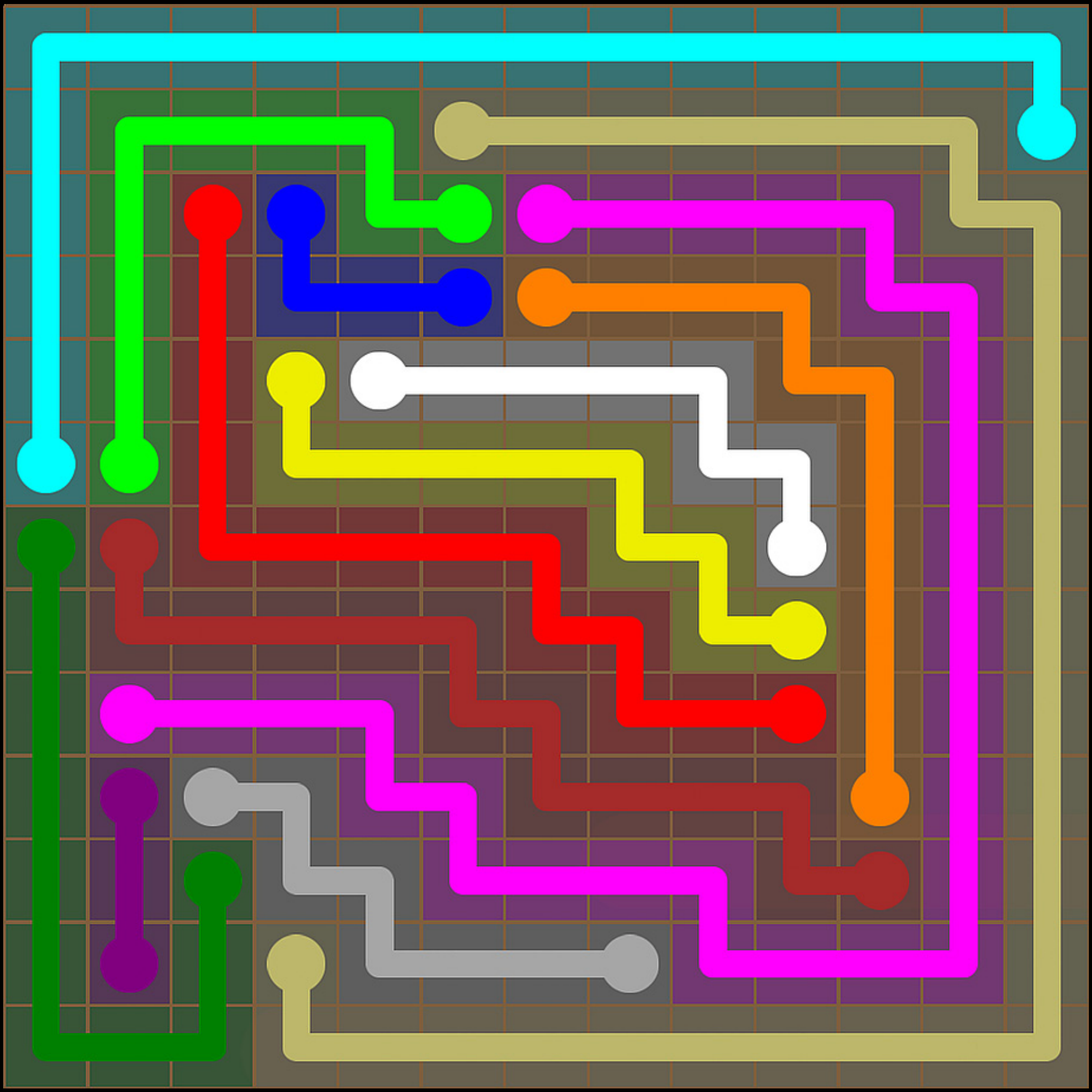
NORTHERN CALIFORNIA NoCOUG ORACLE USERS GROUP

Official Publication of the Northern California Oracle Users Group

# NoCOUG JOURNAL

Vol. 29, No. 2 · MAY 2015

$15



**Women in Technology**

*With Gwen Shapira.*

*See page 4.*

**Tuning Oracle SQL with SQLTXPLAIN**

*Book review.*

*See page 6.*

**Fastest is Not Always Shortest**

*Unconventional wisdom.*

*See page 19.*

***Much more inside . . .***

ce Antoinette Meredith Daisy Nada Liz Amie Jeannine Johr
Robbi Kathyrn Steven Gisele Tim Laura Myron Elizbeth
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 Kareen Tom
erri Sab en An enry Bev
holle C Kareem Er Na
Theron Stev Gisel
y Elizbeth Jeanni rice Antoinette M
ie Jeannine Theron elina Robbi Kathyr
inette Meredith Da nnine Theron Johr
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, Jo Dziubek at Andover Printing Services deftly brings the *Journal* to life on an HP Indigo digital printer.

This is the 114th 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.

Copyright © by the Northern California Oracle Users Group except where otherwise indicated.

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

## ADVERTISING RATES

The *NoCOUG Journal* is published quarterly.

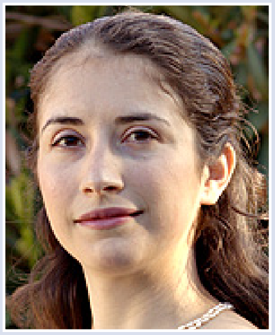
| Size | Per Issue | Per Year |
|---|---|---|
| Quarter Page | $125 | $400 |
| Half Page | $250 | $800 |
| Full Page | $500 | $1,600 |
| Inside Cover | $750 | $2,400 |

Personnel recruitment ads are not accepted.

**journal@nocoug.org**

# Women in Technology

**with Gwen Shapira**


*Gwen Shapira*

***How did you get to be a Big Data expert? What were you doing before that? What did you study in school? And, if we may ask, where did you go to school? You don't have a Californian accent.***

Very long story with a lot of fortunate coincidences.

I'll start at the end. I got my current job at Cloudera through Twitter.

At the time, I was working with a retailer in Japan, as a consultant through Pythian. I was there to help them migrate to Exadata, but their data warehouse system was tightly integrated with Hadoop, and I got to work on this integration too. I got a lot of exposure to Cloudera's distribution—indispensible tools for integrating Hadoop with data warehouses such as Sqoop, Avro, Hive, and Impala. Many of those tools were indispensible but not 100% mature at the time, so I spent a lot of time troubleshooting; since I was alone in Tokyo, I vented on Twitter. I got into interesting and quite useful discussions with a bunch of Clouderans, and eventually Cloudera CEO Mike Olson messaged me and asked to meet when I get back.

We met, we chatted, I talked to his team, and few months later, I decided to leave Pythian and my beloved Exadatas and go help Cloudera customers make use of new types of data and new data systems. To this day, I'm not sure if I was hired because I'm a good consultant or because they wanted the complaints on Twitter to stop.

My adventures with data started a lot earlier. I got my college degree in computer science and statistics, so you can say I was fascinated by data analysis from pretty early in my career. I started working at Mercury Interactive as a developer and later got promoted to a team lead, but I was not very happy leading a team and looked for a way out. When our star Oracle DBA left, I convinced my managers to let me take the DBA role.

The DBA who left was Hanan Hit, of NoCOUG board fame. He continued being a great friend and mentor throughout my career.

A few years later, HP acquired Mercury and I got a chance to relocate to the U.S. HP is where I got started at blogging and later presenting—the job was not very challenging so I was left with plenty of extra time to learn and to write. As a junior DBA, very few people in HP listened to my ideas—I was very surprised to find out that outside of HP people were reading my thoughts, responding, and even inviting me to present!

Presenting at conferences helped me network with a large number of truly brilliant people. One of them is Alex Gorbachev, who offered me a job at Pythian. Pythian is well known for being a place that hires and supports the best DBAs. I jumped at the opportunity, knowing that I would no longer be bored with my day job. That was certainly true. Pythian helped me learn and grow as a DBA and later as a consultant. Both skills were incredibly useful in my big data career.

***In your career thus far, did you find any doors closed or less welcoming to women? Have you had to fight to get the pay or the promotions you deserve? What is your opinion about new Microsoft CEO Satya Nadella's advice to women in response to interviewer Maria Klawe's question?[1] Satya hasn't had the experiences that women have had, but his response was probably colored by his own experience as an Indian immigrant; eventually the system recognized his potential and promoted him. Do you have different advice for women in technology?***

This is a very difficult question to answer. We all have career opportunities and career setbacks. Who can tell what were the influencing factors? It can be my gender, my accent, my personality, or the culture. It is impossible to tell in many cases. I am certain that if you ask yourself "Do I deserve this?" at every turn, good or bad, it is futile. As an engineer in the S.F. Bay Area, I easily earn a very comfortable salary. Do I deserve to earn this much when some people work much harder and can barely feed their children? It is really difficult to know. Life is inherently very unfair to everyone.

[1] Klawe asked: "for women who aren't comfortable with asking for a raise or—sort of saying—who aren't the younger you, let's say, what's your advice for them?" Nadella replied "You know, the thing that perhaps most influenced me in terms of how do you look at the journey or a career, there was this guy whose name was Mike Naples. He was the president of Microsoft when I joined. And he had this saying where he would say, 'Look, all HR systems are long-term efficient, short-term inefficient.' And I thought that that phrase just captured it. Which is, it's not really about asking for the raise, but knowing and having faith that the system will actually give you the right raises as you go along. And that, I think, might be one of the additional superpowers that, quite frankly, women who don't ask for a raise have. Because that's good karma. It'll come back because somebody's going to know that's the kind of person that I want to trust. That's the kind of person that I want to really give more responsibility to. And in the long-term efficiency, things catch up. And I wonder—and I'm not saying that that's the only approach, I wonder whether taking the long term helps solve for what might be perceived as this uncomfortable thing of, hey, am I getting paid right? Am I getting rewarded right? Because reality is your best work is not followed with your best rewards. Your best work then has impact, people recognize it, and then you get the rewards. And so you have to somehow think that through, I think."

I envy Satya for getting all he did without ever having to fight or ask for anything. I'm sure this is his experience, but I don't think this experience is universal. Some organizations are not very good at recognizing talent without some "pushing." This is not a gender issue at all. It is mostly recognizing what it takes to achieve your goals at a particular environment, and sometimes it takes asking.

One of Cloudera's most respected directors gave a presentation called "How to become a technical leader," and one of the lessons was "Ask for extra responsibility. Let everyone know you want to be a manager." I prefer this advice. Many people are afraid to appear ambitious, but I've never seen anything bad happen to people who ask for extra responsibility.

The other lesson I've learned is that career opportunities happen only when people want to work with you. This means that first, people need to know you—networking was key to every career move I made. Second, you need to bring a lot of value and be accountable for your work. We all want to work with people who, when they say they'll do something, they deliver. Perhaps this is what Satya meant when he said that good work is rewarded, but while good work is essential, it is rarely sufficient.

***Which big data products do you work with the most? What advantages do they offer over traditional relational database management systems?***

I work for Cloudera, so I work with products in the Hadoop ecosystem.

I work mostly with HDFS, Hadoop's distributed file system. I work a lot with Sqoop, which lets me copy data from relational databases to Hadoop. I work with Avro and Parquet, two file formats that define schemas for Hadoop files. Parquet is a columnar file format, which lets me efficiently run analytical queries. I use Hive and Impala; both are SQL engines for Hadoop.

I use all these tools to perform large-scale ETL in Hadoop and to perform some types of analytical queries in Hadoop.

The advantages over relational database systems depend a lot on the use case. For one thing, Hadoop is cheap, it can be free or you can license it for around $2,000 a node. If your query or ETL requirements are not super-sophisticated and you just need a lot of spindles and a lot of CPU cores to process large amounts of data, the value proposition is difficult to match.

In addition, Hadoop has a lot of flexibility regarding data types, data ingest, and schemas that relational databases don't have. Many data warehouse projects fail because of the challenges involved in fitting the entire organization into a single schema. Hadoop can give you the flexibility to work around this.

For nerdy DBAs like me, working with open-source software is a real treat. There's little need to guess what the optimizer is doing behind your back or why something is super-slow. You can read the code, run a debugger, and find the issue. If you are lucky, you can even fix it. I'm not sure I can go back to working on closed-source databases and all the strange tricks we did to figure out how the database actually works.

Recently, I've been working a lot on streaming ETL. This mode of ETL doesn't have the large-scale efficiency of the usual batch ETL, but it gives organizations some of the analysis they need in real time. The tools I use for this are Spark Streaming as the stream-processing engine, Kafka as a reliable source of streaming data, and HBase as a NoSQL database where I can rapidly update "materialized views" with the streaming information. I'm still at the beginning of this journey, but so far I'm enjoying the change of pace to a radically lower latency.

***How and where can our readers learn about these products? Any books in particular? Tutorials? Conferences?***

I have to mention my own book *Hadoop Applications Architectures*, which will be published soon by O'Reilly. This book requires some Hadoop knowledge and it goes into the little decisions that are involved in production implementations—things like how to define schemas and how to manage complex ETL workflows. We go through three very common use cases, including ETL and data warehousing.

***"If your query or ETL requirements are not super super-sophisticated and you just need a lot of spindles and a lot of CPU cores, to process large amounts of data, the [Hadoop] value proposition is difficult to match."***

If you need a beginner-friendly book, I recommend *Hadoop the Definitive Guide*, by Tom White, and *Hadoop Operations*, by Eric Sammer. The latter in particular is short and very useful for DBAs looking to expand their skill sets.

Both large Hadoop vendors have courses and certifications. I found Cloudera's classes very useful when I started my career. I took all classes, which gave me a solid start.

The best conferences are Strata/Hadoop World and Hadoop Summit. However neither is really for beginners, so start visiting after you have a bit of experience.

This has been similar to my experience with Oracle conferences: when I was a complete newbie, I couldn't find any session I could understand. It took me two years to appreciate what I heard in my first Oracle conference. The only presenter who is completely clear and interesting to attendees of all levels is Tom Kyte. I wish I knew how he does this.

