Al-Azhar Bulletin of Science Vol. 31, No. 2, (December), 2020, pp. 51-60

A COMPARATIVE STUDY AMONG THE MAIN CATEGORIES OF NoSQL DATABASES

Kamal A. ElDahshan, AbdAllah A. AlHabshy, Gaber E.Abutaleb*

Mathematics Department, Faculty of Science, Al-Azhar University, Cairo, Egypt.

* Corresponding author: gaber_abutaleb@azhar.edu.eg

ABSTRACT

Relational databases are usually used for data storage and retrieval. They are suitable for limited data volume. But when it comes to Bigdata, we need to use more flexible databases that satisfy the need to handle semi-structured and unstructured data. These databases are called NoSQL (Not only SQL) databases. This type of database was developed to interact with data of large volumes. NoSQL databases provide many features such as scalability, availability, replication models, file sharing, and schema-free. This paper's main purpose is to present a comparative study of the five main categories of NoSQL databases; key-value stores, document stores, column family stores, graph stores databases, and object store NoSQL systems. Also, it discusses the famous database management systems for each one of these five categories. The comparison criteria used are performance, scalability, flexibility, complexity, and functionality. Moreover, this paper presents an overview of big data concepts. It briefly discusses the SQL databases versus NoSQL databases in terms of their high-level characteristics. Furthermore, this paper emphasizes the advantages and disadvantages of NoSQL databases. It illustrates the query languages in both SQL and NoSQL databases and represents the most common uses for each category to help users choose the most convenient DBMS for their organization.

Keywords: NoSQL; Key-value stores; Document stores; Column family stores; Graph stores; Object Stores.

1. INTRODUCTION

The main drawback of the relational databases is the inefficiency in handling the data growth. As many organizations gather data with large volumes of semi-structured and unstructured data, relational databases might not be the right storage to use due to the rapid degrading of their performance when data volumes increase[1].To handle all this issue, the NoSQL (not only Structured Query Language) [2] data model was developed by LinkedIn, Google, Facebook, Amazon, and other organizations to handle large data volumes [3]. These organizations have developed their NoSQL DBMS'. Google has developed the Bigtable NoSQL system [4] which uses the column-based, or column family stores. Amazon has developed the Dynamo DB NoSQL DBMS based on the concepts of key-value and document stores [5, 6]. Facebook has utilized the concepts of both key-value and column-based stores to build the Cassandra NoSQL DBMS system [3, 7]. Other document-based NoSQL systems, like Mongo DB and CouchDB, and graph-based NoSQL systems, like Neo4j, were developed by other organizations[8].These developments have been induced by the need to utilize the features and characteristics of the NoSQL systems. Some of these characteristics, like scalability, availability, eventual consistency [9] and replication, file sharding, and high-performance data access, are related to distributed database systems and distributed systems. Other characteristics, such as schema-less and less powerful query languages, are related to data models[3]. The NoSQL systems are introduced in five major categories: key-valuestores, document-based stores, wide column stores, graph-based stores and object stores. This paper introduces an overview of big data concepts and then presents a comparative study among NoSQL systems. It briefly discusses the SQL versus NoSQL databases in terms of the high-level characteristics of each type. The paper explains the advantages and disadvantages of the SQL and NoSQL databases and illustrates the query language used in each type besides the common uses of each type to help decision makers to choose the appropriate database management systems for their organizations.

2-Overview of Big Data Concepts

Big data centered around the technological advancements related to data gathering, storing,

analyzing, and applications in real life. Big data is usually characterized by: volume, veracity, variety, value, volatility, and complexity. The volume of Big data is huge; usually comes in terabytes and petabytes. Big data is being produced very fast and from multiple sources including data from sensors or social media and websites metadata. Big data could be structured, unstructured, or semi-structured and need to be accurate and of high quality to be stored and used to create a business value. Big data complexity comes from the dynamic relationship, where a change in a dataset could result in a change in other datasets. The increasing data availability has resulted in innovating advanced technologies that analyze and harness the data for business value, for example, advanced analytics, machine learning, and artificial intelligence tools [8].

