

The Role of ICT in our Daily Life Applications: Obstacles and Challenges

A Comparative Study of NoSQL Databases

Musbah J. Aqel², Aya Al-Sakran¹ and Mohammad Hunaity³

¹Faculty of Information Technology, Applied Science Private University, Amman, Jordan

²Department of Management Information Systems, School of Applied Sciences Cyprus International University, Lefkosa, North Cyprus

³Faculty of Information Technology, AlBalqa'a Applied University, Jordan

ABSTRACT

A lot of traditional relational databases are still used so far in a very large number of applications. Recently, new data bases technologies have been developed in the need to deal with the increasing amount of complex data. Millions of users would do their updates and reads on web applications, in contrast to traditional DBMSs and data warehouses which have no ability to scale horizontally on these applications. Choosing the most appropriate NoSQL database would be sometimes tricky so it is important to know the features of the NoSQL database. In this survey, most popular NoSQL databases: Cassandra MongoDB, CouchDB, Hbase and SimpleDB are compared. This comparison allows the user to choose the most appropriate database, basing on application's needs. Also, the focus is on the NoSQL models and their descriptions, and when they are best used. Lastly, the compared advantages and disadvantages of these data stores are listed to discuss selecting appropriate NoSQL database which processes huge volumes of data; and provides global overview of these non-relational NoSQL databases.

KEY WORDS: NOSQL DATABASE, RELATIONAL DATABASE MANAGEMENT SYSTEM, STRUCTURED QUERY LANGUAGE.

INTRODUCTION

Big data is a term for data sets that are very large and complex which traditional data processing applications would not handle with them. It is related to the huge development of the Internet, mobile devices and cloud computing. Increasingly, organizations today are facing more and more big data challenges, including analyzing,

capturing, searching, sharing, storing, transferring, visualizing, and querying, updating and information privacy. They have access to all the information, but they do not know how to get value out of it because it is in a semi structured or unstructured format. As a result, they do not even know whether to keep or there is a capability to keep it or not [1].

ARTICLE INFORMATION:


*Corresponding Author: musbahaqel@yahoo.com

Received 12th Nov, 2018

Accepted after revision 29th Dec, 2018

BBRC Print ISSN: 0974-6455

Online ISSN: 2321-4007 CODEN: USA BBRCBA

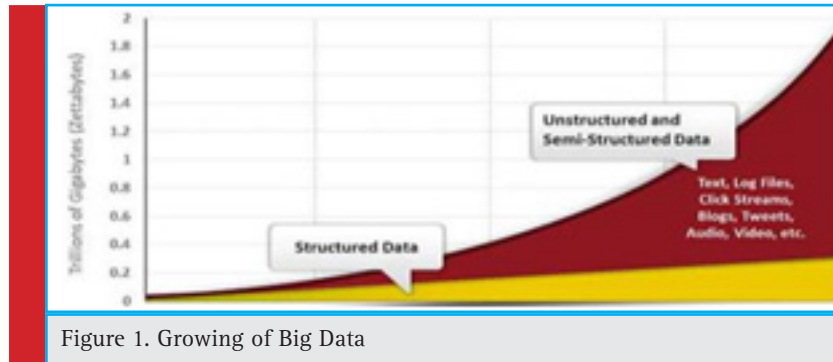
 Thomson Reuters ISI ESC / Clarivate Analytics USA and Crossref Indexed Journal

NAAS Journal Score 2018: 4.31 SJIF 2017: 4.196

© A Society of Science and Nature Publication, Bhopal India 2018. All rights reserved.

Online Contents Available at: <http://www.bbrc.in/>

DOI: 10.21786/bbrc/12.1/3



Big data has the ability to help companies to improve their operations and make faster decisions. This data, when captured, formatted, manipulated, stored, and analyzed would help a company to obtain helpful idea to increase revenues, customers, and improve operations [2].

Nowadays the widely accepted interpretation defines big data as 3 V's. The first one is called Variety. Today data is presented in different formats which are called unstructured data such as texts, videos, images, sound and much more. The second one is Velocity that can be defined as the rate at which data is generated, such as around 100 terabytes of data is uploaded daily on Facebook, YouTube users upload 48 hours of video every minute. And the last one is the Volume of data: terabytes of data being processed daily requires very efficient techniques to store and process data [3].

