International Journal of Advance Research in Science, Management, Engineering and Technology
Volume 1, Issue 4 | September - October 2026 | www.ijarsmet.com
ISSN: 3139-1141

Performance Analysis of Cloud-Based Database Systems

Akshay Ramesh Dere[#1], Dr. Rupali Kalekar[#2]
[#]*MCA Student and Project Guide, Department of Computer Application, Zeal Institute of Business Administration, Computer Application and Research (ZIBACAR), Savitribai Phule Pune University, Pune, Maharashtra, India*
[1]akshaydere285@gmail.com, [2]rupali.kalekar@zealeducation.com

Abstract—

Choosing a database for a cloud application is difficult because published performance results are rarely produced under comparable conditions. This study reviews published evidence on cloud database performance, with primary attention to MySQL and PostgreSQL, and examines how workload mix, concurrency, resource allocation, and deployment configuration shape reported results. The study follows a descriptive, comparative, literature-based methodology. No new cloud benchmark experiments were conducted, and measurements from non-equivalent studies were not pooled. Empirical benchmarks, cloud and architecture studies, and systematic reviews are compared by workload, environment, reported metrics, findings, and limitations. One study on Amazon Web Services reported strong results for managed MySQL under a read-intensive stock market workload, whereas a separate Windows-based benchmark reported lower PostgreSQL execution times in several SELECT scenarios. Because the studies used different datasets, workloads, and deployments, their results do not establish an overall ranking of the two systems. The review also finds that processor, memory, and storage measurements and benchmark configuration details are reported inconsistently. The contribution is an evidence-aware synthesis that supports workload-matched evaluation, multiple performance measures, and transparent reporting.

Keywords— :benchmark reproducibility; cloud databases*; concurrency; database benchmarking; latency; MySQL; performance evaluation; PostgreSQL; throughput; workload characterization*

I. INTRODUCTION

Cloud services have reshaped the way organizations deploy applications and manage data. Rather than maintaining every piece of computing infrastructure on site, an organization can provision database resources as its needs shift. Cloud databases now sit behind e-commerce, education, finance, healthcare, booking, and analytical applications, in which many users may read or update records at once. Database performance bears directly on response time, transaction throughput, capacity planning, and operating cost.

The database engine is not the only thing that decides performance. Many things can shift a measured result: query structure, dataset size, the proportion of reads to writes, concurrency, available central processing unit (CPU) and memory, storage, network conditions, and configuration. A system that handles simple reads well may act differently once updates or mixed transactions become common. Choosing a database therefore calls for evidence from workloads that resemble the intended application. Isolated execution times and broad product claims are poor substitutes.

Earlier studies show why context matters. Singh et al. evaluated relational and non-relational databases on stock-market time-series data, using several workloads in an Amazon Web Services (AWS) environment [1]. Salunke and Ouda compared MySQL and PostgreSQL on a Windows-based machine, using chosen SELECT and INSERT operations, including mixed read/write scenarios [2]. Their findings describe the operations and environment they tested. They cannot be assumed to hold for other datasets or cloud deployments.

How well comparisons are made is a separate concern. Taipalus's systematic review found recurring weaknesses in workload realism and in the detail needed to reproduce database benchmarks [3]. Khan et al. looked across the performance studies of Structured Query Language (SQL) and non-relational (NoSQL) architectures [4]. In their view, system architecture and application requirements should drive the choice of database. Comparative studies of cloud-provider configurations add that deployment choices can change measured performance [5].

Organizations need credible evidence on how candidate database configurations behave under workloads like their own. Published benchmarks rarely make this easy. Datasets, query mixes, hardware, database versions, cloud providers, configurations, concurrency levels, and measurement procedures all vary from one study to the next. Consider an online store that serves product searches while it also processes orders and inventory updates. A configuration that handles isolated reads well may respond differently once writes and many concurrent clients are added. Without comparable conditions, the effects of the database engine are hard to separate from those of the workload and the deployment. This paper tackles the difficulty of comparing MySQL and PostgreSQL from heterogeneous published evidence. It synthesizes reported observations on query-execution time, latency, throughput, workload mix, concurrency, and resource use, and it identifies the factors that limit direct comparison.

This paper sets out to examine what the published evidence says about cloud-database performance, concentrating on MySQL and PostgreSQL. The five specific objectives are these: (1) to review the research on cloud-database performance and benchmarking; (2) to compare reported results across read-heavy, write-heavy, and mixed workloads, where suitable evidence exists; (3) to record the performance measures that studies report (throughput, latency, CPU and memory use, and storage input and output, or I/O); (4) to examine how workload and concurrency shape interpretation;

ISSN: 3139-1141

and (5) to pick out patterns that recur in performance results, along with gaps in methodology. Three questions guide the review. Which performance dimensions are reported most often? How do the published results for MySQL and PostgreSQL differ between operations and environments? Which practices in reporting and evaluation would make future comparisons more dependable?

The original project put forward two hypotheses for a controlled benchmark. H_0 held that, under defined conditions, the tested database configurations show no statistically significant difference in the selected performance metrics. H_1 held that at least one of the selected metrics differs significantly. The paper keeps both as propositions for possible experimental work. It does not test them, because the analysis synthesizes published studies and no common experimental dataset exists. The evidence gathered here comes from three places: evaluations run in the cloud, direct benchmarks of relational databases, and reviews of benchmarking methodology. In practical terms, the synthesis encourages developers and database administrators to test systems against representative workloads and to use more than one operational measure. From an academic standpoint, it keeps results from cloud-hosted systems apart from those taken on local machines, and it points to places where thin reporting blocks comparison. This paper does not set out to compare database products universally; its scope is narrower. MySQL and PostgreSQL form the primary comparison. To help read workload and deployment effects, selected studies on SQL, NoSQL, cloud storage, analytical platforms, and configuration tuning provide background. No claim is made to cover all providers, configurations, workloads, or publications. Its conclusions hold only for the evidence identified and discussed here.

II. Materials and Methods

A. Research Design and Approach

The study is a descriptive, comparative review built on the published literature. Its focus is MySQL and PostgreSQL in cloud settings. Work on wider database architectures and benchmarking is brought in only to supply context. Every finding reported here traces back to the identified publications.

Two kinds of analysis run side by side: a qualitative thematic synthesis and a descriptive comparison of the numbers that the sources report. Studies are sorted by workload, deployment, metric, configuration, and limitations. A numerical comparison is made only when both the values and the test conditions are clear. Where experiments are not equivalent, their results are kept apart. They are never pooled and never read as rankings. One decision rule governs the review: measurements stay separate whenever sources differ in dataset, deployment model, workload, or metric. Numbers are compared only where the conditions permit a meaningful comparison. Elsewhere, the review notes contextual patterns and limitations, and it imputes no missing values.

