Implementation of NoSQL-based Data Warehouses

Ivo ODITIS¹, Zane BICEVSKA¹, Janis BICEVSKIS², Girts KARNITIS²

¹ SIA DIVI Grupa, Riga, Lativa ² University of Latvia, Riga, Latvia

ivo.oditis@di.lv, zane.bicevska@di.lv,
janis.bicevskis@lu.lv, girts.karnitis@lu.lv

Abstract. Data warehouses traditionally are built using relational databases. The main reason for that - data warehouses mostly are used in enterprises with large-scale data sets stored in relational data structures of different legacy systems. Though rapidly developing non-relational databases also are still rather unusual in data processing tasks. This paper discusses the possibilities and advantages of using document-oriented NoSQL database management systems in data warehousing. The paper describes processes and implementation aspects of building data warehouses using a NoSQL data storages. The research is based on practical experience gained during implementing of NoSQL data marts in MongoDB.

Keywords: Data warehouses, NoSQL data, non-relational databases, data denormalization.

1. Introduction

The concept of data warehouse generally is platform-independent, and it does not imply the usage of specific technological means (Bridgwater, 2015). Nevertheless the prevalent understanding of data warehouse (DW) presumes the usage of relational databases. For instance the "Data Warehousing Guide" by Oracle Inc. states: "A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing." Typically it is assumed that DWs are relational data storages and they are processed using means of relational database management systems (RDBMS).

The concept of non-relational (NoSQL) databases refers to a 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 (Woodey, 2014): document databases, key-value stores, graph databases, and column-oriented stores. NoSQL databases are commonly associated with more flexible deployment, high read/write performance as well as scaling to very large data sets.

There is a trend towards increasingly growing market share for NoSQL databases. Market surveys have shown how the NoSQL increases their market share, mostly at the expense of MySQL. Although the maturity of the NoSQL market is assessed to be rather low yet, the Forrester Wave estimated (Han et al., 2003) 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.

Lately efforts of producers of DBMS systems to satisfy the developer community and market demands have resulted in the recent real expansion of functionalities that have brought the two separated worlds considerably close together (Gaspar and Coric, 2017). There can be confidently said that both areas are merging and complementing each other.

The first chapter describes classical DWs and identifies benefits to be expected if using NoSQL technologies instead of RDBMS. The second chapter describes how to create DW using NoSQL data marts and highlights the necessity to use cross-platform denormalization technology. The third chapter deals with reporting issues in NoSQL-based DWs. The paper proposes requirements for reporting support in DW, and it is based on the behavior of classical DWs adding NoSQL-specific features.

2. RDBMS-based vs NoSQL-based DWs

Typically there are several IT systems in an enterprise, every of them covering a different functional segment (e.g. Accounting, HR, CRM). The need of such enterprise's management for more integrated information often leads to the decision to build a DW. As the most of applications use RDBMS data storage the decision to base the DW on a RDBMS data mart seems obvious. Such a decision is compelling if the DW 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).

Roughly speaking the existing RDBMS-based (or classical) DW (Jarke et al., 2003) technology divides all collected data in two groups:

- There is a data that can be measured (e.g. temperatures, costs, speed). 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.
- Only the describing data in DW terminology called dimensions gives meaning to facts. Desirably all the describing data would be related directly as foreign keys to the facts, called star scheme, but sometimes the available data requires more sophisticated DW, called snowflake scheme. There are also other description schemas possible, for instance, the multi-dimensional Cognos Dynamic Cubes of IBM (Beryoza et al., 2015).

The benefit of the classical DW technology in comparison to any self-made data integration is the built-in functionality to combine facts with dimensions. Therefore DW users might not even need to have deep business knowledge do get benefits from it.

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 workaround for specific situations but such solutions are not flexible by definition (Baars and Kemper, 2008). In turn the NoSQL technology claims to have the ability to handle high volumes of structured, semi-structured, and unstructured data.

