Programming and Computer Software, Vol. 29, No. 3, 2003, pp. 1–7. Translated from Programmirovanie, Vol. 29, No. 3, 2003.

Axiomatization of the Evolution of XML Database Schema

S. V. Coox

Department of Mathematics and Mechanics, St. Petersburg State University, Universitetskaya nab. 7/9, St. Petersburg, 199034 Russia
e-mail: qx@acuid.com
Received January 29, 2003

Abstract—The schema of a database consists of constructs that model relationships between its entities. Changes made to the schema with time are called the schema evolution. An axiomatic model of the XML database schema is suggested that automatically maintains its integrity when basic changes are made to the schema. A classification of changes is described.

1. INTRODUCTION

Recently, XML has been gaining popularity not only as a language for developing Web pages and exchanging data between various components of Web applications, but also as a means for storing data with a not strictly defined structure [1]. XML also provides access to data using SQL-like queries [2]. Therefore, we can consider XML databases or, more generally, bases of *semistructured data*. Presently, XML is the most convenient tool for manipulating data that have no strict structure. By an XML database [3], we mean a set of valid XML documents [4] together with the definition of their types called Data Type Definition (DTD). The set of DTD documents is called the database schema; it is an analog of the table structure in relational databases.

When dealing with any data storage, we inevitably face the problem of database evolution. Although the format of the data remains the same (all the data are text documents), changes in the schema are almost inevitable. In the case of XML databases, complicating the matter is the fact that the documents often reside on different servers maintained by different teams; therefore, it cannot be guaranteed that the schemas of such documents will remain invariable or be modified consistently.

Typical changes in an XML document schema are the addition and deletion of tags, modification of the tag nesting model, and similar operations for attributes of a tag. A complete list of all possible changes of the data schema is presented in Section 4. Methods for manipulating those changes are also considered in that section. As a rule, a change in the data schema significantly affects the system as a whole. Particular attention must be given to the two following problems:

1. **The semantics of changes.** How the changes in the schema affect the structure of the data storage?

2. **Change propagation.** This means making changes to documents and queries of existing applications such that those documents and queries remain admissible.

To solve the first problem, a system of schema invariants and a set of rules are usually used such that the application of those rules preserves the schema consitency. This approach was used in system for schema evolution support in object-oriented databases [5, 6]. We systematize the changes in XML data schemas with regard for our model and establish rules for manipulating the changes.

A typical solution to the second problem is an explicit modification of the documents so that they correspond to the new data schema. Another solution is to assign to each document a set of operations that should be applied every time when it is accessed so that it appears as a valid document in the framework of the new schema.

In this paper, we consider the first problem in the general form; namely, we study the maintenance of XML data schema evolution using the model described below. In Section 3, we introduce a set of axioms and require that the model of any correct data schema satisfy these axioms. A classification of changes in a data schema and rules for its modification are given in Section 4. In Section 5, we discuss related studies.

The data model and the set of axioms suggested in this paper are an implementation of the ideas used in the TIGUKAT system [7] for semistructured data.

2. DATA MODEL

Before describing the model, we make some notes concerning the syntax and semantics of XML; specifically, of the part of XML used in DTD. We will use the schema presented in Fig. 1 as a working example.

First, when we consider XML documents as a data storage used by application programs, we can neglect certain features of XML that are immaterial in this case. For example, the order of the subtags *Humidity*, *Temp*, and *Wind* of the tag *Weather* (see Fig. 1) has no effect

0361-7688/03/2903-0001$25.00 © 2003 MAIK “Nauka / Interperiodica”

```
<!ELEMENT Document (Country|Region)*>
<!ELEMENT Country (Region|City)+>
  <!ATTLIST Country name CDATA #REQUIRED>
<!ELEMENT Region (City*,Weather*)>
  <!ATTLIST Region name CDATA #REQUIRED>
<!ELEMENT City (WheatherReport*|Hotel+)>
  <!ATTLIST City name CDATA #REQUIRED>
<!ELEMENT Hotel (Room+)>
  <!ATTLIST Hotel name CDATA #REQUIRED>
  <!ATTLIST Hotel address CDATA #REQUIRED>
<!ELEMENT Weather (Humidity,Temp,Wind)>
<!ELEMENT Humidity (#PCDATA)>
<!ELEMENT Temp (#PCDATA)>
<!ELEMENT Wind (#PCDATA)>
```

Fig. 1. An example of DTD.

