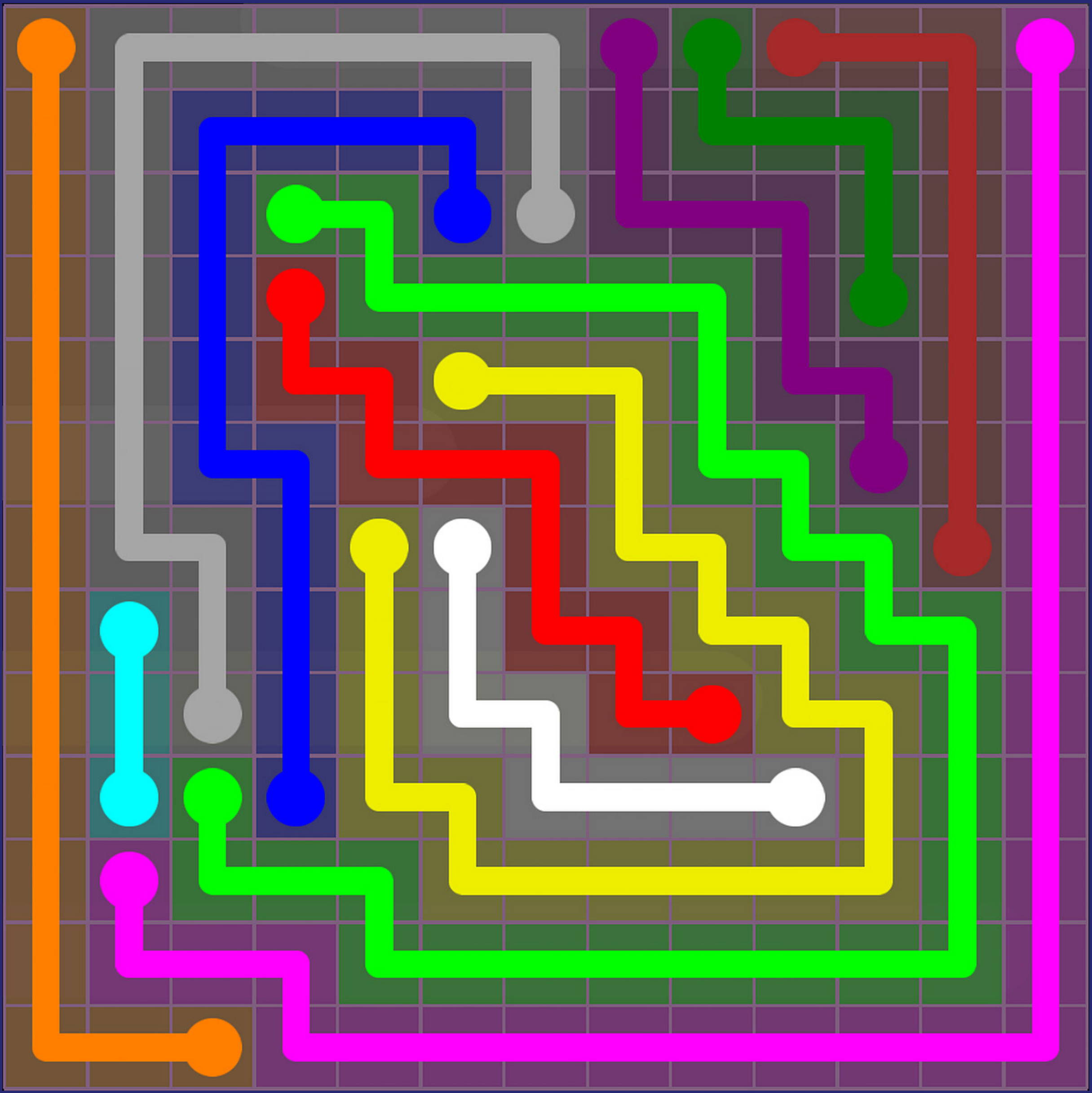
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 29, No. 4 · NOVEMBER 2015 $15



YesSQL(T)

With Carlos Sierra and Mauro Pagano.

See page 4.

Brian's Notes

Oracle Database 12c Security.

See page 7.

The Easy Button

Packaged Analytics.

See page 20.

Much more inside . . .

ce Antoinette Meredith Daisy Nada Liz Amie Jeannine John
Robbi Kathyrn Steven Gisele Tim Laura Myron Elizbeth
e Sun Darell Karan Janine Bert Larry Rodrigo Rachel Todd
an Brittaney Sterling Nedra Simon Christy Guy Clare Rege
m Margaret Dan Sunny Nicho aitlin Kareem Loyd Meg
Brian Torrie Renato Jan Andy Sharlene Andre
na Frank Arlena la Ka en Tom
erri Sab en An enry Bev
holle C areem En Na
Theron Stev Gisel
ty Elizbeth Jeann rice Antoinette M
ie Jeannine Theron elina Robbi Kathyr
inette Meredith Da nnine Theron John
Robbi Kathyrn Gisel te Meredith Daisy

Toad enables connections with a community of millions

The Toad World community? It's experience. It's knowledge. It's tips, tricks, and real-time help straight from the experts. And it's inside the latest version of Toad for Oracle. Realize the power of connected intelligence.

Learn more at **software.dell.com/toadconnected**

DELL Software

Professionals at Work

First there are the IT professionals who write for the *Journal*. A very special mention goes to Brian Hitchcock, who has written dozens of book reviews over a 12-year period. The professional pictures on the front cover are supplied by Photos.com.

Next, the *Journal* is professionally copyedited and proofread by veteran copyeditor Karen Mead of Creative Solutions. Karen polishes phrasing and calls out misused words (such as "reminiscences" instead of "reminisces"). She dots every i, crosses every t, checks every quote, and verifies every URL.

Then, the *Journal* is expertly designed by graphics duo Kenneth Lockerbie and Richard Repas of San Francisco-based Giraffex.

And, finally, David Gonzalez at Layton Printing Services deftly brings the *Journal* to life on an offset printer.

This is the 116th issue of the *NoCOUG Journal*. Enjoy! ▲

—*NoCOUG Journal* Editor

Table of Contents

Publication Notices and Submission Format

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

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

The submission deadline for each issue is eight weeks prior to the quarterly conference. Article submissions should be made in Microsoft Word format via email.

ADVERTISING RATES

The *NoCOUG Journal* is published quarterly.

Size	Per Issue	Per Year
Quarter Page	$125	$400
Half Page	$250	$800
Full Page	$500	$1,600
Inside Cover	$750	$2,400

Personnel recruitment ads are not accepted.

journal@nocoug.org

Carlos Sierra

YesSQL(T)

with Carlos Sierra and Mauro Pagano

Mauro Pagano

What is the history of SQLT? How did it evolve? Who contributed to its development? Why do I need it? How do I get it?

Carlos: SQLT, also known as SQLTXPLAIN, has been around for almost 16 years. On December 2, 2015, it celebrates its Sweet 16! At the beginning, its name was coe_xplain.sql, and it was a simple script to produce an execution plan for a given SQL text. Since its creation, it included metadata for tables and indexes related to the SQL of interest, in addition to the execution plan. Over time, coe_xplain.sql evolved into a more complex script. Then on October 21, 2002, it became SQLTXPLAIN. So I may say SQLTXPLAIN was born 13 years ago, but its predecessor coe_xplain.sql was already 3 years old.

SQLT evolved from a simple script with 350 lines of code to a complete tool with over 150K lines of code! Then there is SQLd360, which is the "new SQLT," but I will let Mauro expand on that.

I was the original developer of SQLT, and when I left Oracle and went to work for the Accenture Enkitec Group (AEG) in August 2013, Mauro took over. But years before that, Mauro contributed many ideas, like the one about SQLT XPLORE. Abel Macias also contributed a lot early on, and now he maintains SQLT at Oracle. Mauro is currently also working for AEG.

Mauro: When Carlos left Oracle, I worked on including most of the 12*c* new features known at the time (not many clients were using 12*c* back then). I always joke, saying I spent most of my time fixing Carlos's bugs, but truth be told, I probably added more than I fixed!

SQLd360 is a natural evolution of SQLT. This brand-new SQLd360 free tool encapsulates our current view on SQL tuning, and it has been built based on the lessons learned with SQLT, on the limitations we wanted to address, and on the approach we now take to SQL tuning.

Carlos: Mauro is very humble. I recognize SQLd360 as "The Next Generation," while SQLT remains like the original "Star Trek." Personally, I like this "next generation" better. Walt Disney said once, "We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths." Mauro is now keeping the effort I started with SQLT moving forward with SQLd360. First he enhanced SQLT, now he works on a new and fresh remake.

I'm sure you have some special success stories of how you solved complex SQL tuning problems using a SQLT report.

Mauro: There are many! I remember one where a DBA walked into his office and found out everything was horribly slow because the execution plan of one of his massively long SQL statements had changed. It was one of those typical cases of "it's so complex nobody wants to touch it, but it works so we are safe," and it was taking 30 seconds instead of a few milliseconds! After a few hours trying to figure out what happened, the DBA decided to create a Service Request for Oracle, and I took it. We asked for SQLT, but the DBA was reluctant to install a tool in production. Fortunately, the issue was so urgent that he decided to go ahead and do it anyway. Once we got the SQLT files it took us 3 minutes to solve the issue! The automatic statistics-gathering job introduced a histogram on a varchar2 column that had very repetitive leading characters (in 11.2, only the first 32 characters are considered when building a histogram), so the CBO moved from seeing millions of distinct values for that column to just a few thousand, big change in selectivity and thus a big change in the execution plan! The DBA was so surprised that he proactively installed SQLT anywhere he could, "just in case I'll need it tomorrow."

Anyway I think what makes SQLT a great tool is not how we were able to solve one specific case, either simple or complex, but the fact that in almost every success story, SQLT was there, playing either a small or big role, but it was there helping the engineer do his/her job in a more efficient way.

Carlos: One I dearly remember is a request to help a client on a recent 10*g* upgrade that caused havoc at this company, where a twin database kept producing different execution plans. Both databases were supposedly identical, yet one produced inconsistent execution plans, changing often and with erratic performance. The other "twin" database was more consistent. The problem required me to create the SQLT COMPARE method, which basically allowed us to identify the root cause. It had to do with column usage and the automatic generation of histograms using a very small sample size and producing an incorrect number of distinct values and missing buckets. There are many complex issues we solved with the help of SQLT, but this one was special since the original hypothesis for all of us was that both systems were identical—and they were indeed in terms of structure and data—but back on 10*g* we were not fully aware of the consequences of "auto" functions of the ever-changing DBMS_STATS package. SQLT provided the irrefutable proof that we needed at that time!

I am in awe. But, if SQLT is the godsend you say it is, why isn't it widely known and used? Is it just a matter of education and evangelization? (I hope this interview helps a little.) Or are there other barriers to adoption?

Carlos: I would have to disagree here. SQLT is widely used around the world. If you ever log a ticket with Oracle Support about a SQL performing poorly, there is a big chance you will be asked to show your issue with SQLT. Keep in mind that according to many engineers in Oracle Support, this SQLT tool saves over 50% of their time when they need to diagnose a SQL statement. Some engineers in Oracle Support even make SQLT a hard requirement in order to progress an issue! The alternative to not using a tool like SQLT is usually a lengthy and painful investigation.

Mauro: SQLT isn't magic; it doesn't do anything anybody with lots of time on his/her hands could not do. What I loved about SQLT during my Support days is that it defines kind of a communication standard between customers and Support. The advantage is twofold: on one hand, clients really simplify and speed up diagnostics collection, plus they make sure not to forget any piece the engineer might need. On the other hand, the Oracle Support engineers receive information in a consistent format so they can focus on the real problems instead of just trying to put pieces together. I remember one customer that was sending me info in Word docs, Excel docs, screenshots of execution plans, and extracts from internal emails at his site; it took me way more time to put the pieces together than to solve the real issue.

I agree with Carlos on SQLT being widely used, especially when dealing with Oracle Support. Outside Oracle I think there are two main barriers to wide adoption of SQLT.

The main one is that it requires an installation. Imagine you are in the middle of a production crisis with your boss over your shoulder waiting for you to fix an issue, and you turn around, look at him or her, and say, "Hey we need a change request [that usually takes days to get processed] to create two new schemas before we can even start using a tool that can help us solve the problem." How do you picture your boss's face? Yeah, me too . . .

The second one, and Carlos may disagree here, is that SQLT is a tool designed by Support to be used by Support; it covers the needs of a Support engineer. It provides a very large amount of information (which translates to long execution times) that, at best, doesn't interest the average DBA/developer. Still, the major drawback of using a tool designed for somebody who looks at SQL tuning issues all day long is, IMHO, that there is very little help in making data analysis easier. It's expected that whoever reviews a SQLT knows what to look for and where. Unfortunately, not everybody outside Oracle Support tunes SQL statements all day long. The first thing that comes to my mind is the tabular format used in the main report. It works well for some types of info (e.g., DBA_TABLES details), but it's painful to review some others (e.g., DBA_HIST_SQLSTAT).

Carlos: Mauro has some good points here, which I need to expand a bit. Mauro and I delivered many sessions of an internal SQL tuning workshop at Oracle while training over 700 Oracle engineers around the world. During those sessions we listened to the needs of many potential users of SQLT, so I would say the content of SQLT was not coming from thin air, and it was not just a product of its authors' prolific imaginations. SQLT actually reflects many needs of many people dealing with SQL tuning issues often. It is just a big tool that collects and displays diagnostic details about pretty much everything that can affect the performance of a SQL statement.

