2020 11th International Conference on Information and Communication Systems (ICICS) 978-1-7281-6227-0/20/$31.00 ©2020 IEEE 10.1109/ICICS49469.2020.239513

SQL, NewSQL, and NOSQL Databases: A Comparative Survey

Tariq N. Khasawneh
Computer science
Princess Sumaya University for Technology
Amman, Jordan
Tariq.Khasawneh.93@gmail.com

Mahmoud H. AL-Sahlee
Computer science
Princess Sumaya University for Technology
Amman, Jordan
mah20198003@std.psut.edu.jo

Ali A. Safia
Computer science
Princess Sumaya University for Technology
Amman, Jordan
Alisafia955@gmail.com

***Abstract*—Non-relational databases have proved to be a viable alternative to the traditional relational database management systems, especially with the ever-increasing volume of data. Also known as "NoSQL" database systems, a scalable management system enabling handling such volumes of data. This work aims at providing an overview of the different NoSQL management systems, classifying them into four main categories: key-value stores, column-oriented, document-oriented, and graph oriented, comparing each of the categories using multiple criteria including the CAP theorem and BASE properties.**

***Keywords*—RDBMS, NewSQL, NoSQL, CAP Theorem, BASE properties.**

I. INTRODUCTION

Data Storage has been and still is one of the main challenges that humankind has faced through its existence recording events, experiences, obtained knowledge has been a primary concern of our species since the dawn of time. Whether it's through cave sculptures, or modern on the cloud storage. Many technologies have been developed to keep track of our species' history such as carving on stones, writing on leather, paper, etc. with the exponential growth of technological solutions, storage techniques have dramatically been improved. In this study we focus on the modern means of recording our history, i.e. database systems.

Generally, a databases system can be defined as a where the data is stored to be accessed whenever needed. Databases has been found as a replacement for the file system in order to overcome some limitations such as redundancy, inconsistency, and concurrent access by multiple users and many more.

Databases can be classified into two main categories based on the connections and relations among the data being stored as following:

1. SQL: the SQL databases are also called Relational databases, it makes use of the SQL queries to analyze, and get the data stored. It is known as relational database because the data stored is highly relational, i.e. it is connected through a predefined schema presenting the relations between the data.
2. NoSQL databases: short for not only SQL, also known as non-relational database or distributed databases, it's a database management system, which doesn't require the data to exhibit high relations among itself, or a predefined schema, it allows the storage of a variety of data types. It can be categorized into four main categories:
    a. Key-value stores.
    b. Column-oriented.
    c. Document-oriented.
    d. Graph-oriented.

The remainder of this paper is organized as follows: The second section provides general definitions of the relational databases, NewSQL and Non-relational databases, stating some examples of each technology, and their pros and cons for them. The third section discusses the main considerations when designing a database management system. In the fourth section, an overview of the criteria for assessing the performance of NoSQL databases is provided. The fifth section make use of the metrics introduced in section four to draw a comparison between each of the NoSQL technologies. Finally, the work is concluded in section 6, suggesting potential future work.

II. RDBMS, NewSQL, and NoSQL:

II.1 RDBMS:

Relational Database Management Systems are database systems with a predefined fixed schema for the data, the data is organized in a table wise manner with high vertical scalability. The relational databases are well suited for applications requiring complex

querying, and can be considered as a way to overcome the Atomicity, Consistency, Isolation, and Durability(ACID) challenges, the RDBMs supports only one data type allowing small variations to the data.

Examples of RDBMS include Oracle, Postgres, MS-SQL, etc.

II.1.1 Examples of RDMS

1-Oracle RDBMS:

It has an open source environment with some limited features, and the enterprise version providing full functionality. It can be described as a fully comprehensive relational database management system that provides a platform to deploy databases with a simple interface (Oracle Call Interface) with support for multiple programming languages. It also provides big data platform that supports non-relational data. [1].

