ORACLE DATABASE PL/SQL

ORACLE PL/SQL Tuning

DBA is blamed for the SQL failure by Developers.
DBA replies- Developers should have taken care of this before deploying anything into Production Database.
Developer: Why you are not aware of this?
Manager: Now onwards, DBA will review all the Queries and approve them before deployment.

Now, the Challenge is: What is the most efficient way to manage this process?

Oracle Tuning Tips- First step to tune your query is to check the EXECUTION Plan and then try to structure and tune your query with proper analysis.
Query Analysis -

Examine the Execution Plan of your SQL- Examine cost, row count and time. Find the expensive operators.
Review filters and Access Predicates- Check how query is interpreted. Review syntaxes and datatypes.
Gather Table Information- Review table definition, if it's a view then review underlying tables, Get the \table size and review statistics
Evaluate Existing Indexes- Get index definition, Check order of the columns and selectivity.
Analyze columns in Where clause- Avoid sub-queries, check the columns if it needs index.
Review Existing Keys and Constraints- Check table relationships, Check primary key definitions. Foreign key constraints can help the optimizer create better execution plans.
Tune the Query- Focus on most expensive operators, Analyze the query having high cost and largest number of rows in result set, Structure your query and Syntaxes. Follow the above steps and rerun your query.

1. Do not use the set operator UNION if the objective can be achieved through an UNION ALL. UNION incurs an extra sort operation which can be avoided. **Use UNION ALL only if you don't want your result set to be unique.**

2. **Select ONLY those columns in a query which are required.** Extra columns which are not actually used, incur more I/O on the database and increase network traffic.

3. **Do not use the keyword DISTINCT if the objective can be achieved otherwise**. DISTINCT incurs an extra sort operation and therefore slows your queries down. One of the example is to change query to fetch data using Group By instead of DISTINCT.

4. **If it is required to use a composite index, try to use the "Leading" column in the "WHERE" clause.** Though Index skip scan is possible, it incurs extra cost in creating virtual indexes and may not be always possible depending on the cardinality of the leading columns.

5. **There should not be any Cartesian product in the query** unless there is a definite requirement to do so. I know this is a silly point but we all have done this mistake at one point

6. Wherever multiple tables are used, always refer to a column by either using an alias or using the fully qualified name. **Do not leave the guess work for Oracle.**

7. **SQL statements should be formatted consistently** (e.g the keywords should be in CAPS only) to aid readability. Now, this is not a performance tip really. However, it's important and part of the practices.

8. **If possible use bind variables instead of constant/literal values** in the predicate filter conditions to reduce repeated parsing of the same statement.

9. **Use meaningful aliases for tables/views**

10. When writing sub-queries make **use of the EXISTS** operator where possible as Oracle knows that once a match has been found it can stop and avoid a full table scan (it does a SEMI JOIN). Instead of IN and Not IN – use EXISTS and NOT EXISTS operator. Exists/Not Exists returns Boolean value whereas IN/NOT IN returns value itself. **Processing Boolean value is faster than processing actual values.**

11. **If the selective predicate is in the sub query, then use IN.** In case you know what are the possible values to be matched with.

ORACLE Topics

12. **If the selective predicate is in the parent query, then use EXISTS.** In this case you will not have any idea about the result set.

13. **Do not modify indexed columns with functions such as RTRIM, TO_CHAR, UPPER, TRUNC as this will prevent the optimizer from identifying the index (Indexed column will not work if used with Function).** If possible perform the modification on the constant side of the condition. If the indexed column is usually accessed through a function (e.g NVL), consider creating a function based index.

14. Try to use an index if less than 5% of the data needs to be accessed from a data set. The exception is a small table (a few hundred rows) which is usually best accessed through a FULL table scan irrespective of the percentage of data required.

15. **Use Equi-joins whenever possible, they improve SQL efficiency**

16. Avoid the following kinds of complex expressions:

- NVL (col1,-999) = …. .
- TO_DATE(), TO_NUMBER(), and so on

These expressions prevent the optimizer from assigning valid cardinality or selectivity estimates and can in turn affect the overall plan and the join method

17. It is always better to write separate SQL statements for different tasks, but if you must use one SQL statement, then you can make a very complex statement slightly less complex by using the UNION ALL operator

18. **Joins to complex views are not recommended**, particularly joins from one complex view to another. Often this results in the entire view being instantiated, and then the query is run against the view data

19. Querying from a view requires all tables from the view to be accessed for the data to be returned. If that is not required, then do not use the view. Instead, use the base table(s), or if necessary, define a new view.

20. **While querying on a partitioned table try to use the partition key in the “WHERE” clause if possible**. This will ensure partition pruning.

21. **Consider using the PARALLEL hint (only when additional resources can be allocated) while accessing large data sets.**

22. **Avoid doing an ORDER BY on a large data set** especially if the response time is important.

23. Consider changing the OPTIMIZER MODE to FIRST_ROWS(n) if the response time is important. The default is ALL_ROWS which gives better throughput.

24. **Use CASE statements instead of** DECODE (especially where nested DECODEs are involved) because they increase the readability of the query immensely.

25. **Do not use HINTS unless the performance gains clear.**

26. Check if the statistics for the objects used in the query are up to date. If not, use the DBMS_STATS package to collect the same.

27. It is always good to understand the data both functionally and it’s diversity and volume in order to tune the query. Selectivity (predicate) and Cardinality (skew) factors have a big impact on query plan. Use of Statistics and Histograms can drive the query towards a better plan.

28. Read explain plan and try to make largest restriction (filter) as the driving site for the query, followed by the next largest, this will minimize the time spent on I/O and execution in subsequent phases of the plan.

