International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 6 Issue 4 PP 62-71, 2025
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V6I4P108

Original Article

AI-Assisted Query Optimization Techniques for Cloud Databases Supporting Hybrid SQL and NoSQL Workloads

Mahathi Kari
Independent Researcher.

Received On: 28/08/2025 ***Revised On: 09/09/2025*** ***Accepted On: 02/10/2025*** ***Published On: 21/10/2025***

Abstract - *Data management in the modern era has changed with the introduction of the paying database solutions which include Database as a Service (DBaaS) with the use of cloud computing. Nevertheless, query optimization in distributed and hybrid systems is highly challenging because of the dynamic workloads, heterogeneity of resources and cross platform data integration. By virtue of being based on standard rules and costs, traditional approaches to rule-based and cost-based optimization do not scale, and cannot easily adjust to the complexity and volume requirements of cloud-based and hybrid SQL-NoSQL databases. The present paper is based on the idea of the use of Artificial Intelligence (AI) and Machine Learning (ML) to optimize queries in a data-driven, adaptive, and autonomous manner. Supervised and unsupervised learning AIs that are used as optimizers are applied to learn cost models, predict optimal execution plans, and dynamically respond to changes in the workload, whereas workload forecasting and resource allocation is possible using deep learning (DL) models such as LSTMs and Transformers. Comparative study of SQLs and NoSQLs system shows that consistency and scalability have a complementary relationship, which makes it crucial to have hybrid system in the cloud ecosystem. The research finds that AI-based query optimization not only enhances efficiency, performance, and adaptability but also preconditions future autonomous and self-tuning database systems, which can work perfectly well in multi-cloud environments.*

Keywords *- AI-Driven Query Optimization, Hybrid SQL-NoSQL Databases, Cloud Database Management, Adaptive Workload Optimization, Cost-based Optimization.*

1. Introduction

In service-oriented computing cloud computing is a very successful paradigm in service-oriented computing. The most common services of cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Any further development of this is Database as a Service (DBaaS) or Storage as a Service. Cloud computing improves sharing of computing power and storage of number of database applications [1]. Cloud computing application can be based on cloud data modeling, and the searching algorithm on cloud computing application is the key issue of cloud computing application. The fundamental principle of database management systems is query optimization, which aims at maximizing the efficiency of query execution with respect to finding optimal execution plans and minimizing resource consumption and response time [2]. With the distributed relational databases, query optimization is even harder since there is the need to coordinate and optimize the execution of queries across the numerous dispersed nodes. The existing optimization methods that have been created based on centralized databases might be inadequate to meet the peculiarities and issues of distributed environment.

Multi-cloud environments and the cloud database systems in particular are very beneficial to the enterprises in terms of scalability, flexibility and availability. They can spread data across various cloud service providers (CSPs), preventing vendor lock-in and realizing cost efficiency, and improving disaster recovery by using redundancy [3][4]. Besides, multi-clouds help in meeting data governance laws through geo-distribution of storage. Nevertheless, these advantages bring about problems, such as data-synchronization, cross-cloud-correlations, and safe query-execution. Such issues require the capability to handle these challenges by using powerful query optimization techniques which can work effectively on heterogeneous platforms and loads.

Moreover, the extensive use of multitenant database architectures has increased the opportunities and challenges in the cloud system [5]. Although multitenancy is cost-efficient and scalable in the sense that it provides the facility of sharing common database infrastructure by several tenants, it also increases the resource contention and variability of query performance. Resource support, workload balancing, and service level agreements (SLA) of such shared environments should optimally be dynamically allocated without affecting the data isolation and consistency of performance. The concept of Artificial Intelligence (AI) and Machine Learning (ML) is reinventing the idea of query optimization by proposing adaptable data-driven approaches that overcome the weaknesses of the conventional rule-based solutions. The AI-assisted optimizers constantly use the query execution data to predict performance, choose the best strategies and dynamically adjust to the workload. Reinforcement learning, cost model learning, and other techniques, allow systems to enhance execution plans via iterative feedback, and cost model learning trains and replaces heuristic estimates with empirically trained models to be more accurate [6]. Such techniques are further expanded by the DL methods which of modeling query-plan relationships, which can then be optimized in an autonomous and self-tuning way.

The latest developments in AI-assisted query optimization have been expanded to include hybrid SQL/NoSQL workloads, with both relational and non-relational data-stores collocated in clouds [7][8]. Such hybrid systems require cross database optimization methods that would combine structured and unstructured data through a common query framework. It has been shown that deep reinforcement learning, predictive modeling and learned index structures can significantly increase scalability, flexibilities, and performance in such heterogeneous environments.

1.1. Structure of this Paper

The paper is structured as follows: Section II presents a query optimization in cloud databases. Section III discusses hybrid SQL and NoSQL cloud databases. Section IV explores AI-driven techniques and models. Finally, Section V concludes and future research directions.

2. Query Optimization in Cloud Databases

Fundamentals of query optimization involve selecting efficient execution plans using cost-based, rule-based, and modern DL approaches to improve performance, accuracy, and adaptability in SQL and hybrid database systems [9]. Several components make up the architecture of the suggested query optimizer shown in Figure. 1.