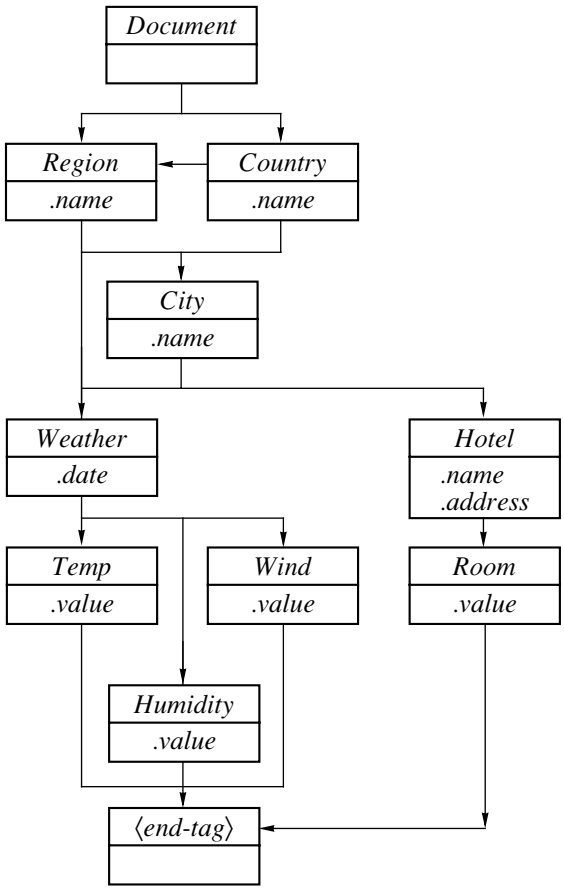

Fig. 2. The graph model of DTD.

on the relationships between the tags (all three tags are included in the tag *Weather* and do not affect each other). However, a document with a different order of those tags is not a valid XML document [4].

Second, the XML syntax admits regular expressions that can impose nontrivial constraints on the simultaneous occurrence of subtags. For example, the construct `<!ELEMENT F ((t,foo*)|bar*)>` subordinates the tags *foo* and *bar* to the existence of the tag *t* in the XML document. However, does it affect the access to the content (data) of this document? It turns out that this does not affect the data at all. We still can write the XML-QL query (see [2])

```
WHERE <F> <foo>$foo</></>
 IN "test.xml"
CONSTRUCT <res><foo>$foo</></>
```

A similar query can be written for *b*ar. Thus, we can iterate through all occurrences of these tags if there are any.

This example shows that the essential elements of DTD are tags with their attributes and their hierarchy. This can be represented in the form of a directed graph in which vertices are tags described in the DTD and edges are inclusion relations (from the ancestor to the descendant). This graph has a distinguished root vertex (the root element of the DTD). We add another constraint: the graph must be acyclic. In a real-life XML document, this constraint can be violated due to two reasons.

1. A recursive tag definition

```
<!ELEMENT Rec (Rec*, Simple+)>
```

We will ignore such edges when constructing the graph, since the inclusion of a tag into itself does not affect the other tags of the document.

2. Indirect recursion

```
<!ELEMENT foo (bar*,…)>
<!ELEMENT bar (foo*,…)>
```

We do not consider such schemas in this paper. In [1] fact, I could not discover any practical example in which the indirect recursion was necessary.

