NoSQL Databases for Big Data Management

Dražena Gašpar, Mirela Mabić
University of Mostar
Faculty of Economics
Matice Hrvatske bb, Mostar, B&H
{drazena.gaspar, mirela.mabic}@sve-mo.ba

Abstract. *NoSQL databases, as a relatively new approach to data organisation and management, arose as a response to the enormous growth of data generated through extensive use of Internet and Web 2.0 applications. Today, the term Big Data is generally used to describe this data explosion, where huge amounts of structured and unstructured data are in question. NoSQL databases show great potential in working with Big Data, especially in situations where relational data model does not fit. The aim of this paper is to present the main features of NoSQL databases that make them the first choice in Big Data management.*

Keywords. NoSQL, Big Data, CAP theorem.

1 Introduction

Today, development of database technology can be analysed through three main stages. The first one stage (1968 – 1971) is related to hierarchical and network database models. The second stage is related to relational database model and started in the 1970s with Edgar Codd paper. Relational databases were a predominant model more than thirty years (1972-2005) and even today they are prevailing databases, especially for ERP (Enterprise Resource Planning) solutions. The relational databases are founded on a formal mathematical theory. They ensure the independence of data presentation (data model) with regard to physical data storage implementation. In the gold period of relational databases, almost every significant database management system (DBMS) shared a common architecture based on the relational model, ACID (Atomicity, Consistency, Isolation, Durability) transactions and SQL language (Harrison, 2015). Relational databases even found a way, through constant innovations and implementation of object oriented features, to respond to object oriented requests, but the era of massive web-scale applications created pressures on the relational database that could not be relieved through incremental innovation (Harrison, 2015).

In the year 2005, Google was definitely the biggest website in the world, and it was from the very beginning faced with the request to deal with the volumes and velocity of data. Almost twenty years ago, Google was among the first companies that faced with Big Data problem and had to invent new hardware and software architectures to store and process the exponentially growing quantity of websites it needed to index (Harrison, 2015). The result of those efforts was Hadoop project which symbolize the beginning of the third stage in database development – non-relational databases. Since 2009 non-relational databases have become famous as NoSQL databases. Actually, NoSQL stands for "not only SQL" (Cattell, 2011), including all non-relational databases, regardless of SQL use. NoSQL databases are distributed, non-relational databases designed for large-scale data storage and massively-parallel data process across a large number of commodity servers (Moniruzzaman & Hossain, 2013).

Since 2012 the term Big Data has become mainstream and buzz phrase. There are multiple and competing definitions of Big Data. But, typically, Big Data is considered to be a collection of huge data in very high volume, variety and velocity in nature that cannot be effectively or affordably managed with conventional data management tools, e.g. classic relational database management systems or conventional search engines (Manyika et.al, 2011). But this definition is more focused on data and it does not reflect the real motivation behind Big Data issue. The real motivation for data capturing does not lay in the fact that there is enough capacity to do that, but in the eternal human need to find a better solution for research or business problems, which is basically search for actionable intelligence (Wu et al., 2016).

Big Data should enable decision makers to make the right decisions based on predictions through the analysis of available data. It means that attributes of Big Data have to be viewed through following three aspects, i.e. domain knowledge (Wu et al., 2016):

- Data domain (searching for patterns)
- Business intelligence domain (making predictions)
- Statistical domain (making assumptions).

Wu et al. (Wu et al., 2016) gave graphical presentation of Big Data definition (Fig. 1), believing that it is comprehensive enough to capture all aspects of Big Data. According to them, the original 3V's (volume, variety and velocity) definition gave a syntactic or logical meaning of Big Data, while their 3^2V's definition represents the semantic meaning, e.g. relationship of data, BI (Business Intelligence), and statistics. In the heart of this definition is machine learning, because without the machine (computer), the mission of learning from Big Data would be impossible (Wu et al., 2016).

