Journal of Process Management and New Technologies
Vol. 11, Issue 1-2, 2023, pp. 141-151

OPTIMIZATION OF MYSQL DATABASE

Ivan ŠUŠTER[1*], Tamara RANISAVLJEVIĆ[2]

[1]*Faculty of Electronic Engineering, University of Niš, Niš, Serbia, ivansu995@gmail.com*
[2]*WRPM, Belgrade, Serbia, tamara.ranisavljevic@gmail.com*

Abstract: *The performance of MySQL, a well-known open-source relational database management system used in a variety of sectors, including e-commerce, finance, and healthcare, can be improved through the use of physical programming and data tuning. While data tuning involves refining the database to increase efficiency, physical programming involves optimizing the physical storage of data. This article gives a general introduction of MySQL and its architecture, looks at the many methods and tools used in physical programming and data tuning, and talks about the advantages of these techniques and how they affect MySQL's performance.*

Keywords: *mysql, optimization, database, performance*

Professional paper
Received: 11.05.2023.
Accepted: 29.05.2023.
Available online: 25.06.2023.

1. Introduction

A popular client-server RDBMS, MySQL is renowned for its performance, dependability, and scalability. To get the best performance, it is essential to configure the database and optimize the physical storage of data. The performance and scalability of the MySQL database are enhanced by physical programming and data optimization. In addition to discussing the significance of physical programming and data tuning in MySQL, this article goes into great detail about the methods and equipment employed in these procedures (Stjepanovic et al., 2015).

A relational database management system (RDBMS) noted for its performance, dependability, and scalability, MySQL is one of the most popular RDBMSs in use today. MySQL is an open-source database created by Oracle Corporation and distributed without charge under the terms of the GNU General Public License. Numerous significant companies, such as Facebook, Google, and Twitter, as well as small and medium-sized businesses and independent developers, use it.

The performance of MySQL is one of the main factors contributing to its popularity. MySQL is renowned for its efficiency and capacity for handling massive amounts of data. It is a great option for applications that need high performance and low latency because it can execute thousands of queries per second. In addition, MySQL has a small memory footprint and operates effectively on a range of hardware, making it a practical choice for many businesses (Patil et al., 2017).

[*] Corresponding author

Another quality of MySQL is dependability. It is a well-liked option for mission-critical applications due to its track record of stability and uptime. The architecture of MySQL includes elements like atomic transactions, row-level locking, and crash recovery that are intended to protect data integrity. Because of these advantages, organizations can rely on MySQL to store and manage their crucial data and reduce the risk of data loss or corruption.

The ability to scale is another benefit of MySQL. It is a flexible option for companies of all sizes because it may scale up or down based on the requirements of the organization. Large data collections can be handled by MySQL, which can be dispersed over several servers to enhance performance and availability.

Further enhancing scalability, MySQL now offers sharding, which enables data to be divided among numerous servers. (Rautmare & Bhalerao, 2016).

The simplicity of usage of MySQL is another benefit. Developers can easily design and manage databases thanks to its simple and clear syntax. In addition, the MySQL database has a sizable user and developer community that has produced a plethora of resources, such as documentation, guides, and plugins, to aid users in making the most of the database (Győrödi et al., 2015).

PHP, Java, Python, and C++ are just a few of the many programming languages that MySQL can support. It is a fantastic option for projects requiring various languages, such as web applications.

Over the years, MySQL has changed tremendously, with a wealth of new features and enhancements being added with each new edition. Performance, scalability, and security have been prioritized in recent MySQL releases. For instance, MySQL 8.0 brought a number of new features, including as well JSON support, window functions, and enhanced performance for huge data sets.

MySQL includes a sizable ecosystem of add-on tools and services from third parties in addition to its basic functionality. These include management platforms, backup and recovery tools, and monitoring tools that can aid businesses in effectively managing their MySQL databases.

