Quote of the day…

Not that anyone should care by now (we are in 10g times after all) ... but that "Hierarchy" package presented last year as a 8i method for doing what sys_connect_by_path does is a bug waiting to happen. ***One needs to understand how it works in order to use it safely.***

Quote of the day…

Not that anyone should care by now (we are in 10g times after all) ... but that "Hierarchy" package presented last year as a 8i method for doing what sys_connect_by_path does is a bug waiting to happen. ***One needs to understand how Oracle works in order to use it safely.***

Things we *think* we ‘know’

Suppose everything we learned, we learned from TV.

Sort of like learning everything we know from the internet.

Things we 'know'

- Some things I've learned from TV
  - Ventilation systems of any building are the perfect hiding place. Not only that, but you can get anywhere in the building using them.
  - Cars and trucks that crash almost always burst into flames
  - When you wake up from a nightmare, you will always sit bolt upright, in a sweat, and breath heavy
  - Creepy music coming from a graveyard always mandates investigation

Things we ‘know’

- Some things I’ve learned from TV
  - When you are outnumbered in a martial arts fight – your enemies will always wait patiently to attack you one by one (waiting for you to knock out their predecessor of course)
  - Having a job of any sort will cause all fathers to forget their son’s/daughter’s birthday.
  - All bombs have very large, red LED display so you know exactly when they will go off
  - When they are alone, all foreigners prefer to speak English to each other

What happens when we “know”

- My car shook at 58-63 mph
- Everyone *knows* when that happens – it must be that your wheels are out of balance/alignment
- Took it in, said balance those tires – and they did.
  - Result – not encouraging, I convinced myself it was a little better but they must not have balanced them right
- So, took it elsewhere, same story
  - Result – the same
- Took it to another place and described the *problem*
  - Wheel was bent, all of the balancing in the world would not help

What happens when we “know”

- My system is going slow
- Everyone *knows* when that happens – it must be that you’re low on CPU or files need be moved
- Added CPU, moved files
  - Result – not encouraging, I convinced myself it was a little better but must not have added/moved enough
- So, tried again, same story
  - Result – the same
- Looked at the *problem*
  - Massive locking/enqueue problem. CPU made it worse, moving files would do nothing

You know more than you think

Suppose that one day you are driving to work and end up arriving late for an important meeting. You aren't able to present your revolutionary idea, so your clients aren't going to use it. You're frustrated by your tardiness and vow to never make the same mistake again. So how do you diagnose the cause in order to avoid a replay? How about this checklist?

- ***Check the car's surface for imperfections***, because surface imperfections can account for a difference of 1 percent or even greater in the car's top speed.
- ***Check the wheel alignment***, because an incorrect camber, caster, or toe angle can cause the car to handle poorly, costing time.
- ***Test the engine to ensure that it is producing 99 percent or more of its rated horsepower***. If it is not, consider rebuilding or replacing the engine.

No, you wouldn't use this checklist; that would be ridiculous. You'd probably diagnose the problem in a completely different way, by asking yourself just one simple question: What took me so long?

Millsap http://otn.oracle.com/oramag/oracle/04-jan/o14tech_perf.html

Quiz Time!

- All of these questions have very easy answers
- Or do they?
- What is ‘obvious’

Things we *think* we 'know'

So, what was the point….

Artemus Ward once wrote, "It ain't so much the things we don't know that get us into trouble. It's the things you know *that just ain't so*."

Updated for 2005

- It ain't so much the things we don't know that get us into trouble.
- It's the things you know
  - *that just ain't so or*
  - *just ain’t so* ***anymore*** *or*
  - *just ain’t* ***always*** *so*

Things Change

- Select INTO
- IN vs EXISTS
- NOT vs NOT EXISTS
- Where nvl(:bv,column) = column
- Array Fetching ***(af.sql)***
- ***And so on…***

‘Obvious’ things “I learned” from the net

- The first thing to do to tune is move files, re-org tables, and rebuild everything in site
  - No need to find the root cause of performance, everyone knows these work
  - Unless…
    - You switched from ANALYZE to DBMS_STATS because you *know* that is the preferred method
    - You *know* method_opt=> null goes “faster” in 8i (but surprisingly, not in 9i…)
    - You don’t *know* why it goes faster, just does
  - It ain’t always so….

