Journal of Science Engineering Technology and Management Science
Volume 03, Issue 04, April 2026 *ISSN: 3049-0952*
www.jsetms.com

Optimizing Database Administration: A Survey of SOQL, MQL, and NoSQL-Based Techniques

Mr. Deepak Mehta
Assistant Professor
Department of Computer Sciences and Assistant ProfessorApplications
Mandsaur University, Mandsaur
deepak.mehta@meu.edu.in

***Abstract*—The rapid growth of cloud computing, big data and enterprise applications has made database administration an essential component of modern information systems. This review paper gives a comprehensive survey on optimization techniques in SOQL, MQL and NoSQL-based database systems to improve performance, scalability and efficient data management. The study discusses the roles of relational and non-relational databases in handling structured, semi-structured, and unstructured data in modern applications. It explores crucial optimization techniques such as query optimization, indexing, caching, replication, partitioning, aggregation, and sharding, which improve query execution and resource utilization. The paper also discusses performance characteristics of Salesforce Object Query Language (SOQL), Mongo Query Language (MQL) and NoSQL systems, highlighting their strengths, limitations and practical applications in cloud-native and enterprise environments. The current challenges of consistency, distributed transactions, interoperability and security in large-scale database systems are also discussed in the review. Emerging directions for intelligent and adaptive database management systems are also discussed, including AI- and machine-learning-based query optimization, autonomous database administration, and hybrid SQL–NoSQL architectures.**

Keywords—Database Administration, SOQL, MQL, NoSQL Databases, Query Optimization.

I. INTRODUCTION (*HEADING 1*)

A database is an organized collection of data stored and managed electronically for efficient access, retrieval, and updating [1]. Database administration focuses on maintaining and securing databases, a task that has become increasingly crucial in today's information-driven business environment. Key areas of focus include database security, performance tuning, data backup and recovery, database monitoring, troubleshooting, and planning for future growth. Databases must adhere to the C.I.A. principles of Confidentiality, Integrity, and Availability, as well as considerations of control, authenticity, and utility [2].

Modern database technologies play a pivotal role in addressing these needs, offering various approaches to handle the dynamic and growing data requirements of e-commerce applications [3][4]. Traditional relational databases, such as MySQL and PostgreSQL, have long been the backbone of data management systems due to their robust transaction management and adherence to ACID (Atomicity, Consistency, Isolation, Durability) principles [5]. NoSQL technologies, including MongoDB and Cassandra, are designed to handle large-scale, unstructured data and provide horizontal scalability. These databases are well-suited for applications where speed and flexibility are prioritized over strict consistency[6].

SOQL, MQL, and NoSQL technologies are widely used in modern database systems to efficiently manage and process large volumes of data. SOQL (Salesforce Object Query Language) is a specialized query language for extracting structured data from Salesforce's object-based environment. SOQL plays a critical role in enabling infrastructure and operations teams to retrieve precise, real-time information from Salesforce's database, supporting the development of comprehensive reports and dashboards. MongoDB stands as a pioneering NoSQL database management system, offering a modern alternative to traditional relational databases. At its core, MongoDB operates on a document-oriented model, in which data is stored as flexible, JSON-like documents within collections. NoSQL is a non-relational database management system. It was designed to store distributed data at a very large scale [7].

Modern database systems face several challenges, including performance, scalability, storage management and query execution. As data volume and user demand grow, efficient database administration is critical to maintaining fast response times and reliable system performance. To improve efficiency, SOQL, MQL and NoSQL databases often employ optimization techniques such as query optimization, indexing, caching, data partitioning and replication [8][9]. These techniques allow for reduced resource consumption, faster data retrieval and support large distributed applications. This survey paper reviews optimization techniques in SOQL, MQL, and NoSQL databases. It focuses on query performance, scalability, database administration challenges, and modern data management approaches.

A. Structure of the Paper

This paper is structured as follows: Section II discusses the foundations of database administration, including relational, NoSQL, and cloud-based databases. Section III presents optimization techniques in SOQL, MQL, and NoSQL systems. Section IV highlights challenges, emerging trends, and AI-based optimization approaches. Finally, Section V concludes the paper with future research directions.

