A review of comparison between NoSQL Databases: MongoDB and CouchDB

Rupali Kaur, Jaspreet Kaur Sahiwal

Abstract- For years relational database has been a critical part of the technology. It has been, and in some cases is still being, the backbone of the large organizations as well as small one. But some loop holes in the relation database gave birth to NoSQL databases. NoSQL is not so newly emerged one but can be considered as the fast growing one. NoSQL stands for Not only Structured Query Language. Many NoSQL databases are available nowadays as per the requirement of the user. In this review various comparisons of these databases based on different attributes is combined. Additionally, light is put on few terms related to NoSQL databases and are explained in detail. Furthermore, comparison between two most commonly used NoSQL is reviewed in detail.

I. INTRODUCTION

As the name suggests, NoSQL databases support unstructured queries with no schema. Article [1] describes the various models for structuring the databases, which includes ACID, CAP and BASE. NoSQL uses the BASE model approach. BASE refers to (i) Basically Available, (ii) Soft state and, (iii) Eventually consistent. In [1] it has also been stated that the common problems that exists in both relational and NoSQL are (i) security issues, (ii) limitations in scalability and, (iii) problems in the availability of data. In another article [2],on the basis of the BASE properties, downsides of the NoSQL have been described as under:

- Not universal as SQL
- Different NoSQL database does different things
- Not as powerful and expressive as SQL
- Not reliable as they are only few years old
- Unlike relational databases, NoSQL is part of only a small ecosystem with fewer applications.

Although relation databases are very simple which facilitates their use but the problem with these arose with the non-uniformity of the growing data. Millons of data is handled by the organizations and it becomes difficult if relational database is used. Also, the area of IoT is before our door and IoT applications are better with NoSQL. To resolve this non-uniformity and massively growing data, NoSQL was introduced. NoSQL is majorly used in cloud, Bid data, IoT or distributed systems. Rigid schema is avoided in NoSQL and availability, scalability and fault tolerance are the important factors or characteristics of NoSQL databases.

MYSQL and MongoDB (NoSQL database) were compared on the basis of the performance for IoT applications by[3] and from the study it was found that in some cases MYSQL is better than the MongoDB while in others vice-versa was true. So, according to [3], if someone wants to choose a better database for IoT, it will depend on the most used query and application requirements.

In another study [4], the author stated the biggest challenge faced by NoSQL as weak consistency, but with the advancement and popularity of NoSQL, new optimizations related to performance as well as other new features have been added along with the updated iteration version.

A. NoSQL Database types

There are four major types:

- Key-Value Store – It contains Hash Table of keys & values. It is a simple database using an associative array as the fundamental datas model where each key is associated with only one value. Example- Riak, Amazon S3 (Dynamo).
- Document-based Store- It contains tagged elements in the form of documents. Rather than structuring the data model in rows and columns (table format) as done in traditional databases, it is kept unstructured leading to varying schema. This in return provides more flexibility in data modeling. Example- CouchDB.
- Column-based Store- It contains only one column of data in the storage block and stores each column continuously either on disk or in-memory and each left column will be stored in sequential blocks. Example- HBase, Cassandra.
- Graph-based-It can be seen as a network database that uses nodes and edges for storage and representation of data where nodes represent entities and edge as a relationship between them. Example- Neo4J.

B. Database Models

There are 3 basic model for database structuring[1]. However we will study about one model in a slight detailed manner-CAP

- ACID: ACID stands for Atomicity, Consistency, Isolation and Durability. Introduced by Jim Gray in 1970s, this model was set to ensure reliability of the database that incorporated it. Atomicity refers to either all or none property. In other words in a database, either all transactions are successful or none. Consistency maintains the integrity of the database system and ensures that none of the transactions are partially successful. The state of the database would remain same at the start as well as at the end of the transaction. Isolation says that even all the transactions running

Revised Manuscript Received on March 10, 2019.

Rupali Kaur, Computer Science, Lovely Professional University, Phagwara, India.

Jaspreet Kaur Sahiwal, Computer Science, Lovely Professional University, Phagwara, India.

together, each transaction should behave independent of the other ones as if they are the only one been executed at that time. Durability ensures the availability of the transactions i.e., even in the event of the system or hardware or any kind of failure, the result should remain the same.

