IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

A Survey of Operational Challenges and Best Practices in MongoDB Database Administration

Dr. Hemant N. Patel
Assistant Professor, Computer Engineering
Sankalchand Patel College of Engineering, Sankalchand Patel University, Visnagar
hp15284@gmail.com

Abstract: *MongoDB is a popular NoSQL database system with a document-oriented data model, scalability across multiple nodes and large amounts of semi-structured or unstructured data. Distributed computing, cloud and big data applications are rapidly gaining popularity and without efficient database management, there will be no reliability, security and optimal performance. It covers every aspect of the administration of the MongoDB database including the very fundamental skills that any Database Administrator should master – configuring, monitoring, backups and recovery of a database, managing replication, sharding, and security. It also examines some of the critical operational issues such as optimization, scalability, data consistency in distributed environments, access control etc. Further optimization could be achieved by adopting best practices such as schema design, indexing and optimization techniques to make the system more efficient and more resourceful. Comparing the MongoDB deployment models can also provide insight into the differences between the models in terms of scalability, availability, fault tolerance and Administration. The results highlight the lack of advanced automation and intelligent management of MongoDB systems today, indicating the need for more advanced solutions that can increase efficiency in the modern data-driven, cloud-based application landscape.*

Keywords: MongoDB, NoSQL Database, Database Administration, Performance Optimization, Query Optimization

I. INTRODUCTION

A database is a structured collection of information stored and maintained electronically that can be accessed, retrieved, updated and processed efficiently. In today's information-rich world, database administration is a key component of the performance, security and reliability of the database. Common database administration tasks involve performance tuning, security administration, monitoring, troubleshooting, backups and recovery, and future capacity planning [1][2]. In addition, database systems will have to be implemented to meet the C.I.A. rules of Confidentiality, Integrity, and Availability, as well as other factors like Authenticity, Control, and Data utility [3].

In the age of Big Data, Cloud Computing and distributed applications, conventional RDBMS are sometimes not suitable for dealing with very dynamic and unstructured data [4][5]. To deal with this, non-Relational (NoSQL) databases have started to gain popularity, which offer a number of data types including document stores, key-value stores, column-family databases, and graph databases [6][7]. Unlike relational databases, NoSQL systems are more flexible and can process semi-structured or unstructured data from multiple sources, or large amounts of data, without relying on a fixed schema or table structure [8].

In the context of NoSQL databases, MongoDB is one of the most popular open source document-oriented databases, because of its flexibility, scalability, and high performance [9]. With flexible schema model that allows dynamic data structures, MongoDB stores information in the form of collections and documents. As opposed to the traditional relational databases, MongoDB can be horizontally scaled, distributed and stored across environments that are available in multiple locations, and process large amounts of semi-structured or unstructured data efficiently [10].

DOI: 10.48175/IJARSCT-36772

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, June 2026 Impact Factor: 8.2

MongoDB provides advanced features like indexing, replication, load balancing, and sharding to enhance performance and scalability in distributed computing systems. Replica sets provide high availability and fault tolerance by distributing data across multiple nodes, and sharding allows workload distribution across servers to effectively handle large datasets [11]. The features of MongoDB make it a suitable data store for modern applications that rely on real-time analytics, cloud resources, Internet of Things (IoT) and big data processing [12].

However, there are some drawbacks to administering MongoDB, such as performance tuning, indexing, replication, sharding, backup and recovery, security, and access control [13]. To operate distributed MongoDB environments effectively, DBAs should use effective operational strategies, such as building effective schemas and indexes, optimizing queries, monitoring systems, automated backup strategies, and access management. In this context, it is important to understand the operational aspects of the MongoDB database and to learn the best practices for ensuring the reliability, scalability, and security of data-driven applications.

A. Structure of the Paper

The following is the outline of the paper: Section II discusses MongoDB database administration and administrator responsibilities. Section III highlights operational challenges in MongoDB administration. Section IV presents best practices for MongoDB administration. Section V provides comparative analysis and literature review. Finally, the paper concludes with future research directions.

