Document-oriented data warehouses : complex hierarchies and summarizability

Max Chevalier[1], Mohammed El Malki[1,2], Arlind Kopliku[1],
Olivier Teste[1], Ronan Tournier[1]

[1] Université de Toulouse, IRIT (UMR 5505), 118 route de Narbonne, Toulouse France
[2] Capgemini, 109 avenue du Général Eisenhower, BP 53655- 31036 Toulouse, France
`{Max.Chevalier,Mohammed.ElMalki,Arlind.Kopliku, Olvier.Teste,Ronan.Tournier}@irit.fr`

Abstract: There is an increasing interest in implementing data warehouses with NoSQL document-oriented systems. In the ideal case, data can be analysed on different dimensions and dimensions follow strict hierarchies that we can use to roll-up and drill-down on analysis axes. In this paper, we deal with non-strict and non-covering hierarchies, common issues in data warehousing a.k.a. summarizability issues. We show how to model these hierarchies in document-oriented systems and we propose an algorithm that can deal with summarizability issues. The new approach is tested and compared to existing approaches.

Keywords: data warehouses, document-oriented systems, NoSQL, summarizability

1 Introduction

There is an increasing interest in implementing data warehouses with NoSQL systems [19] including document-oriented systems such as MongoDB [5]. NoSQL systems are an interesting alternative to relational databases (RDBMS), because they offer interesting scaling, replication and flexibility features. Until now, the different studies have focused on modelling issues, instantiation and OLAP cuboids. The management of complex hierarchies [6,12] is an important issue in data warehousing. We introduce in this paper the management of complex hierarchies and summarizability issues with document-oriented data warehouses.

In OLAP settings, it is common to analyse data on different dimension combinations. During analysis, we can drill-down or roll-up at different levels of detail using the hierarchy of dimensions. It is common to have irregularities in these hierarchies such as non-strict hierarchies and non-covering hierarchies. The latter are also the cause of summarizability issues i.e. we cannot drill-down or roll-up in data. Several solutions have been proposed for summarizability issues, but these solutions are adapted to the relational model [6, 7,8,11,18] With these solutions, it is necessary to alter original schemas and to override attribute values to act as arrays. NoSQL systems have interesting features that can useful for dealing with complex hierarchies. This is the scope of this paper.

In particular, document-oriented systems are an interesting case study for managing complex hierarchies. They support atomic attributes as well as the complex attributes (nested records, arrays, …) for storing the data. Document-oriented systems are one of the most popular classes of NoSQL approaches [5]. Data is stored in documents and documents are grouped in collections [5,3]. Documents have a flexible schema. They contain key-value pairs where keys act as metadata (they represent the data structure). Values can be of simple data type (strings, numbers, dates…), but they can also be arrays or sub-documents. Documents within the same collection can have different schemas. Document-oriented systems have been shown to work well for implementing data warehouses. They can scale horizontally and exploit parallel computation for faster querying. However, until now, the management of complex hierarchies and summarizability issues have not been treated with NoSQL systems in an OLAP setting.

In this context, we extend our previous work on data warehouses implementation with document-oriented systems. We introduce support for storing complex hierarchies and support for data summarization on the complex hierarchies. Our new contribution can be summarized as follows:
we show how we can easily store complex hierarchies in documents
We propose an algorithm for summarizability issues in document-oriented data warehouses. We compare our algorithm to other state-of-the-art algorithms

The rest of this paper is structured as follows. In the next section, we introduce the data warehouse basic notions, the multidimensional data model and the complex hierarchies issues. Then, we propose our approach for modelling, storing and dealing with complex hierarchies. In the following section, we propose experimental work to validate our work. We summarize related work and we end with conclusions.
Data warehouses and complex hierarchies

2 Data warehouses, the multidimensional model, cuboids

To ease data analysis and decision making, it is common to centralize them in data warehouses [4]. These latter are suitable for on-line analysis called OLAP (On-Line Analytical Processing [9]). In this setting, data is modelled with a multidimensional model composed of measurable facts and analysis dimensions. Several analysis topics (called facts) regroup a set of indicators (called measures). The values of these indicators are observed by different analytical axes, also called dimensions. These dimensions are composed by attributes, which represent different levels of detail, which are themselves organized into hierarchies.

