The Journal of Systems and Software 208 (2024) 111872

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Database management system performance comparisons: A systematic literature review☆

Toni Taipalus

Faculty of Information Technology, University of Jyväskylä, P.O. Box 35, FI-40014, Finland

ARTICLE INFO

Keywords:
Database
Performance
Comparison
Database management system
Relational database
NoSQL
NewSQL

ABSTRACT

Efficiency has been a pivotal aspect of the software industry since its inception, as a system that serves the end-user fast, and the service provider cost-efficiently benefits all parties. A database management system (DBMS) is an integral part of effectively all software systems, and therefore it is logical that different studies have compared the performance of different DBMSs in hopes of finding the most efficient one. This study systematically synthesizes the results and approaches of studies that compare DBMS performance and provides recommendations for industry and research. The results show that performance is usually tested in a way that does not reflect real-world use cases, and that tests are typically reported in insufficient detail for replication or for drawing conclusions from the stated results.

1. Introduction

Efficiency is important in effectively all software systems, whether efficiency is measured by response times, how many concurrent users the system can serve, or how energy-efficient the system is (Toffola et al., 2018). Despite its importance, many software systems suffer from efficiency problems (Jin et al., 2012), as optimization has been largely recognized as a complex task (Toffola et al., 2018; Difallah et al., 2013). The more a system holds and handles data, the more the system's performance depends on the database, and the database is often one of the first suspects when a performance issue is detected. The domain of database management systems (DBMS) saw rapid advancements in performance especially in the 1980s and 1990s, as benchmarking competitions between DBMS and hardware vendors led to innovations in DBMS technology that significantly improved DBMS performance (DeWitt and Levine, 2008). Performance improvements are related to DBMS aspects such as different supporting data structures (Valduriez, 1987), and algorithms for sorting (Estivill-Castro and Wood, 1992; Do et al., 2022) and joining (Schneider and DeWitt, 1989; Patel and DeWitt, 1996). Given that DBMSs are annually a multi-billion dollar industry, the performance of a DBMS is one of the most crucial aspects when a company chooses a DBMS for their product or service (Dietrich et al., 1992). As different DBMS performance comparison studies and DBMS vendor white-papers highlight the performance gains of one DBMS over another, it may seem tempting to either consider choosing the fastest DBMS for a business domain or to migrate from one DBMS to another for performance gains. However, as we show and argue in this study, performance is typically tested in very specific contexts which are not necessarily generalizable, and there are other aspects besides performance to consider.

This study was inspired by a study by Raasveldt et al. (2018), which claimed that "*[...] we will explore the common pitfalls in database performance benchmarking that are present in a large number of scientific works [...]*" while consciously refraining from citing example studies. While we agree with their claim based on our personal experiences, we wanted to systematically explore whether this phenomenon is common among performance comparisons, and whether such studies show performance gains of one DBMS over another in a setting that can be replicated. This study is not an attempt to criticize studies comparing DBMS performance, as no scientific study (ours included) is without threats to validity. Rather, based on the survey of the literature, the primary goals of our study are to propagate information on *(i)* how DBMS performance has been tested, *(ii)* how performance has been recommended to be tested, *(iii)* how the performance comparison results should be interpreted, *(iv)* what other aspects besides performance should be considered, and *(v)* what other avenues might be fruitful for DBMS performance testing. Additionally, we provide *(vi)* a relatively accessible background on database system performance, followed by *(vii)* a systematic review of literature on DBMS performance comparisons, *(viii)* describing which DBMSs and which types of DBMSs have been compared with each other, *(ix)* the outcomes of the performance comparisons, and *(x)* by which benchmarks the DBMSs have been compared.

☆ Editor: Dr. Jacopo Soldani.

E-mail address: toni.taipalus@jyu.fi.

https://doi.org/10.1016/j.jss.2023.111872
Received 10 March 2023; Accepted 4 October 2023
Available online 27 October 2023

The rest of this study is structured as follows. In Sections 2 and 3, we provide theoretical background for understanding the results and discussion provided by this study. These background sections are deliberately presented by refraining from using unnecessary information technology-related terms, acronyms, algorithms, or mathematics, to cater to the needs of readers from various backgrounds. For readers more technically inclined or interested, we have provided further reading at the end of Sections 2 and 3. Section 4 details how we searched, selected, and categorized the DBMS performance comparison studies, and Section 5 presents a high-level overview of the results, which is complemented by the Appendix detailing the performance comparison outcomes. In Section 6, we discuss what these findings mean, how they are applicable in industry, and present our recommendations for industry and research based on the findings. Section 7 concludes the study.

2. Database systems

2.1. Database system overview

A database is a collection of interrelated data, typically stored according to a data model. Typically, the database is used by one or several software applications via a DBMS. Collectively, the database, the DBMS, and the software application are referred to as a *database system* (Elmasri and Navathe, 2016, p. 7)(Connolly and Begg, 2015, p. 65). The separation of the database and the DBMS, especially in the realm of relational databases, is typically impossible without exporting the database in another format. In these situations, the database is often unusable by the DBMS, unless the database is imported back to a format understood by the DBMS. Possibly due to this inseparability, both the DBMS and the underlying database are often colloquially referred to simply as *database*. It is worth noting, though, that the former is a piece of software that *does*, while the other is a collection of data that *is*.

Fig. 1 shows a simplified example of a system where the components crucial for a database system and the scope of this study are emphasized. We refer to the components in the figure throughout this study. Several things are worth noting in considering the figure, as we have traded technical precision and comprehensiveness for ease of presentation by depicting only a single end-user, a single software application (some parts typically reside on the end-user's device, while others reside on a separate server), a single DBMS, single hardware components, and a single database. Furthermore, we have not illustrated other DBMS components such as access control, data structures such as metadata, or outputs such as query execution plans. The figure also adopts the view that the database resides in persistent storage — this is not always the case. Additionally, we have depicted merely a centralized database system in which neither the DBMS nor the database has been distributed across multiple nodes. These are willful omissions given the scope of this study.

2.2. Data models

Databases follow one or several data models, i.e., definitions of how and what data can be stored, and sometimes, what operations are available for data retrieval and manipulation. Data models may be conceptual, logical, or physical. Conceptual models such as the Entity-Relationship model (Chen, 1976) do not dictate how data should be stored, but are rather used to describe the interrelations and characteristics of the data. Logical data models such as the relational model (Codd, 1970) are related to how data is stored and presented, but often without describing how the data is physically stored, e.g., which computing node is responsible for storing the data, where the data is located on a disk, and what types of indices (i.e., redundant data structures which facilitate query performance) and physical data retrieval operators are available. One DBMS is not limited to using a single data model (Forresi et al., 2022).

There are several popular logical data models, some of which are inseparable from their underlying physical data models. One of the most prominent logical data models is the relational data model rooted in set theory (Codd, 1970). Relational DBMSs (RDBMS) follow many of the concepts introduced in the relational model. Many of the popular RDBMSs such as PostgreSQL and Oracle Database have adopted data structures from other logical data models as well (Lu and Holubová, 2019). What is common for effectively all modern RDBMSs is that they utilize Structured Query Language (SQL) (ISO/IEC, 2016a,b) to define data structures and to retrieve and manipulate data. Typically, RDBMSs also implement a strong data consistency model which dictates or allows that database operations grouped into a transaction must all succeed or all fail, data must follow defined business logic, successful transactions persist in storage, and concurrent transactions (cf. Bernstein and Goodman, 1981) must result in the same data as if the transactions were serial. At least the last rule can often be loosened in modern implementations to various degrees. These constraints are collectively referred to as the ACID consistency model (Haerder and Reuter, 1983).

NoSQL is an umbrella term for several data models, typically developed or popularized in the first decade of the 2000s (Grolinger et al., 2013). Contrary to the relational model, the data models within NoSQL typically have no formal definitions, and different NoSQL DBMSs implement different data models such as key–value (e.g., Redis), document (e.g., MongoDB), wide-column (e.g., Cassandra) and graph (e.g., Neo4J) (Davoudian et al., 2018; Reniers et al., 2017). Furthermore, these DBMSs often have a distinct query language developed to cater to the particular data structures available in the DBMS's implementation of a data model. While RDBMSs have favored data consistency (Chaudhry and Yousaf, 2020) by eliminating redundant data through logical database design, and through a strong consistency model, NoSQL DBMSs have generally adopted the opposite approach. In several NoSQL data models such as key–value pairs and documents, redundant data are stored at the cost of storage space (Hecht and Jablonski, 2011). This approach enables query languages to be simple (Dey et al., 2014), avoiding complex and potentially slow queries. Furthermore, consistency models are typically less strict than in RDBMSs (Stonebraker, 2010), which facilitates higher performance demanded by, e.g., web applications with a large number of concurrent users (Ramakrishnan, 2012).

Although NoSQL DBMSs popularized several database-related approaches such as non-strict database structures, data availability over data consistency, and relatively effortless database replication (i.e., data is copied over computing nodes) and sharding (i.e., data is divided between computing nodes) (Grolinger et al., 2013), some industry leaders such as Google deemed a strong consistency model and an expressive query language important enough to design a DBMS which incorporates features from both RDBMSs and NoSQL DBMSs (Corbett et al., 2013). These so-called NewSQL DBMSs use the relational model, often with extensions, SQL as their primary query language, and a distributed database architecture (Pavlo and Aslett, 2016). In addition to these three *main* categories of RDBMS, NoSQL, and NewSQL data models, others such as object stores (Kulshrestha and Sachdeva, 2014) and GPU-intensive (Suh et al., 2022) systems are used in specific contexts.

2.3. Query execution

The word *query* typically refers to query language statements that retrieve some data from the database. However, in this study, we use the word query to refer to any data retrieval and manipulation statement for brevity. In times it is necessary to differentiate between data retrieval and manipulation, we use appropriate terms such as *read operations* for data retrieval, and *write operations* for data insertion, updates, and deletes. In this subsection, we describe how queries are

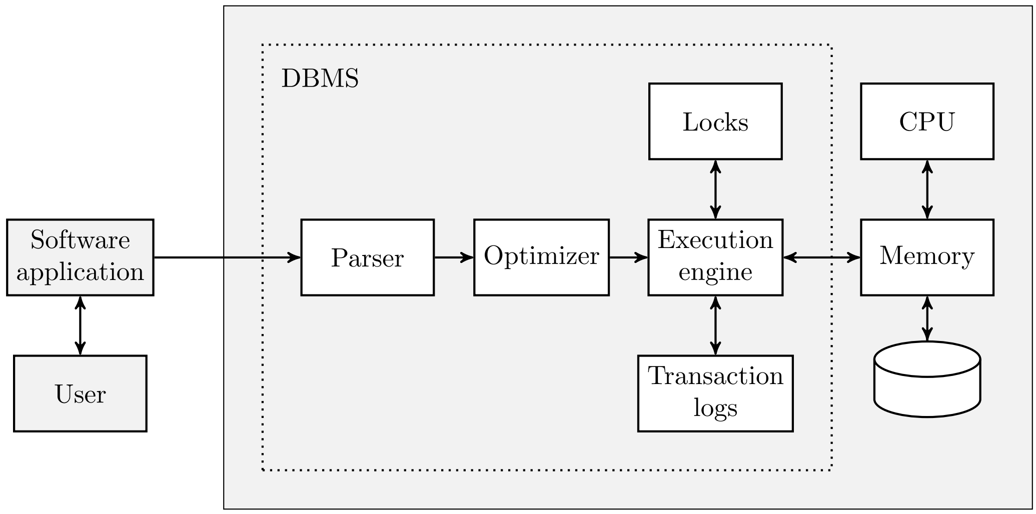

Fig. 1. A simplified view of a database system and the end-user with the emphasis on components relevant to this study; the arrows represent the flow of information from the end-user's device to the database residing in persistent storage; the flow of information back to the software application is not illustrated here; gray rectangles represent boundaries of physical devices.

executed, using mainly general (i.e., not specific to a single DBMS) literature from the domain of RDBMS query execution.

When a user — were it a human actor directly using a terminal, a transaction processing software application, or a database benchmark software — submits a query to a DBMS, a multitude of events must take place before the user receives feedback. Illustrated in a general fashion in Fig. 1, the query parser checks, among other things, that the query is syntactically valid (Hellerstein et al., 2007). If the query passes these (and other) checks, the query is translated to a lower-level presentation and passed to the query optimizer. The optimizer generates one or several query execution plans. These plans consist of physical operators for implementing, e.g., which physical data structures will be utilized in executing the query, and in RDBMSs in particular, how tables are joined together (Graefe, 1993). If several plans are generated, the optimizer evaluates which of these plans is the most effective in regards to, e.g., query execution time (Hellerstein et al., 2007). The accuracy of the optimizer relies on aspects such as database metadata (Christodoulakis, 1984), statistics of previous query executions, and the indices available (Chaudhuri, 1998). Generating effective query execution plans is a complex effort and takes time (Graefe, 1993; Chaudhuri, 1998), but once formulated, the plans can be re-used to a degree.

Next, the query execution engine implements the query execution plan, using the physical operators therein. Simplified, the data objects required by the query are typically first searched from a memory area called the buffer pool which is allocated and maintained by the DBMS. If some or all data is not found, the data is requested from disk. Before accessing the disk, many systems may additionally utilize other areas of memory to avoid disk access (Yang and Lilja, 2018).

Effectively all database systems function in an environment where multiple concurrent end-users use the database. This concurrency presents challenges particularly when the users execute write operations on the same database, e.g., when two or more users withdraw money from the same bank account, concurrently updating the balance (Bernstein and Goodman, 1981). To guarantee that the write operations do not interfere with each other in a way that would cause the data to not represent the real world, DBMSs typically implement concurrency control through locking or versioning data. Effectively, the simpler implementations of locking restrict data objects from being accessed by other operations while the data objects are being modified (Hellerstein et al., 2007). These locking mechanisms may be implemented to ensure that no anomalies happen, or with implementations that theoretically allow some anomalies (Berenson et al., 1995). Typically, the business domain dictates what types of anomalies are tolerated.