Is SQLT completely perfected or is there still room for improvement? Are you continuing to expand its feature set?

Carlos: The same way SQLTXPLAIN improved over its ancestor coe_xplain.sql, these days a new and fresh tool, SQLd360, is slowly but surely replacing SQLT. Why is that? Some core reasons follow: 1. SQLd360 does not install anything on the database (SQLT does), and this is so important for a very large number of clients. 2. SQLd360 is 100% free software, while SQLT requires a My Oracle Support (MOS) account, and some consultants and third parties may struggle to obtain a valid login for MOS or would have to use their client's credentials. 3. SQLd360 makes use of newer technologies including Google Charts, presenting the metadata in a very "Wow!" way, while SQLT focuses on HTML tables. 4. SQLd360 is 13X smaller in terms of lines of code, and it pretty much does the same job as SQLT 5. SQLd360 is much faster to execute than SQLT.

Mauro: Nothing is perfect, especially in IT!

About SQLd360, Carlos said pretty much everything. What I would like to add is an invitation to try out the tool and provide feedback to us so that we can make the tool as useful as possible to the community. The more we hear from the community, the more we can understand what people use the tool for, what is missing, and what might need to be changed or improved. We'd like to think our tools make other people's jobs easier, and it's our way to say "thank you" to a community that every day teaches us so much! (Editor's Note: Please send feedback to **mauro.pagano@gmail.com**.)

Carlos: Again, Mauro is humble here. Remember, SQLd360 is The Next Generation: younger, faster, thinner, sexier, etc. Honestly, I no longer require my clients to install and run SQLT. I simply tell them about SQLd360 and ask for its output. I am proud of SQLT, and I am also proud of SQLd360. The former served its purpose well, and Oracle will continue to maintain and use it for the foreseeable future, but the latter is the future!

Do SQLT, SQLd360, and eDB360 draw upon Statspack data. Or are these tools only geared toward the extremely fortunate 1% who have a license for Diagnostics Pack and Tuning Pack?

Mauro: Where is the 1% coming from?

The goal of all these tools is to make full use of what is available in the database, so depending on the licensed options the output files will include more or less information.

AWR/ASH are a goldmine of information; they are the foundation of basically every serious performance-monitoring tool, and they are, IMHO, the best way to approach the performance challenges that companies face nowadays. Especially from 11*g*, where ASH is able to capture information that is almost impossible to extract outside the kernel (e.g., the execution plan line ID that the SQL was on at the time of the sample).

I have a hard time picturing a production database without Diagnostic Pack.

SQLd360 relies a lot on ASH and there is no SP equivalent, so there would be little benefit in using Statspack. Having said that, even without any additional option, SQLd360 would provide quite a lot of info, focusing on those APIs that are part of the base software.

There are some free alternatives to ASH and I would be very happy to consider them, but the reality is that the Diagnostic Pack is likely to get more and more complex with info that will be harder and harder to extract from "free" views.

Carlos: In my humble opinion, having at least the Diagnostics Pack license should be a must on a production database. This pack, which includes AWR and ASH, as Mauro said, is crucial when it comes to fast and accurate diagnostics of performance issues on a mission-critical Oracle database. Yes we could use Statspack as a second best, and yes SQLd360 (and SQLT) could be enhanced to mine data from Statspack, but on the big scheme of cost and benefit, these requirements fall short. Maybe we need to revisit this topic if more clients drift out of the Diagnostics Pack, but for now we benefit from what AWR and ASH kindly provide to all of us. Just be sure you have the Diagnostics or the Tuning Pack licenses before enabling those features on SQLT and SQLd360.

We're excited about your "Practical SQL Tuning" workshop at the YesSQL Summit in January. What topics are covered?

Carlos: We are also excited. This one-day class focuses on what is really used on real-life cases where we have to do some SQL Tuning. Of course it has to start with some foundations, but it moves rapidly into controlling and manipulating Execution Plans. I'll let Mauro expand on its content. I hope to see many of you there, bringing lots of questions.

Mauro: The workshop covers the 4 "T's" of SQL Tuning: Techniques, Tips, Tricks, and Tools. It takes a practical approach to the topics, keeping the focus on how to actually tackle issues and leverage Oracle functionalities rather than exploring the theory behind each aspect. The session is interactive, with several examples to keep the conversation close to real life.

The first "T," on the Techniques chapter, covers the different approaches to SQL performance, from the fundamental steps to identify bottlenecks in SQL processing to the multiple options available in Oracle to improve SQL performance.

The second "T," on the Tips chapter, is a collection of best practices and lessons learned throughout the years on how to avoid or address SQL performance issues, especially when it comes to the first reason for poor SQL performance: CBO statistics.

The third "T," on the Tricks chapter, groups together all those "mini-tools" you have in your Swiss knife when it comes to tackling and solving SQL problems, covering both custom scripts and standard Oracle APIs.

The last "T," on the Tools chapter, is an overview of what is freely available out there in terms of tools for SQL tuning, with special focus on what we worked on for years: SQLT and SQLd360.

I cannot adequately express my appreciation for your contributions to the Oracle Database community; you guys are just the greatest! Does SQL tuning continue to be your exclusive focus, or are you moving on to greener pastures at Accenture Enkitec Group?

Mauro: While at AEG I still do a lot of SQL tuning, even though it's not my exclusive focus (it wasn't the exclusive focus at Oracle either). We are part of a team led by Carlos that focuses on database health checks, where we take a holistic approach to the overall database as a component of a larger (and more complex) architecture. Basically we focus not only on the database health on its own but also on how applications use (or misuse) the database. I really enjoy the type of work we do because by using a top-down approach, you are able to identify real application bottlenecks and problems that when addressed make customers achieve a level of performance that is almost impossible when focusing on the database alone. So we are now doing SQL Tuning and a lot more. Have you heard of "Tuning as a Service (TaaS)"? Well, we are helping on this new and exciting offering! With TaaS, our clients get some hours of our time, which they can use to tune some challenging SQL or have us look at some global database bottlenecks; it is exciting!

Carlos: I see what I do regarding performance tools as my little contribution to the Oracle community, from which I have received so much. I have learned pretty much all I know about Oracle from reading material others have put together: books, blog posts, articles in magazines, distribution lists, conferences, etc. I just do my share, trying to help our Oracle community to become a better one for the benefit of all of us.

These days Mauro and I are expanding our scope, and we are providing another free software: eDB360. This tool is, for an entire Oracle database, what SQLd360 is for one SQL statement. In other words, it presents a 360-degree view of an Oracle database. We use it every day to perform unbiased health-checks of Oracle databases, as Mauro mentioned. By reviewing the output of this eDB360 tool, we can build a story behind a database's current state and elaborate on sound recommendations to improve the database in terms of scalability, performance, availability, etc. As always, we use our own tools, which helps us to make them better from version to version. Many have asked, "Why do you guys share for free what you use to do your job?" And my answer is, "Why shouldn't we?" I dream of a world where more people freely share knowledge and tools for the benefit of their communities! By the way, have you heard of the new and free Apache Log Analyzer (apalyzer) developed by Dimas Chbane? Maybe a topic for another day. ▲

Carlos Sierra is a regular speaker at Oracle User Group conferences on topics related to Performance and SQL Tuning. He is the author of tools like SQLTXPLAIN, SQLHC, eDB360, and eSP (Enkitec's Sizing and Provisioning). Carlos has almost 20 years of experience in Oracle databases, and many more on legacy UNISYS and IBM mainframes. Carlos currently works as a consultant for Accenture Enkitec Group, where he leads the Oracle database Health-Check team. His home is Seattle now, after 20 years in sunny Florida. He and his wife have raised four responsible young adults, and now they enjoy the peace of an empty nest. Carlos enjoys helping and mentoring others, and contributes to the Oracle community as much as his free time permits.

Mauro Pagano is a database performance engineer with special interest in SQL Tuning and Wrong Results. He is an active member of the Oracle community, always willing to help or mentor other peers, and he enjoys giving back by developing tools and presenting at Oracle User Group conferences. He is the author of SQLd360 and the previous maintainer of SQLTXPLAIN and SQLHC. Mauro has over a decade of experience in the Oracle world, from database tuning to applications development. Mauro currently works as a consultant for Accenture Enkitec Group, as a member of the Oracle database Health-Check team.

Oracle Database 12*c* Security

A Book Review by Brian Hitchcock

Details

Author: Scott Gaetjen, David Knox, William Maroulis

ISBN-13: 978-0071824286

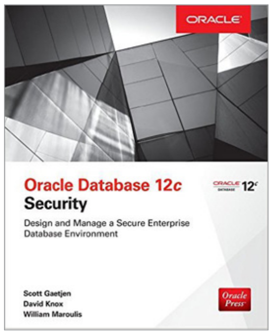
Pages: 560

Year of Publication: 2015

Edition: 1

List Price: $70

Publisher: McGraw-Hill Osborne Media



Summary

Overall review: A very good way to learn a lot about all the many security features and options available for the Oracle 12*c* database. Most of the material applies to recent pre-12*c* versions as well. You don't have to know a lot about 12*c* to understand what is being discussed, although it would be easiest if you knew about container and pluggable databases before you get started. The authors do explain basic 12*c* features as they go along, but if it is the first time you are hearing about containers, it might be a little confusing at first.

Target audience: DBAs, application developers, and security specialists.

Would you recommend this book to others?: Yes

Who will get the most out of this book?: Anyone who is involved with security issues for systems that include an Oracle database.

Is this book platform specific?: No

Why did I review this book?: I saw a copy at a recent NoCOUG conference and liked the table of contents.

Introduction

The overall layout of the book is broken down into three sections: Essential Database Security, Advanced Database Security, and Security and Auditing for the Cloud. For Essential Database Security we are told that user database account mapping is one of the most important decisions we will make, and this will be discussed later. Mapping refers to how users will connect to applications and databases to do their jobs without causing security issues. Most users connect to applications through a browser, and the application connects to the database, which means that for most users, their identity is not known to the database. The terms "access control," "authorizations," and "privileges" will be covered, as they describe the foundations of database security. A new term to me, "entitlement analytics," describes the ways we can examine which privileges users have and how they were assigned. Another new term for me is "Oracle Real Application Security," described as a declarative security model for applications.

For Advanced Database Security, we will learn about Virtual Private Database (VPD), data redaction (DR), and features in 12*c* that help us find sensitive data in our databases. Oracle Label Security (OLS), Oracle Database Vault, and Transparent Data Encryption (TDE) will also be discussed. If you feel your life lacks acronyms, relax, you will have many more to deal with as we move along.

For Security and Auditing in the Cloud, we will learn about what the term "cloud" means from a security perspective and about meeting compliance regulations, and review an approach to setting up all of this database security for yourself.

Chapter 1: Security for Today's World

The first thing we are told is that securing your database can be an overwhelming task, and I agree. If, as you read this book, you feel overwhelmed, you're paying attention. Security is a complicated subject, and it would be even more complicated to try to implement all the things that will be described. We are told that we will discuss how to secure the Oracle database in general and then cover the new features that Oracle has developed specifically for Oracle Database 12*c*. The authors also point out that while they focus in on the Oracle database, we need to look at the entire system within which the database lives. Again, I agree: it is so easy to slip into a detailed discussion of any given Oracle product when, in reality, all of the Oracle products you are dealing with live in an overall system that has many other components and technologies from multiple vendors. This points out just how complicated security issues really are. It is good to learn about Oracle database security, but we need to be careful: it is easy to get lost in what we know best and to ignore the complexity we don't understand as well. Another good point is offered next: many times we assume that our infrastructure will prevent any and all bad actors from getting into our environments. This assumption is identified as among the worst mistakes we can make. We are encouraged to assume that no matter what we do, someone can still get in, and bad things can still happen. This leads to another important point, perhaps my personal favorite, that the insider threat is always present. For all of the new and old technologies we have for securing the database and other components, how can you prevent someone like me, who has full access to the database, from simply doing bad things, ones that you may never even know I have done? We are told that what we learn in this book will protect against insider threats, but I remain skeptical.

Security issues have changed just as quickly as the Oracle database and we are told that many things we do now in the name of database security are based on security concepts that are roughly 15 years old. I don't disagree, but I think the choice of 15 years is interesting. Not to worry: the 12*c* database has lots of new stuff that will replace all the old stuff.

The evolution of security technologies is covered next, starting with the Evolving Four A's, which are authentication, authorizations, access, and auditing. I was waiting to see if it would come up, and it does, that multi-factor authentication is described as essential in modern systems. I have been required to use two-factor authentication for years at work. I have also seen this become available for online brokers, and it is an option I use on my WordPress website. I'm glad to see this book really does seem to be up to date. The discussion centers on users interacting with the database; this has changed, as most users connect to applications that use connection pools to access the database.

Chapter 2: Essential Elements of User Security

The focus shifts to discuss the importance of database accounts and how to manage this efficiently in 12*c*, although it is largely relevant to previous versions as well. Since 12*c* is all about consolidating many databases into one multitenant database, we now have even more sensitive data stored in one place. How you map users to accounts is covered in detail. Identification and authentication are reviewed. Identification covers user-supplied information, such as usernames, and what is called technological identification, such as biometrics.