***You're an active user of social media, aren't you? Of late, I've been getting friend requests from people who I only know at a professional level, which seems strange to me because they're colleagues, not friends. What's the role of social media in our professional lives, if any?***

As I mentioned in the beginning of the interview, social media plays a huge role in my career.

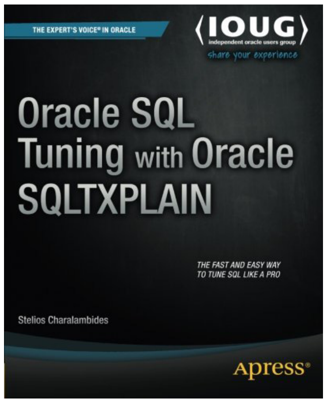
I spent a lot of time on the road with customers. This can be incredibly frustrating and challenging at times, and there are no co-workers around to gripe to. My friends and spouse were also very far away.

Social media became my release valve. I could express my difficulties and frustrations and get sympathy and sometimes even useful help. These days I tweet some of my ideas when preparing slides for conferences. This generates interest in my talks but also lets me discuss my ideas with people. The talks are often much improved as a result.

I also love it when people use social media to give me feed-

*(continued on page 26)*

# Oracle SQL Tuning with SQLTXPLAIN



A Book Review by Brian Hitchcock

## Details

**Author:** Stelios Charalambides

**ISBN:** 978-1-4302-4809-5

**Pages:** 317

**Year of Publication:** 2013

**Edition:** 1

**List Price:** $44.99

**Publisher:** Apress

## Summary

**Overall review:** Very good; a clearly written description of what SQLTXPLAIN does and how to use it effectively.

**Target audience:** Anyone who needs to tune SQL in an Oracle environment.

**Would you recommend to others?:** Yes.

**Who will get the most from this book?:** Anyone who is working with individual SQL performance issues.

**Is this book platform specific?:** No.

**Why did I obtain this book?:** NoCOUG asked me to review this book.

## Overall Review

SQLTXPLAIN is a powerful utility for researching and resolving Oracle SQL performance issues, and this book explains how to use it very well.

Before we go any further, I want to discuss my experience with Oracle SQL tuning in general. I have worked in various IT environments for over 20 years, and I have never had a test environment that was dedicated to testing SQL performance, that was a duplicate of production, or that had any way of simulating the production load. Further, in my experience management was never willing to invest the considerable time and money needed to really track down the issue and resolve it. I don't say this to be mean-spirited or to imply that SQL tuning isn't important. I say this to make sure you understand that when I read this book, I knew I wouldn't be tuning SQL anytime soon and I didn't know what it would be like to have test environments available to me.

Book quality: With the advent of print-on-demand books and the demise of what we now call "traditional" publishing, I've noticed a substantial decline in the quality of the printed books I review. In this book the figures are fuzzy to the point that I can't read the column headings. I found 25 typos in the text, and I'm not a professional copyeditor and wasn't even looking for typos.

## Author's Introduction

The author explains that this is a practical guide to using the SQLTXPLAIN tool, how the book came to be written, and who should and should not buy this book. If you are looking for tuning theory, you are advised to look elsewhere. At the end of this section, we are told that tuning problems are the most complex issues. This book is focused on SQL statement tuning; from my experience, the real performance issues that I have seen recently involve over a dozen servers supporting a database, multiple WebLogic Servers, Web Servers, and associated ID management systems. I believe this level of complexity is more typical of modern commercial environments, and I think the issues we see in these systems are the most complex. The issues of tuning only a single SQL statement in a single database can be complicated, but the context is less involved.

## Chapter 1—Introduction to SQLTXPLAIN

Here we learn what SQLTXPLAIN is and its history. (Note that SQLTXPLAIN will be referred to as "SQLT" from here on.) I had never heard of SQLT before, and the author confirms that I am not alone. We also see that SQLT will not fix your problem SQL; it will help you see what is needed to fix it. I noticed that the discussion starts out assuming you have the one and only problem SQL already identified before you start looking at SQLT. How you decide which single SQL statement is worth tuning is not covered. I wonder if the author uses AWR reports or the ASH utility. The process of getting a copy and installing SQLT comes next. Note that you have to download and install the version of SQLT for your specific version of the Oracle database. You also need to have a license for the tuning and diagnostics packs to make full use of SQLT.

The steps needed to run your first SQLT report are shown. There are two different reports available: SQLXTRACT and SQLTEXECUTE. You can probably guess that SQLXTRACT reports on data found in the database for SQL that has already executed, while SQLTEXECUTE generates the data needed by actually executing the SQL in question. I think being able to execute the SQL is problematic in most production systems, especially if the SQL is altering data.

Starting with screenshots of the SQLXTRACT report that has been generated, we learn about the structure of the report, cardinality and selectivity, and what cost means to the optimizer. The execution plan section of the report is illustrated in detail, and join methods are discussed.

## Chapter 2—The Cost-Based Optimizer Environment

In this chapter, and in most of this book, the way the CBO works is discussed as we learn about using SQLT to diagnose SQL performance issues. The process is a gentle introduction to this subject and would be easy for new DBAs to follow. We learn that the CBO generally makes good decisions if it is given good information.

I found the discussion of system statistics interesting because I don't think I've focused on this before. There are sections for CBO parameters, hints, and the history of the performance of the execution plan. This was, like many things in this book, something I had not seen before. I can see how this information would be very useful when performance suddenly goes bad. The discussion of column statistics, cardinality, and out-of-range values is quite detailed.

The next section presents a case study where a SQL has suddenly slowed down, and SQLT is used to find when the execution plan got worse, along with the index that was added around the same time. It is assumed that you can execute the problem SQL repeatedly. I can see this if the SQL is only doing select, but anything that changes data would require a refresh of the data between each execution. This also assumes you have a test system that accurately models production, which I have not seen in my career.

## Chapter 3—How Object Statistics Can Make Your Execution Plan Wrong

The message here is simple: generating good statistics on the objects in the database is critical. The discussion covers the many ways that good statistics are not gathered, including sample size issues, histograms being used improperly, and many others. It is recommended that a pre-production test system be used to validate the statistics-gathering process before going to production. It's a good, logical idea that I have never seen in practice. I must work at bad places; I just don't see these good ideas actually happening! Another case study is given, this time involving a table that gets truncated every night.

## Chapter 4—How Skewness Can Make Your Execution Times Variable

First we see what skewness is and what it means. I had heard of this in the past, but I had not seen such a clear explanation of how to quantify it. Basically, it means that some predicates will match largely different numbers of rows in a table. The advice is to understand your data, which I agree with, but I support hundreds of databases and I'm not being paid to look at the data in any one database for any length of time. A specific process to measure skewness is shown and how much is too much is clearly defined, which is good. The author gives us something we can really measure in our own data.

Next we see how skewness can upset the optimizer, followed by a lengthy discussion of histograms, their various types, and when to use them. Bind variables and their effects on the execution plan are presented, followed by cursor sharing. The detail given about the various values available for the cursor-sharing parameter was new to me, and I have a better understanding of this now.

Another case study shows how execution time can change due to poor use of histograms.

## Chapter 5—Troubleshooting Query Transformations

Query transformations are described and illustrated with several examples. Somehow we find ourselves in Disneyworld; the trip to get there is an analogy for query optimization and Mickey comes in for some criticism! Oh snap! I'd advise the author to stay away from Mr. Mouse and his highly optimized team of lawyers!

Next we learn about the 10053 trace file, which documents how the optimizer decided on the execution plan. It turns out that SQLT generates this trace file automatically.

Specific query transformations that are covered include subquery unnesting (sounds awkward), complex view merging (check your mirrors), and join predicate pushdown, (which doesn't sound real to me but is).

Specific hidden database parameters are listed, as well as specific hints that affect how the CBO uses query transformation. This ties in to sections of the 10053 trace file, where we can see what parameters were in place, both normal and hidden. A trace file for a specific SQL statement is shown illustrating where we can find details of which, if any, query transformations were used.

The proper way to use hints is covered, followed by an example where the join method is changed. It was interesting to see an example where conflicting hints were specified and what the optimizer did in that case.

Finally, we see sections of a 10053 trace file used to show the actual cost computations that the optimizer performed as it examined various options.

## Chapter 6—Forcing Execution Plans Through Profiles

The topic of SQL profiles was new to me. If a SQL statement has been running well in the past and suddenly runs poorly, you can use SQLT to generate a script that will create the SQL profile from when the execution was good. This script can then be used to apply the SQL profile to the SQL in the present to force it to run well again. We are then told that the SQL profile doesn't actually force anything but is more like a "super hint." This is faster than gathering statistics and looking for other causes of the performance issue. The SQL profile allows you to freeze the execution plan of a specific SQL statement. Note that once you do this, the optimizer won't change the execution plan and, over time, the profile may not be the best plan.

An example of such a script is given, and we see that the SQL profile tells the optimizer how to handle separate "query block names." These are generated by the optimizer as it transforms and examines the original SQL.

We are shown how to use one of the many parts of the SQLT tool to generate the SQL profile. The author tells us that before doing this, we must get Oracle Support to approve what we are doing. I assume this means that using a SQL profile can be dangerous, but no explanation is given.

This script is executed, which generates the script to apply the SQL profile. This script is shown in detail, with specific sections highlighted so we can see where the profile is being applied. This script is executed to actually apply the SQL profile, and the output of this is also shown. An example is given to show the execution plan, which includes a message confirming the use of a SQL profile.

The chapter ends with an example of how to generate the SQL profile on one database and apply it to a remote database.

***Remember:*** don't use a SQL profile without talking to Oracle Support first!

## Chapter 7—Adaptive Cursor Sharing

This chapter covers the many details of adaptive cursor sharing. The author describes this topic as one of the most confusing and misunderstood areas of the optimizer. I have heard of this before, but I've never understood what it is, how it works, and why it is important. This was introduced in 11g to deal with changing bind variables. The need for bind variables is covered, and the cursor_sharing parameter is shown. Bind peeking is explained as well as bind sensitive and bind aware cursors. An example is shown where SQL is set up to see how all of this works when a table has skewed data. The following SQLT report shows what the optimizer did for table data that was rare and common. I learned a lot from this discussion, I had not seen this level of detail before.

## Chapter 8—Dynamic Sampling and Cardinality Feedback

This chapter deals with something I've always wondered about: what happens when the optimizer finds a table with statistics that are clearly bad or nonexistent? Dynamic sampling is explained as well as cardinality feedback, which I had not heard of before. We see how to set up dynamic sampling, and sections of a 10053 trace file show us how the optimizer uses this feature. Cardinality feedback is activated using a hidden parameter that causes the optimizer to store information about each step of the execution plan and compare this data over multiple executions. If these data show a problem, the optimizer will sample the table data. It is noted that dynamic sampling is used as a last resort. If it is happening, you need to gather better statistics! The example SQL discussed runs well on one system and less so on another. The SQLT report is used to find the bad statistics on the system that has the poor performance.

## Chapter 9—Using SQLTXPLAIN with Data Guard Physical Standby Databases

I hadn't thought about it, but since SQLT requires installation in your local database, how would you use it to look at SQL tuning issues that might come up in your Data Guard physical standby? If you are running reporting on the standby database, you may run into SQL tuning issues you don't see on the primary OLTP database. Data Guard is described briefly, and we are reminded that we must often practice whatever disaster recovery process we have. And I am reminded that I have never worked anywhere that did this. The few times there was a process in place, I never saw it used because no one was confident it would work. Another piece of the SQLT toolbox is described, SQLTXTRSBY, which runs in your local database but goes out to the Data Guard physical standby database to gather the data it needs.

It is no surprise that there are limitations on this process, and these are described. Not all features of SQLT are available when looking at SQL running on the standby. How to use XTRSBY is shown and a sample output is discussed.

Another piece of the SQLT toolkit is described, roxtract, which is used on the standby to gather some of the data that XTRSBY can't.

## Chapter 10—Comparing Execution Plans

I have never had this situation, but if you have one database where a SQL statement is running well and another database where it isn't, you can compare the execution plans of both to fix the bad one. I wonder how often this comes up in practice, but I realize my experiences are just that: only what I've seen. The SQLTCOMPARE tool is used to generate reports on both databases, bring the output together, and generate a report showing how the execution plans are different. This tool can be used to look at SQL execution across different platforms. I can see this being useful, for example, in a hardware migration scenario. A practical example is shown, following the many necessary steps.

## Chapter 11—Building Good Test Cases

Test cases, we are told, are for situations where the solution to the tuning problem is not obvious. Assuming all the usual causes of poor SQL performance have been eliminated, using test cases may be the next thing to try. A test case allows for testing all sorts of things away from the main environment. Building the test case in another environment is complicated, and SQLT can help. The tools that build the test case don't usually copy the data (that could make the test case too large), but they can. Test cases can be run on a different platform from the source. Note that the scripts that create a test case make lots of changes to the system where they run, so you must only run them in an environment where you can quickly rebuild everything. The scripts collect all the metadata and statistics needed to reproduce the issue.

Building a test case is described through a detailed example with lots of output shown. I found it interesting that the test case doesn't need the table data because it has all the statistics from the tables in the source system. The test case will result in the same execution plan in the test system even though there is no data.

## Chapter 12—Using XPLORE to Investigate Unexpected Plan Changes

The XPLORE tool is used to measure the impact of changing database parameters. The tool uses a list of the database parameters and changes one at a time to see how the execution plan is affected. The tool was originally designed to do this brute-force analysis to look for bugs in the optimizer when a database upgrade was being tested. How to use this tool is shown and limitations are discussed. XPLORE won't tell you about bad statistics, for example. The fix control facility is described, which allows you to turn specific bug fixes on and off. I had not heard of any of this before. Clearly, this should only be done on a test system.

I also had not heard of the SQL Monitoring Report. This feature is turned on by adding a hint to your SQL, and statistics are captured as the SQL executes. A detailed example is shown where XPLORE is set up and executed, and the SQL monitor reports are generated.

## Chapter 13—Trace Files, TRCANLZR and Modifying SQLT Behavior

SQLT has several tools to help us look at trace files. The 10046 trace file, what it does, and how to use it are described. TKPROF, which is not part of SQLT, is described, followed by TRCASPLIT, the tool in SQLT that processes 10046 and 10053 trace files and generates a more useful output. The TRCANLZR is used to generate one set of aggregate information from multiple trace files, for example, when SQL is processed in parallel.

