Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

VOL. 18, No. 3 · AUGUST 2004 $15

Ride with the NoCOUG Team!

Teamwork Makes a Difference (see page 27)

NoCOUG Goes One-on-One with Tom Kyte

Find out what happens behind the scenes of the AskTom website

Stayin' Alive: High Availability Without Breaking the Bank

Get the ins and outs of high availability from Jeremiah Wilton

Where's the SQL?

Joel Thompson discusses the lasting value of the best practices he learned on a project for a biotech firm

Much More Inside . . .

Knowledge Is Power

It has been said that knowledge is power and with knowledge you can conquer fear. Being empowered and fearless are probably the best tools for improving your career. We hope you find these qualities in the pages of the *Journal*. Each quarter, we research ideas and solicit technical papers from experts that are meant to help make your job more rewarding by having technical information and solutions available to you.

Feedback is the ideal way for us to bring you what you want. Let us know if you want more articles on a particular subject or if formatting an article differently would make it easier to read. We are working toward improving the look and content for you.

—Lisa Loper and Laurie Robbins, *Journal* Editors

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: Lisa Loper, *NoCOUG Journal* Editor, at journal@nocoug.org.

The submission deadline for the upcoming November issue is October 1, 2004. 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

Working Smart vs. Working Hard

Roger Schrag

It has been said that hard work is good for the soul and the pocketbook. This may be true, but there is also a lot to be said for working smart. Especially in lean times when companies want to do more with less, making the smartest use of our work effort is all the more important.

The difference between working hard and working smart came to mind the other day when I got a phone call from a DBA who said their system was running much slower ever since they upgraded their database from Oracle 8i to Oracle 9i. They had called their application software vendor for help, and the vendor had recommended that they increase the database block size to 16 Kb.

After more discussion, I realized this person was calling me just to ask for the best way to get a 16 Kb block size—should they add a new tablespace with the 16 Kb block size or rebuild the entire database? I was stunned when I realized how limited in scope the purpose of the telephone call was.

I decided to probe a little. I saw some red flags and I wanted to help this DBA work smart instead of just working hard. So I asked how bad the performance degradation was. The DBA explained that users were complaining that a screen that used to take one minute to open now took more than an hour. I also asked when the degradation began, and the answer was immediately after the Oracle 9i upgrade.

This sounded fishy to me and, after much lobbying, I got the DBA to let me work with an end user for an hour so that we could generate a SQL trace file and TKPROF report.

Well, guess what? Almost all of the resource usage in the trace file was attributed to one query—a complicated query. A complicated query with hints. A complicated query with a hint telling Oracle to use an index that did not exist! So I called up the DBA and asked about the method used to perform the Oracle 9i upgrade. It turned out that the upgrade had been performed using the export/import method, and the export of the Oracle 8i database was still available on the server.

You can probably guess the rest from here—the missing index was present in the Oracle 8i export file, and its presence in the Oracle 9i database would have made the slow query run very quickly. The index would have been large relative to other indexes in the database, and it seems likely that the database ran out of temporary space (or space in the index tablespace) during import and so the index went missing on the Oracle 9i database. Hence certain queries ran disastrously slow on the new database, thus the user perception that the system ran slower after the upgrade.

It occurred to me that if the DBA had spent a day configuring the database for multiple block sizes and moving data into the new tablespaces, he would certainly have worked hard. But the users would have been no better off. By working smart—collecting the facts, asking the right questions, and digging deeper instead of taking a general rule of thumb from the software vendor as gospel—we were able to avoid a lot of tedious work for the DBA and bring relief to the end users much more quickly.

So the next time you find yourself banging your head against the wall with a problem you just can't seem to fix, or the next time you find yourself in a panic because you can't keep up with the work dumped on your plate, think about how you could be working smarter instead of harder. Taking a few moments to step back and take a fresh look at the big picture and the path you are on could be time well spent!

And a really smart way to spend your time would be to attend the Summer Conference of the Northern California Oracle Users Group at ChevronTexaco in San Ramon on August 19. I am really excited about this conference. We have some pretty incredible folks coming to share their knowledge. Tom Kyte, Jonathan Lewis, Peter Koletzke, Tim Gorman, Virag Saksena, and others will be giving technical presentations in one very full day.

Where else can you learn from all of these industry leaders while staying close to home? Plus, the knowledge you gain will almost definitely help you work smarter! Check out the back cover of this publication for conference details and logistics. As always the event is free for NoCOUG members and costs just $40 for non-members. See you August 19! ▲

One-on-One with Tom Kyte

Have you ever had an experience in your life that made you think: *What if . . .*? I guess we all have. But did you know that if Tom Kyte had taken a position as a help desk analyst after college, he might be a retired AOL executive today? Instead, he's the Tom behind AskTom, the author of two books, *Expert One-on-One: Oracle* (Apress, 2003) and *Effective Oracle by Design* (Oracle Press, 2003), and co-author of *Beginning Oracle Programming* (Apress, 2003). He's also a vice president in the Oracle Government, Education, and Healthcare group and a popular lecturer at Oracle User Group meetings. Sure, he might be retired now, relaxing in one of his three houses, but I think most of us in the Oracle community appreciate that he works for Oracle instead. We wouldn't have an AskTom website (which averages about 50,000 page views per day) or the terrific reference books he's written. He wouldn't be speaking at the NoCOUG Summer Conference, and NoCOUG wouldn't have had the opportunity to interview him for this article!

Tom Kyte

Where did you get your start working with Oracle software?

Tom Kyte: Well, when I graduated with a major in mathematics, I realized you can't do very much with a math degree, so I applied for a job I was qualified for: *programmer wanted—no experience required.* The job was as a programmer working in a multi-database environment for a consulting company. We had several databases, including DB2, Sybase, Informix, and Oracle. That's where I started, and eventually joined Oracle Corporation in 1993 where I've been ever since.

Did you begin developing the AskTom website when you started at Oracle Corporation?

Tom Kyte: No, that came along seven years later. When I joined Oracle, I was a sales consultant, giving technical presentations and working with customers on technical issues. Then, around 1994, I got hooked on Usenet Newsgroups on the Internet. I began answering questions, and I saw this question about renaming a column. I provided an answer, and then saw another answer to the same question, but their solution involved updating a data dictionary table. I quickly replied and showed how updating the data dictionary would result in ORA-600 errors. Then I learned the information on modifying the data dictionary came from a user-submitted tip in *Oracle* magazine. When I contacted *Oracle* magazine, I discovered they didn't have anyone technically reviewing the user-submitted tips. So I began reviewing the tips, and then they asked me to respond to questions that were sent to the magazine. As I began to answer questions, even more questions were coming in, so we set up a way to answer the questions through email. Receiving the questions through email wasn't a very scalable solution, so we needed something else, which became the genesis of the AskTom website. We set up the first version of the site in May 2000.

Do you have help answering questions received at the AskTom website?

Tom Kyte: I probably answer more than 99.9% of them. When I get a difficult question, I consult with one of the people I manage. We each specialize in various core technology areas. In those cases, I start the answer to the question with their name, so they always get the credit. Otherwise, I answer all the questions myself, so I travel with a wireless card in my laptop.

It sounds like you spend a lot of time answering questions. You must really enjoy it.

Tom Kyte: Yes, I really enjoy helping people be successful with the software. Instead of someone getting frustrated or doing something the wrong way, I'd rather they had a place to ask their questions and learn how the software can work for them. Also, many people search the site for solutions from previous discussions.

On AskTom, I noticed you often refer readers to the Oracle Concepts Guide, rather than suggesting they pur-

chase a third-party Oracle book. Do you consider that the best place to start for documentation?

Tom Kyte: Yes. In fact, on the inside back cover of my latest book, *Effective Oracle by Design*, I have a documentation roadmap. At the top of this roadmap, for both DBAs and developers, is the *Oracle Concepts Guide*. It's well written and a great place to start learning many of the concepts behind the software. It gives a 30,000-foot view of how things work, how Oracle is architected, the explanation of multi-versioning, read consistency, and many other important concepts. It's not the dry SQL reference manual, with 29 pages of train track diagrams showing you the syntax of a SELECT statement. It answers questions such as: If you run these two statements at the same time, what is Oracle doing under the covers; It's a really good way to figure out how Oracle works, and any time someone asks, "What Oracle book should I buy?" I say, "Don't buy any book until you've actually read these first," referring to the documentation roadmap.

What do you enjoy most about your job?

Tom Kyte: I enjoy walking into a meeting room full of people who are upset for some reason. The database isn't working the way they thought it should or it's not working as fast as they think it should or it's just not doing what they anticipated. I can usually turn that around in the course of an hour or two. Whether it's describing tools available to identify and tune bad SQL or just explaining how Oracle is instrumented, they begin to understand, and that seems to ease their frustration. So the most fulfilling part is to take people that are really mad at Oracle and turn them into people that are really impressed with Oracle.

How did you become interested in writing books?

Tom Kyte: The first book, *Expert One-on-One: Oracle*, started as a collection of white papers. If you look at the way it's organized, the first third is sort of how Oracle works, transactions, redo, undo, how memory is managed, how the processes work, locking models, etc. The rest of the book arose out of questions asked regularly at the AskTom site. In those chapters, I explain the Oracle feature discussed, describing why the feature is there, how it works, what problems it solves, what problems it doesn't solve, and caveats.

The second book, *Effective Oracle by Design,* is more of a best-practices type of book. It answers the question, if you ask me; this is how I would approach the problem, such as writing SQL, designing an efficient schema, or the best use of the 15 different data structures in the database, index organized tables vs. clusters vs. hash clusters, etc.

Now that 10g has been released, what features are people getting excited about?

Tom Kyte: The manageability capabilities and features seem to be resonating with people. For the most part, there are two kinds of databases; the database that needs a dedicated DBA—it's a really large, multi-gigabyte/terabyte database and/or a high transaction OLTP database—and then there is the other database. This database is usually small, not a lot of activity, and is sitting in a corner somewhere. With this database, you don't have a lot of time to spend figuring out how much memory the SGA needs, how to manage disk space, etc. To setup this database, all you want to do is tell Oracle I have six devices for storage, stripe and/or mirror everything, and distribute the I/O. If you have 1GB of memory on the machine, you tell Oracle you want to allocate 512M to the SGA, 256M to the PGA and it will determine all the specific memory sizing for you. Then, periodically you pull up the DBCONSOLE and it will tell you about any bad SQL, offer to tune it for you; these manageability features offer you a sort of fire-and-forget approach. And more importantly, it creates time for you to spend on the larger, higher-volume databases.

I noticed you are a member of the BAARF party. What are your thoughts on RAID-5?

