# **Information Needed in Tuning a SQL Query**

The point of this document is NOT to teach you how to tune SQL; it is to help you collect the information you need in order to tune or have someone else help you tune SQL.

In order to tune a SQL Query, specific pieces of information are needed. Some of this is obvious yet all of us can point to hundreds of posts on the WWW where someone has asked for help tuning a SQL Statement and then for example, they did not actually post the statement. How does one tune a query when one has never seen it?

So to improve the overall state of affairs in the SQL Tuning space, here are the details of what information is needed to tune a problem query. I will provide SQL Scripts that gather most of this information needed to tune a SQL Query, so as to make it easier for you to provide this information in your discussions with others.

If you are interested in improving your own SQL Tuning skills, then I refer you to my book: Oracle SQL Performance Tuning and Optimization: It’s all about the Cardinalities. This can be found on Amazon (Ctrl+Click to follow the link).

<http://www.amazon.com/Oracle-Performance-Tuning-Optimization-Cardinalities/dp/1501022695>

This book like most books, being relevant to your career development, is likely covered under your company’s reimbursement policy. So if you want the book, ask your Boss if you can get it and expense it. After your boss gives you verbal approval, buy it from Amazon and submit an expense report to get your money back. There is nothing wrong with letting your company invest a few more dollars in your professional growth, especially since you will be spending your personal time reading this book to improve skills which they will be quick to exploit.

In addition, the first chapter of the book is FREE, and the scripts from the book are FREE too. You do not need to buy the book to get these. You can get them from this link below. Reading the first chapter before you buy the book and using the scripts a little, are both good ways to satisfy yourself and your Boss that the investment is worth it. Go here to OraFAQ to download them.

<http://www.orafaq.com/forum/m/634324/#msg_634324>

So, if you are planning on doing a tuning session or asking someone else to do a tuning session with you or for you, then you will need at some point to gather together the following bits of information.

# **Four categories of information we will collect related to SQL Tuning:**

* Basic Diagnosis Information
* Advanced Diagnosis Information
* Active Diagnosis Information (we will ignore this data in this teaching document)
* Documentation Related Information

First there is the BASIC information that is needed to understand the problem space. This basic information gives us an appreciation of the performance issue, and what some potential problems might be. For example, this tell us the join order of tables and lets us judge if this order is a good one or not. Then once an understanding of the query is had, additional information may be needed in order to suggest possible fixes. For example, maybe cardinalities in the plan suggest indexes would help. If so, then we would need first to check the list of indexes that already exist on the plan’s tables. Next, for very difficult problems, it may be necessary to monitor specific aspects of an actively running query or consult AWR data for the same data only historical. For example, we might want to check the efficiency of hash joins to see if they are OPTIMAL or ONE-PASS or if they are the potentially problematic MULTI-PASS. So we need to run the query and monitor its temp space usage, or check AWR for historical usage data. Last, we may want additional documentary types of information to make explaining things easier to others. For example, a data model and query diagram are pictures which provide good artifacts for you to talk to when explaining how you solved the problem.

# **Basic Diagnosis Information:**

* The SQL Statement
* A Business Description of the query’s purpose (what does the business think the query does?)
* The Query Execution Plan (must include PLAN\_OUTPUT and PREDICATES sections)
* Cardinality Information (table row counts, filtered row counts, estimated and actuals)
* Current Runtime (how long does it run in seconds/minutes/hours)
* Expected Runtime (how fast do you want it to run) (as fast as possible is not a good answer!)
* Basic Environment (EXADATA or not?)

With this information, most Oracle Professionals can engage in a meaningful discourse over; what problems might exist, what possible strategies and solutions might be appropriate, and ultimately -- how a query can be made faster.

SELECT COUNT(\*) FROM VW\_EMP\_LOC\_DIM;

DESCRIPTION: Shows basic employment status of every employee at all active locations.

