TechJournal

New York Oracle Users Group

Fourth Quarter 2008

Fourth Quarter General Meeting

Tuesday, December 10, 2008
St. John's University – Manhattan Campus
101 Murray Street

Sponsored by:
All Star Software Systems and Fusion-io

Free for Paid 2008 Members
Don't Miss It!

In This Issue –

ACCELERATE Your Back Office Processes!

Eliminate Data Entry, Retrieve Documents in Seconds and Automate the Approval and Exception Processing of Transactional Documents

All Star Software Systems is an *Oracle Certified Partner* specializing in business process automation. All Star is a *full service* organization who sells, implements and supports all of the software and hardware included in every solution. Our *passion* for success is only exceeded by the *success* of our implementations for our customers.

ORACLE PARTNER

Solutions

- Invoice Processing
- Expense Report Processing
- Sales Order Processing
- Credit Memo Processing
- Check Remittance Processing
- Invoice Delivery
- Mailroom Document Routing

Technology

- Workflow
- Information Capture
- Data Transformation
- Content and Document Management
- Paper and eForms Processing
- Enterprise Faxing
- ERP Report Output Management

- All Star provides solutions that automate business processes. Our solutions capture inbound information (paper, fax, email, reports, eForm, EDI), extract pertinent data, allow users to retrieve it instantly and automate the approval and exception processing of that information.

ALL STAR SOFTWARE SYSTEMS

www.allstarss.com

***ACCELERATING* Business Processes!**

440 Smith Street Middletown, CT 06457

860 613 1500

NYOUG Officers / Chairpersons

ELECTED OFFICERS - 2008

President
Michael Olin
president@nyoug.org

Vice President
Mike La Magna
vicepresident@nyoug.org

Executive Director
Caryl Lee Fisher
execdir@nyoug.org

Treasurer
Robert Edwards
treasurer@nyoug.org

Secretary
Thomas Petite
secretary@nyoug.org

CHAIRPERSONS

Chairperson / WebMaster
Thomas Petite
info@nyoug.org

Chairperson / Technical Journal Editor
Melanie Caffrey
editor@nyoug.org

Chairperson / Member Services
Robert Edwards
membership@nyoug.org

Chairperson / Speaker Coordinator
Caryl Lee Fisher
speakers@nyoug.org

Co-Chairpersons / Vendor Relations
Sean Hull
Irina Cotler
vendorcoordinator@nyoug.org

Chairperson / DBA SIG
Simay Alpoge
dbasig@nyoug.org

Chairperson / Data Warehousing SIG
Vikas Sawhney
dwsig@nyoug.org

Chairperson / Web SIG
Coleman Leviter
websig@nyoug.org

Chairperson / Long Island SIG
Simay Alpoge
lisig@nyoug.org

Director / Strategic Planning
Carl Esposito
planning@nyoug.org

CHAIRPERSON / VENUE COORDINATOR

Michael Medved
venuecoordinator@nyoug.org

EDITORS – TECH JOURNAL

Associate Editor
Jonathan F. Miller
jonathanfmiller@earthlink.net

Contributing Editor
Arup Nanda - DBA Corner

Contributing Editor
Jeff Bernknopf - Developers Corner

ORACLE LIAISON

Kim Marie Mancusi
Kim.Marie.Mancusi@oracle.com

PRESIDENTS EMERITUS OF NYOUG

Founder / President Emeritus
Moshe Tamir

President Emeritus
Tony Ziemba

Chairman / President Emeritus
Carl Esposito
cesposi@bers.nyc.gov

President Emeritus
Dr. Paul Dorsey

Table of Contents

New York Oracle Users Group, Inc.
#0208
110 Wall Street, 11th floor
New York, NY 10005-3817
(212) 978-8890

General Meeting – December 10, 2008 Agenda

sponsored by All Star Systems & Fusion-io

AGENDA

Time	Activity	Track/Room	Presenter
8:30-9:00	**REGISTRATION AND BREAKFAST**		
9:00-9:30	**Opening Remarks** **General Information**	(single session) Auditorium	Michael Olin NYOUG President
SESSION 1 9:30-10:30	**KEYNOTE:** Coding Therapy for Software Developers aka "How does this code make you feel?"	(single session) Auditorium	Steven Feuerstein Quest Software
10:30-10:45	**BREAK**		
SESSION 2 10:45 -11:45	Tuning the Large Pool for RMAN	DBA Room 118	Anthony Noriega
	Automated Testing Options for Oracle PL/SQL	Developer Auditorium	Steven Feuerstein
SESSION 3 11:45 -12:30	**Ask the Experts Panel**	(single session) Auditorium	Michael Olin Moderator
12:30 -1:30	**LUNCH - ROOM 123**		
SESSION 4 1:30-2:30	Oracle Exadata - Extreme Performance	DBA Auditorium	Graham Wood Oracle Corporation
	Introduction to Java--PL/SQL Developers Take Heart	Developer Room 118	Peter Koletzke Quovera
2:30-2:45	**BREAK**		
SESSION 5 2:45-3:45	Take the Guesswork out of SQL Performance with SPA	DBA Auditorium	Mughees Minhas Oracle Corporation
	Forms Roadmap for Traditional Developers	Developer Room 118	Gil Standen Stillman Real Consulting
3:45-4:00	**BREAK**		
SESSION 6 4:00-5:00	Faster Cross-Platform Database Migration with RMAN	DBA Auditorium	John Larkin Devesh Patel JP Morgan Chase
	Advanced Report Printing with Oracle APEX	Developer Room 118	Marc Sewtz Oracle Corporation

ABSTRACTS

9:30-10:30 AM KEYNOTE: Coding Therapy for Software Developers
aka "How does this code make you feel?"

We can't write software without our brains, and our brains come with a full load of "issues." The way our brain remembers the past and projects into the future has a big impact on how we write code. Moving beyond physiology, human psychology also plays its role, making it difficult for us to acknowledge ignorance and ask for help. Steven will in this keynote address offer an intensive coding therapy session to help all attendees come to grips with their innate, unavoidable "issues", making it easier to write better code -- and help *others* on their team write better code.

Steven Feuerstein, one of the world's leading experts in the Oracle PL/SQL language, has studied PL/SQL for over 15 years and serves as PL/SQL Evangelist for Quest Software.

10:45-11:45 AM DBA TRACK: Tuning the Large Pool for RMAN

This presentation discusses the standard and custom settings to size the Large Pool in conjunction with desirable outcome affecting backup size and duration. The speaker discusses a comparison between Oracle recommended values for the Large Pool and the specific effect of sizing the Large Pool with different values, and the outcome affecting the size of backup pieces, overall backup size, and overall backup and restore turnaround time. Several case studies will be included.

Anthony D. Noriega is a computational scientist and IT Consultant, OCP instructor and Adjunct Professor, who has focused his efforts in database technology, network computing, software engineering, and object-oriented programming paradigms. At ADN/ebPortal, Anthony spends most of his time as a database analyst, architect, and developer, and DBA. Anthony holds an MS Computer Science from NJIT, where he was also a doctoral candidate in the same field, an MBA from Montclair State University, and a BS in Systems Engineering from University of the North. He has been a Sr. consultant for Oracle University, IBM, AT&T, Bowne, Canon, Time, and Deutsche Bank.

10:45-11:45 AM DEVELOPER TRACK: Automated Testing Options for Oracle PL/SQL

We all know we should test our code more thoroughly, but who has the time and patience, and isn't our code "good enough" anyway? Did you know that that is widely accepted that if you have a program of 500 lines, you should expect to write 5000 lines of test code? Ah, so *that's* why we don't test our code! If you are not satisfied with the number of bugs in your code or if you sometimes find yourself embarrassed demonstrating your software to users, then attend this session to learn how you can automate the testing process, reduce bugs in your code, and increase confidence in your applications. This session reviews the tools, from open source to commercial, that offer varying levels of testing automation.

Steven Feuerstein, one of the world's leading experts in the Oracle PL/SQL language, has studied PL/SQL for over 15 years and serves as PL/SQL Evangelist for Quest Software.

1:30-2:30 PM DBA TRACK: Oracle Exadata - Extreme Performance

Data warehouses are tripling in size every two years and supporting ever-larger databases with ever increasing demands from business users to get "answers" faster requires a new way to approach this challenge. Exadata is the new hardware/software solution from Oracle targeted primarily at the data warehousing market Oracle Exadata overcomes the limitations of conventional storage by utilizing a massively parallel architecture to dramatically increase data bandwidth between database and storage servers. In this session, we'll examine these limitations and demonstrate how Oracle Exadata delivers extremely fast and completely scalable, enterprise-ready systems.

Graham Wood is an architect in RDBMS Product Development at Oracle in Redwood Shores. After working in performance related areas for many years creating Statspack, ADDM, AWR amd ASH, Graham now works in the RDBMS Product Management group.

1:30-2:30 PM DEVELOPER TRACK: Introduction to Java - PL/SQL Developers Take Heart

If you know PL/SQL, your first view of Java may be a bit discouraging because its object-oriented core makes it look very different. This presentation explains to PL/SQL developers who have had little or no exposure to Java, the basic concepts of and terms used in Java. It provides an overview of the language; reviews the concepts of object orientation; and describes the fundamental Java code structures—classes and methods—as well as control statements, exception handling, datatypes, and variables.

Peter Koletzke is a technical director and principal instructor for the Enterprise e-Commerce Solutions practice at Quovera, in Mountain View, California, and has over 24 years of industry experience. Peter has presented at various Oracle users group conferences more than 230 times and has won numerous user group awards He is an Oracle Certified Master, Oracle Fusion ACE Director, and coauthor with Dr. Paul Dorsey, Avrom Roy-Faderman, and Duncan Mills of seven Oracle Press Books about Oracle development tools.

2:45-3:45 PM DBA TRACK: Take the Guesswork out of SQL Performance with SPA

Businesses want systems that are predictable, performant, and meet service level agreements. However, SQL performance regression is the key cause of poor and unpredictable database performance. In this session, we will describe how you can demystify the working of the Oracle query optimizer and eliminate the unpredictability in SQL performance by using SQL Performance Analyzer (SPA), a feature of Oracle Database 10g and 11g. We will discuss the internals of SPA as well as best practices for its usage.

Mughees Minhas is responsible for the self-managing solutions of the Oracle Database with special interest in areas such as performance diagnostics, SQL optimization and tuning, space management and load testing. He has more than 12 years of experience working with Oracle databases and is currently director of product management in Oracle's Server Technologies division.

2:45-3:45 PM DEVELOPER TRACK: Forms Roadmap for Traditional Developers

The presentation discusses the evolution of Forms to meet the demands of customers and in particular for Forms to integrate and join with the arrival of Web based applications successfully. Oracle has an updated plan (just released, October 2008) to integrate Forms into Web application architecture and attendees of this presentation will get the most clear statement to-date of this plan. The extraordinary benefits of using an upgraded Forms deployment in a Service Oriented Architecture are the drivers behind the Oracle "Upgrade and Integrate" strategy for existing Forms deployments. Oracle Application Server is highlighted as the central product platform chosen by Oracle as part of their strategy to unify Forms with newer Web based services and SOA.

Gil Standen is Managing Partner of Stillman Real Consulting LLC, and more recently, joined the company founded in 1988 by Rich Niemiec, Joe Trezzo and Brad Brown, the well-known TUSC, in Chicago, where he has been Sr. Consultant for the past year. The author has supported a wide range of Oracle RDBMS and application products as a result of his wide-ranging consulting support of Fortune 50, government, and startup clients for many years. Gil has presented at local and national venues including several NYOUG general meetings and at the Australian National Conferences "Oracle with 20:20 Foresight" in both 2005 and 2007.

4:00-5:00 PM DBA TRACK: Faster Cross-Platform Database Migration with RMAN

You can always move a database from one OS to another with DATAPUMP or EXPORT and IMPORT, but these methods are time consuming.You can reduce the time to convert the datafiles by up to 88% or more when you use RMAN to parallelize the datafile conversions. There are some challenges with some versions of Oracle but there are ways around these issues. Based on Oracle documentation you might even get away with converting only a fraction of all the datafiles in the database. Attendees will learn some basics about RMAN, understand cross-platform migration issues and learn to be flexible with Oracle tools.

John Larkin has worked as a DBA for the last 23 years, the last 15 with Oracle. He works for a major financial services company in Delaware and is currently working on a project to migrate 30 databases from AIX to a Veritas-clustered Solaris environment.

4:00-5:00 PM DEVELOPER TRACK: Advanced Report Printing with Oracle APEX

This session will provide an overview of the printing capabilities built into Oracle Application Express (APEX), discuss some of the printing enhancements introduced in APEX 3.1 and include a live demonstration of how to create custom RTF report templates, include charts and images and utilize the print API for scheduled generation and emailing of reports.

Marc Sewtz is a Software Development Manager for Oracle Application Express (APEX) in the Oracle Database Tools Group and is based on New York City. Marc manages a global team of APEX developers and product managers and is responsible for APEX product features such as Oracle Forms to APEX conversion, the APEX reporting engine, tabular forms, PDF printing and the integration with Oracle BI Publisher.

Message from the President's Desk

Michael Olin

Winter, 2008

As we approach the 25th anniversary of the founding of NYOUG, I had planned to devote this President's Message to providing a brief history of our users group. Although I was there when the group was conceived, in a "newsletter" of sorts that was first sent out in 1984 to a mailing list of users of the Oracle RDBMS who were running the database on Data General hardware, I was not one of the handful (about 10) of users at the first NYOUG meeting the following year.

A Bit of History

Looking back through the NYOUG archives (mostly a collection of old newsletters), I realized that while I had a pretty good memory of the general direction that the group had taken over the past decades, (and I even remembered most of the people who had been involved in leading the group), the timeline was not exactly how I remembered it. There were members of the steering committee with short tenures whose involvement I misplaced by five years or more. There were others whom I thought had been only briefly involved, who turned out to have been leaders of the group for extended periods of time. Finally, my research indicated that there were some very active members of the group who arrived on the scene much later and left much sooner than I had remembered. All in all, what I thought would be a brief one page write-up has grown to be a fairly detailed and lengthy history of NYOUG. After all of this, I am still barely halfway through our history. I hope to complete my work on our "official" history at the beginning of our 25th year. Then, I will have to find a way to distill it to one page. You can expect to see the result of this effort in early 2009.

So what do I write about now? Perhaps I can still convey a little bit of NYOUG history. Many years ago, when the number of companies using Oracle's software was relatively small, NYOUG would set up a simple bulletin board outside the room during our general meetings. By the middle of the day, there would be many announcements of job openings posted on that board and members who were looking for employment opportunities gladly copied down the contact information that was posted and even sought out the member(s) who posted the information in order to discuss the opportunity. This worked quite well for a short time, as our community was rather small. As the number of companies looking to employ Oracle professionals vastly outstripped the supply of capable candidates, two things happened:

1. First, a few IT managers who had been happily allowing the members of their department to attend NYOUG meetings and even paid for their memberships, noticed that some of the members of their team were leaving their departments, having found another opportunity through their membership in NYOUG. This made the managers very angry and not only did they stop paying their employee's membership fees, they refused to allow anyone from their department to attend NYOUG meetings. Since NYOUG was a relatively small organization, the impact that managers at even a few large Oracle shops could have on the group was significant. Even more distressing to NYOUG was that just about every technical recruiter in town figured out that our group was a great way to locate that scarce resource, experienced Oracle professionals. These recruiters signed up to join NYOUG in droves and that caused the second problem.
2. As soon as a recruiter paid his/her membership fee, a copy of our membership list was demanded. The officers patiently explained that we did not share our membership list with anyone, not even other members. Many of the recruiters would berate us for this unreasonable policy, demand a refund of their membership fee, and leave, never to be heard from again. Others, however, appeared to have chosen to take matters into their own hands. On several occasions, members of NYOUG would start receiving cold calls from recruiters and body shops the day after a meeting. Coincidentally, the membership list that was being used to check in people at the meeting could not be located. These two developments led to our policy of not allowing job postings at our meetings and strictly limiting the use of our membership list.

NYOUG is LinkedIn

Times have changed. NYOUG currently has well over 800 paid members and an e-mail list of close to 4,000. Job prospects for Oracle professionals in our area, already negatively impacted by the rise of outsourcing, have been hit again by the turmoil in the economy. Now, more than ever, NYOUG needs to concentrate on using the strengths of our network to help our members who are out of work. We have no intention of running a job posting board or selling our membership list. Quite frankly, we would be quickly overwhelmed. In the past, we would frequently receive calls from multiple sources regarding the same job, and we really do not have any mechanism for the steering committee to manage such a venture. However, we do want to provide a way for NYOUG members to network and share information about job opportunities.

To that end, we have set up an official NYOUG group on the professional networking site LinkedIn. We sent an email with details about this group to the membership in November and in just a week or two several dozen members have already joined. For now, anyone on the NYOUG email list can join the group. Group members can start and contribute to discussions on the site, and share information about opportunities with their fellow members. The next logical step would be for us to promote the group to companies and recruiters who would be interested in posting open positions to the NYOUG. LinkedIn already has a mechanism for members to post job opportunities. We just need to determine what the best mechanism would be for inviting such participation while avoiding the types of intrusive contacts that caused problems in the past. Expect to hear more about NYOUG on LinkedIn and possibly other social and/or professional networking sites in 2009.

NYOUG Leadership for 2009

We will hold our annual election of officers at our December General Meeting. Surprisingly, all of the current officers are the only nominees and all have agreed to continue (if elected) in their positions for another year. When I complete my history of NYOUG, you will see that it is indeed possible for any member to become involved in the NYOUG Steering Committee and even become an officer of the group. Perhaps our 25th anniversary year would be the right time for you to step forward and volunteer.

Upcoming Meeting Dates

RMOUG Training Days 2009

DATE: February 11-12, 2009
LOCATION: Denver, CO
REGISTRATION:
http://www.teamycc.com/RMOUG_2009_Conference/Registration.htm

---

IOUG COLLABORATE 2009

DATE: May 3-7, 2009
LOCATION: Orlando, FL
REGISTRATION:
http://www.ioug.org/collaborate09/index.cfm

ODTUG Kaleidoscope 2009

DATE: June 21-25, 2009
LOCATION: Monterey, CA
REGISTRATION:
http://www.odtugkaleidoscope.com

Recovering the Unrecoverable: An Introduction to Data Unloading for the Oracle Database

Jonah H. Harris
jonah.harris@gmail.com

It's a DBA's worst nightmare--the loss of data. Sure, your backup scripts were well-tested and had some real cool exception handlers in them. But now the time has come to recover your data, and guess what? Those exception handlers weren't so cool after all. Data files are missing, you're missing archive logs, and hey, where did that system tablespace go anyway?

This paper explores the concept of data unloading, lists several unloaders, and describes several real life situations where Oracle database recoveries went terribly wrong; resulting in a situation where using a data unloader was the only option left to extract data.

Introduction

Imagine, if you will, a situation where your Oracle database has become corrupted and your backup could either not be restored or was too out-of-date to be worth restoration. Unfortunately, this situation isn't as hypothetical or as rare as one would think.

While poor backup strategies set the stage for this situation, hardware failures are an all-too-common occurrence which can wreak havoc on your database. So, what is a DBA to do when confronted with such a vexing situation? If the corruption is substantial or the latest data must be recovered, the only solution remaining is Data Unloading (DUL).

What is Data Unloading?

Simply put, DUL is the process of extracting (unloading) data from Oracle data files directly; completely bypassing the Oracle Kernel. Unloading does not even require Oracle to be installed.

How is DUL Different from Exporting Data?

While DUL does perform a bulk export of data, it does not require a running server; as does EXP, EXPDP, etc. DUL is strictly an offline operation.

How Does DUL Interact with Database Security Methods?

As DUL reads the Oracle data files directly, user and role-level permissions are completely ineffective. Currently, the only way to protect data from an unloader is to use encryption. However, Oracle's DUL (and soon DUDE) can read a file which uses Transparent Data Encryption (TDE).

Who Can Perform DUL?

Technically, anyone who knows the data type and file storage formats used by Oracle can extract the data. Though, DUL is generally performed by specialized utilities; a software category of which only a handful of tools exist.

If the SYSTEM tablespace is intact, the DUL utilities are able to retrieve the OBJ$, TAB$, COL$, USER$, PART$, and LOB$ data directly; making the extraction process simpler. However, if this is also corrupt, you may have to specify the

basic object metadata information manually. Though, several of the utilities are able to scan the data files and heuristically determine objects to extract.

As for the extraction itself, each DUL utility contains a block parser, the sole responsibility of which is to read the data files block-by-block, decoding the data, and exporting it.

What Tools Exist for DUL?

As mentioned before, there are very few utilities available for unloading Oracle data files. If you should run into a situation where you need this type of utility or service, I've compiled a comprehensive list below:

- **Bernard's Data UnLoader**
  Oracle's official DUL utility was initially written by Bernard van Duijnen as a standalone C application for Oracle Support in the Netherlands. DUL services from Oracle are offered only as a consulting-based engagement and are rumored to be quite expensive. It can also export data to Oracle Export dump files.
- **DUDE/jDUL**
  DUDE, Database Unloading by Data Extraction, is a Java-based DUL utility for Oracle 7-11g written by OakTable member Kurt Van Meerbeeck of ORA-600. DUDE is far-and-away more comprehensive than any other tool on the market and rivals Oracle's own unloader. DUDE is offered as both a service and a time-limited product. Similar to Oracle's DUL, DUDE can also export data to Oracle Export dump files.
- **AnySQL UnLoader (AUL)**
  AUL is a C-based DUL utility written by Oracle ACE Fangxin Lou. Similar to DUDE, AUL is offered as both a service and a product. AUL has the ability to export data to Oracle Export dump files.
- **WisdomForce FastReader**
  While not designed for recovery purposes, FastReader is able to read directly from Oracle data files in order to perform fast data migration and extraction.
- **Oracle Salvage**
  Oracle Salvage is a SQLite-based data unloader written in C by former Oracle kernel developer Scott Martin of Terlingua Software. It is offered as a product.
- **OracleRecovery**
  A MSVC++-based DUL utility from OfficeRecovery. It is offered only as a product.
- **Recovery for Oracle**
  Recovery for Oracle is a Delphi-based DUL utility offered as both a service and a product from a guy in Poland.
- **MyDUL**
  Different from AUL, MyDUL is a non-commercial utility written by Jerry Sun. I am not sure whether it is still publicly available.
- **CLOUT**
  Clever Little Oracle Unloading Tool, my own proof-of-concept DUL utility, is not commercially available.

What Limitations Do Unloaders Have?

While an unloader works most of the time, there are certainly cases where even it will not work, such as zeroed out data blocks and massively corrupted memory-mapped files.

What Types of Cases Can Lead to Requiring an Unloader?

As mentioned earlier, some of the time, the need for an unloader is due to hardware-related failure. However, human-error accounts for the majority of cases requiring an unloader. Several of these types of cases are discussed below.

Backup and Recovery Scenario #1: Database Administratus Ignoramus (i.e. The Idiot DBA)

Face it. We've all known someone who fits this description. In fact, you're probably already thinking of them, or the mess they left behind for you. This person is the one who says, *"If there's no activity during the night, I can make hot*

backups of the database without putting it in backup mode!" This is also the same person who took manual, inconsistent backups that *you* may have to try and recover from. A lack of knowledge is a very dangerous thing when it comes to making sure data is recoverable. In this type of case, an inconsistent backup may make parts of the backup data inaccessible, requiring an unloader to extract.

Backup and Recovery Scenario #2: The developer wrote the backup scripts.

Have you ever been in a situation where the application developer was also the one who wrote the backup scripts? Ever try and change the user that runs the application? Ever run into poor hard-coded assumptions? Backups should be designed and tested by experienced database administrators. In these types of cases, an unloader may be required to extract data from orphaned data files.

Backup and Recovery Scenario #3: Did you swap out that backup tape?

Unfortunately, several companies I've worked for have placed the responsibility of tape management onto non-technical employees. As these employees lack the skills necessary to properly perform such an important task, rarely can you find the correct tape when you need it. And, even if you can locate it, rarely is it a proper, usable backup.

Backup and Recovery Suggestions

Several simple things can help make your backup and recovery methods a success:

- Use common sense
- Use RMAN
- If you don't know, hire an experienced consultant
- Practice restore/recovery scenarios often

Other Reference Material

For more information on this topic, I would recommend that you find a copy of the following documents:

- Note:1031902.6 How does Oracle store internal numeric data?
- Note:69028.1 How does Oracle store the DATE datatype internally?

Jonah H. Harris
Sr. Software Architect, EnterpriseDB
& Independent Database Consultant

Having administered, developed against, and consulted on all versions of Oracle since version 7, Jonah has spent a considerable amount of time specializing in OCI, performance tuning, and researching Oracle Internals. In addition to blogging, (and when time permits) he provides support to the Database, Call Interface, Instant Client, ODBC, PHP, Linux, SQL, and PL/SQL community forums on the Oracle Technology Network.

jonah.harris@gmail.com
http://www.oracle-internals.com/

Not Using Analytics? Shame on You!

Travis R. Rogers
SRVP of Database Development
CDG Management, LLC
travis@jerseyrogers.com

Intended Audience

This paper is intended for individuals that have a basic understanding of relational databases and SQL. Although the paper is Oracle specific anybody with these skills should be able to easily understand the topics discussed.

Scope

The intent is to present a primer of the extended SQL functionality known as Oracle Analytics (aka Analytic Functions or Window Functions). This is not a thorough treatment of the topic just enough information to convince readers of the value in day-to-day SQL tasks, plus provide enough information to get started. Specifically the goals are:

1. Define Oracle Analytics.
2. Learn why analytics should be used.
3. Learn enough to get started.
4. See some examples.

Unless otherwise specified the terms “Oracle Analytics”, “Analytic Functions”, “Window Functions” and “Analytics refer to the same functionality.

