VAWKUM Transactions on Computer Sciences
http://vfast.org/journals/index.php/VTCS@ 2025, ISSN(e): 2308-8168, ISSN(p): 2411-6335
Volume 13, Number 1, January-June 2025 *pp:278-289*

An ACID-BASE Analysis of NoSQL Database Structuring Models for Efficient Data Management

Haresh Kumar Meghwar[1] **and Areej Fatemah Meghji**[1*]

[1]Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan

Keywords: *key-frame, document store, wide column, CAP theorem*

Journal Info:
Submitted:
June 01, 2025
Accepted:
June 11, 2025
Published:
June 22, 2025

Abstract

Due to the massive growth of data in the modern era, NoSQL databases have achieved significant popularity for their ability to scale and adapt, making them a favored option for managing extensive data in distributed systems or cloud databases. The primary goal of this research is to explore the ACID (Atomicity, Consistency, Isolation, Durability), BASE (Basically Available, Soft State, Eventual Consistency), and CAP (Consistency, Availability, and Partition tolerance) database structuring models to conduct a comparative analysis of ACID and BASE of twelve mostly used NoSQL databases. We also categorize each database with Brewer's CAP theorem. This research also compares the functional and non-functional features of databases adhering to the discussed models and explores the variety of data stores from each of the four NoSQL categories (Document, Key-value, Column, and Graph). The research summarizes the suggestions, benefits, and challenges of using NoSQL databases based on their applicability to cloud-based environments.

***Correspondence author email address:** areej.fatemah@faculty.muet.edu.pk
DOI: 10.21015/vtcs.v13i1.2167

1 Introduction

The significant growth of data in recent years has led to the foundation of NoSQL databases as an alternative to traditional relational databases for handling large volumes of data [1]. Today, NoSQL is a popular database of choice specifically designed to ensure good query performance; this is aided by its support of structured, semi-structured, and unstructured data of large, diverse, and constantly evolving nature [2]. The migration from relational databases (RDB) to NoSQL databases was mainly due to schema transformations and the storage of data within RDB in a normalized form, whereas NoSQL databases are known for their support for denormalization [3]. An issue with traditional database systems is their inability to meet the scalability, performance, and availability requirements of big data workloads and provide support to the sheer size of this data. This limitation led to

the emergence of NoSQL database systems [4]. Traditional databases can process structured, semi-structured, and unstructured data to a certain extent, but have limitations in achieving scalability and flexibility while handling large amounts of data and concurrent users. NoSQL databases can support scalability and flexibility while processing the fast-moving, large volume of data [5].

Mapanga and Kadebu explore the various database structuring models, including ACID (Atomicity, Consistency, Isolation, Durability), BASE (Basically available, soft state, Eventual Consistency), and CAP (Consistency, Availability, and Partition tolerance) [6]. They also discuss the security issues, limitations, scalability, and availability of data. ACID is native to RDB and is fully consistent with transactional updates due to centralized indexing. It works like a unit, whereas NoSQL databases are BASE compliant, which is eventually consistent and works like independent units without transactional updates. The scaling ability of NoSQL is provided by BASE as it offers enhanced performance, flexibility, and availability to its users. Recently, organizations have the choice to select models as there are several NoSQL systems that not only support BASE, but also fully conform to the ACID constraints and features [7]. Khan et al. also demonstrate that NoSQL follows BASE constraints, whereas ACID properties are supported by few NoSQL databases up to a certain extent [7], with the Neo4j graph database being the most favored option [9, 10].

Graph databases have been shown to effectively organize and store data with complex dependencies, and documents serve as the primary focus of MongoDB; it uses the BSON format, which is an offshoot of JSON [11, 12]. A research study performed a comparison between MongoDB, Cassandra, Riak, and Neo4j and summarized that all databases are schema-free and horizontally scalable, and also suggested that, except Neo4j, the others do not have complete ACID properties [13]. An ACID compliancy is presented in a research study as comparison of NoSQL versus RDBMS based on certain features such as schema, flexibility, scalability, data validity, data type, data storage, schema, and query language, thereby highlighting that generally, NoSQL supports only a single record transaction and eventual consistency replica system assuming that transactions are commutative [14]. A comparison of a Graph database and a relational database also suggests that Neo4j supports full ACID transactions and can be utilized in both standalone servers (REST interface) and embedded form [15]. Research has also been conducted on NoSQL database features and data models being used in cloud computing environments, emphasizing the power and limitations of each model, including the classification of NoSQL databases under the CAP theorem [16].

