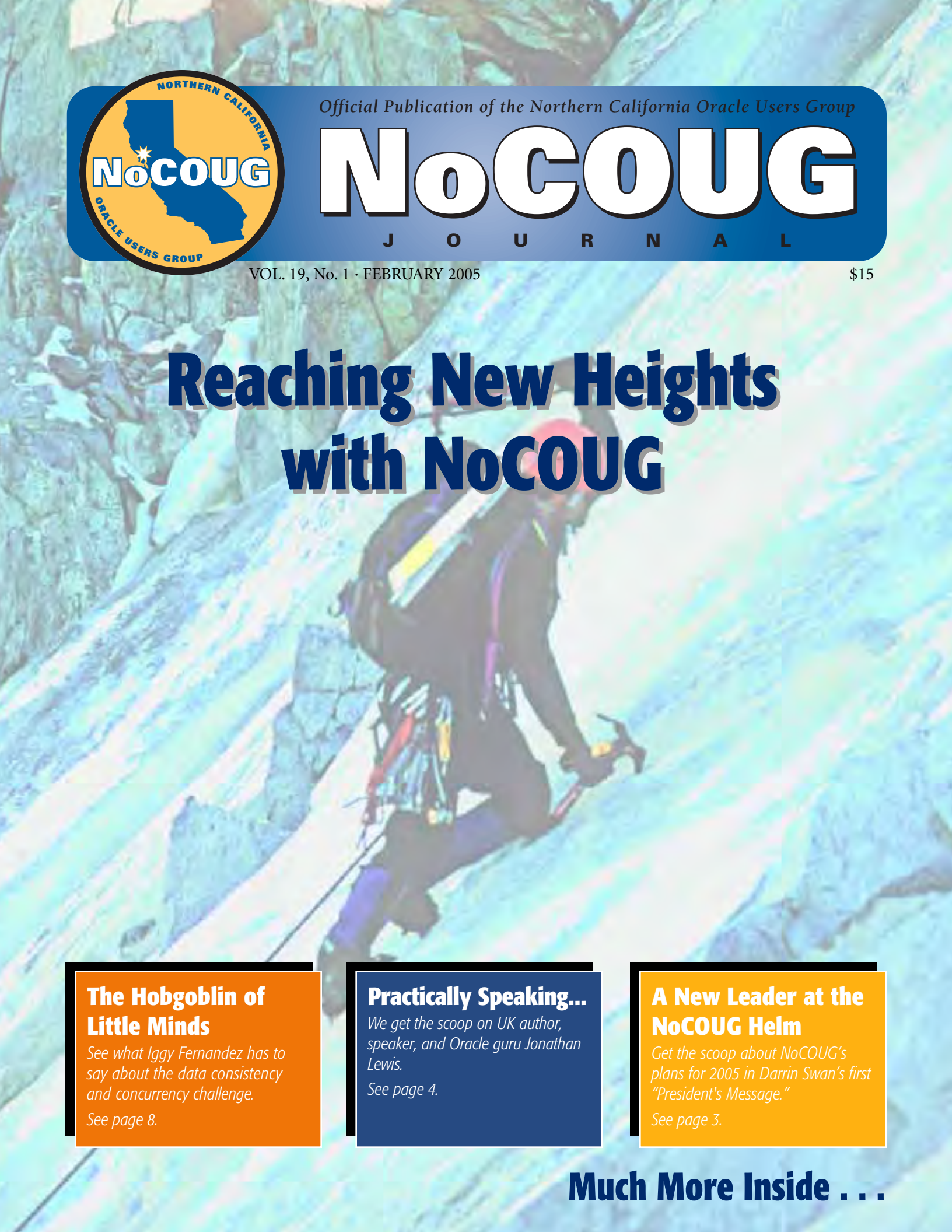
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

VOL. 19, No. 1 · FEBRUARY 2005 $15

Reaching New Heights with NoCOUG



The Hobgoblin of Little Minds

See what Iggy Fernandez has to say about the data consistency and concurrency challenge.

See page 8.

Practically Speaking...

We get the scoop on UK author, speaker, and Oracle guru Jonathan Lewis.

See page 4.

A New Leader at the NoCOUG Helm

Get the scoop about NoCOUG's plans for 2005 in Darrin Swan's first "President's Message."

See page 3.

Much More Inside . . .

Reaching New Heights with NoCOUG

In the past few years of my involvement with NoCOUG, I've seen the organization grow stronger and more successful in achieving its mission of being "dedicated to the education and representation of the users of Oracle Corporation's database and tools software." The NoCOUG board works very hard to help our membership reach new heights in their Oracle careers.

Our conferences continue to attract top speakers, and we have four excellent conferences planned for this year with outstanding learning and networking opportunities. We have also elected a Training Day Coordinator for 2005, so we can build on the two very successful training days we held last year. We will also work to make sure the *NoCOUG Journal* is full of helpful tech tips and interesting articles that will help you with your education of Oracle-related items.

Here's wishing you much success in reaching new heights in your career in 2005. NoCOUG is here to help you get there.

Happy New Year!

—Lisa Loper,
NoCOUG Journal Editor

Table of Contents

Publication and Submission Format

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

The submission deadline for the upcoming May 2005 issue is April 20, 2005. Article submissions should be made in electronic format via email if possible. Word documents are preferred.

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

Happy New Year!

Darrin Swan

"My main goals for NoCOUG 2005 are to have fun and to ensure that NoCOUG members are well informed and armed with the latest Oracle knowledge."

Welcome to the 2005 Northern California Oracle Users Group. My name is Darrin Swan, your new NoCOUG President. I would like to start the year by thanking Roger Schrag and the rest of the 2004 NoCOUG Board of Directors for making 2004 a valuable year for Bay Area Oracle professionals. With another successful year behind us, the bar has been raised and a new benchmark set for your 2005 NoCOUG Board of Directors. We are ready for the charge.

To start the year out right, Oracle Corporation is hosting its Winter Conference on Tuesday, February 8, 2005, at Oracle's Conference Center located in Redwood Shores. This full-day event will start with a keynote address by Thomas Kurian, SVP Development from Oracle Corporation, followed by a powerful lineup of technical presentations from Oracle experts and gurus such as Gaja Vaidyanatha, David Austin, and Jeffrey Jacobs. We have three presentation tracks covering topics ranging from 10g backup and recovery to RAC, data modeling, and 10g data warehousing. Kick off the New Year by joining your friends and fellow Oracle professionals. Visit **www.nocoug.org** for additional event details.

Over the last 4 years I have met many of you, but I thought I might offer a bit of background for those of you who do not yet know me. I have been involved with NoCOUG for about 4 years now and just finished the last 2 years as vice president. Since my first conference in February 2001 as a vendor, I felt a part of an exciting technology movement and community. You have probably noticed me running around during the conferences (as I sometimes cut conversations short, and for this I apologize) to help ensure your optimal conference and educational experience. NoCOUG has helped me to expand my network, and I have made many new friends throughout the years.

In 1992 I began my career in information technology as a computer operator at Hewlett Packard managing HP3000 Unix systems. As many of you know, over the past few years I worked at a startup company focused on bringing to market Oracle database performance optimization solutions. My experience with Oracle spans helping customers deploy proven best practices during application development to increase code quality as well as during testing and ongoing production management to achieve maximum database application performance, scalability, and capacity. If you ever want to chat about Structured Query Language, let me know. I can talk your ear off.

Through a successful acquisition in May 2004, I now enjoy a new and exciting career with Quest Software Inc. Among other things, I manage Quest's business relationship with Oracle Corporation and work on Quest's Business Development and Alliances team. My role will be to NoCOUG's advantage as I work diligently to stay informed about current and future Oracle technologies.

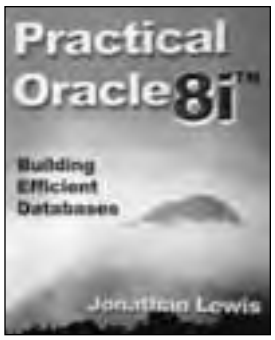
My main goals for NoCOUG 2005 are to have fun and to ensure that NoCOUG members are well informed and armed with the latest Oracle knowledge. Thank you for making our Oracle Users Group a great community to be a part of. See you on February 8.

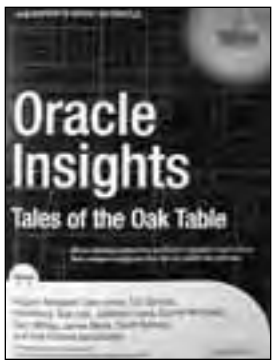
Make it a great year!

Practically Speaking

An Interview with Jonathan Lewis

There are few authors who have written Oracle reference material with the timeless quality found in *Practical Oracle 8i: Building Efficient Databases* (Addison-Wesley, 2001). Jonathan Lewis is an author, teacher, consultant, and owner of the UK-based JL Computer Consultancy, which provides consulting services and training seminars. In fact, last summer thirty NoCOUG members were fortunate enough to attend a one-day tutorial he offered, "Understanding and Assisting the Cost Based Optimizer." Jonathan also contributed a chapter to the informative and entertaining book *Oracle Insights: Tales of the Oak Table* (Apress, 2004).





And, if you don't have either of those books, you can read one of the many articles he's written for the online journal dbazine.com. I had the honor of meeting and interviewing Jonathan after attending his one-day tutorial. And for those who missed his speech at the NoCOUG Summer Conference, hopefully we can convince him to come back again!

Where did you get your start working with computers?

Jonathan: I actually got started when I was 12. I was attending a very forward-looking school, and they bought a computer so they could teach computer science. It was a Hewlett Packard—very expensive—and had a whole 196 instruction spaces with virtually no programming ability, but that's when I got hooked.

Afterward, I went to Oxford University and studied mathematics, then stayed on for an extra year to do a teaching degree. I spent three years teaching mathematics and computer science, which I suppose you could say is where my first official use of computers appeared.

Then I was made an offer I couldn't refuse. I was about to move to a different school, and I received an offer from a businessman to join him in his computer business. I really thought it would be a great opportunity, but I wasn't a terribly good salesman. As you can imagine, that opportunity didn't last very long, but that's when I started doing a little freelance work writing programs. You see, although I wasn't a terrific salesman, businesses that bought the computers always wanted them programmed, so I turned that into freelance work for a while. That's when I was introduced to Oracle, as a freelancer. A company sent me this box of software and said, "This thing called Oracle just came over from America, and we'd like to find out what it does. So could you investigate it for us and take a couple of months and write a report at the end of it?" So I actually sat down and played with Oracle 5.1 for the best part of 2 months. I read the manuals and experimented, and I've never looked back.


Jonathan Lewis

Since you began your studies of computers before college, what drove you to study mathematics instead of computer science?

Jonathan: I guess computers seemed like fun, more of a hobby, but mathematics was interesting and seemed like a more practical career direction. At the time, it never occurred to me that one could actually take computers seriously, get a degree in computer science, and make a living using computers. But mathematics was a very interesting and absorbing subject, so it hasn't hurt.

How are you able to use your background in mathematics and apply it to your work with Oracle?

Jonathan: Study in mathematics taught me how to make sense of a problem. The pattern of thinking you learn teaches you to break the problem down into many smaller problems or steps. Taking this step-by-step approach to find a solution rather than having a single complicated problem simplifies the process. Solving half a dozen simple problems seems easier than a single complicated problem.

