nocoug
JOURNAL

Official Publication of the Northern California Oracle Users Group

VOL. 15, No. 2 · MAY, 2001 $15

Letter from the Editor

Throughout the years that the NoCOUG Journal has been produced, each volunteer Journal Editor has put their own stamp on the publication. Most recently, Cecile Lavoie and John Pon before her, have taken on this tremendous task with skill and dedication. Now it's my turn.

Lisa Loper, Editor

As the new editor of the NoCOUG Journal, I'm excited by the opportunity to take on this role and make some enhancements to the Journal. I also must thank Colleen Childers in my office for her vast contributions in helping me put this issue together. You'll see several changes as you read through this Journal. We've changed the look and feel quite a bit and added more of a focus on our local user community within NoCOUG. We've also reduced the number of technical articles published and started publishing website resources where you can get much more documentation online.

This publication is for you, NoCOUG members. So I am asking you for feedback. What do you think of this new edition of the Journal? What additional features or changes would you like to see? Would you like to contribute with an article sometime? Write to me at journal@nocoug.org with any comments you have.

I look forward to hearing from you. ▲

Been to Our Website Lately?

If you've been to our website recently, you've noticed something different. We've made significant improvements to the NoCOUG website in order to better serve you, our members, and make more resources available to the Oracle user community. Check it out at www.nocoug.org.

In the Meetings area of the website, you'll find up-to-date information about our upcoming meetings, as well as a repository of information from previous meetings. In the Presentations area you'll find white papers and slide shows from previous meetings, available

(continued on page 2)

TABLE OF CONTENTS

–ADVERTISERS–

Been to our Website Lately?

(continued from page 1)

for download. Also, the Links area of the website provides links to many other useful websites for Oracle DBAs, application developers, and managers.

While you are visiting the NoCOUG website, please take two minutes of your time to complete our online survey. It really is fast and easy. The information you provide will help the board of directors bring you the kind of meeting and newsletter content that interests you the most. We won't put you on spam-filled mailing lists, and you don't even have to provide your name and contact information if you prefer to be anonymous. ▲

Register for the NoCOUG Spring Conference online at www.nocoug.org

Other User Groups

Local

NorCalOAUG – Northern California Oracle Applications Users Group

- **Contact:** Cathy Cakebread (650) 562-1167
- **Website:** www.norcaloaug.org

Sacramento

SacOUG – The Sacramento Oracle User Group

- **Contact:** Ravi Verma (916) 705-3261
- **Email:** ravi.verma@telcommand.com
- **Website:** www.sacoug.org

International

IOUG-A – International Oracle Users Group of the Americas

- **Website:** www.ioug.org

U.S. Domestic

OAUG – Oracle Applications Users Group

- **Website:** www.oaug.org

ODTUG – Oracle Development Tools User Group

- **Website:** www.odtug.com

Canvassing calls by employment recruiters to local chapter contacts is strongly discouraged.

Publication and Submission Format

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

The submission deadline for the August issue is July 1st. Article submissions should be made in electronic format via email if possible. Word documents are preferred.

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

NOCOUG BOARD

President
Vilin Roufchaie, Cingular Wireless
vilin.roufchaie@cingular.com

Vice President
Joel Rosingana, Independent Consultant
joelros@pacbell.net

Treasurer/Secretary
Judy Lyman, Independent Consultant
gooma@compuserve.com

Membership
Merrilee Nohr, Design Generations, Inc.
Merrileenohr@compuserve.com

IOUG-A Representative
Ken Leonard, Oracle
ken.leonard@oracle.com

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

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

Member, At Large
Cecile Lavoie, Marimba
clavoie@marimba.com

Vendor Relations
Vacant Position

ADVERTISING RATES

Contact: Nora Rosingana

325 Camaritas Way
Danville, CA 94526
Ph: 925/820-1589

The NoCOUG Journal is published quarterly.

The rates are:

Size	Per Issue	Per Year
Quarter Page	$100	$320
Half Page	$200	$640
Full Page	$400	$1,280
Page 3	$600	$1,920

Personnel recruitment ads are not accepted.

Meet the Veep

by Colleen Childers

Our quarterly meetings are relaxing days for our members, thanks in great part to Joel Rosingana, NoCOUG Board of Directors' Vice President. However, Joel is far from taking it easy that day — it seems as if he gets in several miles of running around! We thought you might like to know a little about Joel and why he is so active with the group.

Here's Joel inviting you to the reception.

Joel has been a member of NoCOUG for nearly ten years. Since Joel typically arrived at the meetings early, he offered a well-needed hand. That quickly led to a position on the Board of Directors. The first role he took on was that of what used to be called "Happy Hour Coordinator." Even now, Joel is very encouraging to members to attend the reception after the meeting. He feels strongly about the social aspect to our meetings in addition to the technical learning opportunities.

Living in a "houseful of computers," Joel is no stranger to technology. He has 45 years of technological experience under his belt. In the early 1990's, on a contract as a COBOL programmer, Joel found his project switching to Oracle. Joel made the switch himself and has been sold since. He really likes how Oracle is straightforward. No doubt that math degree and having braved the trenches of substitute teaching have given Joel his professional edge!

When asked what he gets out of being involved with NoCOUG, Joel says that he enjoys the interaction between members and enjoys meeting new people. Additionally, being able to "get in on the ground floor of what's happening with Oracle" is a big plus. By being on the Board of Directors, he is able to meet what he feels is his personal responsibility to the organization and pay it back for all the good it does.

A lot of what Joel does (as well as the other Board members) is behind the scenes. In addition to helping with the overall direction NoCOUG, Joel has coordinated our host sites for our quarterly meetings. Years ago, when the group was smaller, approximately 50-70 members attended each meeting. However, as membership grew and we approached 200 members at each meeting, we outgrew the space of most our corporate sponsors and had to move offsite. Joel is also representing NoCOUG at the IOUG-A Live! 2001 in Orlando.

Joel, a Bay Area native, lives in the East Bay with his wife of 31 years, Nora. Since retiring (which he's done several times), he has been very active in his local community. He has volunteered with the Boy Scouts of America as well as for the local school system. When he's not busy during the wrestling season with San Ramon Valley High School, he runs a private wrestling club, helping to keep the kids active. Then, when he gets some free time, he enjoys his cabin in the Sierra Nevada Mountain Range.

So, don't be shy the next time you see Joel—introduce yourself. He'd love to meet you! ▲

Don't Miss Our Next Meeting!

The NoCOUG Board has planned a great conference that you won't want to miss. It's taking place on Wednesday, May 16 in Sunnyvale. Sponsored by Lockheed Martin, the one-day conference will have three parallel tracks:

- DBA
- Data Warehouse
- Development

Registration begins at 8:30a.m. at Lockheed Martin, Building 157, 3rd Avenue (at J Street), Sunnyvale. For more details, see our website at www.nocoug.org. ▲

Seen & Heard

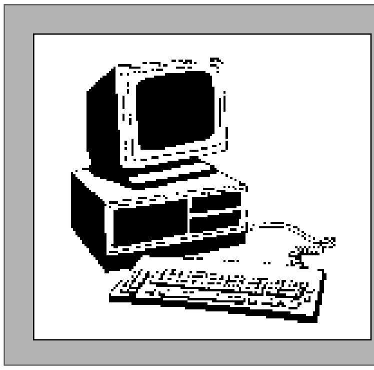
Asked to NoCOUG members:

"What Oracle reference do you most often call upon for help?"

Member **Robin Houston, DBA** at Elan Pharmaceuticals in South San Francisco says the book that she finds most useful is Michael Ault's "Oracle8i Administration and Management." She feels it is comprehensive and a handy guide.

Chris Lawson, Senior Oracle DBA consultant and frequent presenter at NoCOUG meetings, says "Oracle SQL High Performance Tuning" by Guy Harrison is his all-time favorite. "This book is vastly superior to 99% of other purported tuning books. This book has saved me huge amounts of time, and assisted me in solving numerous performance problems, with great benefit to myself and my clients."

Don't forget about Oracle documentation. **Rupert Tang, Oracle DBA** with Instantis, has the electronic version of the Oracle Documentation Library installed on his PC. He calls upon it from time to time to check for syntax, explanations, etc. ▲



Do you have a favorite Oracle reference? Let us know at journal@nocoug.org

The Power of Perl

Perl is a powerful, free, cross-platform scripting language that helps you complete all sorts of tasks quickly. With the DBI:DBD module, your perl scripts can access an Oracle database. At www.cpan.org you'll find a vast library of perl modules available for you to download and use right away. Using already written modules lets you build scripts, tools, and applications very quickly.

As a DBA, you might find perl useful for loading flat file data into an Oracle database, monitoring databases, or collecting database performance statistics—just to name a few uses.

If you'd like to learn more about perl, check out:

www.perl.com

www.perlmonth.com

www.perldoc.com

www.perl.org

Can't get enough of perl? Be sure to check out *Perl Cookbook* by Tom Christiansen and Nathan Torkington from the O'Reilly series. ▲

Export / Import using named pipe with compress

To export to a compressed file:

```
/etc/mknod pipename p
compress < pipename > newfilename.Z &
exp <username>/<password> file=pipename
rm -f pipename
```

To import from a compressed file:

```
/etc/mknod pipename p
uncompress < newfilename.Z > pipename &
imp <username>/<password> file=pipename
rm -f pipename
```

*—from **Stewart McGlaughlin**, Oracle DBA*
www.stewartmc.com/oracle/tips.html

Display Installed Products & Version

This will show you everything installed and its version for your defined $ORACLE_HOME. Make sure your $ORACLE_HOME variable is set correctly.

```
cd $ORACLE_HOME/orainst
      ./inspdver
```

*—from **Stewart McGlaughlin**, Oracle DBA*
www.stewartmc.com/oracle/tips.html

Mark Your Calendars!

The NoCOUG Summer Conference will take place on Thursday, August 23rd at Chevron in San Ramon. See you there!

Get Involved! NoCOUG Volunteer Positions

Board Members

Volunteering for the Northern California Oracle Users Group is a great way to get involved, meet new people, and contribute to the organization. Following is an outline of some of our volunteer opportunities. All are unpaid positions.

In addition to assistance in defining objectives, guidelines, and the direction for the Organization, as defined in the NoCOUG Director's Handbook, a NoCOUG board member takes on various responsibilities and tasks, such as:

- Searching for conference presenters for the quarterly regional NoCOUG meetings
- Serving on committees with various officers to complete projects
- Assisting in the procurement of the various resources and supplies essential to keeping the organization moving forward
- Serving as an officer of the board; i.e., President, Vice-President, Secretary, Treasurer, Director of Membership, Vendor Relations Coordinator, IOUG-A Alliance Representative, Journal Editor, Webmaster, etc. (Each of these positions has specific duties and responsibilities.)

