Building a Document-Oriented Warehouse Using NoSQL

Ines Ben Messaoud, University of Gabes, Tunisia
Abdulrahman A. Alshdadi, University of Jeddah, Saudi Arabia
Jamel Feki, University of Jeddah, Saudi Arabia

ABSTRACT

The traditional data warehousing approaches should adapt to take into consideration novel needs and data structures. In this context, NoSQL technology is progressively gaining a place in the research and industry domains. This paper proposes an approach for building a NoSQL document-oriented warehouse (DocW). This approach has two methods, namely 1) document warehouse builder and 2) NoSQL-Converter. The first method generates the DocW schema as a galaxy model whereas the second one translates the generated galaxy into a document-oriented NoSQL model. This relies on two types of rules: structure and hierarchical rules. Furthermore, in order to help understanding the textual results of analytical queries on the NoSQL-DocW, the authors define two semantic operators S-Drill-Up and S-Drill-Down to aggregate/expand the terms of query. The implementation of our proposals uses MongoDB and Talend. The experiment uses the medical collection Clef-2007 and two metrics called write request latency and read request latency to evaluate respectively the loading time and the response time to queries.

KEYWORDS

Document Warehouse, Galaxy Model, Hierarchical Rules, MangoDB, NoSQL, Semantic OLAP, Structure Rules, Transformation Rules

INTRODUCTION

Documents contain valued information and incarnate pertinent knowledge for decisional processes. Knowledge helps decision-makers interpret the results of business analyses (Inmon, 2002). In (McCabe et al., 2000; Sullivan, 2001), the authors advocate that documents should be warehoused; hence, the Document Warehouse (DocW) concept appears. A DocW is modeled as a Star schema (Tseng et al., 2006; Ben Mefteh et al., 2016), or as a Galaxy (Ben Messaoud et al., 2015; Feki et al., 2013; Pujolle et al., 2011) that is a variant of the Star schema. A DocW organizes textual data for OLAP (On-Line Analytical Analyses) analyses for successful business intelligence purposes (Tseng et al., 2006).

Over the past decade, several digital players (e.g., sensors, social networks) produce unlimited amounts of data so that the data volumes to analyze reach critical sizes (Jacobs, 2009). Nevertheless, current warehousing methodologies become obsolete to handle successfully the growing data volumes as stated in (Krish et al., 2013) and (Chevalier et al., 2015a). To overcome this drawback, NoSQL (Not-Only SQL) appear as a new technology to implement huge databases and, in particular, document warehouses. In fact, there are four types of NoSQL models: key-value, column-oriented,
document-oriented and graph-oriented models. This paper is interested in the document-oriented NoSQL model that offers performance, horizontal scalability options and is convenient to handle complex data structures like nested objects. Furthermore, NoSQL is very suitable for data integration and schema migration (Hecht et al., 2011).

Our objective, in this paper, is to store, manage and query big volume of documents, and better serve decision-makers whose requirements in the analysis of document contents are beyond the classical needs of Data Warehouse (DW) users. More precisely, we aim to improve the performance of the DocW in terms of the response time of queries; as well as the semantic analysis. To do so, we suggest, an approach for the design and implementation of a NoSQL-DocW loaded from heterogeneous XML documents, along with semantic operators.

This approach has two methods called Document Warehouse Builder and NoSQL-Converter. The first accepts XML documents and produces a DocW as a galaxy (Feki et al., 2013; Ben Messaoud et al., 2015). The second method transforms the obtained galaxy into a Document-oriented NoSQL model. We develop two semantic OLAP operators S-Drill-Up and S-Drill-Down to query a NoSQL-DocW.

To implement our proposal, we use MongoDB as a document-oriented NoSQL system. We measure the performance in terms of two metrics, the Write Request Latency (WRL) assesses the data loading time, and the Read Request Latency (RRL) evaluates the querying time. We conduct experiments on the medical collection Clef-2007.

We organize this paper as follows: Section 2 presents the related works in NoSQL warehouses. Section 3 presents our approach for building a DocW modelled as a galaxy. Section 4 details our rules for transforming a galaxy model into a NoSQL-DocW. Section 5 suggests semantic operators for the NoSQL-DocW. Section 6 experiments and evaluates the results. Finally, Section 7 concludes the paper and addresses ongoing work.

RELATED WORKS

Warehousing allows big data management and analysis; NoSQL offers interesting features to implement Data/Document warehouses (Chevalier et al., 2015a). Next, we review relevant works related to NoSQL warehouses.

In (Li, 2010) the author proposed a two-phase approach to transforming a relational database (RDB) into the column-oriented NoSQL HBase. First, they transform the relational schema into HBase schema; secondly, they express the relationships between the source and target schemas by mappings. Nevertheless, this approach applies on the conceptual schema only.

Similarly, the authors of (Freitas et al., 2016) suggested the R2NoSQL approach, which defines conceptual mappings to convert the concepts of a RDB into NoSQL.

The authors of (Dehdouh et al., 2014) presented a benchmark for columnar NoSQL-DW but without formalizing the modelling process. Later, in (Dehdouh et al., 2015), they proposed three approaches to implement the DW using a column-oriented NoSQL model. The first approach NLA (Normalized Logical Approach) uses distinct tables to store facts and dimensions, and uses simple attribute to map measures and dimensional-attributes. The second approach DLA (Denormalized Logical Approach) stores the fact and dimensions together within one table, and uses a simple attribute to map measures and dimensional-attributes. Finally, the third approach DLA-CF (Denormalized Logical Approach by using Column Family) stores the fact and dimensions together within one table, a composite attribute maps measures and dimensional-attributes. However, the NLA approach is inefficient with regard to join-queries. The DLA-CF approach is better than the DLA when the query-attributes belong to the same dimension.