In the work you've written, either in an article or in a response you posted on the Oracle-L list or Metalink, it sounds like you spend a fair amount of time running trial-and-error-type experiments. If that's true, what do you find helpful about this approach to using and understanding Oracle?

Jonathan: I want to know how something is done. There's a great deal in Oracle that's incredibly clever. When Oracle first appeared on my desk 16 years ago, I began reading

the manual, and thought, this is fantastic architecture. From that experience, I've kept the pattern of looking for the new features, looking for the bits and pieces of clever work and the capabilities, and saying, "Well, that's really a smart move, and I wonder how that's being done." Because in a way it's almost like appreciating a work of art. You look at it and you think: I can't quite put a finger on it, but it's been done really well, it's fantastic, it's elegant, it's wonderful. And simply appreciating the elegance of something is enjoyable itself. And when you've got something like Oracle, I find that you have to dig in, you have to really think about it and take things apart and discover how it's been done. I get pleasure in seeing how it's been done and appreciating how well all the pieces work together.

From your work as a consultant, do you have any assignments that stick out as memorable or challenging?

Jonathan: I particularly enjoy starting from scratch. I like when a client asks me to consult at the earliest stages of a project. What they want seems impossible, and projects with the most challenging requirements tend to be the most interesting. You walk in, and the client describes what they want, which includes the leading-edge technology, with the latest versions of software. Translating these extremely challenging requirements into something that can actually work in Oracle is exciting. Then I like to develop a plan for testing possible breaking points and running these tests to see if the architecture really does break or if there are any potential traps. This is an important step in moving from the image of what's needed to the actual mechanical implementation, since there are always pitfalls that inevitably exist right at the leading edge. Unfortunately, I can't tell you about any of them in detail because of confidentiality agreements, but the projects do make for interesting work.

Working as a consultant would seem like a nice balance to the teaching work you do. The best teachers seem to also have practical work experience. Do you also try to teach your clients?

Jonathan: Absolutely. My intent when I'm on a consulting assignment is to make myself redundant. The teaching aspect is very important, passing on the information. Looking at how I spend my time, I spend roughly 1/3 consulting, 1/3 doing tutorial-related work (teaching, keeping material up to date), and 1/3 doing what I like to call R&D—experimenting with the software. In many ways, the consulting work I do helps because I find out what people are doing and need. The consulting also helps me teach because I find out what people are interested in knowing. And the R&D, experimentation with the software, can be the real fun. It can be quite mundane if you never have time to appreciate the architecture.

In looking at your website, I've noticed you teach several classes. How did you become interesting in teaching, and what Oracle topics do you focus on?

Jonathan: The training aspect actually came after writing the book *Practical Oracle 8i: Building Efficient Databases.* I realized having someone talk about the material and answer questions about it might add value or provide a way to reinforce what's on the pages of the book. So, I took portions of the book and turned them into a 3-day seminar, "Optimizing Oracle: Performance by Design." *[For more details, the website is listed at the end of this article.]* The seminar helps to fill in the details, adds background, and gives all sorts of insights. But the seminar is intensive, and not everyone may be ready or or need that level of detail. So, I also developed several one-day tutorials to help address different people's learning styles and time availability.

All the material I teach focuses on the basic Oracle engine. If you have an understanding of the basic engine, you can work outward from there to identify problems with your application.

All the material I teach focuses on the basic Oracle engine. If you have an understanding of the basic engine, you can work outward from there to identify problems with your application. For instance, I've been asked if I know Oracle Financials. No, I don't know Financials, but it's just an application that runs on top of Oracle. But because I understand the Oracle engine, I can identify what's causing the problem. I can determine where resources are being used too aggressively, how the application is using indexes, tables, and tablespaces, etc. The specific application is a minor detail; it's just a way to get distracted.

Do you find it a challenge to try and fix a tuning problem in an Oracle database running a third-party application, which often treats the database as a black box?

Jonathan: I find, whether it's a third-party application or a custom application, it's too easy to say, "This is wrong; it should have been done this way." What's important to most clients is finding the most cost-effective way to say, "We have this limitation; here is a reasonable, relatively low-risk and inexpensive way to make a significant improvement." It may not be the "right" thing to do, but it's a good way to engineer around the problem rather than just saying, "It's wrong; do it again." It's always, "What's the least expensive way of making the most improvement?" So you never actually produce something that's perfect, but you can produce something that is really quite good. There is always a trade-off between how much improvement you can get and how little work can you do to get it [cost to the client]. Finding a nice balance is interesting.

Since having adequate statistics is one of the keys to helping the CBO, what factors should be considered when deciding how often to run table statistics?

Jonathan: It is important for the optimizer to understand your data. However, making sure the optimizer understands the data does not necessarily reveal a correct frequency for running statistics or a correct percentage to calculate against. Although, if you've got a big time window and it won't cause a performance problem, there should be no harm in computing the statistics every day of the week. So,

So, all of a sudden, instead of having six months of doing nothing except writing a book, I was working flat out and writing a book evenings and weekends.

if you can afford the resources and time, go ahead. If you have a limited time window or you can't afford the resources, you should aim to work out the minimum resource you can spend to get the best-quality statistics. Interestingly, you can often get quality statistics with a small sampling of data. There are the few special cases where the optimizer needs some help, where something about the data is sufficiently unusual or variable. It goes back to knowing the application and knowing your data. In the cases where you know the data, you can give the information you've got to help the optimizer and save your system resources.

How important is testing and/or conducting your own experimentation?

Jonathan: I think it comes back to the documentation. You read about a feature and see the examples for something like partition elimination. The documentation will show an example where the feature works, but isn't expanded to show where the feature doesn't work. Also, a critical part of database work—particularly in the design phase—is the proof of concept, asking, "Here's what we want to do, and what if we also do this, or this?" This type of testing and experimentation saves time and effort in the end. Another example: I was reading a question on a technical bulletin board asking, "I have been told if you have more than 15 indexes on a table, Oracle can't see the 16th; is this true?" Well, why not create a table with 16 indexes and write a query that has to use the 16th index and answer the question yourself?

If you had a magic wand and could change one thing about Oracle software, what would it be and why?

Jonathan: I'd put myself out of work. I would like the documentation to be expanded and the quality improved to be a little more realistic. But I know that's impossible because it's already 30,000 pages, and it merely scratches the surface of what the database can do.

How long did it take you to write* Practical Oracle 8i*?

Jonathan: I actually wrote it in the year coming up to 2000. I had a process of phoning about fifty people a week just to remind them I was around and to find out if they had any consulting work for me. Everyone was telling me projects were on hold until March 2000, since all the focus was on Year 2000 work. So, I thought, "What can I do until March 2000? I know; I'll write a book." So, I sat down and started writing and tried to find a publisher. It took me quite a few weeks to find a publisher, and when I did, I signed a contract that said I would deliver the book in March 2000. The next day, clients started phoning me and said the Year 2000 work wasn't going to be that terrible, and they wanted me to start on projects right away. So, all of a sudden, instead of having six months of doing nothing except writing a book, I was working flat out and writing a book evenings and weekends. So, although it took about six months of work, it was while I was also working full-time, and it was quite tough. Then, it took the publisher about six months to get it to print. So all my efforts to hit the deadline were lost.

Do you have any plans for another book?

Jonathan: Yes, I am working on another book on the fundamentals of the cost-based optimizer (CBO). However, I haven't told any publishers about it yet. I'd prefer to finish the book and then find a publisher, because they will only harass me until I finish it. I'm targeting March 2005 for a finish date, and then I'll find a publisher. Hopefully, publishing and printing will be done later in 2005.

This book is focused on describing how Oracle does the key features of the CBO. I think there is a huge gap in this understanding that needs to be filled. I really believe few people realize how straightforward the optimizer is and how few details you need to know in order to understand why the optimizer did something. I'd like to illustrate how straightforward the optimizer is for 80% of what it does. Really, 80% of what's going on is quite easy; it's just the 20%, which everyone has, that is tough. But, if you can get comfortable with the 80%, it makes it easy to deal with the 20%.

What other activities do you enjoy?

Jonathan: Well, life outside of the work I do with Oracle is spent with my family. My wife and I and our two children all enjoy reading. In fact, since we share books, we have a rule in the house: no one can start reading a book that someone else has already started to read. We also like to go to the theatre and enjoy plays by authors such as George Bernard Shaw. And since my daughter is an aspiring actress, we also go to school plays when she's appearing. ▲

For more information on seminars presented by Jonathan Lewis, or for other useful information (papers, scripts, FAQ's) on Oracle topics, visit http://www.jlcomp.demon.co.uk/.

Interview conducted by Laurie Robbins,
NoCOUG Journal ***assistant editor.***

Working with Transportable Tablespaces

By Anu Padki

Recently, I was working on a genomic database, which I had to export import almost every week. In the beginning, when the database was small, normal export import was working fine. However, at one point, when one table got 127 million rows, it became rather difficult. I had to explore the Transportable Tablespace export and found it very efficient and clean. However, I stumbled at times and finally came up with this plan.

My scenario: I wanted to transport the tablespaces dev_tab and dev_idx from the development database to the test database.

The user in the development database was dev1. The user in the test database was test1.

On both databases I logged in as user sys.

Step 1: Examining the source and destination databases.

1. Do not have the tablespace already in the destination database.
2. Drop materialized views from the source tablespace and those in other tablespaces created on objects in the source tablespace; e.g., materialized views created on the tables in the tablespace you want to transport.
3. Drop function-based indexes in the source tablespace.
4. Make sure that the source and target databases have the same character set.

Compare the output of

```
Select * from nls_database_parameters;
```

If the character set is not the same, the transportable tablespaces feature cannot be used.

Step 2: Source database export.

```
1. Execute dbms_tts.transport_set_check('dev_tab,
dev_idx', TRUE);
```

This checks if the tablespaces are self-contained. If they are not, you will see rows in

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

Sometimes, this does not capture the real picture.

I had this query returning zero rows and still had problems with actual exp/imp. It did not show the materialized views created in the user's tablespace using the tables in tablespace dev_tab.

The following works better.

Set the tablespaces in read-only mode.

```
alter tablespace dev_tab read only;
alter tablespace dev_idx read only;
```

1. Run the exp command.

```
host exp transport_tablespace=y tablespaces=dev_tab,
dev_idx
```

Run the query.

```
select obj1_owner, obj1_name, ts1_name,
reason from pluggable_set_check
```

This is a more comprehensive query that shows the objects in other tablespaces built on the objects in the tablespaces you want to transport.

All these objects have to be dropped, even if they are owned by other users and are in other tablespaces.

2. Now copy the datafiles of the tablespaces to the destination path.

   Use mv, cp, scp, or whatever works.

3. Put the tablespaces in read write mode.

```
Alter tablespace dev_tab read write;
Alter tablespace dev_idx read write;
```

Step 3: Destination database import.

1. Import at the destination database.

```
imp transport_tablespace=y
file=expdat.dmp
   datafiles=('/db/dev_tab1.dbf','/db/dev_idx1.dbf',...)
tablespaces=(dev_tab,dev_idx)
tts_owners=(dev1)
fromuser=(dev1)
touser=(test1)
```

Note the datafiles clause; you have to mention the path here if it is not the same as that in the source database.

You can use any other import parameters as well. I always find these useful: statistics = none, grants = n.