Plan hash value: 1232802935

-----------------------------------------------------------------------

| Id | Operation | Name | Rows |

-----------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 1 |

| 1 | SORT AGGREGATE | | 1 |

| 2 | VIEW | VW\_EMP\_LOC\_DIM | 198M|

| 3 | UNION-ALL | | |

|\* 4 | HASH JOIN | | 296K|

|\* 5 | VIEW | | 240K|

|\* 6 | WINDOW SORT PUSHED RANK | | 240K|

|\* 7 | FILTER | | |

|\* 8 | TABLE ACCESS STORAGE FULL | EMP\_DIM | 240K|

|\* 9 | TABLE ACCESS STORAGE FULL | EMP\_LOC\_DIM | 296K|

| 10 | COUNT | | |

|\* 11 | HASH JOIN | | 152M|

| 12 | VIEW | | 8874 |

| 13 | HASH UNIQUE | | 8874 |

|\* 14 | FILTER | | |

|\* 15 | TABLE ACCESS STORAGE FULL| EMPLR\_LOC\_DIM | 8874 |

| 16 | VIEW | | 240K|

| 17 | MINUS | | |

| 18 | SORT UNIQUE | | 240K|

|\* 19 | TABLE ACCESS STORAGE FULL| EMP\_DIM | 240K|

| 20 | SORT UNIQUE | | 251K|

|\* 21 | TABLE ACCESS STORAGE FULL| EMP\_LOC\_DIM | 251K|

-----------------------------------------------------------------------

Predicate Information (identified by operation id):

---------------------------------------------------

 4 - access("EMP\_GID"="EL"."EMP\_LOC\_GID")

 5 - filter("PRTY\_REF\_ID\_RNK"=1)

 6 - filter(RANK() OVER ( PARTITION BY "GRP\_BEN\_CASE\_ID","PRTY\_REF\_ID" ORDER BY CASE

 "EE"."EMPLMT\_STAT\_CD" WHEN 'T' THEN 2 WHEN 'R' THEN 2 WHEN 'D' THEN 2 WHEN 'I' THEN 2 ELSE 1 END

 ,INTERNAL\_FUNCTION("EE"."SRCE\_EFF\_START\_TMSP") DESC ,INTERNAL\_FUNCTION("EE"."EMP\_PK\_ID") DESC )<=1)

 7 - filter(SYS\_CONTEXT('APP1','CURRENT\_SCHEMA')='CLDW\_THPA\_DLV1')

 8 - storage("EE"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EE"."SRCE\_APP\_SYS\_CD"='ELIG')

 filter("EE"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EE"."SRCE\_APP\_SYS\_CD"='ELIG')

 9 - storage("EL"."POPULATION\_STATUS\_CD"<>'D')

 filter("EL"."POPULATION\_STATUS\_CD"<>'D')

 11 - access("A"."GRP\_BEN\_CASE\_ID"="B"."GRP\_BEN\_CASE\_ID")

 14 - filter(SYS\_CONTEXT('APP1','CURRENT\_SCHEMA')='CLDW\_THPA\_DLV1')

 15 - storage("E"."EMPLR\_LOC\_PK\_ID"<>(-1) AND "E"."EMPLR\_LOC\_PK\_ID"<>(-2) AND

 "E"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00')

 filter("E"."EMPLR\_LOC\_PK\_ID"<>(-1) AND "E"."EMPLR\_LOC\_PK\_ID"<>(-2) AND

 "E"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00')

 19 - storage("EE"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EE"."POPULATION\_STATUS\_CD"<>'D' AND "EE"."EMP\_PK\_ID"<>(-1) AND "EE"."EMP\_PK\_ID"<>(-2))

 filter("EE"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EE"."POPULATION\_STATUS\_CD"<>'D' AND "EE"."EMP\_PK\_ID"<>(-1) AND "EE"."EMP\_PK\_ID"<>(-2))

