*Journal of Computer Science & Computational Mathematics*, *Volume 11, Issue 4, December 2021*
DOI: 10.20967/jcscm.2021.04.002

# A Survey of NewSQL DBMSs focusing on Taxonomy, Comparison and Open Issues

Abdullah Muhammed[2], Zul Hilmi Abdullah[1,4], Waidah Ismail[1,3,5,*], Ali Y. Aldailamy [2], Abduljalil Radman[1], Rimuljo Hendradi[3], Radhi Rafiee Afandi[1]

[1]Faculty of Science and Technology, Universiti Sains Islam Malaysia, Nilai, Negeri Sembilan, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, (UPM), Serdang, Selangor, Malaysia
[3]Information System Study Program, Universitas Airlangga, Indonesia Kampus C, Surabaya, Indonesia
[4]Faculty of Information Technology, INTI International University, Nilai, Malaysia
[5]International Halal and Fatwa Center, Universiti Sains Islam Malaysia, Nilai, Negeri Sembilan, Malaysia
[*]Corresponding author email: waidah@usim.edu.my

***Abstract*:** Advancements in internet, cloud, and business intelligence technologies, as well as the emergence of big data, have caused massive traffic in online transaction processing (OLTP) systems. All these impose the needs for effective and efficient data storage and processing which is not available in conventional Relational Database Management Systems (RDBMSs). In the year 2000, a new generation of database management systems (DBMSs) called Not Only Structured Query Language (NoSQL) has emerged to address the scalability of OLTP workloads in a way that was impossible for conventional relational database management systems (RDBMSs). Despite adequately addressing issues linked to scalability and producing better read-write performance than RDBMS, NoSQL DBMSs appear to lack in providing definite assurance of data consistency. Hence, a newer DBMS alternative called NewSQL has emerged. NewSQL DBMS has combined the advantages of both conventional RDBMS and NoSQL. They have the scalable performance of NoSQL DBMSs and the data consistency of traditional RDBMS. To date, there are tens of existing NewSQL-DBMSs, so it is difficult to understand the difference between them, and it is also quite challenging to decide the best solution for a specific task. Hence, this paper presents a comprehensive review concerning NewSQL-DBMSs by emphasizing the following purposes: (1) providing researchers and practitioners guidance that may assist in selecting the most appropriate NewSQL-DBMS, (2) classifying NewSQL-DBMSs based on internal implantation and number of tiers, (3) providing comparisons and analyses of query capabilities, technical characteristics, and security attributes of the prominent NewSQL-DBMSs, and finally, (4) identifying issues and challenges raised with the emergence of NewSQL-DBMSs. In conclusion, NewSQL DBMSs can offers solutions for fault tolerance, horizontal scalability features.

***Keywords*:** DBMS, RDBMS, NoSQL, NewSQL, ACID, BASE.

## 1. Introduction

The continuous development of the internet and cloud, along with the rise of Big Data, has rapidly contributed to the growth of databases sizes. Hence, large-scale data computing power has turned into an essential factor to manage data in massive amounts successfully. To date, a range of Database Management Systems (DBMSs) is equipped with a variety of characteristics. The initial and widely used database solution refers to Relational Database Management Systems (RDBMSs) [1]. They are designed to store and manage data, apart from providing reliable performance in transaction processing with support for data integrity. Nevertheless, RDBMSs are not designed to scale out over multiple servers. Instead, they are intended to operate in a single machine to avoid distributed computing issues and to preserve table mappings integrity [2]. Hence, a new type of DBMSs called Not Only SQL (NoSQL) had been introduced. NoSQL DBMSs are equipped with functions that are not found in the relational database [3]. Nonetheless, NoSQL has missed several features found in RDBMSs, such as immediate consistency and unified interface.

Brewer [4] assessed the principles of these chronological changes in DBMSs, and evaluated the effects of atomicity, consistency, isolation, durability (ACID) and basically available, soft state, eventual consistency (BASE) properties in the architecture and design of DBMSs, hence introducing the CAP (Consistency, Availability, and Partition tolerance) theorem. This theorem is farther elaborated by Gilbert and Lynch [5], which introduced the three concepts and the possible causes of the theory. Based on the CAP theorem and as illustrated in Fig 1, the distributed DBMS has three properties, namely: availability, consistency, and network partition tolerance, with sharing, at most, two of the three properties. However, for distributed DBMSs, it is not feasible to forfeit partition tolerance, thus the need to choose between availability (BASE) and consistency (ACID). The CAP theorem has been under investigation and development as its concept has been misunderstood [6].

Note: CP (Consistency and Partition); CA (Consistency and Availability); PA (Partition and Availability) Based on the CAP Theorem in Figure 1, the system can be consistent and available at the same time, but the problem raises upon network failure. The trade-off between database consistency and availability is required with the presence of network partition [7]. The database can be kept consistent, but partitions that do not receive the latest updates should be kept unavailable until they receive such updates, and until the synchronisation process is complete. Availability may be selected, but it requires forfeiting the consistency for updates that oc-

