Journal of Modern Technology
Vol. 02, No. 02, July 2025, pp. 312-316.
DOI: https://doi.org/10.71426/jmt.v2.i2.pp312-316 , E-ISSN: 3049-3102
Homepage: https://review.journal-of-moden-technology.com/index.php/jmt/index

Optimizing Data Retrieval and Update Operations in PHP-MySQL Applications: A Review

Olta Zagraxha[1*], Brikena Morina[2], Egzona Haskuka[3]

[1*]Department of Computing and Information Technologies, Rochester Institute of Technology, Prishtina, Kosovo, Email: oz3702@g.rit.ed , ORCID: https://orcid.org/0009-0000-7633-0493
[2]Department of Computing and Information Technologies, Rochester Institute of Technology, Prishtina, Kosovo, Email: bm3927@g.rit.edu , ORCID: https://orcid.org/0009-0008-7732-9409
[3]Department of Computing and Information Technologies, Rochester Institute of Technology, Prishtina, Kosovo, Email: eh6882@g.rit.edu , ORCID: https://orcid.org/0009-0004-6467-3033

Article Info

Article history:

Received: June 9, 2025
Revised: July 8, 2025
Accepted: July 9, 2025
First Online: July 10, 2025

Keywords:

Structured Query Language (SQL)
Hypertext Preprocessor (PHP)
MySQL Performance
Query optimization
Artificial Intelligence (AI)

ABSTRACT

One of the most employed technology stacks for developing interactive, database-driven web applications is PHP with MySQL. As systems grow larger, the need for efficient data retrieval and updating becomes more crucial to ensure optimal performance, scalability, and user satisfaction. This study discusses several approaches to deal with the main issues related to PHP and MySQL integration performance. The detailed analysis concentrated on prepared statements, batch and asynchronous operations, normalization, persistent connections, query optimization, indexing, and caching, and the use of sophisticated frameworks. Examples from real-world situations show how the methods can enhance performance. To help direct future development efforts, additional conceptual frameworks are also offered, along with best practices and possible hazards. This study discusses several approaches to deal with the main issues related to PHP and MySQL integration performance. The detailed analysis concentrated on prepared statements, batch and asynchronous operations, normalization, persistent connections, query optimization, indexing and caching, and the use of sophisticated frameworks.

Corresponding Author:

Email address of corresponding author: oz3702@g.rit.ed (Olta Zagraxha)

1. INTRODUCTION

Hypertext Preprocessor (PHP) and MySQL have long been the most popular frameworks for building dynamic web applications. Rasmus Lerdorf's server-scripting language, PHP, was first released in 1994 and quickly became well-known because it significantly improved several crucial web development tasks. In the mid-1990s, MySQL AB released MySQL and quickly gained popularity as a reliable relational database management system. The developers of MySQL were particularly fond of its speed and reliability in handling structured data, as outlined in the 8.0 reference manual [1]. Both novices and experts can understand PHP's complexities due to MySQL's built-in support and the abundance of community resources and assistance. The vast scalability of the entire infrastructure is demonstrated by the numerous significant platforms that were developed with PHP and MySQL, including Facebook, WordPress, Slack, and Wikipedia. With the increasing complexity of web applications, the demand for

convenient access to a suitable database environment has become essential. Applications that use a non-optimized data fetch and update method will run slower, stop responding, and will be expensive to maintain. This study aims to identify fundamental performance problems in PHP- MySQL applications and analyze how these issues can be resolved promptly [2]-[3].

2. OVERVIEW OF PHP AND MYSQL

Originally developed as a set of basic tools for personal home pages, PHP has evolved into a versatile programming language used on over 20 million websites worldwide ("history of PHP"). Despite its role as a bridge between different database systems, PHP's compatibility with MySQL made it a natural choice for creating web applications [4]. PHP provides reliable and secure database access through its PDO (PHP Data Objects) and MySQL extensions. The MySQL speed, dependability, and support for common SQL functions made it a popular choice for relational databases.

It enabled developers to efficiently manage and present complex relational data. Since MySQL is open source, a global community has contributed to its substantial advancement [15]. PHP and MySQL were the foundation of the software stack (Linux, apache, MySQL, and PHP) that was extensively used for creating online applications for almost 20 years. Building dynamic, database-driven websites was made simple for developers by the smooth integration of their tools. However, the demands placed on these systems have increased significantly. Modern applications put additional demands on backend systems because they need to process large amounts of data, be instantly responsive, and be customizable in real time [5]. Developers need to adopt systematic optimization methods that align with the current performance benchmarks.