 21 - storage("EL"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EL"."POPULATION\_STATUS\_CD"<>'D')

 filter("EL"."SRCE\_EFF\_END\_TMSP"=TIMESTAMP' 9999-12-31 00:00:00' AND

 "EL"."POPULATION\_STATUS\_CD"<>'D')

Note

-----

 - dynamic sampling used for this statement (level=4)

 - automatic DOP: Computed Degree of Parallelism is 1

@genfrpspreadsheetcode411g.sql

 Plan Filtered Actual

ID TABLE\_NAME NUM\_ROWS ROWCOUNT Cardinality Cardinality FRP

----- ------------- ---------- ---------- ----------- ----------- ------

 8 EMP\_DIM 6243035 6243035 240117 215414 3.5

 9 EMP\_LOC\_DIM 329699 329699 296337 329699 100.0

 15 EMPLR\_LOC\_DIM 8874 8874 8874 8872 100.0

 19 EMP\_DIM 6243035 6243035 240117 236469 3.8

 21 EMP\_LOC\_DIM 329699 329699 251761 212993 64.6

5 rows selected.

CURRENT RUNTIME: 3 minutes 15 seconds.

EXPECTED RUNTIME: 3 seconds (based on volume of data)

EXADATA: Yes (note “storage” keyword in the query plan)

Given the above information, people can see the important parts of your problem and start to answer significant questions like:

* What is the query Driving Table and is it reasonable?
* What is the query Join Order and is it reasonable?
* What is the Data Volume being fed into the query?
* Does the Current Runtime align with the Number of Rows being manipulated?
* Does the Expected Runtime seem realistic?
* Where is Row Filtering occurring and how much is occurring?
* Where might Plan Cardinalities be wrong?
* What is the basic Query Style (precision or warehouse)?
* Do tables have appropriate Access Methods used against them?
* Are joins being done with appropriate Join Methods?
* Where are there possible Filter Inefficiencies in the query plan?
* Where are there possible Join Inefficiencies in the query plan?
* Does Query Logic match Business Semantics?

# Scripts I will give you for collecting this information include:

* SHOWPLAN11G.sql
* SHOWTPLAN11GSHORT.sql
* SHOWPLANFRPSPREADSHEETCODE11G.sql
* LOADPLANFROMCACHE11G.sql
* LOADPLANFROMHIST1G.sql

# **Advanced Diagnosis Information:**

* Indexes on the tables referenced by the query
* Constraints on the tables referenced by the query
* Column Statistics for specific tables and columns
* Histograms for specific tables and columns
* Various System Parameter Settings
* If EXADATA then statistics showing SMARTSCAN effectiveness

After reviewing this basic information, additional information might be relevant. This could include any of the above, which will be used to consider alternative ideas that will allow you to change something to make your query go faster.

@showplantables11g

 ID OBJECT\_OWNER OBJECT\_NAME OBJECT\_ALIAS

---------- ------------------------------ ------------------------------ --------------------

 18 SCOTT EMPLR\_LOC\_DIM E @ SEL$8

 6 SCOTT EMP\_DIM EE @ SEL$4

 13 SCOTT EMP\_DIM EE @ SEL$6

 7 SCOTT EMP\_LOC\_DIM EL @ SEL$2

 15 SCOTT EMP\_LOC\_DIM EL @ SEL$7

5 rows selected.

@showplantablesunique11g

OBJECT\_OWNER OBJECT\_NAME

------------------------------ ------------------------------

SCOTT EMPLR\_LOC\_DIM

SCOTT EMP\_DIM

SCOTT EMP\_LOC\_DIM

3 rows selected.

@showplanindexesunique11g

TABLE\_OWNER TABLE\_NAME INDEX\_NAME INDEX\_OWNER

-------------- ------------------------------ ------------------------------ --------------

SCOTT EMPLR\_LOC\_DIM PK\_EMPLR\_LOC SCOTT

SCOTT EMP\_DIM PK\_EMP SCOTT

SCOTT EMP\_LOC\_DIM PK\_EMP\_LOC SCOTT

3 rows selected.