‘Obvious’ things “I learned” from the net

- PCTINCREASE should be 1
  - Holdover from dictionary managed tablespaces
  - Brought about because SMON won’t coalesce adjacent free extents in a tablespace with a default pctincrease of 0
  - The reason SMON wouldn’t do that is because you should have set initial=next, obviating the *need* for expensive coalescing.
  - That *would have* had the nice side effect of removing fragmentation
  - But setting pctincrease to 1 killed all of that.
  - It ain’t so anymore (if it ever was – see next slide…)

PCTINCREASE 1

```
ops$tkyte@ORA817DEV> create table t ( x int )
  2 tablespace testing storage ( initial 1k pctincrease 1 minextents 100);
Table created.

ops$tkyte@ORA817DEV> select blocks, count(*) from user_extents
  2 where segment_name = 'T' group by blocks order by 1;


    BLOCKS   COUNT(*)       BLOCKS   COUNT(*)       BLOCKS   COUNT(*)
---------- ----------   ---------- ----------   ---------- ----------
         2          1           66          2           87          1
         5          1           67          1           89          1
        10          5           69          1           90          1
        15          5           70          3           91          1
        20          5           71          1           93          2
        25          5           73          1           94          1
        30          5           74          1           95          2
        35          5           75          2           97          1
        40          5           76          1           98          1
        45          5           78          1           99          1
        50          5           79          1          100          2
        55          5           80          2          101          1
        57          1           81          1          103          1
        59          1           83          1          105          1
        60          3           84          1
        62          1           85          2       49 rows selected.
        64          1           86          1
        65          1
```

‘Obvious’ things “I learned” from the net

- Tables should have one (or few) extent(s)
  - Nugget of Truth with a dictionary managed tablespace and objects you DROPPED or TRUNCATED
  - Releasing extents *was* expensive
  - Allocating extents frequently *was* expensive
  - But, if a table was already in N thousand extents – would putting it in a few make it better?
    - You’d have that horribly expensive release
    - A reload
    - And nothing else
  - Consider Index Access
  - Consider Full table scan
  - It ain’t so anymore (if it ever was…)

‘Obvious’ things “I learned” from the net

- Separate Indexes from Data
  - “*Tablespaces containing tables, and tablespaces containing indices corresponding to them, would be like locating matter and antimatter on the same spindle*”
  - Why? It isn’t like Oracle accesses them in parallel
  - Nugget of truth buried in history, old history, long ago history.
    - Attempt to spread IO out
    - Could have been done with lots of small extents (round robin)
    - *But that would conflict with the previous slide!*
    - It ain’t so anymore

‘Obvious’ things “I learned” from the net

- Google “oracle tuning tips”
  - First hit = Calculate buffer cache hit ratio in the database. Make sure it is more than 80 for an OLTP environment and 99 is the best value.
  - Appears to be the top 10 (not just in)
  - It is a metric, not a goal
  - It ain’t so (ever)
  - “Hey, we increased the buffer cache in our data warehouse, all of a sudden cache buffers chains latches are 35% of our wait time!”

‘Obvious’ things “I learned” from the net

- Remove large-table (or even *all)* full table scans
  - Generally true but…
  - Becomes a mantra for many and isn’t always the case
    - Spend inordinate amount of time hinting and playing games to get indexes used
    - Only to never measure that the full scan was superior
    - Simple example coming up
  - *And* in a warehouse “it just ain’t so”
  - Full scans – something to look for, not something to make extinct
  - It ain’t so

‘Obvious’ things “I learned” from the net

- Remove large-table full table scans

```
I want to update a table with one go on online system. A table
has 200,000 records with 110 columns. When i give the update
command it takes ~ one hrs.I don't know why it is taking so much
time even though I made sure an index is created on app_flg –
that particular field. app_flag has only two values Approved or
unapproved.By default is unapproved.

select count(*),app_flg from test;

170,000   approved
 30,000   unapproved

update test set app_flg='APPROVED' where app_flg='UNAPPROVED'

it took 1hr to update the records and other application online
users processing got slowed down and locks started to occur on
the table.
```

‘Obvious’ things “I learned” from the net

- Remove large-table full table scans

I have solved that problem, the index was dropped. The updates are as fast as you can think.