As a result, any DTD in the XML database not containing the indirect recursion can be assigned a acyclic directed graph with a distinguished root vertex. In this graph, vertices correspond to tags described in the DTD and edges correspond to inclusion relations between tags. Every vertex is assigned a set of attributes of the corresponding tag, which actually are the data stored in the document (simple tags containing only string data will be considered as tags with a single attribute called *value* of type *#PCDATA*). We also add to the graph a dummy end vertex <*end-tag*>, which is included in every leaf tag. The graph shown in Fig. 2 corresponds to the DTD shown in Fig. 1.

The graph thus constructed is suggestive of the type hierarchy of an object-oriented database. This is quite natural, since the tags of an XML document can be considered as objects and tag attributes as properties of those objects. An important difference from the set of types is the absence of the inheritance mechanism in a pure form. However, if tag attributes are considered together with the attributes of containing tags, we can speak of a set of attributes corresponding to the former tag. For example, we can consider the set of attributes

{*City.name*, *Weather.date*, *Temp.value*} for the tag *Temp*, which describes the temperature on a given day in a certain city.

3. AXIOMATIZATION OF CHANGES

It was noted above that typical changes in the data schema of an XML document are the addition and deletion of tags, changes in the tag nesting model, and similar operations for attributes of a tag. The effect of such changes on the hierarchy of tags and the migration of attributes are subject of rigorous analysis, since the introduction of changes in the schema must retain the integrity of the data and keep the schema as close to its initial design as possible.

When dealing with the model, we will use the notation presented in Table 1 (a detailed description is given below). We show how this notation can be used to described the model of a well-formed data schema and define a set of axioms that unambiguously determine the behavior of the schema under modifications.

We denote by $\mathcal{T}$ the set of all tags of the schema and, simultaneously, the corresponding graph. An example of the schema used in the following discussion is shown in Fig. 2.

The hierarchy of vertices of the graph $\mathcal{T}$ is determined by the sets of *immediate predecessors* $P(t)$ for every tag t. An immediate predecessor of t is a tag that can directly contain t, but no transitive inclusion of t is possible. In Fig.2, the tag *Region* can be contained in *Document* both directly and through *City*; therefore, $P(Region) = \{City\}$.

Essential predecessors $P_e(t)$ are the predecessors that must be present in the path from the root to the tag t to retain the semantics of the data. This set, as well as the set $N_e(t)$, is determined by the author of the schema. If the tag s is included in $P_e(t)$, it must remain a predecessor of t under all modifications of the schema except for its deletion from $P_e(t)$ or the deletion of s from the schema. There is a single constraint imposed on the set $P_e(t)$: $P(t) \subset P_e(t)$; i.e., all immediate predecessors of t are essential. Suppose, we declare $P_e(Hotel) = \{City, Region\}$ in the schema in Fig. 2 and the tag *City* is removed from the system. Then, the tag *Region* becomes an immediate predecessor of *Hotel*, since it is its essential predecessor. However, the tag *Country* is not included in $P(Hotel)$, since it was not declared essential.

The *graph of predecessors* $PL(t)$ of the tag t is defined as the subgraph of $\mathcal{T}$ consisting of all tags from which t can be reached, including t itself. It is seen that $PL(t)$ possesses all the properties of the initial graph; namely, this is a directed acyclic graph with a single root vertex (T) and a single end vertex (t). For example, $PL(Weather) = \{Weather, City, Region, Country, Document\}$.

Table 1. Notation of the model

Expression	Description
$\mathcal{T}$	The complete graph of tags
$s, t, T, \perp$	Tags
$P(t)$	The set of immediate predecessors
$P_e(t)$	The set of essential predecessors
$PL(t)$	The graph of predecessors
$N(t)$	The set of native attributes of the tag t
$H(t)$	The set of inherited attributes of the tag t
$N_e(t)$	The set of essential attributes of the tag t
$I(t)$	The set of all attributes of the tag t (interface)
$\alpha_x(f, \mathcal{T}')$	The function f applied to all elements of $\mathcal{T}'$

Native attributes $N(t)$ of the tag t are the attributes defined in this tag. For example, $N(Room) = \{price, space\}$.

