ZAMBIA INFORMATION COMMUNICATION TECHNOLOGY (ICT) JOURNAL

Volume 1 (Issue 1) (2017) Pages 1-4

A Comparative Study of NoSQL and Relational Database

Douglas Kunda

School of Science, Engineering and Technology
Mulungushi University
Box 80415, Kabwe, Zambia

dkunda@mu.ac.zm

Hazael Phiri

School of Science, Engineering and Technology
Mulungushi University
Box 80415, Kabwe, Zambia

***Abstract* - Relational Database and NoSQL are competing types of database models. The former has been in existence since 1979 and the latter since the year 2000. The demands of modern applications especially in web 2.0, 3.0 and big data have made NoSQL a popular database of choice. Choosing an appropriate database model to use is an important decision that developers must make based on the features of a given database model. This paper compares the features of Relational Databases and NoSQL to establish which database is better at supporting demands of modern applications. The paper also brings out the challenges of NoSQL. Finally, the paper concludes by determining whether Relational Databases would completely be replaced by NoSQL database models. The findings revealed that, Relational Databases are based on ACID model which emphasizes better consistency, security and offers a standard query language. However, Relational Databases have poor scalability, weak performance, cost more, face availability challenges when supporting large number of users and handle limited volume of data. NoSQL, on the other hand is based on the BASE model, which emphasizes greater scalability and provides a flexible schema, offers better performance, mostly open source, cheap but, lacks a standard query language and does not provide adequate security mechanisms. Both databases will continue to exist alongside each other with none being better than the other. The choice of the database to use will depend on the nature of the application being developed. Each database type has its own challenges and strengths, with relational database lacking of support for unstructured data while NoSQL lacks standardization and has poor security. Modern applications in web 2.0, 3.0 and big data are well suited to use NoSQL but, there are still many applications that rely on Relational Databases.**

Key words - Relational Databases, NoSQL, Query Language, Security

I. Introduction

Databases have replaced flat files as repositories of large pools of data. Since, the inception of databases, Relation Databases dominated for over 30 years until the year 2000 when NoSQL databases began to replace them in some applications [1]. The nature of application needs did not remain static over time and led to applications that are highly interactive and process large volumes of data, such as e-commerce and social media. Providing interactive features in databases is a major requirement for databases serving web 2.0 and 3.0 applications [2]. The shift in application needs has seen Relational Databases fail to meet the needs of developers and users. Companies such as Google, Facebook and Yahoo have migrated to NoSQL to mitigate the shortcomings of Relational Databases [3]. In spite of these trends, there are many firms that still use Relational Databases. These firms are characterized by limited volume of data that require high levels of consistency.

Relational Databases are based on ACID model i.e. Atomicity, Consistency, Isolation and Durability [4]. Atomicity guarantees completeness of transactions, Consistency provides stability of data in a database, Isolation ensures independence of multiple transactions that are executed at the same time and Durability makes sure that stored transactions do not change state even in the presence of failure. ACID provides consistency and availability as strong properties that made Relational Databases popular. NoSQL, on the other hand is based on the BASE (Basically Available, Soft State and Eventually Consistent) model [3]. The distributed nature of NoSQL brings possibilities of data being partially available when some parts of the distributed database are not operation or cannot be reached hence, the term Basically Available. Soft State allows data to vary overtime with or without input. Eventually Consistent guarantees that data will become consistent in future and not immediately after an operation. BASE gives NoSQL ability to scale easily, offer better performance and greater levels of availability to its users.

This paper is based on a review of past literature and begins with a description of Relational Databases and NoSQL database models. The discussion then progresses to comparing the features of Relational Databases and NoSQL which is then followed by challenges of NoSQL. Based on the features, the authors try to determine whether NoSQL is better than Relational Databases at supporting modern database application needs and whether NoSQL will completely replace Relational Databases.

II. Relational Database And NoSQL Types

There is only one form of Relation Database which is based on the relational model [5]. Many organizations, have adjusted their application requirements to conform to the strict schemas that are fixed in advance in Relational Databases. The strict schema requires the application to conform to the needs of the database instead of the database conforming to the needs of the application. Examples of Relational Databases are MySQL, Microsoft SQL Server and Postgres.

