Tuning SQL without the Tuning Pack

John Larkin
JP Morgan Chase

Who am I

- Originally a mainframe COBOL programmer
- DBA for the last 23 years, the last 15 with Oracle.
  - UNIX (Solaris, Aix, Windows, Linux)
- Recently completed a year-long project to split a mid-size datamart into an ETL/staging database and a separate Published database.
- Employed by chemical manufacturers, publishers, retailers as both employee and independent contractor.
- 10g OCA
- Contact – john.x1.larkin@chase.com / john.larkin1@comcast.net

Overview

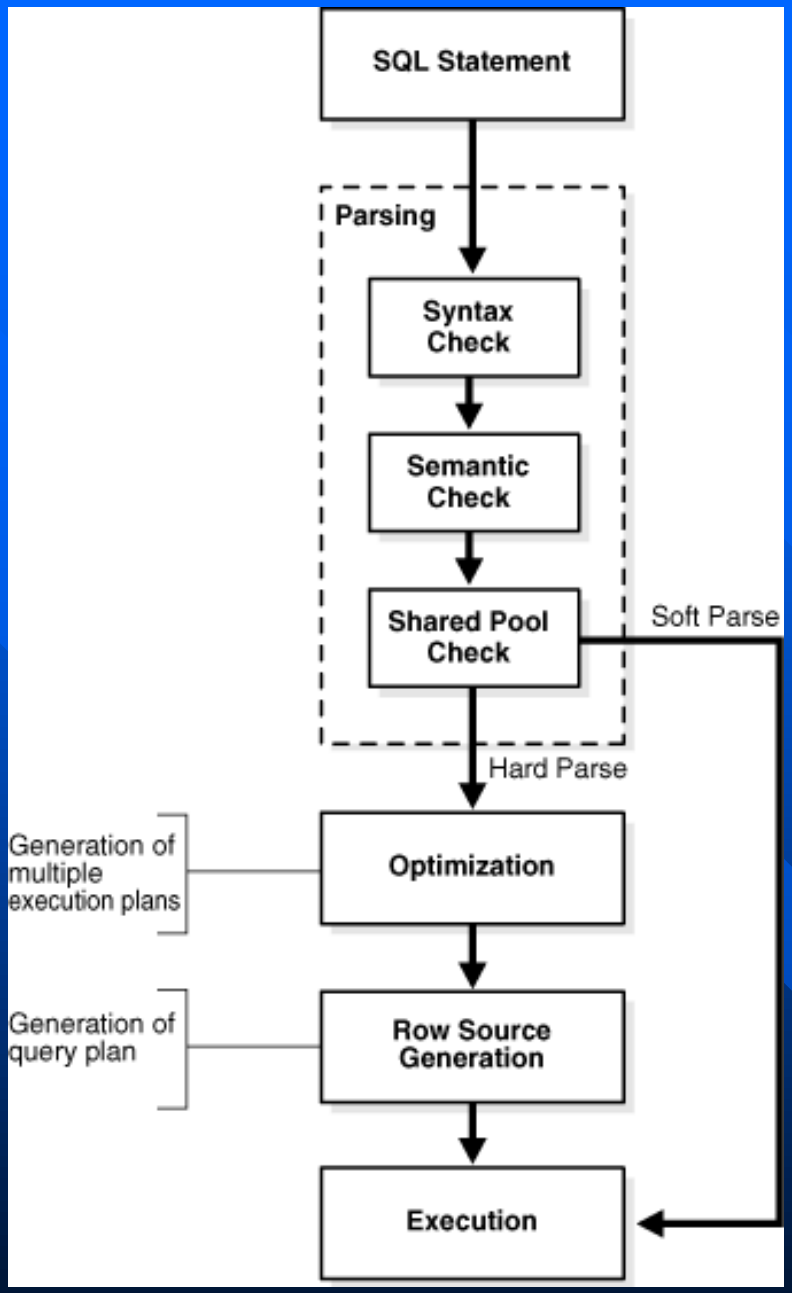
- SQL does not always perform consistently
  - Performs well in QA
  - sub-optimally in production
- How do we fix it ?
  - Money – memory and CPU's
  - Enterprise Manager Tuning Pack
    - » more money - maybe

Scope

- What we will cover
  - » Basic concepts and tools.
  - » SQL*Plus
  - » SQL execution plans
  - » Some statistics
  - » Hints
- What we will not cover
  - » No Enterprise Manager in this presentation.
  - » “SET EVENTS 10046”

Execution Plans

- Execution Plans
  - SQL statements are parsed
  - correct syntax
  - semantic correctness
    - » table and column names
    - » verify permissions
  - Shared_Pool check
    - » hard parse - result is a set of query blocks
    - » soft parse
      - skips row source generation
      - execution