The set of *inherited attributes* $H(t)$ of the tag t is the union of the attributes of all tags from which t can be reached (not including t itself). For example, $H(Wind) = \{Weather.date, City.name, Region.name, Country.name\}$ contains attributes of all tags that can occur in a real document on the path from the root to the tag *Wind*. It is easily seen that the sets of native and inherited attributes of any tag do not intersect.

Essential attributes $N_e(t)$ of the tag t are the attributes that are declared essential for this tag by the author of the schema (similarly to the set of essential predecessors). This set consists of all native attributes and, possibly, some inherited attributes $[N(t) \subseteq N_e(t)]$. The inclusion of an attribute into this set implies that it must remain an attribute of t (either native or inherited) as long as possible under all modifications of the schema. For example, if $N_e(Weather) = \{date, City.name, Region.name\}$ and the tag *Region* is removed from the schema, then the tag *Weather* must be given an optional tag *RegionName*.

The *interface* $I(t)$ of the tag t is the set of all its attributes (the union of $N(t)$ and $H(t)$).

Table 2 shows how the sets described above can be represented in terms of $P_e(t)$ and $N_e(t)$, which are specified by the author of the schema. All modifications of the schema can be also described in terms of these sets. The sets $P_e(t)$ and $N_e(t)$ can be specified and changed either by the user or by the system. For example, when a new tag is defined, the system can prompt the author of the schema to specify the sets of essential predecessors and attributes. Alternatively, the system can assume that all predecessors and attributes are essential or declare essential only the minimally admissible sets $P(t)$ and $N(t)$. The developer of the schema can modify it by adding or deleting tags and attributes and adding or deleting the inclusion relations of tags. These operations modify P_e and N_e; the sets P and N are insufficient for this purpose. For example, when the tag t is included

Table 2. Axiomatic model

Axiom of closeness	$\forall t \in \mathcal{T}, P_e(t) \subseteq \mathcal{T}$	(1)
Axiom of acyclicity	$\forall t \in \mathcal{T}, t \notin \cup \alpha_x(PL(x), P(t))$	(2)
Axiom of the root	$\exists T \in \mathcal{T}, \forall t \in \mathcal{T} \backslash T \in PL(t) \wedge P_e(T) = \emptyset$	(3)
Axiom of the end tag	$\exists \perp \in \mathcal{T}, \forall t \in \mathcal{T} \backslash t \in PL(\perp)$	(4)
Axiom of immediate predecessors	$\forall t \in \mathcal{T}, P(t) = P_e(t) - \cup \alpha_x(PL(x) \cap P_e(t) - \{x\}, P_e(t))$	(5)
Axiom of the ancestor graph	$\forall t \in \mathcal{T}, PL(t) = \cup \alpha_x(PL(x), P(t)) \cup t$	(6)
Axiom of the tag interface	$\forall t \in \mathcal{T}, I(t) = N(t) \cup H(t)$	(7)
Axiom of native tag attributes	$\forall t \in \mathcal{T}, N(t) = N_e(t) - H(t)$	(8)
Axiom of inherited attributes	$\forall t \in \mathcal{T}, H(t) = \cup \alpha_x(I(x), P(t))$	(9)

into the tag s, it is always added to $P_e(t)$; however, it is added to $P(t)$ only if $s \notin PL(t)$.

In the axiomatic model, we will use *operations applied to all elements* of a certain set; they are denoted by $\alpha_x(f, \mathcal{T}')$. This means that the unary operation f is applied to all elements of the set $\mathcal{T}' \subseteq \mathcal{T}$ and the results of its application are included in the resulting set. The variable x is bound in the expression in parentheses and it runs through the entire set $\mathcal{T}'$; the other variables in this expression are free. If $\mathcal{T}'$ is empty, the expression results in an empty set. In the functional notation, α_x can be considered as a λ function $\lambda_x.f$ applied to $\mathcal{T}'$.

