IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4567

XML Storage in Relational Databases: An Approach Combining Description Logic and Statistics

Mourad Ouziri, Christine Verdier
Laboratoire d'Ingénierie des Systèmes d'Information (LISI)
7,av Jean Capelle, 69621 Villeurbanne Cedex, France
{mouziri, cverdier}@lisi.insa-lyon.fr

ABSTRACT

We propose in this paper to jointly use the description logic (DL) and probabilistic approaches to store documents in relational databases. The description logic is used to generate a first database schema by reasoning capabilities over the conceptual part of the documents. The resulting schema is normalized. Indeed, we add to the DL knowledge base, which represents the documents, some ontological assertions specified in the DL formalism. Then probabilistic calculus is used to optimise the generated database schema by computing some statistical measurements over the extensional part of the documents.

1. INTRODUCTION

Document is the main information support in most organizations. It is used as data source to store, visualize and exchange data. It represents the most useful information support for the human beings and particularly the end-users. Data in documents are semi-structured. This means that document structure is not given, the data types are not defined and the data structure may be absent, irregular, implicit or partial. These features make the description and the manipulation of data easy. Databases are based on models which store typed and well structured data (see Kappell, 2001) for a comparison of the relational data model and the semi-structured data). Thus, it is difficult to store semi-structured data in a conventional database management system.

In this paper we propose to create a correct database schema for semi-structured data. We use the relational model which is the most efficient and mature system in database technology to store and query large amount of data. A normalized schema is produced from the conceptual level of the document. To obtain this schema, we use two tools: the description logics (Borgida 1995) to compute the database schema and some statistical measurements to optimize this generated schema.

The combination of the description logic reasoning and probability allows us to create a semantically correct database schema.

We consider two main groups of related works: the schema modeling from the XML tree structure and from the DTD (Document Type Definition) structure. The second paragraph presents our theory: the construction of the schema and its optimization. A running example is given at the end of this paper.

2. RELATED WORKS

2.1. Modeling the XML Tree Structure

In this approach, a XML document is considered as an oriented and labeled tree (Buneman, 1997 and Abiteboul, 1997) 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. A tree is represented by several relational schemas. In the MONET data model (Schmidt 2000), for a XML document, the tree represents all the binary relations between all the nodes of the tree and the relations associating the nodes and their values or their attributes. A method of hybrid so-called storage has been developed in the NATIX system (Kanne, 1999). In this approach, a XML document is seen as a tree structure. An object storage model is used for the physical level in which the sizes of the nodes and pages disks are considered to optimize the storage of the document nodes. In (Florescu 1999), the authors presented six schemas to represent the document graph into relational tables: three for the edges and two for the leaves .

This approach is relatively well adapted for the design of wrappers for federating heterogeneous data sources. Moreover, the rebuilding of the original document is highly reliable and is very simple for exchanging the contents of the database. It is done in a reduced time. We think that this model provides a good medium to establish an algebra, especially for the operations on the structure such as the navigation in the documents, the union, the intersection, the subtraction between documents, etc. This model considers a XML document as a tree structure. The syntactic aspects are obviously treated but the semantic ones aren't.

2.2. Modeling the XML DTD

A DTD (Document Type Definition) is a XML structure description format. In some applications, the lack of DTD makes difficult to extract similarities between the structures of different XML documents and to find an optimal common schema. The generation of the database schema from a DTD is a simple and direct process: the content of the DTD represents a set of instructions defining the structure of the documents. The database schema design from a DTD (Schmidt 2000, Shanmugasundaram 1999) can be made differently according to the data model to generate, namely relational model (Bourret 1999, Klettke 1999), object-relational model (Klettke 2000) or pure object model (Christophides 1994).

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

2.3. Description Logic Approach

The description logic (DL) is a reasoning formalism at the conceptual level. It provides users with various inference capabilities that allow them to deduce implicit knowledge. Extending of database systems with logics capabilities has been studied by different authors (Brodie 1989, Hacid 2000, Gõni 1995). In (Calvanese 1999), a XML DTD is represented in the DL formalism as a tree structure using the *f* and *r* fillers to respectively specify the first and the rest of an element. The *TBOX* is generated from the rules definitions by creating the assertions from the DTD elements and rules. The reasoning functionalities (instance validation, DTD inclusion, disjunction, equivalence, etc.) are then performed on the knowledge base. In (Hacid 2000), an algorithm of DL-based knowledge base generation is presented. It is generated from a XML document and a database schema inference. This algorithm creates a concept for each node and its definition assertion is a concatenation of all sub-elements assertions until the leafs. The database schema is generated by calculating the *lcs* (Least Common Subsumer) of *k* objects, where *k* is a threshold.