29. **If Query requires quick response rather than good throughput is the objective, try to avoid sorts (group by, order by, etc.).** For good throughput, optimizer mode should be set to ALL ROWS.

30. Queries tend to perform worse as they age due to volume increase, structural changes in the database and application, upgrades etc. Use Automatic Workload Repository (AWR) and Automatic Database Diagnostic Monitor (ADDM) to better understand change in execution plan and throughput of top queries over a period of time.

31. SQL Tuning Advisor and SQL Access Advisor can be used for system advice on tuning specific SQL and their join and access paths, however, advice generated by these tools may not be always applicable (point 28).

32. SQL Access paths for joins are an component determining query execution time. Hash Joins are preferable when 2 large tables need to be joined. Nested loops make work better when a large table is joined with a small table.

33. Reduce access count to DB using BULK operations (Less context switching Better Performance) and use frequent commits.

34. Identify high load or top SQL statements responsible for a large share of the Application workload and system resources by reviewing the past execution history available in the system.

35. Implementing correlative actions to generate better execution plan for poorly performing SQL statements.

36. **Balance the workload –** Schedule any non-critical tasks and batch jobs at the night time when there will be less traffic on server. This way we can achieve better performance as it frees up the resources for the more critical programs in a day.

Above steps are repeated until the system performance reaches a satisfactory level or no more statements can be tuned.

Get involved and leave your Comments in the Box Below. The more people get involved, the more we all benefit. **So, leave your thoughts before you leave the page.**

10 comments:

bala subramaniam January 24, 2017 at 4:40 AM

vert useful info Thanks for posting

Reply

Replies

Ravikant Baluni January 27, 2017 at 10:22 AM

You are most welcome.

Reply

Vicky ETL January 28, 2017 at 8:26 AM

Hi Ravi,
I have one question .
could you please explain the first point i.e. Examine the cost , rowcount and time, find expensive operation . With example

Reply

Replies

Ravikant Baluni January 28, 2017 at 9:41 AM

Hi,
that can be examined through the Execution Plan of your Query(Posted in blog how to check this). Get an estimate of the run-time for a given operation.
But one can not completelt rely only on this so better to check with Autotrace also will tell the Optimizer estimates will be returned by any particular operation.
Explain generates a tree with higher to lower cost but Autotrace will also give you the statistics like consistent gets and rows processed.
can also be done by GATHER_PLAN_STATISTICS which captures execution statistics the number of rows actually returned by each operation in the execution plan.
for expensive operators some points are already there in the post to avoid them like do not use SQL functions in the WHERE clauses unless it is really required.
As you know any expression using an indexed column causes the optimizer to ignore the possibility of using an index on that column. Will post about other factors practically like Tuning ora full table scans, tuning with indexes, tuning parallel sql execution etc.....

Nav January 28, 2017 at 9:51 AM

This comment has been removed by the author.

Vicky ETL January 28, 2017 at 9:52 AM

thanks for reply

Reply

Vicky ETL January 28, 2017 at 8:37 AM

what is partition pruning ?

Reply

Replies

Ravikant Baluni January 28, 2017 at 9:52 AM

if you have already implemented PArtitioning then you will understand this better.
using pruning partitions of the table can be ignored if you are not looking for complete information. this can be

- ORACLE Driving Tables
- EXECUTION PLAN and It's Components
- CURSOR_SHARING in ORACLE
- INDEX Usage with LIKE Operator and DOMAIN Index
- DYNAMIC_SAMPLING and its Impact or OPTIMIZER
- NOT NULL and Indexed Column
- ORACLE AUTOTRACE Utility
- Pass COMMA Separated Value to IN Operator
- ANALYTICAL & AGGREGATE Functions Examples
- ORACLE UTL_FILE Package
- ORACLE UTL_FILE Exceptions
- UTL_FILE Operations and Functions
- Comma Separated Values
- RETURNING Table From a Function
- REGULAR Expressions in ORACLE
- RESTRICT DROP/TRUNCATE on TABLE
- Export Table Data to CSV
- IMPORT Data from Flat Files to ORACL Tables
- ORACLE PIVOT/UNPIVOT
- PARTITIONING IN ORACLE
- Equality Test of Two COLLECTION Typ
- Compare and Merge COLLECTION Obj
- NESTED Table Functions
- DETERMINISTIC FUNCTIONS
- HANDLING CURSOR Exceptions
- When CROSS JOIN Will Be Useful?
- DML Error Logging
- Handle CONCURRENT Updates
- Pessimistic and Optimistic Oracle Loc
- Returning REF CURSOR From a Proced
- Prevent VALUE_ERROR Exception
- SYS_REFCURSOR Vs. REF CURSOR

Popular Posts

- Analytic Functions Vs. Aggregate Functions

done by partitioned key using which you will be directed to the partitioned data you want to get from Query.
example if your table is partitioned by Quarter (Jan-Mar) (Apr-Jun) (Jul-Sep) (Oct-Dec)
then where PKey = 'Apr' will prune three of the partitions from processing(rest of the data ignored while processing).In case Pkey > 'Aug' then just two of the partitions (Jul-Sep) (Oct-Dec) will be included.

Reply

It's Siri June 1, 2018 at 9:04 AM

its very useful and everything is clear with examples and its extremely good and recommended to my friends also. please share if u have any other skills can u share your knowledge and any documents. and waiting for your new updates.

Reply

Replies

Ravikant Baluni June 7, 2018 at 8:26 AM

Sure. I will be posting Unix and Sybase blogs soon. Will keep you notify the same.
Or you can check in addition to the below home page:
https://tipsfororacle.blogspot.com/p/homepage.html

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)