II. FOUNDATIONS OF DATABASE ADMINISTRATION

Database Administration (DBA) is the process of managing and maintaining databases to ensure efficient data storage, retrieval, security, and availability. Database administrators are responsible for tasks such as performance

monitoring, backup management, recovery operations, user access control, and query optimization. Effective database administration is essential for maintaining reliability and efficiency in modern enterprise systems.

Modern database systems support both relational and non-relational data models. Relational databases organize data into structured tables and maintain transactional consistency through ACID properties. In contrast, NoSQL databases provide flexible schemas and distributed storage mechanisms suitable for handling large-scale, heterogeneous data. These database models are widely used in cloud computing, big data analytics, and real-time applications [10]. Various tools and techniques are employed to ensure these aspects:

A. Confidentiality Controls

Tools and controls for maintaining confidentiality include encryption at rest and in transit, auditing techniques, real-time threat identification systems, hardware-based blocking, role-based authentication, and row-level data masking.

B. Integrity Controls

Ensuring integrity involves regularly testing database backups, verifying constraints, using CASCADE functionality, following locking protocols, maintaining administrator oversight, performing block-, record-, and table-level integrity checks, and conducting extended logical checks.

C. Availability Controls

Availability is maintained through high-availability and disaster-recovery solutions, clustered database instances (active/passive automatic failover), cloud multi-region availability, query performance tuning, real-time monitoring solutions, backup solutions, and load balancing and splitting across servers.

1) Traditional Relational Databases

Traditional Relational Database Management Systems (RDBMS) store data in structured tables with rows and columns. Relational DBMSs (RDBMS) follow many of the concepts introduced in the relational model. Many popular RDBMSs, such as PostgreSQL and Oracle Database, have also adopted data structures from other logical data models. What is common to almost all modern RDBMSs is that they use Structured Query Language (SQL) to define data structures and retrieve and manipulate data. Typically, RDBMSs also implement a strong data consistency model that dictates or allows that database operations grouped into a transaction must all succeed or all fail, that data must follow defined business logic, that successful transactions persist in storage, and that concurrent transactions must result in the same data as if the transactions were serial. At least the last rule can often be loosened in modern implementations to various degrees. These constraints are collectively referred to as an ACID consistency model [11].

2) Role of NoSQL Databases

NoSQL databases have emerged as an alternative to relational databases, designed to address the challenges posed by high-volume, high-velocity, and heterogeneous data in modern enterprise and cloud-native applications. These databases encompass diverse types, including document-oriented stores (MongoDB, Couchbase), key-value stores (Redis, DynamoDB), column-family stores (Cassandra, HBase), and graph databases (Neo4j, Amazon Neptune), each optimized for specific workloads and access patterns. NoSQL systems offer flexible schema designs, allowing developers to store semi-structured or unstructured data without predefined schemas, facilitating rapid application iteration and supporting dynamic business requirements. Their distributed nature provides horizontal scalability through sharding and replication, enabling high throughput and fault tolerance for large-scale applications, such as social media platforms, real-time analytics engines, IoT infrastructures, and content delivery networks. Querying mechanisms differ from traditional SQL, often relying on API-based access, document queries, or graph traversals. At the same time, consistency models vary from eventual consistency to tunable consistency based on application needs. NoSQL systems trade off some aspects of ACID compliance to achieve scalability and performance, following the BASE model for many workloads [12].

3) Cloud-Based Database Systems

A Database deployed in a virtualized cloud infrastructure is called a cloud database. Cloud enables virtual models supporting ‘everything-as -a-service’; that is why databases can also be included in cloud products and services. Relational databases can be shifted to the cloud – the new model is called Relational Cloud. It enables users to shift the majority of responsibilities to the service operator, such as scaling, backup, privacy and access control [13][14]. Cloud providers can offer ready-made database services or DIY(Do It Yourself) services. Ready-made database services are where the cloud provider installs, configures and maintains a cloud server [15]. DIY service means that the majority of responsibilities are on the cloud user. DIY service can be useful when migrating a database to the cloud with little or no change (lift-and-shift).

NoSQL is a technology that may challenge the dominance of relational databases. NoSQL databases can be scaled horizontally, offer greater flexibility and support schema-less models. These features can be achieved by migrating or building a database in the cloud, leveraging cloud features such as scalability [16].

