A database approach for the document data modeling: Application to the medical surveys

M. Ouziri

Laboratory for Information Systems Engineering
INSA of Lyon
Lyon, France

C. Verdier

Laboratory of Health System Analysis
UMR 5823
Lyon 1 University
Lyon, France

Abstract

In this paper, we propose two parts of the same projects. The first part concerns a new way to store and query documents. The main idea is to propose an automatic translation of XML document in a relational database based on the data semantics. The second part concerns the creation of the documents directly by the user (in HTML format and at the end, in XML format) with the help of a generic system.

Keywords XML document, generic model, ontology, database.

1 Introduction

The various statistics surveys are created and exploited by the organizations that manage them essentially in a manual manner. Indeed, there exists according to our knowledge no system able to build automatically surveys. The most current way to make surveys is described as follows: The documents concerning surveys are, generally, manually reproduced under application software or a word processing. The surveys data are captured and processed in statistics software or in spreadsheets. The origin of our study started from this noting. Our working group was regularly requested to take part in epidemiological surveys. In each study, it was necessary to define the electronic questionnaires and a software application to associate the questionnaires with a database able to memorize the information from the surveys. We thus propose in this work to show how it is possible to computerize these various tasks to make the surveys administrator free from the computer. The main idea is to keep him totally independent from the computer scientist. We supply him a generic tool of documents design for the medical surveys that is stored in a database. Our system thus allows:

- To create documents in XML format [1] (or HTML); in this system part, we propose to translate automatically the documents in XML code. This part is not presented in this paper.
- To generate Automatically the (relational) database schema in which the data are stored,
- To fill in the database with data captured in the documents,
- To query the database via the documents which are used for the creation of the schema. This last item will not be discussed in this article.

The central idea of the article is to communicate with the user only through documents for the operations of structure design, data entry, visualization and query of the data contained in the documents. Last, we use the global semantic represented by the ontology to generate the relational schema corresponding to the documents.

After the presentation of the related work in paragraph 2, we present in paragraph three, our proposition to generate a relational model from a XML document. Then we show how to create easily a document and we briefly present some screens of the software proposed. Finally we present the future prospects.

2 Related works

2.1 The document at the interface level

The computer scientist, if they tried to adapt the user to the computer tool were constrained under the influence and in some cases, the rejection, of the end-users (that was particularly the case for the medical information systems [2,3]) to adopt an opposite attitude. They tried to conform the computer tool to the professions. Several types of research have been elaborated concerning especially, *i*) the interfaces (by including the designers non computer scientists, the ergonomics, the graphic designers and the end-users notably), *ii*) the graphic query of the data and *iii*) the representation of the data in a document. These last two domains of research are looking for facilitating the electronic document using and so making the end-users more autonomous facing the computer tool. In the traditional databases, the documents are, in fact, visualization and data capture forms. It means that the document doesn't really exist; only its structure is real. The data are displayed in the document but are not stored in the document itself. They are stored in the database. That means that the querying and the handling of the data are done from the database and not from the document. The immediate consequence is that the database query languages are still tools for specialists. Researchers thought then about the improvement of this system by proposing some more convivial tools of data querying. Business Objects (BO) for example represents a integrated query system for the end-user [4]. The main idea of this tool is to transfer the computer scientist expertise toward the end-user. BO proposes a tool placed between the database and the user therefore. He chooses the attributes that he wishes to query by a simple click and move them in a suitable window. The system automatically generates the SQL code corresponding to the request. In this same way, we can mention the works of Zloof [5]. In spite of all, these tools remain complex and require a good experience to be used efficiently. One notes in practice that few users really master them and that the requests obtained don't translate still the question that the user would like to make.

