UNIFICATION OF XML DOCUMENT STRUCTURES FOR DOCUMENT WAREHOUSE (DocW)

Ines Ben Messaoud, Jamel Feki, Kais Khrouf
Laboratory Mir@cl, University of Sfax, Airport Road Km 4, Sfax, Tunisia

Gilles Zurfluh
IRIT, Institut de Recherche en Informatique de Toulouse, Université Toulouse 1, Toulouse, France

Keywords: Document warehouse, DTD unification, XML document, Unified tree.

Abstract: Data warehouses and OLAP (On Line Analytical Processing) technologies analyse huge amounts of structured data that companies store as conventional databases. Recent works underline the importance of textual data for the decision making process and, therefore, lead to build document warehouses. In fact, documents help decision makers to better understand the evolution of their business activities. In general, these documents exist in XML format, are geographically distributed and described by multiple and different structures. This paper deals with a method to build a distributed document warehouse. This method consists of two steps: *i) unification of XML document structures* in order to set a global and generic perception/view of the distributed document warehouse, and *ii) multidimensional modeling* of unified documents for decisional purposes. More specifically, this paper focuses on the unification step.

1 INTRODUCTION

Data warehouses (DW) are widely used in many organizations and their benefits are no longer to be proven. They allow storage and analysis of huge amounts of structured business data (Perez and al., 2008). These data are generally numeric values tracing the whole activities of the organization and, therefore, are used as the basis for evaluating/analyzing the business performances as well as for well-founding decisional processes managed by decision makers (i.e., analysts). Recently, enterprises have become aware that DW is solving a small part of their real integration and analysis needs (Pedersen, 2009). Indeed, 20% information is extracted from data warehouses as numeric data and the other 80% information is hidden in non-numeric data (*i.e.*, in documents) (Tseng and Chou, 2006). Consequently, decisional data (i.e., data useful for the decisional process) can be found not only as numeric values directly recorded in database objects (*i.e.*, tables or classes) but may be spread in textual data as in reports. This leads to enrich the Decision Support System (DSS) by including documents. Those documents can be structured, semi-structured or non-structured. As example of documents, we can mention business journals, credit reports and industry newsletters. These documents help decision makers to better understand the evolution of the corporate data over time. However, they represent an important volume to be integrated into the DSS.

On the other hand, the number of documents is permanently growing through time. As a result, decision makers pay out much of their working time to explore documents looking for data that help them in their decisional processes. Obviously, without appropriate design and powerful software tools, decisional analysts can not easily and rapidly explore documents. As a consequence, in many cases, when making important decisions, some relevant documents may be ingnored whereas some irrelevant ones may be considered by intuition. The result may be imperfect because the decision making process is based on incomplete information, or even noisy (Tseng and Chou, 2006). In order to alleviate this problem, documents must be warehoused (McCabe and al., 2000).

In addition, documents are usually stored in several distant geographical sites and may be

Ben Messaoud I., Feki J., Khrouf K. and Zurfluh G..
UNIFICATION OF XML DOCUMENT STRUCTURES FOR DOCUMENT WAREHOUSE (DocW).
DOI: 10.5220/0003502100850094
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 85-94
ISBN: 978-989-8425-53-9

presented in many different formats. Among these formats, the most popular one in use is XML «Extensible Markup Language». So, we focus on XML documents since XML is the most used format for data representation and exchange. In general, those documents are described by several structures. Consequently, a step to unify the structures of XML documents is compulsory in order to produce a global view describing a large document set. This step should be followed by a multidimensional modeling of documents; it enables multidimensional analysis of the documents content. In this paper, we present a method to build a document warehouse. This method is composed by two steps namely: *unification of XML document structures*, and *multidimensional modeling* of these documents. More specifically, we focus on the unification step that produces a global view for the distributed document warehouse (DocW).

This paper is organized as follows. In section 2, we describe the most popular works that treat DocW and unification of structures of XML documents. Secondly, in section 3, we propose our method to build a DocW. Then, we present the process to unify the sructures of XML documents set. After that, we present an example of unification in section 5. Finally, in section 6, we conclude the paper and overview the next steps of our current works.

2 RELATED WORK

In (Inmon, 1994), the author underlines the importance of external contextual information to understand the results of the historical analysis operations. In fact, the external contextual information is usually available in documents (e.g., on-line news and company reports). Those documents can be presented as XML formats. Usually, the structure of XML documents may be different for documents belonging to the same domain (e.g., DTDs of research papers are different). In this section, we first define the concept of DocW and then, we describe the most popular works related to document warehousing. Finally, we depict the relevant works that treat unification of XML document structures.

A DocW is designed to store documents issued from internal and external data sources. In fact, within a DocW, documents are organized for effective analysis, or feature extraction to enable distilled and fruitful business intelligence (Tseng and Chou, 2006). Moreover, a DocW can be seen as a special case of a Content Warehouse. In fact, a Content Warehouse archives Web data composed of texts. This warehouse is derived into *Text Warehouses* when little structure was available and *Document Warehouses* when structure was available (Ravat and al., 2010). In the literature, there are few works regarding DocW; significant ones are (Tseng and Chou, 2006), (Perez, 2007) and (Perez and al., 2008).

