International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 11 Issue 03 March 2023, Page no. – 3311-3317

Index Copernicus ICV: 57.55, Impact Factor: 8.187

DOI: 10.47191/ijmcr/v11i3.05

Approaches to Build and Develop Data Warehouses According to the NOSQL Model

KABAMBA LUBANGI Nico[1], **NYEMBO MPAMPI Augustin**[2]
[1,2] Faculty of Computer Science / Our Lady of Lomami University

ARTICLE INFO	ABSTRACT
Published Online: 28 March 2023	Traditional data warehouses support atomic, consistent, isolated, and durable (abbreviated, ACID) transactions. The ACID property of transactions guarantees the correction of concurrent access to a data warehouse. However, as the CAP[1] theorem explains, this property is always costly in terms of performance. For this reason, traditional data warehouses show their limits when used in a very demanding context, such as the Cloud.
Corresponding Author: **KABAMBA LUBANGI Nico**	The work presented in this article aims to propose approaches to build and develop data warehouses according to the column-oriented NoSQL model.

KEYWORDS: data warehouses, NoSQL model, Approaches

I. INTRODUCTION

Indeed, the different NoSQL models have become standards in the storage and management of massive data. They were originally designed to build databases whose storage model is the “key/value” model. Other models then appeared to take into account the variability of the data: column-oriented, document-oriented and graph-oriented models. To develop massive data warehouses, our choice fell on the column-oriented NoSQL model because it appears to be the most appropriate for processing decision-making queries that are defined according to a set of columns (measures and dimensions) from from the warehouse. In order to exploit columnar data warehouses, for the creation of OLAP cubes, we have proposed the CN-CUBE (Columnar-NoSQL Cube) operator which takes into account the facts and dimensions which are grouped in the same table when the generation of cubes from a denormalized warehouse according to a certain logical model.

sMC-CUBE (MapReduce Columnar-Cube) to build OLAP cubes stored in columns in a distributed environment using the invisible join and the MapReduce paradigm to parallelize processing.

Finally, we are interested in the techniques of grouping attributes for the constitution of families of columns. Our goal is to obtain homogeneous sets of attributes to increase the performance of decisional queries.

We then proposed a strategy for grouping attributes, from an initial load of queries, using two algorithms: a metaheuristic, in this case Optimization by Particle Swarm (PSO), and the algorithm excavation, k-means.

This article is organized as follows. The first section presents general information on cloud computing and business intelligence. The second section deals with the problem of formalizing the column-oriented logical model, and the definitions of the transformation rules. The third section is devoted to the detailed description of the OLAP cube calculation process and the presentation of the CN-CUBE operator. And finally the fourth details the problem related to data access in a warehouse based on a columnar NoSQL system and here we detail the proposed approach.

[1] G. SETH et N. LYNCH, Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services, dans, SIGACT News, 2002, p.51–59.

II. 1. CLOUD COMPUTING AND BUSINESS INTELLIGENCE

A. 1.1. Cloud Computing

Cloud Computing is first and foremost a business need: how to reduce the costs of IT infrastructure? This includes the costs of purchasing the equipment, but also of its maintenance, its security, etc. The idea is therefore to use servers (and more generally resources) from another provider, self-service, and remotely. The IT infrastructure is therefore decentralized and made accessible no matter where in the world. The term “Cloud” was thus born: we access computing resources as if they were placed in the clouds (we do not have them at hand, but we can easily access them from everywhere).

We can generally consider, when we talk about cloud computing, that it is a question of making data and applications accessible and usable through a network. This term refers to both applications as services on the Internet and the hardware and software that make it possible to provide these services[2].

The Cloud can be seen as a commercial service subscription offer. This subscription can take various forms, which are grouped into 3 main categories:

- IaaS (Infrastructure as a Service): this involves renting an IT infrastructure on which an operating system can be installed, etc. It is said to be a "low level" service because it is closer to the machines and technical details than the other services. So we have more flexibility, but more elements to manage ourselves.
- PaaS (Platform as a Service): this time, the supplier has already installed the operating system and the basic services on its machine. And the customer can install their own applications on top of it. This is the case of shared web hosting where the service provider rents you a (virtual) server and on which you can place the files of your website.
- SaaS (Software as a Service): in this case, the software is made available to the customer. There are many different offers, you can access them either via a browser or by installing software on your PC.

