Available for free online at https://ojs.hh.se/

Journal of Intelligence Studies in Business 2 (2012) 32-40

Multiversion Document Warehouse: An Approach to Multidimensional Analysis

Kaïs Khrouf*, Jamel Feki*, Chantal Soulé-Dupuy**

* MIR@CL Laboratory - University of Sfax - Tunisia,

** IRIT - University of Toulouse I – France

Received 15 December 2010; received in revised form 2 March 2012; accepted 27 April 2012

ABSTRACT: Document warehouses allow the storage of selected and filtered heterogeneous documents, as well as their exploitation through multidimensional analyses techniques. However, the content of documents is dynamic and changes across time. In practice, decisional analysts may be interested with various versions of documents. Thus, the document warehouse should store and manage these versions. This paper presents an extended generic model for document warehouses allowing the management of the multiversion documents. In addition, it interests with multidimensional analysis on documents versions.

Keywords: Document warehouse, Multiversion documents, Multidimensional analyses

1. Introduction

Nowadays, Internet allows an exponential evolution of data volumes stored and exchanged among organizations. These evolutions raise new problems: How to deal with changes undergone by documents? What are these changes and how to detect them? For instance, a user revisiting a document might want to be informed of the document changes since his last visit.

In order to maintain various versions of the same warehoused document, we need the concept of document warehouse. The author of (Khrouf & Soulé-Dupuy, 2004) defined the document warehouse as a source of information that is subject-oriented, filtered, integrated, archived (versions), and organized for a process of retrieval, interrogation or analysis.

According to this definition, documents integrated in the warehouse could be historized (i.e., retain their evolution over time through different versions). In order to reach this objective, we propose an extension for the document warehouse meta-model defined in (Khrouf, Feki and Soulé-Dupuy, 2011). This extension is expected to manage *content changes* (i.e., when the document content is modified) and *structural changes* (i.e., when the document structure changes) that can undergo one document or class of documents.

The extended meta-model allows applying techniques of multidimensional analyses on multiversion documents. We distinguish two types of analysis: i) *Multiversion analysis*, i.e., analysis covering all versions for the same document, and ii) *Recent-version analysis*; i.e., analysis relying on the last version of document(s).

This paper deals with the problematic of multiversion document warehouse; it is organized as follows. In section 2 we outline some works devoted to the management of multiversion documents. In section 3, we propose an extended meta-model for document warehouse and, in sections 4 to 6 we detail our approach of multidimensional analyses on multiversion documents integrated in the warehouse. Finally, we give an overview of our software prototype baptized DocWare (*Doc*ument *Ware*house).

2. Related works

For the management of multiversion documents, several theoretical works have been proposed in the literature; furthermore, software prototypes have emerged.

Nicolle, Alvarez & Amghar (2001) consider that the document is a set of independent fragments (parts). They distinguish two types of versions: a document version and a fragment version. In fact, the modification of certain document fragments creates new versions of fragments, and therefore a new version of the whole document.

XyDiff (Cobéna Abiteboul & Marian, 2002) is a component of Xylème (Abiteboul, Cluet, Ferran & Rousset, 2002) to manage different versions of a document. Every modified item is represented as an XML file, stored in a data warehouse and indexed. These files are used thereafter to reconstruct previous versions of documents. XyDiff uses the tree structure of XML documents in order to detect movements and changes taking place on a document.

X-Diff (Wang, DeWitt & Cai, 2003) is an algorithm for integrating the characteristics of XML structures with standard techniques of tree comparison in order to calculate the differences between two versions of an XML document. The main feature of this algorithm is that XML documents are modeled by unordered tree structures, unlike the work of XyDiff.

Rusu, Rahayu & Taniar (2006) propose an approach for extracting rules from the changes of version of dynamic XML documents. Specifically, the authors propose an algorithm that studies the conduct of versions of XML documents in time and thus determines learning rules to predict document changes in the future.

In our work, we are interested not only in the management of document versions (track and detect changes of the document evolution through time), but also for managing the versions of the collections of documents (set of documents gathered in the same class). In addition, we develop a multidimensional analysis approach for these multiversion documents.

3. Meta-model for document warehouses

3.1 Meta-model description

The document warehouse should store pertinent documents in order to apply the multidimensional analyse on these documents; In addition, it should be able to manage the heterogeneity and support the evolution of structures and contents. To do so, we propose the meta-model of Figure 1.