3- Characteristics of SQL databases

Relational database management systems (RDBMSs) use the relational model, which relates the database's tables to each other by referential keys and indexes, making the data storage and retrieval faster than in the old navigational models[10]. SQL is a data query language originally developed by IBM in 1970 to help define and manipulate data. It is embedded in all relational database management systems (RDBMS') [11]. The two main types of RDBMS are the physical architecture and the logical architecture. The physical architecture is concerned with the physical storage of the data in pages, extent, database file, transaction, log files, and the other operating system's files. The logical architecture is concerned with the logical integration and presentation of the data in tables, constraints, views, stored procedures, functions, triggers, etc. [11]. The SQL language manages the following issues: authorization, integrity constraint, views, and concurrency control. It also handled embedded SQL and dynamic SQL. The standard set of the database systems properties includes: atomicity, consistency, isolation, and durability. These properties are often referred to as ACID which assures the database recovery from potential failures that might occur while processing a transaction [12]. The atomicity ensures that either all the transaction operations are performed and reflected in the database or none of them is. The consistency means that the database must stay consistent after performing the transaction. Isolation ensures that multiple transactions may occur concurrently without impacting the database consistency; that is, each transaction is processed in isolated mode. Durability refers to the system's ability to recover the updates of any complete transaction even if a failure in the system or the storage media occurred[11, 13]. Relational database systems avail an advanced level of logic processing, robust declarative query language, metadata and schemas, triggering and consistent relationships, logging and system recovery, multitenant operations and synchronization, roles, indexing and security. These functionalities provide a variety of benefits regarding both the consistency and security of the data. Because SQL databases are developed for data integrity and concurrency control, they are the best option for banking and insurance systems. There is no doubt that data integrity controls a lot of work and high processing power, which results in reaching the limits of the relational databases when it comes to handling data of large volumes. The DBMSs power-fullness could be a disadvantage for efficiency, performance, and flexibility. The fast growth of semi-structured and unstructured data is a major problem that negatively impacts the relational database management systems efficiency [1]. In Big data applications, data varies very widely as it is being gathered from different sources including internet logs, social media, sensors, mobile apps, etc. To handle such data, scalability and flexibility become of critical essence. Scaling SQL systems results in spending much money on expensive hardware in a single node, which make it less efficient and more expensive[11]. After identifying the shortcomings of relational databases in the next section, we will present the advantages of NoSQL databases. Which, based on its characteristics, most organizations tended to use it instead of relational databases.

4- Characteristics of NoSQL Databases

NoSQL databases are developed for enabling the data insertion with out needing a predefined schema. NoSQL systems are non-relational distributed databases designed for large volume data storage and massive parallel processing with in multiple commodities. These systems use the SQL query language and

mechanisms to interact with the data, usually through APIs that interpret the SQL statements to the systems' native query language [13]. NoSQL systems can support a variety of data processing and analysis activities like predictive and exploratory analytics, data ETL (Extraction, Transform, and Load) and OLTP (Online Transactional Processing). Unlike the traditional DBMS and DWHs (Data Warehouses), these systems are highly scalable and can scale to thousands, or even millions, of users perform read/write operations. NoSQL databases are the best fit for organizations that collect a large amount of data. These databases offer increased stability over commodity hardware and focus on the analytical processing of large volume datasets. NoSQL databases don't require consistency to fulfill higher partitioning and availability, which grouped in a property known as BASE [9, 13, 15]. BASE is a property that ensures the NoSQL database reliability despite the loss of consistency. It refers to Basically Available, Soft State, and Eventual Consistent. Basically, Available simply means the system guarantees the reach ability of the data. Soft state means that the data is consistency is the developer's responsibility and the system's state may change at any time. "Eventually" means that at some in the future, data will recover to a consistent state[11]. Whenever ACID becomes not the major concern, NoSQL is the optimal solution and BASE ensures eventual consistency[16]. The major uses of NoSQL databases are: (1) Embedded IR, (2) Parallel processing of data over distributed systems, (3) Explorative analytics on unstructured semi-structured data, (4) large-scale data storage(structured, semi-structured, unstructured)[13], (5) Social Networks, (6) Search Engines, (7) Geospatial analysis, (8) Nuclear modeling, (9) Data warehouses, (10) Caching[17].

5- Advantages and Disadvantage of NoSQL Databases

Advantages: (1) Provide a variety of data models. (2) Scalable[18]. (3) Database administrators are not critically needed (4)Some of these databases are programmed to handle hardware failures, like Riak and Cassandra. (5) Fast, flexible, and efficient[19] (6) Most of them are open-source, which makes them more cost-effective than the conventional commercial databases[2]. (7) NoSQL Better For the Internet of Things[20].