3. COMMON PERFORMANCE ISSUES

While PHP and MySQL can function effectively together, if the system is not optimized correctly, it can lead to several common performance problems. The use of inefficient SQL queries has a significant impact on a system's efficiency. Developers commonly use broad queries, like select, to retrieve all of the fields in a table, even though they are not required [6]-[7]. This approach increases the size of the result set, delays server processing times, and consumes more bandwidth. Such inefficiencies are especially detrimental in complex applications where even minor inefficiencies can lead to notable performance improvements. Additionally, by lengthening execution times and taxing the database server, badly constructed join statements and insufficient data filtering techniques make these issues worse.

3. 1 Database queries that are not essential

Noteworthy database queries are a common problem. When applications repeatedly retrieve the same data from the database instead of storing it locally, it places a substantial burden on the database. In congested traffic scenarios, even small inefficiencies can significantly affect servers, resulting in bottlenecks and slower response times.

3. 2 Delay of data transmission

It is very costly to establish a new connection and then break it after each request. The impact may be negligible for small-scale applications, but it becomes more substantial as the user base expands. The burden of this responsibility can be significantly alleviated by fostering long-lasting relationships and employing sophisticated connection pooling techniques [8].

3. 3 Inadequate oversights of financial transactions

In systems where several tasks are occurring simultaneously, if the transaction control is not executed accurately, the applications may have inconsistent and incomplete updates. Transactions are crucial to ensure that operations are carried out correctly and to maintain the reliability and accuracy of data [9].

4. OPTIMIZATION TECHNIQUES

To optimize the PHP-MySQL interface, a comprehensive and multi-layered approach is necessary.

4. 1 Altering the performance of SQL queries

Asking focused, focused questions is vital. Developers need to meticulously design join procedures, prevent n+1 query issues (where a query is repeated in a loop), and select only the necessary columns for the task ("SQL optimization"). Profiling tools like MySQL explain are essential for rewriting queries to pinpoint bottlenecks. By employing query decomposition, which involves breaking down a large query into smaller, more manageable components, efficiency can be greatly enhanced [10].

4. 2 Data organization and database structure improvement

When using columns in joins, sorts, and searches, it is crucial to position their indexes in a way that optimizes performance. Composite indexes can be employed to enhance queries that involve multiple columns. Developers need to understand covering indexes, which allow the index to provide a complete response to a query without needing to access the entire table. Furthermore, implementing database normalization techniques during the design phase minimizes redundancies and ensures data accuracy. Nevertheless, selective denormalization, which includes additional data redundancy to expedite read operations, may prove beneficial in high-traffic systems.

4. 3 Caching layers

Likewise, enforcing database normalization ways during the design phase minimizes redundancies and ensures data delicacy. Nonetheless, picky denormalization, which includes fresh data redundancy to expedite read operations, may prove salutary in high- business systems.

5. CASE STUDY: SCALING AN E-COMMERCE PLATFORM

One such is working on a mid-sized e-commerce platform with practically a million actual products. Since all product categories started depending on specific searches without any filters, paging a lot during periods of high shopping traffic took more than a minute. Customers regularly leave their shopping carts empty as a result of their poor experiences, which cost the company a significant amount of money. Query refinement was the first stage of optimization, where unnecessary columns were removed, and additional conditions were added to restrict the data that could be retrieved. The indexes have been generated on the item's name, grouping id, and price sections to speed up the sorting and filters process [11]-12].

The network eventually adopted relationship collaboration and ongoing relationships among data bases to reduce the strain of constantly creating fresh connections [13]-[14]. The import in batches procedure was simplified to handle modifications to numerous product entries without overtaxing the database. The platform's database processing time dropped by 79% after completing these three steps. A website's loading times dropped from an ordinary of 6.2 seconds to a less than 1.5 seconds. Within times of high demand, the network's CPU load dropped by 42% and the cart abandonment rate dropped by more than 30%. Over six months, customer satisfaction ratings rose in tandem with a 15% increase in revenue. This case study demonstrates how fundamental optimization techniques can have a significant real-word scenario.