The traditional example in data warehouses concerns sales as the fact and dimensions like customer, date, supplier. For the sake of change, we will use another example from social media OLAP, more precisely the analysis of the tweets (microblogs). In figure 1, we show the multidimensional schema. The *tweet* is analysed according to three dimensions: *Time*, *User* and *Subject*. One of the analysis measures is the popularity of a tweet (the number of times a tweet has been retweeted). At different analysis levels, we may wish to have the total amount of retweets grouped by topic or by category or by month or year. The measures can be observed, for example based on the "time" dimension with three detail levels (day, month, year) organized in a hierarchy with "day" the lower detail level, "month" at a higher level and so on. The hierarchies are useful structures that are employed to ease the pre-calculation of induced agglomeration (for example, calculate the annual sales from the weekly values). Generally, the situations in the real world are modelled according to the simple hierarchies. The associations between the different levels of one simple hierarchy are the type "one-to-many", e.g. one category many sub-categories.

Below, we provide some formalization on the multidimensional data model and OLAP cuboids:

A **multidimensional schema**, namely E, is defined by (F^E, D^E, $Star^E$) where: $F^E=\{F_1,\dots, F_n\}$ is a finite set of facts, $D^E=\{D_1,\dots, D_m\}$ is a finite set of dimensions, $Star^E$: $F^E \rightarrow 2^{D^E}$ is a function that associates facts of F^E to sets of dimensions along which it can be analyzed ($2^{D^E}$is the *power set* of D^E).

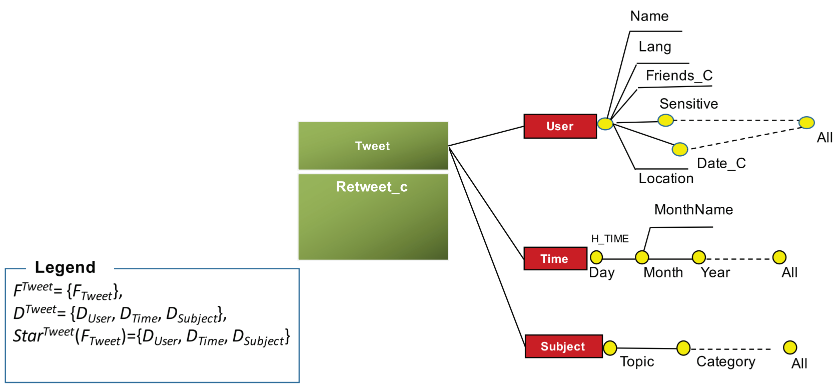
A **fact**, $F \in F^E$, is defined by (N^F, M^F) where: N^F is the name of the fact, $M^F=\{f_1(m_1),\dots,f_v(m_v)\}$ is a set of measures, each associated with an aggregation function f_i.

A **dimension**, denoted $D_i \in D^E$ (abusively noted as D), is defined by (N^D, A^D, H^D) where: N^D is the name of the dimension, $A^D=\{a_1^D,\dots,a_u^D\} \cup \{id^D, All^D\}$ is a set of dimension attributes, $H^D=\{H_1^D,\dots,D_v^D\}$ is a set hierarchies.

A **hierarchy** of the dimension D, denoted $H_i \in H^D$, is defined by (N^{Hi}, $Param^{Hi}$, $Weak^{Hi}$) where: N^{Hi} is the name of the hierarchy; $\mathrm{Param}^{\mathrm{H_i}} =< \mathrm{id}^{\mathrm{D}}, \mathrm{p}_1^{\mathrm{H_i}}, \dots, \mathrm{p}_{\mathrm{v_i}}^{\mathrm{H_i}}, \mathrm{All}^{\mathrm{D}} >$is an ordered set of v_i+2 attributes which are called **parameters** of the relevant graduation scale of the hierarchy, $\forall k \in [1..v_i]$, $p_k^{H_i} \in A^D$; $Weak^{Hi}$: $Param^{Hi} \rightarrow 2^{A^D - Param^{H_i}}$ is a function associating with each parameter possibly one or more weak attributes.

An **OLAP cuboid** O is derived from E, $O = (F^O,D^O)$ such that: F^O is a fact derived from F ($F \in F^E$) with a subset of measures, $M^O \subseteq M^F$; $D^O \subseteq 2^{Star^E(F)} \subseteq D^E$ is a subset of dimensions of D^E. More precisely, D^O is one of the combinations of the dimensions associated to the fact F ($Star^E(F)$).