III. Optimization Techniques In SOQL, MQL, And NoSQL Databases

The topic focuses on various techniques for improving the performance, efficiency, scalability and resource management of modern database systems based on SOQL, MQL and NoSQL technologies. It examines the role of query optimization, indexing, schema design, sharding, replication, caching, and distributed processing in enabling databases to efficiently manage large volumes of structured, semi-structured, and unstructured data. The work also compares optimization techniques used in Salesforce databases, MongoDB systems, and other NoSQL platforms. It discusses existing challenges, performance issues, and emerging trends in cloud-based and intelligent database management systems.

A. Query Optimization Techniques in SOQL.

SOQL is a structured query language tailored for Salesforce objects, fields, and records. Query Optimization Techniques in SOQL focus on improving query performance,

reducing execution time, and enhancing resource utilization in Salesforce databases. Techniques such as selective filtering, indexing, query execution plan analysis, and efficient aggregation help optimize SOQL queries and improve system scalability.

- Understanding Query Execution Plans Optimizing SOQL queries begins with a thorough understanding of how Salesforce executes them. Execution plans reveal how data is retrieved, indexed, and filtered, helping developers identify inefficiencies. By analyzing these plans, organizations can pinpoint queries that scan entire objects unnecessarily or trigger multiple database calls, which increase latency and resource consumption. Proper interpretation of execution plans allows targeted optimization strategies, ensuring queries return results quickly while minimizing system load.
- Indexing Strategies for LargeData Volumes Indexes play a crucial role in improving query performance, particularly for large Salesforce datasets. Standard and custom indexes enable efficient retrieval of frequently queried fields, reducing the need for full table scans. Composite and external indexes can further optimize queries with multiple filtering criteria. In hybrid Unix/Linux environments, indexing must consider data distribution across nodes and replication schedules. Proper index management ensures consistent performance, accelerates reporting, and supports compliance by enabling rapid access to audit-relevant data.
- Efficient Data Filtering and Aggregation Methods. Effective filtering and aggregation are essential to prevent unnecessary data processing. Using selective filters, avoiding wildcard searches, and leveraging relationship queries reduce .processing overhead. Aggregate functions such as COUNT(), SUM(), and GROUP BY should be applied judiciously to minimize computational cost. Combining these techniques with batch processing for large datasets ensures that SOQL queries maintain acceptable performance while supporting regulatory reporting and operational transparency across hybrid infrastructures [17].

B. Query Processing and Aggregation in MQL

MongoDB is a document-based NoSQL database that offers high availability and scalability as its primary features. It stores data in JSON format and provides JavaScript functions to query it, allowing fields to vary from document to document and the data structure to change over time. Query Processing and Aggregation in MQL involve retrieving, filtering, grouping, and analyzing data in MongoDB using JSON-like queries and aggregation pipelines. These techniques improve data processing efficiency, query performance, and real-time analytics in large-scale applications [18].

1) Query Processing in MQL

Query processing in Mongo Query Language (MQL) involves retrieving, filtering, and managing data in MongoDB using JSON-like queries. MongoDB uses operators such as $eq, $gt, and $in to efficiently evaluate conditions. Query performance is improved through indexing and optimization techniques, which help in faster data retrieval and reduced execution time. The generic structure of MongoDB is shown in Fig. 1.

2) Aggregation in MQL

Aggregation in the MongoDB Query Language (MQL) is used to process and analyze large volumes of data by grouping, filtering, and transforming documents. MongoDB uses the aggregation pipeline, with stages such as $match, $group, $sort, and $project, to efficiently perform operations like data summarization, calculations, and reporting. It is widely used in analytics, real-time processing, and big data applications [19].

C. Comparative Analysis of SOQL, MQL, and NoSQL Systems.

Table I compares SOQL, MQL, and NoSQL systems based on data model, scalability, query processing, and use cases. It shows that SOQL is well-suited to Salesforce CRM environments, whereas MQL and NoSQL systems offer greater flexibility and scalability for modern big data applications.