Tom Kyte: Well, Mogens—co-creator of BAARF—is categorically against RAID-5 in all cases. But like every story, there are two sides. Much like politics, there's the left wing and the right wing. And you sort of pick, even if you believe a little bit of the left but you are more right.

I don't like RAID-5 in general, but if you've bought it and it's already configured, there are implementations where it can be successful. If you think about how Oracle works, for a predominantly transaction-based system, not excessively high volume, datafiles on RAID-5 may work well since dbwriter is writing to them in the background. You can also tune that to a degree—sizing buffer cache, checkpoints, etc.—to make sure the data gets out there fast enough. I wouldn't want my redologs on RAID-5 because a user waits for a write to a redolog, but they don't wait for a write to a datafile necessarily. And I don't want my TEMP on RAID-5 because users wait for writes to TEMP, if you are doing a sort and it runs out of sort area size.

In a data warehouse, however, RAID-5 can be deadly because the data warehouse does a lot of direct path operations and a direct path operation waits for writes to disk to complete—it's not buffering through dbwriter. These direct path operations such as ALTER TABLE . . . MOVE, and CREATE INDEX are some of the big and bulky operations you see in a data warehouse environment. And direct path load to RAID-5 is not going to be as efficient as a direct path load operation to RAID 1+0 or 0+1 or just a bunch of disks.

What are some rules of thumb on consolidating database schemas into a single instance, and when should databases be on the same server or different servers?

Tom Kyte: In general, there are two kinds of applications, the custom-developed applications—you write and own the code—and the software you buy—Oracle Financials, SAP,

PeopleSoft, etc.—The software you buy treats the database as a black box. This software will dictate the version of Oracle you can use, with the specific patches necessary, and you really don't want anything else in this database. For everything else, you want as few database instances as possible.

To me, the only right number of Oracle instances on a single computer is one. So when I see a server with five or six Oracle instances running on it, I'm thinking, "Something is wrong here." Or if you have five or six custom-developed applications in separate databases, they should be in a single database instance. For those custom applications, if you took the five instances and the applications and put them in a single instance, you'd find you need less memory because you have less SGA requirements, fewer database processes, and you'd have better control over system resources.

On top of that, you haven't lost anything since the addition of tablespace point-in-time recovery. You can segment the applications inside the database by tablespace. Each application gets one tablespace (which contains both data **and** indexes unless you have a multi-gigabyte application), since the striping is done at the operating system, so separating tablespaces is mostly for administration purposes. For instance, on my server at Oracle we run about 30 different applications, so we have about 30 different tablespaces and they coexist nicely.

The problem with generalizations is they may be appropriate in some cases, but not all.

Another problem with five different instances on a single machine: In five years, I guarantee, each of those instances will be at a different release level, so you will have five different versions of Oracle on that machine. How many DBAs does it take to manage an instance? How many DBAs does it take to manage five times as many instances? There are two different numbers there.

Are you able to incorporate anything from your math background into the work you do at Oracle?

Tom Kyte: Studying math in college required learning a lot of theorems and learning how to prove or disprove them. I definitely carry that concept through. When somebody asks me a question, I don't give them an answer A or B. I give them an answer A, and here's the proof of why that's the case, and here's where it doesn't work so well or the limitations. In mathematics, you also build on prior knowledge. In fact, it's promoted within the field; it's a science that continues to grow but also a science where things change over time. What was known to be true in the past can be proven to be wrong in the future, so you constantly have to change and re-evaluate. I often say: Question Authority. The same is true in the database world; what was true in version 6 might not be true in version 7 and might become true again in version 8.

I also don't like generalizations. For instance, a DBA has a problem with a view, therefore, views are not used anymore, or stored procedures are never permitted for whatever reason. The problem with generalizations is they may be appropriate in some cases, but not all. I heard a presentation on PL/SQL Server Pages (PSPs) and the presenter said, "Unfortunately, we couldn't use this technology at a customer site because they had a rule that you could only use packages. But PSPs won't generate packages. In general it's a good rule, and if you go to my website and my books, I say you should use packages. However, in the case of PSPs, you don't need to use packages because the benefits of packages don't apply. So my math background also taught me to think things through, understanding all the ramifications.

Can you recall a job experience that had a profound effect on your career?

Tom Kyte: Probably the most pivotal point was in my first job. I was working for a company doing government contracting and I found out about this new project, which everyone wanted to join. My manager refused to let me go because he thought I was too valuable on his team. Well, that caused me to go down the hall and volunteer for a project I had turned down because of the commute to New Jersey from Virginia each week. However, finding out I was getting boxed-in caused me to volunteer for that project. And it ended up being the most intense and rewarding database project I'd worked on up to that point. I learned most of what I know about relational theory, concurrency control, distributed transaction, and database processing in general. It's also where I met a group of Oracle consultants, which persuaded me to work for Oracle when I was ready to quit that company and take a new job. Before Oracle, my job was that that of a programmer, analyst, and DBA. I had never done any public speaking or written anything, and that's totally changed now. Perhaps it's also a warning to managers not to box in their employees—but to give them opportunities—because they will just run away.

And finally, what do like to do in your spare time?

I spend most of my free time with my wife and two kids. My weekends are full with the kids and their soccer games and horseback riding. As a family, we also enjoy going to the movies, relaxing at home, or taking a vacation together. ▲

You can access the AskTom website to search the question archives, ask a question, or to find out where Tom is speaking next by visiting asktom.oracle.com.

Acknowledgements

Interview conducted by Laurie Robbins, NoCOUG Journal *assistant editor. Thanks to Andy Rivenes, Bill Reid, and Dale Snearly for their input on technical questions.*

Stayin' Alive: High Availability Without Breaking the Bank

Second Revision, June 2004

by Jeremiah Wilton, Independent Oracle Professional

Part One: Planned Outages*

Preface to the second revision

This is a revised edition of a paper I wrote in 1999 for Oracle Openworld. The industry has seen many changes since the days of Oracle 8.0, and I have tried to integrate those changes into this revision of the paper. Many of my original methods and advice still hold true, but have been embraced and integrated or productized by Oracle, in packages such as DataGuard and RMAN. Overall, the biggest change is that there are simply a lot more ways to achieve high availability now. By including as many of the approaches as I know, I hope that I have made the paper relevant to the widest variety of environments possible.

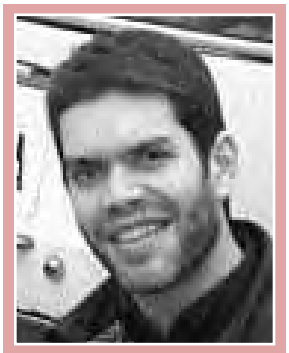
Introduction

Oracle's principal corporate direction in recent years shows an overwhelming attentiveness to features that promote availability and scalability. From Real Application Clusters to DataGuard to Flashback Query, much effort has gone into keeping Oracle's customers from suffering debilitating outages and data losses.

Oracle's emphasis on reliability was reflected in their 9i slogan, "unbreakable." Alongside the huge technical effort, an equally great marketing effort has sought to promote Oracle's reliability. From the customer point of view, this often took the form of a constant drumbeat touting the universal applicability of RAC as a solution to all scalability and availability problems.

In my view, Oracle's RAC-centric marketing sells Oracle short by ignoring a broad spectrum of huge advances in 9i and 10g. These features, including logical standby (part of DataGuard), streams, improved multimaster replication, flashback query, cross-platform transportable tablespaces, and improved RMAN (including the pipe interface), constitute the great adaptability for any size organization that is Oracle's greatest strength.

In addition to the diverse features that allow any Oracle customer to engineer the optimal availability-for-cost configuration for their organization, there are hundreds of low-tech techniques and approaches that can contribute as much, if not more, to the overall reliability of a system than any specific availability feature. This paper covers the judicious application of such creatively tailored approaches, which will provide far greater overall improvement in stability per dollar spent than simply implementing RAC and DataGuard as Oracle's marketing seems to suggest.



Jeremiah Wilton

Globalization and e-commerce have completely altered the IT availability landscape over the past seven years. Beginning in 1997, as the dot coms were making their initial public offerings, with the limelight focused on them in earnest, news sources including *The Wall Street Journal* and *C|net* made conspicuous—and gleeful—note of extended website outages, including a couple of multihour failures by Amazon.com and a spectacular two-day Ebay outage.

Prior to the dot-com boom, many financial institutions and other large Oracle clients were not true 24/7 systems, and allowed weekend and nighttime maintenance windows. They did not concern themselves greatly with the impact of uptime-reducing activities such as upgrades and migrations. Those few organizations that needed true high availability had to do a great deal of engineering on their own. Some went as far as implementing standby databases using AIJ files as early as v.5. The downside of these efforts was a lack of good support from Oracle and few peers to consult in the user community.

Although Oracle Parallel Server (OPS), the precursor of RAC, was available on Oracle v.7, this platform was difficult and impractical to deploy in any environment with a diverse workload. Each node could only share cached blocks with another node by writing those blocks down to disk, where the other node could take possession of them. The whole process meant that it was extremely resource-intensive to run workloads that required the same data on different nodes at the same time. This limitation begged the question, "If we have to separate the workloads anyway, why not use two separate databases?"

* *Editor's Note: This paper has been broken into a two-part series. Part One provides an overview of high availability and dealing with planned outages. Our next issue will feature Part Two, dealing with unplanned outages.*

Replication was in its infancy in the mid-1990s and did not attain stable bug-free maturity until version 8. All in all, Oracle was weak on high availability until the dot coms and the Internet boom forced them to change direction and focus on it. The rest of the industry can thank the dot coms for precipitating the change of direction that resulted ultimately in Streams, logical standby, flashback query, advanced queues, cross-platform transportable tablespaces, solid asynchronous multimaster replication, and RAC.

The planned outage oversight

Because many IT professionals fail to consider planned downtime as outages, there is a tendency to focus on prevention of unplanned outages without sufficiently seeking to reduce the length and number of planned downtimes. Perhaps because of the overall focus on unplanned outages, companies seldom regard planned maintenance as an availability problem, and therefore seldom actively seek to reduce or eliminate them. Such attitudes must fall by the wayside in the arena of global and Internet commerce, as managers and staff realize that all downtime, planned or unplanned, detrimentally affects the bottom line.

I often encounter DBAs and SAs that claim to run 24/7 systems, but still perform cold backups or other unnecessary tasks that make the system unavailable. The common notion that planned outages and "maintenance windows" somehow are exempt or don't count as downtime is an idea that needs to be banished from our field. While it is true that you can schedule such tasks during periods of lowest usage, the mere act of shutting the instance down and starting it back up can be destabilizing due to human error (typo in the initialization file), O/S peculiarity (unremovable shared memory segments), or human error (open database file deleted by mistake, but held open by running instance until shutdown). In addition, many companies that tolerate frequent planned outages may find that business processes could be made more efficient by moving some batch jobs or other work into the time slot that a scheduled outage window used to occupy.