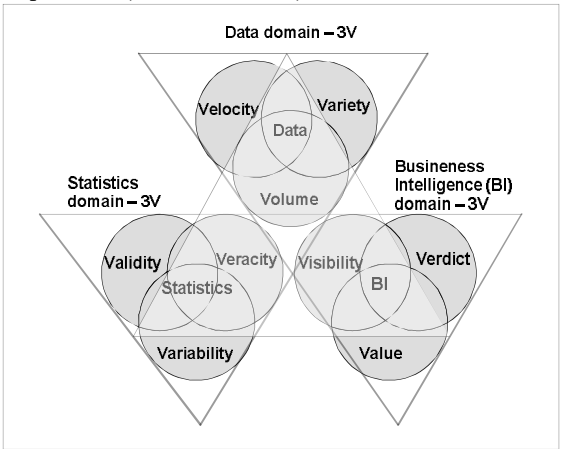

Figure 1. 3^2V's Big Data definition
(Wu et al., 2016)

The Google example has shown that key for resolving Big Data issue lies in developing of an alternative database technology. The examples of other companies faced with Big Data problem, like Yahoo, Amazon, Twitter, additionally confirms that. In this paper, authors analyse main characteristics of NoSQL databases to present why they are more suitable for Big Data management.

2 NoSQL – New Era of Databases

NoSQL databases are answer to challenges related to a huge quantity, velocity, and variety of data that have to be managed, searched and stored by modern database systems. Different concepts and technologies form a foundation for NoSQL database appearance and further development (Harrison, 2015):

- Google File System (GFS) - a distributed cluster file system that allows all of the disks within the Google data center to be accessed as one massive, distributed, redundant file system.
- MapReduce - a distributed processing framework for parallelizing algorithms across large numbers of potentially unreliable servers and being capable of dealing with massive datasets.
- BigTable - a non-relational database system that uses the Google File System for storage.
- Sharding - partitioning the data across multiple databases based on a key attribute, such as the customer identifier. The operational costs of sharding, together with the loss of relational features, made many seek alternatives to the RDBMS.
- AJAX (Asynchronous JavaScript and XML) – programming style that offers far more interactivity to web sites through direct browser communication with a backend by transferring XML messages. XML was soon superseded by JavaScript Object Notation (JSON), which is a self-describing format similar to XML but is more compact and tightly integrated into the JavaScript language. JSON became the de facto format for storing, serializing, objects to disk.

Additionally, the environment in which NoSQL databases have arose and developed is characterized by cloud deployment, mobile applications, social networking, and the Internet of Things. The developers of NoSQL databases have understood from the very beginning that in this new, distributed environment, integrity and consistency of data based on ACID transactions represents a big problem. Namely, relational databases maintain transaction control by using properties like atomicity, consistency, isolation, and durability (ACID) to insure transactions are reliable. The primary consideration of ACID approach is ensuring data consistency and integrity. But, in distributed systems, like NoSQL databases, can arise conflicts related to availability that cannot be fully resolve. The result is CAP theorem, first introduced by Eric Brewer in 2000. The CAP theorem states that any distributed database system can have at most two of the following three desirable properties (McCreary & Kelly, 2014):

- *Strong Consistency* – enabling a single, up-to-date, readable version of data to all clients. Consistency here is concerned with multiple clients reading the same items from replicated partitions and getting consistent results.
- *High availability* – meaning that the distributed database will always allow database clients to update items without delay. Internal communication failures between replicated data shouldn't prevent updates.
- *Partition tolerance* - ability of the system to keep responding to client requests even if there's a communication failure between database partitions.

The CAP theorem proves that it is not possible to create a distributed database that is consistent and available and partition tolerant at the same time. Figure 2 shows that during the implementation of NoSQL databases it is necessary to make trade-off between strong consistency, high availability and partition tolerance.

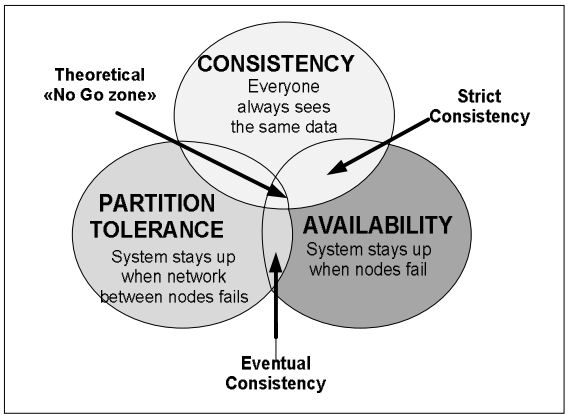

Figure 2. CAP theorem and NoSQL Database (Harrison, 2015)