II. ROLE OF MONGODB DATABASE ADMINISTRATION

The administration of a MongoDB system involves managing, monitoring and maintaining the system to ensure the reliability, security and availability of the database [14]. With the advent of data-driven world, MongoDB Database Administrators (DBAs) are a crucial part of keeping the efficient working of the database system and its stability, especially in a distributed environment. They're in charge of database setup, database management, data backup, access control and disaster recovery. MongoDB DBA's ensure the availability and running of the database in distributed situations. The main roles of a MongoDB Administrator to effectively manage and maintain a database system are depicted in Figure 1.

A. Responsibilities of a MongoDB Administrator

In today's database-centric world where the MongoDB database is used to support performance, availability, scalability and security, the role of the MongoDB Administrator (DBA) is a critical one. The administrator has to perform key functions such as installing, configuring, monitoring, backing up and restoring the database, managing replication and configuration sharding. Ensuring the efficient execution of queries, handling indexes, and distributing resources effectively to achieve smooth data processing are also crucial responsibilities for MongoDB DBA tasks. Furthermore, the use of authentication, authorization and encryption in a Distributed Database System may ensure the reliability and protection of an organization's data [15].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

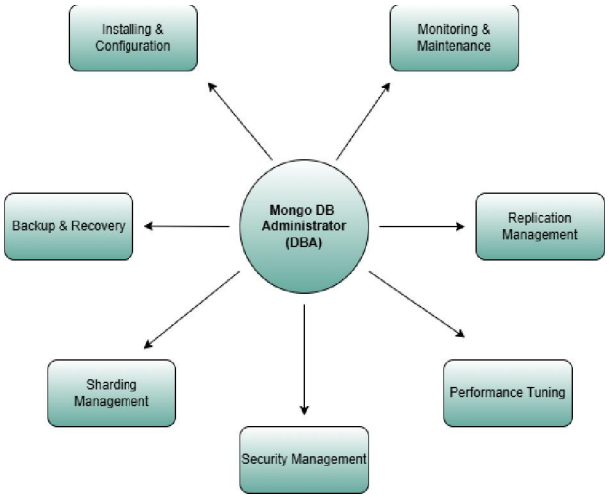

Fig. 1. Responsibilities of MongoDB Administrator

B. Database Configuration and Setup

High-performance, scalable and reliable operations of MongoDB systems depend on the right configuration and setup. The setup phase of the system consists of setting up the database, configuring the server, setting up storage and setting up the network for smooth system operation. There are several ways of deploying MongoDB: Standalone Servers, Replica Sets and Sharded Clusters. To provide fault tolerance and data availability, data can be replicated on other nodes using replica sets, and to handle a large amount of data and heavy workload, horizontal scalability can be achieved by using sharding. WiredTiger storage engine is a popular storage engine for MongoDB that has features of memory efficiency, concurrency control and compression.

The different parameters for storage, indexing, caching or query execution are important and can be configured as part of the database's configuration so that the overall performance of the database is optimal. In addition, security configurations are set during deployment to safeguard sensitive data from unauthorized access – for instance, authentication and authorization protocols. Properly configured can ensure the stability of operations, minimize downtime and optimize resource use.

C. Monitoring and Maintenance

Monitoring and maintenance of the database is one of the main causes that every MongoDB Administrator considers the availability, performance and reliability of the database very critical. Some important metrics to track for monitoring MongoDB are: Server Health, CPU Usage, Memory Usage, Disk Usage, Query Performance, Replication Status, and Networking Performance. Several tools can be used to keep track of the behavior of a database, such as MongoDB Compass, MongoDB Atlas and MongoDB Ops Manager.

Performance Optimization

There are two main goals of optimization with the mongoDB: improve query performance and maximize the use of resources. The primary methods for achieving high performance from MongoDB are indexing, optimizing queries, cache and sharding. Administrators also review query execution plans for any inefficiencies and optimize them for query response time as well. The techniques discussed above will help make MongoDB more effective in analyzing and handling high-volume transactional workloads [16].