Oracle Analytics has been in Oracle since version 8.1.6 and is also known as Analytic Functions, Window Functions and Windowing Functions. It has been adopted, under the name Window Functions, as part of ANSI SQL 2003 OLAP Functions (non-core feature ID T611). Oracle technical deities generally have high praise for the functionality for good reason. There are several places in the Oracle documentation where a definition is attempted:

> *Analytic functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group.*
>
> *Oracle Database SQL Reference Ch. 5*

> *Analytic functions (formally called window or windowing functions) enable you to compute various cumulative, moving and centered aggregates over a set of rows called a window.*
>
> *Oracle Database Data Cartridge Developer’s Guide Ch. 11*

> *It provides access to more than one row of a table at the same time without a self join.*
>
> *Various places in Oracle Docs*

These are all fine, if incomplete, definitions for someone that already understands the basics; but they provide little help for the beginner. The most thorough treatment of analytics in the Oracle documentation is Chapter 21 of the *Oracle Database Data Warehousing Guide* but it does not really attempt a definition and instead launches directly into the details.

In addition to lacking a clear definition, the term Oracle Analytics or Analytic Functions is misleading at best. Unfortunately this nomenclature seems to pigeonhole, in many minds, where and when it should be used. Instead of being a tool that SQL developers use as readily as any other, it seems to be ignored unless some sort of statistical analysis is being discussed/attempted. Why then the name? Well, that's a mystery but I have some theories:

1. The name window function is boring.
2. Analytics is a cool(er) name.
3. Analytics was/is a hot topic.
4. Businesses pay for Analytics but hate paying for windows ;).
5. This functionally is very useful in statistical and OLAP analysis.
6. Analytics is a free word.

Yes, Analytics is a free word. It gets thrown around a lot and sounds pretty impressive but it's real meaning is ambiguous.

> *In reality, the word "Analytics" has not been properly defined by the professional community and may mean different things to different people…*
>
> *Wikipedia*

That explains why use of the word is free but why then is it "cool" and why would businesses be willing to pay for it?

> *…common applications of Analytics include the study of business data using statistical analysis in order to discover and understand historical patterns with an eye to predicting and improving business performance in the future…*
>
> *Wikipedia*

Aha! That explains the perceived business value and at the same time explains why the functionality is pigeonholed. Oracle Analytics is a vaunted, standardized functionality that is poorly defined and poorly named. It needs is a new name and a new definition...

Do you really think that if all the people at Oracle couldn't come up something better, that I would? That's right, I got nothin'. Windowing is probably the best name because understanding the window is the key to understanding the capability. Later there will be a description which is helpful but geared towards kick starting use and not complete enough to be a definition.

Why Are Analytics Not Used More Often?

This conversation assumes that it is not used often. This assumption is based purely on observation does not rely on any data. Although there are probably many reasons why it's not used more often, below are a few.

My App/Project Must Connect to Multiple Databases
Or
My Company/Manager Says I Have to Use Standard/Generic SQL

This is one of those religious debates so re-hashing all of it is of no value. However, there are a couple things that deserve being re-said.

Analytic functions are part of a standard, although may not be supported by all relevant database products. It is possible that, with this knowledge, you can start using analytics right away with no additional debate.
Otherwise, keep in mind that there is an extreme lack of products that successfully use generic SQL. Most products that try, end up with a db specific layer or db specific capabilities built in. Why waste the time and energy to ignore this fact rather than build with a realistic design. There are even fewer (if any) products that get the most out of their hardware investments and use generic SQL.

The desire for generic SQL makes less sense if your company's or product's primary database is Oracle. If you are not using the Oracle specific features you are wasting time and money on development and tons of money on hardware in the "possibility" that someday money might be saved on a conversion.

This may not convince your company to change so at least design with views. If the application SQL generically accesses views, then the views can take advantage the database. The fact that analytic functions are a part of Oracle SQL allows its easy use in views.

I'm Not Doing "Analytical" Work

So what? Even if it were true that this was functionality specifically developed for "analytical" work (whatever that means) if it can be effectively used elsewhere why not? Regardless, this is not functionality specifically developed for analytical work. If you've done significant SQL work, by the end of this paper you will see how analytics fills a gap in SQL.

It's Hard
Or
It's Hard to Read

This is a common but irritating answer and generally shows a lack of willingness to learn something new. The reality is that it is not hard and that once understood it is easier to read than other alternatives. I recently reviewed a PL/SQL procedure which was 5 pages long. The analytic replacement was a single SQL statement less than 1 page long.
Oracle analytics seem hard. Much of this is due to things discussed prior (documentation and name) but also a lot of the reason is just that it is different. Remember that the first time a good C developer sees SQL, it is a mystery because it is so different...but it is still the best way to get at relational data.

I Can Do the Same Thing Using Other SQL and/or PL/SQL So There Is No Reason

Most of the time this is true, but at what cost? The same argument could be made of almost any SQL functionality but everybody can agree that most joins should be done in SQL and not hand coded into each application.
Probably the best way to combat this attitude is to prove the value of Oracle analytics. The procedure mentioned in the prior section originally executed in 20 minutes even after multiple changes (over 10 man hrs) were made to increase performance. Each performance modification made the implementation more obtuse resulting in a procedure that had multiple loops, used a temp table and was 5 pages long. The analytic replacement was a single SQL statement less than 1 page long and used no temporary tables. It took 2 hrs to create and test and executes in half the time (10 minutes).

I'm Waiting on the Tom Kyte Book

Hopefully some day there will be a book, but in the mean time there is good work to be done!

Why Use Analytics?

This will be a short discussion since most of it is a repeat of the above. In addition, much of this will become obvious with some later examples. However, it is worth driving a few things home.

Faster

Most of the time if a self join or PL/SQL is the only way to solve a problem and Oracle analytics could be used, Oracle analytics will perform faster. So far I have not seen a single incident where the analytics function is not better. There was one situation where I was presented with a scalar subquery that performed as good as or slightly better than the equivalent analytics function. The scalar subquery took advantage of an index and its performance depended on which data was being selected. The analytics statement did not need the index and always performed the same regardless of the data being queried. In my opinion, requiring an index to get equal or slightly better performance (sometimes) is not a good solution.

Easier

You won't agree till you learn it. Keep reading, try it out and you'll agree.

More Readable

If you can read SQL you can read Oracle analytics. Once you understand what you're looking at you'll find it more readable than the alternatives.

If You Don't, Somebody Else Will

There are so many uses for Oracle analytics that eventually people will add this to their list of tools. The capability is here to stay and is getting better and faster. You can either start using it now or wait until someone makes you look bad.

The Basics

As promised earlier, this section provides a description that helps get started with Oracle analytics. In addition we'll see a simple example from multiple perspectives and use it to understand what is going on behind the scenes.

How I See Oracle Analytics

Oracle Analytics is…

1. The ability to query a subset of the data from the primary result set (the subset is also referred to as the window).
2. The subset query will return a single value per row and is presented as a column in the final query output.
3. This can be done multiple times in a single query.

The Primary Result Set

The primary result set is everything that could possibly be included in the result set. For example, the primary result set for the below is emp.*, not empno because the possibility of what can be returned is emp.*.

```
SELECT e.empno, d.loc
FROM   emp e
       ,dept d
WHERE  e.deptno = d.deptno
       AND e.job = 'CLERK';
```

A Subset Query Example

```
SELECT empno, sal, deptno
       SUM(sal) OVER (PARTITION BY deptno) as dept_sal
FROM   emp;
```

In this example:

1. emp.* is the primary result set.
2. "SUM(sal) OVER (PARTITION BY deptno) is the analytic function
3. "(PARTITION BY deptno) is the subset query.

The translation of this analytic function is:

> *Get the salary for every employee in the same department as the employee and return the sum.*

The result set looks like

EMPNO	SAL	DEPTNO	DEPT_SAL
7782	2450	10	8750
7839	5000	10	8750
…	…	…	…
7566	2975	20	10875
7902	3000	20	10875
…	…	…	…
7521	1250	30	9400
7844	1500	30	9400
…	…	…	…

Table 1 - Data for Subset Example

Traditional SQL Alternatives

There are at least 3 traditional SQL alternatives to the above example. Each of these alternatives break down when things get more complex but a more in-depth look at them reveals why the analytic function is the best solution for even this simple scenario.

Alternative 1 – Inline View

```
SELECT   e1.empno, e1.sal, e1.deptno, e2.d_sal
FROM     emp e1
         ,(SELECT   deptno, SUM(sal) AS d_sal
           FROM     emp
           GROUP BY deptno) e2
WHERE    e1.deptno = e2.deptno
```

Alternative 2 – Scalar SubQuery

```
SELECT e1.empno, e1.sal, e1.deptno,
       (SELECT SUM(sal)
        FROM   emp
        WHERE  deptno = e1.deptno) AS dept_sal
FROM   emp e1
```

Alternative 3 – PL/SQL

This is a basic piece of code that would loop through the table once and manually summarize the data. Due to its length (compared to the rest of the paper) there is no need to insert the code here, but there is a need to understand the process.

How Does Each Alternative Work?

Ok, so there are multiple ways to do this thing and at least 2 of them are simple and most likely effective. Now let's take a look at what is happening under the covers for each.

SubQuery

The process is basically the same for either the inline view (Alt 1) or the scalar subquery (Alt 2). See diagram below.

1. Get emp data from disk.
2. Put into memory.
3. Get 2nd copy of emp data from disk.
4. Put into memory.
5. Process each appropriately in memory.
6. Merge.
7. Send final view to application.

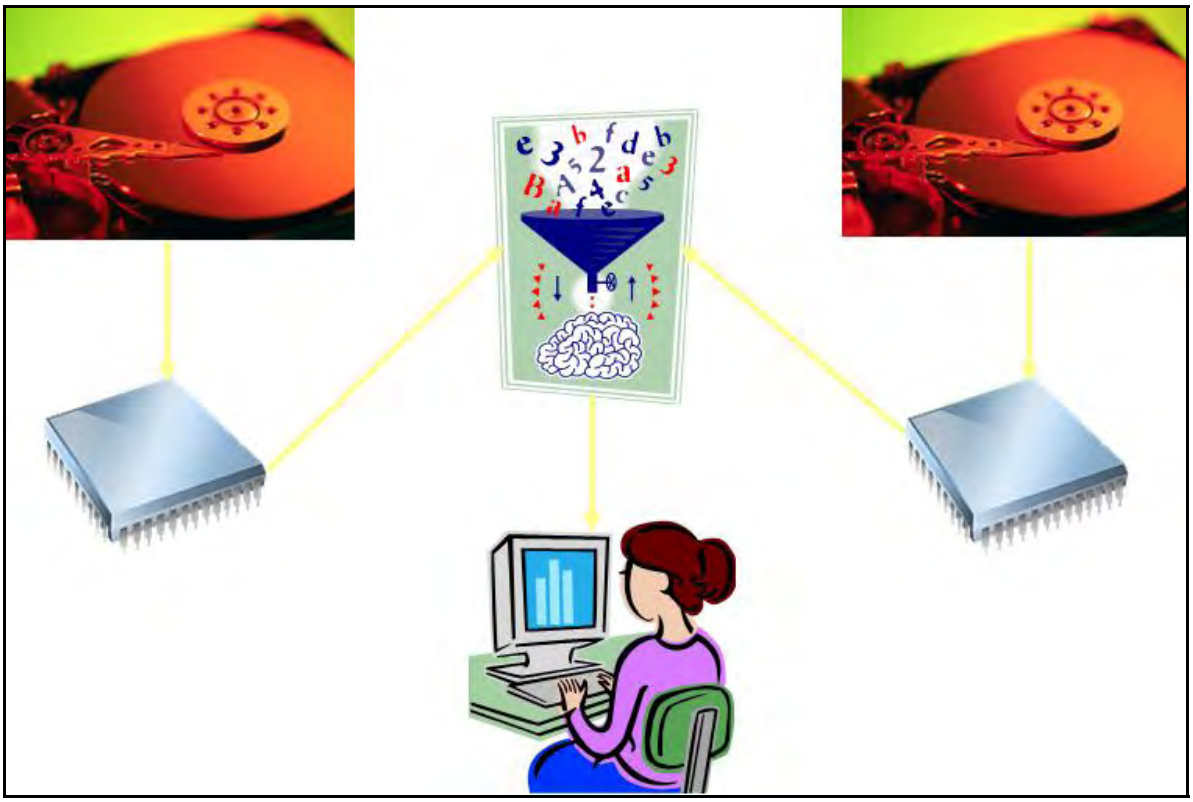


Diagram 1- How Subquery Works

PL/SQL

This process would get the data from disk, put it into memory and then pass it to the PL/SQL block which summarizes it and passes it to the application. Since generally disk reads are bad this seems like a better process than the subqueries (above); however having to pass the data to PL/SQL creates more overhead and results in a slower process.

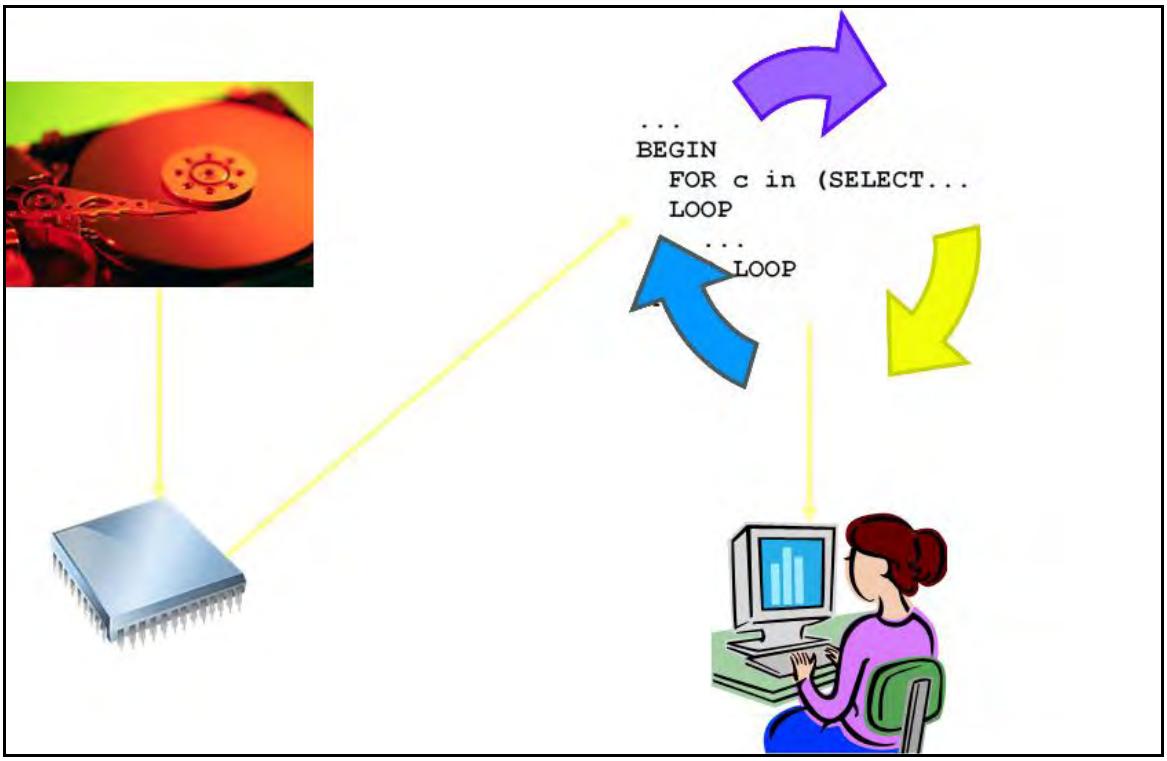


Diagram 2 - How PLSQL Works

Analytics

The analytics function gets the data from disk, puts it into memory and then summarizes the data as outlined in the diagram below. Here we see why it is better:

1. There is only 1 pull from disk.
2. There is no PL/SQL overhead.
3. Basically it follows the best process in low level, optimized code.

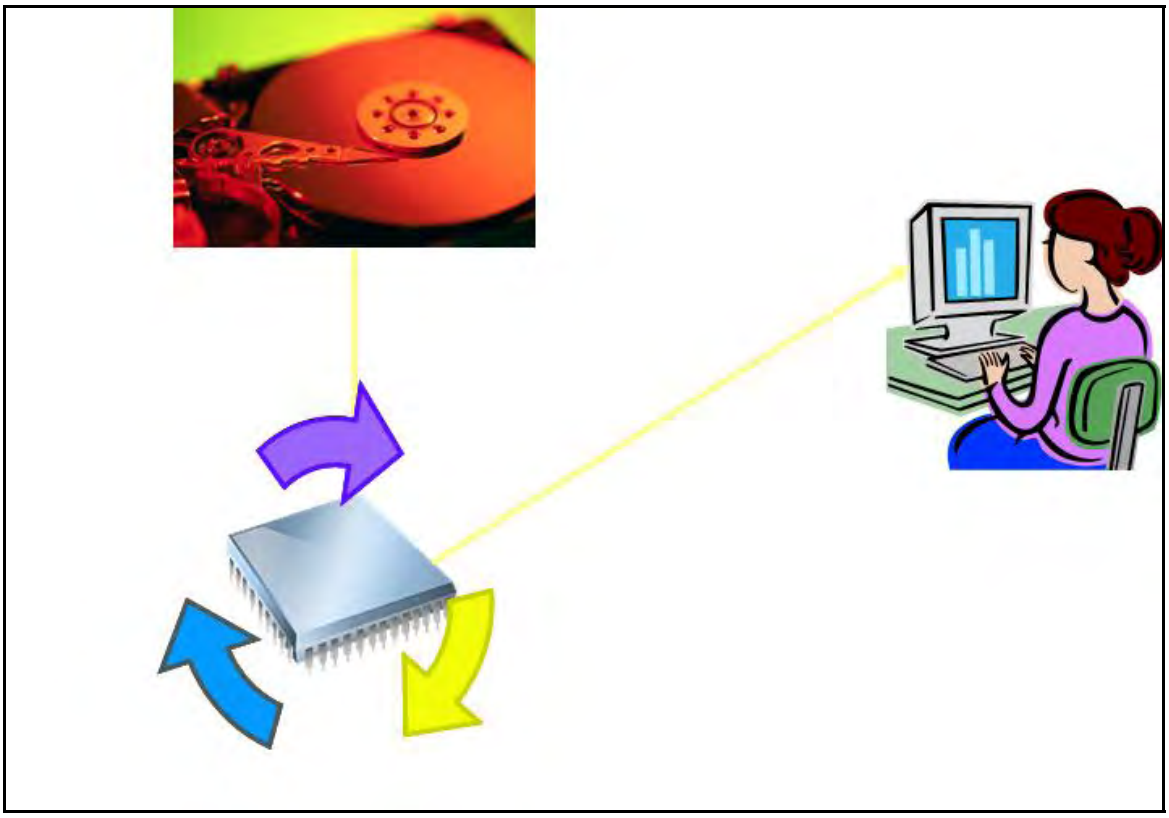


Diagram 3 - How Analytics Works

Summary of Example

Even in this simple example it is pretty obvious that the analytic function is the best. Is it really worth the effort of using a different method than what is already generally known and practiced? This example contains only one summary (department salary) but if there were 3 it would require 4 disk reads for alternates 1&2 and a significantly more complex (i.e. costly) stored procedure. In other words each additional summary using traditional methods results in an exponential increase in effort; however, the analytic function might perform more work but it will not perform exponentially more work.

Before moving on, it is worth noting…

1. Take a look at the example, there is no group by in sight. One major misconception with analytic functions is that they require or use or live with SQL grouping capability…they don't.
2. This is a simple query. With analytics it is best that the base query be simple and remains simple as long as possible.
3. A single analytic function fills one column for each row. You should always think in this fashion, "What piece of data will I get with each individual row?" Better yet when first starting out, "What piece of data do I need with each individual row?"
4. If you had to read data from a text file and provide this same info would you read the file twice and mush the data together or read it once and get what you needed from it? Analytics follow the same basic method you would.

Parts of an Analytic Function

```
<function> OVER (<subset>)
```

<function> are aggregate and other analytic specific functions that are applied to the subset of data. Here is a list of functions with the most useful for day-to-day activities underlined.

- AVG
- CORR
- COVAR_POP
- COVAR_SAMP
- COUNT
- CUME_DIST
- DENSE_RANK
- FIRST
- FIRST_VALUE
- LAG
- LAST
- LAST_VALUE
- LEAD
- MAX
- MIN
- NTILE
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- RANK
- RATIO_TO_REPORT
- REGR_(linear regression)
- ROW_NUMBER
- STDDEV
- STDDEV_POP
- STDDEV_SAMP
- SUM
- VAR_POP
- VAR_SAMP
- VARIANCE

<subset> contains up to three parts which together define the window of data and the order of data inside the window.

1. PARTITION BY <expr>
2. ORDER BY <expr>
3. <window>

The easiest way to create subsets for the beginner is to think in SQL. First ask yourself, "What SQL is required to get the value needed for this column+row from the primary result set?" Once the basic SQL is identified apply the analytic subset syntax. Eventually however, in order to get the real power from analytic functions it will be important to start thinking in "windows of data". This is the reason that "windowing" or "window functions" is the best name for this capability.

Windows into the data

Take this data set:

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL
7900	JAMES	CLERK	7698	30	950
7902	FORD	ANALYST	7566	20	3000
7521	WARD	SALESMAN	7698	30	1250
7566	JONES	MANAGER	7839	20	2975
7499	ALLEN	SALESMAN	7698	30	1600
7934	MILLER	CLERK	7782	10	1300
7698	BLAKE	MANAGER	7839	30	2850
7788	SCOTT	ANALYST	7566	20	3000
7654	MARTIN	SALESMAN	7698	30	1250
7369	SMITH	CLERK	7902	20	800
7839	KING	PRESIDENT		10	5000
7876	ADAMS	CLERK	7788	20	1100

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL
7844	TURNER	SALESMAN	7698	30	1500
7782	CLARK	MANAGER	7839	10	2450

Table 2 – Basic Data Set

Apply "PARTITION BY deptno" and the data is sorted by deptno and the function will only use data where the value is equal to the current row's deptno.

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL
7934	MILLER	CLERK	7782	10	1300
7839	KING	PRESIDENT		10	5000
7782	CLARK	MANAGER	7839	10	2450
7902	FORD	ANALYST	7566	20	3000
7566	JONES	MANAGER	7839	20	2975
7788	SCOTT	ANALYST	7566	20	3000
7369	SMITH	CLERK	7902	20	800
7876	ADAMS	CLERK	7788	20	1100
7900	JAMES	CLERK	7698	30	950
7521	WARD	SALESMAN	7698	30	1250
7499	ALLEN	SALESMAN	7698	30	1600
7698	BLAKE	MANAGER	7839	30	2850
7654	MARTIN	SALESMAN	7698	30	1250
7844	TURNER	SALESMAN	7698	30	1500

Table 3 - Basic Data Set Partitioned

Apply "ORDER BY mgr" and it will order the data by manager id within each partition. Notice that 7839 exists in both the partition for department 10 and department 30 but they are not sorted together. It's like it was supposed to order first by deptno and then mgr. Technically that is how it works the difference is that the "PARTITION BY" command creates a boundary (like the frame of a window).

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL
7839	KING	PRESIDENT		10	5000
7934	MILLER	CLERK	7782	10	1300
7782	CLARK	MANAGER	7839	10	2450
7902	FORD	ANALYST	7566	20	3000
7788	SCOTT	ANALYST	7566	20	3000
7876	ADAMS	CLERK	7788	20	1100
7566	JONES	MANAGER	7839	20	2975
7369	SMITH	CLERK	7902	20	800
7900	JAMES	CLERK	7698	30	950
7521	WARD	SALESMAN	7698	30	1250
7499	ALLEN	SALESMAN	7698	30	1600
7654	MARTIN	SALESMAN	7698	30	1250
7844	TURNER	SALESMAN	7698	30	1500

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL
7698	BLAKE	MANAGER	7839	30	2850

Table 4 - Basic Data Set Partitioned and Sorted

If we stop here and apply the analytic function "SUM(sal)"we get the following which shows that the windows of data for each record with deptno 10 is all records with deptno 10 and etc.

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL	SUM(sal)
7839	KING	PRESIDENT		10	5000	5000+1300+2450 = 8750
7934	MILLER	CLERK	7782	10	1300	5000+1300+2450 = 8750
7782	CLARK	MANAGER	7839	10	2450	5000+1300+2450 = 8750
7902	FORD	ANALYST	7566	20	3000	3000+3000+1100+2975 +800 = 10875
7788	SCOTT	ANALYST	7566	20	3000	3000+3000+1100+2975 +800 = 10875
7876	ADAMS	CLERK	7788	20	1100	3000+3000+1100+2975 +800 = 10875
7566	JONES	MANAGER	7839	20	2975	3000+3000+1100+2975 +800 = 10875
7369	SMITH	CLERK	7902	20	800	3000+3000+1100+2975 +800 = 10875
7900	JAMES	CLERK	7698	30	950	950+1250+1600+1250+ 1500+2850 = 9400
7521	WARD	SALESMAN	7698	30	1250	950+1250+1600+1250+ 1500+2850 = 9400
7499	ALLEN	SALESMAN	7698	30	1600	950+1250+1600+1250+ 1500+2850 = 9400
7654	MARTIN	SALESMAN	7698	30	1250	950+1250+1600+1250+ 1500+2850 = 9400
7844	TURNER	SALESMAN	7698	30	1500	950+1250+1600+1250+ 1500+2850 = 9400
7698	BLAKE	MANAGER	7839	30	2850	950+1250+1600+1250+ 1500+2850 = 9400

Table 5 - Basic Data Set Partitioned and Sorted with Analytic Function

A minor addition to the <subset> will change the calculation completely. The <window> phrase "ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING" translates to "take all rows in the partition that are the current row, 1 row preceding the current row and 1 row following the current row."