Published evidence is extracted and interpreted in a transparent sequence, and this framework ties that sequence to the research questions. MySQL and PostgreSQL are its focus. Wider database research enters only where it helps explain workload and deployment effects. What the framework describes is a process of evidence synthesis. No benchmarking platform has been implemented.

B. Study Selection and Sampling

Scholarly work on database performance forms the intended population. This covers cloud database services, relational database management system (DBMS) benchmarking, MySQL–PostgreSQL comparisons, workload behaviour, concurrency, and benchmark reproducibility. Relevance and traceability decide which publications are selected. The target is at least 25 relevant publications. Peer-reviewed journal and conference papers from 2020–2026 come first, though older foundational works may be included when there is good reason.

A source is eligible if it addresses database performance or benchmarking, gives traceable bibliographic details, and describes its methods or findings well enough to identify them. In principle, duplicates, non-scholarly material, and papers unrelated to the research questions are excluded. The manuscript currently lists 28 references, yet its detailed extraction matrix covers just five studies. Having 28 references does not, on its own, show that every one was screened or formally included in a review.

Research from 2020–2026 gets priority in the search strategy. It covers cloud-database performance, MySQL–PostgreSQL benchmarking, latency, throughput, scalability, and concurrency, and older foundational work is allowed where it is relevant. Preference goes to peer-reviewed journal articles, conference papers, and systematic reviews. No full screening log exists in the paper (search dates, retrieved-record counts, duplicates, exclusions, final inclusion counts), so none of those numbers is reported.

C. Data Collection and Instruments

According to the documented strategy, publisher websites, ScienceDirect, SpringerLink, the ACM Digital Library, IEEE Xplore, and Google Scholar were the discovery sources. Five search phrases were used: “database benchmarking reproducibility,” “managed relational database performance,” “database latency throughput concurrency,” “MySQL PostgreSQL benchmark,” and “cloud database performance.” The manuscript does not state exact search dates or give a complete screening log, so neither should be assumed.

The principal instrument is a structured extraction spreadsheet. Each study is recorded under planned fields: authors, year, and publication first, then the systems tested and where they were deployed, what data and workload were used, how the benchmark was run, which metrics were reported, what was found, and what limited it. Where the included sample is documented, spreadsheet formulas or Python may be used for descriptive counts and summaries.

D. Analytical Dimensions and Review Procedure

This is a literature-based paper, so no new experiment supplies its comparison dimensions. The published studies do. The factors used to compare studies are configuration, benchmark tool, concurrency, workload category, deployment type, and database system. The outcomes of interest are latency, throughput, CPU and memory utilization, storage I/O, and cost, wherever these are reported. Contextual factors are also recorded, since they may explain why findings differ: measurement method, test duration, storage settings, cloud provider, database version, hardware, and dataset size.

The review followed six steps:

ISSN: 3139-1141

- Set the research questions and search terms.
- Search academic databases and publisher sites, and remove duplicates.
- Screen titles, abstracts, and full texts against the inclusion and exclusion criteria.
- Record each study's design, environment, workload, metrics, findings, and limitations in the literature matrix.
- Cluster studies by theme and compare findings only within their own experimental contexts.
- Note recurring patterns, differences, and reporting gaps, and draw conclusions bounded by the evidence.

The evidence is sorted under seven headings, starting with reproducibility and cloud configuration, then resource utilization, concurrency, and workload mix, and ending with throughput and query-execution time together with latency. Counts or percentages are meaningful only after the final set of included studies is fully documented. Source units and test conditions stay attached to each value. A mean, median, or standard deviation is summarised only if the source gives a suitable basis for it. T-tests, ANOVA, and similar inferential tests are not applied to unrelated published results. That would require compatible raw data and assumptions that can be defended.

The extraction matrix holds one entry per core study, noting its findings and limitations alongside the systems tested, the environment, the dataset and workload, the benchmark method, and the metrics reported. Across studies, the synthesis compares storage I/O, CPU and memory use, concurrency, workload category, throughput, and latency, but only where these are reported. Direct numerical comparison is limited to sufficiently similar test conditions. In every other case, results are described in the context where they arose.

E. Quality Appraisal and Comparability Rules

Each study is checked on five points: traceable source details, a clear description of the system and deployment, a defined workload, stated measurement units, and stated limitations. Numbers carry more weight when the tested systems, query types, and summary statistics can be identified. A study that lacks hardware, version, concurrency, or workload details can still feed the thematic synthesis, but its values are never pooled with another experiment's. Every performance claim stays tied to the setting where it was reported.

F. Ethical Considerations

Only published work is used. Each idea, finding, and number from other researchers is credited to its source, and nothing is presented as if the present author had measured it. Bibliographic details need checking against publisher or DOI records. Any reproduced figure or table must respect the source's attribution and licensing terms. Where access to sources or comparability is limited, this is stated openly.

III. Results and Discussion

A. Evidence Base: Cloud Evaluations and Direct Benchmarks

The data analyzed here are findings reported in published studies. The comparison centers on MySQL and PostgreSQL and looks at resource use, concurrent operations, throughput, latency, and query-execution time, so far as the sources report them. The evidence covers cloud-based evaluations, local-machine benchmarks, and reviews of database benchmarking practice.

Metrics, deployment environments, workloads, datasets, and systems all vary among the five core studies that are discussed in detail. Their results are examined in their original settings and are not merged into one numerical dataset. Other sources in the bibliography provide context, but the manuscript holds no detailed extraction matrix for every reference.

Singh et al. evaluated several database systems with historical stock-market data (open, high, low, close, and volume) and workloads that involved reads, inserts, and updates in cloud environments [1]. They examined throughput, latency, storage, and cost. Their findings belong to the financial-data use case and the AWS deployments tested. They should be read as workload-specific evidence, not as a general ranking of relational databases.

Zbořil and Svatá used pgbench, a benchmarking tool for PostgreSQL, to run scenarios against PostgreSQL services from major cloud providers. The scenarios changed the query mix, the number of clients and threads, and the service configuration [5]. Throughput, they found, can vary with the provider's configuration even though the database family stays the same. The study compares PostgreSQL offerings with one another, not PostgreSQL with MySQL. It therefore informs the deployment context but leaves the central pairwise question of this paper unanswered.