Backup and Recovery Maintenance

Backup and recovery systems are an important piece in the puzzle of data security and system reliability. Database backups are periodically done by the admins, for example with mongodump and mongorestore, to avoid any loss of

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

data, and to recover from a disaster. Replication can also provide access to more data and quicker recovery in the event of a hardware failure or accidental loss or corruption of the data. These mechanisms ensure business continuity and help to reduce operational risk.

Security and System Maintenance

Security maintenance is an important point to consider when safeguarding a database system using techniques of authentication, authorization and encryption. This data and operations can only be accessed by authorized users thanks to role-based access restrictions. The system is patched and updated regularly to minimize vulnerabilities and system stability. These will ensure the database and its operation will be more secure and reliable.

III. OPERATIONAL CHALLENGES IN MONGODB ADMINISTRATION

The administration of MongoDB is the process of managing the database in the context of a dynamic computing environment, including its performance, security, and availability, as well as its maintenance. Although flexible and scalable, MongoDB has some operational impediments for the database administrator. The problems are: how to retrieve data (with a suitable authentication and access control), how to optimize the performance of data as the amount of data increases and becomes huge and how to build a solid backup and recovery system to avoid data loss. In addition, administrators are troubleshooting system problems, and handling replication and sharding for high availability and scalability. To maintain efficient, secure, and continuous MongoDB operations in modern applications, it is crucial to address these operational challenges.

A. Performance and Scalability Issues

The distributed nature of MongoDB, along with sharding, allows for high performance and horizontal scaling of the system for efficient handling of large data sets and high workloads. As illustrated in Figure 2, the advantages of sharding are that workload is distributed more effectively and that the database can be used more effectively at scale. However, scalability has issues like selecting the wrong shard key, rebalancing clusters, and administration complexity as well, which can impact performance.

The performance of MongoDB will also be boosted as the use of indexing and query optimization will help decrease the time taken for queries to be executed, thereby increasing the response time. These benefits are provided, but MongoDB runs into a performance consistency problem where a higher write concern and distributed transactions are more reliable and can be slower and impact the system performance. This brings the issue of maintaining consistency of data across distributed nodes important when it comes to MongoDB administration [16].

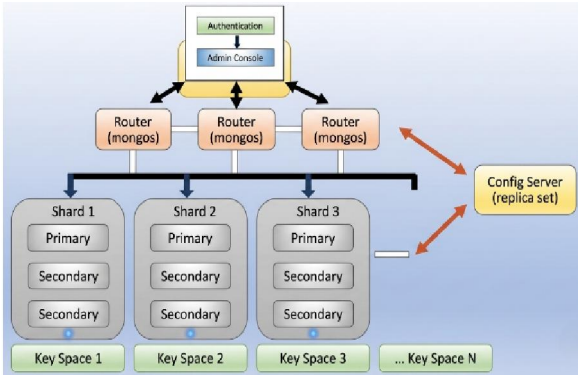

Fig. 2. MongoDB Sharding Architecture for Scalability and Workload Distribution

B. Data Security and Access Control

The key elements to ensure data security in MongoDB are robust authentication and access control practices. The primary way of protection in all database systems is by authentication. The robust authentication features of MongoDB

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, June 2026 Impact Factor: 8.2

ensure users' identities when accessing the database. Supplies interoperable, secure communication of user credentials, and might be performed by using a number of authentication mechanisms, including SCRAM-SHA-1 and SCRAM-SHA-256. MongoDB can also integrate with other authentication systems like LDAP or Kerberos, allowing for centralized user management and authentication.

One of the essential parts of security in MongoDB is access control. Role-based access control (RBAC) is one of the most important properties of RBAC: permissions can be granted to individuals based on their role and responsibility. This level of control makes sure that only authorized users are able to get access to specific databases and perform specific tasks [17].

