WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Examining and juxtaposing the two NoSQL databases MongoDB and CouchDB

***Shahida B***[a]

[a]*CSE department, PDIT, Hospet, India*

*DOI: https://doi.org/10.55248/gengpi.2022.3.9.28*

A B S T R A C T

Relational databases have been an essential component of technology for many years. It has been, and in some cases, still is, the foundation of both large and small organizations. However, a few flaws in relational databases gave rise to NoSQL databases. NoSQL is not a particularly new technology, but it is one that is rapidly expanding. The structured query language is also known as NoSQL. Today, a variety of NoSQL databases are accessible depending on the user's needs. This review combines numerous comparisons of these databases based on various attributes. A few terms relating to NoSQL databases are also clarified and brought to light. Additionally, a detailed comparison of the two most popular NoSQL databases is evaluated.NoSQL is a component of a very narrow ecosystem with fewer applications than relational databases. Relational databases are fairly straightforward, which makes them easy to use, but the non-uniformity of the expanding data causes an issue with them. Organizations manage millions of pieces of data, which makes using a relational database challenging. Additionally, the IoT is right outside our door, and NoSQL is better for IoT applications. NoSQL was developed to address this non-uniformity and the rapidly expanding data sets. NoSQL is mostly utilized in distributed, cloud, and IoT systems. NoSQL databases eschew rigid schemas and prioritize availability, scalability, and fault tolerance as key properties.

Keywords:NoSQL database, MongoDB, CouchDB, Column Database, Key-value Database, Big Data, Big Data Analytics, RDBMS, and NoSQL

## 1. Introduction

NoSQL databases, as their name suggests, support unstructured queries without a schema. The different database structural paradigms, including ACID, CAP, and BASE, are described in this article [1]. The BASE model technique is used by NoSQL. Basically Available, Soft State, and Eventually Consistent are often referred to as BASE. According to [1], the common challenges that affect both relational and NoSQL databases include security concerns, (ii) scalability restrictions, and (iii) issues with data availability. On the basis of the BASE properties, the following negative aspects of NoSQL have been described in another article [2]: Not as general as SQL, many NoSQL databases provide various functions. not as powerful and expressive as SQL; not as reliable because they are still relatively new.NoSQL is a component of a very narrow ecosystem with fewer applications than relational databases. Relational databases are fairly straightforward, which makes them easy to use, but the non-uniformity of the expanding data causes an issue with them. Organizations manage millions of pieces of data, which makes using a relational database challenging. Additionally, the IoT is right outside our door, and NoSQL is better for IoT applications. NoSQL was developed to address this non-uniformity and the rapidly expanding data sets. NoSQL is mostly utilized in distributed, cloud, and IoT systems. NoSQL databases eschew rigid schemas and prioritize availability, scalability, and fault tolerance as key properties. The performance of MYSQL and MongoDB (a NoSQL database) for IoT applications was compared by [3], and it was revealed that in some situations MYSQL is superior to MongoDB, while in others the reverse is true. As a result, according to [3,] the best database for IoT will be based on the most common queries and application requirements.Weak consistency was cited as NoSQL's main obstacle in another survey [4], but as NoSQL has grown in sophistication and acceptance, additional performance improvements and capabilities have been added, along with an updated

* *Corresponding author.* Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: shahida@pdit.ac.in

iteration version. Types of NoSQL databases Four main categories exist: A hash table containing keys and values is contained in a key-value store. A single value is assigned to each key in this straightforward database, which uses an associative array as its primary data architecture. Example: Amazon S3 with Riak (Dynamo). It holds tagged elements in the form of documents in the document-based store. The data model is left unstructured, resulting in a variety of schemas, as opposed to being organized into rows and columns (table format), as is done in traditional databases. In turn, this gives data modeling additional freedom.

