International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Benchmarking SQL Query Optimization Techniques Across Major RDBMS Platforms

Prateek Panigrahy

KIIT University, Bhubaneswar, Odisha

prateek.panigrahy@gmail.com

Abstract

SQL query optimization is central to RDBMS performance, impacting responsiveness, scalability, and resource efficiency. This study benchmarks optimization techniques across Oracle, SQL Server, PostgreSQL, MySQL, and IBM Db2, beginning with theoretical foundations such as cost models, equivalence rules, and statistical usage. It then examines platform-specific strategies, including adaptive query execution, advanced indexing, and machine learning-driven optimization. Using TPC-H and TPC-DS benchmarks, the analysis evaluates optimizer performance under standardized workloads, revealing both shared reliance on cost-based optimization and unique innovations per system. Emerging trends workload-aware cloud optimization, hybrid transactional/analytical processing, and energy-efficient query planning, underscore the evolving landscape, while benchmarking remains essential for system selection, tuning, and guiding research advancements.

Keywords: SQL Query Optimization; Relational Database Management Systems (RDBMS); Benchmarking Frameworks; Cost-based Optimization; Adaptive Query Execution

1. Introduction

The Relational Database Management Systems (RDBMS) remain to process enterprise data at its core, which is the force behind transactional and analytical applications in the industry. The capability of the RDBMS platforms to optimize the queries is critical to the performance of the platforms, as the time an operation takes to run the query can be the difference between the resource utilization and the usability of the system. In the current times, SQL query optimization is one of the largest research and engineering problems in data management. Not only does it aim at the minimization of the execution time, but it also balances the input/output overhead, memory consumed, and parallelism strategies. The other RDBMS systems, like the Oracle Database, Microsoft SQL Server, MySQL, PostgreSQL, and IBM Db2, have developed different philosophies of query optimization, because of the different philosophies of design and performance tuning [1], [2].

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
SQL query optimization in these platforms should also be benchmarked in order to be aware of their relative performance, their strong points, and trade-offs. Benchmarking would allow researchers and practitioners to evaluate the performance of their systems when presented with comparable workloads, to conclude how well query engines can be scaled, and to conclude how query planners can be optimized. The growing demand for real-time analytics, combined with the need to process large volumes of data, highlights the importance of evaluating platform capabilities. It is essential to assess how different systems handle complex workloads, whether in transactional OLTP environments or in analytical OLAP queries [3], [4]. The given benchmarking practice is not just an academic activity, but a practice that has a practical implication. The selection of RDBMS systems to be employed in mission-critical workloads is important, and query optimization can be dramatically successful in influencing the performance outcomes, cost structure, and manageability. As the complexity of the datasets grows and grows exponentially and with the introduction of a hybrid architecture (distributed RDBMS and cloud-based databases) together with the hybrid optimization of query strategies, the adjustment of query optimization strategies also becomes a topical issue [5]. Besides, the advancements in machine learning and adaptive query optimization are transforming the traditional paradigms, and it is important to place the modern benchmarking study in a futuristic perspective [6].

In order to come up with a comprehensive image of the benchmarking methods applied to facilitate the optimization of SQL queries, this paper will analyze the theoretical foundation of query optimization and the central role that it plays in the success of the database [7, 8]. It goes further to talk about the optimization techniques that have been applied on the major RDBMS platforms, and the considerate approach by the providers. The second section is the discussion of benchmarking frameworks, workloads, and evaluation methodologies that have been immensely employed in comparative studies. The article then goes ahead to summarize the findings on the previous empirical standards that have presented an insight into the platform-based strengths. Finally, the paper also retrospectively examines the new fashions and the new surroundings as regards query optimization in the newer RDBMS entities. By relating these sections, the discussion will be formulated according to underlying principles to applied comparisons that will ensure that each transition helps the reader in uncovering the concept of SQL query optimization benchmarking. The second part gives the contextual background based on the re-exploration of the theoretical background of query optimization and its role in the greater context of RDBMS.

2. Theoretical Principles of Query Optimization

Since it has already been introduced, where the need to benchmark was presented, it needs to be initially brought to a theoretical level of optimization of SQL queries. Query optimization is a procedure of determining the most efficient plan for running a specific SQL statement. This comes through high-level declarative queries to low-level strategies of implementation to be executed in physical hardware at minimal resource

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
consumption [9]. At the simplest level, query optimization involves the cost models that approximate the cost of the different execution plans. The distributed environments commonly consider the disk I/O, the CPU cycles, and memory usage, and the network costs used in such models. The optimizer assumes these estimates to select the plan that is least apt to cost a given amount. The problem, however, is that, as is natural to the nature of workloads, data distributions, and runtime conditions, query optimization is computationally intensive and approximation-driven in nature [10].