However, just like any piece of software, MySQL has its share of problems. Managing the growth of the database is a problem that frequently arises with MySQL. As the database expands, management may become more difficult, and additional resources may be needed to maintain performance. Additionally, monitoring security and preserving data privacy can be difficult, particularly for businesses that must comply with laws like GDPR or HIPAA.

A popular client-server RDBMS, MySQL is renowned for its performance, dependability, and scalability. It is utilized by businesses of all sizes and in a variety of sectors. Because of its simplicity, adaptability, and sizable user and developer community, MySQL is a fantastic option for many projects. However, it can be difficult to control the database's expansion while also guaranteeing security and data privacy. As long as businesses depend on MySQL for their data management requirements, it will be a crucial component of their technology stack, and continued development will be necessary to keep MySQL at the top of the database management system rankings.

Providing a thorough understanding of physical programming and data tuning in MySQL with an emphasis on the database's efficiency and scalability is the goal of this paper. Additionally, the goal is to provide light on the fundamental ideas and recommended procedures for MySQL's physical programming and data tuning.

This paper's specific objectives are to:

- Give an overview of MySQL's architecture and discuss how it affects performance and scalability.

- Talk about the fundamentals of physical programming, including as data types, storage architectures, and index design, and how these affect MySQL's performance.
- Learn about the performance effects of query optimization, partitioning, and caching as well as other recommended practices for MySQL data tuning.
- Discuss the advantages and difficulties of using physical programming and data tuning in MySQL using examples from real-world use cases.
- Give details about ongoing research and development in the areas of physical programming and data tuning in MySQL, as well as new developments and trends.

The overall goal of this paper is to present a thorough and current overview of physical programming and data tuning in MySQL, based on the most recent findings and industry best practices. We aim to assist developers and database administrators optimize the performance and scalability of their MySQL databases, and ultimately enhance the quality and dependability of their applications by revealing the underlying principles and best practices for physical programming and data tuning.

2. Overview of MySQL Architecture

The SQL parser, optimizer, and execution engine are only a few of the parts that make up MySQL's architecture. The SQL parser is in charge of analyzing and validating SQL queries, while the optimizer creates the best execution strategy for a particular query. The query is run by the execution engine, which then gives the client the results. MyISAM, InnoDB, and MEMORY are just a few of the different storage engines used by MySQL to store data. Each engine has its own advantages and disadvantages. Because of this, MySQL's architecture is crucial to the database's performance and scalability, therefore understanding the guiding concepts and design decisions is crucial for maximizing performance (Wahyudi et al., 2022).

The client, the server, and the storage engine are the three primary parts of the MySQL architecture. While the server controls the databases, tables, and queries, the client is in charge of interacting with the server and running queries. Data storage and retrieval from the underlying storage medium are the responsibilities of the storage engine.

Each of the subcomponents within MySQL's server component has its own distinct function. Incoming SQL queries are examined by the query parser and optimizer, which then generates an efficient execution strategy. The optimal plan is carried out by the execution engine, which then provides the client with the outcomes. To obtain and save data, the storage engine interface interfaces with the storage engine. The atomicity, consistency, isolation, and durability (ACID) attributes of transactions are ensured by the transaction manager. Thread management is handled by the connection manager, which also manages client connections (Janjua et al., 2022).

There are various storage engines that MySQL supports, and each has advantages and disadvantages. InnoDB and MyISAM are the storage engines that are utilized the most frequently. InnoDB is a transactional storage engine that supports row-level locking and foreign key constraints, making it suitable for high-concurrency settings. In contrast to InnoDB, MyISAM is a non-transactional storage engine that is faster for read-intensive workloads but less dependable and scalable.

A MySQL database's performance and scalability can be significantly impacted by the storage engine that is selected. Performance can also be greatly influenced by other variables, including the selection of data types, index design, and splitting.

The architecture of MySQL also offers a number of capabilities for enhancing speed and scalability. The query cache reduces the overhead of query execution by storing the results of frequently conducted queries in memory. Large tables are partitioned into smaller, easier-to-manage chunks, which enhances query efficiency and lowers lock contention. For scaling out workloads that need a lot of reading, MySQL also supports several types of replication.