CouchDB is an example, a column-based store continually stores each column on a disk or in memory, with the left column being stored in blocks that are sequential. It only holds one column of data in the storage block. Example: Cassandra, the HBase Graph-based databases use nodes and edges to store and represent data, with nodes standing in for individual entities and edges for the connections between them. As an example, consider database models.There are three fundamental models for constructing databases[1–3]. But we'll focus on one model, CAP, in a little more detail. Atomicity, Consistency, Isolation, and Durability are acronyms for ACID. This concept, which Jim Gray first introduced in the 1970s, was created to guarantee the validity of the database it was integrated into. Atomicity can refer to any property or none at all. In other words, either every transaction in a database is successful, or none of them is. Consistency ensures that no transaction is partially successful and upholds the integrity of the database system. The database would be in the same condition at the beginning and end of the transaction. According to isolation, even when all transactions are active at the same time, each one should act independently of the others as if they were the only ones being carried out at that moment. Durability guarantees transaction availability, i.e., that the outcome should hold true regardless of system, hardware, or other failure. Consistency, Availability, and Partition-Tolerance, or CAP, is an acronym. Eric Brewer[5] developed the CAP theorem in 2000. It listed the three crucial elements. The ACID model's definition of consistency is the same in this model. The constant availability of the data is ensured via availability. It is the same as the ACID model's durability. It continuously ensures the database system's availability. Partition-tolerance, which is related to availability, indicates that a system may be partitioned into many parts and remain stable afterward. The creation of partitions, which can be both local and remote, enables the system to be available in the case of a failure and increases fault tolerance. Eric Brewer[5] developed the CAP theorem in 2000. It listed the three crucial elements. CAP is a methodology for database organization that is frequently utilized, including by Amazon and Azure[6]. The CAP attributes are represented by the three vertices. A straight line connecting two vertices can be used to represent the database system. Similarly, a line from a triangle can be constructed with only two vertices. The CAP theorem only allows for two of the qualities to be used, and the third one must be traded off. Each transaction ought to act independently of the others, as if they were the only ones being carried out at the moment. Durability guarantees transaction availability, i.e., that the outcome should hold true regardless of system, hardware, or other failure. Consistency, Availability, and Partition-Tolerance, or CAP, is an acronym. The ACID model's definition of consistency is the same in this model.

The constant availability of the data is ensured via availability. It is the same as the ACID model's durability. It continuously ensures the database system's availability. Partition-tolerance, which is related to availability, indicates that a system may be partitioned into many parts and remain stable afterward. Partitioning is done to make the system available in case of failure and to increase fault tolerance. The divisions created in this way might be both local and remote. CAP is a methodology for database organization that is frequently utilized, including by Amazon and Azure[6]. The CAP attributes are represented by the three vertices. A straight line connecting two vertices can be used to represent the database system. Similar to how a line from a triangle can be constructed with only two vertices, the CAP theorem only allows for two of the qualities to be used, and the third one must be traded off. Consistency: The issues with the CAP paradigm have already been covered. Since the bulk of NoSQL databases use this paradigm, it is obvious that consistency is still the biggest issue. These databases operate according to the idea of eventually consistent rather than ACID transactions. High performance could result from this, but synchronization issues between nodes are added. Data on one node may provide certain read/write results, whereas data on another node may have entirely different ones. Scalability: NoSQL is well-renowned for being incredibly scalable. Scalability, however, can often constitute a barrier to performance. Sharding is a crucial component of scalability. When a database is sharded, it is divided logically or physically and distributed among network nodes. A shard is the name given to each split. Scaling the database is not a problem if the sharding is automated. However, scaling up or down automatically in these cases becomes an issue and must be done manually because not all NoSQL databases offer automated sharding. Inexperience: Despite just being a few years old, NoSQL is still in its infancy. Few organizations have still run into it. Due to a lack of knowledge, many SQL-based initiatives ended up costing the clients much more than they would have if NoSQL had been used instead. If a NoSQL code-intensive project is built with insufficient expertise and understanding, it will undoubtedly cost the client a lot of money. For these problems to be avoided, NoSQL developers must advance. Despite these databases' quick development, it is still unclear how they compare in terms of performance. More than 225 NoSQL databases are accessible right now. Additionally, choosing a NoSQL database is made more difficult by the fact that each one has a different implementation, storage capabilities, configurations, and optimization strategies. In this work, we take a quick look at the most popular NoSQL databases and then try to compare two document-based databases, MongoDB and CouchDB, in more depth.