Finally, as strong consistency models often require that transactions persist in the database and that all or none of the operations in a transaction succeed, locking is typically complemented by transaction logs. These logs are written before write operations are committed to the database, and can be used in reversing earlier write operations if a later write operation in the same transaction fails. All these considerations discussed in this section play a significant role from a performance perspective, which is discussed in the next section.

Further reading on database systems: for readers interested in the basics of database systems, either the undergraduate level textbook by Connolly and Begg (2015), or Elmasri and Navathe (2016) are excellent albeit lengthy introductions covering the topic from several points of view and with the focus on RDBMSs. For readers interested in query processing, we point to studies by Chaudhury (1998), and Hellerstein, Stonebraker and Hamilton (2007). If you are interested in logical relational database design, the book by Date (2019) is an in-depth resource covering both formal and informal approaches. For a survey of literature on NoSQL data models, the study by Davoudian et al. (2018) is an accessible starting point.

3. Performance

3.1. Performance measurement

In general, performance is a measurement of how efficiently a software system completes its tasks. Performance is typically measured in response time, throughput (Hellerstein et al., 2007), or in some cases, utilization of computing resources (Cortellessa et al., 2011, p. 4). Response time is the time taken for a call in the system to traverse to some other part of the system and back. This is also sometimes called latency (Gunther, 2011, p. 10), and in the context of database systems, the response time may be measured as the response time to the first or the last result item (Graefe, 1993). In a broad perspective described in Fig. 1, the response time might be the time taken after the end-user sends a request to the software application (e.g., an online store), which passes the request to a DBMS, which returns a set of data to the software application, which finally presents the data to the end-user's device. In database benchmarking, however, response time might be measured by running the benchmark on the same device the DBMS and the database reside, effectively eliminating inter-device-induced performance drawbacks such as network latency (Patounas et al., 2020;

Delis and Roussopoulos, 1993) and firewalls, and mitigating the effects of other software running on the devices. Although DBMSs perform other tasks besides querying, querying is typically what is measured in DBMS performance testing (Dietrich et al., 1992). While response time is perhaps the least arduous performance metric to measure, it is not often enough for reliable measurement of transaction processing environments (Dietrich et al., 1992) (often dubbed online transaction processing, OLTP). That is, response time might be a metric better suited for long-running queries in decision support environments (often dubbed online analytical processing, OLAP), but as transaction processing environments often process a large number of concurrent transactions, response time alone might not reliably account for the effects of concurrent transactions, unless response time is measured as an average of multiple concurrent transactions.

Performance can also be measured by throughput, i.e., how many transactions the DBMS can execute in a given time frame. Throughput is often expressed as transactions per second (Dietrich et al., 1992) and requires a more sophisticated approach, e.g., benchmarking software. Again, throughput may be measured either locally (i.e., using only the hardware the DBMS and the database reside on), or over a network in case the database is distributed. Alternatively, throughput may be measured by connecting the benchmarking software to the software application, which simulates the throughput of the whole database system by accounting for, e.g., network and the software application (e.g., Kumar and Grot, 2022; Sundaresan et al., 2013). Such an approach arguably requires significantly more investment, but provides a holistic perspective on the performance of the whole system, also uncovering potential performance issues unrelated to the DBMS and the database. Finally, performance may be measured by resource utilization, either CPU time, I/O, memory allocation, or energy consumption (Graefe, 1993) in systems striving for energy-efficiency due to, e.g., limited battery power, or due to environmental concerns (Guo et al., 2022).

In summary, we might consider the measurement of throughput a process that typically requires a simulation of some level, and the measurement of response time as an exact or approximated mathematical method. The former approach requires relatively high investments into the development of such simulations (Cortellessa et al., 2011, p. 142), while the latter often relies on a set of assumptions that do not necessarily reflect real-world scenarios due to inaccuracies in predicting what the real-world scenario ultimately is and how it can change.

3.2. Factors affecting performance

Hardware: An intuitive factor in performance is the power of hardware (Osterhage, 2013, p. 1), and while it is true that most of the local response time is attributed to time taken by CPU processing, memory and disk access, and software waiting for other tasks to complete (Cortellessa et al., 2011, p. 5), first investing in software performance rather than hardware performance is often more cost-effective. That being said, it is generally accepted that memory access is at least four orders of magnitude faster than disk access (e.g., Gunther, 2011, p. 42). That is, if memory access takes minutes (nanoseconds), disk access takes months (milliseconds). These numbers are largely dependent on the speed of memory and the type of disk, but paint a picture of how zealously DBMS optimization strives to minimize disk access. Since memory is typically more expensive than disk storage, keeping the whole database in memory is often unfeasible. Additionally, the underlying hardware is important, as, e.g., some DBMSs have been shown to utilize multi-processor or multi-core environments more effectively than others (Tu et al., 2013). Intuitively, how well a DBMS can exploit parallelism affects the performance of query execution (Tallent and Mellor-Crummey, 2009; Tözün et al., 2013). Ultimately, performance measurement is about gains or losses in percentages, not in, e.g., response times.

Data models: Data models described in Section 2.2 have indirect effects on DBMS performance. Relational databases often follow design guidelines that strive to minimize redundancy to eliminate potential data anomalies caused by redundant data (Codd, 1972, 1975), and to minimize the need for storage space, which in turn typically causes queries to run slower due to a larger number of table joins. In contrast, different NoSQL data models — especially key–value, document, and wide-column — follow design guidelines according to which data structures are designed to efficiently satisfy predetermined business logic queries, with the elimination of redundant data being a secondary concern (Davoudian et al., 2018). It follows that because many NoSQL data structures are designed to serve queries, queries are typically simple (Dey et al., 2014), and their execution requires less computational resources than complex queries in relational databases. As discussed in Section 2.3, locking data objects (both on disk and in memory, and both primary data structures as well as indices), logging write operations, and how memory is managed by the DBMS all play a significant role in DBMS performance (Hellerstein et al., 2007; Stonebraker, 2010). For example, preventing write operation-induced anomalies is a costly action, and the level of granularity of database locks presents significant considerations on write operation performance, which is largely dictated by the ratio of read and write operations.

Distribution: Write operations in distributed configurations pose non-trivial challenges to both performance and data consistency (Delis and Roussopoulos, 1993). In distributed database systems, effectively all transactions must choose either data consistency or data availability (Brewer, 2012; Gilbert and Lynch, 2002). The former guarantees that the data the end-user receives are not stale, with the cost of performance, while the latter guarantees to a degree that the end-user receives data faster, but with no guarantees that the dataset received is the most recent. The preferred approach is largely dictated by business logic.

DBMS and OS parameters: Moving from data models and database system distribution to lower levels of abstraction, operating system (OS) and DBMS parameters and their interrelationships (e.g., page size) can have direct or indirect effects on performance (Dietrich et al., 1992). Additionally, DBMS parameters such as the amount of memory the DBMS is allowed to use for data processing is typically closely related to the amount of memory available. Furthermore, as a query is sent to the optimizer (cf. Fig. 1), it depends on the DBMS internals how efficiently the optimizer can select the most efficient physical operations to implement the query, and what physical operations are available to the optimizer in the first place (Chaudhuri, 1998). For example, MySQL implemented only one physical operation for table joins until 2018,[1] limiting the number of options the optimizer could choose from. Regarding query optimization, the optimizers of RDBMSs in particular are relatively mature and can spot some unnecessary complications in queries, while overlooking others (Brass and Goldberg, 2006). Despite the benefits brought by the optimizers, some queries are inherently slow and can only be optimized through query rewrites.

Physical database design: Last, but definitely not least, physical database design plays a key role in DBMS performance. It has been argued that performance bottlenecks are difficult to find in large systems (Ammons et al., 2004), and that efficiency is gained by focusing on the vital few areas instead of the trivial many (Juran and De Feo, 2010, p. 450). One of the most vital areas in database systems is physical design. In relational databases, efficient physical design is largely achieved through indices, and in NoSQL databases, typically through database distribution over computing nodes. In contrast to a holistic system overview, performance bottlenecks may be easier to find in queries, since many DBMSs provide detailed information on query execution (Fig. 2). PostgreSQL (Fig. 2(a)) lists the physical operations used to execute the query, which of the operations took the most time units, and which indices, if any, were used. For example, it can be seen in Fig. 2(a) that the sequential scan on line 12 accounted for

[1] https://dev.mysql.com/doc/refman/5.6/en/explain-output.html

```
             QUERY PLAN
----------------------------------------------------------------------------------------------------
Hash Join (cost=3802.67..5996.69 rows=469 width=67) (actual time=183.354..189.409 rows=473 loops=1)
 Hash Cond: (o.customerid = c.customerid)
 -> Bitmap Heap Scan on orders o (cost=167.89..2341.57 rows=7753 width=52) (actual time=1.446..5.581 rows=7925 loops=1)
    Recheck Cond: (((paymenttype)::text = ’OC’::text) AND (orderdate > to_date(’20101010’::text, ’YYYYMMDD’::text)))
    Heap Blocks: exact=1772
    -> Bitmap Index Scan on ord_paymenttype_orderdate (cost=0.00..165.95 rows=7753 width=0) (actual time=1.182..1.182
rows=7925 loops=1)
       Index Cond: (((paymenttype)::text = ’OC’::text) AND (orderdate > to_date(’20101010’::text, ’YYYYMMDD’::text)))
 -> Hash (cost=3491.49..3491.49 rows=11463 width=15) (actual time=181.848..181.848 rows=11464 loops=1)
    Buckets: 16384 Batches: 1 Memory Usage: 672kB
    -> Seq Scan on customers c (cost=0.00..3491.49 rows=11463 width=15) (actual time=0.031..178.834 rows=11464 loops=1)
       Filter: ((firstname)::text ~~* ’ma%’::text)
       Rows Removed by Filter: 178095
 Planning Time: 1.964 ms
 Execution Time: 189.543 ms
(14 rows)
```

(a) PostgreSQL query execution plan shows the time units anticipated and taken by each phase of the query execution

```
Tracing session: 4e5c44b0-6714-11ed-97f4-0597e908e9d9
activity                                                                  | source    | source_elapsed
--------------------------------------------------------------------------+-----------+---------------
 Execute CQL3 query                                                       | 127.0.0.1 |              0
 Parsing SELECT * FROM movies_by_tag; [Native-Transport-Requests-1]       | 127.0.0.1 |           1795
 Preparing statement [Native-Transport-Requests-1]                        | 127.0.0.1 |           2003
 Computing ranges to query [Native-Transport-Requests-1]                  | 127.0.0.1 |           2308
 Submitting range requests on 17 ranges with a concurrency                |           |
     of 7 (14.4 rows per range expected) [Native-Transport-Requests-1]    | 127.0.0.1 |           2562
 Executing seq scan across 1 sstables for                                 |           |
  (min(-9223372036854775808), min(-9223372036854775808)] [ReadStage-2]    | 127.0.0.1 |           7768
 Submitted 1 concurrent range requests [Native-Transport-Requests-1]      | 127.0.0.1 |           8006
 Read 3 live rows and 0 tombstone cells [ReadStage-2]                     | 127.0.0.1 |          21272
 Request complete                                                         | 127.0.0.1 |          21747
```

(b) Cassandra query execution plan; some output has been omitted for brevity

Fig. 2. Query execution plans illustrating the physical operators such as *hash join* and *seq scan* chosen by the optimizer.

approximately 94% of the execution time of the whole query (178 *time units* out of 189 ms), probably because the query fetched a large number of records from the database. The query could be optimized by, e.g., selecting a smaller number of records, and showing the results to the end-user by paging them, i.e., showing a subset of results first, and fetching more later if necessary. In NoSQL systems, the query optimizer plays a smaller role due to typically less expressive query languages (cf. Fig. 2(b)). Some NoSQL systems such as Cassandra do not permit the execution of queries that do not utilize the physical structures effectively.

3.3. Database performance benchmarks

There are several database performance benchmarks available, each typically consisting of a sample database and a workload that simulates how the database could be used (Difallah et al., 2013; Qu et al., 2022b). The benchmarks usually measure the efficiency of querying while taking into account factors such as concurrency but disregarding other DBMS tasks such as efficiency in data structure definition or bulk loading (Dietrich et al., 1992).

In the domain of relational databases, the Transaction Processing Council (TPC) benchmarks (e.g., Gray, 1992) are perhaps the most utilized (Dreseler et al., 2020; Tözün et al., 2013), and test the throughput of the DBMS with various parameters. For example, the TPC-A benchmark simulates a database of a bank with four tables and with one transaction, the TPC-B benchmark a database of a wholesale supplier with nine tables and with five transactions, and the TPC-E benchmark a brokerage database with 33 tables and 12 transactions. All these benchmarks have the option for simulating strong consistency, and while TPC-A and TPC-B have transactions typical for transaction processing, TPC-E includes also decision support transactions (Tözün et al., 2013). TPC-A simulates human end-user thinking by waiting between transactions, as a human arguably would wait between clicks in an online bank. TPC-B, on the other hand, does not wait and can be used as a precursor for TPC-A in adjusting DBMS parameters (Dietrich et al., 1992). Alternatively to transaction processing, TPC-H benchmark measures the performance of a DBMS in decision support (Barata et al., 2015; Dreseler et al., 2020).

