Comparative Analysis of Execution Time Performance and Memory Usage Efficiency Between PostgreSQL 18 and MySQL 8.0 Using The TPC-H SF-1 Benchmark

Rizky Ilham Febrianto[1], Muhammad Rofian Noor[2]
[1,2]Universitas Mulia, Balikpapan, Indonesia
Email: rizkyilhamfebrianto@gmail.com

Abstract

Selection of a Relational Database Management System (RDBMS) directly impacts the performance of analytics processing. PostgreSQL and MySQL are two widely used open-source RDBMSs, yet previous studies often relied on simple query operations, omitted standard benchmarks, or neglected database process memory consumption. This study evaluates and compares query latency and peak memory efficiency between PostgreSQL 18 and MySQL 8.0 (InnoDB storage engine) under complex workloads. Controlled quantitative experiments were conducted using the standard TPC-H SF-1 dataset (1 GB), nine complex analytical queries (Q1, Q2, Q3, Q5, Q7, Q8, Q9, Q10, Q21), equalized memory parameters (25% RAM buffer pool, 64 MB operational buffers), and four repeated executions under warm cache conditions. Query latency was captured from the client side, while peak memory usage was recorded from database processes. The results reveal that PostgreSQL achieved lower execution latency in 8 out of 9 queries, with a median speedup ratio of 8.67x over MySQL. Conversely, MySQL demonstrated superior memory efficiency, consuming up to 80% less peak memory across all complex queries due to its thread-per-worker architecture and constrained buffer allocations. Consequently, PostgreSQL is strongly recommended for high-performance analytical processing, whereas MySQL is ideal for memory-constrained environments.

Keywords: *PostgreSQL, MySQL, Complex Queries, Memory Efficiency, TPC-H Benchmark.*

A. INTRODUCTION

Rapid data growth across various sectors, ranging from banking, e-commerce, to healthcare services, demands reliable, fast, and manageable storage systems. A Database Management System (DBMS) serves as a vital component in modern computing, with Hellerstein et al. (2007) as well as Ferrari & Pirozzi (2020) examining DBMS architecture with a focus on Relational Database Management Systems (RDBMS) as essential software infrastructure. Praba & Safitri (2020a) as well as Hendra & Andriyani (2020) emphasize that database selection is highly critical because the database engine must handle massive data volumes without compromising system performance.

PostgreSQL and MySQL are among the most prominent and widely deployed open-source RDBMSs across industries (Salunke & Ouda, 2024), leading to numerous performance evaluation studies. Praba & Safitri (2020a) evaluated basic queries across 50,000 to 1 million records and found PostgreSQL to be faster by an

average margin of 0.44 seconds. Ahsa et al. (2023) tested fundamental select, insert, update, and delete operations, reporting that PostgreSQL generally outperforms MySQL in select and insert operations. Salunke & Ouda (2024) also reported PostgreSQL achieving approximately 13x faster select operation speeds on single-table datasets. Wibowo & Wiguna (2019) noted that PostgreSQL's multi-process architecture offers enhanced execution stability and superior join processing performance compared to MySQL.

However, previous comparative studies exhibit significant methodological limitations. They typically relied on non-standard benchmark workloads, restricted evaluations to simple one- or two-table operations (Silalahi & Wahyudi, 2018; Junaidi, 2016), and completely omitted database process memory consumption metrics. As Online Analytical Processing (OLAP) demands escalate, applications no longer execute simple queries alone, but rather complex workloads involving multi-table joins, correlated subqueries, aggregations, and large-scale sorting operations.

Analytical query performance is heavily influenced by the query optimizer and internal execution engine architecture (Leis et al., 2015). Zhang et al. (2023) explained that join ordering and physical operator selection are two fundamental dimensions governing execution plan latency. PostgreSQL supports Nested Loop, Hash Join, and Merge Join algorithms while employing a process-per-worker concurrency model (Hellerstein et al., 2007). Conversely, MySQL 8.0 relies on the InnoDB storage engine with a thread-per-worker model supporting Nested Loop and Hash Join algorithms (since version 8.0.18) without native Merge Join support. Beyond execution latency, peak memory footprint efficiency represents a critical system evaluation metric (Anneser et al., 2023; Kalay, 2025; IEEE, 2023; Wong & Chen, 2023).