2. Once the import is over, put the tablespaces in the read write mode.

```
alter tablespace dev_tab read only;
alter tablespace dev_idx read only;
```

Even if you have imported to the same user, make sure the user has the desired quota on these tablespaces. ▲

The Hobgoblin of Little Minds

The Data Consistency and Concurrency Challenge

by Iggy Fernandez

Caveat Lector!

Oracle Corporation has not reviewed this essay for accuracy.

A foolish consistency is the hobgoblin of little minds,
Adored by little statesmen and philosophers and divines.
—Ralph Waldo Emerson

Introduction

The above quote appears in the chapter on data consistency and concurrency in older editions of the *Oracle 10g Concepts* manual. Tom Kyte, vice president of Core Technologies at Oracle, characterizes differences in approaches to data consistency and concurrency as *the* fundamental difference between Oracle and the other database vendors, saying that it can be Oracle's best feature or its worst feature (*if* you don't understand it) ([Reference 9]). He also says that if you don't understand it, you are probably doing some transactions *wrong* in your system and that do-it-yourself referential integrity is *almost always* wrong!

Multiversioning—Just talk about it for a bit . . .

- In my opinion *the* fundamental difference between Oracle and most of the rest:
  - It can be the best feature
  - It can be the worst feature (if you don't get it)
- Non blocking reads
- Writes only block writes
- However . . . unless you understand it, you're probably doing some transactions wrong in your system! (DIY RI is almost always wrong)

ORACLE

Fig. 1: A slide from Tom Kyte's presentation at the Northern California User Group Fall 2004 conference.

The Data Consistency and Concurrency Challenge

Oracle has patented the techniques it uses for concurrency control. One of the names on the patent filing is that of Dr. Kenneth Jacobs, a.k.a. "Dr. DBA," currently the vice president of Product Strategy at Oracle. Here is a quote from the patent documents ([Reference 2]).

"To describe fully consistent transaction behavior when transactions execute concurrently, database researchers have defined a transaction isolation level called 'serializability.'

"In the serializable isolation level, transactions must execute in such a way that they appear to be executed one at a time ('serially'), rather than concurrently. […]

"In other words, concurrent transactions executing in serializable mode are only permitted to make database changes they could have made if the transactions had been scheduled to execute one after another, in some specific order,[1] rather than concurrently."

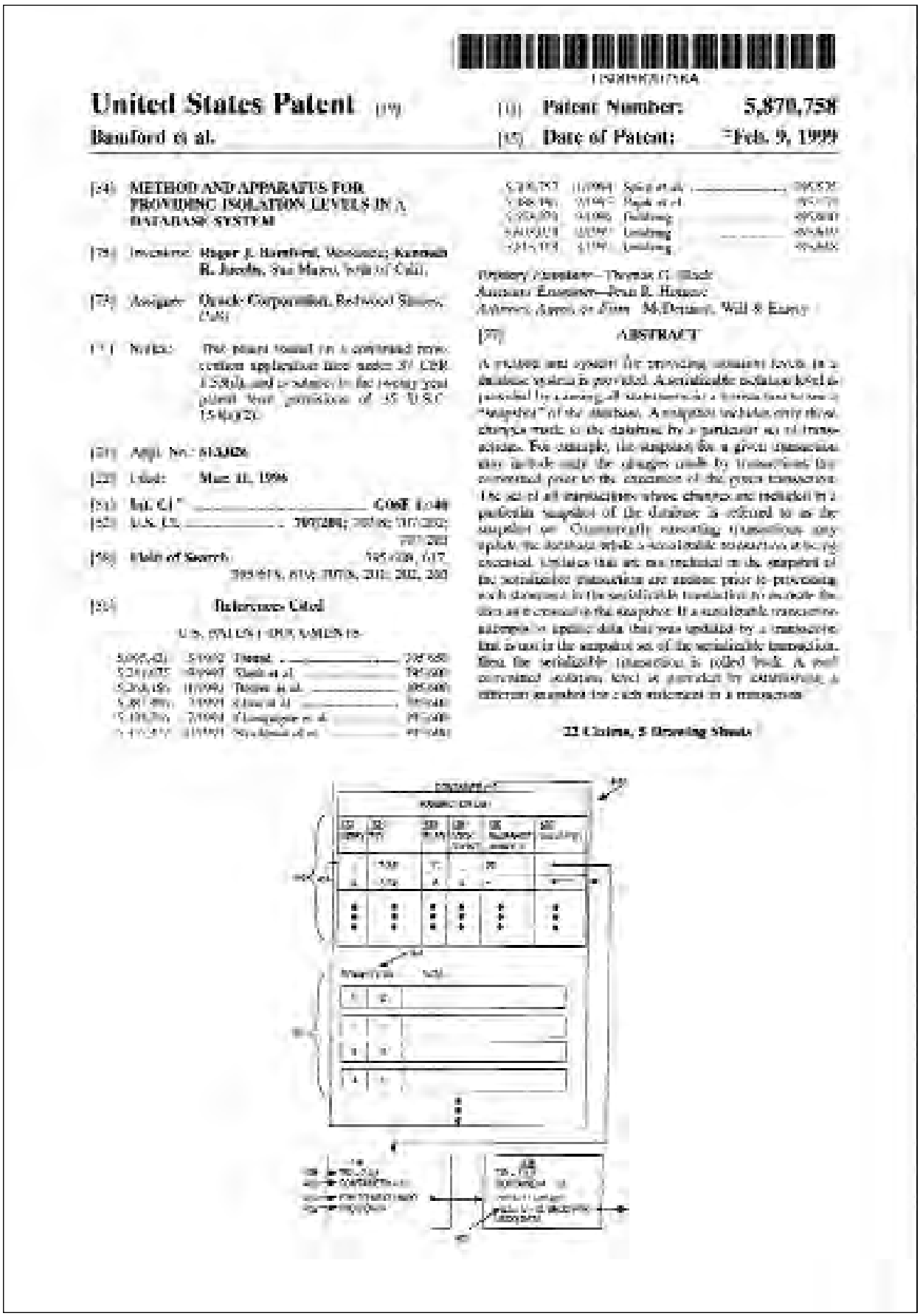
[See Fig. 2 on opposite page.]

The serializability criterion for database consistency is very well known and is even mentioned in the ANSI SQL standard [Reference 1]. Here is an excerpt.

> *"The isolation level of an SQL-transaction defines the degree to which the operations on SQL-data or schemas in that SQL-transaction are affected by the effects of and can affect operations on SQL-data or schemas in concurrent SQL-transactions. […] The execution of concurrent SQL-transactions at isolation level SERIALIZABLE is guaranteed to be serializable.*
>
> *"A serializable execution is defined to be an execution of the operations of concurrently executing SQL-transactions that produces the same effect as some serial execution of those same SQL-transactions. A serial execution is one in which each SQL-transaction executes to completion before the next SQL-transaction begins."*

However, it is not very well known that serializability is *not* the only criterion for database consistency. In other words, serializability is *sufficient* for database consistency but not *necessary*. If a transaction executes *concurrently* with another transaction, and the results are the *same* as if the transactions had executed *serially* in some order, then *obviously* the results are consistent. However, if the results are *different* from any that could be produced if the trans-

[1] Note that different serial orderings of transactions can conceivably produce different results. (For example, multiplying a number by 2 and then adding 3 will produce a different result if the operations are reversed.) Since each such result is permissible when the transactions are executed in serial fashion, they are all permissible when the transactions are executed in concurrent fashion.



United States Patent [19]

Bamford et al.

[11] Patent Number: 5,870,758

[45] Date of Patent: Feb. 9, 1999

[54] METHOD AND APPARATUS FOR PROVIDING ISOLATION LEVELS IN A DATABASE SYSTEM

[75] Inventors: Roger J. Bamford, Belmont; Kenneth R. Jacobs, San Mateo, both of Calif.

[73] Assignee: Oracle Corporation, Redwood Shores, Calif.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: 813,826

[22] Filed: Mar. 11, 1996

[51] Int. Cl.[6] G06F 17/30

[52] U.S. Cl. 707/201; 707/8; 707/202; 707/203

[58] Field of Search 395/608, 617, 395/614, 615; 707/8, 201, 202, 203

[56] References Cited

U.S. PATENT DOCUMENTS

5,095,421 3/1992 Freund 707/8
5,261,069 11/1993 Wilkinson et al. 395/600
5,263,156 11/1993 Bowen et al. 395/600
5,280,612 1/1994 Lorie et al. 395/600
5,333,265 7/1994 Orimo et al. 395/600
5,355,477 10/1994 Strickland et al. 395/600
5,369,757 11/1994 Spiro et al. 395/600
5,388,196 2/1995 Pajak et al. 395/12
5,404,507 4/1995 Lomet 395/600
5,440,732 8/1995 Lomet et al. 395/600
5,504,899 4/1996 Raz 395/600
5,530,939 6/1996 Mansfield et al. 395/800
5,581,753 12/1996 Terry et al. 395/600

Primary Examiner—Thomas G. Black
Assistant Examiner—Jean R. Homere
Attorney, Agent, or Firm—McDermott, Will & Emery

[57] ABSTRACT

A method and system for providing isolation levels in a database system is provided. Each isolation level is provided by creating all the necessary data for one or more "snapshots" of the database. A snapshot includes only those changes made to the database by a particular set of transactions. For example, the snapshot for a given transaction may include only the changes made by transactions that committed prior to the execution of the given transaction. The set of all transactions whose changes are included in a particular snapshot of the database is referred to as the snapshot set. Optionally, executing transactions may update the database while a serializable transaction is being executed. Unless this set was recorded in the snapshot of the serializable transaction are undone prior to processing such transaction in the serializable transaction to include the data as it existed in the snapshot. If a serializable transaction attempts to update data that was updated by a transaction that is not in the snapshot set of the serializable transaction, then the serializable transaction is rolled back. A read committed isolation level is provided by establishing a different snapshot for each statement in a transaction.

22 Claims, 5 Drawing Sheets

Fig. 2: The first page of Oracle's patent filing for "a method and system for providing isolation levels in a database system."

actions had executed serially in some order, then it is still *possible*, however *unlikely* that may be, that the results are *also* consistent.

The serializability criterion for database consistency was introduced a very long time ago, in a paper by IBM researchers ([Reference 5]), and therefore most research on concurrency has focused on this criterion for database consistency. Only very recently, a Microsoft researcher ([Reference 10]) proposed another condition for database consistency called "semantic correctness," which is *less restrictive* than serializability. The following example appears in [Reference 10].