Reducing and Eliminating Planned Outages

Unnecessary outages

Some outages are unnecessary. The king of unnecessary outages is the cold backup. With minimal engineering and the tiniest bit of effort, any Oracle database can be backed up online. The prerequisite to most online backups, such as tablespace hot backup and RMAN, is archivelog mode. It can be safely stated that anyone running in archivelog mode has no valid reason to be taking cold backups. Hot backups achieve the same result, but avoid the destabilization of shutting down and starting up the instance(s). Many scripts for performing online backups on a variety of platforms are available in the public domain.

Another maddeningly common planned outage is the scheduled reboot. Believe it or not, a huge number of companies restart their Oracle instances or even reboot their server hosts on a periodic basis. Server hosts, regardless of the platform, are designed to run for months at a time without rebooting. Any problem that necessitates frequent rebooting should be thoroughly investigated for the root cause, and eliminated through the vendor's support channels. Any such routine rebooting should be done only as a temporary measure until the vendor has provided a patch or upgrade that addresses the specific technical problem that is forcing periodic rebooting.

Necessary outages

Other planned outages, such as migrations and reorganizations, are not altogether unnecessary. But any planned outage can be dramatically reduced in duration and impact through the use of novel approaches and careful scripting and management. No effort should be wasted on planned necessary outages because **they are the outages over which DBAs can have the most control**. The mantra for planned outages should be "why waste time when seconds count?"

The Little Things

Few major architectural configurations will have as much impact on a system's availability as just maximizing the efficiency of the small tasks that contribute to the time line of an outage. Any outage that is worked manually could be scripted to some degree, saving the time spent fumbling and typing. Beyond scripting, a variety of low-tech techniques can drop hours of annual downtime off your availability numbers. A selection of these techniques follows.

Shutdown abort!

As I have mentioned, all efforts should be made to avoid shutting a service down. With each new Oracle version, fewer and fewer configuration changes require a restart to take effect. Many initialization parameters can be changed in a running instance. If you think you must restart, make sure you really have to.

But if you must shut down, how do you do it? Many DBAs issue a shutdown immediate, then wait. Shutdown immediate kicks users off and locks out new logins as soon as you issue it, but often takes a long time to finish in a large system. Usually, DBAs of large systems eventually resort to a shutdown abort. If this happens every time, why not just shutdown abort the first time?

The answer I consistently hear is that it is dangerous. This is certainly not so. Simply put, the best way to shut down an Oracle database is like this:

```
SQL> alter system checkpoint;
SQL> shutdown abort
```

In the few situations such as cold backups and version upgrades where consistent datafiles and controlfiles are required, you can do this:

```
SQL> alter system checkpoint;
SQL> shutdown abort
$ lsnrctl stop
SQL> startup restrict
SQL> shutdown immediate
```

Shutting down the listener prevents CPU resources from being used by processes forking while crash recovery is completing. This speeds startup time in some environments.

A common misconception holds that shutdown abort is somehow more dangerous or reckless than the other shutdown modes, and can result in data being lost. This is not the case. True, shutdown abort kills the Oracle background and shadow processes, and removes the shared memory segments. This may seem ugly, but aesthetics matter little when it comes to availability. The fact remains that **all committed transactions are intact in the online redologs**, even if the data associated with them has not been written to the datafiles. On startup, crash recovery applies online redologs beginning from the time of the most recent checkpoint. When was the most recent checkpoint? How about one second before you shut down?

When recovery is complete, the database is opened for use. Transaction rollback and temp segment cleanup (the two big time wasters for shutdown immediate) take place in the background after startup, when applications can already log in and do work. No user's time is wasted waiting for all uncommitted transactions to roll back.

Consider the diagram below comparing total outage times using abort vs. immediate shutdown on a typical instance.

When starting up after a shutdown abort, the amount of time spent in instance recovery depends largely upon how recently the last checkpoint was issued. By forcing a checkpoint immediately prior to issuing shutdown abort, the redo required to complete crash recovery and bring the database open will be minimal.

The alternative in an active environment to shutdown abort is shutdown immediate, but immediate shutdowns often take too long, rolling back transactions and cleaning up temp segments while precious seconds pass by.

Shutdown abort is useful for very brief downtimes, such as those required to change a nondynamic initialization parameter. In practice on Oracle instances with very large SGAs, such quick bounces can typically take as little as 30 to 40 seconds. In order to expedite planned shutdowns and startups, the same scripts that are devised for reliable and fast startups at machine boot and shutdown should be used for manual shutdowns and startups. The scripts should be used because they can issue commands faster and more accurately than you can type, and can be designed to resolve potential complications such as datafiles left in backup mode.

DDL changes on popular segments

Another commonly cited reason for having to restart is to perform DDL on heavily-accessed objects. When trying to apply a DDL change to some segments, a DBA may encounter the error ORA-00054: resource busy and acquire with NOWAIT specified. This can occur even if the DBA acquires an exclusive DML lock on the table using the lock table command. Unfortunately, locking a table does not guarantee the success of any subsequent DDL statement on the table or associated indexes. DDL statements must obtain a library cache lock in order to perform DDL, and there is no manual mechanism to guarantee possession of a library cache lock.

A common reaction is to restart into RESTRICT mode, perform the DDL, then reopen the database to applications. However, in most cases it should not be necessary to resort to these measures.

Sit 'n' spin

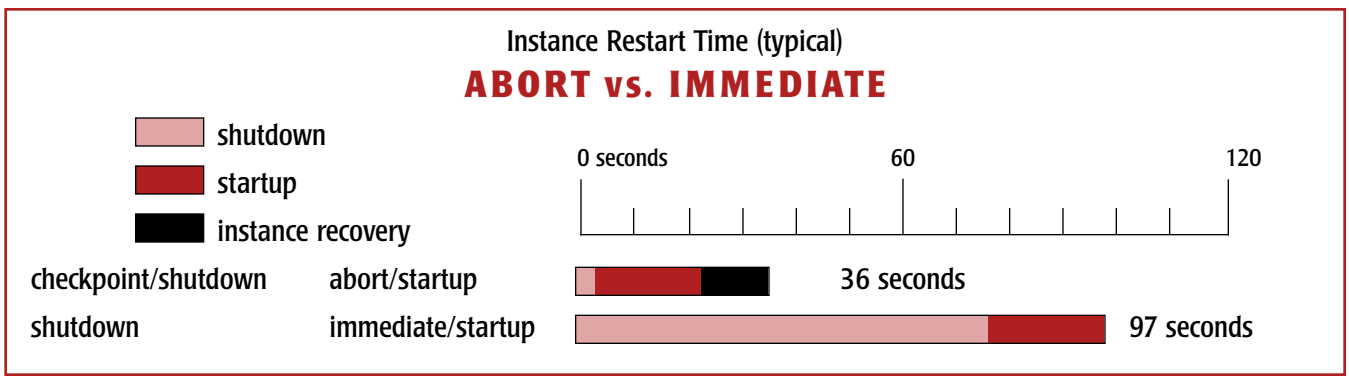
To get a DDL change to apply on such an active segment, write a script that loops trying to execute DDL, stopping only when it finally succeeds. In the unliklely event this tactic continues to fail, one might try disabling selected applications that heavily access the object in question for long enough to perform the DDL, thus leaving the database up and available to other applications.

Jonathan Lewis (http://www.jlcomp.demon.co.uk) wrote a simple stored procedure to do this, and it is included here with his permission:

```
create or replace procedure do_ddl(m_sql varchar2) as
   in_use exception ;
   pragma exception_init(in_use, -54);
begin
   while true loop
        begin
                execute immediate m_sql;
                exit;
        exception
                when in_use then null;
        end;
        dbms_lock.sleep(0.01);
   end loop;
end;
```

Online redefinition

Online redefinition is also possible in Oracle 9i and 10g using the dbms_redefinition package. This package creates an interim table to take DML and store data while the original table is being redefined. It is useful for a variety of activities that cannot be accomplished with a single DDL statement, such as moving or reorganizing a seg-



ment. For details on using this package, consult chapter 14 of the *Oracle Database Administrator's Guide 10g Rel. 1*. For most DDL, however, the PL/SQL spinning method is sufficient and avoids the unnecessary complexity of dbms_redefinition.

Upgrades and patches: Do them instantly!

Oracle patches and upgrades are another huge waste of downtime. Incredibly, many DBAs wait until after they have shut down an instance to get the upgraded or patched software ready. In almost every case, it should be possible to have upgraded software installed and ready before you shut anything down.

- **Upgrades and patchsets** are similar in that they are installed using the Oracle Universal Installer (OUI).
- **One-off patches** are applied with the opatch utility (or patch.sh prior to 9i), which is far less cumbersome than the OUI.

One-off patches

One-off patches are typically modifications to a small piece of Oracle's code to address a specific bug. On Unix operating systems, they usually consist of a new version of a single .o file that the opatch utility places into a .a file (usually libserver<version>.a for the database) in place of an old version using the unix ar command. Afterwards, opatch relinks Oracle using the Unix make command.

Many DBAs have had the unpleasant experience of taking an instance down to patch it, only to discover that there is something wrong with the patch or that Oracle will not relink. Many unnecessary minutes of downtime have been racked up while the DBA searches around for the path to the make command. Fortunately, it is not actually necessary to have the instance down while building a patched executable. Patch application really only requires a few seconds of downtime.

By making small modifications to the patch's configuration file, you can make it call an alternate makefile of your own to relink Oracle. By changing the name of the Oracle (or any other) binary in the alternate makefile, you can stage a new version of Oracle that contains your patch, without ever taking down the instance.

Preparing to relink Oracle on Unix without shutting down

- Find the Oracle makefile, $ORACLE_HOME/rdbms/lib/ins_rdbms.mk.
- Make a copy of the makefile with an alternate name, such as ins_rdbms_new.mk.
- Edit the file and find the makefile target that corresponds to the executable you want to relink. For the Oracle executable, the target is ioracle.
- Change the makefile for the target so that it creates a differently named executable from the one that is running. The following example is for the Oracle executable. Changes are in black:

```
ioracle: $(ORACLE)
-mv -f $(ORACLE_HOME)/bin/oracle_new
$(ORACLE_HOME)/bin/oracle_newO
-chmod 6751 $(ORACLE_HOME)/bin/oracle_new
```

Through 8i: Modifying patch.sh

Before 9*i*, one-off patches for Unix came with a patch.sh script that installs new modules into library files, and then calls make to relink. To make patch.sh ignore the running instance and call the modified makefile, make changes to the setup() section at the top of the file to ignore the running instance and call the alternate makefile:

```
SHUTDOWN=false
MAKE_TRIPLETS="rdbms/lib:ins_rdbms_new.mk:ioracle"
```

Now running patch.sh will create a new patched Oracle executable called $ORACLE_HOME/bin/oracle_new.

