claim that they are meeting this goal. Ironically, the problems with most high-availability solutions aren't really due to flaws in the technology, but how they are implemented. In particular, I have noticed that "hot standby" systems are especially weak links. In my experience, most high-availability systems fail for these reasons:

Critical components are missed: In order for a failover solution to work, every part must failover properly (or have some type of redundancy). If any piece is missed, you don't really have high availability. Unlike school, a score of 99% correct isn't good enough—it gets an F just like the system that is only 1% correct. When your online system is down and customers are mad, management doesn't really care that you got it almost right. At one large insurance company, a critical server failed. As planned, the database failed over just fine, and was 100% usable. Unfortunately, a major part of the application didn't, because the designers neglected to include it in the process.

Incomplete testing of the failover process: In the above case, a simple failover simulation would have immediately uncovered the deficiency. Incredibly, technical management assumed everything would work fine—without testing. In another case I witnessed, a failover didn't work because filesystems had been modified on one node (the current node), but not on the failover node. This flaw was only

How do we design a system that provides this assurance: "People will never make a mistake"?

uncovered when the failover didn't work.

Overlooking the likelihood of human error: By this I mean blunders made after the high-availability system is implemented. I think this is the toughest problem to solve. The key focus in high-availability solutions is usually reliability of the hardware. In reality, however, failures are often the result of human error. Several years ago, I helped set up a Veritas Cluster Server high-availability solution. We tested everything many times. Failover worked flawlessly. A few months later, however, a real outage occurred, and the failover didn't work! Investigation showed that the system administrator had recently deleted a directory that he thought wasn't needed. Of course, the directory contained files vital to VCS. The most obvious answer to this problem is to add strict configuration control, or some form of double-checking to the process. At one firm, a project manager, frustrated by multiple DBA mistakes, briefly implemented a policy of having one DBA look over the shoulder of another DBA while he typed in commands.

Unlike school, a score of 99% correct isn't good enough—it gets an F just like the system that is only 1% correct.

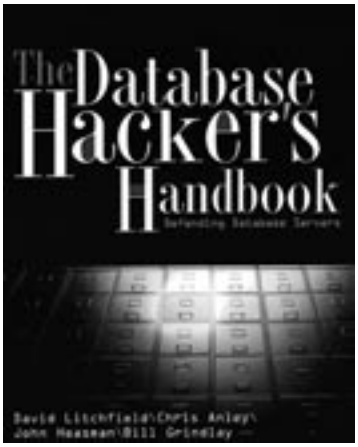
In summary, true 100% system uptime will likely remain a tough goal, but proper planning and testing can increase the uptime dramatically. To really achieve 24x7 however, we need to focus on the root cause of most system outages. The really difficulty problem is not how to increase hardware reliability, but how to increase human reliability. How do we design a system that provides this assurance: "People will never make a mistake"?

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

Database Hacker's Handbook

A Book Review by Brian Hitchcock

Summary



Overall review: Excellent; a book you must read.

Target audience: Anyone responsible for the security of corporate data.

Would you recommend this book to others? Without question.

Who will get the most from this book? Anyone with basic RDBMS experience.

Is this book platform specific? No.

Why did I obtain this book? I saw it on Amazon and wanted to read it. NoCOUG reimbursed me for the cost of the book.

Overall review

There are many books published about RDBMS technologies that you really don't need to know about. This isn't one of those books. You must read this book, or at least the first five chapters, which cover security issues in Oracle databases.

The authors do throw in some things that they don't explain but this is rare. I found myself wanting to know what a "0 day exploit" is, and I found out with a simple Internet search. (It means that someone takes advantage of a security vulnerability on the same day that the vulnerability becomes generally known.)

A reasonable person might question the morality of publishing this information. The reality is that the "bad guys" know all this and so should you. When you have an idea how the hacking happens, you have to deal with the magnitude of the problem and just how poorly protected your systems really are.

Chapter 1: Why Care About Database Security?

I think it is an excellent point that it doesn't really matter which vendor's database you use, it all really boils down to the specific implementation of your database in your environment. It is also excellent that they break down all of the database security flaws into eight groups.

Unauthenticated flaws in network protocols—These are issues in which a hacker can gain access to your database simply by being on the same network. The best protection for this is up-to-date patching.

Authenticated flaws in network protocols—Here the hacker has been able to connect to your database through normal means before the hacking begins. The recommended solution for this class of problems is to have a strong password policy.

Flaws in authentication protocols—The hacker uses a database user password that is detected as it is transmitted in plain text or by other means, and again the recommended solution is to be up-to-date with your patches.

Unauthenticated access to functionality—Features within the database can be executed by a database user who isn't supposed to be able to execute to them. The solution for this class of problems is to use a listener password.

Arbitrary code execution in intrinsic SQL elements—These issues involve causing buffer overflows simply through using SQL statements in unusual ways, and the recommended defense is to be up-to-date with patches.

Arbitrary code execution in securable SQL elements—These problems are caused by functions within Oracle that can be executed in specific ways to cause buffer overflows that allow a hacker to gain access. Patching is the best defense.

Privilege elevation via SQL injection—Here the issue is how to execute PL/SQL stored procedures in specific ways to get privileges that the database user shouldn't have. Patching is the best defense.

Local privilege elevation issues—The hacker executes functions within the database to go out and execute operating system utilities. The recommended solution for the user is to run your database as a low-privilege operating system user. I'm not clear how to do this since I always install Oracle as Unix user oracle.

There are many books published about RDBMS technologies that you really don't need to know about. This isn't one of those books.

Chapter 2: The Oracle Architecture

The vulnerabilities of Oracle Enterprise Manager are explained, among other things. This is a great example of

why I wanted to read this book and why I want you to read it as well. This chapter presents a new and very different view of Oracle. I would not have thought of Oracle Enterprise Manager as a security problem.

Simply from reading this chapter, I believe this book has the most information per page of any Oracle book I've ever seen.

The section on Oracle auditing is fascinating. The authors point out that if database auditing is on, any ALTER USER statement that changes the password is logged in plain text. Other things I learned in this chapter are that Oracle stores passwords in uppercase and this makes it easier for a brute force attack on the SYS.USER$ table.

Chapter 3: Attacking Oracle

The discussion of PL/SQL invoker rights, wrappers, and injection is very interesting, and very valuable. Simply from reading this chapter, I believe this book has the most information per page of any Oracle book I've ever seen. There are several sections in this chapter for which my comment is simply WOW! For example, the explanation of how a hacker can execute a procedure that can assume or inherit the privileges of the SYS user and how to grant DBA_ROLE to PUBLIC using autonomous transactions is just great. How easy it is to use DBMS_SQL also rates a WOW!

The authors explained in detail what conditions are necessary for an anonymous PL/SQL block to be used by a hacker; once that is understood, you can start looking through all of the PL/SQL blocks in Oracle to find others that meet the same conditions and can be exploited in the same way. This also illustrates how software could (and should?) be written to prevent many security issues in the first place.

Now I understand how hackers can find security issues. Once they know what conditions are necessary for them to exploit a piece of PL/SQL code, they can download and install Oracle on their own system and spend as much time as it takes looking for vulnerable pieces of code. This also illustrates why even the Oracle user SCOTT can be exploited. Once you know how to exploit pieces of code supplied by Oracle, even an Oracle database account as innocent as SCOTT can be used to break in. I used to wonder how an account like this could be a problem.

