Using the column oriented NoSQL model for implementing big data warehouses

Khaled. Dehdouh[1], **Fadila. Bentayeb**[1], **Omar. Boussaid**[1] , **and Nadia Kabachi**[1]
[1]ERIC Laboratory/ University of Lyon 2, Bron, France

Abstract - *The column-oriented NoSQL (Not Only SQL) model provides for big data the most suitable model to the data warehouse and the structure of multidimensional data as the OLAP cube and allows it to be deployed in the cloud and a high scalability whilst delivering high performance. In the absence of a clear approach which allows the implementation of data warehouses using this model, we propose in this paper, three approaches which allow big data warehouses to be implemented under the column oriented NoSQL DBMS. Each one differs in terms of structure and the attribute types used when mapping the conceptual model into logical model is performed. We use these approaches to instantiate the conceptual model of the star schema benchmark (SSB) data warehouse into columnar logical models, and show the differences between them when decisional queries are performed.*

Keywords: Big data warehouses, columnar NoSQL model, logical modeling.

1 Introduction

A data warehouse is a database for online analytical processing (OLAP) to aid decision-making. It is designed according to a dimensional modelling which has for objective to observe facts through measures, also called indicators, according to the dimensions that represent the analysis axes [1]. Thus, at the conceptual level, the multidimensional modelling gave birth to the concepts of fact and dimension. The most popular models used to design data warehouses are the star, snowflake, and constellation schemas [2]. These models are then converted to the logical models which depend on the storage mode that will be adapted at the physical level [3]. Classically, the mapping from the conceptual to the logical model is made according to three approaches; ROLAP (Relational-OLAP), MOLAP (Multidimensional-OLAP) and HOLAP (Hybrid-OLAP) [4].

However, with the advent of the big data, the logical modelling adopted by these approaches does not adapt itself to an environment characterized by such amount of data. To solve a part of this issue, other models have appeared such as the column oriented NoSQL. This latter gives a data structure more adequate to the massive data warehouses. Yet, the data warehouse implementation process requires to take into account the recent data structures and should adapt itself to the new technological constraints.

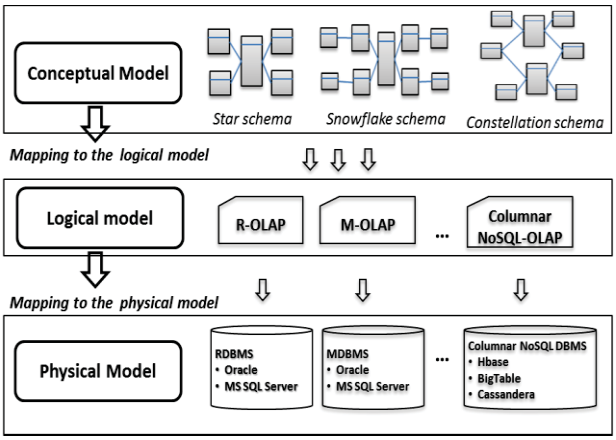

Figure 1: Implementation process for data warehouses

As depicted in the figure 1, the logical model aims at reorganizing the data according to the most appropriate storage architecture for a better taking in charge by the DBMS. It is situated between the conceptual and the physical models of data. In other words , it not only gives more details than the conceptual model on the structuring of data and their relations, but it prepares the transition to the physical level as well; this makes it the most decisive model in the modeling process.

In order to fully benefit from the columnar NoSQL model advantages, and in the absence of a clear approach allowing for implementing the columnar NoSQL data warehouses, we propose in this paper, three types of the conceptual model translations at logical columnar model level namely NLA (Normalized Logical Approach), DLA (Denormalized Logical Approach), and DLA-CF (Denormalized Logical Approach by using Column Family). Each approach leads to a different logical model. We describe each one, and show how we can use these models for implementing data warehouses.