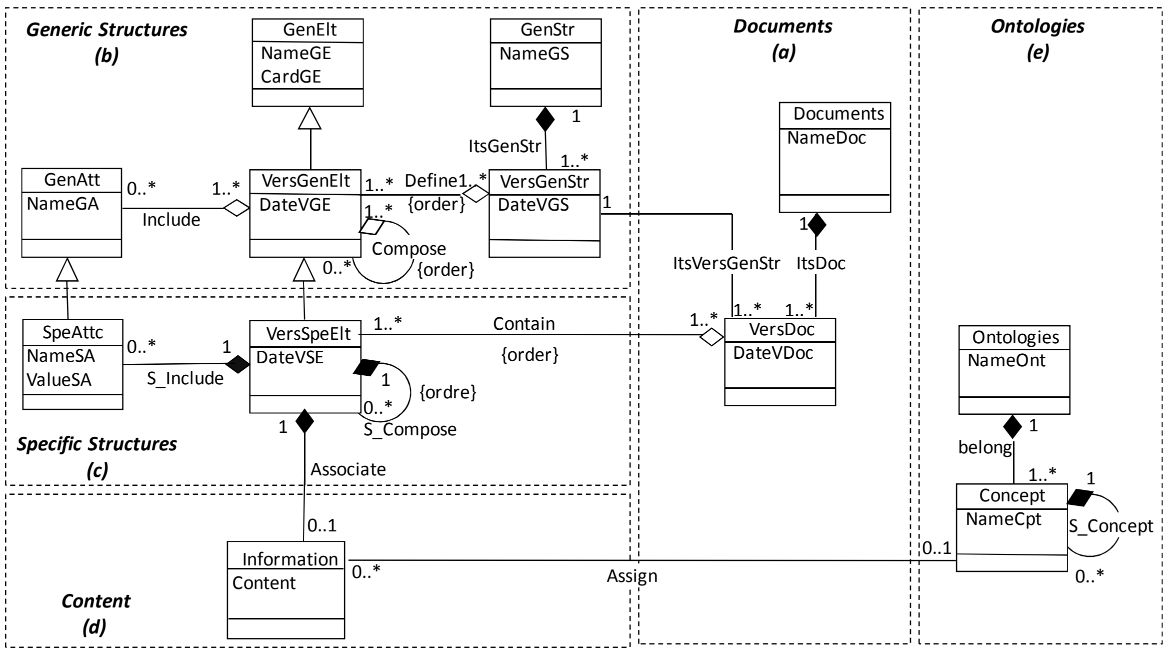

Figure 1: Meta-model for multiversion document warehouses

This metadata includes the following components:

•A set of documents (Figure 1.a) to be integrated in the document warehouse and their different versions (Figure 1.a).

•The hierarchical structure of documents. It is made up of two types of structures:

I. *The generic structure* (Figure 1.b): It is a common structure for a document set. It is composed of a set of versions each of which is defined by a set of versions of generic elements which can be composed of other versions of generic elements. Each of these elements can also be described by generic attributes for example book-Id.

II. *The specific structure* (Figure 1.c): It is associated to a single document and has to be compliant/identical to one among the existing versions of generic structures. This structure is defined by a set of versions of specific elements that can include specific attributes.

•The content (Figure 1.d) is the textual element of the specific structure.

•The semantic layer (Figure 1.e) is defined using domain ontologies. In our context, ontology is composed of a set of concepts hierarchically organized where each leave concept is described by a set of keywords.

3.2 Example

Figure 2 depicts a simple instantiation example for our meta-model of Figure 1. In this example, we manage three versions of the same document *Doc1*:

•*Doc1* is initially compliant to *Version1* of the generic structure *Article* composed of *Title* and *Content.*

•After changes made on the *Content* element, *Doc1* belongs now to the new *Version2* of *Article*.

•After renaming the *Content* element to *Section* composed of two *Paragraphs* (i.e., Dimension and Fact), the new version of *Doc1* is becoming conform to *Version3* of the generic structure *Article*.