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL	SUM(sal)
7839	KING	PRESIDENT		10	5000	5000+1300 = 6300
7934	MILLER	CLERK	7782	10	1300	5000+1300+2450 = 8750

EMPNO	ENAME	JOB	MGR	DEPTNO	SAL	SUM(sal)
7782	CLARK	MANAGER	7839	10	2450	1300+2450 = 3750
7902	FORD	ANALYST	7566	20	3000	3000+3000 = 6000
7788	SCOTT	ANALYST	7566	20	**3000**	3000+3000+1100 = 7100
7876	ADAMS	CLERK	7788	20	**1100**	3000+1100+2975 = 7075
7566	JONES	MANAGER	7839	20	**2975**	1100+2975+800 = 4875
7369	SMITH	CLERK	7902	20	800	2975+800 = 3775
7900	JAMES	CLERK	7698	30	950	950+1250 = 2200
7521	WARD	SALESMAN	7698	30	1250	950+1250+1600 = 3800
7499	ALLEN	SALESMAN	7698	30	1600	1250+1600+1250 = 4100
7654	MARTIN	SALESMAN	7698	30	1250	1600+1250+1500 = 4350
7844	TURNER	SALESMAN	7698	30	1500	1250+1500+2850 = 5600
7698	BLAKE	MANAGER	7839	30	2850	1500+2850 = 4350

Table 6 - Basic Data Set Partitioned, Sorted and Windowed with Analytic Function

The following is important to note:

1. 7839 – This sums only the current row and the row after because there is no row before it in the cursor.
2. 7698 – This sums only the current row and the row before because there is no row after it in the cursor.
3. 7782 & 7369 – This sums only the current row and the row before. Although there is a row after it in the cursor, it is unreachable due to being outside the partition.
4. 7902 & 7900 – This sums only the current row and the row after. Although there is a row before it in the cursor, it is unreachable due to being outside the partition.
5. 7934, 7788, 7876, 7566, 7521, 7499, 7654 & 7844 – This sums the current row, the row before and the row after.
6. The cells in bold show the window used for the row 7876. Visualizing the data in a window will help in defining the <subset> syntax.

This was a lot of work behind the scenes but the query itself is pretty simple:

```
SELECT empno, ename, job, mgr, deptno, sal
       SUM(sal) OVER (PARTITION BY deptno
                      ORDER BY mgr
                      ROWS BETWEEN 1 PRECEDING
                               AND 1 FOLLOWING) AS dept_sal
FROM emp;
```

Examples

Example 1

Problem Statement

Products are registered in a table called prod_p1 containing a product ID (pid), registration date (reg_dt) and a code indicating the product type (type). Entries in this table are sparsely populated meaning multiple products may be registered this month and none registered next month. We need to find:

1. All entries.
2. From the most recent X months (we will use 4 in this example).
3. Prior to the current month (we will use 12/2007 for this example).
4. For a specified product type (we will use 1 for this example).

Need to Know

> *The analytic function **DENSE_RANK** computes the rank of a row in an ordered group of rows and returns the rank as a number. The ranks are consecutive integers beginning with 1. The largest rank value is the number of unique values returned by the query. Rank values are not skipped in the event of ties. Rows with equal values for the ranking criteria receive the same rank.*
>
> *Oracle Documentation*

Base Data

PID	REG_DT	TYPE
93	4/17/2007	2
88	4/30/2007	1
90	5/1/2007	1
92	6/10/2007	1
95	6/22/2007	3
94	8/9/2007	1
97	9/20/2007	3
99	11/1/2007	2
96	11/8/2007	1
98	11/15/2007	1
100	12/1/2007	1

Table 7 - Example 1 Base Data

First eliminate rows the old fashioned way. Since we are looking at Type = 1 that eliminates pid 93, 95, 97 and 99. In addition we only want entries prior to 12/1/2007 which eliminates pid 100. Here is the starting SQL:

```
SELECT a.*
FROM   prod_p1 a
WHERE  a.reg_dt < trunc(SYSDATE, 'MM')
       AND type = 1;
```

Since there is a nice small sample set of data figuring out what other rows should be eliminated is pretty simple. In the remaining rows, the most recent 4 months that have entries are 11/2007, 8/2007, 6/2007 and 5/2007. This means that the row for 4/2007 needs to be eliminated. Also note that there are two entries in 11/2007 and both are desired in the final record set. Eliminating the row for 4/2007 is a two step process which starts by applying DENSE_RANK to the entire population. No partition is needed; however the order by value must be equal by month since the requirements specify month. The order is specified as descending which makes the most recent month the smallest dense_rank value.

```
SELECT a.*,
       trunc(a.reg_dt, 'MM') AS mm_dt,
       dense_rank() OVER
         (ORDER BY trunc(a.reg_dt, 'MM') DESC) AS rank
FROM   prod_p1 a
WHERE  a.reg_dt < trunc(SYSDATE, 'MM')
       AND type = 1;
```

Resulting in…

PID	REG_DT	TYPE	MM_DT	RANK
96	11/8/2007	1	11/1/2007	1
98	11/15/2007	1	11/1/2007	1
94	8/9/2007	1	8/9/2007	2
92	6/10/2007	1	6/1/2007	3
90	5/1/2007	1	5/1/2007	4
88	4/30/2007	1	4/1/2007	5

Table 8 - Example 1 Step 2 Data

The last step will filter out everything except ranks 1-4 by making this query an inline view and applying the filter on the outside.

```
SELECT b.*
FROM   (SELECT
         a.*,
         trunc(a.reg_dt, 'MM') AS mm_dt,
         dense_rank() OVER
         (ORDER BY trunc(a.reg_dt, 'MM') DESC) AS rank
        FROM  prod_p1 a
        WHERE a.reg_dt < trunc(SYSDATE, 'MM')
              AND type = 1) b
WHERE b.rank <= 4;
```

Example 2

Problem Statement

There is a table (sales_p2) which records each individual sale, the employee (empno) responsible and the dollar amount of the sale (amt). Using this and the department table there is a need to see the employee, count of sales, sum of the dollar amount of the sales, average dollars per sale compared to the average dollars per sale for the department.

Need to Know

Aggregating an analytic function is not allowed and will result in the error "ORA-30483 window functions are not allowed here". The below example is illegal due to "MAX" aggregate around the "SUM" analytic.

```
SELECT s.empno
       ,MAX(SUM(s.amt) OVER
         (PARTITION BY e.deptno)) AS d_amt
FROM     sales_p2 s
         ,emp e
WHERE    s.empno = e.empno
GROUP BY  s.empno
```

Step 1

The analytic statements for this are not difficult, this example shows a trick for getting only the data that is needed when using group by's. In order to get the average sale for either department or employee the base data required is the total dollars sold and (divided by) the total number sold. Doing this for each employee is easy and doing this for each department is easy, but what is the best way to put them together. As indicated previously, start simple by getting the department data side by side with the sales_p2 data.

```
SELECT s.empno, s.amt, e.deptno
       ,COUNT(*) OVER
         (PARTITION BY e.deptno) AS d_sls
       ,SUM(s.amt) OVER
         (PARTITION BY e.deptno) AS d_amt
FROM   sales_p2 s, emp e
WHERE  s.empno = e.empno
```

Resulting in …

EMPNO	AMT	DEPTNO	D_SLS	D_AMT
7369	13,227	20	87	1,262,099
7369	26,445	20	87	1,262,099
7369	6,963	20	87	1,262,099
…	…	…	…	…
7499	32,333	30	111	1,799,562
7499	21,826	30	111	1,799,562
7499	8,039	30	111	1,799,562
…	…	…	…	…
7566	23,598	20	87	1,262,099
7566	12,269	20	87	1,262,099
…	…	…	…	…
7782	6,660	10	54	917,949
7782	6,562	10	54	917,949

EMPNO	AMT	DEPTNO	D_SLS	D_AMT
7782	1,482	10	54	917,949

Table 9 - Example 2 Step 1 Data

The only thing left is to group by empno, count and sum...the problem is that d_sls and d_amt are already counted/summed and if you count/sum them again they will be wrong. The trick to this is to use an aggregate function which only returns a value from 1 single row. That is what MAX and MIN do. Here is the final SQL.

```
SELECT   x.empno,
         COUNT(*) AS e_sls,
         sum(x.amt) AS e_amt,
         MAX(x.d_sls) AS d_sls,
         MAX(x.d_amt) AS d_amnt
FROM
 (SELECT s.empno, s.amt,
         COUNT(*) over
          (PARTITION BY e.deptno) AS d_sls,
         SUM(s.amt) over
          (PARTITION BY e.deptno) AS d_amt
  FROM   sales_p2 s, emp e
  WHERE  s.empno = e.empno) x
GROUP BY x.empno
```

Example 3

Problem Statement

In a large production line there are tens of thousands of devices each spewing a variety of events on a regular basis. One of these events indicates a potential device failure; however, experience has shown that these are often false positives. In these circumstances technicians are sent to replace the device but it is working fine. If a technician is not dispatched as soon as a real fault is detected significant production problems can occur. It has been theorized that there is a pattern of events that would better indicate a real failure. In order to determine this, the event data needs to be analyzed.

1. For each instance where the device has issued the first error (status = 0) then:
   -How often did any event follow?
   -If an event followed how often was it an error?
2. Repeat this data up to 3 consecutive errors (i.e. was there a 4th following event and was it an error)

Need to Know

> *The analytic function **LEAD** provides access to a row at a given physical offset beyond that position.*
>
> *Oracle Documentation*

In other words, LEAD allows the query to "peek ahead" in the subset.

Event Table Structure

NAME	TYPE
Dev_id	Number(10)
Evt_dt	Date
Status	Number(2)

Table 10 - Example 3 Data Structure

Step 1

This example is easier than it looks, the hard part is figuring which event is the first ever failure and only evaluating those. Leave that out for now, what if we wanted to know the following about every event?

1. The event code of the current event
2. The event code of the next event (in time order) after the current event for the same device.
3. The event code of the 2nd event after the current event for the same device.
4. The event code of the 3rd event after the current event for the same device.
5. The event code of the 4th event after the current event for the same device.

This information is pretty simple to get with a basic analytic function.

```
SELECT a.*,
       lead(a.status,1) over
        (PARTITION BY a.dev_id
         ORDER BY a.evt_dt ASC) AS next1_stat,
       lead(a.status,2) over
        (PARTITION BY a.dev_id
         ORDER BY a.evt_dt ASC) AS next2_stat,
       lead(a.status,3) over
        (PARTITION BY a.dev_id
         ORDER BY a.evt_dt ASC) AS next3_stat,
       lead(a.status,4) over
        (PARTITION BY a.dev_id
         ORDER BY a.evt_dt ASC) AS next4_stat
FROM   dev_evt a;
```

Step 2

Now back to the real problem of only analyzing the first failure event per device. If we restrict the primary result set to only show failure events then the "LEAD" analytic statements can only see other failure events which is not what is needed. Inlining this querying and using MIN (group by dev_id) doesn't work because it will give us the first of all events and we need the first failure event. There is probably a way to inline and get this data but not without doing a self-join. The good news is that analytics allows an expression inside most functions. To solve this problem we're going to use a case statement to fake out the MIN function to believe that every non-failure occurred in the future. This expression only affects the single analytic return value. Here is another column to be added to the above SQL.

```
MIN(CASE WHEN a.status = 0
         THEN a.evt_dt
         ELSE to_date('1-jan-2099')END)
over (PARTITION BY a.dev_id) AS min_err_dt
```

Final Step

That's all the data we need but the final result set still returns 1 row for every event. There is a pattern that will be applied here that can be seen across our examples…most of the time an analytic function exists in an inline view. Here is the final SQL.

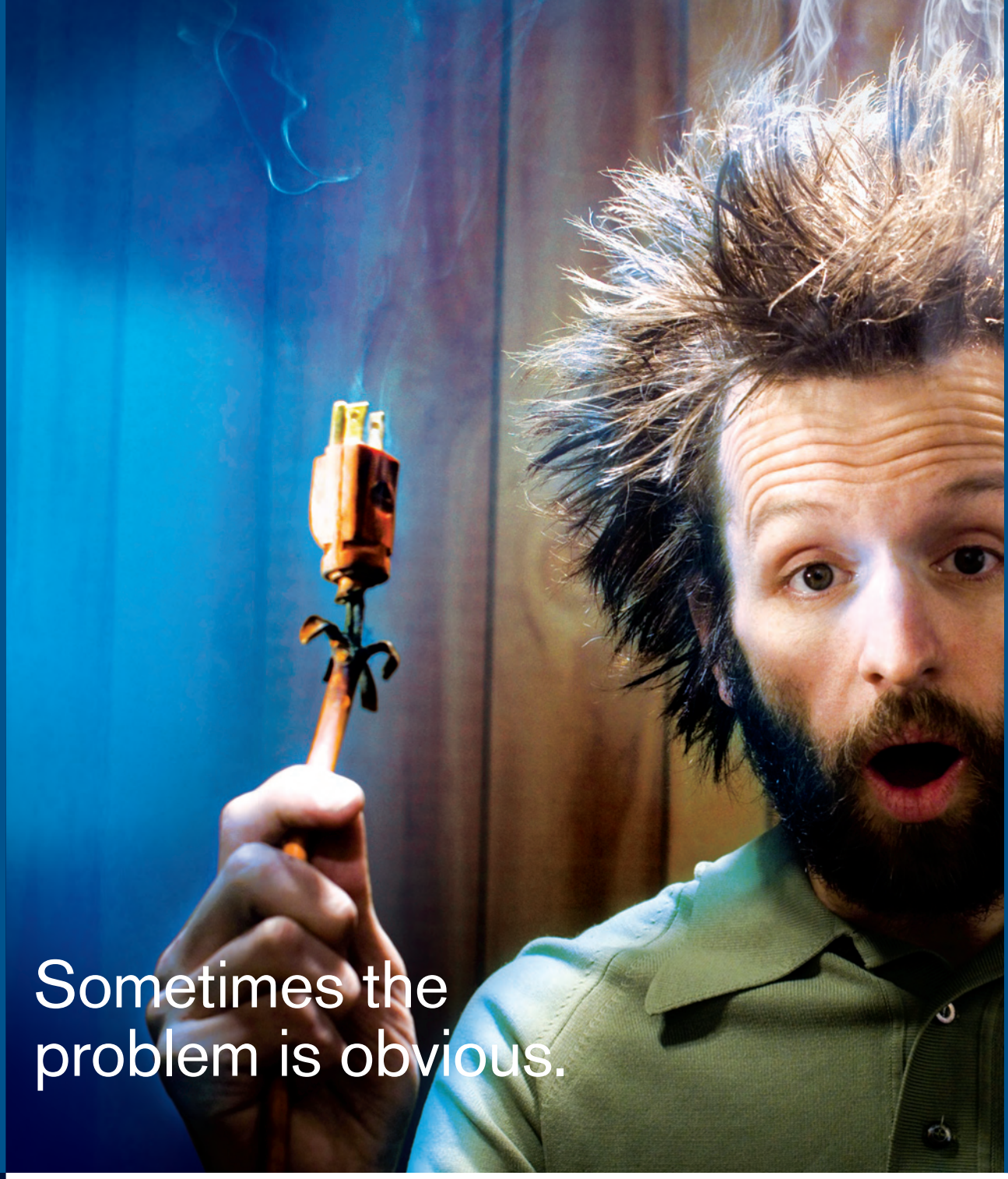
```
SELECT    x.*
FROM (SELECT a.*,
             lead(a.status,1) over
              (PARTITION BY a.dev_id
               ORDER BY a.evt_dt ASC) AS next1_stat,
             lead(a.status,2) over
              (PARTITION BY a.dev_id
               ORDER BY a.evt_dt ASC) AS next2_stat,
             lead(a.status,3) over
              (PARTITION BY a.dev_id
               ORDER BY a.evt_dt ASC) AS next3_stat,
             lead(a.status,4) over
              (PARTITION BY a.dev_id
               ORDER BY a.evt_dt ASC) AS next4_stat,
             MIN(CASE WHEN a.status = 0
                      THEN a.evt_dt
                      ELSE to_date('1-jan-2099')
                 END) over
              (PARTITION BY a.dev_id) AS min_err_dt
      FROM   dev) x
WHERE     x.evt_dt = min_err_dt;
```

In the real world

This example is based on a real world occurrence. The event data being analyzed was 290 million rows and the PL/SQL version ran for 24 hours when it was killed without completing. The analytics statement did the job in less than 2 hours. Although I don't have proof, I'm pretty sure it took me less time to write and test the analytic statement than it did for the other person to write and test the PL/SQL.

Summary

Oracle Analytics is not just for statistical analysis, reporting, OLAP or any other pigeonhole. It is for everyday use by developers who want to provide quality data as efficiently as possible. It is not hard although it takes a little practice. The best way to learn it is to do it…so get busy.



Usually, it's harder to pinpoint.

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

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

A smarter solution makes everyone look brilliant.

CONFIO SOFTWARE

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

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

Value of Embedding Oracle Technologies

Shig Hiura, Oracle Corporation

INTRODUCTION

Oracle has long been the leader in high-performance, reliable and scalable database management services for mission-critical applications in the data center. Today, though, mission-critical applications are deployed not only in the data center, but also on mobile devices, in the network infrastructure and on special-purpose systems. Oracle's commitment to providing best-in-class data management products extends from the data center to edge and embedded applications.

Data Is Everywhere

Applications today have moved beyond the data center. They run at the "edge" – in routers and switches for networking, communication services and OSS/BSS systems for telecommunications, handhelds and data capture devices for mobile field workers, monitoring and control systems in vehicles, and appliances and entertainment devices in consumers' homes. Like the data center applications that preceded them, these applications need fast, scalable and reliable storage services for the data on which they operate. Unlike those data center applications, however, these new applications must manage themselves, running with local data and without a systems administrator.

Oracle's broad embedded database product line satisfies the diverse demands of this new generation of applications. The company's expertise in fast, scalable and reliable database management is now available to the innovators building the next generation of applications at the edge.

The Challenge

In order for your product to succeed in today's world where customers have more choices, your product has to be better in some way. More differentiated capabilities, better performance, easier deployment, and lower cost. And the reality is that you have limited time, money and resources to build your product.

No matter what you are building, whether it is hardware or software, whether it is simple or complex, your application likely needs to manage data in some way. You should consider an embedded database if your application needs to:

- Store data internally
- Run unattended
- Install and configure the database silently
- Deliver lower TCO by minimizing your customer's costs
- Be a complete, turnkey solution

If this is your situation, an Oracle embedded database may be the ideal solution.

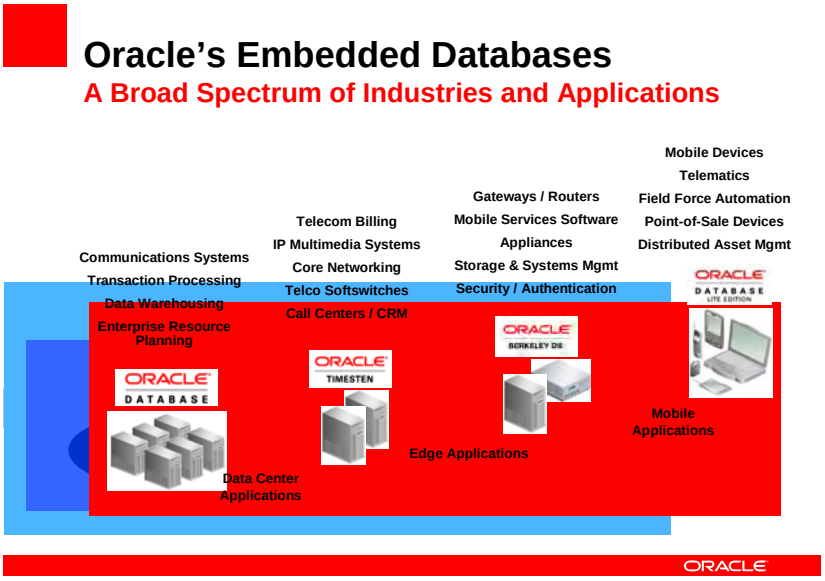
Build or Buy?

There is simply no reason to build embedded data management capability yourself, when a range of proven, commercial off-the-shelf (COTS) products are available from Oracle. Oracle offers you choice and flexibility to meet your unique technical requirements, and our COTS offerings will greatly reduce your engineering cycle/time, cost and risk, while allowing you to focus your scarce resources on your application and its unique value-add, instead of the underlying data management.

The simple fact is that robust data management is hard. Outstanding performance, especially under unpredictable workloads, requires long-term, continuous analysis and tuning. Building highly available, reliable systems that survive failures gracefully requires deep expertise and years of real-life field deployments. Oracle's embedded database products have been proven through millions of mission-critical deployments.

Oracle's Embedded Database Offerings

The demands on a data management solution vary widely with industry and application. Some applications may have well-defined, repeating data access patterns while others require arbitrary queries and manipulations. Some require support of specific data types (such as XML) while others subscribe to traditional content such as plain text and numbers. Some applications require advanced capabilities such as replication and caching while others have a simpler usage model. Some applications run in highly constrained footprints, while resources are not an issue for others. As you can see from these examples, there are a broad variety of database requirements, and no single database can satisfy them all.



To address the variety of requirements for data management solutions, Oracle offers a portfolio of embeddable database products for every industry and application. From enterprise-scale embedded databases to turnkey appliances and handheld devices, embedded Oracle databases supports applications that require robust data management capabilities on the smallest platforms to the largest.

PRODUCTS

Oracle RDBMS

While the Oracle RDBMS is popular in the data center, it is not widely known as an embedded database platform. There are some key technical challenges involved with embedding a database into an application. With Oracle Database 10g/11g, Oracle has made large strides towards creating a truly embeddable database.

These challenges can be classified in the following categories:

Deployment

Installation and configuration of an embedded database must be completely invisible to the user and fully integrated with their application.

The database has to be packaged as a component of the application package, and must be installed as part of the application installation.

The Oracle RDBMS has significantly simplified installation and configuration. The install has been made much faster and lightweight. It is installable from a single CD, which contains binaries for the Personal, Standard and Enterprise Editions along with a pre-created seed database. Oracle installation has been extended to perform pre-requisites check to make sure the target system has the requisite OS patches, memory, CPU etc. The Oracle Universal Installer can be run in a true silent mode for installing and de-installing the software. Along with silent installation the Oracle RDBMS also supports a silent mode de-installation.

Management

An embedded database must be self-managing. All routine administrative tasks must be automated. In addition, the database must be able to adapt to changes in system environment and it should be able to address common problems automatically without requiring any outside intervention. In rare eventuality of errors or problems, the database should provide an easy way to diagnose the problem and recommend possible remedies.

The Oracle RDBMS incorporates significant enhancements to its automated self-managing capabilities, making it ideal for lights-out environments where a DBA is not available. Tasks such as:

- Backup and recovery management
- Space management
- Memory tuning
- Error handling
- Problem diagnostics and resolution

can be automatically addressed through facilities such as RMAN, Flash Recovery Area, Server Generated Alerts, and Automatics Database Diagnostic Monitor (ADDM), to name a few.

Software Maintenance and Support

Software maintenance is a necessary evil that all software applications must deal with. For embedded databases, it is essential that tasks such as upgrades and software patching must be made very simple and as automated as possible.

Oracle makes continuous improvements to its OPATCH and Database Upgrade Assistant (DBUA) to provide silent, scripted and template driven patching and upgrades so that these actions can be performed "behind the scenes" as part of the application's patching/upgrade process. In addition, continuous improvements in the Oracle RDBMS are made so that downtime is minimized during these activities.

Additional details on embedding the Oracle RDBMS can be found in the whitepaper entitled, "The Self-Managing Database: Deploying Oracle Database 10g in Embedded Environments", and can be found at the following URL:

http://www.oracle.com/technology/tech/embedded/pdfs/twp_manage_invisible_database.pdf

TimesTen In-Memory Database

The Oracle TimesTen In-Memory Database is a memory-optimized relational database that delivers very low response time and very high throughput for performance-critical systems. It is targeted to run in the application tier, close to applications, and optionally in process with applications. It may be used as the database of record, or as a cache to the Oracle RDBMS.

The Oracle TimesTen product line consists of a base product, the Oracle TimesTen In-Memory Database, with two options:

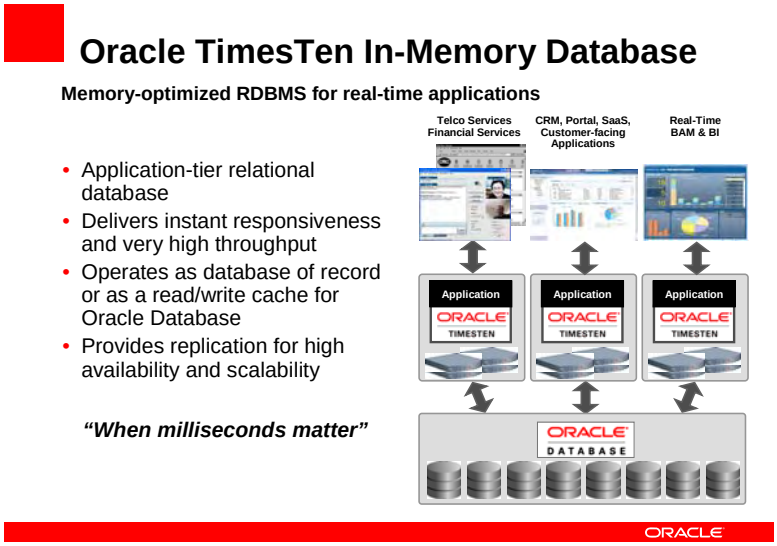
- Replication: TimesTen to TimesTen
- Cache Connect to Oracle

Replication – TimesTen to TimesTen

TimesTen Replication facilitates real-time copying of data between databases. The fundamental motivation behind Oracle TimesTen replication is to make data continuously available to mission critical applications with minimal performance impact. In addition to its role in failure recovery, replication is also useful for distributing user loads across multiple databases for maximum performance and for facilitating online upgrades and maintenance.

