

THE ORACLE ADVISORS FROM A DIFFERENT PERSPECTIVE: ARE YOU A MONKEY OR AN ASTRONAUT?

Karen Morton, Method R Corporation

"It is impossible to underrate human intelligence – beginning with one's own."

–Henry Brooks Adams

INTRODUCTION

Many of you recognize the names Alan Shepard, John Glenn, Scott Carpenter, and Gus Grissom. But do you remember Albert, Gordo, Able, or Miss Baker? These are the names of just a few of the monkeys who played a role in the evolution of the Space Program. But, not too many people think of monkeys when they think of what President Kennedy called the "...most hazardous and dangerous and greatest adventure on which man has ever embarked." [KENN1962]

Well, chimps went first. Monkeys were used to investigate the biological effects of space travel beginning in 1948. However, it wasn't until 1958 that the first monkey actually survived after traveling into space *and* returning to Earth.

In 1959, the first American astronauts, "the Mercury Seven," were announced by NASA, and the press and public soon adopted them as heroes. These men endured stringent physical and psychological training and were selected from an initial set of 508 well-qualified candidates. It was this group that, over the five-year life of the Mercury project, proved that human spaceflight was possible and paved the way for all further spaceflight including the Gemini and Apollo programs.

The monkeys who first were used to test the possibilities of spaceflight were highly trained, just as the astronauts were. But, the training and expectations for the two groups were entirely different. The monkeys were closely managed and trained to react to a set of likely occurrences. All that was expected of them was to simply be there and, if a known condition occurred, to make their trained, automatic response to that condition. For example, if a red light on the panel came on, they were trained to hit a specific red button in response. However, if anything other than the expected happened, they weren't expected to take appropriate action.

Obviously, the human astronauts had the ability to do more than just respond automatically. They could call on their wealth of knowledge, reasoning, and broad training to not just react by rote behavior, but to adjust to new and unknown circumstances. They could also act proactively as events occurred, not waiting until after they ended to follow a checklist of what to do to recover from the problem. Think of the Apollo 13 astronauts and their mission that was ultimately termed a "successful failure." Without the knowledge of the astronaut crew, they would have been lost; they'd have never made it back to Earth. [LOVE1994]

THE SELF-MANAGING DATABASE

Oracle has expended great effort to provide their customers with tools and product features to help simplify administration and increase efficiency of their databases. Starting in version 10, a set of components and related "advisors" were added to help DBAs and developers respond more quickly to performance issues in their database applications. From the Enterprise Manager interface, Advisor Central is our entry point to a host of advisors such as the Automatic Database Diagnostic Monitor, the SQL Tuning Advisor, the SQL Access Advisor, the Segment Advisor, and more. The problem these components were created to address—as Oracle views it—is a common belief among Oracle professionals that database performance optimization is a "complex, time consuming task and requires expensive specialist consulting skills"

[ORAWP2003, p3] to perform. This commonly held belief led Oracle to create these self-managing components to help make the job easier.

In their Oracle white paper entitled “The Self-Managing Database: Automatic Performance Diagnosis,” authors Graham Wood and Kyle Hailey point out that it is ridiculous to dream of a `_MAKE_SQL_RUN_FASTER` parameter and instead note: “Performance diagnosis and tuning is mostly about following a logical, methodical approach. Fortunately computers are pretty good at that!” [ORAWP2003, p4]

They go on to discuss their observations that many people spend a lot of time and effort fixing symptoms rather than getting to the root of performance problems that actually cause the symptoms. And they note that fixing symptoms often results in very little real performance improvement. The paper then proceeds to demonstrate the performance capabilities in Oracle Database 10g by comparing how a DBA might proceed to diagnose a performance problem before having access to the new tools and then how they would go about it with the new tools at their disposal. The example ultimately shows how much effort was likely to go into the “before” method and how little effort was required in the “after” method using the new capabilities.

I think these capabilities Oracle has given us are a certainly a great technological leap forward and I see how, in many cases, they can be used to identify root causes of problems more quickly and guide us to solutions more effectively than we could perhaps do on our own. But, here is where I return to my monkey vs. astronaut story line and the concern I have about Oracle becoming more self-managing. Oracle is the spacecraft, and it is getting smarter and more automated (like the Shuttle vs. the Apollo craft). But, no matter how smart the craft is, it still takes someone to run it. We must decide if we will be an astronaut or a monkey. In other words, as Oracle gets smarter, are we allowing ourselves to get dumber?