Once a user has been identified, the critical step of authentication is next, and multifactor authentication is said to be best practice. This means more than one, perhaps a password and a token or biometric. Database account types are covered, including end users, connection pools, non-person entities (SkyNet?), application schemas, operational DBAs (do we still have them?), and application DBAs. The 12*c* database architecture is described to show that the account types haven't really changed. Further, 12*c* supports all the mechanisms from previous releases. There are new Administrative Privileges in 12*c*, such as SYSDG for Dataguard, SYSKM for key management, and SYSASM for ASM administration. These are to support separation of duty, where different people are responsible for different aspects of database administration so that no one person has all the privileges. In the text the phrase "another set of administrators" is used, which I find amusing, because in my group, we don't have enough bodies to cover the shifts, let alone staff another set of administrators. There are new system accounts associated with these new privileges and new internal tables, such as V$PWFILE_USERS, where we can see which account has which privilege. We are told that it is best practice to lock and expire the password for these accounts and create separate named accounts for individual administrators. I haven't seen this done in the wild. Further, the authors tell us that the operating system for the database is assumed to be secure. I guess I have to assume that, but I was looking for more details about this.

Oracle OS authentication is discussed, including creating users and groups for each admin privilege. It turns out the pluggable databases (PDBs), which are the individual databases you consolidate into one container database (CDB) in 12*c*, require a different process for OS authentication. You have to connect to the container database first and then alter your session to the pluggable database where you want to work. You can't use OS authentication to connect directly to a PDB.

Many options are discussed and many syntax examples are given.

New in 12*c* are common accounts: database accounts that are present in all the PDBs that have been consolidated into the CDB. How to create these accounts and the issues this brings up are covered. For example, common account names must start with "C##," which will take me a while to get used to. It looks like some kind of coding prefix or some kind of wildcard that must get replaced later. How to create accounts in each PDB is discussed as well as some issues with grant and revoke. Options are presented for managing local database accounts and common accounts. This includes passwords and maximum failed logins.

Chapter 3: Connection Pools and Enterprise Users

As mentioned earlier, most clients connect to applications through a browser. For performance reasons we don't want to have users creating their own database connections, so we create connection pools where persistent database connections are used by application users as needed. Each time a database connection is created or destroyed it takes system resources. The application users connect to the application, which connects to the database through the connection pool. This creates the issue that the database doesn't know anything about the end user. This affects security auditing and performance. Connection pools are discussed and the security issues described. Database security can't be based on information specific to the application user. Within 12*c* Oracle offers external authentication mechanisms, including Proxy Authentication, Enterprise User Security (EUS), Kerberos, and Real Application Security (RAS).

Proxy Authentication is part of the Oracle Call Interface (OCI) connection pool. This mechanism starts with creating end user accounts that have an "impossible password," which is an undocumented Oracle feature. My training has been that we don't use undocumented features, but the authors tell us to and provide examples of the required syntax. The idea is that the end users won't have a password in the database at all, but will use a proxy account called APP_POOL to connect. The end user doesn't know the APP_POOL database account password. The example continues showing how this works for application servers, and includes using LDAP.

Enterprise User Security (EUS) uses LDAP to authenticate and authorize database users, which means database users are created in LDAP, not in the database itself. This centralizes user administration in LDAP, supports Single Sign On (SSO) and is more efficient for large numbers of end users. Directory Services are discussed as well as the Oracle components of Identity Management, such as Unified Directory, Internet Directory, and Directory Integration Platform. I promised you lots of acronyms and here they are! The process to set up EUS is shown, including screenshots. For converting an existing database, the User Migration Utility is provided for bulk migration of existing database users. EUS has a performance concern as each user has to be authenticated against LDAP versus the local database. It is also pointed out that this creates a single point of failure; if LDAP is down, the whole database environment is inaccessible.

Kerberos Authentication is covered with a simple example and Real Application Security is covered in Chapter 6.

Chapter 4: Foundation Elements for a Secure Database

Now we look at database privileges and roles, starting with the terms "access control," "authorization," and "privilege." Access control is controlling user access to a resource and access control lists (ACLs) are an example of this. An ACL could be set up to describe a security policy.

Authorization represents the connection between a security policy and the specific privileges a user has. Authorization gets translated into privileges for users.

Privileges are permission to execute an action in the database. Privileges get a lot of coverage here. 12*c* has system and object privileges, as we would expect from previous database versions, and introduces intra-object privileges, which are discussed in Chapter 5. A diagram illustrates the relationship between a security policy, access control, authorization, and privileges for two different users.

System privileges can affect the entire database, which means the container database and all the pluggable databases or can be limited to one specific PDB. This is an example of new considerations we need to be aware of for 12*c*. Examples of the various types of system privileges are covered, along with how to view them in the database, including in the SYSTEM_PRIVILEGE_MAP view.

Object privileges and how to view them is next, followed by column privileges. New to 12*c* is the ability to grant INSERT, UPDATE privileges to individual columns on a table. This is useful with other database security features such as Virtual Private Database (VPD) and Oracle Label Security (OLS), both of which have separate chapters later in the book. The SQL needed to set up column privileges is provided for the mythical EMPLOYEES table. Clearly the authors have a dry sense of humor, as the example shows managers authorizing raises for the employees. Does this really happen? Perhaps I've worked at Oracle too long! Synonyms in 12*c* can have user-friendly names for objects, and SQL is shown for this new feature. The example also points out some subtle security issues that can be created if you aren't careful with how you implement synonyms, specifically how this can affect previously granted privileges when the synonym is dropped. Similarly, the possible interactions between system and object privileges are illustrated using the ever-popular SELECT ANY TABLE privilege. If this is granted to a user, it doesn't matter if you revoke SELECT on a specific table; the user still has access. The WITH GRANT OPTION gets the same treatment as a possible security concern. We are told that good database security requires that we understand how granting and revoking database privileges work. Not surprisingly, roles are addressed next. Roles are shown to be more efficient for managing database privileges for large numbers of database users. We see an example where a specific grant takes effect right away, while a role assignment requires logging out and back in. In 12*c*, roles can be created in the current container (i.e., the current pluggable database), or for all containers (i.e., the single container database and all the pluggable databases). Like common users, roles that affect all databases must have names that start with C##.

Selective privileges are explained, and an example is given where a user can only access the database through an application and not directly. The application can be written to enable the role for the user as needed and then disable it. The user can't access the database directly with the role using something like SQL*Plus. The SQL to set up password-protected roles is shown. This chapter also covers global roles that use LDAP and the confusion that can come up when this is used in addition to some local database roles.

This chapter ends with advice on how to use roles wisely, having too many, how to name them, and some issues around role dependencies.

Chapter 5: Foundational Elements of Database Application Security

While application security may be built into the various layers of an application in JavaScript, Java Servlets, and Java Authentication and Authorization Service (JAAS), among others, the question is posed as to where is best to put the code for security. We are told that the answer is to enforce security as close to the data as possible. An example is that security code in a web application doesn't help if users can access the database directly using SQL*Plus. Any security must be enforced no matter how the user connects to the database. Keeping the security out of the way of the users is described as security transparency, meaning that users are not aware of how security is being enforced, that it doesn't interfere with their work to the point that they demand it be removed.

As a user connects to an application, your security policies will need to store information about the user, and while a database table is one way to go, we are told about the better, faster, more secure way, which is to use an application context. We see a detailed diagram showing how the application context is stored in memory for both a dedicated and a shared server session mode. The context stores name-value pairs that can be queried and changed by users and applications. This feature was created to speed up fine-grained access control (FGAC), part of the virtual private database. The three types of application context are described: Database, which holds data for each user's database session and has a default namespace of USERENV; Global, which is shared among multiple database sessions; and Client, for OCI context values. Examples of what is stored in the USERENV namespace include client IP address, session ID, and many others. All of these values can be accessed using the SYS_CONTEXT SQL function. I found this interesting: one of the optional parameters to this function is the length value. You use this to truncate a value to the specified length, and this is used to verify the data to prevent an attack based on buffer overflow.

Creating an application context requires the CREATE ANY CONTEXT privilege; when you create a context, you need to assign a namespace manager such as a PL/SQL package, used to set the context values. Many other aspects of managing application contexts and using them to improve security are covered, all of which were new to me. For example, when you have connection pools, you have to set and reset the application context as different end users connect to the database. Since I am not an application developer, I don't see these features often.

Chapter 6: Real Application Security

Real Application Security (RAS) is new feature in 12*c*. I had not seen this before, and it turns out that this feature was created specifically to provide a common, declarative security model that can be shared among enterprise applications. This provides better performance because it avoids the overhead of switching database connections to switch between applications. It also deals with the problem of not having the end-user identity information due to using a shared connection pool. This feature was designed to work with Oracle Fusion Middleware (FMW) security as well as any Java-based database application. This means RAS supports FMW products such as Identity Management (IDM), SOA Suite, WebCenter, and Application Development Framework (ADF).

Real Application Security is based on application users and roles, and a lightweight session model, each of which are discussed in detail in this chapter. There are many new things to learn about this, such as using the XS_PRINCIPAL package to create direct login application user accounts, and several examples of the SQL used to manage these accounts are given.

RAS roles and how they are integrated with standard database roles is shown, and a diagram shows how RAS application users, roles, and standard database roles are interrelated. RAS lightweight sessions are created once a traditional database session has been created, and then used to create multiple long-running application sessions, each with its own set of roles, privileges, and context. This is done to support efficient switching between sessions while maintaining needed end-user and application session information.

This leads to coverage of the Java class oracle.security.xs.XSSessionManager, which is used to manage the lightweight sessions in Java. This is one of several sections that cover the details of RAS. Much of this is not familiar to me, as I usually deal with the database and administering FMW, and not application development. A lot of code is shown, which I assume would be useful to developers. More detail is given for server-side event handling and namespaces, and then session performance is discussed. Code is shown that demonstrates that RAS provides a 60% improvement in the time needed to attach and detach from a session. Here we find that RAS was designed and built by the Oracle Database Security team to support the performance, auditing, and session context needs of Oracle Fusion Applications. This helped me understand that RAS wasn't created for me to use but was created to support the much bigger needs of Fusion Applications. That alone made this chapter worthwhile to me. Sometimes I can't really see why some of features of a new database version are so important, but like RAS, it turns out some features were created to support other areas of the overall Oracle technology stack.

This chapter ends with more details of RAS, including privilege management, data security, security classes, and data security policies. This chapter has a lot of detail that would be useful for application developers.

Chapter 7: Controlled Data Access with Virtual Private Database

The previous chapter advocated the use of RAS, which is new to 12*c*. This chapter is for all those environments that are not on 12*c*—at least not yet—and describes the security framework provided for database 8*i* through 11*g*, which is Virtual Private Database (VPD). First up is an introduction to how VPD works, which explains that VPD is implemented using the Row-Level Security (RLS_ package DBMS_RLS), which restricts which rows or columns SELECT or DML statements can access. The restrictions are based on a security function, which is PL/SQL written based on your security policy. Note that 12*c* has VPD as well as RAS, and that in 12*c* VPD is limited to working on the current database; it can't work across multiple pluggable databases.

VPD works by registering a table or schema with a policy function and the type of SQL statement that will be protected. When the SQL statement is executed, the policy function is evaluated and returns a predicate that is appended to the SQL statement's where clause. Note that the execution of the policy function is done first, before the original SQL statement, and the result used to modify the statement, which is then parsed and executed. For example, a VPD policy could look at a user's ID and limit the rows that user sees by adding to the where clause of the original SQL.

One of the benefits of using VPD is that it puts security very close to the data in the database, not in the application where it could be bypassed by a user that could connect directly to the database. It is important to note that while VPD may sound good and may be used simply to get started, it is easy to get lost in the process of defining your security policy requirements. It can be harder to define your security policies than it is to set up VPD.

The following sections describe the VPD components, the row-level and column-level fine-grained control options, and how to use VPD. The authors assert that many times the VPD security policies that are created are too restrictive. A very simple example that I would not have thought of is presented. If you have a table SALES_HISTORY, and the VPD security policy restricts users to only seeing the rows that represent their own sales, then users can't do a simple count of the total number of sales in the table. This is very good information. It is a simple example, but it really shows how easy it is to set up security policies that no one will be willing to live with.

A flowchart is provided to help assess what kind of VPD policy is best based on what you are trying to protect.

This chapter ends with an interesting comment that for any security feature, performance is always a concern. Since VPD adds to the where clause of each SQL statement, there will be a performance impact. We are told that VPD performance issues become SQL statement tuning issues.

Chapter 8: Essential Elements of Sensitive Data Control

The previous two chapters discussed how to limit a column from appearing in a result set using RAS and VPD, which is called static or full redaction. Here we see that in Oracle 12*c* you can set up partial or dynamic redaction that hides part of a result such as a credit card number using Oracle Data Reduction (DR). I get confused because DR to me will always be Disaster Recovery. This is part of the Advanced Security option and works with 11*g*R2 databases as well. Another feature of 12*c* is Transparent Sensitive Data Protection (TSDP), which can find sensitive data in your database. Part of Oracle Enterprise Manager Cloud Control (OEMCC) called Quality Management Data Discovery (QMDD) will analyze databases for sensitive information. With all these acronyms in place, we look at the challenges of protecting sensitive data: what data is sensitive, where it lives, and how to protect it.

