The Challenges of NoSQL Data Warehousing

1st Lucija Petricioli
University of Zagreb, Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia
lucija.petricioli@fer.hr
ORCID: 0000-0003-1878-4969

2nd Luka Humski
University of Zagreb, Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia
luka.humski@fer.hr
ORCID: 0000-0002-6819-8899

3rd Boris Vrdoljak
University of Zagreb, Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia
boris.vrdoljak@fer.hr
ORCID: 0000-0003-0081-172X

***Abstract*—Data warehouses are an important part of decision support systems in business. The volume of data currently being created can at times push the capabilities of relational data warehouses to their limits. A possible step forward is to use NoSQL solutions to model data warehouses, since they were made for the ever-increasing amount of data that various platforms deal with. However, simply deciding a data warehouse should be based on a NoSQL approach does not mean the problem has been solved. The flexibility of NoSQL leads to a host of new problems, such as how to perform various OLAP operations on a data warehouse that does not have a fixed schema or how and when to compute aggregate values. This paper provides an overview of various solutions that have been theorized and presented along with their advantages over relational data warehouses, as well as their drawbacks.**

Keywords*— *NoSQL, data warehouse, OLAP, Big Data

I. Introduction

Every day more and more data are created: data about orders, sales, shipments, e-commerce website clickstreams, etc., each carrying the potential of improving business practices and providing new insights into current operations. However, the increasing volume of data can cause strain to well-established systems that have been in place for years. Relational data warehouse systems at times cannot cope with Big Data due to their inherent properties: the speed they are created at, their heterogeneity, their sheer amount, etc. [1, 2]. A shift to a newer paradigm is tempting and even beneficial in some cases, but not without its own set of problems.

Today’s answer to efficient Big Data storage is NoSQL (Not only SQL) databases. Their four most common and well-accepted types are [3]:

- key-value databases, which store data as key-value pairs;
- column-stores, which store attribute values as columns instead of rows;
- document-oriented databases, which store data as documents made up of tagged elements;
- graph databases, which store data as edges and nodes.

Regardless of type, they promise flexibility and scalability, two aspects that can ease the issues connected to Big Data; issues which relational databases can hardly cope with. These aspects have even resulted in more and more companies adopting NoSQL databases as their preferred type of storage. Nevertheless, these capabilities are not enough to start a mass migration of data warehouses from the relational to the NoSQL paradigm. Since data warehouses are a part of decision support systems, which need careful planning, as well as require certain capabilities of the technologies used to develop them (that NoSQL might not offer), the shift to NoSQL (if necessary) has to be a well-thought out process.

Generally, best practices (e.g. star schema) and benchmarks made for data warehouses have yet to migrate to the NoSQL world. Best practices are often either hard to implement in NoSQL or cannot be implemented due to the differences between the relational and non-relational approach [3, 4]. Data warehouse benchmarks have started a shift towards including NoSQL, albeit a slow one [5, 6]. The solutions currently in place are mature and well established; support is readily available for them, and, unlike NoSQL systems, there are many experts that can use them easily, and develop for them. The lack of NoSQL experts is another issue; NoSQL databases are still relatively new, and they are still in flux. New versions of the same database, new implementations of a NoSQL concept and a change in approach (especially if it is to the same NoSQL database) are frequent occurrences, making NoSQL databases a tough field to gain expertise in [7]. Moreover, since there are more types of database to choose from, selecting the appropriate technology for a task can become a somewhat complicated task.

These are only a small fraction of the reasons a greater adoption of NoSQL systems for data warehousing has not been seen in recent years. This paper aims to highlight the various advantages and drawbacks of switching over to NoSQL technologies. It is organised as follows: section II. lists the advantages NoSQL data warehouses have over relational data warehouses; section III. lays out various approaches to creating a NoSQL data warehouse and the various drawbacks that occur when doing so; and section IV. gives a conclustion to this paper.

