Query Optimization Tips, EXPLAIN Statement and Query Execution Plan

Query Optimization

- With the added complexity of growing data volumes and ever-changing workloads, database performance tuning and MySQL query optimization are now necessary to maximize resource utilization and system performance.
- There are several reasons which make SQL tuning a bit complex for developers.
- Firstly, it requires extensive technical expertise to write and understand different execution plans.
- While writing clean and complete SQL statements is the responsibility of the one who garners thorough knowledge of it.
- Besides its complexity, tuning is very time-consuming.
- Because when you have a large number of SQL statements to sort through, it brings a bit of uncertainty to find out which statements must you tune up and which one should you leave.
- And while every statement is different, their tuning approach also varies according to their respective functionalities.

EXPLAIN Statement

- The EXPLAIN statement provides information about how MySQL executes a statement.
- EXPLAIN works alongside SELECT, DELETE, INSERT, REPLACE, and UPDATE statements.
- It displays information from a built-in MySQL optimizer regarding the statement execution plan and the number of rows scanned in each table.
- Thus we can determine the cost of the query.
- EXPLAIN Select * from sakila.actor limit 100;

Benefits of Query Optimization

- The major advantage of identifying the performance driving factor for database allows you to avoid over-provisioning and reducing cost by right-sizing your servers.
- It also gives you insights into whether moving data storage or adding server capacity will bring improvement in performance or not, and if so, then how much will it be.
- Once tuned properly, the database gives worthwhile performance results with great functionalities. It not only lowers unwanted task load but also optimizes the MySQL database for faster data retrieval.

Optimization Of Database Schema

- The database structure is very crucial in performance optimization.
- **Limiting the number of columns :** MySQL has a limit of 4096 columns per table. Use fewer columns for better performance.
- **Normalize Tables :** Normalizing keeps all data non-redundant. The database that is in this state is called 3NF (third normal form).
- **Use the Most Appropriate Data Types** : Tables should be designed to minimize space used on a disk. Tables that occupy less disk space results in smaller indexes that can be processed in a shorter duration. For example, if a table will host less than 100 records, you should use the TINYINT data type for the unique ID as it takes less space than INT.
- **Avoid Null Values :** Declare columns to be NOT NULL where possible. This enables better use of indexes. NULL values increase the processing power needed for testing whether each value is NULL, making SQL operations slower.

Guidelines for Query Optimization

- First of all, ensure indexing of all the predicates in WHERE, JOIN, ORDER BY, and GROUP BY clauses.
- Strongly recommender indexing of predicates to augment SQL performance.
- Because improper indexing of SQL queries can cause table scans, which eventually lead up to locking problems and other issues.
- Avoid using functions in predicates
- SELECT * FROM TABLE1 WHERE UPPER(COL1)='ABC'Copy
- Avoid using a wildcard (%) at the beginning of a predicate
- The predicate LIKE '%abc' causes a full table scan. For example:
- SELECT * FROM TABLE1 WHERE COL1 LIKE '%ABC'Copy
- In most cases, this wildcard usage brings major performance limitations.

Guidelines for Query Optimization

- Use inner join, instead of outer join if possible
- Use outer join only when it is necessary. Using it needlessly not only limits database performance but also limits MySQL query optimization options, resulting in slower execution of SQL statements.
- Use DISTINCT and UNION only if it is necessary
- Using UNION and DISTINCT operators without any major purpose causes unwanted sorting and slowing down of SQL execution. Instead of UNION, using UNION ALL brings more efficiency in the process and improves MySQL performance more precisely.
- The ORDER BY clause is mandatory in SQL if you expect to get a sorted result
- The ORDER BY keyword sorts the result-set in predefined statement columns. Though the statement brings advantage for the database admins for getting the sorted data, it also produces a bit performance impact in the SQL execution. Because the query first needs to sort the data for producing the final result-set, causing a bit complex operation in the SQL execution.

Guidelines for Query Optimization

- Use inner join, instead of outer join if possible
- Use outer join only when it is necessary. Using it needlessly not only limits database performance but also limits MySQL query optimization options, resulting in slower execution of SQL statements.
- Use DISTINCT and UNION only if it is necessary
- Using UNION and DISTINCT operators without any major purpose causes unwanted sorting and slowing down of SQL execution. Instead of UNION, using UNION ALL brings more efficiency in the process and improves MySQL performance more precisely.
- The ORDER BY clause is mandatory in SQL if you expect to get a sorted result
- The ORDER BY keyword sorts the result-set in predefined statement columns. Though the statement brings advantage for the database admins for getting the sorted data, it also produces a bit performance impact in the SQL execution. Because the query first needs to sort the data for producing the final result-set, causing a bit complex operation in the SQL execution.