Figure1. below shows the big data about 80% of the data generated now is unstructured or semi-structured. The total amount of data is growing very fast [4]

Types of NoSQL Databases

Recently NoSQL database are generated by the huge growth of data mostly in web and mobile applications. If it is to be considered that social media web pages such as Facebook, LinkedIn and Twitter, which are dealing with thousands of terabytes of data, then it must be noticed that besides handling huge data volume, those systems still have to maintain latency, meaning that reading and writing are supposed to be responded immediately [3]. As previously mentioned there are many NoSQL types which recently have appeared with different performances; therefore, they are compared in terms of performance and verified how the performance are related to the database types [5]. In this Section the following NoSQL databases: Cassandra MongoDB, CouchDB, Hbase and SimpleDB are compared in details.

2.1 Cassandra

Facebook continues to be the most popular and the largest social media site that contains thousands of users of the system simultaneously using tens of mil-

lions of servers that are distributed in many data centers around the whole world. There is always a probability any server and any of network components may fail at any given time as any software system needs to be constructed in a way that to deal with failures immediately and efficiency. To meet those Requirements reliability and scalability, Facebook Company has developed new NoSQL Database in 2007 that called Cassandra. Cassandra is the leading NoSQL database that has been developed by Facebook to solve the problem with slow Inbox search across millions messages, in which they had to deal with very large volumes of data in such a way that was impossible to scale with traditional method and to handle Facebook's millions of users searching hits per day. It is an open source non-relational, column oriented distributed system for handling very large amounts of massive structured data distributed over many servers around the whole world [6], while providing highly available services to all users at any given time with no single point of failure. Also, it is used in data-heavy apps like Instagram, Snapchat which daily handle with an average of 100 million photos uploaded, and iCloud which stores over 40 million songs in its database [7, 8].

Cassandra is mainly consists of two systems: Google's BigTable and Amazon's Dynamo. Both systems meet the challenge of scaling, but they do it in different methods: BigTable uses the distributed file system Google already had, while Dynamo is based on a distributed hash table. Cassandra combines the data structure of Big Table, and the high availability of Dynamo [9].

Cassandra uses the eventual consistency model which is used for writing operations and has no central node. Data would be read from or written to any node in a cluster and that provides us with continuous horizontal scalability and has no single point of failure. Because Cassandra uses peer-to-peer fault-tolerance technology, no master/slave setup, failover. This means any node in the cluster would perform users query in the case of any failure happen, the figure1 below shows peer to peer structure [10].

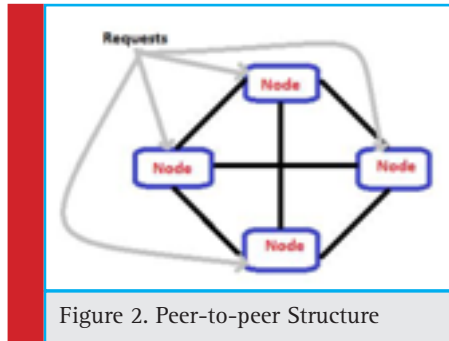


Figure 2. Peer-to-peer Structure

As shown in the figure 2 a peer-to-peer structure means each node is corresponding to the others, and there are replications in the ring. If one node fails, the service will continue of performance. Data on the failed node would still be accessed. The peer-to-peer design also makes it easy to scale by adding new nodes. Because the behavior of each node is separated, in order to add a new server, it is simply need to add it to the cluster [8].

2.1.1 Cassandra API

Cassandra API consists of the following three simple methods:

```
Insert(table,key,rowMutation)
Get (table,key,columnName)
Delete(table,key,columnName)
```

The column Name can refer to a specific column within a column family, a column family, a super column family, or a column within a super column [11].

2.1.2 Facebook Inbox Search

One of the popular applications of Cassandra is Facebook Inbox Search problem, the inbox search system needs to handle a very high speed writing, billions of writings per day and is also required to scale to a very large amount of users. To keep search, data has to be replicated to all data centers which are used by differ-

ent users distributed at different geographic areas. It is noticed that there are very powerful and restricted operational requirements on Facebook’s platform in terms of

Table 1. Describe some production measured numbers are showed for their performance.

Latency	State Search	Term Search
	Interactions	
Min	7.69ms	7.78ms
Median	15.69ms	18.27ms
Max	26.13ms	44.41ms

performance, reliability and efficiency, and to keep up the continuous infinite growth the platform used needs to be highly scalable that dealing with failures in an very efficient way[11].

For each of these super columns the individuals are the columns. In order to make the searches, fast Cassandra provides certain hits for intelligent caching of data; for instance, when a user clicks into the search bar an asynchronous message is sent to the Cassandra cluster to prime the buffer cache with that user’s index. This way when the actual search query is executed, the search results are likely to already be in memory. The system currently stores about 50+TB of data on a 150 node cluster, which is spread out between on different data centers [12].

