Baltic J. Modern Computing . Vol. 4(2016), No. 3, 597-606

NoSQL-based Data Warehouse Solutions: Sense, Benefits and Prerequisites

Zane BICEVSKA, Andrejs NEIMANIS, Ivo ODITIS

DIVI Grupa Ltd, 40-33 Avotu Street, Riga, LV-1009, Latvia

Zane.Bicevska@di.lv

Abstract. The paper discusses the possibilities of using non-relational databases as data warehouses' (DW) data mart – an area traditionally covered by relational databases. The authors highlight potential benefits when using non-relational databases (NoSQL) technology based on the main characteristics of classical relational data base management system (RDBMS) based DW. The paper outlines a creation and production process for DW using a NoSQL data mart and develops requirements for technology necessary for such processes based on practical experience when implementing NoSQL data marts with MongoDB and Clusterpoint DB. Since there currently do not exist of-the-shelf reporting solutions for most of the NoSQL data storages the authors develop requirements for a reporting solution necessary for NoSQL based DW.

Keywords: data warehouse, NoSQL data marts, non-relational databases, universal data browsing, data denormalization.

Introduction

A data warehouse can be defined as "as a central federated repository for all (or significant parts of) the data that an enterprise's various business systems collect - it may be both physical and logical" (Bridgwater, 2015). This definition does not imply the usage of specific platforms or technological means. Nevertheless the Oracle Corporation in the "Data Warehousing Guide" (Lane, 2002) states: "A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing." This definition reflects the prevalent understanding of data warehouses (DW) as typical relational data storages that are processed using means of RDBMS.

NoSQL refers to any database alternative to the relational model that arranges data discretely into tables of columns and rows. Four different kinds of NoSQL databases can be distinguished - key-value stores, document databases, column-oriented databases, and graph databases (Padhy et al., 2011). Key-Value Stores (Seeger, 2009) use big hash tables of keys and values to ensure fast accessing of data; Riak and Amazon's Dynamo are the most popular key-value store NoSQL databases. Document-based NoSQL databases (Pokorny, 2013) store documents made up of tagged elements; Couchbase and MongoDB are the most popular document-based databases. Column-based stores (Abadi et al., 2009) ensure that each storage block contains data from only one column; Google's BigTable, HBase and Cassandra are the most popular column-based databases. Graph-based (Vicknair et al., 2010) are network databases that use edges and nodes to represent and store data; InfoGrid, Infinite Graph and Neo4J are the most popular graph-based databases. NoSQL is commonly associated with more flexible deployment and structure, faster read and write performance as well as scaling to very large data sets.

Recent studies outline the trend towards increasingly growing market share for NoSQL databases. For instance, (Assay, 2015) shows how the NoSQL increases their market share, mostly at the expense of MySQL. (Woodey, 2014) comes to the conclusion: “NoSQL databases are with us to stay. As hyperscale applications and data volumes surpass the capabilities of traditional relational databases, companies have no choice but to alternatives.” Although the maturity of the NoSQL market is assessed to be rather low yet, the Forrester Wave estimated the current adoption of NoSQL to be at 20% in 2014 and saw it doubling by 2017.

For the present the NoSQL databases are mostly perceived as high performance data structures, suitable for look-up and filtering operations, not as wholesome database management systems applicable for complex data processing activities. But it is just a matter of time to build reliable business solutions covering the critical and supporting business functionality. Data warehouse is one of such potential areas, so this paper is devoted to creating of NoSQL-based DW using document-based NoSQL data stores.

The first chapter of the paper outlines the nature of classical RDBMS based DW and identifies several benefits to be expected if using NoSQL technologies instead.

The second chapter describes the creation and production process for DW using a NoSQL data mart and highlights the necessity to use cross-platform de-normalization technology. By contrasting the classical DW creation process the paper identifies the possibility of using agile development approach.

The third chapter deals with the reporting necessary for a NoSQL based DW. Since there do not exist ready-to-use solutions the paper proposes requirements for according support. These requirements reflect on the behavior of according tools for classical DW but add NoSQL-specific features.

The proposed approach has been validated in several pilot projects, the gained results are summarized in the conclusions chapter.

1. Sense of NoSQL-based DW

Typically an organization is using several IT systems every of them covering a different functional segment (e.g. Accounting, HR, CRM). The need of such organization’s management for more integrated information will often lead to the decision for building a DW. Since most of today’s applications use RDBMS data storage the decision to base the DW on a RDBMS data mart seems obvious. As we will conclude later such a decision is compelling if the DW solely shall integrate only some RDBMS based data but it is not self-evident in times of more and more information becoming relevant for the organizations’ decisions (Han et al., 2011).