“BEFORE” AND “AFTER”

Even Oracle admits that the AWR and ADDM and related advisors can’t and won’t find every problem and suggest the correct fix every time. But the simple fact that these components can help in a large percentage of problem cases is what hooks us in. We rejoice in the fact that we no longer need to know how to get to the root of problems ourselves since the database can do it for us. And that’s where the “dumbing down” begins. If Oracle can do all the work for us, then all we need do is sit back and watch the Enterprise Manager Database Control panel until we are alerted to a problem, then simply follow the advise we are given, right?

The “after” example used in the white paper indicates that the DBA receives a complaint about the system being slow from a user and turns to the latest ADDM report for guidance. The finding as determined by ADDM is:

“SQL statements were not shared due to the usage of literals. This resulted in additional hard parses which were consuming significant database time.”

The recommended action to take, representing a 31% benefit (if the problem is corrected), is:

“ACTION: Investigate application logic for possible use of bind variables instead of literals. Alternatively, you may set the parameter "cursor_sharing" to "force".

RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were found to be using literals. Look in V\$SQL for examples of such SQL statements.” [ORAWP2003, p6]

With this information, the DBA now knows what to do and can immediately attempt the recommended actions following the rationale listed (in this case a specific problem SQL statement). What are bypassed are the 8–10 diagnosis steps the DBA would have taken prior to having ADDM available. What is also bypassed is the need for the DBA to be required to know how to perform the diagnosis manually. Sounds great, doesn’t it?

WHAT DO WE REALLY WANT?

As an educator, I consider it my mission to help teach students how to think for themselves and to understand how and why things work as they do. I want them well equipped with the foundational knowledge and confidence to face any performance challenge they may face. At the start of each class I ask my students to answer one question: are you here to be given the secret button to push or the checklist to follow, or be told step-by-step what to do to become a “performance tuning expert?” If they’re really being honest, everyone will say “yes!” Why not? Why wouldn’t we want to find the Holy Grail of performance tuning? Why wouldn’t we want to have a button, a list, or a formula that we follow every time that works without fail? Ah...it’s that last part where it starts to fall apart. We want something that works every time without fail. Frankly, if that something existed, we’d all be out of jobs and they could hire monkeys to do what we now do! If the database could really work this magic itself without anything more from us than to push a button or blindly follow a command, the need for our presence would be eliminated, or at the very least, reduced to such a degree that our value was greatly diminished. Monkeys make a heck of a lot less cash than astronauts do.

However, my students don’t *really* believe this magic formula exists, and they don’t *really* believe that by they’ll find on by attending my class, but they are there with grand hopes on the off chance that it will! They acknowledge that really, deep down they want the easy answer, the quick fix, the big red button to push when things go wrong that will make it all better...and they don’t really want to have to think. What they may not realize is that this desire to not have to think is at the root of “monkey-tizing” themselves! So, then I ask “what do you really want?”

If you distill what I said earlier about the monkeys used in the early years of the space program down to its essence, what you find is this: monkeys were expendable. They had just enough training to accomplish some basic tasks and when things went awry—as they did for over ten years before the first real success—they didn’t survive to try again another day. Now, I know this analogy may seem a bit strong, but I fear that there are way too many monkeys and not enough astronauts among us; and we’re convincing ourselves that it is OK.

When faced with the reality of what Oracle can do for us—which is quite a lot—do we stop learning the how’s and the why’s and only learn the what’s? Most folks will agree that even though they may initially think they want the magic formula, underneath it all, they want to be knowledgeable and confident in their ability to use that knowledge to be able to do what only humans can do—solve the seemingly unsolvable problems. We all have varying degrees of the Medieval knight archetype in us. We want to ride in on our thoroughbred stallion and save the day, then ride off into the sunset followed by the cheers and the accolades of our peers, our managers and our users.

WHEN THE ADVISORS MISLEAD

As helpful as the tools Oracle provides can be, these tools can be misused. I don’t mean misused in the sense of using them improperly, but I mean misused in the sense that they are used instead of and in place of detailed knowledge and understanding of the DBA or developer. Even if an ADDM report can point you to a problem, it’s not as simple as pushing a button in order to fix the problem. The report may accurately identify one or more causes of problems, but currently it will only suggest actions that must be taken. The next step to perform these actions and implement a fix is a task that must be accomplished by us.

But, what happens when the diagnosis and possible remedies we are pointed to do not address the problem at hand? What do we do if the issues Oracle has captured don’t lead to the solution to the real problem? This is certainly possible since the tools diagnose problems based on system-wide metrics. One thing we need to remember is that often the only way to diagnose a problem is to use a business task-focused approach. The steps that are followed in this approach are a restated version of Method R [MILL2003].

