A Comparison of XML-based Temporal Models

Khadija Ali, Jaroslav Pokorný

[1] Czech Technical University, Faculty of Electrical Engineering, Praha , Czech Republic
alik1@fel.cvut.cz
[2] Charles University, Faculty of Mathematics and Physics, Praha , Czech Republic
jaroslav.pokorny@mff.cuni.cz

Abstract. Much research work has recently focused on the problem of representing historical information in XML. This paper describes a number of temporal XML data models and provides their comparison according to the following properties: time dimension (valid time, transaction time), support of temporal elements and attributes, querying possibilities, association to XML Schema/DTD, and influence on XML syntax. We conclude that the approaches of managing time information using XML mostly do not require changes of current standards.

Keywords: XML, temporal XML data model, bitemporal XML data model, versioning XML documents, transaction time, valid time, efficacy time, native XML databases (NXDs).

1 Introduction

Recently, the amount of data available in XML [13] has been rapidly increasing. In context of databases, XML is also a new database model serving as a powerful tool for approaching semistructured data. Similarly to relational or object-relational models in the past, database practice with XML started to change also towards using time in some applications, such as banking, inventory control, health-care, and geographical information systems. Much research work has recently focused on adding temporal features to XML, i.e. to take into account change, versioning, evolution and also explicit temporal aspects of XML data, like, e.g., the problem of representing historical information in XML. XML documents are related to time in two aspects: they contain temporal information and/or their contents evolve with time. Examples of the latter case include normative texts, product catalogues, etc. We consider both aspects in the temporal XML data models.

To manage temporal information in XML documents, a temporal XML data model is required. Based on similar approaches well-known from the field of temporal relational databases [9], many researchers transformed old ideas into the world of hierarchical structures of XML documents. Technically, to develop an XML temporal data model, it is necessary to extend a XML data model by a time dimension. The problem is that there is more XML data models (e.g. Infoset [14], XPath data model [3], XQuery data model [1], etc.) and more times (usually valid and transaction times).

This fact complicates a comparison of various temporal XML data models that occur in literature.

In [9] the relational temporal data models are classified as two main categories: temporally ungrouped models and temporally grouped data models. As opposed to the former, a *temporally grouped* data model can be expressed by relations in non-first-normal-form model or attribute time stamping, in which the domain of each attribute is extended to include the temporal dimension. The hierarchical structure of XML provides a natural environment for use of temporally grouped data models. We describe a number of temporally grouped XML data models and provide their comparison according to the following properties: time dimension (valid time, transaction time), support of temporal elements and attributes, querying possibilities (particularly in languages XPath and XQuery), association to XML Schema/DTD, and influence on XML syntax.

The paper is organized as follows. Section 2 contains a brief overview of some works which have made important contributions on adding temporal features to XML. We describe an XML-based bitemporal data model (XBIT) and its application for versioned documents. Then we describe a temporal XML data model which is able to manage the dynamics of normative texts in time. We introduce also a valid-time XPath data model. The model adds valid time support to XPath language. Then we present a key-based approach for archiving data. The last model introduced is a multidimensional XML model. In Section 3 we summarize all the mentioned models. We briefly analyze their characteristics and subsequently we express our point of view. **In Section 4, we describe shortly an ongoing work of a new XML-based temporal model**. Finally, in Section 5, we present our conclusions and future investigations.

2 Time in XML – an Overview

As usually, an XML data model should provide tools for describing structure of XML data, integrity constraints, manipulation statements, and querying XML data. We mostly omit integrity constraints and in manipulations we focus on promitive change operators for elements and/or attributes: update, insert, and delete.

2.1 XBIT - an XML-based Bitemporal Data Model

The approach introduced in [11] is based on temporally-grouped data model. A temporal XML document is represented by adding two extra attributes, namely `vstart` and `vend`, representing the time interval for which an element is valid. `vstart` and `vend` represent the inclusive valid time interval (`vend` can be set to the special symbol `now`*)* to denote the ever-increasing current date. Each temporal element is assigned also two extra attributes; `tstart` and `tend` to represent the inclusive transaction time interval; `tend` can be set to UC (until changed).

