Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 25, No. 3 · AUGUST 2011 $15

Data, Data, Everywhere

Expert Oracle Exadata

Kerry Osborne, et al., A book excerpt.

See page 12.

Using Services in a Single Instance

By Norbert Debes.

See page 8.

Probably Speaking

Interview with Mogens Norgaard.

See page 4.

Much more inside . . .

Data, Data, Everywhere

Vannevar Bush wrote in the *Atlantic Monthly* in 1945 that "there will always be plenty of things to compute in the detailed affairs of millions of people doing complicated things." This was years before stored-program computers had even been built, so his insight quite remarkable. He was right, of course! As of 2007, "humankind was able to store 2.9 x 10^{20} optimally compressed bytes," growing at 23% per year since 1986 (Hilbert et al. *Science*, Feb 10 2011).

It is our job as database professionals to make sure these huge volumes of data are available. Would an Exadata machine make your job easier? Read the excerpt of Kerry Osborne's new book. Do you need RAC? Read the Mogens Norgaard interview to find out. Or perhaps Norton Debes's article on Oracle Services can help you manage your growing database?

As you consider the Oracle toolset, play around with this historical perspective. When NoCOUG started twenty five years ago, we did even not have row-level locking or hot backups! How did we manage? What will the coming years bring us? Stay tuned! ▲

—*NoCOUG Journal* Editor

Table of Contents

Publication Notices and Submission Format

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

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

The submission deadline for the upcoming November 2011 issue is October 1, 2011. Article submissions should be made in Microsoft Word format via email.

NoCOUG Needs Your Help

by Iggy Fernandez

NoCOUG would not have lasted 25 years without the dedicated effort of volunteers. Won't you consider helping us soldier on for another 25 years?

- **Become a board member.** Board members attend eight board meetings, all four conferences, and take on a commitment such as Webmaster or Editor.
- **Actively help a board member.** You can actively help a board member perform his or her duties but you don't have to attend board meetings.
- **Help at conferences.** You can help with activities such as registration and room monitoring.
- **Become an annual member.** Walk-ins are welcome at conferences but annual memberships are important for budgeting.
- **Forward conference announcements.** Spread the word to colleagues who will benefit from our educational and networking opportunities.

Jonathan Lewis will deliver the keynote address at Conference #99 on Thursday, August 18 at Chevron in San Ramon. In his introduction to Jonathan Lewis' book *Cost-Based Oracle Fundamentals*, Oracle guru Tom Kyte says: *"One of the reasons [Chapter 7] is my favorite chapter is because I still remember hearing this chapter for the first time (not reading it, hearing it). It was about three years ago, at the NoCOUG (Northern California Oracle User Group) Meeting. I attended Jonathan's Histograms session and, for the first time, felt that I truly understood how histograms worked in Oracle."* If Oracle guru Tom Kyte benefited from listening to Jonathan Lewis at a NoCOUG conference, then so will you and I.

I hope to see you on August 18. ▲

2011 Board of Directors. Back row, left to right: Chen Shapira, Dave Abercrombie, Naren Nagtode. Center row, left to right: Scott Alexander, Iggy Fernandez, Omar Anwar. Front row, left to right: Jen Hong, Randy Samberg, Hanan Hit. Not pictured: Eric Hutchinson.

Probably Speaking

by Mogens Norgaard

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

Years ago you said that we probably don't need RAC. Have you recanted yet? Do we probably need RAC?

I still think very, very few shops actually need RAC. Fantastic technology—just like, say, head-up display (HUD) for cars—but few really need it. RAC still has all the hallmarks of something people will want to buy: It increases complexity immensely, it's expensive, it requires specialists that are increasingly hard to find, there are always excellent alternatives—and it's pretty much perpetually unstable. For all those good reasons, more and more customers are using it. Either because manly types like to increase chaos, or because I've been telling people not to use it since around the year 2000. Whenever I recommend or don't recommend something, most customers go out and do exactly the opposite, so in that sense I have a great deal of influence in the market.

Years ago, Mogens wrote a provocative paper titled "You Probably Don't Need RAC," and he has obviously not changed his mind since then. Here are the opening sentences of his paper:

If you've been holidaying in Siberia or similar places for about a year, you have probably not talked to an Oracle Sales rep yet about RAC. But you will no doubt find that there's a voice mail waiting for you when you turn your mobile phone on again after returning home from the vacation.

RAC is being pushed very hard by Oracle. You will get high availability, incredible scalability, a much improved personal life, the ability to partition workloads, buy cheap Linux servers and what have you.

It sounds pretty good. How can anyone say no to that kind of offer?

The closing sentences are as interesting as the opening sentences, especially the very last one.

If you have a system that needs to be up and running a few seconds after a crash, you probably need RAC.

If you cannot buy a big enough system to deliver the CPU power and or memory you crave, you probably need RAC.

If you need to cover your behind politically in your organisation, you can choose to buy clusters, Oracle, RAC and what have you, and then you can safely say: "We've bought the most expensive equipment known to man. It cannot possibly be our fault if something goes wrong or the system goes down".

Otherwise, you probably don't need RAC. Alternatives will usually be cheaper, easier to manage and quite sufficient.

Now please prove me wrong.

The paper was written in the early days of RAC. The technology has matured and improved since the date of the paper and, therefore, a number of the technical details in the paper are no longer valid. However, the underlying message of the paper is that you need to make an informed decision, justify the increased complexity and cost, and consider the alternatives.

Do we probably need Exadata? Is Big Iron the ultimate answer to the great question of life, the universe, and everything?

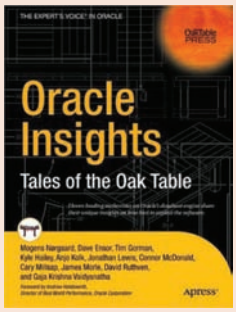
In some ways, Exadata is the new RAC. It's a lot about hardware, uptime, performance, amazing technology—and price. It's also approaching the "Peak of Inflated Expectations" as seen in Gartner's hype cycle, and it will soon set its course downwards toward the "Trough of Disillusionment." Just like with RAC, I simply *love* the technology—a lot of good guys that I like and respect are on it, but few really need it. One of the things I love about it is that there isn't any SAN involved, since I believe SANs are responsible for a lot of the instability we see in IT systems today. I tend to think about Exadata as a big mainframe that could potentially do away with hundreds of smaller servers and disk systems, which appeals hugely to me. On the other hand, the pricing and complexity makes it something akin to RAC—that's my current thinking.

Do we probably need Oracle Database 12c (or whatever the next version of Oracle Database will be named)?

Since Oracle 7.3, that fantastic database has had pretty much everything normal customers need. It has become more and more fantastic; it has amazing features that are light years ahead of competitors—and fewer and fewer are using the database as it should be used (they're using it as a data dump, as Tom Kyte said many years ago), so the irony is that as the database approaches a state of nirvana (stability, scalability, predictability, diagnosability, and so forth—fewer and fewer are using it as it should be used (in my view), and more and more are just dumping data into it and fetching it.

Do we probably need MySQL? Or do we get what we pay for?

(continued on page 6)



Oracle Performance Management

A 2-Day Seminar
With Gaja Krishna Vaidyanatha
November 15–16, 2011

Gaja Krishna Vaidyanatha is a regular at NoCOUG; he's a frequent contributor to our Journal and a frequent speaker at our conferences. He's a member of the Oak Table network of Oracle Database experts and invented the term Compulsive Tuning Disorder for the disease that afflicts so many of us. Gaja will return to NoCOUG in November to deliver his popular Oracle Performance Management seminar updated for Oracle Database 10*g* and Oracle Database 11*g*.

Oracle Performance Management has a reputation as part science, part art, and mostly wizardry. In Gaja's seminar you will learn the core principles of performance management and a systematic methodology for investigating and solving performance problems. The seminar covers all releases of Oracle Database—from Version 7 to 11*g* Release 2—but focuses mostly on Oracle Database 10*g* and above.

- *Oracle Database Architecture*
- *What is Performance Optimization?*
- *Oracle Database Diagnostics*
- *Oracle Wait Interface*
- *Cost-Based Optimizer*
- *Oracle 11g Performance Management*
- *Multiversion Read Consistency*
- *When Should You Optimize?*
- *Operating System Diagnostics*
- *Low-level Trace Methods*
- *Oracle 10g Performance Management*
- *Real-World Case Studies*

Gaja has two decades of Oracle Database experience and is a co-author of Oracle Performance Tuning 101 (Oracle Press) and Oracle Insights: Tales of the Oak Table (Apress). He is a popular presenter at regional, national and international Oracle Database conferences. Gaja is a fierce opponent of Compulsive Tuning Disorder (CTD) in all its forms. He knows it when he sees it and he'll teach you to recognize it and avoid it.

Early-Bird Pricing (Until September 30)

$700 for members

$1,050 for non-members

Regular Pricing

$800 for members

$1,200 for non-members

Register at **www.nocoug.org**. Seating is limited, so register early.

(continued from page 5)

As customers (and especially new, freshly faced programmers) want to use new things instead of things that work and perform, it becomes more and more logical to use MySQL or other databases instead of the best one of them all: Oracle. Since MySQL succeeded in becoming popular among students and their professors, it is immensely popular among them when they leave school (the professors stay, of course, since they don't know enough about databases to actually get a real job working with them outside academia). So MySQL will be used a lot. And it's an OK database, especially if we're talking the InnoDB engine.

Do we probably need certification? Or do we learn best by making terrible mistakes on expensive production systems?

I hate certifications. They prove nothing, and they become a very bad replacement for real education, training, and knowledge. Among Windows and Cisco folks, it's immensely popular, but you can now feed all the farm animals in Denmark (and we've got quite a few, especially a lot of pigs) with certified Microsoft and Cisco people. It's taken by students (what?!? instead of real education, they train them in something that concrete? I find it really stupid), among unemployed (we have a lot of programs for those folks here), and what have you. They're worthless, and a lot of people think it will help them finding a job, thereby providing false hopes and security. YPDNC.