Vendor Relations Coordinator

This is a board position with duties including:

- Assisting vendors before the quarterly regional meetings to answer questions
- Responding to new vendor inquiries
- Being on-hand at the meetings to assist vendors
- Approaching potential new vendors

Conference Assistants

During the organization's quarterly regional meetings, the various duties and responsibilities of this role may include:

- Staffing the registration or volunteer table
- Assisting with speaker coordination and setup
- Distributing and collecting questionnaires/surveys
- Providing assistance wherever it might be needed during the meetings

Webmaster Assistant

The primary responsibilities of this position are:

- Maintaining and enhancing the organization's website (www.nocoug.org)
- Collaborating and working closely with the Webmaster to improve the appearance and functionality of the organization's website

Training Seminar Organizers/ Assistants

The primary role and duties of this position are:

- Making program decisions
- Organizing and driving to completion a full-day training program involving one or more speakers and tracks
- Searching for and coordinating presenters/trainers

Conference Speakers

What better way to share your wealth of knowledge on a subject than to lead a session at one of our quarterly regional meetings? These meetings are usually held in the middle of February, May, August and November, at convenient sites throughout the Bay Area. All presentations—one in the morning and two in the afternoon—last for one hour each. There are three parallel tracks during each conference that vary in subject matter in order to meet the educational needs of our members.

Please contact NoCOUG (board@nocoug.org) to learn more about any of these rewarding volunteer opportunities.

Call For Presentations!

The NoCOUG Board of Directors is in the planning stages for the August 23, 2001 meeting, which will be held at Chevron in San Ramon. We would like to take this opportunity to ask if you or someone in your organization has a presentation or would like to prepare a presentation to share with Oracle users at that meeting. It's a great opportunity to get involved with NoCOUG and share your knowledge.

The meeting will feature technical presentations in three different tracks or subject areas. We haven't finalized the exact track titles yet, but most likely there will be tracks for database administration and application development. Here is what we are looking for in each of these two tracks.

Database Administration

We look for database topics that cover a wide range of areas including tuning, configuration, high availability, maintenance, and tools.

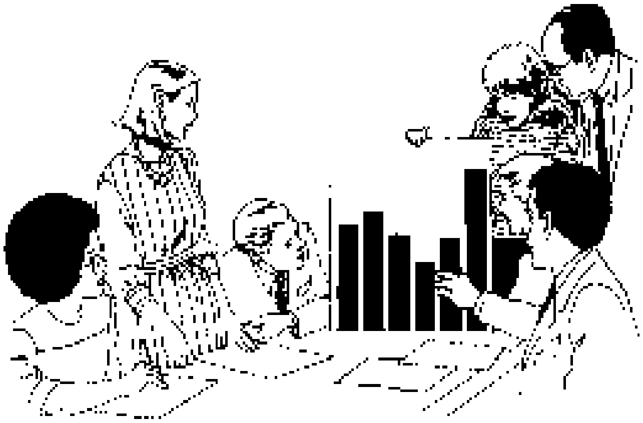
Application Development

We invite presenters covering application servers and application development using various languages and tools, such as Java, Perl, PL/SQL, and Oracle 9iAS.

Sessions are one hour in length. Presentations should run about 50 minutes, allowing ten minutes for questions and answers. Sessions should be technical in nature and include a visual presentation such as a PowerPoint slide show.

If you are interested in this opportunity, please send an email to board@nocoug.org and let us know what topic you would like to present. A NoCOUG representative will contact you. Or, if you would like to present at a future meeting (such as November 2001 or February 2002) please contact us so we can pencil you in!

We look forward to hearing from you.



Presenting a session is a great way to get involved and share your knowledge. (Plus, it looks great on your résumé!)

SPONSORSHIP APPRECIATION

Many Thanks to Our Sponsors

The Northern California Oracle Users Group (NoCOUG) is a non-profit, volunteer organization dedicated to the education and representation of the users of Oracle Corporation's database and tools software within the Northern California region.

NoCOUG has been facilitating quarterly conferences for the last 16 years in order to provide a forum for users and vendors of Oracle products. These conferences provide a networking environment for Oracle professionals to stay current with the rapidly changing Oracle technology and third party tools.

Additionally, NoCOUG publishes a quarterly newsletter that is distributed to members prior to each conference. This publication includes papers in support of topics to be presented and other helpful information relating to products and membership.

NoCOUG is affiliated with the International Oracle Users Group of the Americas and through them with all other Oracle users groups throughout the world.

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low cost membership dues. If your company is able to offer sponsorship at any level, please contact NoCOUG President Vilin Roufchaie at 925-487-9588 or write to board@nocoug.org.

Long-term full event sponsorship:

LOCKHEED MARTIN

CHEVRON

Long-term supplemental event sponsorship:

ORACLE

TREASURER'S REPORT

Judy Lyman, *Treasurer*

Beginning Balance		
January 1, 2001		**$ 24,653.11**
Revenue		
Advertising	-	
Sponsorships	-	
Membership Dues	19,868.00	
Meeting Fees	280.00	
Vendor Receipts	11,320.00	
Interest	153.98	
Total Revenue		**$ 31,621.98**
Expenses		
Regional Meeting	241.28	
Journal	4,609.25	
Membership	3,719.72	
Administration	74.38	
Income Taxes	-	
Board Meeting	448.95	
Web Site	90.00	
Web Site Design	315.00	
Miscellaneous	75.00	
Total Expenses		**$ 9,573.58**
Ending Balance		
March 31, 2001		**$ 46,701.51**

Oracle Database Performance Management

Gaja Krishna Vaidyanatha

Oracle Performance management is part-art and part-science. For practical purposes it can be further classified into 2 types, i.e. proactive and reactive performance management. Proactive performance management involves designing and developing the performance architecture for a system, during the early stages of an implementation. This involves hardware selection, performance & capacity planning, mass-storage selection, I-O sub-system configuration & tuning (i.e. RAID), and tailoring the various components to suite the complex needs of the application and an Oracle database.

Overview

The reactive component involves performance evaluation, troubleshooting, tuning and fine tuning an Oracle environment, within the boundaries of the existing performance architecture. This paper is intended to provide a solid knowledge base for any "reactive" effort. The mission of this paper is to provide the core (minimal) set of parameters and components that require configuration and tuning, to maximize performance in an Oracle database and avoid "*Database and Instance tuning by trial and error*".

Tuning Methodology

First and foremost, specific performance goals need to be set. This will facilitate benchmarking and consistent performance measurement. It is not enough to say "*this database/query runs poorly and it needs to run faster*". The required performance numbers need to be quantified, so that the performance tuner knows what to tune, how to tune, when to tune, and more importantly when to stop tuning. One can needlessly spend time tuning the system without significant gain, if haphazard attempts to alter initialization parameters are performed. The following outlines an effective methodology to engage in a reactive performance tuning effort:

1) Measure and document current performance
2) Document required performance
3) Identify performance bottleneck(s)
4) Determine the cause for the performance bottleneck(s)
5) Tune the required component (Application, Database, I-O, Contention, O-S etc.)
6) Re-measure and document current performance

Benchmark Oracle's Health (Before & After)

Performance snapshots of "*before and after images*" need to be captured during peak activity periods. These snapshots should not be taken immediately after an instance startup, as it will skew the statistics. Also, snapshot(s) should be taken at 15-minute intervals during peak periods as the

TECH TIPS

Identify and Remove Duplicate Records

Identify duplicate records:

```
select COL1, COL2, COL#, COUNT(*)
from <OWNER>.<TABLE_NAME>
group by COL1, COL2, COL#
having count(*) > 1;
```

Remove duplicate records:

```
delete from <OWNER>.<TABLE_NAME> a
where rowid < (
   select max(rowid)
   from <OWNER>.<TABLE_NAME> b
   where b.COL1 = a.COL1
   and b.COL2 = a.COL2
   and b.COL# = a.COL#);
```

*—from **Stewart McGlaughlin**, Oracle DBA*
www.stewartmc.com/oracle/tips.html

relevant performance problems/issues will not "*surface*" over longer runs. Further, if the measuring periods are longer than 15 minutes, there may be a tendency for some performance metrics to cause false alarms. This is because the statistics may again be skewed due to the long duration of the collection. The initialization parameter TIMED_STATISTICS needs to be set to TRUE (this can be done by logging in as SYS and doing an ALTER SYSTEM SET TIMED_STATISTICS = TRUE, for an already running Oracle Instance). TIMED_STATISTICS can also be permanently set to TRUE in the Oracle initialization file (init.ora). There is no measurable overhead in setting this permanently at the instance level.

A given snapshot is begun by running the script $ORACLE_HOME/rdbms/admin/utlbstat.sql from within Server Manager, as this cannot run in SQL*Plus (until Oracle 8.1), because of the need to "*connect internal or connect /as sysdba*". In future versions of Oracle (8.1.x), the Server Manager tool may be de-supported and all DBA tasks may be performed using SQL*Plus.

A given snapshot is ended by running the script $ORACLE_HOME/rdbms/admin/utlestat.sql. On completion of execution of utlestat.sql, a report file called report.txt is created in the current directory with the system's performance statistics. This report contains all relevant metrics of the Oracle instance that was captured during the 15-minute time interval. If multiple snapshots need to be taken, care should be exercised to rename the report.txt from a given run, as it gets overwritten each time. A day and time naming standard may be used to save the multiple iterations of the performance snapshots.

TECH TIPS

Decreasing size of column (which contains data)

Let's say you want to decrease ENAME from VARCHAR2(10) to VARCHAR2(7). Oracle allows you to increase the size of a column but does not allow you to decrease the size of a column. Here's a way to decrease the size of a column which contains data.

```
SQL> desc emp
Name                                    Null?    Type
--------------------------------------------------------
EMPNO                                   NOT NULL NUMBER(4)
ENAME                                            VARCHAR2(10)
JOB                                              VARCHAR2(9)
MGR                                              NUMBER(4)
HIREDATE                                         DATE
SAL                                              NUMBER(7,2)
COMM                                             NUMBER(7,2)
DEPTNO                                           NUMBER(2)
```

```
drop table FIX_IT;

create table FIX_IT as select rowid rowid_xx, ENAME from EMP;
create index FIX_IT on FIX_IT (rowid_xx);

alter table EMP modify (ENAME NULL);
update      EMP    set  ENAME = NULL;
alter table EMP modify (ENAME VARCHAR2(7));
update      EMP    set  ENAME = (
      select ENAME from FIX_IT where rowid_xx = EMP.rowid);

drop table FIX_IT;
```

—from ***Stewart McGlaughlin****, Oracle DBA*
www.stewartmc.com/oracle/tips.html

Apart from existing methods to analyze the report.txt, it is recommended that the report.txt be dissected through an analyzer available at http://www.oraperf.com. This web-site has expert analysis information on report.txt from the performance-tuning gurus of the industry. This website offers a great deal of information on the various components of "*report.txt*", more than what is documented in any published material. The web-site takes a report.txt as input, performs analysis of the metrics and then provides recommendations based on the analysis.

Application Tuning

Even though this paper does not specifically discuss details of application tuning, an honorable mention needs to be made here. Statistically it has been proven that 80% of all Oracle system performance problems are resolved by coding optimal and commonsense SQL. Any application tuning effort involves prudent usage of indexes, appropriate use of full-table-scans, setting the optimal degree of parallelism, selection of the right join methodology, etc.

These are some of the key elements that are required for optimal application performance. There is really no substitute for optimal SQL, as it results in reduced CPU utilization, fewer I-O requests, and better response times. Said in another way, optimal SQL eventually provides a level of application scalability, that hardware cannot offer without spending a heck of a lot of money. And, sometimes even after spending a heck of a lot of money, performance does not increase significantly.

Instance Tuning - Key Initialization Parameters That Need Configuration

The following are the core Oracle initialization parameters (configured in the init.ora file) and have known to provide the biggest impact on instance tuning. All of the below mentioned parameters affect the workings of Oracle and the System Global Area (SGA). Any changes made to these parameters need to be tested before implementing on a production environment. Once the changes are made on production, monitoring of the relevant Oracle dynamic performance views and O-S statistics needs to be done, to detect any unusual behavior caused by the configuration change.