Example 1: A temporal element `title` can be represented in XBIT in the following way:

```
<title vstart="1998-01-01" vend="now"
```

```
    tstart="1997-09-01" tend="UC">Sr Engineer</title>
```

The expression says that `title` with the value `Sr Engineer` is valid from 1998-01-01 until now. This fact is recorded in the database in 1997-09-01.

The model can also support powerful temporal queries expressed in XQuery without requiring the introduction of new constructs in the language; all the constructs needed for temporal queries can be introduced as user-defined libraries. Modifications in XBIT can be seen as the combination of modification on valid time and transaction time history. XBIT will automatically *coalesce* on both valid time and transaction time, for instance, the valid time intervals with the same title value that overlap or are adjacent. Those intervals are merged by extending the earlier interval; this process is repeated until maximal intervals are constructed. At this point, the non-maximal intervals are removed. The used technique is general and can be applied to historical representation of relational data and versions management in archives (see Section 2.2).

2.2 An XML–Based Model for Versioned Documents

An efficient technique for managing multiversion document histories [10] is used by storing the successive versions of a document in an incremental fashion. Elements of an XML document use again attributes `vstart` and `vend` representing the time interval for which this elements version is valid. Elements containing attributes can be supported by representing each attribute as a subelement denoted by a special flag attribute `isAttr`. For instance, if the employee element contains the attribute `empno` then this fact is represented as:

```
<empno isAttr="yes" vstart="1999-01-01" vend="now"> e1
</empno>
```

Change operations on XML documents can be represented in the model for elements. Three primitive change operations are considered, delete, insert, and update. The following is the effect of performing each operation:

Update. When the element *a* is updated at time *t*:

1. a new element with the same name *a* will be appended immediately after the original one; the attributes *vstart* and *vend* of this new element are set to *t* and *now*, respectively.
2. the *vend* attribute of the old element is set to $t - 1$.

Consider the element `<a vstart="2000-01-01" vend="now">123</a>`, and suppose to change the value of *a* to `250` at `"2002-01-02"`. Then we get the following elements:

```
<a vstart="2000-01-01" vend="2002-01-01">123</a>
<a vstart="2002-01-02" vend="now">250</a>
```

Insert. When a new element is inserted at time *t*, this element is inserted into the corresponding position in the document; the *vstart* attribute is set to *t*, and *vend* is set to *now*.

Delete. When an element is removed at time *t*, the *vend* attribute is set to *t* -1. Suppose to delete the element *a* at 2004-01-02. Then we get the following element:

```
<a vstart="2002-01-02" vend="2004-01-01">250</a>
```

The DTD of the versioned XML document can be automatically generated from the original DTD. Two new attributes `vstart` and `vend` are added to each element; an attribute of an element will be converted as a child element. For instance, the temporal element `title` is represented in the DTD as

```
<!ELEMENT title (#PCDATA)>
<!ATTLIST title vstart CDATA #REQUIRED
                vend CDATA #REQUIRED>
```

Due to the temporally grouped features of the model, it is possible to express powerful temporal queries in XQuery. For instance, the query "Find chapters in which the title did not change until a new History section was added" can be expressed as

```
for $ch in document ("V-Document.xml")/document/chapter
let $title:= $ch/title[1]
let $sec:= $ch/section[. ="History"]
where not empty($title) and not empty($sec) and
      $title/@vend = $sec/@vstart
return $ch
```

2.3 An XML-Based Temporal Data Model for the Management of Versioned Normative Texts

In this model [7], four dimensions (publication, validity, efficacy, and transaction times) are used in the context of legal documents. A temporal element is chosen as a basic unit of temporal pertinence. They are used to represent the evolution of norms in time and their resulting versioning. *Publication time* is the time of publication of norms in an official journal. *Efficacy time* usually corresponds to the validity of norms, but sometimes the cancelled norm continues to be applicable to a limited number of cases. *Valid time* represents the time the norm is in force (the time the norm actually belongs to the regulations in the real world). *Transaction time* is the time the norm is stored in the computer system.

An alternative XML encoding schema has been developed for normative text based on an XML-schema which allows the introduction of time stamping metadata at each level of the document structure which is a subject to change. For the management of norms, three basic operators are defined; one for the reconstruction of a consistent temporal version and the other two for the management of textual and temporal changes. Querying uses combination of full text retrieval and XQuery extended by some constructs to deal with time dimensions.