C. Backup, Recovery, and High Availability

MongoDB's backup, availability and recovery are provided by replication and snapshot backup. A replicated MongoDB cluster has a single primary server and a number of secondary servers. For such clusters, the primary server takes the action for any write action before passing it to any secondary servers. This boosts reliability and availability. In a "backup" operation, there will generally be some form of "snapshotting", and "post-snapshotting" operations, in order to assure consistency of the nodes between snapshots, and to guarantee that some specific type of fail will be detected, such as journal corruption. In the recovery process, a system is able to resume normal operations easily due to the replicated servers and primary selection [18].

In the case of MongoDB, the advanced strategy to achieve HA and fault-tolerance is to use a cluster of replica sets of replica sets. In a replica set, primary and secondary nodes accept data asynchronously. It minimizes the risk of downtime in case of unexpected hardware or network failures and does so automatically. If the primary node becomes unavailable, one of the secondary nodes is promoted to primary as the primary; hence, the system continues processing transactions without breaking. Consequently, this replication process naturally fortifies scalability for read operations because they are spread across multiple nodes [19].

IV. BEST PRACTICES IN MONGODB DATABASE ADMINISTRATION

Best practices in MongoDB administration focus on improving database reliability, security, operational efficiency, and system maintainability [20]. As MongoDB is widely used for large-scale and semi-structured data processing, adopting structured administrative practices is essential to minimize operational risks and maintain consistent database performance.

A. Best practices for MongoDB security

Ensuring MongoDB security is essential for protecting sensitive organizational information and maintaining database integrity. Several security practices can be implemented to strengthen MongoDB deployments:

- **Keep MongoDB Updated:** Regularly update MongoDB to the latest stable version to address vulnerabilities and apply security patches.
- **Implement Network Security:** Secure and limit access to databases using firewalls, IP whitelisting and secure network setup. VPNs can also be employed to maintain safe, remote connection.
- **Enable Encryption:** Use SSL/TLS encryption to protect data in transit between the MongoDB servers and client applications to minimize the chance of unauthorized access to data being transmitted.
- **Secure the Deployment Environment:** Regularly patch and secure operating systems running MongoDB servers with appropriate hardening techniques.
- **Implement Role-Based Access Control (RBAC):** Provide user privileges as needed for operation and review access periodically to help reduce security risks.
- **Monitor Security Updates:** Apply all security advisories issued by MongoDB and periodically review for vulnerabilities and recommended mitigations [17].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

B. Effective Schema and Index Design

For administrators, it's crucial to frequently review their indexing strategies to tune their database for optimal performance and to help them support scalable MongoDB environments. Since MongoDB is a schema-less document-oriented database, it is important for administrators to ensure that their data is designed in a logical manner that is suited to their applications and access patterns. When designing schemas, it is important to realize what to store as embedded documents and what to reference other documents. Embedding is good for quicker data retrieval when it helps to access small data sets that aren't that large or highly related to each other, while referencing is good when it helps to access large and highly related data sets.

Indexing is another method used to boost query performance and reduce data retrieval time. Depending on the needs of the query, one might use a single field index, compound index or multikey index. The strategic use of indexes can save on the need for the collection scan, improve response time and optimize resource usage. However, if there are too many or poor indexes, they can lead to storage space overhead and affect the speed of writes. It is important for administrators to review indexing strategies regularly, as they can improve database performance and enable scalable environments where necessary.