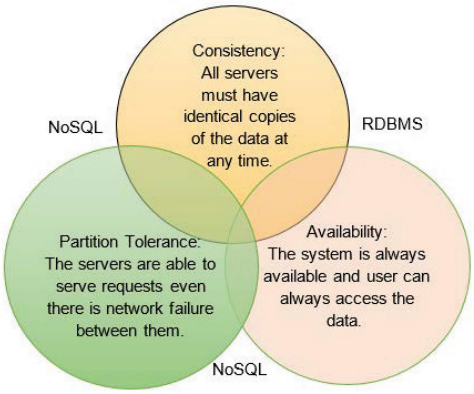


**Figure 1.** CAP Theorem

cur during a network partition. Hence, the trade-off between consistency and availability in distributed DBMSs is always triggered whenever network partition is present. At present, the ideal DBMS should have the ability to scale horizontally or vertically and provide a high level of consistency and availability. This was the reason behind the emergence of a new DBMSs called NewSQL that share many functions of conventional RDBMSs, while offering some benefits of NoSQL technologies. NewSQL is a new class of DBMS technology that aims to produce a combination of the best features of both relational and NoSQL DBMSs. Again, the ACID properties are preserved in this NewSQL DBMSs. They are RDBMSs with capabilities to support SQL query, along with higher complexity and scalability properties of NoSQL [8]. They are used for OLTP web transaction and cloud. In terms of providing efficient performance, most of the NewSQL DBMSs use the principle of the in-memory database to display high performance by keeping all data in the main memory [9].

## 2. The Emergence of NewSQLs

NewSQL is used to distinguish a group of distributed DBMSs in order to have fault tolerance and horizontal scalability features of NoSQL DBMSs, while maintaining the relational model and strong consistency of RDBMSs. NewSQL was introduced by Matthew Aslett in a business report involving 451 groups in the year 2011 [8]. NewSQL DBMSs address the existing issues of OLTP systems. In fact, they have been designed to fully support OLTP systems by providing scalability and the required performance features [9]. They are suitable for applications with read-write transactions, as well as characteristics that include: (1) transactions of short execution times (e.g., INSERT, UPDATE, DELETE), (2) access of a very small amount of data of the entire database using index lookups, and (3) since the transactions are repetitive, they can be re-executed using various inputs [10]. The NewSQL is the most suitable DBMSs for OLTP read-write workload. These new types of DBMSs allow applications to compute a massive number of concurrent transactions that inserts new data and modifies the state of the database. Kumar et al. [11] and [12] asserted that NewSQL offers a more rapid performance than conventional OLTP RDBMSs by approximately 40-50 times. The most interesting feature that distinguishes NewSQL DBMSs from NoSQL DBMSs is that they provide automatic consistency that saves time and effort amidst developers in writing codes that guarantee strong consistency of data in NoSQL DBMSs. Stonebraker [13], [14] , and [15] and Kumar et al. [11] argued for another definition, as NewSQL is a DBMS that has five characteristics, including (1) preserving ACID properties for transactions, (2) lock-free mechanism that of concurrency control, (3) a shared-nothing architecture, (4) SQL as a primary interface for interaction, and finally, (5) architecture that provides higher per-node performance than the conventional single server. As data are partitioned and distributed among a set of machines, each machine has independent performance and access, including independent responsibility for its portion of data. The comparison shown in the Appendix 1. This survey focuses on NewSQL DBMSs solutions. It presents the classification of NewSQL DBMSs based on their architectures and categories. It provides detailed analysis and comparisons of the most important characteristics in almost all available NewSQL DBMSs. The comparisons are presented in the form of tables to ease developers in selecting the most suitable NewSQL DBMS for their tasks. Past studies, such as [16] and [17], compared and evaluated the performance of some NewSQL DBMSs. Study [18] compared and evaluated the performance of three DBMSs, each from different class of DBMSs; NewSQL (VoltDB), NoSQL (MongoDB) and conventional DBMS (MySQL) for sensor readings. Whereas, other works such as [19], [20] and [21] presented a comparison between conventional DBMS, NoSQL, and NewSQL in terms of some general characteristics such as schema, scalability, OLTP support, etc. for handling big data. Other researchers, such as [22] and [23], provided comparisons for the most widely used NewSQL DBMSs, but the compared characteristics were inadequate to cover all essential aspects provided in NewSQL DBMSs. This paper presents a review of the most important characteristics of the security required in any DBMS.

## 3. NewSQL Taxonomy

This section will focuses on the Taxonomy of NewSQL DBMSs that consists of the NewSQL categories and architectures as shown in Figure 2.

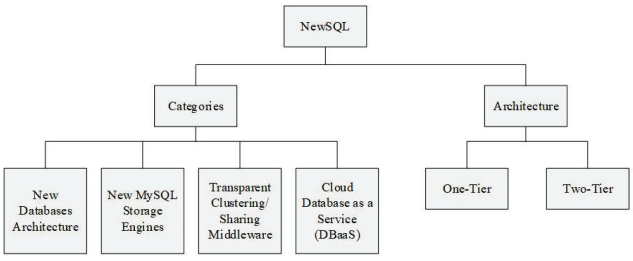


**Figure 2.** Taxonomy of NewSQL DBMSs categories and architectures

### 3.1 Categories of NewSQL

Although NewSQL DBMSs use SQL as their main interface and provide support for relational concepts (e.g., tables and relationships), their actual implementation and internal representation may absolutely differ from that of conventional RDBMSs. Some authors, such as [20] and [24], classified NewSQL into three categories, including New Databases Architecture, New MySQL Storage Engines, and Transparent Sharding Middleware. Whereas, Awasthi et al. [25] categorized then into two categories only which contain New Databases Architecture, New MySQL Storage Engines. Pavlo et al. [26] categorised NewSQL DBMSs into three categories but substituted New MySQL Storage Engines with Cloud Database as a Service (DBaaS). They disregarded New MySQL Storage Engines as NewSQL category. They omitted this category because "MySQL storage engine replacement business for OLTP workloads is the graveyard of failed database projects." Regardless of being the reason for failed database projects for OLTP systems, they were classified as NewSQL DBMSs because they share similar characteristics of NewSQL DBMSs. In this paper, NewSQL DBMSs are classified based on their internal representation into four categories: New Databases Architecture, New MySQL Storage Engines, Transparent Clustering/Sharding Middleware, and Cloud DBaaS which will be discussed in section 3.1.1, 3.1.2, 3.1.3, and 3.1.4 respectively.