In (Tseng and Chou, 2006), the authors present architecture for a DocW where documents are supplied by internal and external sources. Those documents undergo a pre-processing treatment, such as text summarization and text feature extraction (front-end component). After that, the extracted elements are stored as a metadata. The warehouse administrator performs all the operations associated with the management of the documents in the warehouse, such as create dimensions and document indexes for constructing document cubes and archive documents. In addition, the proposed architecture is composed of a back-end component that performs all the necessary operations for the queries management (document access tools, multi-dimensional document query interface, etc.). However, only quantitative reports can be made because indicators are numerical. Furthermore, the authors consider only the textual parts of documents (excluding graphics, multimedia data, etc.).

Another work, (Perez, 2007) and (Perez and al., 2008) propose an architecture that integrates corporate warehouse with a DocW (text-rich XML documents), resulting in a contextualized warehouse. "*A contextualized warehouse is a DSS that allows users to combine all their sources of structured and unstructured data and to analyze the integrated data under different contexts*". In order to build a contextualized OLAP cube, the analyst specifies, by a sequence of keywords, the context of analysis. The resulting cube is called *R-cube* (*R* stands for *R*elevance). This cube is characterized with two specific dimensions namely: *relevance* and *context*. The *relevance* dimension depicts the importance of each fact of the cube in the selected context. But, the *context* dimension represents the documents of the warehouse. Also, each fact in the R-cube is linked to the set of relevant documents that describe its context. However, the proposed approach provides additional information for decisional analysis. Indeed, it returns to the decision maker the relevant documents to an analysis. Furthermore, the process responsible for detecting the facts described in the documents does not consider fact without corresponding corporate facts.

Other works, as (Júnior and Mello, 2008) and (Yoo and al., 2005), treat the unification of the structures of XML documents.

In (Júnior and Mello, 2008), the authors propose an approach for semantic integration of XML instances. This approach is composed of two processes: *Similarity definition* and *Unification*. The similarity definition compares each pair of instances of document and stipulates a similarity scores for them. It generates several sets of XML instances. Those sets contain XML instances semantically similar. The second process (i.e., unification) generates a unified XML representation for each set. It uses domain ontology and a dictionary in order to name and to structure the resulting XML instance. Nevertheless, the comparison of documents may be time consuming because some of these documents can have the same structure. The processing time can be reduced by raising the comparison to the level of documents structures.

(Yoo and al., 2005) propose an algorithm for the unification of DTDs for XML documents having similar structures and belong to the same domain. This algorithm accepts a set of similar DTDs (Document Type Definition) and generates a unified DTD. In fact, a unified DTD plays the role of a global conceptual schema for subjects common to a given domain. The algorithm uses finite automata and tree structure to represent documents. It includes four steps: *Pre-processing*, *DTD representation*, *Uniform DTD generation* and *Post processing*. First, the pre-processing resolves names ambiguities of the elements of DTDs. It unifies the names of elements having the same meaning using an Element Name Resolution Table. In the second step, DTDs are represented as trees and finite automata. Then, trees and finite automata are merged together to create a unified DTD. Finally, the final DTD is verified using a DTD parser. However, the quality of result relies on the completeness of the Element Name Resolution Table.

After this short overview of related works, we focus on the work related to DocW instead of heterogeneity and distribution issues for XML documents. Also, we emphasize that usually the designer does not participate in the process of unification of XML documents. The remaining of this paper presents our method to build a DocW.

3 BUILDING A DocW

Documents help decision makers to better understand the evolution of the corporate data over time. However, they represent an important volume to be integrated into the DSS. In (Tseng and Chou, 2006), the authors claim that 20% information extracted from data warehouses is numeric data and the other 80% information is hidden in documents. In fact, there are several formats of documents, such as XML. XML documents allow the exchange of a wide variety of data on the Web and elsewhere (http://www.w3.org/XML/).

In general, documents are distributed on various sites and their structures differ from one site to another. So, to query several sites of documents, we need a common structure which plays the role of the global schema in a distributed database. Therefore, we propose a method to build a DocW composed of distributed XML documents. This method consists of two steps (Ben Messaoud and al., 2010):

- Unification of XML document structures.
- Multidimensional modeling of documents.

The first step defines a common structure to describe XML documents located at different sites. We can assume that all documents stored at the same site have a unique and common structure.

The second step aims to design multidimensional schemas that highlight the OLAP analysis components: facts, measures, axes (i.e., dimensions), perspectives (i.e., hierarchical levels). It is currently studied as a complementary task for this work (Ben Messaoud and al., 2011).

In the remainder of this paper, we restrict ourselves to detail our proposal for the unification of XML document structures.