Disadvantage: (1)Immature. (2)No standard query language[21]. (3) Some of the NoSQL databases don't comply with ACID. (4) No standard interface. (5) Difficult to maintain [2].(6) Insufficiency of security features[22].

6- Categories of NoSQL Databases

There are some hybrid NoSQL systems and XML databases as well as some other systems that have existed before the term NoSQL became widely used[3]. This section will explain the five main categories of NoSQL databases.

6.1 Document-based NoSQL Stores

Document-based databases are a type of non-relational database that makes structuring data possible; regardless of the schema absence. Document stores were mainly developed for web services as their integration with technologies such as HTTP1 and Java Scriptis easy. This type of database is horizontally scalable by a group of multiple nodes into an integrated system, it distributes the huge data by shard [23]. The schema-free nature of the Document stores omits the need for a pre-defined schema before inserting data. In document databases, users or the processing applications are responsible for specifying the schema; which could result in some disadvantages such as the lack of normalization and referential integrity. On the other hand, the absence of schema provides extreme flexibility to store a large volume of data, and that what most of big data features refer to. For every key (document ID) in the document-based databases, a record can be stored as a value to form the document with its own internal structure[14]. The documents can be inserted and exchanged in standard formats like Java Script Option Nations (JSON), extensible markup language (XML)[13], binary JSON (BSON)[10], or Yet Another Markup Language (YAML)[24]. In the document-based stores, a collection of data contains a set of separated documents. Figure 1 demonstrates a document store called D_USERS that stores a website user's data. The document's version in the D_USERS is indexed by the field Rev (version). Multi-version concurrency control may be used

to resolve concurrent queries is. The Database ensures that each query receives the document version with the most updated number of changes. This is called eventual consistency as it doesn't assure full transactional security. The consistency takes some time to be achieved, which remarkably accelerates the data processing in a tradeoff for the transactional security. In the document-based stores, parallelizing and speeding up the queries can be achieved by the map-reduce technique. This kind of processing occurs in two phases; the Map phase which is corresponding to grouping and the Reduce phase which is corresponding to aggregation in SQL [14]. They also fit for storing semi-structured data that would require using extensive Null values for data completion purpose [13]. Examples of Document-Based NoSQL DBMSs include - but are not limited to – CouchDB (JSON), Mongo DB (BSON)[12, 25], Perservere, Thru DB, Orient DB, Terrastore, RavenDB, and Jackrabbit[26].

6.2 Key-Value NoSQL Stores

An effective simple method of data storage is to store the data in a key-value dataset [s27]. Key-Value stores focused on performance, scalability, and high availability by using a distributed system to store the data [13]. The keys are unique identifiers associated with values used to retrieve and locate these values, which could be simple text or complex objects [6,13], rapidly [24]. In key-value stores, value object can be inserted for any key by a simple SET command. The below figure 2. Shows an example for a key-value store, in which a website user data: first name, last name, email and encrypted password, are stored. For example, the value Jane is stored for the key U17548: first name [14].

Key-value NoSQL stores are schema-less. That is, the data values can be inserted in any format at any time and, without the need for predefined metadata. Being schema-less and the no need for referential integrity make the key-value stores efficient for queries and partitioning and flexible regarding the data types of the data to be stored. Key-value stores are suitable for efficient reading and writing of massive amounts of data. Buffering the key-value pairs in the main memory of the database can enhance the performance even further. This setup is called in-memory database and is enabled by employing technologies that allow caching the value in the main memory [14]. A

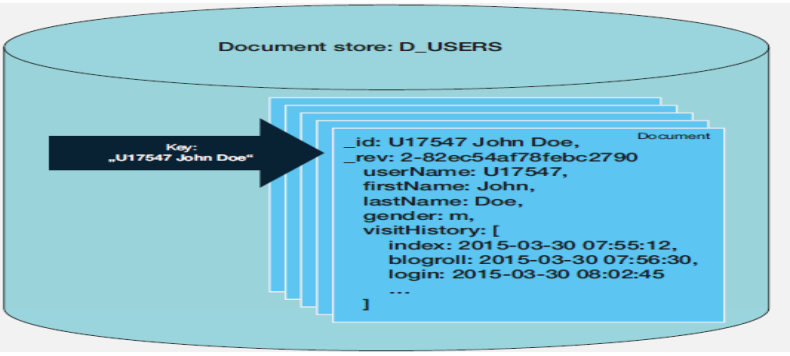

Fig. 1 Document store example

```
SET User:U17547:firstname John
SET User:U17547:lastname Doe
SET User:U17547:email john.doe@blue_planet.net
SET User:U17547:pwhash D75872C818DC63BC1D87EA12
SET User:U17548:firstname Jane
SET User:U17548:lastname Doherty
…
```