The **DB_BLOCK_SIZE** parameter is set prior to database creation. It determines the size of each block within the database. This parameter cannot be changed without recreating the database. In benchmarks, it has been observed that on each doubling of the database block size, response times for I-O intensive batch operations have reduced by ~40%. The rule of thumb to size this parameter is to set it to smaller values (8K) for On Line Transaction Processing (OLTP) or Hybrid applications and to the largest possible value (16K and up) for Data Warehousing (DW)/Decision-Support System (DSS) applications. On most operating systems, using the 32-bit version of the Oracle software, DB_BLOCK_SIZE can be configured as a power of 2 up to16K. For most OLTP/Hybrid applications 8K has been found to be the optimal. This is relevant because, the file system block size for most advanced file systems defaults to 8K. On some operating systems the 64-bit version of the Oracle software supports block size(s) of 32K or higher.

The configuration of db_block_size should usually adhere to the following formula:

DB_BLOCK_SIZE >= FILE SYSTEM BLOCKSIZE >= O-S PAGESIZE.

This will ensure that Oracle-generated I-Os will be optimally serviced and the I-O sub-system will not be stressed by the overhead of redundant or unnecessary I-O. Further, this formula will also ensure that any I-O performed with respect to paging will not incur the overhead of multiple I-O system calls. If this parameter needs to be changed during the life of the database, a full export needs to be done, followed by creation of a new database with the desired DB_BLOCK_SIZE, followed by a full import.

The **DB_BLOCK_BUFFERS** parameter determines the number of blocks in the database buffer cache in the SGA. Since this is the area, where Oracle reads data from and writes data to, and improper sizing can cause serious I-O-related performance problems. Although the required/optimal size of the database buffer cache is dependent on the nature of the application, the size of the database and the number of concurrent users, it is usually the largest component of the SGA. It is normal to find this component of the SGA to comprise 75-80% of the total memory allocated to the SGA. On the other hand, over-sizing this parameter can result in system-wide memory starvation and cause the O-S to page excessively and potentially swap.

This parameter and the SHARED_POOL_SIZE provide the biggest "*bang-for-the-buck*", in tuning the SGA for an Oracle environment. This parameter needs to be increased only if the database buffer cache-hit ratio is persistently below 70%. Even that needs to be validated, as there have been cases where the nature of the application was such that, no matter how much memory was allocated for the database buffer cache, the cache-hit ratio did not change significantly. In other cases, it has been observed that lack of optimized SQL provides a false notion of the cache-hit ratio being very high, but there were serious system-wide performance problems.

Set in bytes, the **SHARED_POOL_SIZE** parameter defines the size of the shared pool area in the SGA. The sizing of this component is very heavily dependent on the nature of the application (i.e., whether it re-uses SQL or whether it generates dynamic SQL etc.). It is also dependent on the number of concurrent users and whether or not the instance is configured to support the multi-threaded server (MTS). When an instance is configured for MTS, the shared pool area needs to be configured significantly larger, as the cursor-state and the user-session-data components of the Program Global Area (PGA) are moved into the shared pool.

Since Oracle 8.0, the LARGE_POOL_SIZE parameter should to be configured when using MTS, Parallel Query and RMAN. A good rule of thumb for most applications is to configure 16 Mb. of shared pool for every 10 concurrent users. This is not a hard-and-fast rule, as the nature of the application will dictate the final sizing of this component. This parameter needs to be modified only if the "*cache-hit ratios*" of the library cache or the dictionary cache are persistently below 90% and 99% respectively. Over allocation of the shared pool area can result in increased "*parse time*" and in some cases abnormal "*hanging*" of SQL statements. If the application does not re-use SQL effectively, any amount of memory configured for the library cache, and dictionary cache, will not improve their cache-hit ratios.

Another factor that needs to be considered in the sizing of the shared pool is the amount of stored PL/SQL code that needs to be readily accessible by the application. The "*core packages*" of the application can be identified, by querying either DBA_SOURCE or USER_SOURCE and their size can be determined by querying DBA_OBJECT_SIZE. Further, to determine whether the stored PL/SQL is "pinned" in memory the dynamic performance view V$DB_OBJECT_SIZE can be queried. Also the procedure SIZES within package DBMS_SHARED_POOL can be used to determine the size of the large packages in the application.

Sets in bytes, the **LOG_BUFFER** parameter defines the size of the redo log buffer area in the SGA. The default is usually four times the database block size, which is not optimal for most environments. For a medium sized Oracle environment, this structure should be about 512 Kb. This is one memory structure where, bigger does not mean better and sizes above 1 Mb. should be questioned. Sizing the redo log buffer larger than required, will cause unnecessary contention for the redo allocation latch. Monitoring the dynamic performance views - v$sysstat and v$waitstat and/or the report.txt for increasing values for "waits" related to redo requests to fine tune this memory structure. It needs to be mentioned here that a badly sized online redo log file can cause "*waits*" for redo requests.

The **DB_WRITERS** parameter can be set for all Oracle implementations that are supported by file systems, and where Direct I-O is unavailable. This need not to be used with raw partitions, as asynchronous I-O is more optimal. It is recommended to set this parameter up to (2 * # of independent disk drives/volumes). This parameter needs modification only if the average write queue length in report.txt is consistently above 1. In Oracle 8.0 and up this parameter has been de-supported and replaced by 2 other parameters namely DB_WRITER_PROCESSES and DBWR_IO_SLAVES. If there is a requirement to set DB_WRITER_PROCESSES to a number greater than 8, then DB_WRITER_PROCESSES can be set 1 and DBWR_IO_SLAVES can be set to "n", where n is the required value subject to (2 * # of independent disk drives/volumes)

Asynchronous I-O (**ASYNC_READ/ASYNC_WRITE/DISK_ASYNCH_IO**) is the process of issuing an I-O call and not blocking until the O-S relays a confirmation about completing the I-O request (more relevant to writes rather than reads). When asynchronous I-O is configured within Oracle, the O-S sends the final confirmation that a write is complete to the DBWR process, only if the data is physically on disk. But in the mean time, DBWR does not have to block and wait for the confirmation.

When I-O sub-systems are configured with a "*write cache*", the O-S has no way of determining whether the data is physically down on disk, as low-level firmware in the I-O sub-system perform the write to disk from the cache. Asynchronous I-O works as advertised only on raw devices and does not add any significant performance benefit when implemented on advanced file systems such as jfs, ufs, vxfs, or xfs. When asynchronous I-O is used (set to TRUE), DB_WRITERS or DB_WRITER_PROCESSES or DBWR_IO_SLAVES should be set to 1, as the combination of ASYNC_WRITE = TRUE and launching multiple DBWR processes, has caused datafile corruption in prior releases of Oracle. Further, the combined usage of asynchronous I-O and multiple DBWR processes or I-O slaves provides no measurable benefit. Normally, either one of the features are used, not both.

Important: Avoid enabling the write cache in the I-O sub-system on disk volumes that house the online redo logs.

The **COMPATIBLE** parameter defines the "*code tree*" for the Oracle executables. This needs to be set to the current release of the Oracle software installed, unless instructed otherwise by Oracle World Wide Support, to get around some platform/version-specific bugs. For example, if the version of the software installed is 8.1.5.0, then it is normal to set this parameter to the same value.

The platform-specific parameter **USE_DIRECT_IO** lets the Oracle DBA "*have the cake and eat it too*" from the perspective of I-O. Direct I-O provides "*raw device*" comparable performance without losing the ease of use provided by file systems. Raw devices tend to be difficult to manage and backup. Raw devices also lend themselves very inflexible when it comes to resizing or adding datafiles. Also, improper or incorrect management of raw devices can result in production downtimes. On the other hand, when using Direct I-O, the reads are performed on file systems, by bypassing the file system buffer cache and thus providing greater throughput. Direct I-O is supported on specialized file systems such as the xfs file system (which is extent based) and is recommended over asynchronous I-O. When using the direct I-O feature on advanced file systems, it is recommended to configure the I-O related parameters in the following fashion:

> ***Raw devices tend to be difficult to manage and backup.***

DB_BLOCK_SIZE = FILE SYSTEM BLOCKSIZE = 8K – For OLTP/Hybrid Systems

DB_BLOCK_SIZE = FILE SYSTEM BLOCKSIZE = Multiple of 16K – Data Warehouse/DSS Systems

The above recommendations assume that the O-S page size is at least 8K.

The **DB_FILE_MULTIBLOCK_READ_COUNT** parameter determines the I-O chunk size for reads while performing full-table-scans and full/range-index-scans. This parameter needs to be set in compliance with the largest I-O chunk size that the O-S can support (if O-S chunk size is 128Kb and the database block size is 8Kb, then DB_FILE_MULTIBLOCK_READ_COUNT usually set to either 16 or 32). But for most implementations setting this parameter greater than the chunk-size of the O-S, will usually not result in any significant I-O performance improvement.

The **HASH_MULTIBLOCK_IO_COUNT** parameter defines the I-O chunk size that is used during hash joins and is significant in Oracle 7.3 or higher. The use of hash joins is significant for good batch report performance from Oracle 7.3.3. It has proven to deliver excellent performance for joins where small tables are joined with very large tables and a significant portion of the large table is processed. While this parameter can be set to the same value as

DB_FILE_MULTIBLOCK_READ_COUNT, it is recommended to monitor the behavior of overall query performance, as setting this parameter too high can influence the cost-based optimizer to use hash joins as the preferred join methodology. In the section on "*Special Parameters*", the method to offset this behavior has been discussed.

The **SORT_AREA_SIZE** parameter specifies the maximum amount of memory, in bytes, that is available for the sorting phase of a sort. The **SORT_AREA_RETAINED_SIZE** parameter sets the maximum amount of memory, in bytes, that is available for the fetch phase of a sort. Both parameters are usually set to the same value. Sorts requiring more space than SORT_AREA_SIZE need to allocate and use temporary segments on disk. It is important to ensure that the ratio of (sorts on disk/sorts in memory) is less than 1%. Caution needs to be exercised while configuring this parameter, as the memory allocated is part of the PGA of the server process. Further, if Parallel Query is enabled, then SORT_ AREA_SIZE should be set keeping in mind the degree of parallelism of a table/index , and the value of PARALLEL_MAX_SERVERS.

By configuring the **SORT_DIRECT_WRITES** parameter, Oracle is enabled to write the temporary segments directly to disk, by bypassing the database buffer cache. When configuring this parameter (set to TRUE), two other related parameters namely SORT_WRITE_BUFFERS and SORT_WRITE_BUFFER_SIZE can be set to 4 and 65536. This may be the only way to enable direct sorting on disk, as the default value of AUTO has very rarely achieved the desired result. From Oracle 8.1 onwards, the only configurable sort parameter is SORT_MULTIBLOCK_READ_COUNT, as all other sort-related parameters are de-supported.

The DB_BLOCK_LRU_LATCHES parameter defines the number of latches that are configured for the LRU list(s) of the database buffer cache. It can be set to its maximum value of (2 * # of CPUs) without any degradation in performance and one can quit worrying about database buffer cache LRU latch contention.

The LOG_SIMULTANEOUS_COPIES parameter defines the number of redo copy latches used for copying redo entries into the redo log buffer. It can be set to its maximum value of (2 * # of CPUs), without any degradation in performance and one can quit worrying about redo copy latch contention. On 7.3.4 and above it can be set to (8 * # of CPUs) on most platforms. In Oracle 8.1 and above, this parameter is de-supported and is automatically configured.