@showplanindexes11g

TABLE\_OWNER INDEX\_OWNER INDEX\_NAME COLUMN\_NAME INDEX\_TYPE UNIQUENES TABLESPACE\_NAME

------------ ------------ ------------- ---------------- ------------ --------- -----------------

SCOTT SCOTT PK\_EMPLR\_LOC EMPLR\_LOC\_PK\_ID NORMAL UNIQUE CLDW\_DATA\_01

SCOTT SCOTT PK\_EMP EMP\_PK\_ID NORMAL UNIQUE CLDW\_DATA\_01

SCOTT SCOTT PK\_EMP\_LOC EMP\_LOC\_PK\_ID NORMAL UNIQUE CLDW\_DATA\_01

3 rows selected.

Elapsed: 00:00:00.14

@showplanconstraints11g

OWNER TABLE\_NAME CONSTRAINT\_NAME C COLUMN\_NAME PCHILD\_OWNER PCHILD\_TABLE\_NAME P INDEX\_NAME

----- -------------- --------------- - --------------- ------------ ----------------- - ---------------

SCOTT EMPLR\_LOC\_DIM PK\_EMPLR\_LOC P EMPLR\_LOC\_PK\_ID PK\_EMPLR\_LOC

SCOTT EMP\_DIM PK\_EMP P EMP\_PK\_ID PK\_EMP

2 rows selected.

I apologize for the line wraps here. Just not much to do about it if the artifact is going to show enough for you to understand it. The script prints better in SQL\*Plus.

@showcolstats SCOTT EMPLR\_LOC\_DIM

COLUMN\_NAME NUM\_DISTINCT NUM\_NULLS NUM\_BUCKETS SAMPLE\_SIZE AVG\_COL\_LEN DENSITY LAST\_ANALYZED GLO USE LOW\_VALUE HIGH\_VALUE

------------------------------ ------------ ---------- ----------- ----------- ----------- ---------- ---------------------- --- --- --------------- ---------------

BATCH\_ID 2 0 1 7620 5 .5 20-jun-2013.19:21:05 YES NO "56473" "56779"

BUS\_EFF\_END\_TMSP 1 0 1 7620 11 1 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

BUS\_EFF\_START\_TMSP 29 0 1 7620 11 .034482759 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

DW\_LST\_UPD\_TMSP 2 0 1 7620 11 .5 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

DW\_POPULATION\_TMSP 2 0 1 7620 11 .5 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

EMPLR\_LOC\_GID 7620 0 1 7620 6 .000131234 20-jun-2013.19:21:05 YES NO "4309040" "4407795"

EMPLR\_LOC\_HRCHY\_LVL1\_DESC 56 0 56 7620 4 .000065617 20-jun-2013.19:21:05 YES NO "AA" "WY"

EMPLR\_LOC\_HRCHY\_LVL1\_NM 2 0 2 7620 26 .000065617 20-jun-2013.19:21:05 YES NO "Employee Work "Employee Work

EMPLR\_LOC\_HRCHY\_LVL2\_DESC 599 0 254 7620 28 .00199495 20-jun-2013.19:21:05 YES NO "-ATLANTA (GEE2 "WAUSA"

EMPLR\_LOC\_HRCHY\_LVL2\_NM 5 0 5 7620 15 .000065617 20-jun-2013.19:21:05 YES NO "EMPLOYEE DEPAR "WORK LOCATION

EMPLR\_LOC\_HRCHY\_LVL3\_DESC 24 0 24 7620 5 .000065617 20-jun-2013.19:21:05 YES NO "96800" "WILSONVILLE"

EMPLR\_LOC\_HRCHY\_LVL3\_NM 5 0 5 7620 32 .000065617 20-jun-2013.19:21:05 YES NO "EMPLOYEE COST "WORK CITY"