> *"For example, a stock trading application might have a buy transaction type that takes as parameters the identity of a stock and the number of shares, n, to be purchased and a result that states 'when each share was purchased no cheaper unbought shares of the stock existed in the database.'*
>
> *"In a semantically correct schedule, two concurrent transactions, T1 and T2, could each buy some shares at $30 and some at $31 per share, even though initially there are n shares available at $30.*
>
> *"First, T1 buys n/2 shares at $30; then, T2 buys n/2 shares at $30; then, since there are no more shares available at $30, T1 buys n/2 shares at $31; and, finally, T2 buys n/2 shares at $31.*
>
> *"When each transaction terminates, its result is true since, when each share was bought, no cheaper unbought shares existed in the database.*
>
> *"The final state could not have been produced by a serializable schedule since the purchase price of all shares bought by one or the other of the two transactions would have been $30."*

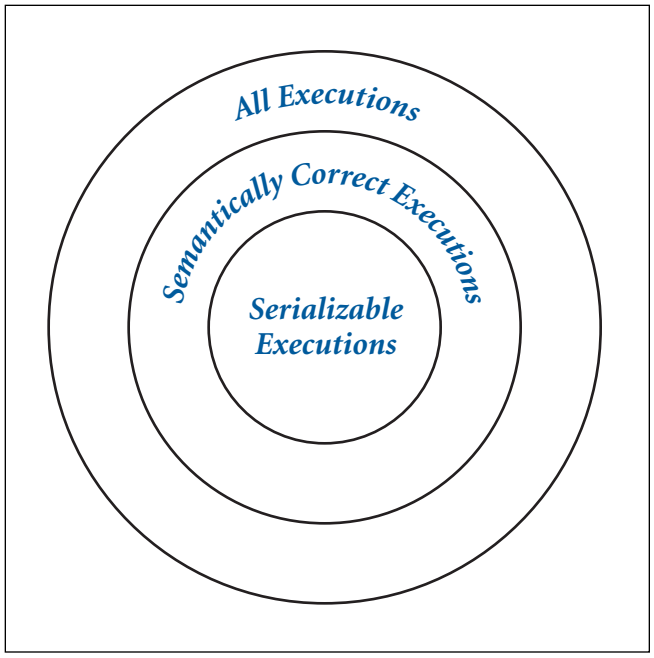


Fig. 3: The relationship between different classes of consistent executions.

Houston, We Have a Problem!

The following quote is from an academic paper by researchers at the University of Massachusetts at Boston [Reference 6].

> *"All major database system products are delivered with default non-serializable isolation levels, often ones that encounter serialization anomalies more commonly than [Oracle's Snapshot Isolation], and we suspect that numerous isolation errors occur each day at many large sites because of this, leading to corrupt data sometimes noted in data warehouse applications."*

The following quote is from the chapter on data consistency and concurrency in the *Oracle 10g Concepts* manual. The language has not changed since the days of Oracle 7 and can be traced to an Oracle white paper written by Dr. Kenneth Jacobs ("Dr. DBA") in 1995 ([Reference 7]).

> *"Although Oracle serializable mode […] offers many benefits compared with read-locking implementations, it does not provide semantics identical to such systems. Application designers must take into account the fact that reads in Oracle do not block writes as they do in other systems.*
>
> ***"Transactions that check for database consistency at the application level can require coding techniques such as the use of SELECT FOR UPDATE** [editorial emphasis added]. This issue should be considered when applications using serializable mode are ported to Oracle from other environments."*

In another place in the same chapter, the caution is repeated, once again using language from the 1995 paper by Dr. Jacobs.

> *"Because Oracle does not use read locks in either read-consistent or serializable transactions, data read by one transaction can be overwritten by another. Transactions that perform database consistency checks at the application level cannot assume that the data they read will remain unchanged during the execution of the transaction even though such changes are not visible to the transaction.*
>
> ***"Database inconsistencies can result** [editorial emphasis added] unless such application-level consistency checks are coded with this in mind, **even when using serializable transactions** [editorial emphasis added]."*

The next quote is from [Reference 6].

> *"The classical justification for lower isolation levels is that applications can be run under such levels to improve efficiency **when they can be shown not to result in serious errors** [editorial emphasis added], but little or no guidance has been offered to application programmers and DBAs by vendors as to how to avoid such errors."*

Part of the problem lies in the fact that the necessary academic research has only recently been completed. Here is another quote from [Reference 6].

> *"When two official auditors for the TPC-C benchmark were asked to certify that the Oracle SERIALIZABLE isolation level acted in a serializable fashion on the TPC-C application, they did so by 'thinking hard about it' […] It is noteworthy that there was no theoretical means to certify such a fact …"*

To summarize, application developers must take into account that the default Oracle isolation level does *not* guarantee consistent results and that *program modifications* may be necessary to guarantee consistent results (even when using stricter isolation levels). This is a good time to repeat Tom Kyte's words of warning in [Reference 9].

> *"Unless you understand it, you're probably doing some transactions wrong in your system! ([do-it-yourself referential integrity] is almost always wrong)."*

Isolation Levels . . . and All That Jazz!

Concurrency control duties put a heavy burden on any DBMS. For example, if a write transaction modifies a data item, it is advisable that other transactions not be allowed to read the modified value until the write transaction commits. "Pessimistic" concurrency control schemes such as those used by Microsoft and IBM (but not Oracle) achieve this by forcing read transactions to acquire "read locks" on the data items they want to read.[2]

A read transaction will not be able to acquire the read lock it desires, if a write transaction has modified the data item in question (thus obtaining an "exclusive lock" on that item), and will be blocked until such time as the write transaction commits or rolls back. The DBMS will enforce this behavior *even if* all transactions are simply reading data and *no* transaction is modifying data (as in the case of a data warehouse), because it has no way of knowing how long this behavior will last.

The "READ UNCOMMITTED" isolation level[3] provides application developers with the ability to signal to the DBMS that read locks are not necessary. The DBMS then no longer has to expend effort in acquiring and maintaining read locks, and performance is thereby improved.

The above reasoning is in line with the previous quote from [Reference 6].

> *"The classical justification for lower isolation levels is that applications can be run under such levels to im-*

[2] Microsoft SQL Server 2005 will partially follow Oracle's lead and provide a nonlocking concurrency scheme similar to Oracle's transaction-level read consistency scheme. However, it will be limited to read-only transactions.

[3] The READ UNCOMMITTED isolation level is not supported by Oracle.

prove efficiency ***when they can be shown not to result in serious errors*** *[editorial emphasis added] …"*

Oracle offers *three* isolation levels, one of which is *not* documented in the Oracle 10g manuals. The default isolation level (activated using the transaction setting "isolation_level=read_committed") provides statement-level consistency.

A second, stricter, isolation level (activated using the transaction setting "isolation_level=serializable") provides transaction-wide consistency and is referred to as "snapshot isolation with the first-updater-wins rule" in the academic literature ([Reference 6]).

A third, very strict isolation level, activated using the database setting "serializable=true" (Oracle 9i and prior versions) or "_serializable=true" (Oracle 10g), guarantees serializability, but only at the expense of *table-level* read locks on *all* tables accessed by the transaction.

Fig. 4 illustrates the relationships between the isolation levels mentioned above. The shaded regions require special attention from the application developer and are the subject of the cautionary remarks by Tom Kyte quoted in the introduction to this essay.

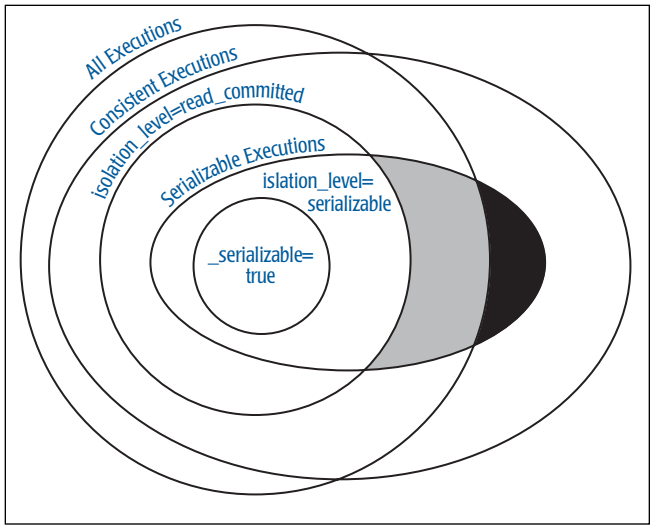


Fig. 4: The relationship between the different isolation levels provided by Oracle.

Statement-Level Consistency

This is the *default* isolation level provided by Oracle. Every SQL statement operates on a database snapshot containing *only* data values that were committed *before* the statement began. Every new statement within the same transaction operates on a *different* snapshot.

Readers do *not* acquire "read locks" on rows satisfying their selection criteria and, therefore, do *not* block writers. Writers acquire exclusive locks on rows that they modify and, therefore, they block other writers, but they do *not* block other readers.

If a statement retrieves a data block and finds that it has been modified since the statement began, it searches the "rollback segments" for the prior version of the block. If the prior version has aged out of the rollback segments, the statement fails with the well-known ORA-1555 error: "Snapshot too old."

This isolation level can cause inconsistent results if used in inappropriate circumstances. For example, it does not prevent the "Lost Update" problem described in [Reference 3] as follows.

"Transaction A retrieves some tuple p at time t1; transaction B retrieves that same tuple p at time t2; transaction A updates [and commits] the tuple (on the basis of the values seen at time t1) at time t3; and transaction B updates [and commits] the same tuple (on the basis of the values seen at time t2, which are the same as those seen at time t1) at time t4.

"Transaction A's update is lost at time t4, because transaction B overwrites it without even looking at it."

Here is a PL/SQL procedure that you can use to simulate the problem. It withdraws the indicated amount from one account and deposits it into a second account. After reading the current balances in each account, it purposely sleeps for 60 seconds to allow you to start another transaction from another database session, this time to transfer money from a third account to the second account (to which you are simultaneously attempting to transfer money in the first session).

You will find that money is successfully subtracted from the first account *and* the third account, but the second account does *not* receive both amounts!

Create a table and populate it as described below. At the start of the test, each account contains exactly ten dollars.

```
create table bank_account (
 account# integer,
 balance number
);
_
insert into bank_account values (1,10);
insert into bank_account values (2,10);
insert into bank_account values (3,10);
```

Here is the PL/SQL program needed for the test. Note that it uses the "sleep" procedure, which is part of the "user_lock" package. To create this package and give execute permissions to public, you will need to log in as SYS and run the "userlock.sql" script in the $ORACLE_HOME/rdbms/admin directory. Also note that the parameter to the sleep procedure is expressed in hundredths of seconds. User_lock.sleep(6000) therefore suspends program execution for 60 seconds.

```
create or replace procedure debit_credit(
  debit_account in integer,
  credit_account in integer,
  debit_amount in integer
)
is
  debit_account_balance number;
  credit_account_balance number;
begin
  select balance
  into debit_account_balance
  from bank_account

  where account#=debit_account;
  –
  select balance
  into credit_account_balance
  from bank_account
  where account#=credit_account;
  –
  debit_account_balance :=
  debit_account_balance - debit_amount;
  –
  credit_account_balance :=
  credit_account_balance + debit_amount;
  –
  user_lock.sleep(6000);
  –
  update bank_account
  set balance = debit_account_balance
  where account# = debit_account;
  –
  update bank_account
  set balance = credit_account_balance
  where account# = credit_account;
  –
  commit;
end;
```

Execute the following command to transfer five dollars from the first account to the second account.

```
execute debit_credit(1,2,5);
```

Before the first command has been completed, switch to another database session and execute the following command to transfer five dollars from the third account to the second account.

```
execute debit_credit(3,2,5);
```

You will find that both statements complete successfully; however, the balance in the second account is only *fifteen* dollars (instead of twenty dollars), even though the balance in the other two accounts has dropped from ten dollars to five dollars. Five dollars has done a vanishing trick!

One might argue that this is an artificial timing-dependent example and that such errors are extremely unlikely to occur in the "real world." This is certainly a persuasive argument, but some organizations may be unwilling to take any chances. Fortunately, transaction-level consistency (discussed in the next section) successfully avoids such errors (and several others) without significant performance penalties.