To compare between the three translations, we have implemented the SSB (star schema benchmark) data warehouse [5] according to our propositions. This implementation was achieved under HBase which is column-oriented NoSQL DBMS. We have noticed that the execution of decisional queries using the SSB data warehouse with the

denormalized approaches (DLA and DLA-CF) takes three times less compared with an implementation with SSB using the normalized approach (NLA). Besides, the use of the family column structure for gathering the attributes belonging to the same dimension leads to the improvement until 10% of the queries' execution time which involve several attributes of the same dimension to perform aggregations.

The rest of this paper is organized as follows. Section two gives the related works of column-oriented data warehouse implementation. Section three introduces the column-oriented NoSQL model. Section four presents the three approaches that we propose for data warehouse implementation under the columnar NoSQL DBMS. Section five describes the conversion rules from a dimensional model towards the logical models according to the three approaches NLA, DLA and DLA-CF. In section six, we conducted experiments to evaluate the star schema benchmark implementation according to our three approaches that we propose. Finally, in section 7, we conclude this paper and give some perspectives.

2 Related work

Although the columnar NoSQL model is widely used for storing and analyzing massive data, it does not have, to our knowledge, any methods or defined rules which allow us to know whether a column-oriented NoSQL data warehouse is well performing or not.

However, some works are aimed at developing data warehouses in columnar NoSQL DBMS. In [6] [7], the author has proposed an approach for transforming a relational database into a column oriented NoSQL database using HBase. However, this approach is limited to the logical level, and does not consider the conceptual model of data warehouses; i.e.: mapping a relational logical model into a column oriented logical model. In recent work [8], we have developed a new benchmark for the columnar NoSQL data warehouse, but without giving the formalization for the modeling process. However, this work is considered as the first work which proposes implemented star data warehouse under column oriented NoSQL DBMS directly from dimensional model.

Another recent work, based on our benchmark, has tried to define a logical model for NoSQL data stores (oriented columns and oriented documents) [9]. However, its column oriented logical modeling has been only limited to the use the columns family concept without considering the attributes which are not necessarily belonging to a column family. Indeed, the columns oriented NoSQL model proposes two kinds of attributes: a simple attribute and a composite attribute (nested attribute). This latter is represented by the concept of column family (see section 3). Thus, columnar NoSQL DBMS rather favors the denormalization of the dimensional model, without necessarily using the column families (composite attribute).

To complete these works by taking the simple attributes case into a count, we propose three candidate approaches which summarize the mapping of the multidimensional conceptual data model into a logical modeling adapted to the column-oriented NoSQL data warehouses.

3 Column oriented NoSQL model

In this section, we present the columnar NoSQL model which is characterized by non-relational logical representation of data, and storing the data of a table column-by-column [10]. It allows data warehouse architecture to be deployed in the cloud and a high scalability whilst delivering high performance [11].

Indeed, the non-relational aspect of this model allows the massive data warehouses to be deployed in a distributed environment when scaling up [12], and the column oriented aspect for storing data is beneficial to the data warehouse when aggregation is performed with value belonging to the same column [13]. However, columnar NoSQL model does not have a mechanism which takes in charge the links between tables; thus, it is assigned to the customer applications level [14].

Consequently, the columnar NoSQL DBMS as HBase favors gathering columns in a single table when storing data. Each column stores data in the form of a "key/value" pair which can be stored in a distributed file system. The combination <row key, column name, timestamp> represents the coordinates of the value as depicted in the figure 2.

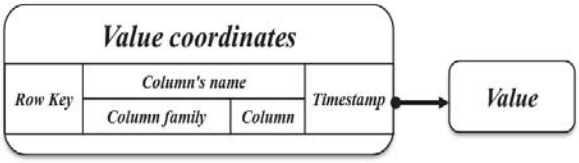

Figure 2: Data structure of the columnar NoSQL model

The row key serves for identifying the column values belonging to the same tuple. The column name allows identifying the attribute of a value; it can be composed by column family name and column name. Indeed, the column may be composite or simple. If the column name is prefixed, this means that the column name consists of the name of the column family (prefix) and the name of the nested column. In this case, it is called composite attribute (belong to a column family), otherwise it is considered as simple attribute. The figure 3 represents the corresponding UML class diagram of a column-oriented NoSQL data model (tables, rows, column families and columns).