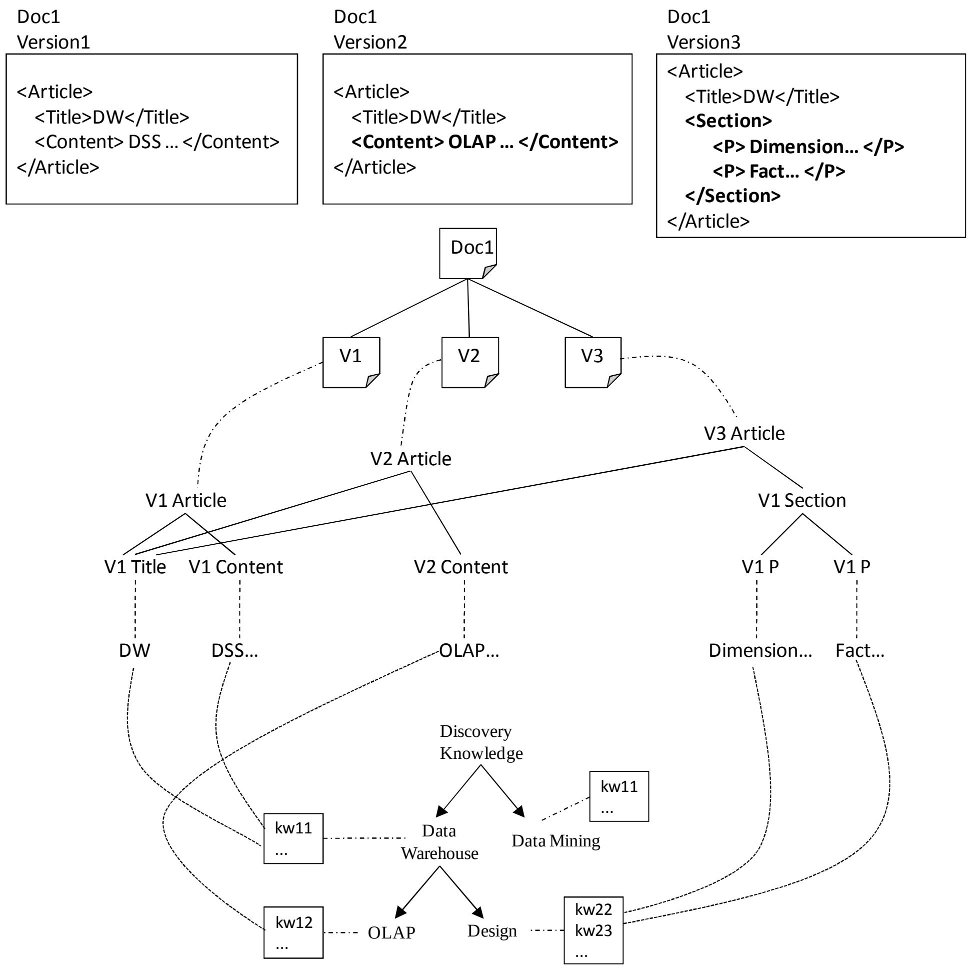

Figure 2: An instantiation example for the meta-model in Figure 1.

3.3 Meta-model advantages

The meta-model we proposed has the following advantages:

•Grouping heterogeneous documents having identical or similar structures into classes. This relies on an algorithm for comparing labeled tree structures (Ben Messaoud, Feki, Khrouf & Zurfluh, 2011)

•Storing various versions of documents due to evolutions.

•Adding up of semantics to the documents by linking the textual content to the concepts of domain-ontologies (Ben Meftah, Khrouf, Feki, Ben Kraiem & Soulé-Dupuy, 2011).

•Applying multidimensional techniques on documentary information. This feature will be detailed in section four.

3.4 Meta-model implementation

As shown in Figure 1, the meta-model is designed using the *Unified Modeling Language* (UML) object-oriented modeling. The meta-model implementation is carried out in an object relational DBMS (Oracle 10g). To ensure this translation, we have used the following transformation rules:

•Classes are transformed into tables.

•For *one-to-many* relationships implementation, we have two alternatives: use one *mono-valuated* link or one *multi-valued* link in the opposite direction. We opted for the mono-valuated link as they facilitate the generation phase of views necessary for the multidimensional analyses.

VersDoc

Id_Doc	DateDoc	ItsVGS
319	04/02/2012	→
716	05/04/2012	→
1426	14/05/2012	→

VersGenStr

Id_VGS	DateVGS
17	01/02/2012
24	05/04/2012

Example 1

•We implement *many-to-many* relationships using multi-valuated links, specifically by using a list of references as nested tables.

VersGenStr

Id_VGS	DateVGS	ItsVGE
17	01/02/2012	→
		→
24	05/04/2012	→
		→

VersGenElt

Id_VGE	DateVGS
67	01/02/2012
68	01/02/2012
85	05/04/2012

Example 2

•For inheritance, we opted for mono-valued links from subclasses to super-classes in order to separate the two structures, *generic and specific*.