B. 1.2. Business Intelligence

Business intelligence (decisional computing in French) refers to the means, tools and methods that make it possible to collect, consolidate, model and restore a company's data in order to provide decision support and enable managers to corporate strategy to have an overview of the activity handled.

This type of application generally uses a data warehouse to store data from several heterogeneous sources and uses batch processing to collect this information.

A BI system is made up of four main categories of tools: ETL, data warehousing, data mining and reporting. The part on which our study focused is based on the second tool, so we will briefly present data warehousing and the multidimensional modeling used for the organization of data in a data warehouse.

III. 2. DATA WAREHOUSE

A data warehouse is a database containing all the functional data of a company.

Its purpose is to provide a set of data serving as a unique reference, used for decision-making in the company through statistics and reports produced via reporting tools.

From an architectural point of view, there are two ways to apprehend it :

- The "top-down" architecture: according to Bill Inmon, the data warehouse is a database at the detail level, consisting of a global and centralized repository of the company. In this, it differs from the data mart, which functionally groups, aggregates and targets data.
- The "bottom-up" architecture: according to Ralph Kimball, the data warehouse is gradually constituted by the company's data marts, thus bringing together different levels of data aggregation and historization within a same basis.

A datamart (data store in French) is a subset of data from a data warehouse.

IV. 3. MULTIDIMENSIONAL MODELING AND MIGRATION TO THE COLUMN-ORIENTED NOSQL MODEL

A. 3.1. Multidimensional modeling

The OLAP concept (Online Analytical Processes) was introduced by E.F Codd, to render analyzes on complex enterprise data along several dimensions[3]. He established a set of rules that a database must have to integrate this concept.

Multidimensional modeling was introduced to facilitate OLAP processing in order to make the conceptual model of data organization more in line with their representation. "The emphasis of OLAP systems is on exible data grouping and efficient aggregation evaluation obtained groups"[4].

---

[2] M. ARMBRUST, et al, *Above the clouds : A berkeley view of cloud computing*. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[3] E.F. CODD, S.B. CODD, and C.T. SALLEY, *Providing OLAP (on-line analytical processing) to user-analysts*, An IT mandate, 1993.

[4] C. LI and X.S. WANG, *A data model for supporting on-line analytical processing*, in Proceedings of the fifth international conference on Information and knowledge management, p. 81-88, ACM, 1996.

To ensure a generic transformation process, we define the multidimensional model employed at the conceptual level[5].

A multidimensional scheme, denoted E, is defined by (F^E, D^E, $Star^E$) or :

- $F^E = \{F_1, \ldots, F_n\}$ a finite set of facts,
- $D^E = \{D_1, \ldots, D_m\}$ a finite set of dimensions,
- $Star^E : F^E \rightarrow 2^{D^E}$ is a function that associates the facts of F^E with sets of dimensions, according to which they can be analyzed (2^{D^E} being the set of parts of the set D^E).

One dimension, denoted $D_i \in D^E$ (wrongly noted D), is defined by (N^D, A^D, H^D) or :

- N^D is the dimension name,
- $A^D = \{ a_1^D, \ldots, a_u^D\} \cup \{id^D, All^D\}$ is a set of dimension attributes,
- $H^D = \{ H_1^D, \ldots, H_v^D\}$ is a set of hierarchies

A hierarchy, noted $H_i \in H^D$, is defined by (N^{Hi}, $Param^{Hi}$, $Weak^{Hi}$) or :

- N^{Hi} is the name of the hierarchy,
- $Param^{Hi} = < id^D, pi^{Hi}, \ldots, p_{vi}^{Hi}, All^D >$ is an ordered set of $vi + 2$ attribute called parameters which represent the tick marks of the dimension, $\forall k \in [1 \ldots vi], p_k^{Hi} \in A^D$,
- $Weak^{Hi} : Param^{Hi} \rightarrow 2^{A^D - Param^{Hi}}$ is a function associating one or more weak attributes to the parameters.
- A fact, noted $F \in F^E$, is defined by (N^F , M^F) or :
- N^F is the name of the fact,
- $M^F = \{f1(m_1^F), \ldots, fv(m_v^F)\}$ is a set of measures associated with an aggregation function fi.

