

Official Publication of the Northern California Oracle Users Group

NoCOUG

J O U R N A L

VOL. 19, No. 4 · NOVEMBER 2005

\$15

Modern Performance Myths

Author Craig Shallahamer sheds some light on Oracle performance analysis. See page 5.

There's No Free Lunch

Our SQL Corner columnist Iggy Fernandez reports on Oracle's concurrency management scheme and the "gotchas" you should know about. See page 8.

Book Review

Our popular book reviewer Brian Hitchcock offers up a review of Oracle Database 10g New Features. See page 20.



Much More Inside ...

Moving On . . .

This is the fourth year that I have volunteered as the editor of the *NoCOUG Journal*. It has been a very enjoyable journey—full of opportunities for creativity and meeting friends in the Oracle user community. After all this time, though, I have decided I would like to contribute in another way and move into another role on the NoCOUG Board.

The NoCOUG Board elections take place in December, so we'll find out soon who will take my place. I will look forward to helping in any way I can. I will also look forward to new challenges in another board position.

Thanks to all who assisted with producing the *NoCOUG Journal* over the past several years. Special thanks go to our (current and past) frequent contributors: Brian Hitchcock, James Koopman, Iggy Fernandez, Chris Lawson, and Laurie Robbins.

—Lisa Loper
NoCOUG Journal Editor

Table of Contents

Editor's Note	2	Advertising Rates	26
NoCOUG Board	2	NoCOUG Fall Conference Schedule.....	28
Publication and Submission Format.....	2		
President's Message.....	3	—ADVERTISERS—	
Renew NoCOUG.....	4	Database Specialists.....	6
Modern Performance Myths	5	Embarcadero Technologies.....	6
SQL Corner.....	8	Quovera.....	11
Performance Tips for Batch Jobs	16	RightOrder.....	11
Book Review	20	Quest Software	12
Sponsorship Appreciation.....	25	Data Domain.....	15
Treasurer's Report.....	25	Confio.....	15
NoCOUG Fall Conference Session Descriptions.....	26	EMC ²	19

Publication and Submission Format

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

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

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

Copyright © 2005 by the Northern California Oracle Users Group. Permission to reproduce articles from this publication, in whole or in part, is given to other computer user groups for nonprofit use, with appropriate credit to the original author and the *Northern California Oracle Users Group Journal*. All other reproduction is strictly prohibited without written permission of the editor. Two copies of each reprint should be sent to the editor.

NoCOUG BOARD

President

Darrin Swan, Quest Software
darrin.swan@quest.com

Vice President

Colette Lamm, Independent Consultant
colette_lamm@yahoo.com

Membership Director

Joel Rosingana, Independent Consultant
joelros@pacbell.net

Director of Conference Programming/Past President

Roger Schrag, Database Specialists, Inc.
rschrag@dbspecialists.com

Webmaster

Eric Hutchinson, Independent Consultant
erichutchinson@comcast.net

Journal Editor

Lisa Loper, Database Specialists, Inc.
lloper@dbspecialists.com

Vendor Coordinator

Diane Lee, Lockheed Martin
diane.c.lee@lmc.com

Secretary/IOUG Representative

Iggy Fernandez, Intacct
iggy_fernandez@hotmail.com

Director of Marketing

Jen Hong, Cisco Systems
Hong_jen@yahoo.com

Director of Public Relations

Les Kopari, Corner Pine Consulting
(650) 802-0563

Training Day Coordinator

Hamid Minoui, Database Specialists, Inc.
hminoui@pacbell.net

Treasurer/Track Leader

Randy Samberg, PalmSource
rsamberg@sbcglobal.net

Member at Large

Naren Nagtode,
Franklin Templeton Investments
nnagtod@frk.com

Other Contributors

Vilin Roufchaie

Brian Hitchcock

NoCOUG Staff

Nora Rosingana, Independent Consultant

Oracle OpenWorld Recap and Wrapping Up 2005 with NoCOUG



Darrin Swan

My anticipation and expectations for Oracle OpenWorld 2005 (OOW) were unlike any others I can remember since I began attending five years ago. I signed up just minutes after receiving my email invitation rather than waiting until the deadline. I attended some of the weekend presentations, which I have never done before. I made special efforts to take in activities throughout the following week that involved meeting with a high volume of diverse Oracle users. I also had opportunities to meet with a number of executives, vendors, fellow NoCOUG members, and others to gain the pulse of their perception of Oracle as it rolled out plans for IT world domination. On the final day, I departed with the general theme that (we) Oracle professionals now have more opportunities than ever before, but there's a lot to learn.

Beginning Saturday, September 17, Moscone Conference Center in San Francisco was swarming with Oracle professionals from around the globe. When you consider everything Oracle announced and launched thus far in 2005 prior to OOW, you can begin to understand the magnitude of potential uncertainty and the multitude of questions provoked by the Oracle customer community. The sequence of Oracle events prompting questions included:

- Acquisition of PeopleSoft/JDE
- Fusion Strategy launch
- Acquisition of Oblix
- Acquisition of Retek
- Fusion Middleware launch
- Oracle 10g R2 launch
- Acquisition announcement of Siebel

The attendees I met arrived with an agenda to seek answers to the questions: How will I be impacted (by the above)? How do I plan accordingly for 2006 and beyond? Another very common question I heard was, How is Oracle going to make all this work? With the latter question in mind, I began attending sessions to get answers and to learn how.

If you were unable to attend Oracle OpenWorld this past September, take a look at the archived OOW agenda and let us know where you would like additional education. The November 10 NoCOUG Fall Conference is scheduled, and we are beginning to plan our February conference hosted by Oracle. If there are areas of uncertainty still looming post OOW, let us know, so that we can dedicate a learning track for our February conference that addresses your questions.

Since this is your last *NoCOUG Journal* installment for 2005, I would like to extend another special thanks to Lisa Loper and Laurie Robbins for doing such an amazing job over the past year in publishing a great educational source for Bay Area Oracle professionals. Thank you again for your contribution and a job well done.

I would also like to thank the rest of the NoCOUG board for their year-long-and-beyond contributions in helping to produce and manage our quarterly conferences. Thank you Colette Lamm, Roger Schrag, Lisa Loper, Randy Samberg, Les Kopari, Joel and Nora Rosingana, Hamid Minoui, Naren Nagtode, Diane Lee, Iggy Fernandez, and Jen Hong.

See you on November 10. ▲

Don't forget to renew your NoCOUG membership for 2006!

NoCOUG annual memberships run from January 1 through December 31. As a NoCOUG member you receive:

- Admission to quarterly NoCOUG conferences, held at different locations around the San Francisco Bay Area.
- A subscription to the *NoCOUG Journal*, the quarterly newsletter of the Northern California Oracle Users Group.
- NoCOUG's annual membership roster of fellow users listed by company and hardware platform, and a telephone directory.

NoCOUG membership for individuals costs \$70 for the year. Individual membership includes admission for one person to each quarterly meeting and one subscription to the *NoCOUG Journal*. NoCOUG membership for businesses costs \$450 for the year. Corporate membership includes admission for up to six people to each quarterly meeting and up to ten subscriptions to the *NoCOUG Journal*.

With your support—not only by your memberships but by your volunteer activities as well—we will continue to be a strong and successful Oracle users group. **Please continue to watch your email in-box for a notice regarding renewing your NoCOUG membership for 2006. ▲**

nocoug.org



Welcome to the
Northern California
ORACLE
Users Group

Refer a friend or colleague to NoCOUG and be qualified to win the following prizes.



Three prizes that you can win:

First Place Prize: \$50 Amazon.com Gift Card
Second Place Prize: \$30 Best Buy Gift Card
Third Place Prize: \$20 Fry's Electronics Gift Card

What a way to tell people about your association with NoCOUG, why you are a member and why you attend the NoCOUG quarterly conferences! Where else can you get up-to-date training on various Oracle topics, in areas such as database, developer, data warehouse and J2EE all in one day. Plus a chance to network with other Oracle Professionals in the Greater Bay Area.

How to enter:

Send email to pr@nocoug.org including your referral's information:

First Name, Last Name, Email address, Profession, Date Joined NoCOUG (new membership only). We will keep track of how many people you referred who joined NoCOUG for the calendar year 2005. The winners will be announced at the Spring Conference 2006.

Please send all your questions and inquiries to pr@nocoug.org

Modern Performance Myths

by Craig A. Shallahamer

Oracle performance analysis has come a long way in the last 20 years. First there was the “just add more resources” approach and tuning the blatantly poor SQL. Then there was ratio analysis followed by wait event analysis. Finally, response time analysis (RTA), in a way, closed the loop. What is so key about RTA is that it brought together and detailed the two elements of response time, that is, service time and queue time. But over the past couple of years, I have noticed that while people talk about Oracle timing and response time analysis, for the most part, people focus only on the wait interface. And this unhealthy focus on the wait interface is where the modern performance myths lie in wait...

Myth #1

The first myth is *decreasing wait event time will always decrease Oracle response time*. In the purest sense, response time is service time plus wait time. In a more realistic sense, response time is the service time from all the various computing subsystems plus all the wait time from all the various computing subsystems. So it's really a summation of all service time and wait time. But for conversation and modeling purposes, we typically simplify the situation by saying, “Response time is service time plus wait time.”

From a DBA perspective, service time is the CPU time consumed by Oracle processes. More specifically, when we measure CPU time we are referring to server process *user mode* CPU time. Let me back up a second. When a CPU is running, it can be running your process (user mode) or operating system *stuff* (system time). For example, this operating system *stuff* could be process scheduling or virtual memory management. Your process could be the Oracle executable or the SQL*Plus executable.

Back to the first myth, decreasing wait event time will always decrease Oracle response time. The most obvious example of this mythical phenomenon is related to latching. When a process is spinning on the latch it is executing Oracle kernel code and consuming CPU time, and therefore **no** wait event is posted. But if a process trying to get a latch has been put to sleep, the wait event is posted because the process is not consuming CPU. It is, in every sense, “waiting.” Therefore, the server process is sleeping and not using any CPU resources. Is it possible for the response time to remain the same or increase while its two components are shifting? Absolutely!