Salunke and Ouda evaluated MySQL and PostgreSQL on a synthetic behavioural-biometric (keystroke) dataset of roughly one million records, using scripted SELECT and INSERT operations [2]. Their primary tests covered retrieval of all records, conditional SELECT queries, and inserts. In the complex tests, SELECT and INSERT activity ran together. In the tested setup, PostgreSQL showed lower reported execution times in several SELECT scenarios, while the reported insert results were closer together. The benchmark ran on a local Windows machine, so it is not a direct comparison of cloud deployments.

B. Workload Mix, Tail Latency, and Benchmark Adequacy

An engine may behave one way under read-only load and another under write-heavy, transactional, or mixed load. Fruth et al. explain that average performance can hide tail latency, in which a small share of slow requests affects the user experience [6]. Qu et al. ask whether current benchmarks represent the behaviour of distributed transactional databases adequately [7]. Taipalus points to recurring problems with workload realism and reporting detail in database-comparison studies [3]. Taken together, these works argue for caution with isolated averages and single-run results.

C. Resource Allocation, Tuning, and Cloud Architecture

Performance also rests on the infrastructure and configuration around a database. Arora et al. study flexible resource allocation for relational Database-as-a-Service systems [8]. Ouyang et al. introduce RCRank, a method for ranking the root causes of slow queries in cloud database systems [9]. Erdelt describes how database benchmarks can be orchestrated in the cloud with Kubernetes [10], and Chen et al. assess the scalability of analytical platforms with an extended

ISSN: 3139-1141

TPC-DS benchmark, a decision-support benchmark from the Transaction Processing Performance Council [11]. Work on cloud analytics benchmarks and production query workloads points the same way: the choice of workload and the platform's architecture both shape evaluation results [12]–[14].

Several further studies cover other angles. One study asks how performance can be differentiated in NoSQL Database-as-a-Service [15]; another examines consistency levels in cloud-based NoSQL systems [16]; a third tests SQL Server and Azure SQL under an application workload [17]. Two more examine how Transport Layer Security (TLS) adds overhead on segmented networks [18] and how a cloud database performs on mobile devices [19]. None of these is a direct MySQL–PostgreSQL comparison. They do widen the view of what can affect database performance in a cloud application.

Design challenges in related areas are taken up by research on cloud-database architecture [20], near-data processing [21], resource management under service-level objectives [22], and elastic query execution [23]. Studies of cloud-native storage services [24], automated configuration tuning [25]–[27], and scalable online transaction processing (OLTP) [28] add that performance can hinge on the storage layer and deployment architecture as well as the engine. These works reinforce the need to document surrounding conditions. None of their measurements is handled as an interchangeable observation in a pooled analysis.

Performance should not be collapsed into a single notion of speed. Latency is the time an operation takes. Throughput counts how much work gets finished in a given period. A system can sustain high throughput while some requests still wait a long time. Suitability can also turn on consistency guarantees, resource allocation, scaling behavior, network overhead, and storage footprint. Work on TLS overhead, consistency levels, mobile use, and cloud analytics shows that performance is partly a property of the whole system, not of the query engine alone [11]–[19].

Configuration research asks how operating settings and resource allocations can be tuned to meet database service goals. Surveys of cloud-database tuning sit beside work on safe dynamic configuration changes, reinforcement-learning-based tuning, and flexible resource allocation [8], [22], [25]–[27]. The literature matters here because two tests of the same engine may use different storage, concurrency, or tuning settings.

Across the architecture and tuning studies, one lesson stands out: benchmark reports ought to document storage, concurrency, and configuration policies. This synthesis attributes a performance difference to a specific tuning setting or architectural feature only if a study isolated that mechanism under controlled conditions.

Workload and concurrency interact. Each type of task (read-heavy, write-heavy, mixed, transactional, or analytical) can expose its own bottlenecks. Throughput may climb as clients are added until a resource or contention limit is hit, while latency distributions may worsen even when average performance looks stable. Fruth et al. underline the role of tail latency [6]. Ziegler et al. turn to a wider question, whether scalable OLTP in cloud environments is a solved problem [28]. Any conclusion about concurrency must stay tied to the workload and platform actually measured.

D. Cross-Study Comparison and Research Gap

The reviewed literature contains three evidence types. Empirical benchmarks measure a defined system and workload. Cross-provider or architecture studies may not isolate MySQL from PostgreSQL. Literature reviews synthesize earlier findings and methodological concerns. Each result is read according to its evidence type, and conclusions from a review are never presented as new numerical measurements.

NoSQL studies add context on trade-offs among data models, consistency, query capability, and horizontal scaling. Those trade-offs matter when database families are compared. They do not name an overall winner between MySQL and PostgreSQL, because the studies share no common measurement basis [4], [15], [16], [20].

Table I lists the five core studies used for the detailed comparison and states what each contributes and where it falls short for the MySQL and PostgreSQL question.

TABLE I. SELECTED STUDIES AND RELEVANCE

Source	System/setting and method	Reported contribution	Limitation for this comparison
[1] Singh et al. (2022)	AWS tests of SQL and NoSQL systems on stock-market time series, with read, insert, and update workloads of different kinds.	Shows performance that depends on the workload; looks at throughput, storage, and cost.	Covers a single application domain and cloud setting; it is not a common MySQL–PostgreSQL benchmark across every workload.
[2] Salunke and Ouda (2024)	Local Windows machine running MySQL and PostgreSQL; SELECT/INSERT scripts and mixed operations.	Direct timing comparison of selected operations, with some concurrent read/write cases.	Run locally, not in the cloud; limited operations and incomplete configuration details restrict generalization.
[3] Taipalus (2024)	Systematic review of studies that compare DBMS performance.	Flags problems with workload realism and incomplete benchmark reporting.	Evidence from a review, not a new cloud experiment.
[4] Khan et al. (2023)	Systematic review covering SQL and NoSQL architectures and their performance.	Gives context on architectural trade-offs and application requirements.	Wider in scope; cannot stand in for workload-matched pairwise testing.
[5] Zbořil and Svatá (2025)	PostgreSQL services from several cloud providers, tested with pgbench scenarios.	Indicates that provider configuration can shape reported throughput.	Compares PostgreSQL offerings with each other, not MySQL with PostgreSQL.

The studies supply complementary evidence, not interchangeable evidence. In Singh et al., several database technologies are tested in the cloud under conditions tied to specific workloads [1]. Salunke and Ouda give a direct MySQL–PostgreSQL timing comparison from a local environment [2], while Zbořil and Svatá compare provider configurations for PostgreSQL [5]. Taipalus and Khan et al. offer methodological and architectural context [3], [4]. This paper keeps these differences intact and does not pool results from non-equivalent tests. Table II sets out the dimensions

ISSN: 3139-1141

along which the studies differ.

