Neenu

27 March 2019

Offshoring

Tips to Improve MySQL Query Performance

MySQL is the most popular open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

A single poorly-designed SQL query will pose a significant threat to the overall performance of your application. Therefore, optimizing query performance is essential. MySQL comes with tools that help us in the optimization of queries. Let's have a look at the most important and useful tips to improve MySQL Query for speed and performance.

1. Optimize Your Database

You need to know how to design schemas to support efficient queries. Well-designed queries and schema are crucial for your application to work properly.

Optimizing your MySQL queries alone will not bring excellent database performance. A well-structured database is crucial along with an optimized query. Otherwise, in the event of data surge, database performance will be adversely affected.

The following strategies will help you to optimize your database.

a. Normalize Tables

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data. It divides larger tables to smaller tables and links them using relationships. This is to avoid having fields that would be null and making sure that all fields in the table only belong to one domain of data being described.

For example, in the employee table, the fields could be id, name, social security number, but those three fields have nothing to do with the department. Only employee id describes which department the employee belongs to. So this implies that which department an employee is in should be in another table.

b. Use Optimal Data Types

MySQL supports different data types and choosing the correct type to store your data is vital to have good performance. MySQL supports data of numeric types, date and time types, and string (character) types (integer, float, double, date, date_time, Varchar, text, etc.). Different data types serve different purposes. When creating your tables you need to understand what type of data each column will hold and choose the most fitting data type.

If a field expects a date value, using a date_time data type is the best because you don't have to run complicated functions to convert the field to date when retrieving records using SQL. Use integer values if you expect all values to be numbers. When it comes to computation, MySQL can do better with integer values as compared to text data types such as Varchar (stores variable-length character strings and is the most common string data type).

c. Avoid Null Values

Allowing null values (absence of any value in a column) in your database is a really bad idea unless the field can logically have a null value. The presence of null value can adversely affect your database results.

For instance, if you want to get the sum of all orders in a database, the anticipated result might behave badly if a particular order record has a null amount. This kind of misbehavior may not happen only if you have used MySQL ‘ifnull’ statement to return alternative value.

d. Avoid Too Many Columns

The biggest downside to having many columns is extra IO and storage overhead. Having wide tables can be extremely expensive and causes storage overhead. It is ideal no to go above a hundred unless your business logic specifically necessitates this.

As opposed to creating one wide table, splitting it apart into logical structures can be beneficial. Suppose you are creating an employee table. In certain instances, you realize that an employee can have multiple addresses. Then it is better to create a separate table for entering employees’ addresses that refer back to the employee table using the ’employee_id’ field.

2. Optimize Joins

Reduce the join statements in queries. An SQL statement with a poorly designed pattern that involves a lot of joins may not work well. A rule of thumb is to have utmost a dozen joins for each query.

3. Index All Columns Used in ‘where’, ‘order by’, and ‘group by’ Clauses

INDEXES. A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

When creating an index, the equality conditions in the WHERE and JOIN conditions are really important. For instance, conditions such as name = ‘John’ will allow the database to filter most of the rows from the table and run through a small number of rows to return the required results. Thus, we should start indexing by adding these columns to the index.

Then, you should only add one of the most selective range conditions as MySQL can’t handle more of them. In some cases when there are no range conditions, it is logical to add the GROUP BY / ORDER BY columns, assuming the ordering is done in only one direction (ASC / DESC).

GROUP BY clause is used for ordering the result and hence if:

-the correct order of the index is used or
-only leftmost columns are used in group by
-the leftmost column is used in WHERE clause and rest in the correct order in GROUP BY clause index would be used.

4. Use Full-Text Searches

MySQL full-text search (FTS) is far much faster than queries using wildcard characters. To add a full-text search index to the students’ sample table, we can use the below MySQL command:

```
mysql>Alter table students ADD FULLTEXT (first_name, last_name);
```

```
mysql>Select * from students where match(first_name, last_name)
AGAINST (‘Jones’);
```

In the above example, we have specified the columns that we want to be matched (first_name and last_name) against our search keyword (‘Jones’).

Only a single row will be scanned even if our students’ database has huge rows and this will speed up the database.

5. Optimize Like Statements With Union Clause

Consider a situation when you want to run queries using the comparison operator ‘or’ on different fields or columns in a particular table. Here, if you use the ‘or’ keyword excessively in WHERE clause, there are chances that MySQL optimizer may incorrectly choose a full table scan to retrieve a record.