EMPLR\_LOC\_HRCHY\_LVL4\_DESC 7407 0 254 7620 17 .000144844 20-jun-2013.19:21:05 YES NO "4izzzbnbhgf, L "Zyzzzghjrtx, F

EMPLR\_LOC\_HRCHY\_LVL4\_NM 4 0 4 7620 21 .000065617 20-jun-2013.19:21:05 YES NO "EMPLOYEE DEPAR "WORK LOCATION

EMPLR\_LOC\_HRCHY\_LVL5\_DESC 107 0 107 7620 5 .000065617 20-jun-2013.19:21:05 YES NO "10100" "TESTING"

EMPLR\_LOC\_HRCHY\_LVL5\_NM 4 0 4 7620 33 .000065617 20-jun-2013.19:21:05 YES NO "EMPLOYEE COST "N/A"

EMPLR\_LOC\_PK\_ID 7620 0 1 7620 10 .000131234 20-jun-2013.19:21:05 YES NO "56473000000001 "56779000001849

GRP\_BEN\_CASE\_ID 4 0 4 7620 7 .000065617 20-jun-2013.19:21:05 YES NO "620920" "902236"

POPULATION\_STATUS\_CD 1 0 1 7620 2 1 20-jun-2013.19:21:05 YES NO "I" "I"

SRCE\_EFF\_END\_TMSP 1 0 1 7620 11 .000065617 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

SRCE\_EFF\_START\_TMSP 29 0 1 7620 11 .034482759 20-jun-2013.19:21:05 YES NO "UNSUPPORTED DA "UNSUPPORTED DA

SRCE\_SYS\_CD 1 0 1 7620 5 .000065617 20-jun-2013.19:21:05 YES NO "ITMS" "ITMS"

22 rows selected.

I had to use a different schema here since the objects from the view we are using as our example here, did not have any histograms.

@showhistogram dbsnmp bsln\_baselines instance\_name

OWNER TABLE\_NAME COLUMN\_NAME ENDPOINT\_NUMBER ENDPOINT\_VALUE CARDINALITY PCT ACTUAL\_VALUE

------- -------------------- -------------------- --------------- -------------- ----------- ---------- ------------

DBSNMP BSLN\_BASELINES INSTANCE\_NAME 3 5.2646E+35 3 38 edadev1

DBSNMP BSLN\_BASELINES INSTANCE\_NAME 6 5.2646E+35 3 38 edadev2

DBSNMP BSLN\_BASELINES INSTANCE\_NAME 7 5.2646E+35 1 13 edadvs2

DBSNMP BSLN\_BASELINES INSTANCE\_NAME 8 5.2646E+35 1 13 edadvs3

4 rows selected.

Elapsed: 00:00:00.13

16:47:57 SQL> @showparameters

KEYWORD NAME VALUE ISDEFAULT

---------- ---------------------------------------- ------------------------------ ---------