TABLE I. COMPARATIVE ANALYSIS OF SOQL, MQL, AND NOSQL SYSTEMS

Feature	SOQL	MQL	NoSQL Systems
Database Type	Salesforce relational database	Document-oriented database	Non-relational databases
Data Model	Structured objects	JSON/BSON documents	Flexible schema models
Query Language	SOQL	MQL	API/document queries
Scalability	Cloud scalability	Horizontal scaling	High distributed scalability
Consistency	Strong consistency	Tunable consistency	BASE/eventual consistency
Optimization	Query optimization, indexing	Aggregation, sharding	Replication, partitioning
Main Use Cases	CRM applications	Real-time web apps	Big data, IoT, analytics
Key Advantage	Secure CRM integration	Flexible schema	High scalability

Major Limitation	Salesforce dependent	Complex joins	Operational complexity

IV. Challenges And Emerging Trends In Database Systems

The explosive growth of data and cloud-based applications has posed challenges in scalability, query performance, security and efficient data management for modern database systems. Emerging trends such as AI-based optimization, cloud-native databases [20][21], automation and distributed database technologies are enhancing the efficiency and performance of modern database systems.

A. Challenges and Limitations of NoSQL Databases

The lack of SQL is the biggest drawback of NoSQL databases. Almost every NoSQL database uses its own way of creating queries (e.g., an access language) that differs from one database to another, depending on the model or specific product. This makes it impossible to create a unique language that would work with these types of databases. Due to their different approaches and data structures, the learning curve for NoSQL databases is steep. This is also the reason for the creation of hybrid databases that allow queries to be written in the SQL language over NoSQL databases. They noticed problems with denormalization, secondary indexes and JOIN processes. If the data is classified across different databases, a problem occurs when switching databases because there is no unique language. This can cause problems in the system, and SQL databases guarantee security with such data storage structures.

JOIN operations are missing, a deficiency many authors consider directly related to the absence of the SQL language. However, some databases, such as MongoDB, offer certain ways of linking (referencing) based on CRUD operations. JOIN operations affect not only the creation of complex queries but also their performance. The lack of ACID transactions often forces developers to implement their own code to provide these features, leading to a complex system. NoSQL databases, by achieving better performance in CRUD operations, exhibit weaker ACID transaction capabilities [22].

B. Improving Query Performance using Machine Learning

The application of machine learning (ML) to improve query response times, specifically for SOQL (Salesforce Object Query Language) queries, is one of the most promising future research directions in Salesforce database optimization. Machine learning algorithms can identify patterns in queries and optimize SOQL queries using historical data [23]. With the use of ML, Salesforce can auto-learn based on query execution time, detect bottlenecks and use forecasting models to enhance the performance of queries in real-time. e.g., ML could offer the most effective indexing policies or indeed dynamically propose alterations to query specifications. Such new developments would lead to significant performance improvements in large-scale Salesforce implementations, where query performance is vital to maintaining system speed and reliability [24].

C. Hybrid SQL–NoSQL Architectures

Hybrid database architectures integrate the strengths of both SQL (Structured Query Language) and NoSQL (Not Only SQL) technologies to manage and process data. These architectures aim to provide a unified solution that leverages the advantages of both SQL and NoSQL, such as the robust querying capabilities of SQL databases and the flexible, scalable nature of NoSQL databases[25]. SQL databases handle structured, transactional operations, while NoSQL systems manage scalable, high-volume, unstructured data. This integration enhances query efficiency, scalability, resource utilization, and real-time data processing in modern enterprise applications.

D. Cloud-Native and Autonomous Database Systems

Cloud-Native Databases (CNDBs) are increasingly integrating Artificial Intelligence (AI) technologies to improve database performance, scalability and automation [26][27]. Unlike traditional databases, AI-enabled CNDBs can dynamically optimize query execution, workload balancing, and resource allocation. The integration of AI significantly improves operational efficiency and reduces manual database administration efforts. However, the development of AI-powered CNDBs remains in its early stages, and many systems lack standardized maturity models to evaluate their capabilities [28]. As a result, database developers face challenges in effectively implementing AI technologies, while users encounter difficulties in selecting suitable CNDB platforms. Therefore, there is a growing need for maturity models and evaluation frameworks to support the analysis, optimization, and continuous improvement of AI-empowered cloud-native database systems [29].

V. Literature Review

Recent studies highlight the importance of hybrid architectures and intelligent optimization methods for modern cloud and enterprise applications.