- CAP: CAP stands for Consistency, Availability and Partition-tolerance. CAP theorem was introduced in 2000 by Eric Brewer[5]. It stated the three very important components. Consistency, as described by the ACID model, is the same in this model as well. Availability ensures the availability of the data all the time. It is equivalent to the Durability from ACID model. It ensures the availability of the database system all the time. Partition-tolerance linked to the availability, this property means that a system can be divided into numerous partitions and is stable even after that. Partitions are done in order to make the system available in the event of failure, making the system fault tolerant as well. The partitions so made can be local and remote also. CAP is widely used model for database structuring, including Amazon and Azure[6].

This theorem can be best described by CAP triangle as shown in figure 1. The three vertices refer to the CAP properties C, A and P. The database system can be seen as a straight line between two vertices. Since a line from a triangle can formed using only two vertices, similarly from a CAP theorem only two of the properties can be incorporated and the third one have to been traded-off.

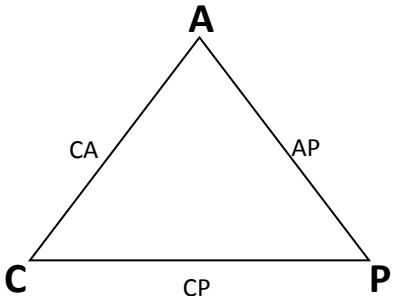

Figure 1: CAP theorem

Table I. Trade-off between CAP theorem

S.No.	Trade-off	Result	Description
1.	CA	Non-partitioned	Ensures availability and consistency
2.	AP	Partitioned	System is available and partitioned but is not consistence in terms of writes
3.	CP	Partitioned	System is partitioned and consistence but only with read access so as to compromise the loss due to Availability trade off.

However, even being used by large organizations, CAP theorem had few problems. The major problem was that due to trade-off, three kinds of distributed systems arose and there were hardly any differences between CA and CP [1].

Based on the triangle, there arise three scenarios as described in Table I.

- BASE: Full formed as Basically available, Soft state and Eventually consistent, BASE is a flexible form of ACID model to counter the issues faced by the latter. It was introduced as the technology was migrating towards NoSQL approach. Basically available feature says that if the system or any component of the system fails, the system will surely be able to give a response but that response may or may not be able to revert the consistent data surely. Implies, the basic features of the system will be available all the times to the users. Soft state property says that the system will always be in the soft state i.e., while the system is being updated, the transactions will still proceed. The system may or may not be in the static state always, so there are background changes going on due to eventual consistency, leaving the system in soft state. Eventually consistent states that sooner or later all the changes will be propagated to all the partitions (or replicas) and the system will become system. Rather like the ACID model, BASE does not wait for the system to be consistent after every transaction but ensures that it will be at some point of time, making the need of strict consistency flexible.

C. Issues with NoSQL Databases

Although NoSQL is fast growing database system, they have new world issues, just like every coin has two sides [7]. Some of the security issues [8] are discussed below:

- Grey area: There are very lesser known facts about NoSQL databases till date. Even though many large organizations make use of these not so newly introduced databases, there are some facts that remain unexplored. Thinking about the scalability feature of NoSQL, there some wrong notions about it. The actual truth is that on one side the small organizations can make use of this technology but on the other side the large organizations making use of NoSQL may still feel the need of deploying SQL. Until these areas remain unexplored, efficient use NoSQL technology cannot be made.
- Security: Despite the use of efficient structuring model, the list for the security problems with NoSQL database is fairly lengthy [9]. As providing security to any system is incredibly difficult, security remains one of the biggest challenges of NoSQL [10]. Especially in case of MongoDB, there is no in-built security till date, all is user incorporated. Programmer needs to create as many as possible doors to secure a system but an intruder or hacker needs only one key to one of these doors to penetrate into the system. That one chance can compromise the sensitive data. There have been many attempts [11] [12] to secure data of NoSQL like using AES [13] techniques but still security is a challenge for NoSQL