There are three responses on CAP theorem (Simon, 2012):

1. *Sacrificing Tolerance* – there is no defined system behaviour in case of a network partition. Two phase commit is one of the attempts in resolving this. It supports temporarily partitions, like node crashes, lost messages and similar, by waiting until all messages are received.
2. *Sacrificing Consistency* – partition data can still be used, but since the nodes cannot communicate with each other there is no guarantee that the data is consistent. In that case, optimistic locking and inconsistency resolving protocols can be used.
3. *Sacrificing Availability* - since data can only be used if its consistency is guaranteed, it implies pessimistic locking because it is necessary to lock any updated object until the update has been propagated to all nodes. If a network partition is in question, it might take quite long until the database is in a consistent state again, so system cannot guarantee high availability anymore.

The most of NoSQL databases have to loosen up the requirements on Consistency to reach better Availability and Partitioning. The result is approach known as BASE (Basically Available, Soft-state, Eventually consistent).

The meaning of this acronym is following (Celko, 2014):

- Basically available - This means the system guarantees the availability of the data as per the CAP theorem. But the response can be "failure," "unreliable" because the requested data is in an inconsistent or changing state.
- Soft state - The state of the system could change over time, so even during times without input there may be changes going on due to "eventual consistency," thus the system is always assumed to be soft as opposed to hard, where the data is certain.
- Eventual consistency - The system will eventually become consistent once it stops receiving input, meaning if no additional updates are made to a data item, all reads to that item will eventually return the same value.

Table 1: ACID and BASE approach comparison

ACID (RDBMS)	BASE (NoSQL)
strong consistency	weak consistency (=> allow stale data)
Isolation	last write wins
Transaction	program managed
robust database	simple database
simple code (SQL)	complex code
available & consistent	available & partition-tolerant
scale-up (limited)	scale-out (unlimited)
shared-something (disk, memory, processor)	shared-nothing (parallelizable)

Basic availability means that the temporarily inconsistency is allowed to systems in order to ensure that transactions are manageable. Soft-state means that some inaccuracy is temporarily allowed, and data may change while being used to reduce the amount of consumed resources. Eventual consistency means eventually when all service logic is executed; the system is left in a consistent state.

In the Table 1 is presented comparison of these two approaches, ACID and BASE (Vanroose & Thillo, 2014).

The BASE approach is far away from ideal solution. It can be very costly, because once when one give up

ACID guarantees, it is then up to developers to explicitly code in their applications the logic necessary to ensure consistency in the presence of concurrency and faults. The complexity of this task has sparked a recent backlash against the early enthusiasm for BASE (Chao et al., 2014). Some authors stressed that the process of designing applications that should to resolve concurrency anomalies in their data can be very apt to errors, time-

consuming and probably not worth the performance gains (Shute et al., 2012).

Unlike relational databases that focus on consistency, NoSQL databases, founded on BASE approach, focus on availability. They relax the rules and allow reports to run even if not all portions of the database are synchronized. Their mission is to keep the process moving and deal with broken parts at a later time, so they are ideal for web storefronts, where filling a shopping cart and placing an order is the main priority (McCreary & Kelly, 2014).

3 NoSQL Databases for Big Data Management

The previously described characteristics of NoSQL databases, especially their distributed features, qualify them for Big Data management. Namely, a *big data* class problem is any business problem that's so large that it can't be easily managed using a single processor. This class of problems requires the use of the more complex environment of distributed computing.