Z. Haider and Z. Huma (2026) systematically evaluate key optimization techniques—including indexing strategies, query optimization, denormalization, caching frameworks and data partitioning—demonstrating their effectiveness in reducing bottlenecks and improving response times. In addition, it explores emerging database paradigms such as NoSQL systems, in-memory databases, and cloud-native database solutions, assessing their suitability for modern, high-demand applications with dynamic workloads and real-time processing requirements [30].

Y. Weng and J. Wu (2025) provides a comprehensive comparative analysis of PostgreSQL and MongoDB, focusing on their suitability for AI use cases. Key evaluation criteria include data modeling, query complexity, scalability, ACID compliance, indexing, and integration with AI frameworks. PostgreSQL excels in scenarios requiring strict data consistency, complex querying, and structured data, making it ideal for financial modeling, scientific research, and feature engineering. Conversely, MongoDB's schema-less design, horizontal scalability, and native support for semi-structured data align well with real-time analytics, IoT, and evolving AI datasets [31].

B. Dhillon (2024) examines the role of SOQL query tuning as a core strategy for improving Salesforce CRM performance, with particular emphasis on hybrid Unix infrastructures supporting enterprise-grade workloads. It highlights optimization techniques, AI-assisted monitoring, and automation frameworks that improve execution efficiency while reducing latency. The review further explores the integration of AI-driven solutions that provide predictive insights, autonomous query optimization, and adaptive workload management. Case studies across the finance, healthcare, retail, and government sectors illustrate practical applications, benefits, and limitations [32].

M. Nuriev et al (2024) delve into the critical aspect of enhancing query performance in MongoDB through meticulous index optimization. It begins with an introduction to MongoDB's unique document-oriented data storage approach and its inherent scalability, which sets the stage for understanding the importance of efficient query processing. The discussion progresses to highlight the pivotal role of indexes in MongoDB, emphasizing their function in expediting data retrieval and the necessity of optimizing them to ensure peak database performance. A detailed exploration is provided of methodologies for identifying fields suitable for indexing, considering factors such as query frequency and the specific use of fields in query operations [33].

V. Rodrigues (2023) explores the fundamentals of SOQL, highlighting key differences from SQL, the role of governor limits, and their implications for query design. It examines performance optimization strategies, including indexing, query selection, and query plan analysis, and emphasizes the importance of secure, auditable query execution [34].

S. E. A. Youssef (2023) explores various performance optimization techniques, including indexing, query optimization, partitioning, caching, and load balancing. Additionally, it delves into integrating SQL and NoSQL databases in modern data architectures, examining how hybrid approaches can leverage the strengths of both to meet the demands of large-scale enterprise applications. The challenges of ensuring data consistency, handling distributed transactions, and maintaining performance in a mixed-database environment are analyzed, along with proposed strategies to overcome these obstacles [35].

Table II highlights key challenges and research gaps in SQL, NoSQL, SOQL, and hybrid database systems. Most studies lack intelligent AI-driven optimization, scalability management, and efficient hybrid integration.

TABLE II. COMPARATIVE ANALYSIS OF RECENT STUDIES ON SOQL, MQL AND NOSQL-BASED TECHNIQUES

Authors	Focus	Challenges	Limitations	Research Gap	Future Work
Z. Haider and Z. Huma(2026)	Indexing, query optimization, denormalization, caching, partitioning, NoSQL, in-memory and cloud-native databases	Balancing consistency and scalability, workload prediction accuracy and integration complexity	Managing dynamic workloads, reducing bottlenecks and maintaining real-time processing efficiency	Need for intelligent autonomous optimization frameworks for cloud-native hybrid systems.	AI-based adaptive optimization, scalable real-time database architectures
Y. Weng and J. Wu (2025)	PostgreSQL vs MongoDB for AI Applications	Managing structured and semi-structured data together, scalability vs consistency trade-offs	Limited to only two database systems; lacks enterprise-scale benchmarking	Need for generalized AI-centric database evaluation models across multiple DBMS platforms	AI-integrated hybrid database architectures and adaptive schema evolution
B. Dhillon (2024)	SOQL Query Tuning in Salesforce CRM	Query latency, governor limits and workload balancing in multi-tenant environments	Focused mainly on Salesforce and Unix-based enterprise systems	Limited research on universal AI-assisted optimization methods applicable across heterogeneous cloud systems	Autonomous CRM optimization systems and predictive resource allocation techniques
M. Nuriev et al. (2024)	MongoDB Index Optimization	Index maintenance overhead, balancing insert and retrieval performance	Concentrates mainly on indexing; less focus on security and distributed transactions	Lack of intelligent indexing systems integrated with hybrid SQL–NoSQL environments	AI-assisted indexing and adaptive query execution mechanisms
V. Rodrigues (2023)	SOQL Fundamentals and Performance Optimization	Governor limits, secure query execution and scalability challenges	Restricted to Salesforce query architecture; lacks hybrid cloud integration analysis	Need for scalable SOQL optimization methods integrated with AI and hybrid databases	Smart query analyzers and automated optimization tools
S. E. A. Youssef (2023)	Hybrid SQL–NoSQL Performance Optimization	Distributed transaction management, synchronization and maintaining performance consistency	Limited focus on security, governance, and automated workload handling	Need for intelligent hybrid database management frameworks for enterprise applications	Automated workload distribution and secure hybrid transaction systems