II. NoSQL Advantages over Relational Data Warehouses

Not so long ago, relational data warehouses were more than sufficient for all data analyses companies were eager to conduct. Hardware improvements were promising to speed up calculations and allow more data to be processed. However, some cases have surfaced in which the amount and type of data to be processed (in real time) exceeds the capabilities of relational data warehouses, even though they are still sufficient for various analyses.

A star schema represents the way relational data warehouses are usually modeled. An example of a star schema can be seen in Fig. 1. It dictates the structure of the data that are stored into the data warehouse and the queries that can be applied to the data warehouse.

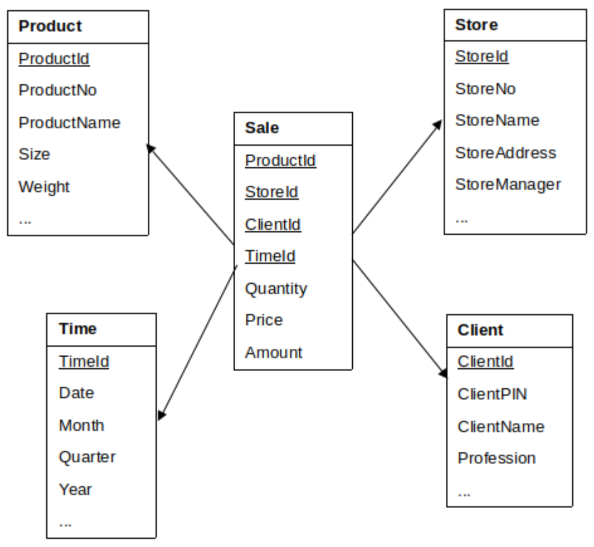


Fig. 1. Example of a star schema used for tracking the sales of a company.

The data in a relational data warehouse should be structured, because semi-structured or unstructured data can be difficult to process and extract information from [1–3, 8–12], which is becoming more and more problematic due to a surge in the amount of collected unstructured data [1, 2]. Relational data warehouses offer very little flexibility; every change has to be done to the schema directly, which means agile development methods based on a data warehouse are mostly out of the question [3, 12]. Sometimes, the adjustment or expansion of a schema that would be necessary due to new requirements cannot be done in a meaningful manner [9]. More often than not, using relational data warehouses in a distributed setting leads to data duplication and noticeable increases in query processing times due to having to fetch table fragments from various nodes [4, 9–11]. Established solutions can come with high licensing costs (if proprietary) [9], and they can be slow to implement due to experts being necessary to do so [3]. In particular, they require a fair amount of specific infrastructure (e.g., data marts, Extract-Load-Transform – ETL processes, On-Line Analytical Processing – OLAP, reporting tools, etc.) and experts that are well versed in their development [12]. The experts also have to work together closely to develop a functional system. Furthermore, relational data warehouses' rigid structure can lead to some tables being heavily populated by NULL values. NULL values become quite problematic if they represent any dimension value or if they are part of a foreign key in any of the fact tables, so much so data warehouse designers try to avoid them by assigning them different values, redefining them or even adding special rows to the dimension tables [10]. Finally, joins can still occur in normalized relational data warehouses, bringing with them a steep price due to needing time and memory resources [10, 11]. This is why denormalized relational data warehouses are more popular.

On the other hand, NoSQL data warehouses are highly scalable and very flexible [2, 8–10, 13–15]. Their horizontal scalability is one of their greatest strong points; a new node can be added to a distributed system without problems and the database can utilize the disk of the node right away. Both the scalability and the flexibility of NoSQL databases result in much less data duplication than a relational database would require. Any new record can be written onto any disk the database has access to. They do not store NULL values [8, 10], they simply omit an attribute from a record if its value is unknown, which results in less disk usage. The data that are stored into NoSQL databases do not need to be transformed into a certain format, which skips the Transform part of ETL processes [7–9] and considerably speeds up the process of data loading. The databases' write operations are considerably faster (partially due to not performing data transformation) [3, 7, 8, 14, 17], and they can store substantial amounts of data (connected to their easy scalability) [3, 8, 14].