1. Determine the business task that is performing sub-optimally that you want to fix next.
2. Collect detailed statistics about how this task spends its time while it is performing poorly.

3. Execute repairs to reduce the task's response time and/or resources consumed, or prove that is it already as efficient as it can affordably be made.
4. If problems still exist, then go to step 1.

Many people believe that the downside to a business task-focused approach is that they do not have access to the information or the people with the information that can help them determine the tasks in need of optimization. It seems much easier to just focus on what the "system" has to tell us in order to find and fix problems. It's easier to review an ADDM report and watch the system-wide indicators using Enterprise Manager than to actually understand the business.

Here we are again, back to the monkeys. It's easier for a monkey to watch a panel and execute a prescribed action. There is little thinking involved. There is little need to question or probe. But, when the prescribed action doesn't successfully avert the problem, what does the monkey do? Nothing. He can't do anything because he has neither the needed knowledge nor the understanding of even what questions to ask. Enter the astronaut.

The astronaut considers the suggested recommendations, but based on knowledge of the problem task, he can take another approach. Consider again the Apollo 13 mission. Their spacecraft was damaged but the total scope of the damages was not visible to their system diagnostics and other tools. It became obvious fairly quickly that their original mission would not be completed—they would not be able to land on the moon. The mission changed to simply getting safely back to Earth. They had to gather detailed information and focus on the tasks that would get them home. And, in the end, they made it. Yes, they had assistance from advisors at mission control, but it was their ability to think on their own that was the ultimate key to them getting successfully home.

The cost to your business of following the wrong advice can be substantial. Blindly following advice that doesn't directly impact where your business is suffering can cost hours, days, or even weeks of time and money. If you are working on the wrong thing, the real problem remains and the business continues to suffer.

IT TAKES HUMAN INTELLIGENCE

For all the good ADDM and the advisors provide, there is currently no substitute for the knowledge and experience of the human in front of the console. Thank goodness! The advisors and tools can help us, but the best tool we have is our own intelligence.

As an example of how human intelligence can often triumph over simply following advice, let's look at a query that was reported as performing poorly by a user.

```
SQL> get cust_disc.sql
 1  select cust_no, tot_orders_amt, 0 as disc_rate
 2  from
 3  (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt
 4   from customer2 c, ord2 o
 5   where c.cust_no = o.cust_no(+)
 6   group by c.cust_no ) t2
 7  where tot_orders_amt = 0
 8  union
 9  select cust_no, tot_orders_amt, .1 as disc_rate
10  from
11  (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt
12   from customer2 c, ord2 o
13   where c.cust_no = o.cust_no
14   group by c.cust_no ) t2
15  where tot_orders_amt > 0 and tot_orders_amt <= 100000
16  union
```

```

17 select cust_no, tot_orders_amt, .15 as disc_rate
18 from
19 (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt
20   from customer2 c, ord2 o
21  where c.cust_no = o.cust_no
22   group by c.cust_no ) t2
23 where tot_orders_amt > 100000 and tot_orders_amt <= 500000
24 union
25 select cust_no, tot_orders_amt, .2 as disc_rate
26 from
27 (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt
28   from customer2 c, ord2 o
29  where c.cust_no = o.cust_no
30   group by c.cust_no ) t2
31* where tot_orders_amt > 500000

```

I executed the code in my test instance and then located the query under Top Activity of the SQL Tuning Advisor in Enterprise Manager to review execution statistics and also to check on the advisor's recommendations for improvement. I used the SQL Tuning Advisor to create a comprehensive analysis of the statement and received this finding and recommendation:

ORACLE Enterprise Manager 10g Database Control

Database Instance: hotsoos > Advisor Central > SQL Tuning Results: SQL_TUNING_acase > Recommendations for SQL ID:gqjckur6q41c4

Logged in As SYS

Recommendations for SQL ID:gqjckur6q41c4

Only one recommendation should be implemented.

SQL Text
select cust_no, tot_orders_amt, 0 as disc_rate from (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt from customer2 c, ord2 o where c.cust_no = o.cust_no(+) group by c.cust_no...

Select Recommendation

Select Type	Findings	Recommendations	Rationale	Benefit (%)	New Explain Plan
Restructure SQL	An expensive "UNION" operation was found at line ID 1 of the execution plan.	Consider using "UNION ALL" instead of "UNION", if duplicates are allowed or uniqueness is guaranteed.	"UNION" is an expensive and blocking operation because it requires elimination of duplicate rows. "UNION ALL" is a cheaper alternative, assuming that duplicates are allowed or uniqueness is guaranteed.		