3. NORMALIZED SCHEMA GENERATION AND OPTIMISATION

The system proposed is at the junction of the DL and the XML tree structure approaches. The DL approach offers a good tool to formalize the knowledge, a well-known structure to extract semantic constraints. So, the system is based on DL and probability to generate a normalized and optimized relational schema to store and to query the XML documents collection. To generate a normalized schema, semantic relations between the documents concepts are necessary since the documents give only information on the structural relations between the concepts (entities). However, normalization is a semantic level task that can't be carried out only starting from the information extracted from the XML documents. Semantics is represented by a domain ontology that is merged as assertions in the DL-based knowledge base (KB) which is created from the XML documents. The semantics is here extracted from the total conceptual diagram implying all the terms of the domain.

The system is working as the following. For each document, the only intentional part of a KB is created. The KB is developed in the DL formalism. The *lcs* (least common subsumer) of each equivalent concept [Cohen, 1992] are computed over the KB. These concepts are deduced from the ontology and DL reasoning. The ontological assertions are taken into account because they are useful to generate a normalized schema. Then, the schema generated is optimized by reducing the number of generated tables. This is done by computing the Bayes probability on the documents instances. The figure 1 presents the general architecture of the system.

3.1. Normalized Schema Generation

To generate the relational schema, the procedure presented in (Hacid 2000) is extended with the addition of the semantic relations between the concepts of the documents. Indeed, the use of information concerning the semantic nature is necessary because documents give only information on the structural relations between the concepts. Semantics is represented by a domain ontology. Thus, the assertions generated from the XML documents representing the intentional definitions of their concepts are merged with those extracted from the ontology representing the semantic relationships between the documents concepts.

The DL-based knowledge base is created following these steps (see [Baader 1991] for the syntax and the semantic of the terminological constructors used):

Step 1: Create a knowledge base from the XML documents collection.

For a document leaf *l*, we add to the KB the assertion : $l \subseteq$ Type where *Type* $\in$ *[String, Number, Date]* is the concept representing the type of the concept *l* extracted from the ontology.

Starting with the leafs and for a concept node *C*, we insert to the KB the assertion $C \subseteq \forall C_l \Pi \ldots \forall k. C_k$ for each outgoing arc labeled *l,…, k* of the node *C* such that $l \subseteq C_l$ and $k \subseteq C_k$ are in the KB (see the paragraph 4).

Step 2: Using the synonym relationships given by a domain ontology, we insert to the KB the assertions $A \equiv C_A$ for each concept *A*, where C_A is the canonical term of all synonym terms. For example, if *Patient* and *Human* are two terms in the documents concepts and if we suppose that in the domain ontology, the canonical term of all terms designing a person is *Person*, thus, we add to the knowledge base the assertions:

Human ≡ Person Patient ≡ Person

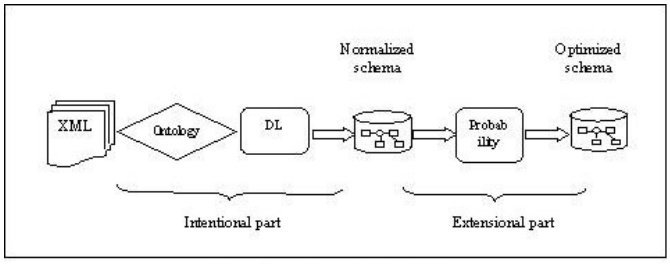

Figure 1. Schema generation process

Step 3: for the *(N, N)* entity relationships, we add to the KB the assertions relating the concepts and specifying the common attributes. For example, *Patient* and *Doctor* are two entities linked by the *(N, N)* association *Exam* which has an attribute *result*. Our reasoning algorithm creates a third relational table which links the two tables corresponding to *Patient* and *Doctor* and contains the common attribute *result*. With this end, we add to the KB this ontological assertion:

Exam $\subseteq \forall$ examined. Patient $\Pi \forall$ examiner. Doctor $\Pi \forall$ result. String Π (=1 examined) Π (=1 examiner) Π (=1 result)

and we modify the descriptions of *Patient* and *Doctor* by respectively adding, the inverse of the roles *examined* and *examiner* (noticed $examined^{-1}$ and $examiner^{-1}$) as follows:

Patient $\subseteq \ldots \Pi \forall$ examined^{-1}. Exam Doctor $\subseteq \ldots \Pi \forall$ examiner^{-1}. Exam

Step 4: now, the KB contains all the needed assertions representing the documents concepts and their semantic relationships. So we can compute the first relational schema from the KB. Its optimization will be presented in the next paragraph. The relational schema corresponds to the most specific structure for each equivalent concepts. To get the equivalent concepts, we build a graph *G (V, A)* where each node (*n*) represents a concept (*C*) and for each equivalence (≡) relationship between two concepts C_i and C_j, an edge is created from n_i to n_j, corresponding to concepts C_i and C_j respectively. The equivalent concepts are obtained by computing connected sub-graphs of *G*. For each connected graph, we compute the *lcs* concept of the concepts represented by its nodes. A relational table is created for each *lcs* concept (*Patient* and *Exam*); we call this table *primary table*. Its attributes correspond to *lcs* concept roles (without inverse roles). We create tables also for the roles which do not participate in the *lcs* such as the roles *address* (of *Patient*) and *doctor* (of *Exam*). We call these tables *secondary tables*.

3.2. Schema Optimization

A schema optimization is used to reduce the number of the tables generated at the previous steps. The normalized relational schema is generated from the intentional part of the KB (*TBOX*) and the optimization operation is based on the statistical calculus performed over the extensional part (*ABOX*). This process is based on the data distribution (from the documents collection). The roles of the secondary tables are put back in the primary tables and the secondary tables are deleted from the DB schema. The process is performed as follows. For each role, we compute two statistical values: the conditional probability and repetition frequency of a generated table attribute. A conditional probability is calculated for each attribute of the secondary tables. If this attribute probability tends towards 1 then the attribute is put in the primary table and deleted from the secondary table. The second measure calculated is used in the inverse sense of the first one. If the repetition frequency of an attribute in a primary table is greater than a threshold, then this attribute is deleted from its primary table and we create a secondary table for it. The two tables are linked by a foreign key.

3.2.1. Conditional Probability

This measure is computed for each attribute of secondary tables. This measure is aimed to reduce the number of tables that are generated during the attributes transfer between the primary and secondary tables. The primary table attributes can contain some null-value tuples when the attribute is not an attribute of the *lcs*. If the number of null-value of an attribute is not large compared to the size of its primary table then its is transferred from the secondary table to the primary one. Therefore, there will be less joints when evaluating queries. The conditional probability, *P(<child> / < parent>)*, to have an element *child* as sub-element (child element) of another element *parent* (parent element) expresses the percentage of the element *parent* having the element *child* as sub-element, over the documents collection. This probability is computed using the Bayes theorem. Thus,

$$P(<child> / <parent>) = \frac{P(<child>.<parent>)}{P(<parent>)}$$

The element probability is calculated as:

$$P(<element>) = \frac{\text{occurrence number of } <element>}{\text{number of documents}}$$

by simplification we write,

$$P(<child> / <parent>) = \frac{\text{occurrence number of } <child> \text{ as sub-element of } <parent>}{\text{appearance number of } <parent> \text{ in the collection}}$$

we formulate it as:

$$P(<child>/<parent>) = \frac{\sum S(<child>, <parent_i>), \forall i \in [1, n]}{n}$$

where *n* is the occurrence number of the element <*parent*> over the documents collection, $<parent_i>$ is the i^{th} occurrence of the element and *S* is a function defined as:

$$S(<child>, <parent_i>) = \begin{cases} 1 & \text{if } <child> \text{ appears at least once in } <parent_i> \\ 0 & \text{else} \end{cases}$$

This probability is a correlation measure of an element <*child*> with its parent <*parent*>. For example, if *P(<address>/<Patient>)* is greater than a threshold (near to 1), then the attribute which corresponds to <*address*> is put back from the secondary table (*Patient1*) to the primary table (*Patient*).

