Sharon combines her technology background with her Regis degree to help empower her own community. How will you be influential?

The School of Computer & Information Sciences in Regis University’s College for Professional Studies offers four undergraduate and six graduate degrees that immerse students in the latest technologies and software. Students benefit from our state-of-the-art learning environments, experienced faculty and strong industry relationships.

Busy adults choose Regis University for the academic rigor and flexibility.

- Regionally accredited and nationally ranked
- Accelerated 5- and 8-week classes
- Online and campus-based learning
- Multiple start dates throughout the year

> CLASSES START SOON

Focused on Delivering the right Solution for the right Project

Arisant focuses on understanding business requirements to ensure the most effective implementation of the right solution. Our strategy is aimed at providing an honest and expert brand of consulting services for both the Private and Public Sectors. Contact us today to discuss any of the following services, and more.

- Consulting Services
  - Architecture
  - Analysis
  - Design
  - Implementation
  - Project Management
- Staffing/Support
- Managed Services
- Managed Hosting Services
- Oracle Software Sales
- Oracle Hardware Sales
EDITOR & DIRECTOR
Pat Van Buskirk
NewsletterDir@rmoug.org

SUBSCRIPTIONS & TECHNICAL ADVISOR
Heidi Kuhn
admin@rmoug.org

CONTRIBUTING WRITERS
Michael S. Abbey
Stelios Charalambides
Jonathan Gebnick
  Dan Hotka
  Jeffrey Jacobs
  Kellyn Pot’Vin
  Carlos Sierra
  Stefan Zielinski

SQL>Update is published quarterly by
Rocky Mountain Oracle Users Group
5409 S. Union Court
Littleton, CO 80127
(303) 929-6266
Email: admin@rmoug.org
www.rmoug.org

Please submit advertising and editorial ma-
terial to the NewsletterDir@rmoug.org. RMOUG
reserves the right to edit, revise or reject all ma-
terial submitted for publication. Liability for errors
in ads will not be greater than the cost of the
advertising space. ©2013 Rocky Mountain Oracle
Users Group

monthly features

4 Letter From Our President
4 Stan Yellott Scholarship Fund
24 OBIEE Tips & Techniques
  Dan Hotka
26 RMOUG Board Focus
  Kellyn Pot’Vin
28 RMOUG Board of Directors
30 RMOUG Calendar of Events
31 August 2013 QEW

A special Thank You to Heidi Kuhn, Tim
Gorman and Kellyn Pot’Vin, without
whose continuing help, this publication
would not be possible.

features

6 An Average Article
  by Jonathan Gennick
  A Lot More To Statistics

10 Can We Simplify The Process
  of SQL Tuning?
  by Stelios Charalambides and Carlos Sierra
  SQLXPLAIN (SQLT) To The Rescue

14 Slicing Tables With Oracle Partitioning
  by Stefan Zielinski
  The Best Thing Since Sliced Bread

16 Application Development With Oracle
  by Jeffrey Jacobs
  Advanced Queuing

20 Human Reliability
  by Michael S. Abbey
  Learning Lessons The Hard Way

On the Cover:
Stefan Zielinski is an Oracle DBA (since 2005) and a long time C and PL/SQL developer. He currently
works for Dell Services (formerly Perot Systems). The cover photo of Maroon Bells was taken on a family trip
to Aspen in September of 2012. Stefan enjoys travelling and hiking in his spare time.
RMUOG was formed in 1984 by a small group of Oracle professionals, including Rik Williams, Richard Cross, Pete Watson, and LeeAnn and Bob Suehrstedt. This year marks the 30th year for the organization, and of course a lot has happened over those years, and there is a lot planned for the coming year.

The new board of directors for the coming year 2013-2014 has been elected and seated as follows...

- **Executive Director**: Heidi Kuhn
- **Emeritus board members**: Kathy Robb, John Jeunnette
- **Past president**: Peggy King
- **Member-At-Large**: Sruthi (Kumar) Annamnidu
- **Director of Scholarships**: Dr. Robert Mason
- **Director of Membership**: Mark James
- **Director of Social Media**: Bobby Curtis
- **Director of Education**: Dan Hotka
- **Director of Special Interest Groups (SIGs)**: Vincent Giasolli
- **Hyperion/EPM SIG**: Chris Chase
- **MySQL SIG**: George Trujillo
- **Women In Technology (WIT)**: Kellyn Pot’vin
- **Webmaster**: Art Marshall
- **Director of Vendor Relations**: Chris Ostrowski
- **Director of Newsletter**: Pat Van Buskirk
- **Director of Training Days**: Kellyn Pot’vin
- **Director of Programs**: Carolyn Frye
- **Secretary**: Ron Bich
- **Treasurer**: Tom Green
- **Vice President**: John Peterson
- **President**: Tim Gorman

Mr. Annamnidu, Dr. Mason, Mr. James, Mr. Curtis, Mr. Hotka, and Mr. Giasolli are all serving their first year on the board of directors, so please congratulate them and wish them well if you get the chance!

Over the years, during the normal course of everyone’s life or career, people move nearer or move farther. Some of the people on this board have recently moved away to other states, namely Peggy King to Arizona and John Jeunnette to New Hampshire, but we value them and they are kind and generous enough to continue helping RMOUG and the Oracle community in Colorado. We also have some people who have recently moved into Colorado, or *back* into Colorado, namely Vincent Giasolli from Hawaii.

Because RMOUG also has members and long-time contributors around the country and around the world, and because technology has made the world a little bit smaller, we are also experimenting with board members who live in other states, namely Bobby Curtis from Georgia and Dan Hotka from Iowa. While there are some roles on the board that such out-of-state members could not fulfill, we believe that we have found ideal roles which can be performed from any location on the planet.

The other members of the board have served many years with RMOUG and their names are familiar, but each year of service is a choice and not an obligation, so I’m personally incredibly grateful for the inspiration and dedication these professionals and all-round good folks bring.

Please welcome them and congratulate them when you get the chance, as well!

So now, let’s get down to business...

What are some of the things we can look forward to during the coming year?

- **DBLab Meetups**
  Chris Chase and the Hyperion/EPM special interest group are especially active, holding meetups at Regis University, the Oracle campus in south Denver (DTC), and at Denver University. The MySQL SIG is also interested in setting up more free evening labs for students and professionals as well. Those interested in these sessions, please sign up at http://www.meetup.com/RMOUGLabs/.

- **Summer Quarterly Educational Workshop (QEW) at Elitch Gardens on Friday August 9**
  A great idea becomes even better! Last year, RMOUG held its summer QEW at Elitch Gardens, presenting professional seminars in the morning and then freeing up members and guests to enjoy the theme/water park with their families throughout the rest of the day. This year, with sponsorship from Oracle Corporation, we’ll be having a “Technology Day” and lunch from 8:00am until 2:00pm, and then members and their families will be free to enjoy the park for the rest of the day. This is a great way to take advantage of the best training opportunities that Oracle has to offer, and spend some quality time with colleagues, friends, and family. So mark your calendar for Friday August 9, 2013 at Elitch’s! For more information, watch the RMOUG website at http://www.RMOUG.org/

- **Women In Technology (WIT) sessions**
  RMOUG has had several WIT sessions at Training Days and at the Spring QEW at the Oracle/Sun campus in Broomfield in May, as well as separate evening sessions in casual coffee-house settings in Denver. These will continue, providing a supportive forum for women in technol-
ogy to discuss opportunities and obstacles and generally encourage positive ways of advancing theirs and others careers in this field. If you're interested in participating in this community, open to men as well as women, there is a group on Facebook at https://www.facebook.com/OracleWomenInTechnology. If you don't care to use Facebook, please contact Kellyn Pot'vin at dbakevlar@gmail.com. Come on and get involved, or let us know what would help you become involved!

- **Training Days 2014**

  One of the half-dozen most highly-regarded conferences for Oracle professionals throughout the world, the 25th annual RMOUG Training Days will be held from Wednesday through Friday, February 5-7, 2014 at the Colorado Convention Center in downtown Denver. Those of you who have already attended in past years know the astounding diversity and depth of the sessions, yet the conference also offers opportunities to mix and chat with colleagues and gurus alike. Opportunities like volunteering for the “Speaker Shuttle” from DIA to downtown during the days prior to the conference allow RMOUG members the opportunity to do a nice favor for an out-of-town guest, and perhaps have a half-hour to chat about a particular Oracle topic. The “ACE Lunches” will be expanded to both days of the conference, and a handy “table map” to find the table with your preferred Oracle ACEs and ACE Directors. The wildly popular Oracle Technology Network (OTN) sessions will be expanded throughout the conference, and Tom Kyte will be the keynote speaker. Watch the RMOUG website at http://www.RMOUG.org for more information, and don’t miss this amazing conference.

- **Stan Yellott Scholarship Fund**

  With the arrival of university faculty Dr Bob Mason and Mark James on the RMOUG board, we look forward to formalizing the scope and processes underlying the Stan Yellott memorial Scholarship Fund. Part of the mission of RMOUG is to encourage new entrants into the Colorado information technology industry, and the scholarships are a key part of that. RMOUG is also looking to encourage middle-school and high-school students to consider careers in information technology, so if you are interested in helping with these efforts, please contact Dr. Bob and Mr. James, whose contact information can be found on the RMOUG website at scholarshipdir@rmoug.org and membershipdir@rmoug.org.

There is so much more that we’ll be doing this year, but surely these will be the highlights. If you are interested in volunteering to help year-round at RMOUG, or at one of the RMOUG events, please contact our executive director, Heidi Kuhn at heidikuhn@rmoug.org and we’ll get you plugged in.

Thank you all so much, and let’s get to work!
An Average Article

by Jonathan Gennick

Many of us learned to compute averages at a young age. With
me it was in elementary school. Total a group of numbers. Divide by
their quantity. Write down the result and label it as “the average”.
Recently I’ve been revisiting the concept of average, finding a lot
more to it than what I’ve just described.

Measure of Central Tendency

Imagine you’re giving career advice to a group of students.
You might be at your local high-school’s Career Day, for example.
Wanting to give a sense of your good fortune from working in Oracle
Database over the past decade, you could say to the students:
“In 2003 I received a 10% raise, in 2004 a 4% raise, in 2005 a
12% raise, in 2006 an 8% raise, in 2007 a 9% raise, in 2008 a 4%
raise, in 2009 a 1.5% raise, in 2010 a 14.5% raise, in 2011 an 11.5%
raise, and in 2012 an 8.5% raise. Compare that to my colleague
in SQL Server who received an 11% raise in 2003, a 1.5% raise in
2004, a 9.5% raise 2005, a 10.5% raise in 2006, a 10% raise in 2007,
a 1% raise in 2008, a 3% raise in 2009, an 11.5% raise in 2010, a
13% raise in 2011, and an 8.5% raise in 2012.”

Of course, this is incomprehensible. The kids’ eyes will glaze
over and they will all move to become Java developers instead.

On the other hand, you might hold their attention if you
reduce the gibberish to just a couple of numbers giving the ten-
dency of each group. For example:
“During the past decade my average yearly raise was 8.3%
versus the 7.5% average raise of my SQL Server colleague.”

In this way, averages allow us to rise above the detail and
talk at a higher, more abstract level about what it is we are trying
to describe. We can communicate the big picture succinctly and
swiftly.

Averages in the Plural

There is more than one type of average. You can make differ-
ent choices depending upon what you’re trying to communicate and
the picture you’re trying to paint. The previous section paints a pic-
ture of salary increase over time. But what’s the typical raise like?
You can begin by counting the most common raise amount:
“My most common raise has been 4%.”

A 4% raise paints a pretty dismal picture and that Java devel-
oper career is starting to look more appealing. Maybe it’s better to
talk in terms of a mid-point:
“I have about the same number of raises above 8.75% as below
it.