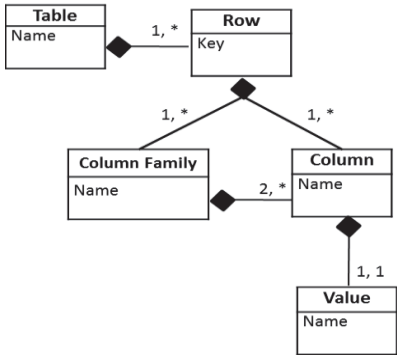

Figure 3: UML class diagram of the concepts of a column-oriented NoSQL data model.

Finally, the timestamp allows checking data coherence. Each value is allocated a timestamp by the system for the purpose of data consistency. It is noteworthy that data replication across different machines (nodes) required by the data management in a distributed environment (Eventually consistent) sometimes leads to different versions of the same data during updates. The timestamp associated with each value means that it is the most recent version which will be taken when a query is entered into the database [15].

Moreover, HBase and Cassandra from the Apache Foundation and BigTable from Google are three examples of column oriented NoSQL DBMS. In the next section, we present a logical model which allows implementing data warehouses under a column-oriented NoSQL DBMS.

4 The logical model for the columnar NoSQL warehouses

In order to implement the big data warehouses within the column-oriented NoSQL model, we propose three approaches namely NLA (Normalized Logical Approach), DLA (Denormalized Logical Approach), and DLA-CF (Denormalized Logical Approach by using Column Family). Each one differs in terms of the structure and the attribute types used when mapping is performed.

The first one uses different tables for storing fact and dimension, and use only the simple attribute for representing measure and dimension attributes. The second approach proposes storing the fact and dimensions into one table, and uses only the simple attribute for representing measure and dimension attributes. The third approach proposes storing the fact and dimensions into one table, and uses only the composite attribute for representing measure and dimension attributes. These approaches are described below.

4.1 Normalized Logical Approach (NLA)

This approach proposes to map the dimensional model of data warehouse by the normalized approach as the relational does. In order to achieve this, the classic dimensional models are converted towards relational logical models. The dimensions and the facts are stored separately on different tables. To ensure the links between these two entities (dimension-fact), the dimension table identifier is duplicated in the fact table. However, without the referential integrity constraints in the columnar NoSQL DBMS, it is the responsibility of the customer application level to check this scheduler.

4.2 Denormalized Logical Approach (DLA)

This approach proposes transforming the data conceptual model into a model based on a large structure (table) called *BigFactTable*, which keeps the facts and the dimensions joined. On the opposite of the normalized approach which separates the facts from their dimensions, this approach favors the denormalization of the schema by integrating, in the same table, the fact and the dimension. This process is very frequent in the modeling of the data warehousing, particularly in the management of the dimensional hierarchies; as explained by [16]. To map measures and dimensions into logical model, this approach uses the simple attribute proposed by the columnar NoSQL model. Thus, the fact and dimension values are now identified by the same identifier (row key) of the table, and we have no longer to achieve joining between tables when aggregation is performed.

4.3 Denormalized Logical Approach by using Column Family (DLA-CF)

This approach proposes transforming the data conceptual model into columnar NoSQL logical model as Denormalized Logical Approach does. However, this approach uses the composite attributes to map the measures and dimensions instead the simple ones. Indeed, each dimension is mapped into a column family and the attributes belonging to the same dimension are gathered in one column family. This allows attributes belonging to a given dimension to be shared in the same disk space which improves the column access time especially when decisional query involves several attributes of the same dimension (hierarchy: year, trimester and month) to perform aggregations.

5 Mapping from the dimensional model to the columnar NoSQL logical models

In order to define the rules that cover the mapping process from the dimensional model to the candidates' columnar NoSQL logical models defined above (section 4), we first formalize the different instantiations that lead to these logical models.