5. 1 Best practices

To guarantee long-term efficiency and scalability, developers must consistently follow many best practices throughout the entire lifecycle of the application.

To initiate the development process, programmers should create modular and manageable code using modern frameworks like Laravel, symfony, or yii2. Frameworks often incorporate caching mechanisms, object layers, and query optimization tools that simplify many best practices. Second, it is vital to incorporate profiling tools into the development process. Developers can employ blackfire. Io, MySQL's explain plan, PHP's xdebug, and the new relic to pinpoint areas for improvement and obtain a comprehensive view of performance problems. Thirdly, there are several methods for storing, such as query cookies, server-side object caches, http storing (with varnish), and browser caches for static pages. Fifth, it's critical to monitor important performance metrics like server CPU utilization, consumption of memory, database load, and question execution time. Monitoring applications like Grafana, Prometheus, and Datadog are used to automatically track these observations. Finally, the development team needs to foster a performance-conscious culture. Grafana, Prometheus, and Datadog are some of the tools that help automate the monitoring process. It is vital that the development team cultivate a culture of performance consciousness. This encompasses the incorporation of performance testing into the continuous integration/continuous delivery pipelines, a focus on enhancing code efficiency through reviews, and ongoing training for the team to stay informed about the latest technologies and best practices.

5. 2 Projected advancements in PHP-MySQL efficiency

The optimum setting for PHP-MySQL applications is constantly changing and improving because of developments in technology. Major progress has been made in immediately gathering in PHP 8.3 and later versions, which improve CPU-bound processes and increase the competitiveness of PHP lessons in fields that were previously ruled by smaller-scale language learning like C++ and Go. Oracle's in the cloud MySQL. The heat wave service combines transactional (OLTP) and analytical (OLAP) features into a single MySQL database instance. This significantly streamlines application design and expedites intricate analyses by eliminating the need to duplicate and synchronize data across multiple platforms. Serverless databases, like AWS's aurora serverless, are becoming more popular and widely used. These solutions eliminate the need for developers to manually manage database server capacity, as they automatically adjust to meet the demands of the system. In serverless systems, the focus is on the application logic and business requirements rather than managing the infrastructure. GraphQL is being utilized as a substitute for conventional REST APIs, resulting in the creation of optimization techniques. GraphQL resolves the problem of excessive data retrieval that can arise with REST-based APIs by allowing clients to specify the exact information they require. Incorporating GraphQL into PHP-MySQL stacks requires the utilization of supplementary tools, but the resulting productivity enhancements are substantial when dealing with extensive and complex datasets. Finally, the database engines are integrating machine learning technologies. It is expected that future versions of MariaDB and MySQL will incorporate artificial intelligence (AI) models, enabling real-time performance tuning, intelligent query plan selection, and predictive indexing, thereby offering even greater opportunities for enhancement without the necessity of human involvement. To create robust and scalable PHP-MySQL applications for the future, developers need to stay updated on the latest advancements in this domain.

6. CONCLUSION

By enhancing the data getting and providing processes of PHP-MySQL programs it is so important that we have efficiency, security and fast answers from the website. This is not merely a scientific finding. This is more than just a scientific discovery. From carefully crafting SQL queries to implementing caching and batch processing, developers have a plethora of optimization techniques available to them. It is crucial to comprehend that optimization is an ongoing process and should not be viewed as a one-time task. the expansion of groups of users, the creation of programs, and the changing needs. In the highly competitive field of contemporary web development, businesses that place a high priority on thorough profiling, frequent updates, and tried-and-true best practices have a better chance of success. Best practices, which include serverless database servers, combined OLAP-OLTP systems, AI performance tuning, and shifting frameworks, are expected to change as new technologies are introduced. Developers who remain proactive and flexible will set the standard, creating apps that can cater to the needs of millions of users globally. In the future, PHP-MySQL applications will transform from vulnerable, outdated systems into powerful, scalable engines of innovation through careful, balanced, and forward-thinking optimization.

DECLARATIONS

Conflict of Interest: The authors declare that there is no conflict of interest.
Funding: This research received no external funding.
Availability of data and materials: No data is available in this article.
Publisher's note: The Journal and Publisher remain neutral about jurisdictional claims in published maps and institutional affiliations.
Acknowledgments
The authors would like to formally acknowledge *Dr. Debabrata Samanta*, CIT Program Head & Assistant Professor, Rochester Institute of Technology, Kosovo, Email: dsamanta@auk.org for his exceptional guidance.