C. Query Optimization Techniques

Optimization Technics in query for MongoDB are the techniques that can be used to improve the efficiency of queries and to get back the data [21]. With so much data stored in the current applications, query optimization plays an important role in the performance of the databases as very few resources are being used by the application. In MongoDB, some of the key techniques for optimizing major queries are:

- **Indexing Optimization:** Creating the right indexes, such as single-field, compound and multikey indexes, to reduce the number of collection scans required and to speed up data retrieval.
- **Query Execution Analysis:** Using tools such as explain() to view query execution plans and determine if indexes are used efficiently, and to identify potential blockages.
- **Efficient Query Design:** Only retrieving required fields, restricting the number of returned results and avoiding extra complex queries to reduce processing overhead.
- **Aggregation Pipeline Optimization:** Optimize aggregation stages such as $match, $group, $sort, and $project to improve the performance of analytical queries and minimize resource consumption
- **Schema Design Optimization:** Use embedding or referencing correctly to avoid unnecessary operations and speed up query execution time.

These methods can be used together to optimize resource usage, scalability, and efficiency in a MongoDB query system, resulting in more effective and efficient systems. Table I presents a comparative analysis of MongoDB deployment models

TABLE 1: COMPARATIVE ANALYSIS OF MONGODB DEPLOYMENT MODELS

Parameters	Standalone MongoDB	Replica Set	Sharded Cluster
Architecture	Single database server	Multiple replicated nodes	Multiple shards with replica sets
Data Availability	Limited	High	High
Fault Tolerance	Low	High-throughput failover	High throughput replication and distribution
Scalability	Vertical scaling only	Moderate scalability	Horizontal scalability
Performance	Suitable for small workloads	Better read performance	High performance for large datasets
Data Distribution	No distribution	Data replicated across nodes	Data partitioned across shards
Backup and Recovery	Manual and limited	Improved recovery through replicas	Complex but highly reliable

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, June 2026 Impact Factor: 8.2

Administrative Complexity	Low	Moderate	High
Cost and Infrastructure	Low	Moderate	High
Typical Use Cases	Development and small applications	High availability systems	Big data and large-scale applications

V. LITERATURE REVIEW

The studies that have been reviewed primarily address MongoDB performance, scalability, schema flexibility, and query optimization, comparing with other databases. However, there is limited research that covers operational challenges like security, monitoring, backup, and recovery in MongoDB, which motivates the study of these challenges.

Ciumac et al.(2026) present a study of two databases, MongoDB and RavenDB, focused on the implementation of the core CRUD (Create, Read, Update, Delete) operations. A case study was created to assess the effectiveness of mobile and web apps to manage product orders. The IIoT data characteristics were a key driver for the application - lots of data and many transactional interactions. The experiments were done on the data sets ranging from 1000 to 1000000 records to ensure a statistical control. The findings resulted in very significant differences between the performance of the two systems, especially when it comes to response time, and MongoDB was much faster than RavenDB, with a very large amount of data. However RavenDB has its own strengths: auto indexing, built-in ACID compliance and better relational retrieval and bulk ingestion support [22].

Nuriev et al. (2024) delve into the importance of optimizing MongoDB indexes, in-depth, for boosting query performance. A brief introduction to the document-oriented nature of MongoDB's data storage method, as well as the support that is built into it for scalability, is provided to establish some level of understanding of its importance to good query processing. The discussion continues by emphasizing the importance of indexes in MongoDB, how they can make data retrieval faster, and why it is important to optimize them for the optimal performance of your database. The approaches to identify fields that can be used for indexing are discussed extensively based on various factors such as: Frequency of queries, etc. to use fields in query operations etc. [23].

Achanta (2024) examines how to develop an optimized architecture for hybrid cloud database solutions that combine SQL Server and MongoDB when using Azure-based architectures, as well as the pros and cons. Hybrid cloud is becoming increasingly popular among businesses for its enhanced data management, scalability, and flexibility. Through this research, the authors will demonstrate how organizations can leverage the Azure cloud platform to speed up, secure and synchronize their data by combining the relational database capabilities of SQL Server with the NoSQL capabilities of the MongoDB database [24].

Remala and Mudunuru's (2023) article will make a detailed comparison between MongoDB and Amazon Redshift, examining their architecture, features, and use cases. The report delves into the pros and cons of each platform, exploring key features such as performance, scalability, consistency, and cost-effectiveness. Furthermore, it explores strategies to optimize Redshift and MongoDB data management, such as selecting the most efficient schema designs, query optimization, and data loading techniques [25].