If we generate OLAP cuboids using all dimension combinations of one fact, we have an OLAP cuboid lattice (also called a pre-computed aggregate lattice or cube).



Name
Lang
Friends_C
Sensitive
User
All
Tweet
Retweet_c
Date_C
Location
MonthName
H_TIME
Time
Day
Month
Year
All
Subject
Topic
Category
All
Legend
$F^{Tweet}=\{F_{Tweet}\}$,
$D^{Tweet}=\{D_{User}, D_{Time}, D_{Subject}\}$,
$Star^{Tweet}(F_{Tweet})=\{D_{User}, D_{Time}, D_{Subject}\}$

Fig 1 A multidimensional conceptual schema allowing the analysis of Tweets

2.1 Complex hierarchies

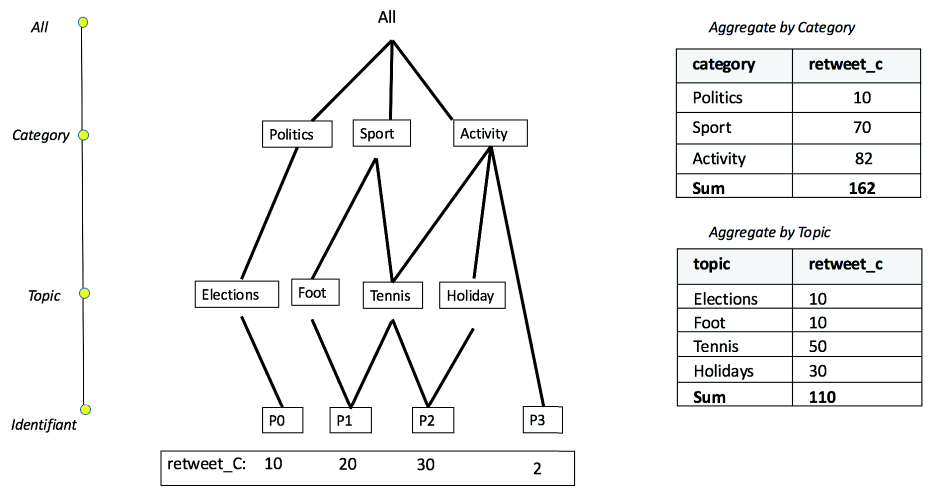
In the real world, it is often the case when hierarchies are irregular. We say that the hierarchy is complex when it is a non-strict hierarchy and/or a non-covering hierarchy [6]. We will illustrate and define the above.

In figure 2, we show an example of complex hierarchy. The example is taken from an OLAP application on Twitter. The *subject* is one of the analysis dimensions and its attributes form a hierarchy *id-topic-category-all*. We can see that the tweet "*P1*" has two topics "*Foot*" and "*Tennis*"; the topic "*Tennis*" falls within two categories "*Sport*" and "*Activity*". This corresponds to a *many-to-many* relationship on *tweet-topic* and *topic-category*. This is called non-strict hierarchy.
The tweet P3 has no topic, but it falls within the category "*Activity*". This corresponds to a *one-to-any* relationship *([1..0-*])* on *tweet-topic*. This is called non-covering hierarchy. Now, we can define:

- A hierarchy is said to be non-strict when a child of a given level can have more than a parent of the superior level [11,15].
- A hierarchy is said to be non-covering if a dimension value can have no direct upper parent [11,15].

The complex hierarchies cause summarizability issues [13, 10] i.e. it is not easy to perform drill-down and roll-up analysis on data, because of potential missing or redundant information. One element can be considered several times or none when computing a pre-aggregate (for example the sum of measures by category when a product appears in multiple categories).

Let us illustrate the summarizability issues with our example from figure 2. If we count re-tweets by topic (Figure 2), we obtain a total of 110 while the exact total is 62. The tweets *P1* and *P2* have been counted several times (twice each) which distorts the calculation of aggregates. If we wish to have the amount of re-tweets by category for the aggregate results at the level of topics (Figure 2), we obtain a result of 162 in place of 62. The topic *Tennis* is attached to two categories. Furthermore, the erroneous aggregate results at the level of topics are reflected in the superior hierarchical levels.