One of the most significant works of theory when it comes to query optimization research is the notion of the equivalence rules. These rules allow the optimizer to rephrase a query into multiple logically equivalent expressions, which makes the set of possible execution plans larger. An illustration is that one can use join commutativity and associativity to convert join operations with radically different execution costs, which can be rearranged by the optimizer. The optimizer is then left to search exhaustively or heuristically through this plan space in order to reach an acceptable solution [11]. This plan space has long been explored through dynamic programming techniques that scale exponentially with query size and are thus more complex. Hence, the present-day optimizers are a combination of randomized with pruning algorithms, combined with heuristics, as a complexity control mechanism. Additionally, advanced RDBMS systems possess adaptive query optimization founded on dynamically altering the execution plans based on dynamically evolving runtime feedback, in which cost estimates are ineffective [12].

Another point that has been highlighted in the theoretical framework is that of statistics in query optimization. Histograms, sampling techniques, and statistical summaries of the distributions of data are used to estimate selectivity, join cardinalities, and filter effectiveness. Such errors in estimates may result in inefficient query plans, and this is the reason why, in recent studies, machine learning models capable of improving the accuracy of the prediction were in focus [13]. Though the conceptualization of query optimization has already been quite developed, its application using actual interfaces is unevenly distributed throughout the RDBMS systems. System properties of accuracy, speed, and scalability are traded among various systems depending on the design priorities and workloads. These bases were further elaborated in the next paragraph, wherein the practice of query optimization by leading vendors of RDBMS is discussed as a means of laying the groundwork for benchmarking.

3. Query Optimization Approaches in Major RDBMS Platforms

Quitting the theoretical aspect of query optimization, one now has to examine the way the different RDBMS platforms are putting these theories into practice. Each of the systems has come up with its own way of query optimization, and this is dependent on

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
how the system is built and the type of workloads that the system is intended to support.

Large-scale RDBMS systems employ different optimization strategies based on the nature of the workload and the specific capabilities of the system. Oracle database includes a cost-based optimizer (CBO), which includes extensive statistics and dynamic changes of plans during execution [14, 15, 16]. Microsoft SQL Server, as well, is cost-based optimized but concentrates on query hints, plan guide, adaptive joins, and advanced indexing, like column store indexes of analytic loads. In more modern forms, cost estimation rule-based strategies were common in PostgreSQL, and the extent to which they could be extended was considerable, as well as their transparency and increased parallel query support. MySQL is a compromise between rule and cost-based options and excels at simple queries of OLTP but poorly on complex analytic loads, although recent additions to the join strategy give it tuning capabilities [17, 18]. To facilitate the selectivity estimation and the query performance, IBM Db2 integrates the cost-based optimization, machine learning, and advanced machine learning in terms of reordering joins, parallelization, and statistical views.

These variances relax the specialization of strategies on which RDBMS platforms cope with the optimization dilemma. These strategies are especially relevant to the implications when the systems are put before benchmarking of the standardized workloads. As the practical optimization methods have been established on the platforms, the second part will explain the methodologies that are used to benchmark optimization of SQL queries in a controlled environment.