## Chapter 14—Running a Health Check

SQLT, as useful as it is, must be installed in the local database. In situations where you can't install SQLT, you can use a separate tool called SQL Health Check (SQLHC). This is a free download from Oracle. It is a set of scripts that do not install anything in the local database and generate a subset of the information you get from SQLT. Running SQLHC is shown, as is the report that is generated. Several of the individual scripts are described in detail.

## Chapter 15—The Final Word

This final chapter opens with a summary of the entire book. Tuning methodology is discussed in general to show how SQLT fits into the overall process. SQLT is described as the best tuning utility for focusing on one SQL statement because it gives you the overall picture in a format that is easy to navigate.

## Appendix A—Installing SQLTXPLAIN

This appendix shows, in detail, the steps to install SQLTXPLAIN. The screen output you will see during the installation is shown, along with notes about possible issues. For example, the tool requires about 2 MB of tablespace and requires a local database connection. The possible licensing levels are covered, as well as how to install the tool remotely and how to automate the installation, i.e., a non-interactive or silent install. Finally, the process to uninstall is explained.

## Appendix B—The CBO Parameters (11.2.0.1)

The XPLORE method of SQLTXPLAIN reports the effects of changing CBO parameters. To support this, SQLTXPLAIN contains a list of all the CBO parameters it uses. This appendix lists all of these parameters and includes hidden parameters. There are explicit warnings that we should not use any of the hidden parameters without first consulting with Oracle Support. Selected hidden parameters are described in detail, followed by the complete list of parameters that SQLTXPLAIN can use, for version 11.2.0.1.

## Appendix C—Tool Configuration Parameters

This appendix covers all of the available configuration parameters for SQLTXPLAIN. Each parameter is described as well as the requirements to make use of the parameter and dependencies between parameters.

## Conclusion

This book is an excellent introduction to a set of SQL tuning utilities that I had not heard of before. If you are tuning individual SQL statements and you have a test database, the various tools that make up SQLT should be very useful to you.

## Postscript—Questions I have for the author

While reading the book, the following questions came to mind. They are not answered in the book so I'm hoping that the author will read them and take the time to respond.

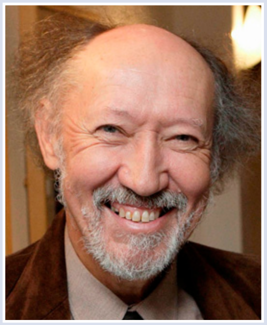
- Tuning takes time. How do we know in advance that the time we spend tuning will be worthwhile? If we spend 10 hours tuning, how do we know that we will save 10 hours of database performance over the next week or month or quarter?
- How do we quickly determine that a specific tuning problem is worth the time we think we will need to fix it?
- How long do we pursue a tuning situation before we decide it isn't worth the time we think we will save with the improved SQL?
- My experience is only that: what I have experienced. In your experience, do most DBAs work in environments where they have duplicate or test databases where they can test SQL with the same data and the same workload as the production system?
- In my many years of DBA work, I have never seen a disaster recovery setup that was actually used. I have worked in several places that had what was called a disaster recovery system, but I never saw it actually used. In your experience, how many organizations actually switch between primary and disaster recovery systems on a regular basis?
- The group I work in now supports multiple customers with hundreds of databases. We have no idea what the data looks like in any one customer's database. To tune effectively, we need to "know the data" in the database. Can I effectively tune SQL in this environment?
- It has been many years since I have seen a customer's system where the performance was limited by the database. The database is almost always much faster than all the middle tiers that are doing application processing between the database and the end user. How relevant is SQL tuning for modern, multi-tier application environments?
- SQL tuning focuses on one problem SQL statement. How realistic is this? How often do complex systems have one and only one SQL that is causing all the performance issues?
- Here's a specific example from my recent experience: One of my coworkers received a customer complaint about slow performance. They reviewed the system and decided that several new indexes would help the slowest SQL run faster. They spent a week preparing new indexes and got the customer to approve the downtime to make the needed changes. But after the new indexes were implemented, performance was worse. The SQL that was taking the most resources had changed during the week spent tuning. The problem SQL the week before was not the problem SQL a week later. The customer was not happy. How could this have been handled better? ▲

*Brian Hitchcock worked for Sun Microsystems for 15 years supporting Oracle databases and Oracle Applications. Since Oracle acquired Sun, he has been with Oracle supporting the On Demand refresh group and, most recently, the Federal On Demand DBA group. All of his book reviews and presentations—and his contact information—are available at* **www.brianhitchcock.net**. *The statements and opinions expressed here are the author's and do not necessarily represent those of Oracle Corporation.*

*Some Comments on Iggy Fernandez's Paper*

# The Rise and Fall of the NoSQL Empire

by C.J. Date


*C.J. Date*

Overall I like this paper, and I agree with its general message. Though I have to say I don't understand the title! The paper has nothing to say about the rise of "the SQL empire," nor does it show it has fallen, either; if anything, in fact, it shows it hasn't. *[Editor's Note: Fernandez's NoSQL paper was inadvertently printed with the title "The Rise and Fall of the SQL Empire."]* Indeed, the general message of the paper—the message I just said I agreed with—is that there's no conflict between what has come to be known in the industry as "NoSQL" and the relational model. That said, however, I do have a number of comments and observations at the detail level, and the purpose of this note is to document those comments and observations. For convenience, what follows is keyed to section headings in Fernandez's paper. All otherwise unattributed quotes are from that paper.

## Functional Segmentation

"Functional segmentation is the underpinning of NoSQL technology, but it does not present a conflict with the relational model; it is simply a physical database design decision." Absolutely correct. I couldn't agree more.

"The collection of functional segments could be regarded as a single distributed database. However, distributed transactions are forbidden in the NoSQL world. Functional segmentation can therefore result in temporary inconsistencies . . . ." I agree with the first sentence here. As for distributed transactions and temporary inconsistences, however, I have a different perspective on what's really going on with consistency in the NoSQL world (and indeed elsewhere), which I'll elaborate on in the next section but one.

## Sharding

"Sharding is an essential component of NoSQL designs but it does not present a conflict with the relational model; it too is simply a physical database design decision." Absolutely correct. I couldn't agree more.

"In the relational model, the collection of . . . shards can be logically viewed as a single distributed database." Again I basically agree, modulo my reservations (hinted at above) regarding consistency.

## Replication and Eventual Consistency

I have no problem with the broad intent of this section or with the business requirement the NoSQL developers are aiming at in this connection. As indicated earlier, however, I do have a different perspective on what's really going on here. I also disagree, somewhat, with the quote from Codd in this section. Let me try to explain:

- First, to say that a database (distributed or otherwise) is consistent merely means, formally speaking, that the database conforms to all stated integrity constraints. Now, it's crucially important that databases *always* be consistent in this sense; indeed, a database that's not consistent in this sense, at some particular time, is like a logical system that contains a contradiction. Well, actually, that's exactly what it is—a logical system with a contradiction. And in a logical system with a contradiction, you can prove anything; for example, you can prove that 1 = 0. (In fact, you can *prove* that you can prove that 1 = 0 in such a system!) What this means in database terms is that if the database is inconsistent in the foregoing sense, you can never trust the answers you get to queries (they may be false, they may be true, and you have no way in general of knowing which they are); all bets are off. That's why consistency in the foregoing sense is crucial.
- But consistency in the foregoing sense isn't necessarily the same thing as consistency as conventionally understood (consistency as understood outside the world of databases in particular). Suppose there are two items *A* and *B* in the database that, in the real world, we believe should have the same value. They might, for example, both be the selling price for some given commodity, stored twice because replication is being used to improve availability. If *A* and *B* in fact have different values at some given time, we might certainly say, informally, that there's an inconsistency in the data as stored at that time. But that "inconsistency" is an inconsistency as far as the system is concerned *only if the system has been told that A and B are supposed to be equal*—i.e., only if "*A* = *B*" has been stated as a formal constraint. If it hasn't, then (a) the fact that *A* ≠ *B* at some time doesn't in itself constitute a consistency violation as far as the system is concerned, and (b) importantly, the system will nowhere rely on an assumption that *A* and *B* are equal.
- Thus, if all we want is for *A* and *B* to be equal "eventually"—i.e., if we're content for that requirement to be handled in

the application layer—all we have to do as far as the database system is concerned is omit any declaration of "*A* = *B*" as a formal constraint. No problem, and in particular no violation of the relational model.

Now perhaps you can see why I don't entirely agree with that text of Codd's. Codd says:

> There are, of course, several possible ways in which a system can . . . respond to [inconsistencies]. In one approach the system checks for possible inconsistency whenever an insertion, deletion, or key update occurs. [*Incidentally, I don't know why Codd says "key update" specifically; surely checking should be done on all pertinent updates?*] Naturally, such checking will slow these operations down.

Well, the system can only "respond to" those inconsistencies it knows about, or in other words those that correspond to formally declared constraints. For such inconsistencies, it simply *must* do the checking whenever a pertinent update occurs; there's no alternative, because not to do that checking is to risk having a database for which all bets are off (see above). What's more, I don't agree that such checking will slow the system down. If the user has bothered to declare the constraint, presumably he or she wants it enforced—for otherwise there's no point in declaring it in the first place. And if the user wants that constraint enforced, and if the system isn't going to do it (by which I mean do it properly, by which I mean doing the checking on all pertinent updates), then the user is going to have to do it instead. Either way, the checking has to be done. What's more, I would hope that the system could do the checking more efficiently than the user; thus, I think that, far from the operations being slowed down, they should be speeded up, so long as the system does the right thing and shoulders its responsibility properly.

Back to Fernandez's paper. Fernandez goes on to say: "ACID has *nothing* to do with the relational model per se." Well, I agree with the broad sense of this observation, though I do have some reservations regarding ACID in general which it probably isn't appropriate to air in detail here. Let me just say that the relational model does at least tacitly require the database system never to lose information, and the A, I, and D features of transactions are aimed at satisfying this goal. In that sense, we might at least say those features of ACID are a mechanism for satisfying a certain relational requirement. (I disagree with the C feature, however, for reasons my earlier remarks on consistency should be sufficient at least to suggest. I don't want to get into further details on this issue here.)

One final small point on this section: Fernandez uses the phrase "post-relational object-oriented database management systems." Actually, I would say that while OODBMSs might be "postrelational" from a chronological point of view, they're actually *pre*relational in terms of their database functionality.

## The False Premise of NoSQL

"The belief stems from an unfounded assumption that has found its way into every relational DBMS—that every table should map to physical storage." I would replace the phrase "every relational DBMS" here by "every mainstream SQL DBMS" (there are certainly exceptions to what Fernandez is claiming here), but overall I agree with most of the paragraph in which this sentence appears.

"It would also be perfectly legitimate to provide a non-relational API [to a relational system] for the important use cases." Legitimate, yes, but I believe it would be quite unnecessary, as I tried to explain in my contribution to the interview with Hugh Darwen and myself that appeared in a recent issue of the *NoCOUG Journal* ("No! to SQL! No! to NoSQL!," *NoCOUG Journal* 27, No. 3, August 2013).

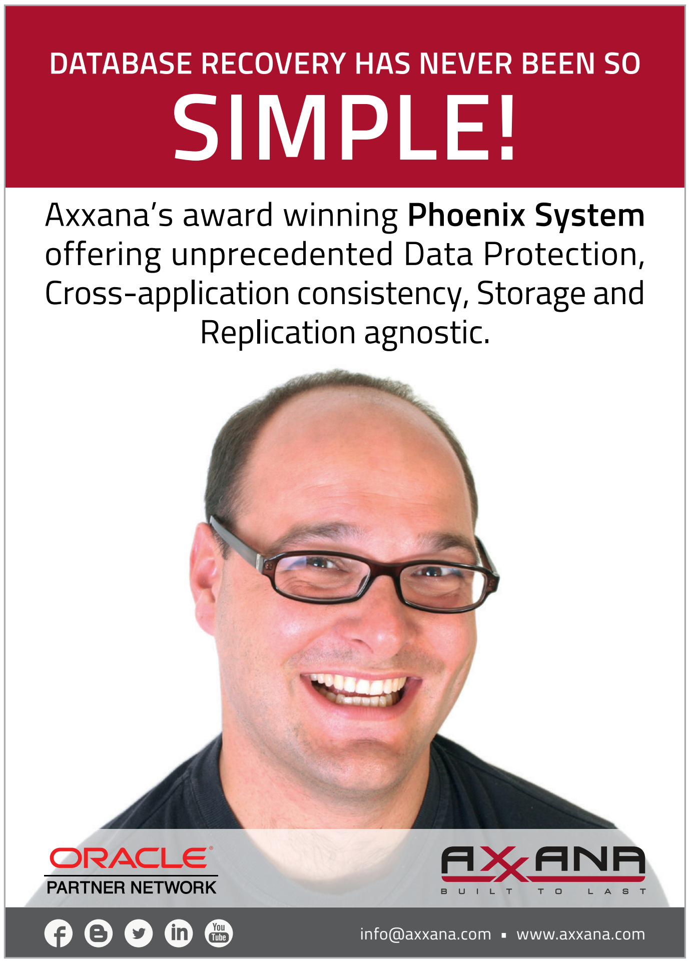
"The key-blob . . . approach . . . would be called 'zeroth' normal form in relational terminology." No, it wouldn't. There's no such thing as "zeroth normal form" in relational terminology. In fact, there's no such thing as an unnormalized relation—the very phrase is a contradiction in terms. *All* relations are normalized, in the sense that they're in at least first normal form. To suggest otherwise is to do the cause of genuine understanding a serious disservice.

## NoSQL Buyer's Guide

Here I'd just like to say that I agree very strongly with the remarks by Codd quoted in this section.