9i and up: Modifying etc/config/actions and etc/config/inventory

For Oracle 9*i* and higher, one-off patches on Unix come with two files under etc/config/:

Change the actions file to call your alternate makefile:

```
<make change_dir="%ORACLE_HOME%/rdbms/lib"
make_file="ins_rdbms_new.mk"
make_target="ioracle"/>
```

Change the inventory file to ignore the running instance:

```
<instance_shutdown>false</instance_shutdown>
```

Now running opatch will create a new patched Oracle executable called $ORACLE_HOME/bin/oracle_new.

The 30-second patch outage

With the successfully patched executable in place, a fast shutdown and restart are all that is required to apply the patch:

```
SQL> alter system checkpoint;
SQL> shutdown abort
$ mv -f $ORACLE_HOME/bin/oracle_new
$ORACLE_HOME/bin/oracle
SQL> startup
```

The Big Things

While many activities require either no or little downtime to perform, some do require major outages. For those, you can make the difference between a two-hour and a five-hour outage by planning and scripting carefully ahead of time. A selection of efficient approaches to major planned outages follows.

Moving a database without extended downtime

From time to time, especially in a rapidly growing or changing organization, the need arises to move a critical service from one server to another, either locally or over great distances. For those who do not wish to incur extended outages for such operations, there are number of strategies. Depending on the approach, micro-downtime strategies can be employed to move a database, change database block size, or split the contents of one database into several other databases. The following techniques can be used together, or individually to make planned maintenance shorter.

Floating (service-based) IP addresses

Oracle databases can be moved from one host to another more easily if a special IP address other than the host IP address is established for each Oracle service. That IP address can then be moved from one host to another, allowing a diverse application base to find the newly moved service. All Unix platforms on which Oracle is supported and also Windows support the configuration of multiple IP addresses on an network interface. On many Unixes, this can be accomplished with the ifconfig tool:

First, after any Oracle listeners are stopped, the IP address is removed on the original host:

```
root# ifconfig en0 stop 192.168.1.110
```

Then, after the database is made available on the other host, either by standby failover or restore from backup or any other method, the IP address is added to an interface on the destination host.

```
root# ifconfig alias le0 192.168.1.110
```

On 32-bit Windows, IP addresses may be similarly added and deleted via the advanced settings in the networking control panel. Good luck scripting this.

If you tried to move a service without moving a service-based IP address, you would have to update all client tnsnames.ora files manually wherever they exist to reflect the new machine's IP address. A less cumbersome change could be made using Oracle Names Server, but that is not guaranteed to cover all clients who may have set up a connection to the server unbeknownst to Names Server. For instance, programs that use the JDBC thin client have host and port hardcoded.

Service-based IP addresses are the native failover redirection method included in the vendor scripts for hardware cluster products such as HP MC Serviceguard, SunCluster, and Compaq TruCluster, where one node must rapidly assume the services of another. The technique is equally applicable to a variety of planned maintenance situations involving failover from one host to another. By including this method in standby failover scripts, the time spent fiddling around during a failover can be reduced.

Moving a database using hot-copy and physical standby switchover

A database can be moved to a new host almost instantaneously and with minimal interruption of service using physical standby switchover.

It is important to understand the difference between a physical standby failover and a physical standby switchover. In a switchover, the standby neatly picks up where the primary left off in the redolog sequence. No resetlogs is performed when opening the standby, so if the primary is shut down at SCN 12345, the standby picks up at SCN 12346. Prior backups remain valid, because there is an unbroken redolog stream that can still be applied to them.

In a failover, something prevents the DBA from obtaining all the logs from the primary before opening the standby. In this scenario, **actual transactions** that were successfully committed on the primary **are lost forever**. This procedure is never used as part of a planned outage, because it loses transactions and invalidates prior backups by performing a resetlogs when the standby is opened.

The first step to performing a switchover is to establish a standby on a secondary on the target host. Copy the database there while open and operating using tablespace backup mode or RMAN. As of Oracle 10g, RMAN can perform copies directly to a secondary host using the "RMAN Pipe Interface." Roll forward the database copy in standby mode, applying redologs. When the predetermined outage time arrives, shut both the old and new recovering databases down, and perform a graceful standby failover to bring the service up on a new host, using the following steps:

- Shut down the primary
- Copy the last few archived redologs from the primary to the standby
- Begin copying the online redologs from the primary to the standby
- Meanwhile, roll forward the standby until it has consumed all the archived redologs
- Copy the controlfiles from the primary to the standby
- Disable the primary by renaming the controlfiles
- Start up the standby using "startup mount"
- Issue a "recover database" command and apply the last few logs
- Open the database using "alter database open"
- Move the service-based IP address from the primary to the standby
- Start the listeners on the standby, and the service is now running on the target host

This same procedure can be accomplished using DataGuard role management services, by following these steps:

- Shut down the listeners on the primary.
- Make the primary into a standby. On the primary:

```
SQL> alter database commit switchover to
physical standby with session shutdown;
SQL> alter system checkpoint;
SQL> shutdown abort
SQL> startup mount
```

- Make the standby into a primary.

```
SQL> alter database commit switchover to
primary;
```

```
SQL> shutdown abort
SQL> startup
```

- Move the service-based IP address from the primary to the standby.
- Reconfigure DataGuard's log apply services so that the new primary's logs are shipped to the new standby.

If you plan to move the service back to the original host, you can replace the controlfiles on the old primary with standby controlfiles and keep it recovering the archived redologs from the newly opened service.

Many users of the standby feature do not realize how useful it can be for planned outages. The graceful failover method, where the database is opened noresetlogs, allows the former primary to be immediately pressed into service as a standby. This allows easy maintenance on one set of hardware (host and disks) while the database runs on the secondary host. Once maintenance is completed and the database on the primary server (now running as a standby) is caught up, the failover can be performed again, in the other direction, so that the database is once again running on the original server, and the standby on the secondary. This is often referred to as "role reversal" or "graceful switchover and switchback."

To perform a graceful failover to a running standby, you can write your own scripts or use DataGuard. Using my own scripts, I have performed very rapid failovers of one or two minutes total unavailable time for the service. Since some of the activities can be performed simultaneously and in parallel (such as copying the online logs while still recovering the standby), the total time to perform the failover can be kept very short.

When the service has been moved, it is important to disable the database on the old host by moving or renaming the initialization file or the controlfiles so that nobody accidentally starts it up, causing it to diverge from the carefully preserved redolog stream. DataGuard achieves implicitly this by converting the controlfile to a standby controlfile.

Low-Impact Reorganizations and Conversions

Some types of migrations between hosts of the same platform require changes to the physical datablocks comprising the database. Such reorganizations are not candidates for physical standby switchover because a physical standby must be block-for-block identical to the primary database. Some of the activities that fall under this category are:

- Block size changes
- Character set conversions

Replicated reorganization

There is a novel low-impact technique that can be used to make major conversions with minimal interruption. The technique uses multimaster replication or streams.

- First, establish a physical standby database of the source database on the target host, and get it rolling forward and caught up with the primary.

- During a brief planned outage, open the standby and establish multimaster replication between the tables on the source database and target database.
- As long as applications are not allowed to log into the target database, it can be shut down and reorganized using export/import. In the meantime, replication changes bound for the target database are queued up on the source.
- Once reorganized, allow the target database to accept all the queued replication changes until the databases are once again logically in sync.
- Point the users and applications at the target database, and remove replication during a second brief downtime.

Smoke and mirrors

As of Oracle 9i, multiple block sizes and multiple character sets are permitted in a single database. One technique for block size or character set conversion even simpler than above is to gradually move segments from tables of one block size or character set to tables of another block size or characterset using "alter table . . . move" and "alter index . . . rebuild." This can be done in a series of brief low-impact outages to one table or index at a time. If a very large table is to be converted, consider the following smoke-and-mirrors technique:

- Create an empty table with the new character set or block size
- Create a view that is defined as a UNION ALL of the source and target tables
- Create a synonym that points applications at the view instead of the source table
- Write a PL/SQL, Java, or OCI program that automatically moves rows from the source table to the target table in discrete transactions using row locking
- Upon completion, point the synonym at the target table, and drop the view and source tables

This procedure is largely incorporated into the Online Redefinition feature, available in 9i and 10g. For details on using this package, consult chapter 14 of the *Oracle Database Administrator's Guide 10g Rel. 1.*

A note about character sets

Many character sets are actually supersets or subsets of others. For instance, Latin 1 (WE8ISO8859P1) is a superset of US7ASCII. In such cases, no actual block changes are needed to convert from a subset character set to a superset. An old method that was popular before Oracle addressed this was to simply update the SYS.PROPS$ table and restart. This worked in many cases because the ASCII values were the same under the new character set. From 9i, it is possible to "alter database character set" For more information on the gotchas of character set conversion, see chapter 11 of the *Oracle Database Globalization Support Guide 10g Rel. 1.* ▲

About the Author

Jeremiah Wilton was the founding DBA and a 6-year veteran of Amazon.com's database group. A frequent speaker at the lecterns of US and overseas Oracle conferences, he enjoys sharing his novel approaches to availability and scalability with the world community of Oracle DBAs. He was a recipient of an honorary Oracle Certified Master certificate in 2000, and a keynote speaker for IOUG-A at Oracle Openworld 2000. Jeremiah now teaches seminars and classes to groups on crisis management and Oracle administration. He is also available for remote emergency DBA services. For more information, visit his website at http://www.speakeasy.net/~jwilton.

To be continued in the next issue of the NoCOUG Journal—*Part Two: Unplanned Outages*

TRAINING DAY

Training Day with Jonathan Lewis

Dear NoCOUG members and friends,

Jonathan Lewis will be speaking at the NoCOUG Summer Conference on August 19 at ChevronTexaco in San Ramon. He was able to include us in his timetable because he had a previous engagement in the USA which made it possible for him to stop over for one extra day in Northern California.

Unfortunately, the previous engagement had to be cancelled at short notice. Rather than disappoint the NoCOUG audience, Jonathan will fly direct from the UK to California for our event, and he has arranged to hold one of his one-day tutorials the day before the NoCOUG conference at a special budget price to cover the costs. We are pleased to make this training day opportunity available with a special discount for NoCOUG members.