Let us recapitulate the essence of a classical DW (Jarke et al., 2003). Roughly speaking the existing DW technology divides all collected data in two groups.

On the one hand there is data that can be measured – e.g. costs, temperatures, speed. But such data - in DW terminology called facts - will have no business value if not used in context of time intervals and/or other describing attributes. On the other hand only the describing data - in DW terminology called dimensions – gives meaning to facts.

Desirably all the describing data can be related directly as foreign keys to the facts, called star scheme, but sometimes the available data requires more sophisticated DW, called snowflake scheme (Levene et al., 2003) . The benefit of the classical DW technology in comparison to any self-made data integration is the built-in flexibility when combining facts with dimensions – the DW user might not even need to have deep business knowledge do get benefits from it.

Nothing in the discussion so far indicates that NoSQL technology could provide any benefit in comparison to classical DW technology. To identify such benefits it is necessary to recollect some of the main reasons why NoSQL technology has become popular. As stated by one of the NoSQL market leaders MongoDB one of such benefits is “the ability to handle high volumes of structured, semi-structured, and unstructured data” (https://www.mongodb.com/scale/benefits-of-nosql)

Classical DW technology implicitly enforces all describing data to be structured and cannot really deal with semi-structured and non-structured data. Of cause it will always be possible to implement a work-around for specific situations but such solutions are not flexible by definition (Baars et al., 2008).

Today’s pampered Google-used users will expect a DW solution to act more in the Internet browser style than to enforce the user to act within pre-defined structures. Users will expect the DW to provide answers even they had entered nothing than search texts. Since NoSQL technology by nature provides a good support regarding semi-structured and unstructured data (Kaur et al., 2013) we shall investigate later in the document possibilities to combine the text search capabilities of NoSQL technology with possibilities to receive something similar as provided from a classical DW.

On the other hand we must be aware that NoSQL data storages by their nature are not created for to optimize numerical operations for a big count of data records. Although all the NoSQL storages support several numerical data formats and allow operations like SUM, AVG and COUNT that support is always worse than classical DW support for facts (Leavitt, 2010).

Summarizing the discussions so far we can expect benefits when using NoSQL technologies if a substantial part of the data collected in the data mart is semi-structured or non-structured and if we do not face too specific requirements regarding the facts necessary in the DW.

2. Implementation of NoSQL-based DW

Another benefit often mentioned in connection with NoSQL is the high flexibility in the development process (Fehling et al., 2011). MongoDB promotes this aspect with “working well with today's software development methodologies that involve agile sprints and frequent code pushes” (https://www.mongodb.com/scale/benefits-of-nosql).

Later in this chapter we will outline an approach to enable agile-style DW development when using NoSQL technology. But let us first discuss the design and implementation of a classical DW recognizing that there are two main reasons retaining a flexible design and implementation process.

The first reason is that there is a need for a lot of specific infrastructure to design and implement a classical DW (Mrdalj, 2011). It requires a whole bunch of technology to set up a classical DW (data mart, ETL, OLAP, reporting tools) and all these components require several IT specialists to work coordinated. Before an end-user gets even a trivial report generated from a classical DW the major part of the DW implementation project time and money will have already been spent.

The second reason is the specific knowledge necessary to set up a classical DW (Fahey et al. 1998). Especially the creation of DW dimensions out of the data provided from specific applications is non-trivial intellectual handwork. The DW designer has to deal with several aspects like time dimension design, grouping/banding, parent/child hierarchies etc. Data issues not known or addressed by the users come up and have to be solved. Although there is a huge amount of literature covering the several design aspects (Devlin et al., 1996), (Todman, 2000), experienced DW specialists still are the best weapon when dealing with the DW implementation. The dimension design and implementation usually is the most time-consuming part in any DW project and, unfortunately, also the part being most inflexible when it comes to changes (Golfarelli et al., 1998). Agile style development of classical DW against this background is nearly impossible (Rahman et al., 2013).

When considering the necessary environment for a NoSQL based DW we will recognize that beside the data mart it requires a methodology and a technological component to transport the involved RDBMS data to the NoSQL data mart. It's obvious that it makes no sense to rebuild the relational data structures when creating the NoSQL based data mart. De-normalization will be necessary and that de-normalization should be as easy to handle as possible (Shin et al., 2006).

