IJSES
Volume 8, Issue 11, pp. 23-26, 2024.
International Journal of Scientific Engineering and Science
ISSN (Online): 2456-7361

Optimization of queries to MySQL and PostgreSQL Databases for High-Load Applications

Mykhailo Shumilov
CTO at Vadimages, Vancouver, WA, USA
Kharkiv, Ukraine

***Abstract*— *Optimization of queries to MySQL and PostgreSQL databases for high-load applications is aimed at improving system performance and reducing response time under heavy loads. The aim of the research is to develop methods that can reduce the load on servers and speed up the execution of complex queries. The methodology includes indexing analysis, caching usage, data partitioning, and the use of various types of joins such as JOINS and subqueries. The results show that query optimization and the use of indexing significantly improve performance, however, they require taking into account the specifics of a particular application and the type of database used. In conclusion, the importance of choosing the optimal approach depending on the application architecture and load profile is emphasized, as well as the need for further research in the field of automating query optimization using machine learning*.

***Keywords*— *Query optimization, MySQL, PostgreSQL, indexing, caching, high-load applications*.

I. INTRODUCTION

In the current conditions of information technology, databases play a key role in the functioning of high-load applications. As the number of users and the volume of data increase, it becomes critically important to ensure fast query execution and minimal response time. MySQL and PostgreSQL, as popular database management systems (DBMS), are used in numerous applications, from simple websites to complex enterprise systems. However, as data volumes grow and the number of users increases, the load on database servers increases, requiring the search for and implementation of query optimization methods to maintain stable and high performance.

Query optimization for MySQL and PostgreSQL databases becomes a relevant task for developers and administrators seeking to reduce server load and improve system efficiency. Various methods, such as indexing, caching, and partitioning, allow for more rational resource management; however, their selection and application require a deep analysis of the application architecture and data specifics. Improper use of optimization techniques can lead to performance degradation and reduced system reliability, which highlights the importance of research in this area.

The purpose of this work is to study and develop effective query optimization methods for MySQL and PostgreSQL databases, aimed at improving the performance of high-load applications and minimizing server response time under intensive load conditions.

II. MATERIALS AND METHODS

PostgreSQL became a strong competitor in the database market, challenging the long-standing dominance of MySQL. According to the StackOverflow 2023 developer survey, PostgreSQL took first place with an impressive 45% developer adoption rate. On the contrary, MySQL has an indicator of 41%, which reflects a noticeable shift in the preferences of web developers [1].

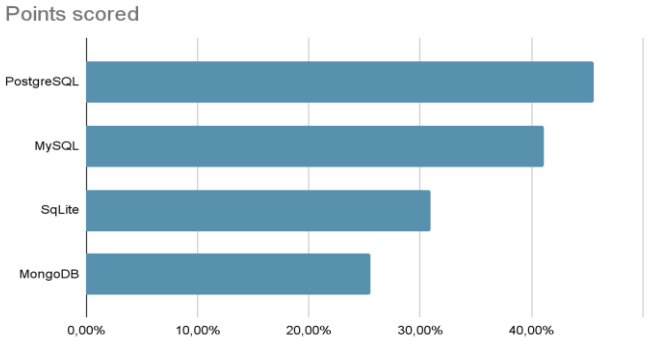

FIGURE 1. Developer opinions on databases [1].

MySQL is a widely used relational database management system (RDBMS) that implements the capabilities of the SQL language. Initially developed for small and medium-sized databases, this system is now capable of handling significant volumes of data. MySQL was developed using the C and C++ programming languages, which allows it to comply with SQL standards, but it also includes a number of extensions, focusing on speed and reliability [2].

MySQL functions are built-in procedures that allow various operations and calculations with the data stored in the MySQL database. MySQL has several key characteristics, which are described in Table 1 for clarity.

Due to these characteristics, MySQL is widely used in various fields—from small projects to large enterprise systems. It is particularly effective for web development, content management systems, e-commerce platforms, as well as business-critical applications where a reliable and high-performance database management system is required [3]. MySQL supports various data types that define the format and structure of data in the columns of database tables. Each column must be associated with a specific data type, determining what data is permissible to insert and in what format [4].