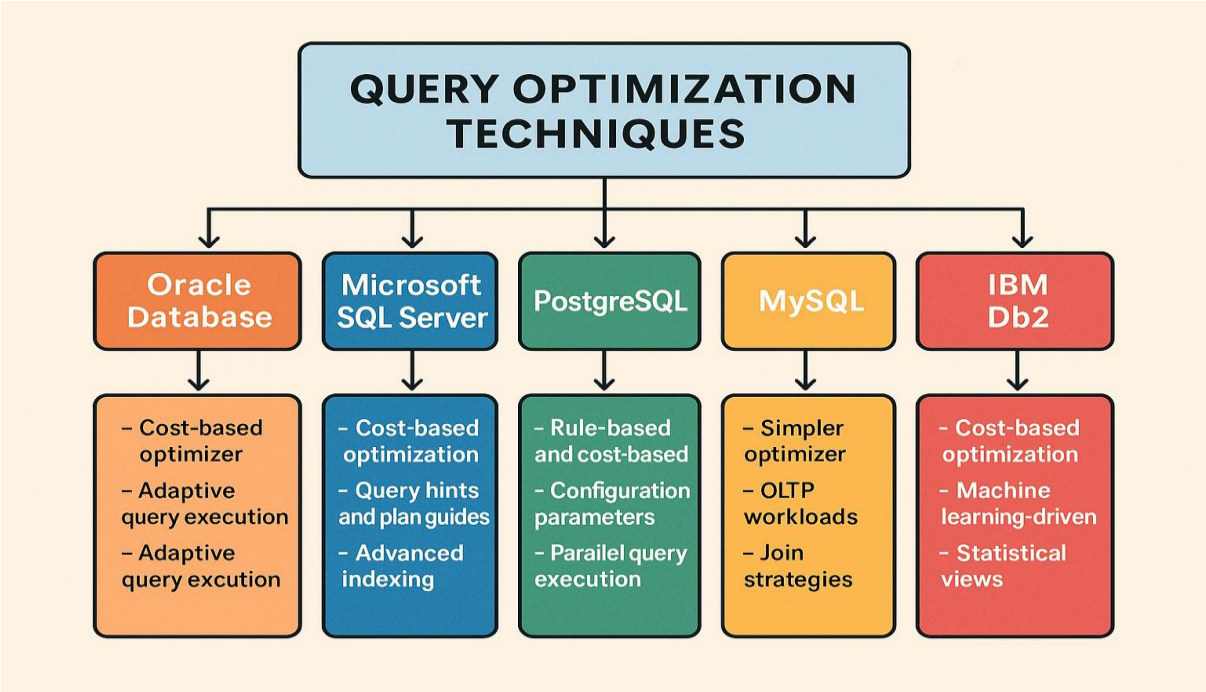

Figure 1: Comparison of query optimization techniques across major RDBMS platforms

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4. Benchmarking Frameworks and Workloads

Benchmarking can be used to a great extent in optimization of SQL queries on various RDBMS systems to provide insights about the performance when the tables are under load. It can be done in an extremely wide range of common frameworks [19, 20]. oLTP workload TPC-C measures the load of OLTP transactions, which are the normal workload; TPC-H and TPC-DS transactions measure the decision support workload and the analytical workload with advanced queries, joins, and aggregations. These benchmarks, coupled with scale factors that scaled the sizes, assisted in comprehending the reordering process of the optimizers, cardinality estimation, indexing decisions and parallelization. By the emphasis on the deviation that entails the systems, e.g. MySQL, whose performance is increasing in direct proportion to the data size, benchmarking will be able to provide a useful, relative value of the effectiveness of optimizers.

The other synthetic and real loads, which are not the standardized loads, are also used in the comparative studies. The synthetic benchmarking allows the researcher to disaggregate part of the query optimization, such as the sequence of joins or the use of indices [21, 22]. However, the application-specific patterns also do not fare well in being represented in the synthetic queries that are offered in the real-world workload, however. It is this which guarantees the generalizability, to say the least, contextual relevance [23]. Benchmarking methodologies also emphasize on repeatability and transparency. The queries must be executed using the identical hardware, software, and setup to ensure that the variations in the performance are the outcome of an optimizer conduct and do not reflect the variability of the environment. The measures were mostly query response time, throughput, CPU usage, memory usage as well and plan stability. In academic studies, energy efficiency has recently become one of the benchmarking metrics of determination, as witnessed in the recent formulation of sustainable computing [24].

The other methodological issue is query caching and reuse of plans. Considering several RDBMS systems maintain the previously generated execution plans within the cache, benchmark systems must consider the possibility of query performance either based on the new optimization or the old execution. This is an important distinction, where optimizers might be very good at one-time queries but not in adaptive queries, where query parameters and data distributions change on a regular basis [25]. Benchmarking can also be used as a diagnostic measure to expose optimizer misestimates, inefficient join strategies or resource bottlenecks. Indicatively, in a comparison of Oracle and SQL Server regarding TPC-DS, research has indicated that both systems can come up with efficient plans to facilitate aggregations, but the Oracle optimizer is more effective in cases where correlated subqueries are used, as opposed to SQL Server, which, in most cases, will not be able to properly estimate cardinality [26]. This kind of knowledge aids administrators in tuning their systems, and it also gives the researchers some advice on areas where the optimizers are weak and where further improvement should be made. Having benchmarking frameworks and

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
workloads as a key base, the discussion can now shift to comparative results. An analysis of the empirical results of benchmark experiments among RDBMS systems can help understand how the theoretical and practical methods of optimization mentioned above can be converted into practical performance differences.