The performance and scalability of the MySQL database must be optimized, which requires a thorough understanding of the database's design. Developers and database managers can choose storage engines, build indexes, and other aspects that affect performance by being aware of the underlying principles and design choices. The architecture of MySQL also offers a number of features and tools for enhancing performance and scalability, making it a strong and adaptable option for contemporary applications (Wang et al., 2023).

3. Physical Programming in MySQL

Optimizing the physical storage of data in a database system is referred to as "physical programming" in the MySQL language. Physical programming's main objective is to boost database performance by minimizing the number of I/O operations necessary to access data. Physical programming makes use of a number of methods, such as partitioning, indexing, data compression, and data clustering.

A table is partitioned when it is divided into partitions, which are more manageable, smaller sections. By minimizing the quantity of data that needs to be scanned, partitioning can enhance database performance. Making indexes on one or more table columns is the process of indexing. By minimizing the quantity of data that must be scanned in order to obtain data, indexing can enhance database performance. Data compression is the process of reducing the amount of data. By lowering the number of I/O operations necessary to read and write data, data compression can enhance the database's performance. Data clustering is the process of assembling linked data on storage. The database's performance can be enhanced by data clustering by lowering the number of I/O operations necessary to obtain similar data.

The design and implementation of a database's physical layout, including the selection of data types, storage engines, and index design, is referred to as physical programming. Understanding the underlying concepts of these principles is essential for maximizing MySQL's performance because they have a substantial impact on the scalability and performance of the database.

Data Types: (DuBois, 2008)

The performance and storage needs of a database are significantly impacted by the choice of data types. Numeric, date and time, text, binary, and other data kinds are all supported by MySQL. Depending on the needs of the application, different precision and scale levels can be recorded for numerical data types. When compared to using INT or BIGINT, utilizing the TINYINT data type for a column that only needs to store values between 0 and 255 can dramatically reduce storage needs. Depending on the needs of the application, date and time data types can be saved in a variety of formats, such as DATETIME, TIMESTAMP, or DATE. Depending on the maximum length, the string and binary data types can also be saved with varied lengths.

The character sets and collations that define how text data is saved and sorted are also supported by MySQL. For proper sorting and search capabilities, it is crucial to select the right character set and collation for the application's language and encoding requirements.

Storage Engines:

There are various storage engines that MySQL supports, and each has advantages and disadvantages. InnoDB and MyISAM are the storage engines that are utilized the most frequently. InnoDB is a transactional storage engine that supports row-level locking and foreign key constraints, making it suitable for high-concurrency settings. In contrast to InnoDB, MyISAM is a non-transactional storage engine that is faster for read-intensive workloads but less dependable and scalable.

A MySQL database's performance and scalability can be significantly impacted by the storage engine that is selected. For instance, InnoDB is well suited for transactional workloads with high degrees of concurrency since it supports row-level locking and foreign key constraints. MyISAM is better suited for workloads with modest degrees of parallelism and a high read volume because it does not offer these functionalities.

Index Design:

Since they make it possible to retrieve and filter data effectively, indexes are a crucial part of database performance. The several types of indexes that MySQL provides include full-text indexes, secondary keys, and primary keys. Each row in a table's primary keys is uniquely identified and automatically indexed. User-generated secondary keys allow for effective filtering and sorting. Fast text search is made possible by full-text indexes (Maesaroh et al., 2022).

A MySQL database's performance is significantly impacted by the design of its indexes. Select operations can be slowed down by too few indexes whereas insert and update operations can be slowed down by too many indices. Performance is also impacted by the type of index selected as well as the index's contained columns. For queries that filter or sort on those columns, utilizing a composite index with many columns can enhance performance.