## 2. Review of Literature

**No-SQL Related Studies**

The use of NoSQL is quickly expanding as big data and Internet of Things (IoT) applications develop. The amount of study in this area is also growing quickly. Five popular NoSQL databases, including Redis (Key value store), MongoDB (Document value store), CouchDB (Document value store), Cassandra (Column family store), and HBase, are compared in [4], for instance (Column family store). In this study, two experiments were carried out, and the outcomes were evaluated using the following two criteria: (1) data loading, (2) work tasks being carried out. According to [4], Redis is best suited for loading and executing workloads, but not when dealing with exceptionally huge amounts of data. Databases for documents and column families performed around averagely. Additionally, it was discovered that master-slave architectures were preferable to master-slave designs. similar to [14], which provided comparisons across all four types of NoSQL databases based on distributive qualities as well as functional and non-functional elements. In a

different study [15], quantitative measurements were used to examine and compare MongoDB, CouchDB, and Cassandra (under different conditions of workload and different datasets).

A better database wasn't specifically recommended because it was discovered that several databases were suitable depending on the situation and the requirements of the application. According to some other publications, as [16], multiple databases should be employed in various contexts after reviewing comparisons between them on both qualitative and quantitative criteria. [17] reviewed the top scientific publications from 2010 to 2016 to outline Google's Big Table, Amazon's Dynamo, and Apache's Cassandra in depth. It begins by providing a summary of the aforementioned methods and how these large companies handle big data, utilizing NoSQL rather than conventional databases. On the basis of Database Applicability, System Performance, Scalability, Availability, and Data Operation, he compared all of the aforementioned methodologies and provided results in accordance [18]. A functional and practicable comparison of various methods of recording altered data has been provided in [19]. The Column-Family Scan, Snapshot Differential, Audit Column, and Trigger-Based It was noted that it is challenging to determine which of the available strategies is the best in this situation as well. In essence, this means that the developer should analyze the application's requirements as well as the performance and constraints of the various techniques before choosing the one he would use. The related forms of NoSQL databases, graph-based and document-based, are compared in other publications like [20] and [21].

**Comparing Different Types of No-SQL Databases**

On the basis of quantitative characteristics, a very detailed comparison of several NoSQL databases has been published in [15], and it was found that different types of NoSQL databases are acceptable for varied application requirements. In another paper [14], the functional aspects, non-functional features, and distributive qualities of all forms of NoSQL databases (key-based, column-based, document-based, and graph-based) have been compared d application requirements. In another paper [14], the functional aspects, non-functional features, and distributive qualities of all forms of NoSQL databases (key-based, column-based, document-based, and graph-based) have been compared [13]. All of these are discussed, which aggregates comparisons based on many criteria provided by [14]. The comparison is based on distributive qualities, non-functional features, and functional features. Denormalization, a single aggregate, atomicity, unordered keys, derived tables, composite keys, composite and aggregation, aggregation, and grouping, adjacency lists, nested sets, and joins are among the functional aspects. Similar to that, distributive features include scaling, replication, and sharding and splitting. Performance of queries, data scalability, flexibility of the schema, database structure, and value complexity are examples of non-functional aspects. It is clear that every database exhibits varied behavior depending on the situation.