Győrödi et al. (2022) main aim of this study is to compare and contrast the effect of each database on the performance of the application when processing CRUD queries. The document-based databases MongoDB and MySQL were used to develop a case-study application that was analyzed. These databases are designed to emulate and streamline the functions of the service providers that rely on massive amounts of data. The results demonstrate the performance of both of these databases from a variety of data volumes, and based on the results, further research and conclusions were drawn to select the best one for a big data application [26].

Achanta (2022) emphasizes how proactive infrastructure design can transform a sluggish, ineffective, and inflexible MongoDB on-prem from a static structure to a fluid, effective, and elastic system. These guidelines and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

recommendations aim to help IT architects, DevOps engineers, and system administrators deploy MongoDB with confidence, resilience, and prescience [27].

Table II indicates that there are limited studies on operational issues such as security, monitoring, backup and automation, while there are more studies on performance and scalability

TABLE 2: COMPARATIVE ANALYSIS OF LITERATURE ON MONGODB DATABASE ADMINISTRATION

Authors	Focus Area	Challenges	Limitations	Research Gap	Future Work
Ciumac et al. (2026)	Comparative performance analysis of MongoDB and RavenDB for CRUD operations in IIoT-based applications	Managing high-volume transactional data, maintaining fast response time, handling large-scale CRUD operations	Focus limited to performance metrics; lacks discussion on security, backup, replication, and operational administration	Limited research on operational management practices and long-term maintenance of MongoDB in IIoT environments	Future studies can investigate automated administration, fault tolerance, and secure MongoDB deployment frameworks for industrial systems
Nuriev et al. (2024)	MongoDB query performance enhancement through index optimization	Identifying optimal indexing fields, improving query speed, reducing retrieval latency	Concentrates only on indexing techniques without addressing broader database operational issues	Lack of integrated solutions for automated indexing, monitoring, and performance management	Future research may focus on AI-driven indexing mechanisms and intelligent MongoDB performance optimization systems
Achanta (2024)	Optimization of hybrid cloud architectures integrating SQL Server and MongoDB on Azure	Data synchronization, cloud scalability, security management, and hybrid integration complexity	Limited practical evaluation of administrative workload and operational risks in hybrid cloud systems	Need for deeper analysis of MongoDB administration practices in cloud-native and hybrid environments	Future studies can explore automated orchestration, cloud monitoring tools, and disaster recovery mechanisms
Remala and Mudunuru (2023)	Comparative analysis of Amazon Redshift and MongoDB focusing on scalability and data management	Managing consistency, scalability, query optimization, and cost-effectiveness	More focused on analytics and architecture rather than operational database administration	Insufficient research on operational best practices such as backup strategies, replication handling, and workload balancing	Future work can develop intelligent monitoring and optimization frameworks for MongoDB-based big data systems
Győrödi et al. (2022)	Comparative performance evaluation of MongoDB and MySQL for	Handling large data volumes and ensuring application performance	Focuses mainly on CRUD performance and ignores security, monitoring, and	Lack of studies addressing operational scalability and MongoDB	Future research can analyze real-time administration techniques and automated resource

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, June 2026 Impact Factor: 8.2

	CRUD requests in big-data applications	efficiency	maintenance aspects	administration challenges in big-data environments	management in MongoDB systems
Achanta (2022)	Proactive infrastructure design and deployment strategies for MongoDB on-premises	Building resilient, scalable, and elastic MongoDB infrastructures	Recommendations are mostly conceptual with limited empirical validation	Limited investigation into operational failures, security risks, and real-time administration challenges	Future studies may develop standardized frameworks for MongoDB infrastructure monitoring, security, and operational automation