Aggregate by Category

category	retweet_c
Politics	10
Sport	70
Activity	82
Sum	**162**

Aggregate by Topic

topic	retweet_c
Elections	10
Foot	10
Tennis	50
Holidays	30
Sum	**110**

Fig 2 Example of non-strict and non-covering hierarchy and summarizability issues

3 Complex hierarchies and document-oriented systems

3.1 Document-oriented data model formalism

Document-oriented systems store documents in collections and are key-value stores. A unique key identifies every document (the value) that will be called identifier. The document is itself a set of key-value pairs. Keys define the structure of the document and act as meta-data. Each value can be an atomic value (number, string, date…), a sub-document or an array. Documents within documents are called sub-documents or nested documents. We distinguish the document instance from the document structure/schema. The document structure/schema corresponds to a generic document without atomic values i.e. only keys. A document instance belongs to a collection *C* and has an identifier, *id*. We refer to this document as *C*(*id*). We use the following symbols: ":" separates keys from values, "[ ]" denotes arrays, "{ }" denotes documents and a comma "," is used to separate key-value pairs from each other. Using this notation, we provide an example of a document instance:

```
User (30001): {

      name: "John Smith",
      addresses: [{city: "London", country: "UK"},
            {city: "Paris", country: "France"}],
      phone: {prefix: "0033", number: "61234567"}}
```

This example document belongs to the “User” collection, it has 30001 as identifier and it contains keys such as “name”, “addresses”, “phone”. The addresses value is an array of sub documents and the phone value is a sub-document.

3.2 Mapping the multidimensional model and complex hierarchies

The formalism that we have defined earlier allows us to define a mapping from the conceptual multidimensional model to each of the logical models defined above.
The data model that we will propose is inspired by our previous work [3]. It takes into account that document-oriented implementations of data warehouses work better with flat models i.e. one fact and its dimensions are stored in one collection. This is different from RDBMS where we normalize data and we have one table for the fact and one table per dimension.

Our mapping can be explained in two steps:

(i) For a given fact, all dimension attributes are nested under the respective attribute name and all measures are nested in a subdocument with key “measures”. This model is inspired from our work. This corresponds to the following mapping:

- Each conceptual star schema (one F^i and their dimensions $Star^E(F^i)$) is translated in a collection C.
- The fact F_i is translated in a compound attribute Att^{CF}. Each measure m_i is translated into a simple attribute Att^{SM}.
- Each dimension $D_i \in Star^E(F^i)$ is converted into a compound attribute Att^{CD} (i.e. a nested document). Each attribute $A_i \in A^D$ (parameters and weak attributes) of the dimension D_i is converted into a simple attribute Att^A contained in Att^{CD}.

(ii) For attributes within complex hierarchies, we use arrays. There are three cases: In this case, the attribute can have no values (non-strict hierarchy)

- The attribute value has no value i.e. non-covering hierarchy
- The attribute value has one value i.e. normal behavior
- The attribute value has many values i.e. non-strict hierarchy

Below, there is an example from the Twitter case study. A combination of fact and dimensions will be stored in one document that looks as the following:

```
{
"User": {
        "user_id": "1704005545",
        "user_screen_name": "ann2thingelse",
        "user_friends_count": "150",
        "user_utc_offset": "28800",
        "user_time_zone": "Irkutsk",
        "user_created_at": "Tue Aug 27 07:16:41 +0000 2013",
        "user_lang": "ko",
        "user_location": ""
        },
"Time": {
        "id": "619883842770370560",
        "created": "Sat Jul 11 14:59:59 +0000 2015",
        "timestamp": "1436626799658",
        "day": "11",
        "month": "6",
        "year": "2015"
        },
"Subject": [{"topic": "football", "category":[ "football", "Activity]"},
               {"topic": "senat", "category": ["Politycs"]}],
Fact": {
       "Retweet_c": "15"
       }
}
```

3.3 Algorithm for managing complex hierarchies

In this section, we propose an algorithm that can deal with non-strict and non-covering hierarchies.

Let *C* be a collection corresponding to an OLAP cuboids or detailed data. We will interest to one dimension *d* and a potentially complex hierarchy *H*. The data in *C* is described at some level of granularity; we suppose the lowest level of granularity corresponds to some attribute *a.* Our goal is to group data on another dimension attribute from *H* that stands higher in the hierarchy, say attribute *b.*
Furing aggregation, we suppose we want to apply *sum(m)* an aggregation function on one measure *m*.