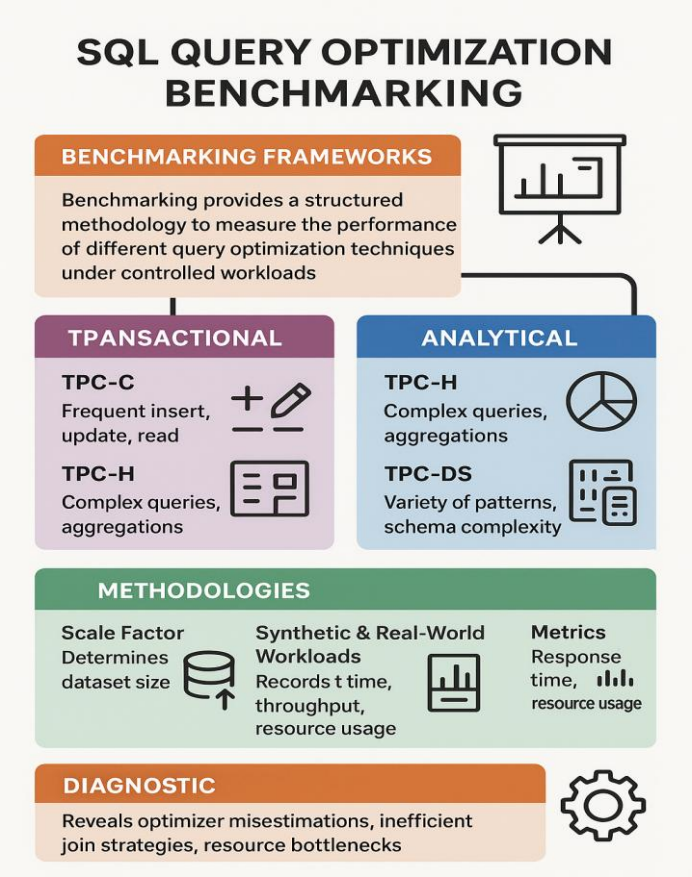

Figure 2: A visual summary of SQL query optimization benchmarking frameworks

5. Comparative Results of SQL Query Optimization Benchmarking

Based on the concept of benchmarking frameworks and workloads, it is now necessary to assess the performance of the various RDBMS platforms during intensive benchmark tests. Comparative performance provides useful information about the strengths and weaknesses of query optimization methods, pointing out the areas of strong performance and areas of weakness in the platforms. The Oracle Database has always performed well in query optimization, both in the TPC-H and TPC-DS workloads. Its optimizer is based on cost, and it is combined with adaptive query execution, which enables Oracle to produce efficient plans for complex analytical queries. On multi-way join and aggregation benchmark queries, Oracle frequently presents reduced response times up to competitors, particularly on greater magnitude parameters. It is also characterized by the fact that its ability to adapt execution plans on the fly results in more stability in its performance, which minimizes the risk of catastrophic plan misestimation [27].

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Microsoft SQL Server provides good performance in the analytical load, and when advanced indexing, such as the column store indexes, is used, the I/O is minimized and allowing the use of vectors. It is also good at star-schema queries and when it needs to aggregate, but sometimes poor cardinality estimates cause poor join ordering [28]. Its plan caching and reuse features help to produce consistent performance in mixed OLTP-OLAP. Recent versions of PostgreSQL have been altered more positively, and parallel query execution has enhanced performance on full-table scan and aggregation-intensive TPC-H queries. Nevertheless, it remains slower than Oracle and SQL Server in performing complex joins and subqueries based on cost estimation and adaptive optimization constraints, but it has the advantage of open-source extensibility and so remains popular in academic and experimental research [29].

MySQL is mostly optimized towards OLTP loads, and it scales well on small transaction loads. But this query optimizer can have a hard time with complicated analytical problems such as TPC-DS, primarily because of poor join reordering and the employment of less sophisticated cost models. Recent advances in join algorithms and indexing have been making incremental improvements but large-scale analytical workloads still present much challenge [30]. Conversely, IBM Db2 competes directly with Oracle on decision-support workloads with the use of statistical views to provide precise cardinality estimates and machine learning-informed optimization to support a wide range of queries with heavy aggregations and enormous joins.

Table 1: Comparative Summary of SQL Query Optimization Performance Across RDBMS Platforms

RDBMS Platform	**Strengths in Query Optimization**	**Limitations under Benchmarks**	**Distinctive Features Observed**
Oracle Database	Consistently efficient execution plans in analytical benchmarks (TPC-H, TPC-DS); strong adaptive query execution	Complexity in tuning; high resource overhead for very large workloads	Adaptive joins, dynamic plan re-optimization
Microsoft SQL Server	Excellent use of column store indexes; stable performance with plan caching in repetitive workloads	Occasional inaccuracies in cardinality estimation lead to suboptimal joins	Advanced indexing strategies, plan guides
PostgreSQL	Significant gains with parallel query execution in recent versions; flexible extensibility	Struggles with large join reordering and correlated subqueries	Open-source adaptability, user-tuned configuration

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

RDBMS Platform	Strengths in Query Optimization	Limitations under Benchmarks	Distinctive Features Observed
MySQL	Fast performance on small-scale OLTP workloads; efficient for simpler queries	Limited performance on large analytical queries; basic cost models	Lightweight optimizer, incremental improvements
IBM Db2	Highly accurate selectivity estimates using statistical views; strong performance in aggregation-heavy queries	Requires expert configuration; higher complexity in deployment	Machine learning–driven optimization, advanced join reordering

6. Emerging Trends and Future Directions in SQL Query Optimization

The new trends of optimization of SQL queries are leaning more towards the ancient cost-based optimization of the query to the intelligent, adaptive and context-based optimization. These systems have a chance to enhance the performance and minimize the estimation errors by using advanced methods. They can be more responsive to the dynamic and changing environment, enabling machine learning, adaptive execution and workload-aware tuning. Variability of resources, distribution, and heterogeneous workloads of data are other issues caused by cloud, hybrid transactional/analytical (HTAP), distributed, and federated platforms. The new hardware technologies, such as the utilization of in-memory databases and in-memory utilization of GPU acceleration do also impact the optimization strategies. All in all, query optimization in the present is gradually transforming into a blend of the old approaches with the adaptive and automated approaches and hardware-centric approaches to address the dynamic and multidimensional information environment in the most efficient way possible.