As shown the table1 above Cassandra can support a very high update throughput while delivering low latency [12].

2.1.3 Cassandra Characteristics

The schema of Cassandra is very flexible and does not require any database schema design as well as adding and deleting fields are very convenient. And it is Supports range queries and the scalability is very high such that a single point of failure does not affect the all cluster and it supports linear expansion [13].

Table 2. below shows how Cassandra Differ from a Relational Database

Relational Database	Cassandra
Handles moderate incoming data velocity	Handles high incoming data velocity
Data arriving from one/few locations	Data arriving from many locations
Manages primarily structured data	Manages all types of data
Supports complex/nested transactions	Supports simple transactions
Single points of failure with failover	No single points of failure; constant uptime
Supports moderate data volumes	Supports very high data volumes
Centralized deployments	Decentralized deployments
Data written in mostly one location	Data written in many locations
Supports read scalability (with consistency sacrifices)	
Deployed in vertical scale up fashion	Deployed in horizontal scale out fashion

2.1.4 Cassandra Data Model:

Key-value data model means each value corresponds to a Key [14].

Column-oriented: column-store systems completely vertically partition a database into a collection of individual columns that are stored separately. By storing each column separately on disk, these column-based systems enable queries to read just the attributes they need, rather than read all rows from disk and discard unneeded attributes once they are in memory [15].

Document: document database and Key-value is very similar in structure, but the Value of document database is semantic, and is stored in JSON or XML format [14].

2.1.5 Security Issues in Cassandra

All passwords in Cassandra are encrypted using of MD5 hash function so that the passwords are very weak. If any malicious user can attack client authorization, user can extract the data because there is no authorization mechanism in internode message exchange. Cassandra is potential for denial of service attack because it performs one thread per one client and it does not support inline auditing [16]. In order to avoid this, the application must encrypt any sensitive information before writing it to the database. Also, operating-system level mechanisms should be used to prevent access to the files by unauthorized users [9].

Cassandra uses a query language called Cassandra Query Language (CQL), which is something like SQL. The

Table 3. Below Show the Different Measurements for Each Data Model.

Data Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key-Value Store	High	High	High	None	None
Column	High	High	Moderate	Low	Minimal
Document	High	High	High	Low	Low
Oriented Store					

authors show that injection attack is possible on Cassandra like SQL injection using CQL. Cassandra also has problem in managing inactive connection [16].

Cassandra uses a query language called Cassandra Query Language (CQL), which is something like SQL. The authors show that injection attack is possible on Cassandra like SQL injection using CQL. Cassandra also has problem in managing inactive connection [16].

2.2 Mongo DB

Mongo DB is an open-source document-oriented database written in C++ that uses JSON, which is used in a schema that require less data model. MongoDB's mainly provides horizontal scalability by using the automatic sharing. Replication is also supported using locks and the asynchronous master-slave model which means writing operations are only processed by the master node and reading operations can be made from both the master node and from one of the slave nodes. Writings are distributed to the slave nodes by reading from the master's operation log. Clients of database have the ability to select which kind of consistency models they wish, by defining whether reading from secondary nodes are allowed and from how many nodes the confirmation must be obtained [17].

In Mongo DB, document manipulation is a strong focus, also the database provides different frameworks and ways of interacting with documents. These can be

queried, sorted, projected, iterated with cursors, and aggregated, among other operations. The changes to a document are guaranteed to be atomic. Indexing can be used on one or several fields (implemented using B-trees), with the possibility of using two-dimensional spatial indexes for geometry-based data. There are many different programming interfaces supported by MongoDB, with most popular programming languages having native bindings [17].

2.2.1 Mongo DB Features

MongoDB stores data in JSON documents. JSON provides a rich data model that maps to native programming language types, and the dynamic schema makes it easier to develop your data model than with a system that enforces specific schemas such as a RDBMS [13].

Power: MongoDB provides a lot of the features of a traditional RDBMS such as secondary indexes, dynamic queries, sorting, rich updates, and easy aggregation. This gives more functions that you are used to from an RDBMS, with the flexibility and scaling capability that the non-relational model allows [13].

Ease of use: MongoDB is very easy to install, configure, maintain, and use. Because it provides few configuration options [8]. This means you can begin right into developing the application, instead of spending a lot of time to search and try to make database configurations [13].

MongoDB can do very strong consistency by using two parameters: first set to read only from the master, meaning that only one node will be accessed for read. Another way is to set “write” parameter to “replica acknowledged”, which ensures that write is successfully completed on all the nodes. These techniques actually force the data store to the synchronous replication and therefore decrease the performance [3].

