ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 6 Issue 1 February 2026
Paper Id: JETA-V6I1P111 / Page No: 89-98

Original Article

A Survey of MySQL Database Administration Techniques and Best Practices

Hari Babu Dama
Senior Database Application Architect, Randstad Digital LLC (Bank of America), Plano, Texas, USA.

Received Date: 15 February 2026 Revised Date: 23 February 2026 Accepted Date: 28 February 2026

Abstract: *MySQL is an open-source relational database management system that is most popular and recognized by its reliability, flexibility, performance, as well as high security properties. With growing data-driven application use in the healthcare sectors, among others, such as finance, e-commerce, and cloud computing, MySQL database management becomes more of a best practice in terms of system upkeep, information quality, and efficiency. This paper provides an in-depth overview of MySQL database administration methods and best practices including its architecture, storage, transaction management, security, performance optimization, backup and recovery mechanisms, and high-availability solutions. The major issues related to the management of large-scale, distributed, and cloud-based MySQL environments, such as scalability, data security, and a combination with analytical workloads, are addressed. The paper focuses on the significance of monitoring, indexing, query optimization, replication, and automated fault-tolerance solutions to ensure the high performance and reliability. This survey offers viable recommendations to database administrators, system architects and organizations that aim to design secure, scale and high-performance MySQL database systems to support contemporary and high-volume data-intensive applications.*

Keywords: *MySQL, Database Administration, Database Security, Storage Engines, Transaction Management.*

I. INTRODUCTION

Database management systems form the backbone of modern enterprise applications, web and data-driven applications, as it provides functionality to store, retrieve and manage structured information in a very efficient manner. MySQL is one of the open-source relational database management systems with the greatest growth rates due to its versatility, reliable, highly performing, and well secured [1]. MySQL finds wide applications in different industries such as in healthcare sector, banking, e-commerce, and retail industries where the ability to handle data and system availability is vital to the continuity of business operations and decision making.

As the volume of digital transformation efforts and the complexity of data environments continue to increase rapidly, the administration of MySQL databases is a crucial need of an organization [2]. In modern MySQL deployments, the traditional on-premise deployment has been refined to include cloud-based and hybrid architectures, posing new administrative problems in the areas of scalability [3], [4], availability, performance optimization and security [5]. DBAs therefore need to develop holistic approaches in managing such environments and at the same time maintain integrity of the data and the reliability of the system. MySQL database administration has a vital component of security.

One of MySQL's primary features that is relevant to data security is its robust user authorization mechanism. This method ensures that only authorized personnel may access sensitive data by allowing database administrators to define specific access privileges for different users [6]. MySQL database administration has a vital component of security. Such mechanisms can assist the organizations to comply with the regulatory requirements and minimize the risk of data breach.

Recovery and backup plans are also very instrumental in ensuring MySQL database reliability[7][8]. MySQL has been configured to react to both the logical and physical methods of backup, which gives the administrators the option to select both methods depending on the size of the system, the performance needs, and the recovery goals [9]. Logical backups, which are often done with the aid of programs like mysqldump, provide some flexibility in that they allow the choice of specific databases or tables to be backed up and come in handy especially in a large or distributed setup. Although faster in the recovery of a complete system, physical backups can result in increased resource overheads when deployed at large scale.

Moreover, good password management and access control policies are also considered important aspects of MySQL security administration. Storing passwords securely, conducting regular privilege audits, and following the least-privilege principles can help reduce the security risks and system integrity [10]. Together, these administrative methods are the basis of creating secure, scalable, and high-performance MySQL database systems that make MySQL a reliable application in the current data-driven applications.

A. Structured of the paper

This paper is structured as follows: Section II presents the fundamental components and architectural framework of MySQL. Section III discusses data administrative techniques of MySQL. Section IV examines key challenges in MySQL database administration. Section V provides an overview of recent studies on MySQL techniques, the research conclusion and future directions are presented in Section VI.

II. CONCEPT OF MYSQL DATABASE

The design of the MySQL server makes it unique among database servers and incredibly flexible. In complicated environment, like web applications, MySQL is a dependable and adaptable database. The architecture of the relational database MYSQL is tiered. The database system's architecture is composed of three layers. An architecture consists of three layers: a client-end or query execution end, a storage engine, and a server resource end (see to Figure. 1).