There are many types of NoSQL databases and from the literature reviewed, four are prominent. These include Key-Value, Document oriented, Column databases and Graph databases [6]. In key value, data is stored as a collection of key and value pairs, where the key is a single element in a database identified by its attribute and the value is the value of the attribute [7]. Key value is easy to use but, does not support handling of relationships between data items. Examples of key value database include Memcached and Redis [3]. Document oriented databases use the key and document as attributes where the key refers to the whole document [8]. Examples of document oriented databases are MongoDB and CouchDB, which are well suited to handling complex data structures but, still lack the ability to handle relationships among data items. Column databases contain rows/columns similar to Relational Databases but, each column is stored in a separate file. A key in column oriented databases refers to a column. Other attributes stored include the value and a timestamp. Bigtable and Cassandra are examples and by design, these are less flexible but, offer greater throughput. Graph databases represent data as connected graphs and are based on graph theory [8]. Graph databases are less scalable but, support greater connectivity. Examples include GraphDB and OrientDB.

III. FEATURES OF RELATIONAL DATABASES AND NOSQL DATABASE MODELS

A. *Closed and Open Source*

Relational Databases consists of both open source and proprietary platforms [5]. The proprietary types of Relational Databases like Oracle are often able to scale better than open source counterparts such as MySQL. However, many NoSQL database models are open source such as MongoDB, CouchDB and Cassandra [9]. The open source nature of NoSQL offers greater opportunities for researchers in investigating features of a database and provides cheaper storage for users that cannot afford proprietary database models.

B. *Scalability*

Relational Databases, usually scale up, where hardware upgrades must be made to one server to make it more efficient. This increases the amount of effort required from administrators in upgrading Relational Databases [1]. This method of upgrading also faces challenges in terms of hardware limitations which are fixed by design and cannot be altered. For example, the maximum amount of RAM or secondary storage that is supported by hardware has a fixed value that is determined by the manufactures of the hardware. This means that Relational Databases have the ability to scale but, there will always be a limit on the level of scalability as it is determined by the hardware. To offer scalability, NoSQL require the use of commodity server's i.e. scaling horizontally [8] [10]. Scaling horizontally is not significantly affected by hardware limitations because smaller, cheaper and less powerful server machines can be combined to offer higher levels of scalability instead of having one expensive server. This ability makes implementation easy as virtual machines can be used as commodity servers in scenarios where actual hardware cannot be acquired. Virtual machines can be added and removed without degrading the performance of the database. Modern Internet applications like social media require high levels of scalability which is not adequately addressed in Relational Databases but, is efficiently provided in NoSQL [16].

C. *Cost*

Relational Databases that are better are proprietary and therefore, require great amounts of investment from organizations and individuals that want to benefit from their advanced features. Additional hardware for upgrades also adds other additional costs. This makes Relational Databases to be an expensive approach to data storage [7]. NoSQL is mostly open source which makes it to be a cheaper alternative to Relational Databases [9]. The ability to use virtual machines as commodity servers further reduces the cost of maintaining a NoSQL database, making NoSQL a compelling cheap data store for organizations.

D. *Volume and Variety of Data*

Internet applications have increased the volume of data that databases are expected to handle [11]. The internet has seen the emergence of web 2.0 and 3.0 which have increased the volume and variety of data that must be stored. The coming of big data has also increased volume and variety of data. Relational Databases have failed to handle the large volumes of data coming from these sources. NoSQL is excelling at handling large volumes of data making it suitable for data intensive internet applications [12]. This can be seen from companies such as Google, Facebook and Yahoo that have migrated to NoSQL [3].

E. *Availability*

The number of users and the time spent accessing data has increased, with examples such as social media, ecommerce and cloud storage taking the lead. By design, Relational Databases usually suffer from single point of failure even for very powerful servers [5]. Availability is further limited because Relational Databases scale up. Single points of failure do not fit well in today's modern internet applications on which users are very reliant on to support them in their daily lives. Therefore, the distributed nature of NoSQL makes a better choice to provide availability to users all the time even in the presence of hardware failures [10]. The Basically Available nature of NoSQL makes it possible to have access partial parts of a database in the presence of failure. Users are guaranteed of continued access to the database irrespective of the failures with the system.

F. *Performance*

Relational Databases require much more time to process information making them slow as compared to NoSQL that are fast at processing [13]. The performance of NoSQL improves further as it retrieves data from volatile memory, unlike Relational Databases that retrieves data from non-volatile memory. By design, volatile memory is faster than non volatile

memory. In internet Search applications, NoSQL has outperformed Relational Databases when searching for information [14]. Experiments have been conducted to test performance of both NoSQL and Relational Databases. A comparison of Relational Database to MongoDB showed that MongoDB had better performance for read, update and basic queries while SQL only performed well at updating non key-attributes [15].

G. Complexity

Relational Databases create complex data in circumstances where data to be stored by users is difficult to convert into tables [1]. The emphasis on storing structured data in Relational Databases brings this complexity. Relational Databases complex queries and transactions may not be required in some scenarios where simple read or write operations can suffice such as in social media. NoSQL can store both semi structured and unstructured data [16]. The ability of NoSQL to store both semi structured and unstructured data provides flexibility required to support multiple varieties of data in their raw state without loss of information. For example, converting an audio recording of customer complaint to text for storage in Relational Databases, leads to a loss of information about the mood of the customer. Such information can be preserved in NoSQL, as the recording can be stored in its state without conversion.