2.2.2 MongoDB Architecture

A MongoDB cluster is different from a Cassandra cluster. The most noticeable difference is the lack of Homology: not each node in a MongoDB cluster is the same [9].

2.2.3 Security Issues in MongoDB:

All data in MongoDB is stored as plain text and there is no encryption mechanism to encrypt data files. This means that any malicious user with access to the file system can extract the information from the files. It uses SSL with X.509 certificates for secure communication between user and MongoDB cluster and intra-cluster authentication but it does not support authentication and authorization when running in Sharded mode. The passwords are encrypted by MD5 hash algorithm and MD5 algorithm is not a very secure algorithm. Since mongo uses Javascript as an internal scripting language, authors in show that MongoDB is potential for scripting injection attack [16].

2.3 SimpleDB

Amazon’s SimpleDB is a Web service that provides basic database functions of information indexing and querying in the cloud [18]. SimpleDB has been published in 2007. As the name indicates its model is very simple, it has collection of simple operations such as Select, Delete, GetAttributes, and PutAttributes on documents. SimpleDB is simpler than other document stores, as well as it does not allow nested documents. Also it is support eventual consistency. Like most of the other systems, it does asynchronous replication. Unlike key-value data stores, and like the other document stores, SimpleDB supports more than one grouping in one database: documents are put into domains, which support multiple indexes. You can simply put domains and their meta-data. Select operations are on one domain, and specify

a conjunction of constraints on attributes, simply in the form of [19].

```
select<attributes> from <domain> where
<list of attribute value constraints>
```

SimpleDB is appropriate option for quick setup without any administration effort. However, the NoSQL model is main problem to applications already developed with relational databases. To adjust a relational-based application to a cloud platform may increase in a large maintenance effort. In order to decrease a situation like that, It is suggested to use an access layer that makes the translation of SQL requests to the SimpleDB API and returns data in a relational format. It is called SimpleSQL. In this first version, the basic layer is only able to perform the four traditional operations: INSERT, UPDATE, DELETE and SELECT, while SimpleSQL provides more details about its functionality and implementation. SimpleSQL is developed by Microsoft .NET Framework version 3.5, using C# as programming language. Figure 3 shows the layer architecture, which highlights the three steps of an SQL command processing [20].

SimpleDB does not automatically distribute data over servers. Some horizontal scaling can be achieved by reading any of the portions, if you Application don’t effect by having the latest version. Writes do not scale, however, because they must go asynchronously to all copies of a domain. If customers want better scaling, they must do so manually by sharing themselves [19].

SimpleDB has currently constraints, some of which are quite limiting: a 10 GB maximum domain size, a limit of 100 active domains, a 5 second limit on queries, and so on. Amazon does not license SimpleDB source or binary code to run on your own servers [19].

2.3.1 Processing and Return in SimpleDB

When the user identify the command, SimpleSQL translate it to a SimpleDB that is called REST method. All the commands begin with the identification of the target SimpleDB domain from the target table, extracted from the command. DELETE and UPDATE commands return the number of affected items. INSERT returns the result of the operation (success or fail) and SELECT returns the data in a table structure using .NET class DataTable [20].

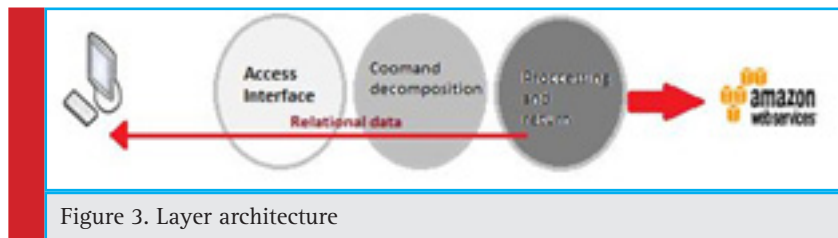


Figure 3. Layer architecture

2.4 CouchDB

CouchDB currently is one of the most popular NoSQL databases. The flexible document structure makes it ideal for using, fault-tolerant database, which supports data formats such as ISON and AtomPub, it provides REST-style API. CouchDB comply with ACID properties to ensure the consistency of data. Also, CouchDB provides a P2P-based distributed database solution that supports bidirectional replication. However, it also has some limitations, such as only providing an interface based on HTTP REST, concurrent read and write performance is not ideal and so on [14].

