database optimization:

Few best practices that can help to improve database speed and efficiency.

1. Keep statistics up to date

Table statistics are used to generate optimal execution plans. If the performance tuning tool is using out-of-date statistics, the plan won’t be optimized for the current situation.

BEGIN

DBMS\_STATS.GATHER\_SYSTEM\_STATS(

gathering\_mode => 'interval',

interval => 720,

stattab => 'mystats',

statid => 'OLTP');

END;

/

Gather statistics during the night. Gathering ends after 720 minutes and is stored in the mystats table:

BEGIN

DBMS\_STATS.GATHER\_SYSTEM\_STATS(

gathering\_mode => 'interval',

interval => 720,

stattab => 'mystats',

statid => 'OLAP');

END;

/

2. Don’t use leading wildcards

Leading wildcards in parameters force a full table scan, even if there is an indexed field inside the table. If the database engine must scan all the rows in a table to find what it’s looking for, the delivery speed of your query results suffers. Other queries may suffer as well, since scanning all of that data into memory will cause the CPU utilization to spike and not allow other queries any time in memory.

3. Avoid SELECT \*

This tip is particularly important if you have a large table (think hundreds of columns and millions of rows). If an application only needs a few columns, include them individually instead of wasting time querying for all the data. Again, reading extra data will cause CPU utilization to spike and memory to be thrashed. You should check the Page Life Expectancy (PLE) to make sure you are not having this issue.

4. Use constraints

Constraints are an effective way to speed up queries and helps the SQL optimizer come up with a better execution plan, but the improved performance comes at the cost of the data requiring more memory. The increased query speed may be worth it depending on the business objective, but it’s important to be aware of the price.

5. Look at the actual execution plan, not the estimated plan

The estimated execution plan is helpful when you are writing queries because it gives you a preview of how the plan will run, but it is blind to parameter data types which could be wrong. To get the best results when performance tuning, it’s often better to review the actual execution plan because it uses the latest, most accurate statistics.

6. Adjust queries by making one small change at a time

Making too many changes at once tends to muddy the waters. A better, more efficient approach to query tuning is to make changes with the most expensive operations first and work from there.

7. Adjust indexes to reduce I/O

Before you dive into troubleshooting I/O directly, first try adjusting indexes and query tuning. Consider using a covering index that includes all the columns in the query, this reduces the need to go back to the table as it can get all the columns from the index. Adjusting indexes and query tuning have a high impact on almost all areas of performance, so when they are optimized, many other performance issues resolve as well.

8. Analyze query plans

Utilizing artificial intelligence to analyze your execution plan and determine how to change it helps databases execute operations more efficiently.

9. Compare optimized and original SQL

When optimizing SQL queries, be sure to highlight changes in the SQL statement so you can compare the original statement with the optimized version. Gather a baseline metric such as logical I/O to compare against as you tune. Don’t make any changes until you are sure the optimized version is accurate (i.e., includes current statistics) and really does improve performance.

10. Automate SQL optimization

Automated SQL optimization tools not only analyze your SQL statement but can also automatically rewrite it or optimize indexes until it finds the variation that creates the most improvement in the execution time of the query.