Several large companies such as Google, Amazon, Facebook, and X (Twitter) have started using the NoSQL model [17]. It needs to be noted that the use of this platform is not without its problems. Organizations need to consider several points while choosing the right model that can be suitable for their system implementation, as each model is known for their support to certain features such as Availability, Scalability, Consistency, Partition tolerance, Security, Standard query language, Performance, and Cost. This research aims to explore the structuring models and their support for four broad categories of NoSQL databases (Document, Key-value, Column family, and Graph). A comparative analysis of ACID-BASE of twelve NoSQL databases(CouchDB, Couchbase, MongoDB, DynamoDB, Redis, Riak, Cassandra, HBase, Bigtable, Neo4j, Orient DB, and Amazon Neptune). We also compare functional and non-functional features of databases adhering to ACID, BASE, and CAP models.

Section 2 presents an overview of NoSQL data models. Section three illustrates the significance, challenges, and benefits of NoSQL databases. In section 4, a comparative analysis of ACID and BASE models of twelve NoSQL databases from each category is presented in terms of supporting the ACID and BASE properties, along with CAP model description, and comparing functional and non-functional features adhering to ACID, BASE, and CAP models. Finally, in section 5, we present a conclusion and future research directions.

2 Overview of NoSQL Databases

NoSQL, which stands for Not Only SQL, was created to handle unstructured and semi-structured data. NoSQL databases are known for their scalability, availability, partition tolerance, system efficiency, and consistency, which makes them the favored option for handling large volumes of data in distributed systems or cloud databases. Many NoSQL databases are considered to be best in industrial practice [18, 19]. Large volume of data often refers to big data processes or large amounts of data that are impossible to process via traditional techniques [20]. NoSQL is a datastore that comprises four main categories of database models: Document databases, key-value databases, column family databases, and Graph databases [21].

2.1 Document-Oriented Databases

Document databases store data in documents in the form of key-value pairs and are transparent to the system. They store data in understandable formats, including XML, JSON, or BSON. Querying the database is not only key-based, but it's defined fields within the records. Some known document databases are MongoDB, Couchbase, and CouchDB.

2.2 Key-Value Databases

An Effective and simple database for managing concurrent access in which data is stored as a pair of unique keys and values. Key-value databases are also termed "hash tables". A few popular databases include DynamoDB, Redis, and Riak.

2.3 Column Family Databases

Column family databases store data in column-oriented format as a series of columns. These databases support data retrieval by column rather than by row. Every column is stored separately, and each row represents a collection of columns. Often called a sparse database for managing null values. A few prominent databases include Cassandra, HBase, and Bigtable.

2.4 Graph databases

In Graph databases, data is stored as nodes and edges. Nodes represent entities, and edges represent relationships. This type of NoSQL database in data is represented as a Graph with interlinked elements. Mostly used for social networks, maps, and data mining. Graph databases include Neo4j, Amazon Neptune, and Graph DB.

3 Significance, Benefits, and Challenges of NoSQL Databases

3.1 Significance of NoSQL

NoSQL databases can handle large volumes of data. These databases also provide a variety of features, including availability, scalability, fault-tolerance, consistency, and flexibility due to their schema-free structure. They are known for their horizontal scalability, meaning that the number of servers used by these models is increased to scale up the system rather than upgrading the hardware of the system [13, 22]. In contrast, RDBs using SQL are not suitable for big data due to their limited support for semi-structured and unstructured data. These systems also exhibit issues of scalability, availability, and flexibility due to their strict schema, limited centralized data storage, and normalized data model [23]. NoSQL databases were initially introduced as an alternative to RDB, but recently, their support for managing large amounts of data efficiently, non-uniformity of data, velocity, variety of data, and automatic recovery of systems from faults has made NoSQL much more popular than RDB.

3.2 Benefits of NoSQL

Schema-free structure: The primary benefit of NoSQL is its ability to manage unstructured data. This includes documents containing text, emails, and multimedia files. NoSQL databases are schema-free; there is no need to

define a fixed structure for storing the data. This provides flexibility to store unstructured and semi-structured data and also satisfies the non-uniformity of data [2, 13].

Open Source and Low Cost: Most NoSQL databases are open source due to cheap server expansion, cluster expandability, virtual machines as commodity servers, and the addition of new nodes to provide storage at low cost for those users that can't afford expensive database models.

Scalability and Availability: NoSQL databases are horizontally scalable, meaning multiple servers and virtual machines are combined to provide higher scalability rather than purchasing a single expensive server. Virtual machines like commodity servers can be added or removed at any time without influencing the performance of the database, and provide continued access to all users even in the case of failure. High availability and horizontal scaling features make NoSQL databases suitable for cloud computing applications [24].