If you have an index that can optimize one side of the query and a different index to optimize the other side, a union clause can make the query run faster. For example; consider a case where you are running the below query with the **‘first_name’** and **‘last_name’** indexed:

```
mysql> select * from students where first_name like ‘Jones%’ or
last_name like ‘Jones%’ ;
```

The query above can run far much slower compared to the below query which uses a union operator merge the results of 2 separate fast queries that takes advantage of the indexes.

```
mysql> select from students where first_name like ‘Jones%’ union all
select from students where last_name like ‘
```

6. MySQL Query Caching

As we know, caching is used to improve performance. It will faster the site or application. The MySQL query cache is a global one shared among the sessions. The query cache stores results of SELECT queries enabling the quick return of the query if an identical query is received in future. When the server for the same query, MySQL ask will retrieve the results from the cache instead of running the query again. This will indeed fasten the process significantly. The results will be set in a memory cache like Memcached or Cassandra.

Configuration Directives

Let’s we check how to enable the query caching in a MySQL server.

In order to enable the query caching, we need to add the following configuration directives.

1) query_cache_size=SIZE
2) query_cache_type=OPTION
query_cache_size=SIZE

The first directive that is needed to permit query caching in MySQL servers is the “query_cache_size=SIZE”.

This directive enables us to set the size of memory allocated for caching query results. In a typical server, the default value for this directive will be ‘0’. It means that the query cache is disabled. To enable the query caching we need to set some value to this. We should set the value according to how much memory we are planning to allocate for query caching.

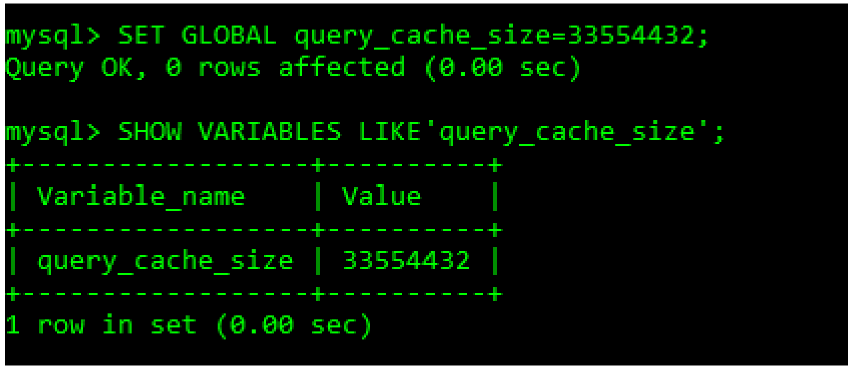
query_cache_type=OPTION
The next configuration directive that is needed to be set to enable query caching is “query_cache_type=OPTION”. Using this directive, we specify which type of query cache we are setting.

There are 3 possible options that can be set for this directive. They are:

1) The zero “0” : – It tells the server the following: Don’t cache the results in or retrieve results from the query cache.
2) The one “1” : – There are query results that start with “SELECT S_NO_CACHE”. If we set the value for this directive as one, it signifies that, to cache all the query results except for those that begin with SELECT S_NO_CACHE.
3) The two “2”:– If we set the value as “2”, it means that cache results only for queries that begin with SELECT SQL_CACHE.

Enabling Query Caching in MySQL

We can set up the caching in the following format. You need to enter into MySQL. Assume that, we are setting up the query cache for 32 Mb. We need to do the following to acquire this.



```
mysql> SET GLOBAL query_cache_size=33554432;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE'query_cache_size';
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| query_cache_size | 33554432 |
+------------------+----------+
1 row in set (0.00 sec)
```

Now, we can append other configuration directives as follows:

```
query_cache_size = 268435456
query_cache_type=1
query_cache_limit=1048576
```

The above commands imply that the maximum size of individual query results that can be cached set to 1048576 using query_cache_limit system variable. The memory size is set in Kb.

Conclusion

In this guide, I have shown you how to optimize your MySQL server. Following these steps along with creating a well-structured database table will definitely improve your MySQL Query performance.

About Neenu

Neenu, Graduate in Computer Science Engineering with more than 7 years of experience in software development. She is a positive thinker, team player, and quick learner with extensive experience in Php, Magento and UI development.

View all posts by Neenu →

2 thoughts on “Tips to Improve MySQL Query Performance”

Meysam

2 May 2020 4:07 PM

It was so helpful, Thanks a lot.

Admin

5 May 2020 10:26 AM

We’re happy that you’ve found it useful.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *	
Name *	
Email *	
Website	

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Newsletter

Get regular updates in your inbox about hot technology trends and latest advancements from our team. Read our Privacy Policy

Enter your email