So far you’ve seen three different averages from the same data,
and there are more! Choosing the right average depends a great
deal upon the message you’re intending to convey.

Arithmetic Mean

The arithmetic mean is commonly the first average we learn,
and it’s the first one SQL “learned” as well. It’s implemented via the
AVG function, though perhaps we’d be better off had the language
designers chosen the name MEAN instead.

I like to think about the mean as being appropriate when the
following are true:
• The values in question are additive
• The math needs to work out accounting-style

Now this isn’t the complete picture. Any statistician will tell
you that I’ve over-simplified with this heuristic, but it’s a reason-
able starting point and leaves you on safe ground.

A good example is the reporting of sales data. Following are
ten days of sales from a fictional gas-station / convenience-store:

```
SELECT *
FROM daily_sales
WHERE daily_date >= DATE '2013-01-01'
  AND daily_date < DATE '2013-01-11'
ORDER BY daily_date;
```

<table>
<thead>
<tr>
<th>DAILY_DATE</th>
<th>SALES_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-JAN-13</td>
<td>17032</td>
</tr>
<tr>
<td>02-JAN-13</td>
<td>15043</td>
</tr>
<tr>
<td>03-JAN-13</td>
<td>16783</td>
</tr>
<tr>
<td>04-JAN-13</td>
<td>23343</td>
</tr>
<tr>
<td>05-JAN-13</td>
<td>25703</td>
</tr>
<tr>
<td>06-JAN-13</td>
<td>19538</td>
</tr>
<tr>
<td>07-JAN-13</td>
<td>18001</td>
</tr>
<tr>
<td>08-JAN-13</td>
<td>17154</td>
</tr>
<tr>
<td>09-JAN-13</td>
<td>26235</td>
</tr>
<tr>
<td>10-JAN-13</td>
<td>14983</td>
</tr>
</tbody>
</table>

Few high-ranking managers want to be bothered with this
level of detail. If you’re the local store manager and are summariz-
ing the first third of the month to your regional manager, you can
convey the big picture succinctly via the mean and the sum:

```
SELECT AVG(sales_amt), SUM(sales_amt)
FROM daily_sales
WHERE daily_date >= DATE '2013-01-01'
  AND daily_date < DATE '2013-01-11';
```

<table>
<thead>
<tr>
<th>AVG(SALES_AMT)</th>
<th>SUM(SALES_AMT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19381.50</td>
<td>193815</td>
</tr>
</tbody>
</table>
You can report that your mean sales are $19,381.50 per day, for a total of $193,815 so far in the month. The math works out too:

$19,381.50 \text{ per day} \times 10 \text{ days} = $193,815

Your revenue goal for the month is $500,000, and you can do some quick math to determine whether you’re on track to meet that goal. Divide the goal by the number of days in the month to obtain an average daily target:

$500,000 \div 31 \text{ days} = $16,129 \text{ per day}

Your current mean sales of $19,381.50 per day is comfortably above $16,129. You have earned more revenue so far in the month than your business plan requires, leaving some cushion in case of a few bad sales days.

Indeed! You will need that cushion. Your next three days show a precipitous drop in sales:

<table>
<thead>
<tr>
<th>DAILY_DATE</th>
<th>SALES_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-JAN-13</td>
<td>16010</td>
</tr>
<tr>
<td>12-JAN-13</td>
<td>4300</td>
</tr>
<tr>
<td>13-JAN-13</td>
<td>2874</td>
</tr>
</tbody>
</table>

Are you still on safe ground? Check your mean once more, to see the effect from the two bad days:

<table>
<thead>
<tr>
<th>DAILY_DATE</th>
<th>SALES_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-JAN-13</td>
<td>16010</td>
</tr>
<tr>
<td>12-JAN-13</td>
<td>4300</td>
</tr>
<tr>
<td>13-JAN-13</td>
<td>2874</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DAILY_DATE</th>
<th>SALES_AMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-JAN-13</td>
<td>16010</td>
</tr>
<tr>
<td>12-JAN-13</td>
<td>4300</td>
</tr>
<tr>
<td>13-JAN-13</td>
<td>2874</td>
</tr>
</tbody>
</table>

Your mean daily sales are now only $16,692.23, but are still above your target of $16,129. Some cushion remains, and there is plenty of time in the month for corrective action.

Note: Further analysis is possible, and even desirable. Computing a running average helps you monitor whether your mean daily sales are trending upward or downward. My article at http://gennick.com.stats/avg.html goes into some detail on doing that.

### Median

I like to think of median as the measure of central tendency most useful in describing the typical experience, or the typical case. The United States Census Bureau, for example, reports on median household income so as to paint a balanced picture not skewed too far in one direction or the other by people in extreme wealth, or in extreme poverty.

Computing the median is simple in concept, but can be memory and resource intensive as all the values must be known and compared against multiple times, and none discarded until the final result is obtained. Figure 1 illustrates the concept, which involves finding or computing a middle value having the same number of other values on either side.

Oracle Database is one of the few database engines to provide direct support for computing the median, and that is through a function appropriately named MEDIAN. With it, you can look at median sales over the first ten days of the month:

```
SELECT AVG(sales_amt) “Mean”,
       MEDIAN(sales_amt) “Median”,
       SUM(sales_amt) “Sum”
FROM daily_sales
WHERE daily_date >= DATE ‘2013-01-01’
AND daily_date < DATE ‘2013-01-11’;
```

<table>
<thead>
<tr>
<th>Mean</th>
<th>Median</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>19381.5</td>
<td>17577.5</td>
<td>193815</td>
</tr>
</tbody>
</table>

Then you can look again after the two days of very poor sales:

```
SELECT AVG(sales_amt) “Mean”,
       MEDIAN(sales_amt) “Median”,
       SUM(sales_amt) “Sum”
FROM daily_sales
WHERE daily_date >= DATE ‘2013-01-01’
AND daily_date < DATE ‘2013-01-14’;
```

<table>
<thead>
<tr>
<th>Mean</th>
<th>Median</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>19381.5</td>
<td>17577.5</td>
<td>193815</td>
</tr>
</tbody>
</table>

The median has not been affected a whole lot by the very low sales on the 12th and 13th. Median is less susceptible than the mean to being “pulled around” by outliers, and for that reason is termed by statisticians as a robust statistic.

The two medians from the preceding queries allow you to paint the following picture of how busy the employee running the cash register might reasonably expect to be on any given day:

“Half the time sales will be less than $17,000 - $17,500, and half the time they will be greater.”

This is of course an incomplete picture. Employing statistics like the median is an exercise in sacrificing detail for brevity. Median divides data into two buckets. If you wish to do so, you can paint a more detailed picture by using the PERCENTILE_CONT function to divide the data into three buckets. For example:

```
SELECT PERCENTILE_CONT(0.33) WITHIN GROUP
      (ORDER BY sales_amt ASC) “1st 3rd”,
   PERCENTILE_CONT(0.66) WITHIN GROUP
      (ORDER BY sales_amt ASC) “2nd 3rd”
FROM daily_sales
```

Median is the middle value, or the mean of two middle values.

Figure 1. Median is the middle value, or the mean of two middle values.
WHERE daily_date >= DATE '2013-01-01'
AND daily_date < DATE '2013-01-14';

1st 3rd 2nd 3rd
---------- ----------
15971.32 17933.24

Now you can do some stout rounding and say generally that:
“About ten days per month will see less than $16,000 in business, another ten days will fall between that and $18,000, and the remaining ten days will be even busier.”

Statistics isn’t always about reducing your data to a single number. Statistics is about trying to find the right balance, the right set of numbers to convey the sense of a given situation with brevity while still being precise and detailed enough to avoid misleading you into taking an incorrect action.

Note: The final example in Figure 1 shows a corner case in which computing the median from an even number of values triggers use of the mean in a way that leads to a largely meaningless result. You can test for that problem by computing the median a second time, arbitrarily discarding one of the input values. Throwing out the case in which ROWNUM = 1 is one possible approach that I demonstrate in my article at: http://gennick.com/stats.median.html.

**Geometric Mean**

What about those raises? They are not additive, so you can’t simply take the arithmetic mean using the AVG function. The math won’t work out. I’ll explain why using your first two raises from 2003 and 2004, which are 10% and 4% respectively:

```sql
SELECT empname, yearnum, emprase
FROM raises
WHERE empname = 'You'
ORDER BY yearnum ASC;
```

<table>
<thead>
<tr>
<th>EMPNAME</th>
<th>YEARNUM</th>
<th>EMPRASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>You</td>
<td>2003</td>
<td>10</td>
</tr>
<tr>
<td>You</td>
<td>2004</td>
<td>4</td>
</tr>
</tbody>
</table>

Say that your starting pay is $X per year. Your 2003 increase of 10% changes that amount to:

\[
X + (X \times 0.10)
\]

Some simple factoring simplifies the math:

\[
X \times 1.10
\]

Your 2004 increase of 4% then extends the series as follows:

\[
X \times 1.10 \times 1.04
\]

Remember that the concept behind computing an average is to find a single value to stand in place of all the separate values in the group. To convey an accurate picture of the preceding two pay raises using a single number requires a value Y such that the following is true:

\[
X \times 1.10 \times 1.04 = X \times Y \times Y
\]

\[
X \times 1.144 = X \times Y^2
\]

\[
1.144 = Y^2
\]

Oracle Database doesn’t provide a geometric mean function, and it’s problematic to multiply many values together due to the risk of overflow. What you can do though, is take advantage of logarithms to turn the multiplying and square-rooting into addition and division. Here’s the gist of what to do:

1. Sum the logarithms to gain the effect of multiplying the input values
2. Divide by the number of values to gain the effect taking the Nth root
3. Take the antilogarithm of the result to translate back into the normal world
4. Begin this approach in our scenario by writing the skeleton of the query:

```sql
SELECT *
FROM raises
WHERE empname = 'You';
```

Execute this query if you like, to verify for yourself that the correct input data is being selected:

```sql
EMPNAME | EMPTECH | YEARNUM | EMPRASE
---------|---------|---------|---------
You      | Oracle  | 2003    | 10      |
You      | Oracle  | 2004    | 4       |
You      | Oracle  | 2005    | 12      |
You      | Oracle  | 2006    | 8       |
You      | Oracle  | 2007    | 9       |
You      | Oracle  | 2008    | 4.5     |
You      | Oracle  | 2009    | 1.5     |
You      | Oracle  | 2010    | 14.5    |
You      | Oracle  | 2011    | 11.5    |
You      | Oracle  | 2013    | 8.5     |
```

Divide the input values by 100 and add 1.0 to each. This is an important step, because you need to get to values that multiply together in order to compute their geometric mean. You need to translate values such 10% and 4% into 1.10 and 1.04 respectively. For example:

```sql
SELECT 1.0 + emprase/100
```

Sum the natural logarithms to gain the effect of computing a product:

```sql
SELECT SUM(LN(1.0 + emprase/100))
```

Divide by the number of input values as a way of taking the Nth root:

```sql
SELECT AVG(LN(1.0 + emprase/100))
```

Apply the antilogarithm via the EXP function. (In mathematics, the exponential function inverts a natural logarithm, and is often abbreviated exp). This gets you back to normal numbers by raising the special value e to the power returned by AVG in the following expression:

```sql
SELECT EXP(AVG(LN(1.0 + emprase/100)))
```

Subtract 1.0 and multiply by 100 to get back to a percentage increase similar to the input values. Include the arithmetic mean for comparison. Beautify the results with some column aliases. Here’s the final query and results:

```sql
SELECT (EXP(AVG(LN(1.0 + emprase/100))) - 1.0) * 100 AS "Geometric Mean",
       AVG(emprase) "Arithmetic Mean"
FROM raises
```
WHERE empname = 'You';
Geometric Mean Arithmetic Mean
-------------------- -------------------
8.28259935 8.35

If you like, you can turn the query into a GROUP BY query to compare results between you and your colleagues:
SELECT empname, MAX(emptech) AS "Tech",
       (EXP(AVG(LN(1.0 + emprase/100)))
         - 1.0) * 100 AS "Geometric Mean",
       AVG(emprase) "Arithmetic Mean"
FROM raises
GROUP BY empname;

EMPNANE   Tech        Geometric Mean Arithmetic Mean
---------- ----------- -------------- --------------
You       Oracle          8.28259935 8.35
8.35
Justin    SQL Server      7.52976617 7.6
7.6
Sean      Oracle          8.17939708 8.25
8.25
Jeff      SQL Server      7.86734146 7.95
7.95

I'll be the first to admit that the absolute differences between these geometric and arithmetic means look trivial. The differences do matter though, especially over time. Given a $70,000 starting pay rate, you will see almost a $1000 more from ten years of 8.35% increases than from ten years of 8.28% increases. Geometric mean is the correct mean to use when the input values are multiplicative.