- Consistency: Problems with CAP model has already been discussed previously. As majority of the NoSQL databases rely on this model, it can already be concluded that the main problem remains consistency. Instead of ACID transactions, these databases follow the concept of eventually consistent. This might provide high performance but adds the problem of synchronization between nodes. While the data on one node may some read/write results, the data on some other node may show completely different result.
- Scalability: NoSQL is popularly known for its highly scalable feature. However, scalability can also become a hurdle on the path to performance. Scalability is highly dependent on sharding process. Sharding refers to splitting a database logically or physically and distributing them among nodes, over a network. Each split is referred to as a shard. If the sharding is automated, there is no problem in scaling the database. But, as not all NoSQL databases support automated sharding, scaling up or down automatically in these scenarios becomes a problem and has to be done manually.
- Inexperience: Even though NoSQL is few years old, it is in its infant years. Not lot organizations have come across it still. Due to inexperience, many projects using SQL had cost the clients way more than it would have if they incorporated NoSQL. A NoSQL code demanding project, if written using poor knowledge and experience of it, will of course cost the client a lot. NoSQL developers need to evolve in order to avoid such issues..

In spite of rapid development of these databases, the performance comparison between them is not yet clear. As of today, 225+ NoSQL databases are available. And all of them have different implementation, storage facilities, configurations and optimization techniques, which makes selection of NoSQL database more challenging. In this paper, firstly we have compared most widely used NoSQL databases in brief and then we try to compare two Document based databases (MongoDB and CouchDB) in detail.

II. RELATED STUDY

With the advancement of Big Data and IoT applications, use of NoSQL is increasing rapidly. Also the research in this field is increasing rapidly. For example, [4] describes the main comparisons between five major NoSQL databases: Redis (Key value store), MongoDB (Document value store), CouchDB (Document value store), Cassandra (Column family store) and HBase (Column family store). Two experiments were conducted in this research and the result was analyzed on the basis of two parameters: (1) Data loading (2) Workloads execution. [4] Concluded that Redis is suited for loading and executing workloads but not when faced extremely large amount of data. Document and column-family databases, showed average performance. It was also found that master-master mode was more advantageous over master-slave architectures. Similarly [14] gave comparison between all four types of NoSQL databases on the basis of functional and no-functional features and also on the basis of distributive properties.

In another work [15], MongoDB, CouchDB and Cassandra were analyzed and compared on the basis of quantitative measures (under different conditions of workload and different datasets). It was observed that under different circumstances and different application requirements a different database is suitable, so a no particular better database was given.

In some other works, as in, [16] has reviewed the comparisons between databases on both qualitative and quantitative measures and obtained same evaluation that different database is to be used in different scenarios. [17] Outlined Google's Big Table, Amazon's Dynamo and Apache's Cassandra in detail by reviewing the top scientific publications ranging between 2010 and 2016. It first gives an overview of above mentioned approaches and how these big organizations are handling Big data using NoSQL instead of traditional databases. He compared all above approaches on the basis of Database Applicability, System Performance, Scalability, Availability and Data operation and gave results accordingly [18].

In [19] a functional as well as performable comparison between different techniques of capturing changed data have been presented. Techniques like Audit Column, Snapshot Differential, Trigger Based and Column-Family scan. In this also it was observed that it is difficult to say that which of the available techniques is the best. So it is basically at the end of the developer, the technique he wants to used, taking into consideration the requirements of the application and also the limitations and performance of the available techniques should be taken into account. Other works like [20] and [21] compare the similar types of NoSQL databases, Graph based and Document based respectively.

III. COMPARISON OF NOSQL DATABSES

Very close comparative analysis of various NoSQL databases has been provided in [15] on the bases of quantitative characteristics and it was observed that different kind of NoSQL database is suitable for different kind of application requirement. In another article [14] comparison between all types of NoSQL databases (key based, column based, document based and graph based) have been done on the basis of functional features, non-functional features and distributive properties. These all are combined under Table

II

Table II: Differences between NoSQL databases

S. No.	Parameters	Key Value store	Document Store	Column store	Graph store
1.	Query performance	High	High	High	Variable
2.	Scalability	High	Variable (High)	High	Variable
3.	Flexibility	High	High	Moderate	High
4.	Structure	Primary key with some value	JSON in form of tree	row consisting multiple columns	Graph – entities and relation
5.	Complexity	None	Low	Low	High
6.	Denormalization	Applicable	NA	Applicable	Applicable
7.	Single Aggregate	Applicable	Applicable	Applicable	NA
8.	Atomicity	Applicable	Applicable	Applicable	NA
9.	Unordered Keys	Applicable	NA	NA	NA
10.	Derived Table	NA	NA	Applicable	NA
11.	Composite Key	NA	NA	Applicable	NA
12.	Composite Aggregation	Applicable (Ordered)	NA	Applicable	NA
13.	Aggregation	Applicable	Applicable	Applicable	NA
14.	Aggregation and Group By	Applicable	Applicable	NA	NA
15.	Adjacency Lists	Applicable	Applicable	NA	NA
16.	Nested Sets	Applicable	Applicable	NA	NA
17.	Joins	NA	NA	NA	NA
18.	Sharding and Partitioning	Auto sharding and no order	Built in and order preserving	Auto sharding and no order	Supports sharding
19.	Scaling	Horizontal	Horizontal	Horizontal	Horizontal
20.	Replication	Relaxed Master Slave	Relaxed Master Slave	Selectable Replication Factor	Causal Clustering using Raft protocol (master slave)