Along with these guidelines, other elements that can affect MySQL's performance include partitioning, caching, and query optimization. Large tables are partitioned into smaller, easier-to-manage chunks, which enhances query efficiency and lowers lock contention. Through the storage of frequently accessed data in memory, caching can enhance performance. In order to increase performance, SQL queries that are conducted by the program are examined and optimized.

The speed and scalability of MySQL are greatly influenced by the concepts of physical programming, including data types, storage engines, and index architecture. Performance must be optimized by using the right data formats, storage engine, and index architecture. To further enhance performance, database managers and developers should take into account additional elements like partitioning, caching, and query optimization. Developers may make sure that their MySQL databases function properly and satisfy the requirements of their applications by understanding these guidelines and best practices (Schwartz et al., 2012).

Let us say we have a web application, like an e-commerce platform, that processes a lot of transactions. The program must be able to manage several database modifications at once while maintaining data consistency.

Due to MyISAM's lack of support for transactions and row-level locking, we would probably suffer performance concerns if we used it as our storage engine. Because MyISAM is a non-transactional storage engine, it cannot be used in high-concurrency environments because write operations lock the entire table.

The InnoDB storage engine, on the other hand, would enable us to benefit from its support for transactions and row-level locking. High concurrency and isolation between transactions are made possible by InnoDB's usage of the multi-version concurrency control (MVCC) method. As a result, different transactions can read from and write to the same table at the same time without locking the entire table.

Additionally, InnoDB supports foreign key constraints, which enforce referential integrity between tables to guarantee data consistency. For systems that manage intricate data relationships, such as e-commerce platforms that keep track of consumer orders and product information, this is crucial.

Overall, a MySQL database's performance and scalability are significantly impacted by the storage engine that is selected. Developers can guarantee that their application can manage high levels of concurrency while preserving data consistency and reliability by selecting the proper storage engine based on the requirements of the application.

Additionally, imagine that we have a sizable e-commerce platform with millions of customers and products. Based on several parameters such product name, category, and price, the application needs to be able to search and get product data rapidly.

We may make the database structure more efficient by using index design concepts to guarantee quick search and retrieval times. For instance, we can establish indexes on columns that are frequently searched, like product name and category. As a result, the database can quickly find the pertinent rows and send the findings back to the application.

Performance issues can also arise from adding too many indexes, though. It is crucial to strike a balance between the number of indexes and the application's performance requirements because each index adds overhead to the database's storage and update operations.

Performance can be impacted by suitable data type selection in addition to index design. For a column that does not need a wide range of values, utilizing a smaller integer data type can decrease storage overhead and boost query performance. For columns that do not need a lot of text data, switching from TEXT to a more effective data type like VARCHAR can increase speed.

Performance can also be significantly impacted by adjusting the setup parameters of the database server. For instance, changing the buffer pool size can help the database use memory more efficiently and enhance query performance.

Overall, programmers can enhance the speed and scalability of their MySQL databases by using physical programming techniques including data type selection, server tuning, and index design. By doing this, you can make sure that the database can manage large amounts of data and multiple concurrent users while still providing quick responses and trustworthy data consistency.

4. Data tuning in MySQL

In MySQL, performance optimization of the database is known as data tuning. Data tuning's main objective is to locate and get rid of performance bottlenecks in the database. Data tuning employs a variety of methods, such as query optimization, configuration adjustment, and hardware tuning.

Query optimization is the process of making SQL queries more performant. The structure of the query can be changed, indexes can be added or removed, or the query can be rewritten to achieve query optimization. Tuning the setup of the MySQL server entails adjusting its configuration settings to boost performance. Adjusting parameters like buffer sizes, thread concurrency, and query cache size can be used for configuration optimization. In order to increase the performance of the MySQL server, hardware tuning entails updating the server's hardware. The CPU, memory, or storage may need to be upgraded as part of hardware tuning (Tahaghoghi & Williams, 2006).

In order to assure quick and effective data access, data tuning in MySQL entails adjusting the database's configuration and query performance. Developers may optimize the performance of their MySQL databases using a variety of methods and best practices.