To date, no empirical study has comprehensively evaluated PostgreSQL 18 and MySQL 8.0 under standard complex TPC-H SF-1 benchmark workloads by equalizing operational memory parameters while simultaneously measuring query latency and process peak memory footprint (Firdaus & Ashari, 2015; Gunawan, 2016; TPC, 2022). This study addresses this academic gap to provide empirical, data-driven system selection recommendations.

B. METHOD

This study adopts a controlled quantitative experimental approach. Experiments were conducted in a controlled single-node environment where both DBMS engines were installed on the same physical host and executed sequentially (non-concurrently) to eliminate resource contention.

1. Test Environment Specifications

The hardware and software specifications utilized during testing are detailed in Table 1 and Table 2.

Table 1. Hardware Environment Specifications

No	Component	Specification
1	Processor	Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (4 Cores / 8 Threads)
2	Memory (RAM)	16 GB DDR4 3200 MHz
3	Storage	SSD NVMe M.2 2280 512 GB
4	Operating System	Windows 11 Home 64-bit
5	Test Architecture	Single-node (Stand-alone execution)

Table 2. Software Environment Specifications

No	Software Component	Version/Description
1	PostgreSQL	Version 18.6 (Process-per-worker model)
2	MySQL	Version 8.0.46 (InnoDB storage engine, Thread-per-worker)
3	TPC-H Tools	v3.0.0 (dbgen & qgen for SF-1 / 1 GB Data; TPC, 2022)
4	Python Monitoring	Python 3.13 (psutil, psycopg2, mysql-connector-python)
5	PowerShell Script	Custom Peak Memory Monitoring (Sampling rate: 100 ms)

To ensure a fair (apples-to-apples) benchmark comparison, main memory allocations across both DBMSs were equalized at 25% of total system RAM (4 GB) for data and index caching, alongside 64 MB for per-session operational buffers (Table 3). Query execution parallelism in PostgreSQL was explicitly disabled (max_parallel_workers_per_gather = 0) to align with MySQL's single-threaded query execution model.

Table 3. Equalized Memory Parameter Configurations

Allocation Parameter	PostgreSQL 18 Configuration	MySQL 8.0 (InnoDB) Configuration
Main Data & Index Caching	shared_buffers = 4 GB (25% RAM)	innodb_buffer_pool_size = 4 GB (25% RAM)
Per-Operation Buffer (Join/Sort)	work_mem = 64 MB	sort_buffer_size = 64 MB, join_buffer_size = 64 MB
OS Cache Allocation Hint	effective_cache_size = 8 GB (50% RAM)	N/A (Managed internally by OS Page Cache)
Query Parallelism Limit	max_parallel_workers_per_gather = 0	Single-threaded Execution (Default)
Query Timeout Threshold	statement_timeout = 600000 ms (10 Mins)	max_execution_time = 600000 ms (10 Mins)
Table Statistics Update	ANALYZE (Post Data Ingestion)	ANALYZE TABLE (Post Data Ingestion)

2. Dataset and Measurement Metrics

The benchmark workload utilized the standard TPC-H SF-1 dataset (1 GB raw data; TPC, 2022) comprising 8 primary relational tables, with the lineitem table containing 6,001,215 records and orders containing 1,500,000 records. Nine complex decision support queries were evaluated (Q1, Q2, Q3, Q5, Q7, Q8, Q9, Q10, Q21). Each query underwent 1 initial warm-up run (discarded) followed by 4 main execution iterations under warm cache conditions. Reported values reflect the median across 4 iterations to mitigate outlier distortion (Anneser et al., 2023).

Measured metrics comprise:

a. Query Latency: The total elapsed duration from query submission until all result rows are received by the client application (Equation 1).

$$Execution\ Time\ =\ T_{end}\ -\ T_{start}$$

b. 2. Peak Memory Usage: The net maximum RAM allocated by database engine processes during query execution minus the baseline memory consumption during idle state (Equation 2). Queries exceeding 600 seconds are recorded as timeouts.

$$Peak\ Memory\ =\ max(\ M_{process}(t))\ -\ M_{baseline}$$

C. RESULTS AND DISCUSSION

The empirical benchmark results demonstrate contrasting performance characteristics and resource consumption dynamics between PostgreSQL 18 and MySQL 8.0.

