future internet

MDPI

Article

NewSQL Databases Assessment: CockroachDB, MariaDB Xpand, and VoltDB

Eduardo Pina [1], Filipe Sá [1] and Jorge Bernardino [1,2,*]

[1] Polytechnic of Coimbra, Institute of Engineering of Coimbra—ISEC, Rua Pedro Nunes, 3030-199 Coimbra, Portugal
[2] Centre for Informatics and Systems of the University of Coimbra (CISUC), Polo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal
* Correspondence: jorge@isec.pt

Abstract: Background: Relational databases have been a prevalent technology for decades, using SQL (Structured Query Language) to manage data. However, the emergence of new technologies, such as the web and the cloud, has brought the requirement to handle more complex data. NewSQL is the latest technology that incorporates the ability to scale and ensures the availability of NoSQL (Not Only SQL) without losing the ACID properties (Atomicity, Consistency, Isolation, Durability) associated with relational databases. Methods: We evaluated CockroachDB, MariaDB Xpand, and VoltDB with OSSpal methodology and experimentally using the Star Schema Benchmark (SSB). The scalability and performance capabilities of each database were assessed. Results: Applying the OSSpal methodology, the results showed that MariaDB Xpand outperformed CockroachDB and VoltDB. On the other hand, we concluded that with Star Schema Benchmark, CockroachDB had better scalability, while VoltDB had a faster query execution time. Conclusions: CockroachDB and VoltDB are the best performing databases in terms of scalability and performance.

Keywords: SQL; NewSQL; OSSpal; SSB; CockroachDB; MariaDB Xpand; VoltDB

Citation: Pina, E.; Sá, F.; Bernardino, J. NewSQL Databases Assessment: CockroachDB, MariaDB Xpand, and VoltDB. *Future Internet* **2023**, *15*, 10. https://doi.org/10.3390/fi15010010

Academic Editor: Wolf-Tilo Balke

Received: 11 November 2022
Revised: 19 December 2022
Accepted: 20 December 2022
Published: 26 December 2022

1. Introduction

Today's business world is accelerating, where advances in web technology and the creation of Internet-connected sensors and mobile devices have resulted in huge datasets that need to be processed and stored like never before. Relational Database Management Systems, also known as RDBMS, are one of the most successful technologies in computing, being the default choice for model adoption in business worldwide and its SQL standard language [1–3]. A DBMS consists of a collection of programs and services that allow the user to insert, update, delete, and query the stored data, making it easier to maintain and access the data [4]. With the huge volume and complex evolution of data, businesses have started to look for alternatives that allow them to work with high volume, velocity, and variety. SQL's increased capacity allows the management of huge amounts of data but is not sufficient for the requirements of Big Data, which demands rapid response and high scalability. With such challenges, Not Only SQL (NoSQL) offers resources such as flexibility and horizontal scalability [5], but despite supporting the capability of highly available data volume technologies, NoSQL systems do not ensure ACID properties [6].

NewSQL emerged as a set of innovative SQL database engines with high performance and scalability. These engines seek to promote the same performance and scalability improvement of NoSQL systems and design solutions that have the advantages of the relational model, and the benefit of using SQL language and fulfilling the ACID properties [7].

In 2011, Matthew Aslett first used the term NewSQL in a business analysis report [8], which discussed the advent of new database systems. NewSQL systems come in different types, targeting different workloads and practices, opening up new opportunities in business, where real-time decisions are critical. Among several use cases, we have

streaming data, either from cameras or sensors, organizing a team and assigning roles and responsibilities, risk monitoring forecast, financial markets, Internet of Things, etc. There is no doubt that without NewSQL, these systems could only be developed using multiple systems at too high a price, and some of these applications would not perform as fast [8]. Moreover, NewSQL databases are designed to meet distributed architectures and scalability requirements, improving performance in such a way that horizontal scalability is a reality and incorporating new storage mechanisms [9].

This paper assesses three of the most popular NewSQL databases according to DB-Engines Ranking 2022 [10]: CockroachDB [11], MariaDB Xpand [12], and VoltDB [13]. The main purpose of this work is to decide which one has the best functionalities based on the OSSpal methodology and experimentally using the Star Schema Benchmark (SSB) to test its scalability and performance.

The open source software assessment methodology, OSSpal, came out as a successor of the Business Readiness Rating (OpenBRR), allowing users to make an evaluation with decisive questions divided into categories [14]. OSSpal methodology combines quantitative and qualitative software evaluation measures, which results in a numerical value that permits the comparison between applications. Some categories, which accommodate common characteristics, will be used to evaluate the open source and commercial solutions. OSSpal methodology has been used by the scientific community in several scientific papers and dissertations [15–19]. However, to the best of our knowledge, this is one of the first studies to apply the OSSpal methodology to NewSQL databases.

Star Schema Benchmark (SSB) is a benchmark designed as an alternative to TPC-H Benchmark. It is based on the TPC-H benchmark model with improvements that implement a traditional star schema. Its goal is to measure the performance of database products and to test a star schema strategy [20].

In the experiments using OSSpal methodology, we conclude that MariaDB has a higher score than CockroachDB and VoltDB. Each NewSQL database was evaluated using a collection of functionalities retrieved from the DB-Engines Ranking 2022, and other common characteristics that allowed us to determine which database is the best. In the experimental evaluation, using the Star Schema Benchmark, CockroachDB presented improved results in scalability, while VoltDB had better results in query execution time.

The main contributions of this work are the following:

- Revealing the strengths and weaknesses of NewSQL databases;
- Best database according to the evaluation of the OSSpal methodology;
- Experimental evaluation of NewSQL databases using a standard benchmark;
- Best NewSQL database regarding performance and scalability;
- Limitations in the practical use of NewSQL Databases.

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 describes the use of the OSSpal methodology and Star Schema Benchmark (SSB). Section 4 analyzes the three popular NewSQL databases, their advantages, and their limitations. Section 5 presents the evaluation based on OSSpal methodology and Star Schema Benchmark. Section 6 discusses the results of the experiments. Finally, Section 7 presents the conclusions and future work.

2. Related Work

There are not many works related to the evaluation of NewSQL databases and their performance. This section presents some of the related works that evaluate NewSQL database performance.

In Reference [21], the authors compared three NewSQL databases, VoltDB, MemSQL, and NuoDB, using the YCSB benchmark. Comparing the results, MemSQL had better results in most workloads, followed by VoltDB. NuoDB had some problems with data ingestion, having worse results.

Another group of authors in Reference [22] performed experiments with four NewSQL databases, VoltDB, NuoDB, CockroachDB, and MemSQL. They applied a set of operations

to evaluate read, write, update latency, and query execution time among the databases. NuoDB was the database with the best performance, followed by MemSQL, VoltDB, and CockroachDB being the database with the highest latencies among the four databases.

In Reference [23], the authors performed a benchmarking test against MySQL and Spanner, a NewSQL database. The authors used datasets provided by the Autonomous City of Buenos Aires. The results proved that when measuring its behavior by increasing the number of records and query complexity, Spanner performed better than MySQL.

In Reference [24], the authors performed an evaluation of two NewSQL databases, MemSQL and VoltDB, using TPC-H benchmark queries with 1 GB of data. During the evaluation, MemSQL performed better than VoltDB.

In Reference [25], the authors compared CockroachDB, MemSQL, NuoDB, and VoltDB with two benchmarks, YCSB and Voter. Comparing the results with both benchmarks, MemSQL performed better, having a higher throughput and lower latency. VoltDB and NuoDB performed similarly, despite NuoDB restrictions with the open source version. CockroachDB presented the worst results, with a lower number of transactions per second.

Another group of authors in Reference [26] performed a benchmarking test with weather data collected from Romanian cities between 2008 and 2020 using several NewSQL databases. The assessment was set with minimum resources, focusing on the read, write, and update latencies, as well as query execution. CockroachDB and FaircomCO presented higher values with write latency. Regarding query execution time, Citus, TIBCO, and VoltDB presented good results, with HarperDB being the database with the best results. Moreover, with update latency, CockroachDB presented inferior outcomes, while other databases had some difficulties with the number of records. Lastly, the read latency test showed that, once more, CockroachDB had the worst outcomes, while Citus and VoltDB presented lower read latencies.

The work conducted in this paper differs from that of the other authors presented in this section because we use a reliable benchmark (SSB) and a methodology (OSSpal) to choose the best NewSQL database. Furthermore, we use the latest available versions of the NewSQL databases.

3. Materials and Methods

This section describes OSSpal methodology and Star Schema Benchmark (SSB). OSSpal methodology is aimed at helping organizations and companies to find high-quality Free and Open Source Software (FOSS) that meets their needs [14]. Star Schema Benchmark is derived from the TPC-H benchmark, which was designed to evaluate the performance of databases with improvements against traditional database warehousing schema [20].

3.1. OSSpal Methodology

OSSpal combines quantitative and qualitative evaluation measures in several categories for software evaluation. Instead of a purely numeric calculated score, OSSpal adds FOSS projects criteria and individual user reviews to these criteria [19]. OSSpal methodology emerged as a successor of Business Reading Rating (OpenBRR) to give a reliable and independent evaluation of open source solutions [16].

According to [14], OSSpal methodology is composed of seven categories, which are the following:

- Functionality: Investigates the quality of software requirements from the users' viewpoint;
- Operational Software Characteristics: Evaluates the security, performance, GUI, and how easy the configuration of the software is;
- Support and Service: Analyzes the software support component for commercial or community applications;
- Documentation: Analyzes suitable documentation and tutorial guides for the software;
- Software Technology Attributes: Analyzes how well the software architecture is designed and how modular, portable, flexible, open, and easy to integrate it is.

- Community and Adoption: Examines the implementation of the component by the community, market, and industry. Moreover, evaluates the activity of the community for the software;
- Development Process: Estimates the level of professionalism of the development process and the project as a whole.

In the assessment of CockroachDB, MariaDB Xpand, and VoltDB, we had to adapt the OSSPal methodology because MariaDB Xpand is not open source. Therefore, the categories 'Community and Adoption' and 'Development Process' were removed because we could not evaluate the activity on forums and the development process of commercial software. Thus, the evaluation focused on the five remaining categories.