4 UNIFICATION OF XML DOCUMENT STRUCTURES

The content of an XML document is encapsulated within elements that are defined by tags. Those elements are hierarchically organized as a tree. Syntactically, two formalisms may be used to describe the structure of XML documents namely: DTD and XSchema. Usually, XML document structures may differ among documents. As a result, when a decision maker needs to retrieve information from several sites where documents are stored, he has to consider this structural heterogeneity. This may lead him/her to write multiple queries; i.e., as many queries as the number of different structures for the document set and, therefore, manually/programmatically build the final result using the returned sub-results. To overcome this

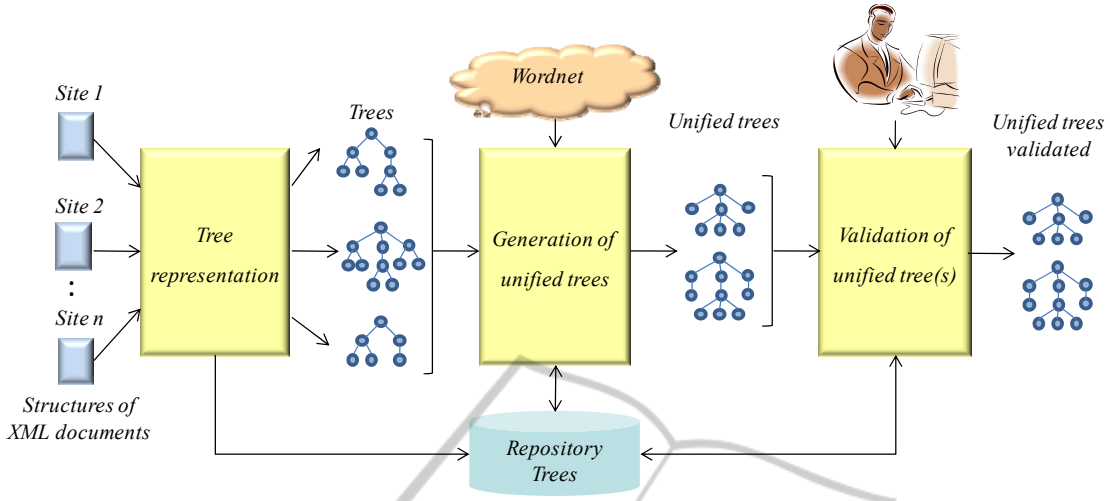


Figure 1: Unification of XML document structures.

problem, we expect to define a unified structure; i.e., a structure convenient to describe multiple heterogeneous XML documents.

Thus, this unified structure will be useful to write a single query that operates on several documents simultaneously.

In order to build this unified structure for structurally heterogeneous XML documents, we propose a method that exploits the XML tree structure. Our unification method consists of the three following steps:

- Tree representation.
- Generation of unified trees.
- Validation of unified tree(s).

Figure 1 shows the sequencing of these steps. The first step translates the XML document structure into the formalism of trees as done in (Yoo and al., 2005) and (Lee and al., 2002). Such a result tree characterizes the site from which it is issued its nodes represent the elements of an XML structure with their cardinalities, and its arcs denote relationships between XML elements. We choose the tree formalism because it is easy to be understood by end-users (i.e., decision-makers). In fact, the usage of trees motivates decision-makers to participate in the next steps, mainly during the unification process. Figure 2 depicts an example of DTD and its corresponding tree; it is for research papers belonging to the site called *Laboratory1*.

In the remainder of this paper, we note a tree T as follows: $T\ (E, r, N, S)$ where E represents the set of all nodes of T, r is its root node ($r \in E$), N is the set of arcs of T and S is the site of XML structure characterized by its DTD. Also, each node $e_i \in E$ is noted as $e_i\ (n_i, c_i)$ where n_i is the node name and c_i represents its cardinality, if any.

According to this notation, the tree of Figure 2 which belongs to the site S {Laboratory 1} is made up of a set of eight nodes E {Article, Title, (Author, +), (Section, +), Name, Affiliation, (Title, ?), (Para, +)} from which one is the root r {Article} and a set of seven arcs N {Article – Title, Article – Author, Article – Section, etc.}.

```
<! ELEMENT Article (Title, Author+, Section+) >
<! ELEMENT Section (Title?, Para+) >
<! ELEMENT Title (#PCDATA) >
<! ELEMENT Para (#PCDATA) >
<! ELEMENT Author (Name, Affiliation) >
<! ELEMENT Name (#PCDATA) >
<! ELEMENT Affiliation (#PCDATA) >
```

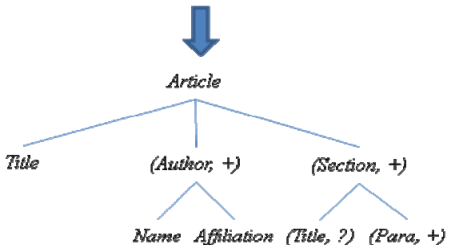


Figure 2: An example of DTD and its tree.

4.1 Generation of Unified Trees

This generation compares all trees resulted from the previous step (i.e., tree representation step) and then produces a unified tree set; i.e., trees not yet validated by the decision-maker. It is composed of the three following sub steps:

- Semantic processing of the tree nodes.
- Similarity calculation.
- Unified trees production.

The semantic processing resolves the semantic ambiguities of the node names, belonging to several trees, by using the *Wordnet* ontology. To do so, nodes having the same meaning are detected and their names are substituted with a unique/standard name. Secondly, for the obtained trees, we define a *similarity factor* that helps to determine which trees are analogous and, therefore, have to be merged. Finally, we produce a unified tree for each set of similar trees according to the similarity factor.

4.1.1 Similarity Calculation

Once all XML document structures are translated into trees, we need to build a unified tree by fusion; that is a global structure for the initial document set. To realize this fusion, we define a *similarity factor* (noted *Sim*) that measures the pertinence of trees to be integrated; its calculation is based on the number of the common nodes between two trees.