In the more general DBMS domain, the Yahoo! Cloud Serving Benchmark (YCSB) is a framework for benchmarking transaction processing in systems with different data models and architectures (Cooper et al., 2010). Due to its extensibility, YCSB can be adapted to different NoSQL data models. YCSB contains different workloads, each with a different ratio of read and write operations. YCSB and its extensions such as YCSB+T typically utilize transactions which consist of single operations and do not enforce strong consistency (Qu et al., 2022b; Dey et al., 2014). The benchmarks described above are by no means an exhaustive list but cover the most popular benchmarks (cf. Section 2.1). Other benchmarks include LUBM (Guo et al., 2005), OLTP-Bench (Difallah et al., 2013), and JOB (Leis et al., 2015). Regardless of the data model and DBMS, transaction processing benchmarks have typically been the de facto method of comparing different DBMSs and hardware (Tözün et al., 2013).

Further reading on performance: for readers interested in physical database operations and query execution from a performance perspective, Graefe (1993) provides an in-depth, DBMS-independent survey. For more information on physical database design, especially indices and how they work, the book by Lightstone et al. (2010) is a detailed and descriptive source. For a practical and concise guide on SQL query optimization, we point readers towards Winand’s (2012) book. Regarding NoSQL DBMS optimization, we suggest referring to the manual of the DBMS of your choice, and always making sure that the source of information is current, as NoSQL systems tend to evolve rapidly.

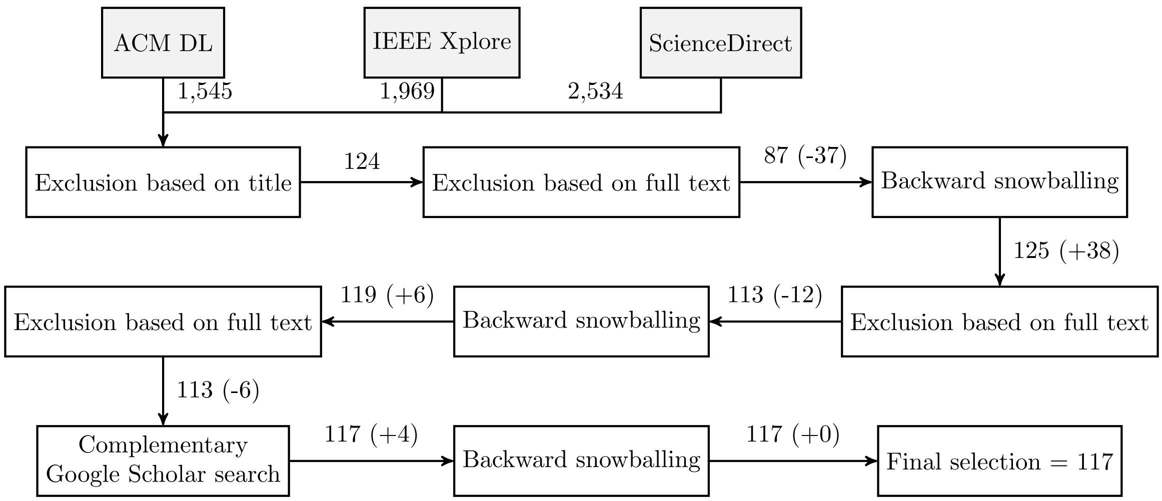

Fig. 3. The study selection process; the numbers refer to the number of primary studies selected in each stage of the process.

Table 1
Search strings.

Database	Search string
ACM DL	[Abstract: performance] AND [Abstract: comparison] AND [[Abstract: database] OR [Abstract: dbms]] AND [Publication Date: (01/01/2000 TO 03/31/2022)]
IEEE Xplore	("Abstract":performance AND "Abstract":comparison AND ("Abstract":database OR "Abstract":dbms))
ScienceDirect	Title, abstract, keywords: performance AND comparison AND (database OR dbms)
Google Scholar	database performance comparison

4. Study selection

4.1. Process and criteria

The DBMSs in this study were selected based on the selected primary studies. That is, we did not choose, e.g., the most popular DBMSs to include, but reported the DBMSs yielded by the primary studies. The results herein may be considered the most popular DBMSs in terms of benchmarking reported in scientific literature. Fig. 3 describes the primary study selection process starting from ACM Digital Library, IEEE Xplore, and ScienceDirect, complemented by subsequent Google Scholar searches. The search strings are detailed in Table 1. To account for potentially missing relevant studies, we conducted three rounds of backward snowballing (i.e., following the lists of references in selected studies), until snowballing revealed no additional studies. A total of 117 primary studies comparing DBMS performance were selected.

Table 2 describes our inclusion criteria applied in the primary study selection. The first four criteria are related to bibliographic details, while the last three criteria are concerned with article focus and content. Regarding criterion *#3*, we excluded academic theses and dissertations (e.g., Coates, 2009) due to the fact that they are typically not peer-reviewed. We also excluded white and gray literature for the same reason, and because those studies are often written or published by partial parties, e.g., DBMS vendors.

We only selected studies that compared query (i.e., retrieving or modifying data) execution performance, not regarding e.g., database replication performance (Elnikety et al., 2006) or performance of different join operations (Kim and Patel, 2010). We also excluded studies that compared a single DBMS performance in different configurations such as hardware, replication strategy, database structure, or query language (Holzschuher and Peinl, 2013) and studies that compared a DBMS with different data-related platforms (Purbo et al., 2020). Studies that reported pseudonymized DBMS names were also excluded. Finally, we only included studies that reported results based on at least seemingly objective metrics and empirical results. That is, studies simply stating the opinions of the authors such as "based on our experiences, we believe MySQL is faster than SQL Server" were not considered.

4.2. Selected studies

The selected 117 primary studies compared the performance of a total of 44 different DBMSs. We categorized these DBMSs into three top-level types defined and discussed in Section 2.2: RDBMSs, NoSQL systems, and NewSQL systems. Five DBMSs not clearly pertaining to any of these three categories were categorized under *other* systems (Table 3). It is worth noting that these DBMS types are not always clear-cut due to the lack of specificity and changing nature of the definitions, and should be interpreted as merely means to compartmentalize the results of this study into a more readable form. Five selected primary studies did not report results implying the performance of one DBMS over another (Padhy and Kumaran, 2019; Schmid et al., 2015b; KumarDwivedi et al., 2012; Faraj et al., 2014; Jing et al., 2009).

Fig. 4 shows the distribution of publication years and the types of DBMSs discussed in the selected studies. Although our criteria allowed for studies from the year 2000, the first studies selected were published in 2008. The figure shows that generally, there is a somewhat constant number of DBMS performance comparison studies each year. It is worth noting that one study may pertain to several types of DBMSs.

5. Performance comparison results

The most popular DBMS performance comparisons compared one or several RDBMSs to one or several NoSQL systems, one NoSQL system to another NoSQL system, or one RDBMS to another RDBMS, respectively. A total of 48 studies compared solely read performance, while 6 studies compared solely write performance. The rest of the studies compared both read and write performance, with the exception of two studies (Cheng et al., 2019; Nepaliya and Gupta, 2015) which were unclear whether they compared write operations. All comparisons and their results per DBMS type are summarized in Fig. 5.

Fig. 6 presents an overview of which DBMSs and DBMS types the primary studies compared. The figure perhaps conveys how both *other* and NewSQL systems are typically compared within their respective DBMS type groups, while RDBMS and NoSQL systems are both

Table 2
Primary study selection criteria.

#	Inclusion criterion
1	Article is written in English.
2	Full article can be accessed.
3	Article is published in a scientific journal, or conference or workshop proceedings.
4	Article is published between 2000 and March 2022.
5	Article focus is on query language statement execution performance comparison.
6	Article focus is on comparing the performance of two or more different DBMSs.
7	Article is based on at least seemingly objective metrics.

Table 3
DBMSs discussed in this study divided into four types.

DBMS type	DBMSs
RDBMS	Access, Azure SQL, Interbase, DB/2, H2, Hive, MariaDB, MySQL Cluster, MySQL, Oracle Database, PostgreSQL, PostgresXL, SQL Server, SQLite
NoSQL	ArangoDB, Azure Document Database, Cassandra, Couchbase, CouchDB, Elasticsearch, Firebase, HBase, Hypertable, memcached, MongoDB, Neo4J, Oracle NoSQL, OrientDB, RavenDB, Redis, RethinkDB, Riak, Scalaris, Tarantool, Voldemort
NewSQL	CockroachDB, MemSQL (now known as SingleStoreDB), NuoDB, VoltDB
Other	BlazingSQL, Caché, Db4o, OmniSciDB, PG-Strom

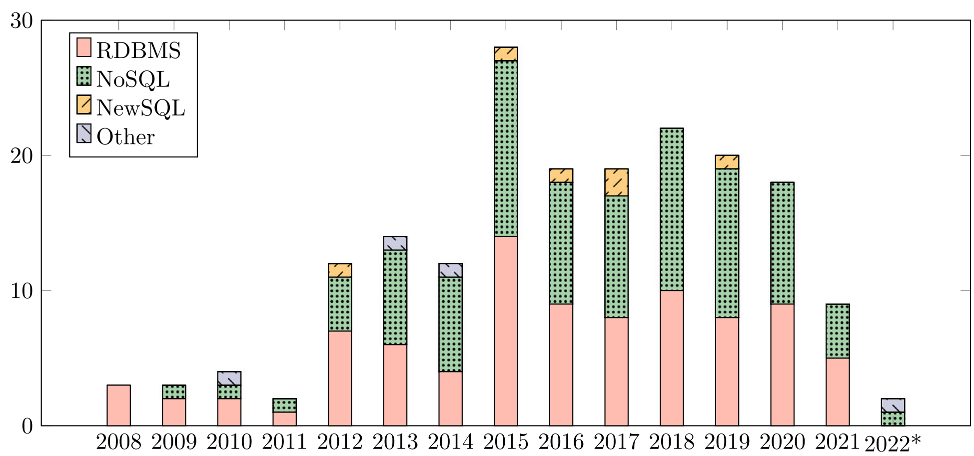

Fig. 4. The number of publications by publication year and DBMS type; the year 2022 was only considered until March.

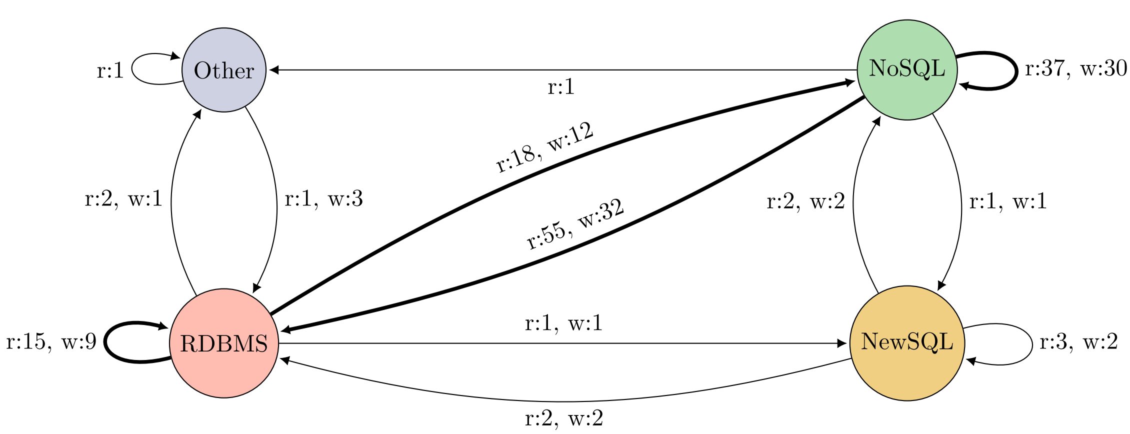

Fig. 5. DBMS performance comparisons overview; a directed edge from node *a* to node *b* represents the number of studies according to which a system of type *a* outperformed a system or systems of type *b* in (r)ead and (w)rite operations, e.g., a NoSQL system outperformed a NewSQL system in read operations in one study, and in write operations in one study; thicker edges visualize the most popular comparisons.