Practical investigations by the authors in this direction have been done by implementing and using a prototype for a cross-platform de-normalization tool (Karnitis et al., 2015). Meanwhile the tool is able to analyze the data structures and relations of several standard RDBMS (MS, Oracle, MySQL, etc.) by analyzing the according meta-data of the RDBMS. The tool allows the user to set up de-normalized data structures without programming simply by browsing data trees or drill-down of the according data, and it lets to export the selected data as XML- or JSON-documents.

The lessons learned can be summarized as follows:

- As expected it is absolutely no problem to create and export denormalized data structures when the describing data is related 1:N. If we think about a Human Ressources system as source it is no big deal to export the several attributes (name, surname, gender, ...) of a person into a flat file.
- Self-referencing data cannot mechanically be denormalized since it would produce endless referenced document chains. Similar to the approaches used in classical DW design it is recommended either to flat the source data by preparing a according view or to denormalize the references only in one direction (e.g. in a management hierarchy store only the respective superior or only the inferiors).
- Pure mechanical approaches end when we face M:N-related data since they carry the risk of producing gigantic amounts of redundant data during denormalization. The decision for the best denormalization approach in a given case will not be only technically driven but will also have to take in mind the user priorities.
- As less hierarchical levels during denormalization are created as more convenient the data will be for reporting.
- Aggregatable numerical data is the biggest challenge in the data mart design process. Since it is self-evident that such data cannot be stored redundant in the data

mart intelectual effort has to be spent to find a workable compromise. When numerical data simply appears in different hierarchy levels (e.g. a bill containing positions and totals) it is not difficult to find any denormalization solution but we face real challenges when there also appear M:N-relations.

Since the de-normalization tool does not only export the data in denormalized form but also stores the meta-data describing the export it is obvious to use that description for generating structures in the NoSQL data mart. Our practical investigations in this field, using both Mongo DB and Clusterpoint platforms, showed that it is generally possible. Since such a feature unifies the design and execution of data mapping it opens the door for an agile development approach.

Of cause not all data from a RDBMS source can be transported 1:1 to a DW – but the same way we would act in case of a classical DW the necessary transformations can be done using data base views or stored procedures. We also recognized that automatically generated data structures on destination side not always are optimal – for a productive use adjustments were necessary to enable the NoSQL data base's full capabilities.

Nevertheless our conclusion is that even a non-optimal structure generation for the data mart is a very worthwhile tool when thinking about the possibility to use an agile development approach.

Summarizing this chapter we can state that NoSQL technology has the perspective to allow agile development approaches under the condition that we have tools supporting the de-normalization for the respective NoSQL based data mart.

3. Reporting for NoSQL-based DW

One of the big strength of classical DW technology is the flexibility regarding the reporting (Chen et al., 2000). Starting from very simple solutions like simply using EXCEL as front-end and ending up with high sophisticated reporting tools the market offers a huge spectre of solutions. Anybody will be able to find a solution solving the requirements of his organization and the search for the best fit in most cases will be more economical than a technical task.

When investigating opinions about the disadvantages of NoSQL technology the lack of reporting support will always be located in top positions (Han et al., 2011). The authors' initial impression when searching for reporting tools fully matched such opinions. Although the authors are located in a small country and basically work with DW for SME even the search for a very simple reporting tool was staggering.

As recourse some effort from authors side was spent to build a simple universal browsing tool. Although these activities were far from the effort necessary to build a stand-alone product the obtained knowledge allows them to define some requirements a customer would expect from a reporting tool supporting a NoSQL data mart.

Of cause the implementation of mentioned prototype had to consider the specific NoSQL data storage; in the research it was Mongo DB. Nevertheless the authors are able to identify several requirements that are independent from the chosen NoSQL data storage, now.

It makes sense to formulate some requirements regarding the data to be stored in a data mart helping to simplify the requirements regarding a universal DW browsing tool.

- Not without reasons classical DW technology offers functionality that allows to hide the technical naming of data objects from the end users. Applying this approach for the NoSQL data storage means that the data base fields should carry names understandable for end users and that all involved fields must be identifyable without an additional context (e.g. “Name” might be a poor naming since it could mean product’s, client’s or employee’s name).
- It is also evident that it is not worth to bring technological data only used in context of the source system from the source to the data mart. E.g. it is senseless to map the numerical identifier of a typical classifier (a table consisting of a numerical identifier and a describing field) to the data mart since the appropriate solution for NoSQL is to add the describing field directly to the described object.
- As mentioned earlier numerical data can cause problems if the denormalization leads to multiple storage of the same information. In cases where we can not avoid the redundant storage of numerical data as result of de-normalization we should ensure that such data can be interpreted as text. On the other hand there sould be ensured that the numerical data stored in the data mart is covering the user requirements fully in the sense that the reporting tool should not be forced to do mathematical operations (e.g. if the source system provides price and amount but not resulting revenue we should store revenue as a third field in the data mart and not leave the respective calculation to the reporting tool).