## Summary

This isn't a comment on the section of this name in Fernandez's paper—I'm just using this heading as a convenient place to compliment Fernandez on a most interesting article. My comments in the foregoing, especially the ones of a critical nature, aren't meant to detract from what I see as a most useful contribution to the ongoing NoSQL debate. Thank you, Iggy! ▲

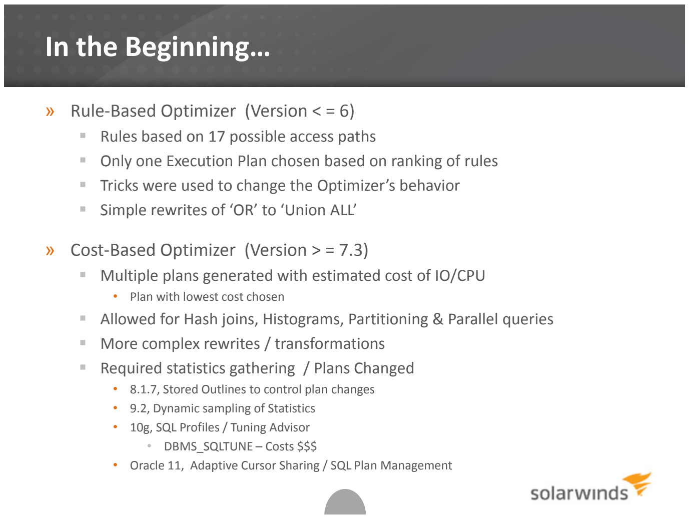

DATABASE RECOVERY HAS NEVER BEEN SO
SIMPLE!
Axxana's award winning **Phoenix System** offering unprecedented Data Protection, Cross-application consistency, Storage and Replication agnostic.
ORACLE PARTNER NETWORK
AXXANA BUILT TO LAST
info@axxana.com ▪ www.axxana.com

# A Short History of the Oracle Optimizer

by Janis Griffin

*Janis Griffin*

In Figure 1, we can quickly see the evolution of Oracle's optimizer and the new capabilities each release produced.



In the Beginning…

- Rule-Based Optimizer (Version < = 6)
  - Rules based on 17 possible access paths
  - Only one Execution Plan chosen based on ranking of rules
  - Tricks were used to change the Optimizer's behavior
  - Simple rewrites of 'OR' to 'Union ALL'
- Cost-Based Optimizer (Version > = 7.3)
  - Multiple plans generated with estimated cost of IO/CPU
    - Plan with lowest cost chosen
  - Allowed for Hash joins, Histograms, Partitioning & Parallel queries
  - More complex rewrites / transformations
  - Required statistics gathering / Plans Changed
    - 8.1.7, Stored Outlines to control plan changes
    - 9.2, Dynamic sampling of Statistics
    - 10g, SQL Profiles / Tuning Advisor
      - DBMS_SQLTUNE – Costs $$$
    - Oracle 11, Adaptive Cursor Sharing / SQL Plan Management

*Figure 1.*

Prior to version 7.3, the optimizer was only rule-based, meaning that there were 16 rules that it always looked for and then produced a plan that followed those rules. There was no deviation from the rules, so DBAs had to come up with tricks to change the behavior. The problem with the rule-based optimizer is that it couldn't scale very well. As datasets grew from gigabytes to terabytes, the rule-based optimizer started to break down. So beginning in v7.3, the cost-based optimizer came into play. This allowed for more sophisticated options such as HASH JOINS, HISTOGRAMS, and PARALLEL QUERIES on PARTITIONED tables.

Unfortunately, since the cost-based optimizer always looked at actual CPU/IO costs to come up with a plan, the plans could change over time due to changes in the system, data size, or database configurations. These changes were not always for the better. In fact, performance regressions could occur and did. That is why with each new Oracle release, different options were developed to try to control plan stability. In 8.1.7, stored outlines were introduced to give hints to the optimizer to stabilize the plan.

In 9.2, dynamic sampling first became available. It gave the cost-based optimizer the option to sample the tables in a query during a hard parse, to determine if better default statistics for unanalyzed segments could be determined (in order to verify its guesses). This type of sampling takes place only at hard parse time and is used to dynamically generate better statistics for the optimizer to use—therefore, the name "dynamic sampling."

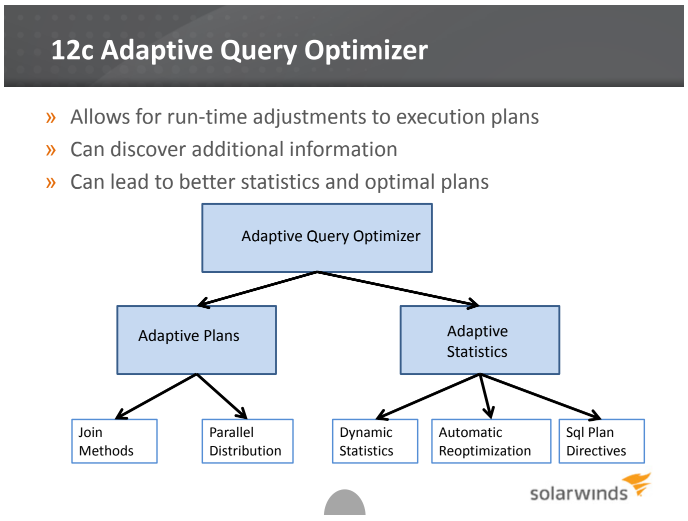
The optimizer uses a variety of inputs to come up with a plan. It uses any and all constraints defined on the table: system statistics—information about your server's I/O speeds, CPU speed, etc., and statistics gathered from the segments involved in the query. The optimizer uses statistics to estimate cardinalities. For example, the number of rows that each step in a plan is expected to return. Those cardinalities are a major variable in computing the cost of a query. When cardinalities are incorrectly estimated, the optimizer may choose an inefficient query plan. The number one reason for the optimizer to generate an inefficient plan is inaccurate cardinality estimations. Per Tom Kyte: "I like to say 'right cardinality equals right plan; wrong cardinality equals wrong plan.'"

So, "right cardinality" is the motivation behind dynamic sampling to help the optimizer get the right estimated cardinality values. Feeding the optimizer more accurate information specific to the query itself will only help the optimizer come up with the best execution plan. Prior to Oracle 12c, dynamic sampling offered 11 setting levels (0 through 10). In 9.2, the default dynamic sampling level value is 1, whereas in Oracle 10.1 and above it defaults to 2. In Oracle 12c, the dynamic sampling name has been changed to "dynamic statistics," where a new level, 11, for automatic statistics gathering has been added.

***"Right cardinality equals right plan; wrong cardinality equals wrong plan."***

Versions 10 and 11 offered more enhancements to plan stability. Most notably in version 10, Oracle came out with ACTIVE_SESSION_HISTORY table, SQL profiles, and the tuning and diagnostics packs, which can be purchased for an additional cost. Version 11 gave us control over BIND VARIABLE PEEKING with adaptive cursor sharing, which I'll go into later, plus BASELINES or SQL plan management (SPM).

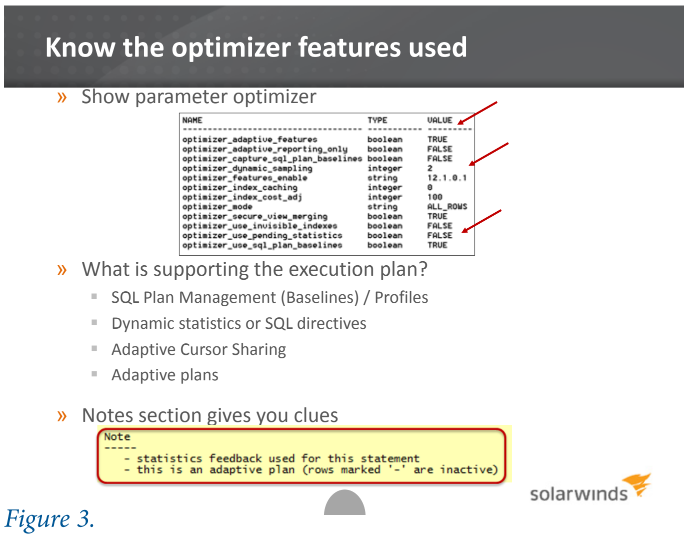
Oracle 12c introduced the adaptive query optimizer, which tries to adapt or make run-time adjustments during the first execution of a plan. Figure 2 shows the components that make up the adaptive query optimizer. Basically you have adaptive plans, which can change the join methods or parallel distribution at execution time, and adaptive statistics, which can change and be stored for future use.


12c Adaptive Query Optimizer

» Allows for run-time adjustments to execution plans
» Can discover additional information
» Can lead to better statistics and optimal plans

*Figure 2.*

In SQL tuning, it's important to know which features the optimizer is using to understand how it can affect the execution plan. To find all of the optimizer features, you can issue a show parameter optimizer command. Notice in Figure 3 that I'm running all of the optimizer features available for version 12.1.0.1. Also, notice that the OPTIMIZER_ADAPTIVE_FEATURES and OPTIMIZER_USE_SQL_PLAN_BASELINES are set to TRUE. Therefore, plans can be supported by a baseline or adapted on the fly. It can also use statistics feedback or SQL directives to change the execution plan. It's important to review the NOTE section of the plan, as it gives you valuable clues on which features the optimizer is using. In the example below, the optimizer used statistics feedback and adapted the plan on the very first execution.


Know the optimizer features used

» Show parameter optimizer

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
optimizer_adaptive_features          boolean     TRUE
optimizer_adaptive_reporting_only    boolean     FALSE
optimizer_capture_sql_plan_baselines boolean     FALSE
optimizer_dynamic_sampling           integer     2
optimizer_features_enable            string      12.1.0.1
optimizer_index_caching              integer     0
optimizer_index_cost_adj             integer     100
optimizer_mode                       string      ALL_ROWS
optimizer_secure_view_merging        boolean     TRUE
optimizer_use_invisible_indexes      boolean     FALSE
optimizer_use_pending_statistics     boolean     FALSE
optimizer_use_sql_plan_baselines     boolean     TRUE
```

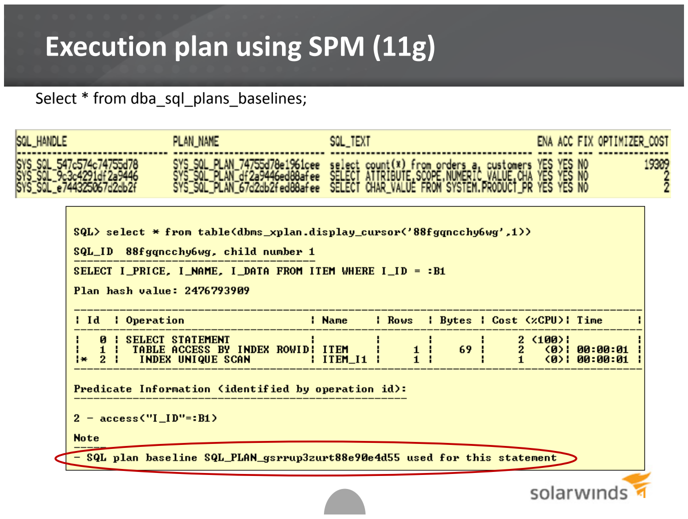
» What is supporting the execution plan?
- SQL Plan Management (Baselines) / Profiles
- Dynamic statistics or SQL directives
- Adaptive Cursor Sharing
- Adaptive plans

» Notes section gives you clues

```
Note
-----
   - statistics feedback used for this statement
   - this is an adaptive plan (rows marked '-' are inactive)
```

*Figure 3.*

Figure 4 shows an example of how an execution plan is supported by a baseline. Baselines tell the optimizer which plan(s) to always use. They are stored in the DBA_SQL_PLANS_BASELINES table for any query executed more than once. This table holds the SQL_HANDLE, PLAN_NAME, and SQL_TEXT for the query, and shows if the baseline has been enabled and accepted. The NOTE section of the execution plan will list the baseline plan name that it's using.


Execution plan using SPM (11g)

Select * from dba_sql_plans_baselines;

```
SQL_HANDLE                PLAN_NAME                      SQL_TEXT                                 ENA ACC FIX OPTIMIZER_COST
------------------------- ------------------------------ ---------------------------------------- --- --- --- --------------
SYS_SQL_547c574c74755d78  SYS_SQL_PLAN_74755d78e1961cee  select count(*) from orders a, customers YES YES NO           19309
SYS_SQL_9c3c4291df2a9446  SYS_SQL_PLAN_df2a9446ed88afee  SELECT ATTRIBUTE,SCOPE,NUMERIC_VALUE,CHA YES YES NO               2
SYS_SQL_e744325067d2db2f  SYS_SQL_PLAN_67d2db2fed88afee  SELECT CHAR_VALUE FROM SYSTEM.PRODUCT_PR YES YES NO               2
```

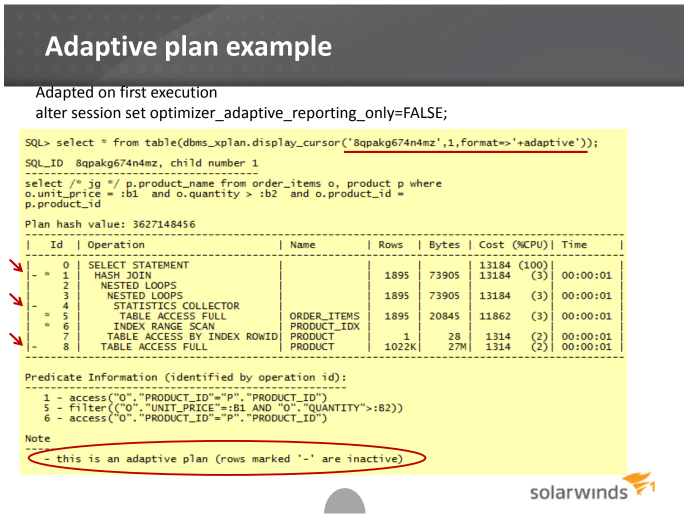
```
SQL> select * from table(dbms_xplan.display_cursor('88fgqncchy6wg',1))

SQL_ID  88fgqncchy6wg, child number 1
-------------------------------------
SELECT I_PRICE, I_NAME, I_DATA FROM ITEM WHERE I_ID = :B1

Plan hash value: 2476793909

-------------------------------------------------------------------------------
| Id  | Operation                   | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |       |       |     2 (100)|          |
|   1 |  TABLE ACCESS BY INDEX ROWID| ITEM    |     1 |    69 |     2   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN         | ITEM_I1 |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