ISSN: 3139-1141

TABLE II. CROSS-STUDY COMPARISON DIMENSIONS

Dimension	Published evidence	Implication for interpretation
Method	Cloud experiments, a local benchmark, a provider-level benchmark, and systematic reviews all appear.	Studies with different designs should not be treated as one controlled experiment.
Dataset/workload	Financial time series, synthetic biometric/keystroke data, pgbench query mixes, and broader review corpora.	The query type and data model influence what is observed.
Primary metrics	Throughput, latency, storage/cost, query execution time, and transactions per second.	Numerical comparison should use only metrics and units that match.
Deployment	Managed cloud services, a local Windows benchmark, and several provider configurations.	Cloud and local measurements need separate labels.
Reporting gaps	Hardware, versions, concurrency, configuration, and CPU/memory/I/O are not always fully specified.	These gaps restrict reproducibility and resource-efficiency conclusions.
Contribution here	A structured synthesis of reported observations and their limits.	Pinpoints patterns and reporting gaps; no new measurements are claimed.

What the literature shows is that workload and deployment context shape database performance. No system can be ranked best for every use case. Some applications need the transaction semantics and consistency features that relational databases can provide. Some NoSQL systems, meanwhile, may scale out horizontally when the data model and access pattern suit them [4], [15], [16]. These wider architectural trade-offs should not be mistaken for direct evidence on MySQL versus PostgreSQL.

The research gap is not a lack of database comparisons. The trouble is that results are hard to compare when performance metrics, concurrency, hardware, database versions, cloud providers, datasets, and workloads all differ from one study to another. Under equivalent workload profiles, the core studies do not report storage-I/O, memory, and CPU measurements in a consistent way. This paper therefore addresses the gap at the level of synthesis: it documents the context of each study and limits numerical comparisons to the conditions under which the values were reported.

E. Main Findings

No universal performance advantage for MySQL or PostgreSQL emerges from the reviewed literature. Relative performance varies with query type, workload composition, deployment, and configuration. In one read-intensive stock-market workload, Singh et al. saw strong results from AWS-managed MySQL, which got through about 100,000 operations in roughly half a minute [1]. That finding is specific to the workload and configuration.

A separate Windows-based benchmark by Salunke and Ouda gave PostgreSQL the lower execution times in several SELECT scenarios, including tests where SELECT operations

ISSN: 3139-1141

were running during inserts [2]. These results describe the tested local setup. They do not show how the databases would compare in equivalent cloud deployments.

Gaps between the two databases were smaller in some INSERT scenarios than in SELECT queries. Its narrative ranges and tabulated summary statistics, however, are not fully consistent in every case, and this source-level issue is treated cautiously below. Resource-efficiency comparisons are further limited because the core studies measure CPU, memory, and storage I/O inconsistently. Taipalus voices a similar worry about unrealistic workloads and patchy benchmark reporting [3]. Table III summarizes the findings reported in these studies.

TABLE III. SUMMARY OF REPORTED FINDINGS

Parameter	Reported observation	Source
Read-intensive workload	For the reported workload, AWS-managed MySQL finished about 100,000 operations in roughly half a minute.	Singh et al. [1]
SELECT execution time	In several SELECT scenarios, PostgreSQL had lower reported times than MySQL.	Salunke and Ouda [2]
Conditional SELECT	In the tested WHERE-condition scenarios, PostgreSQL reported lower times.	Salunke and Ouda [2]
Concurrent reads and inserts	During inserts in the tested scenarios, PostgreSQL kept its reported SELECT times lower.	Salunke and Ouda [2]
Insert operations	Reported INSERT results were closer together than some SELECT results, though the article's ranges and summary statistics are inconsistent.	Salunke and Ouda [2]
CPU, memory, and I/O	Resource measurements in the core studies are not consistent, so no cross-study efficiency ranking can be supported.	Synthesis; [1]–[5]
Reproducibility	Limited workload realism and incomplete method reporting restrict replication and comparison.	Taipalus [3]; Qu et al. [7]

F. Reported Execution-Time Ranges

The execution-time ranges that Salunke and Ouda state in their narrative are reproduced in Table IV [2]. Table V is kept apart and carries the medians from their statistical tables. In the original article, some narrative ranges disagree with the tabulated summaries, most clearly for conditional SELECT and INSERT scenarios.

TABLE IV. REPORTED EXECUTION-TIME RANGES

Tested operation	MySQL range reported in source (ms)	PostgreSQL range reported in source (ms)
SELECT all records	9–12	0.6–0.8
SELECT with a WHERE condition[a]	0.9–1.0	0.09–0.13
SELECT during simultaneous inserts	7–13	0.7–0.9
Basic INSERT operations[a]	0.0010–0.0030	0.0007–0.0014

[a]Narrative range as stated in [2]. The medians tabulated in [2] for this scenario fall below the stated range (see Table V). Requires verification.

ISSN: 3139-1141

G. Figures Based on Published Statistics

Figs. 1–3 represent median and range summaries reported by Salunke and Ouda [2]. They are illustrations drawn from a secondary source, not measurements collected for this paper. Because some narrative ranges in the original article disagree with its summary tables, the figures are best read descriptively, with the source context kept in mind. Fig. 1 compares the reported median execution times, Fig. 2 summarizes the reported distributions, and Fig. 3 shows the minimum-to-maximum ranges with median markers. Table V lists the medians used in the figures.

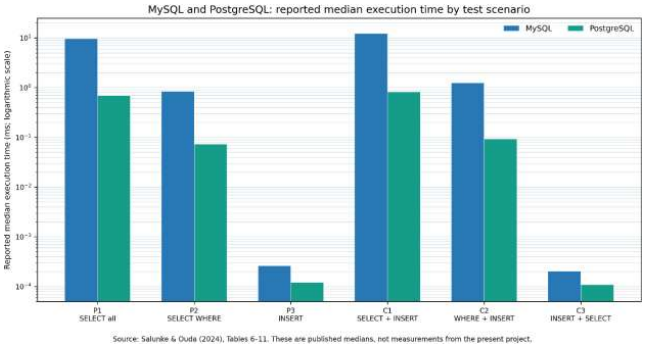


Fig. 1. Bar chart comparing the median execution times reported for MySQL and PostgreSQL. Source: Salunke and Ouda [2], Tables 6–11. A logarithmic vertical axis is used, since the reported medians cover several orders of magnitude. A shorter execution time means faster completion only for the particular operation tested. These results were obtained in a local Windows-based benchmark and do not compare cloud deployments.