2.2 The document at the storage level

Since XML [1] became the standard format of exchange and data representation and represent, therefore, the biggest mass of data stored in data sources or exchanged, numerous works are led today to develop data management and integration systems resulting from XML data sources. Works concerning the integration of the XML formalism to the database applications environment are developed around two big axes. The first consists in conceiving the data in the XML format and in developing a new generation of database, namely the XML document databases, derived from the semi-structured databases [6,7]. Indeed, following this research axis, the researchers chose to store the XML documents as they are, i.e. under their textual shapes (" tagged text "), and developed new query languages therefore, as WebSQL and WebOQL [12] for querying the web and XML-QL [8] or XQL [9] for XML documents query language. Most of these query languages are based on the OEM model [13] of the TSIMMIS project [14] in which a XML document is represented as an oriented labeled graph. The systems based on this approach are Lore [9,10] and Strudel [11] that are semi-structured (documents) data management systems and that were adapted later for the XML documents. The advantage of this model is that it is very well adapted for the semi-structured document representation in which schema and data are joined. On the other hand, the query processing in this model presents the most drawback in which the queries are valued using graph theory and heuristic in spite of the efforts that have been done to develop optimization techniques (Dataguides [15] of Lore). The second approach consists in modeling the XML documents using the relational (or object) data model. Generally, this approach requires a mediating layer that permits to extract the data from XML documents to fill in the database and to restore the XML documents from the database when asked. The advantage of this approach is the very reliable management of the data managed by a relational DBMS. It is based on a powerful relational algebra and uses some techniques at different levels (optimization, indexing, etc. offered by the DBMS).

2.3 Modeling the XML tree structure

In this approach, a XML document is considered as a oriented and labeled tree [16,17] in which the internal nodes represent the elements of the document, the leaves represent the data or the attributes and the edges formalize the relations of element-sub-element inclusion or element-attribute and are labeled by the name of the sub-element. There are several relational schemas to represent a tree. In the MONET data model for the XML documents [18], it represents for a document all the binary relations between all nodes of the tree and relations associating the nodes and their values or their attributes. A method of hybrid so-called storage has been developed in the NATIX system [19]. In this approach, a

XML document is seen as a tree structure and use an object storage model for the physical level in which the sizes of the nodes and pages disks are considered to optimize the storage of the document nodes. Two levels of tree structure are considered in [20] for modeling the internal links within a same XML document and the external links (inter-documents links). An object model is used for the storage of the nodes which are divided into two categories. The nodes of type elements represent the meta-model that serve for the navigation in the document and are stored in a cache memory to be reached in a faster way. The objects nodes that store the data (leaves of the tree) are preserved in big memories (magnetic disks or optic disks). In [21,22], Florescu and al presented six schemas for the representation of a graph with relational tables: three ways to represent the edges and two for the leaves (data).

This approach is relatively well adapted for the design of wrapper for the federation of heterogeneous data sources. Moreover, the rebuilding of the original document is highly reliable and is very simple (is thus done in a reduced time) for exchanging the contents of the base. We think that this model provides a good medium to establish an algebra, in particular, for the operations on the structure such as navigation in the documents, the union, the intersection, subtraction between documents, etc. This model treats a document XML as a tree structure by completely being unaware of the semantic aspect of the information contained in the documents. The updates of the database can produce semantic inconsistencies. In term of space of storage, for each document are stored information about the structure and the data. Two documents with the same structure and concerning the same context are stored independently and it does not consider any relation between them. The results is the duplication of the both documents structure.

2.4 Modeling the XML DTD

A DTD (Document Marks Definition) is a description format of the structure of a XML documents collection. In some applications, the lack of DTD makes difficult to extract the similarities between the structures of the XML documents and to find an optimal common schema. The generation of the database schema from a DTD is a simple and direct operation because the content of the DTD is a set of instructions defining the structure of the documents. The conception of a schema of a database from a DTD [18,23] can be made itself by several ways according to the data model to generate, namely the relational model [24], the object-relational [24,25] or the pure object model [26]. The procedure of relational schema generation from a DTD depends on the model targets.

One of XML features is that it has a lot of similarities with the object model [27]. The generation of the object schema from a DTD is very easy and is straight made. The inconvenience of this approach is bound to the performances of the object model. Indeed, the semantic contributions of the object model can't alone compensate its insufficient performances what stopped it from stabilizing on the commercial plan. However, the relational model (in use since the years 70), that represents a success due to its simplicity, its algebra on which it is built and due to the efficiency of the DBMS to implement it, has been solicited like model and storage support of the XML documents. It still shows its performance and its validity in spite of the emergence of new technologies information. But the relational model is insufficient to represent complex structures such as images, video, sound, etc.. So the system is improved by adding the abstract data types and the concept of object identifier resulting from the object-relational model. The procedures of relational schema generation and relational-object are presented in [28].

The DTD modeling approach requires the existence of a DTD and produces a database schema dedicated to store XML documents, validating the DTD used to generate this database. The use of this database is restricted with a dedicated application and cannot be used with documents containing various structures.