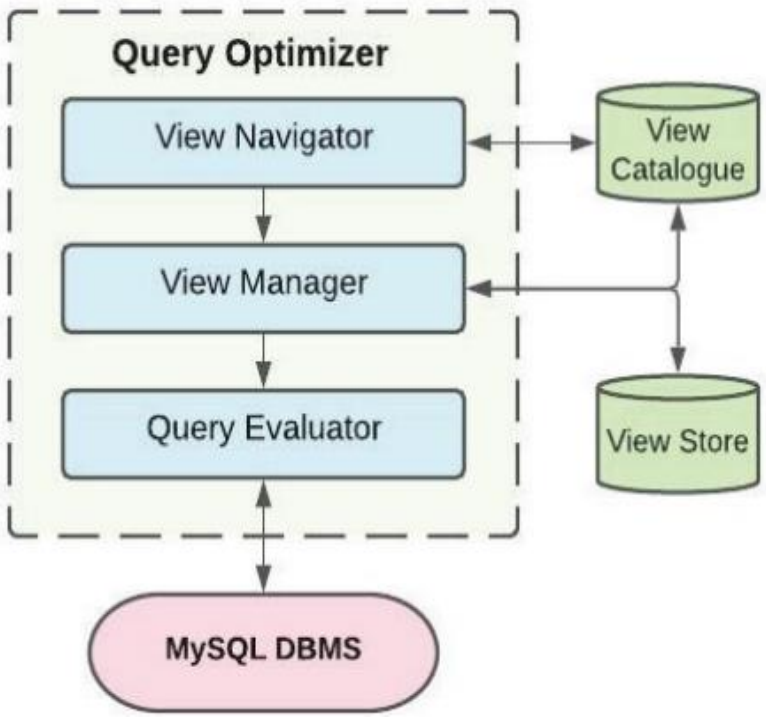

Fig 1: Architecture of Query Optimizer

- The View Navigator uses the view matching technique to navigate the View Catalogue and find relevant views during query evaluation. Each catalogue entry is scanned for the longest match view name recorded in the view store, and the location in the view store where the match was located is returned [10]. Future query assessment also makes advantage of partial interim results. The commutative rule is used by this algorithm for natural joins.
- View Manager refreshes the View Catalogue and keeps created views up to date in the View Store. Along with the read/write timestamp values for each view, it also records the number of view references. When deciding whether to remove views from the View Store and update the View Catalogue accordingly, these values are taken into consideration.
- The query evaluation work is carried out by the query evaluator following the substitution of matched views.
- View Store makes views that are produced from previously assessed queries a reality.
- The View Catalogue keeps track of the list of view names that appear in the View Store. View Manager mapped the view name from the view catalog into the view store.

2.1. Traditional Query Optimization Techniques

An essential part of relational database management systems (RDBMS) since their beginnings has been query optimization (Figure 2). The two main types of traditional optimization strategies are rule-based and cost-based [11].

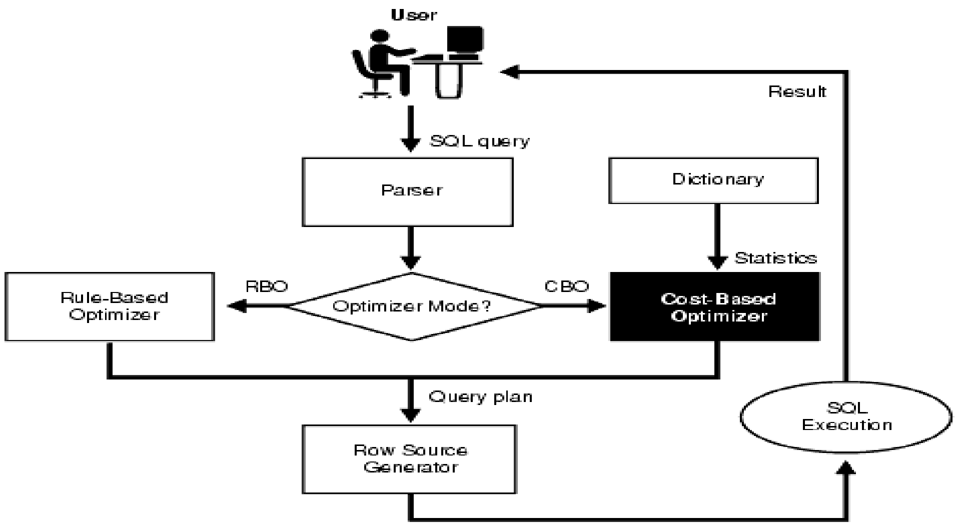

Fig 2: Query Based Optimization Techniques

2.1.1. Rule-Based Optimization

Rule-based optimizers turn a given query into an equivalent but more effective execution plan by using a predetermined set of transformation rules. Typically, these rules are created using expert knowledge and incorporate changes like as join reordering, predicate pushdown, and index selection. Rule-based optimizers are quick and simple, but they are not flexible and might not always generate the best execution strategy, particularly in complicated query contexts.