- The tutorial is the extremely popular: **Understanding and Assisting the Cost Based Optimizer**
- The cost is $225 to NoCOUG members ($275 to non-members)
- The class size will be strictly limited to 30 places
- Location: The Marriott San Ramon, 2600 Bishop Drive, San Ramon, CA 94583
- Date: August 18, 2004

For details of the agenda, booking, and payment see: **http://www.jlcomp.demon.co.uk/tutorial.html#California**

This represents a great opportunity for NoCOUG members to participate in a full day of training led by Jonathan Lewis for a bargain price. The limit of 30 places gives you the opportunity to ask your questions and not get lost in the crowd—but register right away so that you don't get left out!

—Roger Schrag, NoCOUG President

Where's the SQL?

A perspective, implemented with Java

by Joel A. Thompson

In years past, we became accustomed to writing our SQL directly into our program. In my case, this was derived from my days as a "C" programmer where all we had was a proprietary API to access the database, and we were forced to write the SQL statements directly into our program. As I learned Java, I carried these habits into my Java programming. This was the way it was, until about five years ago when I took on a project for a biotech firm. I came into an existing project that was one of the best designed and best executed that I have ever been a part of. They had programming style guides, naming conventions, and they diligently comment coded everything. It was poetry in programming. The conventions and practices that I learned on this project have never left me, and hopefully I can convey one of these best practices adequately enough so that you, too, can get a sense of awe and respect for poetic-programming that leads to a flexible, high-performing, maintainable, modular, portable, and—last but not least—better-quality software. One of the best design choices that I learned on the project was where they placed all their SQL code in the database as procedures and functions.

In my opinion, they designed their system to have the SQL in the database layer for the following reasons:

1) Performance gain: This might be the best reason to put all your SQL into a procedure/function call to the RDBMS. If you can bundle your SQL statements and pass a few arguments to the procedure, then you have saved on fewer calls to the database. An example of this would be when you are using a manufactured primary key in your table. You'd have to issue a call to get the next manufactured ID (generally a sequence number), insert your parent record with the new ID, then insert each of the child records with the new primary key being foreign key (FK). In this situation you would have programmed in three calls to the RDMBS, which can be cut down to **one**!

2) Flexibility: Basically we are talking about performing many SQL statements coupled with logic processing within on call to the RDBMS. Wouldn't it be nice if I could query some set of data, do some logic processing on that set, then query another set of data that is the actual data that I am looking for? With Oracle's PL/SQL, you can do just that. By example, I could create a global temporary table (GTT) (preferably at installation time), and then in my procedure I could query data, insert into this GTT, select records out of the GTT, and perform some logic/rules operations on this data. Finally, I can build up my result set based on the data in the GTT and return to the calling program. (See Listing 1.)

Joel A. Thompson

3) Easier to manage: You don't have to recompile/redistribute your executable if/when you change your database code or even when you make changes to the data definitions. Let's say you want to add a new table join in your SELECT statement. If you put your SQL in your program, this would require that you modify your FROM clause, recompile your Java, and redistribute your executable to your clients. Instead if you called a procedure from your program, all you'd have to do is edit your PL/SQL code and reload it into your database, and you're live.

4) Modularizes database code and cleans up code: Generally speaking, and especially with JSP programming, things get messy when you have SQL embedded into your program. In fact, JSP best practices suggest you build everything into tags and beans that completely hide the Java and SQL from the HTML designer. In this case, consider building your Java BEANS that have procedure calls instead of SQL statements.

5) Portability from schema to schema: The biotech firm wanted their same user interface to work against a completely different database structure. This is where I came in. They asked me to rewrite and remap all the SQL that accessed the old database to work against the new database. This would have been a nightmare and a huge effort if all the SQL was spread throughout their EJB/Java code. Instead, because they designed the SQL into procedures, all I had to do was to rewrite the PL/SQL. I was able to create views, perform logic operations, convert datatypes (like Long to BLOB/ZIP), authorizations, and other advanced PL/SQL techniques completely within the PL/SQL procedures. We made no changes to the EJB/Java tier, and the user interface worked as intended against a completely new schema.

6) Don't want to get hooked into Oracle RDBMS-only solution? Oracle is a powerful database. It is the industry giant. It costs a lot of money, and for a good reason; it provides you all the flexibility, performance, and program-

ming tools needed to get a job done well. I urge you to drop the requirement that your program must work against any database in the future. Unless you are writing a tool that must work against any RDBMS on the market, there is very little reason to hamstring your development efforts with this requirement. Sure PL/SQL is Oracle-only, but it is an amazing development tool and should be used to its fullest capabilities. As a matter of fact, since the PL/SQL is written in one place, namely the Oracle database, if you wanted to change to another vendor, all you'd have to do is rewrite your procedures with that vendor's syntax. One warning: Make sure that the other vendor supports the same functionality that Oracle's PL/SQL provides.

7) Standards compliant? Wouldn't it be great if the JDBC API actually allowed me to make a procedure call with a standard calling convention? Then the implementer (the database vendor) would take care of calling the procedure with the correct syntax. That is correct—JDBC 3.0 defines how to place a procedure/function call. All you have to do is define the procedure in your database and, voilá, you have the best of both worlds: portability and optimization. Unfortunately, Oracle's implementation of JDBC standards is a bit behind, and we are forced to use some Oracle specific type definitions. (See my notes in the code sample provided.)

In conclusion, I urge you to write your SQL statements directly into PL/SQL and call the procedure or function from your Java program. You will gain all the advantages that I mentioned earlier and in the end will create better-quality software. ▲

Joel Thompson is a development manager and software programmer. His favorite technologies (and those he has the most experience in) include Java, C++, HTML, JavaScript, EJBs, JSPs, Servlets, Perl, Shell, and Oracle Database Systems on UNIX and NT platforms. Joel founded RHINO Systems, Inc., in 1997, after working nearly five years for Oracle Corporation as a manager and product developer. He has led many successful website and product development projects and holds a master's degree in computer engineering. Joel can be reached at: joel@rhinosystemsinc.com.

Listing 1:

Java code to call the procedure

```
import java.util.Properties;
import java.sql.Connection;
import java.sql.CallableStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
…
ResultSet rs =null;
// this jdbc 3.0 did not work with Oracle's 9i implementation
classes12.zip)
/* cstmt=conn.prepareCall("{CALL GET_RESULTS(?)}");
cstmt.setLong(1, 1);
rs= cstmt.executeQuery();
*/

final int cursorRefType=oracle.jdbc.driver.OracleTypes.CURSOR;
String sql="{ ? = call get_results(?)}";

cstmt=conn.prepareCall(sql);
cstmt.registerOutParameter(1,cursorRefType);
cstmt.setLong(2, 1);
cstmt.execute();

rs=(ResultSet)cstmt.getObject(1);

// process the results
while (rs.next()) {
    System.out.println("rs.getString(\"ID\") = " + rs.getString("ID"));
    System.out.println("rs.getString(\"NAME\") = " + rs.getString("NAME"));
 System.out.println("rs.getString(\"PHONE\") = " + rs.getString("PHONE"));
}
rs.close();
cstmt.close();
// you must commit to clear the GTT or close your connection
(consider problems with connection pooling)
conn.commit();
```

PL/SQL code demonstrating the use of logic processing and global temporary tables

```
create or replace package RHINOSYSTEMSINC
AS
   type t_ref is ref cursor;
END;
/
create table person (id number primary key, name varchar2(32));
create table phone (person_id number, phone varchar2(32));

create global temporary table gtt_person_temp (
   id number,
   name varchar2(32),
   phone varchar2(32))
ON COMMIT DELETE ROWS ; /* could use ON COMMIT PRESERVE ROWS */

create or replace function GET_RESULTS(name varchar2) return
RHINOSYSTEMSINC.t_ref
is
   v_ret_cursor RHINOSYSTEMSINC.T_REF;
   cursor c_people is select * from gtt_person_temp;
   v_Rec gtt_person_temp%ROWTYPE;
   BEGIN

   insert into gtt_person_temp select id, name,phone from
person where upper(name)
   like upper('%' || name || '%');

   for v_Rec in c_people loop
        if v_Rec.name = 'joel thompson' then
        insert into gtt_person_temp values(100,'Joel Thompson
        was found','111-222-3333');
   end if;
   end loop;
      OPEN v_ret_cursor FOR
      'select * from gtt_person_temp ';

   RETURN v_ret_cursor;
END;
/
```

The Latest News at IOUG!

Kimberly Floss was just re-elected to the International Oracle Users Group and serves as the IOUG president. The IOUG is an international organization affiliated with more than 100 separate local and regional Oracle user groups, including NoCOUG. Kim was gracious enough to take a few moments to answer our questions.

How large is the IOUG, and how many people are on the Board of Directors of the organization?

Kimberly Floss: The IOUG Board of Directors is comprised of 10 volunteers who are leaders in using Oracle technologies, reporting to a membership of more than 7,000 individuals and corporations.

What have been some of the challenges you've faced during your term as IOUG President?

Kimberly Floss: Certainly the current state of the economy has been challenging. Many folks have been let go from their current positions, and companies have been reducing or eliminating budgets for training and/or user group memberships. Ironically, it is in these times that folks truly need to be part of a user group—networking is critical, now more than ever.

You obviously enjoy contributing to the Oracle community. What have you enjoyed the most about your involvement?

Kimberly Floss: The best thing about being involved in the IOUG community—the interaction with others, and the things that I learn. It helps me to keep my technical skills sharp, and also I've made lots of friends and built lots of relationships.

What kinds of things can IOUG members look forward to in the coming year?

Kimberly Floss: We just had our conference, IOUG Live!, last month in Toronto. We will also probably have several regional training events, although the dates have not yet been determined. But if anyone is interested in something in their area, please let us know. I am partial to Las Vegas, but that is just me—I don't get to pick the cities.

Does the IOUG have anything in store related to collaborating with the local user groups?

Kimberly Floss: The IOUG is always looking to collaborate more with the local groups. I personally am involved with both the Chicago Oracle User Group and the Midwest Oracle User Group. But we are always looking to help get speakers, provide grant funding, do events together, and in general, for ways to partner to benefit all Oracle users.

Tell us a little about yourself. How did you initially become involved with IOUG?

Kimberly Floss

Kimberly Floss: Here I was at my desk, when the phone rings, and on the other line is Rich Niemiec [past IOUG Board president and author]. I was shocked, as I had his performance and tuning book on my desk at the time, and I was a little starstruck. He asked me if I was interested in becoming more involved—I was so excited that he was calling me asking me to be more involved that, of course, I said yes.