2 - access("I_ID"=:B1)

Note
-----
- SQL plan baseline SQL_PLAN_gsrrup3zurt88e90e4d55 used for this statement
```

*Figure 4.*

As I stated earlier, the optimizer can now adapt plans on the first execution. It does this by sampling a few blocks of the tables in the query and then changes either the join method (e.g., HASH JOIN or NESTED LOOPS) or parallel distribution method on the fly. You can see if a plan has adapted by looking at two new columns in the V$SQL table. The IS_RESOLVED_ADAPTIVE_PLAN column will contain a Y if the plan was adapted. The IS_REOPTIMIZABLE column instructs the optimizer to do a hard parse on the next run so it can store better statistics about the query.

Figure 5 is an example of an adapted plan. Notice the new format option of '+ADAPTIVE' in the DBMS_XPLAN.DISPLAY_CURSOR call. This format option will tell you which steps were changed or are inactive. In this example, the optimizer sampled a few blocks of the ORDER_ITEMS and PRODUCTS tables via the STATISTICS COLLECTOR step and then decided to do NESTED LOOPS instead of a HASH JOIN. Again, the NOTE section tells us what it did.

*(continued on page 26)*


Adaptive plan example

Adapted on first execution
alter session set optimizer_adaptive_reporting_only=FALSE;

```
SQL> select * from table(dbms_xplan.display_cursor('8qpakg674n4mz',1,format=>'+adaptive'));

SQL_ID  8qpakg674n4mz, child number 1
-------------------------------------
select /* jg */ p.product_name from order_items o, product p where
o.unit_price = :b1  and o.quantity > :b2  and o.product_id =
p.product_id

Plan hash value: 3627148456

-----------------------------------------------------------------------------------------
|   Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT             |             |       |       | 13184 (100)|          |
|-  * 1 |  HASH JOIN                   |             |  1895 | 73905 | 13184   (3)| 00:00:01 |
|     2 |   NESTED LOOPS               |             |       |       |            |          |
|     3 |    NESTED LOOPS              |             |  1895 | 73905 | 13184   (3)| 00:00:01 |
|-    4 |     STATISTICS COLLECTOR     |             |       |       |            |          |
|   * 5 |      TABLE ACCESS FULL       | ORDER_ITEMS |  1895 | 20845 | 11862   (3)| 00:00:01 |
|   * 6 |     INDEX RANGE SCAN         | PRODUCT_IDX |       |       |            |          |
|     7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT    |     1 |    28 |  1314   (2)| 00:00:01 |
|-    8 |   TABLE ACCESS FULL          | PRODUCT     | 1022K|    27M|  1314   (2)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("O"."PRODUCT_ID"="P"."PRODUCT_ID")
   5 - filter(("O"."UNIT_PRICE"=:B1 AND "O"."QUANTITY">:B2))
   6 - access("O"."PRODUCT_ID"="P"."PRODUCT_ID")

Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

*Figure 5.*

# Raiders of the Data Dictionary–Part I

## Indexing for the Workload

by Lothar Flatz

*Lothar Flatz*

In his rules, designed to define what is required from a database management system, Edgar Codd states in rule number four: "The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data."[1]

I have used the exhaustive knowledge that the database holds about its own structure in many ways during my ten years with Oracle Consulting. Never before or again did I use it as exhaustively as in one assignment—the background of this article: a complete re-indexing of a large online database. When the request came in, I hesitated at first. After all, none of my very experienced colleagues at Oracle Consulting wanted to take the task. But no risk, no fun—somebody's got to do it.

In the end, the objectives of my client were met, and the outcome was very positive. There are endless possibilities exploiting the rich statistical information that the database holds about itself. When will you go ahead and uncover the treasure in your own database?

The following tale refers to the indexing of an online database. So if "index" is mentioned, it refers to the conventional B*Tree index. For a data warehouse, different indexing (such as bitmaps) rules are beyond the scope of this article.

### A Consequential Coffee Break

I'll bet that most of the best (and silliest, for that matter) ideas occur when people are taking a break together—just like in this case. I was on a tuning assignment assisting one of my regular customers. I had coffee with a member of the DBA team. The experienced DBA, completely relaxed, was expounding again on his favorite topic: "I'll bet at least 30% of the indexes in our main application are redundant. As a consequence our execution plans are unstable. And that takes up space. I would love to have all indexes double-checked."

I reflected on his words. Did I agree? Yes. In fact, the developers writing the applications usually create the indexes. The developers, however, rarely experience the system under full load. In addition, the indexing is never adjusted to the needs of a particular customer. This means that during a tuning assignment, quite often I must rework the indexing. Moreover, the indexing performed by the original developers might be patched, but it is rarely questioned as a whole. Normally we just add new indexes on top of existing ones. The result is a confusing, dense jungle of indexes.

### What If?

How about starting from scratch and making the investment to get it right from the beginning? Once the application runs for a while we have all the facts we need. The Oracle database is indeed an inexhaustible source of information. All the information required to do high-quality indexing is provided in the shared pool or in the AWR tables.

The idea stuck with me. A few months later an opportunity opened up. One of my clients had enough of half-hearted solutions. He wanted to tidy the index structure of his company's most important applications. Granted, at first I needed all my skills of persuasion to convince the customer that the fundamental indexing should be re-made from scratch. Otherwise, I was convinced that we'd just get bogged down in doing improvements on the basis of existing indexing. Which index should be deleted and which not? Such discussions would paralyze the process unnecessarily and make it inefficient, as I knew from experience.

Of course we could now easily create a big SQL Tuning Set and shift the task to the SQL Tuning Advisor. We would then, however, have to rely blindly on the decisions of the advisor. We could not use the knowledge and experience of the developers and the DBAs.

While the automated tools reduce the complexity of the physical design process, it is not a trivial matter to identify a representative workload, which can be used to drive the index design in its entirety. Second, automated tools do not consider all factors that impact physical design (e.g., the impact of replication architectures). [E.g., compare 4].

Research indicates that a solution by means of automated tools is problematic for big tasks.[2, 3]

We decided to choose a semi-automatic procedure using a team of experts and write our own tools. Thus, each step could be explained and understood. Each team member could contribute ideas. At the end we had a clearly reasoned solution that everyone could support.

### Preparation Phase

First, we formed a team of people with technical expertise on the topic. It consisted of two representatives from the development team of the software vendor, two DBAs as a representative of the hosting company, and myself as an external consultant for Oracle. In two sessions we defined the basic rules for indexing:

- Primary and foreign key will always be indexed
- Naming conventions

- ➤ Design rules reflecting the specific needs of the application (e.g., temporal aspects)
- ➤ Type of physical storage (e.g., tablespace)

## Data Collection Phase

In this phase, we researched and compiled a lot of information about the query conditions used by the application. It was important to include all relevant processes in the evaluation. Thus we considered not only daily processing but also weekly and monthly activities.

This time-consuming phase took several months. However, automatic collection tools did the work; the team had relatively little active work to do. Essentially, the following information was collected:

- ➤ On a column level: the frequency and the relational operators representing the use of columns in query conditions
- ➤ On a column group level: the combination of columns commonly used in queries and the frequency of such usage

This is essentially the same information that the database collects to support the automatic generation of statistics in the sys.col_usage$ table in the data dictionary. As of version 11.2.0.2 column combinations are also supported. Maria Colgan, who at that time was the product manager of the Oracle optimizer, described the corresponding method in her blog [5].

In previous versions of the database column, combinations are not supported and must therefore be extrapolated from other sources. The sources:

- ➤ The current queries in the Shared Pool (you can easily obtain the search criteria from the Filtering and Access Predicates)
- ➤ The top statements from the AWR (here, unfortunately, you have to perform a reparse, since the Filter and the Access Predicates in the DBA_HIST_SQL_PLAN table are empty)

These options are important because you cannot rely solely on the contents of the sys.col_usage$ table. I have seen databases where the content of the sys.col_usage$ table was just representing a small part of overall activity. This is probably due to lack of memory assigned to the shared pool. It is clear that under such circumstance the statistics generation is affected as well.

We also used the dictionary when defining indexes:

- ➤ The selectivity of each column, and the combinations used
- ➤ Primary key and foreign key constraints

## Evaluation Phase

In this phase, the collected information was compiled into a final index design. We used a semiautomatic process that was divided into several stages. Human knowledge was still involved as a last resort in difficult design decisions.

The procedure very roughly follows the pattern of the Merge and Reduction algorithm.[4]

The foundation of the whole process was the column combinations found in the preliminary phase.

Basically, we could have created one index for every search combination discovered, but this would lead to an oversupply of indexes. Since each index can slow down DML operations [e.g., 7], only as many indexes as necessary and as few as possible should be created. The following steps will therefore attempt to optimize the index structure in terms of price/performance.

## Basic Indexing

Usually you will be able to start from a base indexing. Primary keys are normally indexed without much consideration. Foreign keys should also index in general, and for the following reasons:

1. If a foreign key constraint exists, deleting the parent row will cause a potentially harmful lock on the child rows.[6]
2. The Optimizer is not unnecessarily restricted in choosing the best access plan, because an efficient nested loop join from the parent table to the child table will be possible.

## Elimination

In this phase, we eliminated from consideration all the search combinations that can be skipped without a great loss of performance. These in particular were eliminated:

- ➤ Foreign—and primary—keys that have been indexed in the previous step
- ➤ Search combinations that are not very selective
- ➤ Search combinations that are already contained in other search combinations

Let us consider some examples from the well-known area of the address and person data: An index on {Gender} on the person table is normally fairly useless because the search column is not selective enough. There are possible exceptions if frequent searches for minorities occur. For example, a user dealing with women's affairs in the military will frequently search for the rare value. This example also shows that even seemingly simple cases cannot be decided without further consideration. Once again it becomes obvious that fully automatic indexing has its drawbacks.

A candidate index on {City} on the table address can be eliminated if there is another frequent search combination; for example, {City, Street}, which will also allow a search {City} alone. We can go a step further: A three-column index {last name, first name, Age} on the person table can support three different search criteria: {Last name, first name, age}, {Last name, first name}, and {Last name}. The respective search combinations can therefore be eliminated. Note that the following queries cannot be supported and must remain as candidate indexes: {First name, age} {Age} {First name}.

From these examples we learn that the choice of the right order of the columns is crucial for the quality of an index design.

## Synthesis

In this step, we want to merge the remaining search combinations and thereby eliminate additional index candidates. So you can, for example, expand a foreign key with additional search criteria. For example, let "item number" be a foreign key within the sales table. If the sales table is often searched for {item number, date of sale), it makes sense to expand an existing foreign key index on the item number by adding the sale date column.

Sometimes you can eliminate an additional index candidate by a slight deterioration in the index design. As an example, the following search combinations are given: {Place, last name, first name} and {Place, last name, date of birth}.

The candidate index {city, last name, first name, date of birth} can replace the two combinations shown above. Although the search support for {city, last name, and date of birth} is not as optimal as the specialized index, a slight disadvantage can probably be accepted. Place, last name are already good criteria by themselves. For a given place and last name range, "date of birth" can be used as a filter (rather than as an access predicate), which is still pretty effective.

You can see that in this step, a decision controlled by the reasoning of the human designer is particularly important.

## Tests and Implementation

After passing through all these steps, it is relatively easy to establish an index create script. Of course we must now test the result in detail. Oracle's Real Application Testing (RAT) is certainly an option that comes in handy at this phase of the project.

We did a series of tests. For the transition to production we also used a monitoring script to find missing indexes. Of course, you can also use the Oracle Index Advisor, but I believe that it is somewhat cumbersome for this specialized purpose. The new index structure was remarkably stable in production. In the first few months, only a single-digit number of indexes had to be created on the fly.

## Results

In the project just described, we created a new indexing on top of an application that had already been maintained for months by specialized Oracle consultants. We were pleased to see that the total re-indexing resulted in a space savings of 30%, in addition to an overall performance improvement of 30%.

Another very pleasant side effect was the improved stability of execution plans. For a particular access plan, there was now only one index available and not several, making the job of the Oracle Optimizer considerably easier. Consequently, the execution plans became more stable. In particular the plans were less sensitive to all sorts of instabilities, like the ones introduced by bind variable peeking or cardinality estimation errors.

Last but not least: a new index structure with clear rules and their standard names simplifies the daily work of DBAs. No question about it: fact-based indexing is an exciting field. It can make life a lot easier for DBAs, developers, and users.

The fact-based index design has now been in use at this particular client for seven years. So far there have been no negative side effects. Recently, the lead DBA of a company that runs that same application but is still using the legacy index design visited my customer to review the fact-based design. He summarized his experience: "DBAs are so relaxed around here. I want that too." ▲

## References

[1] Codd, Edgar Frank, "Is Your DBMS Really Relational?" Computerworld. October 14, 1985.

[2] Borovica, Alagiannis, Ailamaki. "Automated physical designers: what you see is (not) what you get," DBTest'12, May 21, 2012, Scottsdale, Arizona, USA.

[3] Weikum, Moenkeberg, Hasse, Zabback, "Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering," Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[4] Bruno, N. and Chaudhuri, S. 2007. Physical design refinement: The 'merge-reduce' approach. ACM Trans. Database Syst. 32, 4, 28. Nov. 2007.

[5] Colgan, Maria, "How do I know what extended statistics are needed for a given workload?" blog entry: **https://blogs.oracle.com/optimizer/entry/how_do_i_know_what_extended_statistics_are_needed_for_a_given_workload**.

[6] Oracle® Database Concepts 12c Release 1 (12.1) , Chapter 9, Locks and Foreign Keys, October 2014.

[7] Oracle® Database 2 Day DBA 12c Release 1 (12.1) , Chapter 8, Managing Indexes, November 2014.