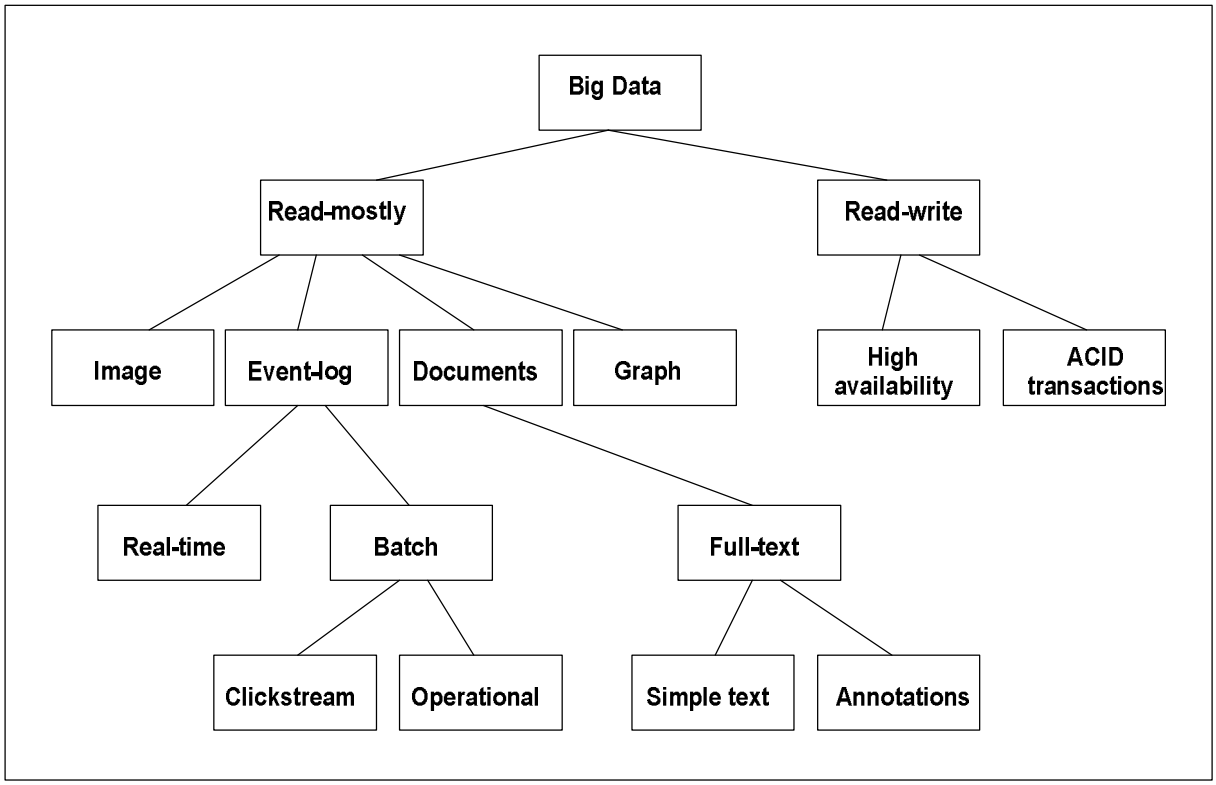

Figure 3. Classification of Big Data problems (McCreary & Kelly, 2014)

The big data problems can be classified as it is shown on Fig. 3.

Today, these different types of big data problems (Fig. 3) are, with more or less success, resolved with different database architectures. According to that, NoSQL database can be classified in four basic categories, each resolving different type of Big Data problems (Fowler, 2016):

- Key-Value
- Wide Column
- Document
- Graph

The key-value type of NoSQL databases uses a key to locate a value (traditional data, BLOBs, files, etc.) in simple, standalone tables, known as hash tables. In that case, searches are performing against keys, not values, and they are restricted to exact matches. The key can be accessed by hashing, indexing, brute-force scans, or any other appropriate method. This is the most primitive model for data retrieval short of a pile of unorganized data (Redmond & Wilson, 2012).
Wide-Column or column-oriented, NoSQL databases got name by their design where data is stored together in columns. By contrast, a row-oriented database (like an RDBMS) keeps information about a row together.

In column-oriented databases, adding columns is quite inexpensive and is done on a row-by-row basis. With respect to structure, columnar is about midway between relational and key-value (Redmond & Wilson, 2012). Wide column databases can be very useful in situations where exist data whose columns can change, and need to be retrieved as a group or aggregate (Fowler, 2016).