**MongoDB and CouchDB**

Both CouchDB and MongoDB fall under the category of document-based NoSQL databases. Document databases, also known as document stores, are typically used to store semi-structured data in document format along with a full description of it. It enables program development and updating without the requirement to consult the master schema. One or more applications for document stores can be found in the administration of content and the handling of data in mobile applications. Other instances of this database include DocumentDb, Couchbase server, and MarkLogic, in addition to MongoDB and CouchDB. [21] has already provided a performance comparison between MongoDB and CouchDB (both of which are document stores). MongoDB and CouchDB will be compared to one another in this section based on a few different criteria. First, MongoDB Startup company 10gen, MongoDB, was founded in 2007. It is one of the standard NoSQL, schema-free databases with comparatively good performance, scalability, and is rich in data processing functions. It is a member of the family of Document stores. The dynamic schemas used in this open source database are created in C++. Documents are organized into collections in MongoDB's architecture based on their structural characteristics. This database uses the BSON format. BSON, which facilitates data exchange and document storage, is JSON's binary representation. Business subjects can be stored in MongoDB in the smallest amount of documents, which can be primarily or secondarily indexed, without being split up into numerous relational documents. Along with the features already described, MongoDB also offers a collection of big replica sets, each of which can hold multiple copies of data.

In replica sets, the primary set is used for all primary operations (read and write), whereas backup sets are employed in the event that the primary set fails. Sharding, which uses a horizontal scaling technique, is a feature of MongoDB. The fact that this document store database runs on numerous servers, allowing data duplication and load balancing, justifies its load balancing capability. In turn, this offers backup in the event of hardware failure. Additionally, it uses a grid file system, which separates a certain file into portions and saves them individually. Some of MongoDB's common characteristics include the following: Ease of building a data model that minimizes the requirement for joins and allows for simple schema evolution. High performance since it doesn't contain join or transactions, which allow for quick access, which boosts performance. High availability because replica sets are used, which offer backup in the event of faults and are also quite robust. Ease of scaling. MongoDB's sharding feature makes it possible for it to run distributed tasks quickly and effectively. This is also made possible by the fact that it allows for horizontal data scaling. Language is incredibly question-rich. The Mongo query language, developed by MongoDB, can be used in place of SQL queries. Utility functions, map, and reduce can also take the role of complex aggregate functions. The MongoDB architecture consists of I client apps, (ii) drivers, (iii) the MongoDB database management system, (iv) data base routing applications, and (v) data. Inc. MongoDB presently oversees MongoDB. Adobe, BBVA, CERN, Department of Veterans Affairs, Electronic Arts, Forbes, and Under Armour are a few businesses that use MongoDB. 2. CouchDB: Similar to Lotus Notes, CouchDB is a product of the Apache Software Foundation and is a document-based NoSQL database with a strong user-friendliness focus. It functions identically like other databases and is a single node database. Although it often starts with a single node instance, clustering is an easy upgrade [15] [18].

One database can be run on numerous servers or virtual machines (VMs). Comparatively speaking, a CouchDB cluster offers higher capacity and availability than a single node CouchDB. It employs the all-purpose language Erlang. It also makes use of Java script and map/reduce, just like MongoDB. Instead of storing data as tables, it does so as a collection of documents. The upgraded CouchDB no longer requires locking the database

during writes because it is lockless. This database's documents use the HTTP protocol, JSON, and have the option of accepting non-JSON attachments. Therefore, CouchDB works with any program or piece of software that supports the JSON format. The data is written and queried using REST API. Also available are document read, add, edit, and delete functions. It employs the ACID model rather than the BASE via MVCC implementation, according to article [21]. allows device replication even when offline, just like MongoDB. Eventual Consistency is a unique replication model that is used. Regarding data reliability, CouchDB is extremely and seriously trustworthy [16]. Append-only crash-resistant data structures are used by single-node databases, whereas multimode or cluster databases can store data redundantly so that it is always available to the user. CouchDB may be scalable from large global clusters to small mobile device clusters. When compared to other databases, CouchDB stands out due to its ability to run on any Android or iOS device. In terms of CouchDB's design, it is distributed and allows for bidirectional synchronization. Due to the use of a unique id, no schema is needed. Although CouchDB adheres to the AP (availability and partition tolerant) characteristic of the CAP model, it actually adheres to the ACID model to avoid traded consistency. The Apache Software Foundation, which founded CouchDB, continues to oversee the database. Talend SA, Akami Technologies, Hothead Games, Inc., GenCorp Technologies, and Vivint Solar Inc. are a few businesses that use this database [17] [18].