VI. CONCLUSION AND FUTURE WORK

Modern distributed database systems require effective administration of the MongoDB database to run efficiently. It includes vital functions such as configuration, monitoring, backup and recovery, security enforcement, performance tuning and more. MongoDB provides several advantages in terms of flexibility, scalability, HA, and more, but only when it is well-managed and well-administered. Major issues that they carry are performance in low-latency scenarios, replication and sharding, access control and security, backup and restore in distributed environments. Some best practices that can help overcome these challenges are as follows: design the schema for maximum effectiveness, create smart indexes, and use query optimization methods. Furthermore, the deployment models are standalone servers, replica sets, and sharded clusters with various scalabilities, fault tolerances, and complexities of the application. The future of developing MongoDB administration will be based on automation and intelligence. Some domains of interest include performance tuning with AI, predictive monitoring systems, self-healing systems and automatic scaling. Furthermore, with the changing landscape of cyber threats, enhancing security efforts for cloud-native and hybrid MongoDB deployments will be increasingly relevant. The enhancements in these areas will help make MongoDB more efficient, reliable and flexible in next-generation data-intensive applications.

REFERENCES

[1] S. Tarakampet, R. Koilakonda, and S. Tatavarthi, “From Vision to Victory: Best Practices for Architecting Enterprise-Wide Compliance Ecosystems,” *Int. J. Comput. Appl.*, vol. 187, no. 87, pp. 31–37, 2026.

[2] C. Patel, “A Review of Multi-Channel CRM Strategies Using Big Data and Cloud Integration,” *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 1, pp. 577–588, 2022.

[3] S. Wanigasooriya, “Database Administration Developments and Research in Contemporary Digital Environments,” vol. 8, no. 1, pp. 577–588, 2024, doi: 10.13140/RG.2.2.25897.81763.

[4] A. Parupalli and H. Kali, “An In-Depth Review of Cost Optimization Tactics in Multi-Cloud Frameworks,” *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 3, no. 5, pp. 1043–1052, Jun. 2023, doi: 10.48175/IJARSCT-11937Q.

[5] P. C. Jakku, P. Sundaramoorthy, S. C. Kondaparthy, T. Desai, R. Jarubula, and A. Nerella, “Enhancing Cloud Security Via Quantum Computing,” *Proc. Data Anal. Manag.*, pp. 377–386, 2026, doi: 10.1007/978-3-032-03558-5_31.

[6] M. Parikh, A. A. Soni, S. M. Shah, and A. R. Jha, “Big Data Workload Profiling for Energy-Aware Cloud Resource Management,” Jan. 2026, doi: 10.48550/arXiv. 2601.11935.

[7] B. Krishnan, A. Thaneeru, R. Lingam, and S. K. Kaata, “The Future of Cloud Data Engineering: Multi-Tenant, Multi-Region Pipelines Leveraging LLM-Powered Data Governance,” in *2025 1st International Conference on Advancement in Futuristic Technologies (ICAFT)*, 2025, pp. 1–8. doi: 10.1109/ICAFT66710.2025.11453308.

[8] N. D. Karande, “A Survey Paper on NoSQL Databases : Key-Value Data Stores and Document Stores,” *Int. J.*

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, June 2026 Impact Factor: 8.2

Res. Advent Technol., vol. 16, no. 2, 2018.

[9] H. B. Dama, "A Survey of MySQL Database Administration Techniques and Best Practices," *ESP J. Eng. Technol. Adv*, vol. 6, no. 1, pp. 89–98, 2026.

[10] A. Chauhan, "A Review on Various Aspects of MongoDB Databases," *Int. J. Eng. Res. Technol.*, vol. 8, no. 5, pp. 90–92, 2019.

[11] A. Batra, A. Pahwa, and D. Sharma, "MongoDB-NoSQL Database: A Survey," *Adv. Comput. Sci. Inf. Technol.*, vol. 2, no. 5, pp. 495–499, 2015.

[12] V. K. Sharma, "Cloud Computing IoT: 5G Focused IoT with Cloud Solutions," *Int. J. AI, BigData, Comput. Manag. Stud.*, vol. 6, no. 3, 2025, doi: 10.63282/3050-9416.IJAIBDCMS-V6I3P103.