The layers of the MySQL architecture used in the query execution are given below:

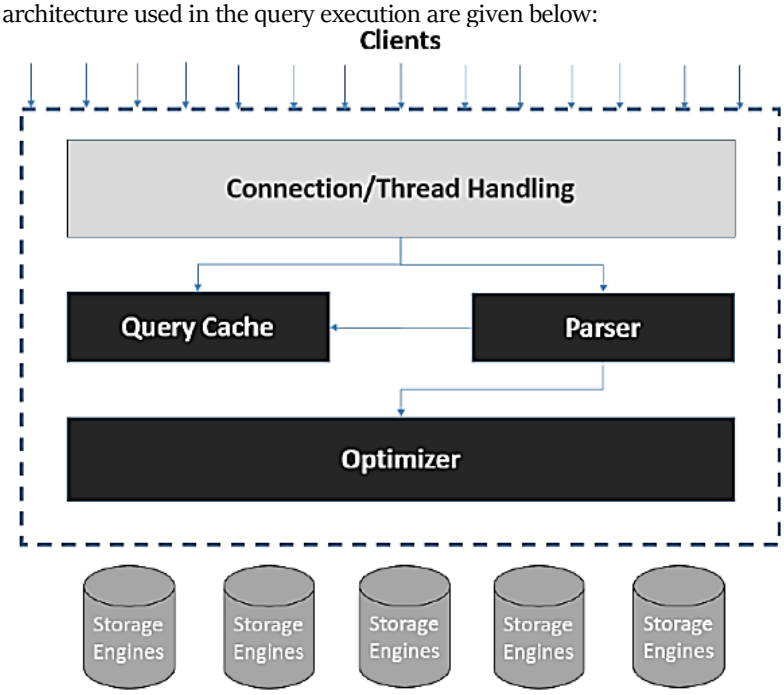

Figure 1 : Mysql Architecture with Core Components [11].

- Client: The client is where all MySQL-accessing applications reside, as is the access server where data is stored and retrieved. ODBC or JDBC are two examples of utility drivers that may be used to link server thread managers.
- Connection Management: Each client connection receives a single thread on the server, and the client's query is processed within that connection. Cache threads are also controlled by the server to save connection management costs [12]. Each client connection request is granted access by the server using host-based authentication, username, and password.
- Optimization/Execution: MySQL parses queries to produce a parse tree, which is comparable to an execution plan, as part of query execution. It then optimizes it by rewriting the query busing variety of ways using indexes and caching. The server's optimizer interacts with storage engines to create the most effective execution plan since the storage engine is impacted by how the server optimizes user requests for optimal performance.
- Metadata Cache: In MySQL, the query and key cache is the primary application for the cache, which plays an essential role in performance. When a client submits a similar query, the server simply retrieves the execution plan from the query cache rather than parsing and running the same question again. In contrast, the key cache is used to store table indexes, which storage engines then employ.
- Storage Engine: The MySQL server's storage engine, which manages the data on the physical discs, is a crucial component. It uses discs to carry out write and read operations on the physical data files on disks. For practically all MySQL needs, InnoDB is one of the most widely used and Oracle-recommended default storage engines.

A. Storage Engines

Several methods are used in MySQL to save the data in files (or memory). By using several methods, users can increase the application's speed or functionality and enhance its overall performance [13]. In MySQL, these various methods and the corresponding supporting functions are referred to as the storage engine (or table type), as seen in Fig. 2.

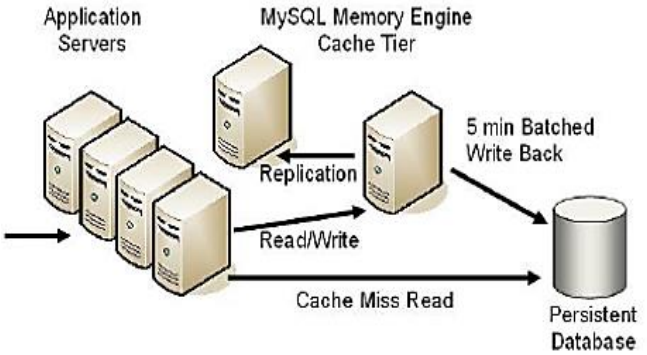

Figure 2 : The Working Principle of Memory Engine.

MySQL originally offered MyISAM, which is the default storage engine in MySQL 5.1. Transactions and foreign keys are not supported by MyISAM. Its quick access speed is its superiority. This engine is mostly used to construct tables in applications that are SELECT & INSERT driven or do not require transactional integrity. In order to leverage this engine, the majority of websites are built on inquiry. Each MyISAM table is kept on the disc as three files.

InnoDB offers MySQL transaction-safe tables with multi-versioned concurrency management features, as well as transaction, rollback, and crash recovery capabilities. Multi-user and deployment capabilities are enhanced by these characteristics. InnoDB's lock function doesn't need to be expanded because row-level locking takes up very little space. SQL queries allow users to flexibly blend InnoDB and other kinds of tables, even inside the same query [14], [15]. InnoDB was created with the intention of optimizing performance while managing massive amounts of data. Out of all the disk-based relational databases, it boasts the best CPU utilization.