As you can see, the advisor recommended that I replace the UNION operations with UNION ALL. So, I modified my query, executed it again and then checked what the advisor had to say.

ORACLE Enterprise Manager 10g Database Control

Database Instance: hotsoos > Advisor Central > SQL Tuning Results: SQL_TUNING_acase_unionall > Recommendations for SQL ID:fa6236y9maty

Logged in As SYS

Recommendations for SQL ID:fa6236y9maty

Update Message
There is no recommendation.

SQL Text
select cust_no, tot_orders_amt, 0 as disc_rate from (select c.cust_no, nvl(sum(o.total_order_price),0) as tot_orders_amt from customer2 c, ord2 o where c.cust_no = o.cust_no(+) group by c.cust_no...

Select Recommendation

Select Type	Findings	Recommendations	Rationale	Benefit (%)	New Explain Plan

Nothing. So, I suppose my problem is fixed, right? Let's compare the before and after by using SQL*Plus and reviewing the AUTOTRACE output:

Before:

Elapsed: 00:00:01.51

Execution Plan

Plan hash value: 3344621431

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		160	2560	191 (78)	00:00:03
1	SORT UNIQUE		160	2560	191 (78)	00:00:03
2	UNION-ALL					
* 3	FILTER					
4	HASH GROUP BY		10	160	48 (11)	00:00:01
* 5	HASH JOIN OUTER		12890	201K	44 (3)	00:00:01
6	INDEX FAST FULL SCAN	CUST2_PK	1000	5000	2 (0)	00:00:01
7	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 8	FILTER					
9	HASH GROUP BY		50	800	48 (15)	00:00:01
10	NESTED LOOPS		12890	201K	44 (7)	00:00:01
11	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01
* 13	FILTER					
14	HASH GROUP BY		50	800	48 (15)	00:00:01
15	NESTED LOOPS		12890	201K	44 (7)	00:00:01
16	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 17	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01
* 18	FILTER					
19	HASH GROUP BY		50	800	48 (15)	00:00:01
20	NESTED LOOPS		12890	201K	44 (7)	00:00:01
21	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 22	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01

Predicate Information (identified by operation id):

3 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)=0)
5 - access("C"."CUST_NO"="O"."CUST_NO"(+))
8 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>0 AND
 NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)<=100000)
12 - access("C"."CUST_NO"="O"."CUST_NO")
13 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>100000 AND
 NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)<=500000)
17 - access("C"."CUST_NO"="O"."CUST_NO")
18 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>500000)
22 - access("C"."CUST_NO"="O"."CUST_NO")

Statistics

0 recursive calls
0 db block gets
39419 consistent gets
0 physical reads
0 redo size

```

26289 bytes sent via SQL*Net to client
1107 bytes received via SQL*Net from client
68 SQL*Net roundtrips to/from client
1 sorts (memory)
0 sorts (disk)
1000 rows processed

```

After:

Elapsed: 00:00:01.50

Execution Plan

Plan hash value: 927543932

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		160	2560	184 (77)	00:00:03
1	UNION-ALL					
* 2	FILTER					
3	HASH GROUP BY		10	160	46 (7)	00:00:01
* 4	HASH JOIN OUTER		12890	201K	44 (3)	00:00:01
5	INDEX FAST FULL SCAN	CUST2_PK	1000	5000	2 (0)	00:00:01
6	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 7	FILTER					
8	HASH GROUP BY		50	800	46 (11)	00:00:01
9	NESTED LOOPS		12890	201K	44 (7)	00:00:01
10	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01
* 12	FILTER					
13	HASH GROUP BY		50	800	46 (11)	00:00:01
14	NESTED LOOPS		12890	201K	44 (7)	00:00:01
15	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 16	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01
* 17	FILTER					
18	HASH GROUP BY		50	800	46 (11)	00:00:01
19	NESTED LOOPS		12890	201K	44 (7)	00:00:01
20	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01
* 21	INDEX UNIQUE SCAN	CUST2_PK	1	5	0 (0)	00:00:01

Predicate Information (identified by operation id):

```

2 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)=0)
4 - access("C"."CUST_NO"="O"."CUST_NO"(+))
7 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>0 AND
          NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)<=100000)
11 - access("C"."CUST_NO"="O"."CUST_NO")
12 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>100000 AND
          NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)<=500000)
16 - access("C"."CUST_NO"="O"."CUST_NO")
17 - filter(NVL(SUM("O"."TOTAL_ORDER_PRICE"),0)>500000)
21 - access("C"."CUST_NO"="O"."CUST_NO")

```

Statistics

```

-----
      0 recursive calls
      0 db block gets
39419 consistent gets
      0 physical reads
      0 redo size
24813 bytes sent via SQL*Net to client
  1107 bytes received via SQL*Net from client
      68 SQL*Net roundtrips to/from client
      0 sorts (memory)
      0 sorts (disk)
  1000 rows processed

```

Even with this simple comparison of the before and after query executions, the query doesn't appear to have been noticeably improved. I note a response time savings (for this one test) of .01 seconds (before = 1.51 seconds and after = 1.50 seconds). The consistent gets are the same and there is one less memory sort for the after version.