Table 2: Emerging trends and future directions

Trend	Description	Key Examples / Systems	Benefits / Implications
Machine Learning–Driven Optimization	Uses historical execution data to predict cardinalities and costs more accurately than static models	IBM Db2, Microsoft SQL Server	Reduces plan misestimations; supports self-tuning systems

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Adaptive Query Optimization	Adjusts execution plans at runtime based on observed deviations	Oracle Database	Mitigates performance degradation in dynamic workloads
Workload-Aware Optimization in Cloud	Considers cloud-specific factors like multi-tenancy, elastic scaling, and cost-efficiency	Cloud-based RDBMS (AWS RDS, Azure SQL)	Balances latency with resource costs; improves performance in cloud environments
Hybrid Transactional/Analytical Processing (HTAP)	Optimizes for both OLTP and OLAP workloads simultaneously	HTAP-enabled platforms	Efficiently handles concurrent transactional and analytical queries
Workload-Driven Automated Tuning	Automates index creation, query rewriting, and resource allocation	Self-managing RDBMS features	Reduces DBA burden; improves query performance automatically
Distributed & Federated Optimization	Decomposes queries across multiple nodes or heterogeneous data sources	Distributed RDBMS, Federated DBs	Optimizes data locality, partitioning, and communication costs
Energy-Efficient Optimization	Considers power consumption alongside traditional performance metrics	Research prototypes	Balances execution time with sustainability goals
Hardware-Aware Optimization	Leverages in-memory databases, GPUs, and non-volatile memory for query execution	GPU-accelerated joins, In-memory DBs	Maximizes hardware potential; reduces execution time for large-scale analytics

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
The future of SQL query optimization is moving audaciously to the systems which are autonomous, context-aware and able to continue improving themselves. Such innovations as self-healing optimizers will dynamically detect the variations in schema or workload, or bad execution plans and will automatically fix them. This will be enabled by cross-linguistic embeddings that will support multilingual query understandings, where multilingual enterprises can normalize and optimize workloads in multilingual environments. The more intricate workload behaviour will be automated by the generative AI models, the more intricate tasks will be summarised, performance reports for the executives, and query rewrites. Cloud-native architectures will also improve optimizers to be cost-conscious, to combine performance and financial efficiency. All this makes the optimization of queries a clever, strong and active component of the current data infrastructure.

7. Metrics & Benchmarks for Evaluating Pipeline Success

In order to assess the effectiveness of the existing data processing and optimization pipelines, there should be evident metrics to assess the performance, reliability, adaptability, and semantic strength. Traditional measures such as latency and throughput remain key; however, new work is being performed that necessitates additional measures to measure the quality of system with schema drift, multi-lingual input, and flexible query patterns. Measures should also show how transformations are correct, how well the execution plan is stable, in addition to the system has the capacity to maintain its performance under various data distributions. When the KPIs are standardized, companies can compare pipelines in various settings, compare the performance of the tools, and measure the benefits of AI-based optimization. The following table is a list of the key metrics that are more commonly applied in benchmarking complex data engineering and query optimization systems.

Table 3: Key Metrics and Benchmarks for Pipeline Success

Metric / KPI	Description	Benchmark Method	Relevance
Pipeline Latency	Measures end-to-end processing time for data ingestion, transformation, and query execution.	Time-to-process (TTP) tests under controlled workloads.	Critical for real-time analytics and operational decision-making.
Query Accuracy / Transformation Fidelity	Evaluates the correctness of normalization,	Ground-truth comparison and validation sampling.	Ensures reliability of downstream analytics and ML models.

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	enrichment, and schema mapping.		
Schema Drift Tolerance	Assesses the pipeline's ability to adapt to evolving schemas without failure or manual intervention.	Simulated schema-change stress testing.	Essential for dynamic, multi-source environments.
Multilingual Coverage	Measures how effectively the system handles multilingual text, metadata, and semantic normalization.	Cross-language test sets and LLM-assisted evaluations.	Important for global enterprises with diverse data sources.
Plan Stability	Gauges the consistency of query execution plans across changing workloads.	Repeated execution under varied parameters.	Prevents performance regressions and unpredictable latency spikes.
Throughput	Amount of data processed per unit time.	Load testing at increasing scale factors.	Determines scalability and operational capacity.
Resource Efficiency	Tracks CPU, memory, and I/O consumption relative to output.	Profiling using system monitoring tools.	Supports cost optimization in cloud environments.