Would you encourage others to volunteer, either at the local or national level? If so, why?

Kimberly Floss: I would encourage everyone to get more involved, at any level. The rewards are so much greater than the time and effort that you put in. The networking, the friendships, the technical knowledge—I have gained so much. I wish I could speak at more conferences, to give more back, the masters have given so much to me . . . I will always wish I could do more.

How did your career lead you to where you are now? What, specifically, prompted you to get involved with Oracle technology?

Kimberly Floss: I have been working with databases for almost 15 years. I guess you could say I just love the technology. While I had worked on some other database platforms, I had a burning desire to master Oracle as well. So when the job opportunity came, I took it.

What are your favorite pastimes/hobbies?

Kimberly Floss: I have no more hobbies—I have a new baby, and he has usurped all of my hobbies. I still teach part-time, at night at a community college, though—SQL programming and database administration. Also, I am currently writing a book, *Oracle SQL and Index Internals*, due out by Rampant Publishing later this spring. And I just finished writing an experience-based Oracle certification exam for ITC2. But when I have spare time, I do enjoy reading and shopping.

What book are you currently reading?

Kimberly Floss: I am reading two—one fiction book and one Oracle book—I always read two at a time, depending on my mood. I have an hour ride on the train each way, so I have plenty of time to read or sleep. I am reading Nora Roberts' *Birthright* and *Oracle Performance Troubleshooting* by Robin Schumacher. ▲

Interview conducted by Lisa Loper, NoCOUG Journal *editor.*

The Problem We Fixed by Doing Nothing!

By Chris Lawson

An East Coast wheel manufacturer uses the "Alligator" Order processing system, with Oracle 8.1.7.4 as the database repository. Alligator coordinates all order-related information sent to the customer, such as order confirmation, backorders, and notification of sales events. Naturally, the Alligator application is critical to the business, and any service interruptions are quickly escalated to upper management. For our part, we DBAs were careful to monitor database performance, so that we could head off any performance problems before they became critical.

One day a DBA noted a potential problem in a large table. Analysis showed that the huge *WHEEL_DATA* table (300 million rows) had many chained rows. While not a serious problem yet, we believed that the row chaining was probably slowing down queries, as well as causing the table to consume more disk space than necessary.

We performed a reorganization of this table using a simple export and import. This maintenance took most of the weekend due to the huge size of the table. After the import was completed, the application was restarted, and the users were notified. Although the database maintenance seemed to go smoothly, we quickly discovered that there were big performance problems. We began to receive urgent calls and emails containing alarming news.

Crisis: Day 1

The users reported that routine functions seemed to take forever. Doing some simple queries on the database, we quickly spotted a bizarre problem: The Alligator application had launched 22 sessions, and each one was updating every single row in the *WHEEL_DATA* table! Obviously, Alligator was doing something terribly wrong, but why? We had changed nothing in the application. We had only changed one database table.

After investigation, we discovered that many SQL statements—even simple queries—appeared to be stuck for no reason.

We called Alligator support and asked them why the application was behaving so wrongly. The analysts suggested, "The table is corrupted." This suggested cause seemed impossible—how can a bad database table cause the application to change the SQL statements it runs?

The Alligator support analyst provided a hint: He said that the "metadata" for the database was wrong. After further inquiry, Alligator explained that the application looks at the Oracle data dictionary, and uses this information to build SQL queries. That is, the application *changes* its SQL depending on the design of the Oracle tables!

We compared the production database to the development database in order to find any differences. We were thinking that perhaps we missed an index on the big table. We were pretty close—we discovered that the *WHEEL_DATA* table no longer had a primary key. (This happened because the export had accidentally been performed with the option "Constraints = N.")

The missing constraint was the key to solving the (first) crisis. Since we had overlooked the primary key on the *WHEEL_DATA* table, the application chose to not include any *Where* clause.[1] After we added the primary key, the application no longer issued SQL without a *Where* clause—but that wasn't the end of the problems.

Crisis: Day 2

With the primary key fixed, we all thought the problems were over. Sadly, we were all wrong. Although we no longer had queries missing the *Where* clause, the Alligator users reported that their jobs still ran slowly.

After investigation, we discovered that many SQL statements—even simple queries—appeared to be stuck for no reason. The problem was not just for one or two SQL statements. Instead, many queries ran badly, for no apparent reason.

Runaway *smon*

We noticed that *smon* seemed to be very active, using 100% of one CPU. We didn't know, however, if the high *smon* activity was a symptom of some other problem, or if it was the problem. We also noticed that the redo logs were switching rapidly. This seemed very odd, since most sessions executed queries, not transactions. Thus, there should have been little transaction logging.

Finally, we found a hot backup in progress. Of course! That would explain the massive log switching. We quickly killed the hot backup and issued the command Alter

[1] We'll let the reader decide if dynamic SQL generation based on Oracle's data dictionary is a good idea.

Tablespace . . . End Backup to restore the tablespace to normal status.

Surprisingly, stopping the backup job did not solve the performance problem! After some discussion, we decided to focus on the only change that we had made—the *WHEEL_DATA* table. Tests revealed an interesting fact about this table: Queries on the table took many times longer than normal. For instance, scanning the first million rows required 1 million reads! In contrast, a scan of a similar large table required only 6,000 reads.

Moving the bad table

Since we couldn't figure out what was wrong with the *WHEEL_DATA* table, we wondered if we could somehow quickly move the table to speed things up. A quick test showed that if we copied the first million rows to another table, the query on the second table would run fine. This suggested a possible resolution: If moving rows worked well for 1 million rows, then it would probably work well for the entire table. Thus, we started a transaction to Create Table as Select . . . with the NoLogging option.

Unfortunately, the slow table reads also made moving rows impractical. Watching the transaction progress in the V$Session_Longops view, we soon realized that moving all the rows would take nearly 24 hours to complete!

Having run out of options, we asked the administrators to begin retrieving the tapes to restore the database to the "pre-reorg" state—"just in case."

Crisis: Day 3

On day three, we regrouped and focused our efforts on analyzing *smon*. Why was *smon* consuming so many resources? We confirmed that even with no sessions whatsoever, *smon* consumed close to 100% of one CPU.

We considered two possible causes for the *smon* activity: *smon* was coalescing free extents or it was cleaning up temporary extents. If it was doing either of these two things, we should be able to confirm it via a few simple queries. In particular, we could watch the total number of free and temporary extents. Here are the queries we used:

```
Select Count (*) From Dba_Free_Space;
Select Count (*) From Dba_Extents Where Segment_Type='TEMPORARY';
```

TECH TIPS

Oracle Performance Management

If you're looking for some resources on Oracle peformance management, you may want to check out the white papers available on Quest Software's website. There are a variety of topics to choose from. All of them are available for free, though some require you to provide some information to gain access to them. Here are some sample topics:

- Optimizing Oracle 9*i* Instance Memory
- Using Oracle's Redefinition Package for Performing Online Reorganizations
- Back to Basics: Assets, Liabilities, and the Bottom Line in RDBMS Application Development Lifecycles
- Data Modeling: Common Mistakes and Their Impact
- Learn How to Partition in Oracle 9*i* Release 2
- Oracle Database Performance Management
- Resolving Oracle Latch Contention

Check out the entire Oracle white paper listing at: http://www.quest.com/content/list.asp?ContentTypeID=1&Format=table&nav=%2Fsolutions%2Fnavigation%2Exml&cat=8 ▲

Oracle support agreed with our conclusion that we were witnessing a monstrous rollback. This was comforting to know, but didn't really help our users.

The query results revealed that each value was small (and unchanging). This was a surprising result and showed that we didn't yet know what *smon* was doing. If *smon* wasn't cleaning up extents, what was it doing?

The V$Wait facility

To answer the *smon* question, we turned to the V$Wait Event facility. Recall that this facility reveals what is holding up any session. Queries on V$Session_Wait showed that *smon* was always waiting for sequential reads. We were able to narrow this down further by using the *P1-P3* parameters. Remember that the *P1-P3* parameters provide more detailed information for a particular wait event. In the case of reads, *P1* shows the file number, and *P2* shows the block number.

Given the file and block information, it was easy to find out exactly which object was being queried. It turned out that *smon* was reading these objects:

```
95% of the waits:   Rollback Segment R07
4% of the waits:    Index #2 on the table WHEEL_DATA.
1% of the waits:    Table WHEEL_DATA
```

The above findings were critical to unraveling the mystery. There should have been zero activity in the rollback segments, since the Alligator application had been turned off. Nevertheless, session-level statistics for the *smon* SID confirmed that *smon* was constantly generating rollback records. We also knew what the rollback affected because we saw wait events on the *WHEEL_DATA* table and index. Thus, the rollback must be due to some aborted transaction on that table.

Closing in on the root cause

One possible reason for a huge rollback related to the recent import of the *WHEEL_DATA* table: Had some part of the import aborted at some point, leading to a massive rollback? This idea could not be supported, however. Repeated checks of the import log for the *WHEEL_DATA* table showed that there were no aborted or failed transactions; the import had run just fine.

Finally, we remembered an important set of events from day 1 of the crisis. On day 1, the application had gone berserk because we had missed the primary key on the *WHEEL_DATA* table. When the application was first started, 22 sessions tried to update every row in the *WHEEL_DATA* table. Eventually, all 22 sessions had to be killed—leading to the massive rollback that caused *smon* to work so hard.

We realized that solving the first crisis (by killing the sessions) had led to the second crisis (huge *smon* activity.) Unfortunately, understanding the cause didn't automatically correct the problem.

The Crisis Comes to a Close

Oracle support agreed with our conclusion that we were witnessing a monstrous rollback. This was comforting to know, but didn't really help our users. Was there some way we could speed up the rollback? One idea was to change the init.ora file to do the rollback using multiple threads. This suggestion, however, was not feasible. We couldn't use any form of parallel rollback because we were using the Standard version of Oracle—not the Enterprise edition.

Looking at the session statistics for *smon*, we estimated that a total of 20 hours would be required to accomplish the rollback. Of course, the rollback had been ongoing for almost an entire day already, so we didn't know for sure how much longer it would take.

Fortunately for us, while monitoring the progress of the huge rollback, it completed on its own. Immediately after it finished, sample queries of the large *WHEEL_DATA* table showed a huge improvement. We notified the users, who restarted the Alligator application.

Epilogue

Once again, we DBAs had solved a perplexing mystery. Everything was back to normal, and the users were happy (at least until the next crisis). Our skilled analysis and sound judgment led us to the root cause of the problem and a restoration of good performance.