The 'Functionality' category is computed differently from the other categories. Functionality is evaluated based on a range of key features that an application already has or should have. The process to analyze this component is as follows:

- Outline key characteristics to analyze, assigning a score from 1 to 3 (1—slightly important; 2—important; 3—very important);
- Rank the characteristics in a cumulative sum (from 1 to 3);
- Standardize the previous result on a score from 1 to 5.

Therefore, the Functionality category will have the following scale:

- <65%: 1—Unacceptable;
- [65–80%]: 2—Poor;
- [80–90%]: 3—Acceptable;
- [90–96%]: 4—Good;
- >96%: 5—Excellent.

The rest of the categories are composed of four steps:

- First step: Select and identify all the software components to be analyzed and measured taking into consideration the evaluation criteria.
- Second step: Attribution of weights for the categories and the measures:
  - Each category is assigned a percentage of importance, making a total of 100%;
  - Each measure within the category is ranked by its importance;
  - Inside the category, assign the importance by percentage, making a total of 100% of all the measures of a category.
- Third step: In each category, score each measure used and assign it a value between 1 and 5, using the following scale: Unacceptable (1), Poor (2), Acceptable (3), Very Good (4), Excellent (5).
- Fourth step: Calculate the final OSSpal value based on the score for each category and the weighting factor.

3.2. Star Schema Benchmark (SSB)

The Star Schema Benchmark was designed to evaluate the performance of databases with improvements against traditional data warehousing schema [20]. The TPC-H benchmark has been criticized for not adopting Ralph Kimball's model of data marts, and not adhering to Edgar F. Codd's data schema definition of a Third Normal Form (3NF) [27]. Given these drawbacks in TPC-H, we chose this benchmark to evaluate the scalability and performance of CockroachDB, MariaDB Xpand, and VoltDB. SSB queries are based on the TPC-H model, from which 13 queries were defined, given the SSB schema [20]. These 13 queries are called query flights (Q{Query Flight}:N{Number of Query}) and are divided into four groups. Each group of queries is responsible for selecting the following data:

- Query Flight 1: The queries Q1.1, Q1.2, and Q1.3 restrict the revenue, with different ranges and filter factors, to find possible revenue increases. Overall, they calculate the increase in revenue that would have resulted from eliminating certain discounts by a certain percentage for products shipped in a given year.

- Query Flight 2: The queries Q2.1, Q2.2, and Q2.3 restrict the data in two dimensions, by comparing revenues from all orders in all years for suppliers in a given region and for a given product class.
- Query Flight 3: The queries Q3.1, Q3.2, Q3.3, and Q3.4 focus on restrictions in three dimensions, calculate revenue volume over a given time period by customer country, supplier country, and year within a given region.
- Query Flight 4: The queries Q4.1, Q4.2, and Q4.3 represent a 'What-If' sequence. It begins with query Q4.1 using a group by on two dimensions and weak constraints on three dimensions, and measure the aggregate profit, which is defined as (lo_revenue—lo_supplycost).

4. NewSQL Databases: CockroachDB, MariaDB Xpand, and VoltDB

This section describes the three NewSQL databases: CockroachDB, MariaDB Xpand, and VoltDB. These databases were selected as the three most popular NewSQL databases according to the DB-Engines Ranking 2022 [10].

4.1. CockroachDB

CockroachDB is a scalable database management system that was built in 2014 to support demanding OLTP workloads while maintaining simultaneously strong consistency and high availability [28]. CockroachDB, as its name implies and according to its creators, is disaster-resistant through automatic replication and recovery mechanisms. CockroachDB is a distributed relational database that scales without much effort, which is consistent with ACID transactions and provides a traditional SQL API for structuring, manipulating, and querying data [29]. Moreover, this database was built on a consistent key–value store, with crash recovery capabilities, and offers horizontal scalability, present in many NoSQL applications.

CockroachDB's architecture is designed in layers to make it easier to manage. Figure 1 shows a diagram of its architecture [30].

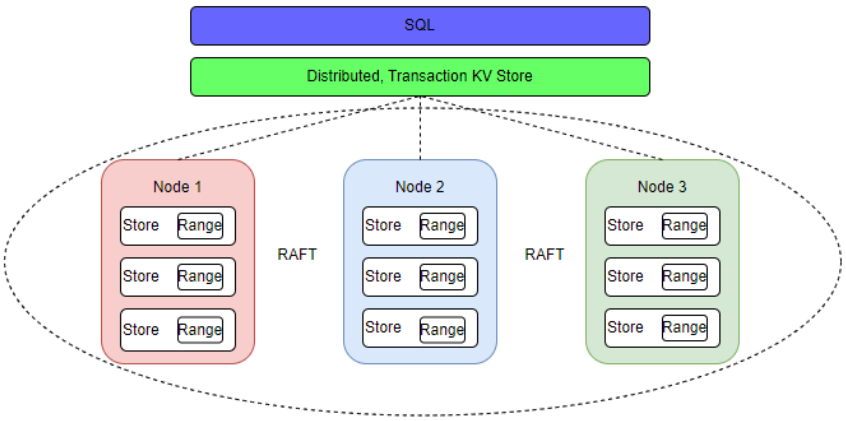


Figure 1. CockroachDB architecture diagram.

CockroachDB is divided into layers, and each one has the following functionalities:

- SQL Layer: The highest level of abstraction for developers. This layer adds support for a wide range of SQL expressions and syntax from PostgreSQL libraries, with some modifications;
- Distributed Key–Value Store: This layer was implemented as a monolithic sorted map, allowing to run multiple computers in parallel, being able to work with larger datasets;
- Distributed Transactions: It is not necessarily considered a part of the layered architecture, but a necessary component of the system. Implementing distributed transactions allows to connect the layers of the architecture: from SQL to stores and ranges on each node;

- Nodes: They are mostly considered as physical machines, virtual machines, or containers that include stores. The distributed key–value (KV) store routes messages to nodes;
- Store: Each node in the database can contain one or more shops, and in turn, each shop can contain many ranges;
- Range: Ranges are the lowest-level unit of key–value data and every store contains ranges. Each range is used to sort items in specific partitions within the store, using the Raft consensus algorithm. The Raft algorithm is a variant of Paxos, which corresponds to a family of protocols that solves consensus in a network of unreliable processors.

Figure 2 shows an example of three nodes running in a cluster, having access to the CockroachDB web user interface (UI).

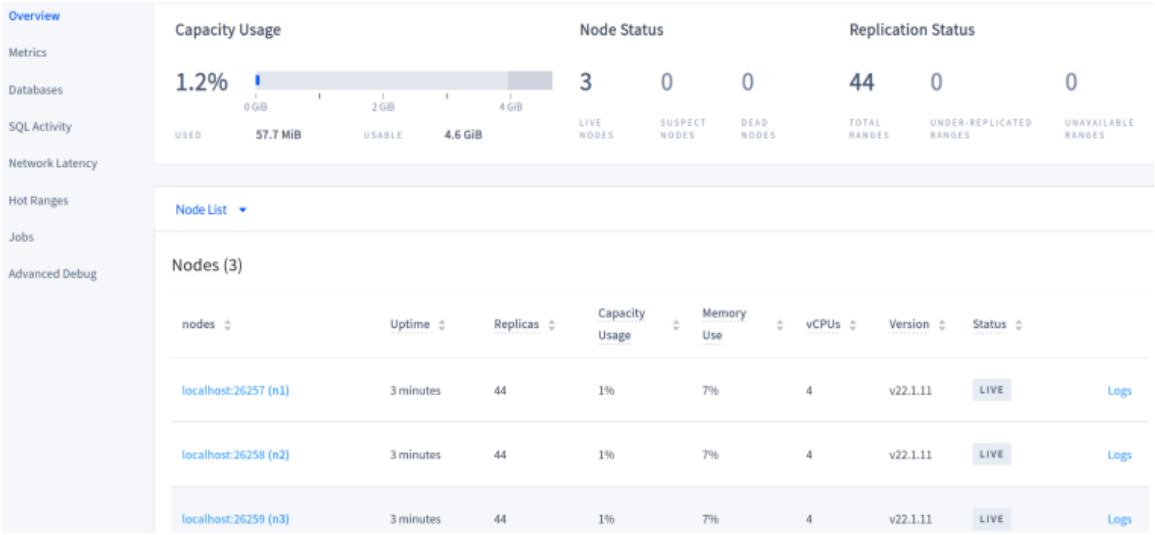


Figure 2. CockroachDB v21.2.10 web user interface cluster.

The main advantages of CockroachDB are [31]:

- Cockroach offers customer support with migrations from other databases, as well as helping users with difficulties in using this database;
- CockroachDB can be used anywhere. It can be deployed in virtual machines, containers, Amazon web services, and many other applications;
- CockroachDB maintains data integrity and is able to survive crashes, due to the use of ACID properties;
- CockroachDB offers high performance and availability;
- CockroachDB scales horizontally and offers cloud support;
- CockroachDB has extensive documentation and tutorials guides;
- Supports PostgreSQL libraries to make use of SQL commands.

The main limitations are the following [32]:

- CockroachDB does not support database transactions with high complexity, since this database's purpose is speed;
- To make full use of CockroachDB, we need to pay for the enterprise version, whereas the core version can be used for most purposes with several functionality restrictions;
- Multi-region tables cannot be restored into tables that are not multi-region tables;
- SQL statements comprising numerous subqueries modifying the same table can cause corruption;
- CockroachDB does not support the use of RESTORE with multi-region table localities;
- The SET command, which allows modifying one of the session configuration variables, does not ROLLBACK in a transaction;
- JSONB/JSON comparison operators are not implemented.

Overall, CockroachDB is a database that has several functionalities, which allows working with a large data volume and scale without much effort.