Chapter 4: Oracle: Moving Further into the Network

This chapter explains how the Oracle RDBMS itself can be seen as a "shell" because it can be used to execute operating system functions. In this manner a hacker can access the entire filesystem of the server machine, execute operating system commands, and access the network as well. A specific example explains how the Oracle listener was used to execute operating system functions as the operating system user oracle. One wonders why Oracle didn't cover all these issues at once. Surely these issues are known to those who wrote the software. This example doesn't give you a good feeling about the quality of commercially available software.

Chapter 5: Securing Oracle

While the previous chapters have been very educational and almost entertaining, this chapter attempts to tackle the really important and most difficult issue of all. Once you appreciate the depth, breadth, and complexity of the security issues, what exactly do you do about them?

Oracle is hard to secure because it is so big and so complex. A reasonable person would say that this argues for using simpler RDBMS software. Take MySQL for example. A simpler RDBMS to be sure, but what is the MySQL vendor doing? They're adding "features" as fast as they can, which adds to the size and complexity of their product. And why are they doing this? To meet market demands! Perhaps the enemy is us?

Perhaps the enemy is us?

The authors offer a list of recommendations on how to improve the security of our Oracle installation. While all of them make sense, and all of them have probably been heard before, I'm not convinced that most of them are really practical. Encrypting network traffic is a good idea, but our license agreement didn't include that specific (and expensive) Oracle functionality. The authors recommend that new account creation should be approved by a security officer. My group supports about 2000 databases; how many security officers should we hire? They also recommend that we use a strong password policy. Again I agree completely, but tell users of my mission critical systems that every 30 days one or more of the databases supporting their systems will lock out the database user that the application server uses to connect to the database. And, of course, I would need to maintain an up-to-date list of all the applications that might access all of the databases in my environment, so I can make sure they'll have up-to-date passwords. And none of the application code will have hard-coded passwords, correct? Auditing is certainly a good idea, but who will review the audit trails for all of the databases in my environment? We are told that we should check all PL/SQL modules to make sure that they include AUTHID CURRENT_USER, for example. Can you imagine reviewing all of the modules in Oracle Applications? We will also need someone to create and maintain a secure PL/SQL coding standard and do code reviews.

All of this ignores the threat from within.

The hard part in all of these recommendations is balancing the business need for security against the cost of the security while taking into account the benefits of the complex database features that you want for your business. I would have found this chapter much more valuable if it told me how all these suggestions were put in place for a large IT organization, and how they evaluated the costs versus the benefits. It's much easier to make these recommendations than it is to implement them.

The hard part in all of these recommendations is balancing the business need for security against the cost of the security.

Chapters 6–26: DB2, Informix, Sybase, MySQL, SQL Server, PostgreSQL

These chapters cover the same issues for each of these vendors' RDBMS products. I did read them all and I think they are worth your time, if only to see just how differently each of these products is architected. Since this is primarily a review for users of Oracle, I didn't review these chapters.

Conclusion

The real issue with security is balancing the cost of implementing improved security against the cost of a successful break-in. We can all agree that more security is better, but at what cost? And how do you determine the cost to your business of a security breach?

The complexity of Oracle invites security issues, but your business benefits from the advanced features available. Simpler software may be more secure, but then we want more features. More features (complexity) breed more security flaws, which require more patching, and the patching becomes more complex—what's wrong with this picture?

And, no matter how technically proficient we think we are, all of this ignores the threat from within. If those who are approved for full access decide to embark on their own nefarious schemes, none of what is discussed here will matter. The internal threats are much harder to deal with and this book doesn't offer any advice on these issues.

About the Authors

All are members of Next Generation Security Software Ltd.

David Litchfield holds the unofficial world record for finding major security flaws, including a major breach in Oracle 9 database servers.

Chris Anley has published white papers and security advisories on SQL Server, Oracle, and other database systems.

John Heasman has authored security advisories on high-profile products including Microsoft Windows, RealPlayer, and Apple QuickTime.

Bill Grindlay has worked on a generalized vulnerability scanner as well as NGSSquirrel database security scanners. ▲

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

SQL Sucks! – Part I

by Iggy Fernandez

Iggy Fernandez

The other terror that scares us from self-trust is our consistency; a reverence for our past act or word, because the eyes of others have no other data for computing our orbit than our past acts, and we are loath to disappoint them. . . . Bring the past for judgment into the thousand-eyed present, and live ever in a new day. . . . A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines. . . . Speak what you think now in hard words, and tomorrow speak what tomorrow thinks in hard words again, though it contradict every thing you said today. **—Ralph Waldo Emerson**

Summary

In the first part of this multi-part article, we argue that query optimizers (including the Oracle query optimizer) are severely handicapped in their ability to generate good query execution plans and that, therefore, the promise of relational technology has gone largely unfulfilled. In the second part of this article, we will discuss the workarounds.

A Personal Note

In a previous issue of the *Journal*, I wrote a couple of articles extolling the power of SQL and received the following comment from Ravi Kulkarni of Phoenix, AZ.

"I think it is a stretch to say that the Oracle optimizer can optimize every possible SQL statement written using convoluted but severely limited syntax, the same way as an expert DBA with a procedural language."

Subsequent to the publication of those articles, I had occasion to participate in a major SQL tuning effort for a Fortune 500 company and encountered many cases in which the Oracle query optimizer was generating poor query execution plans. It appeared that Ravi was right!

Technical Background

SQL is what is called a "nonprocedural" language; i.e., an SQL query simply identifies a subset of data in the database without specifying how to go about extracting that data. The *promise* of relational technology was that the SQL programmer would be relieved from *most* of the responsibility for making the best use of computer resources and would only need to consider the "logical data model," not the "physical data model." The intention was that the major responsibility would be divided as follows:

1. First, the responsibility would devolve to the database engine, which would provide a selection of data processing algorithms (e.g., merge join, hash join, parallel processing, etc.), data storage methods (e.g., index-organized tables, table clusters, data partitioning), and indexing methods (e.g., b-tree indexes, bitmap indexes, function-based indexes).
2. Second, the responsibility would devolve to the database administrator who, in partnership with the application architect, would convert the "logical" database design into a "physical" database design and choose storage methods and indexing methods geared to the requirements of the application. The database administrator would also work with the users of the application to ensure that there was sufficient computer capacity (CPU, memory, I/O bandwidth, and network bandwidth) to meet the current and future needs.
3. Finally, the responsibility would devolve to a special component of the database engine called the "Query Optimizer," which would rely on statistical information such as row counts and histograms to make decisions about table processing order and index usage and to choose from among available data processing algorithms. The goal being to identify the "query execution plan" that minimizes the consumption of computer resources and the time required.

In this article, we will present certain defects of the query optimizer and argue that these defects are incurable. Note that we are not singling out Oracle in particular. The same defects exist in the technology of every other vendor.

Incurable Defects!