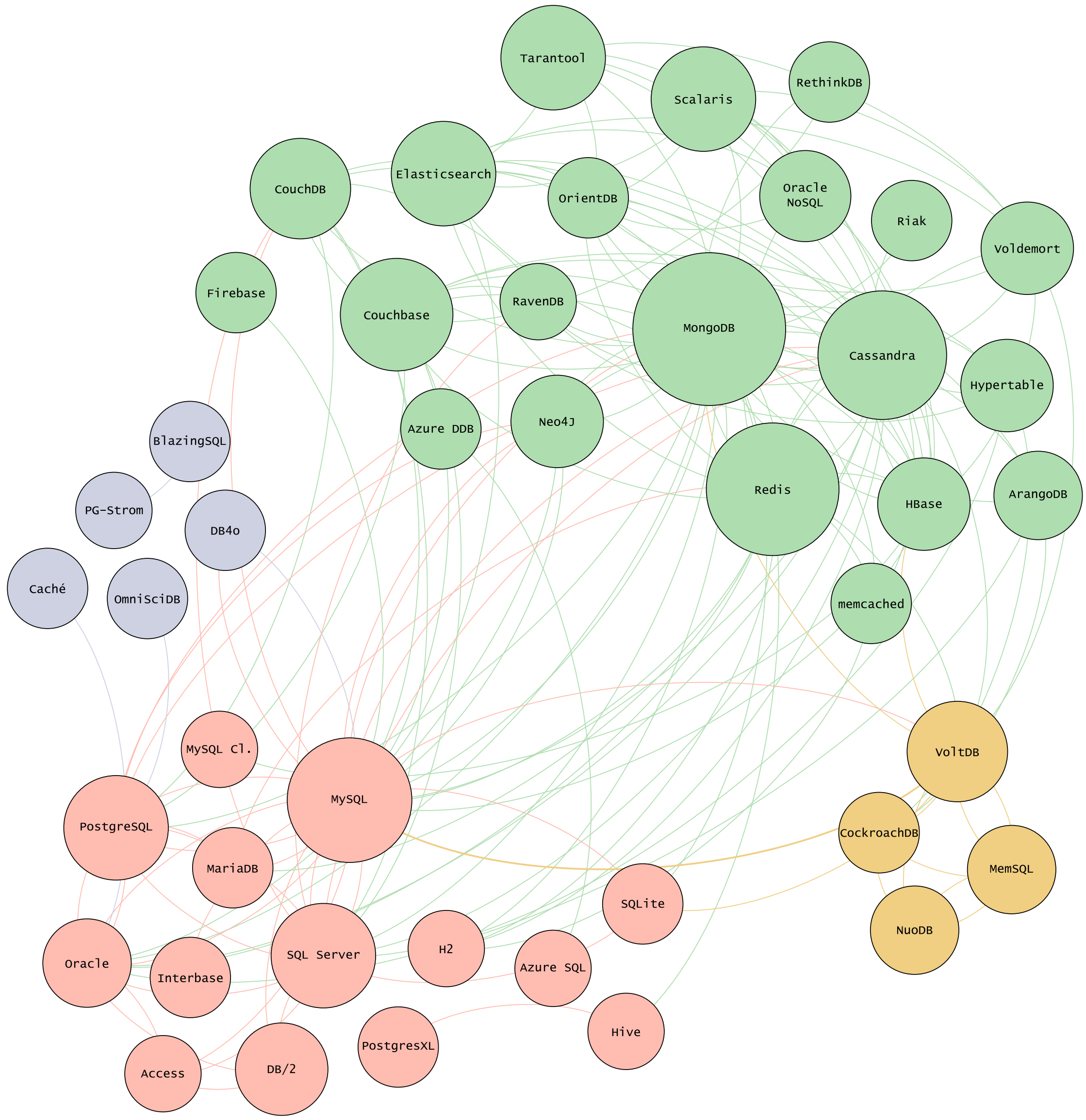

Fig. 6. An overview of read operation performance comparisons between NoSQL systems (green, upper right), NewSQL systems (yellow, lower right), RDBMSs (red, lower left), and other systems (blue, upper left); a clockwise turning edge from node *a* to node *b* depicts node *a* outperforming node *b*, and the color of the edge corresponds to the type of the outperforming node, e.g., Caché outperforms PostgreSQL according to one or several studies; the size of a node represents out-degree, i.e., larger nodes have outperformed more systems than smaller nodes.

compared within their respective groups as well as with each other. Additionally, the size of the nodes such as MongoDB, Redis, Cassandra, and MySQL show that these DBMSs typically outperform the DBMSs they are compared to. Due to their length, the detailed results from the primary study comparisons are presented in the Appendix, which includes tables detailing which DBMSs outperformed which.

Regarding the benchmarks defined in earlier scientific literature, the most popular was YCSB, which was utilized by 15 primary studies (approximately 13%) (Abramova and Bernardino, 2013; Abramova et al., 2014a,b; Gandini et al., 2014; Schreiner et al., 2019; Seghier and Kazar, 2021; Yassien and Desouky, 2016; Abubakar et al., 2014; Kashyap et al., 2013; Swaminathan and Elmasri, 2016; Tang and Fan, 2016; Klein et al., 2015; Araujo et al., 2021; Hendawi et al., 2018; Rabl et al., 2012). The second most popular benchmark was the TPC-H benchmark and its variations, utilized by five primary studies (4%) (Almeida et al., 2015; Fotache and Hrubaru, 2016; Oliveira and Bernardino, 2017; Suh et al., 2022; Vershinin and Mustafina, 2021). It is worth noting, though, that two of the studies (Oliveira and Bernardino, 2017; Vershinin and Mustafina, 2021) seemed to have executed the *queries* of TPC-H, instead of running the benchmark and accounting for, e.g., the effects of concurrent transactions. One primary study utilized the OLTP-Bench benchmark (Tongkaw and Tongkaw, 2016), one the LUBM benchmark (Franke et al., 2013), and one, in addition to TPC-H, the JOB benchmark (Suh et al., 2022). Regarding the benchmarks formulated by the primary study authors, 25 primary studies (21%) reported using ad hoc queries instead of earlier defined benchmarks to compare the performance of DBMSs. These queries were defined verbatim in the primary studies. In contrast, 70 of the primary studies (60%) compared DBMS performance using undisclosed ad hoc queries, likely formulated by the study authors. In other words, 22 primary studies (19%) used some type of earlier defined database benchmarking suite. The performance tests of these 22 primary studies and what aspects of the environment they reported are detailed in Table 4.

Table 4
An overview of primary studies using previously defined benchmark software and which aspects of the testing environment they explicitly disclosed; performance measurements abbreviated as ET (execution time) and TP (throughput).

Study	Explicitly reported				Benchmark	Measurement	Nodes
	DBMS versions	Hardware	DB structure	DBMS parameters			
Abramova and Bernardino (2013)	yes	yes	no[a]	no	YCSB	ET	1
Abramova et al. (2014b)	yes	yes	no[a]	no	YCSB	ET	1
Abramova et al. (2014a)	yes	yes	no[a]	no	YCSB	ET	1
Abubakar et al. (2014)	no	no	no[a]	no	YCSB	ET	1
Almeida et al. (2015)	no	yes	logical only	no	Star Schema Benchmark	ET	1
Araujo et al. (2021)	yes	yes	no[a]	no	YCSB	ET, TP	2
Fotache and Hrubaru (2016)	no	yes	logical only	no	TPC-H	ET	5
Franke et al. (2013)	yes	yes	no	no	LUBM-based	ET	9
Gandini et al. (2014)	no	yes	no[a]	no	YCSB	ET, TP	2-9
Hendawi et al. (2018)	yes	yes	no[a]	no	YCSB	ET, TP	8
Kashyap et al. (2013)	yes	yes	no[a]	no	YCSB	ET, TP	up to 5
Klein et al. (2015)	yes	no	no[a]	no	YCSB	ET, TP	9
Oliveira and Bernardino (2017)	no	yes	logical only	no	TPC-H	ET	1
Rabl et al. (2012)	yes	yes	no[a]	no	YCSB	ET, TP	16 and 24
Schreiner et al. (2019)	no	yes	no[a]	yes (default)	YCSB, Voter	ET, TP	3
Seghier and Kazar (2021)	yes	yes	no[a]	no	YCSB	ET	1
Suh et al. (2022)	yes	yes	no[a]	yes (default)	TPC-H	ET	3
Swaminathan and Elmasri (2016)	yes	yes	no[a]	no	YCSB	TP	up to 14
Tang and Fan (2016)	yes	yes	no[a]	no	YCSB	ET, TP	4
Tongkaw and Tongkaw (2016)	yes	yes	logical only	no	Sysbench, OLTP-Bench	TP	1
Vershinin and Mustafina (2021)	no	yes	logical only	no	TPC-H	ET	1
Yassien and Desouky (2016)	yes	yes	no[a]	no	YCSB	ET, TP	1

[a] The YCSB benchmark defines a single-table with n columns (or loose equivalents in non-relational data models).

6. Discussion

6.1. General discussion

The difficulty of rigorous performance testing is perhaps one of the root causes of why optimization is difficult, and several studies have highlighted the complexity of performance testing due to, e.g., the effects of DBMS parameters (Purohith et al., 2017), testing environment settings (Wang et al., 2022), and how well the data in the performance test database reflects the real application data (Qu et al., 2022a). Is it also important whether an impartial actor has carried out the performance test, or whether the test results are published e.g., by a DBMS vendor (DeWitt and Levine, 2008). However, this is sometimes difficult to assess and can be mitigated by simply explicitly reporting the test so that it can be replicated and verified by others.

Despite the fact that we were aware of some DBMS performance comparison studies as they have been touched on in previous works, we were surprised by the extent the few examples presented in the previous works (Raasveldt et al., 2018; Wang et al., 2022) generalize to so many studies on the subject. For example, in read operations, MongoDB outperforms Cassandra according to ten studies, Cassandra outperforms Redis according to four studies, and Redis outperforms MongoDB according to six studies (cf. Appendix), leading to a situation of $Mo > Ca > Re > Mo$, where MongoDB is both the best and the worst performing DBMS. Furthermore, as discussed in Section 5, few of the selected studies reported the test setting in enough detail for replication. Unfortunately, without sufficient details for replicating an experiment, such experimental results can claim any outcome (Raasveldt et al., 2018). One aspect that was typically reported was some details about the hardware the test was run on, i.e., processor make and model, clock rate, memory size, and disk size. Without other details about the DBMS parameters, parallel execution, etc., these details are inconsequential. Despite the importance of the topic of DBMS performance comparisons, with the exception of one study (Rabl et al., 2012), no primary studies were published in major data management fora such as ACM SIGMOD or VLDB.

6.2. Considerations for industry

6.2.1. Consider the environments in performance testing studies

If the environment in which the performance testing was carried out does not provide sufficient details, whatever the study states, you may interpret the results as if they do not generalize to other environments. That is, if you are in the process of deciding on a DBMS for your application, or perhaps considering changing one DBMS to another, consider whether the performance comparison study you are reading presents a similar use case. Compare your business domain to that presented in performance comparison studies, remembering that a single, sometimes even a seemingly inconsequential parameter (cf. e.g., data types in SQLite in Purohith et al., 2017) may change the results. DeWitt and Levine (2008) aptly describe performance comparisons as the *maximum* potential performance gain of one DBMS over another. The performance gain in your particular environment might be less, or it might be that the DBMS that performed better in the comparison performs worse in your environment.

One important aspect of the environment is the physical setup. Different hardware has been shown to affect DBMS performance, as some DBMSs exploit parallelism more efficiently than others (Marek and Rahm, 1992; Jiang et al., 2010), effectively meaning that if a test was performed on one single-core CPU, the results might not generalize to distributed and multi-CPU environments. Additionally, different hardware aspects such as the relative sizes of different CPU memory caches may significantly affect DBMS performance, making performance comparisons between different hardware a complex task (Ailamaki et al., 1999; Wang et al., 2022). In distributed environments, which were rarely tested in the primary studies, it is worth considering whether data availability is prioritized over data consistency, as the latter setup is typically significantly slower. Benchmarks that simulate concurrent users should also be considered separately from performance tests that merely execute queries sequentially. Concurrency introduces several challenges, many of which severely affect performance (Wang et al., 2022). For example, SQLite uses database locking on a level of granularity which makes concurrent writes slow, but this has no negative effects on single-user writes (Obradovic et al., 2019). Unfortunately, some studies have shown that developers do not widely understand concurrency-related security aspects (Warszawski and Bailis, 2017), and that concurrency-related performance problems are understudied (Yu and Pradel, 2018). Some have even stated that the research has not been focusing on relevant issues (Pavlo, 2017).

Intuitively, different business domains have different databases and they are used in different ways. For example, in some domains, the end-users typically read data, while in others, write operations are more common. The ratio of read and write operations in a performance test plays a crucial role, as some DBMSs are specifically designed for specific workloads (Cooper et al., 2010). The credibility of testing results is also related to how well the test database and data therein represent the target environment (Qu et al., 2022a). Furthermore, in business domains such as online stores, there are typically popular products, and thus the data related to them are targets of a relatively large number of database operations. For generalizable benchmarking results, the benchmark must account for such skewness in database use, rather than, e.g., randomly querying data objects. It is also worth considering how the performance tests have tested performance. For example, is your application about inserting 10,000 rows in bulk, but one row at a time randomly generated by the application? If it is not, you should not consider this type of benchmark results as an indication of how well one DBMS performs compared to another in your particular business context. It is also worth considering that decision support benchmarks such as TPC-H test performance in environments that can be fundamentally different from transaction processing environments. Finally, even similar business domains can have a myriad of different technical implementations.

We have discussed some of the particulars involved in database system design in this subsection, and in Sections 2 and 3, from which one can infer what has often been repeated in database system research: the environments and their optimization is a task so complex (Graefe, 1993; Cooper et al., 2010) that DBMS optimization is a whole profession (Raasveldt et al., 2018). It follows that there are several threats to rigorous DBMS benchmarking. Even though RDBMS optimization is widely and deeply studied in both academic and industry contexts, RDBMS optimization remains a complex task. In the domain of NoSQL DBMSs, there exist far fewer scientific studies simply due to the age of the NoSQL DBMSs, and the heterogeneity of NoSQL data models. Additionally, there are several querying anti-patterns to avoid, such as performing joins in the software application instead of the DBMS, or paging query results by utilizing ordering, limiting and offset. All these points considered, a reader of a performance comparison study must trust that the performance comparison study writers have been able to optimize the database systems to a similar degree for the performance comparison results to be credible. This requires particularly specific, in-depth expertise when DBMSs with more than one data model are compared. Furthermore, decades of benchmarking software development by entire councils (e.g., TPC) cannot simply be skipped by writing a set of (often arbitrary) queries, running them on two or more DBMSs in a single-user environment, recording response times, and consequently stating that one DBMS is faster than another. Although this was the case in over 80% of the selected primary studies, we do not consider this sufficient.

In summary, if it is possible that changing even one of the environmental aspects discussed above may affect the performance test results significantly, it seems reasonable to argue that, no matter how many DBMS performance comparison studies state that one DBMS outperforms another, these DBMSs were not tested in an environment that is the same as your environment, and thus have little concern in the decision of which DBMS is performance-wise the best fit for your environment.

6.2.2. Consider other aspects besides performance

There are other aspects besides response time or throughput to consider when choosing a DBMS. Performance gains, such as those provided by many NoSQL systems, rely heavily on redundant data to minimize the complexity of queries, thus providing faster response times. Naturally, storing redundant data increases the cost of storage, and may lead to data inconsistencies. Another comparison perspective is related to the features provided by the DBMSs compared. Intuitively, a DBMS that is tailored for a specific purpose outperforms a general-purpose DBMS (Raasveldt et al., 2018; Stonebraker et al., 2007). For example, one primary study (Bartoszewski et al., 2019) noted that while MongoDB outperformed PostgreSQL/PostGIS in most of the tests, MongoDB provides only a subset of the geospatial operations provided by PostGIS. If the rest of the operations needed by the business domain need to be implemented in the software application, it is not realistic to assume that such task is either trivial to implement, nor trivial to implement in a way that outperforms the solutions offered by existing DBMS features.