2.4 A Valid Time XPath Data Model

In the valid time XPath data model [12] a list of disjoint intervals or instants that represent the valid time is added to each node of the original XPath data model.

- Every node of the tree structure of an XML document is associated with the valid time that represents when the node is valid, no node can exist at a valid time when its parent node is not valid.
- The valid time of any node is a superset of the union of the valid times of all its children as well as all its descendents. The valid time of the root node should be a superset of the union of the valid times of all nodes in the document.
- The valid time of an edge is determined by the valid time of the nodes at the edge's two ends (if both nodes are valid, an edge can exist). The valid time of the edge is result of t_1 □ t_2, where t_1 and t_2 are the valid times of the edge's two ends.

The XPath is extended with an axis to locate the valid time of a node. A valid time axis is added to XPath to retrieve nodes in a view of the valid time for a node. The axis returns a list of nodes relative to the context node. Each node in an XML document has a corresponding valid time view containing its valid time information. Here, a valid time list can be viewed as an XML document. Each time in the valid time list is denoted as a `<time>` element. The content of the `<time>` is unique to the view. Below we show the valid time view of an element in the commonly used Gregorian calendar; "year", "month ", and "day" element nodes are nested under "begin" and "end" of each view.

```
<validtime>
    <time>
      <begin>
        <day>31</day>
        <month>Jan</month>
        <year>1999</year>
      </begin>
      <end>
        <day>now</day>
        <month>now</month>
        <year>now</year>
      </end>
    </time>
</validtime>
```

Remind that any calendar can be used. The commonly used calendar is Gregorian calendar; however there are other calendars that are widely used by people in different regions.

The valid time axis selects the list of nodes that form a document-order traversal of the valid time view. By this constraint, the nodes in the valid time axis are ordered according to the document order traversal of the valid time view. The valid time axis of a node contains the valid time information of the node as if it had originated from an XML document (a document order refers to the standard document order as it is specified in Infoset).

Since the `<time>` elements in the valid time view are ordered by the actual time they represent, these `<time>` elements selected by the valid time axis are also in this order.

Example 2: Below are some simple examples of using the valid time axis to query within the default view of the valid time.

`v/valid`	specifies the valid time axis of the node *v*.
`v/valid::day`	selects all the day nodes in the axis.
`v/valid::time[2]`	selects the second time node in the axis .

2.5 Key-Based Model for Archiving Data

In this archiving technique [2], a document is viewed as unordered set of XML elements. The elements can be uniquely identified and retrieved by their logical keys; elements have timestamps only if they are different from the parent node.

Key-based approach is used for identifying the correspondence and changes between two given versions based on keys. In contrast to diff-based approach which (i) keeps a record of changes – a "delta" - between every pairs of consecutive versions, (ii) stores the latest version together with all forward completed deltas –changes between successive versions- that can allow one to get to an earlier version by inverting deltas on the latest version, the Key-based approach can preserve semantic continuity of each data element in the archive. An element may appear in many versions whose occurrences are identified by using the key structure and store it only once in the merged hierarchy.

A *key* is a pair (Q, $\{P_1,\ldots, P_k\}$) where Q and P_i, $i \in [1, k]$, are path expressions in a syntax similar to XPath. Informally, Q identifies a target set of nodes reachable from some context node and this target set of nodes satisfies the key constraints given by the paths, P_i, $i \in [1, k]$.

Example 4: Consider the XML document

```
<DB>
   <A> <B>1</B> <C>1</C> </A>
   <A> <B>1</B> <C>2</C> </A>
</DB>
```

The document satisfies the key`(/DB/A,{C})`but it does not satisfy the key `(/DB/A,{B})`since both `A` elements have the same key path value, i.e., `<B>1</B>.`

This archiving technique requires that all versions of the database must conform to the same key structure and the same schema as well.

All the versions are merged into one hierarchy where an element appearing in multiple versions is stored only once along with a timestamp. The main idea behind nested merge is:

- Recursively to merge nodes in *D* (incoming version) to nodes in *A* (the archive) that have the same key value, starting from the root.

- When a node *y* from *D* is merged with a node *x* from *A*, the time stamp of *x* is augmented with *i* (the new version number).The sub-trees of nodes *x* and *y* are then recursively merged together.
- Nodes in *D* that do not have corresponding nodes in *A* are simply added to *A* with the new version number as its time stamp.
- Nodes in *A* that no longer exist in the current version *D* will have their timestamps terminated appropriately, i.e., these nodes do not contain timestamp *i*.