We suppose data is modelled with the mapping we have defined earlier i.e. dimension attribute values within complex hierarchies will be stored with arrays.
To preserve summarizability, we propose on the data model we have proposed the following:

Non-strict hierarchies resolution: The problem with non-strict hierarchies is that we aggregate measures multiple times when we have multiple values in the groub_by dimension attribute. To deal with this issue we propose the use of two variables/fields:

– The ***real value***, which will be displayed for analysis. The real aggregation value is obtained, while aggregating all the measures *m* from the parent attributes of a^H in b^{H}. This value is calculated without taking into account the number of parents for each child attribute.

– The ***aggregate value***: which, it, will be used uniquely for calculating the superior hierarchical level. The ***aggregate value*** is calculated differently. For each attribute *a*, the algorithm calculates the number of parents it has in b^H that we call *parents(a)*. If the child attribute has a single parent ($|parents(a)|=1$) the measure will be aggregated one time. If the attribute has several parents ($|parents(a)|>1$), the algorithm will count the number of parents *P* (the number of elements in the array) then add the measure aggregated value *sum(m)* will be divided by the number of parents $|parents(a)|$. In this way the measure will not be aggregated as many times as that of the parents.

Non-covering hierarchies: For treating non-covering hierarchies, we use classical approach, that regroup all the orphan values in an artificial value called *others*. For example, for an aggregation hierarchical level b^H a *others* value is created and contains all the orphan values of the hierarchical level a^H. This solution is used already in the relational model [7, 8].

Algorithm SCHS: Algorithm pseudocode for aggregating data (summing) on a measure groupig by a dimension attribute of potential complex hiearchy

```
Input: C // Collection of documents to a cuboid of dat
Param : Prameter used for aggregation
For doc in C do
        If doc.b= ∅ then
                doc.b ← NewParam(other)
                SumAgg[doc.b]+=doc.m
                SumReal[doc.b]+=doc.m
        Else
                For v in doc.b
                        $SumAgg[v] += \frac{doc.m}{|doc.param|}$
                        $SumReal[v] += doc.m$
        Endif
End
```

4 *Experiments*

4.1 Experimental setup

We propose two sets of experiments.

The first set is about instantiating a data warehouse with the data model we proposed. We use for this purpose data from the Twitter case study. We load data and we study performance on a set of OLAP queries.

The second set of experiments is about validating our algorithm for data summarization with complex hierarchies. We also compare our approach to two approaches from state-of-the-art namely:

- The approach of Pederson and al [12]: an approach that is considered as a reference approach for the summarizability issues
- The approach of Hachicha and al [6]; that also uses a correction strategy when aggregating.

These two approaches are meant for the relational model; we have adapted them for document-oriented systems.

Hardware: The experiments are done on a cluster composed of 6 PCs, (4 core-i5, 8GB RAM, 2TB disks, 1Gb/s network), each being a worker node and one node acts as dispatcher.

Dataset: The data is obtained with the Twitter API for data streaming. Tweets are returned in JSON data format with each tweet having 67 data fields. We process tweets to follow the data model we have defined earlier. We also add a dimension called *subject* that has as attributes *topic* and *category*. These extra data is fictional and we introduce here arbitrarily non-strict hierarchy issues and non-covering hierarchy issues.

Queries: We test our approach to implemente the conceptuel model to logical model, on 3 query sets. Three query sets are created with 3 queries per set. The query complexity increases from Q1 to Q3. Q1 involves 1 dimension, Q2 involves 2 dimensions and Q3 involves 3 dimensions.

4.2 Experimental results: Data warehouse instantiation and validation

In the first set of experiments, we have concentrated in transforming and loading data into MongoDB with the pre-defined model of data.
After loading data, we focus on interrogation. In the following table, we show query execution times on 9 queries on 5 different settings: 1 shard, 2 shards, 3 shards, 4, shards, 5 shards. We can observe that augmenting the number of shards reduces the query time. This is easy to explain. The query is executed in parallel across shards.