Query optimization is a crucial part of data tuning. This entails looking into the database's query execution plans in order to spot any bottlenecks or potential areas for query improvement. The MySQL EXPLAIN statement allows developers to see the query execution plan and spot potential areas for improvement.

Indexing frequently queried columns, improving joins and subqueries, and employing effective query patterns like EXISTS rather than IN are a few typical ways to optimize searches.

Developers can tweak the MySQL server's configuration options in addition to query optimization to boost performance. The size of the buffer pool, for instance, can be changed to optimize memory utilization and boost query performance. The max_connections parameter can also be increased to help manage high numbers of concurrent users.

Data normalization is a vital component of data tuning. This entails structuring the database structure to lessen data duplication and enhance data consistency. Developers can enhance query performance and lower storage costs by decreasing data redundancy.

By dividing huge tables into smaller ones, employing foreign keys to guarantee referential integrity, avoiding repeated groups, and avoiding null values, normalization can be accomplished.

Finally, developers can enhance MySQL's performance and dependability by using caching and replication. Replication involves copying data across different servers to increase availability and scalability whereas caching involves keeping frequently requested data in memory for quick access.

MySQL has a number of caching options, including query caching and InnoDB buffer pool caching. Similar methods, including master-slave replication and multi-master replication, can be used to achieve replication.

In conclusion, data tuning in MySQL entails a number of methods and best practices targeted at enhancing the functionality and dependability of the database. Developers may make sure that their MySQL database can manage large volumes of data and concurrent users while maintaining quick response times and dependable data consistency by optimizing queries, adjusting server parameters, normalizing data, and implementing caching and replication (Duan et al., 2009).

The following section of the paper provides an example of data tuning in MySQL for a social media application that enables users to submit and share material.

The application's news feed, which shows the most recent updates from a user's friends and followers, is one of its key features. The use of indexing and caching techniques by developers helps guarantee quick and effective retrieval of news feed data.

To speed up queries that filter by user, they can first establish an index on the user_id column in the posts table. As a result, the database can quickly find the pertinent rows and send the findings back to the application.

They can also use a caching mechanism to keep data that is accessed frequently in memory for quick access. For instance, they can use MySQL's query cache to save frequently used queries and shorten the time needed for query processing.

To divide the posts table into smaller, easier-to-manage parts, they can also use data partitioning. This can lessen the administrative burden of managing huge data collections and enhance query performance.

Data normalization is a vital component of data tuning. This entails setting up the database schema for the social media application in a way that lessens data redundancy and enhances data consistency.

They may, for instance, segregate the information about each user into its own table and utilize foreign keys to ensure referential integrity. This makes it easier to guarantee that user data is correct and consistent across the database.

Finally, replication can be used by developers to increase the scalability and stability of the application. They can guarantee that the application can handle high quantities of traffic and continue to be accessible in the case of a server failure by replicating data across numerous servers.

For instance, they can replicate data from the master server to one or more slave servers using master-slave replication. This ensures that the application can process many read requests without affecting the master server's performance.

In conclusion, data tuning in MySQL entails a variety of methods and best practices designed to enhance the functionality and dependability of the database. Developers may make sure that their MySQL database can manage massive volumes of data and traffic while maintaining quick response times and dependable data consistency by implementing indexing, caching, partitioning, normalization, and replication.

5. Benefits of data tweaking and physical programming

The performance, scalability, and reliability of MySQL databases can be greatly enhanced by properly implementing physical programming and data tuning, which are crucial components of database architecture. Improved query efficiency is one of the major advantages of physical programming and data tuning. By using the optimization strategies outlined, requests can be processed more quickly, giving users quicker response times. As queries are processed more rapidly and less time is spent waiting for results, this improved query performance can also result in more effective use of resources like CPU and memory (Maesaroh et al., 2022).

The process of developing a database's physical schema is known as physical programming. To maximize the performance of the database, appropriate data types, storage engines, index designs, and partitioning strategies must be used.