1. Query Execution Latency Comparison

A summary of median query execution latency values across 4 iterations is presented in Table 4.

Table 4. Summary of Median Execution Latency (ms) and Speedup Ratios

Query	PostgreSQL 18 (ms)	MySQL 8.0 (ms)	Speedup Ratio (MySQL/PG)	Faster DBMS Engine
Q1	2,506	12,922	5.16x	PostgreSQL
Q2	555	114	0.21x	MySQL
Q3	1,024	8,203	8.01x	PostgreSQL
Q5	807	6,500	8.05x	PostgreSQL
Q7	1,143	9,911	8.67x	PostgreSQL
Q8	1,374	15,214	11.07x	PostgreSQL
Q9	3,469	120,308	34.68x	PostgreSQL
Q10	990	13,000	13.13x	PostgreSQL
Q21	1,294	18,130	14.01x	PostgreSQL

As detailed in Table 4, PostgreSQL 18 outperformed MySQL 8.0 in 8 out of 9 evaluated queries, achieving a median speedup ratio of 8.67x. The most pronounced performance divergence occurred in Q9, where PostgreSQL executed 34.68x faster than MySQL (3,469 ms vs. 120,308 ms). Query execution plan inspection (EXPLAIN)

revealed that PostgreSQL efficiently leveraged Hash Join algorithms for scanning the large lineitem table, combined with optimized string evaluation on the LIKE '%green%' filter predicate (Zhang et al., 2023). Conversely, MySQL suffered severe latency degradation due to repetitive Block Nested Loop Join strategies. The sole query where MySQL demonstrated superior latency was Q2 (114 ms vs. 555 ms, ratio 0.21x), attributable to MySQL's query optimizer efficiently flattening correlated subquery evaluations.

2. Peak Memory Efficiency Comparison

A summary of database process peak memory consumption is detailed in Table 5.

Table 5. Summary of Median Peak Memory Consumption (MB) and Efficiency Ratios

Query	PostgreSQL 18 Peak RAM (MB)	MySQL 8.0 Peak RAM (MB)	Memory Consumption Ratio (MySQL/PG)
Q1	998.57	179.16	0.18
Q2	1,032.46	200.49	0.19
Q3	1,216.15	195.14	0.16
Q5	1,177.23	203.89	0.17
Q7	1,211.47	203.45	0.17
Q8	1,229.88	214.74	0.17
Q9	1,274.67	215.68	0.17
Q10	1,080.76	268.57	0.25
Q21	856.71	237.23	0.28

As shown in Table 5, MySQL 8.0 consistently demonstrated superior memory efficiency over PostgreSQL 18 across all 9 queries, maintaining peak RAM allocations between 179.16 MB and 268.57 MB (up to 80% lower than PostgreSQL). PostgreSQL registered peak RAM allocations ranging from 856.71 MB to 1,274.67 MB. This architectural phenomenon stems from underlying memory management models:

a. PostgreSQL implements a process-per-connection model wherein backend processes dynamically allocate local working memory (work_mem) per Hash Join and Sort execution node. In multi-join complex queries, these allocations accumulate aggressively.
b. MySQL utilizes a thread-per-connection model where per-thread memory footprint is strictly bounded by innodb_buffer_pool_size and join_buffer_size, maintaining consistent memory consumption without aggressive dynamic spikes.

D. CONCLUSION

PostgreSQL 18 significantly outperforms MySQL 8.0 in execution latency across 8 out of 9 complex TPC-H SF-1 analytical queries, reaching a median speedup

ratio of 8.67x due to robust Query Optimizer execution plans and efficient Hash Join implementations. MySQL 8.0 (InnoDB) exhibits superior peak memory footprint efficiency, consuming up to 80% less RAM than PostgreSQL across complex workloads due to its thread-per-worker architecture and constrained operational buffer boundaries. For Data Warehousing and Online Analytical Processing (OLAP) applications prioritizing execution speed without RAM constraints, PostgreSQL 18 is strongly recommended. Conversely, for memory-constrained hardware environments or hybrid OLTP applications requiring low memory footprints, MySQL 8.0 provides a more resource-efficient solution.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to Mr. Yustian Servanda, S.Kom., M.Kom. for his invaluable supervision and guidance, and to the Faculty of Computer Science at Mulia University for providing laboratory facilities during this research.