4.2. MariaDB Xpand

MariaDB Xpand is a new product from MariaDB that focus on scaling SQL without sacrificing the relational data model and ACID transactions. It was formerly known as ClustrixDB before being acquired in 2018 by MariaDB Corporation. It is no longer a separate solution but is now included as part of MariaDB Enterprise Server as a plugin. MariaDB Xpand, shortly defined as Xpand, was developed to handle large volumes of data and scales the database more efficiently [33]. This means that scaling out the cluster is executed easily and automatically by Xpand itself. Xpand excels at reading and writing scaling data applications with absolute availability in the case of multiple zone failures. Additionally, Xpand offers columnar indexes to enable ad hoc operations, and personal analytics on transactional data [34]. On the other hand, Xpand manages data with multiple copies shared between nodes. In case one node fails, the other nodes will use the copied data from the faulty node, without external intervention, maintaining availability. Furthermore, it provides strong consistency of records, guaranteeing ACID properties of OLTP transactions. According to documentation, MariaDB Xpand is based on Xpand Performance Topology [35]. The Xpand Performance Topology consists of one or more MaxScale nodes and three or more Xpand nodes since the minimum number of nodes is three [36]. In Figure 3, we can see an example of Xpand Performance Topology.

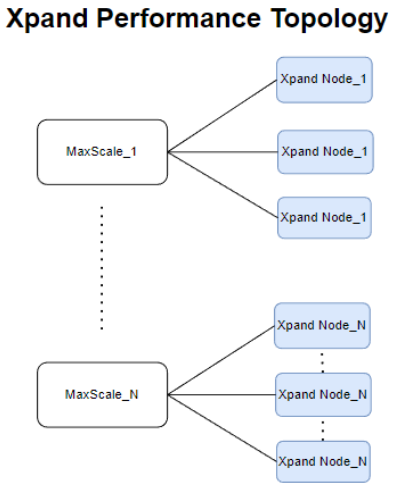


Figure 3. Xpand Performance Topology.

The MaxScale nodes monitor the health and availability of each node and accept clients and application connections. The Xpand nodes receive queries from MaxScale nodes, store data in a distributed manner, and execute queries using parallel streaming. Figure 4 shows an example of the MariaDB Xpand user interface.

The main advantages of MariaDB Xpand are the following [37]:

- It provides distributed SQL capabilities and is ACID-compliant;
- It is highly available due to maintaining replicas of each slice (logical representation of data that are saved in a partition of a disk, which contains pieces of user database and tables), allowing to recover from a node failure without losing data;
- It can maintain multiple replicas of each slice and is zone-aware, allowing to recover from multi-node failures or zone failures without losing data;
- Its rebalancer maintains data distribution, meaning that a node or zone failure causes the creation of new replicas for each slice, and the rebalance then redistributes the data;
- Performs operations in parallel through the nodes to have the latest data;

- It scales out because each node can read and write, plus reads are lockless. Writes do not block reads, and additional nodes can be added to increase capacity.

However, MariaDB Xpand also has the following limitations [37]:

- Difficulty with migrations with modern software;
- Recovery of crash from data replication takes too long;
- No option to export stored procedures' query results.

Overall, MariaDB Xpand is a SQL distributed database that allows scaling out without much difficulty, with the ability to access data through computers around the world. On the other hand, it has limitations in analyzing query performance and in the ability to migrate data with modern applications.

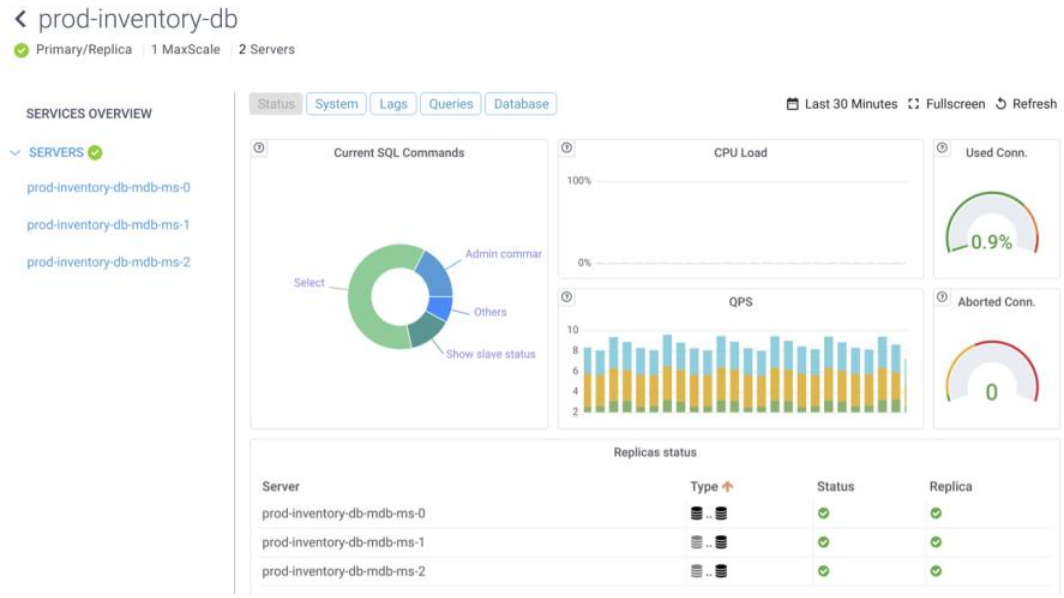


Figure 4. MariaDB Xpand v6.0.3 graphical user interface.

4.3. VoltDB

VoltDB is an in-memory SQL database for modern applications with the ability to manage data at an unprecedented scale, volume, and accuracy [38]. The idea behind VoltDB is to be faster than traditional relational DBMSs on a certain class of applications. VoltDB is based on ACID properties upon a distributed architecture designed for scalability and high availability. This system was designed in 2010 and its architecture is a shared-nothing, in-memory, and distributed relational database [6]. This solution makes use of horizontal scalability, which allows scaling data by adding more nodes (instances of machines) to the cluster [38]. For high availability, VoltDB provides four features that ensure the data remain consistent in the face of external or internal failures:

- Snapshots;
- Command Logging;
- K-safety;
- Database Replication.

Figure 5 shows how VoltDB partitions work within the cluster.

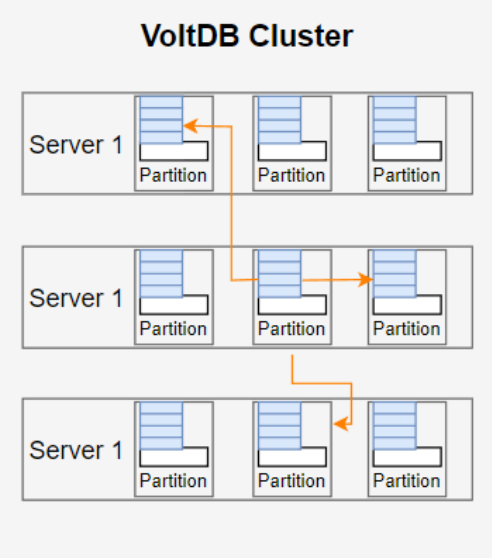


Figure 5. VoltDB architecture components.

Each VoltDB partition acts autonomously by allowing the cluster to manage parallel transactions [39]. When a procedure requires data from multiple partitions, one node acts as a coordinator and distributes the necessary work to the other nodes.

Figure 6 shows an example of the VoltDB user interface.

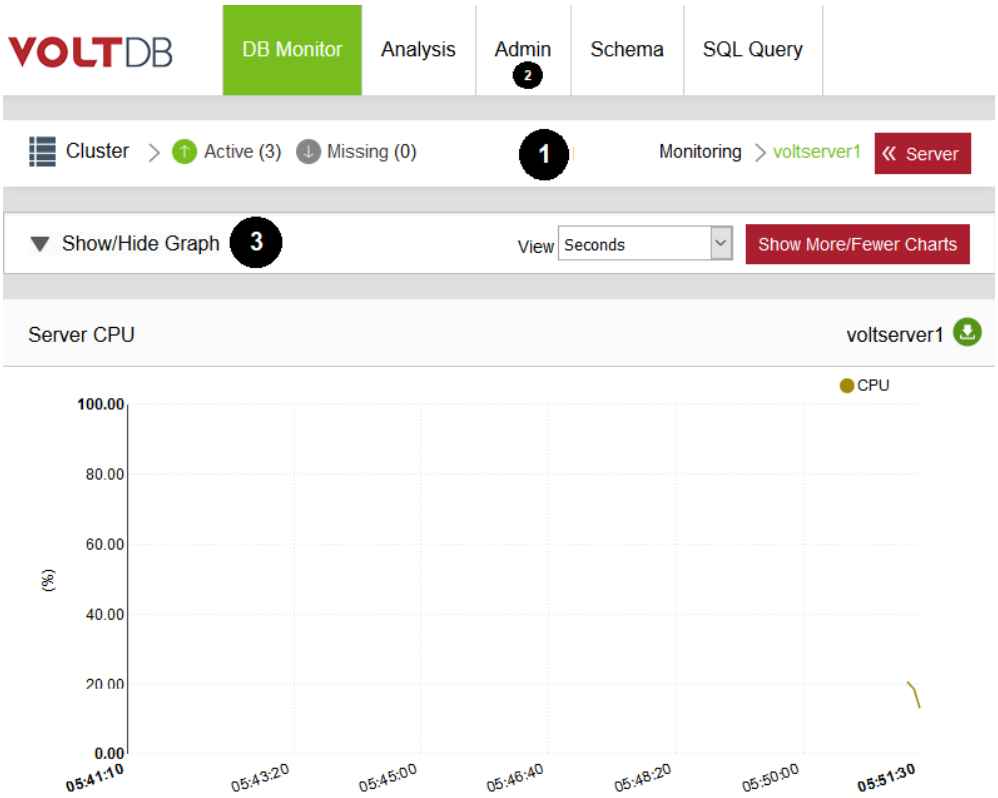


Figure 6. VoltDB v11.3 deployment manager interface.

Figure 6 presents the user interface, which is divided into three parts, where each one is responsible for a specific task [40]:

(1) Shows the number of nodes associated with the current database. It allows to add or remove nodes from the database;
(2) Configuration properties of the current database;
(3) VoltDB displays CPU, memory, latency, and transaction metrics to help the user analyze the execution of the database.