**OraPub Oracle Performance Training**

**Become an expert in Oracle Performance!**
Come for a full week of amazing training

**San Jose | September 21 - 25, 2015**
Oracle Performance Firefighting – September 21 - 23
Advanced Oracle Performance Analysis – September 24 - 25

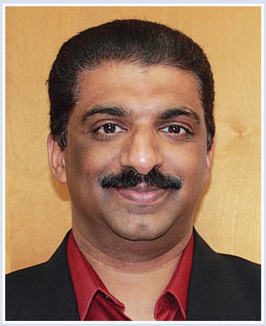
**NoCOUG Members SAVE 10%**
Use "NoCOUG10" at checkout

**QUESTIONS?** Contact Katrina Shallahamer today – Katrina@orapub.com
**Register at www.orapub.com**

OraPub TRAINING

# 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).

## IFILE in TNSNAMES.ORA

If you run Oracle E-Business Suite (EBS), you will be familiar with the use of IFILE in LISTENER.ORA, SQLNET.ORA, and LISTENER.ORA files (we'll call these files ".ORA files" collectively, instead of repeating their names again and again). Oracle lets us define custom entries (that are additional or replacement entries for those provided by Oracle EBS) in the IFILEs. The IFILE in EBS configuration is always the very last line in the .ORA files. The position of the IFILE entry has lot of significance. One might think you can place the IFILE entry anywhere in the .ORA files and the effect would be the same. This is partially true, because the effect would be same only if there are no duplicate entries. I will come back to the EBS IFILE use after the example.

Here is a tnsnames.ora file entry—ORCLRICEW is the entry I am going to use for demonstration, and you see an IFILE reference before the ORCLRICEW entry.

```
IFILE=/u01/app/oracle/cc12dev/11.2.0.4/network/admin/tnsnames_ifile.ora
ORCLRICEW = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = host07)(Port = 1525))
(CONNECT_DATA = (SID = oc1ricew)))
```

The IFILE has the following entry:

```
ORCLRICEW = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = host15)(Port = 1524))
(CONNECT_DATA = (SID = oc1prtst)))
```

Though the entry is the same, the host and instance it connects are different.

Let's check which entry SQL*Plus and tnsping picks to connect:

```
$ sqlplus system/*@orclricew
SQL> select host_name, instance_name from v$instance;
HOST_NAME            INSTANCE_NAME
-------------------- --------------------
host07               oc1ricew
```

It took the entry from the tnsnames.ora; this is good, though we had the IFILE before the STARRCEW entry in tnsnames.ora file.

I am going to move the IFILE reference to after the ORCLRICEW entry, as below, in the tnsnames.ora file.

```
ORCLRICEW = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = host07)(Port = 1525))
(CONNECT_DATA = (SID = oc1ricew)))
IFILE=/u01/app/oracle/cc12dev/11.2.0.4/network/admin/tnsnames_ifile.ora
```

Let's connect again.

```
$ sqlplus system/*@orclricew
SQL> select host_name, instance_name from v$instance;
HOST_NAME            INSTANCE_NAME
-------------------- --------------------
host15               oc1prtst
```

Wow! It did pick the entry from the IFILE.

Let's do one more test. Change the tnsnames.ora as below:

```
IFILE=/u01/app/oracle/cc12dev/11.2.0.4/network/admin/tnsnames_ifile.ora
ORCLRICEW = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = host07)(Port = 1525))
(CONNECT_DATA = (SID = oc1ricew)))
ORCLRICEW = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(Host = host11)(Port = 1526))
(CONNECT_DATA = (SID = oc1patch)))
```

And connect again:

```
$ sqlplus system/*@orclricew
SQL> select host_name, instance_name from v$instance;
HOST_NAME            INSTANCE_NAME
-------------------- --------------------
host11               oc1patch
```

Surprise again (for me)! It did pick the duplicate entry at the bottom of the tnsnames.ora file.

Maybe this is documented somewhere in an Oracle manual or My Oracle Support, but I could not find it (please tweet or email me if you know). So, the conclusion is:

- TNSNAMES.ORA file is read from the bottom up (or Oracle does not stop reading when it finds the first entry; it continues to read until it reaches the bottom) and picks the very last entry if there are duplicate entries.
- It does not matter if the duplicate entry is coming from IFILE or the main file.
- It does matter where you place the IFILE in the main .ORA files.

Back to Oracle EBS. Since autoconfig always overwrites the .ORA files and keeps the IFILE untouched, if you want to change the connection properties of entries like <dbname>_BALANCE or <dbname>_806_BALANCE (you would if you have a RAC database with multiple nodes), you can put your own well-balanced (or) using SCAN (or) target-directed entries in the IFILE and the configuration will be preserved through autoconfig runs. The EBS application will use the entry from the IFILE as IFILE is the last line in the .ORA files when autoconfig creates these files.

## Invisible Columns in Oracle Database 12*c*

In Oracle 12*c* database, you can make a column INVISIBLE. the The COL# column in COL$ is updated to 0, thus the column is not included in the "SELECT *" or "INSERT INTO VALUES", unless specifically selected. When you make an INVISIBLE column VISIBLE, the COL# is assigned the highest available. Thus the column becomes the last column in the table (for display purposes, not for storage purposes). So, if you accidentally make a column INVISIBLE and correct it by changing to VISIBLE, the column order changes. So, if the application uses "SELECT *" or "INSERT" without column names, it might break!

Why would you make a column invisible? There are not many reasons why you would suddenly make a column invisible. One situation that comes to mind is if you want to test the waters before dropping the column from a table—to figure out if something breaks or someone yells. Oracle provides an option to mark a column as UNUSED before you DROP, and do ALTER TABLE . . . DROP UNUSED COLUMNS at a later time. Once you mark a column as UNUSED, there is no going back to undo the action. So marking it INVISIBLE before dropping it is a good idea. Another use could be that you have a running application used by many teams. Before you collaborate with everyone on a table change, you could test the changes in the table by creating the new column as invisible, do your basic tests, and then talk to the other teams and make the column visible.

Let me demonstrate. Create a table with the following characteristics: Columns 1, 4, and 7 are regular columns with no specialty; x2 is invisible, x3 is virtual, x5 is invisible and virtual, and x6 is identity.

```
SQL> create table hidden_test (
     x1 number,
     x2 number invisible,
     x3 number generated always as (x1/2) virtual,
     x4 number,
     x5 number invisible generated always as (x1/4) virtual,
     x6 number generated always as identity,
     x7 number );

Table created.
```

Query the column properties of the table just created.

```
SQL> col column_name format a5
SQL> select column_name, user_generated, virtual_column, hidden_column,
            identity_column, column_id, segment_column_id, internal_column_id
     from dba_tab_cols
     where table_name = 'HIDDEN_TEST';

COLUM USE VIR HID IDE  COLUMN_ID SEGMENT_COLUMN_ID INTERNAL_COLUMN_ID
----- --- --- --- --- ---------- ----------------- ------------------
X7    YES NO  NO  NO           5                 5                  7
X6    YES NO  NO  YES          4                 4                  6
X5    YES YES YES NO                                                5
X4    YES NO  NO  NO           3                 3                  4
X3    YES YES NO  NO           2                                    3
X2    YES NO  YES NO                             2                  2
X1    YES NO  NO  NO           1                 1                  1
```

Notice all columns have USER_GENERATED as YES; the invisible columns are marked as HIDDEN_COLUMN=YES. Now, mark x4 as unused and x2 as visible.

```
SQL> alter table hidden_test set unused (x4);
Table altered.

SQL> alter table hidden_test modify x2 visible;
Table altered.
```

Query the properties again:

```
SQL> select column_name, user_generated, virtual_column, hidden_column,
            identity_column, column_id, segment_column_id, internal_column_id
     from dba_tab_cols
     where table_name = 'HIDDEN_TEST';

COLUMN_NAM USE VIR HID IDE  COLUMN_ID SEGMENT_COLUMN_ID INTERNAL_COLUMN_ID
---------- --- --- --- --- ---------- ----------------- ------------------
X7         YES NO  NO  NO           4                 5                  7
X6         YES NO  NO  YES          3                 4                  6
X5         YES YES YES NO                                                5
SYS_C00004 NO  NO  YES NO                             3                  4
_14060322:
52:06$

X3         YES YES NO  NO           2                                    3
X2         YES NO  NO  NO           5                 2                  2
X1         YES NO  NO  NO           1                 1                  1
```

The x4 column name marked for drop got renamed, and got a system-generated name. The rename is to facilitate adding a column with the same name to the table. The column x4 also got property changes: USER_GENERATED became NO and HIDDEN_COLUMN changed to YES. Also, the COLUMN_ID is released; thus the column will not be visible in "SELECT *" and "DESCRIBE". The UNUSED column still maintains the same INTERNAL_COLUMN_ID.

When x2 was made VISIBLE, it got a new COLUMN_ID assigned (the highest available; thus the column becomes the last column in "SELECT *" and "DESCRIBE"). Its hidden status changed to NO.

What happens to the ID columns, when the UNUSED column is dropped?

```
SQL> alter table hidden_test drop unused columns;

Table altered.

SQL> select column_name, user_generated, virtual_column, hidden_column,
            identity_column, column_id, segment_column_id, internal_column_id
     from dba_tab_cols
     where table_name = 'HIDDEN_TEST';

COLUMN_NAM USE VIR HID IDE  COLUMN_ID SEGMENT_COLUMN_ID INTERNAL_COLUMN_ID
---------- --- --- --- --- ---------- ----------------- ------------------
X7         YES NO  NO  NO           4                 4                  6
X6         YES NO  NO  YES          3                 3                  5
X5         YES YES YES NO                                                4
X3         YES YES NO  NO           2                                    3
X2         YES NO  NO  NO           5                 2                  2
X1         YES NO  NO  NO           1                 1                  1
```

The COLUMN_ID got reordered after the unused column was dropped (X4 was dropped; hence X6 and X7 got new column IDs—X5 never had a column ID assigned as it is INVISIBLE). The internal column ID, which includes the ID for INVISIBLE columns, also got adjusted. After the column was dropped, the INTERNAL_COLUMN_ID also got adjusted.

Tip: By default SQL*Plus DESCRIBE will not show the invisible columns in a table. If you want to see the invisible columns in DESCRIBE, use SET COLINVISIBLE ON.

## The LREG Process in 12*c*

Oracle Database 12*c* introduced a process named LREG—Listener Registration. The LREG registers information about the database instance and dispatcher processes with the Oracle Net Listener. When an instance starts, LREG polls the listener to determine whether it is running. If the listener is running, then LREG passes it relevant parameters. If it is not running, then LREG periodically attempts to contact it. In releases before

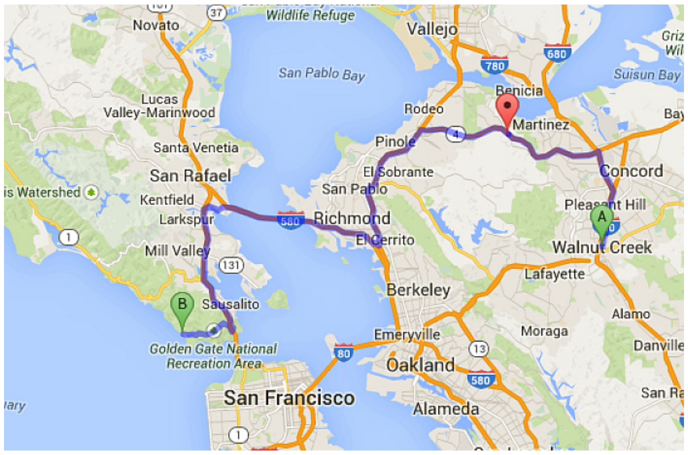
*(continued on page 26)*

# Shortest Isn't Always Fastest

by Tim Gorman

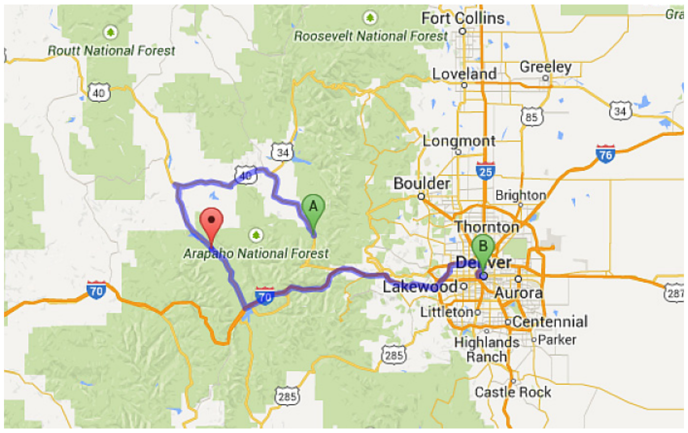
Tim Gorman

***Editor's Note:** If you lived in Walnut Creek in the East Bay and wanted to take the day off and go to Rodeo Beach in the Marin Headlands to search for carnelian gemstones like the one on the front cover of the previous issue of the* NoCOUG Journal*, you probably would not want to take the most direct route at the same time that everybody else is driving to work. The most direct route takes you through the Caldecott Tunnel, over the Bay Bridge, and across San Francisco city streets. Instead, your best bet is to go north toward Martinez, take CA-4 west, I-80 west, I-580 west, and, finally, US-101 south. That's 50% more miles, but you will assuredly get to the beach much faster than if you shared the road with all the poor souls trying to get to work.*


Walnut Creek, CA to Rodeo Beach

It's a problem many of us in Colorado have encountered: we want to drive from Winter Park to Denver. Easy: go over Berthoud Pass, down to I-70, and take I-70 east all the way in Denver. Should only take an hour and a half, tops. Except when Berthoud Pass has been closed due to winter weather. What then? Do we start heading east on US Highway 40 until the snow becomes higher than the windshield? Don't laugh: plenty of people would do essentially just this.

Or, would you turn the other way, head west on US Highway 40 all the way to Kremmling, then follow Highway 9 down to Silverthorne to I-70, through the Eisenhower Tunnel, and on into Denver. Yes, this route covers three times as many miles, but you get to Denver faster. It's not a great choice, but the first step is the most important: turning in the opposite direction and going that way as fast as you can. it's counterintuitive, but sometimes it works.