3 Our proposition for the document storage and design

We use the relational DBMS which offers a very effective storage system and a query language founded on algebraic operators that gives a 100% response to the requests in an optimized time. Nevertheless, the relational model suffers from some disadvantages related to the way in which it stores the data, namely the tables. To store complex data, several tables are necessary. Consequently, to recover the data of the complex objects, it is necessary to carry out joins between the tables concerned. Our aim is to reduce the number of joins by reducing the number of tables generated.

In our vision, we consider the two aspects that a XML document carries, namely the structure and the semantic of the data. Indeed, during the translation from the XML structure to the relational schema (phase of modeling), we take into account these two essential aspects of a XML document in order to establish a better correspondence between the two storage models. In addition, the generation of the schema is performed directly from the documents, without taking into account the DTD. So it widens the application field because it is not only applied to the documents with DTD but also to those that don't have any DTD.

3.1 Generation of a normalized relational schema

In this section, we will present a step for the automatic design of normalized relational schemas to store and to query the contents of XML documents. To generate the relational schema, we will follow the same step presented in [29] while respecting the semantic relations between the concepts of the documents. Indeed, the use of information of a semantic nature is necessary since the documents give only information on the structural relations between the entities. However, normalization is a semantic level task that can't be carried out only starting from the information extracted from XML documents. Semantics (in our case it concerns the total conceptual diagram implying all the terms of the domain) is represented by an ontology of the domain which can be implemented by using a definition language of ontologies [30] such as OIL or CLASSIC (based on the descriptions logic).

To generate the schema, we first create a relational table for each complex element (tag) at the first level in the tree representation of the XML document. To reduce the number of tables, the root element is often ignored because, in the majority of cases, the editors of XML documents add the root element to the document to improve its presentation. For a sub-elements *SE* of an element *E* in the XML document for which we have already created a table, the generation of the diagram goes on as follows:

- If the sub-element *SE* is of simple text type (PCDATA) and that the relation between *E* and *SE* is of type *(1, 1)* in the ontology, *SE* will become an attribute in the table of *E.*
- If sub-element *SE* is of text type (PCDATA) and that the relation between *E* and *SE* is of type *(1, N)* in ontology, we will create a table for *SE* and we connect the two tables by adding the key of the table of *E* like foreign key in the table of *SE*,

```
<patient nss=12>
      <name> <\>
      <firstname> <\>
      <adress> <\>
      <adress> <\>
<\patient>
```

Patient
nss
name
firstname

Adress
nss
adress

- If sub-element *SE* is of complex type (itself containing sub-elements) and that the relation between *E* and *SE* is of type *(1, N)* in ontology, a table for *SE* is created and the key of the table of *E* is integrated like foreign key in the table of *SE*,

```
<patient nss=12>
      <name> <\>
      <encounter>
            < physician >
            <date>
      <\ encounter >
      < encounter >
            < physician >
            <date>
      <\ encounter >
<\patient>
```

Patient
nss
name

- If sub-element *SE* is of complex type (containing itself sub-elements) and that the relation between *E* and *SE* is of type *(1, 1)*; in this case either we create a table for the sub-element *SE* which contains the key of the table *E* like foreign key or we connect directly the sub-elements of *SE* with the element *E* by putting the attributes of the table *SE* in the table of *E* and we consequently remove the table *SE.*

```
<patient nss=12>
      <name> <\>
      <father>
            <name> <\>
       <\father>
<\patient>
```

Patient
nss
name
parent-name

- If, in ontology, the relation between *E* and *SE* are of type *(N, N)*, a third table is created to produce a diagram in the third normal form.

```
< invoice n°=12>
    <cust-name> <\>
    <date> <\>
    <article code=F15>
        <design>
        <qty>
    <\article>
    <\article code=G01>
        <design>
        <qty>
    <\article>
<\patient>
```

Invoice
n°
cust-name
date

ArticlesofInvoice
n°
code
qty

Article
code
design

In this last case, we note that the *qty* element depends not only on its parent element *article* but also on the *invoices* element (grandfather). This relation is expressed in the ontology and can't be extracted from the documents.