Though, the strong architecture of dimensions and facts dictates some specific limitations for DW usage, too. For instance, once a DW is created, it is hard to change its structure, so the exact facts and dimensions should be defined prior to the creating of a

DW. Likewise, an initial meaning of data could be practically lost when focusing only on aggregation of facts.

Today's users expect DW solutions to act more in the internet-style than to enforce the user to act within predefined structures. Users may expect the DW is able to pro-vide answers even if only search texts are entered. As NoSQL technology by nature provides a good support regarding semi-structured and unstructured data (Kaur and Rani, 2013) the possibilities should be investigated to combine the text search capabilities of NoSQL with possibilities to receive similar results as provided from a classical DW. But there should be taken into account, that NoSQL data storages are not created for numerical operations on big data sets. 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).

Thereby it can be concluded NoSQL data storages are able to support the following:

- Data aggregations in documents from several information systems; for example a document would contain full order details, including data about all delivered goods and customer's information;
- It is possible to create documents including the entire data set from external information systems; even if it is not initially clear which data will later be used in data analysis (dimensions and facts are not exactly defined yet);
- It is possible to track the context of data origins if necessary from the document containing resulting (numeric) values.

There should be noted that the document-oriented approach makes the DW more transparent and easier-to-build. For instance, if the DW is built for management of orders' data, the most natural data object in such DW would be an order (document XML or JSON format) containing full information about deals (orders, sales, etc.), customers (name, delivery address, etc.), ordered/sold goods (with their parameters like color, size, etc.), quantity, price, taxes. Furthermore, creating such a structure the DW designer would not have to consider a new dimension in DW for VAT as the respective field would already be a part of the order's document and therefore accessible for all data operations – filtering, grouping, aggregation, etc. The NoSQL approach simplifies the creating of DW as clear data objects or documents are used in a high conceptual level, in contradiction to the classical DW which requires transformations between different relational data models to optimize and to use the data.

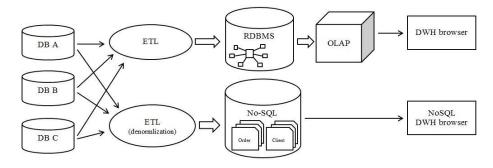


Fig. 1. Processing of RDBMS-based DW and NoSQL-based DW

The figure above illustrates how both types of DW are processed. In the case of NoSQL based DW not only the process is much simpler but also the created DW structure and content are more transparent.

Of course the usage of NoSQL-based approach in building of DW significantly increases the data volume to be stored and leads to data duplication. So obviously, such a solution would not be applicable to big volumes of data. In contrast, small and medium-sized enterprises, especially regarding the nowadays available data processing capacity, may have positive results.

3. Data extraction for NoSQL-based DW

Another often mentioned benefit of NoSQL is the high flexibility in the development process (Fehling et al., 2011).

Building classical DW a lot of work must be invested in defining of dimensions and their hierarchy; adding of new dimensions is also quite labor intensive. During the creating of a NoSQL DW the action takes place at the level of business objects, the level of abstraction is much higher, and hence the building of the DW is faster.

NoSQL DW is created in several stages: physical models of used information systems -> conceptual models of used information systems (i.e. master data model)-> a common conceptual model -> aggregates or data objects (of different types) -> documents (each aggregate is a document of one type) -> data transformation process.

Denormalization allows linking of objects from several information systems in the conceptual (business objects) level. It offers a unified view to all data of an enterprise. But it also raises problems: how to identify the same objects in different information systems unambiguously and how to refresh data? Usually data objects are identified (with unique IDs) in all information systems used within an enterprise; if not, the classical data mining methods must be used – classification, feature vectors, etc. (AnHai et al., 2012).

Data refreshing can raise several problems:

- 1. each row from the relational database may be included in very many documents of the NoSQL data base; hence the potential changes could affect very large part of the data;
- 2. changes in multiple rows (in one or many tables) may lead to repeated changes in the same document several times;
- 3. the denormalized documents should contain unique identifiers (primary keys) of all initial data records, otherwise problems with identification of changeable documents may arise.