5.1 Formalization

Given data warehouse dimensional model DW composed by the couple (F, D). F represents the set of measures $F = \{M_1, M_2, \dots, M_q\}$, and D represents a set of dimensions $D = \{D_1, D_2, \dots, D_j\}$. Each dimension D grouped a set of attributes represents the axe of analysis used for observing the measure attributes M_q, it is defined by $D_j = \{At_1^j, \dots, At_k^j\}$

Definition 1 (NLA)

According to NLA, the instantiation of DW leads to map the fact *F* and the dimensions *D* to separate tables called respectively *FT* (fact table) and *DT* (dimension table). The simple attribute is used for representing both the dimension and measure attributes; such as, $\exists\ E \subset DT_j : E \to DT_j.At_k^j$, and $\exists\ E' \subset FT : E' \to M_q \wedge E$. It means that for each dimension table, there is at least an attribute which uniquely identify all the attributes of dimension, and there is another one which identify both all measure attributes and the identifiers of dimensions.

Definition 2 (DLA)

The instantiation of DW according to DLA leads to map the fact *F* and the dimensions *D* to the same table called *BigFactTable* (*BFT*). This approach uses the simple attribute for representing both the dimension and measure attributes. Thus, we consider that the logical modeling is performed according to DLA, if and only if there is a sub set of attributes *E* included in *BFT*, such as: $\exists\ E \subset BFT : E \to BFT.At_k^j \wedge BFT.M_q$. It means there is at least an attribute which uniquely identify all the attributes of dimension and all measure attributes in the *BigFactTable*.

Definition 3 (DLA-CF)

According to DLA-CF, the instantiation of DW leads to map the fact *F* and the dimensions *D* to the same table called *BigFactTable* (*BFT*) by using the column family structure *CF*. Indeed, all measure attributes are gathered into a column family, and each dimension is converted to a column family, too. Thus, the dimension attributes which belong to the same dimensions are gathered into a column family. We consider that the logical modeling is performed according to DLA, if and only if there is a sub set of attributes *E* included in *BFT*, such as: $\exists\ E \subset BFT : E \to BFT.CF_j.At_k^j \wedge BFT.CF.M_q$. It means there is at least an attribute which uniquely identify all the attributes of dimension and all measure attributes in the *BigFactTable*.

5.2 Mapping rules from the dimensional model

At the conceptual modeling level, the dimensional model is independent from the details related to data structuring and the environment implementation; hence, we adopt the dimensional model as presented by [14] without any expansion or modification. However, we expose, in this section, the rules which allow to map a dimensional model already established towards a logical model according to the three approaches that we propose in this work.

Conceptual model to NLA: in order to instantiate from the conceptual model by using this approach, the following rules must be applied:

(R1) Each fact becomes a table called fact table *FT*, and each dimension becomes a table called *DT*.

(R2) Each measure $M \in F$ is translated within FT as a simple attribute ($i.e\ FT.M$).

(R3) Each dimension *D* and each attribute $At_k \in D$ is mapped into *DT* as a simple attribute (i.e. $DT_j.At_k^j$), and the *FT* is completed by simple attribute $DT_j.At_k^j$ (the value reference of the linked dimension).

Conceptual model to DLA: in order to instantiate from the conceptual model by using this approach, the following rules must be verified:

(R1) Each fact and dimension is converted in one large table called *BigFactTable BFT*.

(R2) Each measure *M* ∈ *F* is translated within *BFT* as a simple attribute ($i.e.: BFT.M$).

(R3) For all dimensions *D*, each attribute $At_k \in D$ is translated into a simple attribute (i.e. $BFT.At_k$).

Conceptual model to DLA-CF: in order to instantiate from the conceptual model by using this approach, the following rules must be checked:

(R1) Each fact and dimension is converted in one table called *BigFactTable BFT* as composite attributes (column families).

