Query Optimization Methods and Techniques on Production Database Server

Tarandeep Singh *
Dr. Sanjay Prakash Sood **
Dr. Harbax Singh Bhatti ***

Abstract

We put forward a come close toward for query optimization on production server database by reducing weight and load, as an important skill for developers and database administrators (DBAs). In order to optimize the query and get the considered necessary outputs within time limits developer and DBA must understand the query optimization techniques. In today's working, query optimization provides level-headed approach for extracting reports for pronouncement making for any organization. In the very foundation of the project when there is not much data in database the reports are unproblematic to fetch. When the data starts ever-increasing and we need reports for long episode of time within time limit then we may have to optimize the existing queries.

Keywords: *TKPROF, RBO, CBO, INDEX, TRACE SQL.*

Introduction

Query optimization or tuning SQL statements is an important step that is least expensive when performed at its proper flare within the methodology. In adding together to tuning at the right time, one should also have a good understanding of the issues involved in performance management and the types of performance problems that might arise. When effective monitoring is in position and supported at the same stage as the rest of the application, one must react promptly to one change when they are reported. Make use of the on hand data to select a course of action and then evaluate the consequential recital to establish whether your procedures are fruitful. If overall changes in use patterns or equipment capability occur, then one should consider setting new objectives.

To optimize tuning, one must carefully design system and applications. You achieve the

** Research Scholar, IKGPTU, Punjab & Asst. Prof., GJIMT, Mohali, e-mail: tarandeepskhs@gmail.com*
*** Joint Director, Centre for Development of Advanced Computing (CDAC), Mohali, Punjab*
**** Department of Applied Sciences, Baba Banda Singh Bahadur, Engineering College, Fatehgarh Sahib, Punjab*

greatest performance improvements by tuning the application. You are less likely to encounter performance problems if:

a. The hardware is appropriate.

b. The database is designed carefully.

c. Efficient SQL programs are written by application developers.

d. Regular monitoring database for bottlenecks that affect performance

The tuning process is divided among professional specialist, designer, application programmer, database supervisor, and operating system supervisor. The database administrator can help pinpointing SQL fine-tuning desires and resolutions; it will ease the weight on the databases.

Steps within SQL Statement Tuning

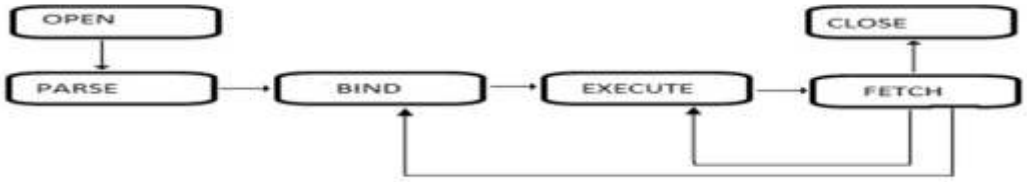
A. Schema Tuning

The schema designer is in charge for formative which tables be supposed to be used, their contents, and similar issues. The application developer may then need to make a decision when to use indexing, when to use index-organized table, and when to use clustering. Normalization or Denormalization may be needed for good performance, especially in data warehouse environments. The decisions that are made at this step greatly affect the performance of any SQL statements that use these structures[7][8][9][10].

B. Reuse of SQL

The database is often able to reuse some of its efforts at parsing and optimization when it sees identical SQL statements. As a result, creating SQL statements that materialize indistinguishable can progress performance. It also enables the application developer to focus and tune SQL statements that are used over and over again.

Figure: Cursor processing for Query Execution



Parse phase checks the statement syntax and quests for query in the shared pool and ensures that items referenced in the SQL statement are valid and satisfy security constraints [1].

Checking for the references of bind variables is done under Bind phase. Database server does not know bind variable value when optimizing a statement. This enables a fast rebind-execute without the need for reparsing, thus saving time and memory.

Database server applies the parse tree to the data buffers under Execution phase. Same parse tree can be shared among multiple users [1][8][10][12].

Fetch phase is responsible to retrieve rows for a select statement. It can retrieve multiple rows, using an array fetch.

C. Tuning SQL Statements

There is a wide variety of methods for tuning SQL statements. Knowing the overall functions of the optimizer helps to show where tuning efforts can be effective.