Large Volume and Support for Variety of Data: NoSQL databases are known for their support of large amounts of variety of data, making them a favorable option for extensive data applications. NoSQL, also known as non-relational, offers more functionalities for managing various formats of data compared to traditional relational databases [25].

3.3 Challenges of NoSQL

Lacks Standard Query Language: NoSQL databases represent data in diverse models and use a variety of query languages, unlike traditional databases [26]. This makes it difficult for developers to understand all implementations of NoSQL. Multiple ways of querying NoSQL limit the number of queries supported; each implementation must provide its unique queries [27].

Poor Security and Lack of Consistency: NoSQL databases suffer from poor security, making them more vulnerable to attacks. Enhancing the security measures becomes crucial when multiple unidentified entities are participating in the transactions over the network, however, it provides limited security as data at rest is not encrypted and can be attacked by scanning known port numbers [22, 27]. NoSQL databases compromise on consistency when storing data. This lack of consistency due to the distributed nature of the data is a big limitation for companies and organizations that require strong consistency.

4 Comparative Analysis of Acid, Base, and CAP Model of NoSQL

ACID is native to traditional relational databases, but several NoSQL databases support full ACID constraints. On the other hand, the BASE (Basically available, soft state, and eventually consistent) model is native to the NoSQL databases due to the distributed nature of data. In this context, we have taken twelve databases from each of the four categories of NoSQL to conduct a comparative analysis of ACID, BASE, and CAP theorem properties.

4.1 ACID

The properties of atomicity, consistency, isolation, and durability offer reliable processing of database transactions, and they enforce strong consistency in partitioned databases. However, for achieving high availability and speed, a weaker form of consistency known as eventual consistency (asynchronous transactions) is used.

Atomicity means that either all transactions succeed or no transaction goes through. It guarantees the completeness of transactions. Consistency ensures that data stays consistent and valid according to the defined rules. It provides stability of data in the database. Isolation ensures that all transactions are in isolation, which means no transaction will have access to or effect on other transactions in an intermediary state. Each transaction is independent of the other. Durability means that once the transaction is committed, it will persist in the system even if the system breaks down due to power loss or crashes. Committed transactions will survive and do not change the state even in the case of the occurrence of failure. These concepts have also been depicted in Figure 1

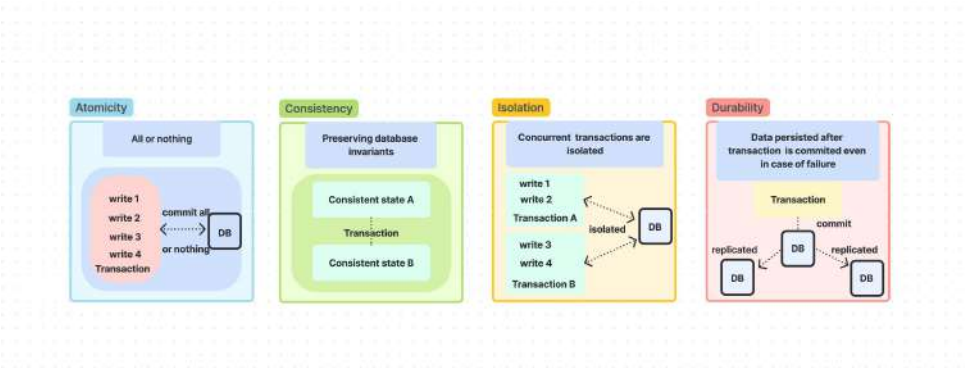

Figure 1. ACID

Table 1. ACID Analysis of Document Databases (COUCHDB, COUCHBASE, AND MONGODB)

ACID Properties	COUCHDB	COUCHBASE	MONGODB
Atomicity	Limited Atomicity (Doc level)	Limited Atomicity (Doc level)	Support Atomicity (Doc level)
Consistency	Eventual consistency by default - Strong consistency when requested	Eventual consistency (by default) - Strong consistency (when requested)	Strong consistency by default - Eventual consistency when requested
Isolation	Limited Isolation (Doc level)	Support Isolation (Doc level)	Complete Isolation (Doc level)
Durability	Durable (Recorded in JSON on disk)	Durable (Recorded on disk and replicated to other nodes)	Highly Durable (Recorded in BSON Write ahead log on disk)

NoSQL databases are known for their ability to support large amounts of data. Scalability and flexibility make them the favored option for big data industries. There are more than 225 NoSQL databases at present [28]. NoSQL databases are mostly used nowadays and support ACID constraints to varying degrees depending on the configuration and deployment of the database. Table 1, Table 2, Table 3, and Table 4 illustrate the ACID properties of each four categories of NoSQL databases.