It leads to the conclusion that technically much easier is to delete all documents and to denormalize the data from the scratch again, or to apply a little optimization (it can improve performance but not necessarily does it) and to replace all changed documents with new ones.

The NoSQL technology has the perspective to allow agile development approaches if there are methodology and tools supporting the denormalization and synchronization for the respective NoSQL-based data mart.

It's obvious that it makes no sense to rebuild the relational data structures when creating the NoSQL based data mart. A process of denormalization will be necessary, and the denormalization should be as easy to handle as possible (Shin and Sanders, 2006).

Denormalization of relational data and transforming of it to document-oriented data base is a topical and for the present insufficiently solved problem. The solutions are either trivial (table-> collection of documents) or individually programmed processing routine for each specific. The authors have done practical investigations in this direction using a prototype for a cross-platform denormalization tool. The tool's prototype is able to analyze data structures and relations of several standard RDBMS (MS, Oracle, MySQL, etc.) by analyzing the according meta-data of the RDBMS. The prototype allows the user to set up denormalized data structures without programming simply by browsing data trees or drilling-down the according data. There was also a possibility to export the selected data as XML or JSON documents (Karnitis and Arnicans, 2015).

The lessons learned can be summarized as follows:

- It is a quite trivial task to create and export denormalized data structures when the data is related 1:N. Good working algorithm is spreading over all links from the root data until the denormalized document contains data from all tables. If any table is accessible via several ways, the shortest will be automatically used to gather the necessary data.
- Self-referencing data cannot mechanically be denormalized since it would
 produce endless referenced document chains. It is recommended either to
 flat the source data by preparing an according view or to denormalize the
 references only in one direction (e.g., in a hierarchy store only the respective
 superior or only the inferiors).
- As less hierarchical levels during denormalization are created as more convenient the data will be for reporting.
- Aggregation of numerical data is the biggest challenge in the data mart design process. A considerable intellectual effort has to be spent to find the best structures as such data cannot be stored redundant in the data mart.

Since the denormalization process 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 of structures in the NoSQL data mart.

Any DW requires a solution to synchronize the data in the data mart with the data available in the data sources. The DW implementation has to find a compromise between user requirements regarding data actuality and the technical limitations of involved technology. All considerations relevant for implementing of DW synchronization in a classical DW environment will be relevant also for NoSQL-based DW:

- DW designers will have to decide on the necessary frequency of the DW update;
- they will investigate the possibilities to do data updates limited for time periods versus a complete reload of all data;
- in case of partial data refreshing, the changed data must be identified.

In particular the last two considerations may significantly influence the refreshing of data in a NoSQL DW:

- it should be identified which NoSQL documents are affected by data changes;
- the affected documents or their parts should be replaced by updated ones with denormalized data.

At present, the existing tools do not offer functionality of data synchronization for NoSQL-based DWs. The existing data denormalization tools just transform the result of each data request to a simple NoSQL document. When building NoSQL DW, also

complex documents should be built, the content of which exceeds a content of one flat, relational data table; for instance a document about customers may contain all kind of relevant information – name, several addresses, signatories, account numbers, etc.

4. 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 spectrum 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 is always located in top positions (Han et al., 2011). The authors' initial experience when searching for reporting tools fully matched such opinions. Although different tools are offered for NoSQL DW browsing, they lack the data analysis functionality (grouping and aggregation).

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 for hiding
 the technical naming of data objects from 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
 identifiable without an additional context.
- 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 cannot avoid the redundant storage of numerical data as result of denormalization we should ensure that such data can be interpreted as text. On the other hand there should 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 (hereafter – UB) it is worth to remind about successful concepts used for classical DW (Chaudhuri and Dayal, 1997). A user setting up a OLAP-based report in EXCEL will operate with two panes – one containing the meta-data (in case of OLAP dimensions and facts) and one with the report containing the data. The meta-data 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).

A UB should use an analogue approach: the NoSQL based DW provides the metadata to the UB and the user is able to select/deselect the data objects to be included in the report and is also provided with additional information how to use the meta-data.