[13] B. P. Singh, "Convergence of AI and Zero Trust: Enabling Continuous Verification Across Hybrid Cloud Environments," *Int. J. Intell. Syst. Appl. Eng.*, vol. 14, no. 1s, pp. 339–348, Apr. 2026, doi: 10.17762/ijisae.v14i1s.8181.

[14] M. R. C. Mukkolakkal, "InfraLLM: A Generic Large Language Model Framework for Production-Grade Microservice Auto-Scaling in Cloud Infrastructure," *Int. J. Sci. Res. Mod. Technol.*, vol. 4, no. 11, pp. 113–123, 2025, doi: 10.38124/ijsrmt.v4i11.1023.

[15] M. Rathore and S. S. Bagui, "MongoDB: Meeting the Dynamic Needs of Modern Applications," *Encyclopedia*, vol. 4, no. 4, pp. 1433–1453, 2024, doi: 10.3390/encyclopedia4040093.

[16] M. R. Dhanagari, "MongoDB and Data Consistency: Bridging the Gap between Performance and Reliability," *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 2, pp. 183–198, 2024, doi: 10.32996/jcsts.

[17] S. K. Singh, P. Dubey, and G. K. Shukla, "MongoDB in a Cloud Environment," *Don Bosco Inst. Technol. Delhi J. Res.*, vol. 1, no. 1, pp. 13–18, 2024, doi: 10.48165/dbitdjr.2024.1.01.03.

[18] A. Kathpal and P. Sehgal, "BARNS: Towards building backup and recovery for NoSQL databases," 2017.

[19] T. Studies, "Scaling with MongoDB: Solutions for Handling Big Data in Real-Time," *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 5, pp. 246–264, 2024, doi: 10.32996/jcsts.

[20] K. Jangiti, "Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments," in *SoutheastCon 2026*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476363.

[21] M. Kari, "AI-Assisted Query Optimization Techniques for Cloud Databases Supporting Hybrid SQL and NoSQL Workloads," *Int. J. Emerg. Res. Eng. Technol.*, vol. 6, no. 4, pp. 62–71, oct. 2025, doi: 10.63282/3050-922X.IJERET-V6I4P108.

[22] M. Ciumac, C. A. Győrödi, R. Ștefan Győrödi, and F. M. Costea, "Performance Evaluation of MongoDB and RavenDB in IIoT-Inspired Data-Intensive Mobile and Web Applications," *Futur. Internet*, vol. 18, no. 1, p. 57, Jan. 2026, doi: 10.3390/fi18010057.

[23] M. Nuriev, R. Zaripova, O. Yanova, I. Koshkina, and A. Chupaev, "Enhancing MongoDB query performance through index optimization," *E3S Web Conf.*, vol. 531, p. 8, Jun. 2024, doi: 10.1051/e3sconf/202453103022.

[24] P. R. D. Achanta, "Optimizing Hybrid Cloud Database Architecture: Integrating SQL Server and MongoDB in Azure Environments," *Int. J. Sci. Res. Manag.*, vol. 12, no. 2, pp. 1815–1826, 2024, doi: 10.18535/ijsrm/v12i12.ec05.

[25] R. Remala and K. R. Mudunuru, "Strategic Data Management: Comparing Amazon Redshift and MongoDB," 2023.

[26] C. A. Győrödi, D. V. Dumșe-Burescu, D. R. Zmaranda, and R. Ș. Győrödi, "A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management," *Big Data Cogn. Comput.*, vol. 6, no. 2, p. 49, May 2022, doi: 10.3390/bdcc6020049.

[27] P. R. D. Achanta, "Overcoming Infrastructure Challenges in MongoDB On-Premises with Smart Design Patterns," *ICONIC Res. Eng. JOURNALS*, vol. 6, no. 3, pp. 305–311, 2022.