In the same context, the authors of (Chevalier et al., 2015a) translate the star model of the DW into two NoSQL models namely Column-oriented and Document-oriented. Later, in (Chevalier et al., 2015b), they improve their proposal using the concepts table, column-family and column of the column-oriented NoSQL model, to map the star model into a column-oriented NoSQL model. They
transform the star as follows: one table where the fact is a column-family and each measure is a column; and every dimension becomes a column-family where each dimensional-attribute is a column.

In order to adapt the star model for a collection of documents, they use, in (Chevalier et al., 2015c), the concepts of the NoSQL document-oriented model; i.e., document, composite attribute and simple attribute. Indeed, the fact becomes a composite attribute and each measure a simple attribute. A dimension is transformed into a composite attribute (nested document) and each parameter and weak attribute becomes a simple attribute. Nevertheless, their rules neglect to convert the hierarchies of dimensions. This lack of hierarchy in the result NoSQL-DW is a real handicap because hierarchies are crucial for the drilling operators.

To alleviate this problem, the authors of (Yangui et al., 2016) propose two approaches: one for a column-oriented NoSQL-DW and one for a document-oriented NoSQL-DW. For each approach, they distinguish two types of transformations: simple and hierarchical. The simple transformation converts the Star model of the DW into a NoSQL model without hierarchies; it uses column-family/collection to store measures and dimensions. The hierarchical transformation treats the hierarchies in the NoSQL model by using the super-column/document concepts. Nevertheless, these two approaches are experimented on a small set of requirements, which does not help to assess the benefits/drawbacks seriously of the NoSQL-DW. Note that the authors do not suggest dedicated OLAP operators.

The authors of (Sellami et al., 2018) have taken into consideration the transformation of hierarchies into a graph-oriented NoSQL model by using the concepts of node, relation, label and property. The main lack is the authors experiment neither their transformation rules nor evaluate the implemented graph-oriented NoSQL-DW.

Regarding the DocW field, the authors of (Ben Messaoud et al., 2017) and (Ben Messaoud et al., 2018) propose an approach to implement a NoSQL-DocW. They transform the galaxy model of the DocW into a column-oriented and document-oriented NoSQL model. Once again, this work distinguishes two types of transformations simple and hierarchical.

Table 1 summarizes our literature review, based on the following six criteria.

C1: Transform a DB into NoSQL.
C2: Transform a DW into NoSQL.
C3: Transform the DocW into NoSQL.
C4: Adopt NoSQL.
C5: Propose transformation rules towards NoSQL.
C6: Experiment/Evaluate the proposed rules.

So far, we have given an overview of pertinent work regarding NoSQL warehouses. These works convert the DW/DocW model into NoSQL. Note that major works bypass hierarchy that are crucial for the drilling operators. In addition, most existing approaches were interested in transforming the numeric DW into NoSQL against few works interested in transforming the DocW into NoSQL.

Section 3 suggests a novel approach that aims to build a NoSQL-DocW from XML documents while paying particular attention to the hierarchy concept that is fundamental for OLAPing.

**APPROACH FOR BUILDING A DOCUMENT-ORIENTED NOSQL WAREHOUSE**

DW methodologies are for numeric data (e.g., sales activity), they are inappropriate for new data structures and requirements. Indeed, they are inefficient for building/handling huge data volumes (Krish, et al., 2013; Chevalier et al., 2015a).

The advent of the new NoSQL technology, has allowed the DW community to store, manage and query big data, along with better performance and horizontal scalability features. In fact, horizontal scalability enables to increase the hardware resources by adding application server instances (Lemberger et al., 2015; Chandawni, 2016). Considering these advantages, we are interested in this technology for warehousing XML documents.

We present an approach for building a document-oriented NoSQL-DocW from a collection of XML documents. Figure 1 depicts the two methods of this approach called *Document Warehouse Builder* and *NoSQL-Converter*.

The *Document Warehouse Builder* accepts a set of XML documents in the same domain with their structures (DTD and/or XSD) and then produces a galaxy model. Firstly, it generates a limited number of unified structures to be validated; it treats both acronym and synonym ambiguities of structures nodes, referring respectively to a dictionary of acronyms and the WordNet lexical resource.

The unified structures are then validated according to the analytical requirements. Secondly, we translate each unified structure into a galaxy model by using ten identification rules for dimensions, attributes, etc. Finally, the output galaxies are syntactically validated referring to eleven quality-design constraints dedicated to the multidimensional galaxy model. The *Document Warehouse Builder* is detailed in (Feki et al., 2013) and (Ben Messaoud et al., 2015) and illustrated with examples.

**Figure 1. Approach for building a document-oriented NoSQL Warehouse**
The NoSQL-Converter, the first contribution in this paper, converts each galaxy into a document-oriented NoSQL model as detailed in the following section.

**NOSQL-CONVERTER**

NoSQL-Converter produces a document-oriented NoSQL Warehouse by applying six rules (Section 4.3) on the galaxy resulting from the Document Warehouse Builder; the resulted NoSQL-DocW differs from the literature works by considering the hierarchies of dimensions.

For this conversion, we introduce the galaxy model of the DocW, and then the document-oriented NoSQL model. Thereafter, we define the transformation rules.

**The Galaxy Model**

In (Pujolle et al., 2011), the authors suggested the galaxy model for DocWs (Figure 3-a), it inherits the concepts of the multidimensional model (Inmon, 2002; Kimball et al., 2013) and pays special attention to the dimension concept. We formalize its concepts.

**Galaxy Model:** A galaxy model is a set of dimensions connected by one node or more. In a galaxy, the conventional Fact concept is voluntary hidden. Formally, a galaxy model $G$ is defined by the triplet $(G^Na, G^D, G^No)$ where:

- $G^Na$: name of the galaxy;
- $G^D = \{D_1, \ldots, D_n\}$: set of $n$ ($n \geq 2$) dimensions;
- $G^No = \{N_1, \ldots, N_m\}$: set of $m$ ($m \geq 1$) nodes.