B. 3.1. Conversion to column-oriented NoSql model

Implementing a data warehouse in a columnar NoSQL system takes into account the specifics of the physical data storage environment.

Remember that the latter are organized into families of columns made up of a set of attributes. Thus, it is a principle of vertical partitioning of data.

Figure 5 shows an example representing a table *T* with 7 attributes and 3 families of columns. Each family *CFi* consists of a set of attributes each having a value, each row of data is referenced by a row key *Ri*.

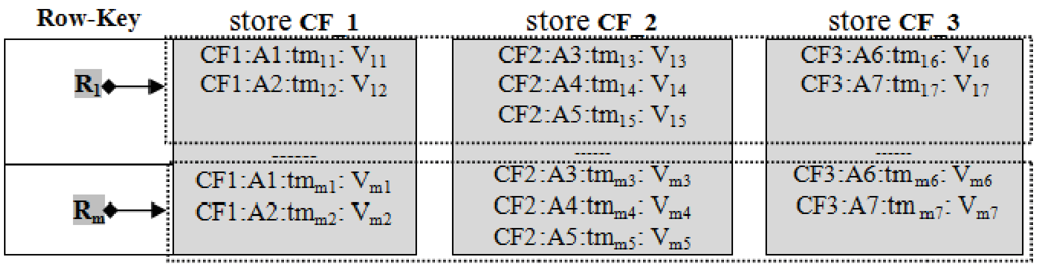

Figure 5: Column-oriented storage of a table

In reality from a storage point of view (Figure 5), all the data, referring to the same row key (Rowkey), are stored together, the name of the column family then acting as the key for each of its columns and the row key as the key to all the attributes of the same “record” in the relational sense of the term. We can notice in this storage technique that two elements influence the speed of execution of queries: the choice of columns and that of rows. For a request relating, for example, to the values of attributes {*CF*_1 : *A*1, *CF*_2 : *A*4, *CF*_3 : *A*6}, the table is traversed at the level of the column families 3 times; the search will be done on 3 different data partitions: CF_1 , CF_2 and CF_3 with a complete scan of the values of the attribute concerned. This then requires a high execution time to reach the values.

However, the implementation of data warehouses, according to a columnar NoSQL model, is based on denormalization methods, which groups both fact and dimension data in the same table. This forces the duplication of dimension data for each fact instance, which minimizes the benefit of grouping attributes into column families. Therefore, to avoid access to several families of columns (several metadata) and a complete scan of all the data, it is important to propose a grouping of the data which is rather appropriate to the needs of the users. Additionally, typical NoSQL implementations run entirely in memory, thus incurring a cost. This will be particularly high depending on the size of the data volume. Therefore, the problem that arises when building the table is how to define the right number (threshold) of column families for a good attribute grouping strategy.

[5] F. RAVAT, O. TESTE, R. TOURNIER, et G. ZURFLUH, *Algebraic and graphic languages for OLAP manipulations*, IJDWM, Orlanda, p. 17–46, 2008.

We will discuss, in the next section, the approach proposed for the implementation of data warehouses according to the column-oriented NoSQL model.

C. 3.3. Grouping columns into families

1) 3.3.1. Principle

Our method for implementing a relational warehouse (star or snowflake schema) according to a column-oriented NoSQL logical model is based on the following process :

- Extraction of attributes from fact and dimension tables;
- Grouping of attributes and construction of column families according to OEP or k-means;
- Generation of a schema (metadata) of the data warehouse in the NoSQL model in columns according to the groupings obtained;
- Data preparation and warehouse loading

2) 3.3.2. Formalization

A data warehouse is defined by $D = \{d1, d2, ..., dj\}$ dimension tables and a fact table F of n attributes. Each dimension table $(d_m)_{m=1...j}$ is composed of several attribute A_m^i with $i \in [1, k]$, such as $d_m = \{A_m^1, A_m^2, \ldots, A_m^k\}$ where k can vary from dimension to dimension. We also consider a set of queries $Q = \{q_1, q_2, ..., q_q\}$. These exploit the entire warehouse schema through selection, join and aggregation operations (MAX, COUNT, SUM, etc.).