#### 3.1.1 New Databases Architecture

This category of NewSQL DBMSs has a completely new architecture. They are designed from the beginning to support scalable performance, instead of modifying or extending the existing architectures. In precise databases designers coded this type of database from scratch to provide the best features of NoSQL DBMS and conventional RDBMS. They were built based on completely new distributed stores designed to work on multiple nodes. These DBMSs were designed in accordance to the distributed architectures that provide several features, such as fault tolerance and high availability using replication, as well as a lock-free concurrency control that allows executing a large number of transactions in multiple nodes. They also operate on shared-nothing resources that support distributed processing of queries.One advantage that differ the main memory NewSQL solutions from others is that they can support databases larger than the amount of memory available in the system. This advantage allows the system to continue working in the main memory mode without changing to disk-oriented mode. This mechanism is implemented by unloading a subset of data to the disk when the size of the database is bigger than the memory size. The most famous instances of this NewSQL category include Google Spanner [27], VoltDB [28], MemSQL [29], NuoDB[30], Clustrix [31], CockroachDB [32], H-Store [33], HyPer [34], SAP HANA[35], Drizzle [36], and Altibase [37].

#### 3.1.2 New MySQL Storage Engines

With the advent of new requirements for new OLTP application, such as high scalability and transactions concurrency, a set of highly optimised storage engines for MySQL is developed to meet the requirements. These storage engines are designed to tackle issues related to scalability of MySQL built-in storage engines (InnoDB is the default built-in storage engine of MySQL). These New MySQL storage engines improve MySQL read-write OLTP workloads through the mechanism of distribution, as well as read-intensive Online Analytical Processing (OLAP) applications. Moniruzzaman [20] reported that the main advantage of these storage engines is that they provide a similar SQL interface with the capability to scale better than MySQL built-in storage engines. They are redesigned to provide high availability through the use of replication, high throughput by utilizing computing power the distributed machines, and low latency while providing the ability to scale out.

#### 3.1.3 Transparent Clustering/Sharding Middleware

These DBMSs make the database appears to the application and users as a centralised unit. Nevertheless, it is split into multiple portions and distributed among multiple machines. These systems still utilise original OLTP systems, while providing the ability to scale out by dividing the database into smaller chunks. The two methods of implementing this approach are Transparent Clustering and Transparent Sharding Middleware [24]. The greatest advantage of this category of NewSQL DBMSs is that with the use of any of the above approaches, it usually allows the reuse of the existing ecosystem and skillsets, in order to avoid the need to rewrite the code or perform any data migration. Most of them use open source databases and work well on commodity multicore servers to save up a great percentage of the cost. This paper presents the surveyed characteristics of the most prominent four NewSQL DBMSs, which are AgilData [38], MariaDB MaxScale [39], ScaleArc [40], and Continuent Tungsten [41].

#### 3.1.4 Cloud Database as a Service

With the advancement in the cloud computing industries and the rapid development of services provided by the cloud, therefore, the database is now provided as a service by most cloud providers, wherein this service is called Database as a Service (DBaaS). Lehner et al. [42] have reported that DBaaS is a cloud service that provides users with the same functionality as any proprietary databases that operate on their own dedicated servers. It allows users to access an automatically-managed database. Many cloud storage providers offer DBaaS, but most of them offer a managed instance of a conventional, single-node RDBMS, (e.g., MySQL, Oracle) or managed instances of distributed NoSQL DBMSs (e.g., MongoDB, Cassendra): famous examples include Microsoft Azure SQL, Google Cloud SQL, Rackspace Cloud Database (Supports Redis, MongoDB, Cassandra databases), and ScaleGrid NoSQL (Supports Redis and MongoDB databases). This present survey only considered cloud DBaaS that provides NewSQL solutions. Some famous examples of such providers are Amazon Aurora [43] and ClearDB [44]. Refer to Table 1 that presents a summary of the key advantages and disadvantages of NewSQL categories.

**Table 1.** Differences advantages and disadvantages of NewSQL categories

| Characteristic | RDBMS | NoSQL | NewSQL |
|---|---|---|---|
| DBMS Model | ACID | BASE | ACID |
| OLTP | Not fully supported | Supported | Fully supported |
| Data analysis (OLAP) | Supported | Supported | Supported |
| Schema flexibility | No | Yes | No |
| Flexibility of data format | No | Yes | Yes |
| Distributed architecture | No | Yes | Yes |
| Scale out (horizontal)/Scale up (vertical) | No/Yes | Yes/Yes | Yes/Yes |
| Increasing data size effect in performance. | Slow | Fast | Very Fast |
| Overhead in performance | Huge | Moderate | Minimal |
| Popularity | Huge | Growing | Slowly growing |
| Programming interface | SQL | APIs | SQL&APIs |
| Data model | Relational | Column-Oriented, Key-Value, Document, and Graph | Relational, key-value |
| Query complexity | Low | High | Very high |
| Storage | On-disk & Cache | On-disk & Cache | On-disk & Cache / Memory & on-disk |
| Security | High | Low | High |
| Cloud support | Not fully supported | Supported | Fully supported |