Within TSDP we have new packages, DBMS_TSDP_MANAGE, which we use to define sensitive data by logical name, and DBMS_TSDP_PROTECT to define policies for DR or VPD. An example is shown, with screenshots, of using OEMCC to examine your database for sensitive information. The example is fine, but it assumes that sensitive data is going to be really obvious, like Social Security numbers, for example. If they are stored alone I guess that is okay, but how do we know there isn't sensitive data that would be much harder to define and detect? A lot of data that could be sensitive will not be so obvious. Configuring TSDP to detect sensitive data types is shown, which includes mapping to specific columns. How to create policies and map them to the sensitive data types is next, followed by setting up redaction. This uses the new package DBMS_TSDP_PROTECT, which provides procedures to enable policies to control how sensitive data will be displayed.

This chapter ends with a good example of the impact of all this security stuff, namely the need to redact bind variables in audit trails. I had not thought about it, but the value of a bind variable could be shown in an audit trail and that might need to be redacted. I think this chapter is interesting, as it shows how to set up a needed functionality, but I would like to see how this would be applied to a real enterprise application where the sensitive data is much more complicated than just Social Security numbers. I warned you about lots of acronyms, didn't I? This chapter really delivers on that front!

Chapter 9: Access Controls with Oracle Label Security

Oracle Label Security (OLS) controls row-level access to table data and is built on top of VPD, which has already been covered, and uses the DBMS_RLS package. Can you imagine what the PL/SQL manual for 12*c* must be like? The features that make OLS uniquely different from VPD and RAS are the declarative policies and the lack of technical programming, and it turns out we can use VPD, RAS, and OLS separately or together. A history is given of where all this came from, starting with the need for finer-grained access control than GRANT statements. Oracle consulting came up with Secure Access (SA), and the SA acronym still appears in some of the package names.

A functional example of OLS is given that shows exactly how OLS works using the SALES_HISTORY table. OLS works with the GRANT privilege to control access to each row of a table; you have to have access to the table before you get to the OLS restrictions. Each user has a label and each row in the table has a label, and OLS adds to the WHERE clause of the SQL to control each user's access to the rows in the table.

OLS and VPD are compared to answer the question of which is better and when to use each one. The answer is that OLS has all the features that have been built and tested for a security setup, whereas VPD requires that you set up and maintain your own code and other pieces of the installation. Since OLS has been tested, it may also make it easier to pass certain security accreditations. The OLS label component is shown to have three parts: levels, compartments, and groups, and we see that OLS allows for ten thousand levels for a single OLS policy. I can't really imagine how you would design and maintain such a complex set of policies, but 12*c* will support it. The row labels used by OLS have numeric values as does the user's label. Typically, when the user's label value exceeds that of the row, access is granted. You can also have lots of compartments and groups, and groups can be set up in a hierarchical structure. It is interesting to see how we display the labels using SQL, using the LABEL_TO_CHAR function. The many different types of user labels are shown. This could get very complicated, and I wonder how many customers really set all this up.

The processes of installing OLS and administering it are covered, including screenshots from the Database Configuration Assistant. Registering and enabling OLS has to be done in the root container first and then in the pluggable database, and the SQL for this is shown. You can use the LBACSYS database account for OLS administration, but as with other database features, we are told to lock this account and set up named administrator accounts, which are granted the LBAC_DBA role.

The chapter ends with a detailed example of setting up OLS for the SALES_HISTORY table. When I say it is detailed, I am not exaggerating: it really covers all the steps, to the point that I wonder again who actually does all this for all the tables needed for an enterprise application. The performance impact of OLS is also discussed, and table partitioning is recommended. The authors tell us that OLS and VPD are sometimes described as being "too complicated" to use. I can see why.

Chapter 10: Oracle Database Vault: Securing for the Compliance Regulations, Cybersecurity, and Insider Threats

DBV was introduced in 2005 to offer the features needed by businesses to deal with all the emerging cybersecurity threats. I noticed that the threat from internal employees was brought up, something that I think needs a lot more attention. It doesn't matter how good your security is if your own DBA is selling your data. We are told that DBV changes the way DBAs do their job.

First is a history of privileged accounts, specifically for the Oracle database, starting with the SYS user, which has all access—similar to the root user for Linux. This leads to the DBA role and then the need to audit these powerful users. We need to have these roles, but we need to be able to keep track of what they are doing.

We need SYS to own the core database objects and to do things like apply patches, so we can't remove SYS. But SYS presents a big security risk, so we need to control SYS, and this is one of the things DBV was designed to do. SYS can't get around the security provided by DBV. Security features of DBV are discussed, starting with multifactor authentication, which requires more than just a password to gain access. Conditional security provides context-based security, which can be seen as a conditional grant—access is only granted under specific conditions, such as where a user is connecting from. It also provides adaptive security to respond in near real time to threats, and separation of duty (SoD), so that no single person has all authorizations needed to perform administrative tasks. Conditional auditing provides fine-grained auditing, so that only what we decide needs to be audited is audited. The message here is that with DBV, security is not a rigid set of roles with a defined set of privileges but is much more responsive to the context of the request for access.

The components of DBV are discussed, primarily the declarative framework provided that transparently evaluates all SQL and determines if execution should be allowed. This is based on a set of tables in the DVSYS schema. A web-based interface within OEMCC (remember, that is Oracle Enterprise Manager Cloud Control 12*c*) can be used to set up the security policies for DBV or a PL/SQL package called DBMS_MACADM.

The DBV policies are specific to one container, the root database or one of the pluggable databases. Many more pieces of

DBV are described, factors to be considered when granting access, rules for decision points in the authorization process, realms for collections of objects that need to be secured as a group, command rules, and secure application rules. The process of configuring and enabling DBV is covered, with many SQL examples and the observation that DBV must be configured in the root container first before it can be used in a PDB. Some of the many PL/SQL packages and database views available for DBV administration are covered.

Finally, a detailed example of setting up DBV is presented, which covers a lot of ground. It is clear that implementing DBV is a large project.

Chapter 11: Oracle Transparent Data Encryption: Securing for the Compliance Regulations, Cybersecurity, and Insider Threats

Each of the many security components available in 12*c* provides features to deal with different security issues. Now we cover encryption, which helps with several specific issues. It happens regularly that an employee laptop with sensitive data is lost or stolen, and with it, a large amount of sensitive data. If the hard drive is encrypted, then the risk is reduced. Similarly, the data stored in the Oracle database is at risk anytime one of the hard drives is removed or a backup is made and stored off-site. If the data is encrypted when it is stored in the database, then we prevent access to the data when a hard drive is removed or a backup made.

The history of database encryption offered in the Oracle database is documented, going back to 8*i*. Transparent Data Encryption (TDE) was first seen in 10*g*R2 and was a big change. TDE is much more integrated with the database than previous offerings. Implementing encryption is done within the DDL commands used to define how columns or tablespaces are stored. Once implemented, the word "transparent" means that you don't have to encrypt or decrypt data, it is all done transparently without needing any changes to your applications. This is a big deal because security measures that get in the way are less likely to be used.

This chapter covers the basics of cryptography and some history of its use, going all the way back to the Roman Empire, and the reasons you don't want to set up your own encryption scheme. The things needed to encrypt data, a key and an algorithm, are illustrated. I had not previously heard that longer encryption keys may or may not make the encryption stronger. On television (I watch way too much TV!) they are always impressing us with how many bits are involved in the encryption key, as if that means anything to the viewers. The difference between symmetric and asymmetric encryption is explained, and for both, the issue is getting the keys to the right parties. This leads to a discussion of public key encryption, which does away with the key distribution issue but is slower. Public key algorithms are slower, so they are used to transmit the key needed for faster symmetric encryption that is then used to move the data. I found it interesting how the compromise between security and performance is handled.

A detailed example of encrypting credit card data in the SALES_HISTORY table shows how we could see the sensitive data using the strings command without ever needing to access the database, as well as what the sensitive data looks like after encryption. A great deal more detail about how all this works is covered. Note that TDE is only for data that is "at rest," which means data that is in the database. As data is selected or modified, it is decrypted and manipulated, and only when the data is returned to the database is it again encrypted. We are referred to the "standard network encryption feature" to deal with encrypting data that is moving through the environment. I found this to be one of the best chapters for me from a DBA perspective, because TDE is something I could see being used by many people since it doesn't require application development and can be set up entirely in the database.

Chapter 12: Audit for Accountability

We start with a statement that auditing is not exciting and that we all should do it but rarely do. I agree; I can only think of one workplace where I saw database auditing happening, and the only thing we did with the database auditing was delete it whenever it got close to filling up the assigned tablespace. I also agree with the next statement, that many times we don't know what to do with the audit records. This chapter explains why auditing is necessary and not that hard to do.

We start with the security cycle, which is described as a process that begins with prevention that includes access control; moves on to detection—detecting that however good your access control may be, it isn't perfect; and ends with response—what you do when you detect a security issue. I agree with the authors when they state that many times the focus is all on the prevention, on the thoroughness of the access control, as if that is all that is needed. Assuming no one will ever break in is not a good plan, especially when your own DBA could decide it was time for some off-the-books activity.

Auditing provides the detection phase of the security cycle. You can't have a response unless you detect the event. It is vital that you audit the right data, processes, and users, and you must review the audit records regularly. We are told that we must not look at auditing as overhead, something that negatively impacts performance without any benefit.

Several audit methods are reviewed, starting with two that are outside the database: infrastructure and application server logs, which come from network switches; firewalls and application servers; and application auditing, which is dependent on the auditing that the application developers decided to code into the application. Next are two methods that are inside the database: trigger auditing and database auditing. Trigger auditing is transparent to existing applications, which is one way to add auditing when it wasn't built into the application. Four kinds of database auditing are described: mandatory SYS auditing (MSA), traditional auditing (TA), fine-grained auditing (FGA), and Oracle unified auditing (OUA). Even more acronyms!

The advantages and drawbacks of each of these four options are explained.

The details of setting up database auditing are shown, and like some other 12*c* features we have seen so far, this has to be configured in the root container (CDB), and all the PDBs share the auditing setup. I'm not sure this will really work in an environment with many different databases serving widely different application, user group, and political agendas. The one database that needs the most auditing will force that level of detail on all the other databases, and there could be way too much auditing going on. The chapter also covers the options for setting up separate users to control and conduct auditing, further details on how to determine what to audit, and storage issues that arise from auditing.

Chapter 13: An Applied Approach to Multitenancy and Cloud Security

This final chapter offers to tie together everything that has been discussed in all the previous chapters, to provide a pragmatic approach to setting up a secure multitenant database system. This can also be applied to a traditional, non-multitenant database as well. We are reminded to apply only as much security as needed to meet compliance or regulation standards, and that we need to have multiple layers of security, each one making it less likely that anyone can get all the way through.

This pragmatic approach is broken into four sections: system baseline and configuration, installing and securing the application, data encryption, and locking down your system. Each of these sections has many detailed steps to follow. For example, under the system baseline section, we find steps covering personnel security, configuration management, equipment, virtualization, operating system, users, groups, and several others. This is a very detailed list of what should be done. I would have liked to see some discussion of how this process could be applied to an existing system; I think many of the detailed steps would be much more difficult to implement when many of the configuration decisions have already been made.

Under the section covering system baseline and configuration there is a list of steps to consider for installing 12*c* software. Here we find a good discussion of how to choose how many Oracle Homes you need. I had not thought about this. You can have a single Oracle Home for the 12*c* software for multiple CDBs and all the PDBs that will be contained in them, but you may want multiple Oracle Homes so that not all of the databases are affected by an outage. Patching comes to mind. If you have hundreds of databases, can you have all of them down at once to patch the single shared Oracle Home?

The level of detail included in these sections is impressive. I would recommend that you look at this last chapter if you are in any stage of planning a new installation or a security review for an existing system. I had not worried about the chain of custody for the patches applied to the database, but it is on the list of things to consider.

Another interesting section points out that availability should be part of your security plan because many attacks will simply try to put your online business offline. I was puzzled by the specific advice to have a backup site in Lake Tahoe, California. Seriously? Have you looked at real estate prices there? I don't know of many data centers close to the lake, but maybe that is a sign of just how secure they really are.

Conclusion

I enjoyed reading this book and I learned a lot about 12*c*. I do think this book adds to my feeling that the enterprise software we are all working with is too complex to be effectively supported. The Oracle 12*c* database has many cool features, but who has time to understand, configure, and manage all of them? Who has the personnel resources to set up all the different roles needed for true separation of duty, for example? This is why we still do what is quick and easy. I still access databases using "/ as sysdba," and I see many database users with more privileges than they absolutely need. We do these things because we are pressed for time, and customers do not want their next rollout delayed because an application user doesn't have the needed privileges on a table. I think this is the real security issue for all of our software systems, but no one really wants to talk about it. We don't know how to fix the real issue—that no one has the time and resources to really address security—so we focus on new features and hope for the best.

"We assume that our infrastructure will prevent bad actors from getting into our databases. This is one of the worst mistakes we can make. No matter what we do to prevent it, someone can still get in, and bad things can still happen. Furthermore, the insider threat is always present."

I'm surprised, although maybe I shouldn't be, at how much of this book talks about security features that can only be set up when designing and coding the applications that run against the database. This points out just how hard it is for a DBA to do much about security in real time. If the applications are sending data to anyone that asks for it, what can be done to the existing applications?