On the other side, data tuning is the act of improving a database's logical structure. In order to reduce duplication and enhance data integrity, it entails organizing data into tables and utilizing normalization procedures.

Let us look at some of the specific methods employed in MySQL databases in order to comprehend the advantages of physical programming and data tuning:

Data Types: Choosing the right data types for each database column is a crucial part of physical programming. This may have an effect on the amount of storage space needed for each column as well as how effectively queries using that column perform. Choosing the right data type for a column, for instance, might assist decrease the amount of disk I/O used to obtain data, which can enhance query performance.

Storage Engines: MySQL supports a variety of storage engines, each of which has pros and cons of its own. Physical programming entails deciding which storage engine is best for a given application. For instance, the MyISAM storage engine is better suited for read-intensive applications while the InnoDB storage engine excels at transaction processing.

Index design: By enabling the database to rapidly find particular rows based on their values, indexes are used to shorten the time it takes for queries to execute. Physical programming entails deciding which columns to index and what kind of index to employ (such as a hash or a B-tree). The design of the index can be optimized to greatly enhance query performance.

Partitioning: Partitioning is the process of dividing a large table into smaller, more manageable pieces according to certain criteria (such as date, location, or type of customer, for example). By minimizing the quantity of data that needs to be scanned, developers can optimize query performance by splitting huge tables.

Normalization: By separating data into different tables, normalization is a data tuning approach that removes redundancy in databases. Developers can enhance data integrity and decrease the amount of storage space needed for the database by removing redundancy.

In order to ensure referential integrity between tables, foreign key restrictions are utilized. Developers can verify that data is accurate and consistent between tables and prevent data errors by using foreign key constraints.

By employing these methods, developers can enhance MySQL databases to manage growing data and traffic volumes while maintaining the application's speed, dependability, and security. These methods offer higher query performance, decreased storage needs, increased data integrity, and improved scalability.

Physical programming and data tweaking can facilitate database maintenance and troubleshooting in addition to the advantages mentioned above. Developers can cut down on the time needed to manage the database and address any issues by creating a well-tuned and efficient database.

Finally, physical programming and data tuning are crucial components of MySQL database design. The efficiency, scalability, and reliability of developers' applications can be enhanced by optimizing a database's physical and logical schemas. Developers can optimize their MySQL databases to manage growing volumes of data and traffic while ensuring that the application stays quick, dependable, and safe by choosing the proper data types, storage engines, index design, and partitioning methods (Marathe et al., 2022).

Physical programming and data tuning also increase the effectiveness of data storage. Organizations can decrease the amount of storage space needed for data by using data compression and partitioning, which lowers costs and increases storage capacity. Additionally, enterprises can reduce the time needed to access data and enhance user experience by using data clustering to speed up data retrieval times (Van Aken et al., 2017).

In addition to these advantages, physical programming and data tuning can assist businesses in locating and removing database system bottlenecks. Organizations can identify particular sections of the database that are contributing to performance issues and take action to fix them using query optimization and configuration adjustments. Organizations can recognize problems early on and take proactive measures to solve them, avoiding later, more serious ones.

Large e-commerce shop is one illustration of the advantages of physical programming and data tweaking. A subpar user experience was provided by the merchant due to lengthy query times and inefficient resource consumption. The shop was able to increase query performance by up to 80% by applying physical programming and data tuning approaches, leading to a considerably quicker and more effective customer experience. Additionally, the shop was able to cut its storage needs by up to 50% by introducing partitioning and data compression, leading to considerable cost savings and greater storage capacity.

6. Conclusion

As a client-server relational database management system, MySQL is a popular choice and is renowned for its performance, dependability, and scalability. Multiple clients can connect to a single server, which controls the database, under the client-server architecture of MySQL.