(R2) Each measure $M \in F$ is mapped as a simple attribute and included in a column family into a *BigFactTable* ($i.e.: BFT.CF.M$).

(R3) Each dimension D is translated into a composite attribute (i.e. $BFT.CF_j$), and each attribute $At_k^j \in D_j$ is translated as a simple attribute included in the CF_j (i.e. $BFT.CF_j.At_k^j$).

The matching between entities from the conceptual model with those from the logical models that we propose is shown in the following table:

Conceptual model	NLA	DLA	DLA-CF
Fact F	FT	BFT	BFT
Measure M	FT.M	BFT.M	BFT.CF.M
Dimension D	DT	BFT	BFT.CF
Dimension attribute At	DT.At	BFT.At	BFT.CF.At

Table 1: Matching between the conceptual model and the logical models.

6 Experiments

In this section, we have evaluated the performances of the star data warehouse under the columnar NoSQL DBMS. For this reason, we have implemented a decisional benchmark SSB within HBase columnar NoSQL DBMS according to three (3) approaches. The first one implements the SSB following the normalized logical approach NLA; we called this data warehouse NLA-SSB. The second approach denormalizes the schema of data warehouse and implements the SSB according to the denormalized logical model without using the family columns DLA. We called it DLA-SSB. The third and last approach implements the SSB according to the denormalized logical model by using column family DLA-CF. We called it DLA-CF-SSB. To achieve this evaluation, we conducted two experiments to study the impact that the choice of approach used to implement a data warehouse under the column oriented NoSQL DBMS may have on the execution time of the decisional queries.

6.1 Test environment

In order to perform our experiments within a column oriented NoSQL and distribute environment, we have put in place a non-relational and distributed storage and processing environment [17]. This environment is based on a private Cloud Computing architecture produced using the Hadoop-2.6.0 and a HBase-0.98.8 DBMS, for managing data in a distributed environment. In order to simplify data handling and boost the performance of the HBase DBMS, we strengthened this configuration with an SQL interface for HBase, called Phoenix-4.1.0. This latter is an open source and allows the data handling at the HBase level (scan, put and get) to be combined to express a selection of data and to apply filters [18].

The test environment is a cluster made up of 25 machines (nodes). Each machine has an intel-Core TMi5-3220M CPU@3.30 GHZ processor with 8GB RAM. These machines operate with the operating system Ubuntu-14.04 and are interconnected by a switched Ethernet 100 Mbps in a local area network. One of these machines is configured to perform the role of Namenode in the HDFS system, the master and the Zookeper of HBase [19]. However, the other machines are configured to be HDFS DataNodes and the HBase RegionServers. Although the private Cloud architecture we used is limited in terms of capacity (number of nodes composing the cluster), it is sufficient to allow us to deploy a non-relational data warehouse with scaling-up and to apply a queries set in a distributed environment.

6.2 Data set

In order to perform our experiments, we used data generators of SSB which are available according to normalized[1] and denomalized[2] approaches [8], and we populated NLA-SSB, DLA-SSB, and DLA-CF-SSB data warehouses according to SF = 1000, this allows to generate fact table with 6×10^9 tuples of data sample.

6.3 Queries set

For our experiments, we used a queries set composed of eight (8) queries which are divided into two categories as depicted in table 2. The first category is composed of four (4) queries; they gradually increase in the number of dimensions involved when aggregation is performed. Each query in this category uses one attribute per dimension. The second category is composed of four (4) queries in which they involve only one dimension and gradual increase in the number of dimensions attributes when aggregation is performed.