Each qi query contains a set of attributes.

Let $F = \{FC_1, FC_2, ..., FC_w\}$ be the set of column families that will be generated.

The number of column families (w) is defined such that: $2 \leq w \leq W$.. To control the maximum number of column families to create, a threshold W is set a priori.

3) 3.3.3. Extracting attributes from fact and dimension tables

Our technique for implementing a warehouse based on a columnar NoSQL model is based on a statistical analysis of the most frequent queries and relies on both qualitative and quantitative information on data usage. These concern the relations, the set of attributes and the tuples accessed, the predicates relating to the attributes, the type of requests, the number of read requests, the frequency of execution of a request, the site from which the query is executed, the storage capacity and the cost of transferring data between sites. This first step then consists in processing all the attributes (present in the Select and Where clauses) relating to the query load, to build the attribute usage matrix (MUA) and the affinity matrix (MAA).

To do this, we were inspired by the work of Navathe et al[6]. The latter use the principle of affinities between attributes to design groups of attributes.

4) 3.3.4. Construction of column families

In this step, our goal is to define a logical NoSQL schema of the warehouse that best optimizes data access for queries. Our solution is to implement a process aimed at grouping attributes that are frequently queried together. This grouping will make it possible to form families of columns making up the logical schema of the NoSQL warehouse. For this, we have chosen to use two types of algorithms: (1) an OEP metaheuristic developed by Eberhart et al.[7] ; (2) and the k-means algorithm[8]. Our choice to use these two OEP and k-means algorithms is motivated by the fact that we can control the number of classes (column families) in k-means; this turns out to be an advantage, as long as we want to limit the number of column families.

In addition, EOP offers the possibility to control the number of attributes per group (family of columns), and to have the same advantage as k-means. This helps us build column families with the same number of attributes to balance the load.

a) a. Particle Swarm Optimization (PSO)

This algorithm is inspired by swarms of insects or animals and their movements in groups to find food. Initially the swarm is randomly distributed in space, each particle having a random speed. Then the particles move through the search space based on limited information, i.e. each particle has to decide its next move and its new speed. It does this by linearly combining three pieces of information: (1) its current speed; (2) its best performance: each particle is able to evaluate the quality of its position and to remember its best performance, i.e. the best point through which it has already passed; (3) the best performance of its neighbors (its informants): each particle can interrogate its nearest neighbors to find out their best performance in order to decide on its movement. Particle swarm algorithms can be applied to both discrete and continuous data.

[6] S. NAVATHE, S. CERI, G. WIEDERHOLD, et J. DOU, *Vertical partitioning algorithms for database design*, ACM *Transactions on Database Systems (TODS)* 9 (4), 680–710, 1984.

[7] R. C. EBERHART, J. KENNEDY, et al., *A new optimizer using particle swarm theory*, In, *Proceedings of the sixth international symposium on micro machine and human science*, Volume 1, p. 39–43. New York, NY, 1995.

[8] J. MACQUEEN, et al., *Some methods for classification and analysis of multivariate observations*, In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, Volume 1, p. 281–297, 1967.

b) b. Adaptation of the OEP algorithm for the grouping of Attributes

Our problem is defined by:

- $R = \{A_1, A_2, ..., A_{nb}\}$: set of attributes.
- Q : total number of frequent requests
- Fq : access frequency of the request q, for q = 1, 2, ..., Q
- W : maximum number of column families ($2 \leq W$)
- P : number of attributes per family of columns ($1 \leq P < nb$)
- C : set of physical constraints (number of nodes in the cluster, storage capacity of the nodes, memory space used by the machines, size of the warehouse, etc.)
- S = {S1, S2, ..., St}: set of all feasible solutions, such that t is the maximum number of iterations of the OEP, $(Si)_{i=1,..,t} = \{FCi_1, FCi_2, .., FCi_w\}$, où les $(FCi_j)_{j=1,..,W}$ are finite subsets of P attributes of R.
- F: an objective function which takes its values on S.

The problem consists in finding a solution S* ∈ S optimizing the value of the objective function F such that: F(S*)≤F(Si), for any element Si ∈ S, and F the objective function to be minimized.