Table 2 presents nine axioms that describe the schema in terms of P_e and N_e. The validity of these axioms implies that changes to the database schema are automatically supported (it is sufficient to make proper modifications to the sets P_e and N_e). Axioms (1)–(4) determine the structure of the graph (acyclicity and the uniqueness of the root and end vertices); the five other axioms determine the sets *P*, *PL*, *I*, *N*, and *H*.

1. Axiom of closure. Tags from $\mathcal{T}$ are subtags of the tags from $\mathcal{T}$ (i.e., the set $\mathcal{T}$ is closed).

2. Axiom of acyclicity. Ensures that the graph is acyclic.

3. Axiom of root. There exists a unique tag *T* (in the example, this is *Document*) that contains all other tags of the system. This axiom can be relaxed when the schema is a forest. In this case, every connected element of the schema can be considered individually and must satisfy this axiom.

4. Axiom of the end tag. There exists a unique tag $\perp$ that is a descendant of every tag of the system. This dummy tag was added to the schema so that the leaf vertices would not be distinguished as a special class of vertices.

5. Axiom of immediate predecessors. Immediate predecessors of a tag are those and only those essential ancestors from which it is impossible to reach the desired tag through other tags. By this axiom, one can automatically construct the set of immediate predecessors if the sets of essential ancestors of tags $P_e(t)$ are given.

6. Axiom of the ancestor graph. The graph of ancestors of the tag *t* includes the tag *t* and recursively all ancestors of all its immediate predecessors. By this axiom, the set $PL(t)$ can be automatically constructed for any tag.

7. Axiom of the tag interface. The interface of a tag (the set of its attributes) consists of the union of the sets of native and inherited attributes. By this axiom, the set $I(t)$ can be automatically constructed for every tag.

8. Axiom of the native tag attributes. The native attributes of a tag are its essential attributes that are not inherited. By this axiom, the set of native attributes $N(t)$ can be automatically constructed for any tag.

9. Axiom of inherited attributes. The set of inherited attributes of a tag is the union of the interfaces of its immediate predecessors. By this axiom, the set of inherited attributes $H(t)$ can be automatically constructed for any tag.

Using the graph in Fig. 2 as an example, we explain how this set of axioms works.

Axioms 1, 3, and 4 are evidently valid (all tags belong to the set $\mathcal{T}$, T = *Document*, and $\perp$ is the dummy *<end-tag>*, which was added to the system.

Axiom 2 (acyclicity): none of the tags is an ancestor of any its immediate predecessor.

Axiom 5 (immediate predecessors). Assume that the following sets of essential ancestors are given:

$$P_e(Weather) = \{Region, City\},$$
$$P_e(City) = \{Region, Country\},$$
$$P_e(Region) = \{Country\},$$
$$P_e(Country) = \{Document\}.$$

Verify that the set of immediate predecessors $P(Weather) = \{City\}$. Indeed,

$$X = P_e(Weather) - \cup\alpha_x(PL(x) \cap$$
$$\cap P_e(Weather) - \{x\}, P_e(Weather)) =$$
$$= \{Region, City\} - \cup\alpha_x(PL(x) \cap$$
$$\cap \{Region, City\} - \{x\}, \{Region, City\}) =$$
$$= \{Region, City\} - ((PL(Region) \cap$$

$$\cap \{Region, City\} - \{Region\}) \cup (PL(City) \cap$$
$$\cap \{Region, City\} - \{City\}))$$
$$PL(Region) = P\{Document, Country\}$$
$$PL(City) = \{Document, Country, Region\} \Rightarrow$$
$$\Rightarrow X = \{Region, City\} - (\emptyset \cup \{Region\}) =$$
$$= \{City\} = P(Weather).$$

The set of axioms provides an automatic mechanism for calculating the new structure of the data schema upon modifying the sets of the essential ancestors $P_e(t)$ or the essential attributes $N_e(t)$. Changes in one of these sets are the basis for the schema evolution, and the axiomatic model makes it possible to represent the relationships between tags and their attributes in terms of those sets; this, in turn, can be used for the description and comparison of schemas (and, as a consequence, of the schema evolution).