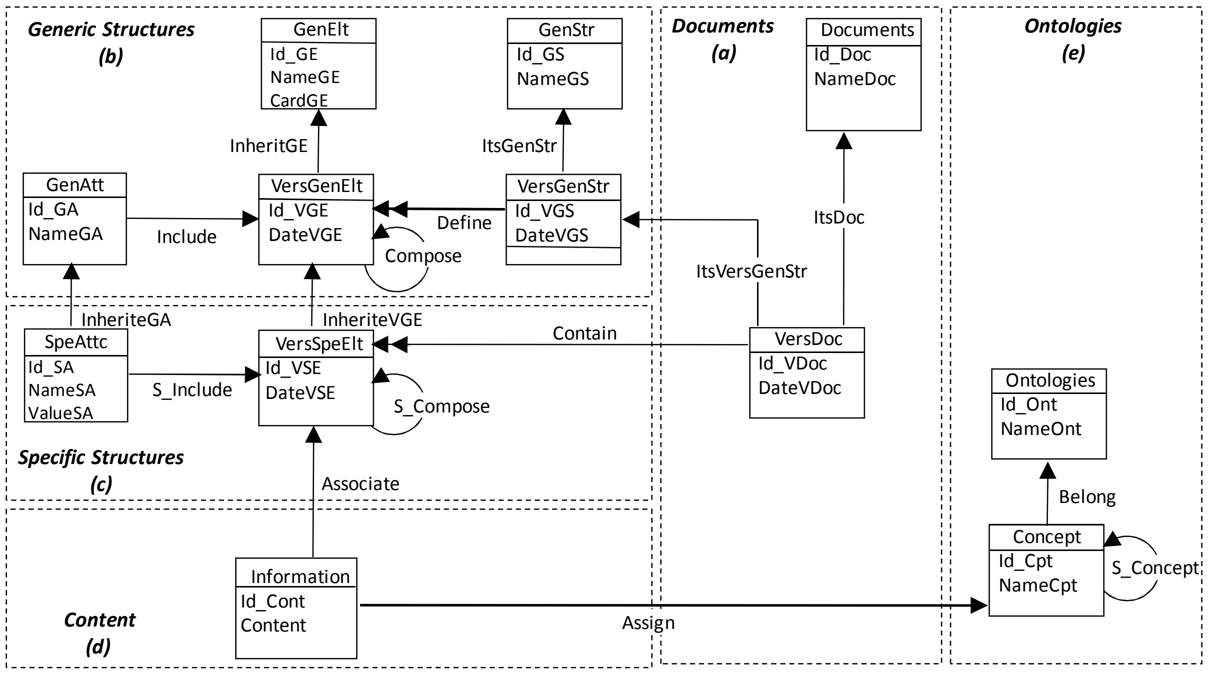

Figure 3: The navigational diagram of the proposed meta-model in Figure 1

3.5 Meta-model instantiation

The integration of a document into the warehouse is accomplished through the three following steps:

I. Extraction of the *specific structure* for the document by using a parser; it includes the document tags and its hierarchical structure.

II. Comparison of the *specific structure* of the document with the *generic structures* stored in the warehouse. This step is accomplished through an algorithm which calculates a similarity degree to compare labeled tree structures (Ben Messaoud, Feki, Khrouf & Zurfluh, 2011).

III. Insertion of the document content, information and list of keywords into the warehouse while linking the textual information to one or more concepts that also are characterized by keywords. We use the information retrieval techniques to perform this step (reference)**.**

4. Multidimensional analyses

The document warehouse is intended to allow decision-making. To do so, we adopt the multidimensional model (Kimball & Ross, 2002) that considers an analyzed subject as a point within a space having several dimensions. This model relies on the concepts of *fact* and *dimension*. The fact represents the subject to be analyzed as the number of articles and, the dimensions represent the context of recording the fact such as *Author*, publication *Year* and *Conference*. Dimensions are made up of attributes organized, from the finest to the greatest granularity, into hierarchies.

Figure 3 describes our proposed multidimensional process to analyze textual information stored in the document warehouse.

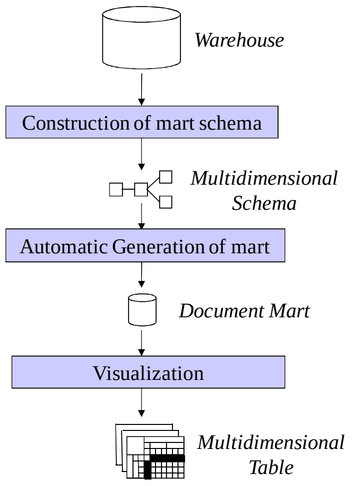

Figure 4: Multidimensional analysis process

In following section, we detail the first two phases of this process.