8. Comparison of tooling as far as Orchestration and Cloud-Native ETL

Orchestration and ETL tools analysis is a critical process to the progression of scalable and maintainable data pipelines. Each of the platforms varies in model of performance, integration with ecosystem, enjoyable development and uniformity in business operation. Open-source workflow managers such as Apache Airflow, Dagster and Prefect are powerful workflow managers with a high level of abstraction and developer experience. At the same time, cloud-native ETL systems like AWS Glue, Azure Data Factory, and Google Cloud Data Flow, offer fully managed serverless ETL

Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
to scale to large-scale loads of batch and streaming. The choice of the correct tool will depend on such factors as the intricacy of the pipeline, the latency requirements, the cloud platform and engineering skills available. A summary comparison of the same has been provided in the table below so as to be able to select the tools informedly depending on the objectives of the pipeline.

Table 4: Comparison of Orchestration and Cloud-Native ETL Tools

Tool	Type	Execution Model	Ease of Development	Strengths	Limitations	Best Use Case
Apache Airflow	Orchestrator	DAG-based scheduling	Moderate; more boilerplate	Mature ecosystem, highly extensible	Steeper learning curve	Complex enterprise batch workflows
Dagster	Orchestrator	Asset-driven jobs	High, strong abstractions	Data lineage, testing, modern UI	Smaller ecosystem	ML/analytics pipelines with asset tracking
Prefect	Orchestrator	Task/flow-based	Very high; minimal boilerplate	Flexible, intuitive, cloud-ready	Some features cloud-linked	Rapid pipeline development, hybrid workflows
AWS Glue	Cloud ETL	Serverless Spark jobs	Moderate; code-first	Deep AWS ecosystem integration	Less visual; complex for large pipelines	ETL workloads in AWS-centric environments

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Azure Data Factory	Cloud ETL	Drag-and-drop orchestration	Very high; no-code friendly	Enterprise connectors, hybrid support	Limited for advanced transformations	Enterprise ETL integrating on-prem + Azure
GCP Dataflow	Cloud ETL/Streaming	Apache Beam batch + streaming	Moderate; Beam expertise needed	Excellent scalability, real-time processing	Higher learning curve	Large-scale streaming + analytics workloads

9. AI/ML Integration in Query Optimization

In query optimization with the emergence of artificial intelligence and machine learning, there is a paradigm shift between the classical paradigms of rule-based and cost-based optimization into intelligent and adaptive systems with the ability to learn based on the assumptions about real workloads. With the database environment ever becoming dynamic, heterogeneous and large-scale, even with complex data relationships that the traditional optimizers are unable to handle, schema change, unpredictable patterns of queries and even complex data relationships. Then, the AI/ML is a way out of such deficiencies, as the solution can make inferences automatically with semantic comprehension and self-tuning mechanisms to enhance precision and efficiency. Such innovations as model-driven anomaly detection, reinforcement-learned plan selection, and semantic enrichment are supported by the aid of LLMs and are more autonomous, robust, and context-aware. In this section, the authors discuss certain AI-related innovations that could help transform the future of SQL query optimization.

9.1 Machine Learning Detection and Inference of Anomalies

Another illustration of how machine learning may be used to play a role in the modern query optimization is a custom-off-the-shelf automated schema inference and workload anomaly detectors. ML models have the capacity of generalizing the manner in which structural patterns are learnt with the help of historical query logs, thus, infer time varying relationships within a schema, without human intervention- particularly in dynamic and multi-source conditions. Also, anomaly detection algorithms detect abnormal query behavior, e.g. suspicious latency spikes, inefficient join operations or abnormal data distributions, which can be an indicator of index degradation or schema drift. The ML-driven systems that will have the constant system telemetry will enable

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
the administrators to detect the faults in the system prior to noticing the poor performances of the applications.

9.2 Multilingual Multimodal LLM

The Large Language Models (LLM) present new semantically enriching actions of comprehending, reforming and enhancing SQL workloads. They are able to standardize the SQL comments and annotations that are multilingual and the user generated description and these must be consistent across the international teams of operations. Semantic enrichment is also improved by LLM i.e. query intent recognition, high level business query translation to optimized SQL etc. and therefore facilitates transparency and debugging since they are able to simplify a complicated query plan and provide an automatic human readable plan. In addition to that, the embeddings on the basis of LLM contribute to the integration of the semantically close workloads and enhance the performance of the patterns detection of the heterogeneous systems.

9.3 Adaptive Self-tuning pipelines

Reinforcement learning (RL) permits query optimizers to acquire an adaptive and self-tuning structure, which refines its capacity on actual loads over time. Relative to the non-reinforced cost models, RL agents find the optimal join orders using trial and error, in an iterative fashion and get feedback in the form of rewards based on the execution performance. It renders the RL especially useful in a data-drifting setting, where past statistics are no longer a predictor of the behavior illustrated by the system at the time. The RL based optimizers will identify those strategies that are more successful than the previous time-tested heuristics, in a query that involves a multi-join, in this case, the complicated query.