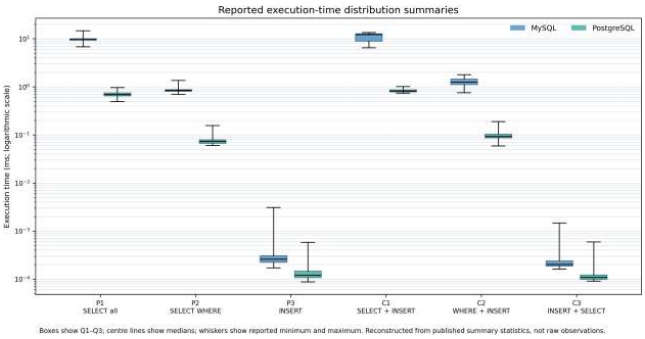


Fig. 2. Box-and-whisker summaries reconstructed from published five-number statistics. Boxes mark the reported first and third quartiles, center lines mark medians, and whiskers show the minima and maxima given in the source. The summaries are reconstructed and are not raw run-level data. The narrative ranges for the WHERE-condition and INSERT scenarios in the source article do not agree with its own tabulated statistics (see Tables IV and V), so the matching parts of the reconstructed distribution should be treated as provisional and requiring verification.

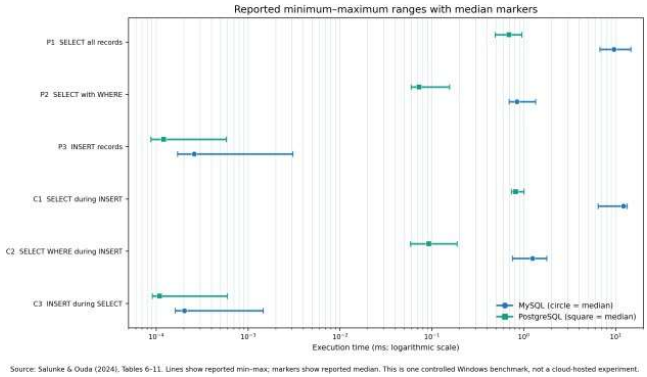


Fig. 3. Reported minimum-to-maximum execution-time ranges with median markers. Each line runs from the minimum to the maximum reported in the source, and a marker shows the reported median. All values come from the Windows-based benchmark in [2].

TABLE V. PUBLISHED MEDIAN VALUES USED IN FIGS. 1–3

Scenario	MySQL median (ms)	PostgreSQL median (ms)	Ratio of MySQL to PostgreSQL median (unitless)
P1: SELECT all records	9.61042	0.692291	13.88
P2: SELECT with WHERE[a]	0.843838	0.0726841	11.61
P3: INSERT records[a]	0.000260	0.0001207	2.15
C1: SELECT during INSERT	12.2288	0.817204	14.96
C2: SELECT WHERE during INSERT	1.25388	0.0929318	13.49
C3: INSERT during SELECT	0.00020265	0.000109	1.86

[a]The tabulated median lies outside the narrative range reported for this scenario in [2] (see Table IV). Requires verification against the source.

Medians were copied from Tables 6–11 of [2]. Each ratio divides the MySQL median by the PostgreSQL median. All figures are published summaries, and none were measured in the present study. The source article's narrative ranges conflict with some of its tabulated values, so the ratios compare tabulated medians only, and they should be read with caution.

In the transcribed medians, the largest MySQL-to-PostgreSQL ratios appear in SELECT-related scenarios, including SELECT during INSERT. Ratios for INSERT-related scenarios are smaller. This pattern describes the source benchmark and the transcribed values. It should not be taken as an estimate of any general cloud-wide effect. Any calculation built on these medians inherits the limitations and inconsistencies of the original source.

This paper calculates no new mean, standard deviation, confidence interval, p-value, or significance test. The cited studies use different methods and offer no common run-level dataset suitable for meta-analysis. Ratios of reported medians are arithmetic comparisons. They are not evidence of statistical significance or causation. Inferential analysis would need compatible raw or repeated measurements and a justified statistical model.

ISSN: 3139-1141

H. Evidence Coverage and Reporting Quality

Fig. 4 summarizes what four core sources contribute. Empirical cloud results come from Singh et al. [1], while direct MySQL–PostgreSQL timings from a local benchmark come from Salunke and Ouda [2]. Taipalus and Khan et al. contribute review-level evidence on benchmarking and database architecture [3], [4]. The matrix covers only these four sources and does not represent a final coding of every reference in the bibliography.

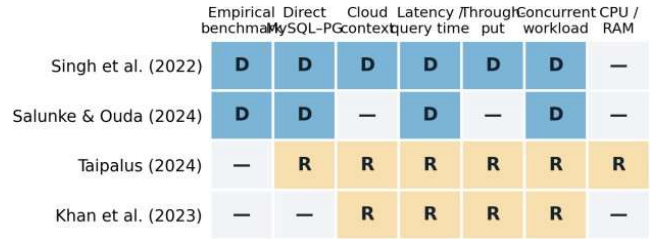


Fig. 4. Evidence matrix covering four of the core sources cited in this paper. In the matrix, "Direct" marks an empirical measurement or evaluation and "Review" marks a synthesis of earlier literature. "Not a primary reported result" means the source does not quantify that item as a principal outcome.

No line graph of increasing workload or concurrency is included. What was extracted is a set of summaries for separate scenarios. It is not one ordered series of comparable measurements across workload or concurrency levels. Joining these summaries would suggest a trend that the available data do not establish. A graph of that kind belongs in the paper only once comparable workload-level measurements are available from the original study.

I. Objective-Wise Results

Table VI relates each objective of the review to what the examined literature shows.

TABLE VI. OBJECTIVE-WISE RESULT SUMMARY

Objective	Finding from the literature	Status
Identify performance parameters	The core empirical studies report latency, execution time, and throughput; evidence on resource use is less consistent.	Addressed by analysis of the literature
Compare MySQL and PostgreSQL	AWS MySQL did well in one read-intensive cloud workload [1]; PostgreSQL showed lower reported SELECT times in the local benchmark [2].	Comparative observations tied to context
Examine workload and testing conditions	Outcomes differed with query type, concurrent operations, and environment.	Dependence on workload identified
Identify limitations in existing evaluations	Differing methods and incomplete reporting restrict direct comparison and replication.	Gaps in methodology identified

J. Interpretation and Explanation of Differences

The findings do not support ranking MySQL and PostgreSQL on a single metric or benchmark. Dataset, workload, deployment, and evaluation objective all differ between the AWS stock-market study and the local Windows benchmark. Their results therefore answer different questions and do not directly contradict each other. Concurrency, storage, configuration, caching, indexing, and query structure may all influence results. This synthesis cannot isolate those effects, because the studies did not follow a common controlled protocol.

