SQL - DATABASE TUNNING

It takes time to become a Database Expert or an expert Database Administrator. This all comes with lot of experience in various database designs and good trainings.

But the following list may be helpful for the beginners to have a nice database performance −

- Use 3BNF database design explained in this tutorial in RDBMS Concepts chapter.
- Avoid number-to-character conversions because numbers and characters compare differently and lead to performance downgrade.
- While using SELECT statement, only fetch whatever information is required and avoid using * in your SELECT queries because it would load the system unnecessarily.
- Create your indexes carefully on all the tables where you have frequent search operations. Avoid index on the tables where you have less number of search operations and more number of insert and update operations.
- A full-table scan occurs when the columns in the WHERE clause do not have an index associated with them. You can avoid a full-table scan by creating an index on columns that are used as conditions in the WHERE clause of an SQL statement.
- Be very careful of equality operators with real numbers and date/time values. Both of these can have small differences that are not obvious to the eye but that make an exact match impossible, thus preventing your queries from ever returning rows.
- Use pattern matching judiciously. LIKE COL% is a valid WHERE condition, reducing the returned set to only those records with data starting with the string COL. However, COL%Y does not further reduce the returned results set since %Y cannot be effectively evaluated. The effort to do the evaluation is too large to be considered. In this case, the COL% is used, but the %Y is thrown away. For the same reason, a leading wildcard %COL effectively prevents the entire filter from being used.
- Fine tune your SQL queries examining the structure of the queries *andsubqueries*, the SQL syntax, to discover whether you have designed your tables to support fast data manipulation and written the query in an optimum manner, allowing your DBMS to manipulate the data efficiently.
- For queries that are executed on a regular basis, try to use procedures. A procedure is a potentially large group of SQL statements. Procedures are compiled by the database engine and then executed. Unlike an SQL statement, the database engine need not optimize the procedure before it is executed.
- Avoid using the logical operator OR in a query if possible. OR inevitably slows down nearly any query against a table of substantial size.
- You can optimize bulk data loads by dropping indexes. Imagine the history table with many thousands of rows. That history table is also likely to have one or more indexes. When you think of an index, you normally think of faster table access, but in the case of batch loads, you can benefit by dropping the index*es*.
- When performing batch transactions, perform COMMIT at after a fair number of records creation in stead of creating them after every record creation.
- Plan to defragment the database on a regular basis, even if doing so means developing a weekly routine.

Built-In Tuning Tools

Oracle has many tools for managing SQL statement performance but among them two are very popular. These two tools are −

- **Explain plan** − tool identifies the access path that will be taken when the SQL statement is executed.
- **tkprof** − measures the performance by time elapsed during each phase of SQL statement processing.

If you want to simply measure the elapsed time of a query in Oracle, you can use the SQL*Plus command SET TIMING ON.

Check your RDBMS documentation for more detail on the above-mentioned tools and defragmenting the database.