Transaction-Level Consistency

This *nondefault* isolation level avoids most errors that can occur at the default isolation level. It is referred to as "snapshot isolation with the first-updater-wins rule" in the academic literature. Every SQL statement operates on a snapshot of the database containing only data values that were committed before the *transaction* began. *Every* statement within the *same* transaction operates on the *same* snapshot.

The other significant difference between this non-default isolation level and the default isolation level is that Oracle will *abort* a transaction that attempts to modify a data item that was modified *after* the transaction began.[4] This is called the "first-updater-wins" rule. If you use this isolation level to run the test described in the previous section, the second transaction will abort with the following error.

```
ORA-08177: can't serialize access for this transaction
```

Write Skew

While transaction-level consistency does a good job at avoiding a plethora of errors, including "Lost Updates" ([Reference 3]) as well as "Dirty Reads," "Non-repeatable Reads," and "Phantoms" ([Reference 12]), it is subject to a class of error referred to as "Write Skew" ([Reference 6]).

[4] Oracle will also abort a transaction if it cannot verify that the data item was not modified after the transaction began. The details can be found in the *Oracle 10g Concepts* manual ([Reference 11]).

Here are three examples of "Write Skew." The first example is taken verbatim from the chapter on data consistency and concurrency in the *Oracle 10g Concepts* manual ([Reference 11]).

"One transaction checks that a row with a specific primary key value exists in the parent table before inserting corresponding child rows. The other transaction checks to see that no corresponding detail rows exist before deleting a parent row.

"In this case, both transactions assume (but do not ensure) that data they read will not change before the transaction completes. The read issued by transaction A does not prevent transaction B from deleting the parent row, and transaction B's query for child rows does not prevent transaction A from inserting child rows.

"This scenario leaves a child row in the database with no corresponding parent row."

The second example is from [Reference 6].

"Suppose X and Y are data items representing bank balances for a married couple, with the constraint that X+Y > 0 (the bank permits either account to overdraw as long as the sum of the account balances remains positive). Assume that initially X = 70 and Y = 80.

"Transaction T1 reads X and Y, then subtracts 100 from X, assuming it is safe because the two data items added up to 150. Transaction T2 concurrently reads X and Y, then subtracts 100 from Y, assuming it is safe for the same reason."

TECH TIPS

Unbreak Broken DBA Jobs

Sometimes DBA jobs are broken due to an instance being down for maintenance or backup. It is easy to find out which jobs are broken in the database and "unbreak" them. Just follow the simple steps listed below:

—login in as sysdba
—Execute the following sql:

```
spool unbreak_dba_jobs.sql
select 'exec sys.dbms_ijob.broken('||job||', false);'
from dba_jobs where BROKEN='Y'
spool off
```

—Execute the unbreak_dba_jobs.sql. It will execute command sys.dbms_ijob.broken(job, false) and unbreak this job.
—Select the following query to list any broken jobs.

Capture SQL and Performance Issues with 10046 Event

—Log in as sysdba or any user with dba privilege.
—Run the following command to turn the trace on for your user session.

```
ALTER SESSION SET EVENTS='10046 TRACE NAME CONTEXT FOREVER, LEVEL 12';
```

—Run the following command to turn the trace off for your user session.

```
ALTERSESSION SETEVENTS '100046TRACENAMECONTEXT OFF';
```

—Run the following command to turn the trace on for the entire system.

```
ALTER SYSTEM SET EVENTS='10046 TRACE NAME CONTEXT FOREVER, LEVEL 12';
```

—Run the following command to turn the trace off for the entire system.

```
ALTERSYSTEM SETEVENTS '100046TRACENAMECONTEXT OFF';
```

Note that there are different levels of trace, depending on the issue that you are trying to capture.

Go to the udump directory to fetch the trace file after turning the trace off. ▲

—Submitted by Jen Hong

The final example is paraphrased from Chapter 3 ("Locking and Concurrency") in Tom Kyte's best-selling book, *Expert One-On-One Oracle* ([Reference 8]).

> *"Two tables, A and B, initially contain no rows. Session 1 uses transaction-level consistency and executes the command 'insert into A select count(*) from B.' Session 2, also using transaction-level consistency, executes the command 'insert into B select count(*) from A.' Both sessions then commit successfully.*
>
> *"Both table A and table B now contain a single data item with value 0. It is easy to see that this cannot happen if the sessions were executed serially in some order."*

Ensuring Serializability of Transaction-Level Consistency

While transaction-level consistency does not *always guarantee* consistent results, it *is* possible for a set of transactions using transaction-level consistency to operate "with serializable effect." For example, [Reference 6] rigorously proves that the transactions constituting the TPC-C benchmark ([Reference 12]) *always* operate with serializable effect when using transaction-level consistency.

[Reference 6] explains how to determine if the transactions constituting an arbitrary application always operate with "serializable effect" when using transaction-level consistency. However, automated tools are not yet available for the purpose, and therefore this sort of analysis may not be feasible in a system containing thousands of different transaction types.

There are two ways to *force* serializable results when using transaction-level consistency. The first is to force Oracle to acquire *table-level* read locks on *every* table that is read or modified during a transaction. This is achieved using the database initialization parameter "serializable=true" (Oracle 9i and prior versions) or the "hidden" parameter "_serializable=true" (Oracle 10g). Readers are invited to use this setting to retest the examples of "Write Skew" listed in the previous section and convince themselves that the potential for inconsistent results is eliminated by this draconian measure.

Table-level read locks will play havoc with concurrency and are not likely to be an acceptable solution for very many organizations.

Table-level read locks will play havoc with concurrency and are not likely to be an acceptable solution for very many organizations. Fortunately, the "first-updater-wins" rule can be leveraged to create a "sufficient condition" that guarantees serializability when using transaction-level consistency.

The rule states that serializability is guaranteed under transaction-level consistency if, for *every* pair of write transactions defined in an application, *exactly one* of the following rules holds.

1. The transactions *demonstrably* operate on separate sets of tables or on separate regions of a set of tables. For example, an application might enforce a rule that a write transaction may read and modify the data of *only one* department of the organization. In such a case, if any two write transactions are reading or modifying the data of different departments, then they are *demonstrably* operating on separate regions of a set of tables or on completely separate sets of tables. Note that Oracle *cannot* enforce such a rule. It *has* to be enforced by the application itself.
2. Both transactions update *at least* one common record. If a suitable record does not exist, then an artificial record can be created. ([Reference 6] refers to this strategy as "materializing the conflict.") If the transactions happen to run concurrently, then the "first-updater-wins" rule will prevent both transactions from succeeding (which is the needed behavior).

The above rule is a slightly more restrictive form of a rule presented in an academic paper published a few months ago ([Reference 4]). It is "sufficient" for serializability (when *all* transactions in the application use transaction-level consistency), but not necessary. In other words, it may be *overly* restrictive in some cases. For example, the "Write Skew" problem discussed in [Reference 11] may be more simply circumvented by using "SELECT FOR UPDATE" when performing referential integrity checks or by directly embedding these referential integrity checks into the DBMS using foreign key constraints.[5]

Summary

It is important to understand each isolation level and choose one that maximizes concurrency but avoids inconsistent results. In some cases, program modifications are necessary to avoid inconsistent results. ▲

Iggy Fernandez is the lead DBA for a Silicon Valley startup and is Oracle 10g certified. Previously, he was the manager of database administration for Corio, an application services provider (ASP), and was responsible for a mixed portfolio of nearly one thousand Oracle and SQL Server databases. He is interested in best practices for Oracle database administration and is writing a book called A Structured Approach to Oracle Database Administration using Oracle 10g, *which seeks to apply IT service management (ITSM) techniques to Oracle database administration. You can contact him at iggy_fernandez@hotmail.com.*

[5] Oracle uses "SELECT FOR UPDATE" when checking foreign key constraints.

Acknowledgments

I am grateful to Venkat Devraj, CEO of ExtraQuest Corporation and the author of *Oracle 24x7 Tips and Techniques*, and to Ravi Kulkarni, senior database administrator at Corio, for carefully reading this essay and providing helpful comments.

References

[1] ANSI. X3.135-1992, *Database Language SQL*, 1993. Available at http://www.cs.pdx.edu/~len/587/sql-92.pdf.

[2] R. Bamford, and K. Jacobs. *Method and Apparatus for Providing Isolation Levels in a Database System.* United States Patent No. 5,870,758, 1996. Available at http://www.uspto.gov/, on payment of a $3 download fee.

[3] C. Date. *An Introduction to Database Systems*, Sixth Edition. Addison Wesley, 1994, Chapter 14.

[4] S. Elnikety, F. Pedone, and W. Zwaenepoel. *Generalized Snapshot Isolation and a Prefix-Consistent Implementation.* 2004. Available at http://icwww.epfl.ch/publications/documents/IC_TECH_REPORT_200421.pdf.

[5] K. Eswaran, J. Gray, R. Lorie, and I. Traiger. *The Notions of Consistency and Predicate Locks in a Database System.* 1976. Available at http://www.cc.gatech.edu/classes/AY2005/cs4803enc_fall/papers/NotionsOfConsistency.pdf.

[6] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha. *Making Snapshot Isolation Serializable.* 1996. Available at http://www.cs.umb.edu/~isotest/snaptest/snaptest.pdf.

[7] K. Jacobs, R. Bamford, G. Doherty, K. Haas, M. Holt, F. Putzolu, and B. Quigley. *Concurrency Control: Transaction Isolation and Serializability in SQL92 and Oracle 7.* Oracle White Paper, Part No. A33745, 1995. Available on request from Oracle Support.

[8] T. Kyte. *Expert One-On-One Oracle.* Wrox Press, 2001, Chapter 3.

[9] T. Kyte. *Inside Multiversioning.* Slide presentation at the Northern California User Group Fall 2004 Conference, 2004. Available at http://www.nocoug.org/download/2004-08/RWCons.ppt.

[10] S. Lu, A. Bernstein, and P. Lewis. *Correct Execution of Transactions at Different Isolation Levels.* 2004. Available at http://www.cs.wayne.edu/~shiyong/papers/ tkde04.pdf.

[11] Oracle. *Concepts. 2004*, Chapter 13. Available at http://www.oracle.com/technology/software/index.html.

[12] TPC. *TPC-C Benchmark Specification.* Available at http://www.tpc.org/tpcc/.

Mark your calendar for our Spring Conference:

Thursday, May 19th

at Lockheed-Martin in Sunnyvale, CA.

Executing External Programs from Within Oracle

by James F. Koopmann

James F. Koopmann

Finally, Oracle 10g has given us the ability to execute external programs from within Oracle. Let's take a look at what we need to do to expose this new feature.

DBMS_JOB

It must first be stated, or we might get confused, as I did, that the DBMS_JOB package that was widely used in earlier versions of Oracle is being left behind in Oracle 10g for the greater capabilities of the DBMS_SCHEDULER package. This new package allows us to manage jobs with greater versatility.

DBMS_SCHEDULER

This new job scheduler allows the database to handle a much fuller set of scheduling and monitoring capabilities. The vast amount of capabilities of the scheduler is beyond the scope of this article, but I would encourage you to explore the manuals and determine which of the features will fit in your environment. I am confident you will switch from the old DBMS_JOBS to the new DBMS_SCHEDULER. If there were one feature that would get you to switch to DBMS_SCHEDULER, I think it would be its extended ability to handle a multitude of recurring time intervals. Also, with this new scheduler we have the ability to control the execution of internal database applications as well as external applications. The new DBMS_SCHEDULER has three distinct parts.