Mode

Mode is the last of the averages that I'll cover. The mode is simply the value that occurs most frequently. For example, you can apply the STATS_MODE function and learn that your most common raise amount is 4%:
SELECT STATS_MODE(emprase)
FROM raises
WHERE empname = 'You';
STATS_MODE(EMPRASE)
-------------------
4

Be careful though, because sometimes data can be bimodal, meaning that there are two values tied for most occurrences, or even more than two can be tied. The raise amounts are actually septimodal (seven modes) when viewed across all employees. For example:
SELECT e, ce
FROM (     SELECT e, ce,
            MAX(ce) OVER () AS mce
        FROM (             SELECT emprase AS e,
                               COUNT(emprase) AS ce
                     FROM raises
            GROUP BY emprase
     )
)

WHERE ce = mce
ORDER BY e;

E   CE
--- ---
1.5 3
4   3
9   3
10  3
10.5 3
11.5 3
12  3

The foregoing query may look intimidating, but the logic is straightforward:
1. The innermost query returns a list of percentages and occurrence counts
2. The middle query adds a third column with the maximum occurrence count
3. The outer query restricts the result to those rows in which the occurrence count is at maximum

Mode can apply to non-numeric data. This is unlike the case with mean and median. You can apply mode to names, to determine the most frequent surname, or to cars to determine the most common paint color, and so forth.

Note: John Myles White provides an excellent unification of mode, median, and mean in his blog entry at http://www.johnmyleswhite.com/notebook/2013/03/22/modes-medians-and-means-an-unifying-perspective.

The Choice

I've touched on four common measures of central tendency:

Arithmetic Mean: A good choice when the input values are additive and you want the math to work out accounting-style.

Median: Robust against the effects of outlier values; helpful when your intent is to convey a sense of the typical experience.

Geometric Mean: The right choice when you want the math to work out, but the input values represent a sequence of multiplications rather than additions.

Mode: The most frequently occurring value – useful with nominal data such as surnames and paint colors.

These and many others are colloquially termed as averages. It’s not always possible to look at a data set and say that this or that type of average should be used. In the case of pay raises, it was the nature of the data in representing a series of multiplications that seemed to drive the choice of the geometric mean, but that choice was in part due to the picture being painted to students of a career in Oracle Database and the long-term effect on pay. One could also legitimately employ the median to give a sense of the typical case – half your raises will be above 8.75%, and half below, for example.

The right choice depends in part upon the nature of your data, but also upon what it is that you’re trying to convey. Careful thought needs to be given to both aspects in order to make a choice that will inform and be helpful rather than misinform and mislead.

Acknowledgments: Dr. Myron Hlynka from the University of Windsor’s Department of Math and Statistics reviewed this article for accuracy from the standpoint of statistics. Stéphane Faroult from RoughSea Ltd. reviewed for Oracle Database correctness. My sincere thanks to both for their help and support.

Jonathan Gennick has a long history of writing on topics such as Oracle Database and its dialect of SQL. He’s currently on a journey through Oracle SQL’s built-in statistical functions, and you can follow him on that journey by visiting http://gennick.com/stats.
Can We Simplify The Process of SQL Tuning?

by Stelios Charalambides and Carlos Sierra

The Oracle cost-based optimizer (CBO) is a truly amazing piece of coding. It is at the heart of the RDBMS engine and pumps out the answers to queries constantly in a variety of environments, platforms and databases designs. Most of the time the optimizer gets it right and only rarely does the answer come very slowly. It is of course important to get the database design right, to make sure the right access paths are in place and that the right data is in the right tables for the queries expected, and that the database is properly normalized (although not too much). Let’s not get into a discussion of the fourth normal form and focus instead on how we can simplify the process of diagnosing a SQL statement when the CBO has provided a suboptimal execution plan.

Most of the time when query performance goes wrong it is because of these inputs:

- Object Statistics
- Database design
- Query structure
- Improper CBO hints

There are of course other reasons why queries might go wrong; for example, bugs or operating system resource constraints, but the ones in the list above are the ‘popular’ ones. With so many choices for how to get it wrong, it becomes a big job to figure out how to tune a query or to figure out how to improve a slow query.

**SQLTXPLAIN (SQLT) To The Rescue**

With such a complex task it would be helpful if there was a single tool that could collect all the information on a single query, the statistics on the related objects, index and table metadata, column histograms, history of statistics collected, an execution plan for the query as well all other execution plans related to the query. It would also be nice to see what setting the optimizer had. Maybe some non-standard settings in the environment and captured bind variables would also help with execution plans that used them. All of these would be really useful to help diagnose poorly running SQL. SQLT is this tool. Not only does it collect all this information, it collects much more. The list is too long to go into details here. You could be thinking, “Oh sure, one more tools to license.” The good news is that you can use this tool for free!

We’ll look at how to install SQLT first, and then we’ll look at the simplest report called an XTRACT report. There are other reports and different ways to use SQLT under different circumstances and for different purposes, but XTRACT is probably the most used report. It is easy to collect and extremely useful. So let’s look at SQLT installation first, and I promise you it takes five minutes. I know this because there is a You Tube video ([URL](http://stelioscharalambides.com/sqlinstallation-thefirststep/) which shows the installation of SQLT, with commentary, and it takes less than 5 minutes.

**Installing SQLT**

The starting point for the installation is Oracle Metalink note 215187.1. Near the bottom of that note is a link to download the latest version of SQLT (currently 11.4.5.7). I am going to go through the steps for 11.2, although there are versions of SQLT for 11.1, 10.2, 10.1 and 9.2). If you are on the older versions of the Oracle database just install the appropriate version. Once you’ve downloaded and unzipped the file from the note (make sure to put this in a dedicated directory), you will see a top level with an HTML documentation file and a number of sub-directories:

- doc
- input
- install
- run
- utl

Let me just describe what these directories are for before we begin.

The “doc” directory is a change log for SQLT itself, to describe new features and bug fixes back to SQLT 11.4.0.1. Input contains example steps to create SQL to test SQLT and gives you examples of usage. The three most important directories are “install”, “run” and “utl”. As I’m sure you’ve guessed “install” contains the script and associated files needed to install SQLT as well as the code to drop SQLT if you need to install a newer version, for example. The important thing to remember is that you install from the “install” directory and run the XTRACT report from the “run” directory. In fact most of the useful routines are in the “run” directory, although there certainly are some useful scripts in “utl” also. But let’s not get ahead of ourselves. The important task is to install SQLT. You will need SYS access to install SQLT and I can assure you it is a robust and safe tool, with a small footprint. The script I am about to run creates two schemas SQLTXADMIN and SQLTXPLAIN. Until recently there was only one schema SQLTXPLAIN, but this was changed recently due to security standards within Oracle. Now SQLTXADMIN holds the procedures, functions and so forth of SQLT and SQLTXPLAIN is the account you use.

So now we know which directory the install is to be found in. We also know it has a small footprint and only takes 5 minutes to install. Let’s go through the steps
1. Log in as sys “sqlplus / as sysdba”
2. From the “install” directory “@sqcreate”. This starts the installation script. There are a few self announcements to confirm the version of Oracle
3. The first prompt after the installation is started is for an optional connect identifier.

Specify optional Connect Identifier (as per Oracle Net)
Include “@” symbol, i.e. @PROD
If not applicable, enter nothing and hit the “Enter” key.
This connect identifier is only used while exporting SQLT repository every time you execute one of the main methods.
Optional Connect Identifier (i.e.: @PROD):

This optional input is used only if you want to do a remote install. In most cases you want to install SQLT in the local database so you press ENTER to accept the default.

4. Then you are prompted for the SQLTXPLAIN password. This is the password for the account which you use to allow access to the SQLT repository. You are prompted to confirm it.

PL/SQL procedure successfully completed.
Define SQLTXPLAIN password (hidden and case sensitive).
Password for user SQLTXPLAIN:oracle
Re-enter password:oracle
PL/SQL procedure successfully completed.

5. Then you are prompted to select a tablespace to use for SQLT and a temporary tablespace to use. Any tablespace will do. If you have a DBA admin tablespace, that is usually a good choice.

... please wait
TABLESPACE FREE_SPACE_MB
------------------------------- ---------
USERS 246
Specify PERMANENT tablespace to be used by SQLTXPLAIN.
Tablespace name is case sensitive.
Default tablespace [UNKNOWN]:USERS
PL/SQL procedure successfully completed.

... please wait
TABLESPACE
-------------------------------
TEMP
Specify TEMPORARY tablespace to be used by SQLTXPLAIN.
Tablespace name is case sensitive.
Temporary tablespace [UNKNOWN]:TEMP
PL/SQL procedure successfully completed.

6. You’re almost there. Now we have to specify the account name which runs the SQL that we are investigating. This has caused some confusion in the past so let me clarify that a little bit. If you have an SQL “select sysdate from dual;” and it is run by APP1 then you should enter APP1 at the next prompt. The user APP1 will then be granted the role SQLT_USER_ROLE. If you press enter at the next prompt no user will receive this role and the installation will continue and complete, but you will have to manually assign this role to the user if you want APP1’s SQL to be investigated with SQLT. Here is the example from the installation.

The main application user of SQLT is the schema owner that issued the SQL to be analyzed. For example, on an EBS application you would enter APFS.
You will not be asked to enter its password.
To add more SQLT users after this installation is completed simply grant them the SQLT_USER_ROLE role.
Main application user of SQLT:APP1
PL/SQL procedure successfully completed.

7. Finally we make facilities available based on the licensing level of your system. If you have the diagnostic and tuning pack then enter “T”. If you have neither of these enter “N”. If you only use the diagnostic pack enter “D”. Here is the example installation showing this phase of the installation. In my case I press ENTER to accept the default which is to include facilities for both the diagnostics and tuning pack. If this changes at a later date you can also change the SQLT setup without re-installing.

SQLT can make extensive use of licensed features provided by the Oracle Diagnostic and the Oracle Tuning Packs, including SQL Tuning Advisor (STA), SQL Monitoring and Automatic Workload Repository (AWR).
To enable or disable access to these features from the SQLT tool enter one of the following values when asked:
“T” if you have license for Diagnostic and Tuning
“D” if you have license only for Oracle Diagnostic
“N” if you do not have these two licenses
Oracle Pack license [T]:

8. The final step is to get a cup of coffee. As I said earlier the installation takes less than five minutes so that’s probably just enough time to get your coffee and take couple of sips.