Next, general theoretical aspects of PostgreSQL will be considered. PostgreSQL is a powerful object-relational database management system (DBMS) with open-source code,

known for its high reliability and performance. Created in 1986 at the University of California, Berkeley, PostgreSQL is the evolutionary continuation of the Ingres project and combines the advantages of both relational and object-oriented data models. This DBMS offers a wide range of features, including support for complex queries, ACID-compliant transactions, Multi-Version Concurrency Control (MVCC)-based parallelism management, as well as advanced indexing capabilities and support for the JSON format, making it suitable for working with both traditional structured data and modern applications requiring the handling of unstructured data [5].

TABLE 1. Key characteristics of MySQL [3].

Characteristics of MySQL	Description
Relational data structure	MySQL implements a relational model approach, organizing information in the form of tables with columns and rows. SQL, a structured query language, is used to perform various operations for querying and managing data.
Open-source	MySQL is an open-source system, providing users with the ability to not only use the software for free but also modify and distribute it. A commercial version, MySQL Enterprise, is also available with additional features and support for corporate clients.
Cross-platform	The software supports various operating systems, including Windows, macOS, and Linux, allowing it to be used in a wide range of environments and across multiple platforms.
Scalability flexibility	MySQL can be adapted to work with both small applications and large enterprise systems. The system supports different storage engines, such as InnoDB and MyISAM, which offer various performance and functionality options.
High performance	Due to its speed and efficiency, MySQL is used to create high-load websites and applications. It handles large volumes of data and supports simultaneous connections from multiple users.
ACID compliance	The MySQL database system adheres to ACID standards, ensuring reliable transaction execution and data integrity. The main ACID principles include atomicity, consistency, isolation, and durability, which are essential characteristics of reliable database management systems.
Support for replication and clustering	MySQL includes features for replication and clustering, enabling data to be distributed across multiple servers. This improves fault tolerance, load balancing, and increases system resilience.
Data security	The system provides security features such as access control, data encryption, user authentication, and auditing, ensuring the reliable protection of sensitive information.
Active community and technical support	A large community of developers and users has formed around MySQL, offering extensive documentation, educational materials, and support. Commercial clients also have access to support from Oracle.
Integration with various technologies	MySQL can be integrated with many programming languages and frameworks, thanks to the availability of drivers and connectors, making it a versatile tool for software development.

One of the key advantages of the PostgreSQL database management system is its architectural structure. Like many commercial DBMSs, PostgreSQL supports a client-server interaction model, which brings significant benefits to both developers and end-users. The core of PostgreSQL is the database server process, which runs on a single server. It is important to note that PostgreSQL currently lacks the high availability technology used in other commercial enterprise-level database management systems. Such systems provide load distribution across multiple servers, enabling greater scalability and resilience to external factors.

Application access to PostgreSQL data is carried out through a specialized database process that ensures client-server interaction. This means that client programs do not have direct access to the data, even if they are running on the same machine as the server [6].

Summarizing the above, it can be said that PostgreSQL and MySQL are relational database management systems that organize data storage in tables connected by common column values. For example, a company stores information about customers in a "Customers" table, which contains columns such as customer ID, name, and address. Product data is recorded in a "Products" table, which includes columns for product ID, name, and price. To track customer purchases, a "Customer Orders" table is created with columns that indicate the customer and product IDs.

The main similarities between PostgreSQL and MySQL include the use of SQL for performing read and data modification operations. Both platforms are open-source and supported by an active developer community. Additionally, both systems offer built-in features for backup, replication, and access control [7]. However, despite their conceptual similarities, there are many differences to consider when selecting and implementing these systems (Table 2).

TABLE 2. Differences between MySQL and PostgreSQL [8,9].

Characteristic	MySQL	PostgreSQL
ACID compliance	ACID compliance is achieved when using InnoDB and NDB Cluster.	Full ACID compliance in all configurations.
Concurrency mechanisms	Does not support full multiversion concurrency.	Uses MVCC for safe concurrent reading and updating of data.
Data indexing	Supports B-tree and R-tree indexing for hierarchical data ordering.	Supports multiple index types (trees, hash, partial, and expression indexes).
Data types	A relational DBMS with limited object handling.	An object-relational DBMS with support for inheritance and hierarchies, adapted for Java and .NET.
View support	Supports views but does not allow creating materialized views.	Supports materialized views to accelerate complex queries.
Stored procedures and triggers	Supports stored procedures using SQL, with triggers limited to standard functions.	Supports stored procedures in other programming languages, with triggers offering extended capabilities such as INSTEAD OF.