## 3. Discussion

Comparing CouchDB and MongoDB, as they are document store databases. It was already mentioned, their commonalities outweigh their differences. Supported languages, indexing, and sharding are a some of the similarities. Even though there isn't much of a difference between them, there are several parameters where they can be compared. Both of these databases undoubtedly had a different objective when they were created, albeit a very minor one. Despite the ease with which both can be scaled over numerous nodes, CouchDB supports availability whilst MongoDB supports consistency. Both incorporate it, albeit in different ways, using the replica set. These sets in MongoDB offer rigorous consistency, which implies that there are two different sorts of nodes: primary and secondary. While secondary ones are utilized to offer redundancy in the event of hardware failure, primary ones are used for writing and reading operations. However, CouchDB incorporates eventual consistency, which allows functions to be executed on nodes without the consent of other nodes. In addition, it incrementally transfers changes made between the two nodes in the document to keep them in sync. Following is an explanation of how to compare the two databases in detail based on querying: Due to its SQL-like query syntax, MongoDB is typically favored over MapReduce. Additionally, the same is chosen above the alternative in the case of dynamic queries. However, if the Mongoose driver is used with MongoDB, it will introduce the constraint of the same schema, although this is not the case with Couch DB. If another MongoDB Node.js driver is used, this constraint can be easily avoided. Additionally, in CouchDB an index must be created for querying purposes and stored using the map function before the query can be executed using cURL; whereas, in MongoDB no such step is necessary and the database may be queried directly using the db command. <database name>. <query>({}). Comparing the databases is based on a few criteria, including the languages used to develop each database, the languages that each database supports, the supported platforms, the storage data structure, the cap features, the replication, the map reduce functions of both databases, the indexes, the query format, the support for sharding, and the license. Both databases' data structures have previously been covered before in this study. Regarding replication, MongoDB has a single master replication system with integrated auto-election. On the other hand, CouchDB provides low latency access to the data regardless of its location for both master-master and master-slave replication. Both MongoDB and CouchDB use the B-Tree index structure when it comes to indexing.

## 4. Summary

We examined MongoDB and CouchDB, two document-based NoSQL databases. As it is seen, the system will be chosen based on the project's priority. The replication technique and platform support are two key distinctions. Additionally, it is evident from the comparisons that MongoDB is a superior option than CouchDB if an application requires greater efficiency and speed. CouchDB is an obvious solution if the user has to execute his database on a mobile device and also needs multi-master replication. Additionally, MongoDB is more appropriate than CouchDB if the database is expanding quickly. The key benefit of adopting CouchDB is that, unlike MongoDB, it is supported on mobile devices (Android and iOS). In other words, based on different application needs, several databases will be needed. Because MongoDB has a format for querying that is similar to SQL and is simpler in the former, we have found that it is marginally superior to CouchDB. Also, MongoDB is a far superior option when doing dynamic queries. Since research on the security of both databases is still going on, it is hard to say which one is better and safer.

## REFERENCES

[1] Abramova, V., & Bernardino, J. (2013, July). NoSQL databases: MongoDB vs Cassandra. In Proceedings of the international C* conference on computer science and software engineering (pp. 14-22).

[2] Ali, W., Shafique, M. U., Majeed, M. A., & Raza, A. (2019). Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics. Asian Journal of Research in Computer Science, 4(2), 1-10

[3] Becker, M. Y., & Sewell, P. (2004, June). Cassandra: Flexible trust management, applied to electronic health records. In Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. (pp. 139-154). IEEE.

[4] Berg, K. L., Seymour, T., & Goel, R. (2013). History of databases. International Journal of Management & Information Systems (IJMIS), 17(1), 29-36.

[5] Bjeladinovic, S., Marjanovic, Z., & Babarogic, S. (2020). A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components. Journal of Systems and Software, 168, 110633.

[6] Chandra, D. G. (2015). BASE analysis of NoSQL database. Future Generation Computer Systems, 52, 13-21.

[7] Chen, J. K., & Lee, W. Z. (2019). An introduction of NoSQL databases based on their categories and application industries. Algorithms, 12(5), 106.