Cache Connect to Oracle

Cache Connect to Oracle creates a real-time, updatable cache for Oracle data, residing in the application tier. It offloads computing cycles from backend systems and enables remarkably responsive and scalable real-time applications. Cache Connect to Oracle loads a subset of Oracle data into TimesTen, propagates updates in both directions, automates pass through of SQL requests for non-cached data and automatically resynchronizes data after failures.



Berkeley DB

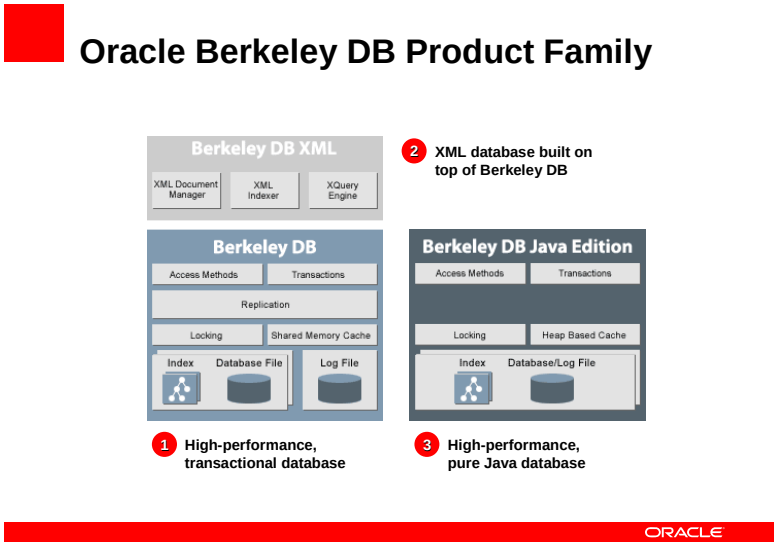
The Oracle Berkeley DB family of high performance, open source, embeddable databases allows developers to embed in their applications a fast, transactional database engine with industrial grade reliability, scalability and availability. For Independent Software Vendors, device and equipment manufacturers, and enterprises building applications for internal use, the Oracle Berkeley DB family of products provides fast, local persistence with zero administration.

The Oracle Berkeley DB products link directly into your application, eliminating the performance penalty of client-server architectures. Berkeley DB provides simple function-call interfaces, allowing programmers to store and retrieve information quickly and easily. Data is stored in the application's native format, eliminating the need for translation or mapping.

Berkeley DB supports full ACID transactions and recovery for data integrity, multiple processes and multi-threading for high concurrency, and replication for high scalability and availability. It is lightweight, designed for high performance, scales up to carrier and enterprise class applications and scales down to mobile devices and other constrained environments.

Berkeley DB is very flexible and configurable, giving the application developer control over how resources are allocated, the amount of memory dedicated to caching records, the degree of concurrency, support for recovery and more. It includes full source code for easier porting, integration, debugging and optimization.

It is also important to understand what Berkeley DB is not. It is not a database server that handles network requests. It is not an SQL engine that executes queries. It is not a relational or object-oriented database management system. It is an embeddable database engine that has been designed to be simple, fast, small and reliable.

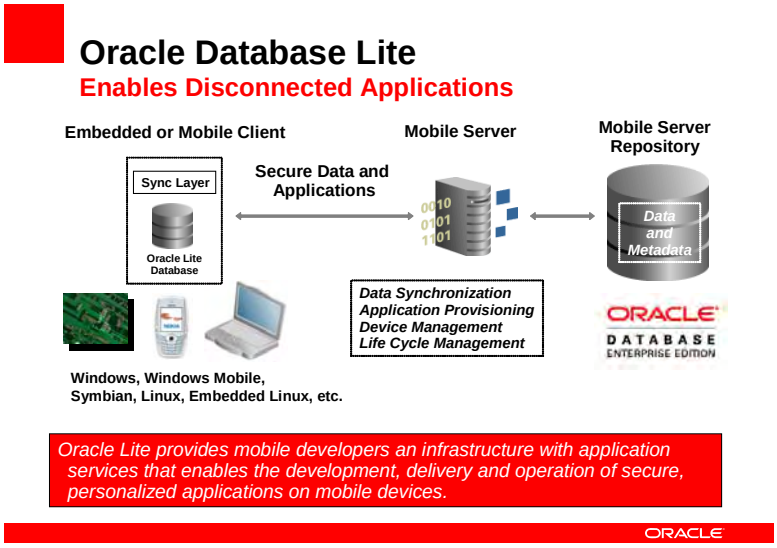


Oracle Lite

Employees are increasingly working away from their desks and require access to corporate data normally found only on desktop devices connected to enterprise networks. Wireless connections for mobile users offer the promise of remote access to enterprise data but persistent wireless connections are not always possible, practical or desirable.
Wireless connections are not always possible because mobile workers may be in an environment that does not have wireless coverage. Often it is cost-prohibitive and impractical to use a persistent wireless connection because applications do not have a real-time data access requirement. Finally, a wireless connection may not be desirable because the very presence of a wireless connection could compromise an application or user's security.

Developers require an end-to-end infrastructure with application services that enables the development, delivery and operation of secure, personalized applications on mobile or embedded devices.

Oracle Database Lite (Database Lite) is a complete solution and includes more than a small footprint client database. Oracle Database Lite 10g is a full-featured, integrated infrastructure for mobile application development, deployment and management. It extends the grid environment by enabling mobile access to applications that rely on information maintained by the enterprise Oracle database in the grid.
The main components of Oracle Database Lite are 1) client stack (featuring the lightweight database) available on multiple platforms including various flavors of Windows 32-bit, Windows Mobile, Linux and Symbian OS 2) Mobile Server for synchronization and scalable deployment and management of applications, users, and devices and 3) developer tools that enable quick and simple application development.



Mobile Client

The Oracle Database Lite client includes various components that operate in concert to facilitate a seamless user experience and easy remote administration. The client includes a small-footprint database (and associated utilities) optimized for handhelds, laptops and small business environments, supporting single or multi-user deployments. The client also provides rich data synchronization support: a synchronization agent that synchronizes automatically in the background, a GUI application that allows manual invocation of synchronization (mSync) and APIs that can be invoked by your application. The device agent on the client allows administrators to remotely manage the device by sending commands or querying system and application data. Finally, an update utility supports application life-cycle management by allowing new versions of the application to be downloaded by mobile workers in the field.

Mobile Server

The Oracle Database Lite Mobile Server is required in the middle tier to allow mobile clients to synchronize data with the enterprise database. In addition, it provides life cycle management functionality for deployment and management of applications, users, and devices.

Mobile Development Kit

Oracle Database Lite provides critical support for rapid development and deployment of off-line mobile applications. Developers can install the Mobile Development Kit (MDK) from the Oracle Database Lite installation medium and install not just the client database and the Database Lite runtime binaries, but also utilities and GUI based tools (MDW and Packaging Wizard) that simplify describing, deploying and debugging a mobile application. Furthermore, the Oracle Database Lite mobile development kit contains code samples that accelerate the development of mobile applications.

CHOOSING THE RIGHT EMBEDDED DATABASE

Selecting the right database will depend on application and deployment platform requirements.

Use Oracle RDBMS when:

- Advanced features (spatial, triggers, stored procedures) are required
- Clustering via RAC or integration with Fusion Middleware is required
- Footprint is not a constraint

Use TimesTen In-Memory Database when:

- Extremely Low latency and high throughput required

- Oracle caching is needed
- Flexible data access via SQL is required
- Working data set fits into memory

Use one of the Berkeley DB Family products when:
- Low latency and high throughput required
- Data access is predictable
- XML/XQuery is required

Use Oracle Lite when:
- Devices may be occasionally disconnected from the network
- Synchronization to Oracle RDBMS is required
- Small footprint is required
- Flexible data access via SQL is required

CONCLUSION

According to IDC's 2007 report, Oracle is the market leader in embedded databases worldwide. Thousands of leading software ISVs and hardware OEMs embed Oracle technologies into their products. There are several hundred million deployments of embedded Oracle technology around the world, and Oracle offers developers the broadest selection of embeddable databases to fit their diverse requirements.

Oracle Instant Client for Windows XP

Eunhee Lee, Return Path, Inc.

Introduction

Recently Oracle introduced the Instant Client as an alternative to full installation of the Oracle Client to run your own applications or SQL*Plus.

I learned about this at the last NYOUG meeting and I tried to install it on a Windows XP platform; however there are a couple of useful troubleshooting tips to be aware of when connecting to a remote Oracle database, requiring light installation of small libraries on Windows using Oracle Instant Client.

What is Oracle Instant Client?

The Oracle Client must be installed if you wish to access an Oracle database. Oracle Instant Client allows you to run your applications without installing the standard Oracle client, and uses significantly less disk space. Even SQL*Plus and ODBC can be used with Instant Client.

In order to install the Standard Oracle Client**,** use the following steps:

1. The Oracle Universal Wizard is used to install the Oracle client software with the proper modules.
2. Check for space on the hard drive. You do not want to get an "out of space " message at the end of the installaction. Oracle 8*i* needs about 300MB, and 9*i* needs 1GB.
3. Configure Oracle Net connections to the databases
4. It takes more than 20 minutes to finish the installation.

Download Oracle Instant Client

Download the Basic and SQL*Plus Instant Client packages from the following website: (http://www.Oracle.com/technology/tech/oci/instantclient/index.html)

You also might want to go ahead and download all of the other packages and keep them together as a single version of this product. Even if you do need one of the additional support libraries, they may come in handy if versions become an issue and download very quickly.

The contents of the basic package are shown in Figure 1.

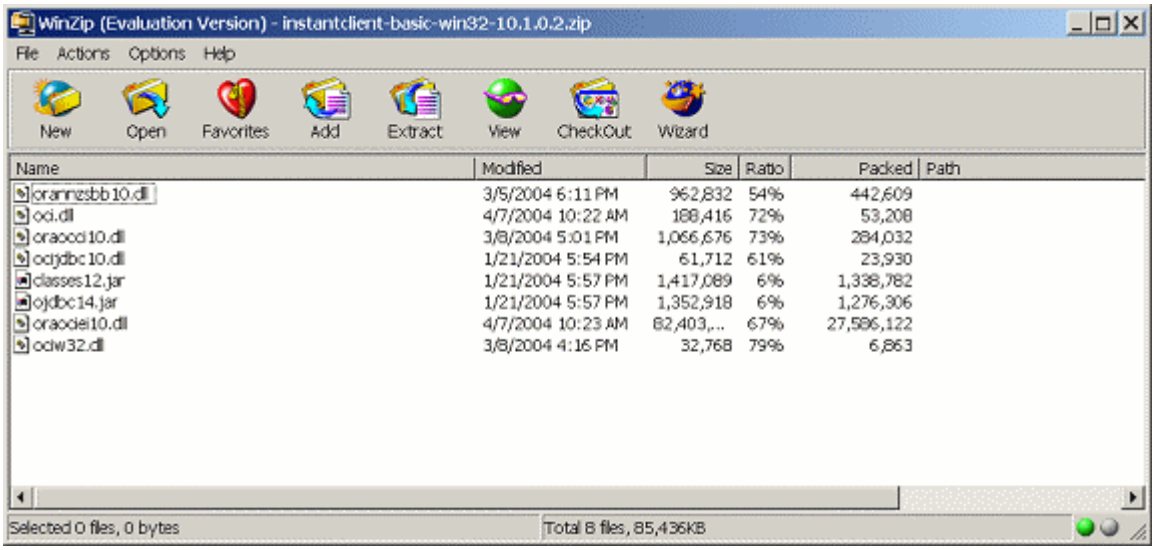


Figure 1: Basic Package Contents

The contents of the SQL*Plus Package are shown in Figure 2.

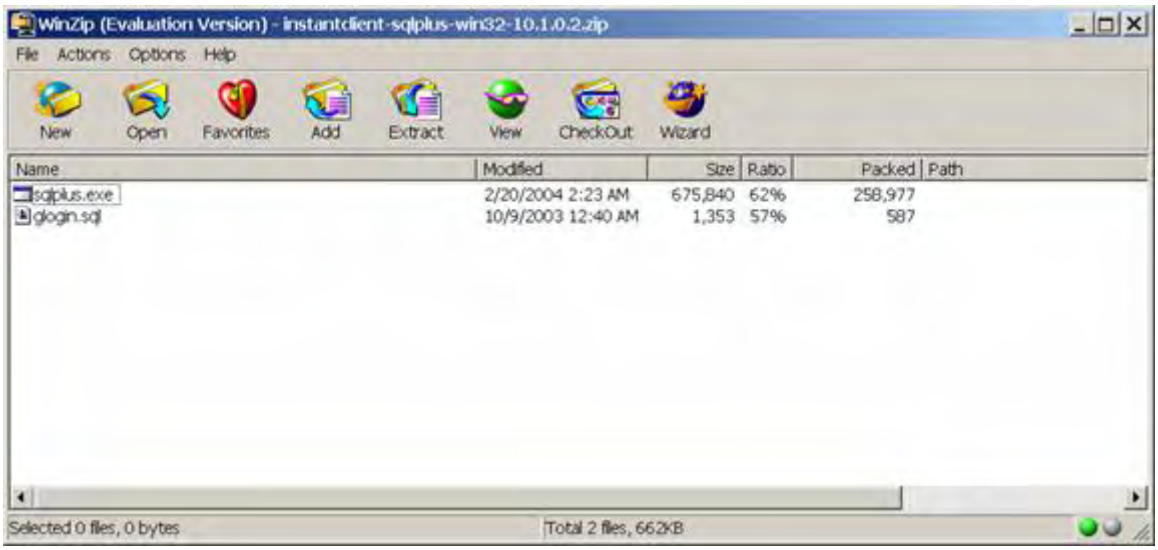


Figure 2: SQL* Plus Package Contents

Installing Oracle Instant Client

Use the following steps to install Oracle Instant Client (Figure 3 shows the contents of the Instant Client Directory)

1. Download the appropriate Instant Client packages for your platform. (All installations REQUIRE the Basic or Basic Lite package.)
2. Unzip the packages into a single directory such as "instantclient."
3. Set the library loading path in your environment to the directory in Step 2 (e.g. "instantclient").
   - UNIX platforms = LD_LIBRARY_PATH
   - On Windows = PATH should be used
4. Start your application and enjoy.

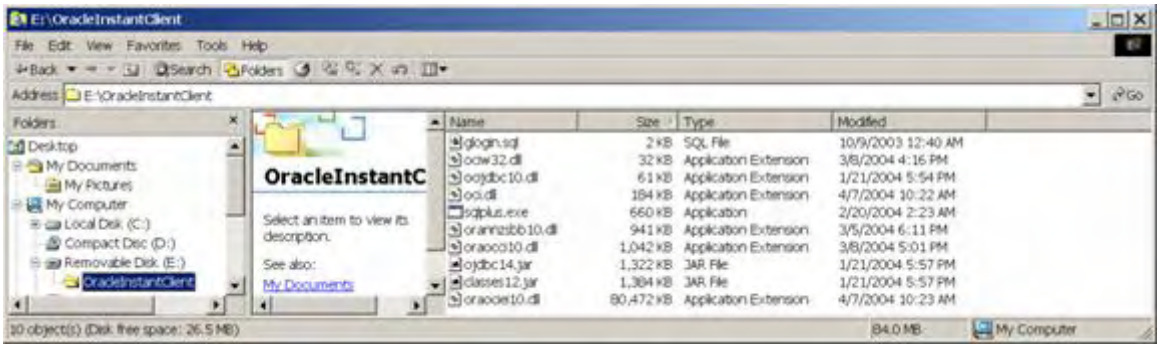


Figure 3: Instant Client Directory Contents

Set Environment Variables

These directions assume that your client directory is located in drive E:

1. SET PATH=E:\OracleInstantClient;%PATH%
   - This is used for finding the executables.
2. SET TNS_ADMIN=E:\OracleInstantClient
   - This is only for finding the tnsnames.ora file. You do not have to have this in the Instant Client directory but if you were on a database box or one that already has the Oracle Client on it, where else would you put it?
3. SET LD_LIBRARY_PATH=E:\OracleInstantClient
   - Used to find the shared libraries shipped with Instant Client.
4. SET SQLPATH=E:\OracleInstantClient
   - If you use a glogin.sql file, you will need to set this to find it.

*Connection Methods for SQL*Plus*

Whether you install the Instant Client on the database server or on a true client machine, you will always need to specify some form of an Oracle Net connection name or identity. Here are some examples.

1. After modifying the tnsnames.ora file that is in the directory pointed to by TNS_ADMIN you can use the following
   - Method: sqlplus <user>/<password>@<tns_entry>
   - example: sqlplus scott/tiger@tnsname
2. If you would rather not use the tnsnames.ora naming, you can bypass this by using a more descriptive connect string.
   - Method: sqlplus <user>/<password>@//<machine>:<port>/<service_name>
   - example: sqlplus scott/tiger@//server.domain.com:1521/tnsname

Using ODBC

Keep the following in mind when using ODBC:

1. There is a bug when installing the ODBC requiring you to copy the mfc71.dll and msvcr71.dll.
2. Copy them in c:\winnt\system or the instant client (not winnt\system) directory
3. Reference : **http://forums.Oracle.com/forums/thread.jspa?threadID=334846&tstart=0**

Other Considerations

When using the Instant Client, you may or may not encounter the following in your particular environment.

Connecting as SYSDBA/SYSOPER	You will need to put a password file on the database server to which you are connecting.
NLS_LANG	For Instant Client, you do not reference an $ORACLE_HOME and thus there is no setting for NLS_LANG. You may need to set this as an environment variable.
Uninstalling Instant Client	Just remove the directory where you unzipped the executables and libraries. You can also unset/remove the environment variables.

The Instant Client is available on a variety of platforms such as Linux, Solaris, HP, IBM AIX, and Windows and supports the following connection types depending on the packages you download.

Basic	OCI, OCCI, and JDBC-OCI applications
JDBC Supplement	XA, Internationalization, and RowSet operations under JDBC
SQL*Plus	SQL*Plus
ODBC	ODBC

The installation procedures for each of these packages at least reads as quick and easy as the SQL*Plus Just be sure to visit and read the release notes for the specific packages, as there are additional, be it minor, variables or installation procedures to follow.

Conclusion

There are a number of benefits associated with connecting your applications to an Oracle instance using Instant Client:

1. It is free
2. It downloads quickly.
3. It uses a small software footprint.
4. There is no reliance on a typical Oracle CD installation
5. It is very easy to deploy.
6. There is no loss of features from the full-blown client version.
7. Vendors may package their applications more easily without reliance on customer side installation.

About the Author

Eunhee Lee is a Senior Database Architect at ReturnPath Inc. in New York City with over 10 years of Oracle Database and software development experience. She earned her Master's Degree in Computer Science and Engineering at HongIk University, South Korea and New York University.

For Mission Critical Systems, Do You Have Both Eyes Open?

For your Oracle systems, see how GoldenGate gives you one solution for both high availability and real-time data integration.

- **Oracle 8i or 9i to 10g/11g migrations** – No Downtime
- **Active-Active** – Highest Availability
- **Direct feeds for Data Warehousing** – Enable Operational BI
- **Off-load Data to a Real-Time Reporting Database** – Better Performance
- **Heterogeneous Data Replication** – Extremely Flexible

With GoldenGate, you solve many business needs with one technology.

- Global Headquarters: +1 415 777 0200
- sales@goldengate.com
- **www.goldengate.com**

GOLDENGATE®

Real-Time Access to Real-Time Information

End User Monitoring with Oracle Enterprise Manager

Nadu Bharadwaj, Oracle Corporation

Executive Overview

Service providers face increasingly demanding customers who want to perform tasks faster, remotely and accurately. The goal of every provider is to provide a high quality of service to their consumers so that the potential rewards of satisfied customers, increased revenues, cost savings, and greater efficiency are realized. Insight into how customers experience services can be the basis for aligning the IT infrastructure to the business objectives of the organization. In turn, this alignment improves the delivery and cost effectiveness of the key processes and services that drive competitive advantage. Oracle Enterprise Manager provides breadth and depth of solutions to manage end user experience. With Enterprise Manager, service providers have a holistic solution that addresses the challenges of optimizing application service levels, rising administrative costs, and maintaining customer satisfaction while maximizing company profits. Administrators have the ability to monitor their business systems from the top down and trace the experience of their end-users as they enter and navigate through applications. Now, end user experience can be monitored from an active or a passive standpoint. With active monitoring, administrators are able to pre-empt performance problems before they impact end-users and affect business profitability. With always on passive monitoring of critical business applications, the business users as well as administrators can identify application issues that reduce revenues or increase costs, respectively. Powerful reporting and dashboard capabilities aligned with simplified and expedient root cause analysis round out the solutions from Enterprise Manager.

Service Level Management with Grid Control

Enterprise Manager Grid Control employs a service level management methodology that comprehensively monitors and manages services from an end-to-end perspective.

Service Modeling

A “service” in Grid Control is defined as an entity that exposes a useful function to end-users. Grid Control enables modeling of real world services by mapping the end user’s view of a service to its underlying IT infrastructure. Services can be of different types- Web application, Forms application, or Generic and Aggregate service. Services can consist of one or more ‘service tests’ defined using any of the supported service protocols. Supported service test types include: infrastructure (DNS, FTP), Web (HTTP/S transactions, SOAP, HTTP Ping), mail (POP, SMTP, IMAP, LDAP, NNTP), network infrastructure (ICMP ping, port checker), database (TNS Ping, JDBC Script, SQL Timing), Oracle Forms transactions, and custom scripts.

Complex service types that can be modeled include packaged applications such as Oracle Applications, including E-Business suite, Siebel and Peoplesoft, as well as Oracle Collaboration Suite. Custom Web based and Forms based applications can also be modeled. Administrators are also able to model complex Aggregate Services, which are comprised of one or more subservices.

In addition to the wide variety of types of services that can be modeled, services can also be represented graphically via the Topology view. The Service Topology feature enables visual modeling and viewing of the dependencies of a service including its subservices and the systems on which they run. The Services Topology provides an at-a-glance view of all critical components of each and every service and subservice, as well as points of failure within the service infrastructure and the interdependent impact of these failures.

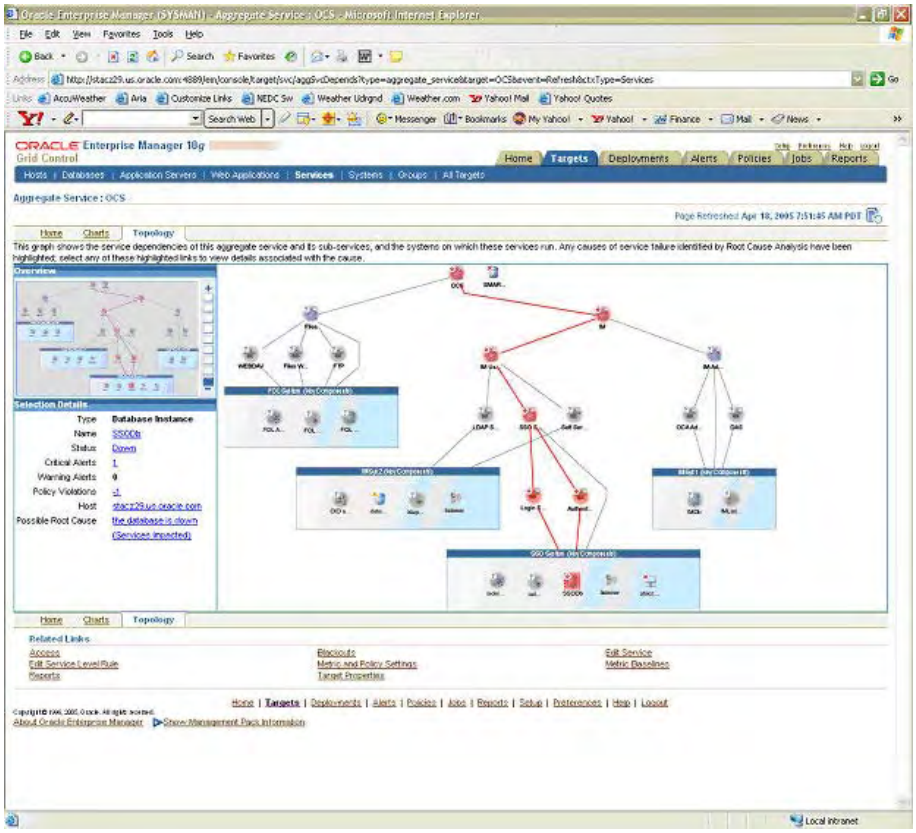


Figure 1: The Services Topology shows both the topology and status of Services and their system components as well as the relationships between them, enabling application administrators to quickly check the overall status, and view the cause and impact of an infrastructure failure.

Service Level Compliance

Service level compliance is measured to ensure contracted SLA's are being met. Grid Control defines Service Levels as the percentage of time during business hours that a service meets specified availability and performance criteria. Grid Control allows customization of a service levels based on several criteria:

- Business hours, which are defined as the specific days of the week and timeframes on those days during which the application must function without availability or performance issues.
- Performance metrics affecting the availability of the service.

Grid Control displays the actual service level percentage and compares it with the user-defined expected service level goal. Additionally, using the Services Dashboard, administrators can also determine whether or not service levels are compliant with business expectations and goals.