**Dimension:** A dimension is composed of a set of attributes called parameters and organized into hierarchies. Every parameter may have labels as weak attribute(s) (i.e., descriptive attribute(s) like the Product-name associated with parameter Product-code). A dimension $D$ is defined by the triplet $(D^Na, D^P, D^H)$ where:

- $D^Na$: name of dimension $D$;
- $D^P$: non-empty set of parameters and weak attributes of $D$;
- $D^H$: non-empty set of hierarchies of $D$.

**Hierarchy:** A hierarchy arranges semantically its parameters in several levels from the finest to the highest granularity. Formally, it is defined by the triplet $(H^Na, H^P, P^WA)$ where:

- $H^Na$: hierarchy name;
- $H^P = \{P_1, \ldots, P_q\}$: set of $q$ ($q \geq 1$) parameters of hierarchy $H^Na$;
- $P^WA$: function, associates each parameter with its weak attributes.

**Node:** A node links compatible dimensions, i.e., dimensions used together significantly within the same analytical query or set of queries. Formally, a node is a couple $(N^Na, N^D)$ where:

- $N^Na$: node name;
- $N^D$: function, links each dimension to its nodes. Naturally, a dimension can participate in several nodes but nodes cannot be directly linked together. In other terms, given a galaxy $G (G^Na, G^D, G^No)$ \forall N_i \perp N_j \perp G^No then \exists a link $(N_i, N_j)$ ($i \neq j$).

The galaxy in Figure 3-a models research articles with five dimensions ($D$-Article, $D$-Author…) connected via a single node, along with their hierarchies. For instance, the $D$-Conference dimension has one hierarchy with three parameters $Id$-$D$-Conference, $P$-Series (conference short name) and $P$-Audience (National or International). $Id$-$D$-Conference has three weak attributes (WA-Acceptance-rate, WA-Editor, and WA-Name).
The Document-Oriented NoSQL Model

The document-oriented NoSQL model stores collections of documents (Figure 2); its three basic concepts are collection, document, and attribute formalized hereafter.

Collection: A collection $C$ is a set of stored data; it is defined by the couple $(C^{Na}, C^0)$ where:
- $C^{Na}$: name of the collection $C$;
- $C^0 = \{D_1, \ldots, D_n\}$: set of $n$ documents.

Document: A document represents the main storage unit of the document-oriented NoSQL model.

It consists of a set of key/value pairs (i.e., attributes). Formally, a document is defined by the couple $(Doc^{Na}, Doc^{KV})$ where:
- $Doc^{Na}$: document name;
- $Doc^{KV} = \{A_1, \ldots, A_m\}$: set of attributes, i.e., pairs of key/value.

Attribute: An attribute is the main component of a document. It can be simple when its value is atomic (e.g., integer, string), or composed when it is a document. A simple attribute is defined by $(SA^{Na}, SA^V)$ where:
- $SA^{Na}$: name of the simple attribute;
- $SA^V$: value of the simple attribute.

Whereas, a composed attribute is defined by $(CA^{Na}, CA^V)$:
- $CA^{Na}$: name of the composed attribute;
- $CA^V = \{D_1, \ldots, D_j\}$: set of documents.

From Galaxy to NoSQL

In order to implement the NoSQL-DocW, we define two categories of transformation rules: Structure transformation rules (structure rules for short), and Hierarchy transformation rules (hierarchy rules for short).

The structure rules transform the conventional structure (i.e., schema) into a document-oriented NoSQL model as a collection of documents.

The hierarchy rules stores data of the galaxy into a document-oriented NoSQL model, as an additional separated set of documents. Note that these rules take into consideration the hierarchies, as their presence in the result model is fundamental for analyses.

Structure Transformation Rules

For transforming the galaxy into a NoSQL-DocW (Document-oriented Warehouse), we define three Structure transformation rules on a Galaxy $G$.

Rule SR1: Transform the structure of $G$ into a NoSQL-DocW as a collection of documents containing as many NoSQL documents as dimensions and nodes in $G$:
- The collection name is $C^{Na}$ (same name as the galaxy $G^{Na}$), prefixed with “Structure” (e.g., the structure of the galaxy model Research-Paper (Figure 3-a) is a collection named Structure-Research-Paper (Figure 3-b); and
- The documents of the collection are the nodes and the dimensions of $G^{Na}$.

Rule SR2: Transform each dimension $D$ of $G$ into a document $Doc$ where:
- The document name $Doc^{Na}$ is the dimension name $D^{Na}$;
- The document identifier is a surrogate key $^1$;
- The document has many composed attributes so that:
  - Each hierarchy $(H \hat{I} D)$ becomes a composed attribute;
Each parameter \((P \in H)\) is described by three attributes: Parameter-Name, Rank in the hierarchy and Weak-Attribute-Name, where:
- The value of the Parameter-Name attribute is the name of parameter \(P\);
- The value of the Rank attribute is the position of \(P\) in its hierarchy \(H\);
- If \(P\) has a weak attribute called \(WA\), the value of Weak-Attribute-Name is \(WA\); otherwise, Weak-Attribute-Name is ‘Null’.

**Rule SR3:** Transform each node \(N\) in \(G\) into a document \(Doc\) such as:
- The document name \(Doc^{\text{Na}}\) is the name of the node \(N^{\text{Na}}\);
- The identifier of \(Doc\) is a surrogate key;
- The attributes of \(Doc\) are the names of the dimensions linked to node \(N\).