**Table 2.** Comparison of the most important characteristics between conventional RDBMS, NoSQL, and NewSQL

| Characteristic | RDBMS | NoSQL | NewSQL |
|---|---|---|---|
| DBMS Model | ACID | BASE | ACID |
| OLTP | Not fully supported | Supported | Fully supported |
| Data analysis (OLAP) | Supported | Supported | Supported |
| Schema flexibility | No | Yes | No |
| Flexibility of data format | No | Yes | Yes |
| Distributed architecture | No | Yes | Yes |
| Scale out (horizontal)/Scale up (vertical) | No/Yes | Yes/Yes | Yes/Yes |
| Increasing data size effect in performance. | Slow | Fast | Very Fast |
| Overhead in performance | Huge | Moderate | Minimal |
| Popularity | Huge | Growing | Slowly growing |
| Programming interface | SQL | APIs | SQL&APIs |
| Data model | Relational | Column-Oriented, Key-Value, Document, and Graph | Relational, key-value |
| Query complexity | Low | High | Very high |
| Storage | On-disk & Cache | On-disk & Cache | On-disk & Cache / Memory & on-disk |
| Security | High | Low | High |
| Cloud support | Not fully supported | Supported | Fully supported |

### 3.2 NewSQL Architecture

The NewSQL DBMSs are designed based on a distributed architecture. It is clear that in the design of NewSQL DBMSs architecture, the most important consideration is scalability. They are designed to preserve SQL-standard and its transactionality of RDBMSs while providing scalability [19]. This section classifies NewSQL DBMSs into one-tier or two-tier architectures based on where computation and storage take place. The system that performs computation and storage in the same node is considered as one-tier architecture, while two-tier DBMS architecture refers to a system that has different nodes for computation and storage. Most NewSQL DBMSs are designed to function in a distributed architecture, but there are differences in the architecture among them [20]. The differences in design and architecture offer potential features in some areas. For example, NuoDB is designed to be deployed and used through a cloud. MemSQL was developed for maximum usefulness in clustered analytics, but its capability of consistent trade-off usually slumps for complete ACID transactions. Some NewSQLs were designed based on one-tier architecture to provide the best performance, but with limited features in scalability. Hence, the trade-off is between scalability and performance. NuoDB has the best scalability, while VoltDB has better performance but poor scalability as it is designed based on the one-tier architecture [45].

### 3.3 Characteristics

Many NewSQL solutions can manage data in an organisation or on the cloud. When an organisation decides to use NewSQL DBMSs, many characteristics must be considered to make the right decision. The main characteristics should be considered are the vital querying capabilities, technical characteristics, and security measurements of NewSQL DBMSs. Figure 3 illustrates the classification of characteristics in NewSQL DBMSs. This section focus more on the technical characteristic of the NewSQL DBMS that consists of partitioning, replication, consistency and latency, main memory, concurrency control, and data compression.

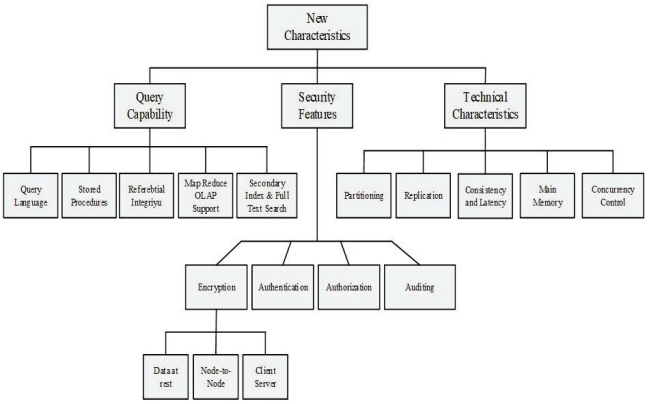


**Figure 3.** Characteristics of NewSQL DBMSs

Also, the comparison of the most important characteristics between conventional RDMBS, NoSQL and NewSQL can be found in Table 2.

#### 3.3.1 Partitioning

Partitioning is a powerful function that allows databases to scale by means of dividing it into partitions (shards), whereby each partition is located in varied machines. The two partitioning methods for this database technology are vertical

partitioning and horizontal partitioning [46]. Vertical partitioning involves splitting table columns among many nodes, while horizontal partitioning involves placing a range of rows into varied tables. Vertical partition is used by NoSQL DBMSs, whereas, horizontal partitioning is deployed by NewSQL DBMSs. The four primary criteria of automatic horizontal partitioning are range, round-robin, hash, and list partitioning [47]. Range partitioning is a technique that creates dedicated partitions for a range of values in a table. The most popular example of range partitioning is the use of dates. Using date ranges to partition tables allow data in a specified period to be stored within the same partition. Creating a list of the partitioned table requires assigning a list of values in the description of the partitioning key of each partition. Shared-nothing systems include independent machines connected by high-speed networks [48]. This technique allows each machine to store a portion of the database locally on its disk. The partitioned database involves distributed transaction processing that distributes the query execution process among multiple partitions. Upon completing the executions, it collects the results together to a single result [49]. All NewSQL-DBMSs have this function, which can partition data and can scale-up to hundreds or even thousands of nodes. However, distributed transaction suffer from increased latency issue as the data is distributed among multiple partition. This latency relies on many factors, such as the number of partitions and the amount of data required to be moved over the network. The recent concern about partitioning is to minimise the amount of data transferred between nodes when the query is placed. Many researchers have proposed partitioning schemas that reduce data transfers between nodes as much as possible. Pavlo et al. [22] proposed horticulture, an automatic database tool, which minimises the number of the partition needed to be accessed per query, as well as the distributed transactions number in the system. Thus, this tool reduces the overhead that results from the coordination between many partitions.