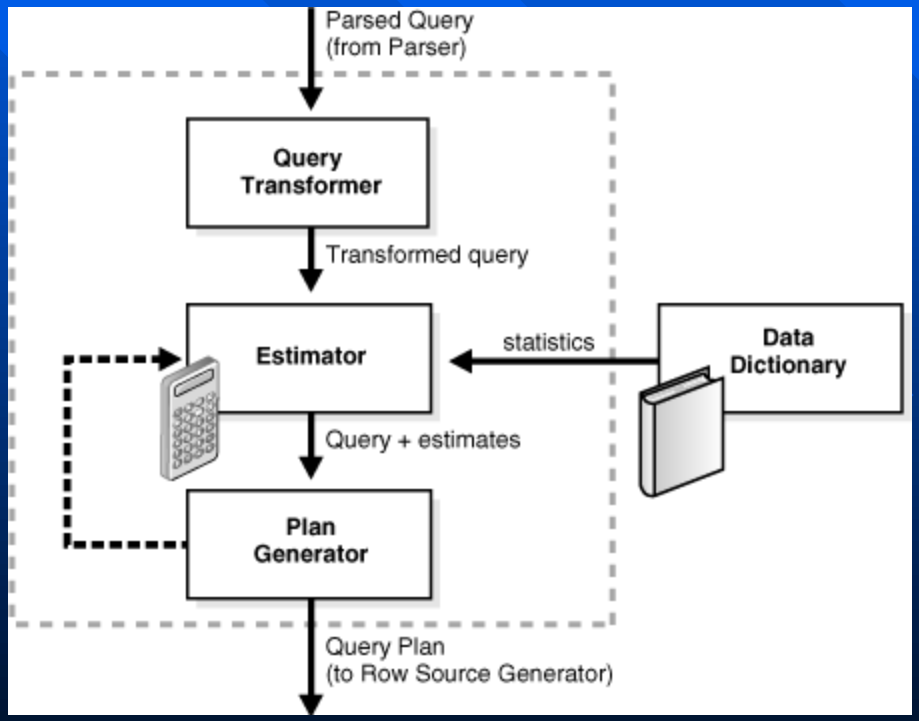

SQL Statement
Parsing
Syntax Check
Semantic Check
Shared Pool Check
Soft Parse
Hard Parse
Generation of multiple execution plans
Optimization
Generation of query plan
Row Source Generation
Execution

Execution Plans

- The optimizer, or CBO.
  - query_rewrite - query transformer make simpler equivalent forms.
  - Generates potential plans
    - » access paths and hints.
  - Estimator calculates cost for each
    - » dictionary statistics or estimate.
    - » Cost - proportional to resource consumption.
      - CPU consumption
      - disk throughput
      - generates measures for Selectivity and Cardinality.
        percent of rows returned from the row set
        – query predicate

Execution Plans

- Cardinality -number of rows to be processed (estimated) by an operation.
- init.ora parameters
- optimizer_index_cost_adj,
- db_file_multiblock_read_count
- sort_area_size



Execution Plans

- Plan Generator
  - » compares costs (relative)
  - » chooses the lowest-cost plan.
- Row Source Generator
  - » Receives optimal execution plan
  - » produces an iterative binary program
    - the query plan, with a series of steps
    - each step returns a row set
      - used by the next step
      - the last step returns the row set to the application.
  - » Row SOURCE - a Row SET returned from a step in the execution plan along with a control structure that can iteratively process the rows.

Execution Plans

» produces a row source tree

- collection of row sources
  - shows an ordering of tables
  - access method for each table
  - join method for tables when needed
  - data operations – filter / sort / aggregation.

– SQL Execution - the only mandatory step for DML.

» SQL engine executes each row source

» execution plan - read right to left , bottom to top.

- Start with the right-most entry in the “table” that is the farthest down.
- Data retrieval starts here.

Getting the PLAN

- Explain Plan syntax or Autotrace in sqlplus.
  - Not perfect
- **EXPLAIN plan for <sql statement> set statement_id = ‘myplan’**
  - does not execute the query
  - Parses
  - submits it for optimization
  - row source generation.
  - stored in sys.plan_table.
  - query plan_table for the execution plan.

Getting the PLAN

- Select * from TABLE(DBMS_XPLAN.Display).
  - » named plan
  - » child cursors – DBMS_XPLAN.Display_Cursor.
  - » for the last statement run in the current session.
  - » pull the information for any plan that is still loaded by providing the SQL_ID and optionally the child number.
  - » Not always the plan you will execute
    - No direct access to the user account
    - separation of duties
    - Execution environment may not look like the job owner
    - data / metadata differences.
    - sort_area_size
    - time of day - batch loads then gather statistics.