5. Phase 1: Construction of the document mart schema

Let us remember that a generic structure gathers a set of documents having identical or similar structures. The decision makers can focus on a generic structure to perform his/her analyses. The first step consists in (1) selecting the analysis context through the choice of the generic structure on which analyses will be applied, and then (2) selecting the type of analysis: Analysis covering all versions or relying only on the last version of documents.

During *step two*, the decision-maker selects the multidimensional schema components, one *fact* and a set of related *dimensions*:

•A fact represents a subject of analysis, composed of a set of attributes describing the business activity. These attributes are called *measures* or *indicators* and have numeric values. As an example, let us consider the fact *Publication* that has the measure *Number of published articles.*

•The dimensions represent the analysis axes of measures. This means that the measures of an activity are observed according to these different dimensions. For instance, measures of the *Publication* fact can be analyzed according to the several dimensions as *Author, Year,* and *Concept.*

In addition, the decision-maker indicates the order of dimensions and the aggregation function (Count, Sum, Max, Min and Avg) to be applied to the fact measures.

In the *third step*, the decision-maker can select specific values or introduce predicates in order to filter data for analysis. We distinguish two types of data filtering:

•*Dimension filtering* through which the user can select values on a dimension.

•*Fact filtering* where the user restricts the values of the fact measures using the comparison operators (<, >, <>, <=, >=, =).

Example:
Let us analyze the number of *Publications* addressing the *Data warehouse* concept by *Author* and by *Year*.

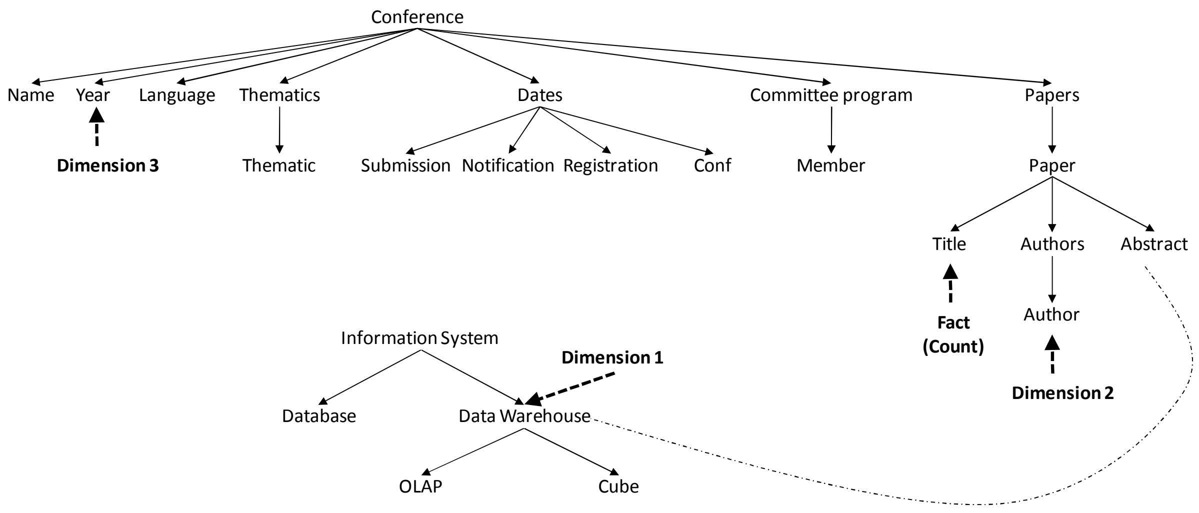

Figure 5: Affectation of analysis components

Once all these document mart schema-components are defined, the next phase generates the document mart. In our approach, this generation is automatically performed.

6. Phase 2: Automatic generation of document mart

The decision-maker task is now completed and the automatic generation produces a document mart instantiated from the warehouse. To simplify this generation, we decompose it into two complementary steps namely *view generation* for each analysis component, element or concept, and *joining and grouping* generated views.

6.1 Views generation for analysis component

The first step is to recover the identifiers of the versions of documents belonging to the same generic structure and concerned by the analysis.

•Multiversion analysis

```
SELECT Id_VDoc
FROM VersDoc VD
WHRE VD.ItsVersGenStr.ItsGenStr.SaGS.NameGS =
'NameGS';
```

•Recent version analysis

```
SELECT vd.ItsDoc.Id_Doc, Max(DateVDoc)
FROM VersDoc VD
GROUP BY VD.ItsDoc.Id_Doc;
```

Secondly, we recuperate trough a sub-query three attributes:
(1) The identifier of each document.
(2) The identifier of the common ancestor of analysis components.
(3) The concerned information.