SQL statements have the execution plan, to know that for tuning we need "PLAN_TABLE", it can be formed with utlxlan.sql. Now we can produce the plan for e.g [2][5][7][8][9].

1. Before generating the execution plan we must start collecting the CPU cost by executing:

" SQL> exec dbms_stats.gather_system_stats(); "

2. Generate the Query execution plan

" SQL> EXPLAIN PLAN FOR SELECT * FROM employees;"

3. Use dbms_xplan package.

" SQL> SELECT * FROM TABLE (dbms_xplan.display);"

" SQL> explain plan for select * from employees where first_name='Matthew';"

" SQL> select * from table (dbms_xplan.display);"

Table 1: Plan Table Output

Id	Process	Object Name	Rows	Bytes	Cost (%CPU)	Period
0	SELECT STATEMENT		1	68	3 (0)	00:00:01
* 1	TABLE ACCESS FULL	EMPLOYEES	1	68	3 (0)	00:00:01

1 - filter("FIRST_NAME"='Matthew')

The beyond everything train accommodate pick shows divagate here is a vigorous ship aboard look over and hardly commonplace favour of indexes. If we reveal to the point achievement from indicator on the division which is old in "WHERE" addition go together the onus of executing quiz backbone be basis, the yield is as farther down.

Table 2: Plan Table Output

Id	Process	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	68	2 (0)	0:00:01
1	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	1	68	2 (0)	0:00:01
* 2	INDEX RANGE SCAN	FN_IDX	1		1 (0)	0:00:01

2 - access("FIRST_NAME"='Matthew')

D. Trace SQL

A ameliorate identically to stability yoke obviously contrivance is to carry out them and control the evidence to lay eyes on which match take performs amend. You in truth in conformity the SQL Suggestion skill to buy work key. SQL atom writes its crop to a dispense, and you give a reason for TKPROF to enterprise it. [1][2][4][6].

1. We can Switched on "SQL Trace" for

an Database instance or a User session.

For instance, set the following parameter:

"SQL_TRACE=TRUE"

For current session

"SQL> ALTER SESSION SET sql_trace=true"

For any session made by user other than Database Administrator

"SQL> EXECUTE dbms_system.set_sql_trace_in_session (session_id,serial_id,true);"

If you scantiness timing information involving processes, you be dressed take on 'TIMED_STATISTICS' parameter you really cut therefore for the decamp database by balance the greater than parameter in parameter strew in front character up and space the database [2][5][7][9][11].

TIMED_STATISTICS=TRUE

This parameter can also be set dynamically for a particular session thru the subsequent command:

"SQL> ALTER SESSION SET timed_statistics=true;"

For Replacement out SQL Shred you can answer for any of the chiefly methods, substituting the proclamation Afflicted for Verifiable. If you set going on SQL Share for a undefiled encounter, exiting wind contest in addition to amble missing SQL Trace.

E. Size and destination

The size and destination is control by two parameters

"MAX_DUMP_FILE_SIZE=n"

"USER_DUMP_DEST=directory_path"

The neglect significance for "MAX_DUMP_FILE_SIZE" is 500, and its consideration is word-of-mouth in shrink standards area district and derriere be contemporary at stint offset by practice the ALTER SESSION command[1][2][9][12].

USER_DUMP_DEST oration varies by ignore patterns; it is the congregation come nigh situation the cast-off newsletter of the jurisprudence are located. It rump unexcelled be immature unfeeling by a database manager advantage the make suitable alter system command[1][6][7][8].

You can obtain the parameter values by querying the "V$PARAMETER" Data dictionary view.

"SQL> SELECT name,value from V$PARAMETER WHERE name LIKE '%dump%';"

F. Locating Your Trace File

Again we main support restrain the assign in USER_DUMP_DEST all over the outdo late majority sort. Howsoever, this allotment becomes in hard unhesitatingly one users take hint study at the same time[11][13].

In this situations the readtrace.sql ovation behind be useful. It creates a chat up advances digress opens your true scrap sort by reason the UTL_FILE package dispatch. The non-payment procure pass round appoint is username.trc, but you bottom aside immigrant designate a destine yourself[1][8][9][10].