When defining requirements regarding a universal browser for NoSQL based DW (in the following named UB) it is worth to remind about successful concepts used for classical DW (Chaudhuri et al., 1997). A user setting up a OLAP-based report in EXCEL will operate with two panes – one containing the metadata (in case of OLAP dimensions and facts) and one with the report containing the data. The metadata part allows to select the dimension/fact to be included in the report and also indicates status information (e.g. if a filter is applied).

In our opinion a UB should use an analogue approach – the NoSQL based DW should provide the metadata to the UB and the user should be able to select/deselect the data objects to be included in the report and also should be provided with additional information regarding the usage of the metadata.

The user should be enabled to save the selected metadata and the UB should provide functionality to manage such selections. We should not fail to mention that during the implementation of the prototype we learned that these metadata requirements in a NoSQL based environment at far more challenging than they are for a RDBMS (Sen et al., 2005). Every RDBMS provides some kind of functionality to operate with metadata but NoSQL by nature is much more flexible.

In case of MongoDB in one collection (the analogue to a RDBMS table) every single document (the analogue to a RDBMS record) can be structured differently. But it is advisable to put uniform documents in one collection; otherwise there could problems with metadata occur, especially regarding document descriptions, filtering and aggregation functions.

We would also expect that the UB would support the same basic operations a classical DW based reporting tool does – setting up filters and apply sorting. Of course such features depend on the data types supported by the NoSQL data storage. In case of Mongo DB we would expect filtering possibilities for boolean, date, numerical and string data types. For classifiers type of data (limited pre-defined amount of possible values) we would expect a feature allowing to set up a filter with multiple selection possibility. We should be able to set up sorting orders independent from the applied order regarding data representation. We would expect that we could assign to every numerical field the information whether it is possible to aggregate the data as Count, Min, Max or Avg or it is to be used only as text. We should be able to store the filter, sort and aggregation settings as part of the "views".

All the previously mentioned features will be available in more or less any reporting tool supporting classical DW but there is one feature we would expect for UB that we are not used to meet in the classical DW environment – the Google-like search. We would expect to have a input field allowing to enter any text and such a filter is applied on all texts stored in the NoSQL data mart. We would expect that there are possibilities to refine that search (include/exclude, and/or, */? , intervals) either using some search operators or by enhanced input possibilities. Of course the search criteria should also be stored as part of the "view".

Generally there are several possibilities regarding the data presentation; we investigated three types – tree, JSON, nested tables. We came to the conclusion that a tree representation in some specific cases is quite helpful – especially to explore a single Mongo DB document – but it won't be a presentation management would expect for a DW product. Similar is true for JSON representation – it is helpful for development and when investigating details of single records but also not useable for management purposes.

In result we expect nested tables to be the mandatory representation form for a UB.

Firstname	**Lastname**	**Address**	**DateOfBirth**	**Education years**
John	Smith	Riga, Latvija		6
+ **hobby**				
+ **education**				
Peter	Brown	London, UK	1960/03/15	14
+ **hobby**				
- **education**				

school	**from**	**to**	**years**
Oxford university	2010	2013	3
Some school	1999	2010	11
Sum			**14**

Basically nested table data representation seems to be also a feasible solution to present M:N related data – especially since the nesting can be continued also on lower data hierarchy level. But then again too deep nesting will confuse the user more than it would help him. As mentioned earlier avoiding obsolete data hierarchy levels during data design seems to be the adequate solution.

Since a DW must be able to deal with aggregated numerical values the UB must be able to group the data. Using our favorite nested table data representation we would expect the UB to group the data on level of every nested table representing the aggregations under the according raw data of the column.

Sales country	Sales person	Product	Price	Amount	Total
Latvia	Jānis	Apple	20.00	500	10 000
Latvia	Jānis	Peach	30.00	300	9 000
Latvia	Juris	Peach	30.00	300	9 000
Latvia	Juris	Strawberry	15.00	200	3 000
Latvia	Juris	Plum	12.50	100	1 250
Count			5	5	5
Sum				1400	32250
Avg			21.50	280	6450

We would expect that we are able to include/exclude data fields for every nested table and that we can change the sequence of the data fields in the representation. The selection and the applied aggregations should be stored as part of the "view".