We really need to prepare the ground adequately with an overview of the inner workings of the query optimizer, but we only have space for the barest minimum of examples. Suppose that A is a "parent table" and B is a "child table" and that the optimizer is presented with a query of the following sort.

```
select * from A, B
 where <filtering criteria to be applied to A>
   and <filtering criteria to be applied to B>
   and <criteria to match records of A and B>
```

The optimizer can use either table A or table B as the "driving table" for the query. It will instruct the database engine first to filter the chosen table using the applicable filtering criteria and then to match the resulting record set with matching records from the other table using the

applicable matching criteria. It is obvious that the key to the choice of the driving table is the size of the record sets that result from applying the applicable filters to each table. For example, if the optimizer knew in advance that the filters on table B were very restrictive, then it would be wise to choose table B to be the driving table.

Now consider an example involving three tables.

```
select * from A, B, C
 where <filtering criteria to be applied to A>
   and <filtering criteria to be applied to B>
   and <filtering criteria to be applied to C>
   and <criteria to match records of A and B>
   and <criteria to match records of B and C>
   and <criteria to match records of A and C>
```

Since only two tables can be processed at a time, the optimizer must decide which two tables to process first, and the resulting record set must then be joined with the remaining table. For example, if the optimizer had some way of knowing that the matching criteria to be used to associate records of table B with those of table C were very restrictive, then it would be wise to process those tables first, so as to reduce the work required in finding matching records in table A. *Once again, we see that estimation of the number of qualifying records at each step of query processing is the key to the problem.*

The optimizer attempts to solve the problem by using statistical information such as row counts and histograms. A histogram is a summary of the data values in any one column of a table; e.g., we might have a histogram that tells us that California is the most populous state in the Union. However, histogram summaries for single columns are of no use when we are faced with more than one filter. Similarly, histograms offer no help in assessing how many rows in one table will match rows of another table.

Consider the following two examples.

```
select * from CarSales
 where Manufacturer = 'Toyota'
   and Model = 'Celica';

select * from CarSales
 where Manufacturer = 'Toyota'
   and ModelYear < 1975;
```

Now, the Oracle optimizer can use histograms to estimate the percentage of sales satisfying any single one of the criteria listed in the above SQL queries but has no way of accurately estimating what percentage of sales satisfy two or more criteria. To us it is obvious that all Celicas are Toyotas and that, therefore, the percentage of Celicas manufactured by Toyota equals the percentage of cars that are Celicas. It is also obvious to us that Toyota was not making cars prior to 1975 and that, therefore, the percentage of Toyotas sold before 1975 is, in fact, zero. Oracle, on the other hand, assumes that there is never any correlation between data items and, therefore, uses the following formulae.

```
Probability(Manufacturer = 'Toyota' and Model =
'Celica') =
    Probability (Manufacturer = 'Toyota')
  x Probability(Model = 'Celica')

Probability(Manufacturer = 'Toyota' and
ModelYear < 1975) =
    Probability(Manufacturer = 'Toyota')
 x  Probability(ModelYear < 1975)
```

In the first case, Oracle has underestimated the size of the result, and, in the second case, it has overestimated.

The error worsens as the number of filters increases, as in the following query, in which Oracle will further underestimate the number of rows in the result set.

```
select * from CarSales
 where Manufacturer = 'Toyota'
   and Model = 'Celica'
   and ModelYear >= 1975
```

Finally, consider what might happen if we encounter an "OR" conjunction such as the one in the following example.

```
select * from CarSales
 where Manufacturer = 'Toyota'
   or Model = 'Celica'
```

Let X, Y, and Z be any three filters. For readers familiar with the language of probability and statistics, we state the following results.

```
Probability(X and Y) =
    Probability(X) x Probability(Y given X)[1]

Probability(X and Y and Z) =
    Probability(X)
 x  Probability(Y given X)
 x  Probability(Z given X and Y)[2]

Probability(X or Y) =
    Probability(X) + Probability(Y)
 -  Probability(X and Y)

Probability(X or Y or Z) =
    Probability(X)
 +  Probability(Y)
 +  Probability(Z)
 -  Probability(X and Y)
 -  Probability(X and Z)
 -  Probability(Y and Z)
 +  Probability(X and Y and Z)
```

Since Oracle has no knowledge of "conditional probabilities," it uses the following alternatives (which are accurate if and only if X and Y are so-called "independent events") and consequently exposes itself to the dangers of underestimation and overestimation.

```
Probability(X and Y) =
    Probability(X) x Probability(Y)

Probability(X and Y and Z) =
    Probability(X) x Probability(Y) x
Probability(Z)

Probability(X or Y) =
    Probability(X) + Probability(Y)
  - Probability(X) x Probability(Y)

Probability(X or Y or Z) =
    Probability(X)
  + Probability(Y)
  + Probability(Z)
  - Probability(X) x Probability(Y)
  - Probability(X) x Probability(Z)
  - Probability(Y) x Probability(Z)
  + Probability(X) x Probability(Y) x
Probability(Z)
```

[1] The probability that the second filter is also satisfied when the first filter is satisfied.

[2] The probability that the third filter is also satisfied when the first and second filters are both satisfied.

We now turn our attention to the problem of joining tables using record matching criteria to associate records from one table with those of another. Let A and B be two tables and let *NR_A* and *NR_B* represent the number of rows in A and B respectively. Further, let *NDV_A* represent the number of distinct values in the joining column of A and let *NDV_B* represent the number of distinct values in the joining column of B. In the absence of better information, we could assume that each distinct value of the joining column in B is equally represented; i.e., it occurs exactly (*NR_B / NDV_B*) times. If we could further assume that every value in the joining column of A is represented in the joining column of B, then the number of records resulting from the joining operation would be *NR_A x* (*NR_B / NDV_B*). A symmetric argument could be used to estimate the answer to be *NR_B x* (*NR_A / NDV_A*). Oracle chooses the minimum of these two answers, which can be expressed as (*NR_A x NR_B*) / *maximum*(*NDV_A, NDV_B*).

Once again, we see that Oracle is using assumptions that may be far from the truth. As the number of filter criteria and the number of tables grow, the errors also grow and the query optimizer's strategy is undermined.

A participant on the "Ask Tom" website expressed the problem perfectly with the following comment.

"I became interested in the CBO's selectivity calculations trying to understand why it comes up with some of the ridiculously low cardinality estimates (like 1 when in reality there are 80,000+) which then lead to disastrous access plans that take hours, provided they finish at all, instead of minutes or seconds."[3]

Concluding Remarks

We should warn the reader that the opinions we expressed in this article don't have mainstream support. The mainstream opinion is that most, if not all, failures of Oracle's query optimizer can be remedied by improving the quality of the statistical information on which the optimizer relies. Here, for example, is a quote from an article published this year in the journal of the Independent Oracle Users Group (IOUG).

"One of the greatest problems with the Oracle Cost-based optimizer was the failure of the Oracle DBA to gather accurate schema statistics. . . . The issue of stale statistics and the requirement for manual analysis resulted in a "bum rap" for Oracle's cost-based optimizer, and beginner DBAs often falsely accused the CBO of failing to generate optimal execution plans when the real cause of the sub-optimal execution plan was the DBA's failure to collect complete schema statistics."[4]

Further Reading

Document 68992.1 in the Oracle knowledge base (Metalink) is a good source of information on the formulae and assumptions used by the Oracle query optimizer. ▲

Iggy Fernandez is a DBA for the Verizon phone company. You can reach him at **iggy_fernandez@hotmail.com**.

[3] **asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950_P8_DISPLAYID:4344365159075**.