Another consideration is the availability of suitable workforce, which is closely related to the DBMS technology and its maturity. It is not surprising that as query languages such as SQL have been a topic of effectively all information technology-related curricula in higher education for several years (Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, 2013; The Joint Task Force on Computing Curricula, 2015), there is a relatively large number of professionals fluent in SQL, as opposed to new query languages. Some studies have also shown that strong consistency models (Corbett et al., 2013) and the SQL language (Cass, 2022) are desired as skills as well as features in a DBMS. That is, it is worth considering how feasible it is to implement a database system with each specific technology, and DBMS performance is only one of the important aspects to consider.

Finally, as the primary studies typically considered performance in terms of response time or throughput, we have approached the topic from a similar viewpoint. However, as discussed in Section 3.1, performance may be measured by the usage of computing resources, which can be a goal conflicting with response time (Chaudhuri, 1998). It is typical that increasing parallelism through multiple CPUs lowers response time, but increases the total amount of work due to the parallelism overhead (Osterhage, 2013, p. 13). Finally, it has been shown that migrating data from one DBMS to another is all but trivial, and prone to fail due to a lack of clear methodologies (Thalheim and Wang, 2013) — especially when the DBMSs differ in data models and query languages (Kim et al., 2018). Therefore, migrations such as RDBMS ⇔ RDBMS or RDBMS ⇔ NewSQL are arguably less complex than migrations such as NoSQL ⇔ NoSQL, RDBMS ⇔ NoSQL or NewSQL ⇔ NoSQL.

6.3. Considerations for research

6.3.1. Consider using existing guidelines for testing and reporting

Database benchmarking guidelines are not a novel invention in database system research and have been described in detail (Gray, 1992) and in short (Dietrich et al., 1992) in the early 1990s, and as a reader-friendly checklist later (Raasveldt et al., 2018). Additionally, benchmarking pitfalls have been discussed in numerous studies in respected database systems fora (Wang et al., 2022; Dreseler et al., 2020). Based on the primary studies, however, neither of these lines of research has been widely applied in practice. Database benchmarking has been argued to be difficult (Raasveldt et al., 2018), as environmental parameters such as the nature of data (Qu et al., 2022a), DBMS parameters (Wang et al., 2022), and data types (Purohith et al., 2017) can all have significant impacts on performance testing results. Furthermore, benchmarking tools have received critique (Reniers et al., 2017; Grolinger et al., 2013) despite the fact that some of the tools have been under development for decades. Therefore, we urge researchers, at the very least, to consider whether using a performance test suite of one's ad hoc queries is credible when well-known performance benchmarks are freely available.

As for reporting, Raasveldt et al. (2018) provide a 24-point checklist for fair benchmarking. Some of the points are concerned about how performance is tested, and others about how the testing is reported. A performance comparison that cannot be replicated may present whatever results (Raasveldt et al., 2018). Furthermore, an empirical study without reproducible evidence should be considered an opinion of the authors, rather than an empirical study. Indeed, at the start of the NoSQL movement, we have witnessed several studies with high praise for the strengths of different NoSQL products, yet with little or no critical notions addressing the acknowledged shortcoming of such DBMSs. Therefore, we would caution the reader from inferring from these results that one DBMS performs better than another. Rather, each such argument should be carefully scrutinized and interpreted in a specific context, like in the primary study assessing the performance of GPU DBMSs (Suh et al., 2022), in which performance between DBMSs was compared, but the comparison was merely one aspect of the study.

6.3.2. Consider a different approach to DBMS-DBMS testing

Especially for a junior researcher, comparing the performance of one DBMS to another may seem like a relatively simple research setting to both carry out, and also justify based on the prevalence of the DBMS industry. We hope that the arguments presented in previous studies as well as here have highlighted that neither of these points are as clear-cut. Following the guidelines (e.g., Raasveldt et al., 2018) can make performance testing a time-consuming task, and in many cases, perhaps overly time-consuming, and given the considerations on the generalizability of the results, the results may not be of interest in other environments. Alternatively, not following guidelines introduces significant threats to validity. While generalizability is hardly an intrinsic value, concluding that, e.g., MySQL outperforms PostgreSQL in “my webstore” but not in others unless they have similar data, hardware, number of end-users, etc., does not carry the implication of being as scientifically impactful result as saying that, e.g., MySQL will always outperform PostgreSQL. Therefore, we must either perform the performance comparisons with rigor and accept that the results do not probably generalize, or perform the comparisons without scientific rigor and state sophisms. Since the latter is hardly ethically sound, DBMS performance comparisons should be limited to domains where the goal of a study is not the generalizability of the results, but the betterment of the very particular domain the study concerns (e.g., Ameri et al., 2014).

Given the arguments above, we propose that future studies, if inter-DBMS performance must be compared, consider taking a different approach to performance testing. First, using a wide range of database system optimization experts to ensure that all aspects of the system are fairly optimized, and avoiding situations where one system is optimized beyond diminishing returns, while the other is barely optimized. We challenge research teams to explicitly disclose which authors optimized which systems, for authors to further one’s intellectual investments in the performance comparison. These solutions should be benchmarked by a party independent of all optimization teams, and fair benchmarking guidelines should be utilized. Second, after the benchmarking has been carried out, we urge researchers to consider what causes the differences in performance, and critically compare those aspects as well, as gains in performance arguably have root causes such as loosened consistency or increased storage space. Nonetheless, performance comparisons of two or more DBMS with different data models should be considered particularly complex. Unfortunately, such comparisons seem to be the most popular (cf. Fig. 5).

6.3.3. Consider other use cases besides DBMS-DBMS testing altogether

It is worth noting that benchmarking software has other use cases besides inter-DBMS performance comparisons. Instead of comparing one DBMS to another, researchers might consider testing the performance effects of different hardware (Do et al., 2011), DBMS parameters (Wang et al., 2022), operating system parameters, query languages (Holzschuher and Peinl, 2013), physical configurations such as database distribution, physical structures such as different indices, or different levels of data consistency.

6.4. Limitations and threats to validity

It might be that some relevant studies are missing from this literature review. However, it was not our intention to select primary studies to quantitatively demonstrate that one DBMS outperforms another by the number of studies corroborating such an argument. Rather, the results verify previous observations (Raasveldt et al., 2018) according to which many of such comparisons are problematic and should be interpreted with care, if at all. Nevertheless, we have strived to include at least most of the primary studies that fit our criteria (Table 2) by several rounds of snowballing (Fig. 3) as well as a complementary literature search. Furthermore, as the DBMS classification (Table 3) and the interpretation of the primary study results (Appendix) involve human judgment, it is possible that another group of researchers may attain at least slightly different results.

7. Conclusion

Several database management system performance comparisons have been conducted and published as both vendor white-papers as well as in scientific fora. The approaches and reporting in such studies have been criticized in previous literature. In this study, we systematically surveyed 117 DBMS performance comparison studies. What seemed to be common among the selected primary studies is that they lack sufficient detail for reproducibility. Scientific, peer-reviewed research of high external validity concerning database management system performance comparison is effectively scarce. Based on the review of literature, we presented several considerations for the industry as well as database system researchers. Namely, we argued for considering *(i)* the *environments* (i.e., business domain, amount of data, amount of concurrent users, hardware, database distribution, read/write operation ratio, etc.) when interpreting the results of DBMS performance comparison tests, and for considering *(ii)* other aspects besides DBMS performance when choosing a DBMS or changing one DBMS to another, and for researchers to consider *(iii)* using existing guidelines in performance testing and reporting the testing environments transparently, to consider *(iv)* different approaches to performance testing when one DBMS is compared to another, and to consider *(v)* other use cases for performance testing besides comparing the performance of one DBMS to another. The results highlight how rarely benchmarking software is used in performance testing, how often different DBMSs with different data models are compared with each other, how often performance testing results in different studies conflict with each other, and why. This study was not an attempt to argue the performance gains of one DBMS over another using primary studies. That is, please do not cite this study by consulting the Appendix and stating that *$DBMS_1$ outperforms $DBMS_2$*.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jss.2023.111872.

References

Ailamaki, Anastassia, DeWitt, David J., Hill, Mark D., Wood, David A., 1999. DBMSs on a modern processor: Where does time go? In: VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK, No. CONF. pp. 266–277.

Ammons, Glenn, Choi, Jong-Deok, Gupta, Manish, Swamy, Nikhil, 2004. Finding and removing performance bottlenecks in large systems. In: Odersky, Martin (Ed.), ECOOP 2004 – Object-Oriented Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 172–196.

Barata, Melyssa, Bernardino, Jorge, Furtado, Pedro, 2015. An overview of decision support benchmarks: TPC-DS, TPC-H and SSB. In: Rocha, Alvaro, Correia, Ana Maria, Costanzo, Sandra, Reis, Luis Paulo (Eds.), New Contributions in Information Systems and Technologies. Springer International Publishing, Cham, pp. 619–628.

Berenson, Hal, Bernstein, Phil, Gray, Jim, Melton, Jim, O'Neil, Elizabeth, O'Neil, Patrick, 1995. A critique of ANSI SQL isolation levels. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. SIGMOD '95, Association for Computing Machinery, New York, NY, USA, pp. 1–10.

Bernstein, Philip A., Goodman, Nathan, 1981. Concurrency control in distributed database systems. ACM Comput. Surv. 13 (2), 185–221.

Brass, Stefan, Goldberg, Christian, 2006. Semantic errors in SQL queries: A quite complete list. J. Syst. Softw. 79 (5), 630–644, Quality Software.

Brewer, Eric, 2012. CAP twelve years later: How the "rules" have changed. Computer 45 (2), 23–29.

Cass, Stephen, 2022. SQL should be your second language. IEEE Spectr. 59 (10), 20–21.

Chaudhry, Natalia, Yousaf, Muhammad Murtaza, 2020. Architectural assessment of NoSQL and NewSQL systems. Distrib. Parallel Databases 38 (4), 881–926.

Chaudhuri, Surajit, 1998. An overview of query optimization in relational systems. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. PODS '98, Association for Computing Machinery, New York, NY, USA, pp. 34–43.

Chen, Peter Pin-Shan, 1976. The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. 1 (1), 9–36.

Christodoulakis, S., 1984. Implications of certain assumptions in database performance evauation. ACM Trans. Database Syst. 9 (2), 163–186.

Coates, Sean Steven, 2009. Comparing the Performance of Open Source and Proprietary Relational Database Management Systems (Ph.D. thesis). Northcentral University.

Codd, Edgar F., 1970. A relational model of data for large shared data banks. Commun. ACM 13 (6), 377–387.

Codd, Edgar F., 1972. Further normalization of the data base relational model. Data Base Syst. 6, 33–64.

Codd, Edgar F., 1975. Recent investigations in relational data base systems.

Connolly, Thomas, Begg, Carolyn, 2015. Database Systems, sixth ed. Pearson.

Cooper, Brian F., Silberstein, Adam, Tam, Erwin, Ramakrishnan, Raghu, Sears, Russell, 2010. Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing. SoCC '10, Association for Computing Machinery, New York, NY, USA, pp. 143–154.

Corbett, James C., Dean, Jeffrey, Epstein, Michael, Fikes, Andrew, Frost, Christopher, Furman, J.J., Ghemawat, Sanjay, Gubarev, Andrey, Heiser, Christopher, Hochschild, Peter, Hsieh, Wilson, Kanthak, Sebastian, Kogan, Eugene, Li, Hongyi, Lloyd, Alexander, Melnik, Sergey, Mwaura, David, Nagle, David, Quinlan, Sean, Rao, Rajesh, Rolig, Lindsay, Saito, Yasushi, Szymaniak, Michal, Taylor, Christopher, Wang, Ruth, Woodford, Dale, 2013. Spanner: Google's globally distributed database. ACM Trans. Comput. Syst. 31 (3), 1–22.

Cortellessa, Vittorio, Di Marco, Antinisca, Inverardi, Paola, 2011. Model-Based Software Performance Analysis, Vol. 980. Springer.

Date, Chris J., 2019. Database Design and Relational Theory: Normal Forms and All that Jazz. A Press.

Davoudian, Ali, Chen, Liu, Liu, Mengchi, 2018. A survey on NoSQL stores. ACM Comput. Surv. 51 (2).

Delis, A., Roussopoulos, N., 1993. Performance comparison of three modern DBMS architectures. IEEE Trans. Softw. Eng. 19 (2), 120–138.

DeWitt, David J., Levine, Charles, 2008. Not just correct, but correct and fast: A look at one of Jim Gray's contributions to database system performance. SIGMOD Rec. 37 (2), 45–49.

Dey, Akon, Fekete, Alan, Nambiar, Raghunath, Röhm, Uwe, 2014. YCSB+T: Benchmarking web-scale transactional databases. In: 2014 IEEE 30th International Conference on Data Engineering Workshops. pp. 223–230.

Dietrich, Suzanne W., Brown, M., Cortes-Rello, Enrique, Wunderlin, S., 1992. A practitioner's introduction to database performance benchmarks and measurements. Comput. J. 35 (4), 322–331.

Difallah, Djellel Eddine, Pavlo, Andrew, Curino, Carlo, Cudre-Mauroux, Philippe, 2013. OLTP-Bench: An extensible testbed for benchmarking relational databases. Proc. VLDB Endow. 7 (4), 277–288.

Do, Thanh, Graefe, Goetz, Naughton, Jeffrey, 2022. Efficient sorting, duplicate removal, grouping, and aggregation. ACM Trans. Database Syst..

