IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 4, Issue 2, March 2024

Database Management Systems: An Examination using NoSQL

Anju Santosh Yedatkar
Assistant Professor, Computer Science Department
Sarhad College of Arts , Commerce And Science, Katraj, Pune, India
anjaliyedatkar@gmail.com

Abstract: *Solutions that can accomplish infinite scalability, high availability, and vast parallelism are needed to meet the ever-changing demands of modern data management. Achieving excellent performance standards. When managing massive collections of both organized and unorganized data sets that traditional RDBMS are unable to handle, the new breed of applications such as business intelligence, enterprise analytics, customer relationship management, document processing, social networks, Web 2.0, and cloud computing needs to scale horizontally to thousands of nodes. The pace the point at which data is produced by interactive applications used by numerous concurrent users in distributed processing involving a vast number of servers and managing Big Data applications has surpassed the capacity of relational databases, emphasizing the adoption of databases NoSQL. Reducing has been addressed by NoSQL database systems.*

Keywords: Relational databases, NoSQL databases, database management systems, ACID, CAP, and BASE

I. INTRODUCTION

The introduction of digital devices and the swift modifications in industrial dynamics across several domains, such as research and technical knowledge, raised the need for high-quality and efficient goods and services. Around this time, automated bookkeeping systems and assembly automation equipment were introduced, along with the automation of real-world activities. Manufacturing Systems, to name just a few. With flat file databases acting as a data management system, these systems could only manipulate text and numerical data. Data protection, organization, storage, aggregation, updating, retrieval, measurement, collecting, transcription, and validation were made possible by this. As technology continues to progress, flat file databases proved insufficient since they were unable to meet the demands of expanding data, data security, and new data kinds. Furthermore, flat file databases didn't include any information about the data; reading the files required further knowledge. Numerous inefficiencies were caused by the lack of a standard for both data storage and communication to and from the database. Cord developed the relational theory in the 1970s, which paved the way for the creation of systems for managing relational databases (RDBMS) to address the issues raised by style. The shift from the complexity of SQL-based servers to NoSQL database systems is a result of businesses throughout the globe, such as Amazon, Facebook, Twitter, and Google, adopting new methods to scale and store massive volumes of data. A class of database management systems called NoSQL was created to address the shortcomings of RDBMSs in specific scenarios. Its schema-less design sets it apart from conventional relational databases. It is therefore appropriate for usage with unstructured data. As these A subset of what SQL can achieve, along with a few more functions, are typically provided by engines' query language. This document is structured as follows: Section II will examine an overview of NoSQL databases. NoSQL database categories are covered in Section III, NoSQL query languages in Section IV, NoSQL database structural models, and finally, conclusions and future work

II. NOSQL DATABASES

The advantages of the NoSQL database strategy include increased flexibility in data storage and manipulation, performance, and allowing for easy scalability. These NoSQL databases come in various varieties, each good for a particular use. Some instances are Disney, bit.ly, Foursquare, and MongoDB, whose deployments are among others, CERN, Source forge, and the New York Times. Facebook used Cassandra and Hadoop (Apache) mostly for Inbox Search. Subsequently, it was made available to the public and is currently a high-level project of the Apache Software Foundation. Digg, Twitter, Reddit, Rackspace, Cloud Kick, Cisco, and others use it. Amazon utilizes Neo4J, Adobe, Cisco, and Voldemort, among other technologies. Although NoSQL uses either CAP or BASE, RDBMS is transaction-oriented and founded on the ACID concept. But with so many options available, choosing among the top NoSQL databases for your interactive web application might be challenging.

The features of a NoSQL:

- Easily accommodate evolving schemas and data.
- High-speed performance
- Support complex queries.
- Flexibility in deployment

There are four primary categories.

1. The data schema in document databases is very adaptable and can change from record to record. They keep data.in forms that allow for flexibility when working with data of all kinds: BSON or JSON.
2. Key-value stores have a straightforward structure and store data as a straight forward set of key-value pairs. Key-value stores are easily scalable and can do extraordinarily fast reads and writes because to this straightforward concept.
3. Wide column stores record enormous amounts of data in a row-and-column layout. They resemble relational databases in format even though they are classified as NoSQL databases. Each row doesn't need to have an equal number of columns, which sets them apart from relational databases.
4. Since data components and their relationships are kept as graphs (as in graph theory, not as line graphs) in graph databases, they have a fundamentally different structure. Numerous types of links between data points and within clusters of data points can be found using complex queries. and among clusters of data points.