These sub-queries are merged by the SQL Union operator to obtain a single view. The sub-query the system generates is the following.

```
SELECT
  'Id_VDOC',                                          (1)
  i.Associate.S_Compose.S_Compose….ID_VSE,  (2)
  i.Content                                           (3)

FROM Information i                                    (4)

WHERE i.Associate
   IN (Select nt.AdrVSE
       From The (select vd.Contain
                 From VersDoc vd
                 Where ID_Doc= 'ID_Doc')nt);(5)

--If the dimension is a generic element
AND i.Associate.InheritVGE.InheriteGE.NameGE=
'NameGE'                                              (6)

--If the dimension is a concept
AND i.contain.NameCpt='NameCpt'                       (7)
```

Where:

(1) Document identifier

(2) Identifiers of specific elements those inherit from the first common ancestor of all analysis elements.

(3) Content of the specific element.

(4) Meta-model table name.

(5) Selection of the specific elements belonging to the document *ID_Doc*.

(6) Selected name of the generic element (when a dimension is based on a generic element).

(7) Name of the concept on which a dimension is based.

Note that the fact view is generated in the same way like dimensions; the *S_Compose* denotes the link between a specific element and its father specific element so then the occurrences of *S_Compose* equal the number of levels between a chosen element and its ancestor.

As an example, for the *Year* dimension (cf. Figure 5) and the document 314 the system generates the following script.

```
SELECT
    '314',
    i.Associate.S_Compose.ID_VSE,
    i.Content

FROM Information i

WHERE i.Associate
   IN (Select nt.AdrVSE
       From The (select vd.Contain
                 From VersDoc vd
                Where ID_Doc= '314')nt);

AND i.Associate.InheritVGE.InheriteGE.NameGE=
'Year'
```

The ancestor element of the analysis components (*Abstract*, *Author*, *Year*, *Title*) is *Conference*. There is one level between *Year* and *Conference*. That's why *S_Compose* is 1.

For the analysis component *Data Warehouse* concept (cf. Figure 5), the system generates the following script for the same document Id 314.

```
SELECT
'314',
i.Associate.S_Compose.S_Compose.S_Compose.ID_
SE,
i.Content

FROM Information i

WHERE i.Associate
   IN (Select nt.AdrVSE
       From The (select vd.Contain
                 From VersDoc vd
                Where ID_Doc= '314')nt);

AND i.contain.NameCpt='Datawarehouse'
```

The number of levels between *Abstract* and *Conference* (ancestor element of the analysis components) is 3. Thus the occurrences of *S_Compose* equal 3.

6.2 Joining and grouping generated views

After generating the view for the fact and its dimension views, we follow by linking these views on their two first attributes, thus we generate a new view called *Joint*. For our running example, it is the following.

```
CREATE  VIEW  Joint  (DataWarehouse,  Year,
Author, Title) AS
SELECT DataWarehouse, Year, Author, Title
FROM DataWarehouse d1, Year d2, Author d3,
     Title f
WHERE d1.doc = d2.doc AND d2.doc = d3.doc
AND   d3.doc = f.doc  AND d1.Anc = d2.Anc
AND   d2.Anc = d3.Anc AND d3.Anc = f.Anc;
```

To generate the final view that describes the document mart we *Group by* all dimensions and apply the *Count* function.

```
CREATE  VIEW  Result  (DataWarehouse,  Year,
Author, Nb) AS
SELECT    DataWarehouse,    Year,    Author,
Count(title)
FROM   Join
GROUP BY DataWarehouse, Year, Author;
```

Figure 6 displays the result, obtained with the generated view, in a multidimensional table.

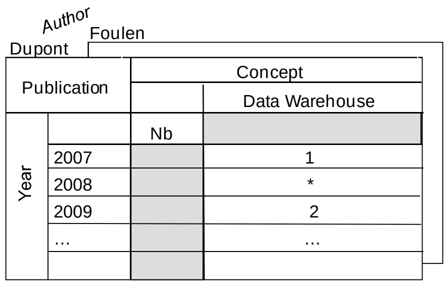

Figure 6: Multidimensional table

7. DocWare prototype: Experimentation

To validate our proposals we developed the software prototype *DocWare* (***Doc***ument ***Ware***house) for the integration and the analysis of textual data. Specifically, *DocWare* provides the two following main features: First it determines the *generic* and *specific structures* of documents and then inserts these documents automatically into the document warehouse, and secondly assists the administrator (or even skilled decision-makers) during the construction of the document mart.