H. Query Language

Relational Databases have a strong foundation and well documented literature about SQL. SQL is the only data manipulation language that all Relation Databases use [5]. However, there are minor variations of SQL implementations for the various Relational Databases in use. The strong foundation provided by SQL, makes Relational Databases popular among developers because of the shorter learning curve on any implementation of Relational Database. This foundation still lacks in NoSQL as it relies on object oriented API for data manipulation [1]. Each implementation of NoSQL has its own data manipulation language which, require developers to spend time learning when developing on different type of NoSQL model than the one they are accustomed to. Having multiple ways of querying NoSQL, limits the number queries supported because each implementation must provide its own unique queries [17]. The demands of web 2.0 and 3.0 calls for agile development approaches and NoSQL may fail to meet these demands, since development time is increased by developers who need to learn the language of implementation.

I. Consistency

Relational Databases offer stronger consistency with the strict schema [8]. This feature makes Relational Databases to sacrifice availability as the two are not complimentary. Strong consistency is good for providing uniform view of data immediately after operations are performed. However, there are applications such as social media were flexibility is more important than consistency [16]. NoSQL provides greater availability but, has poor consistency [7]. So for social media, NoSQL is more suitable as a storage option than Relational Databases.

J. Security

Relational Databases face some security challenges such as SQL injection and cross site scripting. Despite these challenges SQL has strong security mechanisms that are used to protect the data which include authentication, authorization, encryption, integrity and auditing [1]. The security mechanisms are part of the database. In NoSQL security is not part of the database but, is handled by middleware [7]. This leaves the database to be vulnerable to attacks. Further, the security mechanisms implemented in middleware should be implemented in a way that does not compromise scalability and performance.

TABLE I
COMPARISON OF RELATIONAL DATABASE AND NOSQL

	Criteria	Relational Database	NoSQL
1	Variety	Both open source and closed platforms [5]	NoSQL mostly open source [9]
2	Scalability	Scales up by upgrading hardware of a single server [1].	Scale horizontally using commodity servers [8]
3	Cost	expensive approach for data storage [7]	Cheaper as it is open source and inexpensive upgrade [9]
4	Volume of Data	Handle limited data [11].	Handle large volumes data especially in Big Data [12].
5	Availability	suffers from single point of failure [5]	Distributed nature provides availability to users all the time in the presence of hardware failures [10]
6	Performance	require much more time to process information making them slow [13]	tends to have better query performance [16]
7	Complexity	create complex data in circumstances were data to be stored by users is difficult to convert into tables [1]	store both semi structured and unstructured data which is less complex [16]
8	Query Language	SQL is the only data manipulation language that all Relation Databases use with minor variations in implementation [5]	Each implementation of NoSQL has its own data manipulation language [19]
9	Consistency	has strong consistency with the strict schema [8]	has poor consistency with a schema less approach [7]
10	Security	Has strong security mechanisms that are used to protect the data [1]	Leaves security to be handled by middleware and is not part of the database [7]

IV. CHALLENGES OF NOSQL

One of the challenges of NoSQL is that it lacks a standard query language [19] [20]. There are more than 50 implementations of NoSQL, with each providing its own language and interface [8]. This has hindered the wide

acceptance of NoSQL as it is difficult for developers to master all implementations of NoSQL manipulation languages. Therefore, NoSQL has fewer users than Relational Databases [10].

Another challenge of NoSQL is poor security as it is still an immature technology [21]. By design, NoSQL offers limited security because emphasis is placed on data handling. NoSQL databases can be attacked by scanning Known port numbers and the data at rest is not encrypted [17]. For NoSQL data that is in transit, SSL transport can be used but, it is not turned on by default as is the case of MongoDB [21]. NoSQL has insufficient logging capabilities making it more vulnerable to insider attacks which cannot be traced easily.

V. Choosing a Better Database Model

Relational Databases are easy to implement, robust, consistent and secure but, they are too rigid [18]. NoSQL performs well in handling huge volumes data, supports unstructured data but, is less consistent and unsecure. It is not possible to conclude that one database is better than the other [8]. Each database model may be chosen depending on the application to be developed. For small applications requiring strong consistency, a developer may choose Relational Databases and for large dynamic databases, a developer may choose NoSQL. In web 2.0, 3.0 and big data applications NoSQL is a better choice than Relational Databases.