Figure 3-a illustrates the galaxy Research-Paper and Figure 3-b is its NoSQL-DocW result of applying rules SR1 to SR3. The result is six documents: The document Node1 describes the node of the galaxy, and the five documents D-Article, D-Author, D-Date, D-Keyword and D-Conference model the five dimensions. For instance, D-Author has two composed attributes (H1-D-Author and H2-D-Author representing the two hierarchies), each of these composed attributes defines the parameter and weak attribute of its hierarchy.

**Hierarchy Transformation Rules**

*Hierarchy rules* completes the NoSQL-DocW with the galaxy hierarchies by applying three rules TR1 to TR3 defined in (Ben Messaoud et al., 2018).

**Rule TR1:** Transform the galaxy \(G^{\text{Na}}\) into NoSQL document-oriented database as a collection of documents such as:
The collection name is $C^Na$ (same name as the galaxy $G^Na$); and
- The documents in the collection are the nodes and the dimensions of $G^Na$.

Obviously, the transformation process expands to the concepts dimension and node. Figure 4-a results from applying rule TR1 on the galaxy (Figure 4, left) transformed into six documents: $D$-Article, $D$-Author.

**Rule TR2:** Transform each dimension $D$ of the galaxy into a Dimension-document where:
- The name of the Dimension-document is denoted $Doc^DNa$, it is the dimension name $D^Na$;
- The identifier of the Dimension-document is the identifier of $D$;
- The Dimension-document has as many composed attribute(s) as hierarchies in $D$, such as:
  - The name of each composed attribute is the name of its hierarchy, and its values are the parameters and weak attributes of its hierarchy.

Clearly, a collection is a set of documents and a document contains a set of simple and/or composed attributes. Moreover, a galaxy model is a set of dimensions described by parameters and weak attributes organized into hierarchies. Therefore, each dimension turns into a document where each hierarchy becomes a composed attribute.

Figure 4-b is the transformation of the $D$-Author dimension into a Dimension-document. The Dimension-document $D$-Author (Figure 4-b, right) depicts the transformation (rule TR2) of the two hierarchies $H1$-$D$-Author and $H2$-$D$-Author, each hierarchy becomes a composite attribute as for $H1$-$D$-Author transformed into one composite attribute having three simple attributes: $Id$-$D$-Author, WA$-$Author$-$Name, and $P$-Affiliation.

**Rule TR3:** Transform each node $N$ in the galaxy $G$ into a Node-document $Doc$:
- The name of the Node-document is $Doc^NNa$; it is the name of node $N^Na$;
- The identifier of $Doc$ is a surrogate key;
- The attributes of $Doc$ are the identifiers of the dimensions linked by node $N$.

In a galaxy, each node links several dimensions. In the NoSQL-DocW, a collection contains a set of documents and a document is the principal concept of this model. Subsequently, each node in the galaxy turns into a Node-document which attributes are the identifiers of all dimensions linked to the node.

Figure 4-c depicts the result of TR3. $Node1$ became a Node-document with six attributes ($Id$-$Node1$, $Id$-$D$-Author, $Id$-$D$-Article...).

The proposed rules allow transforming the DocW galaxy into a document-oriented NoSQL. In practice, we can implement the NoSQL-DocW as two collections of documents: structure and data collections.

Note that the resulting NoSQL-DocW is textual-data centric; therefore, the conventional numeric OLAP operators are no longer applicable in this context. This motivated us to identify an urgent need to define new OLAP operators appropriate for aggregating textual data. The next section introduces the second contribution of this paper; due to space limitation, it defines two OLAP operators only.

**SEMANTIC OLAP OPERATORS FOR NOSQL-DOCW**

Few OLAP operators have been studied in the NoSQL-DW context, and even less in NoSQL-DocW. In (Dehdouh, 2016), the authors propose the MC-Cube (MapReduce Columnar Cube) aggregation...
Figure 3. (a) A galaxy and (b) its NoSQL transformation

(a) Galaxy Model: Research-Paper

(b) Collection: Structure-Research-Paper

Figure 4. (a) Example of application of rule TR1, (b) Transformation of three hierarchies of the D-Author dimension, and (c) Example of application of rule TR3
to compute OLAP cubes from NoSQL column-oriented DW. In (Gallinucci et al., 2018; Gallinucci et al., 2019), the researchers propose a four-stage approach to query a NoSQL-DocW.

In the context of graph-oriented NoSQL, the authors in (Castelltort et al., 2014) suggest an OLAP oriented data structure and introduced OLAP queries based on the Cypher declarative language. The majority of works were interested in OLAPing numeric data; nevertheless, textual data has not been sufficiently addressed although it hides meaningful information (Tseng et al., 2006).

Textual analysis in NoSQL environment needs specific OLAP operators completely different from those used in numeric DWs. As this represents a key issue, we suggest two semantic OLAP operators \textit{S-Drill-Down} and \textit{S-Drill-Up}, which deaggregate textual terms in the cells of the result-table of an OLAP query, by referring to the business domain concepts modeled as a taxonomy; we exemplify these operators.

**Example**

Given the galaxy in Figure 4-a implemented as a NoSQL-DocW, we assume the decision-maker needs to find research areas of papers’ authors by Year of publication. First, (s)he analyzes keywords per Author and Year. This uses two specific operators to the galaxy \textit{FOCUS} and \textit{LIST} (Ravat et al., 2007) described by the following syntaxes 1 and 2:

- \textbf{FOCUS} (G^{Na}, DF^{Na}, D) \hspace{1cm} (1)

where:

- \( G^{Na} \): name of the input galaxy;
- \( DF^{Na} \): dimension to consider as a fact;
- \( D = \{ D_1 \ldots D_n \} \): non-empty set of \( n \) dimensions.

- \textbf{LIST} (T) \hspace{1cm} (2)

where:

- \( T = \{ T_1 \ldots T_n \} \): non-empty set of \( n \) terms.