Analyzing Table 1 to Table 4, we can see that each NoSQL database supports ACID constraints to varying degrees depending on the applicability of the particular database. MongoDB at the document level, DynamoDB at an item level, Big Table at a row level, and Neo4j at a transaction level have shown good support for atomicity. All NoSQL databases mostly support eventual consistency, however, MongoDB, Neo4j, and OrientDB prioritize offering strong consistency and eventual consistency when required. NoSQL databases offer mostly limited isolation; however, MongoDB offers complete document-level isolation, whereas DynamoDB offers serializable isolation,

Table 2. ACID Analysis of Key-Value Databases (DYNAMODB, REDIS, AND RIAK)

ACID Properties	DYNAMODB	REDIS	RIAK
Atomicity	Support Atomicity (Item level)	Limited Atomicity (Command level)	Limited Atomicity (Key level)
Consistency	Eventual consistency by default - Strong consistency when requested	Eventual consistency only (by default)	Eventual consistency by default - Strong consistency when requested
Isolation	Serializable Isolation (Item level)	Limited Isolation (Command level)	Limited Isolation (Object-key level
Durability	Highly Durable (Recorded in multiple Availability zones AZs)	Durable (Recorded in memory and persisted data asynchronously on disk)	Durable (Recorded in a cluster across v-nodes)

Table 3. Acid Analysis of Column Family Databases (CASSANDRA, HBASE, and BIG TABLE)

ACID Properties	CASSANDRA	HBASE	BIG TABLE
Atomicity	Limited Atomicity (Row level)	Limited Atomicity (Row level)	Well defined Atomicity (Row level)
Consistency	Eventual consistency(by default) - Strong consistency (when requested)	Eventual consistency (by default) - Strong consistency (when requested)	Eventual consistency (by default) - Strong consistency(when requested)
Isolation	Support Isolation (Full Row level)	Limited Isolation (Row level)	Limited Isolation (Row level)
Durability	Durable (Recorded in commit log on disk)	Highly Durable (Recorded in HDFS)	Highly Durable (Recorded in replicas)

Table 4. Acid Analysis Of GRAPH DATABASES (NEO4J, ORIENTDB, and AMAZON NEPTUNE)

ACID Properties	NEO4J	ORIENTDB	AMAZON NEPTUNE
Atomicity	Support Atomicity (Transaction level)	Support Atomicity (Transaction level)	Limited Atomicity (Transaction level)
Consistency	Strong consistency only (by default)	Eventual consistency (by default) and Strong consistency(when requested)	Eventual consistency (by default) and Strong consistency (when requested)
Isolation	Reasonable Isolation (Transaction level)	Support Isolation (Transaction level)	Limited Isolation (Transaction level)
Durability	Fully Durable (Recorded in committed transactions on disk or SSDs)	Durable (Recorded in committed transactions on disk)	Durable (Recorded in multiple Availability zones AZs)

and Neo4j offers reasonable isolation. Durability is supported by NoSQL databases to varying extents. It can be seen that CouchDB, Couchbase, Redis, Riak, Cassandra, OrientDB, and Amazon Neptune have offered durability at the basic level, however, Neo4j is fully durable. Based on the analysis from the tables, it can be seen that MongoDB and Neo4j offer the best support to ACID constraints when compared to others.

4.2 BASE

NoSQL databases are BASE compliant and focus on high availability, which differentiates them from ACID. Basically Available means that the system ensures some level of availability of data even in the case of node failure. As depicted in Figure 2, Soft State means that the system ensures data is in varying states over time and not necessarily consistent. Eventually Consistency means that the system ensures that the data is eventually consistent in the future, but not immediately at every transaction or every moment. As BASE is native to NoSQL, its properties are mostly supported by NoSQL. The BASE's primary motivation is its ongoing availability, so it mainly focuses on the availability of data rather than consistency. The BASE fulfills the need to scale up the increasing traffic over the web and makes it possible to run on large clusters, which relational databases were unable to do [7, 24]. The BASE properties of various databases have been presented in Table 5.

From Table 5, it can be seen that NoSQL databases support BASE constraints to varying extents. It should be noted that the soft state property has not been discussed in Table 5, as all NoSQL databases support the soft state property because of their ability to tolerate inconsistent data that eventually converges to a consistent state. Most of the NoSQL databases are eventually consistent, however, fewer support strong consistency as well. MongoDB, HBase, Big Table, and Redis offer basic availability, focusing on consistency. On the other hand, Neo4j, OrientDB, and Amazon Neptune support high availability as well as consistency. Couchbase, CouchDB, Cassandra, DynamoDB, and Riak offer high availability while sacrificing consistency.