I want to be clear that none of what I'm saying is a criticism of the authors in any way. Their job was to communicate the security features of 12*c*, and they have done a great job of that. I'm also pleased that the performance impact of the various security features was explicitly brought up and discussed. Too often, new features are presented as magic cure-alls that have no impact. ▲

Brian Hitchcock works for Oracle Corporation where he has been supporting Fusion Middleware since 2013. Before that, he supported Fusion Applications and the Federal OnDemand group. He was with Sun Microsystems for 15 years (before it was acquired by Oracle Corporation) where he supported Oracle databases and Oracle Applications. His contact information and all his book reviews and presentations are available at **www.brianhitchcock.net/oracle-dbafmw/**. *The statements and opinions expressed here are the author's and do not necessarily represent those of Oracle Corporation.*

Check your member ID at nocoug.wildapricot.org. If it is 6431187, you have won a copy of Oracle Database 12*c* Security. Collect your prize at the fall conference on November 20 at PayPal Town Hall.

New Indexing Features in Oracle Database 12*c*

by Darl Kuhn

Darl Kuhn

This article discusses a couple of Oracle Database 12*c* topics: multiple indexes on the same column combinations and indexing extended columns. When indexing the same column combinations, you must use different physical index types, and only one index can be designated as visible. With indexing extended columns, this requires either using a virtual column and associated index or a function-based index. These features provide you with additional options that can be utilized to enhance performance in new and creative ways.

Multiple Indexes on the Same Column Combinations

Prior to Oracle Database 12*c*, you could not have multiple indexes defined on one table with the exact same combination of columns. For example:

```
SQL> create table t(x int);
Table created.

SQL> create index ti on t(x);
Index created.

SQL> create bitmap index tb on t(x) invisible;
ERROR at line 1:
ORA-01408: such column list already indexed
```

Starting with 12*c*, you can define multiple indexes on the same set of columns. However, you can only do this if the indexes are physically different; for example, when one index is created as a B*Tree index and the second index as a bitmap index. Also, there can be only one visible index for the same combination of columns on a table. Therefore, running the prior CREATE INDEX statements works in an Oracle 12*c* database:

```
SQL> create table t(x int);
Table created.

SQL> create index ti on t(x);
Index created

SQL> create bitmap index tb on t(x) invisible;
Index created
```

Why would you want two indexes defined on the same set of columns? Say you had originally built a data warehouse star schema with all B*Tree indexes on the fact table foreign key columns, and later discover through testing that bitmap indexes will perform better for the types of queries applied to the star schema. Therefore, you want to convert to bitmap indexes as seamlessly as possible. So you first build the bitmap indexes as invisible. Then when you're ready, you can drop the B*Tree indexes and alter the bitmap indexes to be visible.

Extended Datatypes

Before discussing indexing extended columns some explanation is required regarding the nature of an extended datatype. With the advent of Oracle Database 12*c*, the VARCHAR2, NVARCHAR2, and RAW datatypes can now be configured to store up to 32,767 bytes of information (previously, the limit was 4,000 bytes for VARCHAR2 and NVARCHAR2, and 2000 bytes for RAW). Prior to Oracle 12*c*, the maximum length allowed for VARCHAR2 and NVARCHAR2 datatypes was 4,000 bytes, and 2,000 bytes for the RAW datatype. Starting with Oracle 12*c*, these datatypes can be configured to store up to 32,767 bytes.

For reference, listed next are the steps for enabling extended datatypes for a non-container, single instance database. These steps must be performed as SYS:

```
SQL> shutdown immediate;
SQL> startup upgrade;
SQL> alter system set max_string_size=extended;
SQL> @?/rdbms/admin/utl32k.sql
SQL> shutdown immediate;
SQL> startup;
```

Note: Refer to the *Oracle Database Reference* guide for complete details on implementing extended datatypes for all types of databases (single instance, container, RAC, and Data Guard Logical Standby).

Be aware that once you've modified the MAX_STRING_SIZE to EXTENDED, you cannot modify the value back to the default (of STANDARD). It's a one-way change. So if you try this, first ensure this operation is performed on a test database before applying the change to a production database. If you need to switch back, you will have to perform a recovery to a point in time before the change was made—meaning that you'll need RMAN backups (taken prior to the change) or have the flashback database enabled. You can also take a Data Pump export from a database with extended datatypes enabled and import into a database without extended datatypes enabled, with the caveat that any tables with extended columns will fail on the import.

After enabling the extended datatype, you can create a table with an extended column, as follows:

```
SQL> create table t(et varchar2(32727)) tablespace users;

Table created.
```

If you describe the table it will show the large definition:

```
SQL> desc t
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 ET                                     VARCHAR2(32727)
```

You can manipulate the extended VARCHAR2 column via SQL just as you would a nonextended column; for example:

```
SQL> insert into t values(rpad('abc',10000,'abc'));
SQL> select substr(et,9500,10) from t where UPPER(et) like 'ABC%';
```

The extended datatype is internally implemented as a LOB. Assuming that the T table is created in a schema not containing any other objects, you'll get the following when querying USER_OBJECTS:

```
SQL> select object_name, object_type from user_objects;

OBJECT_NAME                   OBJECT_TYPE
----------------------------- ---------------
SYS_LOB0000019479C00001$$     LOB
SYS_IL0000019479C00001$$      INDEX
T                             TABLE
```

You can further verify the LOB segment details by querying USER_LOBS:

```
SQL> select table_name, column_name, segment_name, tablespace_name, in_row
  2     from user_lobs where table_name='T';

TABLE_NAME COLUMN_NAME SEGMENT_NAME                TABLESPACE_NAME IN_
---------- ----------- --------------------------- --------------- ---
T          ET          SYS_LOB0000019479C00001$$   USERS           YES
```

You have no direct control over the LOB associated with the extended column. This means that you cannot manipulate the underlying LOB column with the DBMS_LOB package. Also, the internal LOB associated with the extended datatype column is not visible to you via DBA_TAB_COLUMNS or COL$.

The LOB segment and associated LOB index are always stored in the tablespace of the table that the extended datatype was created in. Following normal LOB storage rules, Oracle stores the first 4,000 bytes inline within the table. Anything greater than 4,000 bytes is stored in the LOB segment. If the tablespace that the LOB is created in is using Automatic Segment Space Management (ASSM) then the LOB is created as a SecureFiles LOB; otherwise it is created as a BasicFiles LOB.

If you have an application that deals with character data greater than 4,000 bytes but less than or equal to 32,727 bytes, then you may want to consider using extended datatypes. Also, if you're migrating from a non-Oracle database (that supports large character columns) to an Oracle database, the extended datatype feature will help make that migration easier, as you can now define large sizes for VARCHAR2, NVARCHAR2, and RAW columns natively in Oracle.

Indexing Extended Columns

Now that we have an understanding of extended datatypes, next we'll demonstrate how to index extended columns. Let's start by creating a table with an extended column and then try to create a regular B*Tree index on that column:

```
SQL> create table t(x varchar2(32767));

Table created.
```

Note: If you attempt to create a table with a VARCHAR2 column greater than 4,000 bytes in a database that hasn't been configured for extended datatypes, Oracle will throw an "ORA-00910: specified length too long for its datatype" message.

Next, we attempt to create an index on the extended column:

```
SQL> create index ti on t(x);
create index ti on t(x)
                   *
ERROR at line 1:
ORA-01450: maximum key length (6398) exceeded
```

An error is thrown because Oracle imposes a maximum length on the index key, which is about three-fourths of the block size (the block size for the database in this example is 8K). Even though there aren't any entries in this index yet, Oracle knows it's possible that index key could be larger than 6,398 bytes for a column that can contain up 32,767 bytes, and therefore won't allow you to create an index in this scenario.

That doesn't mean you can't index extended columns; rather, you have to use techniques that limit the length of the index key to less than 6,398 bytes. With that in mind, a few options become apparent:

- ➤ Create virtual column based on SUBSTR or STANDARD_HASH functions, and then create an index on the virtual column.
- ➤ Create a function-based index using SUBSTR or STANDARD_HASH functions.
- ➤ Create a tablespace based on a larger block size; for example, a 16K block size would allow for index keys the size of approximately 12,000 bytes. Having said that, if you need 12,000 bytes for an index key, then you're probably doing something wrong and need to rethink what you're doing. This method will not be explored in this article.

Let's start by looking at the virtual column solution.

Virtual Column Solution

The idea here is first to create a virtual column applying a SQL function on the extended column that returns a value less than 6,398 bytes. Then that virtual column can be indexed, and this provides a mechanism for better performance when issuing queries against extended columns. An example will demonstrate this. First, create a table with an extended column:

```
SQL> create table t(x varchar2(32767));
Table created.
```

Now insert some test data into the table:

```
SQL> insert into t select to_char(level)|| rpad('abc',10000,'xyz')
  2  from dual connect by level < 1001
  3  union
  4  select to_char(level)
  5  from dual connect by level < 1001;

2000 rows created.
```

Now suppose you know that the first ten characters of the extended column are sufficiently selective to return small portions of the rows in the table. Therefore, you create a virtual column based on a substring of the extended column:

```
SQL> alter table t add (xv as (substr(x,1,10)));
Table altered.
```

Now create an index on the virtual column and gather statistics:

```
SQL> create index te on t(xv);
Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T');
PL/SQL procedure successfully completed.
```

Now when querying the virtual column, the optimizer can take advantage of the index in equality and range predicates in the WHERE clause; for example:

```
SQL> set autotrace traceonly explain
SQL> select count(*) from t where x = '800';
---------------------------------------------------------------
| Id  | Operation                           | Name | Rows  |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |      |     1 |
|   1 |  SORT AGGREGATE                     |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |     1 |
|*  3 |    INDEX RANGE SCAN                 | TE   |     1 |
---------------------------------------------------------------
```

Notice that even though the index is on the virtual column, the optimizer can still use it when querying directly against the extended column X (and not the virtual column XV). The optimizer can also use this type of index in a range-type search:

```
SQL> select count(*) from t where x >'800' and x<'900';
---------------------------------------------------------------
| Id  | Operation                           | Name | Rows  |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |      |     1 |
|   1 |  SORT AGGREGATE                     |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |   239 |
|*  3 |    INDEX RANGE SCAN                 | TE   |   241 |
---------------------------------------------------------------
```

Like the SUBSTR function, you can also base a virtual column on the STANDARD_HASH function. The STANDARD_HASH function can be applied to a long character string and return a fairly unique RAW value much less than 6,398 bytes. Let's look at a couple of examples using a virtual column based on STANDARD_HASH.

Assuming the same table and seed data as used with the prior SUBSTR examples, here we add a virtual column to the table using STANDARD_HASH, create an index, and generate statistics:

```
SQL> alter table t add (xv as (standard_hash(x)));
Table altered.

SQL> create index te on t(xv);
Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T');
PL/SQL procedure successfully completed.
```

The STANDARD_HASH works well when using equality predicates in the WHERE clause. For example:

```
SQL> set autotrace traceonly explain
SQL> select count(*) from t where x='300';
---------------------------------------------------------------
| Id  | Operation                           | Name | Rows  |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |      |     1 |
|   1 |  SORT AGGREGATE                     |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |     1 |
|*  3 |    INDEX RANGE SCAN                 | TE   |     1 |
|--------------------------------------------------------------
```

The index on a STANDARD_HASH–based virtual column allows for efficient equality-based searches but does not work for range-based searches, as the data is stored in an index based on the randomized hash value; for example:

```
SQL> select count(*) from t where x >'800' and x<'900';
-------------------------------------------
| Id  | Operation          | Name | Rows  |
-------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |
|   1 |  SORT AGGREGATE    |      |     1 |
|*  2 |   TABLE ACCESS FULL| T    |   239 |
-------------------------------------------
```

Function-Based Index Solution

The concept here is that you're building an index and applying a function to it in a way that limits the length of the index key and also results in a usable index. Here we use the same code (as in the prior section) to create a table with an extended column and populate it with test data:

```
SQL> create table t(x varchar2(32767));
Table created.

SQL> insert into t
  2  select to_char(level)|| rpad('abc',10000,'xyz')
  3  from dual connect by level < 1001
  4  union
  5  select to_char(level)
  6  from dual connect by level < 1001;

2000 rows created.
```

Now suppose you're familiar with the data and know that the first ten characters of the extended columns are usually sufficient for identifying a row; therefore, you create an index on the substring of the first ten characters and generate statistics for the table:

```
SQL> create index te on t(substr(x,1,10));
Index created.

SQL> exec dbms_stats.gather_table_stats(user,'T');
PL/SQL procedure successfully completed.
```

The optimizer can use an index like this when there are equality and range predicates in the WHERE clause. Some examples will illustrate this:

```
SQL> set autotrace traceonly explain
SQL> select count(*) from t where x = '800';
---------------------------------------------------------------
| Id  | Operation                           | Name | Rows  |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |      |     1 |
|   1 |  SORT AGGREGATE                     |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |     1 |
|*  3 |    INDEX RANGE SCAN                 | TE   |     8 |
---------------------------------------------------------------
```

This example uses a range predicate:

```
SQL> select count(*) from t where x>'200' and x<'400';
---------------------------------------------------------------
| Id  | Operation                           | Name | Rows  |
---------------------------------------------------------------
|   0 | SELECT STATEMENT                    |      |     1 |
|   1 |  SORT AGGREGATE                     |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |   477 |
|*  3 |    INDEX RANGE SCAN                 | TE   |   479 |
---------------------------------------------------------------
```

Assuming the same table and seed data as used with the prior SUBSTR examples, here we add a function-based index using STANDARD_HASH:

```
SQL> create index te on t(standard_hash(x));
```

Now verify that an equality-based search uses the index:

(continued on page 26)

SQL and the Art of Problem Solving

by Iggy Fernandez

Iggy Fernandez

'Tis a lesson you should heed,
If at first you don't succeed,
Try, try again;
Then your courage should appear,
For if you will persevere,
You will conquer, never fear
Try, try again.