Some NewSQL DBMSs, such as VoltDB and Clustrix, apply consistent hashing partitioning. Some support more than partitioning criteria, for example, SAP HANA [50] and Continuent Tungsten [51], which support hash, range, and round-robin partitioning, apart from giving options for users to choose the type at the time when creating a partition. Using these conventional automatic partitioning methods for workloads consists of a small number of transactions that touch a few records, which are scattered among multiple partitions, data often needs to be transferred across partitions, which is a relatively expensive operation compared to relational DBMSs [52]. Therefore, some NewSQL DBMS, such as MemSQL, NuoDB, and CockroachDB, have their own models of partitioning, aside from supporting some previous partitioning methods. The only NewSQL solution that has a different partitioning model is Google Spanner.

### 3.3.2 Replication

NewSQL DBMSs have the advantages of high availability, high performance, and fault tolerance [53]. All these advantages are gained due to the support of proper replication mechanisms. The guarantee of data availability in case of server failures is the main reason behind the adoption of replication mechanism in distributed DBMSs. The replication technique is usually designed and implemented based on the DBMS system architecture. The system makes identical copies of the same data and stores them in several separate nodes, clusters, or datacenters. Each copy is called replica, so if there is a failure in any replica, other replicas can be used to serve the user requests. In case of any server failure, the server can recover any lost updates to its replica from other replicas, because all the other replicas are synchronised [54]. However, ideal replication technique must take into consideration simultaneously some principles: consistency of replicas, fault-tolerance in case of failure, load balance of replicas' workload, reduction of data movement in-network, less access time, and available resources. However, deploying all these principles in the same replication technique leads to different trade-offs and performance behaviour.

Based on the system architectures, Wiesmann et al. [55] demonstrated that there are two database replication techniques: multi-master and master-slave replications. In both techniques, the consistency of all replicas is maintained using state machine replication. Some of NewSQL DBMSs such as VoltDB, Drizzle, and Altibase provide both techniques, but users have to choose which replication technique before starting the cluster. It is then obvious that the choice of the most suitable replication technique depends on the application and its environments, such as the failure rate, or the available resources.

### 3.3.3 Consistency and Latency

The simplest definition of consistency in NewSQL DBMSs that fulfils the meaning of consistency property in the ACID model is that all users always have a similar view of data at any point of time even they access the same data from different replicas. In precise consistency guarantees that after update request is committed, any subsequent request can view the same updated data. Having multiple replicas of the data distributed in multiple nodes raises many issues. These issues are based on the mechanism of replication used by the NewSQL DBMS. One important issue is that the database takes longer time to update all replicas with the new state, which in turn, increases response time. In a distributed storage system, there is a paradigm regarding the trade-off of data consistency and response latency. This means; the more consistent is a networked data-shared system, is the response time. Some of NewSQL DBMSs has replication layer that uses off-the-shelf components, such as zookeeper [56] and Raft [57] such as in CockroachDB or their custom implementation of Paxos [58] as has been performed in F1 [59] system, in order to provide consistency for the replicated logs between all the servers through the cluster and to ensure the safety rules of Log Matching. It is common for NewSQL DBMS that is AP to have weak consistency that leads to weak ACID guarantees. For example, MemSQL generally has weak ACID due to its ability to work in two modes of

the CAP theorem, which can be configured to work in CP (consistency, partition tolerance) mode or AP (availability, partition tolerance) mode.

### 3.3.4 Main Memory

From starting, all DBMS systems use disk-based storage architecture. The reading and writing operations to such a storage device are slow when compared to main memory speed. Hence, the data are cached in small blocks at the main memory for access purpose. In most of the modern DBMSs, memory is used to cache blocks when reading from disk and to buffer updates from transactions, while preserving a backup copy in the disk. This generates two copies of data; one each for disk and memory. The main difference is that the database of MMDBs resides permanently on the main physical memory [60]. This eviction allows dropping a subset of the data from the memory to empty some space for new elements. Most NewSQL DBMSs apply the heuristic algorithm of memory management. The NuoDB, for instance, uses the recently-least used algorithm. Since these algorithms keep the keys of the released data in the memory, there is no potential saving the memory. Hence, some systems, such as H-Store, apply another technique to address the problem of a database that is larger than that of the available memory [61]. DeBrabant et al. [62] proposed the new technique called 'anti-cache' for H-store DBMS. Upon exhaustion of memory, the anti-cache technique allows moving cold data to the disk safely for transactions. The volatility of RAM is the current potential technical hurdle with in-memory databases. When power loss occurs intentionally or otherwise, volatile RAM losses all data. However, with the advent of non-volatile random access memory or Non-Volatile Memory (NVM) technology, in-memory databases can produce the same performance and maintain data in the case of power loss. Arulraj [63] presented the design and implementation of a new DBMS tailored specifically for NVM. His work concentrated on three aspects of DBMSs: (1) storage management, (2) indexing, and (3) logging and recovery.