The objective function F is used to measure the quality of solutions $(S_i)_{i=1,..,t}$ obtained after each iteration of OEP. Our cost function is inspired by the work of Derrar et al.[9], which consists in evaluating, during the design phase, the relevance of the generated partitioning schemes. Initially, this function is calculated using the square error (Square-Error) and taking into account the frequency of access of requests to the attributes of the different groups. The squared error of the global partitioning scheme of the relation R is calculated as follows:

$$E_w^2 = \sum_{j=1}^{W} \sum_{q=1}^{Q} [(fq)^2 \times s_j^q (1 - \frac{(s_j^q)}{P})]$$

With s_j^q the number of attributes in the group FC_{i_j} on a partitioning scheme If accessed by query q.

In order to find the optimal solution S^*, we use the OEP-FC algorithm which exploits the objective function F. This function evaluates, for each iteration i of the algorithm, the square error E_w^2 of the Si-scheme generated during an iteration. It is therefore a question of determining a scheme $S*$ which minimizes the value of E_w^2, under the set of constraints C, this function is defined as follows:

$F(S^*) = Min\ (E_W^2, S_i)_{i=1,..\ ,t}$

c) c. Application of k-means for attribute grouping

This algorithm takes as input a set of points and an integer k; the problem consists in dividing the points into k groups by minimizing the sum of the squares of the distances between the points of a group and its center. The points are then the attributes and their distances are the affinities. k-means takes as input the attribute affinity matrix (AAM) and the number of clusters k, and returns the column families.

5) 3.3.5. OLAP operators

OLAP operators allow, from a data warehouse, to extract data cubes corresponding to analysis contexts. A data cube is a multidimensional structure which makes it possible to represent the fact to be observed (measurements) according to several axes of observation (dimensions)[10]. The cube calculation allows to have aggregations beyond the limits of Group by. This consists in calculating all the aggregates according to all the levels of all the dimensions.

A column-oriented NoSQL database is a very suitable storage model for data warehouses and online analytics[11]. Indeed, columnar modeling is naturally appropriate to the multidimensional data structure of OLAP cubes. Unfortunately, we can deplore the fact that currently, NoSQL DBMS do not yet have aggregation operators to build OLAP cubes. To overcome this problem, we propose in this article an aggregation operator, called CN-CUBE (Columnar NoSQL CUBE), for column-oriented NoSQL DBMSs.

This operator is used to calculate OLAP cubes from warehouses implemented using column-oriented NoSQL databases. CN-CUBE uses a view, result of an extraction query, defined on the attributes (dimensions and measures) necessary for the calculation

[9] H. DERRAR, O. BOUSSAÏD, et M. AHMED-NACER, *Répartition des données d'un entrepôt basé sur l'optimisation par essaim particulaire*, dans, *EDA*, p. 141–149, 2008.

[10] J. GRAY, S. CHAUDHURI, A. BOSWORTH, A. LAYMAN, D. REICHART, M. VENKATRAO, F. PELLOW, et H. PIRAHESH, *Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub totals,* dans *Journal of Data Mining and Knowledge Discovery 1*, p. 29–53, 1997.

[11] D. JERZY, *Business Intelligence and NoSQL Databases*, dans *Information Systems in Management 1*, p. 25–37, 2012.

of the OLAP cube corresponding to an analysis need. This strategy makes it possible to reduce disk access by avoiding the return to the warehouse data for the calculation of the various aggregates.

Moreover, to calculate all the aggregates of the OLAP cube, CN-CUBE exploits the positions of the values in the columns, to calculate the different aggregates of the OLAP cube. We implemented the CN-CUBE operator and evaluated its performance on a data warehouse that we created within the column-oriented NoSQL DBMS HBase with Hadoop. The choice of the HBase DBMS and the Hadoop platform is motivated by their distributed context necessary for the storage and analysis of big data. The warehouse is fed by real data of the open data type which describes traffic injury accidents.