2-Postgres RDBMS:

An open source object RDBMS that can work as a distributed database. It is scalable in dealing with large data volume (which means that it could be extended as needed). Additionally, it can store complex data. It uses the SQL language and extends it. [2]

3-MS-SQL:

It is not an open source RDBMS that works on windows and Linux operating systems. It has an enterprise platform that consists of two engines one for processing commands and queries called relational engine and the other used for storing, managing, and indexing database files which is called the storage engine. It lacks some functionalities such as partitioning, some optimization and indexing techniques. [3] [4]

II.1.2 Limitations of RDBMS [5]:

1- Scalability: Most SQL oriented databases don't support distributed data processing and storage, making it a challenge to work with data of high volume, therefore, resulting in a need for a server with exceptional computing capabilities to handle the large data volume, which an expensive undesirable solution.

2- Flexibility and complexity: The need for a predefined structure of the data, i.e. explicitly defining a schema for the data being stored is considered one of the main reasons for migrating toward schema-less non-relational databases which allows data storage with no predefined schema.

II.2 NewSQL:

NewSQL systems can be considered as a modification to the traditional SQL systems where it is found to tackle some of SQL shortcomings, as it runs on several nodes on many datacenters giving the authority of data management to the local system.

NewSQL is not only used to overcome some limitations of the SQL, but it could also be an alternative for NoSQL in certain applications where the need for analytics and decision making have to be found on-request with high consistency.

II.2.1 Some examples of NewSQL are:

NuoDB: It is a distributed DBMS that is that could be described as a low latency database. It allows the user to interact with it in a transactional way by supporting ACID operations, and data replication. [6] [7]

VoltDB: It is an in-memory distributed database that is considerably faster than SQL databases. it is a flexible database that supports JSON storage. It is best suited for applications with frequent read and low writing frequency [8].

Clustrix: it is a distributed database that support real-time analytics, it is optimized for massive transactions. It supports some business intelligence tools and fast data retrieval. [6]

II.2.2 Advantages and limitations: [9]

The main advantages of NewSQL are:

1. It enhances data consistency.
2. Similar to SQL queries.
3. Could use SQL extensions.
4. It provides NoSQL clustering style but uses familiar SQL modeling and queries.

However, NewSQL has some limitations, for example [9]:

1. It's not appropriate in applications with higher volume than few terabytes.
2. It does not support full access to the traditional SQL tools.

II.3 NoSQL:

NoSQL is a non-relational database management system that has a flexible schema. It is used with distributed data, especially with applications involving big data, and applications with real time data such as web logs and applications.

NoSQL databases support different types of data such as structured, semi structured, and unstructured data. It is designed to cover some of the drawbacks of relational database management system, specifically when dealing with big data of high variability. NoSQL can be described as a horizontally scalable database that is suited for applications with hierarchical data stores.

NoSQL has a set of features, including [10] [11] [12]:

1- It can run on low specs devises.
2- Does not require high speed internet.
3- Easy to use.
4- High performance.
5- Flexible and elastic scaling tool.
6- Low complexity and latency.
7- Fault tolerance and high availability.

NoSQL is primarily suited for applications with [10] [11] [12]:

1- No need for ACID.
2- Flexible schema.
3- No constraints and validations to be executed in database.
4- Temporary data.
5- Different data types.
6- High data volume.

III. CONSIDERATIONS WHEN BUILDING A NoSQL SOLUTION

When building a NoSQL database system, three main criteria need to be considered

III.1 CAP Theorem:

It stands for Consistency, Availability, and Partition tolerant, three criterions to be considered when implementing a non-relational database solution. it has been demonstrated that t no distributed system architecture meets more than two of these criteria [13].

a. Consistency: this property implies the data updates needs to be broadcasted across data nodes (shards), such that all users can retrieve and query the modified data.
b. Availability: The need for at least one available serving node in case of failure.
c. Partition tolerance: the system will stay up even if there is a network failure or the connection was cut off between the nodes.