Let's suppose a DBA is hyper-focused on reducing wait time. The top wait event is *latch free* (10g example: *latch free: cache buffer chains*). And let's suppose the instance parameter *spin_count* is to be changed. By decreasing the spin count, each spin cycle (when a server process repeatedly asks for the latch) will consume less CPU. But let's say the DBA increased spin count. During each spin cycle more CPU will be consumed. This will increase CPU consumption, that is, the service time.

Look what happens to the latch free wait event . . . it has decreased because less sleeping occurs. Because response time is service time plus wait time, while the wait time has decreased, it is very possible that the increased service time has matched or surpassed the decrease in wait time. Since response time is service time plus wait time, the response time may have increased! But wait, it gets better!

Since the DBA is hyper-focused on wait events, if wait time has decreased, he will think performance has improved! The insidious part to all of this is, nearly always, when there is significant latch contention there is no excess CPU. As a result, more time will be spent spinning than is gained by reduced sleeping, with a net result of increased response time! So remember, decreasing wait event time may not decrease Oracle response time.

Myth #2

The second myth is *decreasing wait event time also decreases end-to-end response time*. To get a clear picture of this myth, the computing system architecture is very important. When we measure response time, being DBAs, we typically start from the database server, move into the network, and then finally the client process. All of our

When the wait interface came along it meant great things for DBAs. Unfortunately, it took many years before it was accepted into the main-stream.

trace file and performance view timing is based upon this perspective. But this perspective is different from an “end-to-end” perspective.

End-to-end performance is what the user feels. Picture a user holding a stopwatch and calmly telling you (now that is a myth!) a specific query takes 7 seconds. That’s end-to-end response time. The problem we as DBAs run into is that measurements typically stop at the client process, not the user’s browser! And what about the time between the Oracle client process and the user’s browser? Well . . . typically we assume it is not significant. Ouch!

This was a safe bet in a classic client/server architecture, but not anymore. It is very common for DBAs to speak about measuring response time and not even think that their measurements are not true end-to-end response time measurements. The network time between the client process and the browser can be significant. If the user experience does not match what you are measuring, don’t forget about

that piece of the architecture. The myth is believing that profiling an Oracle session takes into account all the time associated with the user’s experience, that is, end-to-end response time.

Myth #3

The third myth is *profiling sessions is the best way to diagnose performance problems*. All the current hype about profiling a user’s session would leave you to believe it is the holy grail of performance diagnosis. This performance myth is very discreetly and conveniently not talked about among the most noted Oracle industry experts.

The myth believes that when a session is profiled, that is, a session level response time analysis is performed, the reported CPU time and the wait time are the sole result of the session being profiled. While this may sound true, ask yourself this: Couldn’t the session be impacted, affected, and “slow” because other crazy things are occurring on the system? Obviously!

For example, let’s talk about a DBA’s experience. We’ll call this DBA Frank. Let’s suppose Frank’s CRM session is being profiled. The response time analysis clearly shows the performance issue is a specific SQL statement that keeps requesting blocks to be read from the IO subsystem. So we should focus on Frank’s CRM SQL, right? Wrong. The hidden and very important assumption is that Frank is operating independently from all the other users and



There's No Substitute for Experience

Our team represents some of the most knowledgeable and experienced in the industry. We are authors and speakers with long careers as Oracle experts, averaging 12 years. Our specialty is providing remote DBA services and onsite Oracle database consulting.

We offer a free consultation to discuss:

- Increasing uptime and reliability
- Minimizing downtime and data loss
- Optimizing performance
- Reducing cost of database operations

Call Us Today!

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

ORACLE | CERTIFIED SOLUTION PARTNER


Database Specialists



processes on the system. And this is obviously an incorrect assumption. There could be other users populating the buffer cache with blocks Frank is not interested in. As a result, Frank's data must be brought into the cache from disk. It is a real possibility that by tuning the other SQL (i.e., not Frank's SQL), Frank's response time will dramatically improve!

The myth believes the time associated with the profiled session is the sole result of the session being profiled. A more complete performance diagnosis would be to perform both a session level and a system level response time analysis taking into consideration the end-to-end performance issues. This is why in my *Advanced Reactive Performance* class and in my tools (available free at orapub.com) I stress both session level and system level response time analysis.

Myth #4

The fourth myth is a little tricky to understand. It is *CPU time and wait time are not interdependent*. This is the foundational statement: Service time has a limit but wait time is limitless. Each CPU subsystem has a maximum fixed amount of CPU power it can provide. If you have a one CPU machine, that machine can provide a maximum of 60 seconds of CPU each minute. If you have a 10 CPU machine, it can provide a maximum of 600 seconds of CPU each minute. The situation is distinctly different with wait time.

Wait time is not fixed and is limited only by the workload (assuming the hardware configuration is constant). If the workload is relatively small, wait time may be near zero. But if you keep increasing the workload, wait time will reach infinity . . . it has no limit.

Once all the CPU time/power is consumed, Oracle processes will wait. This will result in increased wait event time and therefore increased response time (i.e., decrease in performance). However, if the system had additional CPU power, more CPU power would be consumed and there would be less wait time. A good example of this was discussed in *Myth #1* in regard to latching.

Once the relationship between service time and wait time is understood, DBAs will naturally start watching more closely how Oracle is impacting the operating system. This is good, because understanding how Oracle impacts and interacts with its host operating system is key to diagnosing and solving performance problems.

Having observed this interaction for many years now, I've found that performance diagnosis is dramatically improved by observing Oracle wait time, Oracle CPU time, and *the percentage of CPU that Oracle is consuming from the operating system*. I have found that once Oracle consumes more than around 75% of the available CPU, actual CPU utilization is at least 80% and CPU-related wait event (e.g., latching) time starts to dramatically increase . . . and performance decreases.

When the wait interface came along it meant great things for DBAs. Unfortunately, it took many years before it was accepted into the mainstream. But now the situation is beginning to swing too far toward wait event analysis without regard for *service time, response time, the system architecture, and the host operating system*. A hyper-focus on wait events can become an unhealthy diversion and undermine an otherwise solid performance analysis. When my students and consulting clients started to fixate on wait events, I had to take a step back and provide more information to help expand their perspective. What I saw happening with different situations revealed these four myths. I hope you have found these myths thought-provoking. Better yet, I hope your performance diagnosis and performance analysis will improve for the better! ▲

© 2005 Craig A. Shallahamer

About the Author

A recognized Oracle server authority and quoted as being "An Oracle performance philosopher who has a special place in history of Oracle performance management," Mr. Shallahamer brings his unique experiences to many as a keynote speaker, a sought after teacher, a researcher and publisher for improving Oracle performance management, and also as the founder of the grid computing company, BigBlueRiver.

Mr. Shallahamer spent nine years at Oracle Corporation personally impacting hundreds of consultants, companies, database administrators, performance specialists, and capacity planners throughout the world. He left Oracle in 1998 to start OraPub, Inc., a company focusing on "Doing and helping others do" both reactive and proactive Oracle performance management. He has personally trained over 1000 DBAs on five continents in thirteen countries. Mr. Shallahamer is also the key designer and engineer behind HoriZone™, OraPub's performance prediction and risk analysis product.

When Mr. Shallahamer is not working on Oracle performance management, he is fly fishing, praying, backpacking, playing guitar, or just relaxing around a fire.

Mark Your Calendar!
You won't want to miss our first conference of 2006. It will take place on Tuesday, February 14, at Oracle Corporation in Redwood Shores.



Iggy Fernandez

Concurrency—There's No Free Lunch

By Iggy Fernandez

It's the best possible time to be alive, when almost everything you thought you knew is wrong.

—The character Valentine in *Arcadia*, a play by Tom Stoppard

Unless you know how it works, you will write programs that corrupt data. It is that simple.

—Concluding remark of the chapter on “Locking and Concurrency” in Tom Kyte’s best-selling book, *Expert One-On-One Oracle*

A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines . . . Speak what you think now in hard words, and tomorrow speak what tomorrow thinks in hard words again, though it contradict every thing you said today.

—American philosopher Ralph Waldo Emerson

Summary

Oracle’s concurrency management scheme is poles apart from that used by the other popular database management engines such as DB2 UDB, MS-SQL Server, Informix and Sybase. In fact, Tom Kyte characterizes it as the fundamental difference between Oracle and other database vendors because Readers do not acquire any locks and, therefore, Readers do not block Writers, and Writers do not block Readers.

This gives Oracle a tremendous advantage in the concurrency game, but, as in most other things in life, there is no free lunch. Tom Kyte bluntly says that “unless you understand it, you’re probably doing some transactions wrong in your system!”

In this essay, we present a number of examples of how things can easily go wrong if you don’t understand Oracle’s concurrency management scheme. Readers with a theoretical bent should refer to the theoretical article on Oracle’s concurrency management scheme published in the February 2005 issue of the *NoCOUG Journal*. The article is available for downloading at http://www.nocoug.org/download/2005-08/SERIALIZABILITY_-_White_Paper.pdf.

Oracle’s concurrency management scheme is extremely attractive. One might ask why the other engines don’t simply emulate Oracle’s scheme and dispense with the need for locking and blocking. The answer is that they use a stricter interpretation of the “serializability” requirement, i.e., the requirement that concurrently executed transactions produce the same result as some serial execution of those same transactions. The confusion is best explained by the following quote taken from Reference [2], *A Critique of ANSI SQL Isolation Levels*:

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 Oracle User Group Fall 2004 conference.

ANSI SQL defines four levels of isolation [...] Each isolation level is characterized by the phenomena¹ that a transaction is forbidden to experience [...] However, the ANSI SQL specifications do not define the SERIALIZABLE isolation level solely in terms of these phenomena. Subclause 4.28, “SQL-transactions”, in [the ANSI standard] notes that the SERIALIZABLE isolation level must provide what is “commonly known as fully serializable execution.” The prominence of the table compared to this extra proviso leads to a common misconception that disallowing the three phenomena implies serializability.²

The Oracle Application Developer’s Guide (Reference [6]) is very clear on the subject.

Sometimes you must take steps to ensure that the data read by one transaction is not concurrently written by another. This requires a greater degree of transaction isolation than defined by SQL92 SERIALIZABLE mode.