1. Schedule, which is the definition of the dates, times, and recurring events that should happen. *Note: We will not be creating any schedules in this article.*
2. Program, which defines the task or collection of tasks a schedule or job will run.
3. Job, which is the definition of when a program will run.

Create a Program

As stated above, the program is a definition of what external program or application we will run. Listing 1 and Table 1 give you examples of how to define an external program to execute and what the parameters are used for. Listing 2 also gives you the external shell contents that are being executed from this scheduled program. Within this shell you could execute anything that you would normally execute from within a shell script.

Listing 1.
Create a Program to Execute an External Application

```
BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM (
program_name          => 'VMSTAT_PGM',
program_type       => 'EXECUTABLE',
program_action  => '/home/oracle/vmstat.sh',
enabled          => TRUE,
comments            => 'generate vmstat output'
);
END;
/
```

Table 1.
CREATE_PROGRAM Parameters

Parameter	Description
program_name	This parameter allows you to assign a unique name to the program.
Program_type	This parameter indicates the type of program that will be run. This type can take three values : plsql_block, stored_procedure, or, for our example, executable.
Program_action	This parameter defines what will be run.
enabled	This parameter is a flag that determines if the program should be enabled when created.
comments	Provide any comments to the schedule here.

Listing 2.
The External Program

```
[oracle@bier oracle]$ cat vmstat.sh
/usr/bin/vmstat >> /tmp/vmstat.LST
```

Create a Job

Now that we have defined a program to execute, we must tell the database when to execute it. This is done by the creation of a job. Listing 3 and Table 3 give you an example of how to create the job and the parameters associated with the create statement.

Listing 3.
Creating a Job to Execute

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
   job_name        =>  'VMSTAT_JOB',
   program_name          =>  'VMSTAT_PGM',
   repeat_interval   =>
'FREQ=SECONDLY;INTERVAL=5',
   enabled         =>  TRUE,
   comments          =>  'Every 5 seconds');
END;
/
```

Table 1.
CREATE_JOB Parameters

Parameter	Description
job_name	This parameter allows you to assign a unique name to the program you are creating.
program_name	This parameter allows you to associate a program name with the job you are creating.
repeat_interval	This parameter defines the recurring time interval for this job.
enabled	This parameter is a flag that determines if the job should be enabled when created.
comments	Add any comments to what you are doing here.

How to View Scheduler Information

To take a look at the jobs that have run and their success or failure, you can run the following SQL, found in Listing 4. If you are concerned with just the status of a job and if it is scheduled to run or not, you can issue the SQL in Listing 5.

Listing 4.
Show Status of Previously Run Jobs

```
SQL>SELECT JOB_NAME, STATUS, ERROR#
      FROM DBA_SCHEDULER_JOB_RUN_DETAILS
      WHERE JOB_NAME = 'VMSTAT_JOB';

JOB_NAME        STATUS           ERROR#
──────────      ──────────       ──────
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
VMSTAT_JOB      SUCCEEDED        0
```

Listing 5.
Status of Scheduled Jobs

```
SQL> SELECT JOB_NAME, STATE FROM
DBA_SCHEDULER_JOBS
    WHERE JOB_NAME = 'VMSTAT_JOB';

JOB_NAME        STATE
──────────      ─────────
VMSTAT_JOB      SCHEDULED
```

Looky, Mom, I can delete from a DBA view.

```
SQL> DELETE FROM dba_scheduler_job_run_details;
SQL> COMMIT;
```

Output from Our Example

Just to see that we actually generated some output from our external job call from within Oracle, take a look at Listing 6.

Listing 6.

```
[oracle@bier oracle]$ cat /tmp/vmstat.LST
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  97268  187756  588084  0    0    87   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  97268  187792  588048  0    0    87   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  97140  187840  588000  0    0    87   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  1      0  98612  187864  587976  0    0    87   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  98612  187920  587920  0    0    86   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  98612  187968  587872  0    0    86   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  0      0  98548  188016  587824  0    0    86   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
1  0      0  98548  188056  588044  0    0    86   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  1      0  98612  188096  588004  0    0    86   102  533  98  6   1   89  4
procs-----------memory-------swap-------io--------system--------cpu----
r  b   swpd   free   buff   cache   si   so   bi   bo   in   cs  us  sy  id  wa
0  1      0  97012  188124  587976  0    0    86   102  533  98  6   1   89  4
```

Droping the Program and Job

If you should ever want to drop the newly created program and job, you can use the following DBMS_SCHEDULER drop procedures.

```
BEGIN
DBMS_SCHEDULER.DROP_PROGRAM ('vmstat_pgm');
END;
/

BEGIN
DBMS_SCHEDULER.DROP_JOB ('vmstat_job');
END;
/
```

The ability for us as DBAs to extend internal database scheduling to call external applications is invaluable. Personally, I no longer need to rely upon cron job entries and their limited ability to schedule my external procedures and DBA tasks. Now *all* scheduled database tasks can be scheduled within my database, where I have control. This is a great day. ▲

James F. Koopmann is dedicated to providing technical advantage and guidance to companies in information technology. Over the years, James has worked with a variety of database-centric software and tools vendors as strategist, architect, DBA, and performance expert. He is an accomplished author, appearing regularly in printed publications across the Web, and speaking at local area user groups as well as industry conferences. He may be reached at jkoopmann@pinehorse.com or www.pinehorse.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, Darrin Swan, at darrin.swan@quest.com. ▲

Long-term event sponsorship:

LOCKHEED MARTIN

CHEVRONTEXACO

ORACLE CORP.

Thank you! Year 2005 Gold Vendors:

- Churchill Software
- Confio Software
- Database Specialists, Inc.
- DataMirror
- Embarcadero Technologies
- EMC Corporation
- Quest Software
- Quovera
- Verican
- Veritas

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

TREASURER'S REPORT

Judy Lyman, *Treasurer*

Beginning Balance		
October 1, 2004		**$ 60,967.09**
Revenue		
Membership Dues	2,170.00	
Meeting Fees	1,066.00	
Vendor Receipts	9,000.00	
Training Day	---	
Advertising	---	
Interest	28.95	
Miscellaneous	2,537.93	
Total Revenue		**$ 14,802.88**
Expenses		
Regional Meeting	15,141.59	
Journal	3,782.47	
Membership	20.00	
Administration	1,211.24	
Website	---	
Board Meeting	722.02	
Training Day	---	
Marketing	225.00	
Paypal	74.34	
Miscellaneous	---	
IRS	85.00	
FTB	---	
Insurance	506.00	
Total Expenses		**$ 21,767.66**
Ending Balance		
November 30, 2004		**$ 54,002.31**

Bitmap Indexes

By Scott Martin

Queries that test for the equality (or non-equality) of a particular "low cardinality" field to a value benefit dramatically from bitmap indexes. Consider

```
select name, ssn from patients where state = 'OH';
```

A bitmap index on "state" is substantially smaller than a B-tree index on "state." In a B-tree index, Oracle stores the key value and the rowid containing the key value for every row in the base table. In a bitmap index, Oracle stores the key value once, the lowest rowid containing the key value, the highest rowid containing the key value, and a highly efficient representation of all the rowids between the first rowid and the last rowid for the given key value.

The Paper Route

Oracle engineers had to choose an internal representation for a bitmap index. To better illustrate the problem they faced, I would like to introduce an analogy—a paper route. On this route a house either receives a daily paper or not. There are at least three ways to represent this route—a simple ordered list of the addresses of the houses that receive the paper, the address of the first house to receive the paper plus a list of deltas on how to get to the next house, and finally the address of the first house followed by a string of ones and zeros, indicating which house, starting from the first, receives the paper.

The best approach depends upon several factors; the most important two are the size of an address of a house and the percentage of houses receiving the paper. The first approach, the ordered list, is essentially how a B-tree index solves the problem. All the addresses (a.k.a. rowids) that receive the paper would be listed, in order, in the leaf nodes of the index under the key value "YES." The choice between the second approach (deltas) and the third approach (bitmaps) depends upon the frequency of houses that receive the paper. So, which approach does Oracle use for bitmap indexes? Both. As the average density of houses receiving the paper gets closer to and above the cost to store a delta, more and more houses are represented by bitmaps.

It will be shown that even at a modest cardinality (say 50, one for each state), the percentage of rows represented by a bitmap in a so-called bitmap index is quite low. I want to be clear that even with these medium cardinality columns, an Oracle bitmap index is probably a better solution than a normal B-tree index, particularly for large datasets. However, the Oracle bitmap index will contain very few bitmaps. We will be returning to this paper route analogy (to discuss "houses per block" and "residents per house"), but for now let us get on with a few concrete examples.

Let's Get Started

Let us introduce our example table of patients.

```
SVRMGR> create table patient as
  2>   select to_char(rownum-1, 'FM0000')              pid
  3>   , to_char(mod(abs(sys.dbms_random.random),   1000), 'FM000' ) || '-' ||
  4>   to_char(mod(abs(sys.dbms_random.random),     100),   'FM00' ) || '-' ||
  5>   to_char(mod(abs(sys.dbms_random.random),     10000), 'FM0000') ssn
  6>   , to_char( mod(abs(sys.dbms_random.random),  50),    'FM00')   state
  7>   , to_char( mod(abs(sys.dbms_random.random),  1000),  'FM000')  areacode
  8>   , sysdate      admit
  9>   , rpad('x', 200)  filler
  10>  from all_objects o, all_objects p
  11>  where rownum <= 10000
  12>  ;
Statement processed.
SVRMGR> create bitmap index pat_state on patient(state) storage (initial 2000K);
Statement processed.
SVRMGR> create bitmap index pat_area on patient(areacode) storage (initial 2000K);
Statement processed.
SVRMGR> select pid, ssn, state, areacode from patient where rownum <= 5;
PID   SSN          STA  AREA
----  -----------  ---  ----
0000 329-82-0340   31   642
0001 705-92-9764   35   781
0002 173-46-3092   32   903
0003 870-42-6877   03   148
0004 276-84-5897   49   953
5 rows selected.
```

Our example makes use of the random number package provided in dbmsrand.sql to create a table of 10,000 randomly generated patients. The state column, with each state averaging 200 patients, is the classic example of a good column for a bitmap index. The area code column, with each area code averaging 10 patients, is a marginal case for a bitmap index, but it is included here for comparison. Let us use the 20 patients in area code 346 as our first example of how Oracle represents the bitmap index. By using our unique meta-views on Oracle data we can see how each of the twenty rowids is represented in the bitmap.