The _DB_BLOCK_HASH_BUCKETS parameter defines the number of hash buckets that are available for the database buffer cache. This has a direct impact on the length of the chain that a server process has to traverse, to identify and read a given block in the database buffer cache. This parameter can be set to the value of DB_BLOCK_BUFFERS, so that every block will be in its own chain. This parameter need not be configured if there is no evidence of "cache buffer chain" latch contention in the v$latch dynamic performance view or report.txt.

> ***To distribute I-O operations against rollback segments, it is recommended that the rollback segment tablespace be striped across multiple disks.***

The **_DB_BLOCK_WRITE_BATCH** parameter controls the chunk size for normal writes in the database. This parameter needs to be configured optimally such that the I-O sub-system is not overburdened, but the database writes that occur are completed within a reasonable amount of time. This parameter need not be manually configured for most databases, but when done so, is usually not set higher than (I-O chunk-size of the O-S/db_block_size).

The **DB_BLOCK_CHECKPOINT_BATCH** parameter controls the chunk size for writes that occur during a checkpoint in the database. This parameter needs to be configured optimally such that the I-O sub-system is not over burdened, but the database writes that occur are completed within a reasonable amount of time. This parameter need not be manually configured for most databases, but when done so, is usually not set higher than (I-O chunk-size of the O-S/db_block_size).

The rule of thumb for the number of rollback segments (**ROLLBACK_SEGMENTS**) is (# of concurrent users/4). Rollback segment sizing needs to be done in a proactive fashion, such that the extent sizes are adequate and the number of MINEXTENTS is 20. The formula to size the extents of a "*normal*" rollback segment, is to take the average value of USED_UBLK from v$transaction and round up to the next "*power of 2*".

It is obvious that the rollback segments used for large operations need to be sized significantly larger, to avoid dynamic extension of the rollback segment at runtime. To distribute I-O operations against rollback segments, it is recommended that the rollback segment tablespace be striped across multiple disks.

The **PROCESSES** parameter defines the number of processes that can attach to the Oracle SGA and should be set keeping in mind the SEMMNS O-S kernel parameter. Setting this beyond the value of SEMMNS will result in failures on database startup.

The **OPTIMIZER_MODE** parameter defines the method of optimization adhered by the Oracle Optimizer and is recommended to be set to CHOOSE from Oracle 7.3.3 and up to enable the cost-based optimizer.

The **PARALLEL_MIN_SERVERS** parameter defines the minimum number of parallel query slaves that are launched and retained from instance startup. This parameter can usually be set at (2 * # of CPUs).

The **PARALLEL_MAX_SERVERS** parameter defines the maximum number of parallel query slaves that can be simultaneously launched during the life of an instance. This parameter can usually be set at (4* # of CPUs).

The **PARALLEL_SERVER_IDLE_TIME** parameter defines in minutes, is the amount of time that a parallel query slave can be idle, before it is deallocated. This parameter needs to be set based on the activity and usage of parallel query on the system, but can be initially set to 10-15 minutes.

The **PARALLEL_SERVER_MIN_PERCENT** parameter defines the minimum number of parallel query slaves that need to be available for a given operation to be executed in parallel. If the number of available parallel query slaves is below this number (percentage), then the operation is executed serially. This parameter can be set to 0 (default), if there is guarantee of the required number of parallel query slaves most of the time. However, if the system cannot afford to execute parallel operations in a serial fashion, then this parameter can be set the required number. A parallel operation that does not procure this minimum percentage of parallel query slaves will fail to execute and generate an application error.

PLATFORM OR APPLICATION-SPECIFIC PARAMETERS

There are other platform-specific parameters and they should be individually studied, tested, investigated and then implemented on production databases.

1) **MLOCK_SGA/LOCK_SGA/_LOCK_SGA** etc. are all parameters that facilitate locking of the SGA in memory and preventing it from being paged/swapped. Proper use of this parameter will prevent the paging algorithm from considering the memory pages used by the SGA for page-outs. This provides a significant performance benefit especially during periods of heavy activity. This parameter should not be used if the system is not configured with an optimal amount of memory to begin with.

2) If the O-S supports "*vector-based*" I-O, **USE_READV** can be set to true, but I-O benchmarks before and after setting this parameter need to determine its usability and efficacy.

3) In past experiences, it has been noted that on most platforms the parameter **CURSOR_SPACE_FOR_TIME** needs to be set to FALSE (default), as setting it to TRUE, has caused significant slowdowns in library cache access times. Although this bug was fixed in Oracle 7.3, this parameter should not be set to TRUE, unless there is an application for it (e.g., a bar-code scanner application that continuously scans and accesses Oracle).

4) The parameter **ENQUEUE_RESOURCES** from Oracle 7.3 onwards needs lesser attention, if pure "*TEMPORARY*" tablespaces are used in the database. This will prevent Oracle from having to acquire and use the "*ST enqueue*" while performing sorts on disk on a temporary tablespace of type – PERMANENT. Acquiring an "*ST enqueue*", is apparently an expensive operation in a busy system and can be avoided by configuring pure "*TEMPORARY*" tablespaces. ENQUEUE_RESOURCES does require configuration and tuning, if there is evidence of "*Enqueue Contention or Enqueue Waits*", on monitoring the dynamic performance views of the instance.

> *So let's face the fact — 80% of all system performance problems can be fixed by writing optimal SQL.*

5) The OPEN_CURSORS parameter defines the maximum number of open cursors that a session can have open. This parameter is application specific, and should be set based on application knowledge or the evidence of sessions reaching the maximum allowable limit of open cursors.

SPECIAL PARAMETERS

The following parameters should not be modified from the default values until all documented performance-tuning techniques have been exhausted. The working efficacy of these parameters should be tested thoroughly before implementing them in a production environment. These parameters are most significant for packaged applications where the code is usually inaccessible. It is also relevant for environments that require hash joins for their "*batch processing*", but want to hold back the optimizer from being over-influenced by the hash join method. Most of these parameters are relevant in Oracle 7.3.5 or above.

The **OPTIMIZER_MAX_PERMUTATIONS** parameter defaults to a value of 80000. When set to a value below 80000, say between 79000-79999, it forces the optimizer to try up to 4 different tables as the driving table for queries that involve joins. This results in the optimizer picking the least expensive of the 4 plans that it generates. Usually, the default behavior is to build a plan with the smallest table as the driving table. The default behavior may not always generate the most suitable plan, especially for packaged applications. The result of setting this parameter is a nominal increase in "*parse time*", but a significant potential in reduction of "*execution time*" of SQL statements.

The **OPTIMIZER_INDEX_COST_ADJ** parameter defaults to a value of 100. The range of values for this parameter is 1-10000. When this is set to a low value (1-10), the optimizer is more encouraged to perform index scans over full table scans.

The **OPTIMIZER_SEARCH_LIMIT** parameter defaults to a value of 5. When this is set to 1, the optimizer is totally discouraged from considering "*cartesian products*". With its default value, the optimizer can and will perform cartesian products for queries with 5 or less tables in the FROM clause. This behavior of performing cartesian products is suitable for small tables, but for obvious reasons is a NO-NO for large tables. Depending on the nature of the application this parameter needs to be tweaked.

The **OPTIMIZER_INDEX_CACHING** parameter defaults to a value of 0. The range of values for this parameter is 0-100. When this is set to 99, the optimizer is encouraged the use of nested loops over other join methods. Starting from 7.3, with 733_PLANS_ENABLED = TRUE and HASH_JOIN_ ENABLED = TRUE, the default value for this parameter will influence the optimizer to consider "hash joins" over all other joins. Hash joins are suitable for applications when small table(s) are joined with very large table(s), and a significant portion of the large table(s) is processed. Manually configuring this parameter to a value of 99 does not turn off hash joins, but does hold back the optimizer from going with hash joins as the default join method.

I-O Tuning

I-O tuning is a critical component of system tuning and it involves among other tasks, spreading hot files across multiple drives/volumes, employing optimal striping methodologies, identifying I-O-sub-system bottlenecks, identifying controller bottlenecks, and choosing the optimal level of RAID, based on the type of application.

I-O tuning should be done after gaining a full understanding of the workings of Oracle and studying the architecture of the Oracle RDBMS. I-O tuning should always be preceded and succeeded by I-O statistics monitoring, such as average service time, IOPS, average length of the disk queue etc. The paper entitled "Implementing RAID on Oracle Systems" provides detailed discussion on I-O configuration and its details by using real-life examples from production Oracle systems.

Contention Tuning

Most of the contention problems related to Oracle can be proactively managed by configuring the concerned initialization parameters. Improper configuration of the latch parameters in the init.ora can cause contention, but rather than blowing this out of proportion, the required parameters can be configured and proactively dealt with.

The relevant parameters to affect latch contention have been discussed in the "*Instance Tuning*" section.

Database objects such as tables, can have two points of contention. The first one is the number of "*freelists*" that are configured (default is 1). The freelist structure maintains the blocks in the table that are available for insert. It is relevant to configure this for those tables where a significant number of "*concurrent*" inserts occur. To proactively counter freelist contention, the parameter FREELISTS can be configured table creation. A value of (2 * # of CPUs) is considered optimal.

The second source of contention is relevant for indexes also, and it is the number of transaction slots configured in the object block header (default 1). Transaction slots are areas in the block header, where a transaction conceptually "*signs in*", using its transaction id, so that any rows that it modifies can be identified at the row-level using the slot number taken by a given transaction. If all existing transaction slots are currently utilized by other transactions, then the server process requiring a transaction slot, will allocate a new slot by borrowing 23 bytes from the block's PCTFREE. This is relevant for those objects that experience a high number of concurrent transactions. For transaction slot contention, the parameter INITRANS needs to be configured. A value of 4, for an 8K block size database is considered optimal in most cases.

Business today is performance, but...
Performance doesn't just happen.
Make it happen.
And make it *Precise.*

O-S Tuning

Tuning of O-S kernel parameters should be done per the guidelines defined in the Installation and Configuration Guide. Most O-S kernel parameters can retain their default values. There are some that do require attention and these are discussed in the following sections.

Kernel Tuning

Two key UNIX kernel parameters that require special attention are SHMMAX and SEMMNS. The former defines the size of the largest shared memory segment that the kernel will support and the latter defines the number of semaphores in the system. SHMMAX should be configured to support the largest shared pool area for any database. While having the entire SGA in one shared memory segment is good, the performance difference in having the SGA split across multiple shared memory segments versus 1 segment is less than 1%.

SEMMNS should be configured to support the maximum number of processes that will attach to all SGAs on a given machine. If a database needs to be support 1000 users, Oracle will need 1000 semaphores just to support the user population (not to mention the required semaphores for other background processes). It is useful to mention here that SEMMNS can be configured at (2 * # of processes on the system), to ensure that the system does not undergo semaphore starvation, as Oracle will procure the 1000 semaphores up front, regardless of whether a 1000 processes attached to the Oracle SGA or not.

It is worthy to mention here that, the upper bound for the file system buffer cache (which is usually configured by a kernel parameter), should usually be limited to 10-15% of total RAM, as failure to configure this parameter on some flavors of UNIX, can cause significant degradation in performance. On some flavors of UNIX the performance degradation can be noticed in the form of intense paging, process inactivation and swapping. This is because in some cases, default kernel parameters allow up to 50% of total RAM for the file system buffer cache.