To minimize the number of schema tables, we perform the optimization rules presented in [29] as follows. First, the root element of the XML document is eliminated from the database schema. It is only useful to have more readability and to have a unique root element in the document. The root element plays a main role in our procedure. It gives the global semantic information about the content of the document. Thus, we use it to select the part of the ontology which is used to generate the schema or to select the appropriate ontology when using domain ontologies. Secondly, some document elements are ignored (passed through) and their sub-elements are linked to the parents. For example, the first-name and last-name elements (see the following document) are attributes of the patient element. Thus, the name element is ignored (passed through) and its two sub-elements are considered as attributes of the patient element. This is extracted from the concept relationships of the ontology.

```
<patient nss=12>
    <name>
        <first-name>
        <last-name>
    <\name>
    <father>
        <name>
            <first-name>
            <last-name>
        <\name>
    <\father>
<\patient>
```

This rule can be applied at many levels. The first-name and last-name elements of the father element can be linked directly to the patient element by passing through the two elements name and father. This is possible because the constraint cardinality (provided in the ontology) between the patient concept and the father concept is 1-* (a patient has one and one father). The document result is represented as follows:

```
<patient nss=12>
    <first-name>
    <last-name>
    < father's first-name>
    < father's last-name>
<\patient>
```

4 A generic data model for the medical surveys

We said in the head of this article that we were requested to take parts to epidemiological surveys. We proposed, few years ago, a software called *Internet Enqueteur* based on a web architecture to create easily surveys. The main idea of the system is to propose a generic tool for surveys investigators. The user chooses (in a list) or creates the variables of the surveys (patient_name, glycaemia_rate, age, etc.) and the type of the variable (kg, mmol/l, etc.). He displays the variables in the screens and then, in the end of the process, validates its. So the document is ready. The system automatically fills in the database for the particular survey.

4.1 Concept of genericity

The main idea which underlies our approach is that the majority of the attributes used in a field (documents of surveys in our case) are reusable from one surveys to another. Studies showed that in a new data-processing application, only 20% of the programs are really new [31]. It is then possible to determine invariant trade and thus to obtain a concept of genericity per field. We consider three levels of genericity: global, by field and application. The reader could find more details concerning this approach in [32].

In the category of genericity per field, three actors intervene in the design of the document: the computer scientist who creates the generic data model, the expert of the field which creates the document and the end-user who fills in the document. Each actor has a level of expertise which allows the communication with the other actors: the computer scientist perfectly knows the technology and the data-processing, the expert must have a light knowledge of data processing and a very strong knowledge of his domain. The end-user can fill in the document and will be able to query it directly.