This reading is consistent with Taipalus's concern that unrealistic workloads and incomplete reporting limit how far benchmark results generalize [3]. Khan et al. widen the context to SQL and NoSQL architecture, yet they stop short of ranking MySQL against PostgreSQL in a cloud-only setting [4]. Zbořil and Svatá demonstrate that PostgreSQL performance can depend on the provider's configuration [5]. Studies of cloud-native storage and tuning [24]–[28], analytical workloads [11]–[14], benchmark orchestration [10], root-cause analysis of slow cloud queries [9], and resource allocation [8] all give further support for documenting test conditions.

One apparent difference stands out. In the read-intensive stock-market workload, AWS-managed MySQL did well, whereas in the local benchmark PostgreSQL had lower SELECT execution times. This is not enough to call the results contradictory. The two studies differ in dataset, query mix, deployment model, hardware, and how they measured. Neither result should be carried over to a different workload or platform before further validation.

Network conditions, resource allocation, concurrency control, storage behaviour, default settings, caching, indexing, and query design could each account for the gap. They remain possible explanations, not established causes, because the reviewed studies share no protocol that isolates them. The defensible conclusion is that each performance advantage must be read within the workload and conditions its source reported.

K. Practical and Theoretical Implications

Query-execution time describes a single operation, whereas throughput measures how much work is completed over time. Concurrency tells us how many clients or operations are active at once. Latency distributions reveal whether a small share of requests take far longer than typical ones. CPU, memory, and storage-I/O measurements can help explain the resources spent to reach a given result. The core sources are not consistent in which of these dimensions they report.

The kinds of measures also differ. Some studies give operation counts over time, others give execution-time ranges or medians for individual queries, and the reviews give conclusions. These should not be placed on a common scale without defensible normalization. For that reason, the figures show scenario-specific values and name the source environment, rather than presenting a pooled benchmark.

In practice, database selection should start from the target application's workload. A system is stressed in different places by simple reads, complex joins, frequent inserts, or mixed read/write operations. For a fair direct benchmark, the datasets, schemas, query definitions, data volumes, and resource allocations would need to be equivalent. Measurement units, concurrency, warm-up, test duration, deployment model, storage type, instance size, cloud region, configuration, indexes, and database versions should be recorded as well.

Benchmark reports should go beyond mean response time. Cost, scalability, storage I/O, CPU and memory use, run-to-run variation, throughput in consistent units, 95th and 99th percentile (P95 and P99) latency where available, and median latency together give a fuller picture. Tail latency can expose

ISSN: 3139-1141

slow requests that averages conceal [6], and workload realism and reproducible reporting remain important [3], [7].

The view of database performance supported here is multidimensional and depends on context. No application-independent best option can be read from latency and throughput alone. Interpretation is affected as well by workload, concurrency, resource allocation, architecture, deployment, and measurement method. What the paper adds is a synthesis of published comparisons and of their limitations. It offers neither a new database system nor an independent performance estimate.

L. Summary of Major Findings

Nothing in the evidence justifies saying that MySQL or PostgreSQL is faster in every situation. Outcomes shift with deployment, configuration, concurrency, query complexity, read/write distribution, and operation type. Singh et al.'s MySQL result concerns a single read-intensive AWS workload [1]. The PostgreSQL SELECT results of Salunke and Ouda, by contrast, concern selected operations in a local benchmark [2].

Applications that must serve requests while updating data need concurrent reads and writes evaluated separately. Cost, scalability, storage I/O, CPU and memory use, latency variation, and throughput deserve weight next to latency and execution time. Since the core studies report resource use inconsistently, neither database can be ranked on resource efficiency. Replication and comparison across studies are further restricted by differences in test configuration, concurrency, workloads, datasets, hardware, and versions [3], [7]. Table VII summarizes the major findings and their implications.

TABLE VII. SUMMARY OF MAJOR FINDINGS

Finding area	Summary	Implication
Workload dependence	Relative performance shifts with the workload and test conditions.	Evaluate against the intended application and avoid any universal ranking.
Read-intensive cloud workload	AWS-managed MySQL did well in the specific workload that Singh et al. studied [1].	Read the result as specific to the workload and configuration.
SELECT and concurrent tests	In the local benchmark by Salunke and Ouda, PostgreSQL recorded lower SELECT times [2].	In the target environment, test representative SELECTs and concurrent read/write operations.
Metric coverage	Evidence on CPU, memory, and I/O is inconsistent.	Gather several performance and resource metrics together.
Reproducibility	Missing method detail limits comparison and replication.	Document tools, versions, settings, workload, and measurements.

M. Recommendations for Database Evaluation

The evidence supports a workload-driven selection process. Candidate databases should be evaluated with representative application queries and data, not chosen from a single published result. If direct benchmarking becomes feasible, MySQL and PostgreSQL should run on equivalent datasets, schemas, indexes, query definitions, and resource allocations, under read-heavy, write-heavy, and mixed workloads. Differences that cannot be brought under control need to be written down.

A reproducible benchmark reports median latency and, where available, tail percentiles. It also gives throughput in consistent units and variation across repeated runs, followed by CPU and memory utilization, storage I/O, and cost. Concurrency should rise in documented steps, with repeated measurements at each level. Where permitted, measurement units, configuration files, database versions, data-generation steps, scripts, run counts, and warm-up procedures should be kept. Table VIII lists the recommended evaluation practices.

TABLE VIII. RECOMMENDED EVALUATION PRACTICES

Practice	Recommended action	Benefit
Workload definition	Note query types, read/write mix, data size, and application behaviour.	Makes the test more relevant to the use case.
Controlled comparison	Keep conditions equivalent and record unavoidable differences.	Cuts down confounding factors.
Repeated measurement	Repeat every scenario; report medians and variation.	Lessens reliance on one unusual run.
Multi-metric evaluation	Measure latency, throughput, resources, I/O, and cost where available.	Gives a wider view of operational suitability.
Concurrency testing	Test several simultaneous load levels using representative queries.	Reveals how performance changes as demand grows.
Reproducibility	Keep scripts, configuration details, versions, and data definitions.	Lets others review and repeat the benchmark.

Before any testing begins, the expected read/write ratio, the dataset size, and the principal application transactions should be defined. The cloud region and resource configuration should stay constant where possible, and the warm-up and run sequence should be documented. Each reported value needs a label saying whether it is a mean, median, percentile, range, or throughput rate. Labeled this way, summaries from secondary sources cannot be confused with measurements collected in this project.