Data objects can be retrieved with a simple query using the key:

```
GET User:U17547:email
> john.doe@blue_planet.net
```

Fig. 2 Key-value store example

weakness of this model is the lack of some traditional capabilities such as atomicity, consistency, etc. Another weakness is that maintaining more values as keys become more difficult while the data volume increases [28]. The data model used in key-value stores has no query languages but rather a set of operations that can be used by the application programmers [24]. The key-value store's simplicity makes it ideal to fast and scalable retrieval of the values needed for the applications activities like retrieving product names or managing user profiles or sessions [13] and Telecom directories[20]. Examples of this type of NoSQL DBMS' are: Redis, Voldemort (LinkedIn), Riak, and Berkeley DB [13, 25].

6.3 NoSQL Graph-Stores

The Graph database model has a structuring schema, which makes it significantly different from the other NoSQL data models. A graph database is a collection of nodes (entities or businesses), the relationships between nodes(i.e. edges), and key-value pairs attached to the nodes and relationships [1,29]. This database stores the data in graphs that use the Index-Free adjacency technique in which each node holds a pointer pointing to the adjacent node. Graph databases are schema-free and come up with efficient storage of semi-structured and unstructured data [15,29]. In Graph databases, the queries are expressed as traversals which make them much faster than the relational databases[2]. The below figure 3 demonstrates an example of the graph databases called the G_USERS. It illustrates information of a website with web pages, users, and the relationships between them. For every node in, the system can specify its direct neighbor, without considering all the edges. This property is a key advantage of the graph databases and called the index-free adjacency[14].

Unlike in the relational databases, the schema in Graph stores is implicit. That enables the direct insertion for the data objects belonging to a not-yet-existing node or edge type without defining the type first. And based on the available information, the DBMS triggers the changes in the schema and thereby creates the respective type[14]. Graph databases can be used in multiple applications like cloud management, Artificial-Intelligence-based recommendation systems, access control, social networking, bioinformatics, network management, security, OLTP [30], etc. Clustering and achieving the sharding are not that easy in the graph-based stores [2]. Some examples of Graph databases DBMSs are: Neo4j [25], Hyper GraphDB, Sones GraphDB, InfiniteGraph, InfoGrid, and Allegro Graph[13].

6.4 Column-Based or Wide Column NoSQL Stores

Column-based databases optimize the key-value stores by providing further structure. It has been proven, in certain uses, that to optimize the read operations it is better to store the data into the relational datasets per column, not per row. This is due to the occasional need for a thorough retrieval of all the columns in one row; however, there are groups of columns that are usually read together[31 ,10]. A useful way

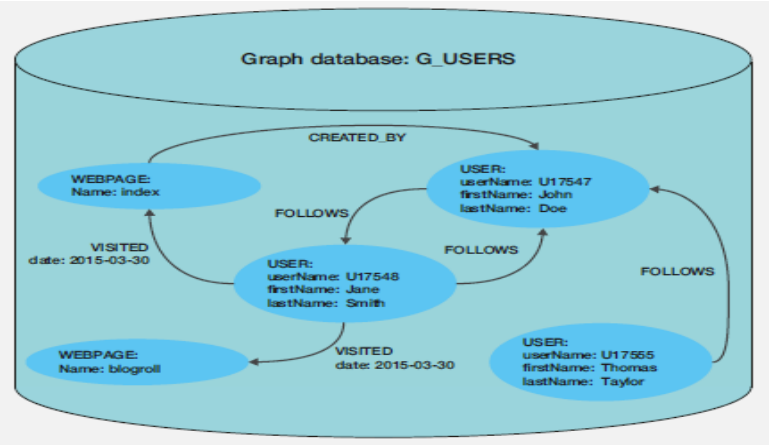

Fig. 3 Graph store example

to optimize the data access is to structure the data in column families—groups of columns — as storage units. In 2008, Google released its column-based Bigtable DBMS. This remarkably influenced the development of the wide-column databases. The row key is an addressing of a database object in column-based stores within which there is another structure divides the row into multiple columns, all addressed by the same key. Timestamp versions are associated with the new entries in the table. Each storage unit is identified by a combination of column key, row key, and time stamp and is referred to as a cell. To control the access to the dataset, its column families are used for granting privileges of reading and writing to users and applications [10]. Column families are the only schemas that are defined when creating the table. The mechanism used to minimize the data retrieval time is by storing the data of the same type in one column family in one row of the dataset and reding it together [2]. Figure 4demonstrates how data is stored in the column family stores[13].