### 3.3.5 Concurrency Control

Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems which include lost updates, uncommitted data, and inconsistent retrievals. Thus, Concurrency control is a basic concept that NewSQL databases should support. It is a mechanism used by DBMS to overcome conflicts when many users attempt to access or alter a piece of data at the same time. It is designed and implemented in DBMSs to control the simultaneous access to the database in order to preserve data integrity. Concurrency control provides transparency to each user that his transaction is the only one being processed in the system. As for ACID properties, concurrency control provides atomicity and isolation guarantees in the system. It uses MV2PL to make isolation between reading and write transactions [31]. Most of the NewSQL DBMSs that belong to transparent sharding middleware, such as AgilData, MariaDB, and Tungsten, apply MV2PL. The drawback of multi-version concurrency controls is that every update or delete transaction demands the creation of a new version of the record. This poses non-negligible performance overhead if there are many indexes in the table. That is because; the changes need to be distributed to all indexes. Consequently, many versions will emerge from the record, each corresponding to the state of the record at some point of time. The number of record versions has to be diminished as many as possible. Sadoghi et al. [64] proposed a latch-free optimistic and pessimistic 2-maximum versions concurrency control (2VCC) model to reduce transaction block and lock wait time.

**Table 3.** Technical characteristics of NewSQL DBMSs

| Category | DBMS | Partitioning | Replication | ACID Support | Concurrency control | Main Memory Storage | Compression |
|---|---|---|---|---|---|---|---|
| New Database architecture | Google Spanner | Yes. Different partitioning model. | Multi-master | *Fully-ACID* | MV2PL | No | NA |
| | NuoDB | Yes. Range and List | Master-slave | *Fully-ACID* | MV2PL | Yes | NA |
| | VoltDB | Yes. Consistent hashing | Multi-master Master-slave | *Fully-ACID* semantics | TO | Yes | No |
| | Clustrix | Yes. Consistent hashing | Master-slave | *Fully-ACID* | MV2PL | Yes | NA |
| | CockroachDB | Yes, *List*, *Range*, and Unique | Master-slave | Fully-ACID | MV2PL | No | NA |
| | HyPer | Yes, Range and hash | Master-slave | Fully-ACID | MVOCC | Yes | Yes |
| | H-Store | Yes. Range and hash | Multi-master | Fully-ACID | TO | Yes | NO |
| | MemSQL | Yes. *Hash partitioning* | Master-Slave | Weak-ACID | MVOCC | Yes | Yes |
| | SAP HANA | *Hash*, *range* and *round-robin* | Master-slave | Fully-ACID | MV2PL | Yes | Yes |
| | Drizzle | Yes | Master-master Master-slave | ACID-compliant | MVCC | No | NA |
| | Altibase | Yes List, range and hash. | Master-Slave Multi-master | ACID compliant | MV2PL | Yes | Yes |
| Transparent Sharding Middleware | AgilData | Yes range, round-robin | Master-Slave | ACID-compliant | MV2PL | No | Not studied |
| | MariaDB MaxScale | Yes, list and range | Master-master Master-slave | Fully-ACID | MV2PL | No | Not studied |
| | ScaleArc | Yes, range, list, hash | Master-slave | *ACID*-compliant | Mixed | No | Not studied |
| | Continuent Tungsten | *Yes, Hash, range* and *round-robin partitioning* | Master-slave | Fully-ACID | MV2PL | No | Not studied |
| New MySQL | TokuDB | Yes, range | Master-slave | Fully-ACID | *MVCC* | No | Not studied |
| | InfiniDB | Both vertical and horizontal | Master-slave | *ACID*-compliant | *MVCC* | No | Not studied |
| DBaaS | Amazon Aurora | No | Master-slave | *ACID*-compliant | *MVCC* | No | Not studied |
| | ClearDB | No | Master-slave | fully *ACID* | MV2PL | No | Not studied |

### 3.3.6 Data Compression

Almost all NewSQL DBMSs that are based on new architecture is designed to use main memory as the default storage for the entire database. However, main memory capacities are still limited when compared to HDD and other secondary storage. Therefore, data that resides permanently in the main memory are required to be compressed in order to store data as much as possible data. Compression of the database has many advantages such as reducing the size of the database to be stored entirely in the main memory and enhancing the performance by reducing the amount of data transferred from and to main memory. Table 3 presents a summary of the comparison between all the internal processes and the technical characteristics for all the surveyed NewSQL DBMSs in the Appendix 5. HyperDB presents a data compression by

including the cold records in a data block which maintains a flat structure without pointers. Data block contains all the required information to reconstruct the record again as well as the type of the compression used. By maintaining a flat structure without pointers, based on the hotness and coldness of the data blocks.

## 4. Research Recommendations/Opportunities

Although NewSQL DBMSs offer great capabilities, there are still many issues and challenges in these systems. This section depicts some of these issues and challenges in order to give opportunities for further investigations and researches. Replication: replication of data in database field has been examined and assessed. However, across multiple geographically distributed system or cloud-based solutions, replication needs implementation that is more effective in minimizing data movements between nodes as well as in moving data as fast as possible across the network when needed. Thus further examinations and assessments of which storage model, replication granularity, and data propagation techniques are more effective for geographically distributed system and cloud-based DBMSs. Main Memory: when a disk-based copy of the database is larger than the available main memory, in-memory databases apply the eviction mechanisms to evict a subset of the database out to persistent storage so as to reduce the occupied space in its memory. These mechanisms use internal tracking approach to identify obsolete records for eviction. Even though all records that are not recently accessed are removed from the memory, their corresponding internal tracking and indexed fields are kept in the memory for subsequent tracking. Such internal tracking increases the overhead of memory management and reduces the evicted memory space. Partitioning: NewSQL DBMSs use partitioning to divide huge data into multiple fragments and distribute them among multiple servers. NewSQL DBMSs deploy partitioning to gain high scalability. However, querying partitioned database moves data across nodes in order to serve the query. Moving data among servers leads to network congestion and increases the response time of the query.