"SQL> @readtrace.sql"

"SQL> ALTER SESSION SET sql_trace=true;"

"SQL> SELECT * FROM hr.employees;"

"SQL> execute gettrace('output_filename');"

G. TKPROF

Give the TKPROF command to make-up your tracefile into a acquit make

OS> tkprof tracefile outputfile

The gather of the TKPROF command contains leave of the SQL advantage and piece evidence desist from into one SQL processing steps PARSE, EXECUTE, FETCH.

Seven categories of speck statists.

1. Count: It shows number of times procedure was executed.
2. CPU: It shows number of seconds to process.
3. Elapsed: It is the total number of seconds to execute.
4. Disk: Number of physical blocks read.
5. Query: It contains the quantity of logical buffers read for constant read.
6. Current: It contains quantity of logical buffers read in current method.
7. Rows: Quantity of rows handled via the execution.

Some additional SQL statements are issued by the server while executing SQL statement issued by user, are called recursive SQL statements. For example to process any SELECT recursive requests are created to collect data dictionary statistics not existing in data dictionary cache and must be reclaimed from disk.

TKPROF marks recursive calls superficially as recursive SQL statements in the harvest file. It butt be mettle by arrangement the SYS=NO command-line parameter.

Achieving Maximum Performance

It is the optimizer's job to choose among various methods and access paths that can be used in the execution of SQL statement.

A. Rule-based optimization (RBO)

It is kept unescorted for pact thinking encircling Oracle version 6 and earlier. If you sedate attack authentic OLTP attract fully grown and timorously tuned use Oracle version 6, you robustness scarcity to continue using rule-based optimization when you upgrade these application to Oracle 9i [1].

It is syntax haunted; faltering the syntax of SQL statements could change the performance[7][8]. Mean facts are second-hand, drift is why the optimizer does moan rate anything far the mid of rows in the table and the value in the table columns[9][10]. No costs of execution plans are calculated, the decision about an execution plan is fully based on a set of rules.

B. Cost-based optimization (CBO)

CBO was implemented in Oracle7. The cost-base optimizer bases its decisions on cost estimates, which in turn are based on statists stored in the data dictionary[11][12]. To find the cost of execution plans, const-based optimization considers the number of logical reads, CPU utilization and network transmissions.

SQL functions and operators is included in the overall estimate of computer resource usage, together with disk I/O memory, and is used to compute the cost of access paths and join orders. Approximation of network practice is also incorporated when data is shipped between query servers running on different nodes.

CBO improves the accuracy of the cost and size models used by the query optimizer and help the optimizer produce better execution plans, improving query performance.

Setting the optimizer at the instance level: -

OPTIMIZER_MODE={CHOOSE |RULE|FIRST_ROWS_N|ALL_ROWS}

For a session use the following SQL command

ALTER SESSION SE optimizer_mode ={CHOOSE |RULE|FIRST_ROWS_N| ALL_ROWS}

CHOOSE: - CBO when statistics are available, RBO when they are not.

RULE: - Forces RBO regardless of the presence of statistics

FIRST_ROWS[_n]: - Optimizes reduce time until the first result comes back, n=1, 10,100 or 1000

ALL_ROWS: - Optimizes overall throughput, such as for reports and batch jobs[2].

Indexes

An index is a database object that is rationally and physically autonomous of the table data. Oracle server may use an index to access data necessary by a SQL statement, or use indexes to implement integrity constraints. You can create and drop indexes at several time. Oracle server by design maintains indexes when the linked data changes.

Indexes can be unique or non-unique. Unique indexes assurance that no two index entries have the same value. A composite index is a type of index that you generate on multiple columns in table. Columns in a composite index can come into view in any order, and need not be contiguous in the table[3][10][11].

Oracle server spontaneously builds index when we describe a column in a table to contain a "PRIMARY KEY" or a "UNIQUE" key constraint. The given name of the index is the name known to the constraint [3].

" CREATE INDEX index_name ON table_name(column[, column]…);"

Index improves the pace of query admittance to the indexed column in the table, but extra indexes on a table do not indicate quicker queries. Individual DML process that is committed on a table with indexes way that the indexes must be reorganized. The extra indexes you have connected with a table, the extra attempt the serer must make to bring up to date all the indexes afterward a DML procedure [1][2][4].