Getting the PLAN

- **set AUTOTRACE TraceOnly**
  - execution plan without execution
  - Other options
  - two main choices
    - » Don't execute the sql –
      - like a standard EXPLAIN PLAN
    - » run the statement to completion
      - more details about the SQL statement
        - difficult if long running. Isn't that why we're here?
        - discuss workarounds later
  - good for getting an execution plan at the start

Getting the PLAN

- subtle difference
- TRACEONLY option
  - » Poor wording
    - Assume you only get a trace as output
    - Might assume that it will return quickly
    - Must run to completion - “ONLY” refers to what you will receive from the query.
    - “ONLY” get the trace and plan returned to the screen.
    - the data is what you will not see.
    - No scrolling
- What about what is already running?
  - » later

STATISTICALLY Speaking :

- Can estimate “Statistics” when costing
- Best to provide accurate statistics 10g
- 10g STATISTICS_LEVEL = TYPICAL
- default - “GATHER_STATS_JOB”
- problematic if mid-stream
  - can leave you with tables that are “empty”
  - datamart or data warehouse, turned off.
  - do not mix ANALYZE and the new stats
  - dbms_stats.gather_table_Stats(‘SCOTT’, ‘EMP’, estimate_percent => 15);

STATISTICALLY Speaking :

- Can gather system stats
  - positive or negative - be careful.
- SET EVENTS 10046 - out of scope
  - Provides very good wait based information
  - Trace file in the databases udump directory.

What's the PLAN?

- AUTOTRACE and Explain Plans
- Execution plan plus statistics and data
  - the first thing that you see.
  - AUTOTRACE is based on the SQL running to completion to obtain a plan and statistics.

- @TRACE_vs_Explain.txt (18)

What's the PLAN?

- Reading from bottom-right to top-left.
  - Index fast full scan using the PK_Databases index.
  - expect one or two columns in the select list
    - » Index Organized Table
      - all of the data is available
      - Sorting adds additional step.

- @Lg_running.txt

What's the PLAN?

- Grab SQL from v$sql_plan
  - list of active sessions
  - Find session of interest. (username, login time, osuser, event, program - sqlplus / SqlNav, Toad).
  - Get SID and use sys.V_$PX_Session to find the SID for the Query Coordinator.
  - Use this to obtain the SQL and the current execution plan.
- Several more steps - remote objects(db link)
  - significant cost

What's the PLAN?

- From the 2 row sources
  - » data is Hash Join'ed
  - » sent parallelly into a buffer sort
  - » finishing the "Select" part of the CTAS
- Data loaded into the new table in parallel.

- Missing - list of temporary segments
  - » monitor progress of a load.

What's the Problem?

- Identifying bad plans
  - key indicators that almost always mean trouble
  - Look for the "guaranteed" problems
    - » Start with database links
      - need fully qualified object names
      - hidden links in synonyms – ask the developers
  - Run an explain plan
    - » if it's running, compare them
      - Any discrepancies ? Track them down.
        - logon triggers, different database, bind peeking.

What's the Problem?

- Move on to problem determination.
- Looking for database links and nested loops
  - Top 2 trouble makers for me
  - Remote table access problematic for the optimizer
    - » appears to treat them as a black box,
      - diplays the information in a manner that suggests that.
      - may have problems accessing statistics across the database link or in mating that information with the statistics from the local objects.
  - Nested Loops
    - » Tom Kyte despises looping – "Use a SQL statement"
    - » The more data you try to push the worse they perform
      - convert Nested Loops to Hashing.
- Sample - 27 hours before it died with a snapshot too old
- @EXPLAIN_Bad_SQL.log

What's the Problem?

- Nested Loop
  - try to get rid of that by first
  - Use DRIVING_SITE hint.
    - » treat the instance where the specified table resides as if it was the site from which the query was submitted.
    - » runs the query from that location.
    - » pick the row source with the largest amount of returned rows as the driving site.
    - » reduce the amount of data pushed through the network
    - » ran for more than 24 hours now runs in less than 4.
    - » @Run_Corrected_SQL.log

What's the Problem?

- database link is hidden within a synonym
  - decompose the synonym into it's base format
  - force the driving_site hint
  - often used with views that will not behave
  - Recent results by a colleague
  - Watch the query execute
    - » V$_session_LONGOPS
      - estimated time to completion
      - early indicator of whether your tuning works
      - Shorter turnaround
      - watch query execute on both sides of the database link.

What's the Problem?

- use v$sort_usage
  - Sorting - get an idea of where the query is in the overall process.
- parallelize a query
  - optimizer chooses to not parallelize ocasionally
  - quicker response, but don't over do it.

Conclusion

- The Optimizer
  - Very good but does not always get it right.
  - It may need our help
  - Several tools to determine an optimal plan.
  - Hints can help it choose the better plan
  - We can improve performance if we know what to look for.

ERROR Messages

- SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
- SQL> select max(timestamp)

  from plan_table;

02-FEB-10

- SQL> select count(*) from plan_table;
- 120
- SQL> delete from plan_table;

Acknowledgements

- special thanks to members of Marketing DBA group for showing me their tuning tips and tricks.(Linda, Dave and Suresh) Their insight proved invaluable in learning more about the optimizer and the ways we can manipulate it.

References

- Oracle
  - 11g adminstrators guide
  - 10g Performance and Tuning Guide

More Fun

- Profiles / Outlines

Miscellaneous Ramblings

- Profiles / Outlines

THANK YOU !

- Contact information
  - John.X1.Larkin@chase.com

THANK YOU !

- Contact information
  - John.X1.Larkin@chase.com

THANK YOU !

- Contact information
  - John.X1.Larkin@chase.com