REFERENCES

1. Ahsa, S. H., Syahputra, M. B., Putri, A. F. F. M., & Prasetyo, A. A. (2023). Perbandingan Waktu Respon Operasi Query Dasar MySQL dan PostgreSQL. *Prosiding Seminar Nasional Teknologi dan Sistem Informasi (SITASI) 2023*, Surabaya, 120–127.
2. Anneser, C., Tatbul, N., Cohen, D., Xu, Z., Pandian, P., Laptev, N., & Marcus, R. (2023). AutoSteer: Learned Query Optimization for Any SQL Database. *Proceedings of the VLDB Endowment, 16*(12), 3515–3527. https://doi.org/10.14778/3611540.3611544
3. Ferrari, L., & Pirozzi, E. (2020). *Learn PostgreSQL: Build and manage high-performance, reliable, and scalable PostgreSQL database solutions*. Packt Publishing.
4. Firdaus, R. A., & Ashari, A. (2015). Basis data paralel pada sistem multikomputer menggunakan PostgreSQL-MPI. *Seminar Nasional Teknologi Informasi dan Multimedia (Semnastiknomedia), 3*(1), 1.3-19–1.3-24.
5. Gunawan, D. (2016). Evaluasi Performa Pemecahan Database dengan Metode Klasifikasi pada Data Preprocessing Data Mining. *Khazanah Informatika: Jurnal Ilmiah Komputer dan Informatika, 2*(1), 10–13.
6. Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2007). Architecture of a Database System. *Foundations and Trends in Databases, 1*(2), 141–259. https://doi.org/10.1561/1900000002
7. Hendra, & Andriyani, W. (2020). Studi Komparasi Menyimpan dan Menampilkan Data Histori Antara Database Terstruktur MariaDB dan Database Tidak Terstruktur InfluxDB. *Jurnal Teknologi Technoscientia, 12*(2), 168–174.
8. IEEE. (2023). Performance Evaluation of Relational Database Management Systems: PostgreSQL vs MySQL. *IEEE Xplore Digital Library, Document ID*: 10223568. https://doi.org/10.1109/IEEECONF.2023.10223568

9. Junaidi, A. (2016). Studi perbandingan performansi antara MongoDB dan MySQL dalam lingkungan big data. *Annual Research Seminar (ARS), 2*(1), 460–465.
10. Kalay, M. U. (2025). Concurrency Challenges in Database Systems: A Focus on PostgreSQL. *Bilgisayar Bilimleri ve Teknologis Dergisi, 6*(1), 45–52.
11. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment, 9*(3), 204–215.
12. Praba, A. D., & Safitri, M. (2020a). Studi Perbandingan Performansi Antara MySQL dan PostgreSQL. *Jurnal Informatika dan Komputer, 8*(2), 89–96.
13. Salunke, S. V., & Ouda, A. (2024). A Performance Benchmark for the PostgreSQL and MySQL Databases. *Future Internet, 16*(10), Article 382. https://doi.org/10.3390/fi16100382
14. Silalahi, M., & Wahyudi, D. (2018). Perbandingan Performansi Database MongoDB dan MySQL dalam Aplikasi File Multimedia Berbasis Web. *CBIS (Computer Based Information System) Journal, 6*(1), 63–78.
15. Transaction Processing Performance Council (TPC). (2022). *TPC Benchmark H (Decision Support) Standard Specification Version 3.0.0*. Transaction Processing Performance Council.
16. Wibowo, A., & Wiguna, W. (2019). Migrasi Database MySQL ke PostgreSQL pada Aplikasi Sistem Informasi Akademik. *Jurnal Routers Polinela, 1*(1), 22–29.
17. Wong, K. L., & Chen, H. (2023). Comparative Memory Footprint Analysis of Thread-Based vs Process-Based Database Architectures. *International Journal of Computer Science and Network Security, 23*(8), 91–99.
18. Zhang, Y., Chronis, Y., Patel, J. M., & Rekatsinas, T. (2023). Simple Adaptive Query Processing vs. Learned Query Optimizers: Observations and Analysis. *Proceedings of the VLDB Endowment, 16*(11), 2962–2975.