It is worthwhile to outline a few opportunities provided by the use of NoSQL. One record in NoSQL database is a whole document containing one or several uniform rows. For instance, the first table represents the case that the person has attended two educational institutions. In this example there could be reasonable two types of requests:

- Find all persons who have attended the Oxford University and show all educational institutions they have attended (Peter Brown has exactly two);
- Find all persons who have attended the Oxford University and show only those educational institutions which correspond to the searching condition (Peter Brown has attended only one such educational institution).

If there would be an appropriate user interface the data could be analyzed using both types of requests.

Likewise the aggregating function can be used in different ways. For example the total education period of the persons from the table could be calculated and shown as an additional column characterising the document, or as a total amount broken down by different educational institutions.

Thereby, on the one hand, a UB of NoSQL may offer additional filtering and data analysis possibilities, on the other hand, it creates new challenges for UB developers to develop specific user interfaces because until now this type of functionality was not available in the classic DW.

Summarizing we can say that based on the experience of some prototype development we propose a UB to provide functionality both to manage metadata and for data representation. Although it is likely that data in a DW is basically structured the implementation of UB metadata management for most NoSQL data storages seem to be a quite challenging task. Regarding data representation we favor nested data representation allowing to add numerical data aggregations on level of every nested table.

4. Conclusions

Based on experiences gained during development and use of prototypes the authors believe that the future of NoSQL based DW is promising:

- NoSQL based DW have the potential to unify benefits of classical DW technology with the simple-to-use of the Internet times including the Google-style search,
- A strong de-normalization technology unifying the design and production aspects is the key to enable agile development approaches for NSQL based DW,
- From-the-shelf reporting tools for NoSQL data storages are necessary to enable a break-through of NoSQL based DW usage.

Acknowledgment

The research leading to these results has received funding from the research project "IT Competence Centre" of EU Structural funds, contract No. L-KC-11-0003 signed between IT Competence Centre and Investment and Development Agency of Latvia, Research No. 1.8 “Technology for denormalization of relational data bases”.

References

Abadi, D.J., Boncz, P.A., Harizopoulos, S. (2009) Column-oriented database systems, *Proceedings of the VLDB Endowment,* Volume 2 Issue 2, August 2009, pp. 1664-1665, DOI: 10.14778/1687553.1687625

Assey, M. (2015) NoSQL databases eat into the relational database market, *TechRepublic*, http://www.techrepublic.com/article/nosql-databases-eat-into-the-relational-database-market/

Baars, H., Kemper, H.G (2008) Management Support with Structured and Unstructured Data—An Integrated Business Intelligence Framework, *Information Systems Management, Volume 25, Issue 2*, 2008, pp. 132-148, DOI:10.1080/10580530801941058

Bridgwater, A. (2015) How the IT universe moves to software-defined data warehouse life, and everything, *ComputerWeekly.com, CW Developer Network Computer Magazine,* http://www.computerweekly.com/blogs/cwdn/2015/04/how-the-it-universe-moves-to-software-defined-data-warehouses.html

Chaudhuri, S., Dayal, U. (1997) An overview of data warehousing and OLAP technology, *ACM SIGMOD Record, Volume 26 Issue 1*, March 1997, pp. 65-74, DOI: 10.1145/248603.248616

Chen, L., Soliman, K.S., Mao, E., Frollick, M.N. (2010) Measuring user satisfaction with data warehouses: an exploratory study, *Information & Management, Volume 37, Issue 3*, 1 April 2000, pp. 103–110, DOI: 10.1016/S0378-7206(99)00042-7

Devlin, B., Cote, L.D. (1996) Data Warehouse: From Architecture to Implementation, *Addison-Wesley Longman Publishing Co*., ISBN: 0201964252

Fahey, L., Prusak, L. (1998) The Eleven Deadliest Sins of Knowledge Management, *California Management Review, Vol. 40 No. 3*, Spring 1998, pp. 265-276, DOI: 10.2307/41165954

Fehling, C., Leymann, F., Schumm, D., Konrad, R., Mietzner, R., Pauly, M. (2011) Flexible Process-Based Applications in Hybrid Clouds, *2011 IEEE International Conference on Cloud Computing (CLOUD)*, 4-9 July 2011, pp. 81 – 88, ISBN: 978-1-4577-0836-7, DOI: 10.1109/CLOUD.2011.37