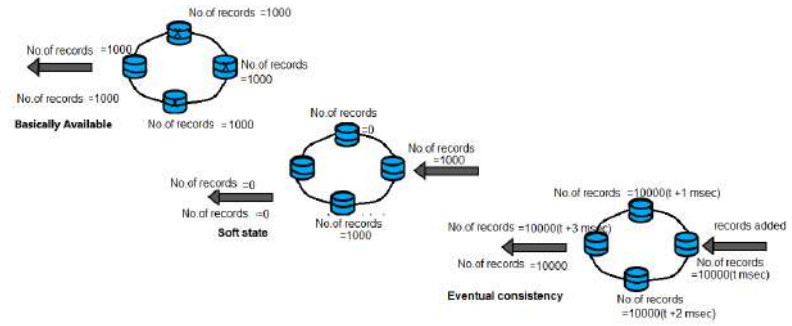

Figure 2. BASE

4.3 CAP Theorem

As depicted in Figure 3, CAP is an abbreviation of Consistency, Availability, and Partition tolerance. This theorem, put forth by Eric Brewer in 2000, states that a distributed system can exhibit only two of these three properties at any given time. Developers need to select two of the most important among them as per their needs, while compromising on the third property. The primary goal is to maximize consistency and availability for specific applications [7, 16]. Figure 3 categorizes NoSQL databases as either supporting AP (Availability and Partition tolerance) at the cost of consistency, CP (Consistency and Partition tolerance) sacrificing availability, or CA (Consistency and Availability) at the cost of partition tolerance. Couchbase, Cassandra, CouchDB, DynamoDB, and Riak reside on the AP side of the CAP theorem, whereas MongoDB, HBase, Big Table, and Redis lie on the CP side of the CAP theorem. Neo4j and RDBMS reside on the CA side of the CAP theorem. OrinetDB and Amazon Neptune support primarily CA and partially AP, depending on the cluster configuration. CAP is a popular model for organizing databases, also utilized by Amazon, Azure, and various other vendors. Table 6 presents a comparison of functional and non-functional features of databases adhering to ACID, BASE, and CAP models.

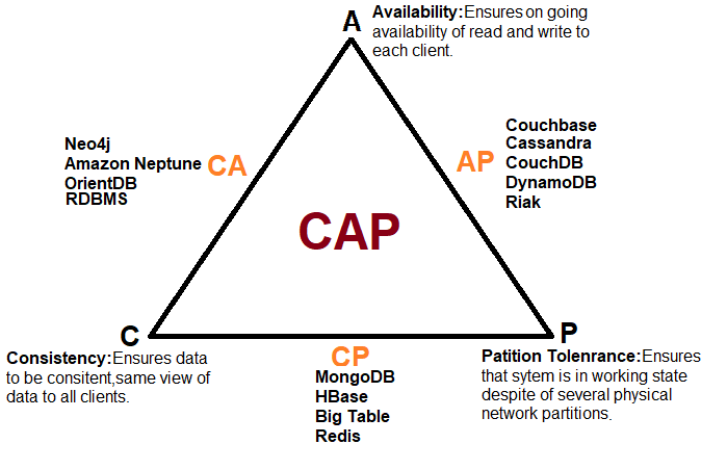

Figure 3. CAP THEOREM

All three structuring models have their prioritization of schemes for providing efficient data integrity. These models can be chosen based on their support of certain features like scalability, availability, partition tolerance, consistency, security, standard query language, performance, and cost. The ACID model is suitable for organizations that favor cost and focus on strong consistency, security, reliability, support for limited data, a standard

Table 5. BASE Analysis of NoSQL Databases

DATABASE (TYPE)	BASICALLY AVAILABLE	EVENTUALLY CONSISTENT
Document		
MongoDB	Ensures basic availability	Strongly consistent as well as eventually consistent (favors consistency and partition tolerance)
CouchDB	High availability over consistency	Eventually consistent (favors availability and partition tolerance)
Couchbase	High availability over consistency	Eventually consistent (favors availability and partition tolerance)
Column		
Cassandra	High availability over consistency	Eventually consistent (favors availability and partition tolerance)
HBase	Ensures basic Availability	Eventually consistent (favors consistency and partition tolerance)
Bigtable	Ensures basic Availability	Eventually consistent (favors consistency and partition tolerance)
Key-value		
DynamoDB	High availability over consistency	Eventually consistent (favors availability and partition tolerance)
Redis	Ensures basic Availability	Eventually consistent (favors consistency and partition tolerance)
Riak	High availability over consistency	Eventually consistent (favors availability and partition tolerance)
Graph		
Neo4j	High availability as well as consistency	Strongly consistent as well as Eventually consistent (favors availability and consistency)
Orient DB	High availability as well as consistency	Eventually consistent (favors availability and consistency)
Amazon Neptune	High availability as well as consistency	Eventually consistent (favors availability and consistency)