CouchDB is not only a NoSQL database, but it is also a web server for applications developed in JavaScript. some of features of using CouchDB as a web server is that applications in CouchDB can be deployed by simply putting them into the database and that the applications can directly access the database without any overhead of a query protocol [21].

CouchDB is a schema-less database which is called “document storage”. Each document is store in JSON, which is a human-readable markup language similar to XML, but requiring much less overhead[22], Also CouchDB provides durability when the system down. All updates will be deleted to disk on commit, by writing to the end of a file. By default, it flushes to disk after every document update. Together with the MVCC mechanism, CouchDB’s durability also provides ACID semantics at the document level. Clients call CouchDB through a RESTful interface [19].

2.5 HBase

HBase is an open source project, written in Java and developed by the Apache software foundation. It is an open source implementation. HBase works with Apache’s Hadoop Distributed File System (HDFS) as basic data storage and it is considered as column-oriented. HBase is a good option for high performance real time queries of very huge amounts of distributed data [23].

In HBase, data is stored in the form of HBase tables (HTable) that are multidimensional sorted maps. The index of the map is the row’s key, column’s name, and a timestamp. Columns are grouped into column families. Column families must be created before data can be stored under any column key in that family. Data is maintained in order by row key. Finally, each column can have multiple versions of the same data indexed by their timestamp. A read or write operation is performed on a row using the row-key and one or more column-keys. Update operations on a single row are atomic. Any update performed is immediately visible to any read operation. HBase exports a non-blocking key-value interface on the data: put, get, delete, and scan

operations. HBase closely matches the scale-out properties assumed for NoSQL databases [24].

2.5.1 Security Issues in HBase

The Security in HBase depends on SSH for inter-node communication, which supports user authentication by the use of SASL (Simple Authentication and Security Layer) with Kerberos. As well as it supports authorization by ACL (Access Control List) [16].

Different performance in Read, Write and delete in MongoDB, Cassandra and CouchDB

In this section, Evaluation some of NoSQL databases categories with a matrix on basis of few attributes design, integrity, indexing, distribution, system have been presented. See table 4 [25].

3.1 Compared between MongoDB, CouchDB, and Cassandra in Read Operation

First experiment compares the time taken to read values corresponding to given keys from the cluster. The below table summarizes the results. In the tables, the number of operations refers to the number of times a given operation (such as read) is executed in the test [26].

Table 4. Shows the Time for Reading					
Number of Opertions	10	50	1000	10000	100000
MongoDB	8	14	138	1085	10201
CouchDB	23	101	1819	19508	176098
Cassandra	115	230	2385	19758	228096

Sorted by read performance the list of databases: is MongoDB, CouchDB, and Cassandra. Of these, Cassandra is column-family databases; and MongoDB, and CouchDB are documents-oriented databases. It is observed there is no correlation between the data model and performance. As well as the read performance of MongoDB is better than CouchDB and Cassandra [26].

3.2 Compared between MongoDB, CouchDB, and Cassandra in Writing Operation

Second experiment measures the time taken to write key value pairs to the bucket. If the key-value pair already exists in the bucket, this amounts to updating the existing value [26].

Table 5. Below Summarizes the Results in Writing Operations						
Number of Opertions	10	50	100	1000	10000	100000
Mongo DB	61	75	84	387	2693	23354
Couch DB	90	374	616	6211	67216	932038
Cassandra	117	160	212	1200	9801	88197

Sorted by write performance we have the list of databases: MongoDB, Cassandra, and CouchDB. The write performance of CouchDB is worse than Cassandra and MongoDB.

3.3 Compared between MongoDB, CouchDB, and Cassandra in Delete Operation

The last experiment measures the time taken to delete key-value pairs from the bucket [26].

Sorted by delete performance the list of databases is: MongoDB, Cassandra, CouchDB. The delete performance

of MongoDB is better than Couchbase and Cassandra [26].

4. Different Criteria for Comparison between NoSQL Databases SimpleDB, CouchDB, Cassandra and HBase:

4.1 Comparison between NoSQL Databases in in Terms of Design Side:

The following table7 shows some of Nosql criteria for comparison such as classification , Protocol, License, Storage Type and Query Method used for each NOSQL Databases and Fault Tolerance.

Table 6. Shows the Time for Deleting and Summarizes the Result

Number of Opertions	10	50	100	1000	10000	100000
Mongo DB	4	15	29	235	2115	18688
Couch DB	71	260	597	5945	67952	705684
Cassandra	33	95	130	1061	9230	83694

Table 7.