Do, Jaeyoung, Zhang, Donghui, Patel, Jignesh M., DeWitt, David J., Naughton, Jeffrey F., Halverson, Alan, 2011. Turbocharging DBMS buffer pool using SSDs. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. SIGMOD '11, Association for Computing Machinery, New York, NY, USA, pp. 1113–1124.

Dreseler, Markus, Boissier, Martin, Rabl, Tilmann, Uflacker, Matthias, 2020. Quantifying TPC-H choke points and their optimizations. Proc. VLDB Endow. 13 (8), 1206–1220.

Elmasri, Ramez, Navathe, Shamkant B., 2016. Fundamentals of Database Systems, seventh ed. Pearson.

Elnikety, Sameh, Dropsho, Steven, Pedone, Fernando, 2006. Tashkent: Uniting durability with transaction ordering for high-performance scalable database replication. SIGOPS Oper. Syst. Rev. 40 (4), 117–130.

Estivill-Castro, Vladmir, Wood, Derick, 1992. A survey of adaptive sorting algorithms. ACM Comput. Surv. 24 (4), 441–476.

Forresi, Chiara, Francia, Matteo, Gallinucci, Enrico, Golfarelli, Matteo, 2022. Cost-based optimization of multistore query plans. Inf. Syst. Front. 1–27.

Gilbert, Seth, Lynch, Nancy, 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33 (2), 51–59.

Graefe, Goetz, 1993. Query evaluation techniques for large databases. ACM Comput. Surv. 25 (2), 73–169.

Gray, Jim, 1992. Benchmark Handbook: For Database and Transaction Processing Systems. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Grolinger, Katarina, Higashino, Wilson A., Tiwari, Abhinav, Capretz, Miriam A.M., 2013. Data management in cloud environments: Nosql and NewSQL data stores. J. Cloud Comput.: Adv., Syst. Appl. 2 (1), 22.

Gunther, Neil J., 2011. Analyzing Computer System Performance with Perl::PDQ. Springer.

Guo, Yuanbo, Pan, Zhengxiang, Heflin, Jeff, 2005. LUBM: A benchmark for OWL knowledge base systems. J. Web Semant. 3 (2), 158–182.

Guo, Binglei, Yu, Jiong, Yang, Dexian, Leng, Hongyong, Liao, Bin, 2022. Energy-efficient database systems: A systematic survey. ACM Comput. Surv..

Haerder, Theo, Reuter, Andreas, 1983. Principles of transaction-oriented database recovery. ACM Comput. Surv. 15 (4), 287–317.

Hecht, Robin, Jablonski, Stefan, 2011. NoSQL evaluation: A use case oriented survey. In: 2011 International Conference on Cloud and Service Computing. pp. 336–341.

Hellerstein, Joseph M., Stonebraker, Michael, Hamilton, James, 2007. Architecture of a database system. Found. Trends Databases 1 (2), 141–259.

Holzschuher, Florian, Peinl, René, 2013. Performance of graph query languages: Comparison of Cypher, Gremlin and native access in Neo4j. In: EDBT '13. Association for Computing Machinery, New York, NY, USA, pp. 195–204.

ISO/IEC, 2016a. ISO/IEC 9075-1:2016 - SQL - Part 1: Framework. Technical Report, ISO/IEC.

ISO/IEC, 2016b. ISO/IEC 9075-2:2016 - SQL - Part 2: Foundation. Technical Report, ISO/IEC.

Jiang, Dawei, Ooi, Beng Chin, Shi, Lei, Wu, Sai, 2010. The performance of MapReduce: An in-depth study. Proc. VLDB Endow. 3 (1–2), 472–483.

Jin, Guoliang, Song, Linhai, Shi, Xiaoming, Scherpelz, Joel, Lu, Shan, 2012. Understanding and detecting real-world performance bugs. In: Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation. PLDI '12, Association for Computing Machinery, New York, NY, USA, pp. 77–88.

Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, 2013. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. Technical Report, ACM, New York, NY, USA, 999133.

Juran, Joseph M., De Feo, Joseph A., 2010. Juran's Quality Handbook: The Complete Guide To Performance Excellence, sixth ed. McGraw-Hill Education.

Kim, Ho-Jun, Ko, Eun-Jeong, Jeon, Young-Ho, Lee, Ki-Hoon, 2018. Migration from RDBMS to column-oriented NoSQL: Lessons learned and open problems. In: Lee, Wookey, Choi, Wonik, Jung, Sungwon, Song, Min (Eds.), Proceedings of the 7th International Conference on Emerging Databases. Springer Singapore, Singapore, pp. 25–33.

Kim, You Jung, Patel, Jignesh, 2010. Performance comparison of the R*-Tree and the quadtree for kNN and distance join queries. IEEE Trans. Knowl. Data Eng. 22 (7), 1014–1027.

Kumar, Rakesh, Grot, Boris, 2022. Shooting down the server front-end bottleneck. ACM Trans. Comput. Syst. 38 (3–4).

Leis, Viktor, Gubichev, Andrey, Mirchev, Atanas, Boncz, Peter, Kemper, Alfons, Neumann, Thomas, 2015. How good are query optimizers, really? Proc. VLDB Endow. 9 (3), 204–215.

Lightstone, Sam S., Teorey, Toby J., Nadeau, Tom, 2010. Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and more. Morgan Kaufmann.

Lu, Jiaheng, Holubová, Irena, 2019. Multi-model databases: A new journey to handle the variety of data. ACM Comput. Surv. 52 (3).

Marek, Robert, Rahm, Erhard, 1992. Performance evaluation of parallel transaction processing in shared nothing database systems. In: Etiemble, Daniel, Syre, Jean-Claude (Eds.), PARLE '92 Parallel Architectures and Languages Europe. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 295–310.

Obradovic, Nikola, Kelec, Aleksandar, Dujlovic, Igor, 2019. Performance analysis on Android SQLite database. In: 2019 18th International Symposium INFOTEH-JAHORINA. INFOTEH, pp. 1–4.

Osterhage, W., 2013. Computer Performance Optimization. Springer.

Patel, Jignesh M., DeWitt, David J., 1996. Partition based spatial-merge join. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. SIGMOD '96, Association for Computing Machinery, New York, NY, USA, pp. 259–270.

Patounas, Georgios, Foukas, Xenofon, Elmokashfi, Ahmed, Marina, Mahesh K., 2020. Characterization and identification of cloudified mobile network performance bottlenecks. IEEE Trans. Netw. Serv. Manag. 17 (4), 2567–2583.

Pavlo, Andrew, 2017. What are we doing with our lives? Nobody cares about our concurrency control research. In: Proceedings of the 2017 ACM International Conference on Management of Data. SIGMOD '17, Association for Computing Machinery, New York, NY, USA, p. 3.

Pavlo, Andrew, Aslett, Matthew, 2016. What's really new with NewSQL? SIGMOD Rec. 45 (2), 45–55.

Purbo, Onno W., Sriyanto, Sriyanto, Suhendro, Suhendro, Aziz, Rz Abd, Herwanto, Riko, 2020. Benchmark and comparison between hyperledger and MySQL. TELKOMNIKA (Telecommun. Comput. Electron. Control) 18 (2), 705–715.

Purohith, Dhathri, Mohan, Jayashree, Chidambaram, Vijay, 2017. The dangers and complexities of SQLite benchmarking. In: Proceedings of the 8th Asia-Pacific Workshop on Systems. APSys '17, Association for Computing Machinery, New York, NY, USA.

Qu, Luyi, Li, Yuming, Zhang, Rong, Chen, Ting, Shu, Ke, Qian, Weining, Zhou, Aoying, 2022a. Application-oriented workload generation for transactional database performance evaluation. In: 2022 IEEE 38th International Conference on Data Engineering. ICDE, pp. 420–432.

Qu, Luyi, Wang, Qingshuai, Chen, Ting, Li, Keqiang, Zhang, Rong, Zhou, Xuan, Xu, Quanqing, Yang, Zhifeng, Yang, Chuanhui, Qian, Weining, Zhou, Aoying, 2022b. Are current benchmarks adequate to evaluate distributed transactional databases? BenchCouncil Trans. Benchmarks, Stand Eval. 2 (1), 100031.

Raasveldt, Mark, Holanda, Pedro, Gubner, Tim, Mühleisen, Hannes, 2018. Fair benchmarking considered difficult: Common pitfalls in database performance testing. In: Proceedings of the Workshop on Testing Database Systems. DBTest '18, Association for Computing Machinery, New York, NY, USA.

Ramakrishnan, Raghu, 2012. CAP and cloud data management. Computer 45 (2), 43–49.

Reniers, Vincent, Van Landuyt, Dimitri, Rafique, Ansar, Joosen, Wouter, 2017. On the state of NoSQL benchmarks. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. In: ICPE '17 Companion, Association for Computing Machinery, New York, NY, USA, pp. 107–112.

Schneider, Donovan A., DeWitt, David J., 1989. A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. In: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data. SIGMOD '89, Association for Computing Machinery, New York, NY, USA, pp. 110–121.

Stonebraker, Michael, 2010. SQL databases v. NoSQL databases. Commun. ACM 53 (4), 10–11.

Stonebraker, Michael, Bear, Chuck, Çetintemel, Uğur, Cherniack, Mitch, Ge, Tingjian, Hachem, Nabil, Harizopoulos, Stavros, Lifter, John, Rogers, Jennie, Zdonik, Stan, 2007. One size fits all? Part 2: Benchmarking results. In: Proc. CIDR.

Sundaresan, Srikanth, Magharei, Nazanin, Feamster, Nick, Teixeira, Renata, Crawford, Sam, 2013. Web performance bottlenecks in broadband access networks. SIGMETRICS Perform. Eval. Rev. 41 (1), 383–384.

Tallent, Nathan R., Mellor-Crummey, John M., 2009. Identifying performance bottlenecks in work-stealing computations. Computer 42 (12), 44–50.

Thalheim, Bernhard, Wang, Qing, 2013. Data migration: A theoretical perspective. Data Knowl. Eng. 87, 260–278.

The Joint Task Force on Computing Curricula, 2015. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical Report, ACM, New York, NY, USA.

Toffola, Luca Della, Pradel, Michael, Gross, Thomas R., 2018. Synthesizing programs that expose performance bottlenecks. In: Proceedings of the 2018 International Symposium on Code Generation and Optimization. In: CGO 2018, Association for Computing Machinery, New York, NY, USA, pp. 314–326.

Tözün, Pınar, Pandis, Ippokratis, Kaynak, Cansu, Jevdjic, Djordje, Ailamaki, Anastasia, 2013. From A to E: Analyzing TPC's OLTP benchmarks: The obsolete, the ubiquitous, the unexplored. In: EDBT '13. Association for Computing Machinery, New York, NY, USA, pp. 17–28.

Tu, Stephen, Zheng, Wenting, Kohler, Eddie, Liskov, Barbara, Madden, Samuel, 2013. Speedy transactions in multicore in-memory databases. In: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. SOSP '13, Association for Computing Machinery, New York, NY, USA, pp. 18–32.

Valduriez, Patrick, 1987. Join indices. ACM Trans. Database Syst. 12 (2), 218–246.

Wang, Yang, Yu, Miao, Hui, Yujie, Zhou, Fang, Huang, Yuyang, Zhu, Rui, Ren, Xueyuan, Li, Tianxi, Lu, Xiaoyi, 2022. A study of database performance sensitivity to experiment settings.. Proc. VLDB Endow. 15 (7).

Warszawski, Todd, Bailis, Peter, 2017. ACIDRain: Concurrency-related attacks on database-backed web applications. In: Proceedings of the 2017 ACM International Conference on Management of Data. SIGMOD '17, Association for Computing Machinery, New York, NY, USA, pp. 5–20.

Winand, Markus, 2012. SQL Performance Explained. Markus Winand.

Yang, Jinfeng, Lilja, David J., 2018. Reducing relational database performance bottlenecks using 3D XPoint storage technology. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering. TrustCom/BigDataSE, pp. 1804–1808.

Yu, Tingting, Pradel, Michael, 2018. Pinpointing and repairing performance bottlenecks in concurrent programs. Empir. Softw. Eng. 23 (5), 3034–3071.

Primary studies

Abramova, Veronika, Bernardino, Jorge, 2013. NoSQL databases: Mongodb vs Cassandra. In: Proceedings of the International C* Conference on Computer Science and Software Engineering. C3S2E '13, ACM Press, Porto, Portugal, pp. 14–22.

Abramova, Veronika, Bernardino, Jorge, Furtado, Pedro, 2014a. Experimental evaluation of NoSQL databases. Int. J. Database Manag. Syst. 6 (3), 01–16.

Abramova, Veronika, Bernardino, Jorge, Furtado, Pedro, 2014b. Which NoSQL database? A performance overview. Open J. Databases (OJDB) 1 (2), 17–24.

Abubakar, Yusuf, Adeyi, Thankgod Sani, Auta, Ibrahim Gambo, 2014. Performance evaluation of NoSQL systems using YCSB in a resource austere environment. Perform. Eval. 7 (8), 23–27.

Almeida, Rafael, Furtado, Pedro, Bernardino, Jorge, 2015. Performance evaluation MySQL InnoDB and microsoft SQL server 2012 for decision support environments. In: Proceedings of the Eighth International Conference on Computer Science & Software Engineering. C3S2E'15, ACM Press.

Ameri, Parinaz, Grabowski, Udo, Meyer, Jorg, Streit, Achim, 2014. On the application and performance of MongoDB for climate satellite data. In: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, Beijing, China, pp. 652–659.

Araujo, Jose Maria A., de Moura, Alysson Cristiano E., da Silva, Silvia Laryssa B., Holanda, Maristela, Ribeiro, Edward de Oliveira, da Silva, Gladston Luiz, 2021. Comparative performance analysis of NoSQL Cassandra and MongoDB databases. In: 2021 16th Iberian Conference on Information Systems and Technologies. CISTI, IEEE, Chaves, Portugal, pp. 1–6.