REFERENCES

[1] Sheng L, Fang S, Xiaoyu W, Haijiao Y, Shibao H, Xiang Y, Baiyu Y. Design and Implementation of Power Tax Data Query Tool Based on Private Information Retrieval. In2024 IEEE 4th International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA) 2024 Dec 6 (Vol. 4, pp. 352-356). IEEE. 10.1109/ICIBA62489.2024.10868610

[2] Kyrychenko I, Tereshchenko G, Smelyakov K. Optimized Indexing Method in a Hybrid Image Storage Model for Efficient Storage and Access in Big Data Environments. In2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET) 2024 Oct 8 (pp. 1-4). IEEE. 10.1109/TCSET64720.2024.10755763

[3] Li S, Liu W, Wu Y, Zhao J. Generative Architecture for Data Imputation in Secure Blockchain-enabled Spatiotemporal Data Management. Journal of Web Engineering. 2024 Jan;23(1):111-63. 10.13052/jwe1540-9589.2315

[4] Wu X, Gong Q, Chen J, Liu Q, Podhorszki N, Liang X, Klasky S. Error-controlled Progressive Retrieval of Scientific Data under Derivable Quantities of Interest. InSC24: International Conference for High Performance Computing, Networking, Storage and Analysis 2024 Nov 17 (pp. 1-16). IEEE. 10.1109/SC41406.2024.00092

[5] Keskin M, Yalman AA, Teper E, Keçeci S, Rençberoğlu E. Enhancing Fine-Grained Image Retrieval with Advanced Embedding and Loss Functions. In2025 7th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (ICHORA) 2025 May 23 (pp. 1-4). IEEE. 10.1109/ICHORA65333.2025.11017113

[6] Mengmeng S, Zhibin L, Qingwei W, Man H, Feiyang X. An Effective Retrieval Method to Improve RAG Performance. In2024 7th International Conference on Data Science and Information Technology (DSIT) 2024 Dec 20 (pp. 1-5). IEEE. 10.1109/DSIT61374.2024.10881380

[7] Soni P, Islam SH, Pal AK, Mishra N, Samanta D. Blockchain-based user authentication and data-sharing framework for healthcare industries. IEEE Transactions on Network Science and Engineering. 2024 Mar 26. 10.1109/TNSE.2024.3381723

[8] Aarab A, Oussous A, Saddoune M. Optimizing Arabic Information Retrieval: A Comprehensive Evaluation of Preprocessing Techniques. In2024 IEEE 12th International Symposium on Signal, Image, Video and Communications (ISIVC) 2024 May 21 (pp. 1-4). IEEE. 10.1109/ISIVC61350.2024.10577827

[9] Thomas D, Cinis J, Lea C, Buzzard M. PHP MySQL Website Programming: Problem-Design-Solution. Apress; 2008. https://link.springer.com/book/9781590591505

[10] Powers D. Connecting to MySQL with PHP and SQL. InPHP Solutions: Dynamic Web Design Made Easy 2010 (pp. 303-334). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4302-3250-6_11

[11] Bramer M, Bramer M. Passing Variables to a PHP Script I. Web Programming with PHP and MySQL: A Practical Guide. 2015:127-51. https://doi.org/10.1007/978-3-319-22659-0_10

[12] Hajdini A, Fazliu L, Gjoshi D. Comparative Study of Join Algorithms in MySQL: Cross, Inner, Outer, and Self Joins. Journal of Computing and Data Technology. 2025 Jun. 9 ;1(1):1-9. https://review.journal-of-modern-technology.com/index.PHP/jcdt/article/view/57

[13] Soma A kumar. Hybrid RNN-GRU-LSTM Model for Accurate Detection of DDoS Attacks on IDS Dataset. Journal of Modern Technology. 2025 May 14;2(01):283-91. https://doi.org/10.71426/jmt.v2.i1.pp283-291

[14] SOMA, Arun Kumar. Building Aether Sensor Network using LoRaWAN Network. In 2025 International Conference on Emerging Systems and Intelligent Computing (ESIC). IEEE, 2025. p. 807-814. 10.1109/ESIC64052.2025.10962750

[15] [Online Available]: https://dev.mysql.com/doc/refman/8.4/en/ , Accessed on 10 June, 2025.