	Simple DB	CouchDB	MongoDB	Cassandra	Hbase
Nosql classification	Document Oriented [21].	Document Oriented (JSON) [21].	Document Oriented (BSON) [27].	Column Database[27].	Column Database [27]
Protocol	TCP/IP[21].	HTTP/REST[21]	TCP/IP[21]	TCP/IP[21].	HTTP/REST[21].
License	****	Apache[25]	AGPL (Drivers: Apache)	Apache[25].	Apache
Storage Type	Document [21].	Document[21]	Document[21]	Columns[21].	Columns[21].
Data Storage	S3 (Simple Storage Solution) [21].	Disk [21].	Disk [21].	Disk[21].	Hadoop[21] .
Query Method	String based query	Map/Reduce [21].	Map/Reduce [21].	Map/Reduce[21]	Map/Reduce[21]
Fault Tolerance	****	****	No single point of failure with peer to peer architecture.[27]	No single point of failure with sharding approach as we can configure multiple mongos instances. Single point of failure in master slave approach. [27]	Single point of failure in master slave approach. Can be overcome by failover clustering.[27]

4.2 Comparision between NoSQL Databases in in terms of System and Characteristics and Architecture side:

Table 8. shows the Architecture, Characteristics, programming language written in, Operating system used for each NOSQL Databases and maximum size.

	Simple DB	Couch DB	Mongo DB	Cassandra	Hbase
Written In	Erlang [21].	Erlang [21].	C++ [21]	Java [21]	Java [21]
Operating System	Linux Mac OS	Linux Mac OS	Linux Mac OS	Linux Mac OS	Linux Mac OS
Value size max	****	20 MB[25]	16 MB [25]	2 GB [25]	2 TB [25]
Characteristics	High Available	High	Consistency	High	Consistency
	And Scalable [29].	Availability	Partition	Availability	Partition
		Partition	Tolerance	Partition	Tolerance
		Tolerance	Persistence [29].	Tolerance	Persistence [29].
		Persistence [29].		Persistence [29].	
Architecture	***	***	1. master slave	Peer to peer	Master Slave
			2. peer to peer via sharding [27]	architecture Model [27].	architecture Model [27].

4.3 Comparison between NoSQL Databases (MongoDB, Cassandra, Hbase) in Read and Write Operations:

Table 9. Shows how database is written and read through Mongo DB, -Cassandra, Hbase:			
	Mongo DB	Cassandra	Hbase
Writes	Fast when data is in RAM[27].	Very fast writes [27].	Writes slower [27].
Reads performance	In a master/slave setup, any changes are written to the master and then passed on to slaves. This model is optimized for reading data, as it allows data to be read from any slave. In sharding reads depend on eventual/strict consistency Level [27].	Performance based on consistency level (decreases in performance with increase in consistency level) and replication Factor [27].	Follows strict consistency model and are optimized for reads. Very fast reads in Hbase with Hadoop support [27].

4.4 Comparison between NoSQL Databases (MongoDB, Cassandra, Hbase) in Terms of System Integrity:

Table 10. Shows how each NOSQL Database differs in terms of (Atomicity, consistency, isolation, durability, integrity Model, Replication model Concurrency Control, Transactions and Replication)					
	Simple DB	Couch DB	Mongo DB	Cassandra	Hbase
Atomicity	*****	Yes [25].	Conditional [25].	Yes [25].	Yes [25].
Consistency	NO[28]	Yes [25].	Yes [25].	Yes [25].	Yes [25].
Isolation	****	Yes [25].	No [25].	Yes [25].	Yes [25].
Durability		Yes [25].	Yes [30].	Yes [30].	Yes [30].
Integrity Model	*****	MVCC [25].	BASE [25].	BASE [25].	Log Replication [25].
Replication		Master Slave	Master Slave	MultiMaster	Master Slave
Model		Replication	Replica Replication	Replication [2]	Replication
Concurrency Control	None [21].	MVCC (Multi Version Concurrency Control) [21].	Locks (Instant update) [21]	MVCC (Multi Version Concurrency Control)[21].	Locks [21].
Transactions	No[21].	No [21].	No [21].	Local [21].	Local [21].
Replication	Asynchronous [21].	Asynchronous [21].	Asynchronous [21].	Asynchronous [21]	Asynchronous [21]

4.5 Comparison between NoSQL Databases (MongoDB, Cassandra, Hbase) in where to use and the best use for each NOSQL Database:

The following table11 shows the best used and the area of use for each type of DB Types:

Table 11.					
	Simple DB	Couch DB	Mongo DB	Cassandra	Hbase
Best used	For a large number of concurrent access [28]	For accumulating occasionally changing data, on which predefined queries are to be run. Places where versioning is important [25].	If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on big DB. If you wanted CouchDB but your data changes too much, filling up disks [25]	When you need to store data so huge that it doesn't fit on server, but still want a friendly familiar interface to it [25].	Hadoop is probably still the best way to run Map/Reduce jobs on huge datasets. Best if you use the Hadoop/HDFS stack already [25].
Area of use	Amazon company [28]	****	CMS System, Comment Storage [2]	Banking, Finance, logging [2].	****
Main points to use	Good option for fast setup without any a administration effort [20]	DB consistency ease of use [25].	JSON document store [25]	Store huge datasets in almost SQL [25].	Billions of rows X millions of columns [25].