2.1.2. Cost-Based Optimization

Cost-based optimization, on the other hand, involves estimating the cost of various possible execution plans and selecting the one with the lowest estimated cost. The cost is usually quantified in resources used i.e. CPU time, memory, and disk I/O. This is more flexible compared to the rule-b ased optimization approach since it can also be adjusted to various cloud database configurations and workloads [12]. Nevertheless, it is difficult to estimate the cost correctly and computationally costly. In addition, the accuracy of the cost model has a large effect on the effectiveness of the optimizer, inaccurate models may result in poor plans.

2.1.3. Heuristic-Based Techniques

Besides the rule-based and cost-based approaches, heuristic-based algorithms have been suggested to enhance query optimization. They are based on heuristics or rules of thumb to make fast decisions regarding the query transformation to simplify the optimization process. Nonetheless, heuristics are not always the best plan to be used and are usually integrated with other optimization techniques.

2.2. Query Planning and Execution Workflow

Delegation of query planning and execution to DL model in relational DBMSs. Optimizers that are traditional are based on cost models and manual feature engineering and thus they tend to miss many interactions between operators. The dynamic construction of neural units at operator level in each relational operator of a query plan, the proposed plan-structured neural network removes human-designed models [13]. These neural units forecast the operator latencies and transfer performance properties up the execution tree, and are able to model dependencies in the workflow correctly. It is based on the real query execution structures and thus it allows accurate latency prediction and execution strategies optimization, far better than the classical heuristic and cost-based models.

2.3. Opportunities and Strategies for Cloud-based Query Optimization

There are some strategies for query optimization as discuss below:

2.3.1. Auto-scaling and Resource Management

Optimization of queries should use cloud-native capabilities like auto-scaling and tagging of resources to redistribute resources dynamically in response to the nature of workloads [14]. Databases are able to increase or decrease the resources automatically with performance indicators and workload trends to maximize the performance of queries.

2.3.2. Serverless Computing

Serverless interface, including AWS Lambda or Azure Functions, provide a chance to optimize query execution by providing finely-grained resource allocation and charging on real resource usage. The serverless computing models can be customized to employ query optimization techniques in order to support scalable and cost-efficient processing of queries.

2.3.3. Geo-distributed Query Optimization

The ability of cloud providers to provide data centers in various regions makes geo distributed query optimization essential in reducing the latency of data transfer and enhancing the response time of queries [15]. Data partitioning, query routing and caching are some of the techniques that can be used to optimize query execution in distributed data centers.

2.3.4. Query Offloading and Edge Computing

Latency and network overhead in specific query types can be reduced by offloading query processing (to edge devices or edge computing infrastructure). Edge computing paradigms can be acquired with query optimization strategies, which distributes queries processing functions further down the data-tree or end-user to enhance performance.

2.3.5. Cost-aware Query Optimization

Cost-consciousness in query optimization models can be used to make performance-cost decisions. Using the factors, e.g., instance pricing, and data transfer costs, as well as resource utilization, query optimization algorithm can be used to ensure that query plans are optimized to achieve minimum overall cost of operations allowed and satisfy performance goals.

3. Hybrid SQL and Nosql Cloud Databases

The SQL databases support the integrity of structured data as ACID compliant and relational schema design, which makes them best suited to handle transactions in a consistent manner [16]. Conversely, NoSQL databases are scaled, flexible and schema-less databases that are suitable to dynamic and high-volume data in the cloud. Therefore, SQL-NoSQL integration facilitates relational and scalability to address the current data-driven application requirements.

3.1. Characteristics of SQL Databases

SQL databases (also relational databases) are designed to manage structured data which has defined schema. They arrange data in tabular format consisting of rows and columns; information between tables is connected by primary and foreign key and these attributes guarantee data integrity and eradicate redundancy. One of their major characteristics is that they support ACID properties Atomicity, Consistency, Isolation and Durability that can ensure transactions are reliably and predictably processed. A standard query language (SQL) is used in SQL [17] databases to query and manipulate data in an efficient manner. The popular systems used include MySQL, PostgreSQL, Oracle, and SQL Server which are very popular because of their reliability, scalability, and high adherence to the SQL standards. These properties are what make SQL databases the best to use when accuracy, consistency and structured data management is needed.

3.2. Features of NoSQL Databases

NoSQL databases are meant to be stored and retrieved with high availability and high scalability of large-scale data, particularly in the cloud. Compared to the traditional relational databases with their high trade-offs of the imposed ACID properties and fixed schemas, the NoSQL databases are schema-less, and a wide range of data types can be stored. The paper identifies three key characteristics Scale-out, which spreads data over a number of machines to bring out scalability and elasticity [18], Flexibility, no fixed schema is needed, and supports dynamic and semi-structured data; and Data Replication, which offers redundancy and fault tolerance, albeit at the expense of immediate consistency. These characteristics render no SQL optimal to work with Big Data in applications with high concurrency such as Google, Amazon and Facebook.

3.3. Need for Hybrid SQL-NoSQL Integration

SQL databases are stronger in providing ACID properties and absolute data integrity, but are weaker in “sparse dimensionality, design rigidity, and support” for data types in a distributed, large-scale data environment. On the other hand, to address this gap, NoSQL databases with BASE properties have arisen, which are scalable and flexible, but do not provide high consistency [19]. Therefore, the authors suggest that “Polyglot Persistence permits a single database to handle a single part of the application and share identical data which are utilized by another database”, and hybrid SQL-NoSQL integration is essential to address various technical and business data requirements indicated in the Figure 3.