III.2 BASE properties:

It is a database model that might be considered as an alternative of the ACID (Atomicity, Availability, Isolation, Durability) model, It's not appropriate for all cases, but it can provide a flexible mechanism for the database model, this approach stands for Basic Availability, Soft state, and Eventually consistency [14].

a. Basic Availability: the model that supports BASE availability tries to drop the consistency to provide an extreme state of availability by allowing partial failure instead of the whole system failure. The main concern in this property is to provide a fault tolerant solution for the replicated databases.
b. Soft state: in some cases, the data can be volatile, stored and recovered easily and can be regenerated, in case the existence of the failure or for the eventual consistency.
c. Eventual consistency: It is related to the synchronization process, i.e. it allows having a state where the nodes are in an inconsistent state. However, after a certain time, there is a guarantee based on certain rules that all nodes would return to a consistent state.

IV. COMPARISON BETWEEN THE NoSQL MODELS

IV.1 Comparison Criteria:

This section is concerned with giving an overview about the metrics commonly used to compare the different NoSQL models. [14]

IV.1.1 Persistence:

It represents the ability to store the data in a permanent state, it doesn't mean that the data can't be removed, but in a state that can't be lost, this is achieved through several techniques including storing the data on non-volatile hardware, databases, distributed file systems, etc...

IV.1.2 Replication:

This metric is an approach to ensure satisfying the availability feature in a certain database, as it stores several replicas of the data on several nodes to guarantee data availability, which provides a backup plane in case of failure, however, the presence of these replicas with the ability to modify (read) some without editing the others could lead to inconsistency problem.

IV.1.3 Sharding:

Sharding refers to the ability of the database to divide the data into several nodes enabling fast analysis and processing, this feature is not available in all databases, because in this case, the users have

to deal with data partition manually, or wait a considerable amount of time until the system processes all the data.

IV.1.4 Consistency:

The NoSQL databases allows multi access for writing and reading of the data at the same time, this ability of having the same data among all replicas and for all users is the consistency metric which has been covered in the ACID operations for the SQL databases, and the eventual consistency for the NoSQL.

IV.1.5 Query method:

Each database employs a different technique for data querying, such as MapReduce, get, and Java based queries.

IV.1.6 Implementation language:

Different databases are implemented using different programming languages, and support API for additional languages.

IV.2 Comparing Key-value databases:

General definition:

Also known as Key-Value stores, Key-Value database is often considered the simplest type of NoSQL databases. It implements a hash table for storing unique keys pointing to their corresponding data values. It is also considered the a highly flexible NoSQL database alternative to the traditional relational databases. It is characterized by its simple schema, high read/write velocity without frequent updates, and high performance and scalability.

Some of the popular key-value databases include:

1- Riak: A distributed NoSQL key-value data store, offering high availability, fault tolerance, scalability, and operational simplicity. It offers both, an open source version and an enterprise version. RIAK is inspired by Amazon's Dynamo, a highly available key value data store [15]. Fault tolerance is ensured by data replication and automatically distributing the data across the cluster improving performance and resilience.
2- Infinispan: A distributed NoSQL Key-Value datastore developed by Red Hat. IT supports transaction, MapReduce. Infinispan is able to ensure data's persistence to the filesystem through pluggable architecture.
3- Hazelcast: An in memory data grid based on Java, Hazelcast is an open source datastore. It supports horizontal scaling through even distribution of the data among the nodes of a computer cluster.
4- Redis: An open source distributed in memory key-value database supporting a variety of data structures, such as lists, maps, sets, sorted sets, and strings.
5- Voldemort: a distributed key-value datastore for high scalability storage [16], Voldemort is used by LinkedIn.

A comparison between different key-value databases with respect to persistency, replication, sharding, consistency, API provided, method of query, and CAP support is provided in table 1.