Do we probably need ITIL? Should we resist those who try to control and hinder us?

When you begin doing "best practices" stuff like ITIL, you've lost. You're pouring cement down the org chart in your shop, and God bless you for that—it helps the rest of us compete. "Best practices" means copying and imitating others that have shops that are unlike yours. Standardizing and automating activity in brain-based shops always seemed strange to me. The results—surprise!—are totally predictable: jobs become immensely boring, response times become horrible, queues are everywhere, and nothing new can happen unless a boss high up *really* demands it. It's Eastern Europe—now with computers. Oh, and it's hype; it's modern right now but will be replaced by the next silly thing (like LEAN—what a fantastically stupid idea, too). Maybe we'll have LEAN ITIL one day? Or Balanced Score Card–adjusted ITIL? Or Total Quality Management of LEAN ITIL?

The funny thing is that Taylor's ideas (called "scientific management") were *never* proved, and he was actually fired from Bethlehem Steel after his idiotic idea of having a Very Big Hungarian lift 16 tons in one day (hence all the songs about 16 tons), because he cheated with the results and didn't get anything done that worked. Not one piece of his work has ever been proved to actually work. His "opponent" was Mayo (around the 1920s), with his experiments into altering the work environment (hence the constant org changes and office redos that everybody thinks must be good for something)—and his work has never been proved either. And he cheated too, by the way, which he later had to admit. So all this management stuff is bollocks, and ITIL is one of its latest fads. I say: Out with it. Let's have our lives and dignities back, please.

NoCOUG membership and attendance has been declining for years. Do we probably need NoCOUG anymore? We'll celebrate our 25th anniversary in November. Should we have a big party and close up the shop? Or should we keep marching for another 25 years?

No. Oracle User Groups are dead as such. Just like user groups for mainframe operators or typesetters. You can make the downward sloping angle less steep by doing all sorts of things, but it's the same with all Oracle user groups around the world. I think I have a "technical fix" or at least something crazy and funny that can prolong NoCOUG's life artificially: move onto the Net aggressively and do it with video everywhere. Let it be possible to leave video responses to technical questions (why doesn't Facebook have that?); let it be possible to upload video or audio or text replies to debates and other things via a smartphone app. Let there be places where the members can drink different beers at the same time and chat about it (and show the beer on the screen), etc., etc. In other words: Abandon the real world before all the other user groups do it—and perhaps that way you can swallow the other user groups around you and gradually have World Dominance.

It costs a fortune to produce and print the *NoCOUG Journal*. Do we probably need the *NoCOUG Journal* anymore?

I have subscribed to the world's arguably best magazine, *The Economist*, since 1983. Recently they came out with an app, and now I don't open the printed edition any more (I still receive it for some reason). It's so much cooler to have the magazine with me everywhere I go, and I can sit in the bathroom and get half of the articles in there read. It's the way. Magazines should not be available anymore in print. Nor should they (in my view) be available on a silly website that people have to go to using a PC, a browser, and all sorts of other old-days technology. The smartphone *is* the computer now. Move the magazine there aggressively, and in the process, why not create a template that other user groups could take advantage of? Or the Mother of All Usergroup Apps (MOAUA) that will allow one user group after another to plug in, so people can read all the good stuff all over the world?

I'm writing a book on physical database design techniques like indexing, clustering, partitioning, and materialization. Do we probably need YABB (Yet Another Big Book)?

No, certainly not. Drop the project immediately, unless you can use it as an excuse to get away from the family now and then. Or, if you *must* get all this knowledge you have out of your system, make an app that people can have on their phone and actually USE in real-life situations. Abandon books immediately, especially the physical ones.

Interview conducted by Iggy Fernandez

Using Services in a Single Instance Environment

by Norbert Debes

In this article, I will address some of the lesser-known and even undocumented aspects of services. Services are often mentioned in the context of Real Application Clusters (RAC) and Grid Computing. For example a service may be provided by one or more RAC instances. Additional RAC instances may be started to support higher demand for a service, thus offering more computing power in the form of additional CPUs and more I/O bandwidth. In this article I will point out that services are just as useful in a single-instance environment.

What's in a Service?

The term "service" or "service name" is used in varying contexts throughout Oracle documentation and error messages. Many but not all references to service name in the documentation try to disambiguate the term by referring to a net service name whenever a connect identifier defined by a naming service such as local naming (tnsnames.ora) is mentioned. Unfortunately, service and service name are not used consistently throughout Oracle documentation. Here's an example from the glossary of Oracle Database Net Services Reference 10*g* Release 2 (10.2) B14213-01:

> **local naming**
> A naming method that locates network addresses by using information configured and stored on each individual client's tnsnames.ora file. Local naming is most appropriate for simple distributed networks with a small number of services that change infrequently.

This excerpt implies that services are defined in the file tnsnames.ora. However this is not the case. What tnsnames.ora contains is a list of connect identifiers that may reference a service offered by a certain DBMS instance using the directive (SERVICE_NAME=<service_name>).

Instance Service Name vs. Net Service Name

To disambiguate the term "service name," one might use the denomination Net (from Oracle Net) service name for items defined in tnsnames.ora or a directory service such as LDAP. However the term "connect identifier" is the most unambiguous and is also used in the error message ORA-12154 provoked below:

```
$ sqlplus -l -s x/y@unknown.service
ERROR:
ORA-12154: TNS:could not resolve the connect identifier specified
SP2-0751: Unable to connect to Oracle.  Exiting SQL*Plus
```

By the way, if this error message takes more than a fraction of a second to appear, it is because the default setting of NAMES.DIRECTORY_PATH is (tnsnames, ezconnect, ldap) or a custom setting that contains more than just tnsnames. If you use NAMES.DIRECTORY_PATH=(tnsnames), the error is thrown immediately, but all other name resolution methods, including easy connect naming, are disabled.

I call the service names a DBMS instance registers with a listener "instance service names". "Instance service names" are defined either through the parameter SERVICE_NAMES or with the package DBMS_SERVICE[1] in Oracle10*g* and Oracle11*g*. The command tnsping accepts connect identifiers (or Net service names), whereas the list of services returned by the command lsnrctl services contains instance service names. Connect strings, such as ndebes/secret@ten.oradbpro.com, contain connect identifiers. The body of a connect identifier's definition includes either an instance service name or an ORACLE_SID (SID=<ORACLE_SID>[2]). A connect identifier's definition in tnsnames.ora has the following format:

```
<connect_identifier> =
(DESCRIPTION =
  (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(HOST = <host_name>)(PORT =
<port_number>))
  )
  (CONNECT_DATA =
     (SERVICE_NAME = <instance_service_name>)
  )
  )
```

Mind the keyword SERVICE_NAME in the body of the DESCRIPTION section. The setting of SERVICE_NAME is an instance service name, and in Oracle10*g* and later versions it is reflected in the column V$SESSION.SERVICE_NAME and in SQL trace files. In Oracle10*g*, all configured instance service names are in DBA_SERVICES.NETWORK_NAME. Why NETWORK_NAME? These are the instance service names registered with an Oracle Net Listener or Connection Manager. Client sessions that connect by using a connect identifier that contains SID=<ORACLE_SID> instead of SERVICE_NAME=<instance_service_name> have the service name SYS$USERS in V$SESSION.SERVICE_NAME. This is also true for local sessions established without specifying a connect identifier. These latter sessions use the so-called "bequeath protocol adapter," which takes the setting of SID from the en-

[1] The source code file for DBMS_SERVICE is $ORACLE_HOME/rdbms/admin/dbmssrv.sql.

[2] Angle brackets surround placeholders that require an actual value.

vironment variable ORACLE_SID.

SYS$BACKGROUND (used by background processes of the DBMS instance such as PMON and SMON) and SYS$USERS are both built-in instance service names that cannot be removed or stopped and are not registered with a listener. Even if the parameter SERVICE_NAMES is not set, a service called <db_name>.<db_domain>[3] is registered with all listeners or Connection Manager (CMAN) instances that can be contacted using the parameter values of LOCAL_LISTENER and REMOTE_LISTENER.

In case a TNS listener or CMAN instance is (re)started after the DBMS instance that registers with it, there is a certain delay until the PMON process contacts the listener or CMAN to register instance service names. This delay is eliminated by the following SQL statement: SQL> ALTER SYSTEM REGISTER;

Benefits of Using Instance Service Names

An instance service may be started and stopped at the DBA's discretion. I recommend that each application that connects to a certain DBMS instance do so using a separate instance service name. Correspondingly, each application must use a separate connect identifier if local naming or LDAP naming are used.

Performance Statistics by Instance Service Name

There is a dynamic performance view called V$SERVICE_STATS that reports CPU usage (STAT_NAME='DB CPU') and 27 other statistics, such as physical reads and redo size, for each service. By using separate service names for each application, it is very easy to determine which application uses the most resources. The following query looks at CPU statistics:

```
select s.service_name, s.stat_name, round(s.value/1000000/(sysdate -
i.startup_time)) AS cpu_secs_per_day
from v$service_stats s, v$instance i
where s.stat_name='DB CPU'

order by cpu_secs_per_day desc;
```

Additional performance statistics by instance service name and module may be enabled using DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE.