The main advantages of VoltDB are the following [41,42]:

- In-memory storage: It uses synchronous replication for durability;
- Designed for online/operational transaction processing, OLTP;
- Optimized performance: Each execution engine is single-threaded. It removes latches, locks, and buffer pool management in order to eliminate overhead and legacy architecture systems;
- VoltDB allows to scale without much difficulty;
- Distributed database: In this case, the tables are partitioned, with one partition per node;
- VoltDB uses Java stored procedures to eliminate client–server round-trips;
- VoltDB provides several options for backing up database data and schema;
- High availability: VoltDB contains three capabilities, namely K-safety, network fault tolerance, and live node rejoin.

The major limitations of VoltDB are the following:

- Not optimized for OLAP, since VoltDB is focused on OLTP, and is not good at solving problems that require large database column scans;
- Not optimized to return an excessive amount of data from stored procedures;
- The database must fit into the available memory of the system;
- Not optimized to work with complex queries and huge tables;
- No support for foreign keys and check constraints;
- No Windows version available.

VoltDB is a fast in-memory database that provides high availability and fast query response. On the other hand, VoltDB does not support the Windows operating system.

4.4. Summary of NewSQL Databases Characteristics

This section presents a summary of the main characteristics of CockroachDB, MariaDB Xpand, and VoltDB.

In Table 1, regarding partitioning, each solution uses its own method, whether it is by range or sharding. These methods are present in many NewSQL solutions and support several replication methods. The three solutions have multiple parallel transactions, which ensures that every single database is synchronized.

Table 1. NewSQL databases summary.

	CockroachDB	**MariaDB Xpand**	**VoltDB**
Developer	Cockroach Labs	Cluxtrix Inc.	VoltDB Inc.
Creation Date	2015	2018	2010
License	Open Source	Commercial	Open Source
Latest Release	v22.1.3, 11 July 2022	v6.0.3, 10 March 2022	v11.3, 8 February 2022
Initial Release	2015	2020	2010
Data Model	Relational	Relational	Relational
Server Operating Systems	Linux, macOS, and Windows	Linux, Windows, macOS	Linux, macOS
Data Schema	Dynamic	Yes	Yes
Partitioning method	Horizontal, by key range	Range	Sharding
Replication Method	Multi-source using RAFT	Parallel Streaming	Source and Multi-Source Replication

Table 1. *Cont.*

	CockroachDB	MariaDB Xpand	VoltDB
SQL Support	Yes	Yes	Yes
Transaction Properties	ACID	ACID	ACID
Implemented Language	Go, Typescript, Starlark, and Yacc	C	Java, C++
Supported Languages	C#, C++, Clojure, Go, Java, JavaScript (Node.js), PHP, Python, Ruby, and Rust	C, C++, Java, Javascript, Python	Java, Python, Ruby, Scala, Javascript

Moreover, VoltDB does not support Windows, and regarding supported languages, each database has a variety of programming languages that allows us to create applications to support each database. It is important to notice that all the latest releases are from 2022. Overall, each solution was designed to support large dataset volumes, without losing ACID properties.

5. Experimental Evaluation

In this section, we describe the assessment of each NewSQL Database, CockroachDB, MariaDB Xpand, and VoltDB with OSSpal methodology, and the experimental evaluation using the Star Schema Benchmark.

5.1. OSSpal Evaluation

Using the OSSpal methodology, we first define the weight of each of the five categories: 'Functionality', 'Operational Software Characteristics', 'Documentation', 'Support and Service', and 'Software and Technology Attributes'. Table 2 shows the weights assigned to each category, making a total of 100%. These weights were based on several successful studies, such as [14–17]. However, we had to make some adjustments since MariaDB Xpand is not open source. Given this difference, we recall that two categories were removed: 'Community and Adoption' and 'Development Process'. The category 'Functionality' is the most important, which was given the highest weight (35%). The next category, 'Operational Software Characteristics', is defined with 25%, since it is related to scalability, user interface, setup, and reliability. The third and fourth categories used in this assessment are 'Documentation' and 'Support and Service' with weights of 15% because a suitable database must have good documentation to help with installation and configuration, and feedback from internal or external sources. Finally, the category 'Software Technology Attributes' was assigned with 10% because it measures the use of third-party tools and support languages.

Table 2. Assigning weights to categories.

Categories	Weight
Functionality	35%
Operational Software Characteristics	25%
Documentation	15%
Support and Service	15%
Software Technology Attributes	10%

The next step is to define and evaluate the different characteristics that involve NewSQL solutions to analyze the 'Functionality' category. The features chosen to evaluate the tools were based on [36]. At this stage, the characteristics that suit both types of tools were included. For this, a score was assigned to each feature on the following scale: 1—slightly important, 2—important, 3—very important. Table 3 shows the weights and

marks assigned to each metric according to what we consider to be the most important features of a NewSQL database.

Table 3. Functionality metrics.

Metrics	Weight	CockroachDB	MariaDB Xpand	VoltDB
OLTP	3	3	3	3
CRUD	3	3	3	3
Replication	3	2	2	3
Partitioning	3	3	3	3
Consistency	3	3	3	3
Crash Recovery	3	3	3	3
Triggers (SQL)	2	0	2	0
Indexes (SQL)	2	2	2	2
PL/SQL functions	2	1	2	2
Cloud Support	1	1	1	1
Total	25	21	24	23
% Score		84%	96%	92%
Final Score		3	4	4

After assigning the weight's attribution to all categories, each tool evaluation is performed to assess which database earns the highest score.

The final evaluation in each category, besides 'Functionality', is based on the following formula:

$$\text{Final weight} = (\text{weight assigned} \times \text{metric weight}/100\%) \quad (1)$$

The results of the evaluation are presented in Table 4.

Table 4. OSSpal score by category.

Categories	Weight	CockroachDB	MariaDB Xpand	VoltDB
Functionality	35%	3	4	4
Operational Software Characteristics	25%	4	3.75	3.5
Documentation	15%	5	4.65	4.65
Support and Services	15%	5	5	5
Software Technology Attributes	10%	4.35	3.75	3.75
Total		3.99	4.16	4.10

From the analysis of Table 4, it is possible to conclude:

- In the 'Functionality' category, MariaDB Xpand and VoltDB stand out with scores of 4, which means that they have almost all features that we considered important in evaluating NewSQL databases;
- For 'Operational Software Characteristics', CockroachDB has the highest score;

- In 'Documentation', CockroachDB has the highest score by having very good documentation available for users;
- In 'Support and Services', every single database has a score of 5, which means that all databases have many people who require their services for training and webinars for new updates;
- Finally, in 'Software Technology Attributes', CockroachDB has the maximum score.

As a result, the database with the highest score is MariaDB Xpand with 4.16, followed by VoltDB with 4.10, and lastly, CockroachDB with 3.99.

Although these databases have close values, it will be useful to assess which one has the best performance and scalability by evaluating its performance with a standard benchmark, as in the next section.

5.2. Performance Evaluation with Star Schema Benchmark

In this section, we evaluate each database using the following metrics: loading data time, % CPU used, and memory usage. The 13 queries of SSB are executed five times from Q1.1 to Q4.3 sequentially. At the end of each run, we restart the database to prevent caching effects, and the final result is the average of the five runs.

The tests were performed on a computer with the following characteristics:

- Intel(R) Core (TM) i7-10700 CPU @ 2.90 GHz;
- 32 GB of RAM;
- SSD disk with 496 GB;
- VirtualBox with Operating System Ubuntu 20.04, 7 virtual cores, 28 GB virtual RAM, and 200 GB of allocated disk.

We use a scale factor (SF) = 1, which corresponds approximately to 1 GB of data size, and SF = 10, which is approximately 7 GB.

Table 5 shows the file size for each table of the SSB schema, and the number of rows of each table.

Table 5. File size and data import with SF = 1 and SF = 10.

	SF = 1		**SF = 10**	
Tables	**File Size**	**Rows**	**File Size**	**Rows**
customer	3.2 MB	30,000	32.6 MB	300,000
date	278.2 KB	2557	278.2 KB	2557
part	19.7 MB	200,000	79.3 MB	800,000
supplier	188.7 KB	2000	1.9 MB	20,000
lineorder	641.1 MB	6,001,173	6.6 GB	59,986,217
Total	∼1 GB	6,235,730	∼7 GB	61,108,774

5.2.1. Data File Import with SF = 1 and SF = 10

In this subsection, we compare the loading time in each database to determine which one has the fastest data insertion. The results of the loading time can be seen in Figure 7.

The results of loading the tables in Figure 7 show that CockroachDB has a faster loading time than VoltDB and MariaDB Xpand. Comparing the results from both scale factors, we expect that by increasing the size of the data by 10 times, it would take 10 times longer to load the data. Instead, there is an increase of only 7.17, 1.99, and 2.10 times for CockroachDB, MariaDB Xpand, and VoltDB, respectively. This confirms that all databases scale without much effort when more data are added to the database. Moreover, MariaDB Xpand and VoltDB have similar loading times. Overall, each database demonstrates that with its characteristics and results, they adapt well to large amounts of data.