The user should be enabled to save the selected meta-data and the UB should provide functionality to manage such selections. During the implementation of the prototype we learned that these meta-data requirements in a NoSQL based environment are far more challenging than they are for RDBMS (Sen and Sinha, 2005). Every RDBMS provides some kind of functionality to operate with meta-data but NoSQL by nature is much more flexible. Even in one collection of NoSQL data mart every single document (the analogue to a RDBMS record) can be structured differently, so it is advisable to put uniform documents in one collection. Otherwise, problems with meta-data may 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. We should be able to set up sorting orders independent from the applied order regarding data representation. 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, should be assigned to every numerical field. The filter, sort and aggregation settings should be stored as a 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 an input field allowing to enter any text and such a filter is applied on all texts stored in the NoSQL data mart.

Generally there are several possibilities regarding the data presentation; we investigated three types – tree, JSON, nested tables. The tree representation is quite helpful in some specific cases– especially to explore a single NoSQL document –, but it is not transparent enough to serve as the presentation layer for a DW product. The same is true for JSON representation – it is helpful for development and when investigating details of single records but is also not useable for management purposes. In result we expect nested tables to be the mandatory representation form for a UB (Table 1.).

Table 1.

Firstname	Lastname	Address	Address		Birth	Education years
John	Smith	Riga, LV	Riga, LV			12
+ hobby						
+ education						
Peter	Brown	London, U	London, UK		60	14
+hobby						
- education						
School		From	To		Years	
Oxford		2010 201		3 3		
University						
Some school		1999 20		0 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 too deep nesting will confuse the user more than it would help him

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.

It is worthwhile to outline some opportunities provided by the use of NoSQL. One record in a NoSQL database is a whole document that can contain one or several uniform sets of data. E.g., Table 1 represents the case that a person has attended two educational institutions. In this case two types of requests could be reasonable:

- 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).

Various interpretations can also be applied regarding aggregation. E.g., Table 1 contains the total education period of the persons which can be calculated and represented both as an additional column referencing the whole underlying document or referencing the individual 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.

Source code 1. Meta-data example

```
"columns": {
    "id": { "type": "integer" },
   "number": { "type": "string" },
   "supplier": { "type": "string"},
   "total": {
      "type": "real", "aggr": {"sum", "avg", "min", "max"}
   "warehouse": { "type": "string" },
   "state": { "type": "string" },
   "items": {
      "type": "array",
      "subtype": "document",
      "childs": {
         " id": { "type": "integer" },
         "part_code": { "type": "string" },
         "quantity": {
            "type": "integer", "aggr": {"sum", "avg", "min", "max"}
         "price": { "type": "decimal" },
         "row total": { "type": "decimal" },
         "state": { "type": "string" }
   }
```

5. NoSQL meta-data implementation

To ensure an effective data analysis, it is necessary to create a description of the documents stored in the NoSQL database (a meta-model). The model should describe the data structure, the explanatory data of the data values, as well as the information about the possibilities of data analysis (whose values are summable, usable in the calculation of average values, etc.)

This meta-model (source code 1) could also be used to transfer data from relational to non-relation data base. Referring to this model it is possible to describe how the fields are selected and replaced in the resulting document.

The authors propose to build a document's meta-model as a JSON document which structure corresponds to the data warehouse document extended with the meta-information: the data type of the vertex, and the available operations.

Source code 2. Meta-data example with data source information

```
{
   "ExtractInfo": {
      "Request": "select pwID id, ...., whName as [warehouse]
      from waybills order by pwID",
   "columns": {
     "_id": { "type": "integer" },
      "warehouse": { "type": "string" },
      "items": {
         "type": "array",
         "subtype": "document",
         "ExtractInfo": {
            "Request": "select pwpID as [_id], prtCode as
            [part_code], ... from partWaybillParts_vwFull where
           pwID = @parentId order by pwpID",
         "childs": {
            " id": { "type": "integer" },
            "part code": { "type": "string" },
     }
```

The type in the example denotes types of the values, and the aggr denotes the possible aggregations of types. The element types array and document are distinct. The first one allows you to define an element that contains a list of other elements, the document describes an element that is a structure.