In this example, we use the \textit{FOCUS} to select the analysis topic (i.e., fact) \( D\text{-Keyword} \), and two analysis axes \( D\text{-Author} \) and \( D\text{-Date} \). \textit{LIST} returns a list of keywords from the input set of terms. Expression 3 sets and prepares data for this requirement:

- \textbf{FOCUS} (G^{Research-Paper}
  (D-Keyword, H1-D-Keyword, <LIST, Id-D-Keyword (WA-Keyword)>)
  ((D-Date, H1-D-Date, <P-Year>)
  ((D-Author, H1-D-Author, <Id-D-Author (WA-Author-Name)>)))

(Table 2.A shows an extract of the multidimensional table obtained with expression 3. We voluntary reduce its content to four cells. Naturally, the complete table is difficult to examine efficiently and successfully.)
Since cells in Table 2.A are crowded with keywords, their interpretation is complicated. We define the semantic aggregation \textit{S-Drill-Up} that summarizes the terms in each cell:

\[
\text{S-Drill-Up} (MT, SR) = M_{TR} \tag{4}
\]

where:
- MT: input multidimensional table;
- SR (Semantic Resource): domain taxonomy of concepts;
- MTR: result table, same structure as MT.

\textit{S-Drill-Up} aggregates the set of terms in each cell of \( MT \) referring to \( SR \). It navigates up the taxonomy to get term(s) that summarize the terms in the current cell.

\textit{S-Drill-Up} (expression E2) applied on Table 2.A produces Table 2.B; it uses the IST-Taxonomy (Figure 5). Thus, the number of terms in cells C1, C2 in Table 2.A is reduced compared to their corresponding cells (C1’, C2’ in Table 2.B); cell C1’ shows the most generic term for terms in C1. The decision-maker finds out easily the research areas. If no generic term is found then \textit{S-Drill-Up} looks for the common ancestor of nodes in the taxonomy.

Inversely, to detail the content of cells, we define a semantic \textit{S-Drill-Down} operator that receives a multidimensional table \( MT \), a taxonomy and the level of detail to reach in the taxonomy; it expands the table cells contents:

\[
\text{S-Drill-Down} (MT, SR, DL) = M_{TR} \tag{5}
\]

where:
- MT: input multidimensional table
- SR: domain taxonomy
- \( M_{TR} \): result table, same structure as \( MT \)
- DL: level of detail to reach in \( SR \)

\textbf{Figure 5. Extract of an Information System taxonomy}
Using the IST-Taxonomy, Table 2.C displays the outcome of S-Drill-Down (expr. E3) on Table 2.B. The number of terms in cells increases giving more details about the research areas per year and author.

### S-Drill-Up Algorithm

We present the S-Drill-Up algorithm; since S-Drill-Down is its reverse, we will not detail it. S-Drill-Up requires a domain taxonomy that is a hierarchical classification structure. For instance, it cascades from broad to specific or from parent to children as stated in (Inmon & Linstedt, 2014):

\[
\text{Taxonomy (Node}_1, \ldots, \text{Node}_i, \ldots, \text{Node}_n) \tag{6}
\]

- A taxonomy is a hierarchy representing a non-empty set of distinct nodes (i.e., terms); each node has a level, one Father-node (except the root node) and may have \( n (n^3 \neq 0) \) descendant-nodes:

\[
\text{Node}_i (\text{Level}, \text{Father-node}) \tag{7}
\]

- Level: number of arcs separating the root node and Node\(_i\).
- Father-node: immediate Father-node (term) for Node\(_i\).

To aggregate, S-Drill-Up needs for every term \( t \) in a cell, the distance \( d_t \) (number of arcs) separating node \( t \) and the root node in the taxonomy. We calculate this distance using a function called \( \text{Distance (Node}_N, \text{Taxonomy}_T) \) returning an integer. Afterward, S-Drill-Up compares the distance
$d_j$ with each distance $d_i$ of terms $t_j$ and $t_i$ in the same cell. If $d_j = d_i$ (i.e., the two nodes have the same depth in the taxonomy) and their corresponding nodes have the same father-node $N_f$, then $N_f$ is their aggregate term. Else ($d_j \neq d_i$), if the father nodes of terms $t_i$ and $t_j$ are linked by a father-relationship then the father node between them is the aggregate term. Otherwise (fathers of $t_i$ and $t_j$ have no link), the two terms cannot be aggregated, subsequently, the algorithm retains the two terms. We use a function named $\text{Father (Node } N, \text{ Taxonomy } T)$ to determine the immediate father node of node $N$ in $T$ taxonomy. The $S$-$\text{Drill-Up}$ function implements the $S$-$\text{Drill-Up}$ operator.

Function $S$-$\text{Drill-Up}$ (Table $MT$, Taxonomy $T$) Return Table

Input
$MT$: Multidimensional Table, each cell has $n$ ($n>0$) terms
$T$: Taxonomy

Output
$MTR$: Multidimensional Table, same structure as $MT$

Begin
$MTR = \text{New Table}$ //Empty multidim. Table, same size as $MT$

For each Cell$_i$ in $MT$ do
Term$_{\text{Input}}$={Terms $\in$ Cell$_i$}
Term$_{\text{Output}}$={}

For each Term$_j$ in Cell$_i$ do
$d_1 =$ Distance (Term$_j$, $T$)
// Distance returns the number of arcs separating
// the node Term$_j$ and the root node in taxonomy $T$.
Aggregate-$\text{Term}=''$
//Aggregate-$\text{Term}$: stores the result of aggregation of two terms