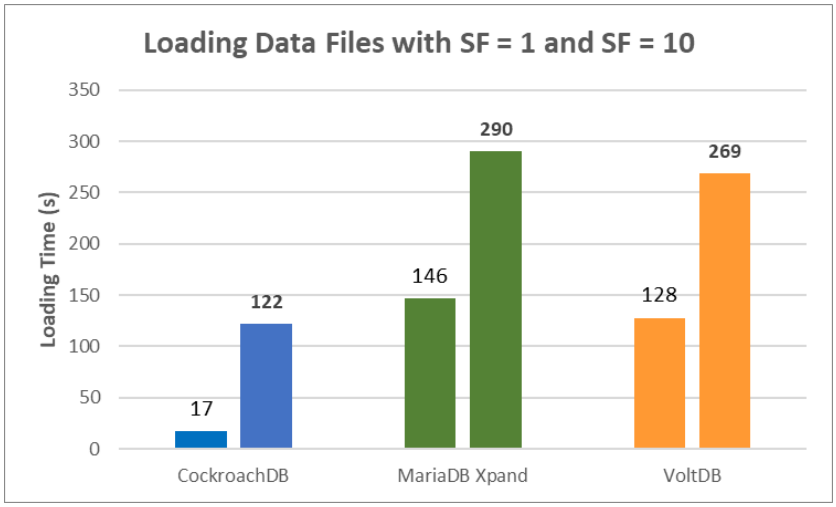


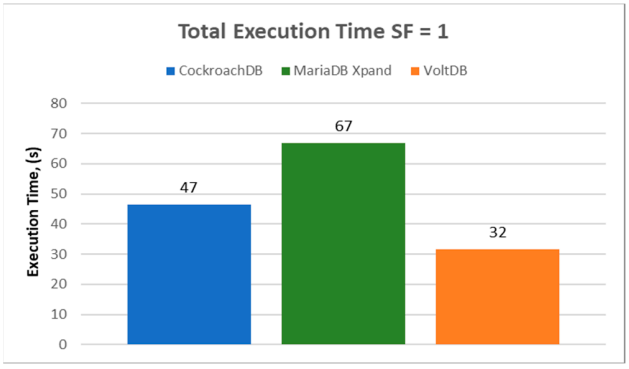
Figure 7. Results for loading data files with SF = 1 and SF = 10.

5.2.2. SSB Performance Evaluation with SF = 1

In this subsection, we evaluate the performance of CockroachDB, MariaDB Xpand, and VoltDB by analyzing query execution time and the average resource usage of RAM and CPU for each SSB query flight group with a scale factor of 1. It should be noted that the values presented are the average of five runs for each query.

The results in Figure 8 show that in general, VoltDB has the best query execution time with SF = 1. Regarding the sum of all SSB queries' execution time, we have 32 ms, 47 ms, and 67 ms for VoltDB, CockroachDB, and MariaDB Xpand, respectively. This is due to the fact that VoltDB is an in-memory database, so the data are easily accessed, while the other databases use disk access. While analyzing VoltDB queries, we observed that from query flight Q2 to Q4, there was a substantial increase in execution time. These queries had a higher execution time than the first query flight Q1 because they searched for information in a larger number of tables and needed to perform more operations to present the results. The remaining group, Q1, searches for information in smaller tables, which explains why there is a shorter execution time.

(**a**) Average Query Execution Time with SF = 1



(**b**) Average Total Time Results with Scale Factor 1

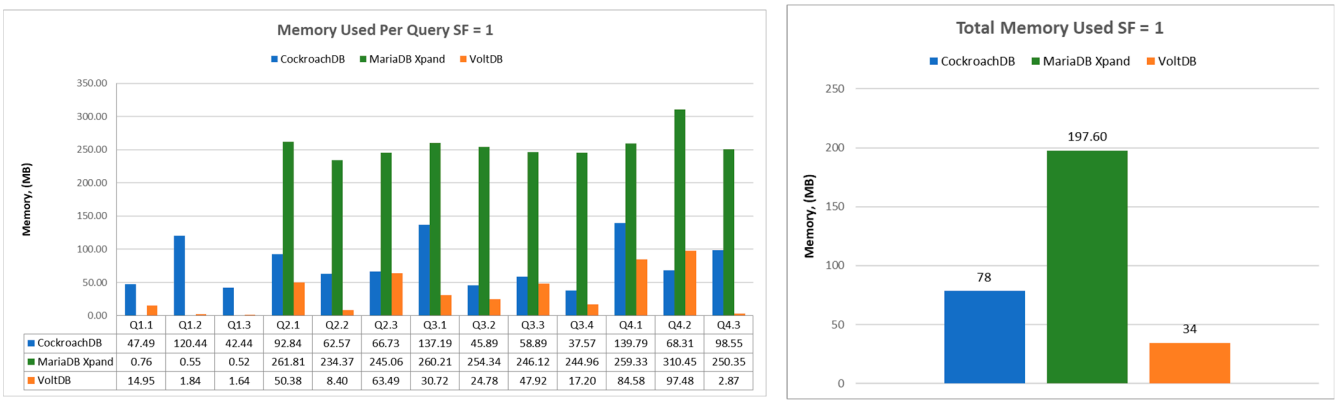
Figure 8. Results of query execution time with SF = 1.

On the other hand, CockroachDB has an almost linear execution time compared to the other two databases. This demonstrates that no matter how complex the query is, the time to process and return a result would be almost the same as the other queries.

Lastly, MariaDB Xpand takes more time to execute the queries. In the first flight group, we were surprised that the execution time is almost the same as VoltDB. We were expecting to see close results to VoltDB, but the remaining queries took more time to process,

especially Q2.1, Q3.1, Q3.2, Q4.2, and Q4.3, which are the queries that access more tables and return the most data.

Figure 9 shows the results of RAM usage in the queries execution with SF = 1. VoltDB presents a better RAM consumption, with an average total of 34 MB, while CockroachDB and MariaDB Xpand results are higher, with 78 MB and 198 MB, respectively. VoltDB has a higher memory consumption in queries Q4.1 and Q4.2, with 85 MB and 98 MB usage, respectively. This last group of queries has more aggregations and data to be retrieved from tables with large data volumes. In the remaining queries, the memory consumption was below 100 MB.



(**a**) Average Memory Used per Query with SF = 1 (**b**) Average Total Memory Used with SF = 1

Figure 9. Results of memory used with SF = 1.

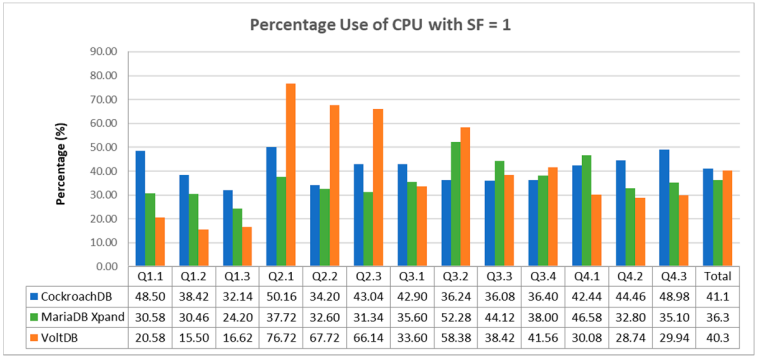
On the other hand, CockroachDB is the second database with the highest memory used, with 78 MB. This database displayed better control over the usage of RAM, with small peaks in queries Q1.2, Q3.1, and Q4.1. Moreover, given the complexity of queries executed in this experiment, CockroachDB has a controlled memory usage.

MariaDB Xpand is the database with the highest use of memory. This database has an increase of 2.54 times the usage of memory comparing to CockroachDB and 5.83 times the VoltDB usage of memory. MariaDB Xpand requires more RAM when queries with more aggregated operations are performed, as we can see with groups Q2, Q3, and Q4. Group Q1 has one aggregator operation so, the use of memory was relatively low.

Figure 10 illustrates the results of CPU used in the queries execution with a SF = 1. MariaDB Xpand is the database with the lowest percentage of CPU usage, with 36.3%, while VoltDB has an average of 40.3%, and CockroachDB with 41.1% of CPU usage. MariaDB Xpand results show an increase in CPU usage from Q1.1 to Q2.2, since more tables were accessed, and from query Q3.2 to Q4.3, we notice a decrease in CPU usage. Even though this group of queries has a higher number of tables with aggregations, MariaDB Xpand displays adequate results with the last group of queries.

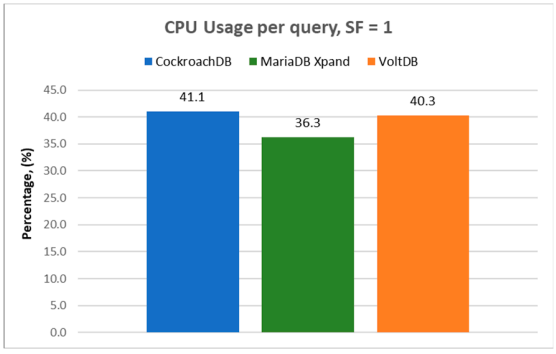
VoltDB is the second database with the highest use of CPU, with 40.3%. By analyzing the queries during VoltDB operation, we notice a substantial increase in queries Q2.1, Q2.2, Q2.3, and Q3.2. These queries have an average use of CPU between 50% and 80% since VoltDB requires more resources to process and return a result.

Lastly, CockroachDB has a higher average of CPU use, with an almost linear percentage of CPU usage from query group Q2 to Q4. These queries have a higher processing time than group Q1 since these queries have more aggregations.



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Total
CockroachDB	48.50	38.42	32.14	50.16	34.20	43.04	42.90	36.24	36.08	36.40	42.44	44.46	48.98	41.1
MariaDB Xpand	30.58	30.46	24.20	37.72	32.60	31.34	35.60	52.28	44.12	38.00	46.58	32.80	35.10	36.3
VoltDB	20.58	15.50	16.62	76.72	67.72	66.14	33.60	58.38	38.42	41.56	30.08	28.74	29.94	40.3

(**a**) Average Percentage Use of CPU core per Query Scale Factor 1



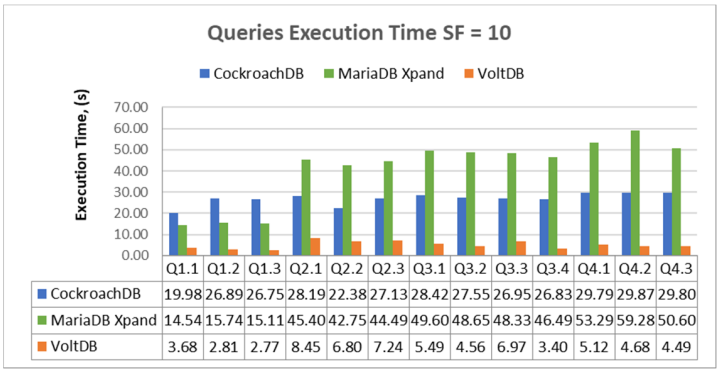
(**b**) Average Total Percentage Used of CPU core Scale Factor 1

Figure 10. Results of percentage usage per core with SF = 1.