By using this definition of documents, it is possible to obtain the data inquires for data loading during the ETL process (source code 2). In the simplest case, assuming that single-level elements of a document can be filled from a single data source, the data loading statement should be added to the defining level of the document.

Thus, if using such a meta-model, it is possible to provide both the information necessary for the analysis and the description of data transfer. Since the same description is used in both cases, it also ensures the integrity of the data warehouse.

This model further illustrates the simplicity of implementing the proposed data warehouse solution. The information necessary for the transfer and analysis of a particular data type is collected in one simple structured JSON document. If the data warehouse must be structurally changed or transaction data is changed, the structure of the data warehouse description should be changed, and the data loading may be repeated according to the new description.

6. Conclusions

Usage of NoSQL to build document-based DWs is only possible thanks to the modern technologies offering more and more power and available dataspace. Even a few years ago such solutions would not be financially affordable, today they are available for middle-sized and small enterprises.

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

- DW may be easier to build when using NoSQL data storage because fewer components are necessary and the resulting data structures are less complex;
- 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;
- NoSQL based DW have the potential to provide new features for data analyses impossible by classical DW systems, and the initial data context may be accessible for analysis purposes;
- Benefits when using NoSQL technologies may be expected if a substantial
 part of the data collected in the data mart is semi-structured or nonstructured and if we do not face too specific requirements regarding the
 facts necessary in the DW.

Acknowledgments

The research leading to these results has received funding from the research project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/16/A/007 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 1.10 "Non-relational data warehouse development technology".

References

AnHai, D., Halevy, A., Ives, Z. (2012). Principles of data integration. Elsevier.

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

- Beryoza, D., Campbell, M.A., Cardorelle, C., Creasey, T., Cushing, D., Da Silva V., David, S., Hagleitner, A., Henderson, I., Howell, D., Kozine, I., Prieto, P., Thompson, P., Vazquez, J., Zhang, Y. (2015). IBM Cognos Dynamic Cubes, IBM Redbooks, ISBN: 9780738440835
- 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, Vol. 26 Issue 1, March 1997, pp. 65-74, DOI: 10.1145/248603.248616
- Chen, L., Soliman, K.S., Mao, E., Frollick, M.N. (2000). Measuring user satisfaction with data warehouses: an exploratory study, Information & Management, Vol. 37, Issue 3, April 1, 2000, pp. 103–110, DOI: 10.1016/S0378-7206(99)00042-7
- 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), July 4-9,2011, pp. 81 88, ISBN: 978-1-4577-0836-7, DOI: 10.1109/CLOUD.2011.37
- Gaspar, D., Coric, I. (2017). Bridging Relational and NoSQL Databases, IGI Global, ISBN: 9781522533863, p. 270
- Han, J., Haihong, E., Le., G., Du, J. (2011). Survey on NoSQL database, the 6th International Conference on Pervasive Computing and Applications (ICPCA), Oct. 26-28, 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, 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. Computational Intelligence, Communication Systems and Networks (CICSyN), 2015 7th International Conference on. IEEE, 2015
- Kaur, K., Rani, R. (2013). Modeling and querying data in NoSQL databases, 2013 IEEE International Conference on Big Data, Oct. 6-9, 2013, pp. 1-7, DOI: 10.1109/BigData.2013.6691765
- Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise?, Computer, Vol. 43, Issue 2, IEEE, pp. 12-14, ISSN: 0018-9162, DOI: 10.1109/MC.2010.58
- Sen, A., Sinha, A.P. (2005). A comparison of data warehousing methodologies, Communications of the ACM - The disappearing computer, Vol. 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, Vol. 42, Issue 1, Elsevier, October 2006, pp. 267– 282, DOI: 10.1016/j.dss.2004.12.004
- Woodey, A. (2014). Forrester Ranks the NoSQL Database Vendors, Datanami, http://www.datanami.com/2014/10/03/forrester-ranks-nosql-database-vendors/

Received February 1, 2018, revised March 22, 2018, accepted March 27, 2018