B. Transaction Management and Concurrency Control

A sequence of read or write activities on a database makes up a transaction. A certain unit of work is completed by the read or write operation. A transaction is a set of read/write activities that only succeeds if every operation within it does as well. When a transaction starts to execute, it can be stopped in one of two ways: Abort or Commit. A transaction is said to be committed when it is carried out successfully and the associated changes are saved, and it is said to be aborted when it does not progress to a successful end or is not finished. The database system keeps the following transaction attributes in order to guarantee data integrity:

- Atomicity: Either all or none of the transaction's activities are accurately represented in the database. Atomicity must be preserved even when there are deadlocks, disc failures, CPU failures, database software failures, and application software failures. Transactions may not always execute properly, and in order to guarantee software atomicity, a failed transaction must not impact the database's state. Cloud recovery techniques must be implemented in order to handle cancelled transactions and reverse any such actions [16], [17].
- Consistency: Database consistency is maintained when a transaction is executed in isolation, meaning that no other transactions are running concurrently. Cloud data handling is still challenging; as data volume and variety increase, pure replication is no longer a practical solution, which causes problems with consistency and efficiency [18], [19].
- Isolation: The system ensures that, for each pair of transactions Ti and Tj, Ti perceive that either Tj completed execution before Ti began, or Tj began execution after Ti completed, even if numerous transactions may run concurrently. As a result, no transaction is aware of other concurrently running transactions in the system.
- Durability: Even in the event of system failures, the database changes made by a transaction remain in effect once it has successfully finished. It is regarded as one of the key characteristics for cloud systems' data center-type applications [20]. A transaction is considered committed when it is successfully completed. When a committed transaction performs updates, the database is changed into a new consistent state that needs to endure even in the event of a system failure.

III. DATABASE ADMINISTRATION TECHNIQUES

The management, optimization, and availability of MySQL Server databases are all tasks that database administrators (DBAs) must perform. Here are some MySQL database administration best practices:

- Database Monitoring and Performance Tuning: MySQL administration is based on good monitoring. MySQL Enterprise Monitor, Performance schema, slow query logs and MySQL Workbench can be used to review the performance. The amount of RAM, CPU, disc input/output, and active connections, and query execution time are some of the critical metrics that DBAs are supposed to monitor in order to find out the bottlenecks, execute the queries in the most efficient way, and enhance the indexing strategies.
- Backup and Recovery Strategies: To guarantee that no data is lost, MySQL databases should have dependable backup and recovery procedures. MySQL Enterprise Backup or a file system snapshot may be used for physical backups, while

mysqldump can be used for logical backups [21]. Tested backup procedures and off-site storage or cloud-based storage should be used on a regular basis to improve ability to recover in the event of a disaster [22].

- High Availability and Disaster Recovery: MySQL has several HA/DR system options, such as source-replica replication, Group Replication, Galera Cluster, and InnoDB Cluster that assist to reduce downtime and maintain uninterrupted availability. Selection of solution is based on system scalability, consistency and fault-tolerance needs.
- Security and Authentication: It is quite clear that the security and integrity of the data is an important aspect which must be preserved. MySQL administration is a very serious issue about security. Role-based access control, authentication plugs, encryption using the help of the SSL/TLS and password policies are used to secure sensitive data. The potential vulnerabilities are reduced through the use of the least privileged principle and auditing access to users as well as keeping up-to-date security patches.
- Indexing and Query Optimization: The use of query optimization and indexing plays a great role in improving the performance of MySQL. Execution plans are usually analyzed with the help of EXPLAIN and MySQL Workbench to identify queries that are inefficient [23]. Efficient index management ensures that read operation is enhanced and unnecessary indexing which can impact write operation is not done. To speed up query execution, create and maintain the proper indexes, but be careful not to over-index as this can affect write performance.

A. MySQL Server Installation and Configuration

Some configuration and installation techniques should be explained as follows:

- Prepare the System: The system is then updated so as to verify that all packages that are already in it are up to date [24]. Basic tools/libraries like wget, tar, gcc and libaio are installed to assist in downloading and extracting the MySQL binaries as well as running them.
- Download and Extract MySQL Binaries: The necessary MySQL binary distribution is downloaded with the help of wget command and the official MySQL site. The download URL is modified to the required MySQL release by changing the version numbers. A custom install directory /app/mysql is made, and the downloaded MySQL tarball is unzipped into this directory.
- Create a MySQL User and Group: A special system user and group with the name mysql is created to execute MySQL service. This enhances security since the MySQL processes not be isolated with other system users.
- MySQL Data Directory Setup: A directory of data is created within the MySQL base directory in order to store database files. The MySQL user is given proper ownership, and the database is brought up through the command mysqld -initialize.
- Configure MySQL: To set up the MySQL parameters, a custom MySQL configuration file (my.cnf) is generated to define such crucial parameters as a base directory, data directory, port number, socket file, error log, and process ID file.
- Set Up the MySQL Service: A service file that is created under systemd is configured to provide management of the MySQL server as a system service. This enables MySQL to be launched, shut as well as tracked using typical systemctl commands.
- Secure MySQL: The mysql_secure set up script is run to enhance safety by defining a root password, uninstalling test databases, turning off remote root logins, and getting rid of anonymous users.
- Verify Installation: The installation is checked through the command-line client login on the MySQL server. A successful login confirm that MySQL is installed and operational.

