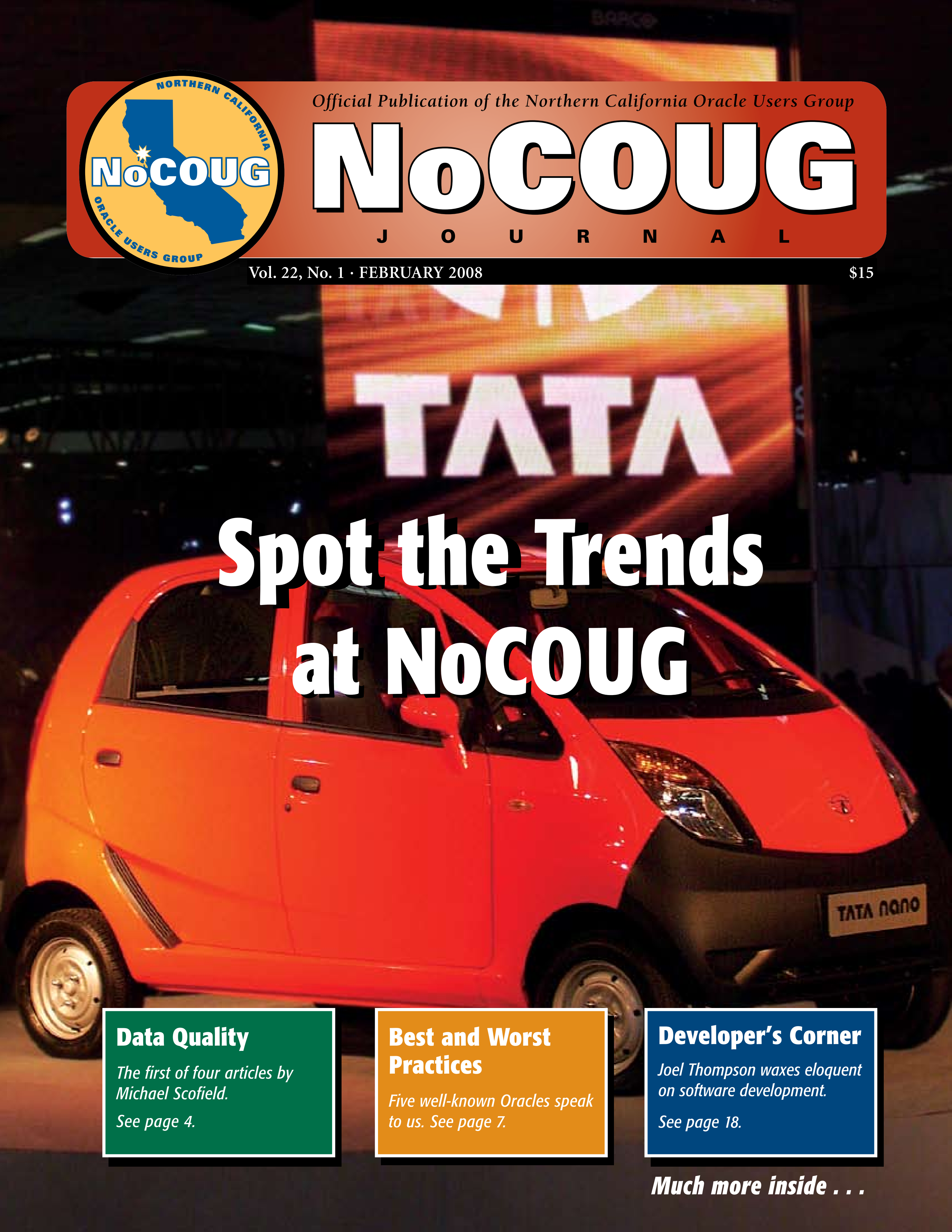
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 22, No. 1 · FEBRUARY 2008 $15



Spot the Trends at NoCOUG

Data Quality

The first of four articles by Michael Scofield.

See page 4.

Best and Worst Practices

Five well-known Oracles speak to us. See page 7.

Developer's Corner

Joel Thompson waxes eloquent on software development.

See page 18.

Much more inside . . .

Spot the Trends at NoCOUG

I recently visited India with a friend. There was a car show in progress in Delhi at the time and we saw the new "people's car," the Tata Nano, which will only cost $2,500 when it rolls off the manufacturing lines in a few months. That is the beautiful little car you see on the front and back covers of this issue of the NoCOUG Journal.

More than 100,000 people visited the Delhi car show every day and I wished that 100,000 people would come to NoCOUG's conferences to spot the new trends in database technology. At NoCOUG's winter conference on February 19, Juan Loaiza will deliver the keynote address on the future of high availability. Juan Loaiza has been at Oracle for almost twenty years and knows more about Oracle's vision than most other people.

Visiting India is the best way to experience the dynamism of a burgeoning young population and comprehend how it will change India and the entire world. In the same way, attending conferences such as those organized by NoCOUG is the best way to understand how database technology is rapidly changing to meet the needs of a rapidly changing world.

I hope you enjoy this issue of the NoCOUG Journal, and I hope to meet you at our winter conference! ▲

—Iggy Fernandez, *NoCOUG Journal* Editor

Table of Contents

Publication Notices and Submission Format

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

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

The submission deadline for the upcoming May 2008 issue is February 28, 2008. Article submissions should be made in Microsoft Word format via email.

Copyright © 2008 by the Northern California Oracle Users Group except where otherwise indicated.

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

ADVERTISING RATES

The *NoCOUG Journal* is published quarterly.

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

Personnel recruitment ads are not accepted.

journal@nocoug.org

2008: Ringing In the New

by Roger Schrag

Roger Schrag

Greetings Oracle professionals! A new year is upon us, and NoCOUG is here to help you develop your career and personal network, just as we have been for over 20 years.

When I started my Oracle career, RDBMS v5.1 and SQL*Forms 2.3 were state of the art. Today it's Oracle Database 11*g*, JDeveloper, and SOA Suite. The technology treadmill moves quickly, so the most successful Oracle professionals take an active role in their career and never stop learning and networking. That's where NoCOUG fits in. Our mission is to provide educational, networking, and leadership opportunities for Oracle professionals in Northern California.

In 2008 you can look forward to conferences with presentations by industry leaders such as Juan Loaiza, Gaja Vaidyanatha, Jonathan Lewis, Bradley Brown, Steven Feuerstein, Cary Millsap, and Tim Gorman—just to name a few. We'll also be holding a special training event with Jonathan Lewis, and we'll be publishing the *NoCOUG Journal* each quarter and adding useful content to **www.nocoug.org** throughout the year.

NoCOUG continues to be the best value for career development in Northern California. Where else does $80 buy you four days of Oracle education throughout the year? Not to mention the valuable information published in the quarterly journal and on the NoCOUG website. Plus, NoCOUG offers great networking opportunities; at a NoCOUG conference you can meet scores of fellow Oracle professionals who live and work right here in Northern California. In addition, NoCOUG training days provide an opportunity to get further in-depth training in specialized areas at discounted prices.

Since you are reading this, the chances are good that you already are a NoCOUG member. If you aren't, I encourage you to join NoCOUG today! You can join online at **www.nocoug.org**. I also encourage everyone to spread the word: Tell your officemates, colleagues, and friends about NoCOUG. You can do a great favor for other Oracle professionals by introducing them to our users group. Enlarging the NoCOUG membership, and thus enlarging our network, benefits all of us.

With the new year, come some changes in the NoCOUG leadership. Our users group is run by a team of a dozen volunteers who take time out of their busy lives to share with the community and make NoCOUG what it is. It takes a lot of energy and time to manage an organization with over 450 members, four conferences per year, a quarterly publication, and a sophisticated website.

I will be your loyal and faithful president this year, with Hanan Hit at my side as vice president. Randy Samberg is stepping up to the role of director of conference programming, and Lisa Loper is moving into the vendor coordinator position. Seven additional volunteers will continue to serve in the same board roles this year as last: Iggy Fernandez as *Journal* editor, Joel Rosingana as membership director, Eric Hutchinson as webmaster, Hamid Minoui as training day coordinator, Naren Nagtode as director of marketing, Diane Lee as IOUG representative, and Jen Hong as secretary/treasurer.

I thank each of these volunteers for their dedication to helping others in the Oracle profession. Also, a special thanks to Darrin Swan, who is stepping down from the board this year after several productive years in such important roles as president and vice president. Finally, I would like to thank Nora Rosingana, NoCOUG's only paid staff. (The pittance NoCOUG can afford to pay Nora doesn't come close to the value she provides the organization.)

I look forward to seeing you at NoCOUG's Winter Conference, February 19 at the Oracle Conference Center in Redwood Shores. Juan Loaiza will kick off the event with a keynote on the future of high availability. *As senior vice president, Systems Technologies, at Oracle Corporation, Juan knows a lot about the subject, including Oracle Corporation's vision and where the Oracle database technology is heading.*

After the keynote there will be 12 technical presentations, including presentations by Oracle's technical team on new features in Oracle Database 11*g* and SOA Suite. Meanwhile, Daniel Morgan, Bradley Brown, and Dan Norris will be flying in to speak on subjects ranging from little-known but very useful Oracle features to the Web 2.0 paradigm. Local Oracle users Iggy Fernandez, Peter Koletzke, and Avrom Roy-Faderman will also give presentations on topics including deadlocks and placement of application business logic.

I look forward to seeing you at the February 19 NoCOUG Winter Conference. Together we can get our careers off on the right foot in 2008! ▲

Fundamentals of Data Quality

by Michael Scofield

Michael Scofield

This is the first of a series of articles about data quality, including the techniques, politics, and tools involved in improving the quality of the data in an enterprise. We will not be pushing any tools or software. But we will be presenting general concepts and principles of managing data and its quality.

While in most organizations data quality is not the primary responsibility of the DBA, the DBA (in the interest of political self-preservation) should be sure that the data-quality topic is covered by someone in the organization. We may hear an angry, oxygen-deprived executive holler at the DBA, "You were the last one with the data; it's your fault!" We know that is not true, but what can you do preemptively to defend against that confrontation? This first article looks at the politics and functions of DQ.

So while the DBA cannot ensure good data quality, the DBA needs to understand what good data quality is, and what functions and processes are needed laterally to the DBA function to maintain an acceptable level of confidence in the data.

Data quality is first and foremost a business issue, not merely a technology issue. True, programming errors can create bad data, but that doesn't happen nearly as often as business design errors. To achieve high data quality, business processes and the applications that support them need to be well designed. The clerk or business user (or even a customer using your web site) should never be in doubt as to what data is required from them in any particular field on a screen. That is where data quality starts. If the interaction is ambiguous, you may get "creative" data ("I don't know what they want, but I think I will enter this . . .").

So while business process owners may be (in a pure sense) responsible for their data (or the data they capture) they may not understand the principles of data quality and good process design. Here is where a mature systems analyst, acting responsibly, will design a high-quality process to prevent data errors coming into the database. The entry of bad data is not solved by simply turning on some referential integrity switch.

Data behavior

Data quality is one characteristic of data behavior—what actually takes place in the production databases as a result of business activity. Data behavior is just what it sounds like—the behavior of data as records are added, changed, and marked for deletion as the result of normal business events. Not all business behavior results in good data.

Data behavior is not data architecture. Most data documentation describes how a data field (a.k.a. "column") is named, its format, and in what table it is located. That is part of data architecture, and might be found in a logical data model. Data documentation may include some business metadata such as the valid values (if a coded field) and what they mean.

But all that is normative documentation (i.e., what the designer intended). That may not be what actually happens in the data field once the business starts using it. Normative documentation could have a "quality" characteristic in that it is clear and unambiguous. And good documentation can contribute to good data quality. But the documentation is not the business data. Data quality is a characteristic of what happens in the actual database, as the result of a living, breathing business. That is data behavior. It is amazing how few techies are even curious about what is actually in the data.

Data governance

Many large bureaucracies are now embracing the latest buzzword, "data governance." This can be an initiative, often backed up with a staff and budget to bring order and consistency to an enterprise's data asset, particularly to the master files. This, by the way, is what "data administration" people were talking about 25 years ago. New name.

One of the values of data governance is to bring stability and reliability to the reference tables and master files. Often, this means restricting change access and developing rules for the timing of changes.

Data governance also is launched by companies that are newly sensitive to the constraints and fines under new federal legislation holding corporate officers accountable for the security and quality of their data and what information they report (as in financial statements). So "compliance" is a popular term to encompass data management and data security. A data governance person may find some allied support from the internal audit department.

Data steward

A data steward is a title some organizations give to someone charged with the responsibility for managing or governing data. It can often be a thankless job, if the individual does not have the authority to bring about necessary changes. This role generally has a business orientation, rather than one of technology. The main technological issue is to have the ability to pose ad hoc queries against any of the data in the scope of responsibility—to actually go and look at what is going on in the data, *and* to achieve that visibility with a minimum of effort (i.e., not waiting weeks for a programmer to write a special report). This visibility can be achieved with a variety of off-the-shelf report writers and query tools (not necessarily expensive DQ tools).