Thus, when considering scalability, it is important to focus on the specific requirements of the application. If a high degree of concurrency and horizontal scalability is needed, MySQL is better suited. PostgreSQL, on the other hand, is the preferred choice for tasks involving complex transactions and analytics.

III. Results and Discussion

This section presents the results of optimization measures aimed at improving query performance under high-load conditions. Optimization in MySQL and PostgreSQL involves transaction management, memory parameter tuning, and the use of query profilers to analyze execution time. PostgreSQL employs various index types, such as B-tree and GiST, to enhance data access speed, as well as monitoring tools like pg_stat_statements, which help identify performance issues [10].

1. Query optimization and database performance improvement. The significance of query optimization. Query optimization is a key step in accelerating application performance. An efficiently organized database significantly improves API response speed and overall user experience. In most cases, users expect immediate responses, and delays can greatly impact system performance. There have been instances where queries took an extended time to execute, affecting the stability and responsiveness of the application. Addressing such issues is crucial to ensuring high system performance.
2. Typical performance issues. One common mistake is the improper use of the "IN" operator. It may seem convenient when quickly joining tables, but as data volume increases, significant delays occur. For example, if there are more than 15,000 records in the database, and a subquery matches half of them, using the "IN" operator can substantially reduce query execution speed. In such situations, using JOIN improves execution speed.
3. The importance of proper indexing. Indexing plays a crucial role in speeding up queries, especially when grouping or sorting data. Without effective indexing, query performance on large datasets can significantly degrade, which is particularly noticeable during analytical operations. However, it is important to use indexes wisely. There are cases where multiple indexes are created on the same column unnecessarily. For example, for fields like post_id and user_id, a combined index may be more efficient than separate ones. It is important to remember that excessive indexing can slow down database performance, as indexes are updated during each insert, update, or delete operation.
4. Deindexing and caching as additional methods. Although normalization is a common practice, data denormalization can improve query performance. Depending on the specifics of the queries, storing the category name directly in the posts table may be more efficient than placing it in a separate table. However, this method requires caution to maintain data integrity. Caching is also a powerful tool, especially if the data rarely changes. Implementing caching reduces the load on the database and increases application response speed.
5. Balancing optimization with other aspects. While query optimization is an important aspect, it is also necessary to consider data integrity, consistency, and ease of maintenance. Over-optimization can lead to complex and difficult-to-maintain code. It is always essential to consider the long-term consequences of the changes being made [11].
6. Monitoring and analysis tools. Using tools such as the MySQL slow query log, EXPLAIN operators, and query profilers helps identify problem areas in queries. Regular analysis of this data aids in understanding where adjustments are needed to improve performance.
7. Handling complex queries. For complex joins, subqueries, or analytical operations, it is recommended to break them into more manageable parts. Using temporary or derived tables can be helpful in some cases. Additionally, pre-aggregating data or using materialized views can significantly improve the performance of complex queries.
8. Writing optimized queries from the start. To write efficient queries, the following steps should be adhered to, which will be described further [11].

Using indexes speeds up data searches, but they must be used correctly. Example of creating an index in MySQL and PostgreSQL (Table 3).

TABLE 3. Example of creating an index in MySQL and PostgreSQL

MySQL	PostgreSQL
CREATE INDEX idx_users_email ON users (email);	CREATE INDEX idx_users_email ON users (email);

Using EXPLAIN for query analysis. This step involves checking query execution plans to understand how they are executed and to identify bottlenecks (Table 4).

