Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 104 (2017) 104 – 111

Procedia Computer Science

ICTE 2016, December 2016, Riga, Latvia

Towards NoSQL-based Data Warehouse Solutions

Zane Bicevska[a], Ivo Oditis[a,*]

[a]*DIVI Grupa Ltd, Riga, Latvia*

Abstract

Data warehousing is a traditional domain of relational databases, and there are two main reasons for that: (1) data warehouses mostly are used in enterprises with large-scale data sets created in different legacy systems with relational data storages, (2) though rapidly developing non-relational databases are still rather unusual in data processing tasks. This paper discusses the possibilities to create data warehouse solutions by using NoSQL database management systems. The main challenge is to find a good balance between characteristics of classical data warehouses using relational data base management systems and opportunities offered by NoSQL database management systems. The paper describes processes of creation and production of data warehouse using a NoSQL data mart and outlines requirements for technology necessary for such processes. The research is based on practical experience when implementing NoSQL data marts with MongoDB and Clusterpoint DB.

Peer-review under responsibility of organizing committee of the scientific committee of the international conference; ICTE 2016

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[1]. 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).

* Corresponding author.
E-mail address: ivo.oditis@di.lv

Peer-review under responsibility of organizing committee of the scientific committee of the international conference; ICTE 2016
doi:10.1016/j.procs.2017.01.080

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. There can be distinguished four different kinds of NoSQL databases[2]: document databases, key-value stores, graph databases, and column-oriented stores. The document-based databases provide the storage of documents made up of tagged elements and they seem to be the most popular NoSQL databases in practice (examples: Couchbase, MongoDB; also in Latvia developed Clusterpoint). The key-value stores use big hash tables of keys and values for fast accessing of data (examples: Riak, Amazon's Dynamo). The graph-based databases use edges and nodes to represent and store data (examples: InfoGrid, Infinite Graph, Neo4J). The column-based stores converts data storage blocks to columns (examples: Google's BigTable, HBase, Cassandra). 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 researches 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[3] 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 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 paper also outlines the possibility of using agile development approach during the process of DW creation. The third chapter deals with reporting issues in NoSQL-based DWs. The paper proposes requirements for accord reporting support. These requirements are based on the behavior of classical DWs adding NoSQL-specific features.

2. Classical 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[3].

Roughly speaking the existing classical DW[4] 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

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 work-around for specific situations but such solutions are not flexible by definition[5]. In turn the NoSQL technology claims to have the ability to handle high volumes of structured, semi-structured, and unstructured data.

Today's users expect DW solutions to act more in the internet-style than to enforce the user to act within pre-defined structures. Users may expect the DW is able to provide answers even if only search texts are entered. As

NoSQL technology by nature provides a good support regarding semi-structured and unstructured data[6] 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[7].

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.

3. Implementation aspects of NoSQL-based DW

Another often mentioned benefit of NoSQL is the high flexibility in the development process[8]. 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 the need for a lot of specific infrastructure to design, implement and set up a classical DW[9] - data mart, ETL, OLAP, reporting tools etc. All these components also require several IT specialists to work coordinated. Usually the major part of the DW implementation project time and money has been spent before end-users get even a trivial report generated from a classical DW.

The second reason is the specific knowledge necessary to set up a classical DW[10]. Especially the creation of DW dimensions 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. Although there is a huge amount of literature covering the several design aspects[11,12], 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[13]. Agile style development of classical DW on this background is nearly impossible[14].

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

The necessary environment for a NoSQL-based DW requires a methodology and a technological component to transport the 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. A process of denormalization will be necessary, and the denormalization should be as easy to handle as possible15.

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.

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
- 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 a 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)
- Pure mechanical approaches end when we face M:N-related data since they carry the risk of producing gigantic amounts of redundant data during de-normalization. The decision for the best de-normalization 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
- Agreeable 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. It is not

difficult to find any denormalization solution when numerical data simply appears in different hierarchy levels but we face real challenges when there also appear M:N-relations

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.

Of course not all data from a RDBMS source can be transported 1:1 to a DW. But the same problem may rise in case of a classical DW, and then 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.

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. The 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.

Nevertheless there are some aspects creating additional challenges when using NoSQL technology instead of classical DW technology. The DW design can be influenced by expected frequency of dimension data changes. Since NoSQL-based data marts will store the describing data (dimension analogue) de-normalized the according attributes may appear in nearly every document stored in a NoSQL-based data mart. In situations with regular dimension data changes the data updating strategy might become quite challenging, especially if a full reload approach cannot be used. For similar reasons we might face challenges in situations where aggregated data cannot be calculated by the reporting tools but has to be stored in the data mart. If these aggregations change in time we face the same situation as described for the changing dimensions.

4. Reporting for No-SQL-based DW

One of the big strength of classical DW technology is the flexibility regarding the reporting[16]. 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[3]. The authors' initial experience when searching for reporting tools fully matched such opinions. As recourse some effort from authors' side was spent to build a simple universal browsing tool.

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 de-normalization leads to multiple storage of the same information. In cases where we cannot 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 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 (in the following named UB) it is worth to remind about successful concepts used for classical DW[17]. 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[18]. 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 No-SQL 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 (see Table 1).

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 (see Table 2).

Table 1. No-SQL documents with person data (hobbies and education).

Firstname	Lastname	Address	DateOfBirth	Education years
John + hobby + education	Smith	Riga, LV		12
Peter +hobby - education	Brown	London, UK	15.03.2960	14

School	From	To	Years
Oxford University	2010	2013	3
Some school	1999	2010	11
sum			14

Table 2. Defining characteristics of five early digital computers.