The Oracle Concepts Manual (Reference [7]) reiterates the difference between Oracle and other database engines.

Although Oracle serializable mode is compatible with SQL92 and 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. This issue should be considered when applications using serializable mode are ported to Oracle from other environments.

But enough with theory, let's bring on the examples!

The First Case of the Disappearing Dollars

The following example illustrates the dangers of the default Oracle concurrency setting ("Statement-Level Read Consistency".) Create a table called BANK_ACCOUNT as follows.

```
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);
```

The following stored procedure transfers money from one bank account to another. It sleeps for sixty seconds (600 centiseconds) after checking the balances in the accounts, and this artificially introduced timing delay gives plenty of time for a second transaction to read the same data and create a problem.

```
create or replace procedure debit_credit(
  debit_account in integer,
  credit_account in integer,
  debit_amount in integer
)
is
  debit_account_balance number;
```

Free Lunch?

Membership in the Northern California Oracle Users Group brings with it one of the few exceptions to the "No Free Lunch" rule! For the low annual membership fee of \$70, members get four issues of the *NoCOUG Journal* and free admission to four full-day multi-track conferences jam-packed with technical presentations. Continental breakfast and a full lunch are served at the conferences (in addition to coffee, soft drinks, and snacks) and the cost of food alone exceeds the annual membership fee. Members owe the free lunch to NoCOUG's generous sponsors, Chevron, Lockheed Martin, and Oracle, who allow NoCOUG to use their conference facilities free of charge, and to the software vendors who advertise in the *Journal* and exhibit their wares at the conferences. ▲

```
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(600);
  --
  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;
```

¹ According to the ANSI SQL Standard: "The following phenomena are possible: 1) P1 ("Dirty read"): SQL-transaction T1 modifies a row. SQL-transaction T2 then reads that row before T1 performs a COMMIT. If T1 then performs a ROLLBACK, T2 will have read a row that was never committed and that may thus be considered to have never existed. 2) P2 ("Non-repeatable read"): SQL-transaction T1 reads a row. SQL-transaction T2 then modifies or deletes that row and performs a COMMIT. If T1 then attempts to reread the row, it may receive the modified value or discover that the row has been deleted. 3) P3 ("Phantom"): SQL-transaction T1 reads the set of rows N that satisfy some <search condition>. SQL-transaction T2 then executes SQL-statements that generate one or more rows that satisfy the <search condition> used by SQL-transaction T1. If SQL-transaction T1 then repeats the initial read with the same <search condition>, it obtains a different collection of rows." (Reference [1])

² The ANSI SQL standard also contains the following proviso: "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." (Reference [1])

In the following transaction history, Transaction A and Transaction B each transfer five dollars to Account #2 and, therefore, the balance in Account #2 should increase by ten dollars.

```
18:09:14 TRANSACTION A> execute
debit_credit(1,2,5);

PL/SQL procedure successfully completed.

18:09:15 TRANSACTION B> execute
debit_credit(3,2,5);

PL/SQL procedure successfully completed.
```

Here are the contents of the BANK_ACCOUNT table after the experiment. The balance in Account #2 is only fifteen dollars instead of twenty dollars.

```
18:09:21 SQL> select * from bank_account;

ACCOUNT#    BALANCE
-----
1           5
2          15
3           5

3 rows selected.
```

Reproducing these anomalous results in the “Real World” is difficult because of the split-second timing that is required, but we cannot guarantee that they will never happen when we use the default Oracle concurrency setting. Fortunately, the problem is easily solved by using the SERIALIZABLE concurrency setting, as in the following example in which Transaction B encounters the error “ORA-08177: can’t serialize access for this transaction”.

```
18:10:41 TRANSACTION A> alter session set
isolation_level=serializable;

Session altered.

18:10:41 SQL> execute debit_credit(1,2,5);

PL/SQL procedure successfully completed.

18:10:42 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

18:10:42 SQL> execute debit_credit(3,2,5);

BEGIN debit_credit(3,2,5); END;
*
ERROR at line 1:
ORA-08177: can't serialize access for this
transaction
```

Here are the contents of the BANK_ACCOUNT table after the second experiment. Transaction B will need to be resubmitted but this is more acceptable than losing a customer’s money.

```
18:10:49 SQL> select * from bank_account;

ACCOUNT#    BALANCE
-----
1           5
2          15
3          10
```

The Second Case of the Disappearing Dollars

The following example is from Reference [3] and shows that Oracle SERIALIZABLE mode will not prevent all anomalous behavior, exactly as Oracle itself warns in the *Application Developer’s Guide* (Reference [6].)

First create a table called BANK_ACCOUNT as follows.

```
create table bank_account (
  account# integer,
  account_type varchar(1),
  balance number
);

insert into bank_account values (1,'C',70);
insert into bank_account values (1,'S',80);
```

The following stored procedure transfers money from one account to another. Negative balances are allowed as long as the sum of the balances in the customer’s checking and savings accounts is greater than zero. Once again, we impose an artificial timing delay after checking the balance in each account.

```
create or replace procedure withdrawal(
  in_account# in integer,
  in_account_type in varchar,
  in_withdrawal_amount in number
)
is
  checking_account_balance number;
  savings_account_balance number;
begin
  select balance
  into checking_account_balance
  from bank_account
  where account#=in_account#
  and account_type='C';
  --
  select balance
  into savings_account_balance
  from bank_account
  where account#=in_account#
  and account_type='S';
  --
  user_lock.sleep(600);
  --
  if (checking_account_balance +
  savings_account_balance >= in_withdrawal_amount)
  then
    update bank_account
    set balance = balance - in_withdrawal_amount
    where account# = in_account#
    and account_type = in_account_type;
  end if;
  commit;
end;
```

Having learned the lessons of the First Case of the Disappearing Dollars, we use SERIALIZABLE mode in an effort to fend off anomalies.

```
09:39:58 TRANSACTION A> alter session set
isolation_level=serializable;

Session altered.

09:39:58 SQL> execute withdrawal(1,'C',100);

PL/SQL procedure successfully completed.

09:40:01 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

09:40:01 SQL> execute withdrawal(1,'S',100);

PL/SQL procedure successfully completed.
```

Our precautions were for naught. The constraint is violated and the sum of the balances in the two accounts falls below zero.

```
09:40:07 SQL> select * from bank_account;

ACCOUNT#  A  BALANCE
-----  -  -
         1  C      -30
         1  S      -20

2 rows selected.
```

What is a programmer to do? He or she must heed Oracle's warning very seriously and create artificial "choke points" or "serialization points" in every read-write transaction. This might limit concurrency but there is no free lunch! One way to create the necessary choke point in this example would be to use "SELECT FOR UPDATE" when reading account balances.

Poor Orphan Annie

The following example is taken from the Oracle *Application Developer's Guide* (Reference [6].)

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. This result occurs even if both A and B are SERIALIZABLE transactions, because neither transaction prevents the other from making changes in the data it reads to check consistency.

Create the PARENT and CHILD tables as follows and insert one row into the PARENT table.

```
create table parent (
  parent_name varchar(8)
);

create table child (
  child_name varchar(8),
  parent_name varchar(8)
);

insert into parent values('Warbucks');
```

The following transaction history shows that it is possible to create an orphan record. This is a good example of the "Do-It-Yourself Referential Integrity" (DIY RI) mentioned in Tom Kyte's presentation.

```
18:25:07 TRANSACTION A> alter session set
isolation_level=serializable;

Session altered.
```

(code continues on page 13)

QUOVERA

Oracle Consulting Solutions Specialists


With over 10 years of Oracle consulting excellence, Quovera has performed more than 300 engagements for over 200 clients. Quovera's specialized Oracle consulting services include:

- ◆ 11i E-Business Suite implementations and upgrades
- ◆ Full lifecycle custom and web development and data warehousing
- ◆ Business intelligence, reporting and business performance management

Quovera also provides expert mentoring and training on JDeveloper, Developer and Designer with its team of renowned Oracle Press Authors and Certified Masters. Quovera brings the experience of its Oracle consulting staff to work with you to improve your Information Technology processes and achieve your business goals.

Quovera, Inc.

800 West El Camino Real, Suite 100
Mountain View, CA 94040
Tel. (650) 962-6319 · www.quovera.com



Query performance slowing you down?

RightOrder QueryEdge: Accelerating the Performance of Resource-Intensive Queries by 10-50x

For more information, go to www.rightorder.com, call 408.583.4000, or drop by our table at the next NoCOUG event!

For the latest analyst whitepaper on improving query performance visit <http://www.rightorder.com/whitepapers/whitepaper>

www.rightorder.com



(APPLICATION CONFIDENCE)

DB2

SQL SERVER

QUEST
CENTRAL™

ORACLE

Three Solid Platforms. One Brilliant Solution.

You work in a heterogeneous environment. Your database management software should, too. With Quest Central you can manage more critical databases with astonishing ease — regardless of platform. It delivers comprehensive production management from administration to diagnostics, analysis, space management, SQL tuning and more — all with built-in expertise and a single intuitive, console-based interface. Try Quest Central. It's not just for Oracle anymore. Download "Surviving in a Multi-database Environment," at www.quest.com/nocoug and find out how Quest Central changes database management forever.



© 2004 Quest Software Inc., Irvine, CA 92618-1699 (408)337-9400 (USA)099

QUEST CENTRAL™: MANAGE MORE DATABASES TODAY

(code continued from page 11)

```
18:25:07 TRANSACTION A> select * from parent
where parent_name='Warbucks';

PARENT_N
-----
Warbucks

1 row selected.

18:25:16 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

18:25:16 TRANSACTION B> select * from child
where parent_name='Warbucks';

no rows selected

18:25:19 TRANSACTION A> insert into child
values ('Annie','Warbucks');

1 row created.

18:25:21 TRANSACTION B> delete from parent
where parent_name='Warbucks';

1 row deleted.

18:25:23 TRANSACTION A> commit;

Commit complete.

18:25:25 TRANSACTION B> commit;

Commit complete.
```

Here are the contents of the two tables after the experiment. Poor Orphan Annie!

```
18:25:28 SQL> select * from parent;

no rows selected

18:25:28 SQL> select * from child;

CHILD_NA PARENT_N
-----
Annie     Warbucks

1 row selected.
```

The way out of this predicament is to use the trusty SELECT FOR UPDATE (at the expense of concurrency) or to define a referential integrity constraint (which uses SELECT FOR UPDATE internally).