TABLE 1:KEY-VALUE TYPES AND CHARACTERISTICS[11][14]

	PERSISTENCE	REPLICATION	SHARDING	CONSISTENCY	API	QUERY METHOD	IMPLEMENTATION LANGUAGE	CAP
RIAK	Bitcask (log-structured store), LevelDB, In-Memory and Multi-backend (different stores for different keys)	Ring (next N-1)	Consistent Hashing	Tunable Consistency (can choose between Eventual or Strong)	PBC	REST-ful API Apache Solr MapReduce	Erlang	AP
INFINISPAN	Simple File Storage, BerkeleyDB, JDBM, JDBC	Ring (next N-1)	Consistent Hashing	Tunable Consistency	HTTP, Java	Get,MapReduce	Java	AP/AC/CP
HAZELCAST	User-defined MapStore,whichcan be persistent	Ring (next N-1)	Consistent Hashing	Strong	HTTP, Java , C#	Get,MapReduce	Java	AP
REDIS	Snapshotsatspecified intervalsby default oranAppend-only file. Both can becombined	Master-Slave	Not supported	Eventual	Java,C,C#,Ruby,Perl,Scala	Get	C	AP
VOLDEMORT	BerkeleyDB, In-Memory, MySQL	Ring (next N-1)	Consistent Hashing	eventual	Java, Python	Get	Java	AP

IV.3 Column-oriented databases:

General definition: a database architecture family which stores the data by column instead of by row. This is done by storing the columns of the tables as rows in the actual storage, enabling an increase in performance in applications requiring real time analytics. Popular Column oriented databases include:

1- HBase: it's an open-source database system that is based on top of HDFS, with low latency because it allows random access and it supports real time data processing. Hbase is designed to be scaled linearly for data with high volume. It provides powerful features such as compression and in memory processing, the main aim of this technology is to provide a table like architecture.
2- Hypertable: an open-source database management system founded by google, and modeled after google big table, that is scalable and compatible with multiple distribute storage systems. It provides high performance and scalability by breaking the tables into a number of key value pairs, distributing them among the different shards. [17]
3- Cassandra: an open source distributed column oriented databased founded by Facebook, providing a scalable database capable of handling a huge amount of data with adjustable consistency, making Cassandra an eventually consistent database [18].
4- Accumulo: an open source distributed big table like database, that can run on top of Hadoop, zookeeper, and provides additional features like compression, and management of access to the cell level. Commonly used when access is required to the cell level [11]

A comparison between different column-oriented databases with respect to persistency, replication, sharding, consistency, API provided, method of query, and CAP support is provided in table 2.

TABLE 2 COLUMN ORIENTED TYPES AND CHARACTERISTICS [11][14]

	PERSISTENCE	REPLICATION	SHARDING	CONSISTENCY	API	QUERY METHOD	IMPLEMENTATION LANGUAGE	CAP
HBASE	built on top of HDFS	Handled on HDFS	By key value, it allows auto sharding	High consistency	Java REST-ful Thrift Avro	MapReduce	Java	CP
HYPERTABLE	HDFS	HDFS	Key-value Allows chunking	High consistency	C++ API Java REST-ful Thrift Avro	HQL (Hypertable query language), MapReduce	C++	CP
CASSANDRA	Proprietary format	Auto partition with zero loss	By row hashing	Eventually consistent	CQL Thrift	MapReduce	Java	AP
ACCUMULO	HDFS	Multi master replication	By row	Eventually consistent	Java REST-ful Thrift	MapReduce	Java	CP

IV.4 Document-Oriented database:

General definition:

A database system designed for the storage, retrieval and management of document-oriented information. One of the main categories of NoSQL databases, since document oriented data constitutes a large portion of semi-structured data available online. Here are some of the most popular document-oriented databases.

1- CouchDB: An open source document oriented NoSQL database, it transfers, stores, processes the data using multiple formats.
2- MongoDB: a cross platform document oriented NoSQL database program, it uses JSON-like documents with schema. It features Ad hoc queries, indexing, replication, load balancing, aggregation, and transactions.
3- Terrastore: a distributed document oriented DBMS that guarantees per-document consistency.

4- RavenDB: a fully transactional document oriented NoSQL database. It is also a multimodal database offering a variety of data models, such as: document, graph, key/value, and time series.

A comparison between different document-oriented databases with respect to persistency, replication, sharding, consistency, API provided, method of query, and CAP support is provided in table 3.