B. Replication and High Availability Techniques

MySQL replication involves transferring data from one MySQL database server (the primary or source) to one or more servers (the replicas or secondaries) in near real-time [25]. It provides redundancy of data, load balancing and fault tolerance. High availability (minimal downtime in case of failure) and scalability (dealing with larger workloads) are impossible without replication. As shown in Fig. 3.

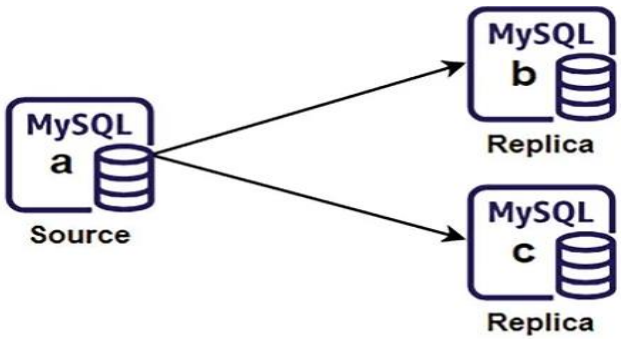

Figure 3 : MySQL Source–Replica Replication Model.

In MySQL, "high availability" (HA) is the attribute of a system to remain operational, without incurring significant impact and downtime caused by hardware failures, network issues, or application errors [26]. HA is essential due to the fact that even a small outage can cost a company revenue, reduced production, and dissatisfied customers, so companies that rely on MySQL databases as critical applications cannot afford to go down.

The following are some of the High Availability (HA) fundamentals:

a) *High Availability (HA):*
- Data Redundancy: Replicas ensure that the data in the primary is replicated so that an eventuality of the primary failure can be handled by the replica.
- Failover Capability: MySQL routers such as MySQL Router or MHA also be used to automate the switch to a replica and reduce downtime.
- Geographic Redundancy: There are replicas in other regions keeping off west-wide failures.

b) *Scalability:*
- Read Scaling: Read requests (e.g. SELECTs) are being processed in replicas which offload the primary and extend read capacity.
- Geographic Scaling: Replicas in other areas minimize the latency experienced by the international users.
- Write Scaling (Limited): Advanced setups like Group Replication or sharding distribute writes.

MySQL high availability (HA) plans can play a significant role to minimize the number of downtimes and ensure that the most important database systems remain accessible in case of Table I failures.

Table 1 : High Availability Techniques

Technique	Description	Advantages	Challenges
Source-Replica Replication	Replication of data asynchronously between one or more replica servers and a source server.	Straightforward setup, affordable, and easy to scale readings.	Possible data loss; no automatic failover without further equipment.
Source-Source Replication	Read and write activities can be handled by both nodes in a dual-source configuration.	Increased redundancy and better load balancing.	Resolving data conflicts can be difficult; careful setup is necessary.
MySQL Group Replication	Automatic failover and fault tolerance in a multi-writer clustering system.	Strong consistency.	More complex to set up.
Galera Cluster	Data consistency is guaranteed across all nodes with a synchronous clustering approach.	Ensures high data consistency.	Increased delay as a result of the synchronous nature

C. Security Management and Data Protection

MySQL has security controls that involve the use of Access Control Lists (ACLs) to access user operations and also has the ability to use an encrypted connection between the client and the server using the SSL protocol. Such security measures applicable to most applications and not specific to MySQL. One needs to understand access privilege management of MySQL and issues access commands like GRANT and REVOKE. Additionally, it is advised that the hashed password be stored in the database rather than the plain text in order to prevent data privacy violations in the case of a database attack [27]. Maintaining a safe system requires routinely examining and modifying user privileges using commands like SHOW GRANT to verify access and REVOKE to revoke access permissions.

Here are some applications given below to ensure the security of the database:

a) *Secure Application-Database Communication.*

Applications that connect to MySQL databases have to have strong security measures in order to safeguard confidential data. Extra care needed when the data is sent through the public or unsecured network, since unencrypted communication can make credentials and transactional data vulnerable to interception.

b) *Application of Encrypted Communication Protocols*

MySQL helps in making the data transmission encrypted, such as using SSL/TLS and SSH protocols, to ensure confidentiality and data integrity. These protocols encrypt the client-server traffic, thus avoiding unauthorized access, man-in-the-middle attacks, and data leakage.

c) Port Management: Network-Level Security

The MySQL services must be guarded through proper network setup. Firewalls and the access control rules should be used to restrict the default MySQL port (3306) in order to only allow connections by trusted IP addresses or application servers.

d) Vulnerability Assessment and Port Scanning

Port scanning tools like Nmap are very common in port scanning to establish open and available ports of a host system. Nmap assists database administrators to assess network exposure, identify misconfigurations and minimize attack surfaces through closing ports not needed.

e) Verification of Port Accessibility

MySQL recommends the use of regular checks on the availability of database port such as telnet and network diagnostic tools. This enable administrators to ensure that the MySQL service is accessible to only authorized systems and not vulnerable to any external threats.