3.2.2. *Repetition Frequency*

This statistical measurement is used jointly with the conditional probability for the attributes of the primary tables. Some elements can appear more than once in their parent elements. This corresponds to the *(*, N)* cardinality between the elements. To avoid redundancy storage of the relational database tuples, these elements are stored in separated tables from their parents tables. For example, if the element *doctor* of *Exam* appears more than once, this statistical measurement is computed for this element to decide if we create or not a separated table for the attribute *doctor*. The computation of this measurement is performed for a child attribute over all the XML documents collection. For an element <*child*> associated to an element <*parent*>, the repetition frequency (*FR)* of <*child*> represents the average of repetition number of the child element in its parent element over all the collection. It is computed as follows:

$$FR(<child>, <parent>) = \frac{\sum Nb(<child>, <parent_i>), \forall i \in [1, n]}{n}$$

where *n* is the occurrence number of the element <*parent*> over the documents collection, $<parent_i>$ is the i^{th} element instance and $Nb(<child>, <parent_i>)$ designs the number of elements <*child*> in the element <*parent*>. If *FR(<child>, <parent>) > threshold* then we create a separate table to store the attribute corresponding to <*child*>.

The thresholds values depends on the database size and mainly on the attributes access frequency. In the figure 2, we present a complete algorithm for generating a normalized relational schema for XML documents collection.

The thresholds values depends on the database size and mainly on the attributes access frequency. In the figure 2, we present a complete algorithm for generating a normalized relational schema for XML documents collection.

Input: Semi-structured graph $G = (V, A, r)$
Output: Normalized relational schema

```
KB = GenKB (G);              // generate the knowledge base of G
Ge = equivalenceGraph(KB); // construct the concepts equivalence graph
Cs = connected (Ge);        // compute the connected sub-graphs of Ge
for each cs_i ∈ Cs          // compute the lcs of the equivalent concepts
    lcs = lcs ∪ computelcs(cs_i);
    construct the non lcs concepts : nonlcs
end for
for each concept c_i ∈ lcs   create a table t_ c_i_lcs;
for each concept c_i ∈ nonlcs   create a table t_ c_i_nonlcs;
// Schema optimization
for each t_c_i_nonlcs
    for each attribute a_i of t_ c_i_nonlcs
        p_i = P(a_i / c_i)    // conditional probability
        if (p_i  > threshold)
            delete a_i  from t_ c_i_nonlcs;
            add a_i to t_ c_i_lcs ;
        end if
    end for
end for
for each t_ c_i_lcs
    for each attribute a_i of t_ c_i_lcs
        f_i = FR(a_i / c_i)   // repetition frequency
        if (f_i >> threshold)
            delete a_i  from t_ c_i_lcs;
            add a_i to t_ c_i_nonlcs ;
        end if
    end for
end for
```

Figure 2. A normalized relational schema generation

4. RUNNING EXAMPLE

In this section, we apply the algorithm presented in the previous section to a XML documents collection represented by the two structures of the opposite documents structures.

The knowledge base generated for these XML documents (extended with the ontological assertion *Exam*) is :

<Patient n°> <name> <\> <name> <\> <address> <\> <Doctor id> <name></> <result></> <\Doctor> <\Patient>	<Patient n°> <name> <\> <Doctor id> <name></> <speciality></> <result></> <\Doctor> <\Patient>

Patient1$\subseteq \forall$ name.String $\Pi \forall$ address. String $\Pi \forall$ doctor. Doctor1 $\Pi \forall$ examined^{-1}. Exam Π ($\geq$ 1 name) Π (=1 address) Π ($\geq$ 1 doctor)
Patient2 $\subseteq \forall$ name. String $\Pi \forall$ doctor. Doctor2 $\Pi \forall$ examined^{-1}. Exam Π (=1 name) Π ($\geq$ 1 doctor)
Doctor1 $\subseteq \forall$ name. String $\Pi \forall$ result. String $\Pi \forall$ examiner^{-1}. Exam Π (=1 name) Π (=1 result)
Doctor2 $\subseteq \forall$ name. String $\Pi \forall$ specialty. String $\Pi \forall$ result. String $\Pi \forall$ examiner^{-1}. Exam Π (=1 name) Π (=1 result) Π (=1 specialty)
Exam $\subseteq \forall$ examined. Patient $\Pi \forall$ examiner. Doctor $\Pi \forall$ result. String Π (=1 examined) Π (=1 examiner) Π (=1 result)
We compute now the lcs of equivalent concepts (manually created):
Patient = lcs (Patient1, Patient2) $\subseteq \forall$ name. String $\Pi \forall$ doctor. Doctor $\Pi \forall$ examined^{-1}. Exam Π (= 1 name) Π ($\geq$ 1 doctor)
Doctor = lcs (Doctor1, Doctor2) $\subseteq \forall$ name. String $\Pi \forall$ result. String $\Pi \forall$ examiner^{-1}. Exam Π (=1 name) Π (=1 result)