Table II combines the comparisons based on different parameters given by [14]. The comparison is based on functional, non-functional features and distributive properties. The functional features include denormalization, single aggregate, atomicity, unordered keys, derived table, composite keys, composite and aggregation, aggregation, aggregation and grouping, adjacency lists, nested sets and joins. Similarly, distributive features include sharding and partitioning; scaling and replicationThe non-functional features include query performance, data scalability, schema flexibility, database structure and value complexity. It can be seen that all the databases behave differently under different conditions and circumstances.

IV. OVERVIEW OF MONGODB AND COUCHDB

MongoDB and CouchDB both are type of Document based NoSQL database. Document database is also called document store and they are usually used to store the document format of the semi-structured data and detailed description of it. It allows the creation and updation of programs without the need of referring to the master schema. Content management and handling of data in mobile application are two of the fields where document store can be applied. Other than MongoDB and CouchDB, other examples of this database include DocumentDb, Couchbase server and MarkLogic. [21] has already given the comparison between MongoDB and CouchDB (both of them are Document stores) on the basis of the performance. In this section we will be giving the comparisons between MongoDB and CouchDB on the basis of few parameters.

1. MongoDB**:** MongoDB was startup of 10gen, originated in 2007. Coming from the family of Document stores, it is one of the typical NoSQl, schema-free databases with comparatively high performance, scalability and is rich in data processing functions. This open source database is written in C++ and makes use of dynamic schemas. The architecture of MongoDB contains documents grouped into collections on the basis of their structure. This database makes use of BSON. BSON is the binary representation of JSON and supports document storage and data interchange. In MongoDB business subjects can be stored in minimum number of documents, which can be indexed primarily or secondarily, without breaking them into multiple relational ones.

Along with the above mentioned capabilities of MongoDB, it also provides with the large replica sets collection where each set can contain more than one copy of data. In the replica sets, all primary functions (read and write) are performed on primary set while secondary sets are used in case of failure of former one. MongoDB incorporates sharding which makes use of scaling process horizontally.

The load balancing property of this document store database is justified by the fact that it runs on multiple servers, thereby providing duplication of data and balancing of load. This in return also provides backup during the hardware failure. It also make use of grid file system which divides the particular file into different parts and stores them separately.

Following are some of the common features of MongoDB:

- Convenience of designing the data model which reduces the need of joins and provides easy evolution of schema.
- High performance, as it contains neither join nor transactions which provide fast accessing and hence performance is increased.
- High availability due to incorporation of replica sets which provides backup during failures and also is highly robust.
- Ease in scalabilty. The sharding property of MongoDB enables it to perform fast and in efficient manner in the distributed functions. This also possible due to the fact that in supports horizontal scaling of data.
- Language highly rich in query. MongoDB has its own query language called Mongo query language which can replace SQL ones. Similarly, utility functions and map or reduce can replace complicated aggregate functions.

The architecture of MongoDB contains, (i) client applications, (ii) drivers, (iii) DBMS Mongod, (iv) data base routing programs, and (v) data.

MongoDB is currently managed by Inc. MongoDB. Some companies incorporating MongoDB are Adobe, BBVA, CERN, Department of Veteran Affairs, Electronic Arts, Forbes, Under Armour.

2. CouchDB: CouchDB, an Apache Software Foundation Product and inspired by Lotus Notes, is also an open source document based NoSQL database which focus mainly on easy use. It is a single noded database, working exactly like other databases. It generally starts with the single node instance but can be seamlessly upgraded to cluster. It allows the user to run single database on many servers or VMs. A CouchDB cluster is provides high capacity and availability as compared to single node CouchDB. It uses Erlang, a general purpose language. Like MongoDB, it also uses java script and map/reduce. It stores data in the form of collection of documents rather than as tables. The updated CouchDB is lockless which means, there is no need lock the database during writes. The documents in this database also make use of HTTP protocol and JSON, along with the ability to attach non-JSON files to them. So, CouchDB is compatible with any application or software that supports JSON format.