The similarity factor of two trees T_i (E_i, r_i, N_i, S_i) and T_j (E_j, r_j, N_j, S_j) noted $Sim(T_i, T_j)$ is calculated according to formula (1).

$$Sim\left(T_i, T_j\right) = \begin{cases} 0.75 \ If \ n_i = c_{i,j} \ and \ n_i < n_j \\ \dfrac{c_{i,j}}{q} \ otherwise \end{cases} \quad (1)$$

Where:

$n_i = |E_i|$, $n_j = |E_j|$, $c_{i,j} = |E_i \cap E_j|$ et $q = n_i + n_j - c_{i,j}$

In this formula, the similarity factor calculation is based on n_i, n_j and c_{ij} representing, respectively, the number of nodes in trees T_i, T_j and the number of their common nodes.

Now, given a set of n trees, we need to determine which trees have to be merged first; i.e., which trees are more closely in structure than others. For this purpose, we define the *Similarity Matrix* (noted *SM*). It is inspired from the matrix presented in (Feki, 2004) used for multidimensional schema integration. Our SM matrix is a triangular matrix with n trees in rows and in columns. Each cell *SM(i,j)* contains the similarity factor *Sim(i,j)* of tree in row i and tree in column j. This matrix facilitates the search of the trees having the highest *Sim*. We note that the comparison of two trees T_i and T_j is meaningful when $Sim(T_i, T_j)$ is greater than a given *threshold* determined by experimentation.

4.1.2 Generation Principle

The unified trees production step merges each two trees identified in the previous step.

In the literature, (Golfarelli and al., 1998) and (Golfarelli and Rizzi, 1999) propose algorithms to treat the attribute tree so as to generate an XML schema from an XML cube. The attribute tree may be pruned and grafted in order to eliminate the unnecessary levels of detail. Likewise, (Boussaid and al., 2006) use the merging operators: *pruning* and *grafting* in order to compare XML document trees and the tree of the multidimensional conceptual model.

In fact, pruning is carried out by removing portions of the trees. Figure 3 (a) shows an example of pruning the two trees T_1 and T_2. The result tree T_3 represents their merged tree. It contains only the common sub-trees to T_1 and T_2. The uncommon nodes are dropped with their sub-trees, if any.

Grafting is used when sub-trees have not the same structure. Figure 3 (b) depicts an example of grafting for T_1 and T_2. In these trees, the sub-trees b-(e, f) of T_1 and b-(e, x, f) of T_2 don't have the same structure (also c-(y, g) and c-(g)). In the result tree (i.e., T_3), the common nodes and their arcs are maintained while the other nodes are dropped. However, the application of the pruning and grafting operators leads to the intersection of trees.

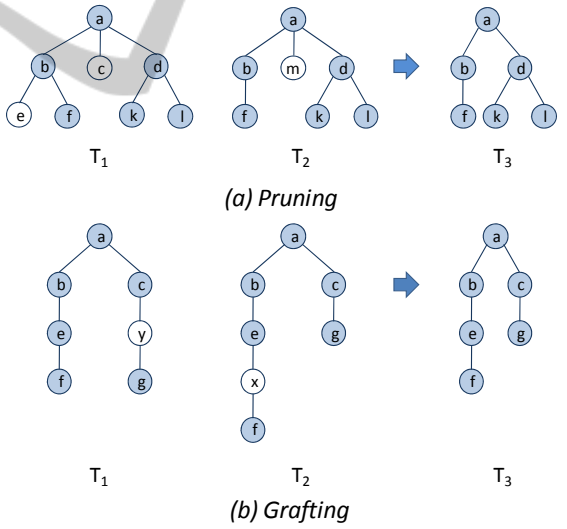


Figure 3: Example of merging trees through pruning (a) and grafting (b).

Let us recall that in our work we want to build a common structure that acts as the global schema in distributed databases. This structure is obtained by unification of XML tree structures issued from several distributed sites. For this unification we need a set of operators: We retain only the merging operator *grafting* as defined in (Golfarelli and al., 1998), (Golfarelli and Rizzi, 1999) and reused in

(Boussaid and al., 2006). Note that the pruning eliminates nodes/sub-trees and therefore is not useful because it leads to eliminate sub-trees. In addition, we propose the following three operators: *fusion by inclusion*, *fusion by union of sub-trees* and *fusion by merging nodes*.

Definition 1. The *fusion by inclusion* ***F-Inclusion*** operator of two trees T_1 and T_2 produces a tree T_3 such as $T_3=T_2$ if $T_1 \subseteq T_2$ and $T_3 = T_1$ if $T_2 \subseteq T_1$. Its syntax is as follows:

F-Inclusion $(T_1, T_2) = T_3$

Input:
- T_1 (E_1, r_1, N_1, S_1).
- T_2 (E_2, r_2, N_2, S_2).

Conditions:
- $T_1 \subseteq T_2$ or $T_2 \subseteq T_1$.

Output:
- $T_3 = T_2$ if $T_1 \subseteq T_2$.
- $T_3 = T_1$ if $T_2 \subseteq T_1$.

Figure 4 (a) gives an example for such a fusion.