TABLE 3 DOCUMENT ORIENTED TYPES AND CHARACTERISTICS [14][11]

	PERSISTENCE	REPLICATION	SHARDING	CONSISTENCY	API	QUERY METHOD	IMPLEMENTATION LANGUAGE	CAP
COUCHDB	CouchDB	B-Tree	Master-Master	Available with extensions	Eventual	HTTP and Json	MapReduce	AP
MONGODB	MongoDB	BSON objects GRIDFS (for big files)	Master-Slave	By field	Strict, or Eventual	HTTP and JSON Mongo wire protocol and BSON	By field, cursors, and MapReduce	AP/CP
TERRASTORE	Terrastore	Terrastore storing support	Master-Slave	Consistent Hashing	Eventual	HTTP and JSON	Conditional queries	AP
RAVENDB	RavenDB	Extensible storage engine	Master-Slave	User defined	Eventual	HTTP and JSON, .NET	MapReduce LINQ	AP

IV.5 Graph databases

General definition:

A graph database is a database that utilized the graph data structures to represent the data and its relational structure. The graph's ability to capture relations among data allows efficient storage and retrieval of the data in a fast manner. Some of the most common graph databases are:

1- Neo4J: an acid compliant transactional graph database. Considered the most popular graph database, and ranked 22nd among databases in general [19].
2- InfiniteGraph: a distributed enterprise graph database. It allows concurrency, consistency, multi-threaded processing, and cloud support.
3- InfoGrid: a web based graph database with components that enable the development of REST-ful web applications on a graph foundation [20].
4- HypergraphDB: a general purpose, open source graph oriented data base management system, that based on directed hypergraphs structure. It gestures powerful data modeling and knowledge representation, graph traversals and relational like queries, customizable indexing, non-blocking concurrent read and writes, and fully transactional and multi-threaded architecture [21].
5- AllegroGraph: a closed source graph oriented database system, mainly used for the retrieval and management of document-oriented information. It is heavily used in commercial projects. [22]

A comparison between different graph databases with respect to persistency, replication, sharding, consistency, API provided, method of query, and CAP support is provided in table 4.

TABLE 4 GRAPH DATABASE TYPES AND CHARACTERISTICS [14][11]

	PERSISTENCE	REPLICATION	SHARDING	CONSISTENCY	API	QUERY METHOD	IMPLEMENTATION LANGUAGE	CAP
NEO4J	Apache Lucene	Master-Slave	Manual	Eventual	Java, HTTP and JSON	Java API	Java	AP
INFINITEGRAPH	Objectivity	Peer to Peer	Rule based	Tunable	C++, Java,python, C#	Graph traversal API	C++	AP/CP
INFOGRID	MySQL, HadoopFS	Peer to Peer	Manual	Eventual	Java, HTTP and JSON	Viewlets, templates HTML, Java	Java	AP
HYPERGRAPHDB	Relations caching + indexes	Master-Slave	Manual	Eventual	Java	Java API	Java	AP
BIGDATA	Indexes (B+ trees)	Master-Slave	Dynamic	Eventual	Java	SPARKQL, RDFS++	Java	AP
ALLEGROGRAPH	indexes	Master-Slave	Manual	Eventual	HTTP and Json	SPARQL, graph traversal API	Lisp	AP

Some studies have compared the main databases for each type with respect to many different features the results of these comparisons are listed in the following tables:

TABLE 5 MOST COMMON DATABASES AGAINST QUALITY ATTRIBUTES [23]

	AEROSPIKE	CASSANDRA	COUCHBASE	COUCHDB	HBASE	MONGODB	VOLDEMORT
AVAILABILTY	very good	very good	very good	very good	mediocre	mediocre	very good
CONCSISTENCY	very good	very good	good	good	average	very good	good
DURABILITY	mediocre	good	good	mediocre	good	good	good
MAINTAINABILITY	good	average	good	good	mediocre	average	mediocre
READ-PERFORMANCE	good	mediocre	very good	average	mediocre	very good	good
RECOVRY TIME	very good	bad	good	unknown	unknown	very good	unknown
RELIABILITY	mediocre	good	mediocre	good	good	very good	unknown
ROBUSTNESS	good	good	average	average	bad	average	unknown
SCALABILITY	very good	very good	very good	mediocre	very good	mediocre	good
STABILIZATION TIME	bad	good	good	unknown	unknown	bad	unknown
WRITING PERFORMANCE	good	very good	good	mediocre	good	mediocre	very good