Swap Tuning

While there is enough documentation to suggest that a system using an Oracle back-end should be configured with swap areas of (2-4 x Total RAM), common sense memory configuration (covered in the next section) suggests otherwise. If the system is configured with adequate memory and the various components are allocated optimally, the need for swap areas is very minimal. In many implementations, it has been observed that with optimum memory configuration, the amount of swap area required is really (1 x Total RAM). This is more relevant, if the SGA is pinned or locked in memory using the required initialization parameters. Ideally if things were configured right, there should be very little paging let alone swapping.

Conclusion

Managing Oracle Database Performance efforts requires adherence to a systematic methodology to ensure that all the core avenues and issues are addressed. Most of the issues can be proactively managed. Understanding O-S specific issues is one of the keys to success. Needless to say, the effort to configure systems with a balanced amount of hardware is of great importance.

So let's face the fact—80% of all system performance problems can be fixed by writing optimal SQL. This paper has attempted to provide you with all the core facts needed to cover the remaining 20%. Also, discipline needs to be exercised in setting reasonable tuning goals and ceasing all efforts when the tuning goals are achieved.

Knowing where one is going is important, but even more important is, knowing when one has reached the destination.

About the Author

Gaja Krishna Vaidyanatha works at Quest Software. His key areas of interests include performance architectures and application & database performance management for data warehouses and transactional systems. He can be reached via e-mail at gajav@yahoo.com. ▲

References

Using Stored Outlines in Oracle 8i for Plan Stability—By Probal Shome, System Performance Group, Oracle Corporation.

This article is excerpted and reprinted with permission from the IOUG-A Live! Conference, May 2000.

Deploying Your Applications to the Internet with the Oracle Forms Server R6i

Barry Hiern, Duncan Mills
Oracle Corporation

Introduction

This paper is designed to provide an architectural overview of some of the key points of the Oracle Forms Server.

Using the Forms server we can run complex applications over the Internet, without compromising on functionality or richness of interface. You can build new applications specifically for Web deployment, or even take your existing Forms Menus and Libraries that you have previously deployed in Client Server and move them to a three tier architecture almost without change. There are some restrictions inherent in using Forms over the web, and these are explained in the section Forms Server Restrictions below.

The architecture of the Forms Server is a little more complex than that of the conventional Client Server implementation of the product, so let's examine it in detail.

Components

The Forms server consists of a Java Client that is downloaded automatically to the end user and three components in the middle tier:

- ***Java Applet:*** When a user runs a Forms session over the web a thin 100% pure Java Client is dynamically downloaded from the Application Server. This Java client provides the user interface for the associated Forms Server Runtime Engine on the server, and handles user interaction and visual feedback such as that generated by navigating between items or checking a checkbox. This applet is the same for any Form that is run using it, you don't have to generate Java code for every application or form that you want to web deploy.
- ***Forms Runtime Engine:*** The Forms Server Runtime Engine performs the same function as the Client Server Runtime Engine would, except that all user interface functionality is redirected to the Java Client (above). The Forms Server Runtime engine is the process that maintains a connection to the database on behalf of the Java Client. The code that will be run by the Forms Server Runtime engine are the same Forms, Menus and Libraries files that would be used for running in Client-Server mode on the same platform. No conversion or re-compilation is required to web-deploy an application!
- ***Forms Listener:*** The Forms Server Listener acts as a broker, taking connection requests from the Java client processes and initiating a Forms Server Runtime process on their behalf. The listener can also maintain a pool of running engines ready for connection, making the connection from the Java Client complete as quickly as possible.

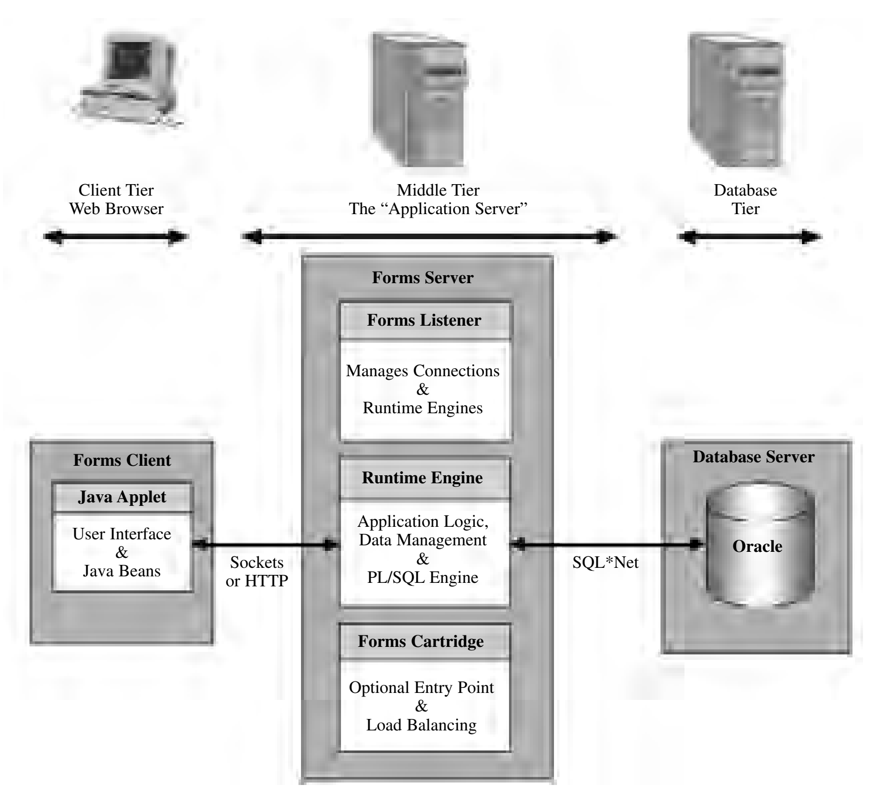


Figure 1. Running a Form on the Web.

- ***Forms Cartridge*** (*optional*): The Forms Cartridge is only used when the Forms Server is configured to run as a "Cartridge" under the Oracle Application Server. It is used as a way to provide a single entry point to many different Forms applications and as a way to administer load balancing.

Connection Process in Detail

So we've looked at the components that make up the Forms server, now to look at the connection process itself. The following diagram outlines the steps taken to establish a Forms session with the Forms Server.

1. The user chooses a Link from a web page, types a URL directly or passes the URL as an argument to the appletviewer.

2. The Web Server interprets the URL that is passed and serves up an HTML page containing an <APPLET> tag which describes the Forms Java client to the Browser. The HTML page may be a *static* HTML page that has all of it's parameters hard-coded into it, or the URL that is passed could call the Forms Server Cartridge if the Oracle Application Server is being used (See Static or Cartridge? for an explanation of the specific processes involved.)

3. The Client receives the HTML file served by the Web Server. The <APPLET> tag in the HTML file will supply the information required to locate the Java Class files that make up the Forms Java Client. Within the APPLET tag in the HTML file you would also supply information about the Form that should be run, and any other parameters that you want to pass to your Forms session, such as the Login information. The APPLET definition also contains instructions on what Forms Server to run and many parameters which can help you to customise aspects of the Java Client such as the Look and Feel, colour schemes etc. The HTML file might also contain other HTML tags such as those to tell the browser to run this particular applet using the Jinitiator Plug-in (see JInitiator).

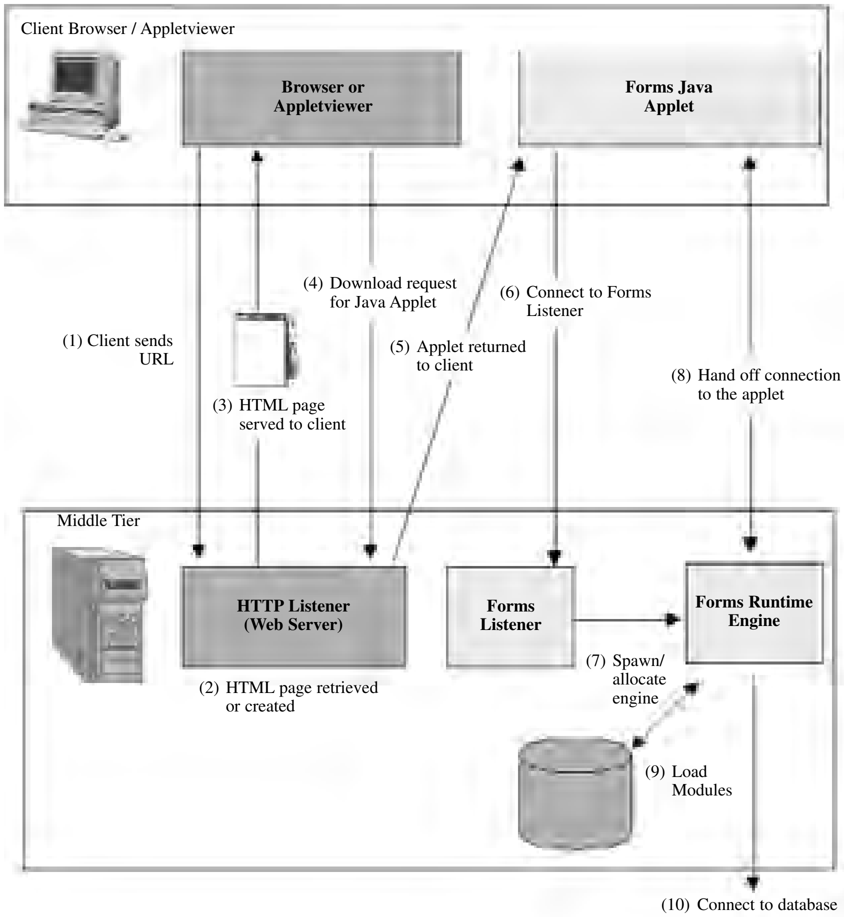

Client Browser / Appletviewer
Browser or Appletviewer
Forms Java Applet
(1) Client sends URL
(4) Download request for Java Applet
(6) Connect to Forms Listener
(5) Applet returned to client
(3) HTML page served to client
(8) Hand off connection to the applet
Middle Tier
HTTP Listener (Web Server)
Forms Listener
Forms Runtime Engine
(7) Spawn/ allocate engine
(2) HTML page retrieved or created
(9) Load Modules
(10) Connect to database

Figure 2. Forms Connection Process in Detail.

4. The Browser asks the Web Server for the Java Class files from the location specified in the HTML file. The APPLET CODEBASE parameter in the HTML file is used to define this. The files may be downloaded individually or as an "Archive". This archive will have an extension of .JAR and can be best thought of as a .ZIP file containing all of individual .CLASS files required by the Applet. The use of a JAR file speeds up the download of the Java client. The ARCHIVE parameter defines which (if any) .JAR file should be used. If this particular HTML file specified that JInitiator should be used, then JInitiator will carry out the additional step of checking the version of the Forms Client Java code available on the Web Server and will only download it if it turns out to be newer that any version that JInitiator currently has cached.

5. The .CLASS or .JAR files are downloaded (if not already present) to the Browser and the Java applet starts.

6. The Java client applet connects to the Forms Server Listener by talking to the TCP/IP Socket defined as the *serverPort* parameter in the APPLET Tag. The Forms Listener will have already been started and will be listening for incoming requests on this particular socket. The Listener must be on the same machine that the Java Client was downloaded from.