```
SVRMGR> select to_char( cdbafile#, 'FM0000') || '.' ||
 2> to_char(cdbablock#, 'FM00000000')  || '.' ||
 3> to_char(crow, 'FM0000')       rid
 4> , decode(btyp, 0, 'DELTA', 1, 'BITMAP')  btype
 5> , rpad(raws, 16) raws
 6> from bitmap_keys
 7> where owner = 'SCOTT'
 8> and name = 'PAT_AREA'
 9> and char01 = '346'
 10>order by 1
 11> ;
RID    BTYPE RAWS
_____________ ___ ________
0001.00017140.0000 DELTA 00
0001.00017177.0006 DELTA c68d0d
0001.00017438.0000 DELTA c0cd5d
0001.00017457.0010 DELTA c2d206
0001.00017462.0012 DELTA c4cd01
0001.00017498.0000 DELTA c0de0c
0007.00005252.0012 DELTA c4f4cedda704
0007.00005256.0004 DELTA c49e01
0007.00005288.0008 DELTA c0a80b
0007.00005332.0002 DELTA c2ce0f
0007.00005338.0014 DELTA c6fc01
0007.00007212.0001 DELTA c1a2a105
0007.00007236.0013 DELTA c5b808
0007.00007261.0004 DELTA c4e408
0007.00007271.0001 BITMAP f8c5030a
0007.00007271.0003 BITMAP
0007.00007379.0014 DELTA c6d026
0007.00007410.0007 DELTA c7f80a
0007.00007438.0000 DELTA c0ef09
0007.00007463.0003 DELTA c3e508
20 rows selected.
```

Eighteen of our 20 rows are represented by delta-type entries (i.e., the next rowid was not close enough to make the use of a bitmap practical). Only the two rowids in block 7.7271 were close together enough to use a bitmap. As we crossed an extent boundary, it is worth noting the size of the field needed to accommodate the large delta. So, how large was the bitmap needed to hold these 20 rows, and how many bytes per row did it take?

```
SVRMGR> select sum(rawl) bytes
 2> , sum(rawl)/20 byteperrow
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_AREA'
 6> and char01 = '346'
 7> order by 1
 8> ;
BYTES  BYTEPERROW
_____ _____
 60     3
1 row selected.
```

Now, of course, if we took the exact same dataset and sorted it by area code, we could represent all 20 rows from area code 346 in a much smaller bitmap.

```
SVRMGR> create table patientsort as
 2> select * from patient
 3> order by areacode;
Statement processed.
SVRMGR> create bitmap index pat_areas
 2> on patientsort(areacode)
 3> storage (initial 2000K);
Statement processed.
SVRMGR> select to_char( cdbafile#, 'FM0000')
 2> || '.' ||
 3> to_char(cdbablock#, 'FM00000000')
 4> || '.' ||
 5> to_char(crow,     'FM0000') rid,
 6> decode(btyp, 0, 'DELTA',
 7> 1, 'BITMAP') btype,
 8> rpad(raws, 16) raws
 9> from bitmap_keys
 10>where owner = 'SCOTT'
 11>and name  = 'PAT_AREAS'
 12>and char01 = '346'
 13>order by 1;
RID    BTYPE         RAWS
_____  ________      ____
0007.00009883.0014 DELTA   06
0007.00009884.0000 BITMAP  f926ff7f
0007.00009884.0001 BITMAP
0007.00009884.0002 BITMAP
0007.00009884.0003 BITMAP
0007.00009884.0004 BITMAP
0007.00009884.0005 BITMAP
0007.00009884.0006 BITMAP
0007.00009884.0007 BITMAP
0007.00009884.0008 BITMAP
0007.00009884.0009 BITMAP
0007.00009884.0010 BITMAP
0007.00009884.0011 BITMAP
0007.00009884.0012 BITMAP
0007.00009884.0013 BITMAP
0007.00009884.0014 BITMAP
0007.00009885.0000 BITMAP  f8260f
0007.00009885.0001 BITMAP
0007.00009885.0002 BITMAP
0007.00009885.0003 BITMAP

20 rows selected.
SVRMGR> select sum(rawl) bytes
 2> , sum(rawl)/20 byteperrow
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_AREAS'
 6> and char01 = '346'
 7> order by 1
 8> ;
BYTES  BYTEPERROW
_____ _____
 8      .4
1 row selected.
```

Although this example illustrates that collocating columns with the same key value can save space, it is understood that this is not practical in most production situations.

The Paper Route Revisited

So, what can be done to reduce the size of a bitmap index given a fixed set of data? To answer this question we will need to better understand how Oracle actually represents individual rowid deltas on disk.

For this we will need our newspaper route example again. This time, however, we are going to introduce two new twists. First, our houses are going to turn into apartment homes, each with precisely 8 apartments. Second, our apartment homes are going to be situated on blocks (ironically enough), with a fixed number of expected homes per block.

Let us represent our customers as the tuple [block.home.apart#]. For our first example, assume we have only three customers, [1.0.2], [1.7.5], and [2.3.1]. Also assume that we have at most 16 homes per block. Oracle has chosen to represent this route by starting with the first home [1.0] and recording apartment #2. To get to the next home, Oracle simply adds the number of homes needed and records the next apartment number. So, to get from [1.0.2] to [1.7.5] simply add 7 homes and record apartment #5—[7.5]. Remembering that we are assuming 16 homes per block, how many homes do we add to get from [1.7.5] to our last customer at [2.3.1]? To get from [1.7.5] to [2.3.1] we need to add 12 homes (9 to get to [2.0.0] and another 3 to get to [2.3.0]). We also need to record apartment #1. Our complete delta from [1.7.5] to [2.3.1] is [12.1]—go 12 homes and deliver to apartment #1. If it so happens that block #1 has only 13 homes, we simply assume it had 16 and go to the third home on block #2.

This is exactly how Oracle represents deltas in bitmap indexes. The block number in the newspaper route corresponds to the disk block address of the rowid. The "home" corresponds to which group of eight rows we are identifying (e.g., first eight, second eight, and so on). The apartment number corresponds to the particular row within the group of eight. Note that Oracle too must know how many "homes" are on a block (i.e., the number of groups of eight rows that can fit on one block).

Unless you give Oracle some hints/limitations (to be discussed later), Oracle must assume the largest number of smallest rows possible per block. A row of all NULLS, the smallest row possible, takes up only 5 bytes—2 in the row directory and 3 in the row itself. So at only 5 bytes per row, Oracle assumes a large number of rows per block (368 on a 2K block).

So Where are These So-Called "Bitmaps"?

As seen before, if the next row is close enough, bitmaps can come into play. A bitmap is just a delta. But instead of an offset from the "eight," a bitmap contains a length followed by a series of bytes. Therefore, a bitmap element can be represented as [delta_eights.length.bitmap]. Each of these "mini-bitmaps" can contain rows from more than one block if some of the rows occur toward the end of one block and the beginning of another *and* Oracle has been given a hint about the number of rows per block. In fact, there is no reason why rows from several blocks could not be in one of these bitmap elements if there are fewer than 8 rows per block.

Consider a particular example from our state bitmap index on the patient table.

```
SVRMGR> select to_char( cdbafile#, 'FM0000')
|| '.' ||
 2> to_char(cdbablock#, 'FM00000000') || '.' ||
 3> to_char(crow, 'FM0000') rid
 4> , decode(btyp, 0, 'DELTA', 1, 'BITMAP') btype
 5> , rpad(raws, 16)     raws
 6> from bitmap_keys
 7> where owner = 'SCOTT'
 8> and name = 'PAT_STATE'
 9> and char01 = '15'
 10>order by 1
 11> ;
RID    BTYPE          RAWS
___    ________       ____
 <cut>
0007.00007249.0005    DELTA c515
0007.00007252.0002    BITMAP f983010441
0007.00007252.0008    BITMAP
0007.00007252.0014    BITMAP
 <cut>
```

An excerpt from the query shows one bitmap element representing 3 rows in block [0007.00007252]—rows 2, 8, and 14.

As these mini-bitmaps clearly store information more efficiently, it would be interesting to compare with the number of bytes used to store the relatively sparse area code bitmap from the more densely populated state bitmap. Each of these bitmaps represents the same 10,000 rowids.

```
SVRMGR> select sum(rawl)  bytes
 2> , sum(rawl)/10000 perrow
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_AREA'
 6> ;
BYTES    PERROW
_____    _____
 312623.1262
1 row selected.
SVRMGR> select sum(rawl)  bytes
 2> , sum(rawl)/10000 perrow
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_STATE'
 6> ;
BYTES    PERROW
_____    _____
 231212.3121
1 row selected.
```

Clearly, STATE is a better bitmap column than AREA-CODE, as it consumes about 30% less space. What percentage of rows in the state index is represented by a bitmap versus the percentage of rows in the area code index?

There are two ways of helping Oracle out in this regard, one "natural," the other a bit more draconian.

```
SVRMGR> select decode(btyp, 0, 'DELTA', 1,
'BITMAP') btype
 2> , count(*)
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_AREA'
 6> and indexlevel = 0
 7> group by btyp
 8> ;
BTYPE COUNT(*)
----- --------
DELTA 9862

BITMAP 138
2 rows selected.
SVRMGR> select decode(btyp, 0, 'DELTA', 1,
'BITMAP') btype
 2> , count(*)
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_STATE'
 6> and indexlevel = 0
 7> group by btyp
 8> ;
BTYPE COUNT(*)
----- --------
DELTA 7625
BITMAP 2375
2 rows selected.
```

Only 1.38% of the rows in the area code bitmap index are represented by bitmaps, whereas 23.75% of the rows in the state bitmap index are represented by bitmaps.

Minimizing Rows per Block

Giving Oracle a better idea of the number of rows per block it can expect and/or enforce helps in two significant ways. First, when computing delta "eights" to get from one block to another, Oracle uses much smaller numbers if it knows that a block contains at most (say) 32 rows. These smaller numbers of delta eights fit into a smaller number of bits. Second, if a block contains at most 32 rows, on a mini-bitmap it may be able to represent a high row from one block and a low row from another without resorting to a delta entry.

There are two ways of helping Oracle out in this regard, one "natural," the other a bit more draconian.

The natural way is to inform Oracle which columns cannot be NULL. This is particularly true for DATES, as a non-NULL DATE occupies 8 bytes (1 byte length, 7 byte value). Making the highest possible column number non-NULL also minimizes the space-saving possibility of trailing nulls.

The more draconian way is through the use of the "ALTER TABLE <table> MINIMIZE_RECORDS_PER_BLOCK" command, which scans the blocks in the current table, finds the largest number of rows per block, and enforces that maximum in the future. This command must be executed before the creation of bitmap indexes, as the indexes themselves do not store the eights per block they are working with. One technique for setting the value is to create the table, populate it with the desired number of small dummy rows, run the "alter table minimize_records_per_block" command to record the desired result, delete all the dummy rows, and then load the good rows.

Let us take a quick look at the space impact of four separate ways of handling rows per block:

1) Do nothing.
2) Set columns to NOT NULL.
3) Load table and then minimize_records_per_block.
4) Preload table with the desired result, minimize_records_per_block, delete dummy rows, and then load the table.

And the results . . .

```
SVRMGR> select o.name
 2> , mod(t.spare1, 32768)  maxrownum
 3> , trunc( mod( t.spare1 , 32768) / 8) + 1eights
 4> , decode(trunc(t.spare1 / 32768), 1, 'YES', 'NO') limited
 5> from tab$ t, obj$ o, user$ u
 6> where t.obj# = o.obj#
 7> and o.name  LIKE ('PATIENT_%')
 8> and o.owner# = u.user#
 9> and u.name = 'SCOTT'
 10>order by 2 desc
 11>;
NAME   MAXROWNUM EIGHTS  LIM
-------- ------ ------ --
PATIENT_NOTHIN   364   46 NO
PATIENT_NOTNUL   200   26 NO
PATIENT_BEST  15 2 YES
PATIENT_MINIMI   14 2 YES
4 rows selected.
```

This is pretty much what we expected. Adding NOT NULL constraints may not have modified the semantics of the table, but it did not help much on the number of eights. Let us look at the effect each of these four approaches has on the length of the bitmap index and the percentage of rowids represented by a true bitmap (versus deltas). For each of the four methodologies, the following SQL was executed.