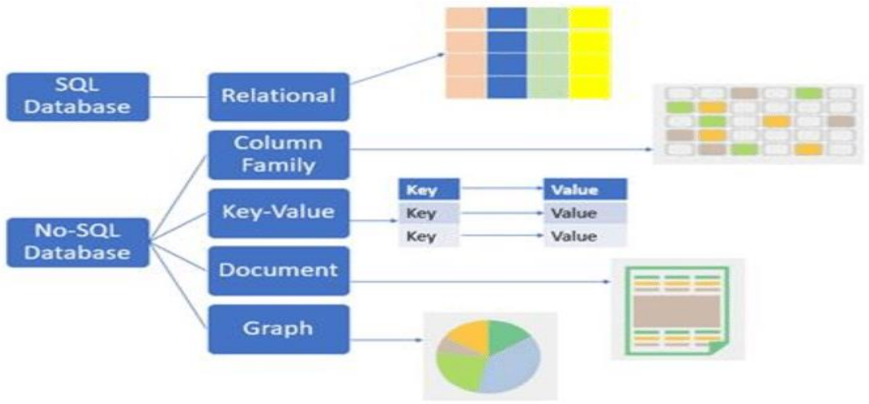

Fig 3: Sql and No-Sql Model.

3.4. Popular Hybrid Cloud Database Platforms

The hybrid cloud database platforms are SQL and NoSQL integrated to offer scalability, consistency and flexibility to global cloud-native applications, distributed transactions, multi-model data, and high availability [20]. Hybrid Cloud Database Platforms, as illustrated in Figure 4.

- **Google Cloud Spanner:** Globally distributed SQL database with horizontal scaling, high consistency and automatic sharding.
- **Microsoft Azure Cosmos DB:** Non-SQL database with support of multiple models with tenable

consistency, low latency and distributed all over the world.

- **Amazon Aurora:** High-performance, fault-tolerant, and auto-scaling database compatible with MySQL/PostgreSQL operation on the AWS cloud.

Fig 4: Logos Of Azure, AWS And Google Cloud Platform.

3.5. Comparative Analysis Of SQL And NoSQL Databases

The Table I comparative the evaluation between the SQL and NoSQL databases points to two opposite paradigms that are influenced by the changing data needs and cloud-native innovations [21].

- **Data Models:** SQL databases have a fixed schema relational structure, which works well for applications requiring sophisticated queries and transactions [22]. However, NoSQL databases are nimbler in data handling and feature flexible data models that can handle semi-structured and unstructured data.
- **Schema Design:** A crucial component of SQL databases, schema design must also be specified. Schema changes are difficult and inconvenient. Because NoSQL databases' schemas may be changed dynamically, they are appropriate for agile development methodologies where needs vary over time.
- **Query Languages:** SQL is the foundation of Structured Query Language (SQL) databases. SQL may be made more robust and standardized to support sophisticated joins and queries. However, NoSQL databases may be built on a proprietary query language to serve their own data model, which may improve efficiency but is not as general as SQL. NoSQL databases do not always employ any standard query language.
- **Performance and Scalability:** SQL databases are often scaled vertically, which might become a bottleneck by adding additional resources to a single server [23]. NoSQL databases are designed to provide horizontal scaling, allowing data to be spread over several servers and effectively managing the strain of large volumes of data at high velocity.
- **Consistency and Availability:** SQL databases adhere to the ACID principles and are more consistent. Cassie the CAP theorem, which states that a distributed system may only offer two of the three properties—consistency, availability, and partition tolerance—is often followed by NoSQL databases. The majority of NoSQL databases prioritize partition tolerance and availability above rigorous consistency, however they do offer eventual consistency.

Table 1: Comparative Analysis of SQL and Nosql Databases

Aspect	SQL Databases	NoSQL Databases
Data Model	Structured and relational model with predefined tables and relationships. Ideal for complex queries and transactional systems.	Flexible data models that can handle unstructured and semi-structured data include document, key-value, column-family, and graph models.
Schema Design	Fixed schema defined upfront; altering schema is complex and may cause downtime.	Dynamic and schema-less design allows agile and iterative development with evolving data structures.
Query Language	Uses Structured Query Language (SQL)—powerful, standardized, and supports complex joins and aggregations.	Uses proprietary or specialized query mechanisms suited to the specific data model (e.g., JSON queries, MapReduce).
Performance and Scalability	Scales vertically by increasing resources (CPU, RAM, storage) on a single server; may face performance bottlenecks at high loads.	Scales horizontally by distributing data across multiple servers, offering superior performance for large-scale and high-velocity workloads.
Consistency and Availability	Ensures ACID (Atomicity, Consistency, Isolation, Durability) compliance for strong consistency and reliability.	According to the CAP theorem, it usually gives availability and partition tolerance precedence over strict consistency, offering eventual consistency instead.

Use Cases	Ideal for applications (like ERP and financial systems) that need integrity, complex transactions, and structured data.	Perfect for social media, IoT, big data, real-time analytics, and content management where scalability and flexibility are essential.