Definition 2. The *fusion by union* operator ***F-Sub-Trees*** *of two sub-trees* produces a tree T_3 composed of sub-trees belonging simultaneously to T_1 and to T_2.

F-Sub-Trees $(T_1, T_2) = T_3$

Input:
- T_1 (E_1, r_1, N_1, S_1).
- T_2 (E_2, r_2, N_2, S_2).

Conditions:
- $E_1 \cap E_2 \neq \varnothing$.
- $\exists\ e_i \in E_1$ and $e_j \in E_2\ \forall\ e_i = e_j$, $Parent(e_i) = Parent(e_j)$ and $Child(e_i) \neq Child(e_j)$.

/* Parent (e_i) returns the parent of node e_i */
/* Child (e_i) determines sub-trees of node e_i */

Output:
- T_3 (E_3, r_3, N_3, S_3) with
- $E_3 = E_1 \cup E_2$
- $N_3 = N_1 \cup N_2$
- $r_3 = r_1 = r_2$
- $S_3 = \{S_1, S_2\}$

Figure 4 (b) illustrates this operator.

Definition 3. The *fusion by merging nodes* ***F-Merging-Nodes*** produces a tree T_3 composed by a specific node.

F-Merging-Nodes $(T_1, T_2) = T_3$

Input:
- T_1 (E_1, r_1, N_1, S_1).
- T_2 (E_2, r_2, N_2, S_2).

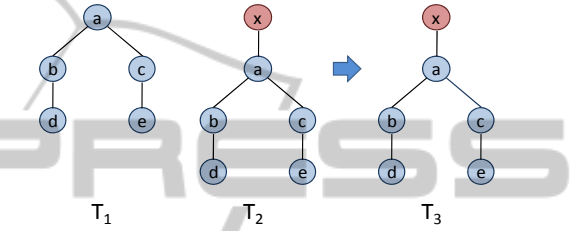
Conditions:
- $E_1 \cap E_2 \neq \varnothing$
- $\exists\ e_i \in E_1$ and $e_j \in E_2\ \forall\ e_i \neq e_j$, $Child(e_i) = Child(e_j)$ and $Parent(e_i) = Parent(e_j)$.

Output:
- T_3 (E_3, r_3, N_3, S_3) with
- $E_3 = (E_1 - \{e_i\}) \cup (E_2 - \{e_j\}) \cup (e_i | e_j)$

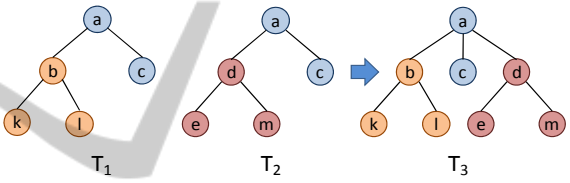
/* ($e_i | e_j$) is a specific node or */
- $N_3 = N_1 \cup N_2$
- $r_3 = r_1 = r_2$
- $S_3 = \{S_1, S_2\}$

Figure 4 (c) shows an example where the two sub-trees b-(e, f) and d-(e, f) become the sub-tree b|d-(e, f) linked to their original common node *a* in T_3. Note that b|d signifies a node named b or d.

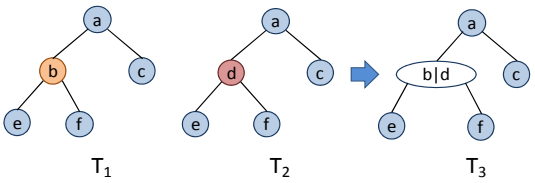
While merging trees, cardinalities are treated in respect to rules defined in (Hachaichi and al., 2010).



(a) Fusion by inclusion



(b) Fusion by union of sub-trees



(c) Fusion by merging nodes

Figure 4: Merging trees through fusion by inclusion (a), fusion by union of sub-trees (b) and fusion by merging nodes (c).

4.1.3 Algorithm to Merge Trees

In order to produce unified tree(s), we propose a *Tree-Merge* algorithm. It uses the similarity matrix to identify trees to be merged, merge them using the operators defined in section 4.1.2 and then produces unified trees. The input and output trees are stored according to the meta-model of Figure 5 where we associate to each node of unified-trees its origin; i.e., one or several sites from which it issues. This association is fundamental because it prepares for querying the distributed DocW.

Algorithm Tree-Merge

```
Input
T = {T1, T2 …, Ti …, Tn}: a non empty
set of n trees where Ti = (Ei, ri, Ni, Si)
Threshold: a real value in [0.5..1[
Output
R = {R1, R2, Rm}: m(m ≤ n) merged trees.
Begin
R := T
Calculate_SM (T) /* calculates the
similarity matrix SM [n, n] */
Size_SM := n
For each i ∈ [1..(Size_SM - 1)] do
 Max := Determine-Max(SM)
 If (Max ≥ Threshold) Then
  For each j ∈ ]i..Size_SM] do
   If (SM[i,j] = max) Then
    Mark row i and column j
     For each K (i<k<j) do
      If (SM[k,j] = max) Then
       Mark row k
      End If
     End For
   End If
  End For
  R = R - {trees corresponding to rows
  and columns marked}
  /* Merge trees corresponding to rows
  and columns marked */
  If (Ti ⊆ Tj or Tj ⊆ Ti) Then
   R := R ∪ F-Inclusion (Ti, Tj)
  Else
   If (Ei ∩ Ej ≠ ∅ and (∃ ei ∈ Ei
   and ej ∈ Ej and ei = ej and
   Parent(ei) = Parent(ej) and
   Child(ei) ≠ Child(ej) )) Then
    R := R ∪ F-Union-Sub-trees(Ti,Tj)
   Else
    If (Ei ∩ Ej ≠ ∅ and (∃ ei ∈ Ei
    and ej ∈ Ej and Child(ei) =
    Child(ej) and Parent(ei) =
    Parent(ej) and ei ≠ ej)) Then
     R := R ∪ F-Merging-Nodes(Ti,Tj)
    End If
   End If
  End If
  Delete rows and columns marked
  Calculated-SM (R)
  Size_SM := |R|
 Else
  Stop
 End If
End For
End.
```

