2004

An XML Representation of the Parametric Data Model for Temporal Data

Seo-Young Noh
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports
Part of the Systems Architecture Commons, and the Theory and Algorithms Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/151

This Article is brought to you for free and open access by the Computer Science at Iowa State University Digital Repository. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
An XML Representation of the Parametric Data Model for Temporal Data

Seo-Young Noh

Department of Computer Science
Iowa State University
rsyoung@cs.iastate.edu

Abstract

This paper introduces an XML representation for the parametric data model handling temporal data. The parametric data model represents an object either in a single tuple or in multiple tuples. This feature reduces query complexities at the user level and provides elegant approaches to manage temporal data. However, this property requires the data model to have flexible attribute value sizes and causes the implementation complexity despite its advantages. To solve this problem, parametric databases have to manage variable attribute value sizes, which are in the same attribute field. XML can provide a feasible solution to represent parametric databases. XML does not have any boundary restrictions. In this paper, we introduce an XML representation scheme for the parametric data model and show that the scheme reflects the properties in the parametric data model.

1 Introduction

Conventional database systems are capable of storing only the current perception of reality and relationships among objects. Such databases enforce currency of data by excluding old data values when newer ones become available. After such currency updates, the old values are lost from the logical level and only the current state remains available. On the other hand, temporal database systems are capable of storing multiple versions of data, thereby allowing users to examine complete object histories [1].

Many temporal data models exist and they have their own merits in their specific application areas. Temporal data models can be broadly classified by three criteria such as domain representation, domain visibility, and time-stamping. Three different types of domain representation schemes exist: point-based, interval-based, and temporal element-based time reference models. Point-based time reference views time domain as a discrete, countable, infinite, linearly ordered set without endpoints. The individual element of the set represents the actual time instances while the linear order represents the progression of time. The example of this data model is shown in SQL/TP [2] and SQLT [3]. Interval-based scheme represents a domain of an object as the continuous maximum time interval. Interval-based temporal data models are introduced in TSQL2 [4] and IXQL [5]. Temporal element-based schemes represent domains of an object as finite unions of time intervals. Examples of this model are introduced in ParaSQL [6] and Tansel’s NTC (Nested Relational Tuple Calculus) [7]. Another criteria is the domain visibility. There are two types of temporal data models based on domain visibility: implicit domain and explicit domain. An implicit domain is invisible to users, but an explicit domain is visible. The last criteria is the time-stamping scheme.
Two different types of schemes exist: tuple-level time-stamping and attribute-level time-stamping. Tuple-level time-stamping assigns a time domain to a single tuple, that is, all attributes share the same domain. Attribute-level time-stamping schemes, however, assign time domains at attribute values.

The parametric data model is temporal element-based with explicit domain and attribute-level time-stamping. The parametric data model’s approach to time dimension is simple, yet elegant. The parametric data model illustrates objects or events in tuples. An object can be stored either in a single tuple or in multiple tuples based on modeling schemes. This feature is one of the main differences found among other temporal data models that model objects in terms of multiple tuples. Due to this property, the parametric data model reduces query complexities without introducing self-join operations. In spite of the advantages, no implemented systems exist. This is mainly because of parametric tuple storage difficulty. Since parametric tuples are not pre-fixed, they cannot be represented in pre-defined or pre-fixed tuple sizes. Therefore, relation or object-oriented database systems are not suitable for the parametric data model.

Over the past few years, many researchers have utilized XML technology in database areas. Because of XML, a new database generation is emerging. XML technology is an emerging technology used in many applications. XML is advantageous because it describes object characteristics such that it can represent unfixed format data. This paper will introduce an XML representation of the parametric data model handling temporal data. The XML representation will reflect the properties and usability of the parametric data model.

The organization of this paper is as follows. Section 2 introduces the parametric data model for temporal data. Section 3 discusses an XML representation. Section 4 introduces a transformation scheme from the parametric data model to an XML representation. Section 5 provides properties of the XML representation. Section 6 shows the applicability of the XML representation. We conclude this paper in Section 7.

2 Parametric Data Model for Temporal Data

In this section, the general concept of the parametric data model will be discussed. In the parametric data model, temporal elements are used to represent domains of objects. An attribute is defined as a function of time. Unlike other temporal data models, attribute values contain their own domains with values. Figure 1 shows an example of a temporal parametric relation, MANAGEMENT whose attributes are Dept and Manager. The relation maintains the history of departments with their managers. Based on the example, we will discuss the concept of temporal elements, temporal attribute values, tuples, and relations.

2.1 Temporal Elements

Time intervals are not adequate to model the history of an object in a single tuple, and they make query languages difficult to use. To obtain timestamps that are closed under the set theoretical operations, the concept of temporal elements is introduced in the parametric data model. The parametric data model assumes that there is a universe of time that consists of an interval [0, NOW] of instants with a linear order \( \preceq \) on it. Here NOW denotes the current instant of time. For simplicity, it is assumed that [0, NOW] is the discrete set \{0, 1, · · · , NOW\}. A temporal element is a finite union of time intervals. A time interval is a temporal element. An instant \( t \) may be identified with the interval \([t, t]\); thus, it is regarded as a temporal element [8]. In MANAGEMENT relation, examples of temporal elements are \([41, 47] \cup [71, \text{NOW}]\), or \([11, 49]\). As expected, the set of all temporal elements is closed under \( \cup, \cap, \) and \( \neg \) (complementation with respect to \([0, \text{NOW}]\)).
### 2.2 Temporal Attribute Values

To capture the changing value of an attribute, a *temporal value* of an attribute \( A \) is defined as a function of a temporal element into the domain of \( A \) [9]. An example of a temporal value of Manager attribute in the first tuple in MANAGEMENT relation is \( \langle [11,44] \text{ John}, [45,49] \text{ Leu} \rangle \). If \( \xi \) is a temporal value, \( J(\xi) \) denotes its domain. Thus \( J(\langle [11,44] \text{ John}, [45,49] \text{ Leu} \rangle) = [11,49] \). A temporal value is also called an *attribute value* or simply a *value*.