Service Availability and Performance Monitoring

Proactive monitoring of service levels ensures that the availability and performance for critical business hours can be provided to end-users. The proactive approach employs the use of service tests that are run from remote beacon locations. Because a slowly responding application is often as unavailable as one with failed components, administrators can set response thresholds that determine acceptable performance of a service. Warning and critical thresholds can be set individually for each user community so that administrators are alerted when service response times are exceeded. Proactively monitoring transactions enables administrators to isolate the bottlenecks within a service, pre-empt problems and make corrections before users are adversely affected.

Service tests represent end user tasks and are used to determine the availability and performance of a service. For Web or Forms transactions, an administrator can define a navigation path within the application to be used as the criteria for determining the service's availability. For example, logging in to a CRM website, searching for a sales report, and

accessing the report could be considered a critical path for service availability. These critical paths can be recorded for Web based and Forms based applications, and the stored transaction or 'service test' can be launched at a user-defined interval from strategic locations across the customer-base.

Service performance as experienced by end-users can be measured through beacons. A beacon is a remote agent that can store transactions and execute them at desired intervals. Through these executions, availability and performance metrics are collected that represent end-users perspective. Response metrics collected by beacons executing the service tests is useful in identifying beacon locations that are experiencing slow performance. For example, if the beacon response time from San Francisco is experiencing slow response times, and another two locations, say London and Paris, are responding well, it may be the case that the problem is a regional San Francisco network problem, and most likely not attributed to the application itself. For Web transactions, drill-down capabilities provide a detailed breakdown of times such as connect, redirect, first byte, html, and non-html time, that was consumed at the transaction, transaction group, or individual page level.

Service monitoring can also be integrated with Oracle Business Activity Monitoring (BAM) metrics. Oracle BAM provides the ability to monitor business events from a variety of sources through active dashboards. Integration with Oracle BAM improves the visibility that business executives have on the important operational elements of the business to continually improve on their business processes and service levels provided.

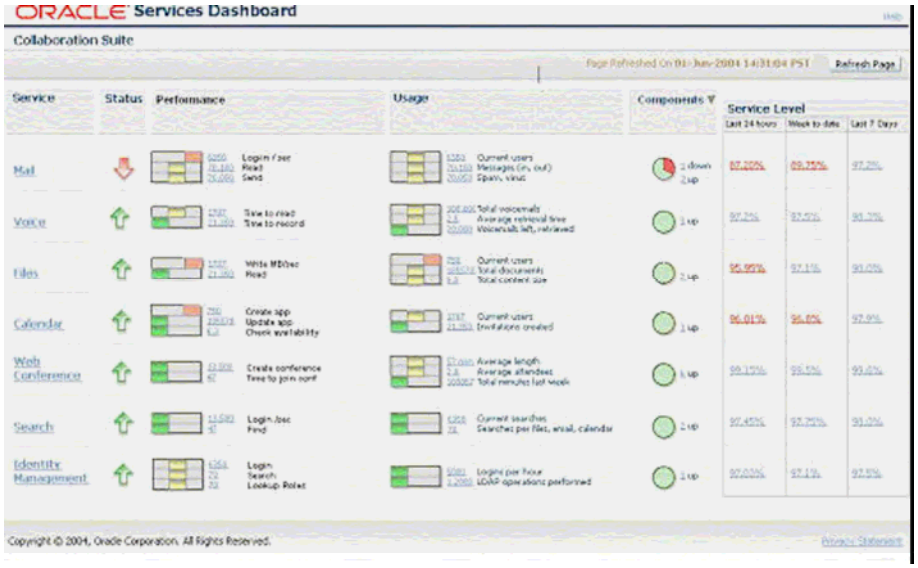


Figure 2: The Services Dashboard enables administrators, managers, and top-level executives to view at-a-glance all service level related information from a central location.

Real User Monitoring with UXinsight

Oracle Real User Experience Insight (UXInsight) is designed for measuring, analyzing, improving, and controlling the availability and performance of all of the above deployment scenarios, including e-business applications, Web transactions, and ERP systems. Its ability to see what the end-users experience, together with its powerful reporting facilities, enables direct insight into every component of even the most complex infrastructures, and delivers an information foundation that services both IT and business.

UXinsight provides enterprises with powerful analysis of network and business infrastructure. Enterprises can monitor the real-user experience passively; set key performance Indicators (KPIs) and service level agreements (SLAs), and trigger alert notifications for incidents that violate them. UXinsight comes with a library of powerful reports that provide both business-orientated and technical-orientated users with the information they need to make effective decisions.

Measuring Real-User Performance

Typically, in order to control the performance and success of a Web application, a number of Key Performance Indicators (KPIs) are defined. The most commonly used KPIs are:

- End-to-end response times of pages and individual objects (URLs).
- Server and internet/network response times.
- Error messages (both technical and functional).
- End user behavior, number of aborts, and so on.

To follow up on all of the above mentioned KPIs, and to obtain a realistic view of the status of the Web applications, companies need to measure real-user experience. To make this step possible, a number of requirements can be identified. First, traffic from all users must be monitored. This is very important because the information is used to cross reference your performance data and users groups which, in turn, can be grouped on location, department, type, status, and so on. Secondly, the data collection method must be 100% non-intrusive. It is not acceptable to disturb the current service by adding extra load on a Web server, or by installing software agents that will affect performance.

Furthermore, companies do not want to change the current application or infrastructure. When a new application release is deployed, or when an additional Web server is added, there must be no (or very limited) changes to the monitoring environment.

Network Protocol Analysis (NPA)

Typically, UXInsight is a system that is installed before the Web servers, behind a firewall in the DMZ; see Figure 3 below. UXInsight's data collection method is based on NPA technology. This data collection method is explained below.

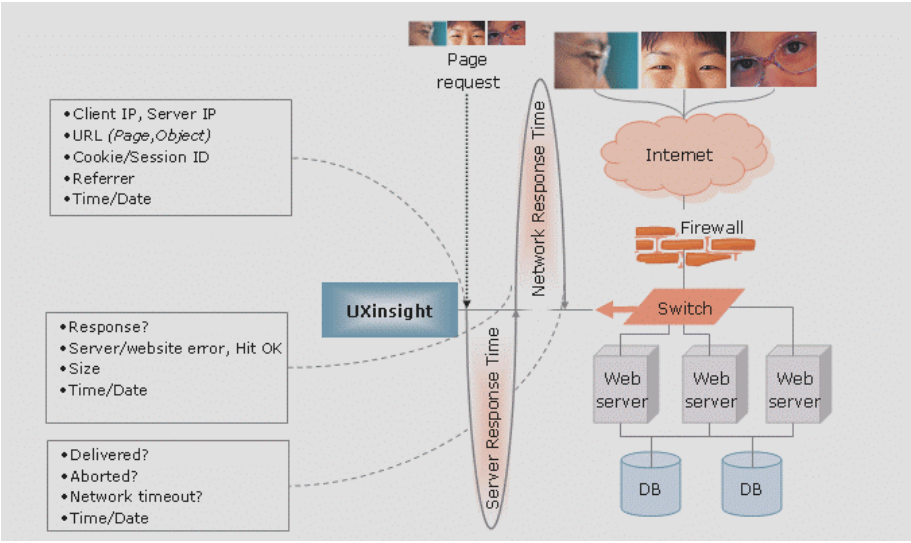


Figure 3: How UXInsight collects data.

When a visitor requests an object, UXInsight sees the request and starts measuring the time the Web server requires to present the visitor with the requested object. At this point, UXInsight knows who requested the page (IP client), which object was requested, and from which server the object was requested (IP server).

When the Web server responds and sends the object to the visitor, UXInsight sees that response, and stops timing the server response time. At this stage, UXInsight can see whether there is a response from the server, whether this response is correct, how much time the Web server required to generate the requested object, and the size of the object.

UXInsight is also able to see whether the object was completely received by the visitor, or if the visitor aborted the download (proof of delivery). Therefore, UXInsight can determine the time it took for the object to traverse the Internet to the visitor, and can calculate the Internet throughput between the visitor and the server (connection speed of the visitor). Rather than defining only a representative usage of your website, UXInsight unlocks your customers' experience while it gathers performance information. This is possible because UXInsight sees exactly how your visitors browse and experience your website.

Reporting Framework

UXinsight provides you with powerful analysis of network and business infrastructure. You can monitor the real-user experience, set Key Performance Indicators (KPIs) and Service Level Agreements (SLAs), and trigger alert notifications for incidents that violate them. UXinsight comes with a library of powerful reports that provide both business-orientated and technical-orientated users with the information they need to make effective decisions.

Checklist	
What's the dashboard reporting?	✓
Are your KPIs showing green or red?	✓
Are the most popular applications properly resourced?	✓
Are key application functions responding quickly?	✓
Are all object types being satisfactory delivered?	✓
Are pages loaded within a satisfactory interval?	✓
Are the slowest loading pages acceptable?	✓
Pages are not too complex?	✓
How well are pages being delivered?	✓
What errors are visitors seeing?	✓
What are the performance killers?	✓
Is the Web infrastructure properly balanced?	✓
Are the servers performing satisfactorily?	✓
Which users see the most errors, and why?	✓
Are transactions being completed at an acceptable rate?	✓
Is our Web site under attack?	✓

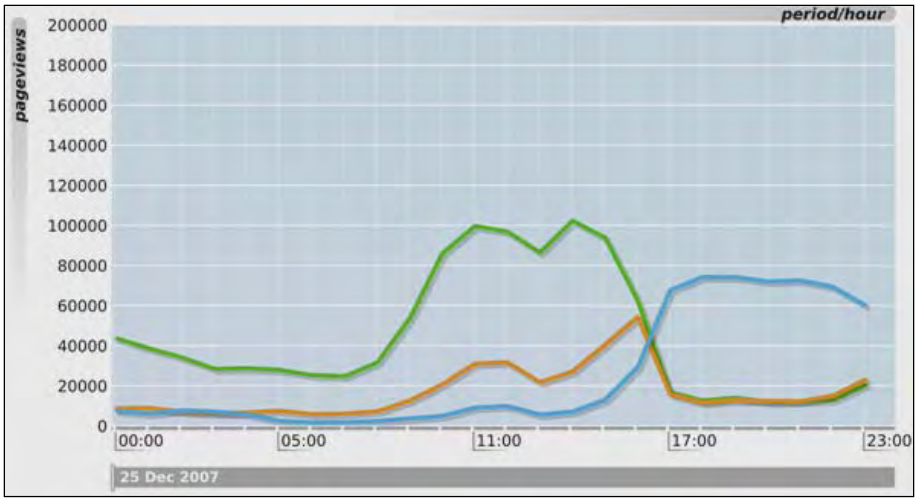


Figure 4: Page loading satisfaction.

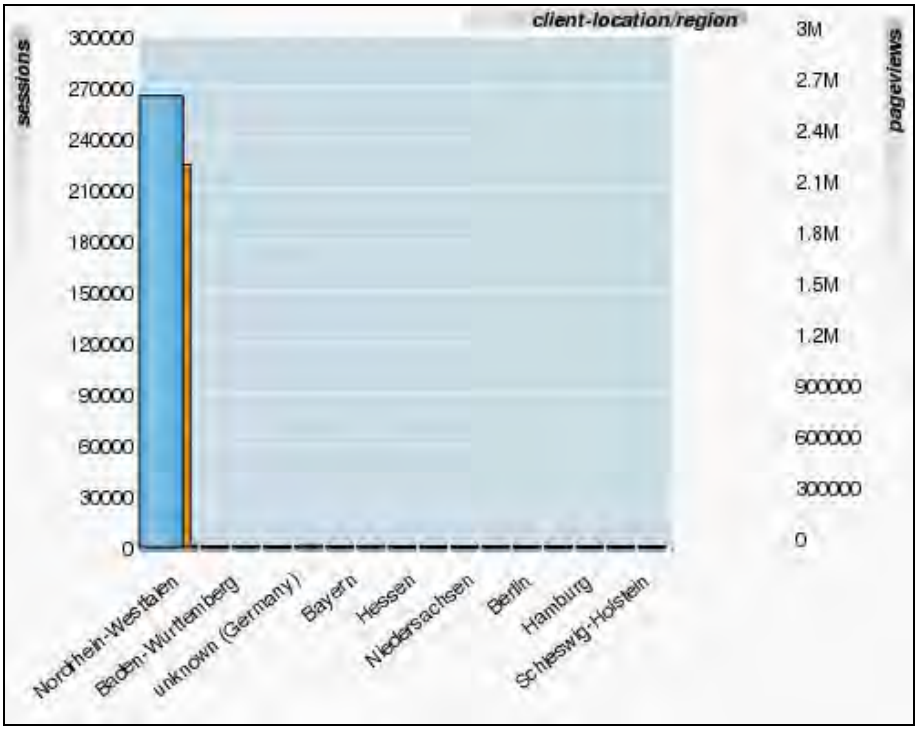


Figure 5: Visitor locations.

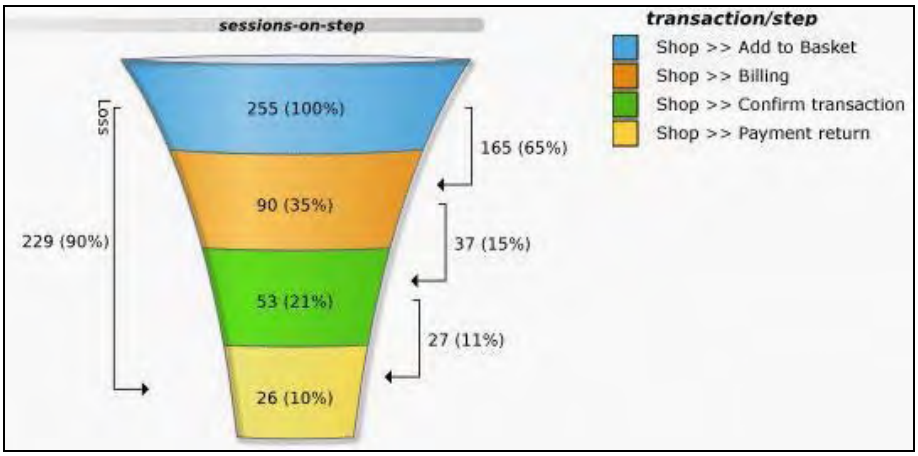


Figure 6: Transaction analysis.

Conclusion

Enterprise Manager with Grid Control and UXInsight provides a comprehensive end user monitoring solution. Enterprises can monitor end-users actively and passively. Both solutions complement each other and provide a framework for IT to measure service availability and performance, as well as real user experience while interacting with those services. The business realizes lost revenues due to users aborting transactions due to faulty application logic or bad performance.

Working with iBATIS in the Oracle Environment

Richard Ji, RealNetworks, Inc.

Introduction

This paper introduces a popular database access tool called iBATIS. iBATIS has gained it's popularity among application developers because of it's simplicity. It enables application developers such as Java developer to easily access database and map SQL result sets to Java objects without coding a single line of JDBC. It also allows them to abstract DB access so they can support multiple DBs easily.

iBATIS supports .NET and Ruby as well, but here we will just use Java as an example. The principal is the same, thus can be applied to .NET and Ruby just as well.

Though iBATIS is meant for application developers, I strongly feel that from an Oracle DBA and Oracle developer's perspective, it is important for us to know and understand these popular database access tools. For couple of reasons:

- It makes it easier for an Oracle DBA and Oracle developer to trouble shoot when an application performance issue or functionality issue arises.
- It helps Oracle DBA and Oracle developer understand how the application accesses database which is the most important part that relates to us.
- It makes it easier for a Java developer and Oracle developer to work together if the Oracle DBA and Oracle developer understands the tool which they are using.

iBATIS is an open source project from the Apache Software Foundation. The iBATIS Data Mapper framework makes it easier to use a database with Java and .Net applications. iBATIS couples Java objects with SQL or stored procedures using a XML descriptor. Simplicity is the main goal of iBATIS as opposite to other tools like Hibernate. Also iBATIS is a SQL to Object mapping tool, it is not a Object to Relational mapping tool.

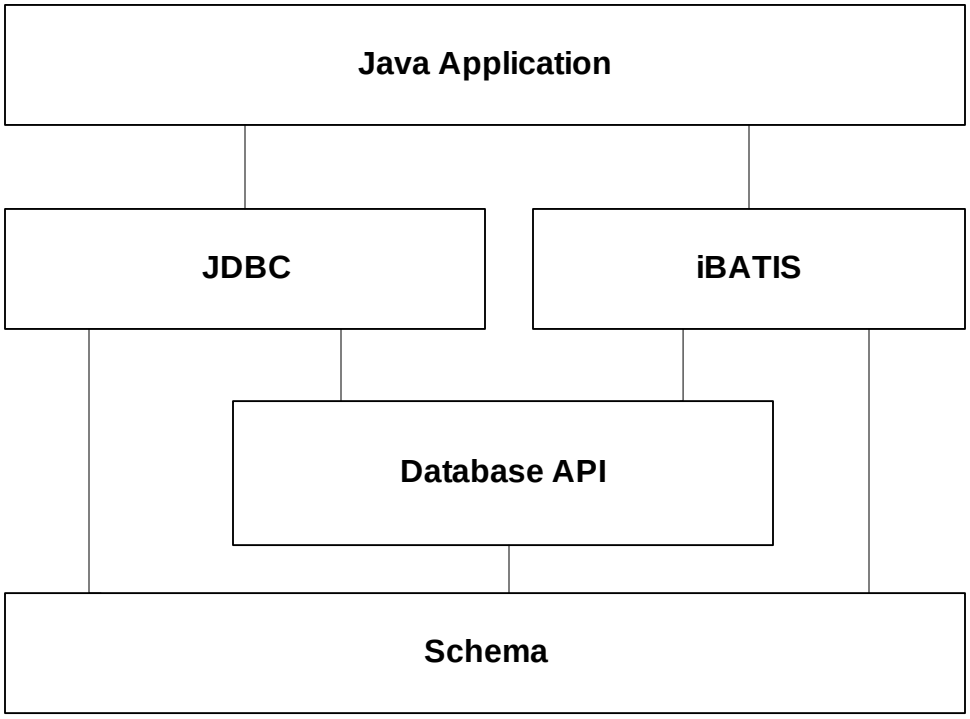
As a person who came from a DBA, Oracle developer and Java developer background. I have seen a lot of struggles between the application developers and Oracle developers and DBAs.

I have seen situations where the application developers would start implementing the application without engaging a database developer until they came to the point of needing real DB access which means they are now done coding and just need some tables created. This is frequently seen in a poorly managed project where functionality specification is poorly scoped and written, deadline are vastly underestimated and without properly consulting all the stake holders in the project. The application developer, in order to race against this deadline, will just plow on with coding, not wanting anything slowing them down, they would simply code something that read/write from a flat file or use something they can run from their PC like MySQL wherever they need data persistence. And it's usually much too late when they engage the database developer. And once the database developer finally started working on it, things start to change, tables are redesigned, SQL are taken out and replaced by stored procedures and the application developer will have to go back and modify their code. Project is now past the deadline.

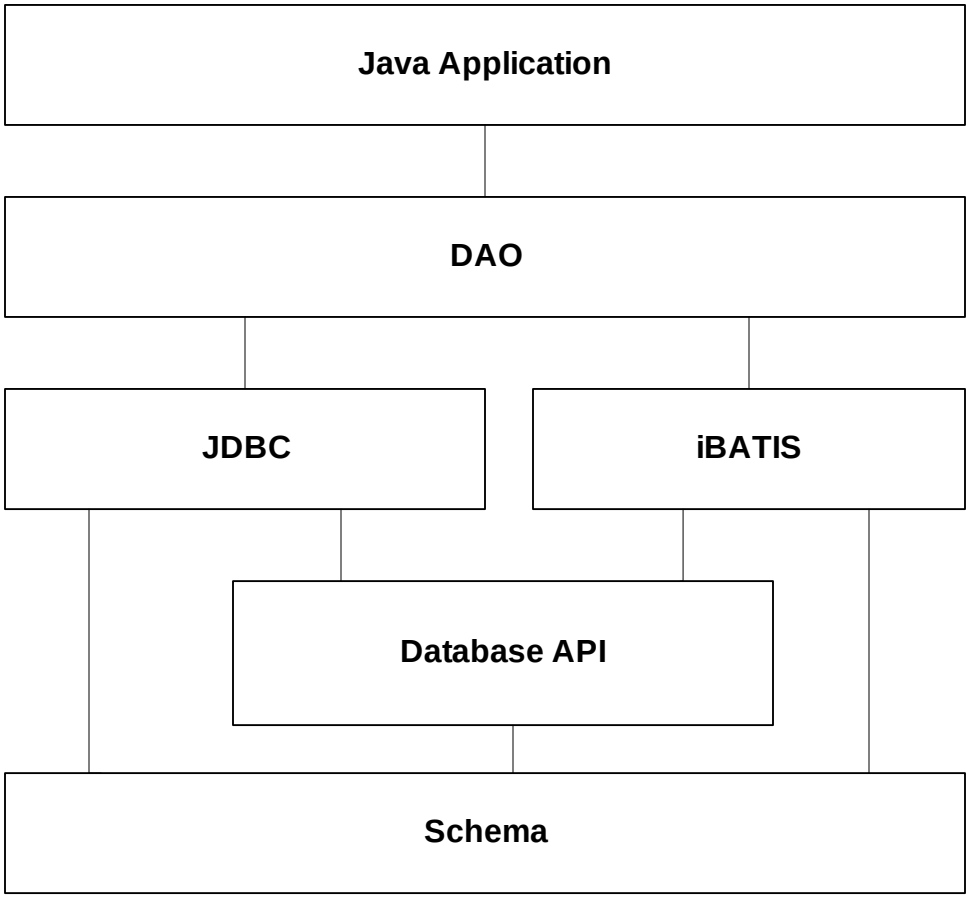
After seeing a lot of these happened, I came to realize something. We, the Oracle DBA and Oracle developer, need to take proactive steps to prevent situations like this from happening in the first place. Because when this happens, the ball is in our hands and time is not on our side. Although it's not our fault but all the pressure and visibility is on us.

So what steps should we be taken to alleviate this? Well, obviously we need to educate everyone including the managers, the project manager and the developers on the importance of database design. And the importance of involving DB team early in the application design.

But I also feel its time for Oracle professionals, especially Oracle developer to change, yet again. We need to step out of our world into their world. There was a time when all DB team and the application development team had to agree on is the database table design. Then stored procedure came and became more and more powerful, so we started building an API layer where all DB access is via stored procedure calls. And we stopped allowing developers to directly embed DML into their code. As long as this PL/SQL API layer is well defined application developers and Oracle developers can each work on their sides.



Now it's time to push it further. Rather than the DB team and application development team agreeing on stored procedure APIs, it's time to push it to the next level. This next level I am talking about is called DAO. DAO which stands for Data Access Object is a design pattern which abstracts the database access or in general any kind of data persistence. Whenever Java developer needs to access data, they call the DAO layer APIs and get objects that contain the data they need back. Inside DAO is the database access implementation. DAO can be making straight JDBC calls, using iBATIS, or other tools.



In this new paradigm, Java developer would simply describe what data is needed in terms of Java Beans. And Oracle developer can implement the DAO using iBATIS or straight JDBC calling database stored procedures. Why do I suggest this way? I think DAO is the layer which clearly separates the application business logic and database business logic. Previously, when we did it via stored procedure APIs, the Oracle developer still don't have first hand control and total control on how the data is accessed. First, there are situations where a Java developer would bypass the stored procedures and code some SQL in their code. We wouldn't know until some times it went into production and caused performance issues. So, what we do is come up with this method to hide the table with one schema and only grant execute on stored procedures to another schema and force the Java developer to access it through this other schema. Then there are times when Java developer would ask for several stored procedures when a single SP call would do and save the number of calls and network round trips between the application server and DB server. For example, I can use one SQL to return master-detail data, rather than making two separate calls. Most of the Java developer don't know that.

```
select deptno, cursor(select e.empno, e.ename from emp e where e.deptno = d.deptno) from dept d;
```

And lastly, I would trust more on an Oracle developer dealing with transactions than a Java developer.

Java developers use to code SQLJ or JDBC in their code which makes it hard for the DB team to look into it to find any trouble causing SQL. We would often have to either ask the developer to give us the SQL they are generating; or monitoring through views and third party tools; or use a capturing tool like JDBC logging or trace their session to get our hands on their SQL. With iBATIS, we know where the SQL are since they are all located inside the XML mapping files. And we can easily test changes without recompile a single line of code. Save the original XML and try out a tuned SQL,

all without involving a developer and edit a line of source code. What's better, you don't even need access to source code. So if this is a third party application, you can still play with the SQL to tune it.

Getting iBATIS

iBATIS is an open source software, all you need to do is download it from ibatis.apache.org. Just download the latest release and unpack it on your computer.

If you want to run the demo included in this paper you will need the scott sample schema. Depending on the version of Oracle, recent versions no longer create the scott schema by default so you will have to run $ORACLE_HOME/rdbms/admin/utlsampl.sql as a DBA user. This will create the scott/tiger user and the demo tables.

To use iBATIS you will need:

- JDK 1.5 or above, download the SDK from java.sun.com
- Oracle JDBC driver, can be found under $ORACLE_HOME/jdbc/lib. For 10g, it's ojdbc14.jar, for 11g it's ojdbc5.jar or ojdbc6.jar.