All the listed benefits of using NoSQL databases make them suitable for storing Big Data. These data have properties that are often described as "Big Data's N Vs", where N started as three and climbed up to seven. The full "Seven Vs of Big Data" are as follows [1, 2]:

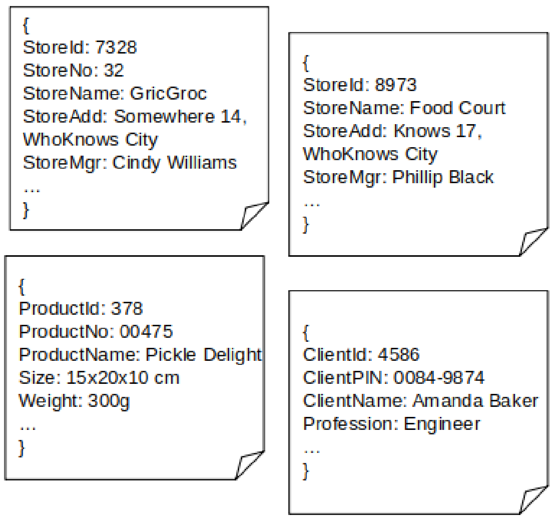
- volume – Big Data is characterized by large amounts of data;
- variety – the data are of different formats and have different levels of structure;
- velocity – the data are generated quickly;
- veracity – the data can be inconsistent or incorrect;
- variability – the data can be interpreted in various ways;
- value – the usefulness of the data cannot always be established;
- visualization – the data need to be visualized to improve or even allow greater understanding.

Even just the first three Vs can cause problems when trying to store Big Data into relational databases, either as transactional databases or as data warehouses. On the other hand, those three Vs are easily solved by using a NoSQL database (the rest of the Vs have to be handled in a different manner – e.g., using an external visualization tool).

III. Different Thoughts on NoSQL Models, Approaches, and Drawbacks

When looking at all the proposed solutions, a clear preference can be seen: authors either use a column-store [5–7, 10, 13, 14, 16, 17, 18] or a document-oriented database [3, 4, 8, 11, 12]. Some uses of graph databases have started to surface, but those are mainly applied to social networks [11]. The following paragraphs summarize the research done on various aspects of using NoSQL databases as data warehouses.

Fig. 2 illustrates the structure (or lack thereof) of a document-oriented database. As can be seen, not all fields are present in documents that are connected to similar entities (e.g., the two documents starting with the attribute "StoreId" – one of them lacks the attribute "StoreNo").



```
{
StoreId: 7328
StoreNo: 32
StoreName: GricGroc
StoreAdd: Somewhere 14,
WhoKnows City
StoreMgr: Cindy Williams
…
}
```

```
{
StoreId: 8973
StoreName: Food Court
StoreAdd: Knows 17,
WhoKnows City
StoreMgr: Phillip Black
…
}
```

```
{
ProductId: 378
ProductNo: 00475
ProductName: Pickle Delight
Size: 15x20x10 cm
Weight: 300g
…
}
```

```
{
ClientId: 4586
ClientPIN: 0084-9874
ClientName: Amanda Baker
Profession: Engineer
…
}
```

Fig. 2. Illustrative example of a document-oriented database used for tracking the sales of a company.