Queries set	Query	Dimension	Attributes	Measure
Category-1	Query 1.1	Date	year,	Sum (revenue)
	Query 1.2	Date, Part	year, category	
	Query 1.3	Date, Part, Supplier	year, category, region	
	Query 1.4	Date, Part, Supplier, Customer	year, category, region (Supplier), region (Customer)	
Category-2	Query 2.1	Part	color	
	Query 2.2		color, type	
	Query 2.3		color, type, size	
	Query 2.4		color, type, size, container	

Table 2**:** Descriptive table of queries set

6.4 Experiment 1

In this experiment, the aim is to study the execution time impact of the normalized and denormalized approaches by

[1] https ://github.com/electrum/ssb-dbgen
[2] https://github.com/Dehdouh/DBGEN-CNSSB

using queries which involve attributes from different dimensions when aggregations are performed. To do this, we applied the first queries set category to NLA-SSB, DLA-SSB, and DLA-CF-SSB data warehouses. The results we obtained are shown in the following figure:

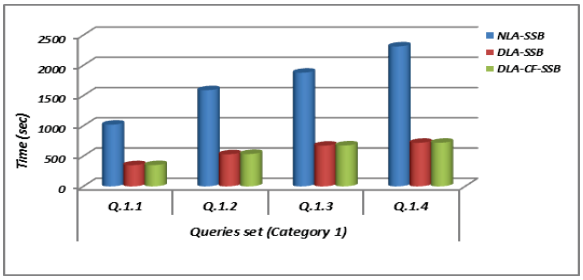

Figure 7: Execution time of the category 1 of queries set

We observed that the denormalized data warehouses represented by DLA-SSB and DLA-CF-SSB show better performance than normalized data warehouse represented by NLA-SSB. Indeed, the query execution times obtained from the data warehouses DLA-SSB and DLA-CF-SSB are better until three times than those executed by the data warehouse NLA-SSB. This is because implementing data warehouse according to normalized approach entails higher costs for materializing the link between dimension and fact especially when the queries involve more joins between the tables for performing aggregations.

However, for denormalized warehouses (DLA-SSB and DLA-CF-SSB), we found that gathering the dimension attributes in a column family does not impact the warehouse performance when the query handles attributes belonging to different dimensions.

6.5 Experiment 2

In this experiment, the aim this time is to study the execution time impact of the normalized and denormalized approaches by using queries which involve only one dimension and gradual increase in the number of dimensions attributes when aggregation is performed. To do this, we applied the second queries set category to DLA-SSB, DLA-CF-SSB, and DLA-CF-SSB data warehouses. The results we obtained are shown in the following figure:

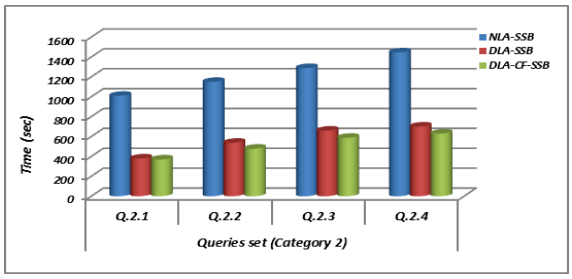

Figure 8: Execution time of the category 2 of queries set

We observed that the query execution times is different for each data warehouse and gives advantage to DLA-CF-SSB. Indeed, the queries used in this experiment (category 2) involve only one dimension when performing aggregations. In the case of NLA-SSB data warehouse, the join between facts table and dimension (Part) is performed. Thus, involving another attribute belonging to the same dimension (Part) entails only additional time related to its scan. This time is lower than the time of joining additional dimension.

On the other hand, we found that DLA-CF-SSB data warehouse performs execution times until 10 % better than DLA-SSB data warehouse. In the context of big data warehouse, this is very important especially in the case of data warehouses characterized by a large number of attributes which compose the dimensions. Indeed, HBase DBMS stores columns by lexicographical order which may sometimes store the columns of the same dimension separately in different disk spaces. Thus, using the column family allows having the attributes belonging to the same dimension stored in the same disk space.

Based on these results, we found that the use of the column family for implementing columnar NoSQL data warehouses gives benefits only with decisional queries handling attributes belonging to the same dimension (i.e.: the dimension hierarchy is involved).