Performance is not the only selection criterion. The decision can also be shaped by service pricing, ecosystem compatibility, schema needs, maintenance effort, and requirements for availability and recovery. A useful comparison asks whether each candidate meets the application's workload and service needs at an acceptable operational effort and cost.

N. Evidence-Informed Decision Guide

Table IX summarizes how the reviewed evidence can inform application-specific testing. It should not be read as a universal ranking of database products.

TABLE IX. EVIDENCE-INFORMED DATABASE SELECTION GUIDE

Evaluation need	What the evidence suggests	Recommended action
Read-intensive workload	MySQL did well in the AWS workload studied by Singh et al. [1].	Consider MySQL a candidate and, when practical, test a similar workload.

ISSN: 3139-1141

Evaluation need	What the evidence suggests	Recommended action
SELECT-query latency	In the benchmark by Salunke and Ouda, PostgreSQL reported lower SELECT execution times [2].	Run the application's actual SELECT statements on both databases, and do not assume the same result in a cloud deployment.
Simultaneous reads and inserts	In the tested concurrent scenarios, PostgreSQL kept its reported SELECT times lower [2].	Benchmark concurrent reads and writes at realistic load levels.
Resource/cost constraints	The available metrics are not enough to name a general resource-efficiency winner.	Measure CPU, memory, I/O, storage, and service cost in the intended environment.

O. Limitations

No cloud-performance measurements were collected independently for this study, which rests entirely on secondary sources. The reviewed papers vary in benchmarking method, deployment environment, configuration, workload, dataset, database version, and hardware. Several also leave out details of CPU, memory, storage I/O, concurrency, or configuration. Only five representative studies receive detailed extraction, although the bibliography has 28 entries, and the intended sample of at least 25 papers has no complete screening log. The studies share no common design and no compatible raw data. For that reason, measurements are not pooled, and no new significance test is run. Long-term behavior is not assessed independently, and findings are not validated in production. The conclusions go no further than the published evidence, and they do not amount to a universal ranking of MySQL and PostgreSQL.

P. Future Scope

Documenting search and screening decisions and enlarging the literature sample would strengthen the evidence. Comparisons across cloud regions, providers, and deployment models could show whether the observed differences hold in other settings. Performance requirements may also differ between application workloads, such as transactional systems, analytics, mobile-facing services, and data-intensive applications.

One option for a future study is to combine a structured review with controlled benchmarking, or to use a mixed-method design. MySQL and PostgreSQL would then be tested with comparable hardware, database versions, datasets, schemas, configurations, and queries, under read-heavy, write-heavy, and mixed workloads. Concurrency should be varied systematically, and repeated runs should allow latency distributions and throughput to be analyzed. Inferential statistics belong in the report only when the data and test assumptions justify them.

Operational effort, backup and recovery, security, reliability, scalability, service cost, tail latency, storage I/O, and CPU and memory use are all candidate variables. Comparisons could also be extended to alternative storage architectures, newer cloud-native databases, serverless offerings, containerized deployments, and managed services. Monitoring traffic, data growth, maintenance, and recovery over a long period would show whether short benchmarks mirror production behaviour.

Production-like or real-world studies could show whether benchmark findings transfer to actual applications. Equivalent conditions should be used where possible, and unavoidable service differences documented. Any new measurements would be future evidence. They would not be results of the present literature-based paper.

IV. Conclusions

This paper examined why cloud-database performance comparisons are hard to make when published studies use different workloads and deployment environments. Using a descriptive synthesis of the literature, the paper reviewed evidence on MySQL and PostgreSQL in terms of concurrency, workload mix, throughput, latency, and query-execution time.

No universal winner emerges. Singh et al. [1] found AWS-managed MySQL performing strongly under a read-intensive stock-market workload. In a Windows-based benchmark, Salunke and Ouda [2] reported that PostgreSQL ran certain selected SELECT scenarios faster. The studies used different conditions, so they do not form a direct head-to-head cloud comparison, and resource-use metrics were also reported inconsistently. What the paper adds is a synthesis, aware of the limits of its evidence, that picks out dependence on workload and shortfalls in comparability and reproducibility. Database selection should rest on representative application workloads and several performance measures. Later comparisons ought to line up their datasets, queries, versions, resource allocations, and measurement procedures.

The contribution is meant to be a structured view of the published evidence, together with the factors that constrain comparison across studies. Methodological gaps and differing results are identified, and questions for future controlled MySQL–PostgreSQL experiments in cloud environments are outlined.

Acknowledgment

I sincerely thank Dr. Rupali Kalekar, my research guide, for her guidance, steady encouragement, and support throughout this project. I am also grateful to Dr. Dinesh Banswal for his encouragement, cooperation, and support. Finally, I thank the Zeal Institute of Business Administration, Computer Application and Research (ZIBACAR), Pune, for the academic environment, institutional support, and resources that contributed to this work.

References

[1] B. Singh, R. Martyr, T. Medland, J. Astin, G. Hunter, and J.-C. Nebel, "Cloud based evaluation of databases for stock market data," *J. Cloud Comput.*, vol. 11, Art. no. 53, 2022, doi: 10.1186/s13677-022-00323-4.

[2] S.V. Salunke and A. Ouda, "A performance benchmark for the PostgreSQL and MySQL databases," *Future Internet*, vol. 16, no. 10, Art. no. 382, 2024, doi: 10.3390/fi16100382.

[3] T. Taipalus, "Database management system performance comparisons: a systematic literature review," *J. Syst. Softw.*, vol. 208, Art. no. 111872, 2024, doi: 10.1016/j.jss.2023.111872.

[4] W. Khan, T. Kumar, C. Zhang, K. Raj, A.M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review," *Big Data Cogn. Comput.*, vol. 7, no. 2, Art. no. 97, 2023, doi: 10.3390/bdcc7020097.

[5] M. Zbořil and V. Svatá, "Performance comparison of cloud databases," *Procedia Comput. Sci.*, vol. 256, pp. 388-395, 2025, doi: 10.1016/j.procs.2025.02.134.

[6] M. Fruth, S. Scherzinger, W. Mauerer, and R. Ramsauer, "Tell-tale tail latencies: pitfalls and perils in database benchmarking," in *Performance*

Evaluation and Benchmarking: 13th TPC Technology Conference, TPCTC 2021, Revised Selected Papers. Cham, Switzerland: Springer, 2022, pp. 119-134, doi: 10.1007/978-3-030-94437-7_8.