[8] Cuzzocrea, A., & Shahriar, H. (2017, December). Data masking techniques for NoSQL database security: A systematic review. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 4467-4473). IEEE.

[9] de Oliveira, V. F., Pessoa, M. A. D. O., Junqueira, F., & Miyagi, P. E. (2021). SQL and NoSQL Databases in the Context of Industry 4.0. Machines, 10(1), 20.

[10] Deka, G. C. (2013). A survey of cloud database systems. It Professional, 16(2), 50-57. IEEE.

[11] Di Martino, S., Fiadone, L., Peron, A., Riccabone, A., & Vitale, V. N. (2019, June). Industrial Internet of Things: Persistence for Time Series with NoSQL Databases. In 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 340-345). IEEE.

[12] dos Santos Ferreira, G., Calil, A., & dos Santos Mello, R. (2013, December). On providing DDL support for a relational layer over a document NoSQL database. In Proceedings of International Conference on Information Integration and Web-based Applications & Services (pp. 125-132).

[13] Gessert, F., Wingerath, W., Friedrich, S., & Ritter, N. (2017). NoSQL database systems: a survey and decision guidance. Computer Science-Research and Development, 32(3), 353-365.

[14] Guimaraes, V., Hondo, F., Almeida, R., Vera, H., Holanda, M., Araujo, A., ... & Lifschitz, S. (2015, November). A study of genomic data provenance in NoSQL document-oriented database systems. In 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 1525-1531). IEEE.

[15] Rodriguez, K. M., Reddy, R. S., Barreiros, A. Q., & Zehtab, M. (2012, June). Optimizing Program Operations: Creating a Web-Based Application to Assign and Monitor Patient Outcomes, Educator Productivity and Service Reimbursement. In DIABETES (Vol. 61, pp. A631-A631). 1701 N BEAUREGARD ST, ALEXANDRIA, VA 22311-1717 USA: AMER DIABETES ASSOC.

[16] Kwon, D., Reddy, R., & Reis, I. M. (2021). ABCMETAapp: R shiny application for simulation-based estimation of mean and standard deviation for meta-analysis via approximate Bayesian computation. Research synthesis methods, 12(6), 842–848. https://doi.org/10.1002/jrsm.1505

[17] Reddy, H. B. S., Reddy, R. R. S., Jonnalagadda, R., Singh, P., & Gogineni, A. (2022). Usability Evaluation of an Unpopular Restaurant Recommender Web Application Zomato. Asian Journal of Research in Computer Science, 13(4), 12-33.

[18] Reddy, H. B. S., Reddy, R. R. S., Jonnalagadda, R., Singh, P., & Gogineni, A. (2022). Analysis of the Unexplored Security Issues Common to All Types of NoSQL Databases. Asian Journal of Research in Computer Science, 14(1), 1-12.

[19] Singh, P., Williams, K., Jonnalagadda, R., Gogineni, A., &; Reddy, R. R. (2022). International students: What's missing and what matters. Open Journal of Social Sciences, 10(02),

[20] Jonnalagadda, R., Singh, P., Gogineni, A., Reddy, R. R., & Reddy, H. B. (2022). Developing, implementing and evaluating training for online graduate teaching assistants based on Addie Model. Asian Journal of Education and Social Studies, 1-10.

[21] Sarmiento, J. M., Gogineni, A., Bernstein, J. N., Lee, C., Lineen, E. B., Pust, G. D., & Byers, P. M. (2020).Alcohol/illicit substance use in fatal motorcycle crashes. Journal of surgical research, 256, 243-250.

[22] Brown, M. E., Rizzuto, T., & Singh, P. (2019). Strategic compatibility, collaboration and collective impact for community change. Leadership & Organization Development Journal.

[23] Sprague-Jones, J., Singh, P., Rousseau, M., Counts, J., & Firman, C. (2020). The Protective Factors Survey: Establishing validity and reliability of a self-report measure of protective factors against child maltreatment. Children and Youth Services Review, 111, 104868