7 Conclusion

Facing the emergence of large and unusual volumes of data (big data), we have proposed three approaches which allow mapping the multidimensional conceptual data model into a logical modeling adapted to the column-oriented NoSQL data warehouses. We have called these approaches; NLA, DLA, and DLA-CF. Each one differs in terms of the structure and the attribute types used when mapping is performed. We have described each one and showed the rules governing the instantiation of the conceptual model. Each approach has its weaknesses and strengths, and the choice depends of the use case.

We have used these approaches for evaluating the performance of SSB data warehouse under distributed environment when applied on decisional queries set. Then, we have observed that the denormalized data warehouses represented by DLA and DLA-CF show better performance than NLA which represents normalized approach. Indeed, the NLA uses less disk memory, but it is quite inefficient when queries with joins are performed.

Morover, we have found that the DLA-CF is more efficient than DLA, but only when query handling attributes belong to the same dimension. Thus, the use of the column family depends of the type of the queries which are applied to the columnar NoSQL data warehouse.

As a perspective, we tend to explore in the next work, the instantiation of data warehouse across other different NoSQL systems namely: key/value, documents-oriented, and graph-oriented to analyze the big data warehouses. These systems give efficient managing of big data corresponding to different contexts.

8 References

[1] Inmon, W. "Building the data warehouse". QED Information Sciences, Inc, 1992.

[2] Kimball, R. "Kimball Dimensional Modeling Techniques", Kimball Group University, 2013.

[3] Coronel, C., Morris, S., Rob, P.: "Database Systems: Design, Implementation, and Management", Cengage Learning, 2012.

[4] Chaudhuri, S., Dayal, U., Ganti, V. "Database technology for decision support systems", IEEE Computer Society, 48--55, 2002.

[5] O'Neil P., O'Neil B., Chen X.: The Star Schema Benchmark (SSB), http://www.cs.umb.edu/\~poneil/ StarSchemaB.PDF, (2009).

[6] Li, C. "Transforming relational database into HBase: A case study", International Conference on Software Engineering and Service Sciences (ICSESS), 683--687, 2010.

[7] Han, D., Stroulia, E. "A three-dimensional data model in hbase for large time-series dataset analysis", IEEE MESOCA, 47--56, 2012.

[8] Dehdouh, K., Boussaid, O., Bentayeb, F. "Columnar NoSQL Star Schema Benchmark", Model and Data Engineering MEDI, 281--288, 2014.

[9] Chevalier, R., El Malki, M., Kopliku, A., Teste, O., Tournier, T. "Implementing Multidimensional Data Warehouses into NoSQL". International Conference on Enterprise Information Systems (ICEIS 2015), 172--183, 2015.

[10] Jing, H., Haihong, E., Guan, L., Jian, D. "Survey on NoSQL database", International Conference on Pervasive Computing and Applications (ICPCA), 363--366, 2011.

[11] Pokorny, J. "Nosql databases: A step to database scalability in web environment", Association for Computing Machinery ACM, 278--283, 2011.

[12] Jerzy, D. "Business Intelligence and NoSQL Databases", Information Systems in Management 1, 25--37, 2012.

[13] Matei, G. "Column-oriented databases, an alternative for analytical environment". Database Systems Journal, 3--16, 2010.

[14] Apache Software Foundation. "The Apache HBase Reference Guide", http://hbase.apache.org/book/joins.html, 2014.

[15] Cattell, R. "Scalable SQL and NoSQL Data Stores", Association for Computing Machinery ACM SIGMOD Record, 12--27, 2011.

[16] Kimball, R., Ross, M. "The data warehouse toolkit: The complete guide to dimensional modeling", Second Edition, Inc, 2002.

[17] Taylor, R. "An overview of the Hadoop-MapReduce-Hbase framework and its current applications in bioinformatics". BMC Bioinformatics Journal. 2010.

[18] James, T. https://github.com/forcedotcom/phoenix/wiki /Performance, 2013.

[19] Hunt, P., Konar, M., Junqueira, F. P., Reed, B. "Zookeeper: Wait-free Coordination for Internet-scale Systems", Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, 11--24, 2010.