[7] L. Qu, Q. Wang, T. Chen, K. Li, R. Zhang, X. Zhou, X. Xu, Y. Yang, C. Yang, W. Qian, and A. Zhou, "Are current benchmarks adequate to evaluate distributed transactional databases?," *BenchCouncil Trans. Benchmarks, Stand. Eval.*, vol. 2, no. 1, Art. no. 100031, 2022, doi: 10.1016/j.tbench.2022.100031.

[8] P. Arora et al., "Flexible resource allocation for relational database-as-a-service," *Proc. VLDB Endow.*, vol. 16, no. 13, pp. 4202-4215, 2023, doi: 10.14778/3625054.3625058.

[9] B. Ouyang, Y. Zhang, H. Cheng, Y. Shu, C. Guo, B. Yang, Q. Wen, L. Fan, and C.S. Jensen, "RCRank: multimodal ranking of root causes of slow queries in cloud database systems," *Proc. VLDB Endow.*, vol. 18, no. 4, pp. 1169-1182, 2025, doi: 10.14778/3717755.3717774.

[10] P.K. Erdelt, "Orchestrating DBMS benchmarking in the cloud with Kubernetes," in *Performance Evaluation and Benchmarking: 13th TPC Technology Conference, TPCTC 2021, Revised Selected Papers*. Cham, Switzerland: Springer, 2022, pp. 81-97, doi: 10.1007/978-3-030-94437-7_6.

[11] G. Chen, T. Johnson, and M. Cilimdzic, "Quantifying cloud data analytic platform scalability with extended TPC-DS benchmark," in *Performance Evaluation and Benchmarking: 13th TPC Technology Conference, TPCTC 2021, Revised Selected Papers*. Cham, Switzerland: Springer, 2022, pp. 135-150, doi: 10.1007/978-3-030-94437-7_9.

[12] A. van Renen and V. Leis, "Cloud analytics benchmark," *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1413-1425, 2023, doi: 10.14778/3583140.3583156.

[13] A. van Renen, D. Horn, P. Pfeil, K.E. Vaidya, W. Dong, M. Narayanaswamy, Z. Liu, G. Saxena, A. Kipf, and T. Kraska, "Why TPC is not enough: an analysis of the Amazon Redshift fleet," *Proc. VLDB Endow.*, vol. 17, no. 11, pp. 3694-3706, 2024, doi: 10.14778/3681954.3682031.

[14] J.V. Szlang et al., "Workload insights from the Snowflake Data Cloud: what do production analytic queries really look like?," *Proc. VLDB Endow.*, vol. 18, no. 12, pp. 5126-5138, 2025, doi: 10.14778/3750601.3750632.

[15] R. Andreoli, T. Cucinotta, and D.B. De Oliveira, "Priority-driven differentiated performance for NoSQL database-as-a-service," *IEEE Trans. Cloud Comput.*, vol. 11, no. 4, pp. 3469-3482, 2023, doi: 10.1109/TCC.2023.3292031.

[16] S. Ferreira, J. Mendonça, B. Nogueira, W. Tiengo, and E. Andrade, "Impacts of data consistency levels in cloud-based NoSQL for data-intensive applications," *J. Cloud Comput.*, vol. 13, Art. no. 158, 2024, doi: 10.1186/s13677-024-00716-7.

[17] A.J. Kadhim and T.S. Atia, "Performance evaluation of on-premise SQL Server and Azure SQL Database using a .NET 8 e-commerce application," *Computers*, vol. 15, no. 8, Art. no. 469, 2026, doi: 10.3390/computers15080469.

[18] S.P. Kannojia and J. Kurmi, "Impact of TLS overhead on segmented network for cloud database systems," *SAMRIDDHI: J. Phys. Sci. Eng. Technol.*, vol. 15, no. 1, pp. 171-175, 2023, doi: 10.18090/samriddhi.v15i01.22.

[19] S. Kot and J. Smołka, "A performance analysis of a cloud database on mobile devices," *J. Comput. Sci. Inst.*, vol. 29, pp. 360-365, 2023, doi: 10.35784/jcsi.3798.

[20] G. Li, H. Dong, and C. Zhang, "Cloud databases: new techniques, challenges, and opportunities," *Proc. VLDB Endow.*, vol. 15, no. 12, pp. 3758-3761, 2022, doi: 10.14778/3554821.3554893.

[21] S. Lin, A.P. Marathe, P.-A. Larson, C. Chen, C. Sun, P. Lee, and W. Yu, "Near data processing in Taurus Database," arXiv preprint arXiv:2506.20010, 2025, doi: 10.48550/arXiv.2506.20010.

[22] H. Liu, R. Li, Z. Zhang, and B. Tang, "Tao: improving resource utilization while guaranteeing SLO in multi-tenant relational database-as-a-service," *Proc. ACM Manag. Data*, vol. 2, no. 4, Art. no. 205, 2024, doi: 10.1145/3677141.

[23] M. Vuppalapati, J. Miron, R. Agarwal, D. Truong, A. Motivala, and T. Cruanes, "Building an elastic query engine on disaggregated storage," in *Proc. 17th USENIX Symp. Networked Systems Design and Implementation (NSDI)*, 2020, pp. 449-462. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/vuppalapati

[24] J. Zhang, W. Jiang, B. Tang, H. Ma, L. Cao, Z. Jiang, Y. Nie, F. Wang, L. Zhang, and Y. Liang, "CDSBen: benchmarking the performance of storage services in cloud-native database system at ByteDance," *Proc. VLDB Endow.*, vol. 16, no. 12, pp. 3584-3596, 2023, doi: 10.14778/3611540.3611549.

[25] J. Zhang, K. Zhou, G. Li, Y. Liu, M. Xie, B. Cheng, and J. Xing, "CDBTune+: an efficient deep reinforcement learning-based automatic cloud database tuning system," *VLDB J.*, vol. 30, pp. 959-987, 2021, doi: 10.1007/s00778-021-00670-9.

[26] L. Zhang and M.A. Babar, "Automatic configuration tuning on cloud database: a survey," arXiv preprint arXiv:2404.06043, 2024. [Online]. Available: https://arxiv.org/abs/2404.06043

[27] X. Zhang, H. Wu, Y. Li, J. Tan, F. Li, and B. Cui, "Towards dynamic and safe configuration tuning for cloud databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2022, pp. 631-645, doi: 10.1145/3514221.3526176.

[28] T. Ziegler, P.A. Bernstein, V. Leis, and T. Binnig, "Is scalable OLTP in the cloud a solved problem?," in *Proc. Conf. Innovative Data Systems Research (CIDR)*, 2023. [Online]. Available: https://www.cidrdb.org/cidr2023/papers/p50-ziegler.pdf