The final screen of the installation shows these lines

VALID PACKAGE BODY 11.4.5.0 TRCA$R
VALID PACKAGE BODY 11.4.5.0 TRCA$T
VALID PACKAGE BODY 11.4.5.0 TRCA$X
Deleting CBO statistics for SQLTXPLAIN objects ...
13:42:58 sqlt$a: -> delete_sqltxplain_stats
13:43:01 sqlt$a: <- delete_sqltxplain_stats
PL/SQL procedure successfully completed.
SQCPKG completed.
TAUTLTEST completed.
SQUTLTEST completed.
SQLT users must be granted SQLT_USER_ROLE before using this tool.
SCCREATE completed. Installation completed successfully.
And that’s it. You’re all ready to go.
An XTRACT Report

Now the interesting part begins. After five minutes of effort we have a great tuning tool for individual SQL installed. Using the tool is just as easy. As I mentioned earlier there are a number of different facilities within SQLT, just so you can look them up. I’ve listed a few here, but really there’s a whole book’s worth of knowledge in the options for SQLT and how to use it.

- **XTRACT** – Collects information on an individual SQL
- **XEXECUTE** – Runs and then collects information on an individual execution
- **XPLORE** – Looks for better execution plans with hidden parameters and fix controls
- **XTRSBY** – Runs XTRACT on a Data Guard instance
- **COMPARE** – Compare two different executions of SQL

There are many videos and presentation of SQLT. Look at the Oracle proactive program for examples and demonstrations [http://stelioscharalambides.com/sqltinstallation-thefirststep/](http://stelioscharalambides.com/sqltinstallation-thefirststep/).

So as an example (I encourage you to install SQLT yourself and do your own example) we will look at this SQL

```sql
select count(*) from dba_objects;
```

This is our starting point. SQLT is about investigating individual SQLs (although TRCANLZR can process multiple SQLs – but that’s a different story). Once we know the SQL we need the SQL ID. We can easily get this with

```sql
SQL> Select sql_id from v$sqlarea where sql_text like 'select count(*) from dba_objects%';
```

Now we have the SQL ID and we have SQLT installed. So now we can get the XTRACT report. Remember we need to run this from the “run” directory.

```sql
SQL> @sqltxtract g4pmrqrgxg3b
```

We will be prompted for the SQLTXPLAIN password (the one we used during the installation). Then the XTRACT report does its work. The end result is a zip file with a name of this format:

```
sqlt_s<nnnnn>_xtract_g4pmrqrgxg3b.zip
```

The `<nnnnn>` is the SQLT ID (not the SQL ID) and the second half of the name contains the SQL ID under investigation. This zip file contains all the information we need to diagnose the SQL of concern. If you unzip this file you’ll see many files in the folder but for let’s just look at

```
sqlt_s<nnnnn>_main.html
```

This is the main HTML report file. We just open this file with any browser and we see at the top the main report (next column).

From here, of course, we can navigate in many different directions, and presented with such a wide choice from the first page many people get lost in the report and do not progress with their task. I would advise the following steps (although based on the task you can deviate from these steps).

1. Check the “SQL Text” hyperlink shows the correct SQL.
2. Look at the “Execution Plans” section so that you can get an idea of the complexity of what you are dealing with.
3. Carefully look through the “Observations” section to see if anything obvious has been highlighted as a problem, e.g. missing indexes, bad histograms, mismatched buckets, stale statistics etc.

From this header page we can, for example, look at the indexes.

Or we might want to look at the metadata for the objects. We need only click on “Tables” then “Metadata” to see this information.
The execution plans for the SQL are also listed.

We can see in the execution plan and all the usual details that we need to try and tune our SQL. Is a HASH JOIN appropriate for this SQL? We can even hover the mouse over certain parts of the report to get more details or click on the “+” to expand details on any of the execution steps to see more details about filters and projections.

There is insufficient space here to show too many screen shots but you can also see optimizer settings. Anything which is out of the ordinary will be highlighted in the “Observations” section; you can also see suggestions to improve the query. My favorite feature is the reporting of over and under estimates which you see in some reports on the execution plan. Here’s an example:

If you want to see a full sample of this SQLT XTRACT report you can find it here: http://carlos-sierra.nte/2013/04/24/sqlt-xtract-report-sample/. With this kind of detail along with the history you naturally get from collecting all the statistical information, you should be able to determine very quickly if a query is performing badly, then why it is performing badly and finally what to do about it.

**Summary**

Tuning is an inherently difficult and time consuming task. This is because it is complex and influenced by many factors. SQLT collects much information on all these factors in a quick and easy to use way. This allows you to investigate SQL performance problems quickly and efficiently and in a methodical manner. Use SQLT regularly on your system and you will find a wealth of information to help you. And keep in mind it is free to use, so there is nothing to lose!

Carlos Sierra is the author of some popular tools for Oracle SQL Tuning, including: SQLTXPLAIN (SQLT), SQL Health Check (SQLHC) and Trace Analyzer (TRCANLZR). He is a regular speaker at Oracle database performance-related events, including several Oracle Users Groups. Carlos currently works in the area of SQL Tuning for the Center of Expertise (CoE) at Oracle Corp. His role includes the design and development of support tools for SQL Tuning, and the development and delivery of internal workshops, also in the area of SQL Tuning. He has contributed to the Oracle community for the past 17 years, and before that he developed software for UNISYS customers.

Stelios Charalambides has more than 20 years experience working with Oracle databases. He is OCP certified from 7 to 11g and has worked as a Senior Consultant DBA on both sides of the Atlantic, dealing with all aspects of system design, implementation, and post-production support, solving practical problems in a wide variety of environments. He now works as a Principal Oracle Support Engineer developing time-critical solutions for tier-one customers with high-profile performance problems. Though born in the UK, Stelios now lives in New Hampshire with his wife, two children, and two dogs.

**Become A Member For These Outstanding Benefits**

- Quarterly Education Workshops
- Special Annual Training Days Rates
- Database Labs
- List Server Subscriptions
- SQL>Update Magazine

www.rmoug.org/member.htm
Slicing Tables
With Interval Partitioning

by Stefan Zielinski, Dell Services

Slicing tables with interval partitioning (a new feature introduced in Oracle Database 11g) is the best thing since ... sliced bread. Finally, DBAs don't have to worry or remember about adding new partitions for the next month, quarter, or year. Interval partitioning creates new partitions automatically when new data arrives, and no space is wasted on empty partitions, if there are gaps in our data.

Interval partitioning is a type of range partitioning. The target table needs to be partitioned on a single column that permits adding a NUMBER or INTERVAL type to it. Interval partitioning supports composite partitioning types, such as interval-range, internal-list, and interval-hash.

This article shows how to use interval partitioning to create a log/debug table and how to implement its retention policy by dropping older partitions (to avoid REDO on DELETE).

Creating a Partitioned Table

Almost every application or at least every database has a user-defined log table to capture application audit or debug information. Very often, those tables are left alone without purging any data, they collect millions of rows that nobody needs anymore, or they generate a lot of unnecessary REDO while being purged, especially when our primary database has its twin physical standby database and logging is forced at the database level. Interval partitioning is a perfect solution for those kinds of tables.

So let’s create our log table and partition it into monthly partitions. As shown below, we can specify more than one tablespace to store our partitions.

```sql
create table log_table
(
    message  varchar2(1000),
    tstamp   timestamp default systimestamp,
    username varchar2(64) default user
) tablespace oraprd01
partition by range(tstamp)
interval (numtoyminterval(1,'month'))
store in (oraprd01,oraprd02,oraprd03,oraprd04)
(partition PART_0000_00 values less than
 (to_date('2000/01/01', 'YYYY/MM/DD')));
```

Let’s look at the first (and only) partition we’ve just created:

```sql
column partition format A12
column pos format A3
column high_value format A16

column interval format A15
select a.partition_name partition, a.partition_position pos, a.high_value,
    decode(a.interval, 'YES', b.interval) interval
from user_tab_partitions a, user_part_tables b
where a.table_name = b.table_name
and a.table_name = 'LOG_TABLE'
order by a.partition_position
/
```

PARTITION POS HIGH_VALUE INTERVAL
------------ --- ------------ ---------------
PART_0000_00 1 TIMESTAMP' 2000-01-01 00:00:00'

Let’s also create two indexes, one local and one global.

```sql
create index local_idx on log_table
(
    tstamp
) local;
create index global_idx on log_table
(
    username
) global;
```

Note that the local index is partitioned the same way as our table, and its first and subsequent partitions have the same names as the table partitions. Its status is shown as ‘N/A’ in the USER_INDEXES.STATUS column and as ‘USABLE’ in the USER_IND_PARTITIONS.STATUS column.

```sql
select index_name, partitioned, status
from user_indexes where table_name = 'LOG_TABLE';
```

INDEX_NAME                     PAR STATUS
------------------------------ --- --------
LOCAL_IDX                      YES N/A
GLOBAL_IDX                     NO  VALID
```