While the data stewardship or data governance function is often somewhat centralized, one of the frustrations is that the power to improve data quality is widely distributed—found in business users, application developers, and many others.

Enterprise Data Assets

A data management program should extend beyond the formal relational databases (what we called "production" 20 years ago). It may include departmental databases and data residing on external platforms (such as outsourced applications, web or otherwise). While the data may be on someone else's server, it is still an enterprise data asset.

An important role of a data governance function is to ensure that all data assets are secure from loss or unauthorized modification.

Data Quality Function

Some large organizations have a formalized data quality function. The roles and responsibilities may vary. The end goal may be to improve or maintain a certain level of data quality. But how that is achieved may differ.

While the DBA cannot ensure good data quality, the DBA needs to understand what good data quality is, and what functions and processes are needed laterally to the DBA function to maintain an acceptable level of confidence in the data.

But before we set goals, we must define "data quality" with greater precision.

Fundamentals of Data Quality

Data quality is a rather sweeping—hence ambiguous—concept. There are many distinct measures of quality. There can be no single, overall score of data quality (even though some executives might wish for one). The measures must be quite specific. Sub-characteristics of data quality include the following:

Scope of data. Scope of data is often assumed or not discussed. This is particularly so when acquiring data from some external source. Are you getting all the tables (or logical entities) that you expect and need? Within a table, you can further ask, are you getting all the rows you expect? Is there a row present for each instance in the real world? That is a question of scope. Even more subtle, are you getting multiple rows for one instance (i.e., duplicate records)?

A large company loaded an externally acquired file to its

It's Registration Time Again!

I know the year just zoomed by. I'm sure it's due to the great 2007 NoCOUG program. The board has another stellar year planned for you. Please register early to continue getting this award-winning *Journal* and streamline your conference registration. For our 2007 members who have not yet registered for 2008, this issue of the *NoCOUG Journal* is complimentary. We know you plan to register.

Check the address label on the back of the *Journal*. If there is an asterisk (*) affixed to your name, you are already registered for 2008. Keep in mind that the mailing list was pulled on 1/26/08. The (*) status indicator will only be used on this issue.

All 2007 members should have received an email or U.S. mail reminder by now. That will include your NoCOUG ID for online registration at **www.nocoug.org**. Contact me with any questions at **membership@nocoug.org**.

Hope to see you at the coming conferences!

—Joel Rosingana
NoCOUG Membership Director

database. The programmer was expecting a set of customers in Indiana, and neglected to do a simple row count, or tally the values found in the STATE code. Rather, he blindly loaded it to the database. This input file actually contained two other states in addition to Indiana, and nobody noticed until the DBMS ran out of overflow space. Don't you just love calls in the middle of the night?

The lesson here is to understand what is in any dataset (including scope) before you do anything important with it.

Cell population. The question is basically this: given the business rules and expectations for when a non-null value should be present, is this column adequately populated? This requires an understanding of business rules. Having a field in a table doesn't necessarily mean it is populated appropriately. Having a GENDER field in the Customer Master File does no good if nobody ever captured the actual data.

Validity. For coded fields, you want to ensure that the values found are what are expected. This "domain" is often known, and either in the documentation or in a reference table. Referential integrity (if turned on) may help ensure validity, but that does not work for externally acquired data.

So if the documentation says STATUS_CODE can be "A", "B", or "C", and you find one record with a "Q" in that field, you probably have a validity problem.

For numeric fields, a range of values may be what is expected, and tests can be run to ensure that the values are in that range. For text fields (such as comments, customer name, address, etc.) there is often little validation that can be done.

Reasonableness. Data can be quite valid and still unreasonable. Tools testing for validation using Boolean logic may not detect unreasonable data, but human intuition often can. "That can't be right!" or, "That doesn't look right."

I had one attendee in New Jersey come up to me after my lecture on data quality and brag, "I can smell bad data." Cool! Wish we had more like her. In the absence of that olfactory skill, we need to devise astute reasonability tests.

Consider the following noon temperature observations:

```
LOCATION     NOON TEMP-F
------------------------
MARYSVILLE            92
SACRAMENTO            91
DAVIS                 39
STOCKTON              94
MODESTO               90
```

Any reasonable person familiar with California's Central Valley will say that temperature in Davis is not correct. Valid? Sure. It may reach 39 on some cold winter nights. But 39 is not reasonable given the context of other data points. Indeed, nearly all reasonability testing involves context data values, either historical or of peer instances.

Accuracy. A fact (the data in a single cell) may be valid and reasonable, and still be flat out wrong. And bad data may be difficult and costly to detect—especially when the data describes an event that cannot be re-created, and for which there were no recording witnesses. Here is where there is potential for fraud: keep the data valid and reasonable, and nobody will suspect a thing!

Precision. Precision is subtly different from accuracy; they are often confused. A numeric value can be stored with precision to the fifth decimal point and still be woefully inaccurate. One way to lie is to feign precision. Yet an accurate value (accurate enough to be useful in many decision-making applications) may not be precise yet still be useful.

Timeliness. A fact (like a customer's mailing address) can be quite accurate—two years ago—but now wrong. He moved. It was never updated. The data is no longer current. This is a data quality problem, but it is also related to data definitions and expectations.

Consistency. Consistency applies to definition and data behavior (such as the domain of values observed). We want to ensure that the meaning of a field is the same now as it was four years ago (particularly important for data warehouses where time-series analysis is often valuable). And we may want lateral consistency, such as all corporate divisions using the same definition for "net sales."

These characteristics describe data in its original or "best" form, either "at rest" in your database, or in initial input files received from outside the organization.

There are a number of other data quality issues in the methods and pathways of delivery and usage. From raw data, processes (machines or people) may create "derived data" through aggregation, filtering, integration, etc. This moves the data toward useable information. The quality of derived data depends, in part, upon the quality of the deriving process.

But these, in my view, fall into a separate category. Flaws in aggregation, translation, filtering, and distribution all can undermine the utility of data. Delays can frustrate business users (especially of business intelligence systems). But they are systemic data delivery issues, not fundamental data quality issues.

Data quality management can be frustrating, especially without the power to improve data or processes. I was director of data quality for a large organization, yet had absolutely no authority to change a single piece of data. I could look at the data and find problems, but then I had to persuade other people (e.g., application owners) to see there was a problem and to alter (improve) their processes. So data quality improvement is usually quite political.

Organizations that think merely purchasing a DQ tool will solve any festering data quality problem may be making a big mistake. And if the DBA is consulted in any way on this decision, the DBA must be ready to discuss the issues rationally. Hence, this series of articles.

Summary

In this article we introduced the key characteristics of data quality, and what some organizations are doing in the data management area. In our next installment, we will look at simple techniques for understanding data behavior, evaluating data quality, and detecting problems. ▲

Michael Scofield is manager of data asset development at ESRI, Inc., in Redlands, Calif., and holds a faculty appointment—assistant professor of health information management—at Loma Linda University. He can be reached at **mscofield@esri.com**.

Best and Worst Practices?

Ask the Oracles!

Gaja Krishna Vaidyanatha: Wikipedia defines best practice as "a management idea which asserts that there is a technique, method, process, activity, incentive or reward that is more effective at delivering a particular outcome than any other technique, method, process, etc. The idea is that with proper processes, checks, and testing, a desired outcome can be delivered with fewer problems and unforeseen complications. Best practices can also be defined as the most efficient (least amount of effort) and effective (best results) way of accomplishing a task, based on repeatable procedures that have proven themselves over time for large numbers of people."

Those of you who have been there and done that with Oracle systems for many years know very well that it is impossible to predict and plan for every scenario that your application and database will present in production. This is especially true when you embark into upgrading any component of your system—especially when you are betting on a new feature in the software to bail your system out of a sticky situation! But you upgrade anyway, because inaction is worse than the current performance status-quo or corner-case bugs that you may encounter in the future. Best practices come in very handy in situations such as these.

In a recent performance management engagement at one of my customer sites, I was privy to a strange (yet seemingly common) situation. The Oracle database was upgraded from 9*i* to 10*g*, and even though all of the pre-upgrade load tests performed in stellar fashion, the database experienced sporadic hiccups in performance during the post-upgrade phase in production. And during these hiccups, the CPUs on the system were constantly and consistently maxed out. This was because the execution plans of the core SQL in the application had gone south. When a 32-CPU machine with an average idle percentage of 85% in Oracle 9*i* gets maxed out in Oracle 10*g*, you know there is a gremlin lurking out there. With adequate performance diagnostic data, it was determined that Oracle's "automatic feature" of "peeking for bind variable values" at hard-parse time was the cause for the pain. To ensure future stability and to guarantee performance capacity during the 2007 holidays, the feature was completely turned off.

How do you plan for a situation such as this, when it did not surface during pre-upgrade performance load tests? Clearly, in this case, the pain had to be experienced before remedial measures could be undertaken. But does that mean that you don't test? No, you still do. Nor does this mean that you take preemptive action and turn off every new feature or modify the default values of every Oracle initialization parameter that is different from the prior release—*especially* when it is an "underscore" parameter.

From an Oracle Performance Management perspective, the following are some key best practices I'd recommend:

1. System Performance Diagnostic Best Practice—Chant and practice the "response time mantra." At the end of the day, performance management is about user experience and application response times, not a slew of database health metrics. If you engage with a specific response time goal in mind, you will stop tuning when you achieve that goal. Remember, Compulsive Tuning Disorder (CTD) is not a good thing!

2. System Performance Diagnostic Best Practice—Utilize mathematical data instead of expert opinions to arrive at *all* of your conclusions. Opinions are not facts and can be very easily protected by the expert with the YMMV (Your Mileage May Vary) camouflage. When you can categorically state that a query's elapsed time is 15 minutes, and 83.5% of its elapsed time is spent in performing single-block I/O requests, then that is a quantified problem that you can attempt to solve.

3. Application Performance Testing Best Practice—Create a realistic test environment, so that major upgrades can be tested for feasibility, reasonability, and stability. Production life is hard enough with testing, and you truly do not want to be out there without testing. Your load tests should reflect reasonability and veracity when compared to your production environment. Testing should be done using realistic production data (or at least production Optimizer statistics), from both the database and application's perspective.

4. System Performance Environment Best Practice—Engage in balanced maneuvers when building and upgrading systems. If you double the capacity of your CPUs, then take the time to review the capacity in your I/O sub-system to determine whether it too requires a similar upgrade. Throwing more or faster CPUs without reviewing the capacity of your I/O sub-system may result in real-time system imbalances. These imbalances will translate into long-term scalability and performance problems.

5. Application Performance Design Best Practice—If a piece of application functionality is data intensive, build it with SQL first. If the code requires complex logic, then build it in the database (yes PL/SQL). Alternatively, if the functionality is processing intensive, then build it in the application tier. In either case, avoid unnecessary single-row processing roundtrips between the database and the application tier.

Based on my experience in the field, I'd recommend that you steer away from the following worst practices:

1. Building "database agnostic applications"—This is a commonly used buzzword line that does not make any sense. The term "agnostic" means "to neither believe nor disbelieve" and to limit one's belief to human experience. Are we saying that if we reinvent the wheel and build database constraints in the application tier, we are actually limiting our belief in Oracle based on our experience? Nonsense! Databases such as Oracle should not act only as "data stores," as they provide way more functionality for scalable performance. Not utilizing the inherent power of the RDBMS is like driving a Lamborghini with 95% of its power turned off.

2. Overusing dynamic SQL at the application tier—If Oracle has to parse every single SQL statement that you present to it because of the dynamism that you have introduced in your application logic, it makes your system that much less scalable. There are application environments where CURSOR_SHARING cannot be set to FORCE and dynamic SQL (non-reusable SQL) will open up a slew of library cache and shared pool latch contention performance problems.

3. Designing without scalability—If you have the luxury of custom designing any part of your system, not ensuring that it will work with large data sets is a worst practice. For example: if you write a query that works well with a one-thousand-row table, not ensuring that it will exhibit a reasonable response time when processing a one-million-row table is a recipe for disaster. The underlying theme here is "design for scalability."

4. Using database ratios to determine the true performance bottleneck—Ratios may be indicators but definitely not "root causes" of performance problems. In all of my engagements with my customers, I have yet to find the need to review a single database ratio to solve a customer's performance problem. The 10046 trace files combined with STATSPACK and AWR reports and some OS commands pretty much define my tuning arsenal. Yes, I do skip the ratios in the aforementioned performance reports. And if I do find ratios to be useful in the future, you will be the first to know!

5. Drinking vendor Kool-Aid due to their stature in the marketplace—Verify the durability and performance of any piece of software or hardware by putting it through the paces in your test environment. Again, remember Best Practice #2—Use mathematical data, not expert opinions. And don't fall prey to sales pitches!