IV. CHALLENGES IN MYSQL DATABASE ADMINISTRATION

The successful storage, retrieval, and manipulation of data, which are all traditional DBA roles, are critical to data science and machine learning. Traditional DBA approaches have significant challenges, despite the dynamism and unstructured form of data used in data science [28]. It identifies the key challenges and provides the solutions to the DBA integration in the data science processes:

- Data Volume and Variety: Some of the datasets that are commonly utilized in data science initiatives include structured, semi-structured, and unstructured data [29]. This variety can be over too much to be managed by conventional databases and hybrid solutions are required.
- Scalability: Scalable database infrastructures that are capable of relying on sudden spikes in data volume or processing needs are required since data science workloads are unpredictable.
- Performance Optimization: Ensuring that databases with complex, resource-intensive queries such as those used in machine learning, can support these queries without compromising performance.
- Data Security and Compliance: The safe management of sensitive data is essential, without violating the law especially when integrating data of many sources [30], [31].
- Integration with Analytical Tools: Integration databases and analytical systems including R, Python and a variety of machine learning systems can be easily integrated.

A. Security Threats and Misconfigurations

MySQL would provide security features including the capability of the use of SSL secured connections between clients and servers and the use of Access Control Lists (ACLs) in user actions. Most applications follow these safety guidelines and they are not application-specific to MySQL. It is important to ensure safe MySQL usage by restricting access to the user table in the system database to the MySQL root account [32].

To manage access it is important to understand the access privilege structure contained in MySQL and use such commands as GRANT and REVOKE. Additionally, it is advised to save the hashed password in the database rather than the plain text in order to prevent data privacy violations in the case of a database attack [33]. Maintaining a safe system requires routinely examining and modifying user privileges using commands like SHOW GRANT to verify access and REVOKE to revoke access permissions.

Applications that interact with MySQL should use the proper security measures to protect the database, which mostly pertains to sending unencrypted data over the Internet. In order to augment data security, one is also recommended to communicate through encrypted protocols such as the use of SSS and SSH.

B. Scalability Challenges in Large-Scale Systems

MySQL scalability of large-scale applications involves the systematic processing of the growing workloads via vertical and horizontal scaling, replicas, load balancing, clustering and distributed architecture to ensure work load performance and availability:

a) Comparison between Vertical and Horizontal Scaling Methods

The MySQL database can be scaled based on its ability to support increasing workloads. At this point, it may explore two predominant scaling approaches: vertical scaling (also referred to as scaling up) and horizontal scaling (also referred to as scaling out) [34]. It provided the knowledge that would be able to make an informed decision-making regarding the database scalability by examining the strengths and weaknesses of both techniques.

b) Different Approaches to Replication

A MySQL database should be built in a scalable and fault-tolerant manner; it requires the use of effective replication strategies. Some of the replication approaches that were discussed in this section include; master-slave replication, multi-source replication, and master-master replication. Having a good understanding of the functionality of all these different strategies exposes the potential benefits of each as well as their disadvantages.

c) Achieving Equitable Distribution of Workload

A balance in the distribution of workload is an essential factor in the context of managing the arrival of new database requests using a small number of servers effectively [35]. During this stage, they discuss some of the load-balancing methods, such as supplier IP affinity, spherical-robin and least-connections. The strategies are necessary to make sure that the database is capable of receiving extensive traffic as well as sustaining the greatest possible performance repertoires.

d) Enhancing Reliability through Clustering and High Availability Measures

To guarantee high availability in an effort to safeguard mission-critical applications in case of potential failures or outages. The given section is devoted to describing methods of clustering that can be beneficial to enhancing redundancy and fault tolerance of a database system, including Galera Cluster and Percona XtraDB Cluster.

e) Harnessing Distributed Databases for Global Scalability

The rate of growing data volumes is increasing exponentially and there is the globalization of applications; Distributed databases are now a basic tool that fulfills the requirements of present times.

V. LITERATURE REVIEW

The literature review points to the MySQL usage in web applications, which are scaled to focus on performance optimization via query tuning, indexing, caching, automation, and proactive administration. The recent studies are compared in Table 2.

Agarwal (2025) provides an extensive overview of optimized forms of database sharding that are specific to MySQL applications. It discusses the basic Sharding concepts, comparisons of various models, and advanced optimization techniques, i.e., consistent hashing, query-aware routing, dynamically re-sharding, and caching. The paper offer a roadmap to help the architects and engineers create scalable, reliable, and high-performance MySQL infrastructures by combining the existing best practices with scholarly work. The results highlight that sharding used and tuned properly can be of major benefit to a database in terms of system scalability, performance, and operational effectiveness [36].

Gupta et al. (2024) An open relational database management system called MySQL was developed using object-oriented programming to create and manage data. While data processing focuses on optimizing data for performance, organizational programming optimize physical storage of data. It provides an overview of MySQL and its architecture, explores the various methods and tools used to manage and organize databases, and discusses the benefits of these techniques and how they impact MySQL performance. Reactive systems, such as Auto Admin, focus on historical workloads, while proactive systems, like QB5000 and DQM, predict future workloads for fully automated optimization [37].