Document-oriented NoSQL databases were designed to store and manage documents. The documents are encoded in a standard data exchange formats (XML, JSON – JavaScript Object Notation, BSON – Binary JSON). Document is similar to a hash, with a unique ID field and values that may be any of a variety of types, including more hashes. Documents can contain nested structures, and so they exhibit a high degree of flexibility, allowing for variable domains. Different document databases take different approaches on indexing, ad hoc querying, replication, consistency, and other design decision (Redmond & Wilson, 2012).

Graph NoSQL databases excel at dealing with highly interconnected data. They are focused on relationship, instead on data. A graph database consists of nodes and relationships between nodes. Both nodes and relationships can have properties—key-value pairs—that store data. The real strength of graph databases is traversing through the nodes by following relationships (Redmond & Wilson, 2012).
Figure 4 shows how these categories of NoSQL databases respond for Big Data problems presented on Fig. 3. It is visible (Fig. 4) that when high availability is in question, that the most of NoSQL databases resolve that problem, while when ACID transactions are in question, only key-value databases, which are the closest to relational databases, can compete. Wide column databases are more suitable for event-log problems, while document databases are the best in document management, and graph databases in graph management. Many of the key NoSQL database types are optimized to satisfy one or more of the Big Data challenges (Fig. 4).

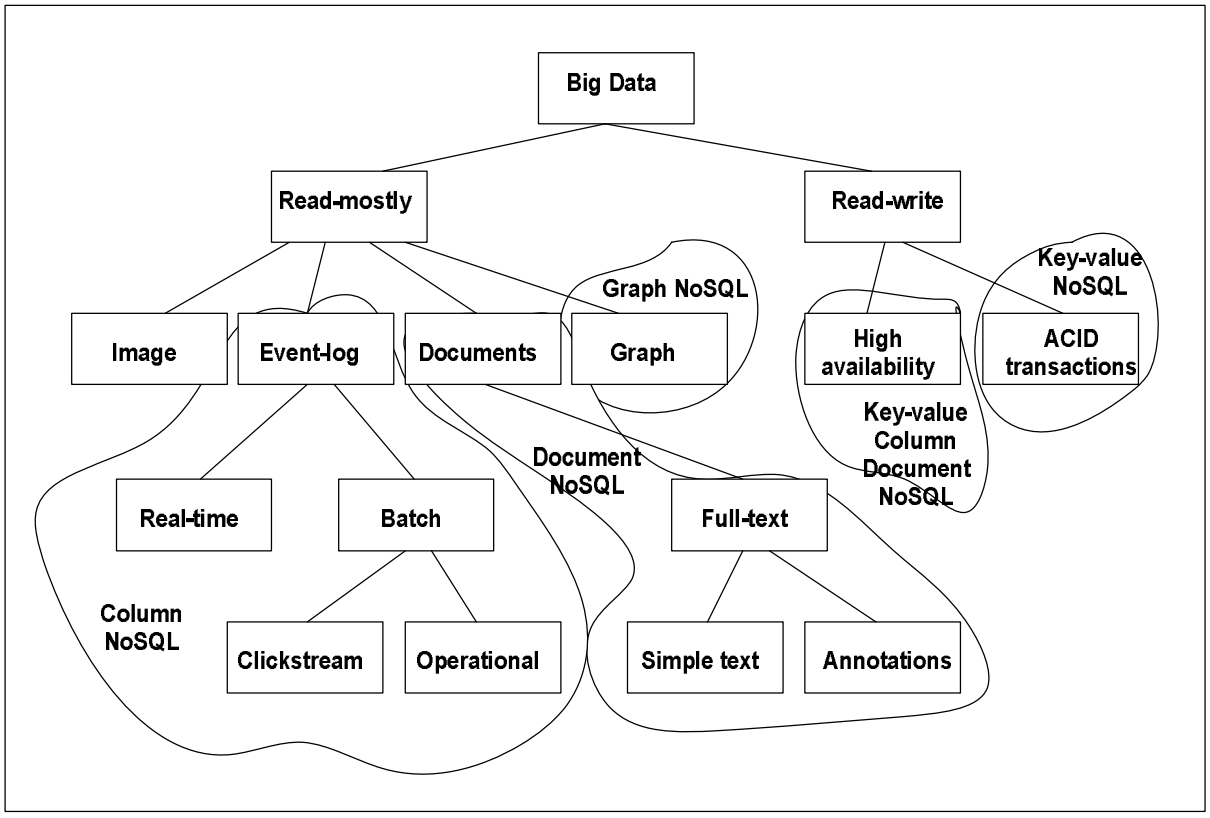

Figure 4: NoSQL databases and Big Data problems

Resolving the Big Data issues is a very challenging task. In order to address the scalability requirements of Big Data, parallel shared-nothing architectures of commodity machines, often consisting of thousands of nodes, have been lately established as the de facto solution. Databases implementing the shared-nothing model often refer to themselves as massively parallel processing (MPP) databases. Figure 5 illustrates the shared-nothing database architecture (Harrison, 2015).

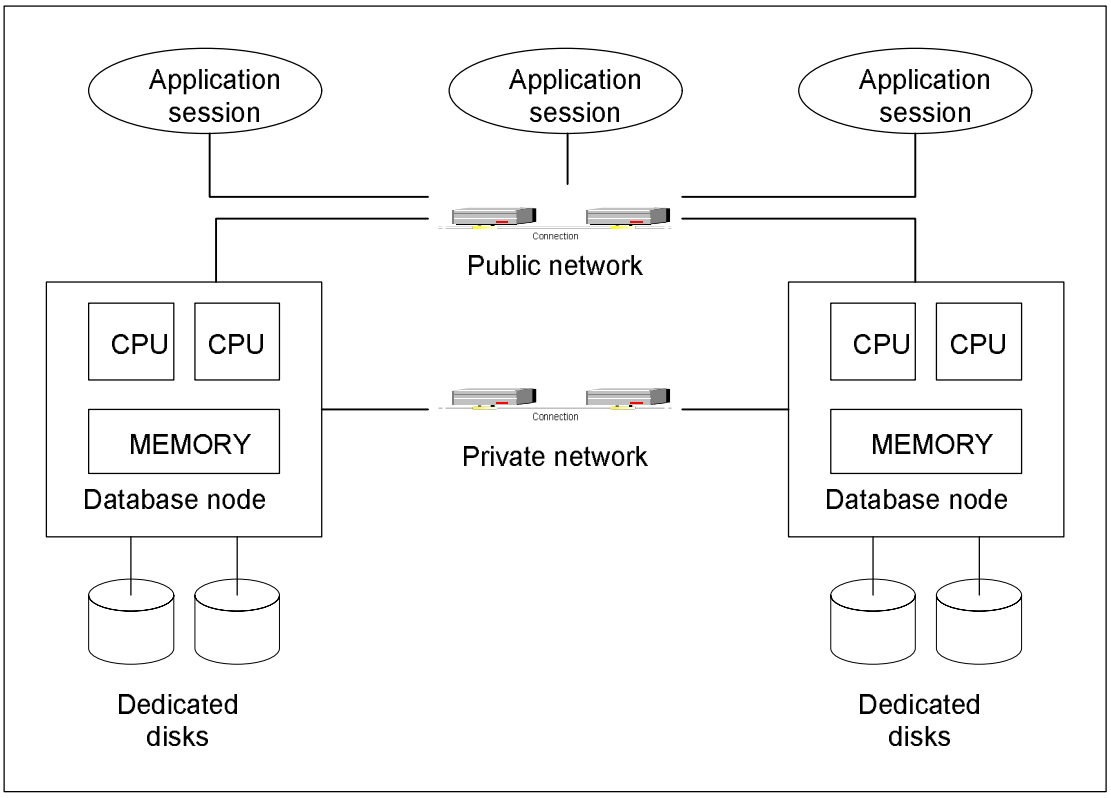

Figure 5. Shared-nothing database architecture (Harrison, 2015)

IT industry developed various systems to support Big Data like MapReduce (Pritchett, 2008), Pregel (Redis, 2016), Spark (Bailis & Ghodsi, 2013) and similar.

The most of NoSQL databases adopted MapReduce (Dean & Ghemawat, 2008) to allow transformation of Big Data over locally distributed nodes instead of transferring large amount of data between nodes. The main functions of MapReduce are map and reduce. The map operation is applied to data on each node and processes the data independently from and in parallel to map operations on the other nodes. The each map operation has as result a collection of lightweight key-value pairs. The reduce operation uses the results of map operation as input and processes them based on keys in parallel. The final values are returned to the user (Ameri, 2016). Since MapReduce allows more data locality and minimizes volume and frequency of data transfer, it is a core component of many Big Data solutions in ensuring system scalability.