7. Upon receiving the connection request from the Java client, the Forms Listener will start a new Forms Runtime engine for this client. You can configure the Forms Server to

hold a pool of unconnected Runtime engines which can be allocated as connection requests come in, thus speeding up the whole connection process. The Runform engine is allocated a new socket, which it will use to talk to the Forms Java Client.

8. The Forms Server Listener hands off the connection to the Java Client and has no further part in the process.

9. Now the Forms Server Runtime Engine allocated to this client takes over, loading the module specified in the serverArgs entry in the HTML file, plus, of course, any libraries and menus that are required by that form.

10. The user is prompted for their database login information, if this had not already been supplied, and the connection to the database server established. The user is now ready to work.

Forms Server Load Balancing

The Oracle Forms Server provides the facility to load balance, across a number of servers on the middle tier, each of which would be running a Forms Server.

Load Balancing will provide you with a much more scaleable configuration. When you are approaching the limits of your current hardware, rather than having to either upgrade or throw out that machine, you can just add more nodes in to run your application and spread the increasing load across several machines rather than one.

When users connect to a server (See below) that is set up for load balancing two extra components of the Forms Server are utilized.

- ***Forms Load Balancing Server:*** The Load Balancing Server , also known as the Metrics server, maintains information about all of the machines currently allocated to the "Server Farm" and what their current loading is.
- ***Forms Load Balancing Client:*** The Load Balancing Client, also known as the Metrics Client, runs on each of the machines allocated to be in the "Server Farm" its job is to simply report back to the Metrics Server with Loading information i.e. the number of Forms Processes that are currently running on that particular machine.

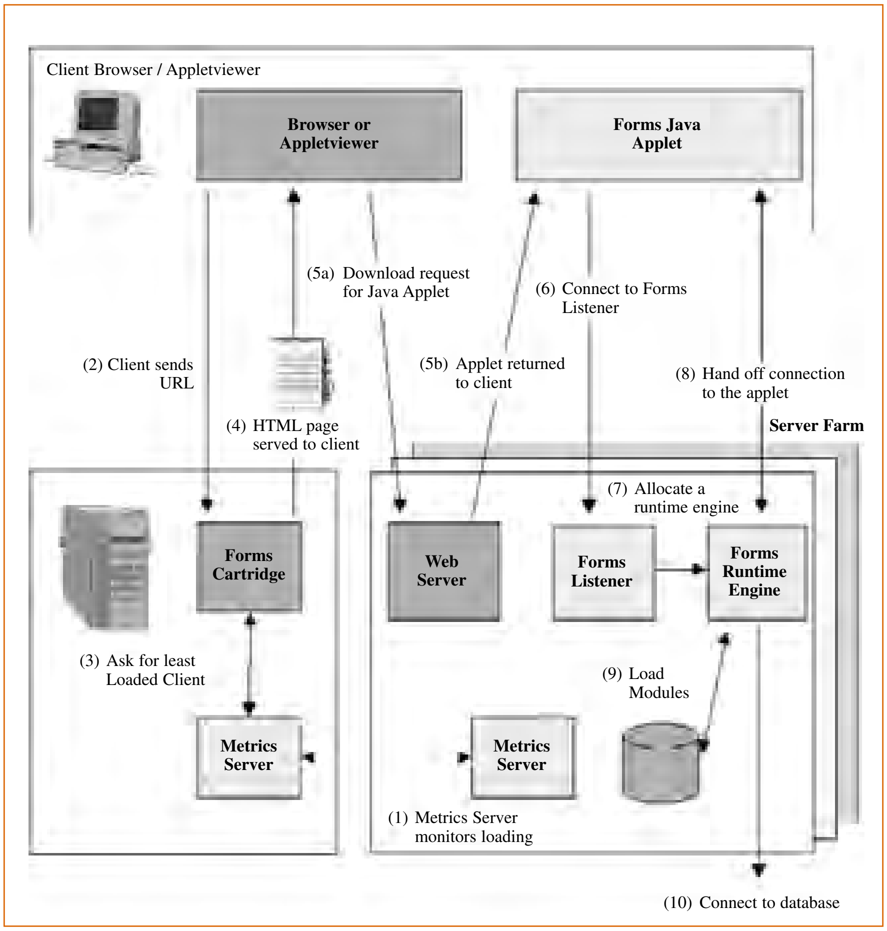

Client Browser / Appletviewer
Browser or Appletviewer
Forms Java Applet
(5a) Download request for Java Applet
(6) Connect to Forms Listener
(2) Client sends URL
(5b) Applet returned to client
(8) Hand off connection to the applet
(4) HTML page served to client
Server Farm
(7) Allocate a runtime engine
Forms Cartridge
Web Server
Forms Listener
Forms Runtime Engine
(3) Ask for least Loaded Client
(9) Load Modules
Metrics Server
Metrics Server
(1) Metrics Server monitors loading
(10) Connect to database

Figure 3. Running a Form in a Load Balancing Scenario.

When a user connects to the server in a load balancing situation, the sequence of events to create a Forms session is slightly is slightly different.

1. The Metrics Server is continually apprised of the process loading on each of the Metrics Clients on machines within the Server Farm.

2. In this case the URL supplied by the user is interpreted by the Oracle Application Server as a request for a Cartridge and the parameters on the URL are passed through to the Forms Server Cartridge to handle.

3. The Forms Server Cartridge will already have been configured to know that it should be load balancing and will ask the Metrics Server for the member if the Server Farm that currently has the lowest number of Forms Server processes running. If your Server Farm consists of a mix of different hardware platforms or configurations then scaling factors can be applied when starting the Metrics Clients. This scaling factor will allow you to skew the loading in favor of more powerful machines, rather than the choice of least loaded server being determined solely by the number of processes running.

4. Having determined the machine in the Server Farm that is best able to handle this new connection, the Forms Cartridge will build an HTML file pointing the Browser at that server for the download of the Java Applet and connection to a Forms Server Listener.

Convergence
Two come together as one.
VERITAS enhances the power of Oracle.
ORACLE
SOFTWARE POWERS THE INTERNET

5. The user's browser will now download the Forms Java Applet from this dynamically determined location, and thereafter the rest of the connection process will take place to this machine.

Load Balancing Futures

In the 6i release of Oracle Forms Server we will be removing the requirement to use Oracle Application Server to facilitate load balancing. Although this will still be an option, we will be supplying a CGI implementation of the Load Balancing software.

Using this CGI implementation you will be able to use *any* webserver (that supports CGI) to host your load balancing.

Static or Dynamic HTML?

When you come to configure the Forms Server you will be faced with the choice of using a "Static" implementation, a "Dynamic" implementation or both, but what does this actually mean? The choice of a Static or dynamic implementation only effects Step 2 in the Forms Startup process.

The so called Static implementation simply means that, the URL that is input into the browser by the end user, will contain the name of an HTML file, within which, will be all of the information required to download the Java Client and start the connection to the Forms Server Engine. This implies that for every entry point into your applications you would have to have a separate HTML file with the relevant Form names and parameters hard-coded into each. As we are just dealing with a plain HTML file here, you can use a static implementation, no matter what Web Server you happen to be using.

The dynamic implementation on the other hand gives you more flexibility, allowing for the creation of the HTML on the fly. In Release 6i this functionality is available through both a CGI interface, an OAS cartridge and in the near future will be available via a Java servlet.

In the case of this implementation, the URL that the end user types in (or clicks on) does not actually point at a real HTML file. Rather it points at an Oracle Application Server Cartridge. When the Oracle Application Server receives such a URL for servicing, it will pass the request on to the Forms Server Cartridge which will then dynamically construct an HTML file which is then passed back to the browser. The Forms Server Cartridge will combine information passed in on the URL, with information entered by the systems administrator in the definition of the cartridge itself, and a template HTML file (which you can use to define a common look and feel for all of the applications that you would be running).

At present if you want to take advantage of the load balancing facility within the Forms Server then you will need to use the cartridge implementation for that.

Communications and Security

When running the Forms Server, the Java Applet implementing the user interface and the Forms Server engine need to communicate. This can be done using a conventional TCP/IP Sockets connection for Intranet Use, or using HTTP/1.1 as a protocol for Internet use where Firewalls prevent the use of a direct socket connection.

In either case, all of the traffic that is transmitted between the Client and Forms Engine is encrypted using 40-bit RSA encoding. This encryption can be turned off optionally, but we would not recommend that sites do this.

For more information on the use of HTTP with Oracle Developer Forms Server and security/firewall issues generally see the Oracle White Paper: *Deploying Internet Applications using HTTP Enabled Oracle Forms Server*

Security constraints imposed by Java

As the thin client portion of the Forms Server is written in Java, it has to comply with certain restrictions that Java Applets[1] have imposed upon them. The first thing that you would notice when running a Forms application over the Web is that the Windows have a bright yellow bar across the button bearing the legend "Warning Applet Window". This is Java's way of reminding you that you downloaded this program from someone else who you can't necessarily trust!

As well as the warning you'll find that the Java Virtual Machine prevents you from carrying out simple tasks such as printing the screen or copying data onto the clipboard. These constraints can of course be a little too restrictive, when you do infact, know where the code has come from, which is true in the case of Oracle Developer. Therefore you can choose to "Trust" Oracle Developer on your machine, by registering a Certificate that we supply, with the Java Virtual Machine. Then, when you download an applet which claims to be the Forms Server Applet, the JVM will check its credentials using the stored certificate information, and if it all matches, the applet will be trusted and able to print, use the clipboard and of course you don't get the bright yellow warning any more!

Performance and Scalability

A lot of work has been done with the Developer Forms Server to make it

[1] It is important to note the difference between a Java Applet and a Java Application. As an Applet is by definition downloaded off of the Web, the Java virtual machine that runs in the web browser or appletviewer strictly controls what the applet can do. This is termed the sandbox, applications running within the sandbox do not have access to objects outside of it such as memory, the local hard drive, printers etc. So if you access a web page that attempts to download a rogue Java application onto your machine, it won't be able to escape from the sandbox and therefore won't be able to damage your machine. In the case of a Java Application, this is where the Java code is sourced from your machine and therefore is implicitly trusted to behave responsibly with your hard-disk etc. in the same way that any normal 'C' program would be.

as performant as possible from end to end. Giving you the kind of performance that you would expect from a Client-Server implementation, but with the vastly reduced administrative overheads of Web Deployment, and no client software installs!

Details of how to leverage the maximum scalability and performance from the Forms Server can be found in the White Paper: *Oracle Developer Server: How to Optimize the Deployment of Internet Applications.*

Additionally if you would like some more "Real" World" information about Oracle Forms Server scalability and performance refer to the following White Papers:

- *Scalable Web Deployment with Oracle Developer Server — A Benchmark Comparison of Client/Server and Web Deployment by Retek Information Systems (December 1998)*
- *Developer Server Scalability Testing (December 1998)*

JInitiator

Oracle JInitiator is Oracle's version of Sun's Java Plug-In, which provides the ability to specify the use of a specific Java Virtual Machine on the client instead of using the browser's default JVM. Oracle JInitiator runs as a plug-in for Netscape Navigator and as an ActiveX component for Internet Explorer; this allows customers to run Oracle Forms Server applications using Netscape Navigator or Internet Explorer.