Killing Sessions

Being able to disconnect all sessions that use a certain instance service name is a useful feature. Let's say that a DBA needs to upgrade a schema that is used by an application that has several open sessions. Instead of killing each session individually using ALTER SYSTEM KILL SESSION, a DBA can disconnect all the sessions that use a certain instance service name through the packaged procedure DBMS_SERVICE.DISCONNECT_SESSION. Here's an example:

```
SQL> SELECT sid, serial#, service_name FROM v$session WHERE
service_name='ELEVEN.oradbpro.com';

       SID    SERIAL#   SERVICE_NAME
----------  ----------  --------------------
        20           4   ELEVEN.oradbpro.com
        21           4   ELEVEN.oradbpro.com
        22           5   ELEVEN.oradbpro.com
        23          23   ELEVEN.oradbpro.com
        24           6   ELEVEN.oradbpro.com
        26           5   ELEVEN.oradbpro.com
        28         202   ELEVEN.oradbpro.com
        30          14   ELEVEN.oradbpro.com
        34           9   ELEVEN.oradbpro.com
        60          48   ELEVEN.oradbpro.com

10 rows selected.
```

Ten sessions have connected using the instance service name ELEVEN.oradbpro.com. All such sessions can be "killed" at once using DBMS_SERVICE.DISCONNECT_SESSION.

```
SQL> exec DBMS_SERVICE.DISCONNECT_SESSION('ELEVEN.
oradbpro.com', dbms_service.immediate)

PL/SQL procedure successfully completed.

SQL> SELECT sid, serial#, service_name FROM v$session WHERE
service_name='ELEVEN.oradbpro.com';

no rows selected
```

Note that the approach shown above works without explicitly starting the affected service using DBMS_SERVICE.START_SERVICE. In other words, you can use DISCONNECT_SESSION even if the services were started based on the value of the parameter SERVICE_NAMES.

At this point nothing keeps the application from connecting again. If the DBA were to enable restricted session, then all connections through all listeners would be disabled and would fail with the error ORA-12526: TNS:listener: all appropriate instances are in restricted mode. Using DBMS_SERVICE.STOP_SERVICE stops a specific service and deregisters it from all listeners. This is a good way to keep an application from connecting while maintenance is being performed.

By the way, the error ORA-12526 can be avoided by adding the undocumented directive (UR=A) to the CONNECT_DATA section of a connect identifier's description. (UR=A) was added in Oracle 10*g* to support connections to an ASM instance (INSTANCE_TYPE=ASM) through a listener. However, it works with RDBMS instances (INSTANCE_TYPE=RDBMS) too. Here's an example:

```
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
System altered.
SQL> SELECT logins FROM v$instance;

LOGINS
----------
RESTRICTED

SQL> connect ndebes/secret@eleven2
ERROR:
ORA-12526: TNS:listener: all appropriate instances are in restricted mode
```

After adding (UR=A) as explained above, the connection goes through:

```
eleven2=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=TCP)(HOST=dbserver)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=ELEVEN2)
    (UR=A) # ignore ORA-12526: TNS:listener: all appropriate instances are
in restricted mode
    )
  )
```

Default Service Names

When a database is created, a service called <db_name>.<db_domain> is automatically created. This service

[3] If db_domain is not set, then the name of the service will be just <db_name> without a domain suffix.

cannot be stopped or removed. It should not be used for applications, but it may be used by DBAs to connect.

You may see two other instance service names when retrieving the list of available services using lsnrctl services:

1. A service called <db_name>_XPT.<db_domain>. This service is related to Data Guard Broker in Oracle Enterprise Edition.
2. A service called <db_name>XDB.<db_domain>. This service is related to XML Database (XDB).

Most environments do not require these instance services. To remove the first service, set the undocumented parameter "__dg_broker_service_names" as shown below.

```
ALTER SYSTEM SET "__dg_broker_service_names" = '' SCOPE=SPFILE;
```

The service <db_name>_XPT.<db_domain> will no longer appear after an instance restart. To eliminate the second service, first remove the dispatcher that references it and then delete the service as follows:

```
SQL> show parameter dispatchers
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
dispatchers                          string      (PROTOCOL=TCP)
(SERVICE=ELEVEN
                                                 2XDB)
SQL> alter system set dispatchers='';
System altered.
SQL> exec dbms_service.delete_service('&svc');
Enter value for svc: ELEVEN2XDB

PL/SQL procedure successfully completed.
```

Adding an Instance Service Name

There are two ways to add an instance service name:

1. Add an identifier to the parameter SERVICE_NAMES.
2. Use the packaged procedure DBMS_SERVICE.CREATE_SERVICE.

Whenever you add an identifier to the parameter SERVICE_NAMES, a row is inserted into the data dictionary base table SERVICE$. You'll see the additional row as you query the data dictionary view DBA_SERVICES. If you later remove the new identifier from the parameter SERVICE_NAMES, the additional row in SERVICE$ is not removed. Hence you may end up with a few obsolete instance service names in SERVICE$ and thus DBA_SERVICES. Since instance services in SERVICE$ alone—without a corresponding service name in the setting of the parameter SERVICE_NAMES—are not started automatically, obsolete service names may go unnoticed.

ALTER SYSTEM

Here's an example of adding an instance service using ALTER SYSTEM:

```
SQL> ALTER SYSTEM SET service_names='webapp.oradbpro.com';
System altered.
```

The preceding command not only creates the new service but also starts it. The following query against V$ACTIVE_SERVICES proves it:

```
SQL> SELECT name, network_name, clb_goal
FROM v$active_services
WHERE name='webapp.oradbpro.com';

NAME                 NETWORK_NAME         CLB_GOAL
-------------------- -------------------- --------
webapp.oradbpro.com  webapp.oradbpro.com  LONG
```

To remove obsolete instance service names, use DBMS_SERVICE.DELETE_SERVICE.

Contrary to what you might expect, service names and network names are case sensitive in SERVICE$. If you intend to trace sessions using DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE, then the instance service name is case sensitive too. Capitalization of instance service names in tnsnames.ora is irrelevant.

DBMS_SERVICE

Here's how you can create a service using DBMS_SERVICE:

```
SQL> begin
  2     dbms_service.create_service(
  3            service_name=>'&service_name',
  4            network_name=>'&network_name'
  5     );
  6  end;
  7  /
Enter value for service_name: WEBAPP_DD
old   3:            service_name=>'&service_name',
new   3:             service_name=>'WEBAPP_DD',
Enter value for network_name: webapp
old   4:            network_name=>'&network_name'
new   4:             network_name=>'webapp'
PL/SQL procedure successfully completed.
```

The new service appears like this in DBA_SERVICES:

```
SQL> SELECT name, network_name, clb_goal FROM dba_services
WHERE name='WEBAPP_DD';

NAME                 NETWORK_NAME         CLB_GOAL
-------------------- -------------------- --------
WEBAPP_DD            webapp               LONG
```

The new service needs to be started as follows to get registered with a listener:

```
SQL> exec dbms_service.start_service('WEBAPP_DD')

PL/SQL procedure successfully completed.
```

I did not specify a domain suffix with the network name to show how the value of the parameter DB_DOMAIN is appended automatically when the new service is registered with a listener. The following excerpt is from lsnrctl services:

```
Services Summary...
Service "webapp.oradbpro.com" has 1 instance(s).
  Instance "eleven", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
```

This output shows that the NETWORK_NAME was registered with a listener, whereas the NAME WEBAPP_DD is sort of an alias that identifies the service when using DBMS_SERVICE.

A Concept for Using Instance Service Names

Using a single database for many applications saves a huge amount of time for setup and administration compared to using a separate database for each and every application. The downside is that one application may monopolize resources. However, the DBA can take various precautions. The maximum number of concurrent sessions can be limited by using profiles (SESSIONS_PER_USER) and setting RESOURCE_LIMIT=TRUE. The parameters PROCESSES and SESSIONS need to be set high enough to accommodate the maximum session count of all applications combined. CPU time can be allotted and the degree of parallelism constrained using the database resource manager.

When a considerable number of services are provided by a DBMS instance, editing the parameter SERVICE_NAMES becomes cumbersome and a bit error prone. If a DBA would accidentally omit or misspell a service name while changing the parameter SERVICE_NAMES, then unplanned downtime would result. Using DBMS_SERVICE is safe in this regard, but without RAC there is no component such as Cluster Ready Services that might start all the services when a database is opened. Remember that services created using DBMS_SERVICE are not started automatically at instance start-up.

Using a database event trigger for the event AFTER STARTUP is the ideal solution for this issue. An AFTER STARTUP trigger fires when a DBMS instance executes ALTER DATABASE OPEN to open a database. Such a trigger can then iterate over all the services in DBA_SERVICES and start each one of them. Thus no special actions need to be performed when a DBMS instance is restarted. Services are started automatically through the database event trigger in the same way as if the parameter SERVICE_NAMES had been permanently set. In fact DBMS_SERVICE calls ALTER SYSTEM SET service_names internally with SCOPE=MEMORY.

Some maintenance procedures require restarting the DBMS instance. If we wanted to make sure that applications won't be able to connect after an instance restart, all we need to do is to disable the database event trigger that starts the application-specific services.

By the way, I avoid giving the ORACLE_SID to end users and developers alike, since I want them to connect using an application-specific instance service name. Thus I can make better use of dynamic performance views, packages, and other features that use instance service names such as V$SERVICE_STATS, V$SERV_MOD_ACT_STATS, DBMS_MONITOR and SQL trace. There are still a few application software components that only work with an ORACLE_SID, but these are rare. A few applications assemble a host name, port number, and ORACLE_SID into a JDBC URL of the deprecated format jdbc:oracle:thin:@<host>:<port>:<ORACLE_SID>. A session opened with a JDBC URL of this format gets the catchall service name SYS$USERS.

Package CDBMS_SERVICE

The package CDBMS_SERVICE („c" for custom) uses DBMS_SERVICE internally. It consists of two procedures that start or stop all services in DBA_SERVICES. The package specification is reproduced here:

```
create or replace package site_sys.cdbms_service is
  procedure start_all_services;
  procedure stop_all_services;
end;
```

The procedure START_ALL_SERVICES is used by a trigger discussed in the next section. STOP_ALL_SERVICES is useful for stopping all services during a maintenance window.