6) 3.3.6. The CN-CUBE operator

The CN-CUBE operator that we propose makes it possible to calculate the OLAP cube from column-oriented NoSQL data warehouses following three phases:

- First phase: It consists in defining a view on the attributes (dimension and measures) necessary for the calculation of the OLAP cube. Data that satisfies all predicates is retrieved from the column-stored data warehouse. Only values that have the most recent timestamp are considered. The result obtained is therefore a relation R composed of the columns which represent the axes of analysis and the column(s) representing the measure(s) to be aggregated;
- Second phase: In this phase, each dimension column of the relation R is hashed with the values that compose it to obtain the list of positions of these values. The values of these lists are binary, they can correspond to "1" or to "0"; the "1" indicates that the hash value exists at this position and "0" otherwise. These lists make it possible to have the aggregates of each dimension separately;
- Third phase: At this phase, the lists of positions of the dimensions at the level of R are associated via the "logical AND" operator. This operation is used to identify the values of the dimensions to combine and the values of the measure to aggregate that correspond to the different combinations. The grouping of the sub-results (totals and sub-totals) of the three phases allows the calculation of the cube.

D. 3.4. Process of running CN-CUBE operator in a cloud environment

To exploit big data, data warehouses need to opt for a necessarily distributed solution in order to be able to scale up. This solution is based on a data storage architecture distributed over several machines and parallelized data processing. Hadoop offers a distributed file management system, called HDFS (Hadoop Distributed File System), designed to store very large volumes of data across multiple machines and a distributed data processing system[12], called MapReduce[13]. The latter is a massively parallel processing model suitable for processing very large amounts of data. It is based on two main steps, Map and Reduce. The Map function divides the processing into sub-processes on the different nodes making up the cluster and the results are grouped together via the Reduce function[14].

The integration of a NoSQL database management system such as HBase with Hadoop, makes it possible to better structure and index the data to avoid browsing the entire cluster when accessing the data (full scan).

However, out of the three phases that characterize the execution of the CN-CUBE operator, the process of running the CN-CUBE operator on a multi-node cluster is done by triggering two main MapReduce jobs. The first executes the first phase which consists in building the intermediate result necessary for the calculations of all the parts of the OLAP cube and consequently providing the aggregates that can be calculated at this level. The result of the first job constitutes the input of the second job which is responsible for calculating from the intermediate result, the rest of the aggregates making up the cube (phases 2 and 3). Thus, phases 2 and 3 can only take place after completion of phase 1. The grouping of the results of the jobs constitutes the OLAP cube.

[12] K. SHVACHKO, H. KUANG, S. RADIA, et R. CHANSLER, *The Hadoop Distributed File System*, IEEE Computer Society, p.1–10, 2010.

[13] J. DEAN et S. GHEMAWAT, *MapReduce: Simplified Data Processing on Large Clusters*. Association for Computing Machinery ACM Commun 51, p. 107–113, 2008.

[14] C. LI et X.S. WANG, *Modèle de données pour le traitement OLAP*, Actes de la cinquième conférence internationale sur la gestion d’informations et connaissances, ACM, p. 81-88., 1996.