10. Data Governance, Security, and Ethics in Query Optimization

As the scale and complexity of data-driven systems changes, the issue of query optimization cannot be viewed through the prism of full-scale performance and efficiency anymore. The modern organizations are dictated by stern rules, ethics and privacy requirements where the management is regarded as an essential part of the database optimization. Execution traces, query logs and tuning metadata might be sensitive information that must be managed securely and processed in a manner that is compliant. Another part of ethics is that the optimizers operate on learning-based models, which will accidentally expose trends in relation to individuals or sensitive data. One of the areas in this segment we explore is the influence of the privacy-saving approaches, data responsibility and international regulation on the construction of the contemporary optimization pipelines. All these will make sure that there will be no loss of performance at the cost of trust, fairness and compliance with the law.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

10.1 Privacy-Preserving Techniques in Query Optimization

Privacy-sensitive techniques have taken on an increased role in the optimization of the query, and machine learning models are handling large volumes of sensitive logs and metadata. Differential privacy enables the optimizers to understand the workload statistics without disclosing the individual query information and adds the calibrated noise to make sure that the identity of users is not known. Better confidentiality is also achieved through Federated learning as the models can be trained using distributed environment without the need to concentrate the raw data which reduces risk in multi-tenant architecture. Encrypted query logs, secure multi-party computation and plan recommendation frameworks are also used to protect sensitive workload characteristics. These approaches ensure efficient optimisation processes in the principles of data-minimization and high-performing processing.

10.2 Ethical Considerations in Handling Sensitive Query Data

The ethical handling of a sensitive query data is impossible without the information of the risks that may be experienced in the process of analyzing the logs, workload modeling and automated tuning. The query traces usually include business logic, personal identifiers or trends of behavior, and these need to be a secret asset. Such information is accessed in a legitimate way and stored with the access control that is strict because of moral governance. It can lead to discrimination, violation of privacy or discriminatory decision-making system in areas such as medical care, human resources, and political analytics since the information generated by queries may be misused. In addition, optimizers that use ML can bring out potentially harmful patterns without intending it when these are trained on biased or sensitive data.

10.3 Regulatory Compliance

Laws such as GDPR, HIPAA and the local data-protection legislation provide very strict demands on the data collection, processing, and storage of query-related information. Such aspects as workload retention policies and access logs are compliance-related. The workings of the contemporary query optimizers are increasingly being automated with tests that uphold privacy-by-design provisions to reveal sensitive fields, safe access, and data flow are restricted to jurisdictional limits. Examples of rule-based rewrites that may be imposed by metadata-aware optimizers are presentation of unauthorised joins, reduction of sensitive attribute exposure, and detection of non-conforming policy-based queries.

11. Conclusion

The paper has compared the process of optimization of SQL queries algorithms of top RDBMS systems, their theoretical basis, their applications, and the results of the compared process. Despite the fact that cost-based optimization is the prevailing method, database platforms vary greatly in terms of their execution plans, flexibility and scalability. The best developed in adaptive optimization and machine learning-

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
performance are Oracle Database and IBM Db2. Adaptive query processing is also built into Microsoft SQL Server, but it has less mature capabilities in certain respects. PostgreSQL is the most efficient with respect to parallel execution, in case of analytical workloads, whereas Server is the most efficient with respect to indexing and plan caching, and MySQL is the most efficient with respect to transactional usage. The variation of the both optimizer performance under scale loads with complex workloads is shown by the TPC-H and TPC-DS standard benchmarks. New trends include machine learning, adaptive runtime management, cloud environment workload intelligent strategies, and hybrid OLAP/OLTP processing, which suggest that the database research community is headed to smarter self-adaptive optimizers. These solutions seek to provide a reasonable trade-off between the performance, cost and energy efficiency. Benchmarking has continued to serve as a valuable guide to practitioners that need to optimize enterprise systems, and researchers that are having a hard time with challenges such as cardinality misestimation and distributed query optimization, to ensure that RDBMS systems are configured to meet current requirements depending on data.

References

[1] Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A. and Price, T.G., 1979. Access path selection in a relational database management system. *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 23–34.

[2] Chaudhuri, S., 1998. An overview of query optimization in relational systems. *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, pp. 34–43.

[3] Ioannidis, Y.E., 1996. Query optimization. *ACM Computing Surveys (CSUR)*, 28(1), pp.121–123.

[4] Neumann, T., 2009. Efficiently compiling efficient query plans for modern hardware. *Proceedings of the VLDB Endowment*, 4(9), pp.539–550.