4. Artificial Intelligence in Query Optimization

AI in query optimization leverages machine learning, deep learning, and reinforcement learning to automate cost estimation, adaptive planning, and workload prediction, enabling self-tuning, efficient, and autonomous database management systems.

4.1. Introduction to AI-Driven Database Management

AI-driven DBMSs embed ML and DL across the stack so the system can learn cost models, tune knobs, predict workloads, and adapt plans instead of relying only on handcrafted rules. These systems aim to reduce human tuning, handle non-stationary workloads, and provide end-to-end [24] automation for performance, availability and cost tradeoffs forming the foundation for "self-driving" or autonomous databases.

4.2. Machine Learning Approaches to Query Optimization

The rising level of complexity of modern cloud databases has been a catalyst for the pursuit of the application of ML techniques in query optimization [25]. ML-based methods aim to leverage historical information about query execution to predict the most efficient execution scheme or to better estimate costs.

4.2.1. Supervised Learning Models

Supervised learning has been utilized to infer query execution time, choose the best indexes and based on query costs. In one instance, it has been found that regression models can be used to estimate the time with which a query is going to be executed given some features such as the number of joins, the size of the tables being used and the presence of indexes. These models are trained using past data, and hence they would be able to adapt to particular database settings. Nonetheless, the performance of the same is greatly reliant on the quality and quantity of the training information.

4.2.2. Unsupervised Learning Models

Unsupervised techniques of learning like clustering have been applied to cluster related queries and optimize them together [26]. These models can use the patterns in the query workloads to implement optimization strategies specific to particular query clusters to enhance the overall performance. However, unsupervised methods often require careful tuning and interpretation, which can be challenging in dynamic cloud database environments.

4.3. Reinforcement Learning for Adaptive Query Planning

Reinforcement learning (RL) treats query optimization (e.g., join-order selection) as a sequential decision problem: the agent explores plan choices and receives feedback (execution cost/latency), learning policies that generalize across workloads and improve with experience. Modern work applies Deep RL, [27] Graph Neural Networks and tree-LSTM encodings to represent plans and states; these approaches can adapt to evolving workloads and sometimes outperform classical enumerative heuristics, though they require careful reward design and sample efficiency strategies. See recent RL-for-join-order papers (RTOS, GTDD, PVLDB tutorial).

4.4. Deep Learning Models for Workload Prediction

Deep models (LSTMs, Transformers, and hierarchical CNNs) are used to forecast query arrival patterns, resource demand, and workload composition so the DBMS can pre-provision resources, adapt indexing/caching, or schedule heavy queries [28]. Systems like Microsoft's Sibyl and Alibaba's AWM show how learned forecasters and pattern miners predict future query templates and volumes to enable proactive tuning and autoscaling. These models help with SLA preservation and cost-aware provisioning in cloud environments.

4.5. Challenges of Query Optimization

The current development, however, has certain limitations on how to make queries optimal on big data workloads in cloud-enabled distributed databases:

- **Scalability:** The current optimization methods can be quite ineffective when dealing with large volumes of data and highly complex queries [29].
- **Dynamic Adaptability:** The conventional query optimizers do not have real-time flexibility to adapt to dynamic patterns of resource availability and workload in the cloud.
- **Cost-Effectiveness:** There is still an open question of the ability to balance between performance gains and cost-efficiency in cloud-based systems.
- **Integration of ML Techniques:** Although potential, ML based query optimization methods should be further developed in order to deal with real-life deployment challenges.

The literature review highlights the significance of the creation of superior query optimization strategies in accordance with the specific requirements of cloud-based derived databases and workloads of big data.

5. Literature Review

The query optimization problem in distributed databases and cloud-computing has been of the meaningful focus over the last decade as a result of growing demand to handle big data workloads efficiently. Query Optimization in Table II comparative literature review.

Spoiala, Barabas and Gal (2025) presents some techniques using MySQL Workbench and Amazon AWS RDS service applied for creating and querying MySQL databases. In this context, it is shown a way to querying a DB instance from a EC2 instance with Ubuntu virtual machine with SQL commands. For the practical implementation of the database MySQL Workbench was used [30].

Bandla (2025) explores the integration of generative AI systems with distributed cloud databases to solve database query optimization challenges while managing operational workloads and making fault predictions and ensuring data consistency initiatives. Through research on generative AI's predictive functions and generative capabilities, formulate innovative techniques that connect automated schema design and adaptive indexing with intelligent data partitioning approaches [31].

Sakib et al. (2025) offers an analysis of NoSQL document store databases focusing on MongoDB, Couchbase, and CouchDB and includes an integrated Proof of Concept (POC) along with a research-based comparison. It starts by giving a general summary of the dynamic nature of the Database Management Systems (DBMS) in emphasizing the importance of the NoSQL databases in addressing the shortcomings of the traditional relational databases. The study involves a contextual analysis and qualitative comparison of the three databases that have performance measures to determine their appropriateness in different applications [32].