For each Term$_k$ in Term$_{\text{Input}}$ do (k<j)
//k=Number of elements in Term$_{\text{Input}}$ - 1 (i.e., $\{\text{Term}_j\}$)
$d_2=$Distance (Term$_k$, $T$)
If ($d_1=d_2$) then
If (\text{Father (Term$_j$, $T$)}=\text{Father (Term$_k$, $T$)}) then
//\text{Father (Term$_j$, $T$)} returns the immediate father
// term of Term$_j$ in the taxonomy
Aggregate-$\text{Term}=\text{Father (Term$_j$, $T$)}$
EndIf
ElseIf ($d_1>d_2$) then
If (\text{Father (Term$_j$, $T$)}=\text{Term$_k$}) then
Aggregate-$\text{Term}=\text{Term$_k$}$
EndIf
Else  // ($d_1<d_2$)
If (\text{Father (Term$_k$, $T$)}=\text{Term$_j$}) then
Aggregate-$\text{Term}=\text{Term$_j$}$
EndIf
EndIf
EndIf
EndIf
EndIf
If (Aggregate-$\text{Term} = ''$) then
Term$_{\text{Output}}=$Term$_{\text{Output}}$ $\cup$ $\{\text{Term}_i, \text{Term}_j\}$
Else
Term$_{\text{Output}}=$Term$_{\text{Output}}$ $\cup$ $\{\text{Aggregate-$\text{Term}$}\}$
EndIf
EndFor
EndFor
MTR.Cell_{1} = \text{Term}_{\text{output}}

EndFor

Return MTR

End Function

Let us explain how $S$-Drill-Up runs on the input set of terms \{Star, OOLAP, MOLAP, ROLAP, Data Warehouse, Logical Design, Snowflake\} in cell $C1$ of Table 2.A.

First, the aggregation result (\text{Term}_{\text{output}}) is empty (algorithm, Line 4). Referring to the taxonomy, Star and Snowflake have the same father node Logical Design which itself belongs to $C1$. Thus, this node represents the aggregate term for the three initial terms (Line 18). As far as, OOLAP, MOLAP and ROLAP have the father node Architecture design (Line 18). Note that Design is the father node of Logical design and Architecture design nodes. Design does not belong to $C1$. Since the father node, Data Warehouse of the node Design belongs to $C1$ then Data Warehouse aggregates Design (Line 26). Figure 6 illustrates the outcome of $S$-Drill-Up on cell $C1$ in Table 2.A.

Figure 6. Terms of cell C1 in Table 2 (yellow) and their aggregate term obtained with S-Drill-Up (green) (Cell C1' in Table 2.B)

Figure 7. Original term in cell C1' (Table 2.B, yellow) and result of S-Drill-Down (green) (C1", Table 2.C)
Let us explain the S-Drill-Down throw an example; Figure 7 illustrates its application starting from $C1'$ of Table 2.B to reach the second level of the IST-taxonomy. To do so, we go down two levels from the level of Data warehouse and then find three terms Conceptual-design, Logical-design and Architecture-design. As the terms of the original cell $C1$ (Table 2.A) in level three (orange color) have the father nodes Logical-design and Architecture-design, we retain only this two terms.

EXPERIMENT AND EVALUATION

In order to validate our approach for building a NoSQL-DocW, we conducted experiments on the medical benchmark Clef-2007.

BENCHMARK DESCRIPTION

Due to the absence of a benchmark, we reuse our galaxy (Figure 8) (Ben Messaoud et al., 2015) generated from 3 DTDs and 1691 XML documents from Clef-2007. For simplicity, we ignore weak attributes.

**Figure 8. Galaxy for Clef 2007 (Ben Messaoud et al., 2015)**

**Test Environment**

To implement the galaxy NoSQL-DocW, we use MongoDB2 and Talend3. We use MongoDB CRUD (Create, Read, Update, and Delete) commands for creating collections, documents, along with simple and composed attributes, and, Talend to load the NoSQL-DocW.

**Evaluation and Discussion**

The transformation rules produce a NoSQL-DocW having two collections of documents, one describing the galaxy structure, and one for data storage.

We measure the efficiency of the implemented NoSQL-DocW with two metrics: Write-Request-Latency (WRL) and Read-Request-Latency (RRL) (Niyizamwiyiitira et al., 2017). WRL measures the loading time for a single write, and RRL measures the response time of a query. Furthermore, to assess the impact of hierarchies on performance, we conduct two tests; one on DocW-1 implemented without hierarchies (Ben Messaoud et al., 2018) and one on DocW-2 having hierarchies. The loading time WRL of 22002 rows is 2.25s for DocW-1 and 8.54s for DocW-2 (Appendix 1). Naturally, DocW-2 is slower due to the parameters to populate.
For the response time RRL of 35 queries (Appendix 2), DocW-2 is more rapid than DocW-1 (Figure 9). Appendix 3 depicts the RRL value of Q3 in the DocW-2.

From this experiment, we conclude that despite the significant loading time of DocW-2, compared to DocW-1, DocW-2 still suitable because the execution time (several queries every day) prevails on the loading time (e.g., once per week or per month).

CONCLUSION AND FUTURE WORKS

DocWs still in an embryonic era since users are looking to explore the semantics in textual documents. This requires further efforts investigating appropriate techniques for data storage and processing (Agrawal et al., 2011), design methods, developing software tools relying on the promising NoSQL technology to manage and analyze deeply and efficiently the semantics.

In this context, the NoSQL technology is an effective solution that can support scalability. Convinced by the efficiency of NoSQL, we have explored it in documents warehouses. Accordingly,
the contributions in this paper summarizes as the proposal of an approach to build and manipulate a DocW developed in NoSQL environment. This approach articulates around two methods Document Warehouse Builder and NoSQL-Converter.

Firstly, Document Warehouse Builder receives a set of XML documents in the same domain, and then generates the DocW galaxy model.

Secondly, NoSQL-Converter uses six transformation rules that convert the galaxy components into a NoSQL-DocW more appropriate for the storage and management of large textual data. We illustrated the transformation on a Research-Paper galaxy.

Thirdly, we suggested two semantic operators; the S-Drill-Up performs an aggregation at one level to find the immediate common father for pairs of terms. Inversely, S-Drill-Down breaks up terms to provide more details about the analyzed topic.