The Case of the Popular Projector

The next example shows that SELECT FOR UPDATE is not a panacea for all problems (if a concurrency limiting strategy can truly be labeled a panacea). The programmer has tried to implement the business rule that a resource such as a Projector cannot be doubly booked for the same time period. Here is the definition of the SCHEDULES table.

```
create table schedules(
  resource_name varchar(25),
  start_time date,
  end_time date
);
```

Here is the stored procedure that is used. It carefully checks that the requested resource has not already been

reserved for an overlapping time period. Once again we introduce an artificial time delay to force the problem.

```
create or replace procedure
resource_scheduler(
  room_name in varchar,
  new_start_time in date,
  new_end_time in date
)
is
  already_reserved integer;
begin
  already_reserved := 0;
  --
  select count(*) into already_reserved
  from schedules
  where resource_name = room_name
  and (start_time between new_start_time and
  new_end_time)
  or (end_time between new_start_time and
  new_end_time);
  --
  user_lock.sleep(600);
  --
  if (already_reserved = 0) then
    insert into schedules values
    (room_name,new_start_time,new_end_time);
  end if;
  --
  commit;
end;
```

Here is a transaction history showing that the above procedure does not prevent a resource from being double-booked for the same time period. Using SELECT FOR UPDATE on the SCHEDULES table will not help in this example. The solution is to create a separate RESOURCES table and update the resource record as part of the transaction. This concurrency-limiting strategy will prevent two users from making a reservation at the same time, even if the reservation is for different time periods. The second transaction will then fail with the ORA-08177 error.

```
18:19:08 SQL> alter session set
isolation_level=serializable;

Session altered.

18:19:08 SQL> exec
resource_scheduler('Projector', '2005/08/31
09:00', '2005/08/31 10:00');

PL/SQL procedure successfully completed.

18:19:10 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

18:19:10 TRANSACTION B> exec
resource_scheduler('Projector', '2005/08/31
09:30', '2005/08/31 10:30');

PL/SQL procedure successfully completed.
```

Here are the contents of the SCHEDULES table at the end of the experiment.

```
18:19:17 SQL> select * from schedules;

RESOURCE_NAME  START_TIME  END_TIME
-----
Projector      2005/08/31 09:00 2005/08/31 10:00
Projector      2005/08/31 09:30 2005/08/31 10:30

2 rows selected.
```

The Case of the Dangling DBA

Here is another example of how SELECT FOR UPDATE is not always the right solution. The programmer has tried to enforce a business rule that not more than 100 students may be admitted to a class. The stored procedure first counts the number of students in the class and only inserts a new record if less than 100 records are found. Once again we resort to an artificial timing delay in order to force the problem.

```
create table schedules (
  course_name varchar(32),
  student_name varchar(32)
);

declare
  i integer;
begin
  for i in 1..99 loop
    insert into schedules values ('DBA 101',i);
  end loop;
  commit;
end;

create or replace procedure
signup(
  in_course_name in varchar,
  in_student_name in varchar
)
is
  signups integer;
begin
  select count(*) into signups
  from schedules
  where course_name = in_course_name;
  --
  user_lock.sleep(600);
  --
  if (signups < 100) then
    insert into schedules values(in_course_name,
  in_student_name);
  end if;
  commit;
end;
```

Here is a transaction history that shows how the business rule can be violated. Using SELECT FOR UPDATE on the SCHEDULES table will not help. We will have to create a COURSES table and update the course record. This will cause the second transaction to fail with the ORA-08177 error.

```
19:05:08 SQL> alter session set
isolation_level=serializable;

Session altered.

19:05:08 SQL> exec signup('DBA 101','Iggly');

PL/SQL procedure successfully completed.

19:05:10 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

19:05:10 TRANSACTION B> exec signup('DBA
101','Ziggy');

PL/SQL procedure successfully completed.
```

Here are the contents of the SCHEDULES table at the end of the experiment, showing that the business rule has been violated because there are now 101 students enrolled in the course.

```
19:05:16 SQL> select count(*) from schedules
where course_name='DBA 101';

COUNT(*)
-----
101

1 row selected.
```

The Case of the Troublesome Tables

The following example is from Tom Kyte's book (Reference [4].) Two tables are initially empty and each transaction inserts a row into one table containing the number of rows in the other table. This time we don't need an artificial timing delay to force the problem!

```
create table a (x int);

create table b (x int);

18:19:18 TRANSACTION A> alter session set
isolation_level=serializable;

Session altered.

18:19:18 TRANSACTION A> insert into a select
count(*) from b;

1 row created.

18:19:26 TRANSACTION B> alter session set
isolation_level=serializable;

Session altered.

18:19:26 TRANSACTION B> insert into b select
count(*) from a;

1 row created.

18:19:27 TRANSACTION A> commit;

Commit complete.

18:19:31 TRANSACTION B> commit;

Commit complete.
```

Here are the contents of the two tables at the end of the experiment. This is not a "serializable" result because it cannot be produced by the serial execution, in any order, of the two transactions. One solution is to create an artificial choke point that allows only one transaction to succeed. This could take the form of a record in another table that both transactions must update.

```
18:19:33 SQL> select * from a;

X
-----
0

1 row selected.

18:19:33 SQL> select * from b;

X
-----
0

1 row selected.
```

We should point out that there is a very good case to be made that the above results are correct (even though they fail the serializability test) because each table accurately reflects the state of the database at a certain point in time. This implies that serializability is *not* the only test of correctness. In mathematical terms, serializability is *sufficient* for correctness but not strictly *necessary*.

And so we end the ditty

Oracle's concurrency management scheme is not a free lunch. Every read-write transaction must be carefully coded so that other transactions do not interfere with it, nor does it interfere with any other read-write transaction. Artificial "choke points" or "serialization points" may need to be created. Readers who wish to study the problem further should read the theoretical article on Oracle's concurrency management scheme published in the February 2005 issue of the *NoCOUG Journal*. The article is available for downloading at http://www.nocoug.org/download/2005-08/SERIALIZABILITY_-_White_Paper.pdf. The chapter called "Data Consistency and Concurrency" in the *Oracle Concepts Manual* (Reference [7]) is also essential reading. ▲

Copyright 2005, Iggy Fernandez

Iggy Fernandez is a Senior DBA with ADP and is Oracle 10g certified. Feedback and comments on this article may be sent to iggy_fernandez@hotmail.com.

References

- [1] ANSI. X3.135-1992, *Database Language SQL*, 1993. Available at <http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt>.
- [2] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. *A Critique of ANSI SQL Isolation Levels*, 1995. Available at <http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=5>.
- [3] 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>.
- [4] T. Kyte. *Expert One-On-One Oracle*. Wrox Press, 2001, Chapter 3.
- [5] 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>.
- [6] Oracle. *Application Developer's Guide*, 2005, Chapter 5. Available at <http://www.oracle.com/technology/documentation/database10gR2.html>.
- [7] Oracle. *Concepts*, 2005, Chapter 13. Available at <http://www.oracle.com/technology/documentation/database10gR2.html>.

Pioneering the Field of Capacity Optimized Storage

The Data Domain DD200 Restorer:
A disk-based recovery storage appliance designed for Oracle database backups:

- Works with leading backup software vendors and RMAN
- Patented Capacity Optimized Storage technology that enables long term data protection at low cost
- Data Inviolability Architecture ensures recoverability of backup data
- Rapid disaster recovery via cost effective site-to-site data replication

Data Domain
Just Another Data Protection

www.datadomain.com
1.877.622.2587

**You've got the skill
now get the tool**

DRFlash™ for Oracle helps you:
Gain visibility of Oracle Wait Events
Accelerate Oracle performance by more than 100%
Cut costs and find problems before they find you

CONFLUO
SOFTWARE

www.confluo.com
info@confluo.com
303.938.8282

EMERUS IT PARTNERSHIP

Performance Tips for Batch Jobs

by Chris Lawson

Batch jobs present their own set of performance challenges. The data set is usually very large, so many of the OLTP tuning tactics simply don't work. In this article I'll discuss some ways of dealing with batch job performance problems. The discussion is divided into three areas:

- General performance tuning
- Multi-threading tactics
- Partitioning considerations

Let's get started with a general list of performance tactics.

Part I. General Performance Tuning

Here is a shopping cart of effective ways to improve performance of batch jobs. I have found these methods to be very useful.

1) Use Set Processing

This is probably the most common performance lapse I see. The point is to avoid looping through millions of rows, issuing SQL calls as you go. Instead, try to process all the rows as a group. For example, looping 1 million times, running one SQL statement each time, is much slower than issuing a single SQL statement that retrieves all 1 million rows.

There are several reasons why set processing is so much faster. First of all, you can take advantage of multi-block reads (discussed next). Multi-block processing can sometimes give you a 10x speedup. In contrast, performing a huge number of single-block reads is rarely a good choice. The multi-block method is especially attractive if your tables are partitioned such that Oracle scans only a few partitions.

Set processing also avoids the degradation due to issuing thousands (or even millions) of separate SQL statements. This degradation is not really eliminated by the use of bind variables. Even if you use bind variables, Oracle must still process each statement and send back the result set. In extreme cases, the time used in sending the SQL statement repeatedly over the network actually creates a bottleneck.

SQL scripts, rather than a procedural language, are oftentimes the better approach. Of course, with SQL scripts, you are pretty much forced to handle the data as a group. Set processing means you may need staging tables to hold intermediate results. This is common practice.

2) Take Advantage of Multi-Block Reads

On most servers, Oracle can read up to 1 Mb (typically 64–128 blocks) at one time. That is why a full table scan

can be performed so quickly. Keep in mind, however, that two conditions must be met:

- 1) The database parameter *Db_File_Multiblock_Read_Count* must be set correctly.
- 2) The table or index being scanned must have extent sizes of at least 1Mb.

If the multi-block parameter is set too low at the database level, you can easily alter your session to set the parameter higher. The second point above recognizes that Oracle will not continue a multi-block scan across extent boundaries. If most of the extents are greater than 1 Mb, it's probably okay. (This is one of those cases where extent sizing really does matter.)