```
SVRMGR> select sum(raw1)  bytes
 2> , sum(raw1)/10000 perrow
 3> from bitmap_keys
 4> where owner = 'SCOTT'
 5> and name = 'PAT_S_NOTHING'
 6> ;
BYTES  PERROW
------ ------
 232002.32
1 row selected.
SVRMGR> select decode(btyp, 0, 'DELTA', 1, 'BITMAP') btype
 2> , count(*)
 3> from bitmap_keys
 4> where owner    = 'SCOTT'
 5> and name     = 'PAT_S_NOTHING'
 6> and indexlevel = 0
 7> group by btyp
 8> ;
BTYPE COUNT(*)
----- --------
DELTA 7625
BITMAP 2375
2 rows selected.
```

In the interest of saving space, the SQL from the other three methodologies has been omitted, but the results are summarized here.

Strategy	Bytes/Row	% of Rows in Bitmap
Nothing	2.32	23.75%
Not Null	2.15	23.75%
Minimize	1.47	51.66%
Best	1.47	51.75%

Reducing the number of possible rows per block has a dramatic effect on the amount of space the bitmap consumes. Notice that with the simple addition of NOT NULL we did not gain any more bitmaps (the eights gap between one block and the next is still far too large). However, Oracle was able to represent some of the deltas with fewer bits. Notice also how "Minimize" and "Best" both occupy approximately the same amount of space, but "Best" permits one more row per block.

Conclusion

Oracle bitmaps, although somewhat of a misnomer, are an extremely efficient way of indexing large volumes of data that have a relatively small number of distinct keys. They should be used only if there is near zero INSERT/UPDATE/DELETE activity, as they support very little concurrency. Informing Oracle about the maximum number of rows per block in the base table through the "alter table minimize_records_per_block" command significantly increases the space efficiency of bitmap indexes. Best results are gained by pre-populating the table with the nearest mulitple of 8 rows, executing the "alter table" command, removing the pre-populated rows, and then loading the data. ▲

About the Author

*Scott Martin is currently the president of Terlingua Software (www.tlingua.com). After graduating with a master's degree from M.I.T., he worked in the Oracle RDBMS development team on versions 6.2 and 7.0 (1988–1992). Since leaving Oracle, Scott has been the principal author of four Oracle utilities—first, "SQL*Trax—The Log Miner for Oracle"; second, a high-speed direct-path unloader for Oracle; third, a parallel direct-path replacement for Oracle Import; and his newest product, "Terlingua Block Viewer for Oracle" (used to prepare this article). Scott has been a speaker numerous times at local user groups as well as at the IOUG. He takes pride in producing "first of breed" products, which are invented by Oracle several years later. Scott can be reached at smartin@tlingua.com.*

NoCOUG Winter Conference

Tuesday, February 8, 2005

Session Descriptions

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

KEYNOTE

Oracle Application Server 10g Release 2 Overview
Thomas Kurian, Senior Vice President, Oracle Corporation

This presentation provides a detailed technical overview of the new features in Oracle Application Server and Internet Developer Suite 10g Release 2, the benefits they offer customers, and how customers can upgrade to Release 2. The session covers new features for J2EE, Web services, enterprise portals, business process automation, security and identity management, wireless and RFID, forms and reports, and business intelligence and analytics. Understand how you can deploy and manage enterprise applications using grid computing technology and learn technical details of how to best utilize Release 2 with your Oracle database. The session also provides best practices for deploying your application server on the Windows and Linux operating systems and in a Microsoft .NET environment.

TRACK 1

Logical E/R Modeling: The Definition of "Truth" for Data
Jeffrey Jacobs, Embarcadero Technologies

Logical entity/relationship models, also referred to as "conceptual" or "semantic" models, define the information requirements of the enterprise, independent of the resulting implementation. A well-defined E/R model is the key to successful development of data-oriented applications. Although most frequently associated with relational databases, the logical E/R model is equally applicable to object-oriented and XML implementation. This presentation will provide an overview of the fundamentals of E/R modeling as the definition of the information requirements of the enterprise. It will focus on the underlying concepts and notations, with a strong emphasis on the semantic content of the E/R model.

Speeding Up Queries with Semi-Joins and Anti-Joins: How Oracle Evaluates EXISTS, NOT EXISTS, IN, and NOT IN
Roger Schrag, Database Specialists Inc.

Optimizing the SQL usually gives the most significant results when DBAs are called upon to "make the system run faster." Using tools like Statspack or Enterprise Manager, it is often easy to find the slow SQL. But how do you make the queries run faster? That is the challenge! In this presentation we will discuss the semi-join and the anti-join, two powerful SQL constructs Oracle offers for use in your quest for faster queries. In particular, we will define these two terms, discuss when and why you might want to use the [NOT] EXISTS or [NOT] IN constructs, and demonstrate how you can use optimizer hints and make minor query changes in order to enable Oracle to use some very powerful and efficient access paths. For certain classes of queries, these features can dramatically reduce logical reads, physical reads, CPU time, and elapsed time. But beware! There are some pretty obscure (and not well documented) requirements that must be met in order for Oracle to deploy the semi- and anti-join access paths. If you fail to dot an I or cross a T, you could be banging your head against the wall for hours! Throughout this presentation we will look at SQL from a real project and demonstrate live the "before" and "after" versions of queries that ran orders of magnitude faster once semi-joins and anti-joins were implemented correctly. Attendees should be very familiar with execution plans and join methods in order to get the most out of this presentation.

Beginning Oracle SQL: Common Idioms
Les Kopari, Independent Consultant

There are many common constructs—frequently used forms of SQL—for the common tasks that we are faced with on a day-to-day basis. For the beginning Oracle developer or DBA, this presentation will list some of these common idioms and the questions they answer. That may help beginners get a quick start.

TRACK 2

It's Time to Do ASH
Gaja Vaidyanatha, DBPerfMan.com

Oracle 10g promises to provide many new diagnostic capabilities for the performance enthusiast. Active Session History (ASH) is one such diagnostic feature that provides insight into the behavior of active sessions that are connected to the database. By retrieving information about active sessions over a period of time, it is now possible to determine the "true nature" of what one or more given sessions in the database is doing. This presentation will

provide an overview of ASH and its architecture and setup.

Oracle 10g Backup and Recovery New Features
Daniel Liu, First American Real Estate Solutions

This presentation introduces two new features in Oracle 10g backup and recovery: Extended Flashback Functions and RMAN Enhancements. When user errors and logical corruptions occur in the 10g database, flashback functionality provides fast and flexible data recovery. The new flashback features include Flashback Database, Flashback Drop, Flashback Table, Flashback Version Query, and Flashback Transaction Query. When physical or media corruption occurs in the database, RMAN delivers an improved and simplified recovery method.

A Structured Approach to Database Administration Using the Principles of ITSM and ITIL
Iggy Fernandez, Intacct Corporation

Change the IT world! Join the ITIL revolution! No amount of technical brilliance can substitute for a disciplined and structured approach to database administration. In the language of the Capability Maturity Model (CMM), most IT organizations rely on "individual heroics." The application of ITSM and ITIL principles generates a structured approach to database administration that is repeatable, defined, managed, and optimizing. Come learn why ITSM and ITIL are being adopted by Fortune 500 companies such as Genentech and eBay. We will also discuss Oracle 10g support for ITSM processes.

TRACK 3

Essentials of Real Application Clusters
David Austin, Oracle Corporation

This presentation describes the basic architecture of a Real Application Clusters (RAC) installation. The RAC software components are placed in the context of the underlying cluster architecture and various disk storage options. The presentation also contains a brief comparison of RAC and grid architectures. The presentation explains how to use some of the key features of a RAC database, including virtual IP addresses and service assignments, plus the general benefits of a clustered database such as high availability and scalability. In addition, the presentation contains guidelines for the installation, configuration, and monitoring of RAC databases.

Minimizing Risks Through Deployment Standardization
Sudip Datta, Oracle Corporation

As the data center grows, it becomes increasingly challenging to manage. This could potentially lead to more software resources and moving parts that are difficult to manage. This also compounds additional risks of noncompliance and vulnerabilities that may lie undetected in the enterprise. This presentation discusses the implementation of a standardized approach to deployment that helps put some discipline in management and hence minimizes the risks. ▲

TECH TIPS

OraSnap: An Oracle Tool with a Twist

OraSnap, which is short for "Oracle Snapshot," is a utility that gathers performance information from an Oracle database and publishes it in easy-to-use HTML pages. Developed by Stewart McGlaughlin, the OraSnap scripts aid in tuning and optimizing your database by giving you easy access to performance statistics collected by Oracle. McGlaughlin says on his site, "These scripts are the same scripts that most of us already have in our arsenal. The 'twist' is the way the information is presented." All of the statistics are presented in a user-friendly way using HTML—which can then be viewed with a web browser. Here are just a few areas covered by the OraSnap scripts:

- Tablespaces
- Rollback
- Users
- Security
- Tables and indexes
- Session statistics
- Shared pool

OraSnap works on Oracle versions 7.3 through 9i. Check it out at http://www.oracle-books.com/orasnap/index.html.

NoCOUG Winter Conference

Tuesday, February 8, 2005 · Oracle Conference Center, Redwood Shores, CA

Please visit **www.nocoug.org** for session abstracts, for directions to the conference, and to submit your RSVP.

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

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

9:30–10:30 **Thomas Kurian, Senior Vice President, Oracle Corporation**

10:30–11:00 **Break**

11:00–12 p.m. **Parallel Sessions #1**

Track 1: *Logical E/R Modeling: The Definition of "Truth" for Data,* by Jeffrey Jacobs, Embarcadero Technologies

Track 2: *It's Time to Do ASH*, by Gaja Vaidyanatha, DBPerfMan.com

Track 3: *Essentials of Real Application Clusters*, by David Austin, Oracle Corporation

12:00–1:00 **Lunch**

1:00–2:00 **Ask Oracle—Question and Answer Session with a Panel of Engineers from Oracle Corporation**

Session descriptions appear on pages 26–27.

2:00–2:15 **Break**

2:15–3:15 **Parallel Sessions #2**

Track 1: *Speeding Up Queries with Semi-Joins and Anti-Joins: How Oracle Evaluates EXISTS, NOT EXISTS, IN, and NOT IN*, by Roger Schrag, Database Specialists Inc.

Track 2: *Oracle 10g Backup and Recovery New Features*, by Daniel Liu, First American Real Estate Solutions

Track 3: *Minimizing Risks Through Deployment Standardization*, by Sudip Datta, Oracle Corporation

3:15–3:45 **Break and Raffle**

3:45–4:45 **Parallel Sessions #3**

Track 1: *Beginning Oracle SQL: Common Idioms,* by Les Kopari, Independent Consultant

Track 2: *A Structured Approach to Database Administration Using the Principles of ITSM and ITIL,* by Iggy Fernandez, Intacct Corporation

Track 3: *Oracle 10g Data Warehouse Features,* by Peter Dalton, Oracle Corporation

5:00–?? **NoCOUG Networking and Happy Hour**

Cost: $40 admission fee for nonmembers. Members free. Includes lunch voucher.

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

(Cover photos taken by Ron Karpel [www.karpel.org] in John Muir Wilderness, CA)

NoCOUG
P.O. Box 3282
Danville, CA 94526

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

Could this be your last issue? See page 25 for details.