First thing to do is make sure JDK works on your box. You type "java -version" on the command line prompt and see the following message:

```
java version "1.6.0_06"
Java(TM) SE Runtime Environment (build 1.6.0_06-b02)
Java HotSpot(TM) Client VM (build 10.0-b22, mixed mode, sharing)
```

Also make sure the Java compile works, just type “javac -version".

```
Javac 1.6.0_06
```

In Java, CLASSPATH is how Java find’s executables. So make sure your CLASSPATH environment variable includes the ibatis library "ibatis-2.3.0.677.jar" and Oracle JDBC library "ojdbc6.jar".

```
export CLASSPATH=./:/home/oracle/lib/ibatis-2.3.0.677.jar:/home/oracle/lib/ojdbc6.jar
```

And that's it, now you are all ready to start using iBATIS and do some Java coding.

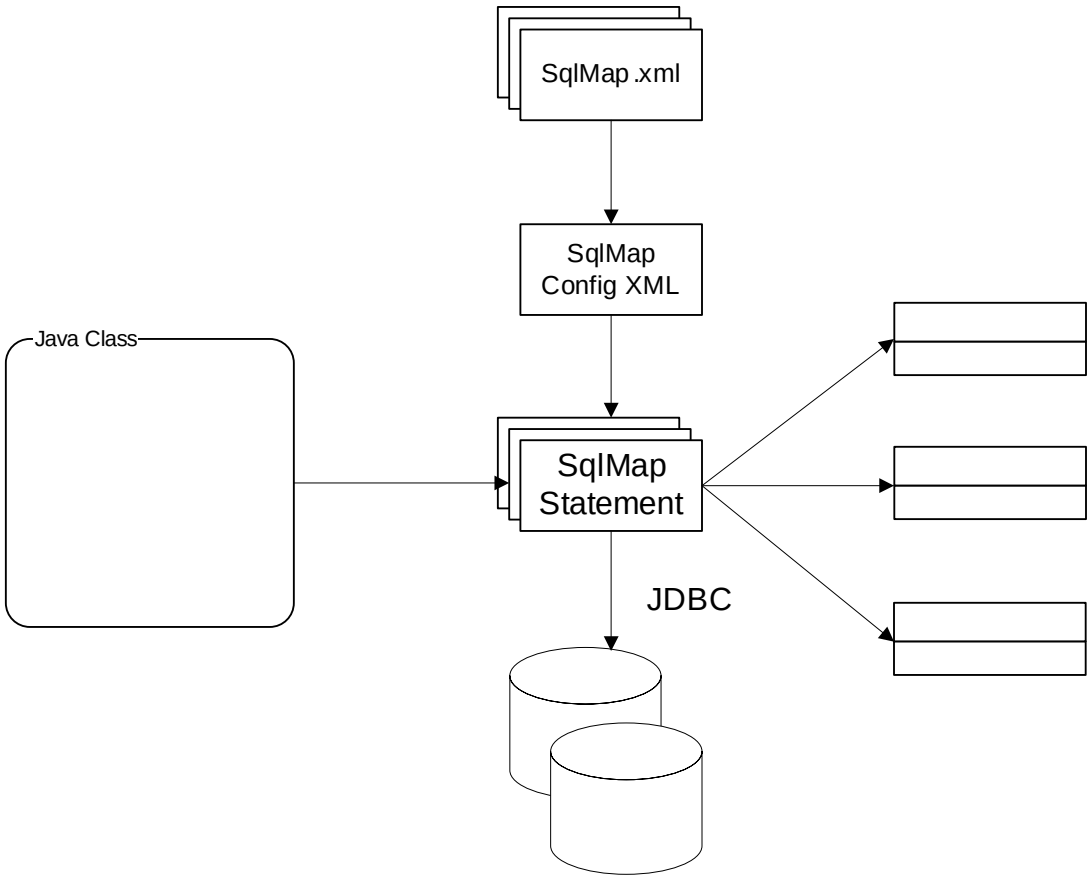
iBATIS Architecture

Before we start coding, it is important to understand the architecture of iBATIS. iBATIS works by mapping SQL and SQL Result set to Java object. All the user has to do is provide this map in a form of a XML file. iBATIS will execute the SQL, fetch the result set if there is any and map it into Java object.

For any database that provides the JDBC driver, iBATIS can access it. Thus using iBATIS can abstract the database layer. iBATIS uses XML files for SQL to Object mapping. There are two types of XML files in iBATIS.

There is the main SqlMapConfig.xml file which contains database connection information and list of other XML files for mapping varies SQL to Object. Then there is the rest of the XML files which provide SQL to Object mapping. Usually one XML file per object.

Then from Java, one would load the SqlMapConfig.xml file and whenever data access is needed, call the available procedures in the XML file using iBATIS calls. So what usually requires at least 10, 20 lines of JDBC coding, is now basically reduced down to just couple of lines.



Our First iBATIS Program

Well, the best way to learn something is start using it. So here we will create a simple Java application which accesses the EMP table from our scott schema. In this example we will use SQL first just to understand how iBATIS works. In the next example we will use stored procedures rather than SQL and use DAO as well.

First we need a SqlMapConfig.xml file. There are sample SqlMapConfig.xml inside the iBATIS software you downloaded. Here is the one I used for our example. Notice the dataSource section which contains the database connection information.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
  <transactionManager type="JDBC" commitRequired="false">
    <dataSource type="SIMPLE">
      <property name="JDBC.Driver" value="oracle.jdbc.driver.OracleDriver"/>
      <property name="JDBC.ConnectionURL"
value="jdbc:oracle:thin:@localhost:1521:ora11a"/>
      <property name="JDBC.Username" value="scott"/>
      <property name="JDBC.Password" value="tiger"/>
    </dataSource>
```

```
  </transactionManager>

  <!-- List the SQL Map XML files. They can be loaded from the
       classpath, as they are here (com.domain.data...) -->
  <sqlMap resource="Debug.xml"/>
  <sqlMap resource="Emp.xml"/>
</sqlMapConfig>
```

Next thing we need is a Java Bean for holding the EMP data that we will be retrieving from database.

Here is the Employee bean I created. This should be saved in a file called Employee.java. There are lots of information on the Web on what a Java Bean is, obviously it's outside of the scope of this paper. But in short, a Java Bean has the fields and get, set methods to store and retrieve values of the fields. Fields can be native types like int, float, or Java Objects.

```
public class Employee {
  private int empNo;
  private int manager;
  private int sal;
  private String empName;
  private String job;

  public Employee () {}
  public int getEmpNo() {
    return empNo;
  }
  public void setEmpNo(int empNo) {
    this.empNo = empNo;
  }
  public int getManager() {
    return manager;
  }
  public void setManager(int manager) {
    this.manager = manager;
  }
  public int getSal() {
    return sal;
  }
  public void setSal(int sal) {
    this.sal = sal;
  }
  public String getEmpName() {
    return empName;
  }
  public void setEmpName(String empName) {
    this.empName = empName;
  }
  public String getJob() {
    return job;
  }
  public void setJob(String job) {
    this.job = job;
  }
}
```

Now we need a XML file that maps SQL result to this Java Bean. I called this file Emp.xml. If you go back to the SqlMapConfig.xml file you will notice that I have this "Emp.xml" file listed under the sqlMap sections. Every time you create a new SQL Map XML file, you have to edit the SqlMapConfig.xml to include it there. Otherwise it won't be recognized.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMap
    PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Emp">

  <!-- Result maps describe the mapping between the columns returned
       from a query, and the class properties.  A result map isn't
       necessary if the columns (or aliases) match to the properties
       exactly. -->
  <resultMap id="EmpResult" class="Employee">
    <result property="empNo" column="EMPNO"/>
    <result property="manager" column="MGR"/>
    <result property="sal" column="SAL"/>
    <result property="empName" column="ENAME"/>
    <result property="job" column="JOB"/>
  </resultMap>

  <!-- Select with no parameters using the result map for Employee class. -->
  <select id="selectAllEmps" resultMap="EmpResult">
    select EMPNO, nvl(MGR,0) mgr, SAL, ENAME, JOB from emp
  </select>

  <!-- A simpler select example without the result map.  Note the
       aliases to match the properties of the target result class. -->
  <select id="selectEmpById" parameterClass="int" resultClass="Employee">
    select
      EMPNO as empNo,
      MGR as manager,
      SAL as sal,
      ENAME as empName,
      JOB as job
    from EMP
    where EMPNO = #id#
  </select>

  <!-- Insert example, using the Employee parameter class -->
  <insert id="insertEmp" parameterClass="Employee">
    insert into EMP (
      EMPNO, MGR, SAL, ENAME, JOB
    )
    values (
      #empNo#, #manager#, #sal#, #empName#, #job#
    )
  </insert>

  <!-- Update example, using the Employee parameter class -->
  <update id="updateEmp" parameterClass="Employee">
    update EMP set
```

```
      MGR = #manager#,
      SAL = #sal#,
      ENAME = #empName#,
      JOB = #job#
    where
      EMPNO = #empNo#
  </update>

  <!-- Delete example, using an integer as the parameter class -->
  <delete id="deleteEmpById" parameterClass="int">
    delete from EMP where EMPNO = #empNo#
  </delete>

</sqlMap>
```

Now that I have mapped SQL to the Employee object, I am going to write a simple Java program to access all of this information.

```
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import com.ibatis.common.resources.Resources;
import java.io.Reader;
import java.io.IOException;
import java.util.List;
import java.util.Iterator;
import java.sql.SQLException;

public class TestEmp {

  private static SqlMapClient sqlMapper;

  static {
    try {
      Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
      sqlMapper = SqlMapClientBuilder.buildSqlMapClient(reader);
      reader.close();
    } catch (IOException e) {
      throw new RuntimeException("Error building the SqlMapClient instance." + e, e);
    }
  }

  public static List selectAllEmps () throws SQLException {
    return sqlMapper.queryForList("selectAllEmps");
  }

  public static Employee selectEmpById  (int id) throws SQLException {
    return (Employee)sqlMapper.queryForObject("selectEmpById", id);
  }

  public static void insertEmp (Employee employee) throws SQLException {
    sqlMapper.insert("insertEmp", employee);
  }

  public static void updateEmp (Employee employee) throws SQLException {
    sqlMapper.update("updateEmp", employee);
  }
```

```
  public static void deleteEmpById (int id) throws SQLException {
    sqlMapper.delete("deleteEmpById", id);
  }

  public static void main (String args[]) {
    Employee e1;

    try {
      System.out.println("Selecting from EMP:");
      Iterator it = selectAllEmps().iterator();
      while (it.hasNext()) {
        Employee emp = (Employee)it.next();
        System.out.println("empno=" + emp.getEmpNo() + " empName=" + emp.getEmpName() + "
job=" + emp.getJob());
      }

      System.out.println("Adding to EMP:");
      Employee emp = new Employee();
      emp.setEmpNo(1000);
      emp.setManager(1000);
      emp.setSal(100);
      emp.setEmpName("Test Emp");
      emp.setJob("CLERK");
      insertEmp(emp);
      System.out.println("EMP added.");

      e1 = selectEmpById(1000);
      System.out.println("empno=" + e1.getEmpNo() + " empName=" + e1.getEmpName() + "
job=" + e1.getJob());
      System.out.println("EMP added.");

      System.out.println("Update EMP:");
      emp.setJob("MANAGER");
      updateEmp(emp);
      e1 = selectEmpById(1000);
      System.out.println("empno=" + e1.getEmpNo() + " empName=" + e1.getEmpName() + "
job=" + e1.getJob());
      System.out.println("EMP updated.");

      System.out.println("Delete EMP:");
      deleteEmpById(1000);
      System.out.println("EMP deleted.");
    }
    catch (SQLException se) {
      se.printStackTrace();
    }
  }
}
```

Run Our First Sample

We are all set to run our first sample program. First we need to compile the JavaBean Employee.java, just do "javac Employee.java". Next we will compile the TestEmp.java, do "javac TestEmp.java". Now we can run it by typing "java TestEmp".

You should see the following output on the screen:

```
Selecting from EMP:
empno=7369 empName=SMITH job=CLERK
empno=7499 empName=ALLEN job=SALESMAN
empno=7521 empName=WARD job=SALESMAN
empno=7566 empName=JONES job=MANAGER
empno=7654 empName=MARTIN job=SALESMAN
empno=7698 empName=BLAKE job=MANAGER
empno=7782 empName=CLARK job=MANAGER
empno=7788 empName=SCOTT job=ANALYST
empno=7839 empName=KING job=PRESIDENT
empno=7844 empName=TURNER job=SALESMAN
empno=7876 empName=ADAMS job=CLERK
empno=7900 empName=JAMES job=CLERK
empno=7902 empName=FORD job=ANALYST
empno=7934 empName=MILLER job=CLERK
Adding to EMP:
EMP added.
empno=1000 empName=Test Emp job=CLERK
EMP added.
Update EMP:
empno=1000 empName=Test Emp job=MANAGER
EMP updated.
Delete EMP:
EMP deleted.
```

That was very easy. We didn't code a single line of JDBC and was able to do all the DB access with a very few lines of code. But what we just did is a Java program directly doing DB access. There is no DAO layer and there is no stored procedure involved.

Our Next iBATIS Program

Let's now do a sample with DAO and stored procedure, the whole nine yards.

First, we will just reuse the Employee.java Java Bean. There is no need to write another one. Now we first design a DAO interface. Using an Interface allows the Java developer to code ahead without waiting for the actual implementation. And now we will code the DAO implementation and write a test case for it.

First let's have a look at the DAO interface.

```
public interface EmpDAO {
    public abstract void add (final Employee employee) throws Exception;
    public abstract Employee get (final Integer uniqueID) throws Exception;
    public abstract void remove (final Integer uniqueID) throws Exception;
}
```

That's it. An interface is just a contract between the codes that defines the APIs. So the Java developer can now code ahead knowing there will be some kind of EmpDAO implementation and that implementation, no matter what we call it, will have the same methods with exactly the same parameters. The Java developer can code methods accepting an EmpDAO and know what methods to call on this EmpDAO object. We can code many implementations of the EmpDAO and pass them in wherever EmpDAO is accepted.

Now before we code our implementation. Let’s do the stored procedures first. Here I have a very simple package that provides procedures to add, remove and return data for the EMP table.

```
create or replace package body emp_util
as

  procedure add (p_empno in emp.empno%TYPE,
                 p_ename in emp.ename%TYPE,
                 p_job in emp.job%TYPE,
                 p_mgr in emp.mgr%TYPE,
                 p_sal in emp.sal%TYPE,
                 p_deptno in emp.deptno%TYPE,
                 status out number)
  is
    ckact                   number;
  begin
    insert into emp (empno, ename, job, mgr, hiredate, sal, deptno)
    values (p_empno, p_ename, p_job, p_mgr, sysdate, p_sal, p_deptno);
    status := SUCCESS;
  exception
    when others then
      dbms_output.put_line(SQLCODE || SQLERRM);
      status := FAILURE;
  end add;

  procedure remove (p_empno in emp.empno%TYPE,
                    status out number)
  is
    record_not_found    exception;
  begin
    delete from emp where empno = p_empno;
    if sql%rowcount = 0 then
      raise record_not_found;
    end if;
    status := SUCCESS;
  exception
    when record_not_found then
      status := FAILURE;
    when others then
      dbms_output.put_line(SQLCODE || SQLERRM);
      status := FAILURE;
  end remove;

  procedure get (status out number,
                 retrc out tRC)
  is
  begin
    open retrc for select EMPNO, nvl(MGR,0) mgr, SAL, ENAME, JOB, DEPTNO from emp;
    status := SUCCESS;
  exception
    when others then
      status := FAILURE;
  end get;

  procedure get (p_empno in emp.empno%TYPE,
                 status out number,
                 retrc out tRC)
  is
  begin
    open retrc for select nvl(MGR,0) mgr, SAL, ENAME, JOB, DEPTNO from emp where empno =
```

```
p_empno;
    status := SUCCESS;
  exception
    when others then
      status := FAILURE;
  end get;

end;
/
```

Now let's create the XML file that maps the stored procedure to the Employee bean.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN" "">

<sqlMap namespace="Employee">

<!--
================================================================================
-->
    <parameterMap id="addMap" class="java.util.Map">
        <parameter property="empNo" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="empName" jdbcType="VARCHAR" javaType="java.lang.String"
mode="IN"/>
        <parameter property="job" jdbcType="VARCHAR" javaType="java.lang.String"
mode="IN"/>
        <parameter property="manager" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="sal" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="deptNo" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="status" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="OUT"/>
    </parameterMap>

    <procedure id="add" parameterMap="addMap" >
        begin emp_util.add(?,?,?,?,?,?,?); end;
    </procedure>

<!--
================================================================================
-->
    <resultMap id="getEmpResultMap" class="Employee">
        <result property="manager" column="MGR" />
        <result property="sal" column="SAL" />
        <result property="empName" column="ENAME" />
        <result property="job" column="JOB" />
        <result property="deptNo" column="DEPTNO" />
    </resultMap>

    <parameterMap id="getEmpMap" class="java.util.HashMap" >
        <parameter property="empNo" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="status" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="OUT"/>
```

```
        <parameter property="emp" jdbcType="ORACLECURSOR" javaType="java.sql.ResultSet"
mode="OUT" resultMap="getEmpResultMap"/>
    </parameterMap>

    <procedure id="get" parameterMap="getEmpMap" >
        begin emp_util.get(?, ?, ?); end;
    </procedure>

<!--
================================================================================
-->
    <parameterMap id="removeMap" class="java.util.Map" >
        <parameter property="empNo" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="IN"/>
        <parameter property="status" jdbcType="INTEGER" javaType="java.lang.Integer"
mode="OUT"/>
    </parameterMap>

    <procedure id="remove" parameterMap="removeMap" >
        begin emp_util.remove(?,?); end;
    </procedure>

</sqlMap>
```

Now we are ready to code the implementation of the EmpDAO. Here we will create a file called EmpDAOImpl.

```
import java.io.Reader;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;

public class EmpDAOImpl implements EmpDAO {

  private static SqlMapClient sqlMapper;

  static {
    try {
      Reader reader = Resources.getResourceAsReader("SqlMapConfig.xml");
      sqlMapper = SqlMapClientBuilder.buildSqlMapClient(reader);
    } catch (IOException e) {
      throw new RuntimeException("Error happened while building the SqlMapClient
instance." + e, e);
    }
  }

    public EmpDAOImpl() {
    }

    public void add (Employee employee) throws Exception {
      try {
        System.out.println("Going to invoke SqlMap: add");
```

```
    HashMap<String, Object> map = new HashMap<String, Object>(7);
    map.put("empNo", employee.getEmpNo());
    map.put("empName", employee.getEmpName());
    map.put("job", employee.getJob());
    map.put("manager", employee.getManager());
    map.put("sal", employee.getSal());
    map.put("deptNo", employee.getDeptNo());
    sqlMapper.update("add", map);
    System.out.println("After invoking SqlMap: add: " + map);

    Integer status = (Integer) map.get("status");
    if (status == null)
      throw new Exception("Received null status value");
    System.out.println("Added unique id: " + employee.getEmpNo());
  }
  catch (SQLException e) {
    throw new Exception("Executing get stored procedure", e);
  }
}

public Employee get (final Integer empNo) throws Exception {
  try {
    System.out.println("Going to invoke SqlMap: get");

    HashMap<String, Object> map = new HashMap<String, Object>();
    map.put("empNo", empNo);
    sqlMapper.update("get", map);
    System.out.println("After invoking SqlMap: get: " + map);

    Integer status = (Integer) map.get("status");
    if (status == null)
      throw new Exception("Received null status value");
    System.out.println("Got unique id: " + empNo);

    //@SuppressWarnings("unchecked")
    ArrayList<Employee> d = (ArrayList<Employee>) map.get("emp");
    if ((d == null) || d.size() > 1)
      throw new Exception("Recieved null user value");

    return d.get(0);
  } catch (SQLException e) {
    throw new Exception("Executing get stored procedure", e);
  }
}

public void remove (final Integer empNo) throws Exception {
  try {
    HashMap<String, Object> map = new HashMap<String, Object>();
    map.put("empNo", empNo);
    sqlMapper.update("remove", map);
    System.out.println("After invoking SqlMap: remove: " + map);

    Integer status = (Integer) map.get("status");
    if (status == null)
      throw new Exception("Received null status value");
    System.out.println("Removed emp no : " + empNo);
  } catch (SQLException e) {
```

```
        throw new Exception("Executing remove stored procedure", e);
      }
    }
}
```

So we have an implementation called EmpDAOImpl that implements the EmpDAO. Again you don't see a single line of JDBC or anything looks like SQL. All you have are iBATIS calls that refer to the names in the Employee.xml file.

Now we just need a test program to access data through the DAO we just implemented.

```
public final class EmpDAOTest {
  private static Employee emp, emp2;

  public static void main (String args[]) {
    EmpDAO d = new EmpDAOImpl();

    emp = new Employee();
    emp.setEmpNo(1000);
    emp.setManager(0);
    emp.setSal(1234);
    emp.setDeptNo(20);
    emp.setEmpName("Alvin");
    emp.setJob("MANAGER");
    System.out.println("test add :");
    try {
      d.add(emp);
    }
    catch (Exception e) { e.printStackTrace(); }
    System.out.println("test add done.");

    System.out.println("test get :");
    try {
      emp2 = d.get(1000);
    }
    catch (Exception e) { e.printStackTrace(); }
    System.out.println(emp2.getEmpNo() + " " + emp2.getEmpName() + " " + emp2.getJob());
    System.out.println("test get done.");

    System.out.println("test remove :");
    try {
      d.remove(1000);
    }
    catch (Exception e) { e.printStackTrace(); }
    System.out.println("test remove done.");
  }
}
```

To run it, first compile the DAO interface, "javac EmpDAO.java", next compile the implementation class, "javac EmpDAOImpl.java". Finally compile the test program to access data through the DAO, "javac EmpDAOTest.java".

To run it, just do "java EmpDAOTest".

Debugging in iBATIS

iBATIS some times hides the exceptions and it is not obvious as to why things didn't work as intended. This means you could miss seeing a SQLException that could of told you the exact ORA- error that happened.

Here is a Debug.xml I created to help produce an Oracle SQL trace from iBATIS. Then you can simply exam the trace file to see what ORA- error was happening.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN" "">

<sqlMap namespace="Debug">

<!--
================================================================================
-->
<!-- Oracle debug statements
-->
<!--
================================================================================
-->

    <update id="ev10046on">
        alter session set events '10046 trace name context forever, level 12'
    </update>

    <update id="ev10046off">
        alter session set events '10046 trace name context off'
    </update>

</sqlMap>
```

Simply include this in the SqlMapConfig.xml file. And now you can do something like this in Java code to start and stop trace.

```
sqlMapper.update("ev10046on");
….
sqlMapper.update("ev10046off");
```

Pros and Cons

Just as the goal for iBATIS stated, it tries to do 80% of the JDBC coding with only 20% of the code.
The pros for iBATIS comes from that statement, it lies in its simplicity. You don't have to know JDBC to do database access and does it with less coding. The cons comes from that same statement, iBATIS doesn't do 100% of what you need. So there is a time you will have to do some JDBC coding yourself. For example, iBATIS doesn't support LOB access. So you would have to code that in JDBC.

But with the DAO layer, this is completely hidden from the application developer. As long as nothing in the DAO interface changes.

Conclusion

iBATIS is a wonderful tool, but just like all tools we use, it has its place. When used correctly it can save productivity and make things easier for both the application developers and Oracle developers.

The most important thing is planning. If you can have the DAO interface well defined and clearly defined, then it's easier for both the application developers and database developers.

I shot an arrow into the air
It fell to earth, I knew not where.

You can't keep going through life not knowing!

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

BEZ
a business lens on IT

The Prediction People™

www.bez.com

Monitoring and Diagnosing Production Applications Using Oracle Application Diagnostics for Java

Rajagopal Marripalli, Oracle Corporation

Production Application Diagnostics Challenges

Critical Java applications suffer from availability and performance problems often when business is at its peak. IT administrators spend lot of time diagnosing the root cause of these problems. They normally do not have enough Application knowledge and must depend on application architects, and developers, to diagnose the problem. All the involved teams cannot use the diagnostic tools in production due to the high overhead they impose, and they must try to reproduce the problems in test or development environment. Many times, the problems are either not reproducible or it takes too long reproduce the problem. The impact on the business is severe, especially when the business is at its peak.

Many Java diagnostics tools in the market are based on Byte Code instrumentation (BCI) technique that requires application instrumentation. It requires in-depth understanding of the application to know which Java components to instrument. In addition to the application knowledge, it also requires specialized skills to use the tools to instrument the application. BCI based tools have high configuration overhead and also add significant performance overhead. They also do not provide visibility for transactions across the Java and DB tiers. All these reasons make the BCI based tools unsuitable for diagnosing application problems in production environment.

Monitoring tools that only alert administrators when the application problems occur are not just sufficient. Administrators need to have enough details to be able to identify the root cause of the problems in production environment without impacting the businesses. Administrators need a lightweight production monitoring tools that can give enough details when the problems occur.

Oracle Application Diagnostics for Java (Java Diagnostics)

Oracle Enterprise Manager 10g Application Diagnostics for Java (Oracle's Java Diagnostics) is a lightweight Java application monitoring and diagnostics solution that enables administrators to diagnose performance problems in production. It reduces the time to resolve performance problems drastically and improves the application availability and performance.

Using Oracle's Java diagnostics administrators will be able identify the root cause of the performance problems in production environment without having to reproduce them in test or development environment. It does not require complex instrumentation of the Application or restarting the application to get in-depth application details. Application administrators will be able to identify deep down Java problems or Database issues that are causing application downtime without any application knowledge.

Low Overhead Java Activity Monitoring and Diagnostics

Oracle's Java diagnostics provide in-depth Java activity monitoring with very low overhead without slowing down the application.

Oracle's Java Diagnostics provides Java application resource consumption such as the requests waiting on I/O, network, and locks, requests that are burning lot of CPU cycles and the requests waiting on database calls immediately after the deployment. Along with the bottleneck resources it also provides the end-user requests that are impacted by the bottleneck resources and the application components that are causing the performance bottlenecks.