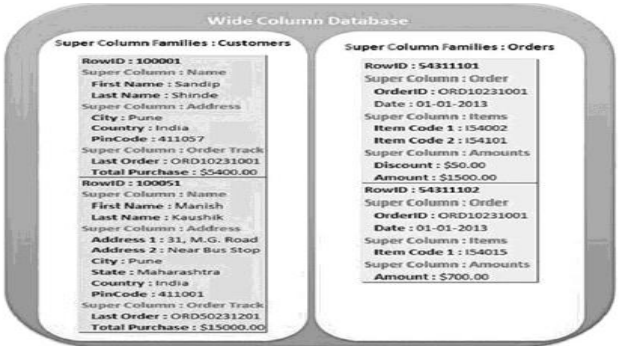

Fig. 4 Column store example

The column-family NoSQL Database Management System is ideal for data distributed storage, especially versioned data due the time-stamping functions of wide-column and column-family; massive scale batch-oriented data processing conversion; storage parsing; predictive and exploratory analytics performed by expert statisticians and programmers [13] and data mining and analytic applications, where the storage of the column family stores is ideal. Common examples for column-oriented DBMSsare HBase and Hypertable; both are derivatives of Big Table [6, 24].

6.5 Object Store NoSQL Stores

An Object-based database is the type of NoSQL database in which data and information are represented as an object. Object-oriented stores can be considered as a combination of object-oriented programming (OOP) and database principles. Encapsulation of data, inheritance, polymorphism, and all the other features of the OOP are offered by other object-based databases. The classes, attributes, and objects in this type of database are comparable to a table, columns, and tuple in a table in RDBMS respectively. Each object is uniquely represented by its object identifier. Data access is much faster in object-oriented databases as the object can easily be retrieved using pointers. Object-based databases ease the agility of modern software development processes [2]. Approaches to the object-oriented stores can be identical to the document-based stores. A major exception is that while the document-based databases explicitly serialize the object to the data, object databases preserve object structure bound to an OOP language like C++ and Java. Compared to other NoSQL models, the object data model complies with the traditional ACID properties relevant to the database reliability. Although this is similar to the traditional relational databases, it is important to consider the object-oriented stores as non-relational databases and to know that they are not queried using SQL[28, 32]. Object-oriented databases are optimal to be used in applications that involve complex object relationships, changing object structures, or if the application defines members that are grouped in collections [2]. Application of scientific research, telecommunication, computer-aided drafting, etc. are common uses for the object-oriented databases. However, the drawback of object-oriented databases is the limited programming language they are bounded to. Also, the object-oriented database is not easy to scale once it exceeds its memory size. Object-oriented databases may not be the best choice whenever the data relationships are simple [2]. Examples of Object Store NoSQL DBMS' include Gem Stone, db4o and objective/DB[28].

7- Query Language

A query language is a set of structured statements used to define, manipulate, and retrieve data in a database. NoSQL DBMS don't use SQL that is used as the query language of the relational databases[33]. NoSQL doesn't have a standard query language and the majority

of the NoSQL DMNSs providers have invented their own query languages[34]. This brings some difficulty when a user wants to switch from one NoSQL DBMS to another. For example, Cassandra supports Cassandra Query Language (CQL), Mongo DB supports the mongo query language, etc[33]. This has created a critical need to develop a common query language that can support a variety of NoSQL database users. UnQL (pronounced as uncle) is a joint effort that brings a commonplace and standardized data definition and manipulation language to the NoSQL platform. UnQL stands for (Unstructured Query Language and is being developed by the creators of Couch and SQLlite [2].