4.6 Comparison between NoSQL databases (MongoDB, Cassandra, Hbase) in terms of Security:

Table 12. Below presents the security issues between each of NoSQL Databases types:					
Assessment Criteria	Mongo DB	Cassandra	Couch DB	H Base	Simple DB
Authentication	Medium [29].	Low [29].	Medium[29]	Medium [29].	
Access Control	High [29].	Low [29].	Low [29].	Medium [29].	
Secure Configuration	Medium [29].	Low [29].	Low [29].	Low [29].	
Data Encryption	Medium [29].	Medium [29].	Medium [29].	Low [29].	
Auditing	Low [29].	Low [29].	Medium [29].	Medium [29].	

Value means.

High: Provides complete support of required features needed to secure data

Medium: Provides a limited set of security feature only and it is advisable to implement missing features

Low: Offers very basic security features or no security at all

CONCLUSION

Basically, the comparison shows NoSQL databases would not replace relational databases, but instead it will become a better option for specific types of projects. And no one of these NoSQL databases is best for all use cases. A user's prioritization of features will be different depending on the application, as well as the type of scalability and availability required. This survey may help the user to choose the most appropriate data store based on the use case, and some examples of applications that fit well with the different data store categories. And, a storage NoSQL. As overall results in terms of optimization, NoSQL databases can be divided into two categories the databases optimized for reads and the databases optimized for updates. Thus, MongoDB optimized to perform read operations and high availability in an unreliable environment, while Colum Family databases, such as Cassandra and HBase have a better performance during execution of updates, but delivering low latency. Also, CouchDB has some limitations, such as only providing an interface based on HTTP REST. Concurrent read and write performance is not ideal.

REFERENCES

- [1] Zikopoulos, P. and C. Eaton, Understanding big data: Analytics for enterprise class hadoop and streaming data. 2011: McGraw-Hill Osborne Media.
- [2] Mayer-Schönberger, V. and K. Cukier, Big data: A revolution that will transform how we live, work, and think. 2013: Houghton Mifflin Harcourt.
- [3] Gurevich, Y., Comparative Survey of NoSQL/NewSQL DB Systems. 2015, The Open University.
- [4] Băzăr, C. and C.S. Iosif, The Transition from RDBMS to NoSQL. A Comparative Analysis of Three Popular Non-Relational Solutions: Cassandra, MongoDB and Couchbase. Database Systems Journal, 2014. 5(2): p. 49-59.
- [5] Bukharmetov, M., et al., Robust Method for Protecting Electronic Document on Waterway Transport With Steganographic Means by Embedding Digital Watermarks into Images. Reliability and Statistics In Transportation and Communication, 2016: P. 32.
- [6] Hecht, R. and S. Jablonski. Nosql evaluation. in International conference on cloud and service computing. 2011. IEEE.
- [7] Khanna Vadivelu, M., magevadi@indiana.edu Indiana University Bloomington.
- [8] Yu, M., Cassandra to back applications. 2011.
- [9] Okman, L., et al. Security issues in nosql databases. in 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications. 2011. IEEE.
- [10] Sakr, S., et al., A survey of large scale data management approaches in cloud environments. IEEE Communications Surveys & Tutorials, 2011. 13(3): p. 311-336.
- [11] Lakshman, A. and P. Malik, Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 2010. 44(2): p. 35-40.
- [12] Jain, R., S. Iyengar, and A. Arora. Overview of popular graph databases. in Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. 2013. IEEE.
- [13] Zvarevashe, K. and T.T. Gotora, A Random Walk through the Dark Side of NoSQL Databases in Big Data Analytics. International Journal of Science and Research, 2014. 3: p. 506-09.
- [14] Han, J., et al. Survey on NoSQL database. in Pervasive computing and applications (ICPCA), 2011 6th international conference on. 2011. IEEE.
- [15] Abadi, D., et al., The design and implementation of modern column-oriented database systems. 2013: Now.
- [16] Sahafizadeh, E. and M.A. Nematbakhsh, A Survey on Security Issues in Big Data and NoSQL. Int'l J. Advances in Computer Science, 2015. 4(4): p. 2322-5157.