Since the archive is in XML, the existing XML query languages such as XQuery can be used to query such documents. The authors of [2] did not discuss the issue of temporal queries in detail.

2.6 A Multidimensional XML Model (MXML)

In the approach [5], we represent multiple versioning not only with respect to time but also to other context parameters such as language, degree of detail, etc.

In a multidimensional XML document, dimensions may be applied to elements and attributes. A multidimensional element/attribute is an element/attribute whose contents depend on one or more dimensions. The notion of *world* is fundamental in MXML. A *world* represents an environment under which data in a multidimensional document obtain a meaning. A *world* is determined by assigning values to a set of dimensions.

Example 5: Consider the *world w* = {(*time*, 2005-12-14), (*customer_type*, student), (*edition*, English)}. The dimensions names are *time*, *customer_type,* and *edition*. The assigned values of these dimensions names are 2005-12-14, student, and English respectively.

The multidimensional element is denoted by preceding the element's name with the special symbol **"@",** and encloses one or more context elements. All context elements of a multidimensional element have the same name which is the name of the multidimensional element. Consider the following MXML document:

```
<book>
  <@isbn>
   [edition = greek]  <isbn>0-13-110370-9</isbn> [/]
   [edition = English] <isbn>0-13-110362-8</isbn> [/]
  </@isbn>
</book>
```

The `@isbn` is a multidimensional element dependent on the dimension `edition`. It has two context elements having the same name `isbn` (without the special symbol "@"). Context specifiers qualify the facets of multidimensional elements and attributes, called *context elements/attributes*, stating the sets of worlds under which each facet may hold. `[edition = greek]`and `[edition = English]` are the context specifiers of `@isbn`. [/] represents the end symbol of a context specifier.

Change operations (update, delete, insert) can be represented in MXML for both elements and attributes. For instance, consider the element `<p a`$_1$ `="9"> v`$_1$ `</p>` and suppose to delete the attribute `a`$_1$ at time point `t`. Then we get the following MXML element:

```
<p a1=[d in{start..t-1}]"9"[/]>v1</p>
```

where a dimension named `d` is used to represent time, `start` is a reserved word representing the start time, `t-1` represents the end time.

The history of the schema of an XML document can be represented easily, for instance, deleting an element, or adding an attribute to an element at a specific time point. The following XML schema description retain the element `r` as optional during the interval `{t..now}`.

```
<xs:element name="r" type="xs:string"
            minOccurs=[d in {t..now}]"0"[/]/>
```

Consider the following XML document:

```
<p>
   <q>v1</q> <r>v2</r> <s>v3</s>
</p>
```

The schema for this document may be encoded in XML schema as follows:

```
<xs:element name="p">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="q" type="xs:string"/>
       <xs:element name="r" type="xs:string"/>
       <xs:element name="s" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```

Suppose to delete the element `r` at time point `t`. Then we get the following MXML element:

```
<p>
   <q>v1</q>
   <@r>
      [d in {start..t-1}] <r>v2</r> [/]
   </@r>
   <s>v3</s>
</p>
```

After deleting the element `<r>v2</r>` at time `t`, it is necessary to modify the document's schema if we want the XML document resulting by applying the deletion to become valid. This change can be represented by turning the element sequence of the above XML schema into a multidimensional element with two facets:

```
<xs:element name="p">
   <xs:complexType>
     <@xs:sequence>
       [d in {start..t-1}]
          <xs:sequence>
             <xs:element name="q" type="xs:string"/>
             <xs:element name="r" type="xs:string"/>
```