8-Comparison of NoSQL Databases Categories

Popescu summarizes the idea of Ben Scofield's who has delivered a generic introduction to NoSQL databases through a categorization of different NoSQL databases as presented in table 1[35]. We add the Object Store NoSQL systems to the table as follows:

Performance evaluation: Database performance was defined by the speed at which it computed basic operations. For benchmarking, was used the Yahoo Cloud Serving Benchmark (YCSB). Workload was applied to a table of 100,000,000 records; each record was 1,000 bytes in size and contained 10 fields. High-performance categories demonstrated good performance with a throughput below 20 milliseconds. Variable-performance categories demonstrated when the workload reached 1,500 operations per second, latency increased to 100 milliseconds. Based on the above, the high-performance range can be considered 20 milliseconds or less and variable performance from 40 to 100 or more [36].

Scalability evaluation: Scalable data architectures have evolved to improve overall system efficiency and reduce operational costs. High scalability: The NoSQL systems can horizontally scale out throughput over many nodes. Graph DB is variable on scalability because it cares about the nature of the relationships between nodes.

Flexibility evaluation: Flexibility: NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes NoSQL databases ideal for semi-structured and unstructured data. High Flexibility: This means no need to define any schema when inserting the data. Moderate Flexibility: this means you have to define the small aspects of the schema like super columns in Column stores.

Complexity evaluation: The domain complexity is how they stack up the data and deal with it. Complexity high in the graph because the data stores in nodes and many nodes may be connected. None on key-value because most of the key-value NoSQL DBMSs are holding the data in memory.

Functionality evaluation: NoSQL databases provide highly functional APIs and data types that are purpose-built for each of their respective data models. High functional: means the database offers a lot of functions they are built-in. Low functional: means the database does not offer a lot of functions. Minimal function: It means that it provides the required limit of functions.

Table 1. Comparison of NoSQL databases categories[17]

NoSQL DB categories	Storage	Performance	Scalability	Flexibility	Complexity	Functionality	DBMS Examples
Key-Value stores	Key and value	High	High	High	None	Variable (None)	Redis, Voldemort
Column stores	Columns	High	High	Moderate	Low	Minimal	HBase and Hypertable
Document stores	XML, JSON, and BSON	High	Variable (High)	High	Low	Variable (Low)	CouchDB, MongoDB
Graph DB	Nodes and edges	Variable	Variable	High	High	Graph Theory	Neo4j, HyperGraphDB
Object Store (added)	Objects	High	Variable (High)	High	Low	Object-Oriented Programming	GemStone, db4o

9- DISCUSSION

In this study, we mentioned to the five main categories of NoSQL databases. These five categories were compared based on specific criteria. These criteria are Performance, Scalability, Flexibility, Complexity, and Functionality. It is noticeable that

- **Performance** is high in Key-Value, Column, Document, and Object stores and variable in the Graph databases.
- **Scalability** is high in Key-Value, Column, and Graph stores while variable in the case of the document and object stores.
- **Flexibility**is high in Key-Value, Document, Graph, and Object stores while moderate in the case of the column store.
- **Complexity** low in column, Document, and Object stores while high in the Graph stores but the fastest is Key-Value stores.
- **Functionality** Variable (None) in Key-Value Stores and Variable (low) in Document Stores while Minimal in Column Stores. Functionality is based upon Graph theory in Graph stores and it is based upon Object-Oriented Programming (OOP) in the case of the Object stores.

According to the above qualitative comparison, we can conclude the following findings: **Document stores** NoSQL DB category is the most convenient for online shopping, event logging, deep analytical processing, content management, and changing data where versioning is important. For example: Customer relationship management. **Key-value stores** NoSQL DB category is the most convenient for serving add content and managing user or product profiles and session management. Key-value stores are convenient to use for rapidly changing data with a foreseeable database size. For example: Real-time analytics and Real-time communication. **Graph Stores** NoSQL DB category is the most convenient category for any application that benefits from relationships among graph nodes such as social media, network search, and fraud detection. Also, Graph stores are convenient to use for graph-style applications such as searching social relations, public transport links, and road maps. **Column Stores** NoSQL DB category is the most convenient category for content management, event logging, and categorizing for analytics. Column Stores are designed to deliver random, real-time, and read or write access to huge data and to run Map Reduce jobs on huge databases; Search engines and Analyzing log data. **Object Stores** NoSQL DB category is the most convenient for storing and retrieving binary large objects such as files, images, videos, and audio files. The Object Stores category is useful for any application that benefits from Object-Oriented Programming (OOP).