Dong et al. (2024) provide a thorough introduction to cloud-native databases. Examine and compile the most recent developments in cloud-native OLTP and OLAP databases, respectively. Part one discusses the three types of cloud-native OLTP database architecture and then goes on to discuss their main strategies, including HTAP optimization, multi-layer recovery, storage layer consistency, compute layer consistency, and data placement strategy. Two categories of cloud-native OLAP databases are described in the second section [33].

Chai and Qin (2024) describes a case study to find a most effective approach to the enhancement of the application performance when using massive and continuously growing datasets. To do this, the performance of the SQL and NoSQL systems in terms of query is compared. The comparison conducted by comparing the run times of the queries run on SQL database and also in the NoSQL database which are in the cloud platform and also using the TPC benchmark. The experiment determines the response time of both types of database and eventually gives an indication of the best way to operate [34].

Olariu (2023) addressed serverless frameworks, namely Amazon Web Services (AWS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). To simplify administration and lessen infrastructure duties, it is advised to first "lift and shift" utilizing IaaS before gradually implementing PaaS components like AWS Elastic Beanstalk or AWS Lambda. Additionally, some AWS-specific cost optimization techniques were investigated. Cost reduction strategies that were found to be successful included using Reserved Instances, optimizing Windows Server and SQL Server licensing, taking into account PostgreSQL as a substitute for SQL Server, and utilizing Linux-based applications and web servers on AWS [35].

Dundjerski and Tomasevic (2022) The work described in this paper focuses on using real-world data to develop an autonomous database troubleshooting system that does root cause analysis by utilizing an expert system and more extensive statistical data science models. Over the course of eight months, a comprehensive assessment study was carried out using a wide range of Azure SQL production workloads for different numbers of databases [36].

Capris et al. (2022) Relational (SQL) and non-relational (NoSQL) data structure interactions are the most crucial factors to take into account when choosing a database. Although all of the solutions are good, customers should consider certain differences before making a decision. Due to their vertical scalability, SQL databases can usually scale server components such as CPU, RAM, and SSD. The horizontal scalability feature is supported by NoSQL databases, however. The NoSQL database's capacity may therefore be increased by adding more servers or by fragmenting (data partitioning) [37].

Table 2: Comparative Literature Review, Query Optimization Techniques for Cloud Databases

Reference	Focus Area	Key Findings	Challenges	Key Contribution	Limitations / Gap
Spoiala, Barabas and Gal (2025)	Database deployment using MySQL Workbench and AWS RDS	Demonstrated practical creation and querying of MySQL databases via AWS RDS and EC2 Ubuntu instances using SQL commands.	Managing secure connectivity between AWS RDS and EC2 instances; ensuring performance consistency.	Provided an applied approach for configuring, querying, and integrating MySQL in a cloud-based environment using MySQL Workbench.	Limited scalability analysis and lack of performance benchmarking between local and cloud-hosted databases.
Bandla (2025)	Generative AI and distributed cloud database optimization	Introduced generative AI-based methods for query optimization, workload prediction, and data consistency management.	Integrating AI models with live database workloads while maintaining real-time response efficiency.	Proposed adaptive schema design and intelligent partitioning through generative AI systems for improved cloud database management.	Lack of experimental validation or comparison with existing optimization algorithms.

Sakib et al. (2025)	Comparative analysis of NoSQL databases (MongoDB, Couchbase, CouchDB)	Conducted proof-of-concept performance evaluation highlighting NoSQL's efficiency for unstructured data.	Balancing scalability with consistency in NoSQL architectures; ensuring fair benchmarking conditions.	Provided a qualitative and quantitative performance comparison to assess NoSQL suitability for various use cases.	Study limited to three NoSQL systems without inclusion of hybrid or NewSQL systems.
Dong et al. (2024)	Cloud-native OLTP and OLAP database architectures	Provided a comprehensive survey of cloud-native database systems and their architectural designs and optimization techniques.	Complexity in integrating OLTP and OLAP layers for hybrid workloads; maintaining consistency across distributed storage layers.	Classified and summarized architectural trends and technical advancements in cloud-native databases.	Theoretical survey lacking implementation or comparative performance evaluation.
Chai and Qin (2024)	SQL vs. NoSQL performance in cloud environments	Compared runtime performance using TPC benchmarks; demonstrated which system performs better for large-scale datasets.	Ensuring uniform experimental conditions across SQL and NoSQL systems.	Offered empirical evidence on performance trade-offs between SQL and NoSQL for scalable cloud-hosted datasets.	Limited scope focusing mainly on query performance; lacks insights into cost and maintenance overheads.
Olariu (2023)	Cloud service models and cost optimization in AWS	Recommended transition from IaaS to PaaS/Serverless for simplified management; discussed cost-saving strategies.	Managing migration complexity and compatibility across service layers.	Proposed a structured roadmap for AWS adoption and effective cost optimization using hybrid service models.	Lacks exploration of multi-cloud or hybrid cloud cost optimization frameworks.
Dundjerski and Tomasevic (2022)	Automated database troubleshooting using data science models	Developed a root cause analysis system combining expert systems and statistical models on Azure SQL workloads.	Handling diverse workload variations and ensuring accurate model generalization.	Introduced an automated troubleshooting framework validated over real-world production data.	System confined to Azure SQL workloads; lacks applicability to other cloud database environments.
Capris et al. (2022)	SQL vs. NoSQL scalability and structure comparison	Highlighted differences in scalability—vertical (SQL) vs. horizontal (NoSQL)—and discussed data partitioning benefits.	Maintaining data integrity across horizontally scaled NoSQL systems.	Provided a conceptual framework for choosing between SQL and NoSQL based on scalability and application needs.	The study is conceptual without empirical validation or performance experiments.