Rahmany et al. (2024) idea of MySQL security comes from database security and mostly refers to attacks that take advantage of flaws in database systems. Many contemporary database systems have broad security flaws, including SQL injection, inference attacks, passive assaults, active attacks, and other database-side attacks. Hackers utilise such techniques to access, alter, abuse, create, or remove data from relational databases in businesses using the application or backend layers. This article examines and discusses several methods for defending MySQL against these attacks [38].

Šušter and Ranisavljević (2023) Performance of a popular open-source relational database management system, MySQL, used in a variety of industries, including e-commerce, finance, healthcare, and so forth can be enhanced through physical programming and data tuning. Data tuning is the practice of enhancing the database to improve its efficiency whereas physical programming is the optimization of data which is in storage. This paper provides a general overview of MySQL and its architecture, examines the numerous techniques and tools of physical programming and data tuning and discusses the benefits of the techniques and their impact on the performance of MySQL [39].

Gao et al. (2023) MySQL database-based application system has progressively expanded into a number of domains, and the MYSQL database's performance has drawn increasing interest. This study examines MySQL database speed optimisation from five perspectives: server-side optimisation, table structure optimisation, schema optimisation, index optimisation, and SQL statement optimisation [40].

Wahyudi et al. (2022) purpose of this research is to learn more about MySQL, also known as My Sequel, an open-source database management system also known as a database management system. The smallest and largest databases and their contents may be created and managed with MySQL, which can also be used to communicate with users. Because of its database

structure, MySQL is also a part of the Relational Database Management System, or RDBMS, which acts as a bridge between the database server and the program when the data retrieval procedure employs the relational database approach [10].

Table 2 : Literature Review on MySQL Optimization, Administration, and Security

Authors	Focus Area	Key Findings	Approaches	Limitations	Future Work
Agarwal (2025)	MySQL Database Sharding & Scalability	Properly designed and tuned sharding significantly improves MySQL performance, operational efficiency, and system scalability	Consistent hashing, query-aware routing, dynamic re-sharding, caching mechanisms	Limited discussion on administrative complexity, monitoring, and security challenges in sharded MySQL environments	Need for comprehensive frameworks addressing administration, security, and automated monitoring for sharded MySQL systems
Gupta et al. (2024)	MySQL Architecture & Performance Optimization	Reactive and proactive optimization systems enhance MySQL performance by managing workloads and physical storage	AutoAdmin (reactive), QB5000, DQM (proactive), architectural analysis	Focuses primarily on performance optimization without integrating security and scalability considerations	Future research should integrate automated performance tuning with security-aware and scalable MySQL administration
Hamidi, Hamraz & Rahmany (2024)	MySQL Security & Attack Mitigation	SQL injection, inference, and active/passive attacks remain major threats; protection mechanisms enhance database security	Encryption, backups, access control, table locking, firewall techniques	Lacks discussion on automation and real-time threat detection in large-scale MySQL deployments	Development of automated and proactive MySQL security monitoring and threat detection systems
Šušter & Ranisavljević (2023)	Physical Programming & Data Tuning	Physical storage optimization and data tuning significantly improve MySQL performance	Physical programming techniques, data tuning, architectural analysis	Limited consideration of distributed, cloud-based, and high-availability MySQL environments	Exploration of physical tuning strategies in cloud-native and highly available MySQL systems
Gao et al. (2023)	MySQL Performance Optimization Techniques	SQL, index, schema, table structure, and server optimization improve MySQL performance	SQL optimization, indexing, schema and table refinement, server-side tuning	Concentrates on tuning techniques with minimal focus on automation and security	Integration of performance tuning with automated administration and security mechanisms
Wahyudi et al. (2022)	MySQL Fundamentals & DBMS Concepts	MySQL is an effective open-source RDBMS for managing databases of varying sizes	Conceptual explanation of DBMS and RDBMS principles	Provides foundational knowledge only, without advanced administration or optimization insights	Need for studies addressing advanced MySQL administration, optimization, and security practices

VI. CONCLUSION AND FUTURE WORK

Modern computing environments are characterized by MySQL database management that covers a wide spectrum of concepts, administration methods, challenges, and best practices required to maintain applications that are data-intensive. MySQL is a stable platform of enterprise, web-based and cloud-driven systems as demonstrated by its modular architecture, multiple storage engines, and strong transaction management, and full security features. Also significant in the attainment of system reliability, scalability as well as data protection are proper techniques of database administration, including constant monitoring of performance, query and index optimization, backup and recovery planning, replication schemes, high-availability configurations and so on. Despite such advantages, there are several common challenges facing database administrators such as how to manage the ever-increasing amounts of data, high rates of security and compliance with regulations, support analytical and ML loads, and operate distributed and highly available databases. Moreover, the evolving deployment architectures such as cloud and hybrid infrastructures make deployment configuration, monitoring and fault management more complex. The direction of future research and development is intensive AI-based administration structures which will be capable of automatically adjusting the performance, detecting anomalies, anticipating faults and reducing security threats. There should be more focus on the cloud-native MySQL architectures, containerized deployment, and complete integration with data analytics and machine learning platforms as well. Such innovations will contribute to the enhancement of the operational efficiency and minimization of administrative overhead, and the role of MySQL as an essential database solution for next-generation data-driven and enterprise-scale applications.