TABLE 6 PERFORMANCE COMPARISONS FOR COMMON DATABASES [24]

	MONGODB	RAVENDB	COUCHDB	CASSANDRA	HYPERTABLE	COUCHBASE
READING TIME FOR 10 OPERATIONS (MS)	8	140	23	115	60	13
READING TIME FOR 50 OPERATIONS (MS)	14	351	101	230	83	23
READING TIME FOR 100 OPERATIONS (MS)	23	539	196	354	103	46
READING TIME FOR 1000 OPERATIONS (MS)	138	4730	1819	2385	420	277
READING TIME FOR 10000 OPERATIONS (MS)	1085	47459	19508	19758	3427	1968
READING TIME FOR 100000 OPERATIONS (MS)	10201	426505	176098	228096	63036	17214
WRITING TIME FOR 10 OPERATIONS (MS)	61	570	90	117	55	60
WRITING TIME FOR 50 OPERATIONS (MS)	75	898	374	160	90	76
WRITING TIME FOR 100 OPERATIONS (MS)	84	1213	616	212	184	63
WRITING TIME FOR 1000 OPERATIONS (MS)	387	6939	6211	1200	1035	142
WRITING TIME FOR 10000 OPERATIONS (MS)	2693	71343	67216	9801	10938	936
WRITING TIME FOR 100000 OPERATIONS (MS)	23354	740450	932038	88197	114872	8492
DELETING TIME FOR 10 OPERATIONS (MS)	4	90	71	33	19	6
DELETING TIME FOR 50 OPERATIONS	15	499	260	95	63	12
DELETING TIME FOR 100 OPERATIONS	29	809	597	130	110	14
DELETING TIME FOR 1000 OPERATIONS	235	8342	5945	1061	1001	81
DELETING TIME FOR 10000 OPERATIONS	2115	87562	67952	9230	10324	805
DELETING TIME FOR 100000 OPERATIONS	18688	799409	705684	83694	130858	7634
ALL KEYS FETCHING TIME FOR 10 OPERATIONS	4	101	67	47	3	nan
ALL KEYS FETCHING TIME FOR 50 OPERATIONS	4	113	115	116	3	nan
ALL KEYS FETCHING TIME FOR 100 OPERATIONS	5	115	19	5	3	nan
ALL KEYS FETCHING TIME FOR 1000 OPERATIONS	19	116	173	76	5	nan
ALL KEYS FETCHING TIME FOR 10000 OPERATIONS	98	136	1063	237	25	nan
ALL KEYS FETCHING TIME FOR 100000 OPERATIONS	702	591	9512	709	159	nan

TABLE 7 BEST FIT APPLICATIONS FOR SOME KEY-VALUE TYPES[25]

	RIAK	REDIS	VOLDEMORT
BEST FOR	High availabilty, partition tolerance, persistence	For rapidly changing data , frequently written, rarely read statistical data	Application with large requirement on data capacity

TABLE 8 BEST FIT APPLICATIONS FOR SOME COLUMN ORIENTED TYPES[25]

	HBASE	CASSANDRA	ACCUMULO
BEST FOR	Real-time access, bulk operations(indexing,..)	When you write more than you read	Access on the cell level

TABLE 9 BEST FIT APPLICATIONS FOR SOME DOCUMENT ORIENTED [25]

	MONGODB	COUCHDB
BEST FOR	Dynamic queries, defining indexes, good performance on big DB	Accumulating , occasionally changing data, pre-defined queries to be run, web use cases and mobile applications

V. CONCLUSION AND FUTURE WORK

We hope this paper provides a helpful overview of database technologies, and acts as a resource of the different non-relational databases to help researchers and people from the industry choose the suitable technology for their specific application. In the future, we would like to design a custom benchmark for the assessment the different NoSQL technologies to provide an empirical standing that enables comparing the different database systems concretely based in experimental results and to cover additional criteria.