6. Conclusion and Future Work

An innovative advancement in cloud database administration is the combination of AI and ML in query optimization. Conventional optimization methods rule-based, cost-based, and heuristic are fundamental and fail to keep abreast of the dynamic and heterogeneous manner of cloud and hybrid SQL-NoSQL environments. These limitations are mitigated by AI-based methods which inject the query optimization process with data-driven flexibility, a perpetual learning, and predictive intelligence. Modern optimizers are able to autonomously operate by altering execution plans, making workload predictions, and optimizing resources through supervised, unsupervised learning models, and reinforcement learning models in real-time. The comparative study of SQL and NoSQL systems highlights the fact that, although SQL is a consistent and structured system with high reliability; NoSQL is scalable and flexible, which requires the integration of both systems to address the changing needs of big data applications. In this type of complex ecosystem, query optimization with the help of AI is relevant to efficient execution in distributed systems, reduces latency, and remains cost-effective by embracing such capabilities as autoscaling, serverless processing, and geo-distributed optimization. The future research work should be on making interpretable, cost efficient, and energy efficient AI models that can work with multi-cloud and hybrid database systems. Additionally, integrating explainable AI and federated learning into query optimization could enhance transparency, security, and performance tuning.

References

[1] A. Bachhav, V. Kharat, and M. Shelar, “Query Optimization for Databases in Cloud Environment: A Survey,” *Int. J. Database Theory Appl.*, vol. 10, no. 6, pp. 1–12, Jun. 2017, doi: 10.14257/ijdta.2017.10.6.01.

[2] O. Oloruntoba, “Architecting Resilient Multi-Cloud Database Systems: Distributed Ledger Technology, Fault Tolerance, and Cross-Platform Synchronization,” *Int. J. Res. Publ. Rev.*, vol. 6, no. 2, 2025, doi: 10.55248/gengpi.6.0225.0918.

[3] V. N. R. Dantuluri, “AI-Powered Query Optimization in Multitenant Database Systems,” *J. Comput. Sci. Technol. Stud.*, vol. 7, no. 4, pp. 802–813, 2025, doi: 10.32996/jcsts.2025.7.4.93.

[4] B. R. Ande, “Enhancing Cloud-Native AEM Deployments Using Kubernetes and Azure DevOps,” *Int. J. Commun. Networks Inf. Secur.*, vol. 15, no. 8, pp. 33–41, 2023.

[5] O. Oloruntoba, “AI-Driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise it environments,” *World J. Adv. Res. Rev.*, vol. 25, no. 2, pp. 1558–1580, Feb. 2025, doi: 10.30574/wjarr.2025.25.2.0534.

[6] S. K. Mamillapalli and R. D. Jeganathan, “Mastering Cloud-Native Performance : Strategies for Optimization,” vol. 3, no. 3, pp. 1–9, 2022.

[7] V. Panwar, “AI-Driven Query Optimization : Revolutionizing Database Performance and Efficiency,” vol. 72, no. 3, pp. 18–26, 2024.

[8] M. R. R. Deva and N. Jain, “Utilizing Azure Automated Machine Learning and XGBoost for Predicting Cloud Resource Utilization in Enterprise Environments,” in *2025 International Conference on Networks and Cryptology (NETCRYPT)*, 2025, pp. 535–540. doi: 10.1109/NETCRYPT65877.2025.11102235.

[9] A. Bachhav, V. Kharat, and M. Shelar, “An Efficient Query Optimizer with Materialized Intermediate Views in Distributed and Cloud Environment,” *Teh. Glas.*, vol. 15, no. 1, pp. 105–111, Mar. 2021, doi: 10.31803/tg-20210205094356.

[10] S. Garg, “Predictive Analytics and Auto Remediation using Artificial Inteligence and Machine learning in Cloud Computing Operations,” *SSRN Electron. J.*, vol. 7, no. 2, 2025, doi: 10.2139/ssrn.5267117.

[11] A. Thirunagalingam and S. Banala, “Enhancing Query Optimization in Cloud-Native Relational Databases : Leveraging Policy Gradient Methods for Intelligent Automation,” *Int. J. Intell. Syst. Appl. Eng. Syst. Appl. Eng.*, vol. 12, no. 23s, pp. 1026–1035, 2024.

[12] B. R. Cherukuri, “Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications,” *Int. J. Sci. Res. Arch.*, vol. 13, no. 1, pp. 3302–3315, Oct. 2024, doi: 10.30574/ijsra.2024.13.1.2035.

[13] R. Marcus and O. Papaemmanouil, “Plan-Structured Deep Neural Network Models for Query Performance Prediction,” *arXiv*, 2019, doi: 10.48550/arXiv.1902.00132.