5.2.3. SSB Performance Evaluation with SF = 10

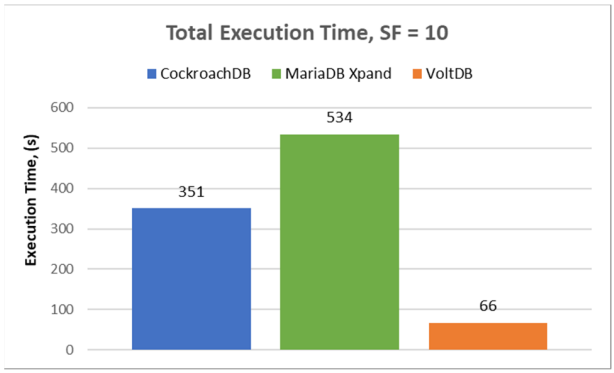
In this subsection, we evaluate the performance of CockroachDB, MariaDB Xpand, and VoltDB by analyzing query execution time and the average consumption of CPU and RAM for each query flight group, increasing the size of data with a scale factor of 10.

In Figure 11, VoltDB, once again, manages to be the fastest database due to its in-memory storage, with an increase of 2.06 times compared to SF = 1. The query execution time is faster compared to the other databases, being below the 10 s mark throughout all queries.



	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
CockroachDB	19.98	26.89	26.75	28.19	22.38	27.13	28.42	27.55	26.95	26.83	29.79	29.87	29.80
MariaDB Xpand	14.54	15.74	15.11	45.40	42.75	44.49	49.60	48.65	48.33	46.49	53.29	59.28	50.60
VoltDB	3.68	2.81	2.77	8.45	6.80	7.24	5.49	4.56	6.97	3.40	5.12	4.68	4.49

(**a**) Average Query Execution Time with SF = 10



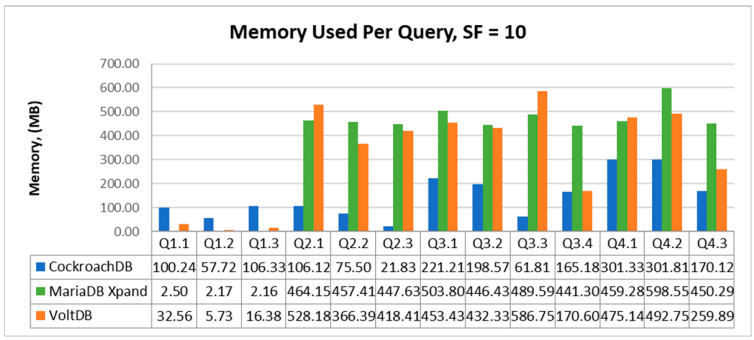
(**b**) Average Total Time Results with SF = 10

Figure 11. Results of execution time with SF = 10.

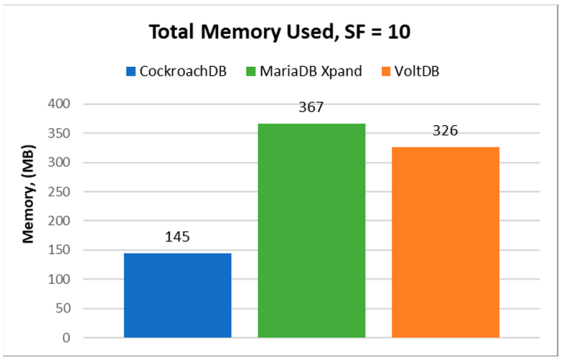
CockroachDB, by increasing the scale factor 10 times, displays an average increase of 7.47 times. The results in Figure 11 show a similar linear execution time in Figure 8 using SF = 1. Furthermore, CockroachDB manages to have a linear execution time despite the complexity of the queries and the increase in data size.

MariaDB Xpand is the database which takes more time to execute the queries, with an increase of 7.97 times compared to SF = 1. In this case, MariaDB Xpand does not show the same performance in the first group of queries Q1 as VoltDB in Figure 8 with SF = 1. In addition, MariaDB Xpand is the database that shows the highest query execution time, with an increase of 1.52 times compared to CockroachDB and 8.1 times compared to VoltDB.

Figure 12 shows the results of RAM percentage consumed in the query execution time with SF = 10. CockroachDB demonstrates better results with memory consumption, with an average of only 145 MB. Throughout all queries, CockroachDB displays good memory control through the execution of each query, with group Q4 being the highest group to consume memory, between 170 MB and 305 MB. The remaining queries remained below 225 MB.



(**a**) Average Memory Used per Query with SF = 10



(**b**) Average Total Memory Used per Query with SF = 10

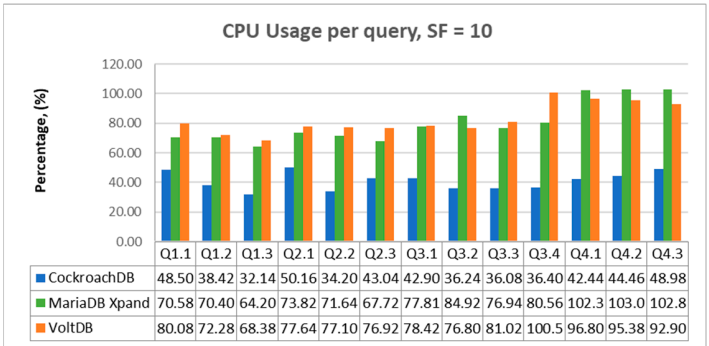
Figure 12. Results of memory usage with SF = 10.

By contrast, VoltDB is the second highest, with a total average memory use of 326 MB. The first query flight group Q1 displays adequate results between 5 MB and 35 MB. The other group of query flights had a higher consumption, between 300 MB and 600 MB, an approximate increase of 9 times the consumption of RAM.

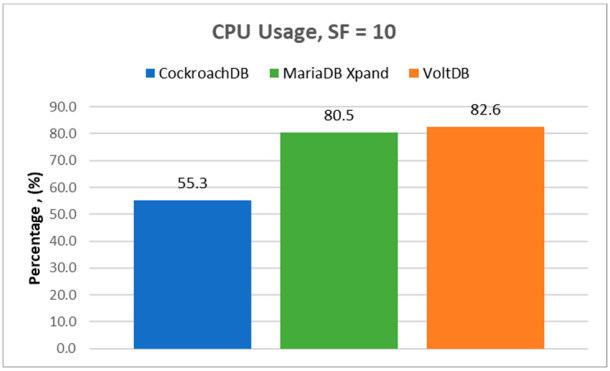
MariaDB Xpand is the database with the highest average consumption of memory, having an average consumption of 367 MB. In query groups Q2, Q3, and Q4, the higher the number of aggregations, the higher the consumption of memory, as this database requires greater memory usage when more aggregations are involved.

Despite these queries, query group flight Q1 displays lower RAM consumption, since these queries search for data only in small tables, having simpler queries.

Figure 13 shows the results of the CPU percentage used in the query execution with SF = 10. CockroachDB is the database with the lowest average CPU use, with 55.3%, while MariaDB Xpand and VoltDB have an additional increase of 1.4 times compared to SF = 1, with 80.5% and 82.6%, respectively. As mentioned before, CockroachDB has sufficient control over its resources, despite the complexity of each query, demonstrating an almost linear percentage of CPU use during the processing time of each query.



(**a**) Average Memory Used per Query with SF = 10



(**b**) Average Total Memory Used per Query with SF = 10

Figure 13. Results of CPU usage with SF = 10.

On the other hand, MariaDB Xpand appears in second place, with an average use of CPU of 80.5%. During the processing time of each query, we noticed that in query flight group Q4, MariaDB Xpand exceeds the 100% scale. It is possible that one or more processes executed at the same time with the query lead to an increase in resources in the virtual machine.

Lastly, VoltDB database has the highest CPU usage. We notice a peak of CPU in query Q3.4, having surpassed the scale of 100%, meaning that during this evaluation, the

benchmark used one full core of the processor. We suppose that during the execution of this query, VoltDB had more processes in the background, leading to an increase in CPU usage. The remaining queries have an average CPU usage between 60% and 100%, being the database with the most resources used during this experiment.

Figure 14 shows the sum of query execution time for all SSB queries with SF = 1 and SF = 10. These results demonstrate that MariaDB Xpand stands out as the database with the highest time to execute all queries. It spent a total of 67 s with SF = 1 and 534 s with SF = 10, which corresponds to about 8 times more. CockroachDB is the second database with 47 s, 7.5 times faster than SF = 10 with 351 s. VoltDB is the fastest database, with 66 s, only 2.1-fold more than SF = 1 with 32 s. Overall, VoltDB showed satisfactory results equally with SF = 1 and SF = 10. This proves that an in-memory database is significantly faster compared with CockroachDB and MariaDB Xpand.

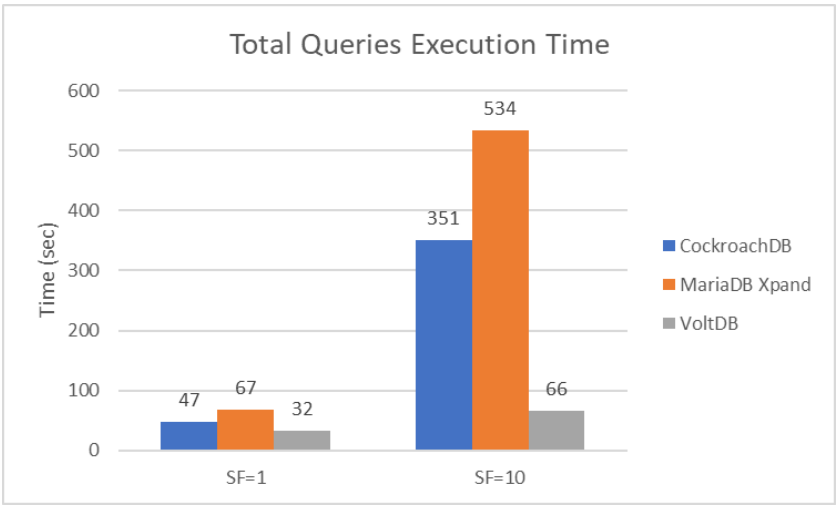


Figure 14. Sum of queries between SF = 1 and SF = 10.