REST API is used to write and query the data. It also offers document read, add, edit and delete. According to article [21], it uses the ACID model rather than BASE by MVCC implementation. Just like MongoDB, supports replication of devices when they are offline. It uses special replication model called Eventual Consistency. CouchDB is highly and seriously reliable in terms of data. Single-node database make use of append-only crash-resistant data structure and multimode or cluster database can save the data redundantly so that it can be made available whenever the user needs it. CouchDB can be scaled along as big clusters as global clusters to as small ones as mobile devices. The ability to run on any Android or iOS devices makes CouchDB to stand out among other databases.

Coming to the CouchDB architecture, it is distributed which supports bidirectional synchronization. It does not require any schema as it makes use of unique id.

Although CouchDB follows AP (availability and partition tolerant) feature of the CAP model, to overcome the traded consistency, it follows ACID model on the practical basis.

CouchDB is still managed by its founder organization, Apache Software Foundation. Some of the companies that incorporate this database are Talend SA, Akami Technologies, Hothead

Games, Inc., GenCorp Technologies, Vivint Solar Inc.

3. Comparison between MongoDB and CouchDB: As already mentioned earlier, MongoDB and CouchDB both are document store database, so rather than differences, they both share similarities more. Some of the similarities include the supported languages, indexing and sharding. The difference between them is very less but still they both can be compared on some parameters. Both of these databases were surely built with a different focus but very slight one at that. Though both can be scaled easily across multiple nodes but where MongoDB supports consistency, CouchDB supports availability. Both incorporates the use of replica set but in a different way. In MongoDB these sets provide strict consistency which implies that the nodes are divided into two types: primary and secondary. Primary ones are used to write and read the operations while secondary ones are used to provide redundancy in case of hardware failure. But in CouchDB, eventual consistency is incorporated where the

functions can be performed on the nodes without the agreement of other nodes and futher, it copies the changes made between the two nodes in the document incrementally so that they are synced continuously. The detailed comparison between both the databases on the basis of querying is explained as under:

MongoDB is generally preferred due to its SQL-like syntax of the queries rather than mapreduce. Also in the scenario of dynamic queries same is preferred over the other one. But if MongoDB being used make use of Mongoose driver, it will introduce the constraint of same schema whereas this is not present in the case of Couch DB. This constraint can be easily avoided if some other MongoDB Node.js driver is used.

Also, in case of CouchDB for quering purposes an index is needed which is stored using map function and then the query can be run using cURL whereas in MongoDB no such procession is required and the database can be queried simply using db.<database_name>.<query>({}).

Table III compares the databases on few parameters namely languages used while developing both databases; languages supported by both; supported platforms; data structure of storage; cap features; replication; map reduce functions of both databases; indexes; query format; sharding support and license. The data structure of both databases have already been discussed earlier in this article. Coming to the replication, MongoDB uses single master replication with built-in auto-election. On the other hand, CouchDB offers both master-master and master-slave replication low latency in accessing the data independent of its location. In case of indexes, both MongoDB and CouchDB make use of B-Tree index structure.

Table III: Result & Comparison between MongoDB and CouchDB

S. No.	Features	MongoDB	CouchDB
1.	Developmental Languages	C++, Javascript	Erlang
2.	Supported	Actionscript, C,	C, C#,.
	Languages	C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala	ColdFusion, Erlang, Haskell, Java, JavaScript, Lisp, Lua, Objective-C, OCaml, Perl, PHP, PL/SQL, Python, Ruby, and Smalltalk
3.	Deployment Platforms	Linux, OS X, Solaris, and Windows.	Android, BSD, iOS, Linux, OS X, Solaris, and Windows.
4.	Database Structure	BSON	JSON
5.	CAP Features	Consistence and Partition tolerant (CP)	Availability and Partition tolerant (AP)
6.	Replication	Master-Slave	Master-Master
7.	Map Reduce	Supports; Using Javascript	Supports; Using HTTP and REST API
8.	Query Format	db.<database_name>.<query>({})	curl-X GET http://<IP_address><database_name><query>
9.	Mobile Support	No Mobile Support	Support Android as well as iOS