Golfarelli, M., Maio, D., Rizzi, S. (1998) The Dimensional Fact Model: a Conceptual Model for Data Warehouses, *International Journal of Cooperative Information Systems, Volume 07*, Issue 02n03, June & September 1998, DOI: 10.1142/S0218843098000118

Han, J., Haihong, E., Le., G., Du, J. (2011) Survey on NoSQL database, *the 6th International Conference on Pervasive Computing and Applications (ICPCA)*, 26-28 Oct. 2011, IEEE, pp. 363-366, ISBN: 978-1-4577-0209-9, DOI: 10.1109/ICPCA.2011.6106531

Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P. (2003) Fundamentals of Data Warehouses, the 2nd edition, *Springer Verlag*, ISBN 3-540-42089-4

Karnitis, G., Arnicans, G. (2015) Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation, In *7th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, 3-5 June 2015, IEEE, pp. 113-118, ISBN: 978-1-4673-7015-8, DOI: 10.1109/CICSyN.2015.30

Kaur, K., Rani, R. (2013) Modeling and querying data in NoSQL databases, *2013 IEEE International Conference on Big Data*, 6-9 Oct. 2013, pp. 1-7, DOI: 10.1109/BigData.2013.6691765

Lane, P. (2002) Oracle9i, Data Warehousing Guide, Release 2 (9.2), Part No. A96520-01

Leavitt, N. (2010) Will NoSQL Databases Live Up to Their Promise?, *Computer, Volume:43 , Issue: 2*, IEEE, pp. 12-14, ISSN : 0018-9162, DOI: 10.1109/MC.2010.58

Levene, M., Loizou, G. (2003) Why is the snowflake schema a good data warehouse design?, *Information Systems, Volume 28, Issue 3*, May 2003, pp. 225–240, Elsevier Science, doi:10.1016/S0306-4379(02)00021-2

Mrdalj, S. (2011) Would cloud computing revolutionize teaching business intelligence courses?, *Issues in Informing Science and Information Technology: Navigating Information Challenges, Volume 8 (Ed. Eli B. Cohen)*, Informing Science Press, 2011, pp. 209-217, ISBN: 978-1-932886-47-4

Padhy, R.P., Patra, M.R., Satapathy S.C. (2011) RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's, *International Journal of Advanced Engineering Sciences and Technologies (IJAEST*), Vol No. 11, Issue No. 11(1), pp. 15 – 30, ISSN: 2230-7818, http://journals.indexcopernicus.com/abstract.php?icid=962583

Pokorny, J. (2013) NoSQL databases: a step to database scalability in web environment, *International Journal of Web Information Systems, Vol. 9 Iss: 1*, pp.69 – 82, DOI: 10.1108/17440081311316398

Rahman, N., Rutz, D., Akhter, S. (2013) Agile Development in Data Warehousing, *Principles and Applications of Business Intelligence Research (Ed. Richart T. Herschel)*, IGI Global, 2013, pp. 286-320, DOI: 10.4018/978-1-4666-2650-8.ch020

Seeger, M. (2009) Key-Value stores: a practical overview, *Computer Science and Media Ultra-Large-Sites SS09*, Stuttgart, http://blog.marc-seeger.de/assets/papers/Ultra_Large_Sites_SS09-Seeger_Key_Value_Stores.pdf

Sen, A., Sinha, A.P. (2005) A comparison of data warehousing methodologies, *Communications of the ACM - The disappearing computer, Volume 48 Issue 3*, March 2005, pp. 79-84, DOI: 10.1145/1047671.1047673

Shin, S.K., Sanders, G.L. (2006) Denormalization strategies for data retrieval from data warehouses, *Decision Support Systems, Volume 42, Issue 1*, Elsevier, October 2006, pp. 267–282, DOI: 10.1016/j.dss.2004.12.004

Todman, C. (2000) Designing a Data Warehouse: Supporting Customer Relationship Management, *Prentice Hall PTR Upper Saddle River*, NJ, USA, 2000, ISBN: 0130897124

Vicknair, C., Macias, M., Zhao, Z., Nan., X., Chen, Y., Wilkins D. (2010) A comparison of a graph database and a relational database: a data provenance perspective, *ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference,* Article No. 42 ACM New York, NY, USA ISBN: 978-1-4503-0064-3, DOI: 10.1145/1900008.1900067

Woodey, A. (2014) Forrester Ranks the NoSQL Database Vendors, *Datanami*, http://www.datanami.com/2014/10/03/forrester-ranks-nosql-database-vendors/

Received March 10, 2016, accepted June 13, 2016