```
              <xs:element name="s" type="xs:string"/>
           </xs:sequence>
        [/]
        [d in {t..now}]
           <xs:sequence>
              <xs:element name="q" type="xs:string"/>
              <xs:element name="s" type="xs:string"/>
           </xs:sequence>
        [/]
      </@xs:sequence>
    </xs:complexType>
  </xs:element>
```

3 Summary of XML-Based Temporal Data Models

So far, we have introduced some works which have made important contributions in providing expressive and efficient means to model, store, and query XML-based temporal data models. In the following subsections we provide a comparison of all mentioned models according to the following properties: time dimension (valid time, transaction time), support of temporal elements and attributes, querying possibilities, association to XML Schema/DTD, and influence on XML syntax.

Time dimension. All the models are capable to represent changes in an XML document by supporting temporal elements, and incorporating time dimensions. Two time dimensions are usually considered: valid time and transaction time. There are several other temporal dimensions that have been also mentioned in the literature in relation to XML. In [7] a publication time and efficiency time in the context of legal documents are proposed.

Temporal elements and attributes. Time dimensions may be applied to elements and attributes. All the models are capable to support temporal elements. In [5] and [10] the temporal attributes are supported. In our point of view, supporting temporal attributes adds an advantage to the model. In [5] versions of an element are explicitly associated as being facets of the same (multidimensional) element. Grouping facets together allows the formulation of cross-world queries, which relate facets that hold under different worlds [6].

Influence on XML syntax. Temporal information is supported in XML much better than relational tables. This property is attributed to the hierarchical structure of XML which is compatible perfectly with the structure of temporal data. Only in [5] the syntax of XML is extended in order to incorporate not only time dimensions but also other dimensions such as language, degree of detail, etc. So the approach in [5] is more general than other approaches as it allows the treatment of multiple dimensions in a uniform manner.

Querying possibilities. In our point of view, the model's power depends also on supporting powerful temporal queries. In [10] and [11] powerful temporal queries expressed in XQuery without requiring the introduction of new constructs in the language are supported. In [12] a valid time support is added to XPath. This support results in an extended data model and query language. In [7] querying uses combination of full text retrieval and XQuery extended by some constructs to deal with time dimensions. The other models in [5] and [2] did not discuss the issue of temporal queries; in [2] elements have timestamps if they are different from the parent nodes. This fact complicates the task of writing queries in XPath/XQuery**;** consider the following XML representation of an archive containing versions 1 and 2.

```
<T t="1-2">
   <db>
      <employee>
         <id>1</id>
         <name>Anas</name>
         <address>
            <city>Prague</city>
            <street>Krouzova 18</street>
         </address>
         <salary><T t="1">22k</T>
                 <T t="2">30k</T></salary>
      </employee>
      ...
   </db>
</T>
```

Note that `T` is a special element represents element's versions by a special attribute `t`. For instance, the tenth line says that the value of `salary` is `22k` and `30k` in the first and second version respectively. The salary of `Anas` in the second version can be expressed easily in XQuery (because salary elements have their own timestamps):