AFTER STARTUP Trigger

As mentioned earlier, there must be some means to automatically start all services in DBA_SERVICES when the value of the parameter SERVICE_NAMES is an empty string. An AFTER STARTUP trigger conveniently implements this functionality. Since the trigger merely calls the package CDBMS_SERVICE, the code is very simple:

```
create or replace trigger sys.start_services after startup on database
begin
  site_sys.cdbms_service.start_all_services;
end;
```

A DBMS instance that is started using STARTUP UPGRADE does not invoke AFTER STARTUP triggers. Thus all application-specific instance services are unavailable after STARTUP UPGRADE. Only the default service called <db_name>.<db_domain> is available with status RESTRICTED, since STARTUP UPGRADE enables restricted mode (V$INSTANCE.LOGINS='RESTRICTED').

Summary

In this article, several benefits of using instance service names were discussed. Separate instance service names for each application are useful, since they ease performance monitoring, tuning, and maintenance. Using the package DBMS_SERVICE instead of setting the parameter SERVICE_NAMES directly is more convenient, since services are added, started, stopped, and removed using PL/SQL instead of editing the string assigned to the parameter SERVICE_NAMES. To compensate for the lack of a component that starts all services in a single instance environment, an AFTER STARTUP trigger is created.

Since Transparent Application Failover (TAF) can be used in a single instance environment and DBMS_SERVICE can enable TAF on the server-side, a reconnect capability might be implemented based on a server-side TAF configuration.[4] However, this is beyond the scope of the article and would require support for TAF by applications.

A SQL*Plus script called cdbms_service.sql contains all the PL/SQL code pertaining to this article and numerous examples of using DBMS_SERVICE. It is available for download at **http://www.nocoug.org/download/2011-08/cdbms_service.sql.txt**. ▲

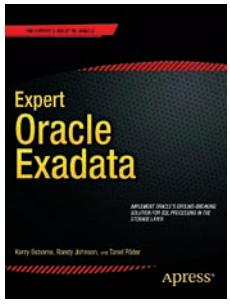
Norbert Debes also wrote "Secret ORACLE - Unleashing the Full Potential of the ORACLE DBMS by Leveraging Undocumented Features," *reviewed here in November 2008, which was subsequently updated and published by Apress as* "Oracle Database Secrets." *He is focused on ORACLE DBMS performance, high availability, database security, backup/recovery and training.*

[4] The parameters with prefix FAILOVER are provided to enable TAF on the server-side using DBMS_SERVICE.CREATE_SERVICE or MODIFY_SERVICE.

Expert Oracle Exadata

by Kerry Osborne, Randy Johnson and Tanel Põder

Details



Authors: Kerry Osborne, Randy Johnsonm, Tanel Põder

ISBN: 978-1-4302-3392-3

Pages: 588

Publication Date: August 1, 2011

Price: $60

Publisher: Apress

Editor's note: Instead of the book review that typically fills these pages, we present this excerpt of this important new book on Oracle Exadata. We have included most of the second chapter. This book is scheduled for publication before this issue of the *Journal* goes to press. We did not have space to print the book's code listings; please see the published book for these listing and the rest of the chapter.

Offloading

Offloading is the secret sauce of Exadata. It's what makes Exadata different from every other platform that Oracle runs on. Offloading refers to the concept of moving processing from the database servers to the storage layer. It is also the key paradigm shift provided by the Exadata platform. But it's more than just moving work in terms of CPU usage. The primary benefit of Offloading is the reduction in the volume of data that must be returned to the database server. This is one of the major bottlenecks of most large databases.

The terms Offloading and Smart Scan are used somewhat interchangeably. Offloading is a better description in our opinion, as it refers to the fact that part of the traditional SQL processing done by the database can be "offloaded" from the database layer to the storage layer. It is a rather generic term, though, and is used to refer to many optimizations that are not even related to SQL processing including improving backup and restore operations.

Smart Scan, on the other hand, is a more focused term, in that it refers only to Exadata's optimization of SQL statements. These optimizations come into play for scan operations (typically Full Table Scans). A more specific definition of a Smart Scan would be any section of the Oracle kernel code that is covered by the Smart Scan wait events. There are actually two wait events that include the term "Smart Scan" in their names, Cell Smart Table Scan and Cell Smart Index Scan. We'll discuss both of these wait events in detail a bit later, in Chapter 10. While it's true that "Smart Scan" has a bit of a marketing flavor, it does have specific context when referring to the code covered by these wait events. At any rate, while the terms are somewhat interchangeable, keep in mind that Offloading can refer to more than just speeding up SQL statement execution.

In this chapter we will focus on Smart Scan optimizations. We'll cover the various optimizations that can come into play with Smart Scans, the mechanics of how they work, and the requirements that must be met for Smart Scans to occur. We'll also cover some techniques that can be used to help you determine whether Smart Scans have occurred for a given SQL statement. The other offloading optimizations will only be mentioned briefly as they are covered elsewhere in the book.

Why Offloading Is Important

We can't emphasize enough how important this concept is. The idea of moving database processing to the storage tier is a giant leap forward. The concept has been around for some time. In fact, rumor has it that Oracle approached at least one of the large SAN manufacturers several years ago with the idea. The manufacturer was apparently not interested at the time and Oracle decided to pursue the idea on its own. Oracle subsequently partnered with HP to build the original Exadata V1, which incorporated the Offloading concept. Fast-forward a couple of years, and you have Oracle's acquisition of Sun Microsystems. This put the company in a position to offer an integrated stack of hardware and software and gives it complete control over which features to incorporate into the product.

Offloading is important because one of the major bottlenecks on large databases is the time it takes to transfer the large volumes of data necessary to satisfy DW-type queries between the disk systems and the database servers (that is, because of bandwidth). This is partly a hardware architecture issue, but the bigger issue is the sheer volume of data that is moved by traditional Oracle databases. The Oracle database is very fast and very clever about how it processes data, but for queries that access a large amount of data, getting the data to the database can still take a long time. So as any good performance analyst would do, Oracle focused on reducing the time spent on the thing that accounted for the majority of the elapsed time. During the analysis, the team realized that every query that required disk access was very inefficient in terms of how much data had to be returned to and processed by the database servers. Oracle has made a living by developing the best cache-management software available, but for really large data sets, it is just not practical to keep everything in memory on the database servers.

Imagine the fastest query you can think of: a single column from a single row from a single table where you actually know

where the row is stored (rowid). On a traditional Oracle database, at least one block of data has to be read into memory (typically 8K) to get the one column. Let's assume your table stores an average of 50 rows per block. You've just transferred 49 extra rows to the database server that are simply overhead for this query. Multiply that by a billion and you start to get an idea of the magnitude of the problem in a large data warehouse. Eliminating the time spent on transferring completely unnecessary data between the storage and the database tier is the main problem that Exadata was designed to solve.

Offloading is the approach that was used to solve the problem of excessive time spent moving irrelevant data between the tiers. Offloading has three design goals, although the primary goal far outweighs the others in importance:

- Reduce the volume of data transferred from disk systems to the database servers.
- Reduce CPU usage on database servers.
- Reduce disk access times at the storage layer.

Reducing the volume was the main focus and primary goal. The majority of the optimizations introduced by Offloading contribute to this goal. Reducing CPU load is important as well, but is not the primary benefit provided by Exadata and therefore takes a back seat to reducing the volume of data transferred. (As you'll see, however, decompression is a notable exception to that generalization, as it is performed on the storage servers.) Several optimizations to reduce disk access time were also introduced, and while some of the results can be quite stunning, we don't consider them to be the bread-and-butter optimizations of Exadata.

Exadata is an integrated hardware/software product that depends on both components to provide substantial performance improvement over non-Exadata platforms. However, the performance benefits of the software component dwarf the benefits provided by the hardware. Here is an example:

```
SYS@SANDBOX> alter session set cell_offload_processing=false;
Session altered.
Elapsed: 00:00:00.06

SYS@SANDBOX> select count(*) from kso.skew3 where col1 < 0;

  COUNT(*)
----------
         2

1 row selected.
Elapsed: 00:00:51.09

SYS@SANDBOX> alter session set cell_offload_processing=true;
Session altered.
Elapsed: 00:00:00.07

SYS@SANDBOX> select count(*) from kso.skew3 where col1 < 0;

  COUNT(*)
----------
         2

1 row selected.
Elapsed: 00:00:00.15
```

This example shows the performance of a scan against a single table with 384 million rows. We ran it once with Offloading disabled, effectively using all the hardware benefits of Exadata and none of the software benefits. You'll notice that even on Exadata hardware, this query took almost a minute. Keep in mind that this was only spread across three storage servers on our V2 quarter rack and did not utilize the flash cache at all. We then re-enabled Offloading, and the query completed in substantially less than a second. Obviously the hardware in play was the same in both executions. The point is that it's the software's ability via Offloading that made the difference.

What Offloading Includes

There are many optimizations that can be lumped under the Offloading banner. This chapter focuses on SQL statement optimizations that are implemented via Smart Scans. The big three Smart Scan optimizations are Column Projection, Predicate Filtering, and Storage Indexes. The primary goal of most of the Smart Scan optimizations is to reduce the amount of data that needs to be transmitted back to the database servers during scan execution. However, some of the optimizations also attempt to offload CPU-intensive operations, decompression for example. We won't have much to say about optimizations that are not related to SQL statement processing in this chapter, such as Smart File Creation and RMAN-related optimizations. Those topics will be covered in more detail elsewhere in the book.