So in the context of Oracle Performance Management, adhering to the best practices listed in this section is a good thing. And by no means is this a comprehensive list of best and worst practices; it is just a beginning. May I remind my readers: "Well begun is half done!" Cheers! ▲

*Gaja Krishna Vaidyanatha has over 15 years of industry technical expertise working with Oracle systems. He is the principal of DBPerfMan LLC (***www.dbperfman.com***), an independent consulting firm specializing in the area of Oracle Database Performance Diagnostics & Management for Fortune 500 corporations. He is the primary author of Oracle Performance Tuning 101, published by Oracle Press, and one of the co-authors of Oracle Insights: Tales of the Oak Table, published by Apress.*

Jonathan Lewis: It's quite hard to say anything sensible or inspiring about a topic like this. Many of the specifics we call "best practices" are extremely obvious common sense, but sometimes circumstances conspire to make it impossible to follow them. Many of the worst practices are things that we should obviously avoid, but again circumstances (often pressures of time) may make them unavoidable.

The worst generic practice I know of, then, is applying a fix to a problem without understanding why that fix might work.

I have a simple approach to problem solving that involves three steps: 1) "What is the problem?"; 2) "Why will my solution fix the problem?"; and 3) "Where am I going to pay for implementing this solution?"

The last question will sometimes tell us that we can't afford to implement the "perfect" solution—perhaps we want to change a heap table to an index-organized table (IOT), but can't because we have too much code that has done something a little exotic with traditional rowids and needs to be rewritten to deal with the "urowid" used for IOTs.

But before we worry about such side effects, overheads, and implementation costs, we need to be confident that our solution really is addressing the root cause; if it isn't, we may spend time and effort implementing a change that seems to fix our problem temporarily—until things go wrong again.

Take a simple example: A query takes an unreasonable amount of time. You check the execution plan and decide the problem might go away if Oracle used a particular index. So you rebuild the index and the query runs much more quickly. Is your job finished?

No doubt the first thing you do is check the execution plan to see that it changed to use the index the way you expected. But does that prove that the performance improvement came from the change in plan? Is it possible that your index rebuild used a table-scan that resulted in the target table being cached somewhere (in the SAN cache, for example) so that accesses to the table were much quicker than they would normally be?

Even if the improvement *was* due to the change in execution plan, do you know *why* the plan changed? Was it because the rebuild packed the index, giving you a smaller leaf_blocks (or blevel, even), or was it because index rebuilds automatically compute statistics in your version of Oracle, and a small change in the statistics (distinct_keys or leaf_blocks being the most likely) made the difference? Will the new plan survive the next statistics collection, even if the index doesn't start growing immediately after the rebuild?

If you don't check, you may end up performing a regular, yet redundant and potentially dangerous, rebuild of this index—and when, exactly, does it need to be rebuilt anyway?

What if the change was due to a change in the distinct_keys? There may be random occasions when the rebuild strategy just doesn't work, or a day may come when the rebuild will no longer work because even the freshly rebuilt index has grown past a critical number of leaf_blocks.

Whenever you make a change that's supposed to fix a problem, try to capture the before-and-after information (e. g., statistics, execution plans, work done, number and type of waits). If there's any doubt in your mind about the root cause, never be afraid to document what you've done and the reasons why you did it—it may save you a lot of time in the future when your "fix" turns out to have been just a lucky coincidence. ▲

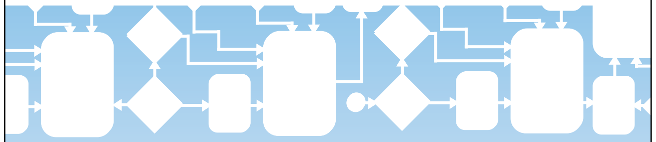
Jonathan Lewis is well-known to the Oracle community as a consultant, author, and speaker, with more than 18 years' experience in designing, optimizing, and troubleshooting on Oracle database systems. His latest book is Cost-Based Oracle Fundamentals, which is the first of three volumes on understanding and using the cost-based optimizer.

Christian Antognini: In recent years, I have witnessed a worrying trend in physical database design. This trend may be called wrong datatype selection. At first glance, choosing the datatype seems like a very straightforward decision to make. Nevertheless, in a world where software architects spend a lot of time discussing things like agile software development, SOA, AJAX, or persistence frameworks, I am convinced it is essential to get back to the basics and understand why datatype selection is important. To illustrate this, I will present four examples of typical problems that I have encountered over and over again. Even though this may all seem very basic, don't underestimate the number of systems that are now running and suffering because of these problems.

At the end of the day, performance management is about user experience and application response times, not a slew of database health metrics. If you engage with a specific response time goal in mind, you will stop tuning when you achieve that goal.
—Gaja Krishna Vaidyanatha

The first problem caused by wrong datatype selection is incorrect or inadequate validation of data, when it is inserted or modified in the database. For example, if a column is supposed to store numeric values, choosing a character datatype for it calls for an external validation. In other words, the database engine is not able to validate the data—it leaves the application to do it. Even if such a validation is easy to implement, bear in mind that every time the very same piece of code is spread to several locations, sooner or later there will be a mismatch in functionality. The example I would like to bring forward is related to the initialization parameter NLS_NUMERIC_CHARACTERS. Remember that this initialization parameter specifies the characters used as decimals and group separators. For example, in Switzerland it is usually set to "." and, therefore, the value Pi is formatted as follows: 3.14159. Instead, in Germany it is commonly set to "," and, therefore, the same value is formatted as follows: 3,14159. Sooner or later, running an application with different client-

Announcing
Embarcadero® EA/Studio™



Join your processes and data with EA/Studio.

- Conceptual modeling
- Real-world process modeling
- Speak the same language as business users

Visit www.embarcadero.com/eastudio to download a free trial.

EMBARCADERO TECHNOLOGIES®

ENTERPRISE STORAGE SOLUTIONS FOR ORACLE.
SCALABLE STORAGE INFRASTRUCTURE.
SCALED DOWN STRESS.

PICK ANY THREE.

SETTLE FOR EVERYTHING WITH NETAPP.

Find out why we're trusted by the world's greatest companies that run Oracle.

Visit netapp.com/bizapps

NetApp® Simplifying Data Management

© 2007 Network Appliance, Inc. All rights reserved. Specifications subject to change without notice. NetApp and the Network Appliance logo are registered trademarks and Network Appliance is a trademark of Network Appliance, Inc. in the U.S. and other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

side settings of this initialization parameter will run into an "ORA-01722: invalid number" if conversions take place because of the use of a wrong datatype in the database.

The second problem caused by wrong datatype selection is the loss of information. In other words, during the conversion of the original (correct) datatype to the database datatype, information gets lost. For example, let's imagine what happens when the date and time of an event is stored with a DATE datatype instead of a TIMESTAMP WITH TIME ZONE datatype. Fractional seconds and time zone information get lost. While the former leads to a very small error (less than 1 second), the latter might be a bigger problem. In one case I have witnessed personally, a customer's data was always generated using local standard time (without daylight savings time adjustments) and stored directly in the database. The problems arose when, for reporting purposes, a correction for daylight savings time had to be applied. A function designed to make a conversion between two time zones was implemented. Its signature was the following:

```
new_time_dst(in_date DATE, tz1 VARCHAR2, tz2 VARCHAR2)
RETURN DATE
```

Calling such a function once was very fast. The problem was calling it thousands of times for each report. The response time increased by a factor of 25 as a result. Clearly, with the correct datatype, everything would not only be faster but also easier (the conversion would be performed automatically).

The third problem caused by wrong datatype selection is that things do not work as expected. Let's say you have to range partition a table, based on a column storing date and time information. This is usually no big deal. The problem is that the column used as partition key contains numeric values. If the conversion from the numeric values is performed with a format mask like YYYYMMDDHH24MISS, the definition of the range partitions is still possible. However, if the conversion is based on a format mask like DDMMYYYYHH24MISS, you have no chance of solving the problem without changing the datatype or format of the column (as of Oracle Database 11*g*, it is possible to work around the problem by implementing virtual-column-based partitioning).

The fourth problem caused by wrong datatype selection is related to the query optimizer. This is probably the least obvious of this short list, and also the one leading to the subtlest problems. The reason for this is that the query optimizer will perform wrong estimates and, consequently, will choose suboptimal access paths. Frequently, when something like that happens, most people think it's the fault of the query optimizer, which once again is not doing its job correctly. In reality, the problem is that information is hidden from it and so the query optimizer cannot do its job. To illustrate this problem, let's take a look at an example. Here, we are checking the estimated cardinality of similar restrictions based on three columns that store the very same set of data (the date of each day in 2008), but based on different datatypes. As you can see, the query optimizer is only able to make meaningful estimations (the correct cardinality is 29) for the column that is correctly defined.

```
SQL> CREATE TABLE t (d DATE, n NUMBER(8), c
VARCHAR2(8));

SQL> INSERT INTO t (d)
  2  SELECT trunc(sysdate,'year')+level-1
  3  FROM dual
  4  CONNECT BY 1 = 1
  5  AND level < trunc(sysdate+366,'year')-
trunc(sysdate,'year')+1;

SQL> UPDATE t SET n = to_number(to_
char(d,'YYYYMMDD')), c = to_char(d,'YYYYMMDD');

SQL> SELECT * FROM t ORDER BY d;

D                  N C
--------- ---------- --------
01-JAN-08   20080101 20080101
02-JAN-08   20080102 20080102
…
30-DEC-08   20081230 20081230
31-DEC-08   20081231 20081231

SQL> execute dbms_stats.gather_table_stats(user,'t')

SQL> EXPLAIN PLAN SET STATEMENT_ID = 'd' FOR
  2  SELECT * FROM t WHERE d BETWEEN
to_date('20080201','YYYYMMDD')
  3                              AND
to_date('20080229','YYYYMMDD');

SQL> EXPLAIN PLAN SET STATEMENT_ID = 'n' FOR
  2  SELECT * FROM t WHERE n BETWEEN 20080201 AND
20080229;

SQL> EXPLAIN PLAN SET STATEMENT_ID = 'c' FOR
  2  SELECT * FROM t WHERE c BETWEEN '20080201' AND
'20080229';

SQL> SELECT statement_id, cardinality FROM plan_table
WHERE id = 0;

STATEMENT_ID CARDINALITY
------------ -----------
d                     30
n                     11
c                     11
```

In summary, there are plenty of good reasons for selecting your datatypes correctly. Doing so may just save you a lot of problems. ▲

*Since 1995, Christian Antognini has been focusing on understanding how the Oracle database engine works. He is currently working as a principal consultant and trainer at Trivadis AG (**www.trivadis.com**) in Zurich, Switzerland. If he is not helping one of his customers get the most out of Oracle, he is somewhere lecturing on optimization. Christian is the author of Troubleshooting Oracle Performance (Apress 2008).*

Dan Tow: From my perspective as a SQL tuning consultant, I have to give three cheers for views. About 75% of my time spent tuning really tricky, non-routine SQL tuning problems is consumed dealing with complex SQL built on complex views. These are the problems that can consume hours, even days, of the best experts in the business, assuring continued, lucrative employment for professionals like myself. Of course, if you are on the employer's side, looking for efficiency, or if you are an overworked developer on a salary, your perspective may be a little different!

SQL is a fabulously flexible query language, but this flexibility is often a curse, as it spools out rope to hang yourself on a plausible-looking query that returns pure nonsense—or worse, that appears to return the desired results in a simple test, but fails in a dozen or more corner cases that the test fails to exercise. Such nonsense is all too likely when dealing with a simple, flat join of tables, but it becomes almost a certainty when dealing with complex SQL built on top of complex views.

A business application database stores data about all the business entities relevant to that application. Except in the rarest of cases, a query should return rows that represent a single type of entity, mapping rigorously one-to-one to instances of that entity (or to aggregations of that entity, in the cases of group-by queries). If the entity is worth a query, it is probably worth a physical table, so the query result, however many tables are in the FROM clause, should map rigorously one-to-one to a subset of rows from a single physical table (which I'll call the "primary" table, here, but which I call the "root detail table" in my book). All the other tables or views mentioned in the query should simply provide lookup data (data pertaining to the primary table, master data under the primary detail data, stored in master tables since the database is normalized) or data necessary to filter out primary-table rows that don't belong in the desired, queried subset. The actual best path to the data might be complex, but one very simple path (which might not be the fastest) should almost always be possible; reach the rows of the primary table, first, all the rows, if necessary, then reach every other queried table (with the possible exception

Many of the specifics we call best practices are extremely obvious common sense, but sometimes circumstances conspire to make it impossible to follow them. Many of the worst practices are things that we should obviously avoid, but again circumstances may make them unavoidable.—Jonathan Lewis

of tables in subquery conditions such as EXISTS and NOT EXISTS) through its primary key, from one or more joins beginning with each row of the primary table. This can be shown as a tree, with the primary table at the root, as described in my book, *SQL Tuning*, and it can be expressed easily with a simple, flat query against simple tables.

Now, here is the point that gets lost when predefined and inline views come into play: use of views doesn't significantly expand the universe of queries that make sense, which can be expressed simply with a flat set of joins to simple tables; complex views just make it easier to write nonsense queries that do not belong to the set of queries that make sense. They make it simpler to obscure the fact that the query is nonsense; they make it much harder to fix the query when it must finally be fixed; and they make it much more likely that the query will be slow as molasses and will require the services of a top SQL tuner!