Reference [4] considers how different ways of organizing data in a document-oriented data warehouse impact OLAP queries. They also develop OLAP cuboids that were previously theorized, but could not be implemented on a relational data warehouse, namely the *nested cuboid* and the *detail cuboid*. Although the authors speculate drill-down operations would be much faster on their proposed cuboids, they run into some implementation problems (one document in MongoDB cannot exceed 16 MB) and the cuboids require more memory and their computation lasts longer than traditional cuboids. When considering the drawbacks of NoSQL, the authors point out that NoSQL data warehouses require more storage space in comparison with relational data warehouses due to duplicating the names of the attributes in every document. They also find any operation in need of joins is greatly slowed down and any optimization can strictly be done manually. In [5], the authors are mostly concerned with adapting a popular relational data warehouse schema benchmark to the NoSQL paradigm. However, they do note that the cost of joins in NoSQL is prohibitively slow and the referential integrity constraint does not exist, leading to issues when adapting the benchmark. The authors of [6] continue the work of [5] and test how the data should be stored in a NoSQL data warehouse to achieve fast querying. They conclude that the solutions greatly vary depending on what needs to be processed and queried, mostly due to trying to avoid joins in NoSQL. In [7], the authors design a prototype of a Twitter data NoSQL data warehouse. They once again try to avoid executing joins and calculating aggregates during query execution. They also comment on not being able to use the newest NoSQL technology due to certain incompatibilities within the software. Reference [13] engages in developing an approach that would directly translate a conceptual multidimensional model into a NoSQL logical model that would be part of an OLAP system (such translations usually use a relational logical model as an intermediate step). Even after extensive experimentation, the authors conclude it is difficult to draw detailed recommendations as to when column-oriented or document-oriented databases could be used for OLAP systems. Reference [14] lists many benefits to using a NoSQL database as a data warehouse and shows a performance comparison between Cassandra and OracleDB. Cassandra is shown to be considerably faster when working with larger amounts of data, but the study does not address potential drawbacks or specifics to using a NoSQL data warehouse. In [15], the authors list various approaches to develop an enriched NoSQL data warehouse. They settle on ontologies as a way of achieving their goal. They contemplate automating the process of data warehouse creation, but they conclude that NoSQL could be semi-automated at best due to the semantics being mixed in with the data. The inconsistency and lack of structure of NoSQL databases complicates any kind of schema modeling. They consider adding a NoSQL data warehouse to an existing relational one a viable option. The authors of [16] propose a new OLAP operator for columnar data stores, since such databases do not have any OLAP operators. They need to use an external application to compute the proposed CN-CUBE (Columnar NoSQL CUBE) because the column store used for prototyping (HBase) cannot handle the various aggregates the cube operator requires. Finally, in [17], similarly to [4, 6], the authors experiment with different ways of organizing data in a NoSQL data warehouse and measure the performance of each type of solution. Once again, they evade joins.

Some authors take a more theoretical approach to NoSQL data warehouses, but they still highlight some issues that arise. Reference [1] offers an overview of various efforts that have been made in the field of storing Big Data meaningfully (in a way that renders them usable). The authors list NewSQL and various middleware that can connect different data sources and present them in a unified manner to the user as potentially interesting solutions. Ultimately, they predict a symbiosis of relational and NoSQL technologies in future data warehouses as neither of the two approaches can fully replace the other due to their inherent differences. The authors of [2] explore the possibility of using MapReduce along with NoSQL databases in a feat to enrich structured data with information extracted from unstructured data. They mention the lack of a declarative query language and variety of implementations as NoSQL characteristics that are difficult to overcome. They add that users often have to write their own custom programs to be able to have at least some ability of running ETL processes on a NoSQL database. In [3], the authors explore the differences between relational and NoSQL databases and they try to focus on which characteristics a reporting tool for NoSQL should have. They ultimately conclude a “best-of-both-worlds” approach may be beneficial because it could mitigate the drawbacks of both methods. The main drawback of NoSQL they identify is that the technologies have immense problems with join operations and aggregate functions due to their loose structure, which makes the development of a reporting tool challenging. The author of [8] gives a general overview of available technologies and why a shift to NoSQL might be a good option. However, the paper indicates that the schemaless structure of NoSQL only shifts the responsibility of keeping some sort of structure onto the developer, along with many aforementioned drawbacks of using NoSQL for data warehousing. The author of [9] also takes a general approach to why a change of data warehouse technology would be pertinent, but, as several other references [1–3], proposes a combined approach, adding that NoSQL databases’ query languages are still severely lacking in capabilities as opposed to regular SQL. Reference [10] is concerned with migrating OLAP queries into the NoSQL world. The author notes that the existing approaches that lead from a conceptual model to a data warehouse are tailored to relational data warehouses, making them difficult to translate to NoSQL. They propose a new approach that would skip the relational phase of modeling, but the resulting method requires a new table whenever new information is needed from the model, which means a lot of data duplication and a need for larger storage capacities. In [11], they propose an approach merging document-oriented databases and graph databases to accurately store the data produced by social networks. The authors theorize this solution would be fast and could store both relationship data and content produced by users, but they do mention that the lack of a standardized query language and the differences between NoSQL data models in general make such development difficult. Reference [12] is concerned with achieving agile development over a data warehouse, and the authors note they are willing to sacrifice joining and aggregate values for the sake of this type of development. They propose a universal browser, a reporting tool that could function with a NoSQL data warehouse. One of the key features of this universal browser would be the option of a “Google-like” search they feel they could not create with a relational data warehouse. When discussing the drawbacks of NoSQL, they admit the Transformation in ETL cannot be avoided completely, since some structure has to be present in the data warehouse for it to be useful.