[4] **www.ingentaconnect.com/content/ioug/sj/2006/00000013/00000001/art00003**.

Brave New World

by Venkat Devraj

Editor's Note: *In our last issue we interviewed Venkat Devraj, the author of* Oracle 24x7 Tips and Techniques*, who briefly touched on the subject of Autonomics, the newest buzz in the IT industry. A quick Google search showed more than two million hits for the phrase Autonomic Software. Intrigued, we invited Venkat back to demystify this mysterious new science and explain its applicability to the world of databases.*

Autonomics is a mechanism that can be used by any software to make itself virtually maintenance free. It is widely heralded to be the way all software will be written in the future to reduce the mundane administrative overhead typically associated with infrastructure and applications software including operating systems, databases, and ERP systems. In the interim, third-party software is emerging that can be installed on top of other applications to make them autonomic.

The Plight of Information Technology

Today, companies take for granted that they need to spend many times their original investment in software applications to hire specialized administrative staff to install, configure, maintain, and tune such software optimally.

Unfortunately, in spite of spending large amounts of money, customers find that their business doesn't necessarily get the performance, stability, or availability it needs, even though their administrators are working lots of hours. So they begin to invest in peripheral tools and point solutions to attempt to help the business gain more value from existing applications and data, as well as to assist the administrators to be more effective in their jobs.

Autonomics combines operational procedures with an expert system to interpret the current state of an environment and automate its care and feeding.

Introducing application subsets and peripheral tools often causes the complexity of the environment to increase, further adding to the number of fires every day, and it becomes even harder for the administrators to keep up. This gets construed as meaning that more head count is needed to deal with the workload and keep users happy. With the ever-increasing need for personnel, IT costs keep spiraling upwards.

Eventually, administrative teams get fragmented across geographies, time zones, platforms, and skill levels. The quality of work keeps getting more inconsistent and unpredictable, causing the typical IT environment to be held together with rubber bands and bubblegum!

Brave New World

Autonomics combines operational procedures with an expert system to interpret the current state of an environment and automate its care and feeding, both proactively and reactively, along with the ability to configure, maintain, and optimize itself. At an abstract level, it is meant to simulate an experienced administrator capable of gathering and analyzing facts and thinking on her feet.

Let's use an example to understand this better: when you get a cut and start bleeding, the blood begins to thicken and clot to avoid further blood loss. When you haven't eaten in a while, your body goes into conservation mode, where it will burn fat first before it starts eating into muscle. These are natural reactions to stimuli that are built inside every normal human body; different parts work in tandem to achieve the desired response. There are several passive activities that the body automatically does in the background while we are tending to foreground tasks. For instance, the heart is pumping blood and the digestive system is working on breaking down the recent meal.

Autonomic software attempts to duplicate the ability of the human autonomic nervous system to proactively carry out tasks that nurture and nourish the system, as well as to respond in different ways to diverse stimuli impacting the environment and to maintain multiple streams of voluntary or involuntary activities. The operational actions and reactions are hardwired into the systems' DNA via a series of crisp rules combined with fuzzy logic to avoid over-rigidity in rules processing. For instance, fuzzy logic focuses on overall system health to figure out the appropriate reaction if a specific threshold is set to 80 and the usage level goes to 79.5 and stays there for an extended period.

Characteristics of Autonomic Systems

For a system to be termed autonomic, it needs to have an awareness of its capabilities and requirements, permissible states of existence of a target environment, and methods to maintain that state as well as deal with deviations. At the same time, it needs the intelligence to identify deviations, make decisions, and retain relevant knowledge in a persistent manner, plus the ability to increase its intelligence when additional data-points

become available. The system needs to have the ability to interpret diverse events and to predict how their correlation will impact the target system.

In the database realm, this means that the system must be able to make sense of key attributes such as platform type and version, data size, database workload patterns, transaction volumes, and so on. In other words, the system needs to be capable of developing an understanding of the target environment and of leveraging this understanding to keep the databases functioning optimally.

Ten DBAs will have ten individual preferences for accomplishing the same task in the same environment, in keeping with their different skill levels and experience.

A thorough baseline of system state is attained by monitoring environmental patterns and collecting key performance indicators over a period of time. The larger the data set, the more comprehensive the understanding. This allows the autonomic system to identify significant deviations from the baseline as problems that need to be addressed, while ignoring temporary spikes such as end-of-month payroll processing. Given that there could be hundreds of servers in a complex IT environment, the metadata tends to build up fast. The autonomics system has to have a set retention period. Any detailed data outside this period can be discarded once the relevant summary data has been rolled up and preserved for comparison purposes.

Useful autonomic capabilities require algorithms ranging from simple math to simulation, artificial intelligence, and neural networking. Rather than expecting every product vendor to develop this from the ground up in their proprietary manner, there is a need for robust third-party autonomic software that can be layered on top of mainstream applications to reduce their administrative burden uniformly.

At the same time, it is unrealistic to expect busy database administrators (DBAs) to pause, review, and adopt any potential game-changing technologies with mere talks of pie-in-the-sky capabilities. Such a system has to come with sufficient out-of-the-box best practices and intelligence to immediately add value. It has to tie in a pragmatic implementation approach with the potential to be further trained via interpretation of collected metadata, as well as trusted scripts, policies, and preferences input by administrators to influence the depth and breadth of autonomics that they are comfortable with deploying over a period of time without jeopardizing their system.

Benefits of Autonomics in Database Administration

There are several obvious benefits, including a reduced need for miscellaneous DBA tools, lowered complexity, the ability to enforce discipline and standards, higher visibility and predictability, and freeing up DBA bandwidth for areas such as functional analysis, capacity planning, performance management, and service-level management. The most tangible advantage is that autonomics provides a series of consistent methods to deal with different activities, regardless of when, who, or what is triggering the methods. Imagine what would happen if each human body attempted to deal with a problem in its own unique way. That's the state of database administration today, and ten DBAs will have ten individual preferences for accomplishing the same task in the same environment, in keeping with their different skill levels and experience.

Concluding Remarks

The larger the number of fragmented tools in use, the more complex the environment, the higher the learning curve, and the lower the chances of those tools being adopted and used consistently across the board. Employee turnover really hurts the company because the replacements take significant time to come up to speed and there will always be knowledge gaps. A true autonomics platform that offers a unified solution for all database platforms in the enterprise will mitigate these issues. ▲

Venkat Devraj is the author of Oracle 24x7 Tips & Techniques *and is working on a new book,* Building Scalable Systems with Autonomics. *Venkat co-founded ExtraQuest Corporation (***www.extraquest.com***), a database technology and services provider in Denver, Colorado.*

Hello, World!
An Introduction to SQLJ

By Krishna Kakatur

Krishna Kakatur

It is impossible to talk about SQLJ without first saying a few words about JDBC. JDBC is part of the Java family, and is used to access relational databases. JDBC applications are portable across databases and operating systems by selecting the appropriate "driver" for each case. However, one has to assign an SQL statement to a variable of type String in order to use the various data access methods offered by JDBC. For large SQL statements, this requires a lot of concatenation, which is not very readable. To make it easier to write database programs, a consortium of database vendors and operating system vendors introduced the so-called SQLJ Standard for embedding SQL statements into Java programs (ISO/IEC 9075-10:2000, formerly ANSI X3.135.10-1998).