[14] Abhayanand and M. M. Rahman, “Enhancing Query Optimization in Distributed Relational Databases: A Comprehensive Review,” *Int. J. Nov. Res. Dev.*, vol. 9, no. 3, p. 13, 2024.

[15] G. Maddali, “An Efficient Bio-Inspired Optimization Framework for Scalable Task Scheduling in Cloud Computing Environments,” *Int. J. Curr. Eng. Technol.*, vol. 15, no. 3, 2025.

[16] 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.

[17] Y. Jani, “The Role Of Sql And Nosql Databases In Modern Data Architectures,” 2021.

[18] B. Sethi, S. Mishra, and P. ku. Patnaik, “A Study of NoSQL Database,” vol. 3, no. 4, pp. 1131–1135, 2014.

[19] P. P. Khine and Z. Wang, “A Review of Polyglot Persistence in he Big Data World,” *Information*, vol. 10, no. 4, 2019, doi: 10.3390/info10040141.

[20] D. Sikeridis, I. Papapanagiotou, B. P. Rimal, and M. Devetsikiotis, “A Comparative Taxonomy and Survey of Public Cloud Infrastructure Vendors,” pp. 1–21, 2017, doi: 10.48550/arXiv.1710.01476.

[21] Y. Jani, “The Role Of Sql And Nosql Databases In Modern Data Architectures,” *Int. J. Core Eng. Manag.*, vol. 6, no. 12, pp. 61–67, 2024.

[22] A. R. Duggasani, “Scalable and Optimized Load Balancing in Cloud Systems: Intelligent Nature-Inspired Evolutionary Approach,” *Int. J. Innov. Sci. Res. Technol.*, vol. 10, no. 5, May 2025, doi: 10.38124/ijisrt/25may1290.

[23] 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.

[24] X. Zhou, C. Chai, G. Li, and J. Sun, “Database Meets AI : A Survey,” 2023.

[25] R. Heinrich, X. Li, M. Luthra, and Z. Kaoudi, “Learned Cost Models for Query Optimization : From Batch to Streaming Systems,” vol. 18, no. 12, pp. 5482–5487, 2025, doi: 10.14778/3750601.3750699.

[26] V. M. L. G. Nerella, “Architecting Secure, Automated Multi-Cloud Database Platforms Strategies for Scalable Compliance.,” *Int. J. Intell. Syst. Appl. Eng.*, vol. 9, pp. 128–138, 2021.

[27] K. Tzoumas, T. Sellis, and C. S. Jensen, “A Reinforcement Learning Approach for Adaptive Query Processing,” 2008.

[28] S. Karimunnisa and Y. Pachipala, “Deep Learning Approach for Workload Prediction and Balancing in Cloud Computing,” vol. 15, no. 4, pp. 754–763, 2024.

[29] C. Bandla, “Query Optimization for Big Data Workloads in Cloud-Enabled Distributed Databases,” *Int. J. Sci. Res.*, vol. 12, no. 8, pp. 2576–2580, Aug. 2023, doi: 10.21275/SR23084171047.

[30] D. C. Spoiala, T. Barabas, and T. O. Gal, “Techniques for Creating and Querying Relational Databases Using MySQL and AWS Cloud Services,” in *2025 18th International Conference on Engineering of Modern*

Electric Systems (EMES), IEEE, May 2025, pp. 1–4. doi: 10.1109/EMES65692.2025.11045576.

[31] C. Bandla, "Leveraging Generative AI for Enhanced Scalability and Efficiency in Distributed Cloud Databases," in *2025 8th International Symposium on Big Data and Applied Statistics (ISBDAS)*, IEEE, Feb. 2025, pp. 280–286. doi: 10.1109/ISBDAS64762.2025.11116985.

[32] F. F. Sakib, S. D. Roy, Z. Rahman, S. Saha, and A. Salam, "NoSQL Database Selection Process: An Integrated Proof of Concept and Comparison for Targeted Document Store Databases," in *2025 4th International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, IEEE, Jan. 2025, pp. 151–155. doi: 10.1109/ICREST63960.2025.10914487.

[33] H. Dong, C. Zhang, G. Li, and H. Zhang, "Cloud-Native Databases: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 12, pp. 7772–7791, Dec. 2024, doi: 10.1109/TKDE.2024.3397508.

[34] S. Chai and Z. Qin, "A Case Study of Cloud Query Performance Comparison Between SQL and NoSQL Database," in *2024 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2024, pp. 716–721. doi: 10.1109/WI-IAT62293.2024.00117.

[35] F. Olariu, "Overcoming Challenges in Migrating Modular Monolith from On-Premises to AWS Cloud," in *2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet)*, IEEE, Sep. 2023, pp. 1–6. doi: 10.1109/RoEduNet60162.2023.10274946.

[36] D. Dundjerski and M. Tomasevic, "Automatic Database Troubleshooting of Azure SQL Databases," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1604–1619, Jul. 2022, doi: 10.1109/TCC.2020.3007016.

[37] T. Capris, P. Melo, N. M. Garcia, I. M. Pires, and E. Zdravevski, "Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation," *2022 Int. Conf. Data Anal. Bus. Ind. ICDABI 2022*, no. February, pp. 214–218, 2022, doi: 10.1109/ICDABI56818.2022.10041513.