- Analyze performance by workload/application
- Predict performance shortfalls in the next 12 months
- Understand the effects of change: new apps, 10g, more users, new hardware...in seconds

BEZ systems The Prediction People™

Real-World Experience

For Oracle database consulting and support, it makes sense to work with a company that has a proven track record. Since 1995 our clients have relied on us for:

- Performance tuning
- Migrations and upgrades
- Backup and recovery strategies
- Database security audits

Plus, we offer ongoing **remote DBA** support plans that are tailored to your business needs and budget.

Call Us Today!

(415) 344-0500 • (888) 648-0500
www.dbspecialists.com

Database Specialists — ORACLE Certified Solution Partner

Here are some rules for using views relatively safely:

1. The view-defining query must, itself, be a sensible query, with results rigorously mapping one-to-one to a single physical table, the "primary table," or to aggregations of rows of that table, in a group-by view. It should be obvious what that primary table is, if necessary through explicit comments built into the view-defining query.

2. If you are going to join to the view from another, more-detailed view or a more-detailed table, then the view should include as one (or more) of its columns the primary key of the view—normally the primary key of the primary table—and the join should be from a foreign key to that complete primary key.

3. The view-defining query should alias each selected expression to the name of the view column, unless the expression is a simple column that already has the same name as the corresponding view column. This saves immense confusion in figuring how to map the view-based query to an equivalent query against simple tables.

4. Where there are inner joins in the view-defining query, these should be commented so that it is clear whether the inner join serves a necessary filtering function—deliberately discarding rows that fail the join—or the join is inner simply because it is always expected to succeed, serving no filtering role. This makes it clear whether the join can be removed when writing the equivalent query to simple tables, if the join only serves to reach reference data in the view that is not needed in the final query.

5. When you can't follow these rules, this is usually a clue that the schema design is broken and should be fixed before it does more damage and is even harder to disentangle. ▲

Dan Tow has 18 years of experience fully focused on performance and tuning, beginning at Oracle Corporation from 1989 to 1998; followed by TenFold Corporation, from 1998 to late 2002; followed by success as an independent consultant under his SingingSQL banner. He invented a systematic, patented method (U.S. Patent #5761654) to tune any query efficiently, which he has taught (or at least introduced) directly to over 1600 people, and this method is extended and elaborated in a course he teaches and in his book, SQL Tuning, published by O'Reilly. Dan lives in Palo Alto, Calif., and is reachable at **dantow@singingsql.com**. *You can view his website at* **www.singingsql.com**, *where you can also find his complete resume.*

Steven Feuerstein: Here is an excerpt from my latest publication, the second edition of *Oracle PL/SQL Best Practices*.

Problem: The exception section of a block can only trap errors raised in the executable section!

That is a little fact that many PL/SQL developers don't realize, and one that causes lots of headaches.

Delaware[1] writes the following packaged function, a classic "getter" of a private variable:

[1] A member of the development team featured in Steven's book.

```
CREATE OR REPLACE PACKAGE fe_config
IS
   FUNCTION get_worst_excuse  RETURN VARCHAR2;
END fe_config;
/

CREATE OR REPLACE PACKAGE BODY fe_config
IS
   c_worst_excuse CONSTANT VARCHAR2 (20) := 'The dog
ate my homework. Really.';

   FUNCTION get_worst_excuse  RETURN VARCHAR2
   IS
   BEGIN
      RETURN c_worst_excuse;
   END get_worst_excuse;
BEGIN
   DBMS_OUTPUT.put_line ('Initialization logic here');
    ... lots of initialization code ...
EXCEPTION
   WHEN OTHERS
   THEN
      fe_errmgr.log_and_raise_error;
END fe_config;
/
```

As far as Delaware can tell, he has set things up so that if anything goes wrong while initializing the package, he will trap and log the error.

Yet when he tries to call the function, he gets an unhandled exception:

```
SQL> BEGIN
  2      DBMS_OUTPUT.put_line
(fe_config.get_worst_excuse ());
  3   END;
  4   /
BEGIN
*
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error:
character string buffer too small
ORA-06512: at "HR.FE_CONFIG", line 3
```

For a solid five minutes, Delaware stares at this simple package, stumped. Then he groans and smacks his forehead. Of course! The exception section that is underneath the package initialization section will trap only exceptions that occur in that initialization section (the executable section of a package). And 20 characters simply aren't enough for the world's worst excuse.

Now, Delaware could simply raise the length of that constant's VARCHAR2 declaration. But he would rather fix the problem in a more long-lasting and fundamental way.

Solution: Don't trust the declaration section to assign default values.

As we've seen, the exception section of a block can trap only errors raised in the executable section of that block. So if the code you run to assign a default value to a variable fails in the declaration section, that error is propagated unhandled out to the enclosing program. It's difficult to debug these problems, too, so you must either:

- Be sure that your initialization logic doesn't ever raise an error. That's hard to guarantee, isn't it?
- Perform your initialization at the beginning of the executable section, preferably in a separate "initialization" program.

Here's what Delaware did with his package:

- ➤ He did no more hardcoding of the VARCHAR2 length. He anchored to a database column instead.
- ➤ He moved the assignment of the default value into a separate procedure, and called this procedure in the package's initialization section.

And here's the code:

```
PACKAGE BODY fe_config
IS
   g_worst_excuse flimsy_excuse.title%TYPE;

   FUNCTION get_worst_excus RETURN VARCHAR2  IS
   BEGIN
      RETURN g_worst_excuse;
   END get_worst_excuse;

   PROCEDURE initialize IS
   BEGIN
      g_worst_excuse := 'The dog ate my homework';
   END initialize;
BEGIN
   initialize;
EXCEPTION
   WHEN OTHERS
   THEN
      fe_errmgr.log_and_raise_error;
END fe_config;
```

Now if that string is too long, the exception section will catch the problem and the error logging will come into play.

Avoid assigning default values in the declaration section, especially if they are function calls or expressions that make it hard to predict the value that will be returned. Instead assign those values in your own, private initialization program that is called on the first line of your executable section. ▲

Oracle ACE Steven Feuerstein is an expert on the Oracle PL/SQL language. He is the author or coauthor of 10 books on PL/SQL (all from O'Reilly Media) including Oracle PL/SQL Programming and Oracle PL/SQL Best Practices. He has been developing software since 1980 and is the PL/SQL evangelist for Quest Software. His current obsession is with improving PL/SQL code quality through automated unit testing. You can find out more about Steven and his PL/SQL-related activities by visiting **www.oracleplsqlprogramming.com**.

Mogens Nørgaard: I never had any critical thoughts about the term "Best Practices" (BP) until Cary Millsap once told me that he much preferred "Worst Practices" (WP). BP sets the bar at a certain level, while WP raises the bar over time.

If we define BP so broadly as to be things like "Look to both sides before you cross a road" or "Take a deep breath before you dive," I'm all for it. Very few people will question this sort of thing, although I was brought up to do left/right/left before crossing the road, while my numerous children are now taught to just do left/right—and although it might make sense to not take a deep breath if you want to be able to sink down to the bottom of your local pool.

If we define BP to be stuff like "Don't leave your hotel room in case of fire" or "Hold your rifle out from your body so you're ready to react," that is something that has changed in recent years, although it seemed fairly fundamental a few years ago.

I'd like to quote a Danish philosopher: "You're always wrong." If you think back, there are very few things we believed some years ago that we fully believe today. We constantly get a little smarter, a little wiser. Things we thought we knew are put on their head.

Our former prime minister Jens Otto Kragh once said, "You've got a point of view until you get a new one."

I would like—without having asked for permission—to point y'all to a few good men, namely Tom Kyte, Jonathan Lewis, Cary Millsap, and such, who never speak about BP, but instead test and demo everything, so that in the end they can say:

"This is how it worked under these circumstances, with these versions, and with this data. But you should always test it yourself under your circumstances."

These fine gentlemen are using BEST EVIDENCE (BE) instead of BP.

As Cary would put it: BE > WP >> BP

OK, it's time for a test in order to wake up the readers:

"Please explain in 42 words or less how anyone can present a paper on best practices about something still in Beta or just released in production."

Please email your answer to **mno@MiracleAS.dk**.

During OOW last year I set up my virtual office at the Chevy's Tex-Mex restaurant right by the Moscone Center in San Francisco so that I could sit there and have good talks (and margaritas) with my friends. Much better than walking around in the City of the Dead Eyes (the exhibition halls) and such places.

It worked very well: People came over, sat down, had a beer or drink or some food, and we talked about good features of the database and other important things in life. Many good ideas came out of it. One of them was from Gunnar Bjarnason, CEO of Miracle Iceland, who suggested that we do a search of the online OOW presentation titles and abstracts for the term "best practices."

So we did. As an old military man, I'll always remember the answer: 762.

A fine caliber in the old days. But also a very frightening number of best practices to follow. I think that number in and of itself speaks volumes about this topic.

Just to make sure you understand me 100%:

BP is BS. ▲

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

Empowering Developers with Hotsos Test Harness

by Karen Morton

Introduction

At Hotsos, we live by the motto "Why guess when you can know?" But over and over again, developers tell me that they are prohibited from accessing v$ views, trace data files, and virtually every piece of valuable information they need to do the job they've been asked to do. They are forced to guess, and it is that guessing that causes so many of the performance firefights in production that Hotsos is often called in to help solve.

What Do Developers Need?

There is a simple list of information needed by developers (or anyone for that matter) in order to optimize performance reliably.

- Statistics—table, column, and index statistics
- Block and Row Selectivity data—data to assess the presence of skew
- Explain plans—to see what plan the optimizer thought it would use
- Execution plans—to see what plan the optimizer actually did use
- Extended SQL trace data—to see the details of exactly how the code executed
- Resource consumption statistics—snapshots of key v$ view data captured during code tests

The Oracle optimizer relies primarily on statistics to make execution plan choices. The developer needs to know what the optimizer knows in order to understand how best to respond to the choices the optimizer makes. The optimizer may make different decisions about plan operations when it thinks a table has a few hundred rows versus several million. Also, the distribution of values in the columns of the tables is an important piece of information to know about the data that SQL is being written against.

Most developers do have access to an explain plan. But it is important to remember that the explain plan is just a guess. It is an estimate of what *should* occur but not a guarantee. I believe the explain plan is best used when viewed in comparison to the execution plan—the actual plan and plan execution statistics—that were used during the execution of a SQL statement. Often, comparing the guess to the actual can point out problems with statistics or syntax choices. For example, if the optimizer thought it was going to be dealing with a table with one thousand rows and ended up having a table with ten million rows, that difference would show up nicely in the row estimates from the explain plan when compared to the row source execution statistics from the execution plan.

Extended SQL trace data is one of the most reliable and accurate ways to locate application problem areas. The trace data can be summarized, and most preferably profiled, to see exactly where the time and resources required by a SQL statement were spent. While many developers could turn on tracing, they don't have access to the trace files that get created without getting a DBA involved to retrieve their trace file.

Finally, the data showing what resources were consumed by a SQL statement completes the picture of how the query performed and how and where it could be improved. Data stored in V$SESSTAT, V$SESSION, VSQL_PLAN, VLATCH, and many others provides key data that can be used in the diagnosis and resolution of performance problems.

The Hotsos Test Harness

In order to help make it easy for all this data to be readily at hand, Hotsos developed a set of simple SQL*Plus scripts that could be used to test any SQL statement or PL/SQL code block and collect all the information discussed above. This tool kit of scripts can be downloaded for free from our website at **www.hotsos.com/educ_downloads.html**. The tool kit contains scripts to display the key table and index statistics information as well as providing a "harness" that captures all the key performance information during a code test and stores it for review and comparison.

Two text documents: Harness Installation Guide.txt and Harness User Guide.txt are included in the download .zip file to guide the new user through the installation and basic use of the scripts. For brevity, I won't go into full detail of the install procedure here; I'll refer you to the installation guide instead. However, I will mention the key elements that are required to be in place for the harness to be used.

The harness installation for an instance has two parts: one part that must be completed by either the SYS user or a specially privileged DBA account and the other part that must be completed by the person wishing to use the harness for testing. The portion that requires SYS or a specially privileged DBA account creates a package (named hotsos_pkg), two tables, two views, a directory, and several public synonyms pointing to these objects. These objects provide the interface between the developer (who will be granted privileges to access and execute these objects) and the trace data files they will create during testing. This will allow each developer to access the trace files they create (and only the files they create) without needing to open the entire USER_DUMP_DEST directory for public access (e.g., _trace_files_public = TRUE) or without requiring the DBA to get involved to locate and provide the traces files back to the developer manually.

Finally, the last step the specially privileged DBA account has to execute is the script named harness_grant_privs.sql. This script must be executed once for each named account that will be using the harness.

The portion of the installation that must be completed by the tester will create several tables and views as well as a handful of procedures that will be used to capture and store test data. The script to execute is called harness_user_install.sql.