—*The Teacher's Manual* by Thomas Palmer, 1840

What would you do if you were lost in a labyrinth of underground caves? In the dark. With no food. Mark Twain tells the story of how Tom Sawyer and Becky Thatcher got separated from their picnic group and got lost in a labyrinth of underground caves.

"The children fastened their eyes upon their bit of candle and watched it melt slowly and pitilessly away; saw the half inch of wick stand alone at last; saw the feeble flame rise and fall, climb the thin column of smoke, linger at its top a moment, and then—the horror of utter darkness reigned!"

"The hours wasted away, and hunger came to torment the captives again. A portion of Tom's half of the cake was left; they divided and ate it. But they seemed hungrier than before. The poor morsel of food only whetted desire."

There's a happy ending of course. If you can't resist a good adventure story, you can read it at **www.cleavebooks.co.uk/grol/twain/tom31.htm**. A couple of chapters later, Tom Sawyer and Huck Finn find something fabulous . . . I'll leave it to you to find out what it was.

But what's that got to do with problem solving? I think that the first rule of problem solving is "If at first you don't succeed, try, try, again." Chris Goerg demonstrated this rule very well in the Fourth International NoCOUG SQL Challenge. His attempts to solve the challenge are on display for all to see at **nocoug.wordpress.com/2015/05/11/fourth-international-nocoug-sql-challenge**. (A more readable summary can be found in the August 2015 issue of the *NoCOUG Journal*.)

Of course, there's more to problem solving than determination. In his 1945 book How to Solve It, one of the 20th century's most notable mathematicians, George Polya, said: "If you cannot solve the proposed problem try to find first some related problem. Could you imagine a more accessible related problem? A more general problem? A more special problem? An analogous problem?"

In other words, convert a problem that you cannot solve into an analogous problem that you can solve.

Consider the following problem: What is the maximum number of positive integers less than or equal to 100 that we can pick such that none of them differ by 9? In the sample tableau shown in Figure 1, I have used pigeons to represent the selected integers and have crossed out the integers that have not been selected.

~~1~~	~~11~~	~~21~~	🐦	~~41~~	~~51~~	~~61~~	~~71~~	🐦	~~91~~
🐦	🐦	~~22~~	🐦	~~42~~	🐦	🐦	~~72~~	~~82~~	🐦
~~3~~	~~13~~	~~23~~	🐦	~~43~~	~~53~~	~~63~~	🐦	~~83~~	~~93~~
🐦	🐦	~~24~~	🐦	~~44~~	🐦	~~64~~	🐦	🐦	~~94~~
~~5~~	~~15~~	~~25~~	🐦	~~45~~	🐦	~~65~~	~~75~~	🐦	~~95~~
🐦	🐦	~~26~~	~~36~~	~~46~~	~~56~~	~~66~~	~~76~~	~~86~~	🐦
~~7~~	~~17~~	🐦	🐦	🐦	🐦	🐦	🐦	~~87~~	~~97~~
🐦	~~18~~	~~28~~	~~38~~	~~48~~	~~58~~	~~68~~	🐦	🐦	🐦
🐦	~~19~~	🐦	🐦	🐦	~~59~~	~~69~~	~~79~~	~~89~~	🐦
🐦	~~20~~	~~30~~	~~40~~	🐦	🐦	🐦	🐦	~~90~~	🐦

Figure 1.

Forty-three integers have been selected in the Figure 1 tableau. We cannot select any more without breaking the rule; for example, we cannot pick the number 1 because the number 10 has already been picked. But the Figure 1 tableau is just a sample—one of many possibilities. We still need to answer the question: What is the maximum number of positive integers less than or equal to 100 that we can pick without breaking the rule that no two of the selected integers differ by 9?

There's a reason I used pigeons to represent selected integers. The problem can be solved easily by converting it into a problem of placing pigeons in pigeonholes. I'm not trying to be facetious. The problem can be solved using the "pigeonhole principle" (the "Schubfachprinzip" or "drawer principle" to fans of mathematical trivia), which simply states that "if n items are put into m containers, with n > m, then at least one container must contain more than one item." Even Homer Simpson would agree.

Each cell in the Figure 1 tableau can be thought of as a pigeonhole in which we can place a pigeon. Therefore the maximum number of pigeons that can be placed in the Figure 1 tableau is 100. But that's not helpful. Instead of the 10x10 Figure 1 tableau, let's use the Figure 2 tableau:

1	10	19	28	37	46	55	64	73	82	91	100
2	11	20	29	38	47	56	65	74	83	92	
3	12	21	30	39	48	57	66	75	84	93	
4	13	22	31	40	49	58	67	76	85	94	
5	14	23	32	41	50	59	68	77	86	95	
6	15	24	33	42	51	60	69	78	87	96	
7	16	25	34	43	52	61	70	79	88	97	
8	17	26	35	44	53	62	71	80	89	98	
9	18	27	36	45	54	63	72	81	90	99	

Figure 2.

There are 54 cells arranged in nine rows in the Figure 2 tableau. With the exception of eight cells in the last column—each cell contains two integers that differ by exactly 9. Therefore, if each cell were a pigeonhole, we could not place more than one pigeon in each cell. Since there are 54 cells (pigeonholes), the pigeonhole principle tells us that if we have 55 pigeons, then at least one cell will be occupied by more than one pigeon. Therefore, we can pick no more than 54 positive integers less than or equal to 100 without breaking the rule that no two of the selected integers differ by 9. The Figure 3 tableau contains the same 43 pigeons as the Figure 1 tableau, but it is much easier to verify. Notice that there are 11 pigeonholes—highlighted in orange—that do not contain a pigeon, because it is not possible to place a pigeon in those pigeonholes without violating the rules of the game; satisfy yourself on this point.

1	🐦	19	28	🐦	46	🐦	64	🐦	82	91	🐦
🐦	11	20	🐦	38	🐦	56	65	🐦	83	🐦	
3	🐦	21	30	🐦	48	🐦	66	75	🐦	93	
🐦	13	22	🐦	40	🐦	58	🐦	76	🐦	94	
5	🐦	23	🐦	41	🐦	59	68	🐦	86	95	
🐦	15	24	🐦	42	51	🐦	69	🐦	87	🐦	
7	🐦	25	🐦	43	🐦	61	🐦	79	🐦	97	
🐦	17	26	🐦	44	53	🐦	71	🐦	89	🐦	
🐦	18	🐦	36	45	🐦	63	72	🐦	90	🐦	

Figure 3.

We have proved that we can select no more than 54 integers less than 100 without breaking the rule that no two of the selected integers differ by 9. But can we really pick that many? A little thought tells us that we can place a pigeon in every pigeonhole as in the Figure 4 tableau.

The preceding example illustrates how converting a problem into another form can make it easier to solve. Chris Goerg used this strategy to great advantage while solving the Fourth International NoCOUG SQL Challenge. The challenge was to determine which of ten possible dates stored in the DateOfBirth table was Cheryl's birthday. Chris neatly recast the problem using the MODEL clause of Oracle SQL.

🐦	10	🐦	28	🐦	46	🐦	64	🐦	82	🐦	100
🐦	11	🐦	29	🐦	47	🐦	65	🐦	83	🐦	
🐦	12	🐦	30	🐦	48	🐦	66	🐦	84	🐦	
🐦	13	🐦	31	🐦	49	🐦	67	🐦	85	🐦	
🐦	14	🐦	32	🐦	50	🐦	68	🐦	86	🐦	
🐦	15	🐦	33	🐦	51	🐦	69	🐦	87	🐦	
🐦	16	🐦	34	🐦	52	🐦	70	🐦	88	🐦	
🐦	17	🐦	35	🐦	53	🐦	71	🐦	89	🐦	
🐦	18	🐦	36	🐦	54	🐦	72	🐦	90	🐦	

Figure 4.

```
with d as (
  select unique
    extract(month from dateofbirth) m,
    extract(day from dateofbirth) d
  from dates
)
select m, d from (
  select * from d
  model
  dimension by (m, d)
  measures(0 s, 0 t)
  rules (

    -- Count the number of times each day is duplicated
    -- Store the count in s

    s[m,d] = sum(1)[m, cv()],

    -- Apply Clue 1a and Clue 1b
    -- Set t to 1 if a date is still in contention

    t[m,d] = case
                when min(s)[cv(), d] > 1 and sum(1)[cv(), d] > 1
             then 1
             end,

    -- Apply Clue 2
    -- Count the number of times each day is duplicated
    -- Store the result in s
    -- Only dates with s = 1 remain in contention after this point

    s[m,d] = case
                when t[cv(), cv()] = 1
             then sum(t)[m, cv()]
             end,

    -- Apply Clue 3
    -- Count the number of dates still in contention in each month
    -- Store the result in t
    -- Only dates with s = 1 and t = 1 remain in contention after this point

    t[m,d] = sum(case when s = 1 then 1 end)[cv(), d]
  )
)
where s = 1
and t = 1;
```

The secret of Chris's solution was his use of two measures (s and t), not just one. Both measures are initialized with the value 0. The progress of his solution can be visualized with a little PIVOT magic:

```
column m format 999
column 14 format a4
column 15 format a4
column 16 format a4
column 17 format a4
column 18 format a4
column 19 format a4

select * from (
with d as (
  select unique
    extract(month from dateofbirth) m,
    extract(day from dateofbirth) d
  from dates
)
select * from (
  select * from d
  model
  dimension by (m, d)
  measures (0 s, 0 t)
  rules (
    -- Add rules here
  )
)
)
pivot (
  min(
    case
      when s is null and t is null
      then ''
      else nvl(to_char(s), '-') || '|' || nvl(to_char(t), '-')
    end
  )
  for d in (14,15,16,17,18,19)
)
order by m;
```

"If you cannot solve the proposed problem try to find first some related problem. Could you imagine a more accessible related problem? A more general problem? A more special problem? An analogous problem?"

This produces the following output when the rules section is empty.

```
   M 14   15   16   17   18   19
---- ---- ---- ---- ---- ---- ----
   5      0|0  0|0            0|0
   6                0|0  0|0
   7 0|0       0|0
   8 0|0  0|0       0|0
```

Bravo, Chris! You can find all the details of his solution in the August 2015 issue of the *NoCOUG Journal*.

P.S. In the Figure 3 tableau, we cannot place another pigeon without breaking the rules of the game. What is the minimum number of positive integers less than or equal to 100 that we can pick such that none of them differ by 9 and such that it is not possible to pick even one more positive integer less than or equal to 100 that does not differ by 9 from the already selected integers? ▲

Mark your calendars for the YesSQL Summit and Practical Tuning Workshop with Carlos Sierra and Mauro Pagano on January 26–28 at the Oracle conference center.

TRAINING WHEN YOU WANT IT

TRAINING MEMBERSHIP INCLUDES

- Online training by Craig Shallahamer
- How-to webinars
- Priority response
- Mastering The Material email follow-up
- 24/7 unlimited access
- Discounts

ORAPUB

Find out more at orapub.com.
Questions? Contact support@orapub.com.

Packaged Analytics—The Easy Button

by Shyam Varan Nath

Shyam Varan Nath

Most businesses deploy Enterprise Resource Planning (ERP), Human Capital Management (HCM), and Customer Relationship Management (CRM) systems to effectively run the business. The HRMS systems like PeopleSoft or EBS HR are great at capturing information but often lack querying and reporting capabilities required to support decision making. For example, an HRMS system may not have an easy way to tell you which employees are likely to leave in the next 60 days, which is critical for employee retention (decision making) and intervention.

HR analytics solutions could be built in two different ways:

- Custom ground-up solutions using Oracle Business Intelligence Enterprise Edition (OBIEE) and a data integration tool like Oracle Data Integrator (ODI)
- Packaged human resource analytics as the starting point for using Oracle Business Intelligence Applications (OBIA).

This article will focus on quickly solving the enterprise reporting and business analytics using OBIEE or OBIA in the HR functional area.

Technical Discussions and Examples

The business users from HR are used to managing the lifecycle of the employee, often referred to as the "Hire-to-Retire" process. While talking to the business users, it is important to understand their business processes and then design/implement the solution, rather than to force-fit a technical solution.

By design, the operational systems such as ERP and HCM—often referred to as "OLTP systems"—are optimized for inserting, updating, and sometimes deleting of transactions. To optimize the storage space usage, the OLTP systems use a third normal data model for the underlying tables. When such a data model is used for query purposes, it often creates complex SQL due to joins and impacts the performance of both the OLTP operations and reporting speed. As a result, most business intelligence (BI) and enterprise reporting systems are based on a data model that is optimized for query and decision support. This often includes pulling the data from the multiple OLTP systems and transforming it to store in a data warehouse (DW). The DW, along with specialized areas called data marts (DMs), are the basis of BI and analysis by the end users. The HRA-related data marts would be Workforce Profile, Payroll, Benefits, and so on.