Column Projection

The term *Column Projection* refers to Exadata's ability to limit the volume of data transferred between the storage tier

and the database tier by only returning columns of interest (that is, those in the select list or necessary for join operations on the database tier). If your query requests five columns from a 100-column table, Exadata can eliminate most of the data that would be returned to the database servers by non-Exadata storage. This feature is a much bigger deal than you might expect and it can have a very significant impact on response times. Here is an example:

[see book for example]

This example deserves a little discussion. First we used a trick to force direct path reads with the _SERIAL_DIRECT_READ parameter (more on that later). Next we disabled Smart Scans by setting CELL_OFFLOAD_PROCESSING to FALSE. You can see that our test query doesn't have a WHERE clause. This means that Predicate Filtering and Storage Indexes cannot be used to cut down the volume of data that must be transferred from the storage tier, because those two optimizations can only be done when there is a WHERE clause (we'll discuss those optimizations shortly). That leaves Column Projection as the only optimization in play. Are you surprised that Column Projection alone could cut a query's response time in half? We were, the first time we saw it, but it makes sense if you think about it. You should be aware that columns in the select list are not the only columns that must be returned to the database server. This is a very common misconception. Join columns in the WHERE clause must also be returned. As a matter of fact, in early versions of Exadata, the Column Projection feature was not as effective as it could have been and actually returned all the columns included in the WHERE clause, which in many cases included some unnecessary columns.

The DBMS_XPLAN package can display information about column projection, although by default it does not. The projection data is stored in the PROJECTION column in the V$SQL_PLAN view as well. Here is an example:

[see book for example]

So as you can see, the plan output shows the projection information, but only if you use the +PROJECTION argument in the call to the DBMS_XPLAN package. Note also that the PK_COL columns from both tables were listed in the PROJECTION section, but that not all columns in the WHERE clause are included. Only those columns that need to be returned to the database (join columns) should be listed. Note also that the projection information is not unique to Exadata but is a generic part of the database code.

The V$SQL family of views contain columns that define the volume of data that may be saved by Offloading (IO_CELL_OFFLOAD_ELIGIBLE_BYTES) and the volume of data that was actually returned by the storage servers (IO_INTERCONNECT_BYTES). Note that these columns are cumulative for all the executions of the statement. We'll be using these two columns throughout the book because they are key indicators of offload processing. Here's a quick demonstration to show that projection does affect the amount of data returned to the database servers and that selecting fewer columns results in less data transferred:

[see book for example code]

Note that the extra column resulted in a great deal of extra time required to complete the query and that the columns in V$SQL verified the increased volume of data that had to be transferred. We've also shown the output of a modified version of the fsx.sql script, which we'll discuss in more detail later in this chapter. For now, please just accept that it shows us whether a statement was offloaded or not.

Predicate Filtering

The second of the big three Smart Scan optimizations is *Predicate Filtering*. This term refers to Exadata's ability to return only rows of interest to the database tier. Since iDB includes the predicate information in its requests, this is accomplished by performing the standard filtering operations at the storage cells before returning the data. On databases using non-Exadata storage, filtering is done on the database servers. This generally means that a large number of records that will eventually be discarded will be returned to the database tier. Filtering these rows at the storage layer can provide a very significant decrease in the volume of data that must be transferred to the database tier. While this optimization also results in some savings in CPU usage on the database servers, the biggest advantage is generally the reduction in data transfer. [see book for example SQL]

First we completely disabled Offloading using the CELL_OFFLOAD_PROCESSING parameter and ran a query without a WHERE clause. Without the benefit of Offloading this query took about 48 seconds. We then enabled Offloading and re-ran the query. This time the query took only about 27 seconds. The savings of approximately 21 seconds was due strictly to Column Projection (because without a WHERE clause for filtering, there were no other optimizations that could come into play). We then used a trick to disable storage indexes by setting the hidden parameter, _KCFIS_STORAGEIDX_DISABLED, to TRUE (we'll discuss that more in the next section) and added a WHERE clause, which reduced the execution time to about 9 seconds. This reduction of an additional 18 seconds or so was thanks to Predicate Filtering. Note that we had to disable storage indexes to be sure that we weren't getting any benefit from that optimization and that all the improvement was due to Predicate Filtering, which brings us to the next topic.

Storage Indexes

Storage Indexes provide the third level of optimization for Smart Scans. Storage Indexes are in-memory structures on the storage cells that maintain a maximum and minimum value for each 1MB disk storage unit, for up to eight columns of a table. Storage Indexes are a little different than most Smart Scan optimizations. The goal of Storage Indexes is not to reduce the amount of data being transferred back to the database tier. In fact, whether they are used on a given query or not, the amount of data returned to the database tier remains constant. On the contrary, Storage Indexes are designed to eliminate time spent reading data from disk on the storage servers themselves. Think of this feature as a pre-filter. Since Smart

(continued on page 16)

Oracle Database Administration, Development, and Performance Tuning... *Only Faster.*

Taking care of your company's data
is an important job.
Making it easier and faster is our's.

DB PowerStudio™

Introducing DB PowerStudio for Oracle. It provides proven, highly-visual tools that save time and reduce errors by simplifying and automating many of the complex things you need to do to take care of your data, and your customers.

Whether you already use Oracle Enterprise Manager (OEM) or some other third-party tool, you'll find you can do many things faster with DB PowerStudio for Oracle.

> **Easier administration** with DBArtisan®, the industry's most respected database administration tool with graphical editors and wizards to streamline routine tasks and reduce errors. In addition, database analytics identify performance, capacity, and storage management issues before they become problems

> **Faster performance** with DB Optimizer™, an agent-less graphical profiling tool that quickly pinpoints performance issues and provides a Visual SQL Tuner so you can quickly understand and optimize poor-performing SQL

> **Faster development** with Rapid SQL™, a full-featured IDE that streamlines SQL and PL/SQL scripting, query building, object management, and version control

> **Simplified change management** with DB Change Manager™, a tool for coordinating database changes across the development lifecycle, enforce database configuration standards, and mask test data to comply with privacy mandates

Best of all, DB PowerStudio tools work across all versions of Oracle from a single interface. And with Embarcadero's enterprise-friendly software licensing and delivery technologies, it's easier to access, track, and manage tools licenses to minimize costs and maximize productivity.

Don't forget Data Modeling! Embarcadero also provides **ER/Studio®**, the industry's best tool for collaborative data modeling.

Go Faster Today. >>> Get Free Trials and More at **www.embarcadero.com**

embarcadero®

(continued from page 14)

Scans pass the query predicates to the storage servers, and Storage Indexes contain a map of values in each 1MB storage region, any region that can't possibly contain a matching row can be eliminated without ever being read. You can also think of Storage Indexes as an alternate partitioning mechanism. Disk I/O is eliminated in analogous fashion to partition elimination. If a partition can't contain any records of interest, the partition's blocks will not be read. Similarly, if a storage region cannot contain any records of interest, that storage region need not be read.

Storage Indexes cannot be used in all cases, and there is little that can be done to affect when or how they are used. But in the right situations, the results from this optimization technique can be astounding.

[see book for example]

In this example we disabled storage indexes (using the _KCFIS_STORAGEIDX_DISABLED parameter) to remind you of the elapsed time required to read through 384 million rows using Column Projection and Predicate Filtering. Remember that even though the amount of data returned to the database tier is extremely small in this case, the storage servers still had to read through every block containing data for the SKEW3 table and then had to check each row to see if it matched the WHERE clause. This is where the majority of the 8 seconds was spent. We then reenabled storage indexes and reran the query, which reduced the execution time to about .08 seconds. This reduction in elapsed time is a result of storage indexes being used to avoid virtually all of the disk I/O and the time spent filtering through those records.

Just to reiterate, Column Projection and Predicate Filtering (and most other Smart Scan optimizations) improve performance by reducing the volume of data being transferred back to the database servers (and thus the amount of time to transfer the data). Storage Indexes improve performance by eliminating time spent reading data from disk on the storage servers and filtering that data. Storage Indexes are covered in much more detail in Chapter 4.

Simple Joins (Bloom Filters)

In some cases, join processing can be offloaded to the storage tier as well. Offloaded joins are accomplished by creating what is called a *bloom filter*. Bloom filters have been around for a long time and have been used by Oracle since Oracle Database Version 10*g* Release 2. So they are not specific to Exadata. One of the main ways Oracle uses them is to reduce traffic between parallel query slaves. They have the advantage of being very small relative to the data set that they represent. However, this comes at a price—they can return false positives. That is, rows that should not be included in the desired result set can occasionally pass a bloom filter. For that reason, an additional filter must be applied after the bloom filter to ensure that any false positives are eliminated. The interesting thing about bloom filters from an Exadata perspective is that they may be passed to the storage servers and evaluated there. This technique can result in a large decrease in the volume of data that must be transmitted back to database servers.

[see book for example]

In this listing we used a hidden parameter, _BLOOM_PREDICATE_PUSHDOWN_TO_STORAGE, to disable this feature for comparison purposes. Notice that our test query ran in about 2 minutes with Offloading and 11.5 minutes without. If you look closely at the Predicate Information of the plans, you will see that the SYS_OP_BLOOM_FILTER(:BF0000,"A""PK_COL") predicate was run on the storage servers for the second run. The offloaded version ran faster because the storage servers were able to pre-join the tables, which eliminated a large amount of data that would otherwise have been transferred back to the database servers.

Function Offloading

Oracle's implementation of SQL includes many built-in SQL functions. These functions can be used directly in SQL statements. They may be divided into two main groups: single-row functions and multi-row functions. Single-row functions return a single result row for every row of a queried table. These single row functions can be further subdivided into the following general categories:

(continued on page 18)

Boost Oracle Performance up to 10x

Challenged with deteriorating application performance?

Are business units demanding real-time analytics to drive revenue?

Bring *tier 0 performance* to your IT environment!