Using the Hotsos Test Harness

Once the test harness has been properly installed, using it is simple. First, start the harness by executing start_harness.sql (see the Harness User Guide.txt file for more details). This will turn on the ability to capture all the resource consumption statistics and related test information using the harness.

Let's do a simple example of using the harness. We'll use a simple statement for our test: select * from emp where first_name = 'David'. I'll save this query into a file named emp.sql so I can use the harness to test it. Before I run the test, however, I'll look at the statistics for the table using the script hstats.sql to get an idea of what to expect.

```
SQL> @hstats
Enter the owner name: op
Enter the table name: employees
=========================================================================================
TABLE STATISTICS
=========================================================================================
Owner        : op
Table name   : employees
Partitioned  : no
Last analyzed: 11/21/2007 11:25:54
Degree       : 1
IOT Type     :
# Rows       : 27392
# Blocks     : 370
Empty Blocks : 0
Avg Space    : 0
Avg Row Length: 82
Monitoring?  : yes
=====================================================================================================
COLUMN STATISTICS
=====================================================================================================
Name            Analyzed    Null?  NDV     Density  # Nulls  # Buckets  Sample  Avg Len  Lo-Hi Values
=====================================================================================================
employee_id     11/21/2007  N      27392   .000037  0        1          27392   5        1, 27392
first_name      11/21/2007  Y      26114   .000038  0        1          27392   12       Adam, Winston9991
last_name       11/21/2007  N      27392   .000037  0        1          27392   12       Abel10092, Zlotkey9903
email           11/21/2007  N      27392   .000037  0        1          27392   13       ABANDA10085, WTAYLOR9991
phone_number    11/21/2007  Y      107     .009346  0        1          27392   15       011.44.1343.329268, 650.509.4876
hire_date       11/21/2007  N      98      .010204  0        1          27392   8        AD 1987-06-17, AD 2000-04-21
job_id          11/21/2007  N      19      .052632  0        1          27392   9        AC_ACCOUNT, ST_MAN
salary          11/21/2007  Y      57      .017544  0        1          27392   4        2100, 24000
commission_pct  11/21/2007  Y      7       .142857  18432    1          8960    2        .1, .4
manager_id      11/21/2007  Y      18      .055556  256      1          27136   4        100, 205
department_id   11/21/2007  Y      11      .090909  256      1          27136   3        10, 110
==============================================================================
INDEX INFORMATION
==============================================================================
Index name     : emp_email_uk
Index type     : normal
Last analyzed  : 11/21/2007 11:25:54
Degree         : 1
Partitioned    : no
Rows           : 27392
Levels         : 1
Leaf Blocks    : 120
Distinct Keys  : 27392
Avg LB/Key     : 1
Avg DB/Key     : 1
Clust. Factor  : 23635
Table Rows     : 27392
Table Blocks   : 370
----------------------------------------
Index name     : emp_emp_id_pk
Index type     : normal
Last analyzed  : 11/21/2007 11:25:54
Degree         : 1
Partitioned    : no
Rows           : 27392
Levels         : 1
Leaf Blocks    : 51
Distinct Keys  : 27392
Avg LB/Key     : 1
Avg DB/Key     : 1
Clust. Factor  : 332
Table Rows     : 27392
Table Blocks   : 370
----------------------------------------
=========================================================================================
INDEX COLUMNS INFORMATION
=========================================================================================
Index Name                             Pos# Order    Column Name        Expression
=========================================================================================

emp_email_uk                               1 ASC    email

emp_emp_id_pk                              1 ASC    employee_id
```

I can see that the table has 27,392 rows and does not have an index on the first_name column. With that information, I can determine that the optimizer will likely choose a full table scan as the plan for my query (since no adequate index exists that matches my query's WHERE clause).

Next, I'll test the query using the harness.

```
SQL> @dosql emp emp test1
Snapshot 1 collection OK.
Snapshot 2 collection OK.

Actual PARSE and STAT line data for emp:test1

PARSE #12:c=0,e=46,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=90278465807

STAT #12 id=1 cnt=29 pid=0 pos=1 obj=72006 op='TABLE ACCESS FULL EMPLOYEES (cr=376 pr=0 pw=0 time=1724 us)'

Explain Plan for emp:test1

Plan hash value: 1445457117

---------------------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows    | Bytes  | Cost   (%CPU)   | Time      |
---------------------------------------------------------------------------------------------
|    0 | SELECT STATEMENT  |              |    1    |    82  |  84        (2)   | 00:00:02 |
|*   1 |  TABLE ACCESS FULL | EMPLOYEES    |    1    |    82  |  84        (2)   | 00:00:02 |
---------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FIRST_NAME"='David')

Statistics Snapshot for emp:test1

Type  Statistic Name                                           Value
----- ---------------------------------------------- -------------
Latch cache buffers chains                                       941
      library cache                                              105
      row cache objects                                          147

Stats buffer is pinned count                                       0
      consistent gets                                            376
      db block gets                                                0
      execute count                                                6
      parse count (hard)                                           0
      physical reads                                               0
      physical writes                                              0
      redo size                                                    0
      session logical reads                                      376
      session pga memory                                           0
      session pga memory max                                       0
      sorts (disk)                                                 0
      sorts (memory)                                               0
      sorts (rows)                                                 0
      table fetch by rowid                                         0
      table scan blocks gotten                                   372

Time  elapsed time (centiseconds)                                 10
```

The dosql.sql script is used to execute my test. It has three input parameters: 1) the name of the .sql file that contains the statement I wish to test (recall that I named my file emp.sql), 2) a name for the test "workspace" (a workspace is just a way to help you organize your tests into categories), and 3) a name for the test "scenario" (a scenario is the name you assign to the test you are executing). In this example, I have named my workspace "emp" and my scenario "test1".

Notice that you get three sections of information when the test completes. A section that displays extended SQL trace data (this shows you the actual execution plan for your test query), a section that displays the explain plan (this shows you what the optimizer guessed for the plan operations and row estimates), and a section for the resource consumption statistics (this shows you the detailed statistics about how your database's resources were consumed by your query execution).

I could now create an index on first_name and compare how the performance would be if an index could be used versus the full table scan. I create the index and run the test again.

```
SQL> @dosql emp emp test2
Snapshot 1 collection OK.
Snapshot 2 collection OK.

Actual PARSE and STAT line data for emp:test2

PARSE #12:c=0,e=48,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=1,tim=90352345394

STAT #12 id=1 cnt=29 pid=0 pos=1 obj=72006 op='TABLE ACCESS BY INDEX ROWID EMPLOYEES (cr=33 pr=0 pw=0 time=440 us)'
STAT #12 id=2 cnt=29 pid=1 pos=1 obj=72009 op='INDEX RANGE SCAN EMPL_FNAME_IDX (cr=4 pr=0 pw=0 time=146 us)'

Explain Plan for emp:test2

Plan hash value: 3072888250

------------------------------------------------------------------------------------------------------
| Id  |  Operation                    | Name            | Rows   | Bytes | Cost  (%CPU) |  Time      |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                 |     1  |    82 | 2        (0) |  00:00:01  |
|   1 |  TABLE ACCESS BY INDEX ROWID  | EMPLOYEES       |     1  |    82 | 2        (0) |  00:00:01  |
|*  2 |   INDEX RANGE SCAN            | EMPL_FNAME_IDX  |     1  |       | 1        (0) |  00:00:01  |
------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("FIRST_NAME"='David')
```

```
Statistics Snapshot for emp:test2

Type  Statistic Name                                               Value
----- -------------------------------------------------- -------------
Latch  cache buffers chains                                          267
       library cache                                                 114
       row cache objects                                             147

Stats buffer is pinned count                                          26
       consistent gets                                                33
       db block gets                                                   0
       execute count                                                   6
       parse count (hard)                                              0
       physical reads                                                  0
       physical writes                                                 0
       redo size                                                       0
       session logical reads                                          33
       session pga memory                                              0
       session pga memory max                                          0
       sorts (disk)                                                    0
       sorts (memory)                                                  0
       sorts (rows)                                                    0
       table fetch by rowid                                           29
       table scan blocks gotten                                        0

Time   elapsed time (centiseconds)                                     9
```

Finally, I compare the differences between my first and second tests to see if I can determine which one performed better. I use the diff.sql script to compare the two tests:

```
SQL> @diff
1st workspace: emp
1st scenario : test1
2nd workspace: emp
2nd scenario : test2

TYPE   NAME                                                        emp:test1          emp:test2        DIFFERENCE
-----  -------------------------------------------------- ------------------ ------------------ -----------------
Latch cache buffers chains                                              941                267               674
       library cache                                                    105                114                -9
       row cache objects                                                147                147                 0

Stats buffer is pinned count                                              0                 26               -26
       consistent gets                                                  376                 33               343
       db block gets                                                      0                  0                 0
       execute count                                                      6                  6                 0
       parse count (hard)                                                 0                  0                 0
       physical reads                                                     0                  0                 0
       physical writes                                                    0                  0                 0
       redo size                                                          0                  0                 0
       session logical reads                                            376                 33               343
       session pga memory                                                 0                  0                 0
       session pga memory max                                             0                  0                 0
       sorts (disk)                                                       0                  0                 0
       sorts (memory)                                                     0                  0                 0
       sorts (rows)                                                       0                  0                 0
       table fetch by rowid                                               0                 29               -29
       table scan blocks gotten                                         372                  0               372

Time   elapsed time (centiseconds)                                       10                  9                 1

emp:test1

STAT #12 id=1 cnt=29 pid=0 pos=1 obj=72006 op='TABLE ACCESS FULL EMPLOYEES (cr=376 pr=0 pw=0 time=1724 us)'

emp:test2

STAT #12 id=1 cnt=29 pid=0 pos=1 obj=72006 op='TABLE ACCESS BY INDEX ROWID EMPLOYEES (cr=33 pr=0 pw=0 time=440 us)'
STAT #12 id=2 cnt=29 pid=1 pos=1 obj=72009 op='INDEX RANGE SCAN EMPL_FNAME_IDX (cr=4 pr=0 pw=0 time=146 us)'
```

Primarily, what I find is that by creating the index on first_name, I reduced the overall amount of work (in terms of logical IO) from 376 to 33. Wow! That's an 11x reduction in the amount of resources needed to execute my query. With just this simple test, I can conclude that my query's performance would be significantly enhanced with the addition of the new index on first_name.

The Bottom Line

The bottom line with using this test harness, or any other, is to get the data needed to formulate an informed and reliable solution for improving the performance of your code. The Hotsos Test Harness is a one-stop shop to capture all the test data you need to review query performance and compare alternatives to see which method provides the best overall performance.

Even if you don't use this test harness for your performance testing needs, the key is to use something! If you can't get access to the information you need, then be very direct with those who would deny you that information: you won't be able to do the job properly that they are asking you to do. This set of scripts makes it easy to get started and provides an outline of what you need to be an effective performance analyst. Don't settle for anything less! ▲

Karen Morton is the director of education services at Hotsos Enterprises, Ltd. With over 15 years of Oracle experience as a developer, DBA, consultant, and educator, she speaks frequently at user groups and conferences. In her role at Hotsos, she leads a team of world-class performance analysts to provide the best Oracle performance education curriculum available.

How to Send Email Using Java

by Joel Thompson

Joel Thompson

In this article I'll first provide an introduction to the developer's corner of the NOCOUG Journal and then offer a "how to" on emailing through Java (with JavaMail and with a Java call to /bin/mail).

This is the first article of the developer's corner for NOCOUG; I am honored to be the author of this section, and I hope that you find these articles clear, interesting, and useful. If you have any questions or need clarification about the topics that I present, please feel free to email me at **joel@rhinosystems.com**. For details of my background, please see the short biography at the bottom of this article.

I will generally focus on software development around Oracle, Java, and SOA technologies; however, I may from time to time, and depending on your feedback, offer articles on C/C++, .NET, Perl, PHP, Ruby, Shell scripting, and more. Perhaps an informal survey would be in order here, and as such, feel free to email me with the top three technologies that you'd like me to discuss.

Before I begin with this article's topic, I'd like to share a bit of my philosophy on software development. I have worked with Oracle databases and software development since 1989. I think Oracle is an excellent platform for software development and database technology; however, I also embrace open-source technologies whenever possible—primarily because of the "free" aspect of using open source. It seems that there are a lot of businesses these days that are following an open-source business model, meaning that they develop a product, open source it, and then charge for their "enterprise edition" or add-ons to the base technology. I don't speak for Oracle, but a similar approach is apparently how Oracle is going. While they don't open source their database technologies, they are giving away the Oracle database XE version as a starter system. This enables businesses to use Oracle technologies for free, knowing that an upgrade path to "enterprise edition" is available if they need it. Oracle is also giving away JDeveloper IDE, which is another big plus for developers and businesses, and an excellent strategy for Oracle.