Programming in SQLJ is a three-step process.

Development: An SQLJ program consists of SQL statements embedded into a Java program. File names are usually suffixed with ".sqlj".

Translation: Translation is a process of converting SQLJ code into Java and then compiling it into a Java Class file. Libraries used in the translation process need to be in the CLASSPATH.

Execution: This process is no different than the one used in executing any other Java Class files. Any run-time libraries required by SQLJ also need to be in the CLASSPATH.

In this article, we explore Oracle's implementation of SQLJ, which is slightly different from other implementations with regard to the names of binaries and processes.

Why SQLJ?

- Productivity is increased by eliminating a lot of low-level programming.
- Less debugging is required because SQL syntax is checked during the precompilation (translation) phase.
- Concatenation of Java strings is eliminated, resulting in improved readability for large SQL statements.
- The programs are portable across databases and operating systems. However, Oracle-specific extensions cannot be used with other database technologies.
- All the power and elegance of the Java language is available to the programmer.

Getting Started with SQLJ

Oracle 8*i* and Oracle 9*i* installations include SQLJ by default. Of the various files installed in ORACLE_HOME, the important ones are as follows (JDK version 1.2 or higher).

- bin/sqlj—Oracle SQLJ translator
- sqlj/lib/translator.jar—Translation drivers
- sqlj/lib/runtime12.jar—Runtime drivers

In the case of Oracle 10*g*, you will need to download JPublisher. Once downloaded, copy the zip file into the ORACLE_HOME, and "unzip" and overwrite any existing files. Then, make sqlj/bin/jpub executable, and add sqlj/bin to the PATH variable. For Windows installations, use jpub.exe, instead of jpub.

Ensure that a Java Development Environment, such as SunJDK 1.2.x (or higher) is installed on the server. Note that a Java Runtime Environment (JRE) is not sufficient to translate and compile SQLJ programs. To check if Java is installed and configured, issue the following command.

```
java -version
```

Ensure that the SQLJ binary is available in the PATH. In other words, ensure that $ORACLE_HOME/bin is in the PATH for Oracle 8*i* and 9*i*, and $ORACLE_HOME/sqlj/bin is in the PATH for Oracle 10*g*. Verify the presence of the binaries by typing "sqlj" or "jpub" at the command prompt without any parameters. This should generate a description of how to use the binaries.

Ensure that the SQLJ Translator and Runtime drivers are in the CLASSPATH. In the case of Oracle 10*g*, you will need to rename classes12_g.jar to classes12.jar.

```
export CLASSPATH=$CLASSPATH:$ORACLE_HOME/
sqlj/lib/translator.jar:$ORACLE_HOME/sqlj/lib/
runtime12.jar:$ORACLE_HOME/jdbc/lib/classes12.jar
```

The First Program

Using SQLJ, let us develop a HelloWorld Java Class, which selects the words "Hello, World!" from the table "dual". Here

is the listing of "HelloWorld.sqlj". Line numbers are shown for easy reference in the notes that follow.

```
 1  import sqlj.runtime.*;
 2  import sqlj.runtime.ref.*;
 3  import java.sql.*;
 4
 5  public class HelloWorld {
 6
 7  public static void main(String[] args) {
 8     Connection c = null;
 9     String dbURL =
10        "jdbc:oracle:thin:@server:1521:ORCL";
11     String userid = "scott";
12     String passwd = "tiger";
13
14     try {
15        DriverManager.registerDriver(
16           new oracle.jdbc.driver.OracleDriver()
17        );
18        c = DriverManager.getConnection(
19           dbURL, userid, passwd);
20
21        // Set the Default Context for the
connection
22        DefaultContext.setDefaultContext(
23           new DefaultContext(c));
24
25        // Select a single row from the
database
26        String helloWorld;
27        #sql { SELECT 'Hello World!'
28           INTO    :helloWorld
29           FROM    dual };
30        System.out.println(helloWorld);
31
32        // Close the database connection
33        #sql { rollback work };
34        c.close();
35
36        } catch (SQLException e) {
37           e.printStackTrace();
38        }
39     }
40  }
```

Notes:

Line 1: The "import" statements make the SQLJ Runtime and JDBC classes available to this script. SQLJ Translator and Java Runtime look for these drivers in the CLASSPATH.

Line 16: Note that we elected to use the thin driver. However, the choices are varied depending on your environment and application. A full discussion is beyond the scope of this article.

Line 23: The database connection allows you to set a number of properties for the Context. However, at a minimum, you need to set the DefaultContext. Otherwise you will see the following:

```
java.sql.SQLException: found null connection
context
```

Line 27: Note the simplicity with which SQL statements are embedded into the program.

- The "#sql" prefix tells the SQLJ Translator to process the line.
- The SQL statement is simply enclosed in curly braces. It can overflow to any number of lines.
- Java variables are identified with a ":" prefix. Note that Java variable names are case sensitive.
- SQL syntax is verified during precompilation.

Line 33: This statement is not really required here, but is included to illustrate that SQLJ is transaction-aware.

Line 36: SQLException must either be caught or thrown by the Java Class. For easier debugging, one should enclose each SQL statement or transaction in its own try-catch block. Having one SQLException for the entire program, or throwing an SQLException, is not a good idea.

Running the Program

Translating with Oracle 8*i* and 9*i*:

```
sqlj HelloWorld
```

Translating with Oracle 10*g*:

```
jpub -sqlj HelloWorld.sqlj
```

The translation process produces a pure Java program, by translating each SQLJ statement into corresponding Java statements, in addition to a Java Class file. Though the generated program is much lengthier than the original, the translator includes nice comments and references line numbers from the original program.

Here is a little sample, corresponding to the ROLLBACK WORK statement.

```
/*@lineinfo:generated-code*//*@lineinfo:33^7*/

//  ************************************************
//  #sql { rollback work  };
//  ************************************************

oracleRollback(sqlj.runtime.ref.DefaultContext.g
etDefaultContext());
```

Executing the Java Class file:

```
java HelloWorld
```

Further Reading

The Oracle SQLJ page has a lot of resources and sample code.

www.oracle.com/technology/tech/java/sqlj_jdbc/index.html. ▲

Krishna Kakatur is an Oracle Certified DBA with Real Networks and has more than ten years of experience using Oracle technology. He can be reached at **kkakatur@real.com**.

Real-Time ETL
A Streams Tutorial

By Ranko Mosic

Ranko Mosic

Your company's OLTP databases are growing fast and your data warehouses and data marts must constantly be refreshed with new data. This common task is accomplished using a variety of tools and techniques:

- Homegrown ETL scripts. These are commonly used. OLTP data is dumped from the database to flat files, the files are transferred to the data mart server where they are loaded into staging tables, and the data mart tables are finally refreshed with the most recent data.
- Oracle Advanced Replication using materialized views with the fast refresh option. Direct communication between databases means simpler processing; there are no flat files involved.
- Third-party products such as Veritas Shareplex.

Oracle Streams is a relatively new technology that promises to replace batch-oriented ETL with streaming ETL. It was introduced in Oracle 9*i* Release 2. Streams can replicate changes between two databases and Oracle promotes it as a replacement for the older Advanced Replication technology.