The Data Marts are characterized by a star schema. Figure 1 shows a typical "analysis" scenario. In this case Sales is a measurable item in terms of product and month (time). This leads to the star schema where Sales is in the center (fact) surrounded by the dimensions. Such a structure allows the user to query the Sales table in terms of any of the dimensions, needing only a single join. Figure 2 shows a star schema from Core Human Resource: Employee Workforce Fact.

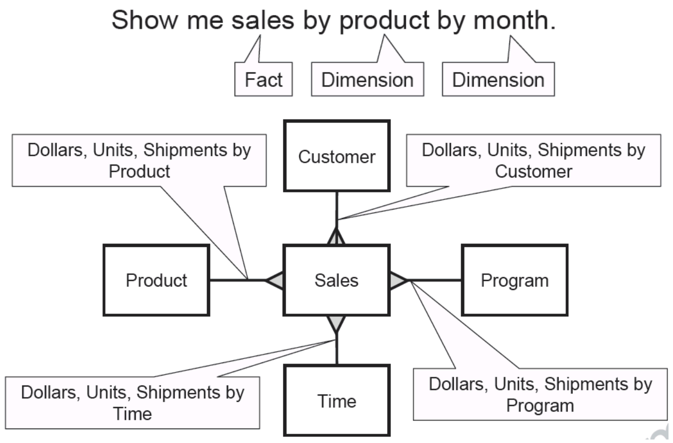

Show me sales by product by month.
Fact
Dimension
Dimension
Dollars, Units, Shipments by Product
Customer
Dollars, Units, Shipments by Customer
Product
Sales
Program
Dollars, Units, Shipments by Time
Time
Dollars, Units, Shipments by Program

Figure 1.

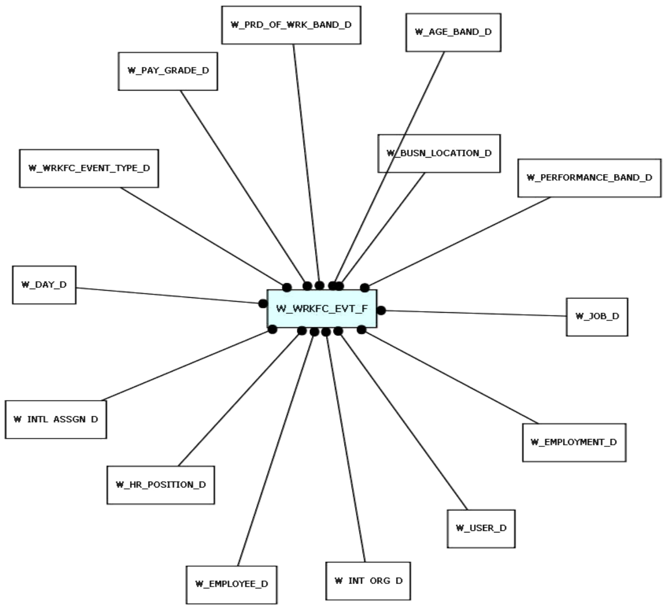


Figure 2.

Once the DW architect has identified all of the possible dimensions, working with the subject matter expert, the star schema gives the end user the flexibility to query the fact table by any dimension using a BI tool's ad hoc query capability. OBIEE can function as such a BI tool. This is a fundamental shift over the traditional reporting paradigm, where most of the business query scenarios have to be flushed out up front to create all those reports. Any new report needs an IT intervention.

Packaged Analytics

Now that we understand how to structure the DW and DM to solve common business analytics problems, we will look at ways to speed up this process and reduce the risks due to unknown factors.

Let us assume that an Oracle shop uses Oracle E-Business Suite (EBS) to run its core operations. Most companies, when implementing an ERP system like EBS, make use of industry best practices and configure it using the ERP's capabilities and flexibilities. Typically, as a good practice, companies do not make massive changes to the underlying table structures and data model; rather, they use the flexible fields to capture the business nuances. With the assumption that 90 to 95% of the table structure of the EBS system will be as shipped, how can we accelerate the development of the DW system? If we build the data integration with the process of extraction, transformation, and loading (ETL), it may take a long time and be risk prone. Systems like EBS contain hundreds—often thousands—of tables in each major functional area. While that option exists, a quicker way is to capitalize on the fact that if 90 to 95% of the table structure of EBS is "as shipped," why not rely on the "as shipped" ETL process to build the DW as the starting point?

This is the approach of the packaged analytics in the form of OBIA. Given the known OLTP source structures, users can implement the known ETL processes to build the DW that answers 80% of the questions that business users tend to ask. Such questions may include total sales by product or by region over a given quarter.

OBIA-HR is a packaged analytics solution that has functional areas like Core HR, Payroll, Absence Management, Compliance and Regulatory Reporting, and so on. The packaged analytics system may look like the one shown in Figure 3. The major components are

➤ Pre-built DW or Oracle Business Analytics Warehouse (OBAW)

➤ Data integration using Data Warehouse Admin Console (DAC) and Informatica

➤ OBIEE to provide the reporting, dashboarding, and ad hoc querying capability based on the metadata

Such an OBIA system, once installed and configured for the given EBS modules and populated with data from the OLTP system, is capable of handling several common business questions, as seen in the interactive dashboard in Figure 4. The end user no longer has to define SQL queries or provide the join criteria between the OLTP tables. This does represent the ideal scenario, but there is a structured methodology to capture the five to ten deviations from the out-of-the-box (OOTB) scenarios of a source system such as EBS. The data integration layer (ETL) can be customized to account for the company-specific changes during the implementation of the EBS. Assuming that the standard process was followed for customization and was documented, the ETL changes can be done in a manner that accounts for those changes. Again, the OBAW's data model is expected to meet 90 to 95% of the data element requirements, and can be extended with new columns and tables as needed.

These changes need corresponding changes to the OBIEE metadata and the reports and dashboards. The degrees of changes in the reports and dashboards will realistically exceed 5 to 10% and often be as much as 30 to 50%. However, the changes to the OBIEE layer are usually faster to do and cost less time and money.

Since the business users usually interact only with the reports and dashboards, it is important to capture their need for reports and dashboards, and to carry them. This can be an ongoing process and often fits the Agile development methodology. Due to access to ad hoc querying, the business users may be able to fetch a lot of data needed for decision making, even while the dashboard and reports are perfected. In fact, the advanced users with ad hoc access often support the process of developing the reports and dashboards on an ongoing basis.

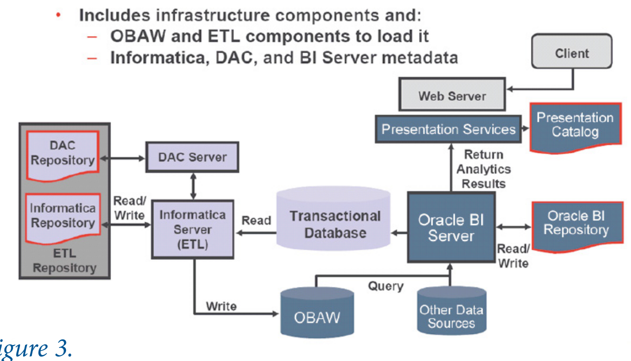


Figure 3.

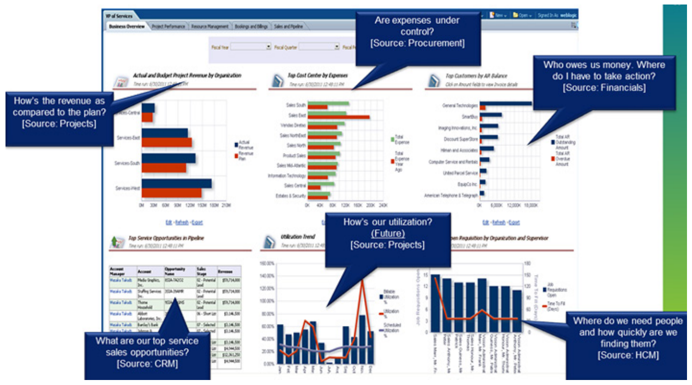

Are expenses under control? [Source: Procurement]
Who owes us money. Where do I have to take action? [Source: Financials]
How's the revenue as compared to the plan? [Source: Projects]
How's our utilization? (Future) [Source: Projects]
What are our top service sales opportunities? [Source: CRM]
Where do we need people and how quickly are we finding them? [Source: HCM]

Figure 4.

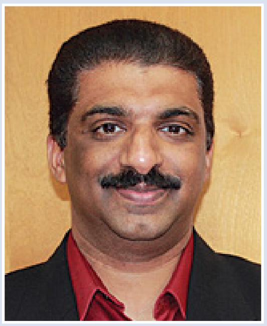
Summary

We looked at the process of solving enterprise reporting for the human resource domain using the BI and DW framework. We looked at how to accelerate the process by leveraging packaged analytics such as OBIA. ▲

Shyam Varan Nath is an IoT and big data analytics architect with General Electric. He has 24 years of industry experience, with a focus on the industrial Internet, advanced analytics, and large data-rich environments. He has implemented large data warehouse, analytics, and IoT solutions. Prior to GE, Shyam worked for IBM, Deloitte, Oracle, and Halliburton. He has an MBA and MS in computer science from FAU, Boca Raton, FL, and a B. Tech. from IIT Kanpur, India. Email: **ShyamVaran@gmail.com***.*

Many Things Oracle

by Biju Thomas


Biju Thomas

Editor's Note: Biju publishes daily Oracle tidbits on Facebook (fb/oraclenotes) and on Twitter (@biju_thomas).

Listener Registration

The LREG process notifies the listeners about instances, services, handlers, and endpoints. In earlier releases, PMON had this responsibility. When the database instance starts up, the LREG process starts up automatically and is a mandatory process. LREG scans port 1521 looking for a listener to register the instance with. If LOCAL_LISTENER and REMOTE_LISTENER parameters are defined, instance is registered with those listeners instead of polling port 1521. LREG communicates the service names and instance load information to the listener. If the listener was not running when the instance started, LREG polls every 60 seconds to check the availability of the listener process.

Log Writer Slave

On multiprocessor systems, LGWR creates LGnn slave processes to improve the performance of writing to the redo log. LGWR slaves are not used when there is a SYNC standby destination. There are a few bugs raised with multiple log writer processes, causing hang issues with the database. The workaround is to disable the multiple log writer processes with the parameter _use_single_log_writer=TRUE. See MOS note 1968563.1 and its reference documents.

SGA Allocator

A small fraction of the SGA is allocated during instance startup. The SAnn process allocates the rest of SGA in small chunks. The process exits upon completion of SGA allocation. This process was introduced to help huge SGA allocations. When the in-memory database feature is in use, we can expect instances with large SGA sizes.

More DB Writers

There can be up to 100 database writer processes (DBWn—configured using DB_WRITER_PROCESSES parameter). The names of the first 36 database writer processes are DBW0-DBW9 and DBWa-DBWz. The names of the 37th through 100th database writer processes are BW36-BW99. Oracle 11*g* supported up to 36 DBWn. It looks like the systems are getting bigger and bigger with database consolidation, and more DB writer processes are required.

RAT Masking Slave

The RAT Masking (RM) Slave Process is used with Data Masking and Real Application Testing. The RM process extracts and masks bind values from workloads like SQL tuning sets and DB Replay capture files.

Transport Monitor

The Transport Monitor (TMON) process monitors the redo transport processes for hangs and death. TTnn processes are redo transport slave processes. The TTnn processes ship redo from current online and standby redo logs to remote standby destinations configured for ASYNC transport.

ASM Related Processes

The Rolling Migration Monitor Process manages the rolling migration procedure for an Oracle ASM cluster. The remote monitoring (RMON) process is spawned on demand to run the protocol for transitioning an ASM cluster in and out of rolling migration mode.

ARSn (ASM Recovery Slave) processes are spawned by the ASM RBAL process to recover aborted ASM transactional operations.

The ASM RBAL background process coordinates and spawns one or more ASM Recovery Slave Processes to recover aborted ASM transactional operations.

The ASM Disk Scrubbing Master Process SCRB runs in an Oracle ASM instance and coordinates Oracle ASM disk scrubbing operations. Related processes are ASM Disk Scrubbing Slave Check Process (SCCn), ASM Disk Scrubbing Slave Repair Process (SCRn), and ASM Disk Scrubbing Slave Verify Process (SCVN)

When ASM disk groups are created with normal or high redundancy mirroring, there is more than one copy of an allocation unit (AU) on the disks. Sometimes logical corruption happens and the AU and its copies get out of sync, resulting in a higher response time from the disk affected by the corruption. The ALTER DISKGROUP SCRUB command is available in Oracle Database 12*c* to report and fix the logical corruption by reading the data from the mirror copies. If you do not fix the logical corruption, the response time will be higher for those corrupted AUs. The REPAIR option automatically repairs disk corruptions. If the REPAIR option is not specified, then the SCRUB option only checks and reports logical corruptions of the specified target.

SQL Developer

SQL Developer can format a .sql file or all .sql files in a directory and use the format.sh or format.bat on the operating system command line.

Keyboard shortcut CTRL+F7 formats SQL code in the SQL Developer worksheet. The SQL Developer distribution ships with format.sh executable to format a number of .sql files in a directory.