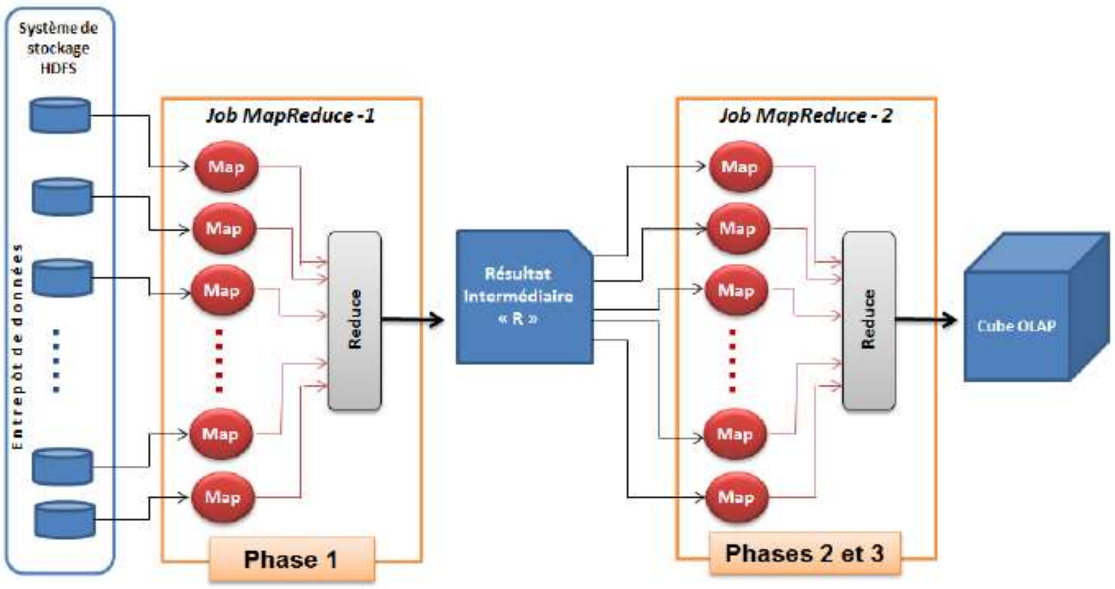

Figure 4: CN-CUBE operator execution process with MapReduce

V. CONCLUSION

This article describes the approach to implement a multidimensional data warehouse in a NoSQL system. We have defined a process that converts a conceptual multidimensional schema (star schema) into logical column-oriented NoSQL schemas. We proceeded to group attributes into families of more relevant columns, using the OEP-FC and k-means mining methods. This allowed us to obtain a performance gain in the processing of decisional queries.

We have proposed in this article, CNCUBE, an operator for calculating the OLAP cube. The advantage of this operator is that it uses the positions of values for the calculation of the aggregates of the OLAP cube. This way of proceeding considerably reduces the input and output flows of data.

REFERENCES

1. Abramova V., et Bernardino J., *Bases de données NoSQL : MongoDB vs Cassandra*. Dans Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E, 13). 10-12 juillet 2013, Porto, Portugal, p. 14-22.
2. Abramova V., Bernardino J., et Furtado P., *SQL ou NoSQL ? Évaluation du rendement et de l'évolutivité*, Int. J. Intégration et gestion des processus opérationnels (IJBPIM), volume 7, numéro 4, 2015, p. 314-321.
3. Adriano Girolamo Piazza, *NoSQL Etat de l'art et benchmark* ; Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES ; Genève, 9 octobre 2013 Haute École de Gestion de Genève (HEG-GE).
4. Andreas Meier, *Introduction pratique aux bases de données relationnelles*, Ed. 2, Springer-Verlag, 2006, p. 68.
5. Bruchez R., *Les bases de données NoSQL Comprendre et mettre en œuvre*, Paris, Eyrolles, 2013, p. 127.
6. Cattell R., *Base de données SQL et NoSQL évolutifs*. SIGMOD Record, volume 39, Issue 4, p. 12-27, Indiana, 2010.
7. Hadrien F., *SQL, NoSQL, NewSQL stratégie de choix*, Bachelor HES (Haute École de Gestion de Genève), 2015.
8. Hecht R., Jablonski S., *Évaluation NoSQL : Enquête axée sur les cas d'utilisation*, Dans Proceedings of the International Conference on Cloud and Service Computing, IEEE. 22-24 novembre 2012, Huashan Road, Shanghai, p. 336-341.
9. Heinrich L., Architecture *NoSQL et réponse au Théorème CAP*. Bachelor HES (Haute École de Gestion de Genève), 2012.
10. Kumar R., Gupta N., Maharwal H., Charu S., *et al.*, *Analyse critique de la gestion des bases de données à l'aide de NewSQL*, International Journal of Computer Science and Mobile Computing (IJCSMC), volume 3, numéro 5, 2014, p. 434-438.
11. Martins G., Bezerra P., Gomes R., Albuquerque F., Costa A., *Évaluer la dégradation du rendement dans les bases de données NoSQL générées par la virtualisation*. Dans Proceedings IEEE of 8th Latin American Network Operations and Management Symposium (LANOMS), 1-3 octobre 2015, João Pessoa, Brésil, p. 84-91.
12. Piekos J., *SQL vs NoSQL vs NewSQL : Trouver la bonne solution*. http://dataconomy.com/2015/08/sql-vs-nosql-vs-newsql-finding-the-right-solution/ (consulté en janvier 2018).