The Oracle JInitiator provides these major benefits:

- It allows the latest Oracle-certified JVM to run in older browser releases.
- It ensures a consistent JVM between different browsers.
- It is a reliable deployment platform. Oracle JInitiator has been thoroughly tested and certified for use with the Oracle Forms Server
- It is a performant deployment environment. Application class files are automatically cached by the JInitiator, providing fast application start-up.
- It is a self-installing and self-updating deployment environment. Oracle JInitiator automatically installs and updates itself like a plug-in or an Active-X component. Local cached application class files will be automatically updated from the application server based on a date-time stamp comparison.

Forms Server Restrictions

Although The Forms server allows you to deploy your existing applications without major changes there are some restrictions imposed by the three tier architecture which you will need to take into account:

- OCX, and VBX controls are not supported. As the user interface is not on the same machine as the runtime engine, there is not way for the control to display. As version 6.0 of Developer Forms Server supports the use of Java Beans, you can replace the functionality provided by the Windows Specific control with a similar Bean
- User_Exits, ORA_FFI and Host commands all execute on the middle tier, not on the Java Client. This implies that they cannot carry out any user interface interaction as the display being used by the user is not the same display as the middle tier.
- TEXT_IO. Reading and writing files using the Text_io package will again take place in the middle tier. You cannot read a "Local" file off of the end users machine, unless the user's drive is shared and visible from the Forms Server machine, or alternatively you use a Java Bean.
- Mouse Move, Mouse Enter and Mouse Leave events. The Triggers that would normally be executed by Forms in a client server deployment have been disabled when running over the web. This is to prevent the network being swamped with mouse messages between the Forms Java Client and the middle tier.
- Local devices, Customers often ask if they can utilise devices such as Bar-code scanners attached to their browser machines and integrate them with Forms running over the web. By default this is not possible as there is (as was mentioned above) no way of programmatically accessing the browsers local machine from within the Java Applet. It is possible to implement this functionality, however, by writing a trusted Java bean which can be embedded into the Forms application and access the local operating system. ▲

This article is reprinted with permission from the IOUG-A Live! Conference, May 2000.

Oracle8i Indexing Choices: Best of Breed

John Jay King
King Training Resources

Abstract

Oracle8i provides expanded options for indexing. Index choices include: traditional B-Tree indexes, Reverse-key indexes, Bit-map indexes, Hash indexes, and Index-Organized Tables. In addition, indexing is now available when using a function or expression on an index column. Choosing the proper index style is an important part of database and application design. Significant performance differences can be achieved by the careful creation of indexes and their subsequent use in applications. This session compares and contrasts the various options available and how to choose from among them. Specific topics include: B-tree indexes, Reverse key indexes, Bitmap indexes, Hash indexes, Index-Organized Tables, the ability to base indexes on functions or expressions, negative impacts of indexes, and hints involving indexes. Attendees will be better equipped to build effective Oracle8i applications.

Introduction

Choosing the appropriate type of index may sometimes be as important as the choice to use an index or not. In the past indexing decisions were largely a matter of what columns to index. Today, we need to explore the options further and also decide how columns should be indexed. The type of index selected may impact performance dramatically. Oracle8i provides a comprehensive set of indexing options that address most application needs. Most of the information in this paper also applies to Oracle8 and to a lesser-extent Oracle 7.3. Notations in the text indicate features that require Oracle8i.

Indexes are traditionally used to guarantee unique values (if defined as unique) and to speed performance by using a key value's rowid to access data rather than a row-by-row search of the tablespace. However, changes to table data that includes index keys can multiply I/O times several times; slowing the speed of inserts, updates, and deletes to keep indexes synchronized.

Types of Indexes

Oracle8i offers different types of indexes: B-Tree (traditional) indexes, Hash-cluster indexes, Bitmap indexes, and Index-Organized Tables. The B-Tree index offers the Reverse-Key variation and both B-Tree and Bitmap indexes allow Function/Expression-based indexes.

Choosing the appropriate indexing method is frequently an important performance-related question. Except for Index-Organized Tables, Oracle's indexes provide a means of translating a key value (one or more columns in the database row) into a rowid. Index use reduces the I/O required to obtain a row to: the I/O or calculations necessary to find the rowid, followed by a direct access using the rowid. This is usually faster than reading all possible rows looking for a match (table scan).

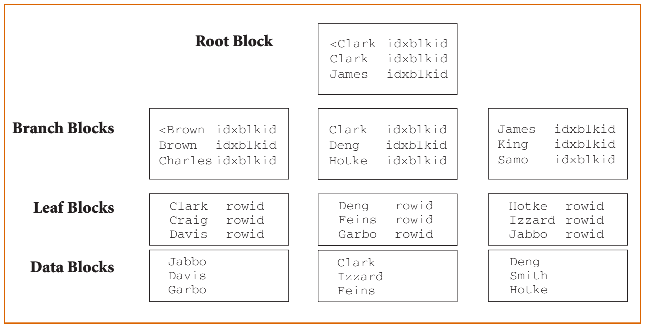
B-Tree indexes store key values sequentially and are traversed from: root block, to branch block (sometimes multiple levels of branch blocks), to leaf block, to data block containing the row. Hash-cluster indexes convert the key value using an algorithm to determine which data block to read. Bitmap indexes contain a bit (0 or 1) for each key value that corresponds to every rowid in the table. Index-Organized Tables actually contain the table row data, so, once an index value is found in the index, the data is immediately available. The different index types are discussed in more detail later in this paper.

B-Tree Indexes

The original type of index supported by Oracle is the B-tree index. These indexes provide a linked-list type access using a file that contains the key values in sequence, with the rowid. In the case of non-unique indexes, multiple rowids might be stored for a given key value. B-tree indexes are a good choice when a given key value is unique or has few duplicates in a table.

A B-tree index is stored as a set of pages in a hierarchy. The first index page is called the "root" and it points to lower-level "branch" index pages. Multiple levels of "branch" pages might exist depending upon the size of table and index keys. The lowest level index page is called a "leaf" page and it contains the address of data blocks containing table rows. The actual key value is stored (unless using key compression) in the various levels on index. If indexes are based upon multiple columns (concatenated keys) the larger index entries can increase the number of levels necessary to address a table's data. Some organizations avoid this issue by assigning key values (sometimes using a sequence generator) that correspond to a concatenated key value. This way, the index entries are smaller, resulting in fewer levels in the index, resulting in faster access (not to mention simpler SQL statements!). Rows with NULL key values are not represented by the B-tree index. Some organizations use a null-capable column as a switch to gain faster access to values that are not null.

Each index requires overhead when updating, deleting, or inserting rows in the base table. If a table is read-only, this additional overhead is avoided. If a table is volatile, it is a good idea to limit the number of indexes that must be maintained.



In the example above, a search for "Garbo" would first read the root block, then the branch block beginning with "Clark" and finally the leaf block beginning with "Deng" to obtain the rowid of the "Garbo" row. This means that the record is retrieved with four block reads (at most in this example) rather than reading each data block in the table.

Index rows are stored in sequence according to key values (as above), however, data is not necessarily sequenced. The Clustering Factor statistic is used by the optimizer to determine whether rows of subsequent reads are likely to be on pages already in the cache.

A "Fast Full Scan" is sometimes selected by the optimizer so that rather than traversing the index hierarchy the Leaf Blocks are read using the same multi-block I/O used by table scans.

Key design can cause some key ranges to become more dense than others, especially when key values are geographic in nature. When key designs lead to index paths being more densely or sparsely populated than other index paths, insert or update of key values can lead to deeper levels of index than would otherwise be necessary. An index "tree" that is unbalanced may inhibit performance of the "Fast Full Scan," rebuilding the index will redistribute the levels. Another solution to the problem of unbalanced key distribution may be the Reverse-Key index.

Reverse Key Indexes

If keys and usage are heavily clustered in a table (e.g. key is generated as next sequential number, and most activity occurs on recent entries; or geographically-oriented and many records are in a specific area), then, Reverse Key Indexes might speed things up. A good example of a geographically oriented key is the Social-Security Number used for taxpayer identification in the United States. This number's first three digits were originally designed to show the region of the country where the person resided when the number was issued. In systems that service the eastern coast of the US, the social-security numbers of New York-based values might skew the indexes of a heavily inserted table using Social-Security Number as its key.

A Reverse Key Index is simply a standard index with the key values stored in reversed form (e.g. '1234' becomes '4321', '1235' becomes '5321'). Actual table data is not changed. By reversing the key values the index blocks might be more evenly distributed reducing the likelihood of some index paths being densely or sparsely populated. Oracle says this is especially useful in Parallel Processing environments but it is also potentially useful any time key ranges become too dense and cause extra processing due to the key density. Be sure to carefully test Reverse-Key indexes to verify that real benefits are occurring.

Syntax to create a Reverse-Key index is to simply add the word REVERSE after the column specification:

```
CREATE INDEX employee_ssn_rev
       ON employee_table (ssn) REVERSE
       /* … rest of index definition … */;
```

Function/Expression-based Indexes (Oracle8i)

Normally, developers attempt to use index columns wherever possible to speed access to large database tables. Until now, this has often been hampered by limitations of index design. Until Oracle8i, index columns always represented the actual value of the column in the database and an SQL WHERE clause needed to specify the original column value or the index would be ignored. Now, an index can represent a column value after some function or expression has been applied. As long as an SQL WHERE clause references a column value after a function or expression exactly as specified (case-insensitive and blanks are ignored) in the index creation, an index may be used by the system. Given the following create index lines:

```
CREATE INDEX … ON EMP (UPPER(ENAME)) …
CREATE INDEX … ON INVENTORY (IN_STOCK + ON_ORDER) …
```

The following two WHERE clause lines may be indexed by the optimizer (requires cost-based optimization):

```
SELECT … FROM EMP WHERE UPPER(ENAME) = UPPER (:hostvar) …
SELECT … FROM INVENTORY WHERE IN_STOCK + ON_ORDER > :LARGE_QTY_ITEMS …
```

These indexes will only be used if cost-based optimization is invoked and if statistics are available on the function/expression-based index. In addition, the query must be rewritable to use Oracle's built-in functions and it must be trusted in order to use user-defined functions:

```
ALTER SESSION SET QUERY_REWRITE_ENABLED = TRUE;
ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED;
```

Bitmap Indexes

When creating B-Tree indexes, columns with few values over many rows (low-cardinality) should be avoided. This means that columns like the Country Code of a customer or Gender of a customer would make poor index columns due to the small number of different values in the table. Bitmap indexes offer performance improvement when created for columns with a relatively small number of values.

When a bitmap index is created, the index contains the key value and a bitmap listing the value of 0 or 1 (yes/no) for each row indicating whether the row contains that value of not.

For an index of customers in a particular country (presume that our organization only does business in the United States (US), United Kingdom (UK), Japan (JA), and Australia (AU)):

COUNTRY	COUNTRY	COUNTRY	COUNTRY
_CD=AU	_CD=JA	_CD=UK	_CD=US
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
0	0	1	0
0	0	0	1

Each entry in the bitmap corresponds to a row in the table, a value of 1 indicates which value that row contains. Bitmaps include all rows, even those with NULL values (unlike B-Tree indexes). Bitmap indexes are significantly smaller than B-Tree indexes for the same table.