Pregel (Redis, 2016) is a specialized model for iterative graph applications. In Pregel, a program runs as a series of coordinated supersteps. With each superstep, each vertex in the graph runs a user function that can update state associated with the vertex, change the graph topology, and send messages to other vertices for use in the next superstep. This model can express many graph algorithms, including shortest paths, bipartite matching, and PageRank.

Spark (Bailis & Ghodsi, 2013) is a fast in-memory data processing system that achieves high performance for applications through caching data in memory (or disk) for data sharing across computation stages. It is achieved with the resilient distributed dataset (RDD) in-memory storage abstraction for computing data, which is a read-only, partitioned collection of records (Datastax, 2016).

The main advantages of NoSQL databases can be summarized as (McCreary & Kelly, 2014):

- Loading test data can be done with drag-and-drop tools before ER modelling is complete.
- Modular architecture allows components to be exchanged.
- Linear scaling takes place as new processing nodes are added to the cluster.
- Lower operational costs are obtained by autosharding.
- Integrated search functions provide high-quality ranked search results.
- There's no need for an object-relational mapping layer. It's easy to store high-variability data.

To be fair, it is necessary to claim that the NoSQL databases suffer from the following weaknesses (Harrison, 2015):

- There are a wide variety of specialized database solutions which exactly fit for an application's requirements.
- A return of the navigational model, e.g. of the situation that existed in pre-relational systems, in which logical and physical representations of data are tightly coupled in an unacceptable way.

One of the great successes of the relational model was the separation of logical representation from physical implementation.

- The inability in most non-relational systems to perform a multi-object transaction, and the possibility of inconsistency and unpredictability in even single-object transactions, can lead to a variety of undesirable outcomes that were mostly solved by the ACID transaction and multi-version consistency control (MVCC) patterns. Phantom reads, lost updates, and nondeterministic behaviours can all occur in systems in which the consistency model is relaxed.
- Unsuited to business intelligence. The document store won't work with existing reporting and OLAP tools. Data in these systems is relatively isolated from normal business intelligence (BI) practices. The absence of a complete SQL layer that can access these systems isolates them from the broader enterprise.

There are several studies on different concurrency and consistency models for NoSQL databases (Padhye & Tripathi, 2012; Lin et al., 2014). As some of those models are hard to implement and the demand for ACID transaction for many applications exists, there is a trend to include transactions on distributed new databases (Neo4j , 2016; Oracle, 2016).

The Beckman 2014 report on database research recommended attention to five research areas (Abadi et al., 2016):

• Scalable big/fast data infrastructures.
• Coping with diversity in the data management landscape.
• End-to-end processing and understanding of data.
• Cloud services.
• Managing the diverse roles of people in the data life cycle.

4 Conclusion

The main driving forces behind the third database generation development are Web 2.0 applications, social networks, Big Data, cloud computing and Internet of Things. Modern database are continuously challenged to meet the needs of applications that demand an unparalleled level of scale, availability, and throughput.

Beside all their limitations, NoSQL databases proved that they can be valuable solutions for different Big Data problems. Big Data players like Google, Amazon, Facebook, Twitter, etc., were first faced with the limitations of relational databases in solving their request, and they have become pioneers in developing and implementing different NOSQL databases. But, it is important to stress out that leading RDBMS vendors (like Oracle) very carefully monitor everything that happens in NoSQL world, and in their systems implement some of the fundamental features of NoSQL databases (JSON interface, shared-nothing sharded distribution, graph compute engine, Hadoop support end, etc.). The next years will show future development of database technology going in the direction of further divergence or convergence towards some unified or hybrid database model.