With JDeveloper you can develop Java and Java Enterprise applications that are completely based on open standards and integrated with open-source technologies. There is a proviso, however, that Oracle's JDeveloper is heavily centered around ADF, which is Oracle's proprietary technology for simplifying software development. The real issue with ADF is that you have to deploy ADF libraries and pay for it (if you don't already own a license to Oracle's Application Server). Also, keep in mind that you can deploy ADF applications to other application servers beside Oracle, such as JBoss. I have worked with JDeveloper using ADF; it is a very efficient and timesaving piece of technology, and well worth the cost. However, some of the businesses I consult for would like to develop without ADF, so I am intimately familiar with both environments. I will tend to write about applications that are more open-source-standards based.

Now, on to the topic at hand: creating and sending an email with Java using JavaMail and, alternatively, calling /bin/mail from your Java program.

Prerequisites

First, in order to send mail, you must have an SMTP server that you are going to send the mail through. This server can be running on your machine (typically a Unix box) or some other server. Sendmail and Postfix are two examples of SMTP server software. Also, it is worth noting that SMTP typically runs on port 25; be aware that some ISPs will block port 25 communication from your server (at their routers). If they are blocking port 25 you will get an error message like "no route to server." You should also check your machine's firewall configuration and make sure you can send outbound traffic on port 25. You can test this from your machine by typing "telnet smtp.server.com 25" from the Unix command line. If you get connected, you have clear passage; if not, you have to run an SMTP server on your machine.

Sending Email Using JavaMail

Let's get started with the code that is used to create a simple email and send it via JavaMail. We will authenticate with a username/password to establish a session with the SMTP server and send a message with this session.

You need to have the JavaMail libraries in your CLASSPATH; these can be found in the J2EE library in JDeveloper. (Right-click your project and select Properties and then Libraries, and search for the J2EE listing.). You can also find the JavaMail libraries in other environments like the Java J2EE reference implementation or jboss/client lib directories.

Listing 1: The code to send an email message from Java using JavaMail

```
package com.rhinosystemsinc.tools;

/**
 * Sample prepared by Joel A. Thompson
 * At http://www.rhinosystemsinc.com
 * All rights reserved.
 */

import java.util.Date;
import java.util.Properties;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

// File: com/rhinosystemsinc/tools/SendJavaMail.java

public class SendJavaMail {

  public static void main(String[] args) {
    SendJavaMail sm = new SendJavaMail();
    sm.send();
  }

  /*
   * This is a simple inner class that extends
   * javax.mail.Authenticator
   * used for gathering a username and password.
   */

  class ThePasswordAuthenticator extends Authenticator {
    String user;
    String pw;

    public ThePasswordAuthenticator(String username,
                                    String password) {
      super();
      this.user = username;
      this.pw = password;
    }

    public PasswordAuthentication getPasswordAuthentication() {
      return new PasswordAuthentication(user, pw);
    }
  }

  public void send() {
    String mailer = “myprogram”;
    //comma separated list of recipients.
    String to = “bob@foobar.com”;
    String cc = “bill@ibm.com,mary@oracle.com”;
    String bcc = null;
    String from = “joel@rhinosystemsinc.com”;
    String subject = “Testing mailing from Java “;
    String body = “This is the body of the test message.\r\n”+
                  “Sent on: “ + new Date() + “\r\n”+
                  “Regards, \r\n” +
                  “Joel” ;

    String smtpserver = “smtp.att.yahoo.com”;
    String smtpport = “25”;

    //my username for smtp.att.yahoo.com was a fully
    //qualified email, like joel@att.com.
    String user = “yourSMTPusername”;
    String password = “yourSMPTpasswd”;

    Properties props = new Properties();

    //default to “localhost”
    props.put(“mail.smtp.host”, smtpserver);

    //default to “25”
    props.put(“mail.smtp.port”, smtpport);

    //uncomment to turn on debugging output which can be useful.
    //props.put(“mail.debug”, “true”);

    //javax.mail.Session
    Session session = null;
    if (user != null && password != null) {
      props.put(“mail.smtp.auth”, “true”);
      session =
        Session.getInstance(props,
                        new ThePasswordAuthenticator(user, password));
    } else {
      session = Session.getDefaultInstance(props, null);
    }

    /*
     * using
     * javax.mail.Message & javax.mail.internet.MimeMessage
     * javax.mail.internet.InternetAddress
     */
    Message msg = new MimeMessage(session);

    try {

      msg.setFrom(new InternetAddress(from));

      /*
       * Parse the given comma separated sequence of
       * addresses into an Array of InternetAddress objects
       * Then set as the desired recipient.
       */
      msg.setRecipients(Message.RecipientType.TO,
                        InternetAddress.parse(to, false));
      if (cc != null)
        msg.setRecipients(Message.RecipientType.CC,
                          InternetAddress.parse(cc, false));
      if (bcc != null)
        msg.setRecipients(Message.RecipientType.BCC,
                          InternetAddress.parse(bcc, false));

      //subject line.
      msg.setSubject(subject);

      //set the body of the message, also consider setContent()
      //for Multipart content
      msg.setText(body);

      //the name of your program.
      msg.setHeader(“X-Mailer”, mailer);

      //Date sent from this computer.
      msg.setSentDate(new Date());

      // send the message!
      Transport.send(msg);

      System.out.println(“\nMail was sent successfully.”);

    } catch (Exception e) {
      System.out.println(“Error: “ + e.getMessage());
    }
  }
}
```

Sending Email Using /bin/mail

The next example of sending email out through your Java program doesn’t involve JavaMail, but rather relies on the Unix operating system’s command “/bin/mail”. In this case the machine’s SMTP software needs to be running in order to send out an email.

You can test to see if you have “/bin/mail” installed by simply typing it in the Unix command prompt, as shown here:

```
 % /bin/mail -v -s testing joel@rhinosystems.com
some message body
.
cc:
%
```

The “-v” turns on debugging so you can see where the message system is or isn’t working.