- ZERO changes to existing servers, storage and applications
- Cost saving by leveraging current infrastructure

GridIron TurboCharger™
Market's First SAN Application Acceleration Appliance

See Customer Success Stories at www.GridIronSystems.com

Schedule a Briefing by Emailing Sales@GridIronSystems.com

GRIDIRON

(continued from page 16)

- Numeric functions (SIN, COS, FLOOR, MOD, LOG, …)
- Character functions (CHR, LPAD, REPLACE, TRIM, UPPER, LENGTH, …)
- Datetime functions (ADD_MONTHS, TO_CHAR, TRUNC, …)
- Conversion functions (CAST, HEXTORAW, TO_CHAR, TO_DATE, …)

Virtually all of these single-row functions can be offloaded to Exadata storage. The second major group of SQL functions operate on a set of rows. There are two subgroups in this multi-row function category:

· Aggregate functions (AVG, COUNT, SUM, …)
· Analytic functions (AVG, COUNT, DENSE_RANK, LAG, …)

These functions return either a single row (aggregate functions) or multiple rows (analytic functions). Note that some of the functions are overloaded and belong to both groups. None of these functions can be offloaded to Exadata. Which makes sense, because many of these functions require access to the entire set of rows, which individual storage cells do not have.

There are some additional functions that don't fall neatly into any of the previously described groupings. These functions are a mixed bag in terms of Offloading. For example, DECODE and NVL are offloadable, but the XML functions are not. Some of the Data Mining functions are offloadable and some are not. Also keep in mind that the list of offloadable functions may change as newer versions are released. The definitive list of which functions are offloadable for your particular version is contained in V$SQLFN_METADATA.

Offloading functions does allow the storage cells to do some of the work that would normally be done by the CPUs on the database servers. However, the saving in CPU usage is generally a relatively minor enhancement. The big gain usually comes from limiting the amount of data transferred back to the database servers. Being able to evaluate functions contained in WHERE clauses allows storage cells to send only rows of interest back to the database tier. So as with most Offloading, the primary goal of this optimization is to reduce the amount of traffic between the storage and database tiers.

Summary

Offloading really is the secret sauce of Exadata. While the hardware architecture does a good job of providing more balance between the storage layer's ability to deliver data and the database layer's ability to consume it, the bulk of the performance gains are provided by the software. Smart Scans are largely responsible for these gains. The primary focus of most of these optimizations is to reduce the amount of data transferred between the storage tier and the database tier.

Editor's note: This chapter continues with many important topics for which we do not have the space to print. These topics include compression, encryption, virtual columns, Data Mining, RMAN, prerequisites, smart Scan Disablers, monitoring, and others. ▲

SERVING UP THE BEST IN DATA INTEGRATION

DBMoto™ delivers TOP PERFORMANCE in Change Data Capture and Data Integration for the most demanding tasks, without stressing your system or budget!

HiT® SOFTWARE

A BackOffice Associates, LLC Company

DBMoto VERSION 7 · *NEW! INFORMIX LOG-READING!*

DBMoto 7 offers Change Data Capture, transaction log-reading for:

Oracle, Informix, MSFT SQL Server, DB2, MySQL, and others!

- Multi-Server Synchronization
- Easy administration, quick to deploy
- DBMoto Verifier™ "compare" tool to assess replication results
- Customizable functions and automated "smart" management features
- Cost-effective – use your own IT team!

If you need quick, reliable, low-cost data integration between Oracle and other relational databases or a Netezza appliance, DBMoto is the most efficient solution available today!

FREE WHITE PAPER!

"Data Integration Cost Considerations"

www.hitsw.com/WP19L

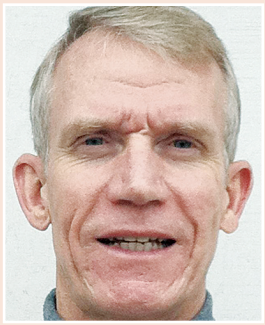
DATABASES SUPPORTED:

Oracle · IBM DB2 for i, z/OS, LUW · Microsoft SQL Server · MySQL · Netezza · Informix · Sybase ASE · SQL Anywhere · Ingres · Cloudscape · PostgreSQL

Evaluate DBMoto for FREE! Download a fully functional version at **www.hitsw.com/DL19A**

T +1.408.345.4001 **www.hitsw.com** **info@hitsw.com**

How is RAC Performance Tuning Different?



Chris Lawson

by Chris Lawson

After working on numerous performance problems on both RAC and single-node systems, I thought it would be helpful to reflect on what I see as differences between the two. I wanted to specifically address the question, *"What exactly do I do differently in performance tuning on a RAC cluster compared to single-node?"*

Note that the perspective of this paper is, above all, *practical*. For just a moment, forget about theories on cache fusion, or marketing fluff about "self-tuning" databases, magical tuning-tools, and the like. I am not interested in what RAC "could do," "should do," or "might do." I am only interested in what I do differently in *practice*—that is, resolving my customers' performance problems.

I think my answer as to "what is different with RAC" will surprise you. First, however, let's remind ourselves about the scenarios where RAC can really make a difference.

Where RAC Helps

RAC offers a good option for improved *availability*, as well as easier *scalability*. So RAC, with it's "scale out" approach, gets around maximum CPU limitations.

That is not at all the same thing as better *performance*. Occasionally, marketing folks, in their zeal, suggest that RAC will actually make things go *faster*. Of course, that's not really true. Using RAC doesn't somehow make things work faster. What the marketers really mean is that the overall throughput of your application will be larger if your application is strictly CPU-limited, and simply needs more processing power. Thus, RAC provides the opportunity to increase the number of threads run concurrently.

We must always remember, however, that there are numerous other bottlenecks that must be considered as well. For example, you might actually be experiencing waits due to available log buffers. In this scenario, adding more CPUs would be counterproductive.

RAC cannot make an application scale, if it otherwise wouldn't. As Oracle's *Admin Guide* states, "If an application does not scale on an SMP system, then moving the application to an Oracle RAC database cannot improve performance."

So, it's worth remembering that performance issues on a single-node system won't magically go away once RAC is installed. If anything, performance issues are intensified unless you actually resolve the underlying issues.

For further information, see my earlier paper, *Performance Tuning: Is RAC Like "Bolt-on Power?"*

The Essence of Performance Tuning Is the Same

In practice, I have found the steps I use to accomplish performance tuning to be nearly identical, whether on a RAC cluster, or on a single-node instance. The differences are mostly pretty mundane details, such as changing V$ to GV$, or trivial (but necessary) details, such as making sure you're querying the correct node.

I think some DBAs will be surprised by my perspective. After all, a RAC system is a lot different from a single-node system. So how can performance tuning be so similar?

The reason is really a tribute to RAC technology—especially the efficiency of cache fusion. Of course, no one would argue that cache fusion is "free," but in practice, I've found that the overhead due to RAC is not very high—in fact, in most cases, I don't even care that I'm on RAC.

RAC Overhead

Let's take a look at the overhead due to cache fusion. These statistics come from a very large database, on an 8-node cluster. In particular, let's look at one very busy node, which handles OLTP queries.

For a recent 24-hour period, let's see what the top wait events were. We'll use the familiar AWR report to glean this information:

Top-5 Wait Events for a 24-Hour Period:	
db file sequential read	277k secs
CPU time	140k secs
gc current block 3-way	24k secs
gc cr grant 2-way	16k secs
log file sync	8k secs

The above chart confirms that RAC-specific events, while certainly noticeable, aren't anywhere close to being the top-wait events. Roughly speaking, they amount to about 15% of the total wait time. Let's examine this issue more closely.

"What?! Does RAC really cause degradation of 15%?"

I don't think so—and here's why: The above RAC wait events—while indeed true delays—are occurring so that *even*

greater delays will be avoided. The inter-node block transfers are occurring in order to avoid disk reads. With RAC, we have a multi-node, monstrous-sized cache, and another node has the desired block cached. So, we do a little work (block transfer) to *avoid* bigger work (disk reads).

In practice, on very large RAC systems, I have found that Oracle spends roughly 1 ms getting a block in order to avoid spending 5 ms for a single block read (sequential read.).

Let's take another view of the impact due to RAC wait events. Again from a recent AWR on a very busy system, here is a list of the top SQL that are impacted by RAC:

SQL ordered Ordered by Cluster Wait Time

Cluster Wait Time(s)	CWT % of Elapsed Time	Elapsed Time(s)	CPU Time(s)	Execs
4,820.79	15.29	31,525.18	2,429.77	300,054
4,649.80	19.62	23,695.91	1,867.30	189,178
4,040.22	59.84	6,751.62	1,181.02	617,979
3,397.00	19.72	17,226.23	1,010.73	319,561
2,443.81	13.98	17,476.77	1,001.37	318,937
2,111.91	14.81	14,259.06	790.73	24,010
2,013.39	10.53	19,127.11	1,003.17	212,313
1,509.99	18.13	8,327.73	502.36	209,420

Observing the statistics above, note that these are the SQL that have the largest cumulative waits due to cluster delays; these are not the typical delays.

We see that the overhead of RAC is about 15%, with the exception of one SQL. Further investigation would be needed to see what is so special about this statement.

On this busy system, the 15% figure represents a rough *upper-range* for the RAC overhead. Statistically, one might say that RAC causes 15% degradation—but only when compared to a single-node instance with a single, similarly-monstrous cache. And that is probably not a realistic comparison.

RAC Differences

In practice, the #1 difference I experience will likely seem absurd at first: simply figuring out on which node the problem SQL is running. With eight nodes, however, this is not always as trivial as it seems. Oftentimes, a user will complain of some issue with a report, but they honestly don't know the node.