Of course, we didn't discuss our solution in great detail. After all, this was the problem we fixed by doing nothing. ▲

Chris Lawson is an Oracle consultant, writer, and editor of the OracleMagician *magazine (www.oraclemagician.com). Chris is also the author of* The Art and Science of Oracle Performance Tuning *(Apress), which describes performance tuning techniques in great detail. You can email Chris at: Chris@OracleMagician.com.*

Comedy Corner—Adding Some Fun into Work

By Laurie Robbins

Working in the technology field can seem mundane at times. Or maybe it's just working in general. So from time to time, it's a welcome diversion to come across something really funny.

For those DBAs out there, have you read Don Burleson's article **"What Type of DBA Are You? Tales From the Trenches"**? It's a fun look at three types of DBAs based on behavior and style of working. Here's the link:

http://www.dba-oracle.com/art_dbazine_what_kind_ dba.htm

And for you developers out there, have you written code using the steps outlined in **Code Like a Klingon**? Or maybe you have worked for a company that uses this standard of programming. It's great fun to read the entire list, but my favorite is #3. I found several versions of this on the web, but here's a link to a funny Top 10 list.

http://www.baetzler.de/humor/klingon_programmer.html

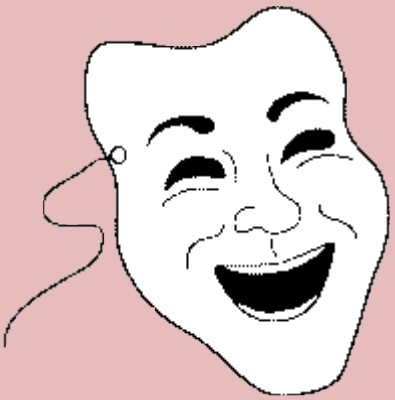
Have you heard the phrase **"Controlling developers is like herding cats"**? Several years ago, EDS took this statement to a new level, using it as the basis for a Super Bowl commercial. You can watch the commercial here:

http://www.superbowl-info.com/bin/edscatherders.mov

or download it here:

http://www.candlelightdreams.com/videos/

Enjoy! ▲



For Anyone Who Needs to Do Hands-on SQL Tuning

Book reviewed by Brian Hitchcock

SQL Tuning—Generating Optimal Execution Plans
by Dan Tow; published by O'Reilly, 2004

Summary

Overall review: Excellent
Target audience: Anyone who needs to do hands-on SQL statement tuning.
Would you recommend to others? Yes, if they need to do SQL tuning.
Who will get the most from this book? Readers should have a general working knowledge of SQL syntax and be aware of the high-level issues involved with SQL tuning.
Is this book platform specific? Yes and no. The overall approach is not platform specific, but the examples provided cover Oracle, SQL Server, and DB2.

Overall Review

I had been asked to review the SQL for a 17-table join that was running slowly in a CRM system. My task was to verify whether Oracle was properly optimizing the SQL statement. Therefore, I had a very specific need to find information relevant to SQL statement tuning. While I am experienced with the overall concepts of SQL tuning, I had never been totally comfortable with the way it is generally done. My experience is that after a lot of analysis, the result is a lot of big words that cover possible causes and possible solutions and maybe this and maybe that, but nothing really clear-cut as to what's good, what's bad, and what could be fixed. I was at the bookstore (yes, a physical bookstore!) looking for books on a different database topic when I saw this book on the shelf. I assumed it would be nothing more than the usual, generic review of all the possible issues with SQL statements running in commercially available RDBMS products. I read the back cover of the book first and quickly scanned over what I felt was the usual stuff, until I found the following statement: "How do you decide which execution plan a query should use?" I've never really had a good answer to this, so I read further.

This is exactly what I think has been missing in the past. Namely, some conclusive evidence that we're doing the right thing. Theory is fine, and theory gets a lot of print, but you'd really like to deliver a concrete answer to a specific question. This book promised to teach me how to do just that. Most of the SQL tuning information I'd seen over many years tends to focus on the mechanics of what the Oracle optimizer was doing and/or the details of SQL syntax. However, the answer I really want is how to determine that what I'm getting from the database is the best possible solution.

The Technique You Will Learn

The technique author Dan Tow provides is summed up on the back cover of the book as a diagram-based method for tuning SQL statements. This means that you look at the SQL statement, then generate a diagram of the tables involved and how they interact. You then examine some simple statistics about each of the tables. From the diagram and the statistics, you can quickly see which tables are the most important to the performance of the query. You then compare the results from your diagram with the plan that your database optimizer has chosen. For Oracle, this means comparing your diagram with the output of the explain plan. From this comparison, you can tell if the database execution plan is the best that can be done.

This technique sounds too good to be true, but it worked for me. Several advantages of this technique (besides the fact that it works!) need to be highlighted. First, you generate the diagram from the SQL just the way it was given to you. This means you don't have to have any insight or years of experience or intuition to apply the technique. You simply take the SQL and move through it to build the diagram of the tables and the interactions.

Second, you gather some simple statistics about the tables, such as row counts and the selectivity of indexes. You use these numbers to compute some ratios that you apply to your diagram. Again, no need for world-class expertise—the author explains exactly how to gather the statistics and how to compute the ratios needed.

Third, with the diagram drawn and populated you can quickly see which tables are defining, or limiting, the performance of the query. This means you can stop worrying about most of the tables and focus on the one or two that are really worth your time. You will know where to apply your time to get the greatest performance benefit.

And finally, note that you don't need full DBA access to the database. This is a real issue for production systems. The people who know the application and the schema best don't always have full access to all parts of the database. This technique only requires database access to the schema that owns the objects in the query being reviewed. You don't need to have access to system tables to apply this technique.

This technique also saves time in that you don't have to worry about all the possible combinations (join order) of tables that you might have to look at for a complex query. The order in which tables are joined has a big impact on the performance of the query. For my 17-table join, the number of possible join orders is huge. Applying this approach eliminates almost all of the tables as being the cause of performance problems and usually (almost always) identifies the one table that must be accessed first to get the best possible performance.

Your users will also like this technique because it cuts through the techno-babble that most technical persons feel compelled to provide to users. End-users don't really care that there are many millions of ways to join 17 tables. They do like getting a single diagram that shows all the tables (this makes it look like you did at least look at the entire query!) and which one or two are the most important for performance. It is more meaningful to them when you can provide a simple explanation as to why one table is so important to performance and all the others aren't. This also means you can answer many related questions all at once without spending any more time. For example, once I had analyzed the 17-table join, it was trivial to answer questions from the developers as to the effect of joining one more (or less) table. Again, saving time and leveraging your analysis are great benefits of this approach. Typically, an application will have groups of queries that are run quite frequently. Once you have diagrammed these common queries, you can answer many more questions about related queries without repeating most of your work.

Chapter by Chapter

Chapter 1 covers why you need to tune queries, something most NoCOUG members already know.

Chapter 2 reviews tables, indexes how joins work, etc. Again, it's basic background, but some of the diagrams are cool.

Chapters 3–4 cover how to generate execution plans and how to change them by changing the query or the indexes and so on. These chapters include specific sections for Oracle, SQL Server, and DB2. This is very good because it ties the general method back to the specifics of the syntax of each of these products.

Chapters 5–6 are, I think, the really good parts. If you don't read anything else (other than my review, of course), you should read these two chapters and then start looking at your specific problem queries. Chapter 5 starts with applying the technique to simple queries giving lots of detail on how this is done. A more complex query follows. These examples and the exercises are great. This is where I determined what I did and didn't understand. The author included solutions to the exercises he provided, and I used them. I strongly recommend that you do the exercises and make very sure you work through the solutions in detail. Again, this is where I learned what I needed to do to apply this technique. Chapter 6 then explains how to verify what the best execution plan is. This means you take the diagram of the query (chapter 5) and you move through it to determine the best way the query can be executed. Some of the special cases in this chapter were way beyond me, but that's okay. They are worth reading just so you can see how well this technique works.

Chapter 7 is where the author delves into unusual situations that I simply haven't seen in my work. To be very clear, the material is great, and I'm sure it's all very valuable, but I haven't needed to use most of it. This isn't a criticism of the material. It is great for those queries that have these unusual structures. Thankfully, I haven't had to deal with them. The basic technique works almost all of the time. For example, I haven't needed to understand cyclic join graphs or diagrams with multiple roots, but you might need to. Also, the section on generating diagrams for subqueries is good, and I did understand that part. I also got useful information from the sections on outer joins (my query had lots of them) and views. I would focus on the most useful chapters that I've identified above so you can quickly generate the diagrams of your queries. If your diagrams contain anything other than a single line between two tables, then I would review this chapter looking for help with your specific query.

Chapter 8 is a reward of sorts. After applying the technique, the reason it works is explained. That is another appealing feature of this author's technique, I can understand why it works. This chapter is brief, to the point, and very worthwhile. This should give you confidence that this approach will work in almost all cases.

Chapters 9 and 10 offer solutions to "special cases" and "seemingly unsolvable problems"—good titles, I think. I read these chapters, but I haven't seen anything like these situations in my work. I suggest you skim the section headings looking for something that applies to you if the basic technique doesn't provide a complete answer for your queries.

Appendix A has the solutions to the exercises that I've already told you to use.

Appendix B is great; it covers another fairly complex example from end to end. I strongly urge you to go through this.

What Could Be Better

Well, okay, you liked the book, but you have to say something critical just to show your readers how smart you are (and prove that it's not your brother who wrote it)—right? Well, after much review I can offer only a few minor criticisms. The technique that I learned from this book worked for me. But I didn't try a large number of queries, so it is possible that I just got lucky. I can't offer you a statistical study of thousands of queries that were analyzed and whose performance was measured to prove the effectiveness of this technique. Still, I know from my years of expe-

rience that the results I provided my users with after applying what learned from this book, were substantially better than what I used to provide.

Further, I'm sure that someone somewhere has a query that will cause this technique to break. After you read and apply this approach, I think you will agree that it works very well for the vast majority of queries that really come up. More important, most of us spend most of our time looking at queries that are pretty straightforward. My 17-table join is very common in modern commercially produced applications.

I would have liked more examples. Yet the examples shown in the book and the detailed analysis of them that is presented allowed me to work through my real-world query on the first try. There had to be a balance between the number of examples covered and the detail presented for each one. I found that this book provided just enough examples for me to learn the technique, but not so many that I never got to the end. And in the end, I was supposed to learn a technique. The number of examples that are presented in detail were enough to help me do that.

The Author

Dan Tow worked for Oracle for many years in the Applications group focusing on performance. This experience exposed him to many sets of SQL that needed to be examined. He has since started his own consulting business providing SQL tuning services.