Listing 2: The code to send an email message from Java using “/bin/mail”.

```
package com.rhinosystemsinc.tools;

/**
 * Sample prepared by Joel A. Thompson
 * at http://www.rhinosystemsinc.com
 * All rights reserved.
 */

import java.lang.System;
import java.util.*;
import java.io.*;

public class SendCommandMail {

  public static void main(String[] args) {
    SendCommandMail sm = new SendCommandMail();
    sm.send();
  }

  public void send() {
    //comma separated list of recipients.
    String to = "joel@rhinosystems.com,iggy_fernandez@hotmail.com";
    String subject = "Testing mailing from Java";
    String body = "This is the body of the test message.";

    Process proc = null;
    Runtime rt = Runtime.getRuntime();
    String cmdStr="";
    String osName="";
    Properties props = System.getProperties();

    //the operating systems name as known to java.
    osName = props.getProperty("os.name");

    try {
      //Check to make sure we are not on Windows!
      if (!osName.toUpperCase().contains("WINDOWS")) {
        // assume UNIX
        cmdStr = new String("/bin/mail -s \"" +
                            subject + "\" " + to);
      }
      //execute the command, it will run as the current user.
      //and the "from" string will be the user@machine-name.com
      proc = rt.exec(cmdStr);

      InputStream is = proc.getInputStream();
      InputStreamReader ir = new InputStreamReader(is);
      BufferedReader br = new BufferedReader(ir);

      //set the body of the message.
      OutputStream os = proc.getOutputStream();
      os.write(body.getBytes());
      os.close();

      String line = "";
      //show the output of the command here.
      while ((line = br.readLine()) != null) {
        System.out.println("output:" + line);
      }
      br.close();

    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Concluding Remarks

The above two samples of code are pretty simple, and hopefully will be helpful to you in your software development. I think it's best to learn by example. All of my examples are tested; however, I am sure there are some environments in which there may be problems. Please feel free to email me and I'll do my best to help you. ▲

Joel Thompson is president of RHINO Systems Inc., a software consulting firm serving the greater Sacramento and Bay Area. He has programmed in Java/J2EE since 1997; prior to that he worked as senior programmer and development manager at Oracle Corporation, beginning in 1989. He resides in Auburn, Calif., with his wife and three children, and enjoys snowboarding, jogging, tennis, and many other athletic activities.

Database Management

Develop	Ensure the highest quality code is developed regardless of user skill set or location
Optimize	Maximize code performance and eliminate time-intensive manual tuning processes
Validate	Performance test your code for scalability before deploying to production

Can Toad® Turn Your Team Into Experts?

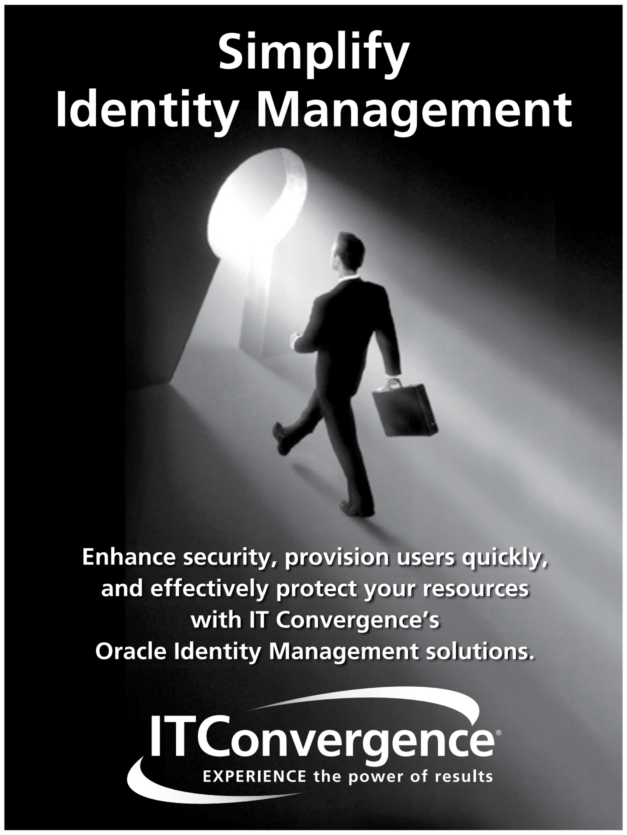
Sit back and relax. You have Toad's development best practices on your side.

You could spend your day tracking down bad code. Or you can take control with Toad® for Oracle.

Promote development best practices in your organization. Read "Implementing Database Development Best Practices" at **www.quest.com/relax**

Toad World community. knowledge. power.

QUEST SOFTWARE®


Simplify
Identity Management
Enhance security, provision users quickly, and effectively protect your resources with IT Convergence's Oracle Identity Management solutions.
ITConvergence®
EXPERIENCE the power of results

Contact us today: **www.itconvergence.com**

Many Thanks to Our Sponsors

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low-cost memberships. If your company is able to offer sponsorship at any level, please contact NoCOUG's president, Roger Schrag, at **www.nocoug.org/contact_us.html?recipient=roger**. ▲

Long-term event sponsorship:

- LOCKHEED MARTIN
- CHEVRON
- ORACLE CORP.
- PG&E

Thank you! Year 2008 Gold Vendors:

- ➤ BEZ Systems
- ➤ Confio Software
- ➤ Database Specialists, Inc.
- ➤ Embarcadero Technologies
- ➤ IT Convergence
- ➤ Network Appliance
- ➤ Princeton Softech
- ➤ Quest Software
- ➤ Roundstone Systems

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

TREASURER'S REPORT

Jen Hong, *Treasurer*

Beginning Balance		
October 1, 2007		**$ 56,101.00**
Revenue		
Membership Dues	1,210.00	
Meeting Fees	1,130.00	
Vendor Receipts	2,500.00	
Advertising Fee	–	
Training Day	–	
Sponsorship	–	
Interest	61.87	
Total Revenue		**$ 4,901.87**
Expenses		
Regional Meeting	10,658.99	
Journal	6,178.90	
Membership	–	
Administration	1,00.00	
Website	–	
Board Meeting	581.21	
Marketing	181.67	
Insurance	500.00	
Vendors	–	
P.O. Box	84.00	
Training Day	–	
Miscellaneous	20.00	
Total Expenses		**$ 19,204.77**
Ending Balance		
December 31, 2007		**$ 41,798.10**

Networking Secrets for Oracle Consultants

by Chris Lawson

Chris Lawson

Those of you who consider yourselves consultants may be tempted to think that you have little need for professional networking. Why bother? After all, like most IT professionals, you probably pride yourself on technical expertise—so why waste time on other "fluff?" While it's possible that you can have a successful career working anonymously behind the scenes, I think you'll be missing out on a lot of opportunities if you don't network. Here's why: Many opportunities in the consulting world come about via your contacts rather than just technical expertise. Many opportunities are filled without ever being advertised in any way.

Here's a surprising fact about connections and opportunities: Even distant connections are often helpful; contacts need not be close friends. Keith Ferrazzi, in his bestselling book, *Never Eat Alone*, points out that about 35% of referrals come from distant connections. This means that you should cast your net pretty wide or say goodbye to a huge chunk of business. If you fail to develop a network of contacts, you are statistically lowering your opportunities.

Of course, you have to be technically competent—there's no question about that. The point is, who will know about your competence? Firms would love to have your expertise, but you have to make it easy for them to find you. You must make a strategic decision to connect to those in your field—decision-makers as well as workers.

Publish White Papers

Most computer professionals have never published anything—nada. Not in a user group journal, not on someone's website, not anywhere. This is a terrible career mistake, especially in light of the various search engines that scour the Web every day. Having your name in print adds credibility—it shows that you've made the effort to research a subject (and besides—you can actually write!). Why not take advantage of the dearth of good writing to put your own talent on display?

Journal editors in user groups are always looking for fresh writers. It's usually just as simple as writing up a good paper in MS Word format and then emailing it to the editor. There's no pay, of course, but that's not what you're interested in at this point. You can even network with others in your field by asking them to review a draft of your work.

It was via publishing white papers that I got a foothold in the publishing world, and was able to write a book on Oracle and easily find a publisher. I had written several in-depth papers on the subject of high-availability systems and published them on the Web. Later, the editor at a major publishing house recognized my background in a certain subject and asked me to review a book proposal. This in turn led him to review my book proposal!

Participate in User Groups

Here's another really easy way to network. Don't be fooled, however—although it's easy, it's also extremely important (as well as fun). I'm always amazed at how few IT consultants participate in user group meetings. I'm not talking about donating lots of time to attend board meetings. I'm just talking about joining and attending. In my field it appears that less than 5% of Oracle professionals in the local area actually attend a given meeting. Once again, why not take advantage of such a gold mine of opportunities? Besides the actual benefit of hearing top speakers in your field, you can easily network with lots of your peers. While networking, don't forget one simple step—ask for referrals. Don't leave this to chance.

Failure to become involved in user groups is a serious strategic blunder. The yearly cost to join most user groups is paltry—perhaps $75 or so. For example, in the Northern California area, one important group for Oracle programmers is **www.nocoug.org**. This volunteer organization publishes an outstanding journal quarterly; the editor is always looking for new writers.

There are similar groups around the world. For example, there is the U.K. Oracle Users Group, found at **www.ukoug.org**. The Los Angeles Oracle Users Group is very active, and is found at **www.laoug.org**. You get the idea—no matter what your niche, there is likely a user group somewhere nearby.

Create Your Own Website

These days, it's pretty easy and cheap to get a rudimentary website going. For a very reasonable cost, you can get lots of exposure. You don't even need a lot of experience in web design. There are lots of inexpensive (or even free) authoring tools available. I used free tools to create my own web pages. You don't have to have an extremely sophisticated website (unless of course that's your field of expertise). After you write a few white papers, simply post them on your website. Then, you can refer interested parties to your site. Also, be sure to add your new URL to your business cards.

Join LinkedIn—Now!

Here's a really easy and free step that you should take right this minute. LinkedIn.com is the #1 professional networking site in the United States. It claims upwards of 11 million members, and many of these members are IT professionals. Recently, LinkedIn began allowing members to upload photos, spawning numerous discussions as to whether photos are helpful to networking. (I can't imagine how it can hurt, so I quickly uploaded mine.)

LinkedIn is free to join (and they don't even spam you!), so sign-up right now. Why in the world would you choose not to connect to other professionals? Ajay Jain has published a useful white paper giving points on networking using LinkedIn: **techgazing.com/TechGazing-WhyLinkedIn-v1.pdf**.

More Suggestions

Besides LinkedIn, there are other popular sites that you may prefer. Two that are very popular are **www.Xing.com** (for European networking) and **www.Ecademy.com**. Each of these sites has its own flavor. Check them out and sign up!

Also, there are many good networking books. I've already mentioned my personal favorite, *Never Eat Alone*, by Keith Ferrazzi. Another choice is *Network Your Way to Success*, by John Timperley.

There are numerous blogs and other interactive forums that can increase your name recognition. In my field, Oracle's OTN forums are an excellent place to share your knowledge.

Once you build your own network, be sure to maintain your contacts. What good is it to build a network of friends and colleagues and then let it all slip away? I like the idea of "pinging." This simply means reaching out regularly to touch base with your friends and acquaintances. I have found that most everyone enjoys hearing from old friends and reestablishing contact.

Don't forgot to reach out and make connections in your off hours. Sports or social groups are a great place to make important contacts. It's hard to beat golf for all-around appeal to a wide number of people; men and women of all ages can play this game. If you really hate golf, try something else—maybe paintball or laser tag is right for you. Besides professional/career blogs, there are all kinds of forums for hobbies and sports.

Finally, when networking, don't worry about whether or not someone is able to help you in return. Develop a reputation for assisting others, without seeking payment in turn. As Keith Ferrazzi suggests, don't keep score. Networking is not a win-lose negotiation where one person benefits at the cost of another. Your purpose is not to exploit others but to help everyone. Always be on the lookout to give someone a leg up—even if you don't benefit. Perhaps this means sending an email reference or offering some encouragement to someone looking for a position. We're all winners when you genuinely invest yourself in others. ▲

Chris Lawson is an Oracle performance consultant who lives with his family in Dublin, California. He is the author of The Art & Science of Oracle Performance Tuning. *Chris's website is* **www.OracleMagician.com**. *In his spare time, Chris enjoys golf, choral singing (bass), and throwing Frisbees to Morgan (the resident border collie). Chris can be reached at* **Chris@OracleMagician.com** *and welcomes all LinkedIn invitations.*

ROUNDSTONE
ROCK SOLID RELATIONSHIPS

Roundstone Systems is Northern California's HP Gold Business Partner authorized in HP's Oracle Solution Network Program.

HP, **Oracle** and **Roundstone Systems** have joined forces to help you reap the benefits of IT consolidation.

HP BladeSystem Servers:

- Increase agility - get products to market faster
- Reduce costs - attain higher system utilization, share resources, reduce management and maintenance
- Increase the overall effectiveness of computing systems and networks

510-764-2080
Gold Business Partner
www.roundstonesystems.com

Enterprise data. Aligned.

You dedicate huge resources to enterprise applications like Oracle,® PeopleSoft,® JD Edwards,® Siebel.® And every year they grow in size and complexity — increasing maintenance challenges. Now you can take back control with Princeton Softech Optim.™ Optim gives you the ability to align application data management to performance goals. Optimizing results, mitigating risk and controlling the cost of every IT investment — servers, software, storage, networks and people. Is this heaven? No, it's the power of Enterprise Data Management.

Learn more at princetonsoftech.com.

princeton softech
The Power of Enterprise Data Management

Bag o' Tricks

by Danny Chow

Danny Chow

For various reasons, DBAs will need to examine characteristics of the objects defined in database. The most common approach is to look at the DDL defining them. For the DBA not having the luxury of third-party tools, we reconstruct these DDL from dictionary views available to us. This is prone to inaccuracy because of new or changed object characteristics introduced to different versions over time.

Oracle does provided a useful yet underutilized package for this purpose, and it could do much more than providing you an accurate DDL, so let's "rediscover" DBMS_METADATA. This function was introduced in V9.0.1 and has been mostly forgotten since then.

DBMS_METADATA package provides a spectrum of useful functions; because of article length, I will provide some examples on GET_DDL, GET_DEPENDENT_DDL, and GET_GRANTED_DDL based on some real-life scenarios.

DBMS_METADATA could handle different object types. TABLE, INDEX, VIEW, PACKAGE, PROCEDURE, FUNCTION, SYNONYM, TABLESPACE, ROLE_GRANT, and OBJECT_GRANT are the ones I happened to use the most.

To make sure the output is correctly displayed, issue "set long" to a big number first; e.g., set long 100000.

Scenario 1: It's your first day of work and there is no document about the application schema, so you decide to start with the tables, sequences, and indexes, etc.

```
SQL> select DBMS_METADATA.GET_DDL('TABLE',table_name,'TEST')
      from   dba_tables
      where  owner = 'TEST';

CREATE TABLE "TEST"."ACCOUNT"
  (     "ACCOUNT_ID" NUMBER(18,0) NOT NULL ENABLE,
                 :
         CONSTRAINT "PK_ACCOUNT" PRIMARY KEY ("ACCOUNT_ID")
                 :
TABLESPACE "TEST_IDX"   ENABLE,
         CONSTRAINT "FK_MKT_CD" FOREIGN KEY ("MKT_CD")
                 :
TABLESPACE "TEST_DATA"
```

Scenario 2: You'd like to determine the default space allocation of a particular tablespace.

```
SQL> select DBMS_METADATA.GET_DDL('TABLESPACE','TEST_DATA') from
dual;

CREATE TABLESPACE "TEST_DATA" DATAFILE
'/u01/oradata/testdb/test_data01.dbf' SIZE 1073741824
LOGGING ONLINE PERMANENT BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL AUTOALLOCATE SEGMENT SPACE
MANAGEMENT AUTO
```

Scenario 3: So, what are the grants on a table?

```
SQL> select DBMS_METADATA.GET_DEPENDENT_DDL('OBJECT_
GRANT','ACCOUNT') from dual;
GRANT DELETE ON "TEST"."ACCOUNT" TO "UPDATE_ROLE"
GRANT INSERT ON "TEST"."ACCOUNT" TO "UPDATE_ROLE"
GRANT SELECT ON "TEST"."ACCOUNT" TO "UPDATE_ROLE"
GRANT UPDATE ON "TEST"."ACCOUNT" TO "UPDATE_ROLE"
```

Scenario 4: You want to determine all the roles granted to a user and the object privileges for the roles granted.

```
SQL> select DBMS_METADATA.GET_GRANTED_DDL('ROLE_
GRANT','APPS') from dual;

GRANT "CONNECT" TO "APPS"
GRANT "UPDATE_ROLE" TO "APPS"

SQL> select DBMS_METADATA.GET_GRANTED_DDL('OBJECT_
GRANT','UPDATE_ROLE') from dual;