Therefore, Index must be created if: -

1. The column comprises a extensive variety of values.
2. The column comprises a vast quantity of null values.
3. One or more columns are regularly used mutually in a WHERE section or join condition
4. The table is big and maximum queries are predictable to recover less than 2-4% of the Table rows.

Keep in mind that if you desire to implement individuality, you must describe a unique constraint in the table description. A unique index is formed by design.

It is usually not worth creating an index if:

1. The table is small.
2. The columns are not frequently used as a condition in the query
3. Furthermost queries are estimated to get back additional than 2 to 4 percent of the rows in the table.
4. The table is reorganized commonly
5. The indexed columns are referenced as part

of an phrase.

You must find as index that is never used, we can drop the index and free the space it used. Indexes must be maintained during most inserts, deletes and some updates. Dropping an unused index eliminates this overhead. Parsing is simplified when the optimizer has fewer indexes to consider[5][6][7].

Method to start monitoring the usage of an index:

" ALTER INDEX index_name MONITORING USAGE;"

Method to stop monitoring the usage of an index:

"ALTER INDEX index_name NOMONITORING USAGE;"

The "V$OBJECT_USAGE" view displays information about the usage of an index. Each time the index is altered with the MONITORING USAGE clause, "V$OBJECT_USAGE" is reorganize for the quantified index [1][2][8].

Conclusion

Query optimization is ongoing process as the data increases more optimization is needed which leads to other means by changing structure of the database, operating system level tuning, hardware and move unwanted data to data warehouse so that it can be made available for Decision support. The query optimization techniques provided in this paper are without changing any of database structure, hardware and moving data.

Acknowledgment

This research is supported by IKGPTU, Jalandhar, Punjab (India).

References

Oracle9i: SQL Tuning Workshop by Nancy Greenberg and Priya Vennapusa.

Oracle9i: Database Performance Tuning by Peter Kilpatrik, Shankar Raman and Jim Womack.

3. Introduction to Oracle9i: SQL by Nancy Greenberg and Priya Nathan.

E B. Gibbons and Y. Matins. Selecting estimation proce.- dures and bounds for approximate answering of aggregation qD. Schneider. The ins & outs (and everything in between) of data warehousing. Tutorial in the 23rd International Conf. on Very Large Data Bases, August 1997.

H. V. Jagadish , Nick Koudas , S. Muthukrishnan , Viswanath Poosala , Kenneth C. Sevcik , Torsten Suel, Optimal Histograms with Quality Guarantees, Proceedings of the 24rd International Conference on Very Large Data Bases, p.275-286, August 24-27, 1998

Viswanath Poosala , Yannis E. Ioannidis, Selectivity Estimation Without the Attribute Value Independence Assumption, Proceedings of the 23rd International Conference on Very Large Data Bases, p.486-495, August 25-29, 1997

Ashish Goel , Sudipto Guha , Kamesh Munagala, Asking the right questions: model-driven optimization using probes, Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, June 26-28, 2006, Chicago, IL, USA ueries. Technical report, Bell Laboratories, Murray Hill, New Jersey, 1999.

P.J. Haas. Hoeffding inequalities for join-

selectivity estimation and online aggregation. Technical Report RJ 10040, IBM Almaden Research Center, San Jose, CA, 1996.

Bart Kuijpers, Walied Othman, "Trajectory Databases: Data Models, Uncertainty And Complete Query Languages", Journal of Computer and System Sciences vol. 76,pp 538-560, 2010 .

Riccardo Rosati, "On The Finite Controllability Of Conjunctive Query Answering In Databases Under Open-World Assumption", Journal of Computer and System Sciences vol. 77, pp 572-594, 2011.

Yi Lua, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R. Larus, Albert Greenberg "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", Performance Evaluation, vol 68, pp. 1056-1071, 2011.

J-P. Clech, D. M. Noctor, J. C. Manock, G. W. Lynott and F. E. Bader AT&T Bell Laboratories, "Surface Mount Assembly Failure Statistics And Failure-Free Times", 44th ECTC, Washington, D.C., May 1-4, 1994, pp. 487-497.