VI. CONCLUSION AND FUTURE WORK

Modern database administration is a vital part of managing large-scale enterprise and cloud applications. This review paper discusses the optimization techniques in SOQL, MQL and NoSQL database systems to improve query performance, scalability, storage management and resource utilization. The study underlined the importance of relational and non-relational databases for efficiently handling structured, semi-structured and unstructured data. Major approaches for improving database performance were discussed, including query optimization, indexing, caching, aggregation,

replication, partitioning and sharding. The review also compared SOQL, MQL and NoSQL systems based on architecture, scalability, consistency and practical applications. The study revealed several challenges, including interoperability, distributed transaction management, security, a lack of standardized query languages for NoSQL systems, and ACID compliance. Future research should focus on developing intelligent, AI- and machine-learning-based database optimization frameworks that enable adaptive query tuning, workload prediction, automated indexing, and autonomous resource allocation. Also needed is research into hybrid SQL-NoSQL architectures to combine strong consistency with distributed scalability. Future work might also address secure cloud-native database platforms, edge-cloud-integrated database systems for IoT applications, self-healing database administration tools and automated monitoring systems to improve reliability, fault tolerance and real-time data processing in large-scale distributed environments.

REFERENCES

[1] 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.

[2] 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.

[3] J. W. Sajja, G. B. Komarina, and N. K. R. Choppa, "Enterprise Data Transformation in the Era of S/4HANA: Real-World Cloud Migration Architecture, Governance Strategies, and Lessons from the Field," *World J. Adv. Res. Rev.*, vol. 26, no. 2, pp. 3596–3619, May 2025, doi: 10.30574/wjarr.2025.26.2.2038.

[4] M. Chanda, "A Low-Cost System for Acquiring Login/Logout Data for On-Ground Racks of in-Flight Entertainment Systems," California State University, 2016.

[5] 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.

[6] F. Qureshi and H. Rasheed, "Comparative Analysis of Modern Database Technologies for Scalable Data Storage in AI-Driven Ecommerce Applications," 2022. doi: 10.13140/RG.2.2.14668.83848.

[7] K. Modhiya, "Introduction to DBMS, RDBMS, and NoSQL database: NoSQL database challenges," 2021.

[8] 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.

[9] H. Ravilla, J. Yarra, and S. Dilip, "Role of SOQL and Database Optimization in Large-Scale Salesforce Implementations," *Int. J. Eng. Archit.*, vol. 3, no. 1, pp. 13–31, Feb. 2026, doi: 10.58425/ijea.v3i1.481.

[10] 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.

[11] T. Taipalus, "Database management system performance comparisons: A systematic literature review," *J. Syst. Softw.*, vol. 208, p. 111872, 2024, doi: https://doi.org/10.1016/j.jss.2023.111872.

[12] V. K. Jangala, "Relational and NoSQL Databases in Enterprise Systems," *Int. J. Contemp. Res. Multidiscip.*, vol. 1, no. 1, pp. 125–131, 2022.

[13] 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.

[14] 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.

[15] 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.

[16] S. Dziubak, "REVIEW OF CLOUD DATABASE BENEFITS," *Mod. Manag. Rev.*, vol. 28, no. 3, pp. 7–16, 2023, doi: 10.7862/rz.2023.mmr.14.