4.2 Validation of Unified Tree(s)

In this third and last step (cf. figure 1), the decision-maker/designer should validate the unified tree(s) by adjusting them to his/her analytical requirements: delete and/or rename nodes. The result tree(s) are saved in a specific repository (cf. figure 5). This repository is later used in querying the distributed DocW.

5 EXAMPLE

In this section, we apply our method on an example of four DTDs (cf. Figure 6) issued from four distributed sites.

Applying the first step, each DTD is translated into a tree (cf. Figure 7). Secondly, semantic ambiguities are resolved by using the *Wordnet* ontology and produces trees shown in Figure 8. Thus, in tree *T1 Writer* is replaced with its synonym *Author* much more present in the other trees. Also, the node *Section* (in T_3) is changed into *Paragraph*. Similarly, *Symposium* (in T_4) is standardised as *Conference*. After that, the similarity matrix is calculated in order to find trees to compare.

In our running example, we set the threshold value to 0.5.

The Tree-Merge algorithm iterations are:

First iteration:

Input: T = {T_1, T_2, T_3, T_4}
n = 4 ; Size_SM = 4

		T_1 1	T_2 2	T_3 3	T_4 4
T_1	1		**0.75**	0.12	0.12
T_2	2			0.16	0.27
T_3	3				0.66
T_4	4				

Since SM[1,2]= 0.75 (is greater than the threshold 0.5) then T_1 will be merged with T_2: T_1 is included in T_2 thus, the algorithm performs the *fusion by inclusion* operator that produces *T'* (which is similar to T_2). The result of this step is the set R = {T_3, T_4, *T'*} (cf. figure 9).

During this iteration, rows 1 and column 2 are marked in order to determine which trees to merge.

Second iteration:

Input: {T_3, T_4, T'}

		T_3 1	T_4 2	*T'* 3
T_3	1		**0.66**	0.16
T_4	2			0.27
T'	3			

As SM[1,2]= 0.66 (> 0.5), then T_3 will be merged with T_4 by applying the *fusion by merging nodes* operator. Indeed, the node (*Paper*) and (*Article*) have the same sub-trees in T_3 and T_4. In the

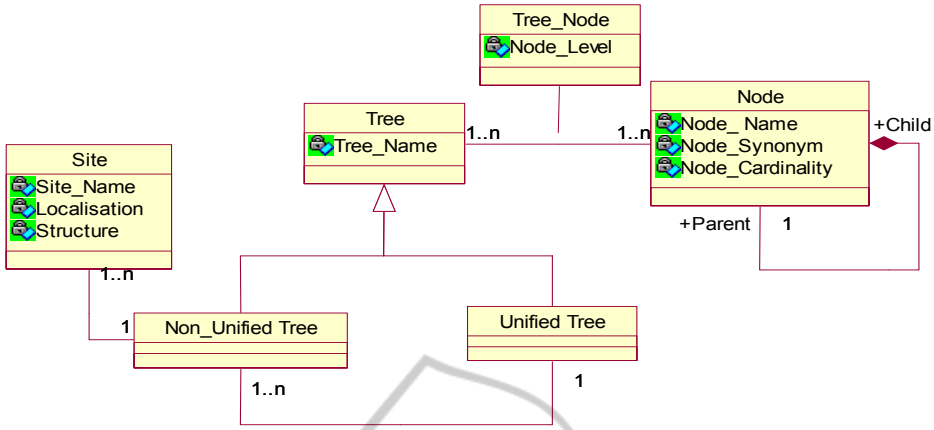


Figure 5: Class diagram for the storage of unified trees.

DTD 1	**DTD 2**
< ! ENTITY Affiliation (University | Industry) > < ! ELMENT Writer (Name, %Affiliation, Style) > < ! ELMENT Name (#PCDATA) > < ! ELMENT University (#PCDATA) > < ! ELMENT Industry (#PCDATA) > < ! ELMENT Style (#PCDATA) >	< ! ELMENT Publication (Title, Author + , Article) > < ! ELMENT Title (#PCDATA) > < ! ELMENT Article (Abstract, Body) > < ! ELMENT Abstract (#PCDATA) > < ! ELMENT Body (#PCDATA) > < ! ELMENT Author (Name, (University | Industry), Style) > < ! ELMENT Name (#PCDATA) > < ! ELMENT Industry (#PCDATA) > < ! ELMENT Style (#PCDATA) > < ! ELMENT University (#PCDATA) >
DTD 3	**DTD 4**
< ! ELMENT Paper (Title, Section +, Author +, Conference) > < ! ELMENT Title (#PCDATA) > < ! ELMENT Section (#PCDATA) > < ! ELMENT Author (#PCDATA) > < ! ELMENT Conference (#PCDATA) >	< ! ELMENT Article (Title, Author +, Paragraph +, Symposium) > < ! ELMENT Title (#PCDATA) > < ! ELMENT Author (#PCDATA) > < ! ELMENT Paragraph (#PCDATA) > < ! ELMENT Symposium (#PCDATA) >

Figure 6: Example of four DTDs.