In summary, VoltDB presents the best results of query execution time for all SSB queries. Its scalability should also be noted: when the dataset size increases by 10 times, the query execution time has only an increment of 2.1 times.

6. Discussion

The main purpose of this paper is to explore the topic of NewSQL databases and to provide an analysis of the best databases, using OSSpal methodology and a standard benchmark (SSB) to evaluate their performance and scalability.

The study conducted shows that using OSSpal methodology, MariaDB Xpand achieves 4.16 points, VoltDB achieves 4.10 points, and CockroachDB achieves 3.99 points, as shown in Figure 15. These values are only determined by analyzing and selecting key functionalities and qualities that each database presents, which can bias the results. Despite having close results, each database has been designed to work with large volumes of data, as well as the capability of scaling. Therefore, having assessed each database, we conclude that MariaDB Xpand is highlighted as the tool with the best characteristics, according to OSSpal categories.

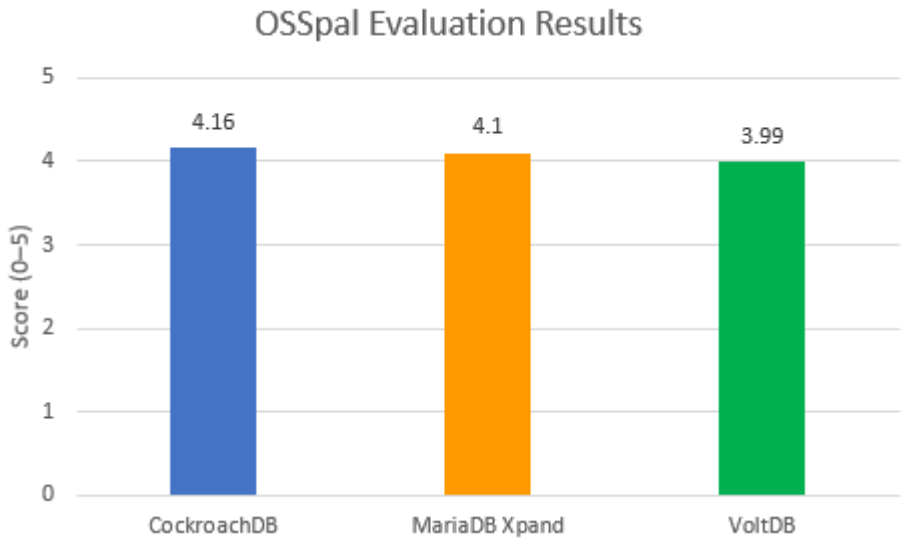


Figure 15. OSSpal evaluation results.

In the experimental evaluation, the Star Schema Benchmark was applied to analyze the performance and scalability of each database, given the scale factor of 1 and 10, corresponding to 1 GB, and ~10 GB of data, respectively. Figure 16 illustrates the total amount of time needed to upload 1 GB and 10 GB of data. We presume that each database would take 10 times to upload the data, but the results proved to be better. This is due to the upload system used by each database. CockroachDB uses an import mechanism that allows data to be processed and stored in key–value stores, passing through several layers of its architecture directly to the storage layer, improving the efficiency of data import, ranking first with a total of 139 s. VoltDB uses an integrated Java mechanism called csvloader, which imports CSV files into the database. This utility allows data to be imported efficiently into memory in 397 s, decreasing performance penalties of disk accesses. However, we expected that being an in-memory database, VoltDB would outperform the other two databases. Lastly, MariaDB Xpand uses a mechanism in Python called clustrix_import, which scans the data to know how much space it needs to allocate in storage before importing data. It also uses a rebalancing system that distributes data throughout the existing nodes in the cluster bringing optimization results. This database, despite having similarities to the MySQL storage mechanism, lagged behind CockroachDB and VoltDB.

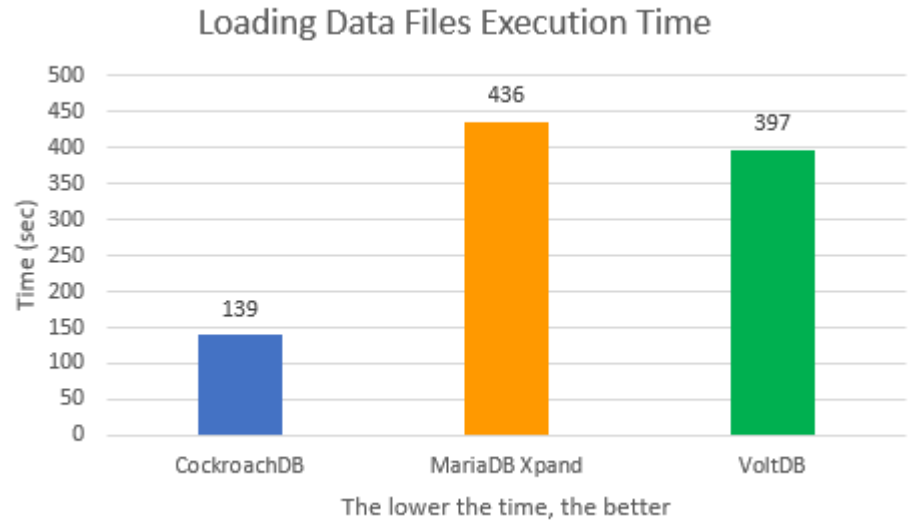


Figure 16. SSB total loading times with SF = 1 and SF = 10.

Additionally, to evaluate the performance of CockroachDB, MariaDB Xpand, and VoltDB, we used 13 suitable queries provided by SSB benchmark. These queries diverge into four groups with three or four queries each. As the number of groups increases, so does the complexity of queries. These 13 queries comprise several instructions designed to assess each database's capabilities. With SF = 1, VoltDB, being an in-memory database, displayed better results than CockroachDB and MariaDB Xpand. This type of database is generally faster and more efficient than disk databases because data can be accessed and processed directly from memory. Among the four query groups, VoltDB's best performance was with query group 1, which involves less complexity and table scans. This database works better with queries that do not include a large scan of data and complex queries. With this circumstance, the results of the execution time of other queries' performance were noticeable. Even increasing the scalability to SF = 10, VoltDB once again displayed fast execution time. In this part of the assessment, we experienced a decrease in query execution time from group 2 to group 4.

CockroachDB was also a database that showed interesting results. We did not expect that there would be an almost linear execution time between all queries, either with SF = 1 or SF = 10. We analyzed each query with the help of the 'EXPLAIN' command to inspect and tune the query. This command is employed to analyze queries with a complex structure that returns large amounts of data. To optimize the queries, this command restructures the queries in small levels of processing. The command displayed how that process is performed, and the conclusion was that by searching tables with large volumes of rows, CockroachDB would take a loss of performance. To prevent this situation, the 'EXPLAIN' command suggested the implementation of secondary indexes. Secondary indexes allow for the copying and sorting of data from rows, based on indexed columns, which efficiently access the rows of the database. These adjustments may be relevant in future research work.

Lastly, the MariaDB Xpand database did not perform as expected. Although the evaluation using OSSpal showed that this database has the best features, the experimental results with the execution of different query instructions using SF = 1 and SF = 10 showed poor results. At the beginning of the performance evaluation, we were surprised to achieve a fast execution time with the first group of queries. Despite having the features of MySQL and a high capacity to work with complex queries, the remaining results showed no such determination.

On the other hand, as with the majority of studies, the current study's design is subject to threats to validity. The first is the use of OSSpal methodology in business environments. The evaluation of tools by business entities may influence the answers and the assessment results, which could affect the overall validity of the analysis of different tools. Therefore, such results should be handled carefully. The second is the limitation of hardware components, RAM and CPU, to evaluate the three NewSQL databases. With this scarcity of hardware resources, we could not perform a deeper experimental evaluation with Big Data. However, the results presented in Section 5, with 1 GB, and ~10 GB of data, could display a starting point to further research opportunities with NewSQL databases. The third is the lack of studies that use a methodology to assess NewSQL databases that states its qualities and functionalities, despite the number of works referenced in Section 2.

As with any database, there are practical implications with managing this type of databases. The first is the ability to provide the same level of data durability that a disk-based system offers. The other is the cost of expensive hardware, RAM, to operate similarly to disk systems. This second implication is a significant factor for large-scale systems. The third is in-memory data storage capacity. If in-memory databases cannot store large volumes of data, there can be extremely poor performance.

Last but not least, the integration with other systems is a limitation. This type of database may not be compatible with various software or databases, which may limit its use in certain situations. It is essential to understand what a database has and what it offers, which could be reflected in later experiments. However, these results should be interpreted with care, as versions of database tools may vary over time.

In summary, these experiments allowed us to better understand NewSQL databases and how they work with limited hardware resources, as well as their behavior with 1GB and 10 GB data, and how in-memory databases can execute fast queries compared to disk-based systems.

7. Conclusions and Future Work

NewSQL databases are the most recent technology bridging the gap between SQL and NoSQL. Since there are many NewSQL solutions in the market, users have difficulties in choosing the most appropriate database for their needs without drastically changing their application.

In this paper, we analyzed three popular NewSQL databases with the highest scores in the DB-Engines Ranking in the year 2022. Firstly, the OSSpal methodology was used, which showed that the NewSQL database with the best score was MariaDB Xpand, with 4.16 points, showing a higher score in several categories, followed by VoltDB with 4.10 and CockroachDB with 3.99. Using the Star Schema Benchmark, we experimentally evaluated performance and scalability, and we concluded that all databases demonstrated adequate scalability for inserting large volumes of data without much effort, with CockroachDB achieving the best results as it uses a fast bulk data loading. In the performance experiments, VoltDB showed better results in query execution time than CockroachDB and MariaDB Xpand, due to it being an in-memory database. On the other hand, CockroachDB showed lower consumption of CPU and RAM resources. The experiments conclude that with OSSpal methodology, the best database is MariaDB Xpand, although the benchmark performance results were poor. VoltDB had the best execution time performance, and CockroachDB had better scalability results.