In the remainder we illustrate some functionalities of *DocWare* through the following example. Suppose we want to count the number of scientific papers dealing with the *Data Warehouse* concept, by *Author* and publication *Year*.

•CONTEXT

Accessing the document warehouse content we find that the documents describing the papers are grouped into the generic structure *Conference*. It contains all necessary elements to perform the analysis (Abstract, Year and Author).

•APPROACH
We follow the three steps of our approach.

I. Choice of analysis context:
We start by defining the generic structure for the document mart to be constructed. Thus, the system displays. Among the list of stored structures in the warehouse, we choose the generic structure *Conference* that will be visualized by a tree (Figure 7).

II. Selection of analysis components:
We specify the role (dimension or fact) of elements to build the mart by using contextual menus. Chosen elements are automatically highlighted by using different shapes and colors for dimensions (read) and facts (yellow). In our example, we assign the *Data Warehouse* concept to the generic element *Abstract* as the first dimension. Then, we select the generic elements *Year* and *Author* as the second and third dimensions. Finally, the measure is the count of *Titles*.
To assign a concept to a generic element, DocWare displays the list of all existing ontologies in the warehouse; this enables us to choose the appropriate ontology (cf. Figure 8).

III. Filtering:
As we want to analyze the count of papers for the authors of this paper, we apply a filter on the third dimension. The system displays all *Author* values; among them we select the three following names: Kaïs Khrouf, Jamel Feki and Chantal Soulé-Dupuy.

•RESULT
To visualize the result, DocWare creates views according the approach described in section 6 and displays the result multidimensional table (cf. Figure 9).