```sql
select index_name, partition_name, status
from user_ind_partitions where index_name = 'LOCAL_IDX';
```

INDEX_NAME      PARTITION_NAME     STATUS
--------------- ------------------ --------
LOCAL_IDX       PART_0000_00       USABLE

Let’s insert a few rows for different months and years and see what partitions have been created. A couple of observations can be made:

- The new partitions have system-generated names.
- There are no empty partitions between the Jan-2013 and Sep-2021 partitions.
insert into log_table (message, tstamp) values ('Dec 2012 data', to_date('2012/12/01', 'YYYY/MM/DD'));
insert into log_table (message, tstamp) values ('Jan 2013 data', to_date('2013/01/01', 'YYYY/MM/DD'));
insert into log_table (message, tstamp) values ('Sep 2021 data', to_date('2021/09/01', 'YYYY/MM/DD'));
commit;

PARTITION    POS HIGH_VALUE       INTERVAL
------------ --- ---------------- ---------------
PART_0000_00   1 TIMESTAMP' 2000-
              01-01 00:00:00'
SYS_P1002      2 TIMESTAMP' 2013- NUMTOYMINTERVAL
              01-01 00:00:00'  (1, 'MONTH')
SYS_P1003      3 TIMESTAMP' 2013- NUMTOYMINTERVAL
              02-01 00:00:00'  (1, 'MONTH')
SYS_P1004      4 TIMESTAMP' 2021- NUMTOYMINTERVAL
              10-01 00:00:00'  (1, 'MONTH')

Dropping Oldest Partitions

Instead of removing the old data with a DELETE statement, we can drop the oldest partitions using the code shown below. The only partition that cannot be dropped is the very first partition (PART_0000_00 in our case) that is a base for all other partitions; an attempt to drop it would raise an error ORA-14758. (We can truncate it though.)

Please note that:
• The procedure was defined with invoker’s rights.
• The DBMS_ASSERT package was used to prevent SQL injection.
• The ‘UPDATE GLOBAL INDEXES’ clause was specified to keep the global index in a valid state.

create or replace
procedure drop_interval_partition_test(
p_drop_date date,
p_table_owner varchar2,
p_table_name varchar2)
authid current_user
is
  v_stm varchar(1000);
  v_part_yr_mo number;
  v_drop_yr_mo number;
begin
  v_drop_yr_mo := to_number(to_char(p_drop_date,'YYYYMM'));
  for x in (select a.partition_name,
             a.high_value,
             a.interval
  from dba_tab_partitions a
  where a.table_owner = p_table_owner
  and a.table_name = p_table_name
  order by a.partition_position)
  loop
    execute immediate 'select to_number(to_char(' ||x.high_value|| ' - numtodsinterval(1,'''second'''), ''YYYYMM''))
    from dual'
    into v_part_yr_mo;
    if v_part_yr_mo < v_drop_yr_mo
      -- is this a non-first partition?
      and x.interval = 'YES' then
        v_stm := 'alter table '||sys.dbms_assert.ENQUOTE_NAME(sys.dbms_assert.SCHEMA_NAME(p_table_owner),false)||'.'||sys.dbms_assert.ENQUOTE_NAME(dbms_assert.SIMPLE_SQL_NAME(p_table_name),false)||' drop partition '||sys.dbms_assert.ENQUOTE_NAME(dbms_assert.SIMPLE_SQL_NAME(x.partition_name),false)||' update global indexes';
        execute immediate v_stm;
      end if;
    end loop;
  end;
end;
/

To continue our example, let’s drop all partitions with entries inserted prior to Jan 1, 2013. Two partitions fall in that category: PART_0000_00 and SYS_P1002. As mentioned earlier, the PART_0000_00 partition cannot be dropped.

execute dbmon.drop_interval_partition_test (to_date('20130101', 'YYYYMMDD'), 'USER1', 'LOG_TABLE');

PL/SQL procedure successfully completed.

Let’s confirm that partition SYS_P1002 is gone.

PARTITION    POS HIGH_VALUE       INTERVAL
------------ --- ---------------- ---------------
PART_0000_00   1 TIMESTAMP' 2000-
              01-01 00:00:00'
SYS_P1003      2 TIMESTAMP' 2013- NUMTOYMINTERVAL
              02-01 00:00:00'  (1, 'MONTH')
SYS_P1004      3 TIMESTAMP' 2021- NUMTOYMINTERVAL
              10-01 00:00:00'  (1, 'MONTH')

The corresponding partition of the local index was also dropped:

INDEX_NAME      PARTITION_NAME     STATUS
--------------- ------------------ --------
LOCAL_IDX       PART_0000_00       USABLE
LOCAL_IDX       SYS_P1003         USABLE
LOCAL_IDX       SYS_P1004         USABLE

And the global index is still valid:

INDEX_NAME                     PAR STATUS
------------------------------ --- --------
LOCALIDX                      YES N/A
GLOBALIDX                     NO  VALID

We can schedule a monthly job that calls our procedure, and our log table will never grow beyond its desired final size.

References:
• Oracle® Database Advanced Application Developer’s Guide
  11g Release 2 (11.2) E10471-05
• Expert Oracle Database Architecture, Oracle Database 9i, 10g, and 11g – by Thomas Kyte

Stefan Zielinski is an Oracle DBA at Dell Services. He is an OCP for 9i, 10g and 11g, and RAC 11.2 Admin Certified Expert.
Application Development With Oracle Advanced Queuing

by Jeffrey Jacobs, Sr. Data Architect at PayPal

**INTRODUCTION**

This paper and corresponding presentation are intended to provide the reader and attendee with an understanding of the basic features and capabilities of Oracle Advanced Queuing (AQ) for consideration in application development. It does not cover all of the features and capabilities of Oracle AQ. The reader is cautioned that the author is neither omniscient nor infallible. The reader should consult the Oracle documentation (Oracle® Streams Advanced Queuing User’s Guide and Oracle® Database PL/SQL Packages and Types Reference) prior to beginning application development.

Oracle AQ provided PL/SQL, OCI, JMS and SOAP APIs. While they all offer virtually identical functionality, this paper and presentation refers only the PL/SQL packages, DBMS_AQ and DBMS_AQADM.

**WHAT IS MESSAGING?**

Messaging is the ability to send a message containing data from one application/process to another application/process. It is a widely used technique in distributed systems, particularly high volume OLTP systems. Unlike client server applications, which are typically synchronous, messaging is typically asynchronous; i.e. the sender, referred to as the producer, is not blocked waiting for a reply from the recipient(s), referred to as consumer(s). Oracle Advance Queuing (AQ) does not support synchronous messaging.

Messaging has many uses and advantages. It allows applications and systems to communicate and co-operate in an API independent manner. An order entry system may send a message containing order information to a fulfillment system without requiring access to internal APIs. The same message may also simultaneously be routed to an inventory management system, a customer support application, an email acknowledgment application, etc.

Messages are placed into queues, called enqueuing. The enqueuing applications are called the producers. There is typically no restriction on the number of producers for a given queue.

The application data portion of the message is referred to as the payload. Messages are read and removed from the queue by dequeuing the message. Applications dequeuing messages are referred to as consumers. There are three general categories of messaging:

- Single consumer, a.k.a., point-to-point - a message is dequeued by a single consumer
- Multicast - the producer effectively names designated consumers for the message
- Broadcast - consumers may dynamically gain access to a message queue by subscribing

A robust messaging systems supports a wide variety of features in addition to those describe above. These include:

- Enqueueing/dequeueing messages by criteria other than FIFO, including, but not limited to:
  - Enqueue time
  - Priority
  - Contents of messages
  - Reliability
  - Propagation - pushing messages to destinations
  - Other queues
  - Other databases
  - Other messaging systems (JMS, middleware, gateways)
  - Retention of messages and history of actions
  - Non-repudiation
  - Logging
  - Performance evaluation
  - Warehousing
  - Wide range of message content data types (aka payload), including:
    - Text
    - XML
    - BLOB, LOB, CLOB
    - Structured records
  - Notification to consumers of message availability
  - Guaranteed delivery
  - High performance

Oracle AQ provides all of this functionality. In addition, Oracle AQ also provides the ability to browse messages without dequeuing.

**QUEUE TYPES**

Oracle AQ provides the three types of messaging describe above via two basic types of queues, single consumer queues and multi-consumer queues. A multi-consumer queue may provide both multicast and broadcast capabilities. All queues allow any application with appropriate permissions to enqueue messages.

In a single consumer queue, a given message is dequeued by only one consumer, after which it is removed from the queue. However, multiple consumers may dequeue from the queue, e.g. multiple instances of an application, such a multiple instances of a fulfillment application processing messages from a single order queue.

Single consumer queues have the simplest underlying structure and, when used appropriately, typically offer the highest performance.

Queues need to be started after creation via START_QUEUE. Queues can be stopped via STOP_QUEUE. Both procedures allow control of enqueuing and dequeuing separately.

For multi-consumer queues, the determination as to whether a message is broadcast or multicast is made at the time the message is enqueued; it is not a property of the queue itself.
MESSAGE STATES
A message may be in one of the following states:
• READY – message is available to be dequeued
• WAITING – availability for dequeuing is delayed
• EXPIRED – message has timed out and been moved to exception queue
• PROCESSED – message has been consumed by all consumers

BUFFERED MESSAGING
Buffered messaging is a light weight, non-persistent form of messaging, which can be specified at the time of enqueuing. It is generally only memory resident, and does not support many of the features that are available for persistent messaging. In particular, buffered messages do not support:
• Grouping
• Retention
• Guaranteed delivery
• Array dequeuing

ADVANCED QUEUING (AQ) TABLES
An AQ table is an abstract object type, which may be implemented by one or more underlying tables, indexes and index organized tables depending on whether the AQ table supports single or multi-consumer queues.

An AQ table typically holds one or more queues, which can be created and destroyed dynamically. Multi-consumer AQ tables typically require more management and overhead. AQ tables are created by:

```
DBMS_AQADM.CREATE_QUEUE_TABLE( queue_table IN VARCHAR2,
queue_payload_type IN VARCHAR2, [storage_clause
IN VARCHAR2 DEFAULT NULL], sort_list IN VARCHAR2
DEFAULT NULL, multiple_consumers IN BOOLEAN DEFAULT
FALSE, message_grouping IN BINARY_INTEGER DEFAULT
NONE, comment IN VARCHAR2 DEFAULT NULL,
primary_instance IN BINARY_INTEGER DEFAULT 0,
secondary_instance IN BINARY_INTEGER DEFAULT 0,
compatible IN VARCHAR2 DEFAULT NULL,
queue_type IN BINARY_INTEGER DEFAULT NORMAL_QUEUE,
queue_table IN VARCHAR2,
queue_name IN VARCHAR2,
primary_instance–primary owner of the queue table service
secure IN BOOLEAN DEFAULT NULL, dependency_tracking IN BOOLEAN DEFAULT
FALSE, comment IN VARCHAR2 DEFAULT NULL,
max_retries IN NUMBER DEFAULT NULL, retry_delay IN
NUMBER DEFAULT 0, retention_time IN NUMBER DEFAULT
0, auto_commit IN BOOLEAN DEFAULT TRUE);
```

The relevant parameters are described below:
• queue_table – AQ table name
• queue_payload_type – payload type
• storage_clause – any valid storage clause. Tablespace should always be specified. Oracle recommends using ASSM. If ASSM is not used, INITTRANS and PCTFREE may be set if needed for extremely high transaction queues; this has not been necessary in the author’s experience.
• sort_list – determines the order in which messages are normally dequeued. It applied to all queues and governs the generation of the underlying queries. This can be overridden by certain dequeuing options, but it cannot be changed after creation. The default is enqueue time, which is effectively FIFO.
• multiple_consumers–TRUE or FALSE. All queues in the AQ table are of this type.
• message_grouping – ‘NONE’ or ‘TRANSACTIONAL.
If TRANSACTIONAL, all messages enqueued in one transaction may be treated as a group when dequeuing. See Transaction Protection below.
• comment – a description of the AQ table which will be stored in the table dictionary.
• primary_instance – primary owner of the queue table service

RAC CONSIDERATIONS
Each AQ table effectively creates a service. AQ table structures are typically hot tables with a great potential for hot blocks. To avoid performance issues caused by cache contention, the services should be pinned to a single node (aka node affinity).
• primary_instance specifies the preferred instance on which the service will run. secondary_instance specifies the preferred instance if primary instance is not available. If neither instance is available, a “random” instance is selected.

CREATING QUEUES
Queues are created via:

```
DBMS_AQADM.CREATE_QUEUE ( queue_name IN VARCHAR2,
queue_table IN VARCHAR2,
queue_type IN BINARY_INTEGER DEFAULT NORMAL_QUEUE,
max_retries IN NUMBER DEFAULT NULL, retry_delay IN
NUMBER DEFAULT 0, retention_time IN NUMBER DEFAULT
0, dependency_tracking IN BOOLEAN DEFAULT FALSE,
comment IN VARCHAR2 DEFAULT NULL,
queue_table–the name of the AQ table holding queue.
queue_name–the name of the queue.
queue_type – NORMAL_QUEUE or EXCEPTION_QUEUE.
max_retries–the maximum number of dequeue retries before moving to exception queue; see Transaction Protection below.
retry_delay–after a failure (usually ROLLBACK), the number of seconds before message will be available for dequeuing again.
retention_time–the time the message remains in the queue table after dequeuing.
dependency_tracking - not currently implemented
comment – Queue documentation, which is kept in the data dictionary.
auto_commit - deprecated;
```

ENQUEUE OPTIONS AND FEATURES
There is a wide range of options for enqueuing messages. These options include, but are not limited to:
• Enqueuing a single message.
• Enqueuing an array of messages (PL/SQL or OCI).
• Message Grouping, which treats all messages enqueued in a single transaction as a group.
• Sender Identification.
• Time Specification and Scheduling of message delivery.
• Correlation Identifier, which allows multiple messages queued with a user defined identifier to be dequeued together.
ENQUEUING MESSAGE
The following PL/SQL API is used to enqueue messages:

```sql
DBMS_AQ.ENQUEUE( queue_name IN VARCHAR2,
  enqueue_options IN enqueue_options_t, message_properties IN message_properties_t, payload IN "type_name", msgid OUT RAW);
```

- `queue_name`–the name of the queue in which the message is to be enqueued.
- `payload`–the type definition of the payload, typically, but not limited to, a PL/SQL abstract type
- `msgid`–the unique identifier of the message

**DBMS_AQ.ENQUEUE_OPTIONS_T**
The `DBMS_AQ.ENQUEUE_OPTIONS_T` record contains the options for enqueuing the message as described below:

```plsql
TYPE SYS.ENQUEUE_OPTIONS_T IS RECORD ( visibility BINARY_INTEGER DEFAULT ON_COMMIT, relative_msgid RAW(16) DEFAULT NULL, sequence_deviation BINARY_INTEGER DEFAULT NULL, transformation VARCHAR2(61) DEFAULT NULL, delivery_mode PLS_INTEGER NOT NULL DEFAULT PERSISTENT);
```

The attributes are:
- `visibility`-
  - ON_COMMIT–the message is enqueued as part of the transaction; i.e., enqueuing the message is completed by COMMIT.
  - IMMEDIATE – the message is enqueued immediately in an autonomous transaction.
- `transformation`–specifies a transformation function to be performed before enqueuing (not covered in this paper).
- `delivery_mode`
  - PERSISTENT - the message is stored in the queue table.
  - BUFFERED–the message is only maintained in memory, and may be lost in the event of system failure or database shutdown.
- `sequence_deviation`–deprecated as of 10.2
- `relative_msg_id`–effectively deprecated.
- `sequence_deviation`–effectively deprecated.

**DBMS_AQ.MESSAGE_PROPERTIES_T**
The `DBMS_AQ.MESSAGE_PROPERTIES_T` record is used for both enqueueing and dequeuing operations:

```plsql
TYPE message_properties_t IS RECORD ( priority BINARY_INTEGER NOT NULL DEFAULT 1, delay BINARY_INTEGER NOT NULL DEFAULT NO_DELAY, expiration BINARY_INTEGER NOT NULL DEFAULT NEVER, correlation VARCHAR2(128) DEFAULT NULL, attempts BINARY_INTEGER, recipient_list AQ$_RECIPIENT_LIST_T, exception_queue VARCHAR2(61) DEFAULT NULL, delivery_mode PLS_INTEGER NOT NULL DEFAULT PERSISTENT, state BINARY_INTEGER, sender_id SYS.AQ$_AGENT DEFAULT NULL, original_msgid RAW(16) DEFAULT NULL, signature aq$_sig_prop DEFAULT NULL, transaction_group VARCHAR2(30) DEFAULT NULL, user_property SYS.ANYDATA DEFAULT NULL, enque_time DATE, send_time DATE, delivery_mode BINARY_INTEGER NOT NULL DEFAULT DBMS_AQ.PERSISTENT);
```

The relevant enqueue attributes are:
- `priority` – the priority of the message. This is only relevant if the sorting method specified for the table includes the priority.
- `delay` – specifies number of seconds before a message is available for dequeuing. Default is 0 (NO_DELAY)
- `expiration` – the number of seconds a message is available for dequeuing (after delay). If the message is not dequeued by all subscribers, it will be moved to the exception queue with a status of EXPIRED. This is necessary for multi-consumer queues, as not all subscribers may be able to dequeue the message. Default is the constant NEVER. delivery_mode - DBMS_AQ.BUFFERED or DBMS_AQ.PERSISTENT, determines if the message is buffered or persistent. The default is persistent.
- `correlation` - the ID used for dequeuing by correlation ID. This is a producer supplied value, which allows a logical grouping of messages. Unlike a transaction group, the messages need not be enqueued in a single transaction or by the same producer.

DEQUEUEING FEATURES
Oracle AQ provides very high performance and functionality. Key features include:
- Concurrent dequeues
- Multiple dequeue methods and options
- Array dequeue
- Message navigation
- Waiting for messages
- Retries with delays
- Transaction protection
- Exception queues

**DBMS_AQ.DEQUEUE**
The PL/SQL API is:

```sql
DBMS_AQ.DEQUEUE( queue_name IN VARCHAR2,
  dequeue_options IN dequeue_options_t, message_properties OUT message_properties_t, payload OUT "type_name", msgid OUT RAW);
```

Note that `message_properties_t` is used for both enqueue and dequeue operations.

**DEQUEUE_OPTIONS_T**

```plsql
TYPE DEQUEUE_OPTIONS_T IS RECORD ( consumer_name VARCHAR2(30) DEFAULT NULL, dequeue_mode BINARY_INTEGER DEFAULT REMOVE, navigation BINARY_INTEGER DEFAULT NEXT_MESSAGE, visibility BINARY_INTEGER DEFAULT ON_COMMIT, wait BINARY_INTEGER DEFAULT FOREVER, msgid RAW(16) DEFAULT NULL, correlation VARCHAR2(128) DEFAULT NULL, deq_condition VARCHAR2(4000) DEFAULT NULL, signature aq$_sig_prop DEFAULT NULL, transformation VARCHAR2(61) DEFAULT NULL, delivery_mode PLS_INTEGER DEFAULT PERSISTENT);
```

The `DBMS.AQ.DEQUEUE_OPTIONS_T` specifies the dequeueing options as described below: `consumer_name`–the name of the subscriber.
- `dequeue_mode`. Modes include:
  - REMOVE (with data) – this is the typical dequeuing method. The message may remain in the queue table
for history based on retention period, but it not eligible for future dequeuing (unless via msg_id).

- REMOVE_NODATA—no data is returned, but the message is removed from queue. This may be used for selective cleanup.
- BROWSE – reads the message data, but does not actually dequeue the message. The message remains available for future processing (unless dequeued by another process). Browsing may not be repeatable, and as such there are numerous “gotchas” to be aware of.

- navigation—there are two methods for navigation when dequeuing.
  - FIRST_MESSAGE—This creates a “snapshot” (effectively a cursor); note that this only retrieves messages that were enqueued at the time of the dequeue call.
  - NEXT_MESSAGE – If FIRST_MESSAGE was used, this retrieves the next message in the snapshot. See Default

- Dequeuing below.
  - wait– If no messages are available, the consumer may wait for the next message. The options are:
    - FOREVER – waits forever, which is the default. Typically used for high frequency queues. Note that this blocks the process.
    - NO_WAIT–don’twaitfornextmessage. Typically used for deferred or batch operations, which are initiated by jobs scheduled at regular intervals.
    - Number – the wait time in seconds. Process is blocked while waiting. The next message is dequeued on wake up.

NOTE BENE: Oracle AQ also offers the ability for a process to listen on multiple queues; the functionality is outside the scope of this paper.

DEQUEUE METHODS

There are several methods for dequeuing messages. The default is to dequeue individual messages based on the sort order specified when the AQ table was created.

NOTE BENE: the most efficient navigation method for dequeuing based on the sort order is to use NEXT_MESSAGE without FIRST_MESSAGE. FIRST_MESSAGE always performs a query. However, if NEXT_MESSAGE is used without FIRST_MESSAGE, it will only perform one SELECT in the session; subsequent calls are simple fetches.

Other methods are:

- Correlation ID – dequeue series of message based on correlation as follows:
  - Get correlation id by dequeuing using FIRST_MESSAGE. Dequeue additional messages via NEXT_MESSAGE using the value of correlation until no more messages remain.

The specification for correlation may use pattern matching (%). This method typically requires the addition of an index and generation of statistics to force the underlying queries to use the index on the correlation column.

- Transaction group – similar to correlation, but uses transaction_group set by producer. Should use array dequeuing, but may use same loop as Correlation ID above, but specifying the transaction_group. Pattern matching may also be used.

- deq_condition—similar to SQLWHERE clause, accesses contents of payload object elements or other columns. See documentation for more details about specifying columns and payload elements. Note that using the method supersedes all other methods.
- msgid - dequeue a single message by system-assigned RAW value. This typically requires browsing the queue(s), and is usually used for cleanup and corrections.

DEQUEUE VISIBILITY

Messages may be dequeued in the following modes:

- IMMEDIATE – Messages are removed from the queue in an autonomous transaction. If the application does not have retry capabilities, this will typically offer better performance and scalability
- ON_COMMIT (transaction protection) - Messages are removed from the queue on COMMIT of the transaction. The dequeue operation is treated in the same manner as an INSERT/UPDATE/DELETE. If the transaction fails, either due to ROLLBACK, system failure or shutdown, the retry count is incremented. If the retry count is exceeded, the message is moved to the exception queue, otherwise it remains in the original queue. Note that a system failure or shutdown may not increment the retry count. If retry_delay was specified when the queue was created, the message will not be available for dequeuing for the specified number of seconds.

MESSAGE EXPIRATION

If expiration is specified in message_properties_t.expiration, all consumers must dequeue the message before expiration time. Otherwise, the message is moved to the exception queue. It is generally a good practice to specify expiration for multi-consumer queues, as not all consumers may be active, which would result in the message remaining in the queue indefinitely.

EXCEPTION QUEUES

Each AQ table has at least one exception queue which contains messages that have expired or exceeded retry count from all of the other queues. Messages in an exception queue may be dequeued once by only one consumer for reprocessing. Exception queues should be monitored and periodically emptied either for reprocessing or simply free space.

PROPAGATION

Messages may be pushed to other queues via propagation. Those queues typically, but not always, exist in another database or an external messaging system; the latter is beyond the scope of this paper. Propagation may also be to queues in the same database. The messages are ultimately processed by consumers of the destination queue(s); propagated messages are considered process upon completion of propagation. Propagation may push messages to multiple queues in multiple targets (fan out). Messages may also be propagated from multiple sources into a single queue. The destination queue may be single or multi-consumer, but must be of the same payload type. Propagation is performed by scheduled jobs. A propagation window is a period of time in which propagation can occur, i.e. effectively scheduling the job.

There are two basic modes for propagation between databases:

- Queue to dblink – Effectively deprecated.
- Queue to queue – the target queues are specified.
You may think that based on the title of this article, it is from a motivational therapist who lives in a van down by the river; not quite. How many times have you embarked on a journey through IT land just to find yourself hopelessly un-prepared for the work at hand? I pray your answer is “not often” but alas in way too many cases, some of us have.

We all learned lessons the hard way. How about the DBA that once completely wiped $ORACLE_HOME underneath a running database? How about the hapless DBA that wiped $ORACLE_HOME/dbs with one rm command? There is nothing like planning and planning and, oh ya, did I mention planning? Throughout the balance of this article, we will cover the first few steps in a detailed workplan to rebuild a set of online redo logs on top of a running database—running database you say! Even more reason to practice what I preach—do it right; do it once.

In the Beginning

Upon first examining the requirements for an online redo log rebuild, one easily thinks “piece of cake”. I have done that a bazillion times. I know the nuances and idiosyncrasies of the work at hand and nothing could possibly go wrong/go wrong/go wrong … Based on that adage and one’s confidence since the work has been done before, let’s now look at the first cut at a workplan to perform the task at hand.

| Time: 8:00 |
| Date: 12/4/13 |
| Database: prod |
| Server: margaret |

[ ] connect to server
[ ] change to owner of software binaries
[ ] set oracle environment
[ ] log into sqlplus
[ ] switch logfile
[ ] drop old group
[ ] add new group
[ ] drop another old group
[ ] add another new group
[ ] erase files for old groups

The sad part about the above list of steps is they are indeed significantly more detailed than some DBAs get for such an important task on a production environment. Let’s pick apart the above steps that conjure up a wealth of questions that accompany each.

**connect to server**

As a seasoned DBA the following questions spring to mind:
1. How do I connect—is this a Windows box or Linux as I would look foolish trying to ssh to most Windows servers
2. What login do I use to make my first connection
3. If someone has never connected to this server and knows none of its metadata, where could that information be found

**change to the owner of the software binaries**

More questions:
1. Who indeed might that be as there are 4 copies of the binaries on the machine
2. What command should I use to change to that user

**set oracle environment**

This is pretty straightforward since we already know the desired ORACLE_SID:
1. What happens when I run oraenv and after entering the desired ORACLE_SID I am prompted for ORACLE_HOME
2. If I find the desired ORACLE_SID missing from oratab how do I proceed

**log into sqlplus**

Easier said than done:
1. What login credentials should I use
2. What if I am told insufficient privileges when using SYSDBA
3. What if the O/S does not understand the command sqlplus

**The next 5 commands**

These are the heart of the work and the place where, if one does not do it right, may get to do it many, many more times than just once:
1. What group should I drop as I believe I need a group number
2. What query should I used to ascertain the group numbers
3. What if the group is in use
4. What if the group is not in use but Oracle is not done with it
5. ...
6. ...
We could go on for days. We are now ready to move this into the next step. While doing just that some of the perplexing questions from above will be answered ...

**Version 2**

The following plan is expanded and much more usable than its predecessor; we will only concentrate on the first three steps to get across the point I am trying to make in this section.

Time: 8:00am pacific
Date: Dec 4, 2013

[ ] connect to server

server name = Margaret
IP address = 10.21.11.121
login = pythian
location of login metadata = client sheet
connection method = ssh

COMMAND = ssh pythian@margaret

[ ] change to owner of software binaries

obtain oracle O/S password
location of login metadata = client sheet

COMMAND = sudo su – oracle

CHECK-UP = are we really the oracle user
COMMAND = id
EXPECTED RESULT = uid=501(oracle) gid=20(orainst)

[ ] set oracle environment

ORACLE_SID = PROD

COMMAND = . oraenv

RESPONSE = PROD

CHECK-UP = is the ORACLE_SID as expected
COMMAND = echo $ORACLE_SID
EXPECTED RESULT = PROD

We have made massive amounts of progress since the first version. I commonly use the baseball diamond analogy to explain just how far we have progressed; in this case we are somewhere between first and second. In another land, we are only half way down the pitch. This is what we are missing and is the heart of do it right, do it once. Let’s build the next version.

**Version 3**

Time: 8:00am pacific
Date: Dec 4, 2013

Client contact information:

Primary: Network Ops (NOC)
Technical : Ben Racine

Escalation tree:

Primary: Jenn Leachman
Secondary : Mac Henderson

Please await the go-ahead from one of the client contact personnel to begin the work.

[ ] connect to server

server name = Margaret
IP address = 10.21.11.121
login = pythian
location of login metadata = client sheet
connection method = ssh

COMMAND = ssh pythian@margaret

CHECK-UP = are we on the right server
COMMAND = hostname
EXPECTED RESULT = margaret.pythian.com

[ ] change to owner of software binaries

obtain oracle O/S password
location of login metadata = client sheet

COMMAND = sudo su – oracle

CHECK-UP = are we really the oracle user
COMMAND = id
EXPECTED RESULT = uid=501(oracle) gid=20(orainst)

[ ] set oracle environment

ORACLE_SID = PROD

COMMAND = . oraenv

RESPONSE = PROD

CHECK-UP = is the ORACLE_SID as expected
COMMAND = echo $ORACLE_SID
EXPECTED RESULT = PROD

I think we all get the gist of the above and recognize the importance of planning and formulation of the intimate details of work to be performed. This meticulous planning will protect us against the Achilles heel when performing maintenance ... human error. Where’s a good place to start—have a look at the following video http://www.pythian.com/blog/how-toreduce-mistakes-when-managing-a-database/ that outlines the direction detailed planning can take to mitigate risk and reduce the likelihood of human error. This human reliability approach is the brainchild of a handful of professionals at Pythian where I have worked for eight of the last 14 years. Pythian’s take on reducing errors when managing a database rolls up into an approach tagged FIT-ACER summarized in the next table:

<table>
<thead>
<tr>
<th>F – Focus (SLOW DOWN! Are you ready?)</th>
<th>A – Assess the command (SPEND TIME HERE!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I – Identify server/DB name, time, authorization</td>
<td>C – Check the server / database name again</td>
</tr>
<tr>
<td>T – Type the command (do not hit enter yet)</td>
<td>E – Execute the command</td>
</tr>
<tr>
<td>R – Review and document the results</td>
<td></td>
</tr>
</tbody>
</table>

**Guidelines**

This meticulous planning benefits the consumer by reducing the potential of something going wrong. The most important steps
in preparation of a workplan for carrying out maintenance, no matter how seemingly trivial, are as follows:

- Document the environment
- Assemble contact list
- Checkpoint on date and time of work
- Prepare detailed list of commands
- Insert checkpoint information after each step
- Share material with second resource to elicit feedback
- Incorporate feedback as/if required
- Circulate final plan to colleagues and client personnel

Following the above will ensure the work is done properly leading to a successful implementation and allow you to do it right; do it once.

About the author

Michael S. Abbey is a seasoned presenter in the Oracle database space, having appeared at a wealth of user group and vendor events for the last 23 years. He honed his skills on the core database technology starting with Oracle version 3 in 1986. He was on the team that produced the first offering in the Oracle Press series in 1994. He is a team lead at Pythian who offer managed services that cross legacy as well as some of the latest big data offerings such as MongoDB, Cassandra and Hadoop.
The API to schedule propagation is:

```
DBMS_AQADM.SCHEDULE_PROPAGATION ( queue_name IN VARCHAR2,
  destination IN VARCHAR2 DEFAULT NULL, start_time IN DATE DEFAULT SYSDATE, duration IN NUMBER DEFAULT NULL, next_time IN VARCHAR2 DEFAULT NULL, latency IN NUMBER DEFAULT 60, destination_queue IN VARCHAR2 DEFAULT NULL);
```

The parameters are:
- `queue_name`– the name of the queue to be propagated.
- `destination` – destination dblinks.
- `start_time`– the start time for the propagation; i.e., the time when the job will first be scheduled.
- `duration`– how long propagation lasts in seconds. NULL means the propagation lasts forever (or until stopped or altered).
- `next_time`– a calendar expression (as used by `DBMS_SCHEDULER`) for the next propagation window.
- `latency` – if no messages, how many seconds to wait until checking the queue for message to be propagated. 0 results in propagation as soon as a message is available.

Other APIs to manage propagation are:
- `ALTER_PROPAGATION_SCHEDULE`
- `DISABLE_PROPAGATION_SCHEDULE`
- `ENABLE_PROPAGATION_SCHEDULE`
- `SCHEDULE_PROPAGATION`
- `VERIFY_QUEUE_TYPES`

**AQ TABLE STRUCTURES**

A multi-consumer AQ table has 7 underlying tables, both heap and index organized. The main table with message data for all queues has the same name as specified in `CREATE_QUEUE_TABLE`, e.g. `ORDERS_QUEUETABLE`. Other tables have names beginning with `AQ$`, e.g. `AQ$ORDERS_QUEUETABLE_H`.

A single consumer AQ table creates a single table with main table name; the index structure may vary.

**PERFORMANCE TIPS FOR DEQUEUING**

Using certain features, such as correlation id or transaction grouping, may require additional indexes on the main table. To change the behavior of the queries used by AQ, statistics need to be gathered, as AQ tables are exempt from automatic statistics gathering. However, generating appropriate statistics in a production environment can be problematic due to the volatile nature of queues; stopping the queues to allow messages to build up in order to gather statistics is probably not acceptable to the DBAs. Statistics can either be created manually, or, in a development or QA environment, messages can be enqued without dequeuing. The statistics can then be imported into production for the table. It’s also a good idea to lock the statistics, just to be safe.

**QUERY TO BE TUNED**

Finding the underlying dequeueing query for tuning is not immediately obvious. Look in appropriate V$ or GV$ views or AWR report for the following pattern:

```
SELECT /*+ FIRST_ROWS(1) */ tab.ROWID, ...
  tab.user_data FROM <queue_table_name> -- the name of the main queue table
WHERE q_name = :1 AND (state = :2 and ... ORDER BY q_name, ... FOR UPDATE SKIP LOCKED;
```

**FOR UPDATED SKIP LOCKED** is the “secret sauce” for AQ’s performance. It performs `SELECT FOR UPDATE` only on rows that are not currently locked!!! It also apparently only locks rows when they are fetched, but this has been difficult to confirm. This is not a documented or user supported feature.

**MORE STUFF**

It is not possible to cover all of the capabilities and functionality of Oracle AQ in this paper. Some other features of potential interest include:
- AQ automatically manages space, perform COALESCE as well as removing messages that have passed their retention periods.
- There are numerous APIs for managing all aspects of AQ.
- AQ can propagate messages via external protocols and gateways.
- AQ can be accessed via SOAP.
- AQ can retain the entire history of a message for non-repudiation, logging, etc.

The author strongly urges the reader to consult the appropriate documentation, in particular `Oracle® Database PL/SQL Packages and Types Reference` and `Oracle® Streams Advanced Queuing User’s Guide`.

Jeff Jacobs is an Oracle Ace and has over 20 years’ experience working with Oracle. He is a Senior Site Data Architect for PayPal and a member of the IOUG Exadata SIG board of directors. He served on the board of ODTUG for 15 years, including stints as President and Conference Chair. Prior to joining PayPal, he provided consulting services to numerous Fortune 500 companies and trained over 3000 students in various aspects of Oracle products, database design, modeling and methodology. He has presented at Oracle OpenWorld, Collaborate, ODTUG Kaleidoscope, RMOUG, NoCOUG and other regional user group meetings and webinars.

Become A Member For These Outstanding Benefits

- Quarterly Education Workshops
- Special Annual Training Days Rates
- Database Labs
- List Server Subscriptions
- SQL>Update Magazine

www.rmoug.org/member.htm
Oracle BI Extended Edition (OBIEE)  
Type & Techniques  
By Dan Hotka

I have been working with Oracle's BI tools for years. I am quite the Discoverer expert (a free tool now from Oracle Corp...OBIEE...standard edition) but this tool is now on the back burner and I hardly even field questions on it anymore.

This is part 2 of a 3 part series on getting OBIEE setup (part 1) and creating some reports. Part 3 will expose you to BI Publisher.

You can try these reports out yourself at: http://www.vlamis.com/testdrive-registration/. This cloud image is the Sample207 image discussed in Part 1. So, you can use the cloud environment provided by Vlamis.com or you can use the same walk thru against the download OBIEE environment discussed in Part 1.

Additional walk thru’s and useful documentation can be found at: http://www.oracle.com/technetwork/middleware/bi-enterprise-edition/overview/index.html

Logging into the Sample207 environment:

- http://<ip address>:7001/analytics
  login: weblogic
  password: Admin123

OBIEE (Oracle Business Intelligence Enterprise Edition) is based off the Siebels Analytics tool. This is a complete web development/data access/data formatting environment.

You can:
- Run simple queries
- Run existing reports
- Integrate existing reports into groups that pull in related data and detail (dashboards)
- Produce quality reports and web pages using the integrated BI Publisher tool.

The Reports:
- Column reports
- Matrix reports (also called pivot tables or cross-tabular reports)
- Graphs
- Narratives
- Links
- Objects (such as images)

- Can include prompts to filter to different data using the same report

This is the screen you would see upon logging in. The end user might see a dash board. This screen shows the user what they have available at the click of a mouse. Being the administrator, this user would see anything available.

The ‘New’ menu item then click on ‘Analysis’ brings up the report-building interface.

The report-building interface allows easy access to tables and columns. The representation here, the folders would be tables; the items in the folders would be columns. This is not entirely true though; the folders could be complex joins across your entire enterprise of data (yes...mixing Oracle, other databases, and even spreadsheet data!). The columns can be a data column from a table, a calculation, or a mix of other items.

The subject areas are then displayed.
All of these table objects, columns, reports, and subject areas are all controlled by permissions both in the OBIEE interface and the Weblogic environment.

The ‘Save’ button (upper right corner) is an important button. Safe often and frequently!

Undo...undoes the last action only...usually...

Let’s build a simple report using the B -Training subject area. On the menu bar...select New then Analytics then B – Training.

The report is easy to build. You can drag and drop the columns from the subject area into the ‘Selected Columns’ area or simply double click on them.

This report has
- Product Type from the Products folder
- Month from the Time folder
- 121 Period Ago Rev from the Time Series folder
- (2) Revenue columns from Base Facts.

The easiest way to add an additional column is to drag another column out onto the selected columns area, then click on the ’Options’ icon in with the column name and select ‘Edit Formula’. From here, you can change the contents of the column to almost anything. All SQL functions work here too!

Enter this calculation into the edit window and click ‘Ok’.

$$(("Base Facts","Revenue"/"Time Series","121 Period Ago Rev")-1)*100$$

...If there is a syntax error, this window will not close and will inform you (yes...with the usual meaningful error from Oracle Corp) of the problem it encountered.

Click on the ‘Results’ tab to view your report. Save your report. To rerun your report...simply click on it from your startup dashboard, from the catalog, or ‘Open’ it.

From here, we can fix the columns, add conditional formats, add data formatting, add a heading, then we could add additional features to the report by adding a prompt, a graph, additional columns of detail, additional reports, and so much more. We can easily execute this report. We can include this report into a new or existing dashboard...to be executed with other reports and graphs.

This information should get you started on creating reports.

The next article will introduce you to BI Publisher.

Let me know how I can help you with your OBIEE implementation.

-Dan

Dan Hotka is a Training Specialist and an Oracle ACE Director who has over 35 years in the computer industry, over 29 years of experience with Oracle products. His experience with the Oracle RDBMS dates back to the Oracle V4.0 days. Dan enjoys sharing his knowledge of the Oracle RDBMS. Dan is well-published with 12 Oracle books and well over 200 printed articles. He is frequently published in Oracle trade journals, regularly blogs, and speaks at Oracle conferences and user groups around the world.
In the last two years since joining RMOUG, changes have been constant in my life, both professionally and personally. My first board focus, which was written for the summer issue 2011, during my first quarter on the board of directors found me on a new path in my life. That path has assisted me in achieving many of my goals and I’m proud to be a representative of this fine Oracle user group.

Professionally, we just returned from Copenhagen, Denmark, where Tim and I did a joint keynote for MOW, (Miracle Open World) conference. This was well received and we’ve been asked to present at a number of other European conferences this next year due to the great success. Joint presentations are always an interesting challenge and to do it as a keynote was an additional challenge we’re not sure we want to do too often!

On the home front, my three children have continued to grow and succeed. My oldest, Sam, is now out of school and having a great time teaching himself C++, (hey, it’s as close as I can get him to an Oracle career). My middle child, Cait, is in high school and has even volunteered for RMOUG Training Days, helping us prepare attendee’s bags, register attendees and set up signage. My youngest, Josh, is busy in boy scouts and preparing to go to Jamboree this summer in Washington DC.

One of my most recent ventures has involved Women in Technology, (WIT). I don’t know how any woman with a career in technology can’t have this subject trouble and challenge her. To fathom how, after this many decades since women’s introduction to the workplace, we are still so far behind in promoting women in the technical arena. I’ve chosen to take more and more time addressing the challenges in presentations and panels at different conferences – luckily, I’m not the only one and there is a myriad articles and books on the topic.

I’ve been honing my skills at Enkitec, an Oracle centric partner, (http://enkitec.com) for just over a year now. They offer me the incredible opportunity to present at conferences when I’m not working at home. I work primarily with Exadata environments and enjoy large databases, whenever I’m offered them. I’m a very hands-on DBA and like to spend time digging into the very heart of problems, offering real solutions vs. band-aids or other temporary solutions. I’ve always been impressed with the excellent management and technical opportunities of the company. They have been very supportive.
of my conference schedule and my recent addition of ACE Director from the Oracle Technology Network.

I highly recommend anyone who is blogging, presenting and involved in the Oracle community to consider the ACE program, (http://www.oracle.com/technetwork/community/oracle-ace/index.html) There are two phases of the program- the first, ACE, recognizes what Oracle specialists have accomplished in their career. The second, the ACE Director program, is an ongoing program that requires the individual to provide continued value to the Oracle community and to other supporting programs.

Spreading word about RMOUG around the US and select countries is a benefit of being an ACE or ACE Director. You are offered the opportunity to discuss what a great local user group we have here in the Denver area and the great content/conference we offer each February at Training Days. The conference has received worldwide recognition from so many as the place to go for great technical content and an incredible grass-roots location. Anyone from the Denver area can be proud of the local Oracle user group they belong to. The reviews have been unanimous.

This next year, we’ve already signed on Tom Kyte as our keynote speaker for RMOUG Training Days 2014. This will be a great opportunity to our local attendees who rarely get to see Tom in person. We will be enhancing the “lunch with an ACE” program and utilizing the great Guidebook mobile application more for 2014. We hope to offer more defined, incredibly detailed content and a newer format that will benefit the Oracle specialist even more than before. The WIT, (Women in Technology) program will be enhanced and incorporated into a type of professional development track, along with Oracle career sessions that all received great response at the 2013 conference. We are working on some new, great social media programs, along with better marketing opportunities. The conference will undoubtedly be an even better choice for Oracle training for any database professional.

I am looking forward to the next year, my third year on the board of directors for RMOUG, continued involvement with ODTUG’s KSCOPE, (I was database track lead for their conference this year) and all the other involvement I have with webinars, presenting and writing. I hope to mentor more individuals in the database field and will focus on WIT topics, not just at conferences, but in schools, where we need to aim to reach the women of tomorrow.

I would like to dedicate this article to my middle sister, Kristi Higday, who passed away May 19th after an incredibly brave battle with colon and liver cancer. Shall she return to us as a sunbeam, for the sunlight was where she was always happiest.
Meet Your Board

Tim Gorman
President & Vendors Director
E-mail: President@rmou.org

John Peterson
Vice President
E-mail: VicePresident@rmou.org

Ron Bich
Secretary
E-mail: Secretary@rmou.org

Thomas Green
Treasurer
E-mail: Treasurer@rmou.org

Mark James
Bob Mason-Assistant
Member Relations Director
E-mail: MembershipDir@rmou.org

Pat Van Buskirk
Newsletter Director
E-mail: NewsletterDir@rmou.org

Carolyn Fryc
Vice President & Programs Director
E-mail: ProgramsDir@rmou.org

Art Marshall
Web Director
E-mail: WebDir@rmou.org

Chris Ostrowski
Vendor Relations Director
E-mail: VendorsDir@rmou.org

Dan Hotka
Education Director
E-mail: EducationDir@rmou.org
Kellyn Pot’vin  
Training Days Director  
E-mail: TrainingDaysDir@rmoug.org

Bob Mason  
Mark James-Assistant Scholarships Director  
E-mail: ScholarshipDir@rmoug.org

Bobby Curtis  
Social Media Director  
E-mail: SocialMediaDir@rmoug.org

Kumar Annamnidu  
Member At Large  
E-mail: MembersAtLarge@rmoug.org

Vince Giasolli  
Special Interest Groups/Meetup Director  
E-mail: SigsDir@rmoug.org

Kathy Robb  
Board Member Emeritus  
Arisant, LLC  
E-mail: BoardME@rmoug.org

Heidi Kuhn  
Executive Director  
Voice Mail: (303) 948-1786  
Fax: (303) 933-6603  
E-mail: Admin@rmoug.org

John Jeunnette  
Board Member Emeritus  
E-mail: BoardME@rmoug.org

Peggy King  
Past President  
E-mail: BoardME@rmoug.org
### RMOUG Events Calendar

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
<th>Location</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/15/13</td>
<td>SQL&gt;Update Call for Articles &amp; Cover Photo Submissions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/9/13</td>
<td>QEW RMOUG Summer Quarterly Educational Workshop (QEW), Elitch Gardens in Denver CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8/15/13</td>
<td>SQL&gt;Update Article &amp; Cover Photo Submission Deadline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/15/13</td>
<td>SQL&gt;Update Publication Deadline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/22-26/13</td>
<td>Open World Oracle Open World 2013 in San Francisco, CA (<a href="http://www.oracle.com/openworld">www.oracle.com/openworld</a>)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/15/13</td>
<td>SQL&gt;Update Call for Articles &amp; Cover Photo Submissions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/15/13</td>
<td>QEW RMOUG Fall Quarterly Educational Workshop (QEW), TBA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/15/13</td>
<td>SQL&gt;Update Article &amp; Cover Photo Submission Deadline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/25/13</td>
<td>QEW RMOUG Autumn Quarterly Educational Workshop (QEW), Oracle DTC office in Denver CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12/15/13</td>
<td>SQL&gt;Update Publication Deadline</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/5-7/14</td>
<td>Training Days RMOUG Training Days 2014, Colorado Convention Center in Denver, CO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please note dates are subject to change. For the most current events calendar visit our website at [www.rmoug.org](http://www.rmoug.org).

---

### SPONSOR A QUARTERLY EDUCATION WORKSHOP AND RECEIVE A HALF PRICE AD IN SQL>UPDATE

<table>
<thead>
<tr>
<th>Breakfast</th>
<th>Discounted Ad Rate</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4 Page</td>
<td>$350.00</td>
<td>$175.00</td>
</tr>
<tr>
<td>1/2 Page</td>
<td>$350.00</td>
<td>$312.50</td>
</tr>
<tr>
<td>Full Page</td>
<td>$350.00</td>
<td>$500.00</td>
</tr>
<tr>
<td>Inside Cover</td>
<td>$350.00</td>
<td>$625.00</td>
</tr>
<tr>
<td>Back Cover</td>
<td>$350.00</td>
<td>$750.00</td>
</tr>
</tbody>
</table>

**Help RMOUG Members and Receive Recognition in An Upcoming Issue of SQL>Update**

Contact Carolyn Fryc - Programs Director - 720-221-4432 - cfryc@orsportal.com

---

### Tell Us About Yourself

Join us in sharing your Oracle experiences with other RMOUG members!

Tell us about your life, your job, or share your amusing Oracle anecdotes, tips and secrets!

Please submit all material to NewsletterDir@rmoug.org

---

### Tired of the I-25 Corridor?

60 Acres In Gorgeous Bijou Basin Less Than An Hour From DTC

- Grassy Meadows & Pines
- Excavated Road and Building Site
- Power & Phone At Property Line
- Water Rights
- One Hour From DIA
- Possible Owner Carry W.A.C.

30777 Magic Dog Circle, Kiowa, CO

$269,000

Pat Van Buskirk • Coldwell Banker, Parker Office
(303) 243-0737 Cell
www.patvanbuskirk.com pat@patvanbuskirk.com
Index To Advertisers

Arisant ................................................................. 2
Centroid............................................................ 22
DBAK................................................................. 32
Quarterly Education Workshop.......................... 31
Quarterly Education Workshop Sponsorship........ 30
Regis University .................................................. 2
Real Estate Coldwell Banker................................. 30

Reach A Targeted Oracle Audience Advertise Now!

A full page, full color ad in RMOUG SQL>UPDATE costs as little as 70 cents per printed magazine and even less for smaller ads.

Contact NewsletterDir@rmoug.org

Quarterly Education Workshops
August 9, 2013 • Elitch Gardens

Join us for our next Quarterly Education Workshop in August for presentations by Oracle at Elitch Gardens Water/Theme Park. RMOUG hosts quarterly workshops in May, August and November of each year with the fourth and largest educational event being Training Days in February. Learn about the newest technologies, gain more insight into Oracle techniques and enjoy the camaraderie of meeting with other Oracle professionals.

If you or your organization are interested in partnering with RMOUG to host an upcoming meeting, or to submit an abstract for presentation, please contact

Carolyn Fryc, Programs Director at ProgramsDir@rmoug.org

Watch RMOUG’s Web Page for August Training Topics  www.rmoug.org
Local Presence
Global Experience

People all over the world call us - because we fix systems that have a direct impact on business and revenue

Oracle Database
Data Strategy
Database Administration
Data Warehouse
Disaster Recovery
Performance Tuning

Oracle License Management
License Reviews
Software Compliance

Managed Services
Remote DBA
Remote Applications
Oracle Database Support
Oracle Applications Support

Oracle Applications
E-Business Suite R12
Financials
Supply Chain
Manufacturing
Procurement

www.dbak.com