Bartoszewski, Dominik, Piorkowski, Adam, Lupa, Michal, 2019. The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases. In: Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis. Springer International Publishing, pp. 291–302.

Cheng, Yinyi, Zhou, Kefa, Wang, Jinlin, 2019. Performance analysis of PostgreSQL and MongoDB databases for unstructured data. In: Proceedings of the 2019 International Conference on Mathematics, Big Data Analysis and Simulation and Modeling. MBDASM 2019, Atlantis Press, Changsha, China.

Faraj, Azhi, Rashid, Bilal, Shareef, Twana, 2014. Comparative study of relational and non-relations database performances using Oracle and MongoDB systems. Int. J. Comput. Eng. Technol. (IJCET) 5 (11), 11–22.

Fotache, Marin, Hrubaru, Ionuţ, 2016. Performance analysis of two big data technologies on a cloud distributed architecture. Results for non-aggregate queries on medium-sized data. Sci. Ann. Econ. Bus. 63 (s1), 21–50.

Franke, Craig, Morin, Samuel, Chebotko, Artem, Abraham, John, Brazier, Pearl, 2013. Efficient processing of semantic web queries in HBase and MySQL cluster. IT Prof. 15 (3), 36–43.

Gandini, Andrea, Gribaudo, Marco, Knottenbelt, William J., Osman, Rasha, Piazzolla, Pietro, 2014. Performance evaluation of NoSQL databases. In: Computer Performance Engineering. Springer International Publishing, pp. 16–29.

Hendawi, Abdeltawab, Gupta, Jayant, Jiayi, Liu, Teredesai, Ankur, Naveen, Ramakrishnan, Mohak, Shah, Ali, Mohamed, 2018. Distributed NoSQL data stores: Performance analysis and a case study. In: 2018 IEEE International Conference on Big Data. Big Data, IEEE, Seattle, WA, USA, pp. 1937–1944.

Jing, Yinan, Zhang, Chunwang, Wang, Xueping, 2009. An empirical study on performance comparison of Lucene and relational database. In: 2009 International Conference on Communication Software and Networks. IEEE.

Kashyap, Suman, Zamwar, Shruti, Bhavsar, Tanvi, Singh, Snigdha, 2013. Benchmarking and analysis of NoSQL technologies. Int. J. Emerg. Technol. Adv. Eng. 3 (9), 422–426.

Klein, John, Gorton, Ian, Ernst, Neil, Donohoe, Patrick, Pham, Kim, Matser, Chrisjan, 2015. Performance evaluation of NoSQL databases: A case study. In: Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems. ACM, Austin Texas USA, pp. 5–10.

Kulshrestha, Sudhanshu, Sachdeva, Shelly, 2014. Performance comparison for data storage - Db4o and MySQL databases. In: 2014 Seventh International Conference on Contemporary Computing (IC3). IEEE, Noida, India, pp. 166–170.

KumarDwivedi, Amit, Lamba, C.S., Shukla, Shweta, 2012. Performance analysis of column oriented database vs row oriented database. Int. J. Comput. Appl. 50 (14), 31–34.
Nepaliya, Prateek, Gupta, Prateek, 2015. Performance analysis of NoSQL databases. Int. J. Comput. Appl. 127 (12), 36–39.
Oliveira, João, Bernardino, Jorge, 2017. NewSQL databases - MemSQL and VoltDB experimental evaluation:. In: Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, pp. 276–281.
Padhy, Sarita, Kumaran, G. Mayil Muthu, 2019. A quantitative performance analysis between Mongodb and Oracle NoSQL. In: 2019 6th International Conference on Computing for Sustainable Global Development. INDIACom, IEEE, pp. 387–391.
Rabl, Tilmann, Gómez-Villamor, Sergio, Sadoghi, Mohammad, Muntés-Mulero, Victor, Jacobsen, Hans-Arno, Mankovskii, Serge, 2012. Solving big data challenges for enterprise application performance management. Proc. VLDB Endow. 5 (12), 1724–1735.
Schmid, Stephan, Galicz, Eszter, Reinhardt, Wolfgang, 2015b. WMS performance of selected SQL and NoSQL databases. In: International Conference on Military Technologies. ICMT 2015, IEEE.
Schreiner, Geomar A., Knob, Ronan, Duarte, Denio, Vilain, Patricia, Mello, Ronaldo dos Santos, 2019. NewSQL through the looking glass. In: Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services. ACM, Munich Germany, pp. 361–369.
Seghier, Nadia Ben, Kazar, Okba, 2021. Performance benchmarking and comparison of NoSQL databases: Redis vs MongoDB vs Cassandra using YCSB tool. In: 2021 International Conference on Recent Advances in Mathematics and Informatics. ICRAMI, IEEE, Tebessa, Algeria, pp. 1–6.
Suh, Young-Kyoon, An, Junyoung, Tak, Byungchul, Na, Gap-Joo, 2022. A comprehensive empirical study of query performance across GPU DBMSes. Proc. ACM Meas. Anal. Comput. Syst. 6 (1), 1–29.
Swaminathan, Surya Narayanan, Elmasri, Ramez, 2016. Quantitative analysis of scalable NoSQL databases. In: 2016 IEEE International Congress on Big Data. BigData Congress, IEEE, San Francisco, CA, USA, pp. 323–326.
Tang, Enqing, Fan, Yushun, 2016. Performance comparison between five NoSQL databases. In: 2016 7th International Conference on Cloud Computing and Big Data. CCBD, IEEE, Macau, China, pp. 105–109.
Tongkaw, Sasalak, Tongkaw, Aumnat, 2016. A comparison of database performance of MariaDB and MySQL with OLTP workload. In: 2016 IEEE Conference on Open Systems. ICOS, IEEE.
Vershinin, I.S., Mustafina, A.R., 2021. Performance analysis of PostgreSQL, MySQL, Microsoft SQL server systems based on TPC-h Tests. In: 2021 International Russian Automation Conference. RusAutoCon, IEEE, Sochi, Russian Federation, pp. 683–687.
Yassien, Amal W., Desouky, Amr F., 2016. RDBMS, NoSQL, Hadoop: A performance-based empirical analysis. In: Proceedings of the 2nd Africa and Middle East Conference on Software Engineering - AMECSE '16. ACM Press, Cairo, Egypt, pp. 52–59.

Further reading

Abdullah, Ahmad, Zhuge, Qingfeng, 2015. From relational databases to NoSQL databases: Performance evaluation. Res. J. Appl. Sci. Eng. Technol. 11 (4), 434–439.
Aboutorabi, Seyyed Hamid, Rezapour, Mehdi, Moradi, Milad, Ghadiri, Nasser, 2015. Performance evaluation of SQL and MongoDB databases for big e-commerce data. In: 2015 International Symposium on Computer Science and Software Engineering. CSSE, IEEE, Tabriz, Iran, pp. 1–7.
Afolabi, A.O., Ajayi, A.O., 2008. Performance evaluation of a database management system (A case study of INTERBASE and MySQL). J. Eng. Appl. Sci. 3 (2), 155–160.
Agarwal, Sarthak, Rajan, K.S., 2016. Performance analysis of MongoDB versus PostGIS/PostGreSQL databases for line intersection and point containment spatial queries. Spatial Inf. Res. 24 (6), 671–677.
Aghi, Rajat, Mehta, Sumeet, Chauhan, Rahul, Chaudhary, Siddhant, Bohra, Navdeep, 2015. A comprehensive comparison of SQL and MongoDB databases. Int. J. Sci. Res. Publ. 5 (2), 1–3.
Ahmed, Nadeem, Ahamed, Shakil, Rafiq, Jahir Ibna, Rahim, Sifatur, 2017. Data processing in Hive vs. SQL Server: A comparative analysis in the query performance. In: 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences. ICETSS, IEEE, Bangkok, pp. 1–5.
Almabdy, Soad, 2018. Comparative analysis of relational and graph databases for social networks. In: 2018 1st International Conference on Computer Applications & Information Security. ICCAIS, IEEE, pp. 1–4.
Andjelic, Svetlana, Obradovic, Slobodan, Gacesa, Branislav, 2008. A performance analysis of the DBMS - MySQL Vs PostgreSQL. Commun. - Sci. Lett. Univ. Zilina 10 (4), 53–57.
Baralis, Elena, Dalla Valle, Andrea, Garza, Paolo, Rossi, Claudio, Scullino, Francesco, 2017. SQL versus NoSQL databases for geospatial applications. In: 2017 IEEE International Conference on Big Data. Big Data, IEEE, Boston, MA, pp. 3388–3397.
Baruffa, Giuseppe, Femminella, Mauro, Pergolesi, Matteo, Reali, Gianluca, 2020. Comparison of MongoDB and Cassandra databases for spectrum monitoring as-a-service. IEEE Trans. Netw. Serv. Manag. 17 (1), 346–360.
Bassil, Youssef, 2012. A comparative study on the performance of the top DBMS systems. J. Comput. Sci. Res..
Batra, Shalini, Tyagi, Charu, 2012. Comparative analysis of relational and graph databases. Int. J. Soft Comput. Eng. (IJSCE) 2 (2), 509–512.
Boicea, Alexandru, Radulescu, Florin, Agapin, Laura Ioana, 2012. MongoDB vs Oracle – database comparison. In: 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. IEEE.
Čerešňák, Roman, Kvet, Michal, 2019. Comparison of query performance in relational a non-relation databases. Transp. Res. Procedia 40, 170–177.
Chakraborty, Soarov, Paul, Shourav, Azharul Hasan, K.M., 2021. Performance comparison for data retrieval from NoSQL and SQL databases: A case study for COVID-19 genome sequence dataset. In: 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques. ICREST, IEEE, DHAKA, Bangladesh, pp. 324–328.
Chaudhary, Anurag Singh, Singh, Kanika, Kalra, Sanchi, Kaur, Parmeet, 2018. An empirical comparison of MongoDB and hive. In: 2018 4th International Conference on Computing Communication and Automation. ICCCA, IEEE, Greater Noida, India, pp. 1–4.
Chickerur, Satyadhyan, Goudar, Anoop, Kinnerkar, Ankita, 2015. Comparison of relational database with document-oriented database (MongoDB) for big data applications. In: 2015 8th International Conference on Advanced Software Engineering & Its Applications. ASEA, IEEE, Jeju Island, South Korea, pp. 41–47.
Chopade, Mrs. Rupali M., Dhavase, Nikhil S., 2017. MongoDB, Couchbase: Performance comparison for image dataset. In: 2017 2nd International Conference for Convergence in Technology. I2CT, IEEE, Mumbai, pp. 255–258.
Damodaran B, Dipina, Salim, Shirin, Vargese, Surekha Marium, 2016. Performance evaluation of MySQL and MongoDB databases. Int. J. Cybern. Inf. 5 (2), 387–394.
Deari, Raif, Zenuni, Xhemal, Ajdari, Jaumin, Ismaili, Florije, Raufi, Bujar, 2018. Analysis and comparision of document-based databases with relational databases: Mongodb vs MySQL. In: 2018 International Conference on Information Technologies. InfoTech, IEEE, Varna, pp. 1–4.
Ding, Haijie, Jin, Yuehui, Cui, Yidong, Yang, Tan, 2012. Distributed storage of network measurement data on HBase. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems. IEEE.
Eyada, Mahmoud Moustafa, Saber, Walaa, Genidy, Mohammed M. El, Amer, Fathy, 2020. Performance evaluation of IoT data management using MongoDB versus MySQL databases in different cloud environments. IEEE Access 8, 110656–110668.
Fatima, Haleemunnisa, Wasnik, Kumud, 2016. Comparison of SQL, NoSQL and NewSQL databases for internet of things. In: 2016 IEEE Bombay Section Symposium. IBSS, IEEE.
Filip, Petr, Cegan, Lukas, 2020. Comparison of MySQL and MongoDB with focus on performance. In: 2020 International Conference on Informatics, Multimedia, Cyber and Information System. ICIMCIS, IEEE.
Fioravanti, Sara, Mattolini, Simone, Patara, Fulvio, Vicario, Enrico, 2016. Experimental performance evaluation of different data models for a reflection software architecture over NoSQL persistence layers. In: Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, Delft The Netherlands, pp. 297–308.
Fraczek, Konrad, Plechawska-Wojcik, Malgorzata, 2017. Comparative analysis of relational and non-relational databases in the context of performance in web applications. In: Beyond Databases, Architectures and Structures. Toward Efficient Solutions for Data Analysis and Knowledge Representation. Springer International Publishing, pp. 153–164.
Gomes, Augusto, Lopes, Vitor, Ribeiro, Edward, Lima, Jorge, Costa, Wagner, Garcia, Luis, Holanda, Maristela, 2021. An empirical performance comparison between MySQL and MongoDB on analytical queries in the COMEX database. In: 2021 16th Iberian Conference on Information Systems and Technologies. CISTI, IEEE, Chaves, Portugal, pp. 1–5.
Gunawan, Rohmat, Rahmatulloh, Alam, Darmawan, Irfan, 2019. Performance evaluation of query response time in the document stored NoSQL database. In: 2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering. IEEE, Padang, Indonesia, pp. 1–6.
Gyorodi, Cornelia A., Dumse-Burescu, Diana V., Zmaranda, Doina R., Gyorodi, Robert s., Gabor, Gianina A., Pecherle, George D., 2020. Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application'"s data storage. Appl. Sci. 10 (23), 8524.
Gyorodi, Cornelia, Gyorodi, Robert, Pecherle, George, Olah, Andrada, 2015. A comparative study: Mongodb vs. MySQL. In: 2015 13th International Conference on Engineering of Modern Electric Systems. EMES, IEEE, Oradea, Romania, pp. 1–6.
Hairah, U., Budiman, E., 2021. Inner join query performance: MariaDB vs PostgreSQL. J. Phys. Conf. Ser. 1844 (1), 012021.
Haiyan, Yu, Jingsong, Li, Huan, Chen, Xiaoguang, Zhang, Yu, Tian, Yibing, Yang, 2010. Performance evaluation of post-relational database in hospital information systems. In: 2010 Second International Workshop on Education Technology and Computer Science. IEEE.