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 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 I 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.

Some more complex aggregation requests are emerging in the example of Table 3 containing the data of two shop customers. Stored in a NoSQL based DW such data would allow many interesting requests for data analysis:

- All customers who have ever ordered bikes
- All customers who have ever ordered bikes and the average total amount of their orders
- Count of orders where bikes are included and grouped by bike types (this would result in a rather complex data base request because one order may contain several type of bikes)
- All paid orders with total amount grouped by country
- Others

Table 3. Customer data sample.

Customer	ID	Address
John Smith - orders	10034	UK

Order date	Order nr	Status
03.04.2015 + status history - items	AC-2301	Delivered

Nr.	Name	Count	Price	Sum
1	Bike (street)	1	220	220.00
2	Tire	2	20	40.00
3	Helmet	1	63	63.00

06.07.2015 + status history + items	AC-4506	Deliver
05.05.2016 + status history + items	AC-0607	Paid

Likewise in previous example the aggregations can be done on different data levels:

- Total amount spent by customer
- Total amount of each order
- Total amount of each order where bikes were ordered
- Total amount of money spent for bikes per order

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 add numerical data aggregations on level of every nested table.

5. Conclusion

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
- NoSQL based DW have the potential to provide new features for data analyses impossible by classical DW systems
- 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

Acknowledgements

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

1. Bridgwater A. How the IT universe moves to software-defined data warehouse life, and everything. ComputerWeekly.com. CW Developer Network Computer Magazine; 2015. Available: http://www.computerweekly.com/blogs/cwdn/2015/04/how-the-it-universe-moves-to-software-defined-data-warehouses.html.
2. Woodey A. Forrester Ranks the NoSQL Database Vendors. Datanami; 2014. Available: http://www.datanami.com/2014/10/03/forrester-ranks-nosql-database-vendors/.
3. Han J, Haihong E, Le G, Du J. Survey on NoSQL database. 6th International Conference on Pervasive Computing and Applications (ICPCA). IEEE; 2011. p. 363-366.
4. Jarke M, Lenzerini M, Vassiliou Y, Vassiliadis P. Fundamentals of Data Warehouses. 2nd edition. Springer Verlag; 2003.
5. Baars H, Kemper HG. Management Support with Structured and Unstructured Data. An Integrated Business Intelligence Framework. *Information Systems Management*. Vol. 25 (2); 2008. p. 132-148.
6. Kaur K, Rani R. Modeling and querying data in NoSQL databases. IEEE International Conference on Big Data; 2013. p. 1-7.
7. Leavitt N. Will NoSQL Databases Live Up to Their Promise? *Computer*. Vol. 43 (2). IEEE; 2008. p. 12-14.
8. Fehling C, Leymann F, Schumm D, Konrad R, Mietzner R, Pauly M. Flexible Process-Based Applications in Hybrid Clouds. IEEE International Conference on Cloud Computing (CLOUD); 2011. p. 81 – 88.
9. Mrdalj S. Would cloud computing revolutionize teaching business intelligence courses? Issues in Informing Science and Information Technology: Navigating Information Challenges. Vol. 8. (ed. Cohen EB). Informing Science Press; 2011. p. 209-217.
10. Fahey L, Prusak L. The Eleven Deadliest Sins of Knowledge Management, California Management Review. Vol. 40 (3); 1998. p. 265-276.
11. Devlin B, Cote LD. Data Warehouse: From Architecture to Implementation. Addison-Wesley Longman Publishing Co.; 1996. Todman C. Designing a Data Warehouse: Supporting Customer Relationship Management. Prentice Hall PTR Upper Saddle River; 2000.
12. Golfarelli M, Maio D, Rizzi S. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *International Journal of Cooperative Information Systems*. Vol. 7 (2-3); 1998.
13. Rahman N, Rutz D, Akhter S. Agile Development in Data Warehousing. Principles and Applications of Business Intelligence Research. (ed. Herschel RT). IGI Global; 2013. p. 286-320.
14. Shin SK, Sanders GL. Denormalization strategies for data retrieval from data warehouses. Decision Support Systems. Vol. 42 (1). Elsevier; 2006. p. 267–282.
15. Chen L, Soliman KS, Mao E, Frollick MN. Measuring user satisfaction with data warehouses: an exploratory study. Information & Management. Vol. 37 (3); 2000. p. 103–110.
16. Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM SIGMOD Record. Vol. 26 (1); 1997. p. 65-74.
17. Sen A, Sinha AP. A comparison of data warehousing methodologies. Communications of the ACM. Vol. 48 (3); 2005. p. 79-84.

Zane Bicevska, born in 1971, earned her Doctor degree in Computer Science from the University of Latvia in 2010 and Master degree in Economics from the Johann Wolfgang Goethe-Universität Frankfurt am Main in 2001. She has published more than 15 scientific papers and has worked as a project manager in various software development projects. Since 2010 Zane Bicevska is Assistant Professor in the Computing Faculty of the University of Latvia. Her main scientific interests include project management, business process modelling, smart technologies. Contact her at zane.bicevska@lu.lv.

Ivo Oditis was born in 1974, earned his Doctor degree in Computer Science from the University of Latvia in 2016. He has worked as a project manager in various software development projects. His main scientific interests include NoSQL-databases, business process modelling, smart technologies. Contact him at ivo.oditis@di.lv.