- [17] Grolinger, K., et al., Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, 2013. 2(1): p. 1.
- [18] Leavitt, N., Will NoSQL databases live up to their promise? *Computer*, 2010. 43(2): p. 12-14.
- [19] Cattell, R., Scalable SQL and NoSQL data stores. *AcmSigmod Record*, 2011. 39(4): p. 12-27.
- [20] Calil, A. and R. dos Santos Mello. SimpleSQL: a relational layer for SimpleDB. in *East European Conference on Advances in Databases and Information Systems*. 2012. Springer.
- [21] Padhy, R.P., M.R. Patra, and S.C. Satapathy, RDBMS to NoSQL: reviewing some next-generation non-relational database's. *International Journal of Advanced Engineering Science and Technologies*, 2011. 11(1): p. 15-30.
- [22] Vincent, M.L., F. Kuester, and T.E. Levy. OpenDig: In-field data recording for archaeology and cultural heritage. in *Digital Heritage International Congress (Digital Heritage)*, 2013. 2013. IEEE.
- [23] Weber, S., Nosql databases. University of Applied Sciences HTW Chur, Switzerland, 2010.
- [24] Vilaça, R., et al. An effective scalable SQL engine for nosql databases. in *IFIP International Conference on Distributed Applications and Interoperable Systems*. 2013. Springer.
- [25] Moniruzzaman, A. and S.A. Hossain, Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *arXiv preprint arXiv:1307.0191*, 2013.
- [26] Li, Y. and S. Manoharan. A performance comparison of SQL and NoSQL databases. in *Communications, Computers and Signal Processing (PACRIM)*, 2013 IEEE Pacific Rim Conference on. 2013. IEEE.
- [27] Manoj, V., Comparative study of nosql document, column store databases and evaluation of cassandra. *International Journal of Database Management Systems*, 2014. 6(4): p. 11.
- [28] Stein, R. and V. Zacharias. Rdf on cloud number nine. in *4th Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic*. 2010.
- [29] Zahid, A., R. Masood, and M.A. Shibli. Security of shardedNoSQL databases: A comparative analysis. in *Information Assurance and Cyber Security (CIACS)*, 2014 Conference on. 2014. IEEE.
- [30] Energy management ((HACS4EM) HACS4EM using wireless sensor networks in smart grids. *Procedia Comput Sci* 2014; 32:469-76. <http://dx.doi.org/10.1016/j.procs.2014.05.449>.
- [31] University of Delaware. Car prototype generates electricity, and cash - sciencedaily. *Sciencedaily*.d.<<http://www.sciencedaily.com/releases/2007/12/071203133532.htm>> [accessed 29.11.17].
- [32] Stastny L, Franek L, Bradac Z. Time synchronized low-voltage measurements for smart grids. *ProcediaEng* 2015; 100:1389-95. <http://dx.doi.org/10.1016/j.proeng.2015.01.508>.
- [33] Yilmaz M, Dhansri NR. A smart grid intelligent control framework. *IEEE Green Technol. Conf* 2012;3(19-20):1. <http://dx.doi.org/10.1109/GREEN.2012.6200983>.
- [34] Tarhuni NG, Elkalashy NI, Kawady TA, Lehtonen M. Autonomous control strategy for fault management in distribution networks. *Electr Power Syst Res* 2015;121:252-9. <http://dx.doi.org/10.1016/j.epsr.2014.11.011>.
- [35] Reddy KS, Kumar M, Mallick TK, Sharon H, Lokeswaran S. A review of integration, control, communication and metering (ICCM) of renewable energy based smart grid. *Renew Sustain Energy Rev* 2014;38:180-92.
- [36] Ferrari P, Sisinni E, Flammini A, Depari A. Adding accurate timestamping capability to wireless networks for smart grids. *ComputNetw* 2014;67:1-13. <http://dx.doi.org/10.1016/j.comnet.2014.03.005>.
- [37] Smart Grid Interoperability Panel. About SGIP 2015.<<http://sgip.org/AboutSGIP>> [accessed 29.11.17].
- [38] Samad T and A.M. Annaswamy, "The Impact of control technology-Control for renewable energy and Smart Grid" *www.ieecss.org*. (eds), 2011.
- [39] Yan Y, Qian Y, Sharif H, Tipper D. A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE CommunSurv Tutor* 2013; 15(1):5-20.