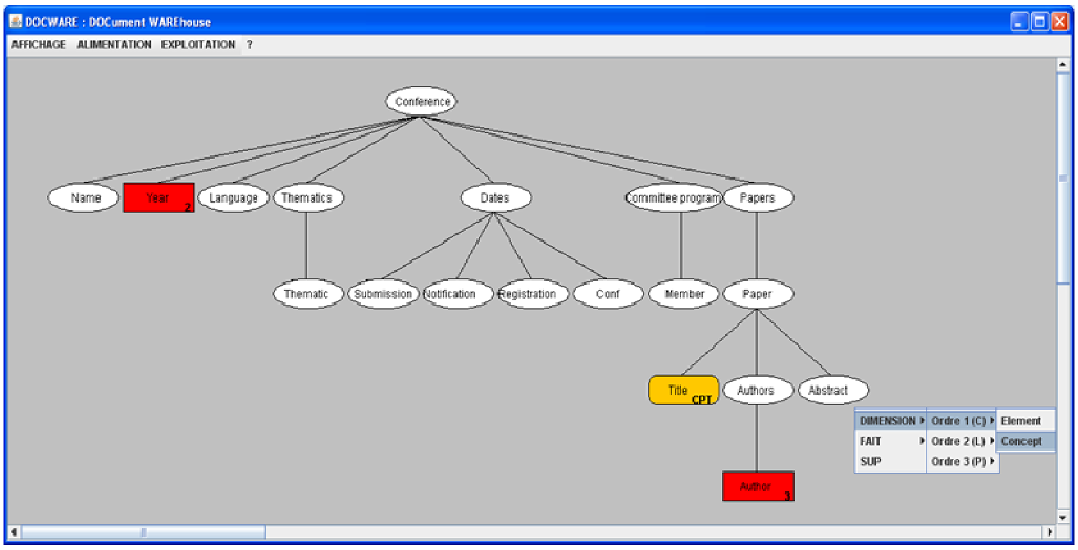

Figure 7: Affectation of a fact and dimensions

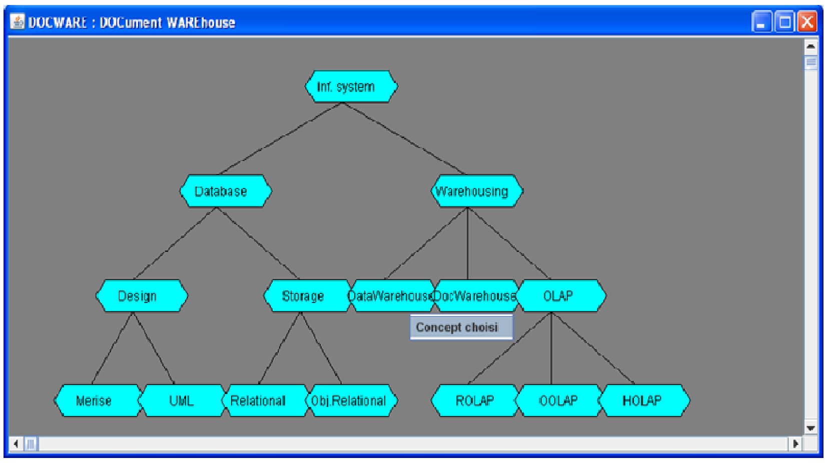

Figure 8: Affectation of concept for the generic element Abstract

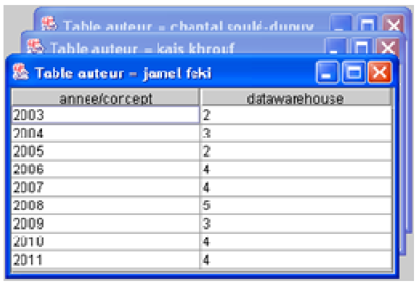
Table auteur = chantal soulé-dupuy
Table auteur = kais khrouf
Table auteur = jamel feki

annee/corcept	datawarehouse
2003	2
2004	3
2005	2
2006	4
2007	4
2008	5
2009	3
2010	4
2011	4

Figure 9: The Result multidimensional table

8. Conclusion

The document warehouse allows flexible manipulation of heterogeneous collections of documents based on their structures and contents. In this paper, we extended the document warehouse meta-model toward a metamodel that supports multiversion document warehouse. This is for integrating a new feature: the management and analysis of multiple versions of documents. As documents evolution may concern their structure and/or content, we addressed the storage of versions compliant to a same document structure, as well as versions compliant to a multiple document structures. Decision makers could be interested with the document evolutions, or even ignore them. Therefore, we suggested two types of analysis on documents namely: i) *Multiversion* analysis; i.e., covering all versions for a same document; and ii) *Recent-version* analysis; i.e., analysis relying only on the last version of documents. In our proposed approach, each document version is compliant to a version of specific structure. Furthermore, various versions of the same document are able to be compliant to several versions of generic structures.

As an immediate perspective, we aim to extend the process of multidimensional analysis by integrating personalization criteria and metadata; this could be done by the user himself or by an assisted process. In addition, semantic aspects during the analysis process are interesting; they can help decision makers to get better analytics.

Acknowledgement

We would like to kindly thank Dr Mohamed Mbarki and Ms Maha Azabou (Master degree student) for their contribution to the implementation of the DocWare system prototype.

References

Abiteboul S., Cluet S., Ferran G., Rousset M.C. (2002). *The Xyleme Project*, Computer Networks, 39(3): 225-238, 2002.

Ben Meftah S., Khrouf K., Feki J., Ben Kraiem M., Soulé-Dupuy C. (2012). *Document Warehouse: Integration of Semantic Structures*, International Conference on Information Systems ans Intelligence Economic, Djerba, Tunisia.

Ben Messaoud I., Feki J., Khrouf K., Zurfluh G. (2011). *Unification of XML Document Structures for Document Warehouse (DocW),* International Conference on Enterprise Information Systems, p. 85-94, Beijing, China.

Cobéna G., Abiteboul S. & Marian A. (2002). Detecting changes in XML documents. In International Conference on Data Engineering (ICDE'2002), p. 41-52, San Jose, California, USA.

Kimball R. & Ross M. (2002). The Data Warehouse Toolkit (2 edition). New York: John Wiley & Sons.

Khrouf K. & Soulé-Dupuy C. (2004). A Textual Warehouse Approach: a Web Data Repository, (p. 101-124). Hershey: Idea Group Publishing.

Khrouf K., Feki J., Soulé-Dupuy C. (2011). An Approach of Multidimensional Analysis of Document. International Conference on Information Systems ans Intelligence Economic, Marrakech, Morocco.

Nicolle C., Alvarez A., Amghar Y. (2001). Managing Versions and Links for Structured Legacy Documents, International Symposium on Information Systems and Engineering (ISE'2001), June 25-28, Las Vegas, Nevada, USA.

Rusu L.I., Rahayu J.W., Taniar D. (2006). Mining Changes from Versions of Dynamic XML Documents, p. 3- 12, Workshop on Knowledge Discovery in XML Documents (KDXD), p. 3-12, Singapore.

Wang Y., DeWitt D.J., Cai J.Y. (2003). X-Diff: An Effective Change Detection Algorithm for XML Documents, International Conference on Data Engineering (ICDE'03), p. 519-530, Bangalore, India.