Finally, to prove our proposals we conducted an experimental work using MongoDB and Talend respectively for the implementation and loading the NoSQL-DocW. We assessed the loading time and the querying time on the medical collection Clef-2007. The experiments taught that the response time of OLAP queries executed on the NoSQL-DocW containing hierarchies is less than the response time of the same queries performed on the NoSQL-DocW devoid of hierarchies; this is motivating because OLAP queries are much more frequent than the ETL process.

We envisage extending the implementation of the operators S-Drill-Up and S-Drill-Down for different aggregation levels. Adding new semantic operators for the NoSQL-DocW is a further research track.
REFERENCES

Agrawal, D., Das, S., & El Abbadi, A. (2011). Big data and cloud computing: current state and future opportunities. In Proceedings of 14th International Conference on Extending Database Technology (pp. 530–533). New York: ACM.

Ben Mefteh, S., Khrouf, K., Feki, J., Ben Kraiem, M., & Soule-Dupuy, C. (2016). A Semantic Approach for XML Document Warehousing and OLAP Analysis. International Journal of Information & Decision Sciences, 8(3), 254–283.

Ben Messaoud, I., Alzaidi, A., Fattouch, N., & Ajala, A. (2018). Transform a Document Warehouse Model into NoSQL Document-Oriented Model. In Proceedings of 32nd International Business Information Management Association (pp. 3908-3920). Academic Press.

Ben Messaoud, I., Ben Ali, R., & Feki, J. (2017). From Document Warehouse to Column-Oriented NoSQL Document Warehouse. In Proceedings of 12th International Conference on Software Technologies (pp. 85-94). SCITEPRESS.

Ben Messaoud, I., Feki, J., & Zurfluh, G. (2015). A Semi-automatic Approach to Build XML Document Warehouse. In A. Fred, J. Dietz, D. Aveiro, K. Liu, J. Filipe (Eds.), Knowledge Discovery, Knowledge Engineering and Knowledge Management: Communications in Computer and Information Science (pp. 347-363). Springer International Publishing Switzerland. doi:10.1007/978-3-319-25840-9_22

Castelltort, A., & Laurent, A. (2014). NoSQL graph-based OLAP analysis. In Proceedings of 6th International Conference on Knowledge Discovery and Information Retrieval (pp. 217-224). SCITEPRESS.

Chandawni, G. (2016). NOSQL data-warehouse. International Journal of Innovative Research in Computer and Communication Engineering, 4(4), 96–104.

Chevalier, M., El Malki, Teste, O., & Tournier, R. (2015c). Implementation of Multidimensional Databases with Document-Oriented NoSQL. In Proceedings of 17th International Conference on Big Data Analytics and Knowledge Discovery (pp. 379-390). Springer.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., & Tournier, R. (2015a). Implementing multidimensional data warehouses into NoSQL. In Proceedings of 17th International Conference on Enterprise Information Systems (pp.108-130). SCITEPRESS.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., & Tournier, R. (2015b). Implementation of Multidimensional Databases in Column-Oriented NoSQL Systems. In Proceedings of 19th East-European Conference on Advances in Databases and Information Systems (pp. 79-91). Academic Press.

Dehdouh, K. (2016). Building OLAP cubes from columnar NoSQL data warehouses. In International Conference on Model and Data Engineering (pp. 166-179). Springer.

Dehdouh, K., Bentayeb, F., Boussaid, O., & Kabachi, N. (2015). Using the column oriented NoSQL model for implementing big data warehouses. In Proceedings of 21st International Conference on Parallel and Distributed Processing Techniques and Applications (pp. 469–475). Academic Press.

Dehdouh, K., Boussaid, O., & Bentayeb, F. (2014). Columnar NoSQL Star Schema Benchmark. In Y. Ait Ameur, L. Bellatreche, & G. A. Papadopoulos (Eds.), Model and Data Engineering (pp. 281–288). Springer.

Feki, J., Ben Messaoud, I., & Zurfluh, G. (2013). Building an XML Document Warehouse. Journal of Decision Systems, 22(2), 122–148.

Freitas, M. C., Souza, D. Y., & Salgado, A. C. (2016). Conceptual Mappings to Convert Relational into NoSQL Databases. In Proceedings of 18th International Conference on Enterprise Information Systems (pp. 174-181). Academic Press.

Gallinucci, E., Golfarelli, M., & Rizzi, S. (2018). Variety-Aware OLAP of Document-Oriented Databases. DOLAP.

Gallinucci, E., Golfarelli, M., & Rizzi, S. (2019). Approximate OLAP of document-oriented databases: A variety-aware approach. Information Systems, 85, 114–130.
Hecht, R., & Jablonski, S. (2011). NoSQL Evaluation A Use Case Oriented Survey. In Proceedings of International Conference on Cloud and Service Computing (pp. 336-341). Academic Press.

Inmon, W. H. (2002). Building the data warehouse. John Wiley & Sons.

Inmon, W. H., & Linstedt, D. (2014). Data Architecture: A Primer for the Data Scientist. Morgan Kaufmann.

Jacobs, A. (2009). The pathologies of big data. Queue, 7(6), 10–19.

Kimball, R., & Ross, M. (2011). The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons.

Krish, K. R., Khasymski, A., Butt, A. R., Tiwari, S., & Bhandarkar, M. (2013). Aptstore: dynamic storage management for hadoop. In 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (Vol. 1, pp. 33-41). IEEE.

Lemberger, P., Batty, M., Morel, M., & Rafaelli, J. L. (2015). Big Data et Machine Learning. Dunod.

Li, C. (2010). Transforming relational database into HBase: A case study. In 2010 IEEE international conference on software engineering and service sciences. IEEE.