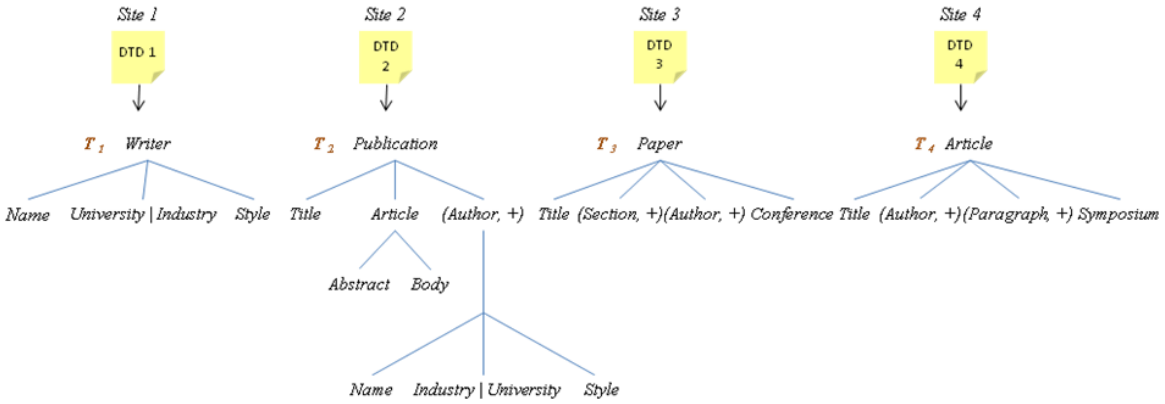


Figure 7: Generated trees for the four DTDs (of Figure 6).

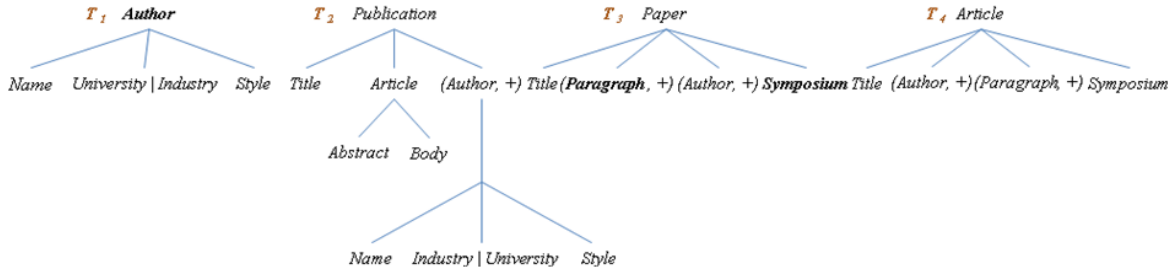


Figure 8: Trees after semantic processing of their nodes.

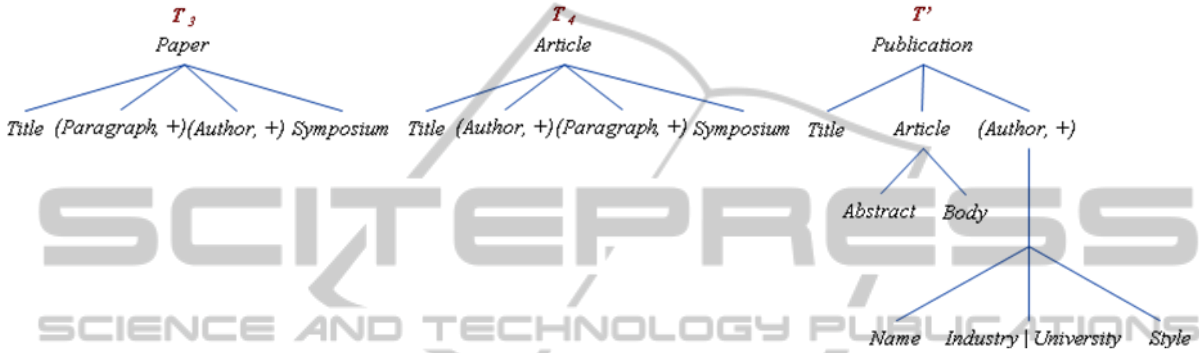


Figure 9: The result trees T_3, T_4 and T'.