References

Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P.A., Carey, M..J., Chaudhuri, S., Dean, J., Doan, A., Franklin, M.J., Gehrke, J., Haas, L.M., Halevy, A.Y., Hellerstein, J.M., Ioannidis, Y.E., Jagadish, H.V., Kossmann, D., Madden, S., Mehrotra, S., Milo, T., Naughton, J.F., Ramakrishnan, R., Markl,V., Olston, C.,Ooi, B.C., Ré, C., Suciu, D., Stonebraker, M., Walter, T., Widom, J.(2016). The Beckman Report on Database Research. *Communications of the ACM*. Vol. 59 No. 2, Pages 92-99. Retreived from: http://cacm.acm.org/magazines/2016/2/197411-the-beckman-report- on-database-research/fulltext.

Ameri, P. (2016). Database Techniques For Big Data. In book ed. Buyya,R., Calheiros, R.N., Dastjerdi, A.V. *Big Data Principles and Paradigms.* Elsevier Inc., USA.

Bailis P, Ghodsi A. (2013) Eventual consistency today: limitations, extensions, and beyond. *Communications of the ACM*, May, pp.55–63.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, 39(4), 12-27.

Celko, J. (2014). *Joe Celko's complete guide to NoSQL : what every SQL professional needs to know about nonrelational databases.* Elsevier Inc., Kindle edition.

Chao, X., Chunzhi, S., Kapritsos, M., Wang, Y., Yaghmazadeh, N., Alvisi, L., Mahajan, P. (2014). Salt: Combining ACID and BASE in a Distributed Database. *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*. October 6–8, Broomfield, CO.

Datastax (2016) CQL for Cassandra 2.1/DataStax CQL 3.1.x Documentation. Cassandra. Retreived from: https://docs.datastax.com/en/cql/3.1/cql/cql_reference/cqlReferenceTOC.html

Dean, J., Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Commun ACM*.January, pp.107–13.

Fowler, A. (2016). *The State of NoSQL 2016: A quick guide to the NoSQL landscape*. Kindle Edition.

Harrison, G. (2015). *Next Generation Databases: NoSQL, NewSQL, and Big Data*. Apress. Kindle Edition

Lin N, Dongming L, Yongqi H. (2014). Optimization method of concurrency scheduling of graphical database transaction. *In: 2014 international conference on computer science and electronic technology (ICCSET 2014)*. pp. 22–6.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute. Retrieved from: file:///C:/Users/user/Downloads/MGI_big_data_full_report.pdf

McCreary, D. & Kelly, A. (2014). *Making Sense of NoSQL*. Kindle Edition.

Moniruzzaman, A. B. M. & Hossain, S.A. (2013). Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. *International Journal of Database Theory and Application.* Vol. 6. No. 4.

Neo4j (2016) Neo4j Graph Database. Retrieved from: http://neo4j.com

Oracle (2016) Oracle Berkeley DB. Retrieved from:http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

Padhye V, Tripathi A. (2012) Scalable transaction management with snapshot isolation on cloud data management systems. *In: Proceedings of IEEE 5th International conference on cloud computing*. 24-29 June.

Pritchett, D. (2008) BASE: An acid alternative. *Queue*. Volume 6 Issue 3, May/June, Pages 48-55.

Redis. (2016) Redis. Available from: http://redis.io/

Redmond, E. & Wilson, J.R. (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. Pragmatic Bookshelf. Kindle Edition.

Shute, J., Oancea, M., Ellner, S., Handy, B., Rollins, E., Samwel, B., Vingralek, R., Whipkey, C., Chen, X., Jegerlehner, B., Littleeld, K., Tong,P. (2012). F1- The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business. *In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. pages 777–778.ACM.

Simon, S. (2012). Brewer's CAP Theorem: Report to Brewer's original presentation of his CAP Theorem at the Symposium on Principles of Distributed Computing (PODC) 2000. CS341 Distributed Information Systems University of Basel, HS2012

Vanroose, P., Thillo, K.V. (2014). ACID or BASE? - the case of NoSQL, GSE DB2 Belgium Joint User Group Meeting IBM. Brussels, 12 June,"What's next?". Retreived from: http://www.abis.be/resources/presentations/gsebedb220140612nosql.pdf

Wu, C., Buyya, R., Ramamohanarao, K. (2016). BDA = ML + CC. In book ed. Buyya,R., Calheiros, R.N., Dastjerdi, A.V. *Big Data Principles and Paradigms*. Elsevier Inc. USA.