The Art of SQL Tuning

By Martin Farber
aka MrOracle, or simply MrO
Qualitec Incorporated
729 Horsepound Road
Kent Lakes, NY 10512

email: Martin.Farber@MrOracle.com
Cell#: 845-222-8176
URL: http://www.MrOracle.com

Who am I, and why am I here?

Pat Choate
Ross Perot's running mate in 1996

- Consulting for almost 30 years.
- Oracle DBA, designer, and developer for about 20 years - since Oracle Version 5.
- UNIX platforms for the last 10 years.
- Tuning, troubleshooting, and architecting for the last 5 years at Priceline.com.

Why an Art not a Science?

- Science implies knowledge and absolutes.
- Art implies skill, experience, observation, and creativity.
- Our media is an optimization plan, and the beauty we create is fast SQL that uses a minimum of resources.

The essence of a Relational Database is Autonavigation.

- Telling the database **what** you want, not **how** to get it.
- Most of the time, this is accomplished with acceptable speed.
- Performance tuning is telling the database **how** to get the data **faster** for those that don’t run at an acceptable speed.

Before SQL tuning - check for contention

- External contention
  - CPU
  - Memory
  - Disk
  - Network
- Internal contention
  - Locks
  - Latches
  - Shared servers

Before SQL tuning - check for viability

- Is the query even **correct**?
- Is the query even supposed to be **run** any more?
- Is there **a better way** to design the process?

Before SQL tuning - check for broader solutions

- Are there init.ora parameters that would address more than just this one query?
- Are there session level parameters that would address more than this one query within this process?
- Have the proper objects been analyzed - or not analyzed, as the case may be?

Optimizer_mode

- Rule
- First_Rows(_N)
- All_Rows
- Choose

Optimizer_mode=RULE

- Has been going away almost as long as our conversion to the metric system.
- Follows a static list of rules, not statistics
- Processes it's plan from the bottom of the FROM list to the top.
- Best features:
  - Consistent execution
  - No need to collect statistics
  - Minimal time spent picking a plan

Optimizer_mode=First_Rows(_N)

- Attempts to provide an execution plan that will return the first row(s) as quickly as possible.
- Tends towards using indexes and nested loops.
- If no statistics are available, on any or all of the tables, it will guess!

Optimizer_mode=All_Rows

- Attempts to provide an execution plan that will return the last row as quickly as possible.
- Tends towards using full table scans and hash or merge joins.
- If no statistics are available, on any or all of the tables, it will guess!

Optimizer_mode=Choose

- Also allegedly going away in 10g, but still under the covers.
- If a query has NONE of its tables analyzed, then it creates a Rule based plan.
- Otherwise, it creates an All_Rows, cost based plan. Even if it has to **guess** about some of the participating tables.

CBO Statistics Collection

- If you're going to run in a Cost mode, you need to collect statistics.
- Oracle's recommended way of collecting is the DBMS_STATS package.
- SYS and SYSTEM objects:
  - should NOT be analyzed prior to 9i
  - probably should not in 9i
  - will automatically be analyzed in 10g

How much analyzing is enough?

- Most people agree that if you can, compute.
- If tables or too large to compute stats in a reasonable window - estimate as much as possible.
- Estimating over 49%, essentially does a compute.
- One suggestion is to estimate the large tables, and compute stats on the indexes.

DBMS_STATS things to be aware of:

- Always back up your statistics - **before** a collection.
- Tables that might be **empty** during collection, but full when the stats will be needed.
- Be certain that you analyzed what you thought you analyzed.
- DBMS_STATS collects info that the optimizer needs, not chaining info, unused space info, etc. that ANALYZE collects.

How do you come by SQL to tune?

- You are writing a statement from scratch, and you want it to run well.
- A developer complains that some particular statement, or process, is too slow.
- Your system has come to a grinding halt!

Seeing your plan

- Autotrace
- Explain Plan
- GUI tools
- tkprof

Autotrace

- Runs in sqlplus, so it's usually available.
- Standard formatting can be unreadable. Try:
  - `set lines 100 wrap on trim on trimspool on`
  - `col plan_plus_exp format a100`
- Doesn't handle DDL. (Eg. Create as Select)
- Doesn't handle statements w/bind variables.
- Traceonly can still take a long time.

Explain Plan

- Always available, even without sqlplus.
- Runs a consistently fast plan.
- You can format the output to your own liking, or use DBMS_XPLAN.
- Handles DDL statements.
- Handles statements w/bind variables.

tkprof

- Timed_statistics should be set.
- Sql_trace or Event 10046 can be set for system, current session, or another session. (dbms_system.set_sql_trace_in_session or set_ev)
- Tkprof, and other tools, can format the data, or it can be read directly.
- Problematic if running Oracle’s MTS, or connection pooling on a middle tier.

Finding offensive statements

- Assuming you’ve eliminated outside causes, and contention - look for SQL that is either running too many times, or is doing too many disk_reads or buffer_gets.
- Look in V$SQLAREA for the statistics, but it only carries 1000 bytes of the SQL_TEXT, so the complete text will usually have to be retrieved from V$SQLTEXT_WITH_NEWLINES.

Sample output: Top10.sql

```
Executions:    6,403,877       Rows_processed:       12,627,846
Disk_reads:           18          Buffer_gets:       19,031,760
%All Reads:            7  Reads per Execution:                3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SELECT CLLI_CD   FROM DP_NPA_NXX_CLLI C  WHERE C.NPA = :b1  AND
C.NXX = :b2
======================================================================
Executions:   10,941,713       Rows_processed:       10,941,713
Disk_reads:           12          Buffer_gets:       12,945,971
%All Reads:            5  Reads per Execution:                1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
INSERT INTO CDR_20000210_S1522_D100 ( REL_MILLISEC,EXM_DATE_TIME
,EXM_MILLISEC,ACM_DATE_TIME,ACM_MILLISEC,RLC_DATE_TIME,RLC_MILLI
SEC,IAM_REL_DUR,IAM_REL_CCS,ANM_REL_DUR,ANM_REL_CCS,CALLING_NATR
_ADDR_CD,CALLING_NATR_ADDR_IND,CALLING_EVEN_ODD_FLG,CALLED_NATR_
ADDR_CD,CALLED_NATR_ADDR_IND,CALLED_EVEN_ODD_FLG,CHARGE_NATR_ADD
R_CD,CHARGE_NATR_ADDR_IND,CHARGE_EVEN_ODD_FLG,ORIG_LINE_CD,CARRI
ER_ID_CD,CARRIER_SELECT_CD,TCIC,JURISDICTION,BACKWD_CHARGE_CD,BA
CKWD_CALLED_STAT_CD,BACKWD_CALLED_CAT_CD,BACKWD_END_TO_END_CD,BA
```

Tuning cost based queries

- From **outside** the query:
  - Set a session level optimizer_mode. (Does not effect PL/SQL.
  - Set optimizer_index_cost_adj to a lower or higher number depending on your desire to increase or decrease the affinity for indices.
  - Manually adjust the statistics stored in the data dictionary.
  - Build missing indices.
  - Force CURSOR_SHARING - if possible.
  - Create hinted views on existing tables.

Tuning cost based queries

- From inside the query:
  - Explicitly declare the optimization mode in a hint. (eg FIRST_ROWS)
  - Use a LEADING hint, or arrange the FROM clause and use an ORDERED hint.
  - Specify the join method in a hint. (eg USE_NL)
  - Use a DRIVING_SITE hint if a database link should be driving the query.
  - Appropriate use of bind variables.