Another problem most of the authors do not address is that, depending on the chosen NoSQL implementation, joins, aggregate functions or both might not even be supported. HBase [5, 6, 13, 16, 17, 18] and MongoDB [3, 4, 12, 13] are the technologies used in most of the papers due to having some support for the aforementioned functions. Three of the referenced papers use Cassandra, and one of them openly states that joins and aggregate functions are not supported [7]. Reference [10] does mention support for some aggregate functions, but does not verify the existence of join operations. The disparity between the two studies is likely due to NoSQL being a fast-evolving field, so some functionalities were most likely added between the writing of the two studies. The author of [14] does not address the issue.

The one reference that moves past just theory and prototypes is [18]. The authors present a NoSQL data warehouse made for integrating various sources of patient data measured and observed in clinical trials. There are several reasons they opt for this approach: there are many data that get produced by clinical trials; the data are heterogeneous due to differences in what is tracked and measured in a study; the studies can change mid-trial and require new attributes to be tracked. They use a modification of ETL – ELT or Extract-Load-Transform. They identify that the data stored in the data warehouse have to be in a certain format once they are accessed, so they provide a browsing tool that stores various data mappings that can be applied to the data once they are fetched from the data warehouse. Once the data have been mapped according to the requirements of a user, they suggest storing them in an application data mart – this way, the correctly formatted data are quickly accessible and do not need to be reformatted for each access. However, some minor transformations still need to be made before the data are stored into the NoSQL data warehouse, namely the extraction of rows of data and flattening of hierarchical structures within the data into individual rows. They mention the ideal solution would be one that would have the flexibility of a NoSQL store and the querying efficiency of a relational database. They achieve a similar solution by using HBase as the NoSQL store and Apache Phoenix as the relational query engine which can translate ANSI SQL queries into HBase table scans, although it seems the authors use the data warehouse purely for storage; all the transformations and joins happen outside of it.

IV. Conclusion

NoSQL is a new, promising approach that could improve data warehousing in modern times. However, some glaring issues still remain, so it requires more effort to become a regularly used and completely viable solution. Since neither relational data warehouses nor NoSQL data warehouses can confidently take each other’s place, a combined solution would probably be the most useful way forward. It could combine the speed and easy scalability of NoSQL with the reliability and computing capabilities of relational data warehouses. Also, columnar stores seem to be the preferred technology when developing a NoSQL data warehouse since they resemble widely used relational data warehouses the most, i.e. share some characteristics with them (primarily similar structure with rows and columns).

References

[1] M. Pticek and B. Vrdoljak, “Big data and new data warehousing approaches,” in *ACM International Conference Proceeding Series. Association for Computing Machinery*, Sep 2017, pp. 6–10.