Oracle Streams uses the same method as Veritas Shareplex to obtain data. The OLTP databases are not directly accessed, so as not to degrade database performance. The source of the data is the archived redo log files. Oracle Streams uses the Log Miner infrastructure to read the redo logs.

For an introduction to Streams and an example of a single table one-way replication procedure, please refer to Sanjay Mishra's article, "Making Data Flow," at **www.oracle.com/technology/oramag/oracle/04-nov/o64streams.html**. In this tutorial, we will describe the steps for setting up one-way replication at the schema level between two databases.

Outline

Our article is organized as follows:

- We first provide a brief overview of the Streams architecture (background processes).
- We will demonstrate the ease with which Streams replication may be configured by showing how to set up one-way replication of the SCOTT schema from the source database (ORCLA.WORLD in our example) to the target database (ORCLB.WORLD in our example).
- We will customize our Streams configuration by excluding a single table from the replication process.
- We will briefly touch upon monitoring of the Streams process.

Architecture

1. The *capture* function is performed by capture processes (c001, c001, etc.) and parallel execution processes (p001, p002, etc.).
2. The *propagation* function is performed by job queue processes (j001, j002, etc.), which propagate messages from the source queue to the destination queue using the job queue infrastructure.
3. The *apply* function is performed by coordinator processes (a001, a001, etc.) and parallel execution processes (p001, p001, etc.).

Configuration Steps

The single PL/SQL procedure MAINTAIN_SCHEMAS in the DBMS_STREAMS_ADM package is all that is required to completely set up replication of database schemas. Certain prerequisite tasks must be completed (Steps 1 through 7 below) before performing this procedure (Step 8 below).

1. Ensure that the source database is operating in ARCHIVELOG mode.
2. Set up the Streams administrator (STRMADMIN in our example) for the source and target databases.
3. Adjust certain initialization parameter settings of both databases.
4. Create a database link from the source database (ORCLA.WORLD in our example) to the target database (ORCLB.WORLD in our example).
5. Create a database link from the target database (ORCLB.WORLD) to the source database (ORCLA.WORLD). This link is needed because we will use Data Pump to export data out of the source database and to transfer it across the network to be imported into the target database.

6. Create a directory on the source machine where Oracle can deposit the replication script.
7. Set the *perform_actions* parameter to FALSE and perform the MAINTAIN_SCHEMAS procedure on the source database to generate the script that configures the replication process. This allows us to verify the script's correctness ahead of time.
8. Set the *perform_actions* parameter to TRUE and perform the MAINTAIN_SCHEMAS procedure on the source database. This will configure and start the replication process.

Configuration Details

For Steps 1 through 4, please refer to the article by Sanjay Mishra that was mentioned above. These steps are identical to the corresponding steps in the single table one-way replication procedure described in his article.

5. Create a database link from the target database (ORCLB.WORLD in our example) to the source database (ORCLA.WORLD in our example).

```
connect strmadmin/strmadmin@ORCLB.world
CREATE DATABASE LINK ORCLA.WORLD
CONNECT TO STRMADMIN
IDENTIFIED BY STRMADMIN
USING 'ORCLA.WORLD';
```

6. Create a directory on the source machine where Oracle can deposit the replication script.

```
CONNECT strmadmin/strmadmin@orcla.world
CREATE OR REPLACE DIRECTORY ADMIN AS '/home/oracle/
Streams';
```

7. Set the perform_actions parameter to FALSE and perform the MAINTAIN_SCHEMAS procedure on the source database to generate the script that configures the replication process. This allows us to verify the script's correctness ahead of time.

```
CONN strmadmin/strmadmin@orcla.world
BEGIN
 DBMS_STREAMS_ADM.MAINTAIN_SCHEMAS(
   schema_names        =>'scott',
   source_database     =>'orcla.world',
   destination_database  =>      'orclb.world',
   capture_name        =>'capture_scott',
   capture_queue_table =>'rep_capture_queue_table',
   capture_queue_name  =>'rep_capture_queue',
   capture_queue_user  =>null,
   apply_name          =>'apply_scott',
   apply_queue_table   =>'rep_dest_queue_table',
   apply_queue_name    =>'rep_dest_queue',
   apply_queue_user    =>null,
   propagation_name    =>'prop_scott',
   log_file            =>'exp.log',
   bi_directional      =>false,
   include_ddl         =>true,
   instantiation       =>
dbms_streams_adm.instantiation_schema_network,
   perform_actions     =>false,
   script_name         =>'schema_replication.sql',
   script_directory_object      =>      'admin'
 );
END;
```

A description of the parameters used above is provided in the appendix to this article.

The schema_replication.sql script contains commands to completely configure the replication of the objects contained in the SCOTT schema.

- Supplemental logging is enabled for all tables in the SCOTT schema. This ensures that all the data required for the proper functioning of Streams is captured in the redo logs.
- The SET_UP_QUEUE procedure is performed to create the capture queue.
- The ADD_SCHEMA_PROPAGATION_RULES procedure is performed to add rules to the propagation rule set.
- The ADD_SCHEMA_RULES procedure is performed to add rules to the capture process.
- A Data Pump network import is run from the ORCLB.WORLD database. The SCOTT schema is exported from the ORCLA.WORLD database and imported into the ORCLB.WORLD database.
- The capture process is started on the ORCLA.WORLD database.
- The apply queue REP_DEST_QUEUE is created in the target database and is configured using the SET_UP_QUEUE procedure.
- The ADD_SCHEMA RULES procedure is performed to define the rules of the apply process.
- The apply process is started.
- Propagation of changes between the two databases is enabled.

8. Set the *perform_actions* parameter to TRUE and perform the MAINTAIN_SCHEMAS procedure on the source database. This will configure and start the replication process. The execution of this procedure will take some time, depending on the schema size, number of objects, volume of data, etc. When it completes, all future DML and DDL changes to tables belonging to SCOTT in the ORCLA.WORLD database will be automatically propagated to the ORCLB.WORLD database.

Customization

We will now customize the replication setup to exclude the EMP table from replication (See Metalink note 239623.1).

1. Determine the name of the capture rule:

```
CONN strmadmin/strmadmin@orcla.world

select r.rule_name, r.rule_owner
from dba_rule_set_rules rs, dba_capture c , dba_
rules r
where c.rule_set_name = rs.rule_set_name
and c.rule_set_owner = rs.rule_set_owner
and rs.rule_name = r.rule_name
and rs.rule_owner = r.rule_owner
and upper(r.rule_condition) like '%:DML%';

RULE_NAME          RULE_OWNER
------------------ ------------------------------
SCOTT15            STRMADMIN

set long 100000
select rule_condition
```

(code continues on page 23)

```
from dba_rules
where rule_name = 'SCOTT15';

RULE_CONDITION
--------------------------------------------------
 ((((:dml.get_object_owner()
= 'SCOTT') and :dml.get_source_database_name() =
'ORCLA.WORLD' )) and
(:dml.get_compatible() <= dbms_streams.compatible_10_2))
```

2. Stop (quiesce) the capture process on ORCLA.WORLD so that the rule can be changed:

```
begin
 dbms_capture_adm.stop_capture('capture_scott');
end;
/
```

3. Alter the capture rule to exclude the EMP table from the replication process.

```
begin
 dbms_rule_adm.alter_rule('SCOTT15',
'(:dml.get_object_owner() = ''SCOTT'' and not ' ||
':dml.get_object_name() = ''EMP'' and ' ||
':dml.is_null_tag() = ''Y'' )');
end;
/
```

4. Restart the capture process:

```
begin
dbms_capture_adm.start_capture('capture_scott');
end;
/
```

From now on, changes to the EMP table in ORCLA.WORLD will not be propagated to ORCLB.WORLD.