[17] P. Grewal and L. G. Vidyapeeth, "Next-Generation Compliance: Integrating SOQL Optimization With Hybrid Unix/Linux Infrastructure Monitoring via Nagios and Zabbix," *Int. J. Trend Res. Dev.*, vol. 8, no. 2, pp. 1–6, 2021.

[18] R. Harshitha and V. R. C, "AN ANALYSIS OF QUERY PERFORMANCE : MONGODB n SQL," *Int. Res. J. Eng. Technol.*, vol. 07, no. 05, pp. 1–4, 2020.

[19] 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.

[20] V. Sanikal, "AI-Enhanced Network Security Framework for Cloud-Edge Integrated Environments," in *2026 Innovations in Machine, Engineering, and Digital Conference (IMED)*, 2026, pp. 1–6. doi: 10.1109/IMED68921.2026.11484417.

[21] 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.

[22] R. Božić, "Advantages And Challenges Of Nosql Compared To Sql Databases - A Systematic Literature Review," *Zb. Rad. Ekon. Fak. BRČKO*, vol. 16, no. 16, pp. 1–8, Sep. 2022, doi: 10.7251/ZREFB2216011B.

[23] H. N. Shantanu Kumar, Shruti Singh, "Resource Management in AI-Enabled Cloud Native Databases: A Systematic Literature Review Study.," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 21, pp. 3621–3630, 2024.

[24] J. Y. Hemadri Ravilla1a and S. Dilip3, "Role of SOQL and Database Optimization in Large-Scale Salesforce Implementations," *Int. J. Eng. Archit.*, vol. 3, no. 1, pp. 1–19, 2026.

[25] L. Peña, "Holistic Approaches to Strategically Integrating SQL and NoSQL Solutions in Hybrid Architectures for Optimized Performance and Versatile Data Handling," *J. Artif. Intell. Mach. Learn. Manag.*, vol. 7, no. 1, pp. 93–115, 2023.

[26] T. P. Patel, "Adaptive Token Routing for Heterogeneous LLM Inference in Edge-Cloud Continuum," in *SoutheastCon 2026*, 2026, pp. 1–7. doi: 10.1109/SoutheastCon63549.2026.11476596.

[27] 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.

[28] J. B. Mehta, "AI-Driven Test Engineering for Cloud-Native Systems," *Int. J. Data Sci. IoT Manag. Syst.*, vol. 5, no. 1, 2026, doi: 10.64751/ijdim.2026.v5i1.297.

[29] P. Parida and N. Senguttuvan, "Responsible Utilization of Cloud in Retail Banking Ecosystem," *Int. J. Comput. Appl.*, vol. 187, no. 49, pp. 34–39, Oct. 2025, doi: 10.5120/ijca2025925835.

[30] Z. Haider and Z. Huma, "Optimizing Database Architectures for High-Performance Web Applications: A Comprehensive Analysis," *Glob. Knowl. Acad.*, vol. 7, no. 1, pp. 1–13, 2026.

[31] Y. Weng and J. Wu, "Database management systems for artificial intelligence : Comparative analysis of postgre SQL and MongoDB," *World J. Adv. Res. Rev.*, vol. 25, no. 02, pp. 0–6, 2025.

[32] B. Dhillon, "Salesforce CRM Performance Optimization Using SOQL Query Tuning in Hybrid Unix Infrastructures With AI Assistance," *Int. J. Nov. Res. Econ. , Financ. Manag.*, vol. 2, no.

4, pp. 1–7, 2024.

[33] 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.

[34] V. Rodrigues, "Optimizing SOQL Queries for Compliance and Security in Salesforce While Running on Hybrid Unix Infrastructure Systems," *Int. J. Trend Res. Dev.*, vol. 10, no. 10(1), pp. 1–7, 2023.

[35] S. E. A. Youssef, "Optimizing Database Performance for Large-Scale Enterprise Applications: A Comprehensive Study on Techniques, Challenges, and the Integration of SQL and NoSQL Databases in Modern Data Architectures," *J. Artif. Intell. Mach. Learn. Manag.*, vol. 7, no. 1, pp. 81–92, 2023.