III. DIFFERENT NOSQL DATABASE TYPES

A database is a collection of structured data that is readily accessible and kept in a computer system. A database is usually managed by a Database Management System (DBMS). Data is stored in nontabular form using NoSQL, a non-relational database. NoSQL stands for Not only SQL. Key-value, wide-column, documents, and graphs are the primary types NoSQL databases, or "Not Only SQL" databases, are a category of databases that provide a mechanism for storage and retrieval of data that is modeled in ways other than relational databases' (SQL databases') conventional tabular relations. NoSQL databases are often used for large-scale and distributed data storage and processing needs. Certain applications and use cases may benefit from these databases, which are made to manage organized, semi-structured, and unstructured data.

Here are some key characteristics and features of NoSQL databases:

1. Schema-less: NoSQL databases are typically schema-less or schema-flexible, meaning that the data stored in them doesn't need to follow a predefined structure. This flexibility allows for easier adaptation to changing data requirements.
2. Data Models: Support for NoSQL databases Different NoSQL database formats are compatible with different use cases and data patterns.
   a. Key-value stores: The most basic type, in which a unique key is used to store each item (value).
   b. Document Stores: Data is arranged in directories and files, but it is also stored as documents (such as JSON or BSON) and collections.
   c. Column-Family Stores: Each column family is seen as a container of rows, and data is stored in columns rather than rows.
   d. Graph databases : are made to show and query relationships between different elements.
3. Scalability: NoSQL databases may effectively handle higher demands by adding more servers to the database because they are frequently built to scale horizontally.
4. Performance: A lot of NoSQL databases are tailored for particular kinds of workloads or queries, which makes them more efficient in some situations.

5. CAP Theorem: A distributed system can only ensure two of the three characteristics, consistency, availability, and partition tolerance according to the CAP theorem. Typically, NoSQL databases are categorized according to the priority given to which two of these attributes.

Popular NoSQL databases include:
Data is stored as adaptable documents that resemble JSON in MongoDB (Document Store).
Wide-column stores, like Cassandra, are made to manage massive volumes of data over numerous commodity servers without the need for a single point of failure.
Redis is an in-memory data structure store that can be used as a message broker or cache (Key-Value Store).
Graph databases like Neo4j are designed with storing and accessing graph data in mind. It is noteworthy that the decision between NoSQL and conventional SQL databases is contingent upon the particular demands of the application, the type of data, and the anticipated requirements for scalability and performance. NoSQL databases are not a one-size-fits-all solution, and the use case and data properties are generally taken into consideration while selecting one.

V. MODELS FOR STRUCTURING DATABASES

As businesses like Amazon, Google, LinkedIn, and Twitter battled to handle previously unheard-of data quantities and operation volumes within strict latency, NoSQL evolved.

1. ACID

The acronym for Atomicity, Consistency, Isolation, and Durability is ACID. It is a collection of characteristics that ensures database transactions are processed reliably. These characteristics guarantee thatDatabase transactions are handled with dependability even when there are mistakes, unanticipated events, or system malfunctions.(2)

1. Atomicity: This guarantees that a transaction is handled as a single, unbreakable work unit. Either every modification performed during the transaction is saved in the database, or none of them are.
2. Consistency: Consistency guarantees that a transaction converts a valid state in the database. Prior to and following the transaction, the database must adhere to a set of predetermined integrity requirements.
3. Isolation: Isolation ensures that the concurrent execution of transactions does not result in interference between them. Every transaction ought to be carried out as though it were the sole one in the system.
4. Durability: Durability guarantees that once a transaction is committed, its effects will persist, even in the case of a system failure (e.g., power outage or hardware failure).

The changes ACID properties are crucial for ensuring the integrity, reliability, and correctness of database transactions, especially in scenarios where data consistency and accuracy are paramount. While ACID properties are a key feature of traditional relational databases (SQL databases), it's essential to note that not all databases, especially those classified as NoSQL databases, strictly adhere to the ACID model. Some NoSQL databases sacrifice certain ACID properties in favor of achieving other goals like scalability and flexibility.