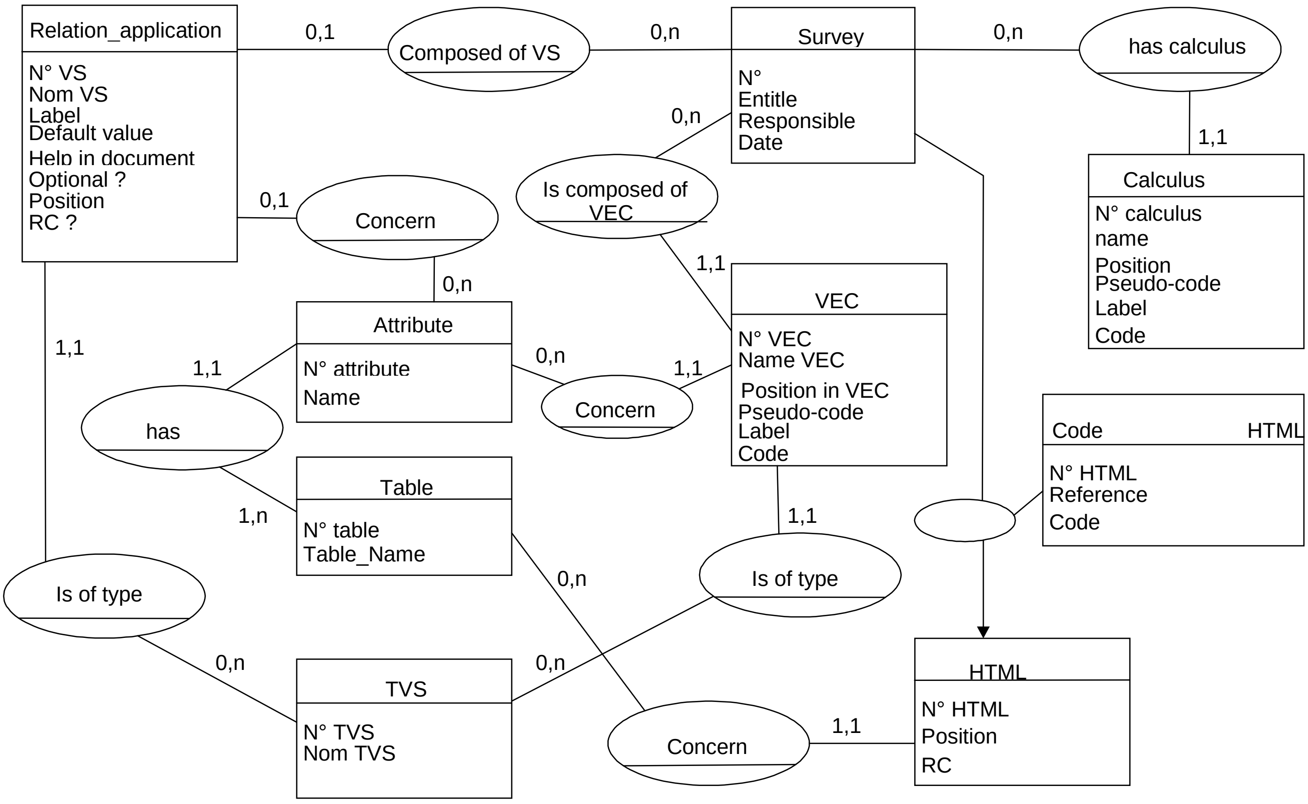

Figure 1 : The generic data model

4.2 generic data model design

The generic data model is conceived around the constraints of creations or following updates: meta database, documents creation, check procedure of data entry in the documents, storage of the values in the meta database, export of the data towards other application softwares. The system considerably simplifies the practice of the statistical surveys but also the structure handling and optimize the data acquisition.

The system stores the contents and the structures of the documents in the generic diagram (figure 1). Indeed, each survey document is associated to survey on which statistical calculations (specific to this study) can be carried out. The surveys document is composed of four parts including the variables of studies (*VS*) specific to each survey. The first relates to the fixed attributes of the patients (name, first name, date of birth, sex, etc). The second part corresponds to variable of studies related to a patient (smoker, glycemia rate, etc).

The third and the fourth part relate to the hospital centers concerning by the studies (fixed attributes and variable of studies). The central table of the model is the *relation_application* table. It represents the *VS*, their values and the way to present them in the document. A type (*TVS*) is associated for each *VS* by the investigator. Association C*oncerns* associates a *VS* to the part of the document of which it takes part of.

4.3 Example of data capture of a Survey

The designer of the surveys designs his document starting from the following screen (figure 2). The expert can create a new survey, duplicate an existing survey by removal of fields and visualize or modify a surveys. The variables of survey can be added, by the expert, for each part of the document (patient and centers variable) via a form. In this screen, the expert can reach several application: to create a new study, to duplicate an existing study (to carry out by field removing), to create and modify the questionnaires or to process statistics about his study.

Figure 2: Survey management

5 Conclusion

We think that the medical handling of information can be made in an effective way only if it is managed in a DBMS which offers a better effectiveness in terms of storage quality, speed and rate of answers to the queries. This led us to study the storage of the medical files, represented in XML documents, in a relational database. The lack of schema in XML documents led us to study the techniques of automatic relational schemas generation starting from XML documents. We show that the only document couldn't be used

to generate a normalized schema. We add to the system the knowledge base represented with ontology. This knowledge enables us to generate normalized schemas in the third normal form. To reduce the number of joins necessary during the evaluation of queries, we observe some rules concerning the optimization of the number of tables created. We also present a generic system for the conception of documents and for the storage of these documents. Today, we study the evolutionary aspects of the documents, particularly concerning the structure and the modeling in the databases. Secondly, we try to develop an algebra for the document processing and document querying designed with the today database technology.

[28] M. Klettke, H. Meyer. Managing XML Documents in Object-Relational Databases. *University of Rostock, Rostocker Informatik-Berichte,* No. 24, 1999.

[29] R. Bourret, C. Bornhövd, A. Buchmann. A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. *IEEE,* 2000.

[30] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner. Ontology-Based Integration of Information : A Survey of Existing Approaches. *Submitted to IJCAI 2001 Workshop: Ontologies and Information Sharing.*

[31] M. McClure. The Three R's of Software automation: Reengineering, Repository, Reutilisability Englewood Cliffs. *NJ: Prentice Hall*, 1992.

[32] F. Laforest. Generic Models: a new Approach for Information System Design. *In 3rd int. Congress Biwit' 97,* Biarritz, 189-1196, 1997.