## 5. Conclusion

Recently, NewSQL DBMSs have emerged as a complete solution that contribute the continuous growth of data storages and computing needs of OLTP systems. This solution offers fault tolerance and horizontal scalability features of NoSQL DBMSs while maintaining the relational model and durable consistency of the RDBMSs. This paper presents the review of a wide range of modern DBMSs denoted for OLTP applications and management of Big Data, especially NewSQL DBMSs. In precise, this paper presents the review of NewSQL DBMSs solutions with the goal of presenting comparisons and analyses of query and security capabilities and technical characteristics of the prominent NewSQL DBMSs. Thus, they can be used by researchers and developers as a guidance that may assist in selecting the most appropriate NewSQL DBMS. The taxonomy and discussion on querying capabilities, technical characteristics, and security attributes, along with the comparison of available NewSQL solutions, is bound to assist practitioners in selecting the best storage solutions in accordance to their needs. Moreover, this study has identified the challenges and open issues in this field that require more investigation and enhancement

**Acknowledgment**

This work was supported by the International Grant USIM/INT-NEWTON/FST/IHRAM/053000/41616 under Newton-Ungku Omar Fund. This publication only reflects the authors' views.

## References

[1] A. Y. Aldailamy, A. Muhammed, W. Ismail, and A. Radman, "Comparative study for load management of hbase and cassandra distributed databases in big data," *International Journal of Engineering & Technology*, vol. 7, no. 4.31, pp. 375–380, 2018.

[2] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access*, vol. 2, pp. 652–687, 2014.

[3] D. G. Chandra, "Base analysis of nosql database," *Future Generation Computer Systems*, vol. 52, pp. 13–21, 2015.

[4] E. A. Brewer, "Towards robust distributed systems (abstract) proceedings of the nineteenth annual acm symposium on principles of distributed computing," *Portland, Oregon, United States*, vol. 343502, p. 7, 2000.

[5] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[6] E. Brewer, "Cap twelve years later: How the" rules" have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012.

[7] D. Abadi, "Consistency tradeoffs in modern distributed database system design: Cap is only part of the story," *Computer*, vol. 45, no. 2, pp. 37–42, 2012.

[8] M. Aslett, "How will the database incumbents respond to nosql and newsql," 2011.

[9] R. Kumar, N. Gupta, H. Maharwal, S. Charu, and K. Yadav, "Critical analysis of database management using newsql," *International Journal of Computer Science and Mobile Computing*, vol. 3, pp. 434–438, 2014.

[10] S. K. Gajendran, "A survey on nosql databases," *University of Illinois*, 2012.

[11] R. Kumar, N. Gupta, S. Charu, and S. K. Jangir, "Manage big data through newsql," in *National Conference on Innovation in Wireless Communication and Networking Technology–2014, Association with THE INSTITUTION OF ENGINEERS (INDIA)*, 2014.

[12] R. Kumar, B. B. Parashar, S. Gupta, Y. Sharma, and N. Gupta, "Apache hadoop, nosql and newsql solutions of big data," *International Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE)*, vol. 1, no. 6, pp. 28–36, 2014.

[13] M. Stonebraker, "Newsql: An alternative to nosql and old sql for new oltp apps," *Communications of the ACM. Retrieved*, pp. 07–06, 2012.

[14] M. Stonebraker and R. Cattell, "10 rules for scalable performance in'simple operation'datastores," *Communications of the ACM*, vol. 54, no. 6, pp. 72–80, 2011.