TABLE 4. Using EXPLAIN in MySQL and PostgreSQL

MySQL	PostgreSQL
EXPLAIN SELECT * FROM users WHERE email = 'example@example.com';	EXPLAIN SELECT * FROM users WHERE email = 'example@example.com';

Limiting the number of data (LIMIT). If queries return too many rows, LIMIT should be used to restrict them.

```
-- MySQL и PostgreSQL
SELECT * FROM logs ORDER BY created_at DESC LIMIT 100;
```

Next, caching at the application level (e.g., Redis) should be used to store frequently requested data, so that database queries are not made every time. Splitting large tables into smaller parts (e.g., by date) helps speed up queries. In PostgreSQL, parallel queries can be used for large tables.

```
SET max_parallel_workers_per_gather = 4;
SELECT * FROM large_table WHERE conditions;
```

Prepared statements reduce execution time for repeated queries, especially if query parameters change.

```
-- MySQL и PostgreSQL
PREPARE stmt FROM 'SELECT * FROM users WHERE email = ?';
EXECUTE stmt USING 'example@example.com';
```

These techniques significantly optimize query performance in MySQL and PostgreSQL. Always test changes in a test environment before applying them in production.

IV. CONCLUSION

In conclusion, query optimization for MySQL and PostgreSQL databases is a crucial aspect of ensuring high performance and stability in high-load applications. The study examined key optimization methods such as indexing, caching, and data partitioning, as well as performance analysis using query profilers. It was found that proper index configuration and the selection of optimal data structures significantly improve query execution speed and reduce server load. However, it is important to consider that excessive use of indexes or caching without regard to application specifics can lead to decreased performance and increased processing time for update operations.

The results of the study confirm the necessity of a comprehensive approach to optimization, taking into account application architecture, load profiles, and data characteristics. It should also be noted that the choice between MySQL and PostgreSQL should be based on the specific requirements and tasks of the application, as each of these DBMS has its own advantages and limitations. As a direction for future research, the application of machine learning methods for automating the query optimization process can be suggested, which would increase the adaptability and efficiency of database management systems under dynamically changing loads and data volumes.

REFERENCES

[1] PostgreSQL vs MySQL — The Rising Popularity of PostgreSQL. [Electronic resource] Access mode: https://medium.com/@franciscomoretti/postgresql-vs-mysql-the-rising-popularity-of-postgresql-69bdd0ab936c (accessed 07.09.2024).

[2] Šušter I., Ranisavljević T. Optimization of the MySQL database //Journal of Process Management and New Technologies. – 2023. – vol. 11. – No. 1-2. – pp. 141-151.

[3] Sovandi K. M. et al. Research on the effectiveness, capabilities and compatibility of MongoDB and MySQL: a comprehensive comparison of NoSQL and relational databases //MIND Magazine (Multimedia Artificial Intelligent Network Database). – 2023. – vol. 8. – No. 2. – pp. 217-229.

[4] Zulfa I., Wanda R. Ranchangan information system of Academician Berbasis on the Menggunakan PHP and MySQL website //KLIK: Kajian Ilmiah Informatika and Computer. – 2023. – Vol. 3. – No. 4. – pp. 393-399. Garbuzov A. PostgreSQL //Scientific and technical conference for students, undergraduates and doctoral students. - 2023. – vol. 1. – pp. 360-363.

[5] Yunal H. T. et al. Postgresql Database Management System: ODAK //The 2023 Conference "Innovations in Intelligent Systems and Applications" (ASIU). – IEEE, 2023. – pp. 1-5.

[6] Sheik B., Chemuduru D. K. Dynamic SQL //Procedural programming using PostgreSQL PL/pgSQL: Development of complex database-oriented applications using PL/pgSQL. Berkeley, California : Apress, 2023. pp. 169-181.

[7] What is the difference between MySQL and PostgreSQL? [Electronic resource] Access mode: https://aws.amazon.com/ru/compare/the-difference-between-mysql-vs-postgresql / (accessed 07.09.2024).

[8] Using MySQL and PostgreSQL in 2023. [Electronic resource] Access mode: https://habr.com/ru/companies/otus/articles/722304 / (accessed 07.09.2024).

[9] The difference between MySQL and PostgreSQL. [Electronic resource] Access mode: https://www.geeksforgeeks.org/difference-between-mysql-and-postgresql / (accessed 07.09.2024).

[10] What are the best ways to optimize your database performance and storage? [Electronic resource] Access mode: https://www.linkedin.com/advice/0/what-best-ways-optimize-your-database-performance (accessed 07.09.2024).

[11] Ramu V. B. Database performance optimization: strategies for efficient query execution and resource usage //International Journal of Computer Trends and Technologies. – 2023. – vol. 71. – No. 7. – pp. 15-21.