DATA WAREHOUSING - TUNING

A data warehouse keeps evolving and it is unpredictable what query the user is going to post in the future. Therefore it becomes more difficult to tune a data warehouse system. In this chapter, we will discuss how to tune the different aspects of a data warehouse such as performance, data load, queries, etc.

Difficulties in Data Warehouse Tuning

Tuning a data warehouse is a difficult procedure due to following reasons:

- Data warehouse is dynamic; it never remains constant.
- It is very difficult to predict what query the user is going to post in the future.
- Business requirements change with time.
- Users and their profiles keep changing.
- The user can switch from one group to another.
- The data load on the warehouse also changes with time.

Note: It is very important to have a complete knowledge of data warehouse.

Performance Assessment

Here is a list of objective measures of performance:

- Average query response time
- Scan rates
- Time used per day query
- Memory usage per process
- I/O throughput rates

Following are the points to remember.

- It is necessary to specify the measures in service level agreement *SLA*.
- It is of no use trying to tune response time, if they are already better than those required.
- It is essential to have realistic expectations while making performance assessment.
- It is also essential that the users have feasible expectations.
- To hide the complexity of the system from the user, aggregations and views should be used.
- It is also possible that the user can write a query you had not tuned for.

Data Load Tuning

Data load is a critical part of overnight processing. Nothing else can run until data load is complete. This is the entry point into the system.

Note: If there is a delay in transferring the data, or in arrival of data then the entire system is affected badly. Therefore it is very important to tune the data load first.

There are various approaches of tuning data load that are discussed below:

- The very common approach is to insert data using the **SQL Layer**. In this approach, normal checks and constraints need to be performed. When the data is inserted into the table, the code will run to check for enough space to insert the data. If sufficient space is not available, then more space may have to be allocated to these tables. These checks take time to perform and are costly to CPU.

- The second approach is to bypass all these checks and constraints and place the data directly into the preformatted blocks. These blocks are later written to the database. It is faster than the first approach, but it can work only with whole blocks of data. This can lead to some space wastage.
- The third approach is that while loading the data into the table that already contains the table, we can maintain indexes.
- The fourth approach says that to load the data in tables that already contain data, **drop the indexes & recreate them** when the data load is complete. The choice between the third and the fourth approach depends on how much data is already loaded and how many indexes need to be rebuilt.

Integrity Checks

Integrity checking highly affects the performance of the load. Following are the points to remember.

- Integrity checks need to be limited because they require heavy processing power.
- Integrity checks should be applied on the source system to avoid performance degrade of data load.

Tuning Queries

We have two kinds of queries in data warehouse:

- Fixed queries
- Ad hoc queries

Fixed Queries

Fixed queries are well defined. Following are the examples of fixed queries:

- regular reports
- Canned queries
- Common aggregations

Tuning the fixed queries in a data warehouse is same as in a relational database system. The only difference is that the amount of data to be queried may be different. It is good to store the most successful execution plan while testing fixed queries. Storing these executing plan will allow us to spot changing data size and data skew, as it will cause the execution plan to change.

Note: We cannot do more on fact table but while dealing with dimension tables or the aggregations, the usual collection of SQL tweaking, storage mechanism, and access methods can be used to tune these queries.

Ad hoc Queries

To understand ad hoc queries, it is important to know the ad hoc users of the data warehouse. For each user or group of users, you need to know the following:

- The number of users in the group
- Whether they use ad hoc queries at regular intervals of time
- Whether they use ad hoc queries frequently
- Whether they use ad hoc queries occasionally at unknown intervals.
- The maximum size of query they tend to run

- The average size of query they tend to run
- Whether they require drill-down access to the base data
- The elapsed login time per day
- The peak time of daily usage
- The number of queries they run per peak hour

Points to Note

- It is important to track the user's profiles and identify the queries that are run on a regular basis.
- It is also important that the tuning performed does not affect the performance.
- Identify similar and ad hoc queries that are frequently run.
- If these queries are identified, then the database will change and new indexes can be added for those queries.
- If these queries are identified, then new aggregations can be created specifically for those queries that would result in their efficient execution.