Hajjaji, Yosra, Farah, Imed Riadh, 2018. Performance investigation of selected NoSQL databases for massive remote sensing image data storage. In: 2018 4th International Conference on Advanced Technologies for Signal and Image Processing. ATSIP, IEEE.
Hassan, Mahmudul, Bansal, Srividya K., 2018. Semantic data querying over NoSQL databases with Apache Spark. In: 2018 IEEE International Conference on Information Reuse and Integration. IRI, IEEE, Salt Lake City, UT, pp. 364–371.
Ilić, Miloš, Kopanja, Lazar, Zlatković, Dragan, Trajković, Milica, Ćurguz, Dejana, 2021. Microsoft SQL Server and Oracle: Comparative performance analysis. In: Book of Proceedings of the 7th International Conference Knowledge Management.
Jaiswal, Garima, 2013. Comparative analysis of relational and graph databases. IOSR J. Eng. 03 (08), 25–27.
Jandaeng, Chanankorn, 2015. Comparison of RDBMS and document oriented database in audit log analysis. In: 2015 7th International Conference on Information Technology and Electrical Engineering. ICITEE, IEEE, Chiang Mai, Thailand, pp. 332–336.
Jose, Benymol, Abraham, Sajimon, 2020. Performance analysis of NoSQL and relational databases with MongoDB and MySQL. Mater. Today: Proc. 24, 2036–2043.
Jung, Min-Gyue, Youn, Seon-A., Bae, Jayon, Choi, Yong-Lak, 2015. A study on data input and output performance comparison of MongoDB and PostgreSQL in the big data environment. In: 2015 8th International Conference on Database Theory and Application. DTA, IEEE, Jeju Island, South Korea, pp. 14–17.
Kabakus, Abdullah Talha, Kara, Resul, 2017. A performance evaluation of in-memory databases. J. King Saud Univ. - Comput. Inf. Sci. 29 (4), 520–525.
Kaur, Karambir, Sachdeva, Monika, 2017. Performance evaluation of NewSQL databases. In: 2017 International Conference on Inventive Systems and Control. ICISC, IEEE.
Khan, Wisal, Ahmad, Waqas, Luo, Bin, Ahmed, Ejaz, 2019. SQL Database with physical database tuning technique and NoSQL graph database comparisons. In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference. ITNEC, IEEE, Chengdu, China, pp. 110–116.
Khan, Wisal, ahmed, Ejaz, Shahzad, Waseem, 2017. Predictive performance comparison analysis of relational & NoSQL graph databases. Int. J. Adv. Comput. Sci. Appl. 8 (5).
Khanna, Deepti, Aggarwal, V.B., Director, J.I.M.S., Dave, India Meenu, 2018. Performance analysis for select, project and join operations of Oracle, My-SQL and microsoft access DBMSS. Int. J. Comput. Eng. Technol. (IJCET).
Kumar, Lokesh, Rajawat, Shalini, Joshi, Krati, 2015. Comparative analysis of NoSQL (MongoDB) with MySQL database. Int. J. Modern Trends Eng. Res. 2 (5), 120–127.
Kumar, K.B. Sundhara, Srividya, Mohanavalli, S., 2017. A performance comparison of document oriented NoSQL databases. In: 2017 International Conference on Computer, Communication and Signal Processing. ICCCSP, IEEE, Chennai, India, pp. 1–6.
Laksono, Dany, 2018. Testing spatial data deliverance in SQL and NoSQL database using nodejs fullstack web app. In: 2018 4th International Conference on Science and Technology. ICST, IEEE, Yogyakarta, pp. 1–5.
Lazarska, Malgorzata, Siedlecka-Lamch, Olga, 2019. Comparative study of relational and graph databases. In: 2019 IEEE 15th International Scientific Conference on Informatics. IEEE, pp. 000363–000370.
Lee, Chao-Hsien, Shih, Zhe-Wei, 2018. A comparison of NoSQL and SQL databases over the hadoop and spark cloud platforms using machine learning algorithms. In: 2018 IEEE International Conference on Consumer Electronics-Taiwan. ICCE-TW, IEEE, Taichung, pp. 1–2.
Li, Yishan, Manoharan, Sathiamoorthy, 2013. A performance comparison of SQL and NoSQL databases. In: 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. PACRIM, IEEE.
Lorincz, Josip, Huljic, Vlatka, Begusic, Dinko, 2020. Transforming product catalog relational into graph database: A performance comparison. In: 2020 43rd International Convention on Information, Communication and Electronic Technology. MIPRO, IEEE, Opatija, Croatia, pp. 523–528.
Magdum, Junaid, Barhate, Rahul, 2018. Performance analysis of DML operations on NoSQL databases for streaming data. In: 2018 Fourth International Conference on Computing Communication Control and Automation. ICCUBEA, IEEE, Pune, India, pp. 1–6.
Mahmood, Khalid, Orsborn, Kjell, Risch, Tore, 2019. Comparison of NoSQL datastores for large scale data stream log analytics. In: 2019 IEEE International Conference on Smart Computing. SMARTCOMP, IEEE, Washington, DC, USA, pp. 478–480.
Makris, Antonios, Tserpes, Konstantinos, Spiliopoulos, Giannis, Anagnostopoulos, Dimosthenis, 2019. Performance evaluation of MongoDB and PostgreSQL for spatio-temporal data. In: EDBT/ICDT Workshops.
Makris, Antonios, Tserpes, Konstantinos, Spiliopoulos, Giannis, Zissis, Dimitrios, Anagnostopoulos, Dimosthenis, 2021. MongoDB Vs PostgreSQL: A comparative study on performance aspects. GeoInformatica 25 (2), 243–268.
Marrero, Luciano, Olsowy, Verena, Tesone, Fernando, Thomas, Pablo, Delia, Lisandro, Pesado, Patricia, 2020. Performance analysis in NoSQL databases, relational databases and NoSQL databases as a service in the cloud. In: Argentine Congress of Computer Science. Springer, pp. 157–170.
Mavrogiorgos, Konstanitnos, Kiourtis, Athanasios, Mavrogiorgou, Argyro, Kyriazis, Dimosthenis, 2021. A comparative study of MongoDB, ArangoDB and CouchDB for big data storage. In: 2021 5th International Conference on Cloud and Big Data Computing. ICCBDC, ACM, Liverpool United Kingdom, pp. 8–14.
Murazza, Muh. Rafif, Nurwidyantoro, Arif, 2016. Cassandra and SQL database comparison for near real-time Twitter data warehouse. In: 2016 International Seminar on Intelligent Technology and Its Applications. ISITIA, IEEE, Lombok, Indonesia, pp. 195–200.
Nyati, Suyog S., Pawar, Shivanand, Ingle, Rajesh, 2013. Performance evaluation of unstructured NoSQL data over distributed framework. In: 2013 International Conference on Advances in Computing, Communications and Informatics. ICACCI, IEEE, Mysore, pp. 1623–1627.
Ohyver, Margaretha, Moniaga, Jurike V., Sungkawa, Iwa, Subagyo, Bonifasius Edwin, Chandra, Ian Argus, 2019. The comparison firebase realtime database and MySQL database performance using wilcoxon signed-rank test. Procedia Comput. Sci. 157, 396–405.
Parker, Zachary, Poe, Scott, Vrbsky, Susan V., 2013. Comparing NoSQL MongoDB to an SQL DB. In: Proceedings of the 51st ACM Southeast Conference on - ACMSE '13. ACM Press, Savannah, Georgia, p. 1.
Patil, Mayur M., Hanni, Akkamahadevi, Tejeshwar, C.H., Patil, Priyadarshini, 2017. A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retrieval operations using a web/android application to explore load balancing — sharding in MongoDB and its advantages. In: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud). I-SMAC, IEEE.
Pereira, Diogo Augusto, Ourique de Morais, Wagner, Pignaton de Freitas, Edison, 2018. NoSQL real-time database performance comparison. Int. J. Parallel Emergent Distrib. Syst. 33 (2), 144–156.
Poljak, R., Poscic, P., Jaksic, D., 2017. Comparative analysis of the selected relational database management systems. In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics. MIPRO, IEEE, Opatija, Croatia, pp. 1496–1500.
Puangsaijai, Wittawat, Puntheeranurak, Sutheera, 2017. A comparative study of relational database and key-value database for big data applications. In: 2017 International Electrical Engineering Congress. IEECON, IEEE, Pattaya, Thailand, pp. 1–4.
Rafamantanantsoa, Fontaine, Laha, Maherindefo, 2018. Analysis and neural networks modeling of web server performances using MySQL and PostgreSQL. Commun. Network 10 (04), 142–151.
Rautmare, Sharvari, Bhalerao, D.M., 2016. MySQL and NoSQL database comparison for IoT application. In: 2016 IEEE International Conference on Advances in Computer Applications. ICACA, IEEE, Coimbatore, pp. 235–238.
Ribeiro, Jardel, Henrique, Jonas, Ribeiro, Rodrigo, Neto, Rosalvo, 2017. NoSQL vs relational database: A comparative study about the generation of the most frequent N-grams. In: 2017 4th International Conference on Systems and Informatics. ICSAI, IEEE, Hangzhou, pp. 1568–1572.
Roopak, K.E., Rao, K.S. Swati, Ritesh, S., Chickerur, Satyadhyan, 2013. Performance comparison of relational database with object database (DB4o). In: 2013 5th International Conference on Computational Intelligence and Communication Networks. IEEE.
Saikia, Amlanjyoti, Joy, Sherin, Dolma, Dhondup, Mary R, Roseline, 2015. Comparative performance analysis of MySQL and SQL server relational database management systems in windows environment. IJARCCE 160–164.
Samanta, Ashis Kumar, Sarkar, Bidut Biman, Chaki, Nabendu, 2018. Query performance analysis of NoSQL and big data. In: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks. ICRCICN, IEEE.
Schmid, Stephan, Galicz, Eszter, Reinhardt, Wolfgang, 2015a. Performance investigation of selected SQL and NoSQL databases. In: Proceedings of the AGILE. pp. 1–5.
Seda, Pavel, Hosek, Jiri, Masek, Pavel, Pokorny, Jiri, 2018. Performance testing of NoSQL and RDBMS for storing big data in e-applications. In: 2018 3rd International Conference on Intelligent Green Building and Smart Grid. IGBSG, IEEE.
Sharma, Monika, Sharma, Vishal Deep, Bundele, Mahesh M., 2018. Performance analysis of RDBMS and No SQL databases: Postgresql, MongoDB and Neo4j. In: 2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering. ICRAIE, IEEE, Jaipur, India, pp. 1–5.
Sholichah, Rahmatian Jayanty, Imrona, Mahmud, Alamsyah, Andry, 2020. Performance analysis of Neo4j and MySQL databases using public policies decision making data. In: 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering. ICITACEE, IEEE, Semarang, Indonesia, pp. 152–157.
Sirish Shetty, B., Akshay, Kc, 2019. Performance analysis of queries in RDBMS vs NoSQL. In: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies. ICICICT, IEEE, Kannur,Kerala, India, pp. 1283–1286.
Stancu-Mara, Sorin, Baumann, Peter, 2008. A comparative benchmark of large objects in relational databases. In: Proceedings of the 2008 International Symposium on Database Engineering & Applications - IDEAS '08. ACM Press, Coimbra, Portugal, p. 277.
Truica, Ciprian-Octavian, Radulescu, Florin, Boicea, Alexandru, Bucur, Ion, 2015. Performance evaluation for CRUD operations in asynchronously replicated document oriented database. In: 2015 20th International Conference on Control Systems and Computer Science. IEEE, Bucharest, Romania, pp. 191–196.
van der Veen, Jan Sipke, van der Waaij, Bram, Meijer, Robert J., 2012. Sensor data storage performance: SQL or NoSQL, physical or virtual. In: 2012 IEEE Fifth International Conference on Cloud Computing. IEEE.

Vicknair, Chad, Macias, Michael, Zhao, Zhendong, Nan, Xiaofei, Chen, Yixin, Wilkins, Dawn, 2010. A comparison of a graph database and a relational database: A data provenance perspective. In: Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10. ACM Press, Oxford, Mississippi, p. 1.

Wei-ping, Zhu, Ming-xin, Li, Huan, Chen, 2011. Using MongoDB to implement textbook management system instead of MySQL. In: 2011 IEEE 3rd International Conference on Communication Software and Networks. IEEE.

Wiseso, Linggis Galih, Imrona, Mahmud, Alamsyah, Andry, 2020. Performance analysis of Neo4j, MongoDB, and PostgreSQL on 2019 national election big data management database. In: 2020 6th International Conference on Science in Information Technology. ICSITech, IEEE.

Xu, Wei, Zhou, Zhonghua, Zhou, Hong, Zhang, Wu, Xie, Jiang, 2014. MongoDB improves big data analysis performance on electric health record system. In: Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 350–357.

Yinfeng Wang, Guiquan Zhong, Lin Kun, Longxiang Wang, Huang Kai, Fuliang Guo, Chengzhe Liu, Xiaoshe Dong, 2015. The performance survey of in memory database. In: 2015 IEEE 21st International Conference on Parallel and Distributed Systems. ICPADS, IEEE, Melbourne, VIC, pp. 815–820.

Zhou, Zhonghai, Zhou, Bin, Li, Wenwen, Griglak, Brian, Caiseda, Carmen, Huang, Qunying, 2009. Evaluating query performance on object-relational spatial databases. In: 2009 2nd IEEE International Conference on Computer Science and Information Technology. IEEE.