Theorem. The set of axioms described above is complete and consistent; i.e., if the sets $P_e(t)$ and $N_e(t)$ for every tag are specified that satisfy the conditions described above, then any well-formed schema satisfies the set of axioms, and modifications of the well-formed schema are controlled by the given sets.

Proof. First, we prove that any well-formed schema satisfies the set of axioms.

Axiom 1 is evidently valid.

Axiom 2. $\circledR t \cong \mathcal{T}$, $t \notin \cup\alpha_x(PL(x), P(t))$. The membership $s \in PL(x)$ means that there exists a path from s to x. Assume, the contrary, that there exists an $x \in P(t)$ such that $t \in PL(x)$. Then, there exists a path in the graph from t to x. Consequently, there is a path from t to t, and the graph contains a cycle, which implies that the schema is not well-formed.

Axioms 3 and 4 are valid for any well-formed schema due to the connectedness of the graph.

Axioms 5 and 6. The proof is by induction on the maximum length of the path from the root to the vertex under consideration. For the root vertex T, Axioms 5 and 6 are evidently valid. Assume that the following equalities hold for any vertex for which the maximum path from the root is no longer than k (due to the acyclicity of the graph, there are no infinite paths):

$$P(s) = P_e(s) - \cup\alpha_x(PL(x) \cap P_e(s) - \{x\}, P_e(s))$$
$$\text{and } PL(s) = \cup\alpha_x(PL(x), P(s)) \cup \{s\}.$$

Consider a tag t for which the maximum path from the root is $k + 1$. In particular, this means that the set $P(t)$ consists of tags for which the length of any path to them does not exceed k (and for at least one of them is exactly k).

$PL(t)$: For all tags from $P(t)$, Axioms 5 and 6 are valid; therefore, the desired set contains the set $P(t)$ and all tags with no direct path to t. It remains to verify that the tags that directly include t but do not belong to $P(t)$ are also included; in other words, these are the tags that include t either directly or through other tags (q in Fig. 3 is an example of such a tag). However, all such paths go through $P(t)$; therefore, there exists a tag $p \in P(t)$ such that $PL(p)$ contains q. It remains to notice that the set $\cup\alpha_x(PL(x), P(t)) \cup \{t\}$ does not contain redundant elements due to the acyclicity of the graph.

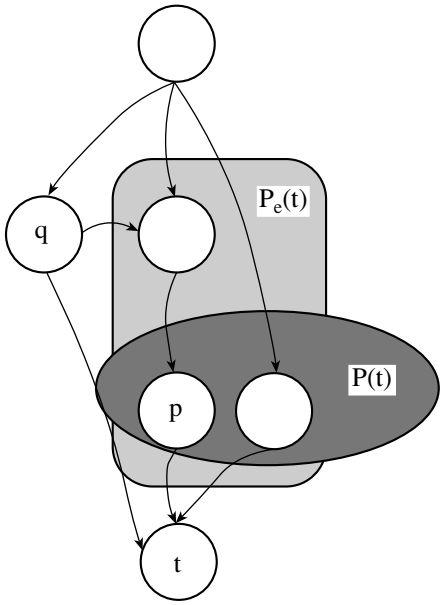

Fig. 3. The sets of essential and immediate predecessors.

$P(t)$: Let us show that all elements of $P(t)$ belong to the set $P_e(t) - (PL(x) \cap P_e(t) - \{x\}, P_e(t))$. Indeed, $x \in PL(x) \cap P_e(t) \circledR x \in P(t)$ and $x \notin PL(s)\ \forall s \in P_e(t) - P(t)$ (otherwise, $x \notin P_e(t)$). It remains to show that the desired set does not contain redundant elements; i.e., $\forall s \in P_e(t) - P(t)\ \exists x \in P_e(t) | s \in (PL(x) \cap P_e(t) - \{x\})$. Such an x exists, since there exists a path from s to t through intermediate tags.