V. CONCLUSION

In this article we have compared two document based NoSQL databases- MongoDB and Couch DB. Table III gives an overview of the main parametric comparisons between these two databases. As we have seen, priority of the project will determine the selection of the system. Major differences include the replication method and the platform support. Also, from the comparisons it is clear that if application requires more efficiency and speed, then MongoDB is better choice rather than CouchDB. If the user needs to run his database on mobile and also needs multi master replication than CouchDB is obvious choice. Also MongoDB is suited better than CouchDB if the database is growing rapidly. The main advantage of using CouchDB is that it is supported o mobile devices (Android and iOS) rather unlike MongoDB. So basically, different application requirements will require different database based on scenarios. We have observed that MongoDB is slightly better than CouchDB as it uses SQL-like structure of querying and the same is easier in the former one. Also, for using dynamic queries, MongoDB is far better choice. Regarding security in both databases, research is still going on and it is hard to say which of these provides better and secure environment.

REFRENCES

[1] I. Mapanga and P. Kadebu, “Database Management Systems : A NoSQL Analysis,” no. 7, 2013.

[2] D. G. Chandra and D. G. Chandra, “BASE Analysis of NoSQL Database,” Futur. Gener. Comput. Syst., 2015.

[3] S. Rautmare, “MySQL and NoSQL database comparison for IoT application,” pp. 235–238, 2016.

[4] Y. Fan, “Performance Comparison between Five NoSQL Databases,” pp. 117–121, 2016.

[5] S. Gilbert and N. Lynch, “Brewer ’ s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services,” pp. 51–59, 2005.

[6] S. Benefico et al., “Evaluation of the CAP Properties on Amazon SimpleDB and Windows Azure Table Storage,” 2012.

[7] L. Okman, N. Gal-oz, Y. Gonen, E. Gudes, and A. Cassandra, “Security Issues in NoSQL Databases,” 2011.

[8] A. Ron, A. Shulman-peleg, and A. P. Ibm, “Analysis and Mitigation of NoSQL Injections,” 2016.

[9] J. Kumar, M. T. Scholars, and V. Garg, "SECURITY ANALYSIS OF UNSTRUCTURED DATA IN NOSQL," pp. 300–305, 2017.
[10] A. Ron, B. Sheba, A. Shulman-peleg, B. Sheba, and E. Bronshtein, "No SQL, No Injection? Examining NoSQL Security."
[11] M. Shih and J. M. Chang, "Design and Analysis of High Performance Crypt-NoSQL," pp. 52–59.
[12] M. Ahmadian, F. Plochan, Z. Roessler, and D. C. Marinescu, "International Journal of Information Management SecureNoSQL : An approach for secure search of encrypted NoSQL databases in the public cloud ☪," Int. J. Inf. Manage., vol. 37, no. 2, pp. 63–74, 2017.
[13] A. D. P. Jr and A. C. Fabregas, "A Secured and Optimized Document Management tool using Advanced Encryption Standard and NoSQL," pp. 167–170.
[14] A. Gupta, S. Tyagi, N. Panwar, and S. Sachdeva, "NoSQL Databases : Critical Analysis and Comparison," pp. 293–299, 2017.
[15] S. N. Swaminathan, "Quantitative Analysis of Scalable NoSQL Databases," 2016.
[16] B. G. Tudorica and C. Bucur, "A comparison between several NoSQL databases with comments and notes."
[17] S. Kalid, A. Syed, A. Mohammad, and M. N. Halgamuge, "Big-Data NoSQL Databases : A Comparison and Analysis of " Big-Table "," pp. 89–93, 2017.
[18] R. Sheth, "Encrypting Data of MongoDB at Application Level," vol. 10, no. 5, pp. 1199–1205, 2017.
[19] F. M. Schmidt, C. Geyer, and A. Schaeffer-filho, "Change Data Capture in NoSQL Databases : A Functional and Performance Comparison," pp. 562–567, 2015.
[20] G. A. P. E, O. S. Pabón, and E. D. E. L. Arte, "A comparison of NoSQL Graph Databases," 2014.
[21] S. K. K. B and S. Mohanavalli, "A Performance Comparison of Document Oriented NoSQL Databases," 2017.