query language, and data integrity. For scenarios where consequences for data loss or corruption are severe while compromising scalability, availability, and performance. Relational databases are native to the ACID model. An ACID model can be best suited for systems that require strong consistency in such as financial transactions which need to be processed reliably and securely; database management where systems require data integrity and consistency; mission-critical systems where systems cannot afford data loss or corruption; defense systems and healthcare systems; real-time systems and so on. ACID ensures predictable and reliable behavior and is suited for financial trading platforms, air traffic control systems, and so on. Organizations that might use the ACID model include banks, healthcare providers, online marketing, database management system vendors, e-commerce platforms, aerospace, and defense systems.

The BASE model can be considered ideal for companies prioritizing high availability, increased scalability, improved performance, adaptable schema, handling large volumes of data, utilizing open source technologies, and cost-effectiveness. It lacks a standard query language and sacrifices security. NoSQL databases are BASE-compliant. Real-time web applications, Big data analytics, IoT sensor data processing, Distributed systems, and Social media platforms that require high availability, scalability, and can tolerate some level of inconsistency. Organizations that might benefit from the BASE model include Social media companies(Facebook, Twitter), E-Learning (Coursera, Canvas), Big data analytics companies (Hadoop, Spark), Real-time web application providers (online gaming, live streaming), E-commerce companies (Amazon, eBay), Cloud providers (AWS, Azure, Google Cloud).

The CAP model states that a distributed system can exhibit only two out of the three characteristics. Developers can either choose CA (Consistency and Availability) while compromising the partition tolerance, CP (Consistency and Partition tolerance) while sacrificing the availability, or AP (Availability and Partition tolerance) while compromising the consistency of data. The CAP model is essential in systems where flexible consistency, high availability,

Table 6. Comparing Functional and Non-Functional Features of Databases Adhering to ACID, BASE, and CAP Models

CRITERIA	ACID	BASE	CAP
Availability	Less Availability	Ensures ongoing Availability	Ensure availability in CA and AP layers
Scalability	Poor Scalability	Better Scalability	Better Scalability
Consistency	Strong Consistency	Eventual Consistency	Ensures Consistency in CP and CA layers
Volume of data	Limited Volume of data	Large Volume of data	Large Volume of data
Cost	High Cost	Low Cost	Low Cost generally for databases residing in CP and AP layers
Variety of data	Less support for a variety of data	High support for variety of data	Highly support for variety of data
Complexity	More complex due to predefined schema, SQL queries, and Joins	Less Complex due to schema-less nature	Less Complex due to schema-free nature
Performance	Weak Performance	Better Performance	Better Performance
Standard Query Language and Joins	Databases that adhere to the ACID model often Support Standard Query language and joins	Databases that adhere to the BASE model lack Standard Query language Support and joins	Databases that adhere to the CAP model lack Standard Query language and joins that reside in CP and AP layers
Flexibility	Less flexible due to strict schema	More flexible due to schema-free structure	More flexible for databases residing in CP and AP layers
Security	Ensures Strong security mechanisms that protect the data	Lacks security as it is not part of the database rather it is handled by middleware	It does not provide direct security guarantees
Atomicity	Ensures atomicity	Does not support atomicity	Does not explicitly ensure atomicity
Aggregation	Databases that adhere to the ACID model Support aggregation	Databases that adhere to the BASE model often Support aggregation except graph store	Databases that adhere to the CAP model often Support aggregation except for graph store
Denormalization	Databases that adhere to the ACID model Support Normalization, e.g., Relational databases	Databases that adhere to the BASE model Support denormalization except for document store. e.g., NoSQL databases	Databases that adhere to the CAP model Support denormalization except for document store. e.g., the relational and NoSQL databases

and scalability are crucial and support for partition tolerance is necessary due to network unreliability. The CAP model is best suited for distributed systems that operate in environments with unreliable networks or partitions, such as distributed systems with multiple data centers, Edge computing platforms, mobile devices, or IoT devices with intermittent connectivity. Organizations that might benefit from CAP include tech giants (Google, Amazon), online gaming companies (Fortnite, World of Warcraft), and IoT companies (smart home device manufacturers). Organizations may choose any of the discussed models based on the features they want to incorporate into their system.