Winter Grove, CO to Denver

When loading large volumes of data into an Oracle database, the same thing can be true. Let's suppose you have 100,000 rows of data to load into a slowly changing dimension table in a data warehouse in an Oracle database. Of those 100,000 rows, 50,000 are new, so they need to be inserted. The other 50,000 rows already have older copies in the table, so those rows will need to be updated with the new data. Often, this is called an "upsert" or "merge" load, a combination of INSERTs and UPDATEs. Most dimension tables have new rows inserted or modified rarely, if at all. However, there are often dimensions that change a little (or a lot) with every load cycle. Typically, these dimensions have something to do with people, such as a dimension for employees or financial accounts or members. While there are a number of ways to merge this data, the most straightforward is to use the Oracle MERGE command, similar to the following . . .

```
MERGE into CURR_ACCT_DIM D
USING (select acct_key from EXT_ACCT_DIM) X
ON (d.acct_key = x.acct_key)
WHEN MATCHED then update
set d.eff_dt = x.eff_dt,
d.attr_01 = x.attr_01, ..., d.attr_99 = x.attr_99
WHEN NOT MATCHED then insert
(d.acct_key, d.eff_dt, d.attr_01, ..., d.attr_99)
values
(x.acct_key, x.eff_dt, x.attr_01, ..., x.attr_99);
```

This does the job, but like the story about driving to Denver over Berthoud Pass, sometimes the most direct and straightforward route is not the fastest way to the destination, especially when a large number of rows are being merged—and most particularly when a large number of rows are being updated.

In many ways, INSERTS are the fastest and most efficient type of data modification in any database, Oracle or otherwise.

- With an INSERT, there is no "search" phase to the operation, just placement of the row(s) into database blocks. From the perspective of transaction logging, not much redo (comparatively) is generated, just the "after image" of the newly inserted row.
- With a DELETE, there is a little more to do. There is a "search" phase to find the row, then remove the row. From the perspective of transaction logging, not much redo is generated, just the "before image" of the newly deleted row from the UNDO segment.
- With an UPDATE, there is a lot to do. There is a "search" phase to find the row, then there is the matter of changing data values in some or all of the columns. If the row grows in size and can no longer fit within the database block, then it must be migrated (similar to "chaining"). And, from the perspective of transaction logging, there is a lot of redo generated, both the "before image" prior to the row being changed, stored within UNDO, and the "after image" after the row has been updated.

Compared to DELETEs and UPDATEs, INSERT operations are a breeze. And they scale very well when run in parallel. There is even a special version of INSERT operations, called "direct path," used for large volumes of data being inserted.

And, as it turns out, INSERTs are the very fastest way to accomplish a boatload of DELETEs or UPDATEs, if you're fortunate enough to be using table partitioning.

Here's how:

First, the easy part—the very fastest DELETE is an INSERT.

For each of the partitions in the table from which rows will be DELETEd:

1. Create a "scratch" table that has the same columns and physical storage attributes as the partition, using a CREATE TABLE . . . AS SELECT (CTAS) statement that queries the partition from which rows will be removed. The WHERE clause of the SELECT statement in the CTAS is the reverse condition of a DELETE statement, finding the rows that will be retained.
    - The INSERT portion and the SELECT portion of the CTAS statement can both be run in PARALLEL.
    - The INSERT portion can use the "direct-path" (a.k.a. APPEND) load mechanism in the CTAS statement.
    - The NOLOGGING clause can be used in the CREATE attributes of the CTAS statement.
    - The COMPRESS clause can be used in the CREATE attributes of the CTAS statement.
2. Perform ALTER TABLE . . . EXCHANGE PARTITION between the original partition and the newly created "scratch" table.
    - As long as there are no global indexes, the EXCHANGE PARTITION operation is practically instantaneous.
3. Drop the "scratch" table.
4. Repeat for all partitions involved.

Fast, easy, and scalable. This is the way to delete hundreds of thousands or millions of rows from a table.

But UPDATEs aren't so easy. So I'm going to set up a scenario to help explain how it can be done. We have a dimension table named CURR_ACCT_DIM that is loaded into nightly. During these nightly loads, 50,000 rows are typically inserted and 50,000 rows are typically updated. The CURR_ACCT_DIM table has one unique B*Tree index supporting the primary key, a column named ACCT_KEY, and 30 single-column bitmap indexes. When the MERGE statement shown above was tested against this table, it took about 8 hours to complete, which is about three times too long.

## What can be done?

Assuming that there is an external table (based on a "flat" file) named EXT_ACCT_DIM with the 100,000 rows of data to be upserted or merged into the CURR_ACCT_DIM table, here is how we do it:

1. Range-partition the CURR_ACCT_DIM table with one (and only one) partition (called PDUMMY).
    - The intent of doing this is only to use EXCHANGEPARTITION during data loading, and not for partition pruning during queries.
    - Create a non-partitioned "scratch" table (named SCRATCH_CURR_ACCT_DIM) that is "shaped" exactly like CURR_ACCT_DIM.

```
create table SCRATCH_CURR_ACCT_DIM parallel nologging compress as
select acct_key, eff_dt, attr_01, ..., attr_99
from (
  select acct_key, eff_dt, attr_01, ..., attr_99,
    row_number() over (partition by acct_key order by eff_dt desc) rn
  from (
    select acct_key, eff_dt, attr_01, ..., attr_99 from CURR_ACCT_DIM
    union all
    select acct_key, eff_dt, attr_01, ..., attr_99 from EXT_ACCT_DIM
  )
)
where rn = 1;
```

    - The innermost subquery is a UNION-ALL that queries all 100,000 of the rows from the EXT_ACCT_DIM and the existing contents of the CURR_ACCT_DIM. If CURR_ACCT_DIM had 10m rows already, then the output from this innermost subquery should be 10.1m rows (i.e., 10m plus 100k rows).
    - The middle subquery exists only to apply the ROW_NUMBER() window function to the results of the innermost subquery, so that we can easily identify the latest EFF_DT value for each ACCT_KEY value.
    - The outermost subquery filters out any rows except the ones with the latest EFF_DT for each ACCT_KEY value (i.e. "RN = 1") and inserts the remaining rows into the SCRATCH_CURR_ACCT_DIM table.
3. Gather table- and column-level statistics on SCRATCH_CURR_ACCT_DIM.
4. Create all LOCAL indexes that exist on CURR_ACCT_DIM onto SCRATCH_CURR_ACCT_DIM with the COMPUTE STATISTICS clause.

*(continued on page 26)*

# Advice for an Oracle Beginner

**by Tom Kyte**

Tom Kyte

*(Previously published in the February 2011 issue of the* NoCOUG Journal*)*

When I first started out in IT, I had no experience, no training, no background whatsoever. I was a math major fresh out of college in the year 1987. I hadn't taken any computer courses beyond the initial "introduction to" type of classes. It wasn't until I got a job as a computer programmer—advertised as "no experience required"—that I started even really using computers.

So, given that I had no experience, no real formal training—how did I get started, how did I get to where I am today? I think it comes down to two simple words: mentorship and participation.

When I first started out as an entry-level programmer, I had an excellent mentor. This was probably the key difference between success and failure for me. My mentor—who back then was about the age I am now (that is, he was old)—took the time to teach me the ropes. He taught me the right way to do things—not the fast way, not the "shortcuts," not the checklist of things to do—but the right way. In many cases, the right way isn't the easy way, isn't the quickest way . . . but it is ultimately the best way. He taught me many things I myself teach these days. Simple things such as "make your subroutine fit on a screen, you have to see it all," "instrument your code to death," "write as little code as you can but as much as you have to," "code defensively; don't trust anyone else to just know what to do with your code," and "test, test, test, benchmark and test again." My mentor made me the programmer I am.

The second item—participation—is what propelled me beyond being just a programmer. In the early 1990s, I started participating in online forums hosted on Usenet. For those who never heard of it, "Usenet" was Twitter, blogging, Facebook—any social network goes here—before any of them were invented. Usenet consisted of a relatively small (by today's standards) group of individuals that would discuss topics of interest to them. My interest was, of course, all things Oracle, and discuss we did. I "met" in a virtual sense on those discussion forums many people I still correspond with and interact with face to face. I learned a lot from them—and they (hopefully!) learned a thing or two from me. It was on these forums that I found the answers to many of my questions—and formulated answers to questions from others. This give-and-take allowed my knowledge of Oracle, programming, and databases in general to expand and grow immeasurably. Participation in the Oracle community is what took

*"Given that I had no experience, no real formal training—how did I get started, how did I get to where I am today? I think it comes down to two simple words: mentorship and participation."*

me from being just another programmer to being "AskTom." It gave me the confidence to write my first book in the year 2000: Expert One on One Oracle. It also gave me the audience for such a book. Without the act of participating, I do not think I would be where I am today.

So, in short, find a mentor. This is crucial. Find someone that you trust, that you respect, that others trust and respect. Learn from your mentor. Then, start participating. Participate in your local user group. Get up in front of an audience and present on some technical topic. Attend conferences. Get active in a discussion forum, such as those on **http://otn.oracle.com**. Don't be afraid to make mistakes (you will; I did), but make sure to learn from them. That would be my advice. ▲

*Tom Kyte is a Senior Technical Architect in Oracle's Server Technology Division. Before starting at Oracle, Tom worked as a systems integrator building large-scale heterogeneous databases and applications, mostly for military and government customers. Tom spends a great deal of time working with the Oracle database and, more specifically, working with people who are working with the Oracle database. In addition, he is the Tom behind the "AskTom" column in Oracle Magazine, answering people's questions about the Oracle database and its tools (***http://asktom.oracle.com***). Tom is also the author of Expert Oracle Database Architecture (Apress, 2005), Expert One on One Oracle (Wrox Press, 2001/Apress 2004), Beginning Oracle Programing (Wrox Press, 2002/Apress 2004), and Effective Oracle by Design (Oracle Press, 2003). These are books about the general use of the database and how to develop successful Oracle applications.*

**If your member ID is 6249118, you have won a copy of *Becoming Steve Jobs: The Evolution of a Reckless Upstart into a Visionary Leader*. Collect your prize at the spring conference on May 21 at UCSC Extension Silicon Valley.**

# How Romeo Won the Heart of Juliet

## Fourth International NoCOUG SQL Challenge

Once upon a time, Romeo, the son of Montague, told his cousin Benvolio that he was in love with Rosaline but she was not returning his affections. Benvolio sang a song by the great American songwriter Stephen Foster:

"There are plenty of fish in the sea
As good as ever were caught."

Meanwhile, Count Paris, a relative of Prince Escalus, asked for the hand of Juliet, daughter of Capulet, in marriage. Capulet organized a grand feast and invited Count Paris. Juliet agreed to talk to Count Paris at the feast. Benvolio suggested that Romeo gatecrash the feast so that Romeo could meet other women. Romeo agreed, but only because Rosaline would also be at the feast. At the feast, Romeo instantly fell in love with Juliet and completely forgot about Rosaline. Romeo then sang another song by maestro Stephen Foster.

"I dream of Juliet with the light brown hair,
Borne, like a vapor, on the summer air;
I see her tripping where the bright streams play,
Happy as the daisies that dance on her way."

Count Paris and Romeo both wanted to know when Juliet's birthday was, so that they could send her an edible arrangement. Juliet connected to an Oracle In-Memory pluggable RAC database and typed the following SQL statements:

```
CREATE TABLE Dates (DateOfBirth DATE NOT NULL PRIMARY KEY);
GRANT SELECT ON Dates TO Paris;
GRANT SELECT ON Dates TO Romeo;
```

Juliet then inserted a number of values into the Dates table, only one of which was her real date of birth. She then took Romeo aside and whispered the *month* of her date of birth into his ear. She then took Count Paris aside and whispered the *day* of her date of birth into his ear.

Romeo declared dejectedly: *"I don't know when Juliet's birthday is but, thank heavens, Count Paris doesn't know either."*

Count Paris exclaimed excitedly: *"Now I do!"*

Not to be outdone, Romeo exclaimed: *"Now I do too!"*

Count Paris whipped out his Samsung Galaxy S6 and ordered a dozen gourmet dipped swizzled strawberries from **ediblearrangements.com**. It was too late for same-day delivery, so he ordered next-day delivery along with a card saying "Better Late Than Never!" Juliet wasn't thrilled but politely thanked Count Paris.

Romeo did not have a smartphone but he had a bike and, having worked as a bike messenger, he knew the streets of Verona like the back of his hand. He pedaled furiously to the Edible Arrangements store on the next block where he bought a dozen gourmet dipped swizzled strawberries and brought them back to Juliet. Juliet was thrilled because, more than anything, she loved to receive gourmet dipped swizzled strawberries on her birthday.

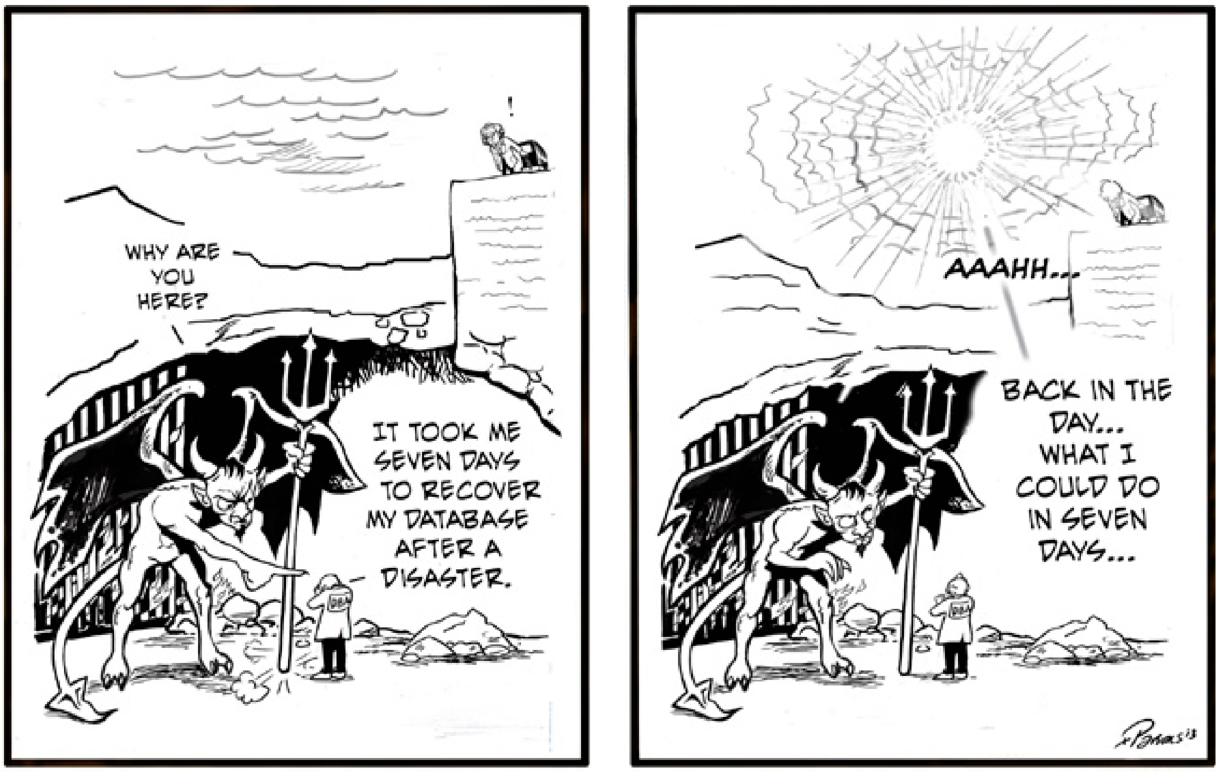
And that's how Romeo won the heart of Juliet. Our story ends with Count Paris singing another verse from Benvolio's song:

"There are plenty of fish in the sea
But, oh, they're hard to be caught."