‘Obvious’ things “I learned” from the net

- Issue frequent commits
  - Theories behind it:
    - Enhances performance (faster)
    - Resource utilization is minimized
    - In other databases that employ read locks…
    - Wonder why JDBC and ODBC autocommit after each statement by default?
  - Realities
    - Leads to ORA-1555
    - Destroys transactional integrity
    - Runs slower
    - Generates more overall undo and redo
  - Your transaction size is driven by one thing – your business rules.
  - It ain’t so

‘Obvious’ things “I learned” from the net

- Indexes need to be rebuilt frequently or on a schedule
  - Since space is never reused (myth)
  - And they get unbalanced (myth)
  - It’ll make them smaller (sometimes yes, sometimes no)
    - 10,000 leaf nodes with 1 entry/leaf -> smaller
    - 10,000 *packed* leaf nodes -> *bigger*
    - Most probable is no change at all
  - It ain’t so
  - www.actoug.org.au/Downloads/oracle_index_internals.pdf

‘Obvious’ things “I learned” from the net

- Most selective fields should go first
  - (yes, the doc bug was filed to fix the question in the docs!)
  - I’d say least selective should (skip scans, compression)
  - In any case – where a=:a and b=:b performs the same regardless of selectivity and ordering of A,B in the index
  - **HOW** you use the index dictates column ordering
    - CREATE TABLE T as SELECT * FROM ALL_OBJECTS *and ask the following*
      - o Get details on Scott’s EMP Table
      - o Show me the indexes owned by Scott
      - o Show me Scott’s objects
      - o Object Name is “most selective”, but should go dead last
  - It ain’t so…

‘Obvious’ things “I learned” from the net

- Views are evil things that slow down performance
  - Nugget of truth – teeny tiny nugget
  - Views when improperly applied to a problem may lead to sub-optimal query performance. BUT
    - It is typically an apples/oranges comparison
    - That is, the result from the view with extra predicates layered on top are *different* then the results from the query without the view
  - Views are tools, no tool is 100% evil or 100% good
  - It ain’t so…

‘Obvious’ things “I learned” from the net

- Count(1) is superior to count(*)
  - Funny thing is – in 8i, count(1) was optimized to be count(*) internally
  - Count(1) was slower (probably why the optimization was made, everyone did it that way and it was slower)
  - It just ain’t so and never was

‘Obvious’ things “I learned” from the net

- Primary keys must have a unique index
  - Used to be true (in 7.3 and before!)
  - Changed in 8.0 with deferrable constraints
  - Can be very useful
    - MV refreshes
    - Update Cascades
  - It just ain’t so anymore

‘Obvious’ things “I learned” from the net

- It is always the database (*always*)
  - Time and attendance application
  - Worked great most of the time
  - Just ported from Informix to Oracle
  - Biggest install to date.
  - “works great on Informix”
  - Was getting totally locked up on Oracle
  - What was wrong…

Amazing things I’ve heard that people “know”

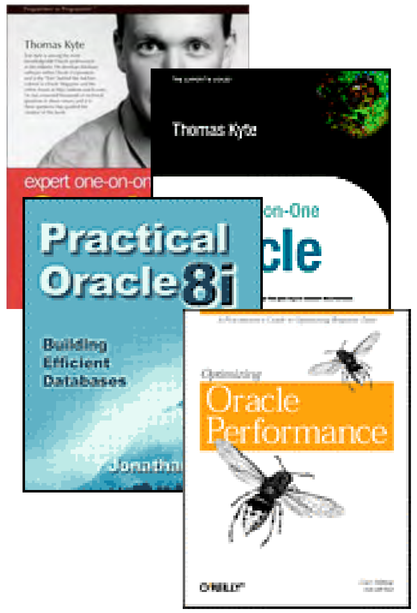
- *Shared server shouldn’t work that way*.
- *No, we don’t have a backup of rollback*. That’s not our data – why should we need that just to recover our database (they “knew” they didn’t need to back that up)
- *No, we just copied the datafiles* – we heard that putting a tablespace in backup mode generated extra redo so we avoided that overhead
- *We just know we can recover*, we don’t need to actually try it out.

5 questions

- What does your group know that it knows it knows
  - We know backup & recovery provably so
- What does your group know that it doesn’t yet know it knows
  - We should have been able to predict the need for “resource x”, had we been watching
- What knowledge does your group lack that it knows it lacks
  - Perhaps the easiest of all if you are honest
- What knowledge does your group lack that it doesn’t know it lacks
  - What haven’t you tested?
- What does your group know that “just ain’t so”
  - Look at your “standard operating procedures”

There are lots of “experts” out there

- Make them prove everything
- Statements that should raise your eyebrows:
  - It is my opinion...
  - I claim...
  - I think...
  - I feel…
  - I *know*
  - It always worked that way
- Things change, expect that
- It only takes a single counter case
- Nothing is 100% good, nothing is 100% evil
  - It is about understanding *when* to do *what* and as importantly – when not to do what



Questions

and

Answers