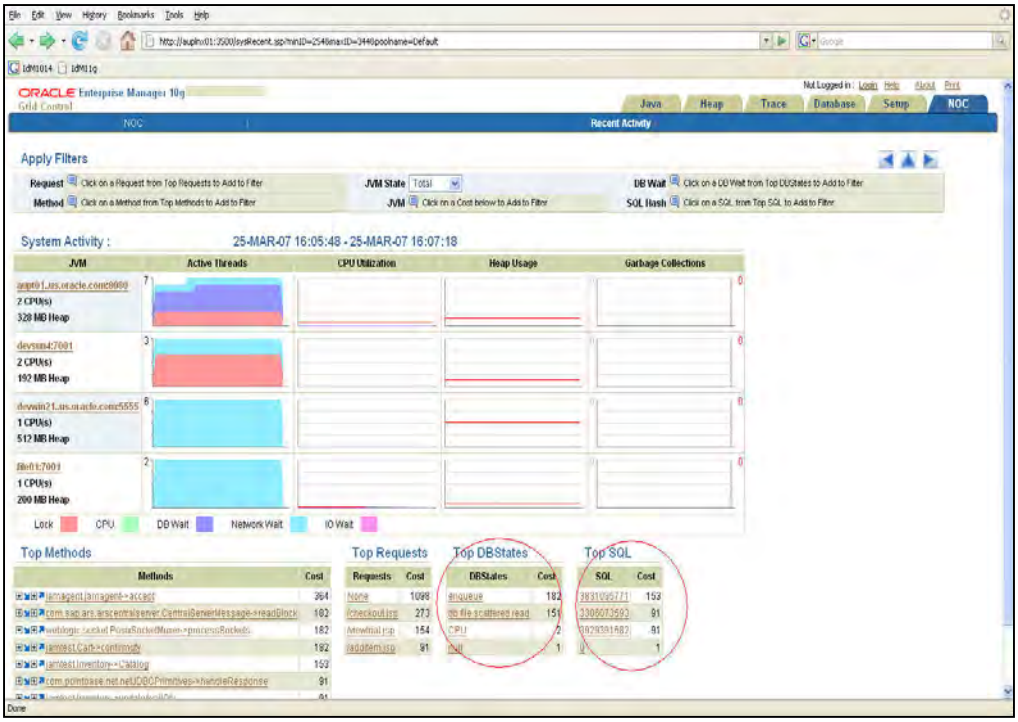
Real-time Transaction Tracing

If a particular request is slow or hanging or if the entire application is slow, administrators can run the real-time transaction trace to view current Java application activity. Administrators can look at the offending threads and their execution stack. They can perform analysis such as how much time a thread spent in waiting for DB wait or wait on a lock. Complex problems such as activity in one thread (or request) affecting the activity in the other thread or rest of the JVM can be found very quickly.

Cross-tier Correlation with Database

If a performance problem is between Java and Database, it requires efforts from multiple teams. It is not easy to narrow down the problem to a particular tier and component within the tier. Resulting in high turn around time to problem resolution.

Cross-tier correlation of individual Java requests with associated database activity shows the database states such as database locks or full table scans and the SQL statements that are causing the Java application problems.



Oracle's Java Diagnostics for DBAs

Database Administrators are often challenged with the issues that are caused by the applications running in the middleware. Database administrators who are solely responsible for Database performance can start diagnosing the problems from Database point of view. They can view the database sessions that are suffering from performance problems and identify the Java threads that are using the database sessions. They pinpoint to the application request and the Java method that is impacted by the Database issue.

Memory Leak Detection and Analysis

Memory leaks are often one of the reasons to slowdown application and lead to application crash. These problems are difficult reproduce in test environments and almost impossible to diagnose in production environments without impacting the application availability.

Oracle's Java diagnostics alert administrators on abnormalities in Java memory consumption. Administrators can use Oracle's Java diagnostics console and take heap dumps in production applications without disturbing the application. They can take multiple heap dumps over a period of time, analyze the differences between the heap dumps and identify the object causing the memory leak.

Heterogeneous Platforms

Oracle's Java Diagnostics supports Oracle, Non-Oracle Application Servers and JVMs from multiple vendors. It also supports standalone Java applications.

Conclusion

Oracle Application Diagnostics for Java is a lightweight production Java application monitoring and diagnostics solution. It reduces the time to resolve performance problems and improves the availability and performance of critical business applications with the help of the following features:

- In-depth visibility of Java activity
- Real-time transaction tracing
- Cross-tier correlation with Database
- Memory leak detection and analysis

Dynamic SQL in a Dynamic World

Michael Rosenblum, Dulcian, Inc.

Introduction

I have presented several times on the topic of dynamic SQL. I never expected that:

- this feature would be completely unfamiliar to so many people
- there were so many ways of misusing the feature.

Due to length limitations, this paper will not include syntax discussions (it takes about 30 pages in my book PL/SQL for Dummies to introduce all of the core concepts). This paper will include some real world cases and examples. The goal is to explain what is really happening when you fire an EXECUTE IMMEDIATE command.

1. Security issues

A currently hot topic in the Oracle world is that of database security. From the point of view of dynamic SQL, there are two questions to address:

1. What privileges do you need to use dynamic SQL?
2. How can you guard against misuse of dynamic SQL?

A. Running Dynamic SQL

The following code shows how to create a user with DBA privileges and a few basic routines:

```
create user nyoug identified by nyoug
default tablespace users
temporary tablespace temp;

grant dba to nyoug;

create or replace procedure nyoug.p_makeTable (i_name_tx varchar2) is
   v_sql_tx varchar2(256):= 'create table ' ||i_name_tx||' (a_tx varchar2(256))';
begin
  dbms_output.put_line('Fired:'||v_sql_tx);
  execute immediate v_sql_tx;
end;

create or replace function nyoug.f_getCount_nr (i_user_tx varchar2, i_table_tx varchar2) return number
is
   v_out_nr number;
   v_sql_tx varchar2(256):='select count(*) from ' ||i_user_tx||'.'||i_table_tx;
begin
  dbms_output.put_line('Fired:'||v_sql_tx);
  execute immediate v_sql_tx into v_out_nr;
  return v_out_nr;
end;
```

Next, you can connect as NYOUG and try to create a table using dynamic SQL and get a row count from SCOTT.EMP:

```
SQL> exec p_makeTable('t_nyoug');
Fired:create table t_nyoug (a_tx varchar2(256))
BEGIN p_makeTable('t_nyoug'); END;

*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "NYOUG.P_MAKETABLE", line 5
ORA-06512: at line 1

SQL> select f_getCount_nr('SCOTT','EMP') from dual;
Fired:select count(*) from SCOTT.EMP
select f_getCount_nr('SCOTT','EMP') from dual
       *
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "NYOUG.F_GETCOUNT_NR", line 7
SQL>
```

Strangely enough, both calls failed with errors that suggest some kind of privilege problems. But the user has a DBA role. What can the developer do? Take some code like the following and fire it manually (expecting some kind of Oracle error):

```
SQL> create table t_nyoug (a_tx varchar2(256));
Table created.

SQL> select count(*) from SCOTT.EMP;
  COUNT(*)
----------
        14
SQL>
```

Surprise! A direct call works, while the same code wrapped in dynamic SQL fails. When this happens, most developers and DBAs will declare dynamic SQL to be unusable and spread this message to everyone else in their companies. This is unfortunate since it would only take one extra step to make everything work, namely referring to the Oracle manuals. The Oracle security model requires all privileges (both system and object) for the code implemented using dynamic SQL to be granted explicitly. This cannot be done with roles. To test this, connect as any valid administrative user and add the following grants to NYOUG user:

```
SQL> grant create table to nyoug;
Grant succeeded.
SQL> grant select on scott.emp to nyoug;
Grant succeeded.
SQL>
```

Now rerunning the initial procedure and function provides completely different results. Everything starts to work exactly as expected:

```
SQL> exec p_makeTable('t_nyoug01');
Fired:create table t_nyoug01 (a_tx varchar2(256))
PL/SQL procedure successfully completed.

SQL> select f_getCount_nr('SCOTT','EMP') from dual;
Fired:select count(*) from SCOTT.EMP
```

```
F_GETCOUNT_NR('SCOTT','EMP')
----------------------------
                          14
SQL>
```

To summarize, if you are planning to use dynamic SQL in production code, first check with your DBA team about how much effort it would take to provide explicit privileges. Otherwise there may be a lot of confusion during various stages of your application development project.

B. Running YOUR Dynamic SQL

The question of code injections has been extensively discussed throughout the Oracle literature. Any dynamic SQL operation requires two things to be passed: structural elements (tables, columns, SQL clauses) and/or the data itself. This means that security should be interpreted in the following ways:

- You can only pass allowed structural elements.
- You cannot pass structural elements instead of data.

To satisfy these requirements, follow just two rules to keep your system safe:

1. If the end user's input is simply data (e.g. values of columns, etc.), these values must be passed to the dynamic SQL using bind variables. Since these variables cannot affect the structure of the query, whatever users type will not cause any problems.
2. If the end user's input will influence the structure of the code, the available options should be based only on the repository elements without the ability to alter the repository itself. That is the sole prerogative of administrators or power users. End users should only select from a previously populated list of functions.

The first rule is significantly easier to illustrate. The way in which Oracle defines bind variables automatically prevents their misuse. Even when passing something like “null OR 1=1” instead of a last name, Oracle would just look for the last name “null OR 1=1” instead of showing the whole table, because the input will be interpreted as a string, and not a part of the query as shown here:

```
SQL> declare
  2    v_tx varchar2(256):='null OR 1=1';
  3    v_count_nr number:=0;
  4  begin
  5    execute immediate 'select count(*) from emp where ename = :1'
  6    into v_count_nr using v_tx ;
  7    dbms_output.put_line('Bind: '||v_count_nr);
  8
  9    execute immediate 'select count(*) from emp where ename = '||
 10    v_tx into v_count_nr;
 11    dbms_output.put_line('Inject: '||v_count_nr);
 12  end;
 13  /
Bind: 0
Inject: 14
PL/SQL procedure successfully completed.
SQL>
```

The repository concept requires more explanation. A number of SQL experts were asked to help implement a very advanced business rules-based system. The problem was not the rules engine itself (even though it was very complicated and included hundreds of millions of records), but the whole deployment cycle. The smallest change to the front-end required a significant effort from multiple resources. Another point of concern was that end users were not exactly sure

what modules were needed in addition to the rules engine. The requirement was something close to one extra small screen every few hours.

In four man-days, a somewhat limited but generic module was built to satisfy the following requirements:
User requests are represented (wrapped if needed) as functions with:

A generic name (header of the pop-up screen)
Up to 5 parameters, each including:

i. Header
ii. Mandatory/not mandatory identification
iii. Data type (NUMBER/DATE/TEXT/LOV)
iv. Optional conversion expression (e.g. default date format in the UI since everything on the web is text-based)
v. Value list name (for LOV datatypes)

Return CLOB

The response is already pre-formatted as HTML so the front-end just needs to present the output on the screen.
All definitions are stored in the repository table accessible only by administrators and not visible to the end-users

The final order of actions was as follows:

1. The end user sees the list of possible options (display names from the repository tables) and selects one.
2. The front-end utility reads the repository and builds a pop-up screen on-the-fly with appropriate input fields and mandatory indicators. If the data type of the input field is LOV, the utility requests the generic LOV mechanism from the appropriate list of ID/display pairs.
3. The user enters whatever is needed and presses "SUBMIT." The front-end calls a special wrapper procedure.
4. The wrapper procedure builds a real function call, passes the required parameters, and returns whatever result is generated.
5. The front-end displays the result.

Now everybody was happy since it only took about five minutes from the moment any new procedure was ready to the moment when it was accessible from the front-end.

From a code injection point of view, everything was completely safe as well. Although the core function uses dynamic SQL, all of the structural elements are declared in the repository table. This means that one side of security protection comes down to a standard audit of repository activities (insert/update/delete). From the other side, end users can only communicate to the system something like "execute routine #N" (where N is selected from the value list), but they have no control over the transformation of #N to SCOTT.DO_SOMETHING. This separation of roles between the person who defines what should be executed and the person who actually uses the functionality is the most important aspect of using a repository-based approach. It allows the system to be flexible enough without creating a security breach.

2. Object Dependencies

Another problem area, especially for beginners, is object dependency. Since dynamic SQL is executed at runtime, there are some pros and cons associated with its use.

A. Con #1: No dependency path to follow up

Oracle does not resolve strings for you, so the USER_DEPENDENCIES data dictionary view is useless. As of this writing, the only effective solution is to use the following two-step process:
Populate repositories with required information, structured similarly to that of the Oracle data dictionary
Generate from the repository using a straightforward, transparent mechanism (so it is clear what becomes what)

Now the whole dependency problem is limited to comparing the Oracle data dictionary with your own. This part is easy. Also you are getting a direct syntax check of the generated code. Since there is one and only one way of transforming a repository into code, everything either works or doesn't.

B. Con #2: No way to determine exactly what will be executed

Since a lot of parameters are either user-entered or constructed on the fly, it is very difficult to predict possible syntax errors without really encountering them. This problem is much more difficult to solve, but the concept of "samplers" looks somewhat promising (if a repository-based approach is not for you):
Generate all possible (or as many as possible) permutations of the code to be executed and create PL/SQL modules with that code. This approach can help to identify any initial code problems
Record all dependencies and keep a simple module that references the same set of objects.
If the sampler becomes invalid, this is an indication of code problems.

C. Pro #1: Reference objects that may not be in the database (yet)

If you have a table for each month, it is easier to create a universal caller instead of writing a huge case statement. An example is shown here:

```
create or replace procedure p_runMonthly(i_dt date)
is
    v_sql_tx varchar2(256):='begin p_run_'||to_char(i_dt,'YYYYMM')||'; end;';
begin
    execute immediate v_sql_tx;
end;
```

D. Pro #2: "Back door" to resolve logical dead loops or hide existing dependencies

Sometimes having dependencies is not a good thing for the following reasons:

- Code generators – If you wrap a call to the generated modules into dynamic SQL, you can refresh these modules without invalidating all dependencies. This can be critical when using JDBC connections to the database, since they are significantly less forgiving (in comparison to Net8 calls).
- Remote objects – Since your maintenance cycle may not be the same as that of other teams involved in your project, there is a chance that you could reference objects via database links while they are un-accessible or invalid. In that case the only safe way out is to wrap all remote calls into dynamic SQL
- Logical loops – Even though it is very difficult, you could create a case when a chain of object references cycles itself. In that case, the only solution is to make one step a dynamic SQL module. An example from a sales/contacts system is shown here:
  - Original design
    - There is a materialized view MV_PEOPLE based on the package function VRAP$PKG.F_PEOPLE_TT that returns an object collection.
    - There is a materialized view MV_PEOPLE_ASSIGN that is based on the package function VRAP$PKG.F_PEOPLEASSIGN_TT that returns an object collection. Inside of this function, a materialized view MV_PEOPLE is used to get the current phone numbers of all assigned representatives.
    - There is a module called nightly (via a job) that fires DBMS_MVIEW.REFRESH for both materialized views in an order.
  - Problem
    - The job fails with "ORA-04068: Existing state of packages has been discarded," because a refresh of the first materialized view is explicitly used in the core function of the second one.
  - Solution
    - In VRAP$PKG.F_PEOPLEASSIGN_TT convert all SQL queries to MV_PEOPLE to dynamic SQL

3. Bulk Operations

Dynamic SQL can be extremely handy in conjunction with the major performance booster of PL/SQL code and bulk operations on collections. The FORALL operator allows you to fire a set of dynamic DMLs at once, but only if you use a parameter in the USING clause. Unfortunately, this does not work with structural elements on the fly as shown here:

```
forall i in dept.first..dept.last
 execute immediate 'delete from emp where deptno=:1' using dept(i); -- works
 execute immediate 'drop table t_'||dept(i); -- doesn't work
```

But the major advantage of this approach is the following statement type:

```
EXECUTE IMMEDAITE <string> BULK COLLECT INTO <collection>
```

Even though dynamic SQL does not support any PL/SQL datatypes, you can always use RECORD-types as the output of a dynamic query. This feature allows you build a query of any complexity on the fly and retrieve its results for future processing in the fastest possible way.

All user-defined SQL datatypes are perfectly valid for all cases, but there is one syntax issue that a lot of people forget. There should be always an object constructor INSIDE of the query as shown here:

```
Create function f_getlov_nt
      (i_table_tx varchar2,i_id_tx varchar2,i_display_tx varchar2,i_order_tx varchar2)
return lov_nt is
    v_out_nt lov_nt := lov_nt();
begin
    execute immediate
      'select lov_oty('
                  ||i_id_tx||','||i_display_tx||')'||
      ' from '||i_table_tx||
      ' order by '||i_order_tx
    bulk collect into v_out_nt;
    return v_out_nt;
end;
```

The explanation is simple. Since you are loading results into a single variable, you should return a collection of homogeneous objects and not a set of columns.

4. Transformation of Cursors for DBMS_SQL

Until Oracle version 11g , when discussing dynamic SQL, there was a very thick line between Native Dynamic SQL and DBMS_SQL package. In this latest version of the DBMS, Oracle declared "functional completeness" and complete cross-compatibility of these implementations with the following results:

- Both support more than 32K of text.
- Both can work with user-defined datatypes etc.

But there was a new extension to the DBMS_SQL package that was expected for many years. Now there is a way to transform the REF CURSOR datatype into the DBMS_SQL.OPEN_CURSOR datatype and vice versa. Last year, I demonstrated a routine that would take a random SQL statement passed as a text and return a CLOB with all retrieved rows. The only problem was that technically the same query had to be parsed twice, once to describe columns via DBMS_SQL and the second time to get data back. In 11g the same functionality could be achieved with only one parse as shown here:

```
create or replace function f_getSqlResults (i_sql varchar2,i_fetch_limit_nr
```

```
number:=64000)
  return clob
is
  v_out_cl   CLOB;
  v_header varchar2(4000);
  v_column varchar2(32000);

  v_sql_tx varchar2(32000);
  v_cur     integer := dbms_sql.open_cursor;
  v_cols_nr number := 0;
  v_cols_tt dbms_sql.desc_tab;
  v_exec_nr integer;
  v_ref_cur SYS_REFCURSOR;

  pragma autonomous_transaction;
BEGIN
  DBMS_SQL.parse (v_cur, i_sql, DBMS_SQL.native);
  DBMS_SQL.describe_columns (v_cur, v_cols_nr, v_cols_tt);
  v_exec_nr:=dbms_sql.execute(v_cur);
  v_ref_cur:=dbms_sql.to_refcursor(v_cur);

  FOR i IN 1 .. v_cols_nr LOOP
    if i = 1 then
      v_column:=' v_row_tx := ''"''||DynResults(i).'||v_cols_tt(i).col_name||'||''"'' ';
      v_header := '"'||NVL (v_cols_tt (i).col_name, ' ')||'"';
    else
      v_column:=v_column||'||chr(9)||''"''||DynResults(i).'||
           v_cols_tt (i).col_name||'||''"'' ';
      v_header := v_header||chr(9)||'"'||NVL (v_cols_tt(i).col_name, ' ')||'"';
    end if;
  END LOOP;

  v_header := v_header||chr(10);
  v_column:=v_column||'||chr(10);'||chr(10);

  v_sql_tx :=
    'DECLARE '||
    '   v_out_cl       CLOB; '||
    '   v_buffer_tx      VARCHAR2(32000); '||
    '   v_buffer_length_nr number:=0; '||
    '   v_row_tx   VARCHAR2(32000); '||
    '   cursor c1 is '||
          i_sql||';'||
    '   TYPE DynamicTable IS TABLE OF c1%rowtype INDEX BY BINARY_INTEGER; '||
    '   DynResults   DynamicTable; '||
    '   PROCEDURE p_addline (pi_tx VARCHAR2) IS '||
    '      v_add_nr number:=NVL (LENGTH (pi_tx), 0);'||
    '   BEGIN '||
    '     if v_buffer_length_nr+v_add_nr > 32000 then '||
    '       DBMS_LOB.writeappend (v_out_cl, v_buffer_length_nr,  v_buffer_tx);'||
    '       v_buffer_length_nr:=v_add_nr; '||
    '       v_buffer_tx:=pi_tx; '||
    '     else '||
    '       v_buffer_length_nr:=v_buffer_length_nr+v_add_nr; '||
    '       v_buffer_tx:=v_buffer_tx||pi_tx; '||
    '     end if; '||
    '   END; '||
```

```
    'BEGIN '||
    '  DBMS_LOB.createtemporary (v_out_cl, TRUE, DBMS_LOB.CALL);'||
    '  p_addline(:1); '||
    '  fetch :2 bulk collect into DynResults limit :3;'||
    '  FOR i IN 1 .. DynResults.COUNT LOOP '||
          v_column ||
    '      p_addline(v_row_tx); '||
    '   END LOOP;'||
    '   dbms_output.put_line(v_buffer_length_nr);'||
    '   DBMS_LOB.writeappend (v_out_cl, v_buffer_length_nr,  v_buffer_tx);'||
    '   :4 := v_out_cl;'||
    'END;';

    EXECUTE IMMEDIATE v_sql_rx
    USING    IN  v_header, in v_ref_cur, IN  i_fetch_limit_nr, OUT  v_out_cl;
    close v_ref_cur;

    return v_out_cl;
EXCEPTION
   WHEN OTHERS
   THEN
      close v_ref_cur;
      return '"<Invalid Query>"';
END;
```

As you see from the example, DBMS_SQL.OPEN_CURSOR is transformed into REF CURSOR, which is a valid datatype to be passed into an anonymous dynamic SQL block. To make the transformation, the original cursor has to be "executed." Simply parsing it is not enough.

This feature can be extremely useful for anyone working with this type of undefined data structures or in cases of heavy usage of REF CURSOR datatypes because now it is possible to get detailed description of the cursor without knowing where and how it was created as shown here:

```
create or replace procedure p_explainCursor (io_ref_cur IN OUT SYS_REFCURSOR)
is
    v_cur     integer := dbms_sql.open_cursor;
    v_cols_nr number := 0;
    v_cols_tt dbms_sql.desc_tab;
begin
    v_cur:=dbms_sql.to_cursor_number(io_ref_cur);
    DBMS_SQL.describe_columns (v_cur, v_cols_nr, v_cols_tt);
    for i in 1 .. v_cols_nr loop
        dbms_output.put_line(v_cols_tt (i).col_name);
    end loop;
    io_ref_cur:=dbms_sql.to_refcursor(v_cur);
end;
```

This simple example illustrates the point:

```
SQL> declare
  2    v_tx varchar2(256):='select * from dept';
  3    v_cur SYS_REFCURSOR;
  4  begin
  5    open v_cur for v_tx;
  6    p_explainCursor(v_cur);
  7    close v_cur;
  8  end;
```

```
  9  /
DEPTNO
DNAME
LOC
PL/SQL procedure successfully completed.
SQL>
```

5. Performance and Resource Utilization

Another major point why a lot of people refuse to try dynamic SQL is that you cannot avoid parsing. It is important to distinguish three completely different situations:

1. EXECUTE IMMEDIATE without bind variables – N hard parses for N calls
2. EXECUTE IMMEDIATE with bind variables – 1 hard parse + N-1 soft parse for N calls
3. DBMS_SQL with separation of parsing and execution stage – 1 hard parse only for every type of calls.

The last case requires a bit non-trivial code. In some cases, this can be a real performance booster for the whole system as shown in the following example:

- The problem:
  - Users upload CSV-files
  - All files use very specific templates
    - Name of file defines type
    - Column headers map directly to table columns (if header is not registered, it will be ignored)
    - 1 row of file = 1 logical group (1..N real rows) – for example, one row could include both sale and cancellation records
    - Group-level validation – rules are applicable to the whole set of rows rather than each insert
- The solution:
  - Universal CSV-loader - build all inserts on the fly

The original implementation of the solution was straightforward, but with real volumes of data (10k+ rows in a single file) DBAs started to complain that the CPU workload became too high. Also, the overall performance of the module was significantly worse compared to estimates (and worsened with increased file sizes). The answer was to implement that single parsing mechanism. Since the module itself is too large, this example only shows the core snippets.

The first step was to prepare the inserts. Each insert type has its own DBMS_SQL cursor (already parsed), stored in the associative array for future access as shown here:

```
Declare
    type integer_tt is table of integer;
    v_cur_tt integer_tt;
    ...
Begin
    ...
    for r in v_groupRow_tt.first..v_groupRow_tt.last loop
        v_cur_tt(r):=DBMS_SQL.OPEN_CURSOR;
        for c in c_cols(v_mapRows_tt(r)) loop
            for i in v_header_tt.first..v_header_tt.last loop
                if v_header_tt(i).text=c.name_tx then
                    v_col_tt(i):=c;
                    v_col_tx:=v_col_tx||','||v_col_tt(i).viewcol_tx;
                    v_val_tx:=v_val_tx||',:'||v_col_tt(i).viewcol_tx;
                end if;
            end loop;
```

```
        end loop;
        v_sql_tx:='insert into '||v_map_rec.view_tx||
                    '('||v_col_tx||') values('||v_value_tx||')';
        DBMS_SQL.PARSE(v_cur_tt(r),v_sql_tx,DBMS_SQL.NATIVE);
    end loop;
```

Now it is only necessary to spin through all of the rows of the uploaded file, find the appropriate row type (from the array) and bind variables, and fire the insert statement as shown here:

```
    for i in 2..v_row_tt.count loop
        for r in v_groupRow_tt.first..v_groupRow_tt.last loop
            for c in v_col_tt.first..v_col_tt.last loop
                if v_col_tt(c).id  = v_mapRows_tt(r) then
                    DBMS_SQL.BIND_VARIABLE(v_cur_tt(r),
                        ':'||v_col_tt(c).viewcol_tx,
                        v_data_tt(c).text);
                end if;
            end loop;
            v_nr:=dbms_sql.execute(v_cur_tt(r));
        end loop;
    end loop;
```

The overall performance improvement for a set of 60,000 rows in the file was about 50 times. For smaller datasets, that number decreased, but it was always significant.