But, according to what the advisor can tell me, there is nothing else to be done to improve this query. However, my business user demands better performance. The query, when executed in production where the data volume is 10 times larger than development, is still "slow."

Using my own knowledge, I determine that I can rewrite the query to use fewer resources as follows:

```

1  select c.cust_no, sum(o.total_order_price) tot_orders,
2         (case when sum(o.total_order_price) = 0 then 0
3             when sum(o.total_order_price) <= 100000 then .10
4             when sum(o.total_order_price) <= 500000 then .15
5             when sum(o.total_order_price) > 500000 then .20
6             else 0 end) as disc_rate
7  from customer2 c, ord2 o
8  where c.cust_no = o.cust_no(+)
9* group by c.cust_no

```

The test results show excellent improvements in both response time (down to .29 seconds and resource consumption (only 191 consistent gets instead of 39,419).

Elapsed: 00:00:00.29

Execution Plan

Plan hash value: 399043451

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1000	16000	46 (7)	00:00:01
1	HASH GROUP BY		1000	16000	46 (7)	00:00:01
* 2	HASH JOIN OUTER		12890	201K	44 (3)	00:00:01
3	INDEX FAST FULL SCAN	CUST2_PK	1000	5000	2 (0)	00:00:01
4	TABLE ACCESS FULL	ORD2	12890	138K	41 (0)	00:00:01

Predicate Information (identified by operation id):

2 - access("C"."CUST_NO"="O"."CUST_NO"(+))

Statistics

0 recursive calls
0 db block gets
191 consistent gets
0 physical reads
0 redo size
26279 bytes sent via SQL*Net to client
1107 bytes received via SQL*Net from client
68 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
1000 rows processed

The moral to this story is that the advisors can't do it all for you. This is a very simple example, but it is intended to show that the advisors can only provide recommendations from a pre-defined set of possible responses to certain situations. Think of how the advisors respond as being like how Google performs a search: you ask for certain keywords and you get back a list of possible matches. You're not guaranteed that the matches are really what you want or need. The list isn't perfect, but it's a starting point. You may find that you need to change your criteria to get a better list or you may find that you have to look at many suggestions from the list to find one that works for you—or to find out that none of them works.

In this example, the fix to the query's performance problem was not found via the advisor. But, if I hadn't had the knowledge to write the query differently, what other choice would I have had? I had to have some skills and knowledge of my own to find the solution to the problem.

CONCLUSION

Follow this advice: don't let Oracle's increasingly smart advisors become the dictator of your career. These tools are there to facilitate our knowledge, not replace it. Use the tools wisely but not as a substitute for your own lack of knowledge.

The bottom-line is about what you choose to do for yourself. And in my opinion, the key is about a thirst for knowledge...not just the knowledge itself. It's about how we strive to learn, understand, and know how and why things work that makes the difference. It is the fulfillment of the desire to *know* that leads us to the knowledge we need to evolve from monkey to astronaut and to be able to solve the problems that seem unsolvable. It's not necessarily the easiest path to take. It requires time, diligence and commitment. But the result is well worth the effort. The result is that you become a respected, invaluable resource with the skills and confidence to impact your business in a positive way. You become an astronaut!

REFERENCES AND FURTHER READING

[KENN1962] John F. Kennedy, *Address at Rice University on the Nation's Space Effort*, Sept. 12, 1962.

[LOVE1994] J. Lovell and J. Kluger, *Apollo 13*. New York: Pocket Books, 1994.

[MILL2003] C. Millsap and J. Holt, *Optimizing Oracle Performance*. Sebastopol CA: O'Reilly, 2003. (Chapter 1 available at www.oreilly.com/catalog/optoraclep/chapter/index.html).

[ORAWP2003] G. Wood and K. Hailey, *The Self-Managing Database: Automatic Performance Diagnosis*, an Oracle white paper, at www.oracle.com/technology/oramag/oracle/04-may/self10g.html, Nov. 2003.