[5] Graefe, G., 1993. Query evaluation techniques for large databases. *ACM Computing Surveys (CSUR)*, 25(2), pp.73–170.

[6] Chaudhuri, S. and Narasayya, V., 1997. An efficient cost-driven index selection tool for Microsoft SQL Server. *Proceedings of the 23rd International Conference on Very Large Data Bases*. Morgan Kaufmann, pp. 146–155.

[7] Bruno, N. and Chaudhuri, S., 2005. Automatic physical database tuning: A relaxation-based approach. *ACM SIGMOD Record*, 34(2), pp.227–238.

[8] Stillger, M., Lohman, G., Markl, V. and Kandil, M., 2001. LEO–DB2's learning optimizer. *Proceedings of the 27th International Conference on Very Large Data Bases*. Morgan Kaufmann, pp. 19–28.

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
[9] Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A. and Neumann, T., 2015. How good are query optimizers, really?. *Proceedings of the VLDB Endowment*, 9(3), pp.204–215.

[10] Markl, V., Raman, V., Simmen, D., Lohman, G. and Pirahesh, H., 2004. Robust query processing through progressive optimization. *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 659–670.

[11] Babcock, B. and Chaudhuri, S., 2005. Towards a robust query optimizer: A principled and practical approach. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 119–130.

[12] Ganapathi, A., Kuno, H.A., Dayal, U., Wiener, J.L., Fox, A., Jordan, M.I. and Patterson, D.A., 2009. Predicting multiple metrics for queries: Better decisions enabled by machine learning. *Proceedings of the 25th International Conference on Data Engineering (ICDE)*. IEEE, pp. 592–603.

[13] Marcus, R. and Papaemmanouil, O., 2018. Deep reinforcement learning for join order enumeration. *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 3–18.

[14] Marcus, R. and Papaemmanouil, O., 2019. Towards a hands-free query optimizer through deep learning. *Proceedings of the 2019 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 1245–1260.

[15] Trummer, I. and Koch, C., 2016. Multiple query optimization on the D-Wave 2X adiabatic quantum computer. *Proceedings of the VLDB Endowment*, 9(9), pp.648–659.

[16] Abadi, D.J., Boncz, P.A. and Harizopoulos, S., 2009. Column-oriented database systems. *Proceedings of the VLDB Endowment*, 2(2), pp.1664–1675.

[17] Harizopoulos, S., Abadi, D.J., Madden, S. and Stonebraker, M., 2008. OLTP through the looking glass, and what we found there. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 981–992.

[18] Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S. and Stonebraker, M., 2009. A comparison of approaches to large-scale data analysis. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 165–178.

[19] Yin, S., Hameurlain, A., & Morvan, F. (2015). Robust query optimization methods with respect to estimation errors: A survey. *ACM Sigmod Record*, *44*(3), 25-36.

[20] Chaudhuri, S., Gravano, L. and Shim, K., 1996. Optimizing queries with materialized views. *Proceedings of the 1996 IEEE International Conference on Data Engineering*. IEEE, pp. 190–200.

[21] TPC, 2014. TPC Benchmark™ H (Decision Support) Standard Specification Revision 2.17.1. Transaction Processing Performance Council.

Volume 38 No. 12s 2025,
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
[22] TPC, 2018. TPC Benchmark™ DS (Decision Support) Standard Specification Revision 2.10.0. Transaction Processing Performance Council.

[23] Boncz, P., Zukowski, M. and Nes, N., 2005. MonetDB/X100: Hyper-pipelining query execution. *Proceedings of the 2005 Conference on Innovative Data Systems Research (CIDR)*. ACM, pp. 225–237.

[24] Klonatos, Y., Koch, C., Rompf, T., & Chafi, H. (2014). Building efficient query engines in a high-level language. *Proceedings of the VLDB Endowment*, *7*(10), 853-864.

[25] Mohsin, S. A., Darwish, S. M., & Younes, A. (2021). Qiaco: a quantum dynamic cost ant system for query optimization in distributed database. *IEEE Access*, *9*, 15833-15846.

[26] Markl, V., Lohman, G. M., & Raman, V. (2003). LEO: An autonomic query optimizer for DB2. *IBM Systems Journal*, *42*(1), 98-106.

[27] Leis, V., Radke, B., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A. and Neumann, T., 2018. Cardinality estimation done right: Index-based join sampling. *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 1405–1418.

[28] Bhargava, B. (2002). Concurrency control in database systems. *IEEE transactions on knowledge and data engineering*, *11*(1), 3-16.

[29] Kumar, A. and Patel, J.M., 2015. Learning-based query performance modeling and prediction. *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering (ICDE)*. IEEE, pp. 1674–1685.

[30] Bhoyar, M., Reddy, P., & Chinta, S. (2020). Self-Tuning Databases using Machine Learning. *resource*, *8*(6).

Received: August 09, 2025