[15] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, "The end of an archi-

tectural era: It's time for a complete rewrite," in *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, 2018, pp. 463–489.
[16] K. Kaur and M. Sachdeva, "Performance evaluation of newsql databases," in *2017 International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2017, pp. 1–5.
[17] J. Oliveira and J. Bernardino, "Newsql databases-memsql and voltdb experimental evaluation." in *KEOD*, 2017, pp. 276–281.
[18] H. Fatima and K. Wasnik, "Comparison of sql, nosql and newsql databases for internet of things," in *2016 IEEE Bombay Section Symposium (IBSS)*. IEEE, 2016, pp. 1–6.
[19] S. Binani, A. Gutti, and S. Upadhyay, "Sql vs. nosql vs. newsql-a comparative study," *database*, vol. 6, no. 1, pp. 1–4, 2016.
[20] A. Moniruzzaman, "Newsql: Towards next-generation scalable rdbms for online transaction processing (oltp) for big data management," *arXiv preprint arXiv:1411.7343*, 2014.
[21] D. Aggarwal and R. Sonika, "Emerging technologies for big data processing: Nosql and newsql data stores," *Int J Eng Comput Sci*, vol. 5, pp. 15 598–15 604, 2016.
[22] A. Pavlo, C. Curino, and S. Zdonik, "Skew-aware automatic database partitioning in shared-nothing, parallel oltp systems," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 61–72.
[23] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: Nosql and newsql data stores," *Journal of Cloud Computing: advances, systems and applications*, vol. 2, no. 1, pp. 1–24, 2013.
[24] R. Kumar and S. Charu, "Newsql databases: Scalable rdbms for oltp needs to handle big data," *International Journal of Modern Computer Science (IJMCS)*, vol. 3, pp. 13–17, 2014.
[25] E. Awasthi, S. Agrawal, and R. Pandey, "A survey on nosql and newsql data stores for big data management," *GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES*, vol. 7, no. 5, pp. 269–277, 2018.
[26] A. Pavlo and M. Aslett, "What's really new with newsql?" *ACM Sigmod Record*, vol. 45, no. 2, pp. 45–55, 2016.
[27] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, pp. 1–22, 2013.
[28] M. Simborg, "What is an in-memory database and how does it work," Jun 2021.
[29] Yhblog, "Memsql is the no-limits database powering modern applications and analytical systems."
[30] "Nuodb technical white paper: Manualzz."
[31] ClustrixDB, "Clustrix / clustrixdb relational database: Mariadb>." [Online]. Available: https://clustrix.com.siteindices.com/
[32] C. Labs, "Cockroachdb: Distributed sql," Jan 2021. [Online]. Available: https://www.cockroachlabs.com/product/
[33] L. Ma, J. Arulraj, S. Zhao, A. Pavlo, S. R. Dulloor, M. J. Giardino, J. Parkhurst, J. L. Gardner, K. Doshi, and S. Zdonik, "Larger-than-memory data management on modern storage hardware for in-memory oltp database systems," in *Proceedings of the 12th International Workshop on Data Management on New Hardware*, ser. DaMoN '16, 2016, pp. 9:1–9:7.
[34] A. Kemper and T. Neumann, "Hyper: Hybrid oltp&olap high performance database system," 2010.
[35] I. Mehndiratta, "Sap hana in-memory computing and analytics for smart businesses," Sep 2017.
[36] D. Developers, "Drizzle in launchpad." [Online]. Available: https://launchpad.net/drizzle
[37] A. Admin, "Altibase - enterprise grade open source database." [Online]. Available: https://www.altibase.com/en/index.php?ckattempt=1
[38] AgilData, "Mysql scaling and big data solutions - agildata," Jan 2021. [Online]. Available: http://www.agildata.com/
[39] M. Zaslavskiy, A. Kaluzhniy, T. Berlenko, I. Kinyaev, K. Krinkin, and T. Turenko, "Full automated continuous integration and testing infrastructure for maxscale and mariadb," in *2016 19th Conference of Open Innovations Association (FRUCT)*. IEEE, 2016, pp. 273–278.
[40] Rasteroids, "Database performance and load balancing software: Scalearc." [Online]. Available: http://www.scalarc.com/
[41] P. Michalák and M. Lang, "Full service mysql database management software," Aug 2021. [Online]. Available: https://www.continuent.com/
[42] W. Lehner and K.-U. Sattler, "Database as a service (dbaas)," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. IEEE, 2010, pp. 1216–1217.
[43] Version1, "Amazon aurora: Cutting edge relational database for the cloud," Oct 2020.
[44] clearDB, "Cleardb – the ultra reliable, geo distributed data services platform." [Online]. Available: https://www.cleardb.com/
[45] A. G. Fior, J. A. Meira, E. C. de Almeida, R. G. Coelho, M. D. Del Fabro, and Y. Le Traon, "Under pressure benchmark for ddbms availability," *Journal of Information and Data Management*, vol. 4, no. 3, pp. 266–266, 2013.
[46] J. L. Harrington, *Relational database design and implementation*. Morgan Kaufmann, 2016.
[47] C. Curino, E. P. C. Jones, Y. Zhang, and S. R. Madden, "Schism: a workload-driven approach to database replication and partitioning," 2010.
[48] S.-H. Kim and B.-h. Roh, "Fast detection of distributed global scale network attack symptoms and patterns in high-speed backbone networks," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 2, no. 3, pp. 135–149, 2008.
[49] A. Y. Aldailamy, N. A. W. A. Hamid, and M. Abdulkarem, "Distributed indexing: performance analysis of solr, terrier and katta information retrievals," *Malaysian Journal of Computer Science*, pp. 87–104, 2018.
[50] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees, "The sap hana database-an architecture overview." *IEEE Data Eng. Bull.*, vol. 35, no. 1, pp. 28–33, 2012.
[51] vmware, "Tungsten replicator 4.0 manual," VMWare Inc, Tech. Rep. [Online]. Available: https://www.vmware.com/pdf/tungsten-replicator-4.0.pdf
[52] R. Nehme and N. Bruno, "Automated partitioning design in parallel database systems," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 1137–1148.
[53] B. Kemme, R. Jiménez-Peris, and M. P. Martınez, "Database replication (synthesis lectures on data management 7)," 2010.
[54] Y. Liu and V. Vlassov, "Replication in distributed storage systems: State of the art, possible directions, and open issues," in *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge*

*Discovery*. IEEE, 2013, pp. 225–232.
[55] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in *Proceedings 20th IEEE International Conference on Distributed Computing Systems*. IEEE, 2000, pp. 464–474.
[56] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems." in *USENIX annual technical conference*, vol. 8, no. 9, 2010.
[57] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.
[58] L. Lamport, "The part-time parliament," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 277–317.
[59] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner *et al.*, "F1: A distributed sql database that scales," 2013.
[60] H. Garcia-Molina and K. Salem, "Main memory database systems: An overview," *IEEE Transactions on knowledge and data engineering*, vol. 4, no. 6, pp. 509–516, 1992.
[61] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. Jones, S. Madden, M. Stonebraker, Y. Zhang *et al.*, "H-store: a high-performance, distributed main memory transaction processing system," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1496–1499, 2008.
[62] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. Zdonik, "Anti-caching: A new approach to database management system architecture," *Proceedings of the VLDB Endowment*, vol. 6, no. 14, pp. 1942–1953, 2013.
[63] J. Arulraj and A. Pavlo, "How to build a non-volatile memory database management system," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1753–1758.
[64] M. Sadoghi, M. Canim, B. Bhattacharjee, F. Nagel, and K. A. Ross, "Reducing database locking contention through multi-version concurrency," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1331–1342, 2014.