2. CAP

Consistency, Availability, and Partition Tolerance are three essential distributed system characteristics that are represented by the abbreviation CAP. Brewer's theorem, or the CAP theorem, was developed by computing Eric Brewer, a physicist, in 2000. Theorem shows that all three of these features cannot be concurrently achieved in a distributed system.

1. Consistency: The most recent write or an error is returned by the system for each read. In a consistent system, all nodes in the system have the same data view at the same time. Trade-off: Achieving consistency may require coordination between nodes, perhaps resulting in lower availability and higher latency.
2. Availability: The system responds to each request, but there's no assurance that the answer has the most recent data. In an available system, every node (or as many as possible) in the system is responsive to read and write requests. Trade-off: Favoring availability may sacrifice consistency, as it might be possible to read data that has not yet been updated across all nodes.

3. Partition Tolerance: In spite of network partitions, or breakdowns in communication between nodes, the system keeps running. In other words, even if some nodes in the distributed system can't communicate with each other, the system still functions.

Trade-off: Achieving partition tolerance may lead to scenarios where nodes operate independently during network partitions, potentially causing inconsistencies that need to be resolved later.

A distributed system must decide between preserving Consistency (C) and Availability (A) when faced with a network partition (P), according to the CAP theorem. Under such circumstances, it is impossible to accomplish both. The CAP theorem is therefore frequently represent a trade-off triangle in which the system is only able to give priority to two of the three qualities. It's important to note that the system can have any level of these, according to the CAP theorem. Instead, it highlights the challenge of achieving all three simultaneously under certain conditions. Different distributed databases and systems make different trade-offs based on their design goals and the specific requirements of the applications they support. have Produced Novel Approaches Aiming to Improve Consistency and Efficiency

4. BASE

In the context of distributed databases and systems, a collection of attributes known by the acronym BASE is frequently employed as a substitute to the ACID properties. Essentially, BASE stands for Soft condition, Eventually consistent, and available. It was coined as a counterpoint to the strict guarantees provided by the ACID properties in traditional relational databases.(2)

Here's an overview of the components of BASE:

Basically Available: The system guarantees availability, meaning that it remains operational even in the face of network partitions, node failures, or other types of failures. Trade-off: During certain situations (such as network partitions or node failures), the system might provide a response even if it cannot guarantee the most up-to-date data.

Soft state: Temporary inconsistencies in the system's state could be caused by asynchronous communication, replication lag, or network latency. Trade-off: Rather than enforcing strict consistency at all times, the system allows for some level of inconsistency between nodes or replicas.

Eventually Consistent: The system will converge towards a consistent state over time, given that the system is allowed to recover from temporary inconsistencies. Trade-off: Instead of immediately enforcing consistency, the system allows for variations in the data across nodes, with the expectation that these variations will be resolved over time. BASE is frequently linked to distributed systems and NoSQL databases that are intended to handle massive volumes of data across numerous nodes and scale horizontally. The trade-offs proposed by BASE are seen as more suitable for scenarios where strict consistency (as provided by the ACID model) is difficult to achieve due to the challenges posed by distributed architectures, network partitions, and the need for high availability and scalability. It's worth noting that the choice between ACID and BASE depends on the specific requirements of the application.

VI. CONCLUSION

The RDBMS shared by both database systems had security flaws, scalability restrictions, data availability irrespective of network splits, timely change propagation for consistency, performance difficulties, and the presence of a single point of failure. All data structures cannot be represented and saved due to the RDBMS's strict schema. The architectural limitations imposed by the databases themselves give rise to these difficulties. It was noted that these DBMSs possess some NoSQL cannot handle relational and transactional data well enough. This raises crucial issues for the implementation of CAP, such as how the system balances consistency and availability in the case of a partition (P). independent data. In the coming years, we'll be investigating ways to combine the best aspects of RDBMSs and NoSQL solutions into a single database model.

REFERENCES

[1]. Syed Abdul Rahman, Rakshitha, 2023, Database Management Systems: A NoSQL Analysis,

[2]. Mapanga, Innocent &Kadebu, Prudence. (2013). Database Management Systems: A NoSQL Analysis.

[3]. Mapanga, Innocent, and Prudence Kadebu. "Database Management Systems: A NoSQL Analysis."