VII. REFERENCES

[1] V. Prajapati, “Cloud-Based Database Management: Architecture, Security, challenges and solutions,” J. Glob. Res. Electron. Commun., vol. 01, no. 1, pp. 07–13, 2025.

[2] G. Maddali, “Enhancing Database Architectures with Artificial Intelligence (AI),” Int. J. Sci. Res. Sci. Technol., vol. 12, no. 3, pp. 296–308, May 2025, doi: 10.32628/IJSRST2512331.

[3] S. Narang and V. G. Kolla, “Next-Generation Cloud Security: A Review of the Constraints and Strategies in Serverless Computing,” Int. J. Res. Anal. Rev., vol. 12, no. 3, pp. 1–7, 2025, doi: 10.56975/ijrar.v12i3.319048.

[4] V. Shah, “Managing Security and Privacy in Cloud Frameworks: A Risk with Compliance Perspective for Enterprises,” Int. J. Curr. Eng. Technol., vol. 12, no. 06, pp. 1–13, 2022, doi: 10.14741/ijcet/v.12.6.16.

[5] R. Tandon and D. Patel, “Evolution of Microservices Patterns for Designing HyperScalable Cloud-Native Architectures,” ESP J. Eng. Technol. Adv., vol. 1, no. 1, pp. 288–297, 2021, doi: 10.56472/25832646/JETA-V1I1P131.

[6] D. Patel, “Leveraging Database Technologies for Efficient Data Modeling and Storage in Web Applications,” Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 10, no. 4, pp. 357–369, Jul. 2024, doi: 10.32628/CSEIT25113374.

[7] V. Verma, “Optimizing Database Performance for Big Data Analytics and Business Intelligence,” Int. J. Eng. Sci. Math., vol. 13, no. 11, pp. 56–75, 2024.

[8] S. K. Chintagunta, “Enhancing Cloud Database Security Through Intelligent Threat Detection and Risk Mitigation,” TIJER – Int. Res. J., vol. 9, no. 10, pp. 49–55, 2022.

[9] Q. Zhou and J. Zhang, “Research on the Design and Implementation of Case Teaching in MySQL Database Course,” Educ. Reform Dev., vol. 7, pp. 145–150, 2025, doi: 10.26689/erd.v7i5.10588.

[10] J. Wahyudi, M. Asbari, I. Sasono, T. Pramono, and D. Novitasari, “Database management in MySQL,” Edumaspul-J. Pendidik, vol. 6, no. 2, pp. 2413–2417, 2022.

[11] D. Gorasiya, “Quantitative Performance Evaluation of Cloud-Based MySQL (Relational) Vs. MongoDB (NoSQL) Database with YCSB.” National College of Ireland, 2019. doi: 10.13140/RG.2.2.29628.59528.

[12] M. G. M. Nyabuto, V. Mony, and S. Mbugua, “Architectural review of client-server models,” Int. J. Sci. Res. Eng. trends, vol. 10, no. 1, pp. 139–143, 2024.

[13] X. Pan, W. Wu, and Y. H. Gu, “Study and Optimization Based on MySQL Storage Engine,” Adv. Intell. Soft Comput., vol. 129, pp. 185–189, 2011, doi: 10.1007/978-3-642-25986-9_28.

[14] A. R. Bilipelli, “Visual Intelligence Framework for Business Analytics Using SQL Server and Dashboards,” ESP J. Eng. Technol. Adv., vol. 3, no. 3, pp. 144–153, 2023, doi: 10.56472/25832646/JETA-V3I7P118.

[15] S. Amrale, “Proactive Resource Utilization Prediction for Scalable Cloud Systems with Machine Learning,” Int. J. Res. Anal. Rev., vol. 10, no. 4, pp. 758–764, 2023.

[16] N. K. Prajapati, “Cloud-based serverless architectures: Trends, challenges and opportunities for modern applications,” World J. Adv. Eng. Technol. Sci., vol. 16, no. 1, pp. 427–435, Jul. 2025, doi: 10.30574/wjaets.2025.16.1.1225.

[17] P. Chandrashekar, “Advancements in Automated Incident Management: A Survey within Cloud-Native SRE (Site Reliability Engineering) Practices,” Int. J. Curr. Eng. Technol., vol. 13, no. 6, pp. 1–9, 2023, doi: 10.14741/ijcet/v.13.6.13.

[18] P. Guleria, “ACID Support and Fault-Tolerant Database Systems on Cloud : A Review,” vol. 1, no. 8, pp. 2011–2013, 2012.