10- CONCLUSION

This paper introduced an overview of big data concepts. It briefly discussed the SQL versus NoSQL databases in terms of the high-level characteristics of each type. The paper explained the advantages and disadvantages of the SQL and NoSQL databases. Moreover, it illustrated the query languages used in each type and the primary uses of each one of them. The common uses of each type have been presented to help decision-makers choose the appropriate DBMS for their organizations. This paper surveyed the five main categories of NoSQL databases; Namely, Key-value stores, Document stores, Column family stores, Graph databases, and Object Store. It presented a comparative study among these categories based upon these comparison criteria: performance, scalability, flexibility, complexity, and functionality. This comparative study reveals the strengths and weaknesses of each of these categories. The most famous database management systems in each category were mentioned.

11- REFERENCES

[1]. Tauro, C.J., S. Aravindh, and A. Shreeharsha, Comparative study of the new generation, agile, scalable, high performance NOSQL databases. International Journal of Computer Applications, 2012. **48**(20): p. 1-4.

[2] Nayak, A., A. Poriya, and D. Poojary, Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, 2013. **5**(4): p. 16-19.

[3] Elmasri, R. and S. Navathe, Fundamentals of database systems. 2017: Pearson.

[4] Chang, F., et al., Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 2008. **26**(2): p. 4.

[5] Sivasubramanian, S., Amazon dynamoDB: a seamlessly scalable non-relational database service, in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. 2012, Association for Computing Machinery: Scottsdale, Arizona, USA. p. 729–730.

[6] Özsu, M.T. and P. Valduriez, NoSQL, NewSQL, and Polystores, in Principles of Distributed Database Systems. 2020, Springer. p. 519-557.

[7] Lakshman, A. and P. Malik, Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 2010. **44**(2): p. 35-40.

[8] Sahafizadeh, E. and M.A. Nematbakhsh, A survey on security issues in Big Data and NoSQL. Advances in Computer Science: an International Journal, 2015. **4**(4): p. 68-72.

[9] Diogo, M., B. Cabral, and J.J.F.I. Bernardino, Consistency Models of NoSQL Databases. 2019. **11**(2): p. 43.

[10] Mahajan, D., et al., Improving the energy efficiency of relational and NoSQL databases via query optimizations. 2019. **22**: p. 120-133.

[11] Binani, S., A. Gutti, and S. Upadhyay, SQL vs. NoSQL vs. NewSQL-A comparative study. database, 2016. **6**(1): p. 1-4.

[12] Kumar, K.S. and S. Mohanavalli. A performance comparison of document oriented NoSQL databases. in 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP). 2017. IEEE.

[13] Moniruzzaman, A.B.M. and S.A. Hossain, NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. CoRR, 2013. **abs/1307.0191**.

[14] Meier, A. and M. Kaufmann, Data Management, in SQL & NoSQL Databases. 2019, Springer. p. 1-23.

[15] Abed, A.H.J.I.J.A.N. and Applications, Recovery and Concurrency Challenging in Big Data and NoSQL Database Systems. 2020. **11**(04): p. 4321-4329.

[16] Kanwar, R., P. Trivedi, and K. Singh, NoSQL, a solution for distributed database management system. International journal of computer applications, 2013. **67**(2).

[17] Zvarevashe, K. and T.T. Gotora, A Random Walk through the Dark Side of NoSQL Databases in Big Data Analytics. International Journal of Science and Research, 2014. **3**(6): p. 506-509.

[18] Chen, J.-K. and W.-Z. Lee. A study of NoSQL Database for enterprises. in 2018 International Symposium on Computer, Consumer and Control (IS3C). 2018. IEEE.

[19] Li, Z. (2018). NoSQL Databases. The Geographic Information Science & Technology Body ofKnowledge (2nd Quarter 2018 Edition), John P. Wilson (Ed). DOI: 10.22224/gistbok/2018.2.4.

[20] Sharma, S.J.S.i.I.P.N., Nosql Better For Internet of Things. 2020. **40**(74): p. 2235-2238.

[21] Amghar, S., S. Cherdal, and S. Mouline. Data Integration and NoSQL Systems: A State of the Art. in Proceedings of the 4th International Conference on Big Data and Internet of Things. 2019.

[22] Zugaj, W., A.S.J.A.J.o.I.S. Beichler, and Technology, Analysis of Standard Security Features for Selected NoSQL Systems. 2019. **3**(2): p. 41-49.