Conclusion

Overall, this book delivers on its advertised promise. I actually read every page in this book; I didn't just skim the book to do a book review. The book is also blissfully short. I was able to read all of it in about four evenings, without missing any of my all-important television. Too many authors feel the need to write 600 pages (I should know . . .) even if they have to include copies of large sections of the Oracle documentation. And as mentioned above, I had a real need to find answers quickly for people who were paying me to support them. Therefore, I read this book with a very critical eye. And in the end I was able to provide my users with specific answers to their questions about the specific SQL they asked me to review. Best of all, they actually think I know something, which is always a pleasant surprise! ▲

About the Reviewer

Brian Hitchcock has worked at Sun Microsystems in Newark, California, for the past nine years. Before that he worked at Sybase in Emeryville, California. Even further into the past he spent 12 years at Lockheed in Sunnyvale, California, designing microwave antennas for spacecraft. He supports development databases for many different applications at Sun, handling installation, tuning, NLS, and character set issues as well as Unicode conversions. Other interests include Formula One racing, finishing a Tiffany Wisteria lamp, Springbok puzzles, Marklin model trains, Corel Painter 8, and watching TV (TiVo rules!). Brian can be reached at brian.hitchcock@aol.com or brhora@aol.com. ▲

Editor's Note: NoCOUG members receive a 20% discount on O'Reilly, No Starch, Paraglyph, Pragmatic Bookshelf, and Syngress books and O'Reilly conferences. Just use code DSUG when ordering online or by phone (800) 998-9938. http://www.oreilly.com/

**** Free ground shipping is available for online orders of at least $29.95 that go to a single U.S. address. This offer applies to U.S. delivery addresses in the 50 states and Puerto Rico. For more details, go to: http://www.oreilly.com/news/freeshipping_0703.html*

NoCOUG Summer Conference

(continued from page 26)

tion regulations (such as Sarbanes - Oxley, HIPAA, etc.) is a pivotal challenge for enterprises today. Learn how to implement a best practice in Data Lifecycle Management (DLM) to manage compliance with these regulations while maintaining optimum performance.

Deploying Oracle Databases on Networked Attached Storage
by Akira Hangi

This presentation will introduce Networked Attached Storage as a production storage platform for Oracle database implementations. With NAS, clients on a network can share information on a storage device via a file server. We will cover best practices to deploy an Oracle 11i Application on NAS storage.

Leveraging Master-to-Master Replication for Efficient Workload Balancing and Data Distribution
by Mark Ketchie

This session will discuss how to reduce system outage time and costs by easily and cost-effectively distributing large Oracle data volumes and workloads across a number of masters and distributed database applications.

Integrating OpenSource and Oracle 10g Spatial with 9iAS Technology
by Tony Lopez

This presentation includes a hands-on approach to creating a secure, available, and scalable geospatial application utilizing the power of Oracle 10g and 9iAS as well as various OpenSource technologies, including Eclipse, Jasper, and Java Server faces. ▲

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 membership dues. If your company is able to offer sponsorship at any level, please contact NoCOUG president Roger Schrag at rschrag@dbspecialists.com. ▲

Long-term event sponsorship:

- Lockheed Martin
- ChevronTexaco
- Oracle Corp.
- PG&E

Thank you! Year 2004 Gold Vendors:

- Churchill Software
- Database Specialists, Inc.
- DataMirror
- Embarcadero Technologies
- EMC Corporation
- Mission Critical 24/7
- Quest Software, Inc.
- Veritas

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

TREASURER'S REPORT

Judy Lyman, *Treasurer*

Beginning Balance		
April 1, 2004		**$ 77,615.96**
Revenue		
Membership Dues	5,315.00	
Meeting Fees	140.00	
Vendor Receipts	5,000.00	
Training Day	4,420.00	
Advertising	---	
Interest	20.59	
Total Revenue		**$ 14,895.59**
Expenses		
Regional Meeting	4,407.61	
Journal	5,613.48	
Membership	---	
Administration	2,303.71	
Website	---	
Board Meeting	667.94	
Paypal	147.81	
Miscellaneous	12.00	
IRS	170.00	
FTB	---	
Total Expenses		**$ 19,496.28**
Ending Balance		
June 30, 2004		**$ 73,015.27**

NoCOUG Summer Conference

Thursday, August 19, 2004

Session Descriptions

Editor's Note: The following are brief descriptions of session descriptions. For more detailed descriptions and up-to-date information, see www.nocoug.org.

TRACK 1

Strategies for Gathering Statistics
by Jonathan Lewis

Statistics are important to the Cost-Based Optimizer, but how often should you regenerate them, how big should your sample size be, do you need histograms, and what can you do when the statistics collected never seem to be good enough? In this presentation, we use a simple working example to show how too many people spend too much (machine) time gathering excessive statistics when they don't really need to.

The Evolution of Optimization 8i to 10g
by Jonathan Lewis

With each release of Oracle, we see more and more options for producing faster code. This presentation looks at SQL optimization from all directions, including a couple of the new features that exist in Oracle 10g to aid in the task of solving performance problems.

Inside Multiversioning
by Tom Kyte

Back to the basics of read consistency and—more important—write consistency. This presentation looks at how SQL statements really are processed, especially those pesky modification statements. Find out why an update that normally takes 10 minutes never finished the other night. You will be amazed at what goes on inside Oracle to deliver a read and write consistent answer.

Scaling to Infinity—Partitioning Data Warehouses in Oracle
by Tim Gorman

Partitioning is crucial to data warehouses, but how do you decide how to physically partition? This presentation is the result of lots of practical experience in physical database design for data warehouses, driven by the combination of business requirements and systems requirements. It will provide straight answers and solid guidelines to best utilize this important feature to ensure decision-support success.

TRACK 2

Common Performance Monitoring Mistakes to Avoid
by Virag Saksena

This presentation will discuss the common mistakes made when monitoring and diagnosing Oracle issues in a Unix environment. Oracle administrators are deluged with large amounts of performance data. Monitoring mistakes often make it harder to separate the right data from the noise. We will discuss low-overhead, high-impact techniques to quickly identify, isolate, and troubleshoot performance issues.

Systems Integration Secrets Using Logical Databases
by Eric Buskirk

Coordinating the schedules of multiple projects can be incredibly difficult for DBAs and data architects. The complexity increases exponentially with each additional application. This presentation encompasses the utilization of a logical database layer comprising views, triggers, and stored procedures to support dynamic database and development environments.

Introduction to ADF in JDeveloper 10g — Is It Oracle Forms Developer Yet?
by Peter Koletzke

In its 10g release, JDeveloper offers a new set of features, libraries, and methods called the Application Development Framework (ADF) that promises to simplify Java development tasks. The presentation discusses how the additional development productivity promised by frameworks and specifically by ADF comes with additional challenges. It also shows the various tools in JDeveloper that support the ADF development method and how they ease the task of connecting Java front-end code to the database.

The Disaster Diary: Real Failures and Real Recoveries
by Jeremiah Wilton

The Disaster Diary series examines the technical details of major real-life database failures at prominent Oracle customer sites. Attendees will learn what led to failures, how technical staff resolved them, and what a DBA could have done to prevent or shorten them. This series breaks down the time spent performing various tasks to resolve problems and shows how DBAs can think critically during a failure to drastically reduce time to recovery.

TRACK 3

Managing Compliance in an Era of Data Growth
by Mohan Dutt

Data growth fueled by compliance with stricter data reten-

continued on page 24

Teamwork at NoCOUG

(COURTESY OF CALCYCLING)

NoCOUG is a successful organization with more than 400 members, and there's no way it could run without teamwork. We have a full and active Board of Directors, plus other volunteers who contribute regularly. All the people on the NoCOUG team contribute in both big and small ways, depending on what they have time for. And it's all of us working together as a team that makes for the great conferences, training days, and other benefits.

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

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

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

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

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

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

–Joel Rosingana
NoCOUG Membership Director
and Past President
Independent Consultant

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

–Darrin Swan
NoCOUG Vice President
Quest Software

NoCOUG Summer Conference

Thursday, August 19, 2004 · Location: ChevronTexaco, San Ramon, CA

Please visit **http://www.nocoug.org** for additional technical presentation details, directions to the conference, and to submit your RSVP.

8:00 a.m. **Registration and Continental Breakfast**

9:00–9:30 **General Session and Welcome**

9:30–10:30 **Keynote:** *Understanding by Design: Why Understanding Is Different from Knowing*—Tom Kyte

10:30–11:00 **Break**

11:00–12:00 **Parallel Sessions #1**

Track 1: *Strategies for Gathering Statistics* by Jonathan Lewis, JL Consulting
Track 2: *Common Performance Monitoring Mistakes to Avoid* by Virag Saksena, Founder, Silicon Valley Performance
Track 3: *Managing Compliance in an Era of Data Growth* by Mohan Dutt, Director —Oracle Delivery, Solix Technologies

Keep checking the NoCOUG website for the most updated Summer Conference schedule!

12:00–1:00 **Lunch**

1:00–1:30 **Roundtable Discussions**

1:30–2:30 **Parallel Sessions #2**

Track 1: *The Evolution of Optimization 8i to 10g* by Jonathan Lewis, JL Consulting
Track 2: *Systems Integration Secrets Using Logical Databases* by Eric Buskirk, President, Verican Systems
Track 3: *Deploying Oracle Databases on Networked Attached Storage* by Akira Hangi, EMC

2:30–2:45 **Break**

2:45–3:45 **Parallel Sessions #3**

Track 1: *Inside Multiversioning* by Tom Kyte, Vice President, Core Technologies for Oracle GEH, Oracle Corporation
Track 2: *Introduction to ADF in JDeveloper 10g — Is It Oracle Forms Developer Yet?* by Peter Koletzke, Technical Director and Principal Instructor, Quovera
Track 3: *Leveraging Master-to-Master Replication for Efficient Workload Balancing and Data Distribution* by Mark Ketchie, Data Mirror

Session descriptions appear on page 26.

3:45–4:15 **Break and Raffle**

4:15–5:15 **Parallel Sessions #4**

Track 1: *Scaling to Infinity—Partitioning Data Warehouses in Oracle* by Tim Gorman, Principal, SageLogix
Track 2: *The Disaster Diary: Real Failures and Real Recoveries* by Jeremiah Wilton, Independent Consultant
Track 3: *Integrating OpenSource and Oracle 10g Spatial with 9iAS Technology* by Tony Lopez, eSpatial

5:15 -?? **NoCOUG Networking and Happy Hour**

Cost: $40 admission fee for non-members. Members free. Includes lunch voucher.

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

NoCOUG
P.O. Box 3282
Danville, CA 94526

Thank you to ChevronTexaco, our meeting sponsor.