Bitmap indexes may only be used for equality-type tests (= or IN). Bitmap indexes are ideally used in conjunction with other indexes to reduce the number or rows returned. The following query will be much quicker given bitmap indexes on the low-cardinality columns COUNTRY_CODE, GENDER, and CREDIT_CARD.

```
SELECT CUSTOMER_ID, LAST_NAME, BALANCE
       FROM NON_GOV_CUSTOMERS
       WHERE COUNTRY_CODE IN ('AU','UK')
             AND GENDER = 'M'
             AND CREDIT_CARD = 'AX';
```

Bitmap index maintenance can be quite expensive. Since an individual bit may not be locked, a single update locks potentially large portions of the index. It is best if Bitmap indexes are used primarily in read-only situations like data warehouses or where concurrent transactions are unlikely.

Hash-Cluster Indexing

In B-Tree and Bitmap indexes, the key value is used to find rows that match. This process requires I/O to process the index before obtaining the data row(s) desired. A hash cluster determines how to find a row by applying an algorithm based upon the key value, the only read is to get the data row. Rows are stored together based upon their hash value.

The hashing algorithm may be specified to use Oracle's internal algorithm, to use the cluster key as the hash function, or to use a user-defined hash function. Hashing is not new to the computing industry, entire books have been devoted to developing optimal hash algorithms.

Hash clusters can be the fastest type of index given that: very-high-cardinality columns are used for the index, only equal (=) tests are used, index values do not change, the number of rows and rows/index value are known at index creation time, and only minimal insert/delete activity will occur. Only one Hash-cluster is allowed per table. The size of the hash index should be known at index creation and it should allow for distribution of rows with few collisions when hashing a specific key (ideally no collisions). If each key hashes to a unique value optimal results will occur. If many keys hash to the same value (a collision) then chaining is likely to occur reducing the benefit of the Hash cluster. There is no graceful way to reorganize a Hash-cluster, the index must be dropped and recreated.

Index-Organized Tables

Beginning with Oracle8, CREATE TABLE has an ORGANIZATION INDEX clause that causes the table's data to be incorporated into the B-tree index representing the table's Primary Key (and only the primary key). This means that the table's data is always available in sequence by Primary Key and many sorts can be avoided by the optimizer. Oracle8i adds the ability to create secondary indexes for Index-Organized tables (not allowed in Oracle8.0). Index-Organized Tables are especially useful for "lookup" type tables that are often used to create drop-down lists in GUI interfaces; since table data is already in the desired sequence, access is faster.

The advent of Index-Organized Tables has caused the new term "heap organized" to describe traditional tables.

A sequential scan of an index-organized table yields all values in sequence by primary key.

The advent of Index-Organized Tables has caused the new term "heap organized" to describe traditional tables.

The entire Index-Organized Table is stored in the index and has no specific rowid. Instead, Oracle uses a "virtual rowid" to provide secondary indexing capability. This is quicker than a scan of the Index-Organized Table, but, not quite as fast as a traditional B-Tree secondary index.

Index-Organized Tables work best when there are few columns in the table/index (a "narrow" table) and the size of a row is small compared to the size of a block. Index-Organized columns may not contain LONG columns, but may contain BLOB, CLOB, or BFILE data. Index-Organized Tables may not be used in a CLUSTER.

Comparing Index Strengths and Weaknesses

For high-cardinality key values, B-tree indexes and Hash-cluster indexes may offer the best alternative. B-Tree indexes work with all types of comparisons and gracefully shrink and grow as table data changes. Hash-clusters work only with equal tests and table growth is a significant problem.

For low-cardinality key values that are not changed by concurrent transactions, Bitmap indexes are often superior to B-tree indexes. Hash-clusters are not a good choice for low-cardinality data (many collisions).

If a key design causes dense or sparse population of index values, it may be good to test using Reverse-key indexes to see if overall performance is improved.

Index-Organized tables are a good choice if tables have few columns, have relatively small rows, and are frequently sorted by Primary Key.

Using Hints to Suggest Indexes

Sometimes, the optimizer may not choose to use an index, or might not choose to use it in the way you intended. In this case you might want to consider using Hints to control the processing of the SQL. Through careful testing of hints, it is sometimes possible to improve on the optimizer's decisions. Be sure to use trace information in addition to Explain output when determining whether or not a hint is useful.

Be careful! Test statements thoroughly before and after adding Hints, your "help" might make things worse (oops!). Be aware that buffering of indexes and data may make the choice of index style less important in some instances.

Revisit decisions to use Hints regularly (at least as often as installations of Oracle releases, probably more frequently). A performance improvement due to hints today, using today's data, in today's environment might not be a good thing tomorrow.

User-Defined Index Types

Oracle8i provides the ability to create a user-defined index type to provide indexing of complex data such as documents, images, video clips, spatial data, or audio clips. Creation uses the object features first introduced in Oracle8 and allows creation of indexes specifically designed for complex applications such as On-Line Analytical Processing (OLAP).

User-defined indexes may be used in conjunction with user defined operators (CREATE OPERATOR).

Specifics concerning User-Defined Index Types may be found in the Oracle8i Concepts manual, no further discussion is provided in this paper.

Conclusion

This presentation attempted to illustrate the indexing capabilities of Oracle8i and to help system developers when deciding not just what columns to index but how to index them. This paper presented various indexing options available and suggested when choosing a particular type of index might be the best choice. As with all performance related issues, test, test, and test again. Just because something appears on paper to be a good choice does not mean that it is best for you. Also, any performance oriented decision must be revisited periodically to make sure that the best choice is still being made. ▲

John King is a Partner in King Training Resources, a firm providing instructor-led training since 1988 across the United States and Internationally. John has worked with Oracle products and the database since Version 4 and has been providing training to application developers since Oracle Version 5. He has presented papers at various industry events including: IOUG-A Live!, UKOUG Conference, EOUG Conference, ECO, RMOUG Training Days, and the ODTUG conference.

John Jay King
King Training Resources
6341 South Williams Street
Littleton, CO 80121-2627 U.S.A.
Phone: 1.303.798.5727 1.800.252.0652 (within the U.S.)
Fax: 1.303.730.8542
Email: john@kingtraining.com Website: www.kingtraining.com

Copies of this paper are available for download from the King Training Resources website (www.kingtraining.com).

Bibliography

Oracle8i SQL Reference, Oracle Corporation
Oracle8i Concepts, Oracle Corporation
Oracle8i Administrator's Guide, Oracle Corporation

This article is reprinted from the IOUG-A Live! Conference, May, 2000.

NoCOUG Spring Conference

Session Descriptions: Wednesday, May 16, 2001

See back page for full schedule.

DBA Track

10:30a.m.
New Features in the Oracle9i Database

Michael Hartstein, Senior Director of Server Technologies, Oracle Corporation

This session provides an overview of the enhancements in the Oracle9i database. Enhancements which increase and improve availability, scalability and performance, security, and manageability are discussed. Also, new features in the areas of business intelligence, application development, content management, and B2B and B2C application integration are covered.

1:45p.m.
Tuning Database Locks and Latches

Hamid Minoui, Oracle DBA, Fritz Companies

In this session Hamid will start with a quick overview of the role locks and latches play in the database server. Then he will present techniques for diagnosing latch and lock contention problems, and how to correct them. Along the way he'll introduce pertinent V$ views, X$ tables, and initialization parameters.

3:15p.m.
Oracle9i Real Application Clusters: Scalability for the Real World

Merrill Holt, Director of Cluster and Parallel Storage Technology, Oracle Corporation

Traditional cluster databases require significant application modifications to exploit cluster scalability for the back-end database. This is not practical for most applications and when possible, proves to be very costly. Oracle9i Real Application Clusters (the next generation of Oracle Parallel Server), with its "shared-cache" architecture, solves this problem for all applications. The presentation will highlight the technology and present proof-points that back up the claims.

Developer Track

10:30a.m.
JDeveloper9i — A Sneak Preview

Christophe Job, Senior Director of Java Tools Development, Oracle Corporation

1:45p.m.
Text in Oracle

Omar Alonso, Senior Product Manager, and Stefan Buchta, Oracle Corporation

This session will explore why text is important in a number of intranet and internet applications and how Oracle deals with text. The session will include demonstrations, including one that uses Ultra Search. For the DBAs in the audience, we will have information on administering text indexes and tuning SQL queries that involve text.

3:15p.m.
Java in the Oracle9i Application Server

Margaret Mei, Product Manager, Oracle Corporation

This presentation will update the audience on Java capabilities in the middle-tier. Topics discussed will include deploying Java Servlets, JSPs, and EJBs on Oracle9iAS.

Data Warehouse Track

10:30a.m.
Managing the Virtual Enterprise with Metrics

Arvindh Balakrishnan, CEO, MetricStream

Most companies have solid business initiatives, but few have the platform in place to successfully drive them. Participants will learn how to successfully accomplish business initiatives, minimize operational disruptions, and better manage and drive business operations, all while realizing the value of existing enterprise applications.

1:45p.m.
Components of a Successful Data Warehouse

Chris Wheaton, Co-Founder, BASE Consulting Group

3:15p.m.
To Be Announced

RESOURCE CORNER

Are you always on the lookout for additional Oracle Resources online? In each edition of the NoCOUG Journal, we'll point out a few noteworthy websites.

For over four years, **RevealNet** has published its handy (and free!) tips on its website through its Pipelines. Each month, RevealNet's Pipelines give a quick helpful hint. There is an Oracle DBA pipeline and a PL/SQL Pipeline for developers. Check it out at: www.revealnet.com/Pipelines/pipelines.htm.

Underground **Oracle FAQs** at www.orafaq.org/faq.htm is an extensive and comprehensive resource site. It contains scripts, white papers, Oracle-related news and discussion forums. You can even find resources to help you pass OCP exams!

Lots of discussions go on at the **LazyDBA** website. Look for the Oracle Scripts section where you'll find scripts on backups, day-to-day monitoring, and more. It's at www.lazydba.com.

Send your favorite resources to journal@nocoug.org for publication. ▲

NoCOUG Spring Conference

Wednesday, May 16, 2001

Location: Lockheed Martin

Building 157 at 3rd Avenue and J Street, Sunnyvale

Please note that Lockheed Martin has special security requirements: All attendees must bring government-issued photo identification (such as a driver's license). Resident Aliens must also bring their green card, and Foreign Nationals on work visas must bring their passport.

Our Spring Conference takes place at Lockheed Martin, Building 157, at 3rd Avenue & J Streets in Sunnyvale.

For session descriptions, check out page 27.

8:30-9:30	**Registration and welcome. Refreshments served.**
9:30-10:15	**Opening Remarks and Announcements / Vendor Introduction**
10:15-10:30	**Morning Break**
10:30-11:30	**Sessions**
	DBA Track: New Features in the Oracle9i Database
	Developer Track: JDeveloper9i — A Sneak Preview
	Data Warehouse Track: Managing the Virtual Enterprise with Metrics
11:30-1:45	**Lunch Break**
1:45-2:45	**Sessions**
	DBA Track: Tuning Database Locks and Latches
	Developer Track: Text in Oracle
	Data Warehouse Track: Components of a Successful Data Warehouse
2:45-3:15	**Afternoon Break**
3:15-4:15	**Sessions**
	DBA Track: Oracle9i Real Application Clusters: Scalability for the Real World
	Developer Track: Java in the Oracle9i Application Server
	Data Warehouse Track: To Be Announced
4:30-???	**Reception/No Host Bar—Location to Be Announced**
	Cost: $40 admission fee for non-members. Includes lunch voucher.

Check NoCOUG's website for more information and last minute changes prior to the meeting at www.nocoug.org

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

NoCOUG
P.O. Box 3282
Danville, CA 94526