The primary tables are Patient (n°, name), Doctor (id, name) and Exam (n°, id, result). The secondary tables are Patient1(n°, address) and Doctor2 (id, speciality). To optimize this schema, we have calculated these probabilities over numerous XML documents:
P(<address>/<Patient>) = 0.98, then we bring back the address attribute to the table Patient and thus we delete the secondary table Patient1 form the schema.
P(<speciality>/<Doctor>) = 0.8, the domain experts says that the attribute specialty is essential, thus, it may be queried frequently. To avoid many joints, we bring back it to its primary table, Doctor.
Fr(<name>/<Patient>) = 1.05, then the name attribute is kept in its primary table. The final schema is given as: *Patient (n°_pat, name, address), Doctor (id_dct, speciality), Exam (n°_pat, id_dct, result)*.

5. CONCLUSION

We think that the handling of the 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 queries processing. For our application domain, this led us to study the storage of the medical files, represented in XML documents, in relational databases. 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 only the document couldn't be used to generate a normalized schema. We add a semantic knowledge to the system, represented by a domain ontology. This knowledge enables us to generate a normalized schema in the third normal form. We have used some artificial intelligence techniques namely, description logic, which allows us to use many reasoning services at the conceptual level. A knowledge base in the description logic formalism is created from the XML documents and is merged with the ontological assertions to express all the semantic information about the documents. We use some probability measurements over the documents to reduce the relational tables number. Consequently, we optimize the queries evaluations.

6. BIBLIOGRAPHY

Abiteboul S. Querying semi-structured data. *In Proceedings of ICD,* 1997.

Baader F., Hanschke, P. A Scheme for Integrating Concrete Domains into Concept Languages. *IJCAI 1991: 452-457, 1991*

Borgida A. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering, 1995.*

Bos B. "The XML Data Model". *http://www.w3.org/XML/Datamodel.html, 1999.*

Bourret R., Bornhövd C., Buchmann A. A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. *Technical report DVS99-1 Dept of CS, Darmstadt univ, Germany 1999.*

Brodie M.L. Future intelligent information systems: AI and database technologies working together. *In J. Mylopoulos and M.L. Brodie, editors, Readings in Artificial Intelligence and Databases, p 623-640. Morgan Kaufmann, 1989.*

Buneman P. Semi-structured data. *In Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1997.*

Calvanese D., De Giacomo G., Lenzerini M. Representing and reasoning on XML documents: A description logic approach. *Journal of Logic and Computation, 9(3): 295-318, 1999.*

Christophides V., Abiteboul S., Cluet S., Scholl M. From Structured Documents to Novel Query Facilities. *In Proc. Of ACM SIGMOD Conf. On Management of Data, Minnesota, 1994.*

Cohen W., Borgida A., Hirsh H. Computing least common subsumers in description logics. *In Proc of AAAI-1992, p 754-760.*

Florescu D., Kossmann D. A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database. *Technical Report, INRIA, France, 1999.*

Göni A., Blanco J.M., Illarramendi A. Connecting knowledge bases with databases: a complete mapping relation. *In Proc. of the 8th ERCIM Workshop. Trondheim, Norway, 1995.*

Hacid M-S, Soualmia F., Toumani F. Schema Extraction for Semistructured Data. *Proc. of the 2000 Int Workshop on DL, Aachen, Germany, p 133-142, 2000.*

Kanne C., Moerkotte G. Efficient Storage of XML Data. *Technical Report, University of Manheim, Germany, 1999.*

Kappell G., Kapsammer E., Retschitzegger W. ML and Relational Database Systems – A Comparison of Concepts. *In Int Conf on Internet Computing (IC'2001) Las Vegas, USA.*

Klettke M., Meyer H. XML and Object-Relational Databases Systems: Enhancing Structural Mappings Based On Statistics. *WebDB (Informal Proceedings) 2000.*

Schmidt A., Kersten M., Windhouxer M., Waas F. Efficient Relational Storage and Retrieval of XML Documents. *In International Workshop on the Web and Databases, Dallas TX, USA, 2000.*

Shanmugasundaram J., Tufte K., He G., Zhang C., DeWitt D., Naughton J. Relational Databases for Querying XML Documents: Limitations and Opportunities. *In Proc of the 25th VLDB 1999 Conference, p. 302-314, Edinburg, Scotland.*