Conclusions

The whole topic of dynamic SQL is just too huge to cover in one paper. But the main purpose of this paper is to encourage IT professionals to expand the number of tools available to solve different kinds of problems. Having some kind of tool that allows us to support unforeseen changes and unpredicted requirements should be always welcomed. However, you should consider complex solutions only when simpler solutions don't work. Even with the best possible technology available, there is no substitute for good analysis and a real understanding of what are you trying to develop.

About the Author

Michael Rosenblum is a Development DBA at Dulcian, Inc. He is responsible for system tuning and application architecture. He supports Dulcian developers by writing complex PL/SQL routines and researching new features. Mr. Rosenblum is the co-author of PL/SQL for Dummies (Wiley Press, 2006). Michael is a frequent presenter at various regional and national Oracle user group conferences. In his native Ukraine, he received the scholarship of the President of Ukraine, a Masters Degree in Information Systems, and a Diploma with Honors from the Kiev National University of Economics.

www.pillardata.com

Pillar Axiom.
Application-Aware Storage for Oracle Applications.

Pillar Axiom® can more than double your efficiency in running Oracle applications, because it sees storage from an applications point of view.

Pillar's integrated support for Oracle Enterprise Manager provides an ideal platform for monitoring your application SLAs. Since Pillar is Application-Aware, business policies are easily enforced and automated for Oracle E-Business Suite, PeopleSoft, Siebel, JD Edwards, and Retek.

- Oracle E-Business Suite
- PeopleSoft
- Siebel
- JD Edwards
- Retek

Pillar's support for Oracle Validated Configurations assures a faster, easier, lower-cost platform for all Oracle 11g Database and BI/DW customers, too. And Pillar and Oracle provide joint accelerator solutions for customers looking to expedite database and applications upgrades.

Find out why Oracle chose Pillar to eliminate cost and drive the highest level of efficiencies. And how we can do the same for you, with Application-Aware storage for Oracle applications. They're made for each other, so you can get the best out of both of them.

Call 1.877.252.3706 or visit www.pillardata.com

The First and Only True Application-Aware Storage™

pillar® DATA SYSTEMS

IBM Information Management Software E-Discovery and Enterprise Data Management — What You Need to Know

IBM Corporation

Brave New World of e-discovery

New products, new technologies, new legislation — and new liabilities! With the advent of stronger laws that focus on data retention and accountability, such as Sarbanes-Oxley, SEC-17a, Basel II and others, companies across industries face a much higher risk for audits and potential lawsuits. Along with increased risk, the associated costs can multiply into millions of dollars.

A key factor in satisfying audit requirements and defending lawsuits, or at a minimum, avoiding costly penalties, is having the capability to retrieve the appropriate information and supporting evidence to meet audit, legal and business requirements. However, with the rise of the "new data center" and "paperless" application environments, almost 95 percent of all business information is created and stored electronically, hence the evolution of electronic discovery ("e-discovery").

According to Forrester Research, e-discovery is the process of collecting, preparing, reviewing, formatting and producing enterprise content that may exist in either paper or digital form in response to internal, litigation, or regulator requests.1 Examples include files, databases, e-mail and any other electronically stored information (ESI) that could be relevant evidence in a lawsuit.

While companies recognize that managing business-critical information electronically offers many advantages, the practice also poses many IT challenges. IT organizations need technology with proven capabilities for managing and locating the specific information in order to deliver a fast response to any e-discovery or audit request. Without an effective enterprise data management strategy and early IT involvement to plan for data retention and audit challenges, the way your company responds to an e-discovery request can mean the difference between success and failure.

Litigation Costs Companies Millions

The third annual litigation trends survey conducted by the international law firm, Fulbright & Jaworski L. L. P., provides some striking statistics. Based on information obtained from 422 in-house law departments worldwide, the survey found that "U.S. companies face an average of 305 pending lawsuits internationally. For large U.S. companies – those with $1 billion or more in annual gross revenue – the number of lawsuits soared to 566 cases, with an average of 50 new disputes emerging each year for close to half of them."2

No industry is immune to the potential risk of litigation. The survey explored thirteen "different industry sectors, including energy, manufacturing, financial services, retail/wholesale, technology/communications, engineering/construction, healthcare and pharmaceutical, real estate, insurance and education, as well as non-profit organizations and trade associations."3 However, the insurance industry, with an average of 1,696 lawsuits to its credit, is the undisputed leader, followed by retailers and energy firms that reported more than 330 caseloads per company.

Another significant statistic is cost. The survey reported that "large U.S. companies commit an average of $19.8 million to litigation," while "counsel at American small businesses say their average dispute spending totaled only $178,000." In

addition, “although the majority of those cases are in U.S. courts, the tide of international disputes is rising – more than one-third of companies said that up to 20 percent of their dockets originate in foreign venues, proof that the U.S. style of litigation is going global.”4

Business Reasons to Prepare for e-discovery

Clearly, the potential for litigation is on the rise, not only in the U.S., but also globally. The associated legal costs and penalties can be prohibitive. In addition, because most cases settle before going to trial, the discovery phase is often the most critical and expensive part of litigation.

For example, in the event of an e-discovery request, it takes time and resources to pull the information together. First, you must identify the appropriate data, which may be managed in different application databases operating on different platforms across your enterprise. Hopefully this data has been maintained in a way that makes it easy to access. Secondly, you must ensure that the integrity of the data and metadata is preserved so that it can be retrieved, accurately and unaltered in its business context.

Next, reviewing the data ensures that you can separate the information that must be produced from the data that is privileged or irrelevant and must be withheld. Finally, you must produce the data in a transferrable medium. Based on the cost per hour or per GB of data, it is easy to calculate the expense of the discovery phase into the hundreds of thousands of dollars.

The following metrics from a recent Gartner Research Report (see Table 1) indicate how quickly costs can multiply:5

Table 1. Cost metrics for e-discovery Task Time

- Identify appropriate data $200 / hour
- Preserve the data $100 to $300 / hour
- Collect the data $200 to $300 / hour
- Review the data $120 to $350 / hour
- Produce the data $1000 to $2,100 / GB

However, even if your organization has never faced an audit or e-discovery request, it simply makes good business sense to be prepared. Many companies start by reviewing their IT infrastructure systems and applications to identify possible areas of contention. In addition to e-mail, also consider the databases that support your custom and packaged business applications, file servers, network access controls, desktop workstations, laptops, blackberries and other mobile devices across your enterprise.

Legal Reasons to Prepare for e-discovery

There are several legal reasons to be prepared for e-discovery. First and foremost are the Federal Rules of Civil Procedure (FRCP), which went into effect on December 1, 2006. The new rules govern lawsuits filed in Federal Court and outline the rights and obligations of all organizations, public or private.

Basically, these rules impact your company’s investment in records retention and information management initiatives. Some of the amendments that are worth noting include:6

- Rule 26(b) defines the discoverability and accessibility of electronic information across a tiered storage environment. This rule also touches on privacy of information in redacted files and the inadvertent use of privileged information produced during discovery. Redaction is the processes of blocking out information as “privileged.” Companies must demonstrate the integrity of their information systems and processes, and show that records and information retention policies are enforced consistently.
- Rule 26(f) directs parties to develop a discovery plan and address issues related to preserving discoverable information. Parties must also agree on methods for producing the data in the desired format. This rule compels

organizations to recognize the strategic difference between backup and archive technologies, when it comes to "preserving" information.

- Rule 34(b) recognizes electronically stored information as a separate category that includes e-mail, video, audio and so on. With regard to electronically generated and managed information, such as application data, metadata is often included in a discovery response. Metadata, which "describes" the data, is necessary for understanding the business context of information and must be preserved.
- Rule 37(f) provides "safe harbor" from sanctions arising from data that is unrecoverable (spoiled) or lost in the normal course of business activities and routine operation of information management systems. However, care should be taken to preserve relevant information.

The FRCP rules recommend that companies implement procedures for capturing and preserving information that relates directly to an actual or pending lawsuit. However, the caveat is that keeping more data than is required makes that additional data available for e-discovery.7 In addition, current or pending litigation warrants a "litigation hold." This means that companies are required to monitor procedures to ensure that relevant data is not altered or destroyed. Effective data retention policies are critical for compliance.

The FRCP rules hold companies accountable for taking "reasonable steps" to implement processes and technology to safeguard data. To adequately prepare for e-discovery, companies must have procedures in place to assess and classify business data and establish appropriate data retention, backup, restoration and litigation-hold policies. The good news is that the technology is available to enforce effective retention policies and support e-discovery initiatives, but there are still many challenges.

What Data Management Challenges Impact e-discovery?

A recent Gartner Research Report notes that, "Organizations are experiencing an increase in the complex demands for the management and retention of historical information, driven by evolving compliance and discovery requirements. The volume of electronically stored content continues to increase dramatically, along with the diversity of content types that need to be managed . . . Organizations will need to intertwine their content governance and retention policies with information archiving technologies to meet their commitments effectively."8

The report recognizes that "Few organizations have archiving strategies for both their structured files and unstructured content that exists in transaction-oriented applications. Rather than defining a content lifecycle management plan that would include archiving, the active data store is typically allowed to grow, putting a strain on the ability of IT to support business continuity, as well as backup and recovery for these critical applications." Gartner goes on to note, "Structured data that is typically stored in databases will require special database archiving solutions."9

Why? Because the ERP, CRM and custom applications that drive your business operations collect large volumes of data that must be managed, maintained and secured. Application data or "structured data" poses additional challenges for e-discovery. This is particularly true when the data is stored in one or more complex relational databases.

For example, if data is referenced from two different databases, it must be archived together, while maintaining the data relationships. A capable database archiving solution must manage complex relational data and keep that data referentially intact. When the data model is straightforward, creating referentially intact subsets of data is relatively simple. However, the databases powering most of today's enterprise applications rely on highly complex data models.

In order to support e-discovery requirements, the ideal database archiving solution must capture data at the business object level of detail and preserve the referential integrity of the data. For example, archiving a complete "paycheck" would include time and labor transaction details, as well as the employee master data needed to fulfill reporting requirements without ties to production data.

Why Is Early IT Involvement Essential?

Communication among IT, business users, compliance experts and legal counsel is essential. Assembling cross-functional teams to define data management policies helps determine what and when data can be archived, where it should be stored, who can have access and how long it should be retained. As you implement your enterprise data management strategy, this team will evolve to manage both structured and unstructured data across the enterprise, not only to support e-discovery and audit requirements, but to improve daily operations as well.

Your records retention program must be articulated, well-executed and uniformly enforced to prevent spoliation and to comply with FRCP e-discovery requirements. Your IT staff can implement database archiving and storage strategies that allow you to meet accessibility requirements for compliance, while lowering costs.

"As an IT professional, you play the dual role of first responder and gatekeeper, ensuring that proper protocols are in place and followed. Staying up to speed on the changes to the laws and technology relating to e-discovery will certainly help your company retain the strategic edge when litigation ensues."10

How Does Enterprise Data Management Support e-discovery Initiatives?

An effective enterprise data management strategy allows you to manage application data throughout its lifecycle. Including comprehensive database archiving capabilities offers a proactive approach for analyzing the different types of data within your organization and defining the value of each set of data at various points in the lifecycle. By automating archive processes based on your business rules, you can segregate current active data from historical transactions, and have more control for implementing retention policies based on the business value of the data and your access requirements.

Ideally, archived reporting and reference data must be preserved in its original business context. You also need capabilities for managing archives on a variety of storage media in an immutable format to ensure compliance and easy access on demand. Enterprise data management archiving capabilities offer a variety of options that help you get the most value from your storage resources.

By archiving historical data, based on its business value, and implementing tiered storage strategies, you can optimize storage utilization and defer the cost of high-speed resources, while providing authorized data access for e-discovery and other reporting requirements. Archived data remains accessible no matter where it is stored. .

An enterprise data management strategy with archiving capabilities can improve the way you manage application data and support e-discovery requirements. Some of the advantages are explained in the following paragraphs.

Managing data growth. Business applications will continue to collect data. However, if left unchecked, accumulating huge volumes of current and historical data can degrade performance, limit availability and increase costs. As a recognized best practice, routine database archiving keeps application databases at a manageable size, reducing capacity requirements and lowering operational costs.

Capabilities for segregating historical from current data enable companies to manage continued growth, while keeping the data accessible for reporting purposes, audits or e-discovery requests. You can define data retention policies for managing different types of historical data to satisfy compliance and e-discovery requirements.

Current application data remains in well-managed production environments, available at peak performance. Historical data can be archived, indexed, compressed and saved to a variety of storage media. In the event of a discovery request, specific archived data can be quickly located and retrieved to control costs.

Accessing and Retrieving Data

Typically, e-discovery requests have tight deadlines for delivering the needed information. Companies need to know where archived data is located and who has access. Capabilities for indexing, locating, accessing and retrieving data quickly ensure a timely response and avoid unnecessary costs and penalties.

Archive indexing improves search accuracy, which is critical for locating needed data. Additional audit capabilities for monitoring who has access and when archived data was last retrieved provide additional levels of control. Enterprise data management options permit authorized access to archived data. In addition to native access and application independent access, an open data management access facility ensures fast data retrieval, while mitigating risk.

Protecting Data Privacy

Corporations must protect all sensitive information, whether it resides in a production system, a development database or anywhere else within organizational boundaries. Redaction, or the process of blocking out information, is an important part of e-discovery. Companies need the capability to mask sensitive data that may not be required as part of an e-discovery request.

Industry analysts recognize that data de-identification provides a viable alternative for protecting data privacy. By implementing capabilities that allow you to mask or transform sensitive information, you have the data management strategies you need to protect data privacy and preserve the integrity of your data.

Defining Data Retention Policies

According to Forrester Research, “There is a compelling case today for retention management — lower e-discovery costs and more efficient compliance processes. Organizations that do not put initiatives in place now not only will lose out on today’s business benefits but also will lose the opportunity to manage information better overall.”11

First you must determine which applications manage data that would most likely be required for e-discovery and group them into categories for archiving based on your business requirements. Next, classifying the data makes it easy to define and document business rules and data retention policies to govern active, inactive and compliance-managed data.

You can prevent information assets from becoming information liabilities by deleting historical records after they are no longer required for compliance or business purposes. For example, bank statements, cancelled checks and accounting records must be retained for seven years. If your company faced an audit or e-discovery request to support bankruptcy proceedings, and these records were not deleted after the retention period expired, they could be requested as evidence. The caveat is that if these records do pertain to the e-discovery request, you would need capabilities to apply a litigation hold to protect the records from “spoilage.”

Capabilities that allow you to apply a litigation hold or define automated policies for data retention and deletion provide added protection. You can ensure that data remains available when necessary for an audit or e-discovery request or that it can be deleted when no longer required for retention purposes.

Differentiating between Backups and Archives

Both database archiving and database backup are vital components of best practices that are essential to a company’s day-to-day operations. Backups make it possible to recover data to ensure business continuity. In contrast, archives are designed for long-term data retention, especially important for e-discovery. However, each plays an equally important, but different, role in enterprise data management.

Consistent backup processing ensures that in the event of a disaster or inadvertent data loss, operational data can be recovered quickly from the last backup point. Successful recovery depends on how often and how well backup procedures are performed. However, because backups are often overwritten, this procedure is not designed to preserve data for the long term and would not support your e-discovery requirements.

In contrast, ongoing database archiving allows you to manage and maintain historical data and associated metadata for reporting and compliance purposes. Your archived data remains accessible on demand to support your e-discovery efforts. As an additional benefit, implementing routine database archiving can significantly reduce backup and recovery times. Keeping databases at a manageable size allows you to maintain acceptable levels of performance and availability.

Storing Data throughout Its Lifecycle

Companies need a variety of cost-effective storage alternatives and capabilities for managing data in an immutable format for e-discovery. Tiered storage strategies enable managing data based on its business value and access requirements, until retention periods expire.

For instance, it makes sense to manage active data in a high-performance storage environment for rapid transaction throughput. Reporting data can be relocated to mid-tier storage, so you control costs, while still meeting service requirements.

Finally, reference data can be maintained in an immutable format on a secure WORM (Write Once, Read Many) device, where defined policies govern its access and ultimate disposal. The flexibility of enterprise data management allows IT organizations to leverage the best price/performance platform for storing enterprise data throughout its lifecycle.

Retiring Applications

Periodically reviewing your application portfolio helps you balance the cost of operating and maintaining your applications against the business value they provide. Your objectives may be to simplify your IT infrastructure by eliminating redundancy, migrating applications to more cost-effective platforms or reducing dependence on systems that are no longer supported by a vendor.

An enterprise data management strategy that includes database archiving ensures access to application data, even after the originating application is taken out of service. Sites can satisfy an e-discovery request by using industry standard methods to query and report on archived business records.

What Benefits Does Enterprise Data Management Deliver?

Ultimately, implementing enterprise data management strategies that include database archiving will deliver measurable benefits to support your audit and e-discovery requirements:

- Manage continued data growth by archiving to segregate current from historical data and storing historical application data in an immutable format as required by FRCP. In addition, keeping databases at a manageable size improves service level performance, increases availability and lowers operational costs.
- Define and enforce automated data retention, litigation hold and disposal policies to ensure that the appropriate information remains available when necessary and is deleted when no longer required for compliance.
- Ensure on-demand access to current and archived data using archive indexing techniques and a variety of access methods, such as native access or application-independent access.
- De-identify sensitive data that may not be required as part of an e-discovery request. Apply a variety of data masking techniques to prevent the inadvertent exposure or privileged information and support redaction requirements.
- Implement tiered storage strategies to manage historical data based on its business value and access requirements. Facilitate managing data cost-effectively throughout its lifecycle.
- Retire or decommission applications that no longer deliver business value. Archived application data remains accessible long after the application is taken out of service.

The ideal solution should address the critical issues that support your audit and e-discovery requirements.

What Should You Look for in an Enterprise Data Management Solution?

The ideal enterprise data management solution should provide a consistent technology and methodology that can be scaled across diverse applications, databases, operating systems and hardware platforms. Find a solution that provides proven capabilities for addressing the critical IT issues that accompany e-discovery and audit requests. Archive and retrieve complete business objects and metadata. Focus on archiving capabilities that ensure the integrity of the data in its complete business context. The ideal enterprise data management solution must have the capability to capture complete business objects and the associated metadata. You should also have capabilities that allow you to selectively retrieve archived transaction records without having to restore, and capabilities for restoring archived data to production or to an alternate target environment.

Integrated archiving and indexing capabilities can promote faster searches across archives. The ideal solution should allow you to establish one or more indexes for each archive to reduce the time it takes to retrieve the precise data you need.

Offer compatibility with a variety of storage alternatives. Determine whether to store archives in a database, in a file or perhaps in different formats at different points in time. Each alternative offers benefits. For example, when you begin an archiving project, your staff can probably manage the archive database easily. However, as time progresses, the archive database can balloon to the size of the original production environment, forcing you to archive that database as well. Any potential storage savings are erased, and project management efforts quickly become burdensome.

In contrast, compressed file formats require a smaller footprint, so you can maximize storage savings. Compressed files can also be indexed, enabling rapid retrieval of archived data. Most importantly, managing archives in a file format offers the broadest range of access methods over the life of the archive.

Provide Long-term Data Access Options

Typical alternatives for accessing archived data include native (application-based) access and application-independent access. Some archive solutions provide only one access method or the other, but few provide both. You need to weigh the advantages of each method for meeting your specific access requirements. For example, native access allows you to interact with archived data through the original application interface. This method allows users to access the information they need, using familiar formats and processes. However, if you plan on retiring an application and still want to retain access to your archives, application-independent access may be the better alternative.

Application-independent access provides the most flexible range of access alternatives over the life of the archive. This approach enables authorized users to interact with archives using industry standard methods and report writers. You can use a variety of industry standard access methods, like ODBC/JDBC, XML and SQL, and reporting tools, like Business Objects™, Cognos® or even Microsoft® Excel. Leveraging this access path, you “future proof” your archives because you retain access long after the originating application has been upgraded, changed or even retired.

Satisfy Data Retention and Litigation Hold Requirements

Next, focus on capabilities for managing archives and retention periods cost-effectively and consistently over the full lifespan of your data. Capabilities for saving archives on a variety of storage media allow you to future-proof methods for managing data, based on its business value and access requirements. Look for a solution that offers a litigation hold feature. You can keep business records accessible until legal retention periods expire and archives can be deleted either manually or by means of automated disposal of expired transaction records.

IBM Optim Supports Your e-discovery Initiatives

Preparing for e-discovery simply makes good business sense. Enterprises must recognize that there is a business value in organizing their information and enterprise application data. Managing enterprise application data for e-discovery poses many IT challenges. To comply with FRCP rules, companies must take some “reasonable steps” to implement processes and technology to support e-discovery requirements.

The IBM® Optim™ Data Growth Solution provides enterprise data management and proven database archiving capabilities that can help you meet audit and e-discovery requirements:

- Archive and retrieve complete business objects and metadata
- Offer compatibility with a variety of storage alternatives
- Provide long-term data access options
- Satisfy data retention and litigation hold requirements

Enterprise data management strategies that include database archiving offer effective methods for managing, storing, accessing and retrieving data to support e-discovery requirements. As a recognized best practice, database archiving also offers other benefits in the form of improved application performance and availability, while reducing risk and lowering operating costs.

About IBM Optim

IBM® Optim™ enterprise data management solutions focus on critical business issues, such as data growth management, data privacy compliance, test data management, e-discovery, application upgrades, migrations and retirements. Optim aligns application data management with business objectives to help optimize performance, mitigate risk and control costs, while delivering capabilities that scale across enterprise applications, databases and platforms. Today, Optim helps companies across industries worldwide capitalize on the business value of their enterprise applications and databases, with the power to manage enterprise application data through every stage of its lifecycle.

For More Information

To learn more about IBM Optim enterprise data management solutions, contact your IBM sales representative or visit: www.optimsolution.com

IBM Software Group
111 Campus Drive
Princeton, NJ
USA, 08540-6400
800.457.7060
609.627.5500
Fax 609.627.7799
www.optimsolution.com

Produced in the United States of America
03-08

1 Barry Murphy and Robert Markham, “eDiscovery Bursts Onto the Scene,” Forrester Research, March 1, 2006.

2 “Fulbright Launches its Third Annual Litigation Trends Findings – U.S.,” Press Release, October 10, 2006.

3 Ibid.

4 Ibid.

5 Debra Logan, “Mapping Technology and Vendors to the Electronic Discovery Reference Model,” Gartner Research, ID Number: G00153110, November 9, 2007.

6 Vivian Tero, “Worldwide Legal Discovery and Litigation Support Infrastructure 2007 – 2011 Forecast: Legal Matter and Compliance Records Repositories Are the Early use Cases of the Active Archiving Architecture,” IDC Market Analysis, IDC#207155, June 2007, p. 28-29.

7 Alan Brill and Michele C.S. Lange, “The New E-discovery Rules: What You Don’t Know Can Definitely Hurt You and Your Company,” Computerworld – Storage, December 11, 2006.

8 Kenneth Chin and Carolyn DiCenzo, “Key Issues for Information Archiving and Retention,” 2007, Gartner Research, ID Number: G00146211, March 9, 2007, p. 2.

9 Ibid. p. 3.

10 Alan Brill and Michele C.S. Lange, “The New E-discovery Rules: What You Don’t Know Can Definitely Hurt You and Your Company,” Computerworld – Storage, December 11, 2006, p. 2.

11 Barry Murphy, “Abysmal: The State Of Retention Management, Forrester Research, July 17, 2007, p. 10.

IME14002-USEN-00

NYOUG 2008 Sponsors

The New York Oracle Users Group wishes to thank the following companies for their generous support.

All Star Software Systems (http://www.allstarss.com)
Bez Systems (www.bez.com)
Confio (www.confio.com)
Fusion-io(www.fusionio.com)
GoldenGate Software (www.goldengate.com)
IBM (www.ibm.com)
Oracle (www.oracle.com)
Texas Memory Systems (www.texmemsys.com)
TUSC (www.tusc.com)
VERSATILE printing applications (www.getvpa.com)

Contact Sean Hull and Irina Cotler for vendor information, sponsorship, and benefits

TUSC acquires WhittmanHart Consulting
– expands into Enterprise Performance Management

ORACLE | Hyperion

With 20 years of experience delivering analytic applications that extend visibility into the enterprise and beyond, TUSC's new Enterprise Performance Management Group ensures that you articulate and seamlessly implement an end-to-end EPM strategy for your business. Leveraging a combination of Oracle EPM solutions as well as proprietary financial templates, driver-based planning models, Industry specific scorecards/dashboards, and reference architectures, we help you drive value from a financial, customer and shareholder value perspective.

The Oracle Experts

TUSC

A ROLTA COMPANY

TUSC's other Core Service Areas

Oracle's E-Business Suite
- Implementations
- Upgrades
- Remote Support

BI / Data Warehousing
- Strategy
- Implementation

Custom Development
- Fusion Middleware
- Web Services & SOA
- Application Migrations

Product Licensing
- Oracle
- Instant SOA

Managed Services
- Remote Support
  - DBA and Database
  - Oracle E-business

Database Services
- Health Checks
- Strategic Planning
- Infrastructure Services
- Full DBA Support
- Remote Support

Training
- Classroom
  - (offsite & onsite)
- Mentoring

ORACLE CERTIFIED ADVANTAGE PARTNER

GSA Contract Holder
Contract GS-35F-0149U