Table 1 Query execution times at different configuration with 400 millions documents, in seconds

#shards/query	*1 shard*	*2 shards*	*3 shards*	*4 shards*	*5 shards*
Q1.1	1070	1042	824	598	497
Q1.2	702	658	433	402	326
Q1.3	697	655	433	408	324
Q2.1	687	656	433	351	286
Q2.2	687	656	433	351	286
Q2.3	687	656	433	352	285
Q3.1	695	676	433	360	285
Q3.2	693	675	433	352	285
Q3.3	693	676	432	353	285

4.3 Experimental results 2: Data summarization with complex hierarchies

In this section, we show results on data summarization (aggregation) using algorithms that fix summarizability issues on complex hierarchies. We compare our approach to the approaches of Hachicha and Pedersen. Results are shown in Table 2 and Table 3. We use two different settings. In the first setting, we consider one configuration server and one data shard (Table 2). In the second setting, we consider one configuration server and 5 data shards (Table 3).

We show in the tables, the execution time to compute a pre-aggregate (OLAP cuboid) on given dimension combinations. We build cuboids on top of each other i.e. we will compute a cuboid from another existing cuboid that is closer to its granularity of data.

We observe the following results. In the average case, our approach works faster than the other approaches from state-of-the-art. We also observe that it is faster to compute top-level cuboids i.e. cuboids that group on few dimensions and top-level attributes. This is easy to explain, because there is less data. In this case, our approach performance is comparable with state-of-the-art approaches.
The above observations are true on both settings: single shard and multiple shards. We can confirm once again that sharding makes computation faster.

Table 2 Cuboids computation times (in seconds) compared on different approaches on single shard setting with 400 millions documents

Aggregate	**Pedersen**	**Hachicha**	**SCHC**
topic-day-location	*21070*	*19070*	*18892*
topic-month	*12067*	*12067*	*11857*
category-month-location	*167*	*63*	*64*
year-category	*109*	*48*	*51*
avg	***33143***	***31248***	***30864***

Table 3 Cuboids computation times (in seconds) compared on different approaches on 5 shards setting with 400 millions documents

Aggregate	**Pedersen**	**Hachicha**	**SCHC**
topic-day-location	*903*	*808*	*604*
topic-month	*597*	*534*	*486*
category-month-location	*36*	*23*	*12*
year-category	*34*	*15*	*10*
avg	***1570***	***1308***	***1112***

5 Related Work

In 1997, the summarizability has studied for the first time on multidimensional data by Lenz and Shoshani [10]. Since then, three approaches for treating the complex hierarchies have been proposed.

The first approach involves schema normalization. In this solution, two solutions are proposed. For the first, the authors propose to resolve the problem at the conceptual level while defining the rules of constraint and of implementation of the conceptual model towards the logic model [7]. In the second solution of normalization, the principle is to separate the correct hierarchies from the hierarchies susceptible to cause aggregate calculation errors. In this context, [11] propose to put the non-strict hierarchies in the new tables, called joint tables also called separated tables by Malinowski and Zimanyi [11]. In 2008, Mazon and al, proposed a conceptual model normalized UML, separating the different associations [13].

In the second approach, data is transformed for treatment of the complex hierarchies. This approach requires the modification of the fact-dimension instances. Pederson and al were the first to propose a solution for this perspective [15] . Three algorithms were thus proposed, Makecover which is responsible for making the covered data. Makestrict, is responsible for transforming the multiple hierarchies to the simple hierarchies. For each element having multiple parents, a parent composed from the fusion of its parents is created and inserted between the two. The last algorithm Makeonto is used to manage onto hierarchies [11] In similar work based on the solution from Pederson, Mansmann and Scholl [10] present a visual tool OLAP which allows for normalizing, browsing and visualizing the different levels of a hierarchy . In their graphic structure, each level of the hierarchy is modeled by a directory.

The third solution has to detect the non-strict hierarchies and non-covering hierarchies, and resolve them at the moment of aggregate computation. These solutions are often accompanied by implementation of operators. In 2005, Horner and Song [7] suggest a script to detect the measures already computed but without ever implementing them. Hachicha and Darmont [6], consider the managing of the summarazibility issues in the documents XML and propose a projection operator which returns a zero result in the case of non-strict hierarchies. In a similar way, Hachicha and Darmont, while drawing from the work of Pederson, propose an operator which operates in multidimensional data XML, by grouping the parents of an element for a single hybrid parent.

The representation of complex hierarchies in conventional relational DBMS turns out to be very complicated, even more so with the explosion of massive data, the bases of relational data shows the benefits of difficulty in management of such massive data. This is why, in this article, we take an interest in a new solution, the systems NoSQL, which seems to respond to the problem of massive data [19] in particular the system document-oriented.