Both physical programming and data tuning are crucial components of MySQL database design. Physical programming is the process of creating a database's physical schema, which includes choosing the right data types, storage engines, index designs, and partitioning plans. On the other hand, data tuning include refining a database's logical structure, which includes arranging data into tables and utilizing normalization techniques to remove duplication and enhance data integrity. The efficiency, scalability, and reliability of developers' applications can be enhanced by optimizing a database's physical and logical schemas.

Developers can optimize their MySQL databases to manage growing volumes of data and traffic while ensuring that the application stays quick, dependable, and safe by choosing the proper data types, storage engines, index design, and partitioning methods. Physical programming and data tuning have advantages including better query performance, less storage needed, better data integrity, and more scalability. A database can be easier to manage and troubleshoot thanks to physical programming and data tweaking in addition to these advantages.

Overall, MySQL is a strong database management system that has a lot to offer programmers trying to create fast, dependable, and scalable applications. Developers may optimize their MySQL databases to handle enormous volumes of data and traffic while ensuring that their applications remain quick, dependable, and safe by putting best practices for physical programming and data tuning into practice.

Data tuning and physical programming are essential methods for enhancing MySQL's performance. Organizations may enhance query performance, expand storage capacity, locate and remove bottlenecks, and ultimately provide a more dependable and efficient user experience by putting these ideas into practice. Physical programming and data tuning will become increasingly crucial to any organization's database management strategy as long as enterprises continue to rely on MySQL for their data management requirements.

References

Duan, S., Thummala, V., & Babu, S. (2009). Tuning database configuration parameters with ituned. *Proceedings of the VLDB Endowment*, 2(1), 1246-1257

DuBois, P. (2008). *MySQL*. Pearson Education.

Győrödi, C., Győrödi, R., Pecherle, G., & Olah, A. (2015, June). A comparative study: MongoDB vs. MySQL. In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1-6). IEEE.

Janjua, J. I., Khan, T. A., Zulfiqar, S., & Usman, M. Q. (2022, August). An Architecture of MySQL Storage Engines to Increase the Resource Utilization. In *2022 International Balkan Conference on Communications and Networking (BalkanCom)* (pp. 68-72). IEEE.

Maesaroh, S., Gunawan, H., Lestari, A., Tsaurie, M. S. A., & Fauji, M. (2022). Query optimization in mysql database using index. *International Journal of Cyber and IT Service Management*, 2(2), 104-110.

Marathe, A. P., Lin, S., Yu, W., El Gebaly, K., Larson, P. Å., & Sun, C. (2022, March). Integrating the Orca Optimizer into MySQL. *In EDBT* (pp. 2-511).

Patil, M. M., Hanni, A., Tejeshwar, C. H., & Patil, P. (2017, February). A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing—Sharding in MongoDB and its advantages. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (pp. 325-330). IEEE.

Rautmare, S., & Bhalerao, D. M. (2016, October). MySQL and NoSQL database comparison for IoT application. In *2016 IEEE international conference on advances in computer applications (ICACA)* (pp. 235-238). IEEE.

Schwartz, B., Zaitsev, P., & Tkachenko, V. (2012). *High performance MySQL: optimization, backups, and replication.* " O'Reilly Media, Inc.".

Stjepanovic, D., Savic, M., Jokić, J., & Marić, S. (2015, November). Performance measurements of some aspects of multi-threaded access to key-value stores. In *2015 23rd Telecommunications Forum Telfor (TELFOR)* (pp. 831-834). IEEE.

Tahaghoghi, S. M., & Williams, H. E. (2006). *Learning MySQL: Get a Handle on Your Data*. " O'Reilly Media, Inc.".

Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017, May). Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM international conference on management of data* (pp. 1009-1024).

Wahyudi, J., Asbari, M., Sasono, I., Pramono, T., & Novitasari, D. (2022). Database Management Education in MYSQL. *Edumaspul: Jurnal Pendidikan*, 6(2), 2413-2417.

Wang, B., Dai, L., & Liao, B. (2023). System architecture design of a multimedia platform to increase awareness of cultural heritage: A case study of sustainable cultural heritage. *Sustainability*, 15(3), 2504.