It is especially important to optimize your reads if you are using Oracle parallelism. Why launch many extra processes if you don't first optimize what a single process can do? That brings us to the next point.

3) Optimize Queries Before Using Oracle Parallelism

As long as there is spare capacity on your server, you can usually improve your full table scans by using the PARALLEL hint. But first, make sure the query runs as efficiently as possible. If the query runs badly with one process, it won't be much better with six or eight—plus, you will be consuming server resources unnecessarily.

As a general rule of thumb, a parallel degree of 6 typically gives excellent performance. Of course, this assumes that you have adequate reserves (both CPU and disk throughput) on your server.

4) Avoid Massive Deletes

Oracle is simply not very fast at deleting millions of rows. Instead, copy and temporarily store the rows you do want, truncate the table, and then put the rows back in. This can easily be 10x faster. If this method is not feasible due to business considerations, consider multi-threading (discussed later).

5) Use Summary Tables and Materialized Views

If you repeatedly access a large table, consider building an aggregate table, or materialized view. A materialized view used in this manner is just like a table, but you can rebuild it with a single command. Remember to use Oracle paral-

lelism to speed up the refresh.

Whether you use a materialized view or an actual table, the idea is to create a “pre-digested” form of the data. You include only the columns and rows that meet your conditions. Once you build the summary table, you can also build custom indexes that optimize your query.

6) Avoid Excessive Commits

A commit after every row will usually wreck performance. Most jobs should commit no sooner than every 1,000 rows. Committing every 10,000 rows would be a good rule of thumb. Unless you are working with many millions of rows, further increasing the commit interval doesn't really offer much improvement.

7) Don't Forget to Analyze New Tables

If you build and populate a new table, don't forget to gather statistics. It's very common to see a performance bottleneck caused by incomplete or inaccurate table statistics. It's not necessary to sample all the rows when gathering statistics; the “estimate” option is usually fine.

8) Turn Off Logging When Feasible

Oracle allows you to turn off transaction logging for a handful of cases: creating or rebuilding indexes, inserting rows, and *Create Table As Select*. To do this, run *Alter Table [name] NoLogging*, and then use the hint *NoLogging*. Remember that deactivating logging means the data cannot be recovered after a disk crash and database recovery. So this is not appropriate in every case.

9) Speed Up Inserts

In cases where you don't really need transaction logging, you can speed up inserts a bit by using the *NoLogging* feature. For inserting rows, set the table *NOLOGGING* (using *Alter Table . . .*) and then use this syntax: *INSERT /*+APPEND */*.

Remember, however, that all the new rows will be placed at the end of the table—above the “high water” mark. This means that if you are performing deletes from the table, these “holes” will never be filled. The table will grow rapidly. Eventually, you will need to rebuild the table to crunch it down.

10) Avoid Building Flat Files

When transferring data to another Oracle database, it's far simpler (and usually faster) to transfer data over a database link. In other words, don't take the data out of the database if you're just going to put it back into another Oracle database.

11) Use Oracle Parallelism

For scans of huge tables or indexes, it is often good practice to invoke Oracle Parallel Query Option (PQO). On most systems, using a parallel degree of 6 gives excellent performance. Of course, this assumes that you have the spare resources (disk and CPU). For example:

```
Select /*+Parallel (T 6) */ Emp, Name from Giant_Table T
```

Remember that Oracle will actually start a total of 2x the degree specified (one set for reading, another set to process/sort). So the example above will actually result in 12 processes.

It is usually best to invoke parallelism via a SQL hint, rather than setting the table to a degree higher than 1. If you set the table to invoke parallelism, Oracle will tend to start up the slave processes for many queries, whether you actually want parallelism or not.

12) Use Bind Variables

If you will be repeating a massive number of SQL statements, it is very important to properly use bind variables. In this way, the database engine avoids a re-parse of each

TECH TIPS

Free Two-Day Oracle Self-Study Course

Oracle Corporation is offering a free, no-strings-attached Oracle 10g DBA course for individuals who are new to Oracle. Billed as a “free DBA self-study tutorial,” it is a two-day course designed for individuals who are new to database administration. Through a series of ten web pages, the course walks participants through basic database administration activities ranging from installing Oracle software to creating a new tablespace. Various tasks are demonstrated using Enterprise Manager and other GUI tools. The course includes topics such as:

- Installing Oracle and Building the Database
- Getting Started with Oracle Enterprise Manager
- Configuring the Network Environment
- Managing the Oracle Instance
- Administering Users and Security

You'll find it at http://www.oracle.com/technology/obe/2day_dba/index.html. ▲

statement. Also, a large number of unique SQL statements tends to flood the shared pool, causing other SQL statements to be released as the engine makes room for the new SQL. This will annoy the DBA.

13) Allow the Database Engine to Crunch the Statistics

Oftentimes, reports will need statistics such as average sale price, etc. It is usually best to allow Oracle to do the number crunching. It is rarely necessary to transfer the actual raw data from the database to the report server.

If you mistakenly transfer huge amounts of data out of the database, you will pay the price of a large network delay, plus the time for the report server to crunch the data. Oracle can easily retrieve a million rows and throw them at the report server. The report server, however, will likely be overwhelmed when you ask it to process a million rows of data.

It is much better to leave the raw data in the database and allow the database engine to crunch the statistics. This can easily be 100x faster.

14) Consider Parallel DML

Similar to parallel queries, you can invoke Oracle parallelism to perform transactions in parallel. There are many restrictions, however, that limit the situations where you can do this. Probably the most significant limitation is that you cannot have an active trigger on the table being manipulated.

My tests show that the gain from executing parallel transactions is not nearly as great as that for parallel queries.

Part II. Multi-Threading Tactics

Let's now turn our attention to a special technique that has proven useful.

For cases where you need to perform lots of near-identical transactions against a massive table, consider a "multi-threading" strategy. Although some program redesign will be necessary, the gain is typically very large. At first, this technique might seem crude, but don't underestimate its power.

Let's try an example to see how this works. Suppose you want to update the title (say, remove the leading blanks) for a set of products at many stores. You will need to update the giant *Product_Line* table, which contains 100 million rows. A typical transaction might look like this:

```
Update Product_Line
Set Product_Title = Ltrim(Product_Title)
Where Product_Number = 900 and Position = 100;
```

Assume that you need to repeat this transaction a total of 100,000 times. Since the table is extremely large, most of the blocks will not be cached. This means that each update will require at least one disk read to retrieve the data block. This will be true no matter how you set up the indexes—you simply need to read the data block off disk.

At a typical disk I/O rate of 100 reads per second, this means a delay of 1,000 seconds—just to get the blocks to change. To this time you need to add the delay for executing 100,000 round-trips across Sql*Net. Assume a fairly quick time for each round-trip of only 5 ms, or 500 seconds, more.

What alternatives do you have? You can choose from the following:

- Option 1 (our baseline option): Run the DML 100,000 times, accessing each row via an index.
- Option 2: Store the list of rows to be updated in a temp table and then rewrite the code to perform a full scan of the *Product_Line* table, using the temp table to identify the rows to be updated.
- Option 3: Use option 2 but invoke Oracle Parallel DML to launch multiple processes.
- Option 4: Run the DML once, using a bulk collect to identify a group of rows to be updated. Each row is accessed via an index. (Note: trying to combine with Parallel DML was a disaster—we produced a massive number of runaway parallel processes.)
- Option 5: Run 10 threads, each of which processes 10,000 rows via the index.

Let's analyze each option in our thought experiment:

Option 1: This is our baseline option. The 100,000 physical reads mean a runtime of about 16 minutes just to access the relevant blocks in the table. To this you must add 8 minutes for the cost of making 100,000 round-trips over Sql*Net. Total runtime: 24 minutes.

Option 2: Perform one full scan instead of 100,000 index lookups. A table this large typically has about 3 million Oracle blocks (and the vast majority will not be cached). Assuming an average read of 2,000 Oracle blocks per second, this means a delay of about 1,500 seconds. Total runtime: 25 minutes.

Option 3: Actual testing with parallel DML shows a performance of 2.5x. This means a runtime of about 25 minutes/2.5 = 10 minutes. (Note that future versions of Oracle might show a better performance gain.)

Option 4: The bulk collect eliminates the numerous Sql*Net round-trips. However, it does not eliminate the 24-minute disk-access time. Total runtime: 16 minutes.

Option 5: Eliminate the Sql*Net round-trips—but you still must account for the disk access time of 16 minutes. The key here is that you spread this time over the 10 threads. Of course, the time will not go down linearly to 1.6 minutes, but many systems will show an excellent trend. Assume a 20% penalty. Total runtime: 2 minutes.

Note that all multi-threading schemes must account for the possibility of locking among the various threads. So each thread must be assigned a group of keys that cannot possibly conflict with any other group. This is really not very difficult in practice.

Also, many threads operating simultaneously on a table bring up the possibility of ITL locking. This happens when multiple processes update the exact same block but there is no room in the header block to expand. This is easily avoided by building a table with INITRANS set to more than 1.

Part III. Partitioning Issues

Most large database tuning involves partitioning. This subject can be very confusing, but it also offers great performance benefits—if used properly. Here are some ideas that I have found useful.

1) Design Partitions to Match the Queries

Partitioning a table does not automatically improve performance. The objective of partitioning is to reduce (or “prune”) most of the data to be searched, leaving a small group that still meets your conditions. In order for this to happen, the partition key must match the condition used in your queries. For example, if most of your queries contain a date restriction, then you could partition by a date-related column and eliminate many of the partitions.

There are two main ways that partitioning can speed up queries:

- 1) If you need to perform a full table scan, Oracle can substitute a full partition scan. In this case, having lots of small partitions really helps.
- 2) If you perform an index scan, the index will be much smaller, and the scan will be slightly more efficient. (This boost may not be too noticeable unless you are doing a huge number of index scans.)

2) Indexes on Partitioned Tables Typically Should Be Local

A local index has partitions that are aligned one-for-one with the table partitions. So, if you have 256 partitions in a table, your local index will likewise have 256 partitions. Local indexes are attractive because they get a similar performance benefit as the table—that is, reduced size of the data set. All things being equal, reducing the size of the data set will improve performance.