5 Conclusion and Future Work

NoSQL databases are famous for their enhanced capabilities and support for large volumes of data. They have provided the best choice for distributed systems, educational sectors, and industrial practices that require greater scalability, availability, and better performance. Our study explored ACID and BASE constraints of the twelve most popular NoSQL databases from each of four broad categories (Document, Key-Value, Column family, and Graph) based on top-reviewed scientific publications from 2010 to 2025. The study also performed an ACID BASE analysis of NoSQL databases and found that NoSQL being native to the BASE model also support ACID constraints at varying degrees as many organizations not only require basic availability but also strong consistency as well; among the twelve explored NoSQL databases, MongoDB and Neo4j offer the best support to atomicity, consistency, isolation, and durability constraints; although other databases also support ACID constraints to varying extents. All BASE compliant NoSQL databases are soft state and offer basic availability. Most of the NoSQL databases follow eventually consistent, however, few support strong consistency as well. MongoDB, HBase, Big Table, and Redis offer basic availability focusing on consistency. On the other hand, Neo4j, OrientDB, and Amazon Neptune support high availability as well as consistency. Couchbase, CouchDB, Cassandra, DynamoDB, and Riak offer high

availability while sacrificing consistency. Organizations adhering to the BASE model and also wanting a flavor of ACID constraints may choose Noe4j and MongoDB databases as they are native to BASE model and also support ACID constraints.

Additionally, we also illustrated the CAP theorem along with the comparison of functional and non-functional features of databases adhering to ACID, BASE, and CAP models. NoSQL databases can be wedded to the ACID as well as the BASE model depending on the nature of their applicability. Our study found that all three structuring models have their own prioritization of schemes for providing efficient data integrity. The ACID model focuses on strong consistency, security, reliability, and data integrity, where consequences for data loss or corruption are severe. e.g., banks and financial institutions, healthcare providers, DBMS vendors, E-commerce platforms, and Defense contractors. The primary focus of the BASE model is on ensuring high availability, increasing scalability, and enhancing performance. e.g., social media companies (Facebook, X), e-learning (Coursera), Big data analytics companies (Hadoop, Spark), and Cloud providers (AWS, Azure, Google Cloud). On the other hand, the CAP model is essential in systems where flexible consistency, high availability, and scalability support to partition tolerance are crucial. The CAP model is best suited for distributed systems with multiple data centers, Edge computing platforms, Mobile devices, or IoT devices with intermittent connectivity. e.g. Tech giants (Google, Amazon), online gaming companies (Fortnite, World of Warcraft), IoT companies (smart home device manufactures). Organizations can choose any of the discussed models based on the features they want to incorporate into their system. In future work, we will study the extended CAP theorem feature PACELC and also focus on how strict ACID constraints can be achieved by NoSQL databases to support absolute transaction integrity and efficient data management.

Author Contributions

Haresh Kumar: Conceptualization, Data curation, Methodology, Visualization, Writing- Original draft preparation
Areej Fatemah Meghji: Conceptualization, Data curation, Writing- Original draft preparation.

Compliance with Ethical Standards

It is declared that all authors don't have any conflict of interest. It is also declared that this article does not contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

References

[1] S. Ferreira, J. Mendonça, and E. Andrade, "Experimental Performance Analysis of Data Consistency Levels in NoSQL Databases," *Software: Practice and Experience*, 2025.

[2] I. Carvalho, F. Sá, and J. Bernardino, "Performance evaluation of NoSQL document databases: couchbase, CouchDB, and MongoDB," *Algorithms*, vol. 16, no. 2, pp. 78, 2023.

[3] N. Bansal, S. Sachdeva, and L. K. Awasthi, "Query-based denormalization using hypergraph (QBDNH): a schema transformation model for migrating relational to NoSQL databases," *Knowledge and Information Systems*, vol. 66, no. 1, pp. 681-722, 2024.

[4] E. Popescu, and A. Radu, "A Comparative Study of Scalability and Performance in NoSQL Databases for Big Data Storage and Retrieval," *International Journal of Social Analytics (IJSA)*, vol. 5 no. 12, pp. 16-27, 2020.

[5] A. Thakare, O. W. Tembhurne, A. R. Thakare, and S. N. Reddy, "NoSQL databases: modern data systems for big data analytics-features, categorization and comparison," *International journal of electrical and computer engineering systems*, vol. 14, no.2, pp. 207-216, 2023.

[6] I. Mapanga, and P. Kadebu, "Database management systems: A nosql analysis," *International Journal of Modern Communication Technologies & Research (IJMCTR)*, vol. 1, pp. 12-18, 2013.