Axioms 7, 8, and 9. Again, the proof is by induction on the maximum length of the path from the root to the vertex under consideration. The basis of induction is obvious. Assume that Axioms 7, 8, and 9 are valid for any vertex for which the maximum path from the root is no longer than k. Consider a tag t for which the maximum path from the root is $k + 1$. We show that $H(t) = \cup\alpha_x(I(x), P(t))$. By the inductive assumption, all three axioms hold for every element of $P(t)$. It remains to notice that the union of $H(x)$ over $x \in P(t)$ is equal to the union of $H(x)$ over $PL(t)$. Thus, Axiom 9 holds, Axiom 8 follows from the fact that the set N_e includes all native attributes of the tag, and Axiom 7 follows from Axioms 8 and 9.

In the next section, we show that the modifications made to well-formed schemas are controlled by the sets $P_e(t)$ and $N_e(t)$.

4. SCHEMA EVOLUTION

In this section, we classify the simplest modifications of the database schema and demonstrate how these modifications can be expressed in terms of the axioms discussed in Section 3. The issue of propagation of the schema modifications to concrete documents remains open for future study.

When discussing schema modifications, we will mean modifications made to the model described in Sections 2 and 3. These modifications can be of one of two types: modification of the internal structure of a tag (i.e., of its attributes) and modification of the inclusion relationships between tags. Thus, we arrive at the following set of schema modifications and the corresponding changes in the sets P_e and N_e (by default, we denote by t the tag being modified and by a the attribute being added or deleted).

1. Modifications of the tag structure

1.1. Addition of an attribute. The attribute being added is included into $N_e(t)$, and the sets N_e, N, and H for the tag t and for all tags that are reachable from t are recalculated. It is also useful to allow the schema designer to add this attribute to the set N_e of some of its descendants at the designer's discretion.

1.2. Deletion of an attribute. The attribute being deleted is excluded from $N_e(t)$, and the sets N_e, N, and H for the tag t and for all tags that are reachable from t are recalculated. It should be noted that if $t \in P(s)$ and $a \in N_e(s)$, then, by Axioms 8 and 9, the attribute a will be added to $N(s)$.

2. Modifications of the data structure

2.1. Addition of a direct tag inclusion relation. Assume that we are adding the inclusion relation of the tag t into the tag s. Then, s is added to $P_e(t)$. If the acyclicity axiom is violated in the process, then such a modification must be rejected by the system. It should be noted that the tag s is added to $P(t)$ only if there is no other path from s to t.

2.2. Deletion of a direct tag inclusion relation. This is a rather complicated operation, since it can result in the appearance of new edges leading from certain ancestors of the deleted tag to its descendants according to Axiom 5. Assume that we are deleting the tag s from $P_e(t)$. Then, s is removed from $P_e(t)$ and all expressions that depend on $P_e(t)$ are recalculated. If this results in the violation of the root axiom, such a modification must be rejected. (Alternatively, we could include t into the root T, but such a modification is made in two steps: T is added to $P_e(t)$ (operation 2.1) and then the relation $s \longrightarrow t$ is deleted (operation 2.2). Similarly, the axiom of the end tag for s can be violated; however, if $s \in P_e(\perp)$, then we can simply add the edge $s \longrightarrow \perp$.)

2.3. Addition of a tag. A new tag t is added to the set $\mathcal{T}$. The new tag must be assigned a set $P_e(t)$; accordingly, the edges incoming into this set will be added, and the tag t will be added to $P_e(\perp)$ in order to satisfy Axiom 4. In the simplest case, $P_e(t) = \{T\}$, and the modifications of $P_e(t)$ can be made using operations 2.1.