Using SQL Developer you can connect to schemas for non-Oracle databases, such as MySQL, Microsoft SQL Server, Sybase Adaptive Server, Microsoft Access, and IBM DB2, and view metadata and data in these databases; and you can migrate these non-Oracle databases to Oracle.

SQL Developer provides read-only access to non-Oracle databases. This read-only access enables SQL Developer to migrate non-Oracle database data to Oracle databases.

Oracle SQL Developer provides an interface to Oracle Application Express, allowing you to browse, monitor, and manage your database applications, and perform the following Oracle Application Express tasks: browse your Application Express applications, export and import applications, drop applications, deploy applications, modify applications, export pages, tune your queries, generate Application Express reports and generate exception reports.

Using SQL Developer you can display SQL Trace output files in a formatted way similar to TKPROF. You can examine the information in the List View, Statistics View, and History panes; each pane includes options for filtering and controlling the display.

To open a .trc file in SQL Developer and see a formatted display of the information, click File > Open, and specify the file; or drag the file's name or icon into the SQL Developer window. You can then examine the information in the List View, Statistics View, and History panes, with each pane including options for filtering and controlling the display.

Many SQL*Plus commands are supported on SQL Developer. The SQL*Plus buffer editor commands are very specific to SQL*Plus and hence not applicable to SQL Developer. SQLDeveloper SQL Worksheet does not support a few SQL*Plus commands, such as archive, copy, oradebug, password, recover, run, startup, and shutdown.

Finally, the SQL Developer command interface is a cool tool if you use SQL Plus more often. SDSQL is a separately downloadable executable available on the OTN download site under SQL Developer. The SDSQL tool includes command history, command completion (by pressing the tab key to complete table names in a query), creating alias names for SQL queries, and the sqlformat command to change the output format (like creating a CSV output). ▲

Biju Thomas is an Oracle ACE, Oracle Certified Professional, and Certified Oracle Database SQL Expert. He is a principal solutions architect at OneNeck IT Solutions, with more than 20 years of Oracle DBA experience. He is the author of Oracle 12c and 11g OCA, and co-author of Oracle 10g, 9i, and 8i OCP certification guides published by Sybex/Wiley. He is a frequent presenter at Oracle conferences and publishes articles for technical journals. Twitter @biju_thomas. Blog **www.bijoos.com**.

HGST
a Western Digital company

Accelerate with PCIe
Server-side Flash Storage

Shared PCIe Flash Pool
managed by
ORACLE
ASM

- HGST FlashMAX® II PCIe cards inside servers used as primary storage with shared access across the cluster
- Oracle ASM performs volume management and data mirroring
- 0.5TB to 72TB of flash in a standard x86 server
- Consistently high performance with linear scalability
- Distributed architecture with no single point of failure
- At a fraction of the SAN cost

DELPHIX®

Database Virtualization Software

▸ Consolidate Infrastructure.
▸ Instantly Provision and Refresh.
▸ Maximize Performance.

www.delphix.com

NoCOUG Fall Conference

Session Descriptions

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

–Fireside AB–

Data-Centric Security: The Key to Digital Business Success

Ulf Mattsson, CTO, Protegrity 9:30–10:30

With the exponential growth of data generation, sharing, and collection stemming from new digital business models fueled by Big Data, cloud computing, and the Internet of Things, we are setting the stage for an almost limitless ability to identify trends, satisfy customer needs, and improve business performance. We are also creating a cybercriminal's paradise, as the old perimeter-based model of data security—based on finding malware and data leaks—has become woefully inadequate at protecting all possible data access points and data stores. New security approaches assume that you are under attack and focus instead on protecting the data itself, even in computer memory—the target for a growing number of attacks. In this session, Protegrity CTO Ulf Mattsson will provide insight to address the following issues:

- ➤ Taking a data-centric approach to balancing the often-competing goals of improving data security while maximizing data value and productivity
- ➤ Managing today's threats while adapting to the ever-changing data security landscape of the future
- ➤ Bridging the gap between security regulations, privacy, and compliance while still being able to provide powerful analysis and data insight
- ➤ Selecting the best methods of protecting sensitive data at rest, in transit, and in use
- ➤ Understanding why tokenization—the most effective form of PCI data security—is increasingly being used to protect PII and PHI as well

–Fireside AB–

Oracle Database 12.2 Active Data Guard New Features

Mahesh Girkar, Oracle 11:00–12:00

Data Guard is Oracle's solution for real-time protection and availability of your Oracle data. Oracle Active Data Guard enhanced it by providing the ability for any standby database to perform live reporting while extending protection guarantees. In this session, you will learn in depth about new enhancements in Active Data Guard that are part of Oracle's upcoming 12.2 RDBMS software release in various areas such as redo transport and apply, standby queries with in-memory performance, and new corruption detection and repair functionality.

Oracle Database 12.2 Sharding for Scalability and Fault Isolation

Srinagesh Battula, Oracle 1:00–2:00

Oracle Sharding is an architectural pattern where data is horizontally partitioned across up to 1000 discrete DBs that do not share hardware and software. It provides massive scalability and fault isolation compared to what is possible by scaling-out or scaling-up a single database. This session covers how the next release of Oracle database automates the deployment of sharded databases with various partitioning schemes, enables elastic scaling and rebalancing, and supports data-dependent routing and cross-shard queries while rendering strict consistency, full power of SQL, developer agility with JSON, and the proven enterprise qualities of Oracle Database EE. Learn how this next-generation Oracle technology provides a complete platform needed for a sharded database.

Keeping Everybody Happy in a Data Warehouse

Yasin Baskan, Oracle 2:30–3:30

Large database operations perform best when executed in parallel. But what about all the other tactical users on your system? How can you guarantee the service levels of such an environment? If you ever wondered how to run different types of highly diversified analytical workloads optimally, or if you are a DBA managing such a system, this session is for you. In this session, learn about managing concurrent workloads—parallel or serial, small or large user populations, strategic or tactical—with the help of technologies like database resource management or fundamental system configuration. Learn how to configure, manage, and monitor your database so that each of your workloads achieves the performance it needs.

–Fireside CD–

Advanced GoldenGate Configuration

Lorrie Yang, Bank of America 11:00–12:00

Since Oracle's purchase of GoldenGate software in 2009, GoldenGate has gained a lot of popularity as the leading choice for database replication when Oracle Database is involved. This presentation—learned from experience—will focus on advanced topics such as conflict detection and resolution setup for multi-master replication and its impact on application design and test platforms.

Faster Database Upgrades

Paresh Patel and Samrat Roy, PayPal 1:00–2:00

Come to this session to understand how database upgrades can be performed with less than one minute downtime and how to reorganize very large databases faster. This session will cover data copy methods, GoldenGate configuration, data validation, SQL execution plan analysis, and database upgrade optimizations.

(continued on page 26)

Many Thanks to Our Sponsors

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

Long-term event sponsorship:

Chevron

Oracle Corp.

Thank you! Gold Vendors:

- Axxana
- Database Specialists
- Dell Software
- Delphix
- EMC
- HGST
- SolarWinds

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

TREASURER'S REPORT

Beginning Balance		
July 1, 2015		**$ 61,154.83**
Revenue		
Individual Membership	1,235.00	
Gold Vendor Fees	2,000.00	
Corporate Membership	1,650.00	
Silver Vendor Fees	1,000.00	
Conference Sponsorships	0.00	
Conference Walk-in Fees	350.00	
Training Day Receipts	0.00	
Journal Advertising	500.00	
Interest	1.45	
Total Revenue		**$ 6,736.45**
Expenses		
Conference Expenses	7,758.63	
Journal Expenses	4,205.63	
Training Day Expenses	0.00	
Board Expenses	129.45	
PayPal Expenses	860.96	
Software Dues	2,160.00	
Insurance	0.00	
Office Expenses	0.00	
Meetup Expenses	0.00	
Taxes and Filings	1,229.90	
Marketing Expenses	0.00	
Total Expenses		**$ 15,687.39**
Ending Balance		
September 30, 2015		**$ 52,203.89**

OLTP Meets Big Data: Challenges, Options, and Future

Saibabu Devabhaktuni, PayPal 2:30–3:30

New database management challenges have emerged as analytics and real-time reporting requirements now have to be handled by OLTP databases. Attend this session to understand how OLTP database technology is evolving to handle the three V's of Big Data: Volume, Velocity, and Variety.

–Town Square A–

Evidence-Based Decision Making: Using AWR and Statspack to Decide If Flash Is Right for You

Mike Ault, IBM. 11:00–12:00

Oracle is a complex I/O-driven database. In this presentation Mike Ault shows how I/O is critical to database performance, how FlashSystem technology with microlatency solves many I/O issues, and how to evaluate your AWR reports to determine if Flash is right for your environment.

Bulletproof Disaster Recovery: Zero Data Loss and Cross-Application Consistency

Eli Efrat, CEO, Axxana 1:00–2:00

The combined solution of Oracle Data Guard and Axxana Phoenix System for Oracle allows recovery of all database transactions to a remote site in a cost-effective manner with high performance and consistency across all applications, regardless of the nature of the disaster and the distance between the sites. This session will cover the following:

- ➤ Understanding the combined solution of Oracle Data Guard and Axxana Phoenix System for Oracle
- ➤ Sizing your DR environment with the combined solution
- ➤ Using Far Sync with the Axxana Phoenix System

All-Flash Storage Arrays for Mixed Workload Consolidation and Copy Data Management

Kevin Closson, EMC. 2:30–3:30

You've heard all about Flash for performance, now take it to the next level with workflow consolidation, integrated copy data management, and self-service. The holy grail would be simple storage consolidation for all databases, business and analytics applications, and nonproduction lifecycle copies. Come learn about scale-out all-Flash arrays and in-memory integrated Copy Data Management, including:

- ➤ Third-party lab testing results for consistent sub-millisecond response times and linear IOPS scaling for massive consolidation of multiple tier-1 workloads and nonproduction copies
- ➤ Integrated Copy Data Management workflow automation for test/dev, data protection, and on-demand analytics
- ➤ Self-service options through AppSync and Oracle Enterprise Manager
- ➤ Using SLOB for testing consolidation opportunities

DATABASE RECOVERY HAS NEVER BEEN SO

SIMPLE!

Axxana's award winning **Phoenix System** offering unprecedented Data Protection, Cross-application consistency, Storage and Replication agnostic.

ORACLE PARTNER NETWORK

AXXANA BUILT TO LAST

info@axxana.com ▪ www.axxana.com

(continued from page 16)

```
SQL> set autotrace traceonly explain
SQL> select count(*) from t where x = '800';
-------------------------------------------------------------
| Id  | Operation                          | Name | Rows  |
-------------------------------------------------------------
|   0 | SELECT STATEMENT                   |      |     1 |
|   1 |  SORT AGGREGATE                    |      |     1 |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| T    |     1 |
|*  3 |    INDEX RANGE SCAN                | TE   |     8 |
-------------------------------------------------------------
```

This allows for efficient equality-based searches but does not work for range-based searches, as the data is stored in an index based on the randomized hash value.

Acknowledgements

This article was derived from material contained in Expert Oracle Database Architecture, 3rd edition, Apress, authored by Tom Kyte and Darl Kuhn. ▲

Darl is currently a DBA/developer working for Oracle. He has written books on a variety of IT topics, including SQL, Performance Tuning, Oracle Internals, Linux, Backup and Recovery, RMAN, and Database Administration. Darl's most recent book is Linux and Solaris Recipes for Oracle DBAs *from Apress.*

Database Specialists: DBA Pro Service

DBA PRO BENEFITS

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

CUSTOMIZABLE SERVICE PLANS FOR ORACLE SYSTEMS

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

DBA Pro's mix and match service components

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

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

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

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

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

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

DatabaseSpecialists

C A L L 1 - 8 8 8 - 6 4 8 - 0 5 0 0 T O D I S C U S S A S E R V I C E P L A N

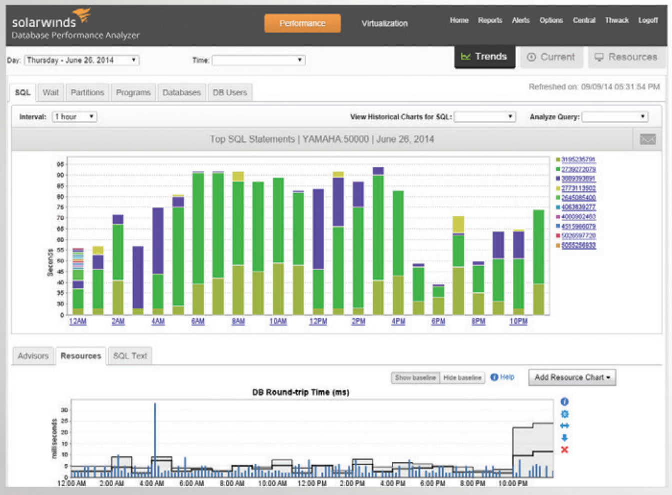
NoCOUG
P.O. Box 3282
Danville, CA 94526

RETURN SERVICE REQUESTED

solarwinds

See what's new in

Database Performance Analyzer 9.0



- Storage I/O analysis for better understanding of storage performance
- Resource metric baselines to identify normal operating thresholds
- Resource Alerts for full-alert coverage
- SQL statement analysis with expert tuning advice

Download free trial at: solarwinds.com/dpa-download