[2] M. Pticek and B. Vrdoljak, “MapReduce research on warehousing of big data,” in 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017 - Proceedings. Institute of Electrical and Electronics Engineers Inc., Jul 2017, pp. 1361–1366.

[3] Z. Bicevska and I. Oditis, “Towards NoSQL-based Data Warehouse Solutions,” in *Procedia Computer Science*, vol. 104. Elsevier B.V., Dec 2016, pp. 104–111.

[4] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, and R. Tournier, “Document-oriented Data Warehouses: Models and Extended Cuboids; Extended Cuboids in Oriented Document,” in *Proceedings – International Conference on Research Challenges in Information Science*, vol. 2016–Augus, 2016.

[5] K. Dehdouh, O. Boussaid, and F. Bentayeb, “Columnar NoSQL Star Schema Benchmark,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8748, pp. 281–288, 2014.

[6] L. C. Scabora, J. J. Brito, R. R. Ciferri, and C. D. De Aguiar Ciferri, “Physical data warehouse design on NoSQL databases: OLAP query processing over HBase,” in *ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems*, vol. 1, 2016, pp. 111–118.

[7] M. R. Murazza and A. Nurwidyantoro, “Cassandra and SQL database comparison for near real-time Twitter data warehouse,” in Proceeding - 2016 International Seminar on Intelligent Technology and Its Application, ISITIA 2016: Recent Trends in Intelligent Computational Technologies for Sustainable Energy, 2017, pp. 195–200.

[8] G. Chandwani, “Nosql Data-Warehouse,” International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, spec. issue no. 4, pp. 96–104, 2016.

[9] S. Müller, “Die neue Realität: Erweiterung des Data Warehouse um Hadoop, NoSQL & Co,” *HMD Praxis der Wirtschaftsinformatik*, vol. 51, no. 4, pp. 447–457, Aug 2014.

[10] D. Prakash, “NosoLAP: Moving from data warehouse requirements to NoSQL databases,” in ENASE 2019 - Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, 2019, pp. 452–458.

[11] H. Akid and M. B. Ayed, “Towards NoSQL Graph Data Warehouse for Big Social Data Analysis,” in *Advances in Intelligent Systems and Computing*, vol. 557. Springer International Publishing, 2017, pp. 965–973.

[12] Z. Bicevska, A. Neimanis, and I. Oditis, “NoSQLbased Data Warehouse Solutions: Sense, Benefits and Prerequisites,” *Baltic Journal of Modern Computing*, vol. 4, no. 3, pp. 597–606, 2016.

[13] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, and R. Tournier, “Implementing multidimensional data warehouses into NoSQL,” in *ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings*, vol. 1, 2015, pp. 172–183.

[14] J. KURPANIK, “NOSQL DATABASES AS A DATA WAREHOUSE FOR DECISION SUPPORT SYSTEMS,” *Journal of Science of the Gen. Tadeusz Kosciuszko Military Academy of Land Forces*, vol. 49, no. 3, pp. 124–131, 2017.

[15] M. Pticek and B. Vrdoljak, “Semantic web technologies and big data warehousing,” in 2018 41st International Convention on Information and Communication Technology Electronics and Microelectronics, MIPRO 2018 - Proceedings, 2018, pp. 1214–1219.

[16] K. Dehdouh, F. Bentayeb, O. Boussaid, and N. Kabachi, “Columnar NoSQL CUBE: Agregation operator for columnar NoSQL data warehouse,” in *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Jan 2014, pp. 3828–3833.

[17] K. Dehdouh, F. Bentayeb, O. Boussaid, and N. Kabachi, “Using the column oriented NoSQL model for implementing big data warehouses,” *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA’15)*, pp. 469–475, 2015.

[18] E. Yang, J. D. Scheff, S. C. Shen, M. A. Farnum, J. Sefton, V. S. Lobanov, and D. K. Agrafiotis, “A late-binding, distributed, NoSQL warehouse for integrating patient data from clinical trials,” *Database*, vol. 2019, no. 1, pp. 1–16, 2019