A global index, on the other hand, does not match the table partitioning. You partition it *differently* from the table. For example, you can have a different number of partitions, or even partition the index using a different key.

In most cases, however, a global index is not partitioned at all—it is just one big segment that spans all the partitions of the table. When people say “global index,” this is usually what they mean—a *nonpartitioned* index.

There is a disadvantage to global indexes that makes them less desirable. If you modify any partition in the table, the global index becomes *invalid* and must be rebuilt. DBAs do not like this feature.

When you create an index, it will be *global* unless you explicitly specify *local*! Similarly, if you add a primary key and specify the *Using Index* clause, the index will be *global*. So it's easy to mistakenly create a bunch of *global* indexes.

Watch your email and don't forget to renew your NoCOUG membership for 2006!

3) Be Aware of Global Index Exceptions

Although most indexes on partitioned tables will likely be local, there are important occasions for which to use a global index. Specifically, you will need a global index when your query conditions *don't match the partition key*.

For example, assume you need to retrieve inventory information as of a certain *Sale_Date*. (That column is indexed.) The table has 256 partitions, but the partitions are keyed on *Factory*, not *Sale_Date*. If the index were local, a query based on *Sale_Date* would require combining the scans of all 256-index segments to get the result. On the other hand, if our index had been created *global*, Oracle could just perform one read of the single (larger) index. This would be far faster. ▲

Chris Lawson is a performance specialist living in the San Francisco Bay Area. He specializes in tuning Retek applications and also publishes the online newsletter The Oracle Magician. Chris can be reached at Chris@OracleMagician.com.

www.EMC.com/relodge.'"/>

EMC
where information lives

From: expecting the world from Oracle
To: getting the universe

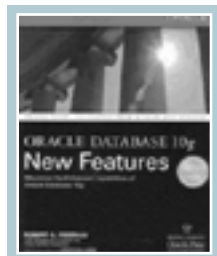
EMC CAN HELP YOU OPTIMIZE ORACLE INFORMATION ACROSS ITS ENTIRE LIFECYCLE. Our services, software, and hardware help you get more from your Oracle database and applications. Developed jointly with Oracle, our solutions give you the power to improve availability, reliability, and flexibility while lowering TCO. You get a custom information infrastructure, proven to work in the most demanding situations — including migrations, upgrades, backups, and peak workloads. Visit www.EMC.com/conditions to learn more and sign up for a live demo. Or call 1 866 464 7381.

Find an authorized EMC reseller: www.EMC.com/relodge.

EMC, EMC logo, and EMC logo with tagline are trademarks of EMC Corporation. © 2005 EMC Corporation. All rights reserved.

Oracle Database 10g New Features

Review by Brian Hitchcock



Title: *Oracle Database 10g New Features*
By: Robert G. Freeman
Published by: Oracle Press
ISBN: 0-07-222947-0

Summary

Overall review: Adequate for its intended purpose.

Target audience: DBAs and application developers.

Would you recommend to others?: No.

Who will get the most from this book?: Those who need a quick high-level review of the new features in the 10g database.

Is this book platform specific?: No.

Why did I obtain this book?: I didn't, I was asked to review it for NoCOUG, and it was given to me.

Overall review

Before I read this book, I had read and reviewed the 10g DBA OCP upgrade exam book. Therefore, I came to this book having already gone over a lot of the technical details of the new features in the 10g database. However, the OCP exam guide was much more focused on the very detailed technical aspects of the new features, while this book gives a much higher level, and arguably more thorough view of the major new features of the 10g database.

The very nature of the task of writing about new features is problematic. First, it is largely a marketing exercise, which isn't necessarily a criticism. The task itself is to get the material out close to—and perhaps even a little before—the general release of the new version of Oracle. Therefore, much of the material may be relatively unproven, in that many of the new features described may or may not be fully implemented in the finished product and will not have been in wide general use by the technical community for any length of time.

This limits the practical value of much of the material, in that you're never sure if what you're reading about will actually be fully implemented in the product you get. Plus, the book describes how much you need to be careful to test the new features before you rely on them.

Another challenge of reviewing a book about new features is that I'm commenting on the book itself *and* the

Oracle features being described in the book. I've attempted to do both, and attempted to be clear on when I was doing the former versus the latter. The reader needs to know a lot about a feature in order to get much out of the text covering the changes to it. In many cases, a feature is described so briefly that it isn't clear what that feature does or how it will be useful.

The reader needs to know a lot about a feature in order to get much out of the text covering the changes to it. In many cases, a feature is described so briefly that it isn't clear what that feature does or how it will be useful.

Chapter Reviews

- 1 Getting Started
 - 2 Server Manageability
 - 3 Performance Tuning
 - 4 Security
 - 5 Availability and Recoverability
 - 6 Business Intelligence
 - 7 Application Development
 - 8 Other Database New Features
 - 9 Oracle Enterprise Manager 10g
- Appendix Oracle Database 10g new processes

Chapter 1 Getting Started

The author starts out by telling us that the “g” in 10g refers to “grid.” He then goes on to point out that no one seems to be sure what grid really means in general, let alone what it means for Oracle 10g in particular. The author then gives four bullet points about grid that don't really tell me what grid means. For example, one of the bullet items states that the grid enables you to “leverage components made available to the grid.” What does that mean? How do I configure Oracle products to be available to the grid? I'm surprised that a book covering 10g new features can't tell me what the grid is. Given Oracle's well-known abilities as a marketing machine, I'm puzzled that they haven't more fully explained, at least in marketing terms, what the grid

and the “g” in 10g mean to Oracle users.

The author covers upgrading to Oracle database 10g with reference to the new Database Upgrade Assistant (DBUA) GUI tool. The author says he prefers manual upgrades, but he unfortunately does not explain the reasons for his preference. I also prefer manual upgrades vs. using a GUI upgrade tool, and my opinion is that it’s better to control all aspects of the upgrade, rather than to run an automated tool and find out later which pieces worked and which didn’t. Next we’re told—not the first and not the last time—that using any new features without extensive testing is unwise.

In addition to the author’s writing, this book also contains commentary by Jonathan Lewis. We’re told in the introduction that Mr. Lewis is known as “Mr. Oracle.” Our first comment from “Mr. Oracle” reiterates that we should not use any of the new features described in this book without extensive testing. Given the description of the testing Lewis says is needed for every new feature in the book, I think you would need a small team of dedicated full-time DBAs just to review the new features. These days, does anyone have those kinds of resources available? Simply telling me that new features are risky is not the practical advice I’d prefer, and it doesn’t provide any insights into how we are supposed to deal with Oracle new features in our real-world jobs.

The author also describes how you can upgrade using “create table as select *.” This puzzles me. I’m not sure there are very many databases where it would be practical to do an upgrade by creating a copy of every table of the old database. I’m not sure that this method would really be of any use to the vast majority of the Oracle user community. A relatively detailed description of ASM (automated storage management) is given, but I’m not clear why this is in a chapter called getting started versus a chapter of its own.

I had a lot of unanswered questions about this new feature. Do I really need a new volume manager? How will the DBAs and SAs interact when the ASM is used? How will the SA do his or her job if the disks are only visible to the Oracle ASM instance?

These are the kinds of issues you need to think about before you can really make an informed decision about whether to use new features or not, and these issues are not covered in this book. These issues will only come up

Given Oracle’s well-known abilities as a marketing machine, I’m puzzled that they haven’t more fully explained, at least in marketing terms, what the grid and the “g” in 10g mean to Oracle users.

as ASM is really used. Taking this step further, think about how ASM would affect outsourced SA support for a data center. Can you implement a new feature like this with the support model your business is currently using? Rolling upgrades are described, but there are many restrictions on this new feature. It becomes clear as you read this book, that at least some of the 10g new features seem only partially ready. Some of them sound like they might be a good idea but they don’t all

seem to be fully implemented yet.

Cluster ready services are described, and after reading four paragraphs, I don’t know what they are or what they do. Sounds like yet another layer of “services,” another set of commands, and potentially new bugs. What will I do with the ability to “assign services to specific instances in the cluster”?

On page 16 we have what I think illustrates what can go wrong with a book on new features. I’m told the name of the new parameter, its default, and its minimum and maximum values, and that’s it. Oh, I’m also told this parameter “defines the initial number of server processes in global cache service that are started to serve inter-instance traffic.” I’m sure this is terribly cool, but I don’t know what any of it does and I don’t see any further description from the author on this.

Chapter 2 Server Manageability

The new feature of automated statistics collection is described and this includes lots of new packages and procedures. I’m not clear how to know which of these packages and procedures to use, when to use them, and who will monitor and maintain all of this, let alone review all the data that will be produced. There is no mention of how this new automated statistics collection feature might affect performance of your application after you migrate to 10g. I would like to have some commentary from Mr. Lewis on this new feature.

This is a good example of the problematic nature of writing a book on the topic of new features. In general, it’s tough to really explain why we need any one of these new features. In 10g we’re told that we should analyze the tables in the data dictionary, but I’m curious as to why we were told not to do this before 10g. The author offers no explanation. Again, this book is mainly a listing of the new features and does not provide much help as to the effects of trying to implement these new features.

Rolling upgrades are described, but there are many restrictions on this new feature. It becomes clear as you read this book, that at least some of the 10g new features seem only partially ready. Some of them sound like they might be a good idea, but they don’t all seem to be fully implemented yet.

The section on the new Scheduler is very thorough. I'm curious how this compares with the UNIX cron utility that I currently use. Again, it would have been helpful for the author to comment on issues like this. Oracle has made it much easier to understand things like the repeat interval when using the job scheduler, since we can now use terms like "daily," which is much better than the previous syntax. This is an example of where this book is good. Learning this sort of specific new feature information makes me actually reconsider using Oracle's job scheduler. In the past I haven't really considered it, because I felt it was so hard to use compared to the UNIX cron.