[19] S. Srinivasan, R. Sundaram, K. Narukulla, S. Thangavel, and S. B. Venkata Naga, “Cloud-Native Microservices Architectures: Performance, Security, and Cost Optimization Strategies,” Int. J. Emerg. Trends Comput. Sci. Inf. Technol., vol. 4, no. 1, pp. 16–24, 2023, doi: 10.63282/3050-9246.ijetcsit-v4i1p103.

[20] M. Menghnani, “Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications,” Int. J. Innov. Sci. Res. Technol., vol. 10, no. 2, 2025, doi: 10.5281/zenodo.14959407.

[21] N. Fathima Nifra and S. Razeeth, “Database backup and recovery: a review with test implementation for MYSQL and NOSQL databases,” 2022.

[22] S. B. Karri, C. M. Penugonda, S. Karanam, M. Tajammul, S. Rayankula, and P. Vankadara, "Enhancing Cloud-Native Applications : A Comparative Study of Java-To-Go Micro Services Migration," Int. Trans. Electr. Eng. Comput. Sci., vol. 4, no. 1, pp. 1–12, 2025.
[23] I. C. Saidu, M. Yusuf, F. C. Nemariyi, and A. C. George, "Indexing techniques and structured queries for relational databases management systems," J. Niger. Soc. Phys. Sci., p. 2155, 2024.
[24] T. Valentine, "Installing and using the MySQL database server," in Database-Driven Web Development: Learn to Operate at a Professional Level with PERL and MySQL, Springer, 2023, pp. 129–143.
[25] I. Suster, D. Karabaševic, A. Sijan, D. Pucar, L. Ilic, and G. Jocić, "Database Replication: Ensuring MySQL Data Integrity," UAESTUS Multidiscip. Res. J., vol. 24, pp. 281–293, 2024.
[26] B. Zhao and Z. Wen, "MySQL High Availability Analysis and Practice," in Proceedings of the 2024 Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence, 2024, pp. 544–551.
[27] Y. R. Akhondzadeh, "Data Protection \& Security in MS-SQL Server DBMS," Int. J. Innov. Comput. Sci. Inf. Technol, vol. 4, no. 1, pp. 7–13, 2021.
[28] S. S. S. Neeli, "Key Challenges and Strategies in Managing Databases for Data Science and Machine Learning," Int. J. Lead. Res. Publ., vol. 2, no. 3, 2021.
[29] A. Kohnke, "Managing Security Across Disparate Database Technologies," ISACA J., vol. 4, 2022.
[30] Y. Macha, "Cloud-Based CRM for Healthcare Data Management: A Review of HIPAA Compliance and Validation Rules," TIJER - Int. Res. J., vol. 10, no. 11, pp. 1–6, 2023.
[31] A. P. and S. Pandya, "Compliance-Driven Data Governance: A Survey on GDPR, and HIPAA in Cloud Databases," Int. J. Curr. Eng. Technol., vol. 12, no. 6, pp. 828–836, 2022, doi: 10.14741/ijcet/v.12.6.18.
[32] A. Hamidi, A. R. Hamraz, and K. Rahmani, "Database security mechanisms in MySQL," Afghanistan Res. J., vol. 4, no. 1, 2022.
[33] G. Wessler Boneti, "Identify threat model and protection offered by different encryption possibilities in MySQL, PostgreSQL and Oracle," 2023.
[34] A. Zulkifli, "'Accelerating database efficiency in complex it infrastructures: Advanced techniques for optimizing performance, scalability, and data management in distributed systems," Int. J. Inf. Cybersecurity, vol. 7, no. 12, pp. 81–100, 2023.
[35] N. Thakur and A. Aditya, "International Journal of Research Publication and Reviews," SSRN Electron. J., no. 5, pp. 7631–7638, 2024, doi: 10.2139/ssrn.4991412.
[36] R. Agarwal, "Optimized Database Sharding Techniques for High-Performance MySQL Applications," Int. J. Comput. Exp. Sci. Eng., vol. 11, no. 4, 2025, doi: 10.22399/ijcesen.4340.
[37] M. Gupta, A. Mittal, S. Singh Bisht, and S. Arora, "Optimizing MySQL Performance: Essential Techniques, Resources, and Best Practices," in 2024 International Conference on Progressive Innovations in Intelligent Systems and Data Science (ICPIDS), 2024, pp. 484–489. doi: 10.1109/ICPIDS65698.2024.00081.
[38] A. Hamidi, A. Hamraz, and K. Rahmany, "Database Security Mechanisms in MySQL," Afghanistan Res. J. - Nat. Sci., vol. 4, 2024, doi: 10.70648/naturalscience.v4i1.66.
[39] I. Suster and T. Ranisavljevic, "Optimization of MySQL database," J. Process Manag. New Technol., vol. 11, no. 1–2, pp. 141–151, 2023, doi: 10.5937/jouproman2301141Q.
[40] P. Gao, Q. Chen, X. Xie, and C. Wang, "Research on Performance Optimization of MySQL Database," in 2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2023, pp. 869–872. doi: 10.1109/ICIBA56860.2023.10165291