```
//employee[name="Anas"]/salary/T[@t="2"]
```

But where `Anas` was living in the second version can not be expressed easily in XQuery. It requires to check timestamps of `db` element since `city`, its parent (`address`), and parent of its parent (`employee`) do not have timestamps.
Association to XML Schema/DTD. A significant advantage will be added to the model if it is not only representing the history of an XML document but also the history of its corresponding XML schema or DTD as well. In [5], [7], and [10] the temporal XML schema/DTD is supported by extending the existing XML schema/DTD. All the mentioned models are summarized in Table 1.

4 3D_XML: a three-Dimensional XML-based model

In this Section we describe shortly an ongoing work of a new XML-based temporal model. Our model is a three-dimensional XML-based model (3D_XML in short) for representing and querying histories of XML documents. The proposed model incorporates three time dimensions, *valid time*, *transaction time*, and *efficacy time* without extending the syntax of XML. An important issue of each data model is its implementation. Native XML databases *NXDs* represent a suitable storage platform when complex time dependent data has to be manipulated and stored, so we chose to implement temporal queries directly in NXDs (particularly *eXist*). 3D_XML is equipped with a set of temporal constructs; valid/efficient times relationships constructs, interval comparison operators, snapshot data construct, etc. We use XQuery to express complex temporal queries, but the expression of these queries is greatly simplified by a suitable library of built-in temporal functions; the preliminary experimental results on query performance are encouraging.

5 Conclusions

In this paper, we showed that an effective temporal information system should provide (i) expressive temporal data model, (ii) powerful language for temporal queries and snapshot queries as well. We conclude that XML provides a flexible mechanism to represent complex temporal data. Its query language XQuery is natively extensible and Turing-complete [8], and thus any extensions needed for temporal queries can be defined in the language itself. This property distinguishes XML temporal querying from that one in relational temporal languages, e.g. TSQL. So, any syntax extension of XQuery towards a temporalness, e.g. τXQuery [4], makes only queries easier to write. The most usual bitemporal representation does not require changes of current XML standards.

Table 1. Features summary of XML-based temporal models. ("-" means that the associated feature is not discussed in the original model; Y and N denote Yes and No, respectively.

Model	Time dimension	Supports temporal elements	Supports temporal attributes	Querying	Extends XML schema/ DTD	Extends XML syntax
XBIT [11]	Valid/ transaction time	Y	N	XQuery	-	N
Versioned Documents [10]	Valid/ transaction time	Y	Y	XQuery	Y	N
Versioned normative texts [7]	Valid/ transaction/ publication/ efficacy time	Y	N	XQuery is extended	Y	N
Valid time XPath data model [12]	Valid time	Y	N	XPath is extended	-	N
Key-based model for archiving data [2]	Valid time	Y	N	-	-	N
MXML [5]	Valid/ transaction time	Y	Y	-	Y	Y

XML can be even an option for implementations of temporal databases (or multidimensional databases) on a top of a native XML DBMS. Our work shows that there are a lot of important topics for forthcoming research. Many research issues remain open at the implementation level, including, e.g., the use of nested relations on the top of an object-relational DBMS, reflecting temporal features into an associated XML query language, etc. The future work is directed to extend 3D_XML model by adding more temporal constructs in order to support more powerful temporal queries. Support of updates will be also a real area of future investigations.

Acknowledgements. The research was in part supported by grant 1ET100300419 of the Information Society Program - Thematic Program II of the National Research Program of the Czech Republic.

References

1. Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J. Siméon, J.: XQuery 1.0: An XML Query Language, W3C Working Draft, 04 April 2005. Available: http://www.w3.org/TR/xquery/
2. Buneman, P., Khanna, S., Tajima, K., and Tan, W.: Archiving scientific data. In: Proc. of ACM SIGMOD Int. Conference (2002) 1-12
3. Clark, J., DeRose, S.: XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999. Available: http://www.w3.org/TR/xpath/.
4. Geo, D., Snodgrass, R.: Temporal slicing in the evaluation of XML queries. In: Proc. of VLDB, Berlin, Germany (2003) 632-643
5. Gergatsoulis, M. and Stavrakas, Y.: Representing Changes in XML Documents using Dimensions. In Proc. of 1st Int. XML Database Symposium (2003) 208-221
6. Gergatsoulis, M., Stavrakas, Y., Doulkeridis, C., and Zafeiris, V. Representing and querying histories of semistructured databases using multidimensional OEM. Inf. Syst., Vol. 29, No. 6. Elsevier Science (2004) 461-482,
7. Grandi, G., Mandreoli, F., Tiberio, P.: Temporal Modelling and Management of Normative Documents in XML Format, Data and Knowledge Engineering, 54:3, Elsevier Science (2005) 227-254
8. Kepser, S.: A Simple Proof of the Turing-Completeness of XSLT and XQuery. In: Proc. of Extreme Markup Languages, Montréal, Québec (2004)
9. Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A., Snodgrass, R.T.: Temporal Databases: Theory, Design and Implementation, Benjamin/Cummings Publishing Company, California (1993) 496-507
10. Wang, F. and Zaniolo, C.: Temporal Queries in XML Document Archives and Web Warehouses. In Proc. of 10th Int. Symposium on Temporal Representation and Reasoning (2003) 47-55
11. Wang, F. and Zaniolo, C.: XBIT: An XML-based Bitemporal Data Model. In: Proc. of 23rd Int. Conference on Conceptual Modeling, Shanghai, China (2004), 810-824
12. Zhang, S. and Dyreson, C.: Adding Valid Time to XPath. In: Proc. of 2nd int. Workshop on Database and Network Information Systems, Aizu, Japan (2002) 29-42
13. W3C: Extensible Markup Language (XML) 1.1. (Third Edition), W3C Recommendation 04 February 2004, Available: http://www.w3.org/TR/xml11/
14. W3C : XML Information Set (Second Edition). W3C Recommendation 04 February 2004. Available: http://www.w3.org/TR/xml-infoset/