LONG_SESSIONS.SQL

Here's an one way to home- in on the problem session. The following script identifies user sessions—across all nodes—that have been connected for at least 8 hours. Keep in mind, however, that a long-connected session is not necessarily a problem, especially if a connection pool is being used:

```
Col Sid Format 99999
Col Serial# Format 999999
Col Machine Format A15 Truncate
Col Event Format A30 Truncate
Col Inst Format 9999

Select Inst_Id Inst,Sid,Serial#,Username,Machine,Event,
Logon_Time,Sql_Id,Prev_Sql_Id
From Gv$Session
where type != 'BACKGROUND'  and event not like 'SQL*Net%'
  and event not like 'Streams AQ: waiting%'
  And Nvl(24 * (Sysdate - Logon_Time),0) > 8
Order By Username;
```

ALL_ACTIVE.SQL

This script finds sessions from a different approach. It shows active sessions on all nodes—not just sessions connected for a long time:

```
SeleCT DISTINCT osuser, gv$session.inst_id, Sid, username,
Substr(program,1,19) PROG , sql_text
From gV$Session, gV$Sql
Where status = 'ACTIVE'
and gv$session.inst_id = gv$sql.inst_id
And username is not null
And gv$session.sql_hash_value = hash_value
and gv$session.sql_address  = gv$sql.address
and sql_text not like '%DISTINCT osuser%'
order by 2;
```

ASH_BY_TIME.SQL

Very often, I hear of a problem report such as, "My report ran too long last night." The user will know the approximate start/end time, but will almost never know the node. (Plus, sometimes, various threads run on multiple nodes.)

One of the first scripts I run is a simple ASH script that categorizes the long-running SQL by node, over a particular time period.

```
With P1 As (Select /*+Parallel(A 6) */
Distinct A.*
From Dba_Hist_Active_Sess_History A
Where Sample_Time Like '22-APR-10%4.%AM'
) Select Instance_Number, Sql_Id, Count(*)
From P1
Group By Instance_Number, Sql_Id Having Count(*) > 20
Order By Count(*)
```

In the script above, I look for the activity, for all nodes, at 4 to 5 A.M.a.m. on April 22. I employ Parallel 6 to reduce the runtime due just a minute or so. I find it convenient to use query subfactoring (the "with" syntax), but of course that is not mandatory.

High-Resource SQL

Many performance drilldown queries are based on V$ views. I've found it easiest to substitute the GV$ view in most cases. Keep in mind, however, that behind the scenes Oracle will perform a union of the underlying V$ views, possibly making the runtime quite a bit longer.

One of my most oft-run scripts queries *V$Sql* to find high-resource SQL. With RAC, you need to slightly alter this to query *GV$Sql*. The result set will show the SQL from each instance meeting the criteria.

Many performance DBAs use scripts similar to the one below, with different filters—e.g., elapsed time, disk reads, buffer gets, etc. You will also want to order the result set to match what exactly you're looking for:

```
Select Inst_id, Sql_Id, First_Load_Time, Round(Elapsed_Time/1000000)
Secs, Rows_Processed, Executions, Buffer_Gets, Disk_Reads, Sql_Text
From Gv$Sql
Where Executions > 0
And Disk_Reads > 1000000
Order By 1
```

In the above script, Oracle references ***Inst_Id***, but the exact field name changes from view to view! Just when you think you've got it figured out, Oracle switches the column name.

Note that Oracle uses slightly different field names to identify the instance, depending on the data dictionary view. I count eight different ways:

```
INST#
INSTANCE
INSTANCE#
INSTANCE_ID
INSTANCE_NUM
INSTANCE_NUMBER
INST_ID
INST_NUMBER
```

After seeing all the ways that Oracle can identify an instance, I feel much better about getting the name mixed up all the time.

Some Other Differences in RAC

AWR Reports

Here's another nit-picky difference: with RAC, you need to pick a node for which you want an AWR report. (at least, if you're running the official AWR report script.). On a two-node cluster, not a big deal. Not quite so trivial on an eight-node system. Of course, you can always guess, and run multiple reports.

RAC-Specific Bugs

In my experience, this has proven to be a major issue. There are some bugs peculiar only to RAC. For example, Oracle 10.2.0.3 has a RAC bug related to parallelism. With RAC, the child parallel processes will often jump to a different node than where the parent query originated. There is nothing wrong with this, but when the query is particularly complicated, with multiple layers of parallelism, Oracle gets confused, and the parallel processes spin forever. Of course, we applied this patch, but in the meantime, there was much confusion as to what was happening.

Conclusion

Over the course of resolving thousands of performance issues—both on RAC and on single-node systems—I have slowly realized that there are few major differences in the performance tuning process. In practice, what I do to resolve a performance issue is amazingly similar. I just remember to ask that question, "What node are you on again?" Of course, depending on your particular business or configuration, you may encounter very specific differences in your RAC setup.

In general, the same clear, logical thinking process works in both environments. If you are adept at solving performance problems in single-node, you will likely be similarly adept in the RAC environment. Likewise, if you are rather clumsy at solving performance problems in single-node... ▲

Chris Lawson is an Oracle Ace and performance specialist in the San Francisco Bay Area. He is the author of The Art & Science of Oracle Performance Tuning, *as well as* Snappy Interviews: 100 Questions to Ask Oracle DBAs. *Chris would be happy to hear from you. He can be reached at* RAC@OracleMagician.com.

Usually, it's harder to pinpoint.

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

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

A smarter solution makes everyone look brilliant.

CONFIO SOFTWARE

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

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

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, Iggy Fernandez, at **iggy_fernandez@hotmail.com**. ▲

Long-term event sponsorship:

CHEVRON

ORACLE CORP.

Thank you! Year 2011 Gold Vendors:

- Confio Software
- Database Specialists, Inc.
- Delphix
- Embarcadero Technologies
- GridIron
- Rolta TUSC

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

TREASURER'S REPORT

Naren Nagtode, *Treasurer*

Beginning Balance		
April 1, 2011		**$ 72,760.17**
Revenue		
Membership Dues	5,227.94	
Meeting Fees	50.00	
Vendor Receipts	6,500.00	
Advertising Fee	–	
Training Day	13,300.00	
Sponsorship	–	
Interest	4.81	
Paypal balance	–	
Total Revenue		**$ 25,082.75**
Expenses		
Regional Meeting	6,899.47	
Journal	3,861.59	
Membership	182.58	
Administration	279.56	
Website	–	
Board Meeting	425.00	
Marketing	–	
Insurance	–	
Vendors	117.50	
Tax	–	
Training Day	18,689.00	
IOUG Registration	925.00	
Miscellaneous	–	
Total Expenses		**$ 31,379.70**
Ending Balance		
June 30, 2011		**$ 66,463.22**

NoCOUG Summer Conference

Session Descriptions

For the most up-to-date information, please visit **http://www.nocoug.org**.

–Keynote–

Optimizing a Two-Table Join

Jonathan Lewis 9:30–10:30

The optimizer is very good at transforming your query into something that you didn't write; sometimes, though, Oracle's transformation is a bad idea; sometimes you can transform a query manually in ways that the optimizer cannot.

In this presentation we examine a single, fairly simple, query to see how many different things we can do to make it more efficient, using strategies which the optimizer, at present, may not consider.

–Room 1220–

Beginners' Guide to Becoming an Expert **Editor's Pick**

Jonathan Lewis 11:00–12:00

In this slide-free presentation Jonathan Lewis will discuss and demonstrate what it takes to become an expert in areas relating to Oracle performance. There will be NO "rocket-science" in this presentation, just simple demonstration and discussion. There will be plenty of time for questions and answers —and Jonathan will be asking most of the questions.

Best Practices for Monitoring Oracle on VMware

Dean Richards, *Confio* 1:00–2:00

Databases on VMware? Heard about this lately? Many of our customers are either virtualizing databases or planning to do so soon. Most of our customers have already implemented virtualization on the VMware platform and databases seem to be the last frontier. I also hear from many groups that have tried running their database on VMware but went back to physical hardware because of performance issues. There are many reasons this may have happened, but quite often the decision to go back to physical was not based on hard facts. When virtualizing, many things change including storage configuration, memory settings, CPU allocations, application changes and many more. Most times, when reviewing the details with these customers, their performance problems were not caused by VMware but rather something else that could have been easily fixed. The underlying problem is that they did not have a clear view of performance once they went to a VMware environment and made uneducated decisions as a result.

Did you know that O/S metrics from a virtual machine (VM) are no longer reliable? For example, if a VM is allocated 8GB of RAM, but has a VM memory limit set to 4GB, what is the memory utilization for that O/S if it's currently using 3.8GB? The O/S will report the memory utilization as 3.8 / 8 or 47.5%.

However, since there is a VM limit in place, the real utilization is 3.8 / 4 or 95%, which is a much different perspective. This is one of only a handful of problems you will run into when using O/S metrics (including tools that collect their data from the O/S) as the basis for performance data. This is also a root cause of why customers made uneducated decisions about moving their databases off VMware.

More and more companies are virtualizing with VMWare, and databases are the next logical step. This presentation will explore the fundamentals of monitoring databases in a VMWare environment because it can be much different than when running on a physical machine. You will learn business and technical benefits of virtualization, master new terms and concepts, pick up useful planning tips and tricks and cover best practices for maintaining optimum performance in a VMWare environment.

NFS Tuning

Kyle Hailey, *Delphix* 2:30–3:30

NFS transmission speeds are already on par with fiber channel and could significantly pass up fiber channel by 2012. NFS is easier and cheaper to install, configure and maintain than fiber channel and includes robust error handling On the other hand NFS has a reputation as being slow. Learn how close FC and NFS are how NFS can be optimally configured and how to analyze for NFS bottlenecks.

Extreme Performance Data Warehousing

Oracle Corporation 4:00–5:00

Extreme query performance and in-database analytics are key requirements for data-driven organizations looking to uncover revenue opportunities and take action on trends faster. However, the need to ask more complex questions and analyze greater volumes of information can conflict with these requirements.