GRANT UPDATE ON "TEST"."ACCOUNT" TO "UPDATE_ROLE"
       :
GRANT SELECT ON "TEST"."ACCOUNT_SEQ" TO "UPDATE_ROLE"
```

The following guides provide detailed descriptions of the DBMS_METADATA package; I encourage you to check them out.

➤ *Oracle Database PL/SQL Packages and Types Reference* contains the complete specifications for the DBMS_METADATA package of functions.

➤ *Oracle Database Utilities* contains detailed examples of the use of the Metadata API.

In summary, DBMS_METADATA provides much more than the examples listed above, and I am sure it will make your daily work life much more productive. I think it will only get better in the versions to come, so use it often and make this utility your personal Bag o' Tricks. ▲

Danny Chow is an independent Oracle consultant with many years of DBA and development experience. He holds certifications in Oracle 7, 8, 8i, 9i and 10g. His email address is **dannychow@earthlink.net**.

NoCOUG Winter Conference

Session Descriptions

For more detailed descriptions and up-to-date information, see **www.nocoug.org**.

—Keynote—

The Future of High Availability

Juan Loaiza, *Oracle Corporation* 9:30–10:30

Your business needs to be online 24/7, 365 days a year. If critical applications, servers, or data become unavailable, your entire business could be jeopardized. Lost revenue, dissatisfied customers, penalties, and negative press will have a lasting effect on your organization's reputation. Oracle Database 11*g* can protect your business from negative outcomes due to planned and unplanned downtime, including the most common cause of failure—human error. In addition, Oracle's Maximum Availability Architecture framework provides clear and concise guidance on implementing best practices using Oracle's proven high-availability technologies. During this session, Juan will highlight Oracle's continuing evolution of technology for achieving high availability. He'll discuss recent development innovations and present a roadmap to help your company invest in guarding against failures to meet your quality of service obligations at the lowest cost.

—Auditorium—

DB Time-Based Oracle Performance Tuning: Theory and Practice

Editor's Pick

Uri Shaft and Graham Wood, Oracle Corporation 11:00–12:00

Oracle 10*g* formally introduced the fundamental concept of DB Time as part of the Server Manageability effort. This concept underlies or is significantly used by many of the manageability technologies of the Diagnostic and Tuning packs in both 10*g* and 11*g*, including ADDM, SQL Tuning Advisor, Access Advisor, and Enterprise Manager. Less prominently but no less importantly, the concept of DB Time is intended to be used as the new lingua franca for Oracle performance tuning. This session will introduce the abstract theory of DB Time and its time-normalized sibling Average Active Sessions. The process of performance tuning using DB Time will be discussed and compared with other current methodologies including those based on wait events and SQL trace. The session will discuss the Active Session History (ASH) technology and its critical relationship to quantifying the expenditure of DB Time in an active system across many dimensions of interest to performance analysts. The automatic analysis of DB Time by the Automatic Database Diagnostic Monitor (ADDM) is also discussed. Example usages of Enterprise Manager's visualization of DB Time will also be presented.

Simplified SQL Performance Management with Automatic SQL Tuning and Real-Time SQL Monitoring

Pete Belknap, Oracle Corporation. 1:00–2:00

SQL tuning is a critical aspect of database performance tuning. Unfortunately this has traditionally been an inherently complex activity, requiring a high level of expertise in multiple domains. Furthermore, SQL tuning is both time consuming and repetitive due to the large volume and evolving nature of the SQL workload and the underlying data. Oracle 10*g* introduced the SQL Tuning Advisor to provide quick recommendations for solutions to common SQL performance problems, including the application of SQL profiles to transparently improve problematic execution plans without rewriting SQL. Oracle 11*g* automates this process through the Automatic SQL Tuning feature, wherein SQL Tuning Advisor runs automatically in system maintenance windows, tests its own recommendations, and can implement SQL profiles when they show a large performance improvement. The result is that Oracle can now automatically recognize and fix many SQL performance problems with minimal or no human intervention. Monitoring long-running and complex SQL during execution and understanding where it spends execution time is another major challenge for the DBA. Simply finding long-running SQL executions can be difficult because statistics are typically aggregated across all executions. Parallel execution plans are particularly difficult to analyze as multiple sessions can simultaneously be working on them, possibly across multiple database instances in RAC environments. Drilling down to discover where time is spent at the plan operation level has also been almost impossible. Oracle 11*g* introduces the Real-Time SQL Monitoring feature to help the DBA by giving a complete and accurate picture of individual SQL executions by exposing the interesting new measurements at each stage of the execution pipeline. In this session we take a deep dive into these two exciting new technologies and see specific examples where they help simplify SQL monitoring and tuning immensely. We will demonstrate how these features have lowered the barrier to entry for competent SQL tuning, introducing new approaches that are both more scientific and more accessible to the broad base of Oracle customers.

SQL Performance Analyzer: Testing the Impact of System Changes on SQL Performance

Khaled Yagoub, Oracle Corporation 2:30–3:30

SQL Performance Analyzer is a component of the Oracle 11*g* Real Application Testing option. This novel technology enables flexible testing of database change scenarios on SQL execution performance. System changes like database upgrade, initialization parameter settings, schema changes, and gathering optimizer statistics may affect many execution plans in a large workload. Today customers attempting to achieve positive benefits through system-level changes can sometimes end up with unanticipated side effects on the execution of some SQL or other due to the complexity of predicting all possible effects over workloads consisting of many SQL statements. SQL Performance Analyzer offers a comprehensive solution that enables users to forecast and analyze how a system change will impact SQL query plans over a large workload, and assess the run-time performance prior to production deployment. The technology identifies potential problems that may occur due to a change and provides solutions for avoiding degraded

SQL performance that is predicted to occur under the change. As well, it measures performance benefits achievable by the change through quantitative estimates of the system's performance in the new environment with high confidence. This comparative analysis of SQL workload response time allows clear and easy assessment of the change. In this session we describe the architecture of the SQL Performance Analyzer, its usage model, and its integration points with other Oracle database components to form an end-to-end solution for managing SQL execution performance in the face of ever-changing system environments.

What Is New in Oracle SOA Suite

Joanna Schloss, *Oracle Corporation* 4:00–5:00

This session provides an overview of the Service Component Architecture Assembly Model and how it simplifies the end-to-end lifecycle management of SOA composite applications. Application designers can now design, package, version, and manage their SOA composite applications (web services, BPEL, Enterprise Service Bus flows, and the like) as one single entity. These capabilities will be illustrated with a demonstration of Oracle SOA Suite 11*g*.

—Room 102—

Oracle Gems

Daniel Morgan, *University of Washington* 11:00–12:00

Oracle Ace Director Daniel Morgan will present a variety of Oracle Gems—10*g* and 11*g* database features that have been hidden from view and that, with a little polish, may be of great value. The gems, some for developers, some for DBAs, will be presented live in SQL*Plus.

Web 2.0

Bradley Brown, *TUSC*. 1:00–2:00

You've heard the term Web 2.0 and might be wondering exactly what it means to you and your business. You might be wondering what Web 2.0 includes. On the other hand, you might be right in the middle of it all, but noticing that more and more Oracle products seem to offer Web 2.0 entry points. Or maybe you're not using Oracle technology for anything but your backend database. You might be wondering which other products should be under consideration, which should be avoided, or what's a good entry-point solution.

Resolving Conflict with the Arrow of Truth

Bradley Brown, *TUSC*. 2:30–3:30

Have you ever run into a conflict at work or at home, but had difficulty resolving the problem? By using the methods described in this presentation (the arrow of truth), you will be able to work through issues that otherwise may appear unresolvable. This method works not just at work but at home too.

Oracle Annoyances for Geeks: Deadlocks and Livelocks

Iggy Fernandez, *Database Specialists* 4:00–5:00

Deadlocks are not well understood and the trace file generated by Oracle when a deadlock occurs suggests that a deadlock is a problem "due to user error in the design of an application or from issuing incorrect ad-hoc SQL." In fact, sessions can deadlock even if they are not contending for the same data and a session can even deadlock with itself! We discuss this and other misconceptions about deadlock, and demonstrate multiple ways in which deadlock can occur.

—Room 103—

Birds Do It: Migrating Forms to Java EE Web—A Case Study

Peter Koletzke, *Quovera* 11:00–12:00

This case study examines the migration of an enterprise-level, character-mode, Oracle Forms 6.0 application to Fusion ADF web technologies. It explains the decision points for choosing Fusion ADF and focuses on how ADF Business Components provide the core pivot point for an ADF user interface and PL/SQL back-end business rules code. The presentation also describes and offers insight on the techniques used to migrate the skills of staff Forms developers to the new technologies. Finally, it lists successes and lessons learned along the way.

Whither Business Logic?

Avrom Roy-Faderman, *Quovera* 1:00–2:00

Where should you put business logic in your web applications? Database people will tell you it should go in the database; J2EE types will say it belongs in the model layer of the application. But there's more to this than a holy war; the question of where to put your business logic can be approached rationally, depending on features of the logic and application. This talk covers principles for placement of business logic in the database or application, as well as the question of logic in the client tier. The talk will also cover Oracle JDeveloper techniques for accommodating different business logic placement.

Oracle Adaptive Access Manager: What, Why, How

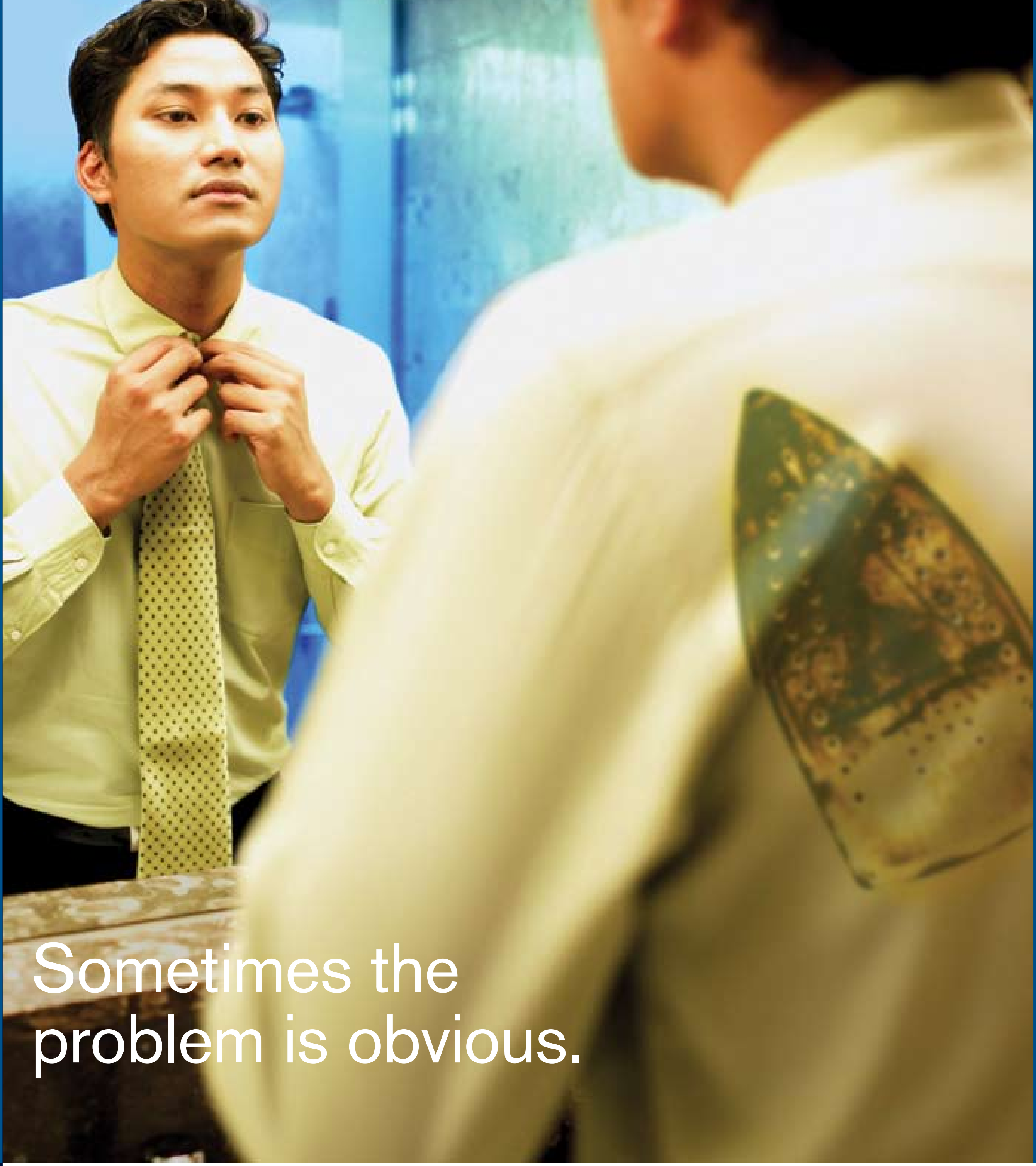
Dan Norris, *Piocon* 2:30–3:30

You're a hacker. You want data. Shouldn't be too hard—just guess a password or two and you can find lots of confidential information. If you're lucky, you'll find a weakly secured web application and watch network traffic as someone logs in to find their username and password. Now that you have the credentials, all you need to do is log in. When you attempt log-in, you find that you're denied access or maybe you're asked to type in the PIN code from your login token (which you don't have). This scene was brought to you by Oracle Adaptive Access Manager (OAAM). Oracle acquired Bharosa in October 2007 to fill the need for a real-time fraud detection capability in the Oracle Identity Management Suite. OAAM uses a database of heuristics and pattern matching to find the "bad" guys. What happens to them is up to you, the OAAM administrator. Come to this session to learn how OAAM protects applications from fraudulent use without any client-side installation.

High-Availability Options for Oracle Database

Dan Norris, *Piocon* 4:00–5:00

From Oracle RAC to cold backups, this session will introduce the various Oracle and third-party features, products, and options that provide high availability for Oracle Database. IT Managers, DBAs, and system architects will benefit from the information provided in this technical session. To help you determine the best choice for your environment, we'll focus on the advantages and disadvantages of each option and discuss the factors that commonly influence the decision-making process. ▲



Usually, it's harder to pinpoint.

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

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

A smarter solution makes everyone look brilliant.

CONFIO SOFTWARE

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

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

NoCOUG Winter Conference Schedule

February 19, 2008, at Oracle Corporation, Redwood Shores, CA

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

Session descriptions appear on page 25.

8:00–9:00 a.m.	Registration and Continental Breakfast—Refreshments served
9:00–9:30	**Welcome:** Roger Schrag, NoCOUG president
9:30–10:30	**Keynote:** *The Future of High Availability*—Juan Loaiza, Oracle Corporation
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Auditorium: *DB Time-Based Oracle Performance Tuning: Theory and Practice* —Uri Shaft and Graham Wood, Oracle Corporation
	Room 102: *Oracle Gems*—Daniel Morgan, University of Washington
	Room 103: *Birds Do It: Migrating Forms to Java EE Web—A Case Study*—Peter Koletzke, Quovera
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Auditorium: *Simplified SQL Performance Management with Automatic SQL Tuning and Real-Time SQL Monitoring*—Pete Belknap, Oracle Corporation
	Room 102: *Web 2.0*—Bradley Brown, TUSC
	Room 103: *Whither Business Logic?*—Avrom Roy-Faderman, Quovera
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Auditorium: *SQL Performance Analyzer: Testing the Impact of System Changes on SQL Performance* —Khaled Yagoub, Oracle Corporation
	Room 102: *Resolving Conflict with the Arrow of Truth*—Bradley Brown, TUSC
	Room 103: *Oracle Adaptive Access Manager: What, Why, How*—Dan Norris, Piocon
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Auditorium: *What Is New in Oracle SOA Suite?*—Joanna Schloss, Oracle Corporation
	Room 102: *Oracle Annoyances for Geeks: Deadlocks and Livelocks*—Iggy Fernandez, Database Specialists
	Room 103: *High-Availability Options for Oracle Database*—Dan Norris, Piocon
5:00–	**NoCOUG Networking and Happy Hour at Chris' New Harbor Bar, 150 Harbor Blvd., Belmont**

Cover photos © Tom Wagner, Tata Nano.
The *NoCOUG Journal* design and production: giraffex inc.

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

NoCOUG
P.O. Box 3282
Danville, CA 94526

RETURN SERVICE REQUESTED