6 Conclusions

In this paper, we have studied complex hierarchies and summarization issues in the context of document-oriented implementations of data warehouses.

First, we have proposed a set of rules to automatically translate the conceptual multidimensional schema at the level of logic NoSQL document oriented systems. Furthermore we have conducted a set of experiments to study the loading processes and interrogation. Then, we have tested our approach on datasets with Twitter tweets. Different volumes have been used. We have used MongoDB as the data base NoSQL. The first results show that our approach offers the best results and the best analysis.

As for future work we hope to conduct investigations at the level of columns oriented models. These two latter use the versioned values a very interesting point for updating data warehouses.

7 References

1. Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, Ronan Tournier. Implementation of multidimensional databases in column-oriented NoSQL systems (ADBIS 2015), Springer, p. 79-91
2. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R., Implementing Multidimensional Data Warehouses into NoSQL, 18th Int. Conf. on Enterprise Information Systems (ICEIS 2016) .
3. Max Chevalier, Mohammed El Malki, Arlind Kopliku, Olivier Teste, Ronan Tournier. Implementing Multidimensional Data Warehouses into NoSQL (ICEIS 2015).
4. G. Colliat. OLAP, relational, and multidimensional database systems. SIGMOD Record 25(3), ACM, pp. 64.69, 1996.
5. Dede, E., Govindaraju, M., Gunter, D., Canon, R. S., Ramakrishnan, L., 2013. Performance evaluation of a mongodb and hadoop platform for scientific data analysis. 4th Workshop on Scientific Cloud Computing, ACM, pp. 13–20.
6. Hachicha, M., C. Kit, et J. Darmont (2012). A Novel Query-Based Approach for Addressing Summarizability Issues in XOLAP. In *COMAD'12, Pune, India*, pp. 56–67. CSI.
7. Horner, J., I.-Y. Song, et P. P. Chen (2004). An Analysis of Additivity in OLAP Systems. In *DOLAP'04, Washington, DC, USA*, pp. 83–91. ACM.
8. Hurtado, C. A., C. Gutiérrez, et A. O. Mendelzon (2005). Capturing Summarizability with Integrity Constraints in OLAP. *ACM Trans. Database Syst. 30*(3), 854–886.
9. R. Kimball and M. Ross. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. John Wiley & Sons, 3rd ed., 2013.
10. Lenz, H.-J. et A. Shoshani (1997). Summarizability in OLAP and Statistical Data Bases. In *SSDBM'97, Olympia, Washington, USA*, pp. 132–143. IEEE Computer Society.
11. Malinowski, E. et E. Zimányi (2006). Hierarchies in a multidimensional model : From conceptual modeling to logical representation. *Data & Knowledge Engineering 59*(2), 348–377.
12. Mansmann, S. et M. H. Scholl (2007). Empowering the OLAP Technology to Support Complex Dimension Hierarchies. Int. Journal of Data Warehousing and Mining 3(4), 31–50.
13. Mazón, J.-N., J. Lechtenbörger, et J. Trujillo (2009). A survey on summarizability issues in multidimensional modeling. *Data & Knowledge Engineering 68*(12), 1452–1469.
14. Morfonios, K., Konakas, S., Ioannidis, Y., Kotsis, N., 2007. R-OLAP implementations of the data cube. ACM Computing Survey, 39(4), p. 12.
15. Pedersen T., Jensen Ch., Dyreson C. A foundation for Capturing and Quering Complex Multidimensional Data. Information Systems, 26(5): 383-423, 2001
16. Rafanelli, M. et A. Shoshani (1990). STORM : A Statistical Object Representation Model. In *SSDBM'90, Charlotte, USA*, Volume 420 of *LNCS*. Springer.
17. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Graphical Querying of Multidimensional Databases. In: 11th East-European Conf. on Advances in Databases and Information Systems (ADBIS), Springer-Verlag, LNCS 4690, pp.298–313 (2007).
18. Stonebraker, M., New Opportunities for New SQL, in Communications of the ACM, 55(11), ACM, pp. 10–11, 2012.
19. M. Stonebraker, S. Madden, D.J. Abadi, S. Harizopoulos, N. Hachem and P. Helland. The end of an architectural era: (it's time for a complete rewrite). *33rd Int. conf. on Very large Data Bases* (*VLDB*), ACM, pp. 1150-1160, 2007.