At the same time, the new job scheduler is a good example of what's wrong with many Oracle features. (This is true of almost all software, not just Oracle.) It is so flexible, and has so many features that a clever DBA could easily deploy a set of scheduled jobs that are very hard to understand and maintain. The administration costs of such features are an argument against implementing them. Yes, something new is possible in Oracle 10g, but that doesn't mean you can afford to implement it. And, with each new release there are a few new things that are simple and useful. The user configurable default tablespace is one of these things. I also like the ability to rename a tablespace, but note that the associated datafile name doesn't change. The ability to drop and shrink tablespace is also very useful.

I also like the ability to rename a tablespace, but note that the associated datafile name doesn't change. The ability to drop and shrink tablespace is also very useful.

Chapter 3 Performance Tuning

Oracle 10g provides us with many more wait events than before, and the description of how these new wait events have been sorted into wait events groups is good. There are changes to the v\$session table to make it easier to see which waits are associated with which sessions, as well as blocking information and wait history.

The automated workload repository (AWR) is described, but it's hard to assess what the real impact of this will be. How easy will it be to get usable results out of this new feature? It's too soon to tell. With all of these new features coming in 10g, I wonder if something like OEM (Oracle Enterprise Manager) may be required to simply keep track of all of the different kinds of repositories and advisories that are now available in 10g.

Learning this sort of specific new feature information makes me actually reconsider using Oracle's job scheduler.

Further, 10g will give us many alerts that will tell us when things may be going wrong, but who's going to review, monitor, and respond to them all? Imagine the data center with hundreds of databases that could potentially generate thousands of alerts on a regular basis. The author does not comment on how these alerts might be implemented or managed.

Automated SGA tuning sounds very good, but upon reading the description given here, it's not fully implemented. You still have to manually set some parameters. Personally, I think we need to simply set the total memory for the SGA, and that's it.

10g offers some new tracing abilities that are good, but also mean you'll be generating lots more data when you do your tracing.

Chapter 4 Security

This chapter briefly describes some improvements over 9i for security features, although there really isn't a whole lot of information here. This chapter is all of five pages long.

Chapter 5 Availability and Recoverability

In this chapter, we learn about recovery through the reset log command, specific changes to log archive formats, new standby database features, and changes to the way you can make backups, including specific RMAN changes. The new flash recovery area is described and this information is useful as it makes clearer how this new feature affects your backup process, and so on.

I find it interesting that Oracle is now telling me how RMAN is improved because it takes actual image copies of my datafiles as backups, whereas previously RMAN was better because it only recorded the block changes. How times change.

There is also a fascinating comment from Lewis. We're told that disks have become much too big these days and that we should only use about 10 GB per disk. However, I can't see myself going to my management and telling them that for a terabyte database where we will be using new 70 GB disks, we will need to purchase 7 terabytes of disk space before mirroring.

While the new capabilities of various flashback features are described, flashback has always puzzled me. The author provides good coverage of the many new flashback features. Being able to flashback the entire database doesn't do me

10g offers some new tracing abilities that are good, but also mean you'll be generating lots more data when you do your tracing.

much good, unless the database is only used by a simple single application. If you use flashback to reverse a single data change, you'll move all of your business back to that time, without regard for the other data changes that have been made since the time you flashed back to. Similarly, you can flash forward as well as flashback, but I'm not sure these are good ideas. Does your business really want to move back and forth in time? How, for example, would this play with the many SOX requirements we now have to deal with? And I'm also concerned about the amount of disk space that would be needed to retain sufficient redo information to support flashback (and flash forward) of a large active database over a reasonable period of time, for example, several days. My users don't usually tell me the very minute that they drop something they meant to keep, it usually takes days for such an event to come to light.

Chapter 6 Business Intelligence

This chapter devotes a lot of its coverage to Oracle Data Pump, which replaces (or, at least, supplements) the export and import utilities. The coverage here is thorough and clear. However, Lewis adds that he "can't get very excited about Data Pump Export." I have had to support many performance issues relating to moving data into and out of databases. Therefore, I would have appreciated expert advice from Lewis on the benefits and best way to use Data Pump.

The rest of the chapter covers new features like big file tablespaces, which allows you to have one very large datafile for a tablespace instead of a group of smaller datafiles. Cross-platform transportable tablespaces are described as well as various enhancements to specific SQL statements that will allow you to do data analysis. New partitioning features are also covered. This material is thorough and clear, but doesn't offer you any insights as to how to determine when these new features will help you with your specific application needs.

Chapter 7 Application Development

Oracle 10g allows you to use regular expressions in your SQL and PL/SQL processing that allow you to do complex pattern matches in your string data. There are clear descriptions of the new PL/SQL packages and information on some things that I've never had to use such as Oracle collections, temporary tables with varray columns, tablespaces for nested tables.

We're told that the new 10g PL/SQL compiler is 50 to 75% faster than the 9i compiler. I would like to have had the author's comment on some testing to verify this improvement but none is offered. I hope that these performance improvements turn out to be true, as that would be a big help to many users.

Chapter 8 Other Database New Features

This chapter covers an even wider range of new features than the previous chapters. Some of these new features are improvements to things I have never used before so I can't really say if these improvements are significant or not. For example, the very detailed description of workspace man-

In our next issue of the *NoCOUG Journal*, watch for another of Brian Hitchcock's popular book reviews. To suggest a title for review, please write to journal@nocoug.org.

ager enhancements, including multiparent workspaces and workspace manager events, may be very useful to those who are familiar with this feature. There's also a description of exporting and importing version data from workspaces, but again I'm not sure if this is a significant feature or not.

There is a section on streams enhancements. I don't think there is enough explanation of what streams even are, so unless you are familiar with streams, this section may be hard to understand. Here again, you need to have some expertise with the specific feature before you can get much out of the sometimes very brief description of the enhancements to specific features contained in this book.

There are several pages describing SQL*Plus enhancements that I found very interesting, because I have used SQL*Plus extensively over the years and these enhancements will be relevant to my day-to-day use of the Oracle database. There is also a section on SQL language new features, however I believe these are relatively obscure features that I don't have enough background to appreciate. For example, I'm not sure how to assess the benefit of the new removal of aggregate and SQL statement length limitations.

Chapter 9 Oracle Enterprise Manager 10g

This chapter focuses on changes and improvements to the Oracle Enterprise Manager (OEM). The description of these changes is thorough and easy to follow. It includes some screenshots of the web pages you'll see as you use the new OEM features. OEM is now browser based, so you can manage your databases from anywhere you have a browser that can connect to your network.

OEM now comes in two flavors, Grid Control for managing multiple databases and application servers and Database Control for managing a single database. It is clear to me that as the authors approached the last three or four chapters of this book, they were running up against time and/or physical space limitations because the amount of information on any given new feature becomes smaller

Vendors Wanted

If your company is interested in being a vendor at a future NoCOUG Conference, contact board@nocoug.org.

and smaller to the point where some are mentioned in name only.

Appendix Oracle Database 10g New Processes

The appendix is all of three-quarters of one page, but is useful in that it identifies all the new operating system-level processes that you may have when you start using the 10g database. Again, this highlights how many more things you may have to install, configure, and manage as you implement the new features in 10g. While I think the new features of 10g will improve many long-standing DBA issues, I think there is also added complexity for some of the newest features that will simply replace old issues with new issues. For those who think each new version of Oracle reduces the need for a DBA, I think it simply changes what the DBA does.

I think it is important to keep in mind that new features are just that, new features. They aren't necessarily good features that will help you with your real-world applications.

Conclusion

With regard to new features in general, not just for 10g, I'm reminded of one of the better training experiences I ever had. This was a third-party training class on Oracle 8i, and the instructor pointed out that in many ways Oracle has to create new features simply to maintain the myth that they are the technology leader in the arena of relational databases. And while the specific example described by the instructor is now well out of date (this was 8i after all), I think the message is still relevant. After all, how many of you are now storing video inside of your Oracle database? But at the time that 8i was new, storing video in the database was an example of how Oracle was superior to the competition. I think it is important to keep in mind that new features are just that, new features. They aren't necessarily good features that will help you with your real-world applications.

This book is a fine attempt to convey a high-level review of all the major new features of the Oracle 10g database. However, I don't see that this book is much better than the marketing materials you can get for free from Oracle's website, so I don't see that you need to buy this book.

To put the whole idea of how we buy enterprise software in perspective, consider how it would sound if we were discussing the new features of a new car. The author would tell us that the new car has big shiny fins, whose purpose is not clear. We would review each of many new things about the car, many of which have not been available before so we wouldn't know what, if any, effect on driving they would have. Finally, "Mr. Automobile" would advise us that after buying, we must completely disassemble the car and thoroughly test each component separately before we should consider driving our new car. And "Mr. Automobile" would be unable to get excited about the car's fuel mileage.

Would you buy a new car this way? Then why do you buy mission critical enterprise software this way? ▲

About the Author

Robert G. Freeman has been involved in Oracle for almost a decade and a half. Having decided many moons ago that selling trucks was not the life for him, Robert moved into computing. From there he started delving into C programming and then he quickly moved into UNIX administration and then to Oracle. As a consultant, Robert has experienced many interesting assignments all over the United States. Additionally, Robert has spoken at many user groups, both national and international. In his free time (does he have free time?) Robert enjoys being with his new wife (whom he totally didn't deserve), his wonderful kids, and Clifford the cat (and no, Clifford isn't eight feet tall). He's also trying to learn French, but is sure he will never get it.

About the Reviewer

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

The 2006 NoCOUG Board of Directors will be elected at the NoCOUG board meeting next month. Stay tuned to learn of any changes to NoCOUG leadership for 2006.

Many Thanks to Our Sponsors

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low-cost memberships. If your company is able to offer sponsorship at any level, please contact NoCOUG's president, Darrin Swan, at darrin.swan@quest.com. ▲

Long-term event sponsorship:

LOCKHEED MARTIN

CHEVRON

ORACLE CORP.