In this presentation, we'll examine how intelligent optimizations in Oracle Exadata and Oracle Database 11*g* can improve data warehouse performance by a factor of 10x, reduce storage footprint by 10x, and meet all your data warehousing requirements, from simple ad-hoc queries to complex predictive analytics.

–Room 1240–

Private Cloud Database Consolidation

Oracle Corporation 11:00–12:00

Oracle Database 11*g*, Oracle Exadata and Oracle Real Application Clusters enable consolidation of multiple applications on clustered server and storage pools—providing unbeatable

fault tolerance, performance and scalability. This keynote will discuss how to consolidate your databases onto a private cloud—and realize the efficiencies of mixed workload consolidation, workload and resource management, and dynamic provisioning for elastic scalability.

Oracle In-Memory Database Cache

Oracle Corporation 1:00–2:00

In today's fast-paced business environment, every second counts. The ability to access, capture and analyze data in real time with low latency and high throughput is a competitive advantage for many companies. In this session, learn how Oracle In-Memory Database Cache enables application developers to use standard SQL and database APIs to dramatically improve application response time and throughput by caching performance-critical subsets of an Oracle Database in the application tier.

The in-memory cache grid enables applications to easily scale-out without service interruption, making Oracle In-Memory Database Cache ideal for a wide-range of applications with requirements for extreme performance in the application-tier.

Oracle 11*g*: Learning to Love the Automatic Diagnostic Repository

Don Seiler, *Pythian* 2:30–3:30

Oracle 11*g* has introduced a whole new system of collecting diagnostic information, the Automatic Diagnostic Repository, or ADR. Gone are the familiar bdump, cdump and udump directories, all hail our new XML overlords! Where's my alert log? And why on earth is it producing so much trace data?!? Like it or not, the ADR is here. It's time to embrace it with both arms to get the most out of our Oracle databases.

In this session, learn about the core ADR features that every DBA will want to be familiar with, including file locations, retention policies and using adrci to view alert log information with some helpful filters. Then go through a scenario to demonstrate the new ADR-based packaging system for Oracle Support Services.

Leveraging Mobile Technology for Oracle Databases Management

Hanan Hit & Faisal Faruqi. 4:00–5:00

Nexscience's technical architects conducted a comprehensive study of how mobile platform and cloud computing can be leveraged to manage Oracle databases in a Software-as-a-Service model. What are the potential benefits, what are the risks and who are the major players currently working on developing solutions for mobile databases management. We intend to present these technical findings for the benefit of the Oracle user community.

–Room 1150–

7 Data Masking Tips & Tricks

Ilker Taskaya , Axis Technologies 11:00–12:00

Data Masking is the act of replacing sensitive data with fictitious but realistic data in order to eliminate the risk of exposure to unauthorized parties. Implementing a sustainable data masking program in a modern enterprise environment can be surprisingly challenging—not just technically and organizationally, but culturally. The reactions and concerns you may encounter from your development and QA teams are understandable. You're not just introducing new controls over confidentiality exposures in development and testing systems. You're shutting off developers' and testers' unfettered access to live customer Production data—something that's been quite normal and convenient for them to date, something they think they just can't do their job without. Any project is only as successful as its level of buy-in from all involved. Getting colleagues to "do things a little differently" than they have traditionally means getting them on board early and anticipating their concerns. This presentation recounts lessons learned from years of implementing data masking programs at some very large companies with Oracle database and applications such as PeopleSoft.

Riak Top to Bottom

Andy Gross 1:00–2:00

In this talk Andy Gross, Basho's Principal Architect, will give you an overview of Riak, Basho's open source, "NoSQL" database, with an emphasis on real world applications and deployments, and what today's DBA can expect when working with Riak in production.

Tentative talk agenda is as follows:

- Riak—An Overview
- Riak for the Ops Professional
- Riak and Relational DBs—Can they coexist?
- Riak in Production

Database Consolidation: Better Data Security at Lower Cost

Oracle Corporation 2:30–3:30

More and more organizations are consolidating their databases and taking advantage of private cloud computing environments to drive down IT costs. But since two thirds of sensitive data in most organizations resides in databases, did you know that consolidation can also improve data security and regulatory compliance?

In this session, you will learn how replacing costly and insecure database silos and adopting a database defense in depth strategy will help safeguard all your mission critical enterprise data. You will also learn about the new Oracle Database Firewall, which offers unprecedented capabilities to monitor database traffic and block threats before they even reach your databases.

High Performance Oracle Database in a Flash

Sumeet Bansal, Fusion-io. 4:00–5:00

Many organizations are deploying NAND Flash Solutions in their Oracle Database environments. This presentation will cover the following topics:

- Centralized Database architecture vs. Distributed Database architecture
- Goals of NAND Flash based architectures
- Oracle ASM and NAND Flash solutions
- Single Instance Oracle environments
- Oracle RAC environments
- Setting up High-Availability with Data Guard for a NAND Flash based Oracle Database
- Real World examples of ioMemory deployments. ▲

Database Specialists: DBA Pro Service

DBA PRO BENEFITS

- *Cost-effective and flexible extension of your IT team*
- *Proactive database maintenance and quick resolution of problems by Oracle experts*
- *Increased database uptime*
- *Improved database performance*
- *Constant database monitoring with Database Rx*
- *Onsite and offsite flexibility*
- *Reliable support from a stable team of DBAs familiar with your databases*

CUSTOMIZABLE SERVICE PLANS FOR ORACLE SYSTEMS

Keeping your Oracle database systems highly available takes knowledge, skill, and experience. It also takes knowing that each environment is different. From large companies that need additional DBA support and specialized expertise to small companies that don't require a full-time onsite DBA, flexibility is the key. That's why Database Specialists offers a flexible service called DBA Pro. With DBA Pro, we work with you to configure a program that best suits your needs and helps you deal with any Oracle issues that arise. You receive cost-effective basic services for development systems and more comprehensive plans for production and mission-critical Oracle systems.

DBA Pro's mix and match service components

Access to experienced senior Oracle expertise when you need it
We work as an extension of your team to set up and manage your Oracle databases to maintain reliability, scalability, and peak performance. When you become a DBA Pro client, you are assigned a primary and secondary Database Specialists DBA. They'll become intimately familiar with your systems. When you need us, just call our toll-free number or send email for assistance from an experienced DBA during regular business hours. If you need a fuller range of coverage with guaranteed response times, you may choose our 24 x 7 option.

24 x 7 availability with guaranteed response time
For managing mission-critical systems, no service is more valuable than being able to call on a team of experts to solve a database problem quickly and efficiently. You may call in an emergency request for help at any time, knowing your call will be answered by a Database Specialists DBA within a guaranteed response time.

Daily review and recommendations for database care
A Database Specialists DBA will perform a daily review of activity and alerts on your Oracle database. This aids in a proactive approach to managing your database systems. After each review, you receive personalized recommendations, comments, and action items via email. This information is stored in the Database Rx Performance Portal for future reference.

Monthly review and report
Looking at trends and focusing on performance, availability, and stability are critical over time. Each month, a Database Specialists DBA will review activity and alerts on your Oracle database and prepare a comprehensive report for you.

Proactive maintenance
When you want Database Specialists to handle ongoing proactive maintenance, we can automatically access your database remotely and address issues directly — if the maintenance procedure is one you have pre-authorized us to perform. You can rest assured knowing your Oracle systems are in good hands.

Onsite and offsite flexibility
You may choose to have Database Specialists consultants work onsite so they can work closely with your own DBA staff, or you may bring us onsite only for specific projects. Or you may choose to save money on travel time and infrastructure setup by having work done remotely. With DBA Pro we provide the most appropriate service program for you.

Database Specialists

CALL 1-888-648-0500 TO DISCUSS A SERVICE PLAN

NoCOUG
P.O. Box 3282
Danville, CA 94526

NoCOUG Summer Conference Schedule

August 18, 2011—at Chevron in San Ramon

Please visit **http://www.nocoug.org** for updates and directions, and to submit your RSVP.
Cost: $50 admission fee for non-members. Members free. Includes lunch voucher.

8:00 a.m.–9:00	Registration and Continental Breakfast—Refreshments served
9:00–9:30	**Welcome:** Iggy Fernandez, NoCOUG president
9:30–10:30	**Keynote:** *Optimizing a Two Table Join*—Jonathan Lewis
10:30–11:00	**Break**
11:00 –12:00	**Parallel Sessions #1**
	Room 1220: *Beginner's Guide to Becoming an Expert*—Jonathan Lewis **Editor's Pick**
	Room 1240: *Private Cloud Database Consolidation*—Oracle Corporation
	Room 1150: *7 Data Masking Tips & Tricks*—Ilker Taskaya, Axis Technologies
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Room 1220: *Best Practices for Monitoring Oracle on VMware*—Dean Richards, Confio
	Room 1240: *Oracle In-Memory Database Cache*—Oracle Corporation
	Room 1150: *Riak Top to Bottom*—Andy Gross, Basho
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Room 1220: *NFS Tuning*—Kyle Hailey, Delphix
	Room 1240: *Oracle 11g: Learning to Love the Automatic Diagnostic Repository*—Don Seiler, Pythian
	Room 1150: *Database Consolidation: Better Data Security at Lower Cost*—Oracle Corporation
3:30–4:00	**Raffle**
4:00–4:30	**Parallel Sessions #4**
	Room 1220: *Extreme Performance Data Warehousing*—Oracle Corporation
	Room 1240: *Leveraging Mobile Technology for Oracle* Databases Management—Hanan Hit and Faisal Faruqi
	Room 1150: *High Performance Oracle Database in a Flash*—Sumeet Bansal, FusionIO
5:00	**NoCOUG Networking and No Host Happy Hour at Izzy's Steaks (bar area), 200 Montgomery Street, San Ramon. 925-830-8620**

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

The *NoCOUG Journal* design and production: Giraffex, Inc., S.F. Cover image: photos.com.