VI. NoSQL As A Replacement For Relational Databases

NoSQL may have become popular but, it will not completely replace Relational Databases [4]. For Big Data, Social Networks, Internet of things, NoSQL will continue to dominate but, there are many applications that will still continue to rely on Relational Databases. NoSQL and Relational Database will continue to exist side by side to complement the shortcomings of each other.

VII. Conclusion And Future Work

In this paper, we have presented a comparison of NoSQL and Relational Databases based on existing literature. The study shows that the features of Relational Databases are well suited to handling limited volume of structured data. The study also reveals that NoSQL features are designed for scalability and performance, with thin layer of security over a non-standard Query language. Future work can be conducted determine the possibility of providing a standard query language for NoSQL.

References

[1] M. Abourezq and A. Idrissi, "Database-as-a-Service for Big Data: An Overview," International Journal of Advanced Computer Science and Applications, vol. 7, no. 1, pp. 157-177, 2016.

[2] A. . T. Kabakus and R. Kara, "A performance evaluation of in-memory databases," Journal of King Saud University – Computer and Information sciences, 2016.

[3] J. Batra and S. Batra, "MONGODB Versus SQL: A Case Study on Electricity Data," Emerging Research in Computing, Information,, 2016.

[4] H. L. Zhen, H. Beda, M. Doug, L. Ying and J. C. Hui, "Closing the functional and Performance Gap between SQL and NoSQL," 2016.

[5] M. A. Mohamed, "Relational vs. NoSQL Databases: A Survey," International Journal of Computer and Information Technology, vol. 3, no. 03, 2014.

[6] S. Priyanka and AmitPal, "A Review of NoSQL Databases, Types and Comparison with Relational Database," International Journal of Engineering Science and Computing,, vol. 6, no. 5, pp. 4963-4966, 2016.

[7] A. K. Zaki, "NoSQL Databases: New Milleneum Database For Big Data, Big Users, Cloud Computing and Its Security Challenges," International Journal of Research in Engineering and Technology, vol. 3, no. 3, 2014.

[8] A. Singh, "NoSQL : A New Horizon in Big Data," International Journal of Scientific Research in Science, Engineering and Technology, vol. 2, no. 2, 2016.

[9] W. Kim, "Web data stores (aka NoSQL databases): a data model and data management perspective," Int. J. Web and Grid Services, Vol. 10, No. 1, 2014, vol. 10, no. 1, pp. 100-110, 2014.

[10] S. Sharma, R. Shandilya, S. Patnaik and A. Mahapatra, "Leading NoSQL models for handling Big Data:," Int. J. Business Information Systems, vol. 22, no. 1, 2016.

[11] A. B. Moniruzzaman and S. A. Hossain, "NoSQL Database: New Era of Databases for Big data Analytics -," International Journal of Database Theory and Application, vol. 06, no. 4, 2013.

[12] A. Nayak, A. Poriya and D. Poojary, "Types of NOSQL Databases and its Comparison with Relational Databases," International Journal of Applied Information Systems, vol. 5, no. 4, 2013.

[13] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes and J. Abramov, "Security Issues in NoSQL Databases," in International Joint Conference of IEEE TrustCom, IEEE ICESS-11, 2011.

[14] Z. Parker, S. Poe and S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB," vol. ACMSE'13, 2013.

[15] S. Srinivas and A. Nair, "Security Maturity in NoSQL Databases – Are they Secure Enough to Haul the Modern IT Applications?," in International Conference on Advances in Computing, Communications and Informatics IEEE, 2015.

[16] J. Kepner, D. Hutchison, H. Jonathan, T. Mattison, S. Samsi and A. Ruether , "Associative Array Model of SQL, NoSQL, and NewSQL Databases," 2016.

[17] E. Barbierato, M. Gribaudo and M. Iacono, "Performance evaluation of NoSQL big-data applications using," Future Generation Computer Systems, no. 37, pp. 345-353, Elsevier, 2014.

[18] L. Rocha, F. Vale, E. Cirilo, D. Barbosa and M. Fernando, "A Framework for Migrating Relational," Procedia Computer Science, vol. 51, pp. 2593-2602, Elsevier, 2015.

[19] K. K.-Y. Lee and W.-C. Tang, "Alternatives to relational database: Comparison of NoSQL," Computer Methods and Programs in Biomedine, no. 110, pp. 99-109, Elsevier, 2013.

[20] A. Makrisa, K. Tserpesa, V. Andronikoub and D. Anagnostopoulo, "A classification of NoSQL data stores based on key design," Procedia Computer Science, no. 97, pp. 94-103, Elsevier, 2016.

[21] P. Atzeni, F. Bugiotti and L. Rossi, "Uniform access to No SQL systems," Information Systems, no. 43, pp. 117-133, Elsevier, 2014.