cpu cpu\_count 6 TRUE

cpu parallel\_threads\_per\_cpu 2 TRUE

cpu resource\_manager\_cpu\_allocation 6 TRUE

dyn optimizer\_dynamic\_sampling 2 TRUE

index optimizer\_index\_caching 0 TRUE

index optimizer\_index\_cost\_adj 100 TRUE

index optimizer\_use\_invisible\_indexes FALSE TRUE

index skip\_unusable\_indexes TRUE TRUE

mode optimizer\_mode ALL\_ROWS TRUE

mode remote\_dependencies\_mode TIMESTAMP TRUE

mode result\_cache\_mode MANUAL TRUE

optimizer optimizer\_capture\_sql\_plan\_baselines FALSE TRUE

optimizer optimizer\_features\_enable 11.2.0.1 FALSE

optimizer optimizer\_secure\_view\_merging TRUE TRUE

optimizer optimizer\_use\_pending\_statistics FALSE TRUE

optimizer optimizer\_use\_sql\_plan\_baselines TRUE TRUE

parallel parallel\_adaptive\_multi\_user TRUE TRUE

parallel parallel\_automatic\_tuning FALSE TRUE

parallel parallel\_degree\_limit CPU TRUE

parallel parallel\_degree\_policy MANUAL TRUE

parallel parallel\_force\_local FALSE TRUE

parallel parallel\_instance\_group TRUE

parallel parallel\_io\_cap\_enabled FALSE TRUE

parallel parallel\_max\_servers 60 TRUE

parallel parallel\_min\_percent 0 TRUE

parallel parallel\_min\_servers 0 TRUE

parallel parallel\_min\_time\_threshold AUTO TRUE

parallel parallel\_server FALSE TRUE

parallel parallel\_server\_instances 1 TRUE

parallel parallel\_servers\_target 24 TRUE

pga pga\_aggregate\_target 0 TRUE

rewrite query\_rewrite\_enabled TRUE TRUE

rewrite query\_rewrite\_integrity enforced TRUE

size bitmap\_merge\_area\_size 1048576 TRUE

size create\_bitmap\_area\_size 8388608 TRUE

size hash\_area\_size 131072 TRUE

size parallel\_execution\_message\_size 16384 TRUE

size sort\_area\_retained\_size 0 TRUE

size sort\_area\_size 65536 TRUE

size workarea\_size\_policy AUTO TRUE

statistics statistics\_level TYPICAL TRUE

statistics timed\_os\_statistics 0 TRUE

statistics timed\_statistics TRUE FALSE

43 rows selected.

With the above information, SQL Tuners can make comparisons between what the plan is using, and what the plan could be using, and what they think might be missing that the plan could have used. They can answer questions like these.

* Are we working in the correct schemas with the correct objects?
* What alternative indexes could have been used?
* Do we think indexes are missing that might provide more efficient access paths for precision queries?
* Are indexes used taking advantage of all relevant columns in query predicates?
* Are constraints missing that could have supplied additional information to the optimizer?
* Do column statistics suggest an OUT-OF-BOUNDS problem?
* Do statistics look old and possibly need re-collection?
* If data is SKEWED, are the necessary Histograms in place to address this?
* Could EXTENDED STATISTICS help?
* Could better DYNAMIC\_SAMPLING help?
* Do database parameters in general reflect a good environment?

# Scripts I can give you for collecting this information include:

* SHOWPLANTABLES11G.sql
* SHOWPLANTABLESUNIQUE11G.sql
* SHOWPLANINDEXESUNIQUE11G.sql
* SHOWPLANINDXES11G.sql
* SHOWPLANCONSTRAINTS11G.sql
* SHOWCOLSTATS.sql
* SHOWHISTOGRAMS.sql
* SHOWPARAMETERS.sql

# **Active Diagnosis Information**

This information is for more advanced problems. Typically it requires running a query to see the resources it consumes, and possibly looking at historical executions of the query to see what has happened before. There are uses for this, but they involve deeper problems and would require many more pages to describe. So for the time being I am skipping this one since 90% of what we can solve can be handled by the BASIC and ADVANCED information we have already collected. At some later date I will come back and fill this in as time permits.

# **Documentation Related Information**

Query Diagram

Data Model for the Query

When documenting your results, or when providing context for others, these pictures provide good references. The help people visualize where your query sits with respect to the entire database. Perspective is usually a good thing.

# Scripts I can give you for collecting this information include:

For this I currently have no scripts to offer. You will need to use other tools to create your pictures. I would suggest TOAD is one possibility. Though it is a manual process, you can drag-and-drop specific tables into a model window and have Toad hook them all up into a picture. Since other scripts you have used here show you the tables involved, you have the necessary information to create a data model diagram for the tables in your query.

So as you can see, it is not difficult at all to acquire the information needed for a tuning session, when you use the scripts. This information will help you and those who assist you, to tune you problem SQL. Please refer back to page 1 above for the location of the FREE scripts.

Good luck, and remember to provide this information whenever you are asking for help.

Kevin Meade