McCabe, M. C., Lee, J., Chowdhury, A., Grossman, D., & Frieder, O. (2000). On the design and evaluation of a multi-dimensional approach to information retrieval. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval (pp. 363-365). ACM.

Niyizamviyitira, C., & Lundberg, L. (2017). Performance evaluation of SQL and NoSQL database management systems in a cluster. International Journal of Database Management Systems, 9(6), 1–24.

Pujolle, G., Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2011). Multidimensional database design from document-centric XML documents. In Proceedings of 13th International Conference on Data Warehousing and Knowledge Discovery (pp. 51-65). Springer.

Ravat, F., Teste, O., & Tournier, R. (2007). OLAP Aggregation Function for Textual Data Warehouse. In Proceedings of 9th International Conference on Enterprise Information Systems, (pp. 151-156). Academic Press.

Sellami, A., Nabli, A., & Gargouri, F. (2018). Entrepôt de données NOSQL orienté graphe: Règles de modélisation. In Proceedings of 12th Conference on Advances Decisional Systems: Big Data & Applications (pp. 442-453). Academic Press.

Sullivan, D. (2001). Document warehousing and text mining: Techniques for improving business operations, marketing and sales. John Wiley & Sons.

Tseng, F. S. C., & Chou, A. Y. H. (2006). The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. Decision Support Systems, 42(2), 727–744.

Yangui, R., Nabli, A., & Gargouri, F. (2016). Automatic Transformation of Data Warehouse Schema to NoSQL Database: Comparative Study. In Proceedings of 20th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (pp. 255-264).

**ENDNOTES**

1 The concept of surrogate key is a sequential value automatically generated for identifiers; it is widely used in ETL (Extract-Transform-Load) processes to standardize the identifiers when data come from heterogeneous sources.

2 https://www.mongodb.com/fr

3 https://fr.talend.com/
APPENDIX 1

Write Request Latency: WRL (On Docw-2)

Figure 10.
## APPENDIX 2

### Queries Description

| Table 3. |
|---|
| **N°** | **Query** | **Dimension(s)** | **Attribute(s)** | **Condition(s)** |
| Q1 | D-Carriage-Case (D-Carriage-Case) | D-Carriage-Case | M-D-Carriage-Case | M-D-Carriage-Case + 5135 |
| Q2 | D-Carriage-Case (D-Carriage-Case) | D-Carriage-Case | M-D-Carriage-Case | M-D-Carriage-Case + 2161 |
| Q3 | D-Carriage-Case (D-Carriage-Case) | D-Carriage-Case | M-D-Carriage-Case | M-D-Carriage-Case + 2233 |
| Q4 | D-Carriage-Case (P-WEBLINK) | D-Carriage-Case | P-WEBLINK | P-WEBLINK = "google it" |
| Q5 | D-Carriage-Case (P-ClinicalPresentation) | P-ClinicalPresentation | WA-ClinicalPresentation | WA-ClinicalPresentation = "Patient suivi au UPAM, hospitalisation pour " |
| Q6 | D-Carriage-Case (D-WA Department) | D-Carriage-Case | WA-Department | WA-Department = "Departement de Pneumologie" |
| Q7 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q8 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q9 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q10 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J.", "Frank Keck" |
| Q11 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q12 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q13 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J.", "Frank Keck" |
| Q14 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q15 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q16 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J.", "Frank Keck" |
| Q17 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q18 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q19 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J.", "Frank Keck" |
| Q20 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q21 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q22 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q23 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q24 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q25 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q26 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q27 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q28 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q29 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q30 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q31 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q32 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q33 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
| Q34 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Christian J." |
| Q35 | D-Carriage-Case (WA Author Name) | D-Carriage-Case | WA-Author Name | WA-Author Name = "Frank Keck" |
APPENDIX 3
Read Request Latency: RRL Of Q3 (On DocW-2)

![Diagram showing Read Request Latency: RRL Of Q3 (On DocW-2)]

Ines Ben Messaoud received his PhD in Computer Science, in joint supervision and joint diploma, from the University of Sfax-Tunisia and University of Toulouse I Capitole-France. Since 2013, she is assistant in the computer science department at the University of Gabes-Tunisia. She is a researcher at the Multimedia, Information Systems and Advanced Computing Laboratory (Mir@cl)-University of Sfax. Her research interests cover Data/Document Warehouse and Big Data.

Abdulrahman A. Alshdadi is assistant professor of Computer Science in Computer Science and Engineering (CCSE) at University of Jeddah. He has been awarded the PhD qualification in cloud computing in February 2018 from University of Southampton, Southampton, UK. His research interests span mainly around Industry 4.0 Prestaining issues of Cloud Computing and Fog Computing Security, Internet of Things (IoT) and Smart Cities, Intelligent Systems, Deep Learning, Data Science Analytics and Modelling. He has published numerous conference papers, Journal Papers and one book chapter. He is now acting as a head of Computer Science and Artificial Intelligent Department (CSAI) as well as Vice Dean of College of Computer Science and Engineering (CCSE) at University of Jeddah, Jeddah, Saudi Arabia.

Jamel Feki received his BS in CS (1980) from the University of Sfax (Tunisia), a Master’s degree (1981) and a Ph.D. (1984) in CS from the University Paul SABATIER (France). He joined the University of Sfax (Tunisia) in 1986 where he is a full professor and member of the Mir@cl research laboratory. Since 2015, he joined the University of Jeddah (Saudi Arabia) where he is a full professor. He has supervised several Ph.D. theses and has published research papers in refereed journals and conferences; he is a co-author of three book chapters. His research interests include Decision Support Systems and Business intelligence: analytical requirements specification, Data Warehouse design methods, DW Integration, Knowledge Warehouses, Big Data, and Data Science. He is a steering committee member in conferences and workshops; he is a PC member in international conferences and reviewer in journals.