As future work, we intend to evaluate more NewSQL databases using the YCSB (Yahoo! Cloud Serving Benchmark) to assess their performance and scalability when facing different workload scenarios, as well as the use of TPC-DS, a Big Data-assisted decision support benchmark, to understand how these databases handle large volumes of data.

Author Contributions: Conceptualization, J.B. and F.S.; Methodology, E.P. and J.B.; Software, E.P.; Validation, E.P., F.S. and J.B.; Formal analysis, E.P., F.S. and J.B.; Investigation, E.P.; Resources, E.P.; Data curation, E.P.; Writing—original draft preparation, E.P.; Writing—review and editing, J.B. and F.S.; Supervision, J.B. and F.S.; Project administration, J.B. and F.S.; Funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grolinger, K.; Higashino, W.A.; Tiwari, A.; Capretz, M.A. Data management in cloud environments: NoSQL and NewSQL data stores. *J. Cloud Comput.* **2013**, *2*, 24. [CrossRef]
2. Date, C.J. *E. F. Codd and Relational Theory*; Technics Publications: Basking Ridge, NJ, USA, 2022; pp. 1–404.
3. Venkatraman, S.; Fahd, K.; Kaspi, S.; Venkatraman, R. SQL versus NoSQL movement with big data analytics. *Int. J. Inf. Technol. Comput. Sci.* **2016**, *8*, 59–66. [CrossRef]
4. Sharma, N. Overview of the Database Management System. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 362–369.
5. Abramova, V.; Bernardino, J.; Furtado, P. Experimental Evaluation of NoSQL Databases. *Int. J. Database Manag. Syst.* **2014**, *6*, 1–16. [CrossRef]
6. Chaudhry, N.; Yousaf, M.M. Architectural assessment of NoSQL and NewSQL systems. *Distrib. Parallel Databases* **2020**, *38*, 881–926. [CrossRef]
7. Stonebraker, M. Newsql: An alternative to nosql and old sql for new oltp apps. *Commun. ACM* **2012**, 6–7.
8. Valduriez, P.; Jiménez-Peris, R.; Özsu, M.T. Distributed database systems: The case for NewSQL. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVIII*; Hameurlain, A., Tjoa, A.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12670, pp. 1–15.
9. Matthew, A. How Will the Database Incumbents Respond to NoSQL and NewSQL. Available online: https://15799.courses.cs.cmu.edu/fall2013/static/papers/aslett-newsql.pdf (accessed on 26 July 2022).
10. DB-Engines Ranking. Available online: https://db-engines.com/en/ranking (accessed on 13 September 2022).

11. Cockroach Labs, The Company Building CockroachDB. Available online: https://www.cockroachlabs.com/docs/releases/v21.2.html (accessed on 22 December 2022).
12. MariaDB Xpand: Distributed SQL Database. Available online: https://mariadb.com/docs/xpand/release-notes/mariadb-xpand-6/6-0-3/#Installation_Instructions (accessed on 22 December 2022).
13. Volt Active Data: Because Milliseconds Matter. Available online: https://docs.voltactivedata.com/v11docs/UsingVoltDB/ (accessed on 22 December 2022).
14. Wasserman, A.I.; Guo, X.; McMillian, B.; Qian, K.; Wei, M.Y.; Xu, Q. OSSpal: Finding and evaluating open source software. In Proceedings of the IFIP International Conference on Open Source Systems, Buenos Aires, Argentina, 22–23 May 2017; Volume 496, pp. 193–203.
15. Calçada, A.; Bernardino, J. Evaluation of Couchbase, CouchDB and MongoDB using OSSpal. In *KDIR*; SCITEPRESS—Science and Technology Publications: Setúbal, Portugal, 2019; pp. 427–433.
16. Leite, N.; Pedrosa, I.; Bernardino, J. Open Source Business Intelligence Platforms Assessment using OSSpal Methodology. In Proceedings of the 15th International Joint Conference on e-Business and Telecommunications (ICETE), Porto, Portugal, 26–28 July 2018; pp. 356–362.
17. António, O.; Jorge, B. OSSPal Assessment of Self-Service BI and Analytics Software. In Proceedings of the CAPSI 2020, Porto, Portugal, 11–12 October 2020; p. 23. Available online: https://aisel.aisnet.org/capsi2020/23 (accessed on 22 December 2022).
18. Ferreira, T.; Pedrosa, I.; Bernardino, J. Integration of Business Intelligence with e-commerce. In Proceedings of the 14th Iberian Conference on Information Systems and Technologies (CISTI), Coimbra, Portugal, 19–22 June 2019; pp. 1–7. [CrossRef]
19. Cardoso, T.; Penela, J.; Rosa, A.; Wanzeller, C.; Martins, P.; Abbasi, M. OSSpal Qualitative and Quantitative Comparison: Couchbase, CouchDB, and MongoDB. In *Marketing and Smart Technologies*; Springer: Singapore, 2022; pp. 141–150.
20. O'Neil, P.; O'Neil, E.; Chen, X.; Revilak, S. The star schema benchmark and augmented fact table indexing. In *TCPEB*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 237–252.
21. Astrova, I.; Koschel, A.; Wellermann, N.; Klostermeyer, P. Performance Benchmarking of NewSQL Databases with Yahoo Cloud Serving Benchmark. In Proceedings of the Future Technologies Conference, Virtual, 5–6 November 2020; pp. 271–281.
22. Kaur, K.; Sachdeva, M. Performance evaluation of NewSQL databases. In *ICISC*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–5.
23. Murazzo, M.; Gómez, P.; Rodríguez, N.; Medel, D. Database NewSQL Performance Evaluation for Big Data in the Public Cloud. In *Cloud Computing and Big Data*; Naiouf, M., Chichizola, F., Rucci, E., Eds.; JCC&BD 2019. Communications in Computer and Information Science; Springer: Cham, Switzerland, 2019; Volume 1050. [CrossRef]
24. Oliveira, J.; Bernardino, J. NewSQL Databases—MemSQL and VoltDB Experimental Evaluation. In Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management—KEOD, Madeira, Portugal, 1–3 October 2017; pp. 276–281, ISBN 978-989-758-272-1. [CrossRef]
25. Knob, R.; Schreiner, G.; Frozza, A.; Mello, R. Uma Análise de Soluções NewSQL. In *Anais da XV Escola Regional de Banco de Dados*; SBC: Porto Alegre, Brazil, 2019; pp. 21–30. [CrossRef]
26. Hahn, S.M.L.; Chereja, I.; Matei, O. Comparison of the Performance of NewSQL Databases Based on Linux OS. In *Data Science and Intelligent Systems. CoMeSySo 2021*; Silhavy, R., Silhavy, P., Prokopova, Z., Eds.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2021. [CrossRef]
27. Sanchez, J. A review of star schema benchmark. *arXiv* **2016**, arXiv:1606.00295.
28. Taft, R.; Sharif, I.; Matei, A.; VanBenschoten, N.; Lewis, J.; Grieger, T.; Niemi, K.; Woods, A.; Birzin, A.; Poss, R.; et al. CockroachDB: The resilient geo-distributed sql database. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 14–19 June 2020; pp. 1493–1509.
29. The New Stack: Meet CockroachDB, the Resilient SQL Database. Available online: https://www.cockroachlabs.com/blog/the-new-stack-meet-cockroachdb-the-resilient-sql-database/ (accessed on 19 July 2022).
30. Google Spanner Inspires CockroachDB to Outrun It. Available online: https://www.nextplatform.com/2017/02/22/google-spanner-inspires-cockroachdb-outrun/ (accessed on 30 August 2022).
31. Start a Local Cluster (Insecure). Available online: https://www.cockroachlabs.com/docs/stable/start-a-local-cluster.html (accessed on 30 August 2022).
32. Known Limitations in CockroachDB v22.1. Available online: https://www.cockroachlabs.com/docs/stable/known-limitations.html#a-multi-region-table-cannot-be-restored-into-a-non-multi-region-table (accessed on 9 October 2022).
33. MariaDB Corporation. Evaluating MariaDB Xpand and Cockroach with Sysbench. [White Paper] March. 2022. Available online: https://go.mariadb.com/22Q2-WC-GLBL-DBaaS-Xpand-vs-CockroachDB-with-Sysbench-DB1139_LP-Registration.html (accessed on 22 December 2022).
34. Deploy Xpand Performance Topology—MariaDB. Available online: https://mariadb.com/docs/deploy/topologies/xpand-performance/xpand-6/ (accessed on 19 July 2022).
35. Deploy Xpand Topology—Enterprise Documentation—MariaDB. Available online: https://mariadb.com/docs/deploy/topologies/xpand/xpand-6/ (accessed on 19 July 2022).
36. Architecture of MariaDB Xpand. Available online: https://mariadb.com/docs/architecture/components/xpand/ (accessed on 16 July 2022).
37. MariaDB Xpand Reviews, Ratings & Features 2022—Gartner. Available online: https://www.gartner.com/reviews/market/cloud-database-management-systems/vendor/mariadb/product/mariadb-xpand (accessed on 18 July 2022).

38. Stonebraker, M.; Weisberg, A. The VoltDB Main Memory DBMS. *IEEE Data Eng. Bull.* **2013**, *36*, 21–27.
39. ODBMS, VoltDB Technical Overview. Available online: http://www.odbms.org/wp-content/uploads/2013/11/VoltDBTechnicalOverview.pdf (accessed on 19 July 2022).
40. Using the VoltDB Deployment Manager Web Interface. Available online: https://docs.voltdb.com/v7docs/AdminGuide/DeploymentWebUI.php (accessed on 20 July 2022).
41. Almassabi, A.; Bawazeer, O.; Adam, S. Top NewSQL databases and features classification. *Int. J. Database Manag. Syst.* **2018**, *10*, 11–31. [CrossRef]
42. VoltDB Active Data Documentation Administrator's Guide. Available online: https://docs.voltdb.com/AdminGuide/ (accessed on 10 October 2022).