[7] K. Machado, R. Kank, J. Sonawane, and S. Maitra, "A comparative study of ACID and BASE in database transaction processing," *International Journal of Scientific & Engineering Research*, vol. 8, no. 5, pp. 116-119, 2017.

[8] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review," *Big Data and Cognitive Computing*, vol. 7, no.2, pp. 97, 2023.

[9] R. C. McColl, D. Ediger, J. Poovey, D. Campbell, and D. A. Bader, "A performance evaluation of open source graph databases," *In Proceedings of the first workshop on Parallel programming for analytics applications*, pp. 11-18, 2014.

[10] D. Anikin, O. Borisenko, and Y. Nedumov, "Labeled property graphs: SQL or NoSQL?.," *In 2019 Ivannikov Memorial Workshop (IVMEM)*, pp. 7-13. IEEE, 2019.

[11] Z. Ait El Mouden, A. Jakimi, M. Hajar, and M. Boutahar, "Graph schema storage in SQL object-relational database and NoSQL document-oriented database: A comparative study," *In International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*, pp. 176-183, Cham: Springer International Publishing, 2019.

[12] Y. Zhu, E. Yan, and I. Y. Song, "The use of a graph-based system to improve bibliographic information retrieval: System design, implementation, and evaluation," *Journal of the Association for Information Science and Technology*, vol. 68, no. 2, pp. 480-490, 2017.

[13] A. Gupta, S. Tyagi, N. Panwar , S. Sachdeva, and U. Saxena, "NoSQL databases: Critical analysis and comparison," *In 2017 International conference on computing and communication technologies for smart nation (IC3TSN)*, pp. 293-299. IEEE, 2017.

[14] A. Makris, K. Tserpes, V. Andronikou, and D. Anagnostopoulos, "A classification of NoSQL data stores based on key design characteristics," *Procedia Computer Science*, vol. 97, pp. 94-103, 2016.

[15] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, D. Wilkins, "A comparison of a graph database and a relational database: a data provenance perspective," *In Proceedings of the 48th annual ACM Southeast Conference*, pp. 1-6, 2010.

[16] S. D. Dhasade, "NOSQL DATABASE," *International Research Journal of Modernization in Engineering Technology and Science*, vol.04, no. 10, 2022.

[17] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, and G. D. Pecherle, "Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application's data storage," *Applied Sciences*, vol. 10, no. 23, pp. 8524, 2020.

[18] S. Gupta and R. Rani, "A comparative study of elasticsearch and CouchDB document oriented databases," *2016 International Conference on Inventive Computation Technologies (ICICT)*, pp. 1-4, 2016.

[19] S. Kalid, A. Syed, A. Mohammad, and M. N. Halgamuge, "Big-data NoSQL databases: A comparison and analysis of "BigTable","DynamoDB", and "Cassandra"," *In 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 89-93, IEEE, 2017.

[20] K. Moharm, and M. Eltahan, "The role of big data in improving e-learning transition," *In IOP Conference Series: Materials Science and Engineering*, vol. 885, no. 1, pp. 012003, IOP Publishing, 2020.

[21] V. Sharma, and M. Dave, "SQL and NoSQL Databases," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 8, 2012.

[22] S. Gupta and K. Agrawal, "NoSQL Cloud based Bigdata technologies for efficient performance evaluation in the modern era," *14th IEEE International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia*, pp.350-4354, 2022.

[23] H. Vera-Olivera, R. Guo, R. C. Huacarpuma, A. P. B. Da Silva, A. M. Mariano, and M. Holanda, "Data modeling and nosql databases-a systematic mapping review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1-26, 2021.

[24] P. Atzeni, F. Bugiotti, L. Cabibbo, and R. Torlone, "Data modeling in the NoSQL world," *Computer Standards & Interfaces*, vol. 67, pp. 103149, 2020.

[25] P. Jakkula, "HBase or Cassandra? A comparative study of nosql database performance," *International Journal of Scientific and Research Publications*, vol. 10, no.3, pp. 808-820, 2020.

[26] M. A. Abdel-Fattah, W. Mohamed, and S. Abdelgaber, "A comprehensive spark-based layer for converting relational databases to NoSQL," *Big Data and Cognitive Computing*, vol. 6, no.3, pp. 71, 2022.

[27] E. Barbierato, M. Gribaudo, and M. Iacono, "Performance evaluation of NoSQL big-data applications using multi-formalism models," *Future Generation Computer Systems*, vol. 37, pp. 345-353, 2014.

[28] R. Kaur, and J. K. Sahiwal, "A review of comparison between NoSQL databases: MongoDB and couchDB," *International Journal of Recent Technology and Engineering*, vol. 7, pp. 892-898, 2019.