2.4. Deletion of a tag. This is the most complicated operation: first, the tag t must be deleted along with its incoming and outgoing edges; then, edges that connect its ancestors with its descendants may be added; and, third, the attributes of t that are essential for its descendants must be expanded over the schema. This modification is made in three steps: first, operation 1.2 is applied to all attributes of t, then all outgoing edges of t are removed (operation 2.2), and, finally, all incoming edges of t are removed.

5. RELATED WORKS

Problems related to the database schema evolution regularly attract attention of researches (see [8, 9]). For relational databases, this problem is solved by DBMS manufacturers (such as Oracle). For object-oriented databases, there are such systems as TIGUKAT [6] and Orion [10]. However, for XML databases, the case is much worse. XML database developers try to solve this problem individually for every particular case.

The problem of supporting the evolution of the object-oriented database schema seems similar to the problem considered in this paper due to the hierarchical data structure. While Orion is a specific solution of the problem based on a system of constraints (invariants) imposed on the system and a set of admissible operations, TIGUKAT suggests a general solution and shows how the system of invariants used in Orion fits into the axiomatic model. The axiomatic model suggested in this paper is an implementation and elaboration of the ideas used in TIGUKAT for semistructured data.

6. CONCLUSIONS

The axiomatic model suggested in this paper provides a general solution to the problem of supporting the evolution of an XML database schema. However, along with evident advantages (it is axiomatized, and an automatic method for changing the schema is available), this model has certain drawbacks; the main drawback is that there is no complete correspondence between the model and the original DTD document. In the scheme described in this paper, the information about relationships between tags of the same level is lost (see the example in Section 2), as also is the information about how many times a subtag is included in the enclosing tag and the information about the types of attributes. In addition, the suggested model does not support the indirect recursion, which can be important in some cases.

It is seen that not every DTD document can be represented in the framework of the model discussed, and not all modification made in the DTD have an adequate projection on this model. The approach suggested in this paper should be considered as a basis for the development of a system for supporting the evolution of XML data schemas.

There is also an interesting and important problem of automation the comparison of similar schemas and the representation of one of them in the form of a list of simple modifications of the other schema resulting in the former schema. The solution of this problem can significantly simplify the automatic indexing of various XML pages.

The problem of expanding the changes of the schema to existing XML documents remained unsolved in this paper. This problem is relatively simple (once the changes have been decomposed into the simplest ones) if the existing documents can be modified. However, if documents should be modified "on the fly," the problem becomes much more interesting. The problem of automated modification of queries that existed at the time when modifications in the database schema were made arises as well.

REFERENCES

1. Rath, H.H., XML: Chance and Challenge for Online Information Providers, *Ph. D. dissertation*, http://www.xml.org/xml/xml_chance_challenge.shtml.
2. http://www.w3.org/TR/NOTE-xml-ql/.
3. Cohen, S., Kanza, Y., Kogan, Y., Nutt, W., Sagiv, Y., and Serebrenik, A., EquiX—Easy Querying in XML Databases.
4. Pitts-Moultis, N., *XML in record time*, San Francisco: Sybex, 1999. Translated under the title *XML za rekordnoe vremya*, Moscow: Mir, 2000.
5. Banerjee, J., Kim, W., Kim, H.-J., and Korth, H.F., Semantics and Implementation of Schema Evolution in Object-Oriented Databases, *SIGMOD Conf.*, 1987, pp. 311–322.
6. Peters, R.J. and Oszu, M.T., An Axiomatic Model of Dynamic Schema Evolution in ObjectBase Systems.
7. Peters, R.J. and Oszu, M.T., Axiomatization of Dynamic Schema Evolution in Object-Bases.
8. Roddick, J.F., A Survey of Schema Versioning Issues for Database System, *Information and Software Technology*, 1995, vol. 37, no. 7, pp. 383–393.
9. *Database Management Systems*, 1995, no. 2.
10. Kim, W. and Chou, H.-T., Versions of Schema for Object-Oriented Databases, *VLDB Conf.*, 1988, pp. 148–159.

SPELL: 1. schemas, 2. acyclicity, 3. Online, 4. EquiX, 5. ObjectBase