Monitoring

Replication is now fully configured. DML as well as DDL changes are flowing from the source database to the target database. How do we determine what is happening to the capture, propagate, and apply processes?

- To determine the state of the capture process:

```
select state from v$streams_capture;
STATE
---------------------------------
CAPTURING CHANGES
```

- To determine the total number of propagated messages:

```
select total_msgs from v$propagation_sender
TOTAL_MSGS
--------------------
15863766
```

- To determine the state of the apply process:

```
select state from v$streams_apply_reader;
STATE
--------------
SPILLING
```

Summary

Oracle Streams is a relatively new tool primarily designed to help move data between databases in real time. The easy setup and good performance make it a welcome addition to a DBA's toolset.

Appendix

Here are the explanations of the parameters of the MAINTAIN_SCHEMAS procedure.

- *schema_names* is the name of the schema that will be propagated (SCOTT in this example).
- *perform_actions* is the parameter that specifies if a script should actually be executed. If the value is TRUE, then configuration actions will actually be performed. If it is set to FALSE, then the procedure will only generate a script.
- *capture_name* is the name of the capture process configured to capture changes made on the source database.
- *capture_queue_table* is the name of the queue table for the queue used by the capture process.
- *capture_queue_name* is the name of the queue used by the capture process.
- *capture_queue_user* is set to NULL in our example, meaning that the procedure will not grant any privileges.
- *propagation_name* is the name assigned to the propagation.
- *log_file* is the name of the Data Pump log file.
- *bi_directional* is set to FALSE in our example because we are only creating one-way replication from the source to the target database.
- *include_ddl* is set to TRUE in our example, which means that both DML and DDL changes will be replicated from the source to the target database.
- *instantiation* is set to INSTANTIATION_SCHEMA in our example. It means that target schema instantiation will be done using Data Pump.
- *perform_actions* is set to FALSE in our example because the procedure is only generating a script for review.
- *script_name* is set to "schema_replication.sql" in our example. This script will be written to the Oracle directory object "admin". It will contain all the steps to configure the replication process.
- *script_directory_object* is set to "admin". This is the name of the Oracle directory object. ▲

Ranko Mosic, B. Eng., is an independent consultant and Oracle DBA based in Toronto, Canada. He has 14 years of experience as an Oracle DBA, has provided Oracle DBA consulting services to clients across North America, and was profiled in Oracle Magazine. *He can be reached at* **ranko.mosic@gmail.com**.

Spring Conference Abstracts

	Auditorium	ECR Conference Room	X7 Conference Room
10:45 a.m. to 11:45 a.m.	**"Hints and How to Use Them"** **—Jonathan Lewis** This presentation explains what hints are, and why the name "hint" was a big mistake. We will examine a few 10053 trace files to demonstrate that hints are orders—but orders that it may be impossible to obey. After looking at a few of the common misunderstandings about hints, we will go on to discuss the pros and cons of using hints, and the degree of care needed when using them.	**"Oracle JDeveloper 10*g* and JavaServer Faces: High-Performance UIs on the Web"** **—Avrom Roy-Faderman** After a brief discussion of the general architecture of Oracle's Application Development Framework (ADF), this presentation focuses on JSF and ADF Faces. It explains how to use them to create a UI with much higher performance and interactivity than is common in web applications today.	**"Database Security—the Past, the Present, the Future"** **—Mark Kraynak** The presentation includes a live database hacking demonstration. Learn about next-generation blended database attacks (SQL injection, web worms, phishing, etc.); the use of dynamic database query profiling to prevent data leakage and identify theft; and security practices that address compliance requirements such as SOX 404, HIPAA, etc.
12:45 p.m. to 1:45 p.m.	**"What's Up with dbms_stats?"** **—Terry Sutton** For years Oracle has been telling us to use the dbms_stats package instead of the analyze command to gather statistics to be used by the Oracle optimizer. But your choice of which options to use with dbms_stats can dramatically affect your results. The focus here will be on actual experience, not on what the documentation says.	**"Performance Diagnostics Using STATSPACK Data"** **—Tim Gorman** STATSPACK has been largely misunderstood due to the inadequacies of the single report that is provided with it. In fact, it provides a powerful data warehouse of analytic data that can be used to quickly gain understanding of the trends affecting application performance. A method and scripts are supplied and demonstrated.	**"Strategies and Tools for Centralizing and Automating Database Management"** **—Matthew Zito** Companies have more databases; more data; and increasingly complex availability, security, and compliance demands than ever before. We will discuss strategies and tools for centralizing and automating database management that will save time and make the DBA's life easier.
2:00 p.m. to 3:00 p.m.	**"Application Development Tuning Best Practices"** **—Peter Koletzke** This presentation explains some best practices for making applications run most efficiently. It provides tips for SQL optimization, such as breaking up complex queries and materialized views, and ways to make PL/SQL more efficient, such as BULK COLLECT and FORALL. On the tools side, Oracle Forms applications and J2EE applications are discussed.	**"Services for a DBA: May Your Workloads RIP (Run In Peace)"** **—David Austin** This two-hour presentation reviews the history of Oracle database services from Oracle8*i* until Oracle10*g* R2. It includes details of implementing workload management and high-availability options in the current release using services with connection pools, Fast Application Notification, and load-balancing options.	**"What Does Sarbanes-Oxley Have to Do with the Management of Databases?"** **—Steve Lemme** Top of the mind among business executives today is how to meet new regulatory compliance and corporate governance requirements. New laws are changing the way companies collect, retain, and manage information. DBAs need to understand what is happening in the corporate business world and how it will directly impact their job role.
3:30 p.m. to 4:30 p.m.	**"Scalable, Fault-Tolerant NAS for Oracle—The Next Generation"** **—Kevin Closson** This presentation focuses on architectural differences between various NAS offerings and why NAS is a reasonable deployment option for Oracle. It also covers a Proof of Concept of Oracle 10*g* R2 RAC attached to a new NAS architecture called Symmetric Multi-Headed NAS.		**"Religion, Revelation, Revolution! Best Practices and Projects for Managing Databases"** **—Steve Lemme** The complexity of managing databases has increased so significantly, some are considering outsourcing for relief. New laws are increasing the pressure on IT. This presentation focuses on the new business challenges and best practices, as well as how DBAs should be educating themselves and spending their time.

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, Darrin Swan, at darrin.swan@quest.com. ▲

Long-term event sponsorship:

LOCKHEED MARTIN

CHEVRON

ORACLE CORP.