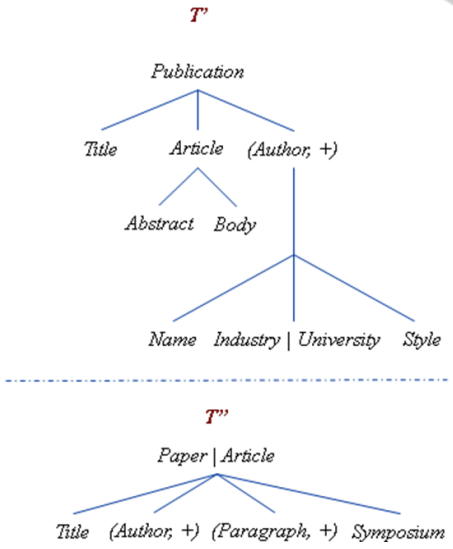


Figure 10: The result trees T' and T''.

result tree T'', the node (*Paper*|*Article*) appears. The set R is composed of the two trees T' and T'' (cf. figure 10).

Third iteration:

Input: {T', T''}

		T'	T''
		1	2
T'	1		0.16
T''	2		

Since SM[1, 2] = 0.16 (< 0.5), then there are no trees to merge. As a consequence, T' and T'' represent the final set of unified trees. As a final step of our method, these trees are presented to the decision-maker/designer in order to adjust them to their analytical requirements.

6 CONCLUSIONS

In this paper, our main concern was to enrich the data warehouse with textual data in order to improve the quality of decisional analysis. More specifically, we have proposed a method to build a document warehouse. This method is composed of two main steps: unification of XML document structures, and multidimensional modeling of documents. We have focused on the unification step which unifies heterogeneous XML document structures issued from several sites. To do so, we have translated XML structures into trees, solved semantic ambiguities of their nodes using *Wordnet* ontology, and then merged those trees producing unified tree. Finally, the unified trees are presented to the user in order to validate them according to his analytical requirements. We note that the unified trees represent a common structure that will be used later in querying documents.

Currently, we are developing a software prototype to support our proposed unification method. As a long term perspective for this work, we will propose a querying language for the distributed document warehouse.

REFERENCES

Ben Messaoud, I., Feki, J., Zurfluh, G., 2010. Unification des structures des documents XML pour l'entreposage de documents. In *ASD'10, Cinquième Atelier sur les Systèmes Décisionnels*, pages 1-12, ISBN 9973-9900-2-0, Sfax, Tunisie.

Ben Messaoud, I., Feki, J., Zurfluh, G., 2011. Modélisation multidimensionnelle des documents XML. In *EDA'11, 7ème journée francophones sur les Entrepôts de Données et d'Analyse en ligne*, Clermont Ferrand, France (To appear).

Boussaid, O., Ben Messaoud, R., Choquet, R., Anthoard, S., 2006. Conception et construction d'entrepôts en XML. In *EDA'06, 2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne*, Versailles, France.

Golfarelli, M., Maio, D., Rizzi, S., 1998. Conceptual Design of Data Warehouses from E/R Schema. In *HICSS'98, Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, pages 334–343, Washington, DC, USA.

Golfarelli, M., Rizzi, S., 1999. Designing the Data Warehouse : Key Steps and Crucial Issues. In *Journal of Computer Science and Information Management 2(3)*, pages 88–100.

Feki, J., 2004. Vers une conception automatisée des entrepôts de données : Modélisation des besoins OLAP et génération de schémas multidimensionnels. In *MCSEAI'04, 8th Maghrebian Conference on Software Engineering and Artificial Intelligence*, pages 473-485, ISBN 9973-37-193-3, Sousse, Tunisie.

Hachaichi, Y., Feki, J., Ben-Abdallah, H., 2010. Modélisation multidimensionnelle de documents XML centrés-données. In *Journal of Decison Systems*, vol 19/3, pages 313-345.

Inmon, W., H., 1994. *Building the Data Warehouse*. John Wiley & Sons.

Júnior, C., A., S., Mello, R., S., 2008. An ontology-driven process for unification of XML instances. In *Brazilian Symposium on Multimedia and the Web*, pages 242-249, Vila Velha, Brazil.

Lee, M., L., Yang, L., H., Hsu, W., Yang, X., 2002. XClust: clustering XML schemas for effective integration. In *CIKM'02, Proc. of the ACM International Conference on Information and Knowledge Management*, pages 292–299, McLean, Virginia,

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 23th Annual International ACM SIGIR Conference*, pages 363–365.

Pedersen, T., B., 2009. Warehousing The World: A vision for Data Warehouse Research. In *Kozielski S., Wrembel R. (Eds.): New Trends in Data Warehousing and Data Analysis*. Annals of Information Systems, Vol.3.

Pérez, M., J., M., 2007. *Contextualizing a data warehouse with documents*. Thèse de doctorat. Université Jaume I, Spain.

Pérez, M., J., M., Berlanga, L., M., R., Aramburu, C., M., J., Pederson, T., B., 2008. Contextualizing data warehouses with documents. In *Decision Support System (DSS)*, Elsevier, pages 77-94.

Ravat, F., Teste, O., Tournier, R., Zurluh, G., 2010. Finding an application-appropriate model for XML data warehouses. In *Information Systems*, volume 35, issue 6, pages 662-687.

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

Yoo, C., S., Woo, S., M., Kim, Y., S., 2005. Unification of XML DTD for xml Documents with Similar Structure. In *Computational Science and its Applications* – ICCSA, LNCS 3482, pages 954-963.