REFERENCES

[1] https://www.oracle.com/a/tech/docs/database19c-wp.pdf

[2] [Stonebraker, M., & Rowe, L. A. (1986). *The design of Postgres* (Vol. 15, No. 2, pp. 340-355). ACM

[3] https://info.microsoft.com/rs/157-GQE-382/images/EN-US-CNTNT-white-paper-DBMod-Microsoft-SQL-Server-2019-Technical-white-paper.pdf

[4] Database Differences: Microsoft SQL Server vs. Oracle Database. (n.d.). Retrieved from https://www.cybrary.it/0p3n/database-differences-microsoft-sql-server-vs-oracle-database/.

[5] Stonebraker, M. (2010). SQL databases v. NoSQL databases. Communications of the ACM, 53(4), 10-11.

[6] Kumar, R., Gupta, N., Maharwal, H., Charu, S., & Yadav, K. (2014). Critical analysis of database management using newsql. International Journal of Computer Science and Mobile Computing, 3, 434-438.

[7] Moniruzzaman, A. B. M. (2014). Newsql: Towards next-generation scalable rdbms for online transaction processing (oltp) for big data management. arXiv preprint arXiv:1411.7343.

[8] Stonebraker, M., & Weisberg, A. (2013). The VoltDB Main Memory DBMS. IEEE Data Eng. Bull., 36(2), 21-27.

[9] Piekos, J. (2015). SQL vs, NoSQL vs. NewSQl: finding the right solution. *Retrieved June*, *11*, 2018.

[10] Kumar, R., Parashar, B. B., Gupta, S., Sharma, Y., & Gupta, N. (2014). Apache Hadoop, NoSQL and NewSQL solutions of big data. *International Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE)*, *1*(6), 28-36.

[11] Acharya, B., Pandey, M., & Rautaray, S. S. SURVEY ON NoSQL DATABASE CLASSIFFICATION: NEW ERA OF DATABASES FOR BIG DATA.

[12] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In *2011 6th international conference on pervasive computing and applications* (pp. 363-366). IEEE.

[13] Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data, 2(1), 18.

[14] Corbellini, A., Mateos, C., Zunino, A., Godoy, D., & Schiaffino, S. (2017). Persisting big-data: The NoSQL landscape. Information Systems, 63, 1-23.

[15] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... & Vogels, W. (2007, October). Dynamo: amazon's highly available key-value store. In *ACM SIGOPS operating systems review* (Vol. 41, No. 6, pp. 205-220). ACM.

[16] (n.d.). Retrieved from http://www.project-voldemort.com/voldemort/.

[17] Khetrapal, A., & Ganesh, V. (2006). HBase and Hypertable for large scale distributed storage systems. Dept. of Computer Science, Purdue University, 10(1376616.1376726).

[18] Cassandra, A. (2014). Apache cassandra. Website. Available online at http://planetcassandra. org/what-is-apache-cassandra, 13.

[19] Engines Ranking. (n.d.). Retrieved from https://db-engines.com/en/ranking/graph dbms.

[20] InfoGrid. (n.d.). Retrieved from https://dbdb.io/db/infogrid.

[21] A Graph Database. (n.d.). Retrieved from http://www.hypergraphdb.org/.

[22] (n.d.). Retrieved from https://allegrograph.com/.

[23] Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(1), 18

[24] Li, Y., & Manoharan, S. (2013, August). A performance comparison of SQL and NoSQL databases. In *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)* (pp. 15-19). IEEE.]

[25] B. H. K. P. H. J. (2017). A Survey of Non -Relational Databases with Big Data. International Journal on Recent and Innovation Trends in Computing and Communication, 5(11), 143 -. Retrieved from https://ijritcc.org/index.php/ijritcc/article/view/1291