The obvious question remains: what was the day of the feast? Formulate your answer as an ANSI-standard SQL query of minimum length and attach it as a comment to the challenge announcement on the NoCOUG blog **nocoug.wordpress.com**. Judging criteria will include correctness, originality, efficiency, portability, readability, and order of receipt. The winner will receive an Apple Watch Sport or gift card of equal value. The August Order of the Wooden Pretzel will be conferred on the winner, in keeping with the celebrated comment by Steven Feuerstein in the August 2006 issue of the *NoCOUG Journal: "Some people can perform seeming miracles with straight SQL, but the statements can end up looking like pretzels created by someone who is experimenting with hallucinogens."* The current knights of the August Order of the Wooden Pretzel—Alberto Dell'Era (Italy), Andre Araujo (Australia), Rob van Wijk (Netherlands), Ilya Chuhnakov (Russia), and Lukasz Pluta (Poland)—are not eligible to participate. Additional prizes may be awarded at the discretion of NoCOUG. The contest will close at a time of NoCOUG's choosing. NoCOUG and the judges reserve the right to publish and comment on any of the submissions with due credit to the originators. NoCOUG reserves the complete right to clarify, interpret, or modify the contest rules at any time. NoCOUG's decisions are final. ▲

## Dr. DR

by Rich Parsons



!
WHY ARE YOU HERE?
IT TOOK ME SEVEN DAYS TO RECOVER MY DATABASE AFTER A DISASTER.
AAAHH...
BACK IN THE DAY... WHAT I COULD DO IN SEVEN DAYS...

Dr. DR is brought to you by Axxana.

**Medical tools have come a long way since the early days. Have your database tools?**

Embarcadero® DB PowerStudio.® Everything you need to manage, develop, and optimize your databases from a single interface, no matter how many DBMSs you throw at it.

- DBArtisan®
- Rapid SQL®
- DB Optimizer®
- DB Change Manager®

**Download your free trial today**
www.embarcadero.com/downloads

©2014 Embarcadero Technologies, Inc. embarcadero®

EMBARCADERO.COM | SALES 415.834.3131 | CUSTOMER SUPPORT 877.783.5332

DELPHIX®

Database Virtualization Software

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

www.delphix.com

# NoCOUG Spring Conference

## Session Descriptions

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

### –Keynote–

**Database Technology Trends**—*Sehmuz Bayhan, eBay* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9:30–10:30

This keynote covers how databases have evolved over the past 30 years and how Big Data and the cloud are revolutionizing the database industry.

*Sehmuz Bayhan is the senior director of database infrastructure at eBay.*

### –Room 106–

**Introduction to Query Performance Tuning: A 12-Step Program, Part I**—*Janis Griffin, SolarWinds* . . . . . . . .11:00–12:00

Performance tuning can be complex. It's often hard to know which knob to turn or button to press that will give you the biggest performance boost. The presentation will detail 12 steps to quickly identify performance issues and fix them. It will show how to quickly identify performance inhibitors in order to avoid poor performance in the future. Finally, the participant will be able to identify and understand how new Oracle features can change and/or support different execution plans.

**Introduction to Query Performance Tuning: A 12-Step Program, Part II**—*Janis Griffin, SolarWinds* . . . . . . . . 1:00–2:00

**The Scaling Database Saga at Intuit: Applying New Learnings and Techniques For Infinite Scale**
—*Tushar Thakker, Intuit*. . . . . . . . . . . . . . . . . . . . . . . . . . .2:30–3:30

Scaling a database on any platform is one of the most difficult challenges in today's environment. This is especially true when the database needs relational features. There are many architectural patterns that have been conceived and tried over the years to solve the problem of database scale. For example, architects have used replication patterns, but with this approach there is full downtime, up to 100%, while your primary database switches to the fail-over database. Another option is to use "beefier" machines to host your database, but this approach presents various hardware limits with CPU and memory, and it can be an expensive route. A third possibility is to have a cluster of database servers (e.g., Oracle RAC), with each server automatically cooperating with the others. But this option has its own set of issues, limits, and complexities. In this session we will learn how Intuit is using a new approach utilizing a swimlane architecture along with SOA to scale our core offering with the intended outcome of being able to scale infinitely.

*Tushar Thakker is the chief data architect in the Small Business Group at Intuit.*

### –Room 107–

**RAC Cache Fusion Internals**—*Riyaj Shamsudeen* . .11:00–12:00

Have you ever wondered how Oracle RAC maintains consistency, considering that a block can be modified in any instance buffer cache? How does the transaction locking mechanism work concurrently with the RAC consistency mechanism? Come to this session to learn intricate details about various cache fusion internals. I will do a live demo of a three-node RAC cluster and show various internals about cache fusion. Understanding the internals is essential to appreciating the underlying RAC wait events. I will map the internals to the related RAC wait events to improve practical use of the internals in your day-to-day life. Further, I will use Swingbench to show the importance of log I/O throughput in a RAC database.

**In-memory Internals**
—*Riyaj Shamsudeen, Oracle ACE Director* . . . . . . . . . . . 1:00–2:00

In this session I will present internal details about in-memory options, processes, and memory structures; v$ views; and some of the practical issues that we encountered while testing a multi-terabyte database with this product.

I will also discuss a few internal troubleshooting events to understand how IM is populated and scanned, and so on. In addition, we will take heap dumps to review memory structures.

**Health Checks and Diagnosability for the Oracle Cloud**
—*Sandesh Rao, Oracle Corporation* . . . . . . . . . . . . . . . . .2:30–3:30

Cloud environments require heavy emphasis on standardization and automation while implementing many components of the Oracle stack. Invariably, when problems arise Oracle Support will need the correct first failure diagnostics in order to provide timely and correct solutions. Oracle has provided tools like ORAchk, Collection Manager, and Oracle Trace File Analyzer (TFA) Collector to address these requirements. ORAchk is an automation framework of proactive health checks for a growing number of components of the Oracle technology stack. Collection Manager provides a central repository and dashboard that tie together collections from across the enterprise. TFA Collector automates the process of efficiently gathering first failure diagnostics.

*Sandesh Rao is the Senior Director, RAC Assurance at Oracle Corporation.*

### –Room 116–

MongoDB 101—*Ahbaid Gaffoor, eBay*. . . . . . . . . . . . .11:00–12:00

In this session we will introduce MongoDB to the Oracle DBA. We will cover installation, running a mongo database, and basic data modeling and CRUD operations. We will then conclude with a discussion of replication and sharing options. Beginners are welcome; having some knowledge of JSON is helpful but not required.

*(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
- Embarcadero Technologies
- SolarWinds

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

## TREASURER'S REPORT

Eric Hutchinson, *Treasurer*

| | | |
|---|---|---|
| ***Beginning Balance*** | | |
| January 1, 2014 | | **$ 56,867.13** |
| **Revenue** | | |
| Individual Membership | 13,625.00 | |
| Gold Vendor Fees | 12,000.00 | |
| Corporate Membership | 11,000.00 | |
| Silver Vendor Fees | 3,000.00 | |
| Conference Sponsorships | 2,500.00 | |
| Conference Walk-in Fees | 2,449.12 | |
| Training Day Receipts | 3,430.03 | |
| Journal Advertising | 3,750.00 | |
| Interest | 4.97 | |
| **Total Revenue** | | **$ 51,759.12** |
| **Expenses** | | |
| Conference Expenses | 29,851.47 | |
| Journal Expenses | 14,927.51 | |
| Training Day Expenses | 2,211.43 | |
| Board Expenses | 1,931.53 | |
| PayPal Expenses | 1,395.59 | |
| Software Dues | 857.60 | |
| Insurance | 581.00 | |
| Office Expenses | 197.47 | |
| Meetup Expenses | 160.63 | |
| Taxes and Filing | 0.00 | |
| Marketing Expenses | 0.00 | |
| **Total Expenses** | | **$ 52,114.23** |
| ***Ending Balance*** | | |
| December 31, 2014 | | **$ 56,512.02** |

*(continued from page 5)*

back about my presentations—what worked, what didn't. I recently got book reviews via Twitter!

***Do you have a life outside work? What are you passionate about other than big data?***

In the last year a lot of my free time was spent writing a book. I really want my weekends back!

When I'm not writing a book, I like the outdoors. California has amazing nature and I can't get enough. I hike, bike, run, climb, and camp. Later this year, my husband and I are travelling to Peru to hike in the Andes. I'm super-excited about this already! ▲

*Gwen Shapira is a Solutions Architect at Cloudera and leader of the IOUG Big Data SIG. She studied computer science, statistics, and operations research at the University of Tel Aviv, and then went on to spend the next 15 years in different technical positions in the IT industry. She specializes in scalable and resilient solutions and helps her customers build high-performance large-scale data architectures using Hadoop. Shapira is a frequent presenter at conferences and regularly publishes articles in technical magazines and her blog.*

*(continued from page 13)*

Here our story pauses but I'm sure that it will resume on a future date because—if history is any indication—Oracle Corporation will continue to surprise and delight us with new optimizer capabilities for many years to come. ▲

*Janis Griffin is an Oracle ACE and a database performance evangelist at SolarWinds, the maker of Database Performance Analyzer (formerly Confio Ignite) for Oracle. Information on Database Performance Analyzer can be found at* **www.solarwinds.com/dpa-oracle**.

*(continued from page 18)*

Oracle Database 12*c*, PMON performed the listener registration. In the pluggable database scenario, the PMON process cannot do its primary job of process recovery and buffer cache cleanup if it also has to register all those services for the many PDBs and keep track of connections and load balance them (server side load balancing in RAC). Hence the birth of the LREG process. ▲

*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**.

*(continued from page 20)*

5. Finally, use "alter table CURR_ACCT_DIM exchange partition PDUMMY with table SCRATCH_CURR_ACCT_DIM" to publish the new contents of dimension table for end users.

In this way the UPDATE/MERGE can be performed with only INSERT/SELECT statements, and the resulting table can be compressed using Oracle9*i* "basic" compression or, if on HCC-enabled storage, compressed using one of the HCC methods instead, thanks to the user of direct-path APPEND inserts. If need be, the NOLOGGING attribute can be set on the "scratch" table so that the direct-path APPEND loads don't generate redo, with the normal caveats and cautions when doing so.

Sometimes the fastest way to Denver is through Kremmling. And sometimes the fastest DELETE or UPDATE is an INSERT. ▲

*Tim Gorman is a technical consultant for Delphix, who enable database and storage virtualization to increase the agility of IT development and testing operations. He has co-authored six books, tech-reviewed eight more, and written articles for RMOUG SQL_Update and IOUG SELECT magazines. He is an Oracle ACE Director, a member of the Oak Table Network, and has presented at Oracle OpenWorld, Collaborate, KScope, Hotsos, and local Oracle users groups all around the world. Tim lives in Westminster, Colorado, with his wife, Kellyn, and their children.*

*(continued from page 24)*

**Wresting control of your Oracle data with Heat Map and ILM in Oracle DB 12*c*—***John Kanagaraj, eBay*. . . . . . . . . . . .1:00–2:00

Oracle Database 12*c* introduced the Heat Map as well as various partitioning and compression enhancements that can help you wrest back control of your data. In this session, you will learn how this works and how to implement Information Lifecycle Management (ILM) for Oracle-based databases to take charge of data growth.

**Tips for DBAs and Developers—***Govindan Katteri, Pearson School Systems*. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2:30–3:30

Of utmost importance when writing code is knowledge of the caveats and finer aspects of the database or language one is dealing with. While the manual might describe a feature, it does not explain how best to use it, when to use it, critical aspects of the feature, and how to mix and match features. This session describes a number of hidden gems for Oracle DBAs and developers. ▲

# 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

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

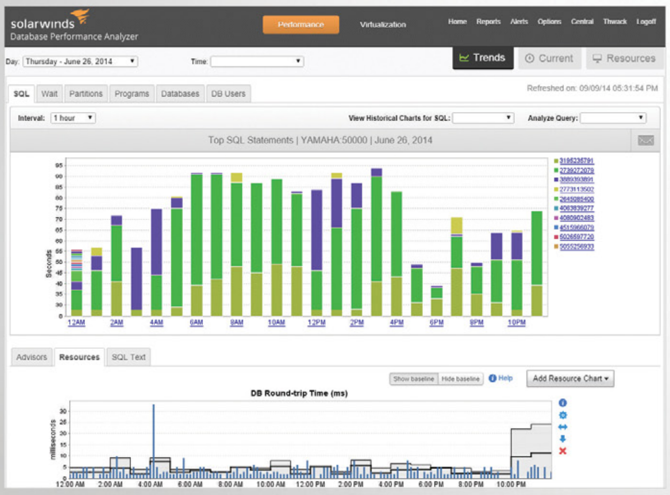
**NoCOUG**
P.O. Box 3282
Danville, CA 94526

**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***