Thank you! Year 2005 Gold Vendors:

- ▶ Confio Software
- ▶ Database Specialists, Inc.
- ▶ Data Domain
- ▶ Embarcadero Technologies
- ▶ Quest Software
- ▶ Quovera, Inc.
- ▶ RightOrder

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



TREASURER'S REPORT

Randy Samberg, *Treasurer*

Beginning Balance

July 1, 2005 \$ 54,929.75

Revenue

Membership Dues	1,810.00	
Meeting Fees	800.00	
Vendor Receipts	5,500.00	
Training Day	980.00	
Advertising	---	
Interest	57.11	
Miscellaneous	---	
Total Revenue		\$ 9,147.11

Expenses

Regional Meeting	10,172.42	
Journal	6,995.67	
Membership	34.95	
Administration	11.47	
Website	---	
Board Meeting	585.33	
Training Day	29.64	
Marketing	---	
PayPal	74.00	
Miscellaneous	77.48	
IRS	---	
FTB	331.00	
Insurance	---	
IOUG-Rep	---	
Total Expenses		\$ 18,311.96

Ending Balance

September 30, 2005 \$ 45,764.90

NoCOUG Fall Conference

Thursday, November 10, 2005

Session Descriptions

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

KEYNOTE

Why: Why “Why?” Is Probably the Right Answer

Tom Kyte, Oracle Corporation

This short keynote presentation will present Tom’s view on why “why” is probably the right initial answer to give to most technical questions; why just answering the question as asked can likely cause more harm than good.

Track 1:

All About Binds

Tom Kyte, Oracle Corporation

We’ll briefly go over why using bind variables is extremely important with regard to performance, scalability, and even security, but quickly move into topics such as: Do I always want to bind? (Surprisingly, the answer is no.) What is bind variable peeking? Is it good or evil in disguise or a bit of both? So the developers don’t bind; is cursor_sharing=force/similar appropriate system-wide? (Emphasis will be on the reasons why setting cursor sharing at the instance level is not such a good idea.) What is the real difference between cursor_sharing=force/similar and which should we use under what circumstances? The presentation will be practical, with many examples and hard numbers you can use in your day-to-day work.

Logical Standby Databases for Reporting

Mark Bole, BIN Computing

Not quite ready for a data warehouse, but still want a high-performance reporting environment that doesn’t impact

your on-line transaction (OLTP) users? New in Oracle 9i, the Data Guard logical standby database is one answer. One or more core schemas are automatically synchronized with your live database, as frequently as you wish, without creating any load on the source database. Once updated, the core data is protected from any alteration, yet any other tables and schemas can be created in the same database to extend your data model and provide reporting performance. Materialized views, query rewrite, and other typical data warehouse techniques are all available.

Transportable Tablespaces and Data Pump

Caleb Small, caleb.com

These are fast and efficient methods of moving large amounts of data between databases and across platforms. The Transportable Tablespace feature literally allows you to unplug tablespaces from one database and plug them into another (some restrictions apply). Data Pump, the new generation of Import/Export, expands this functionality by bridging the gap across platforms. This presentation includes a live demo of moving relational tables and indexes from a Linux database to Windows, with many of the caveats and gotchas exposed.

Advanced Triggers—Procedural DDL, Security, Auditing, and More

Caleb Small, caleb.com

Traditional table-based triggers allow implementation of complex business rules inside the database, creation of custom security rules, and detailed auditing, among other things. System and DDL triggers expand this functionality to system events such as startup/shutdown, login/logout, and individual DDL statements. This session explores how to make your database more secure and flexible; track or prevent DDL changes; implement rules based on data, user or time values; special rules for data loading, debugging triggers, and more.

Track 2:

Tips and Tricks for Customizing Portal Reports Portlet Modules

Peter Koletzke, *Quovera*

Portal (from version 3 to 10g) contains an Oracle Reports Portlet that allows you to easily present a parameter form. This form calls standard Oracle Reports RDF (and REP) files. As with most of the development tools in Portal, the Reports portlet is completely functional but the interface it offers is very basic and you will probably want to customize it. This presentation offers tips and techniques you can use to tweak the functionality and user interface of these Reports modules. After a short review of the basics, it describes undocumented or little known techniques such as customizing the template, defaulting fields to specific values, removing defaults from a previous session, hiding the schedule tab, changing the DESFORMAT prompt, creating a multi-select control, coding a shuttle (left-to-right) select control, changing the button labels, modifying the font, conditionally displaying fields based on other fields,

ADVERTISING RATES

Contact: **Nora Rosingana**

325 Camaritas Way · Danville, CA 94526

Ph: (925) 820-1589

The NoCOUG Journal is published quarterly.

Size	Per Issue	Per Year
Quarter Page	\$100	\$320
Half Page	\$200	\$640
Full Page	\$400	\$1,280

Personnel recruitment ads are not accepted.

customizing the error page, and adding separator lines. Although this presentation offers intermediate-level techniques, a short review of the Reports portlet will allow beginners to assimilate the topics.

Easy HTML DB

Michael Cunningham, *The Doctors' Company*

HTML DB is a web development environment with the power to build and deploy full-featured web applications. Key features include wizard-driven application building, graphical query building, and built-in transaction concurrency management—really! For those of you who find information on asktom.oracle.com, did you know it is built using HTML DB? This session will be primarily a demo with a discussion of features—including new features in version 2.0. The live demo will include building a non-trivial application in less than 10 minutes using the built-in wizards. Can't believe it? Come and see.

JDeveloper 10g and Oracle ADF Business Components: Getting the Most Out of Your Data

Avrom Roy-Faderman, *Quovera*

The core of any J2EE application is its business services layer—how it interacts with the database. This interaction is not as trivial as it might seem: databases and J2EE applications live in entirely different worlds—one relational, the other object-oriented; one SQL-based, the other Java-based; one where manipulating terabytes of data is standard, one where a few hundred megabytes can bring a system to its knees. Oracle JDeveloper 10g and its Application Development Framework (ADF) provide a number of options for bridging the object/relational divide, but the easiest to learn and most productive to use is ADF Business Components (ADF BC) technology. After a brief discussion of the general architecture of Oracle ADF, this presentation focuses on ADF BC. It explains the various component types, demonstrates how they fit into the framework, and shows how you can use this knowledge to maximize your application's data processing efficiency and flexibility. Finally, audience members will see a demonstration of the tools JDeveloper provides to help develop business components and integrate them into applications. This presentation is appropriate both for those who have never used ADF and those who have had some hands-on experience but would like to learn more about the Business Components portion of the framework.

Track 3:

High Availability and Disaster Recovery Techniques and Options

Alok Pareek, *GoldenGate Software*

Drawing on ten years of experience at Oracle Corporation in database development and within the recovery/high availability group, this presentation provides a way to view High Availability and Disaster Recovery needs, common approaches, and evaluation of available IT solutions. Attendees will be presented with various systematic views and availability concerns for high availability (no outage

scenarios), unplanned outages, and planned outages. They will learn how to better understand and categorize database failures. Finally, there will be discussion about differentiating and evaluating existing HA/DR technologies, including conventional backup, RAID, block-level replication, mirroring, and transactional data management, among others.

Practical Space Management in Data Warehouse Environments

Hamid Minoui, *Database Specialists, Inc.*

Managing space in a data warehouse environment is one of the challenging tasks for data warehouse DBAs. Not only are there many data files and tablespaces to manage, but also the size of individual segments and partitions tends to be very large. This presentation addresses best practices in effectively managing very large databases and data warehouse environments with the goals of efficiently using the existing disk space by reducing waste and managing data growth while enhancing query performance.

Data Vault—What's the Combination?

Jeffrey Meyer, *Denver Public Schools*

What is a data vault and why should I use it? The answer is that a data vault is a concept that has been pioneered by Dan Linstedt. In this presentation, you will learn what a data vault is and how a data vault allows data warehouses to move to the next level of scalability and ease of use. ▲

TECH TIPS

Oracle FAQ Script Exchange

“**T**hese scripts are potentially dangerous . . .” That is the warning on the Orafaq.com script exchange page. Of course, you should always review, customize, and test anything that could affect your database. Keeping this in mind, you could spend hours poking around the Oracle FAQ site looking at the scripts and tools available. Here are a few of the sections you'll find at <http://www.orafaq.com/scripts/index.htm>.

- General SQL Scripts
- General PL/SQL Scripts
- General DBA Scripts
- Object Management Scripts
- Space Management Scripts
- Security Auditing Scripts
- Database Performance Tuning Scripts
- Backup and Recovery Scripts
- Advanced Replication Scripts
- UNIX Shell Script ▲

NoCOUG Fall Conference Schedule

November 10, 2005, at the Computer History Museum

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

8–9AM	Registration and Continental Breakfast —Refreshments served
9–9:30	General Session and Welcome —Darrin Swan, NoCOUG President
9:30–10:15	Keynote: “Why: Why ‘Why?’ Is Probably the Right Answer,” Tom Kyte
10:15–10:45	Break
10:45–11:45	Parallel Sessions #1 Track 1: <i>All About Binds</i> by Tom Kyte, Oracle Corporation Track 2: <i>High Availability and Disaster Recovery Techniques and Options</i> by Alok Pareek, GoldenGate Software
11:45AM–1PM	Lunch
1–2	Parallel Sessions #2 Track 1: <i>Logical Standby Databases for Reporting</i> by Mark Bole, BIN Computing Track 2: <i>Tips and Tricks for Customizing Portal Reports Portlet Modules</i> by Peter Koletzke, Quovera Track 3: <i>Practical Space Management in Data Warehouse Environments</i> by Hamid Minoui, Database Specialists, Inc.
2–2:15	Break
2:15–3:15	Parallel Sessions #3 Track 1: <i>Transportable Tablespaces and Data Pump</i> by Caleb Small, caleb.com Track 2: <i>Easy HTML DB</i> by Michael Cunningham, The Doctors’ Company Track 3: <i>Data Vault—What’s the Combination?</i> by Jeffrey Meyer, Denver Public Schools
3:15–3:45	Raffle and Refreshments
3:45–4:45	Parallel Sessions #4 Track 1: <i>Advanced Triggers—Procedural DDL, Security, Auditing, and More</i> by Caleb Small, caleb.com Track 2: <i>JDeveloper 10g and Oracle ADF Business Components: Getting the Most Out of Your Data</i> by Avrom Roy-Faderman, Quovera
5PM–	NoCOUG Networking and Happy Hour TBD Cost: \$40 admission fee for nonmembers. Members free. Includes lunch voucher.

Session descriptions appear on page 26.

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

NoCOUG
P.O. Box 3282
Danville, CA 94526

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