Thank you! Year 2006 Gold Vendors:

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

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

TREASURER'S REPORT

Diane Lee, *Treasurer*

Beginning Balance		
January 1, 2006		**$ 33,871.73**
Revenue		
Membership Dues	18,230.00	
Meeting Fees	760.00	
Vendor Receipts	10,500.00	
Interest	84.47	
Total Revenue		**$ 29,574.47**
Expenses		
Regional Meeting	7,246.68	
Journal	7,304.85	
Membership	439.34	
Administration	2,154.96	
Website	275.00	
Board Meeting	622.36	
Marketing	1,215.11	
Vendors	107.70	
FTB	800.00	
Total Expenses		**$ 20,166.00**
Ending Balance		
March 31, 2006		**$ 43,280.20**

News Roundup

by Ravi Kulkarni

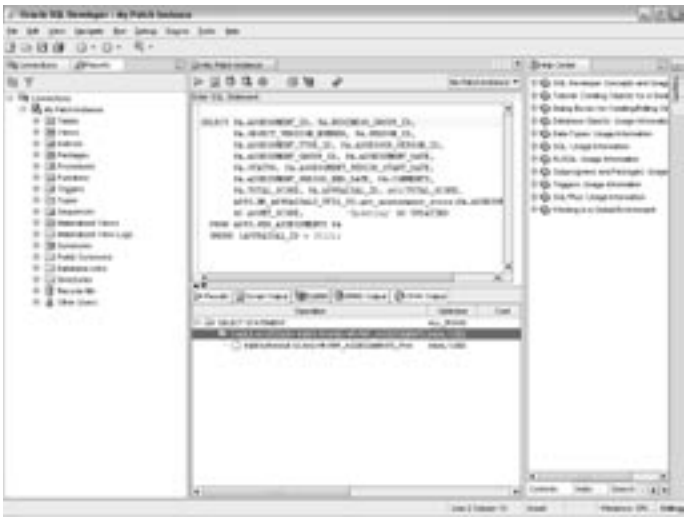
Our intrepid database gumshoe trolls the highways and the byways of the World Wide Web, looking for newsworthy items.

Ravi Kulkarni

Big Fish Eats Little Fish

Oracle continues its feeding frenzy. In February, Oracle acquired Sleepycat Software, a provider of storage engine software for MySQL databases. This follows last year's purchase of InnoBase, another maker of storage engine software for MySQL databases.

Oracle plans to use the technology in embedded database products, but some users of MySQL worry that Oracle may be trying to stifle the competition. Be that as it may, Oracle is definitely concerned about competition from the open source movement, if the recent release of the free 10*g* "express" edition is any indication.



Is It a Bird? Is It a Plane? It's Raptor!

Oracle has finally released a decent Windows GUI called SQL Developer and code named Raptor. It provides a graphical interface for SQL and PL/SQL, a PL/SQL debugger, an Export wizard, user-defined reports, Explain Plans, and more.

The popular Toad program may finally face some competition. It is a great feeling to be freed from the tyranny of SQL*PLUS after two decades.

RAC in the Box

Oracle, Red Hat, and VMware have collaborated to put together a package that will have you up and running on a fully configured and operational Linux-based RAC database in less than an hour, or so the claim goes. All that VMware needs to work its magic is a Windows XP machine with a decent amount of memory.

It is a great opportunity for Oracle database administrators and programmers to experiment with some really cool technology—including Real Application Clusters 10g R2, Oracle Clusterware, Automatic Storage Management, HTML DB and Oracle Warehouse Builder—without having to endure lengthy installation routines. Go to **www.oracle.com/technology/tech/linux/vmware/index.html**.

Imitation Is the Sincerest Flattery

SQL Server now offers Read Consistency as Oracle has done for many years. Readers will no longer block writers; neither will writers block readers. But row locks will still escalate to table locks, so deadlocks will continue to rule the SQL Server world.

Oracle still leads in features as well as performance. Read more about it at **www.oracle.com/technology/deploy/ performance/withSS2005.html**. ▲

Ravi Kulkarni has 20 years of IT experience in roles ranging from programmer to CEO. He is currently an Oracle applications administrator for IBM in Phoenix, AZ. His interests outside work include yoga and early childhood development. His address is **kulkarniravi@yahoo.com.**

A Funny Thing Happened at Work

Every week we do a code push for a certain software development group. The code push always includes an SQL script containing one or more statements to be executed by a DBA. One time, the script ran quicker than usual. We checked it and all it contained was "`SELECT * FROM DUAL`". The force of habit is very hard to resist.

—Submitted by Mark Cippel, CA

One day, website traffic and database traffic went through the roof. We checked the database and found that all the queries involved a certain promotion code. A quick Google search showed that the code had been posted on several Internet message boards. The promotion was a $15 discount on any $15 purchase! Apparently it had been intended for loyal customers only, but it's not easy to keep a secret in the Internet age.

—Submitted by Mark Cippel, CA

Do you have a funny story to share? Send it to journal@nocoug.org

NoCOUG Spring Conference Schedule

May 18, 2006, at Lockheed Martin, Sunnyvale, CA

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

8:00–9:00A.M.	**Registration and Continental Breakfast**—Refreshments served
9:00–9:30	**General Session and Welcome**—Darrin Swan, NoCOUG President
9:30–10:15	**Keynote:** *How to Test*—Jonathan Lewis, JL Computer Consultancy
10:15–10:45	**Break**
10:45–11:45	**Parallel Sessions #1**
	Auditorium: *Hints and How to Use Them*—Jonathan Lewis, JL Computer Consultancy
	ECR Conference Room: *Oracle JDeveloper 10g and JavaServer Faces: High-Performance UIs on the Web* —Avrom Roy-Faderman, Quovera
	X7 Conference Room: *Database Security—the Past, the Present, the Future*—Mark Kraynak, Imperva, Inc.
11:45A.M.–12:45P.M.	**Lunch**
12:45–1:45	**Parallel Sessions #2**
	Auditorium: *What's Up with dbms_stats?*—Terry Sutton, Database Specialists
	ECR Conference Room: *Performance Diagnostics Using STATSPACK Data*—Tim Gorman, SageLogix
	X7 Conference Room: *Strategies and Tools for Centralizing and Automating Database Management* —Matthew Zito, GridApp Systems
1:45–2:00	**Break**
2:00–3:00	**Parallel Sessions #3**
	Auditorium: *Application Development Tuning Best Practices*—Peter Koletzke, Quovera
	ECR Conference Room: *Services for a DBA: May Your Workloads RIP (Run In Peace)—Part I* —David Austin, Oracle Corporation
	X7 Conference Room: *What Does Sarbanes-Oxley Have to Do with the Management of Databases?* —Steve Lemme, Computer Associates
3:00–3:30	**Raffle and Refreshments**
3:30–4:30	**Parallel Sessions #4**
	Auditorium: *Scalable, Fault-Tolerant NAS for Oracle—The Next Generation*—Kevin Closson, PolyServe, Inc.
	ECR Conference Room: *Services for a DBA: May Your Workloads RIP (Run In Peace)—Part II* —David Austin, Oracle Corporation
	X7 Conference Room: *Religion, Revelation, Revolution! Best Practices and Projects for Managing Databases*—Steve Lemme, Computer Associates
4:30–??	**NoCOUG Networking and Happy Hour at Faz, 1108 North Mathilda Avenue, Sunnyvale.**

Session descriptions appear on page 24.

Cover photos © Tom Wagner. Mount Timpanogos, Utah.

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

NoCOUG
P.O. Box 3282
Danville, CA 94526

RETURN SERVICE REQUESTED

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
SAN FRANCISCO, CA
PERMIT NO. 11882