[23] Suma, S. and F.J.I.J.o.A. Alqurashi, A comparison study of NoSQL document-oriented database system. 2019. **8**(1): p. 27-31.

[24] Kaur, K. and R. Rani. Modeling and querying data in NoSQL databases. in 2013 IEEE International Conference on Big Data. 2013. IEEE.

[25] Kamal, S.H., H.H. Elazhary, and E.E.J.I.J.A.C.S.A. Hassanein, A qualitative comparison of NoSQL data stores. 2019. **10**(2): p. 330-338.

[26] Tudorica, B.G. and C. Bucur. A comparison between several NoSQL databases with comments and notes. in 2011 RoEduNet

دراسة مقارنة بين الفئات الرئيسية لقواعد بيانات NoSQL

الملخص

عادة ما تستخدم قواعد البيانات العلائقية لتخزين البيانات واسترجاعها. وهي مناسبة لحجم البيانات المحدود. ولكن عندما يتعلق الأمر بـ Bigdata ، نحتاج إلى استخدام قواعد بيانات أكثر مرونة تلبي الحاجة إلى معالجة البيانات شبه المنظمة وغير المنظمة. تسمى قواعد البيانات هذه قواعد بيانات NoSQL(ليس فقطSQL) تم تطوير هذا النوع من قواعد البيانات للتفاعل مع البيانات ذات الأحجام الكبيرة. توفر قواعد بيانات NoSQL العديد من الميزات مثل قابلية التوسع والتوافر ونماذج النسخ ومشاركة الملفات وخالية من المخطط. الغرض الرئيسي من هذه الورقة هو تقديم دراسة مقارنة للفئات الرئيسية الخمس لقواعد بياناتNoSQL ؛ مخازن القيمة الرئيسية ، ومخازن المستندات، ومخازن عائلة الأعمدة، وقواعد بيانات مخازن الرسم البياني، وأنظمة تخزين العناصر .NoSQL. كما يناقش أنظمة إدارة قواعد البيانات الشهيرة لكل واحدة من هذه الفئات الخمس. معايير المقارنة المستخدمة هي الأداء وقابلية التوسع والمرونة والتعقيد والوظائف. علاوة على ذلك، تقدم هذه الورقة لمحة عامة عن مفاهيم البيانات الضخمة. يناقش بإيجاز قواعد بيانات SQL مقابل قواعد بيانات NoSQL من حيث خصائصها عالية المستوى. علاوة على ذلك، تؤكد هذه الورقة على مزايا وعيوب قواعد بيانات .NoSQL. يوضح لغات الاستعلام في كل من قواعد بيانات SQL و NoSQL ويمثل الاستخدامات الأكثر شيوعًا لكل فئة لمساعدة المستخدمين على اختيار DBMS الأكثر ملاءمة لمؤسستهم.

international conference 10th edition: Networking in education and research. 2011. IEEE.

[27] Çimrin, K.M. and Y. Daşdemir. NoSQL Database Systems: Review and Comparision. in International Conference on Artificial Intelligence towards Industry. 2018.

[28] Dindoliwala, V.J. and R.D. Morena, Survey on Security Mechanisms In NoSQL Databases. International Journal of Advanced Research in Computer Science, 2017. **8**(5).

[29] Khan, S., et al., Bivariate, Cluster and Suitability Analysis of NoSQL Solutions for Different Application Areas. 2019.

[30] Khan, W. and W.J.I.J.A.C.S.A. Shahzad, Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases. 2017. **8**: p. 523-530.

[31] Davoudian, A., L. Chen, and M.J.A.C.S. Liu, A survey on NoSQL stores. 2018. **51**(2): p. 1-43.

[32] Badlani, P., NoSQL in action-A new pathway to database. International Journal of Science and Research, 2016. **5**(6): p. 872-877.

[33] Babić, A., D. Jakšić, and P.J.Z.V.u.R. Poščić, QUERYING DATA IN NOSQL DATABASES. 2019. **7**(1): p. 257-270.

[34] Bathla, G., et al., Comparative study of NoSQL databases for big data storage. 2018. **7**(26): p. 83.

[35] Strauch, C., U.-L.S. Sites, and W. Kriha, NoSQL databases. Lecture Notes, Stuttgart Media University, 2011. **20**.

[36] Jaxenter website. https://jaxenter. com/ evaluating-nosql-performance-which-database-is-right-for-your-data-107481.html. available at(5/06/2020).