### 2.3 Homogeneity, Temporal Tuples, and Temporal Relations

A *homogeneous tuple* \( \tau \) over a schema \( R \) is a function from \( R \) such that for every attribute \( A \) in \( R \), \( \tau(A) \) is a temporal value of \( A \) and all the temporal values in the tuple have the same domain. Informally, a tuple is a concatenation of temporal values whose temporal domains are the same. The assumption that all temporal values in a tuple have the same domain makes the tuples homogeneous. Suppose \( \tau \) is a tuple. Then the temporal domain of \( \tau \) is the temporal domain of any attribute and is denoted by \( J(\tau) \).

In parametric model, the entire history of a real world object is accumulated in a single tuple. A temporal relation \( r \) over \( R \), with \( K \subseteq R \) as its key, is a finite set of nonempty tuples such that no key attribute value of a tuple changes with time, and no two tuples agree on all their key attributes. A key in parametric model is not required to be minimal or nonempty [9].

### 3 An XML Representation

In this section, an XML representation for the parametric data model, XML(R), will be introduced. Since XML(R) is an XML document, it consists of nodes and edges like an ordinary XML document. However, XML(R) has more specific definitions such as element nodes, value nodes, and subtrees. The purpose of these new definitions is to provide elegant transformation schemes from a parametric relation to an XML document without losing any properties. We first define an XML document \( D \) with nodes and edges. In order to reflect the properties of the parametric data model, nodes are classified into two types—elements and values. Elements consist of four different types of elements. By using the definitions of elements and nodes, we define subtrees which are bases of XML(R).

- **Definition 1: XML Document**
  Let \( D \) be an XML document. \( D \) is defined as \( D = (N, E) \), where \( N \) is a set of nodes and \( E \) is a set of edges in \( D \).
• **Definition 2: Node**
  Let $N$ be a set of nodes in $D$. There are two types of nodes: element, and value nodes, denoted as $N^e, N^v$, respectively. An element node is non-terminal node, that is, it has at least a child node when it is used in a tree. In contrast, a value node is a terminal node that does not have a child node. $N^e$ and $N^v$ are disjoint such that $N^e \cap N^v = \emptyset$ and $N = N^e \cup N^v$.

• **Definition 3: Element**
  Set of elements, $N^e$, consists of four different types: Dunit (Domain Unit), Pdom (Parametric Domain), Tunit (Terminal Unit), Iunit (Internal Unit) elements.

  A **Dunit element** is a parent element of a value node. It represents a maximal continuous time domain of a value (time interval). We denote a set of Dunit elements as $N^d$ and $N^d \subset N^e$.

  A **Pdom element** is a parent element of Dunit elements. It represents a temporal element of an object. We denote a set of Pdom elements as $N^p$ and $N^p \subset N^e$.

  A **Tunit element** is a parent element of a Pdom element and a value node. It represents a data value and its domain. We denote a set of Tunit elements as $N^t$ and $N^t \subset N^e$. It is used to represent a temporal value.

  An **Iunit element** is a parent element of a Pdom element and either Iunit or Tunit elements. It is used to internally connect between Iunit elements or between Tunit elements. We denote a set of Iunit elements as $N^i$ and $N^i \subset N^e$.

  The element set, $N^e$, is partitioned by $N^d, N^p, N^t$, and $N^i$, that is, $N^d \cap N^p \cap N^t \cap N^i = \emptyset$, where the size of each subset of $N^e$ is positive. The set of elements satisfies $N^e = N^d \cup N^p \cup N^t \cup N^i$ in XML document $D$.

Based on the definition of four types of elements and a type of value node, the edges in $D$ can be defined as connections between them.

• **Definition 4: Edge**
  Let $E$ be a set of edges in an XML document $D$. An edge $(p, c) \in E$ is one of following, where $p, c \in N$:

  $$(p, c) = \begin{cases} 
  p \in N^d \text{ and } c \in N^v & \text{Dunit and Value} \\
  p \in N^p \text{ and } c \in N^d & \text{Pdom and Dunit} \\
  p \in N^t \text{ and } c \in N^p & \text{Iunit and Pdom} \\
  p \in N^i \text{ and } c \in N^t & \text{Iunit and Tunit} \\
  p \in N^i \text{ and } c \in N^i & \text{Iunit and Iunit} \\
  p \in N^t \text{ and } c \in N^p & \text{Tunit and Pdom} \\
  p \in N^t \text{ and } c \in N^v & \text{Tunit and Value} 
  \end{cases}$$

Up to this point, an XML document, $D$, has been described by lowest level characteristics such as edges and nodes. An XML representation, XML(R), has some restrictions to represent a parametric data model to an XML document. For example, a node (or single element) can not be XML(R) even though a single element can be an XML document. In addition, XML(R) should be a tree with at least height of 6. Therefore, it is required to define subtrees because XML(R) is a composition of subtrees.

• **Definition 5: Set of Subtrees**
  Let $T$ be a set of subtrees in an XML document $D$. $T$ can be defined as $T = (N', E')$, where
\[ N' \subseteq N \text{ and } E' \subseteq E. \] T has five different types of subtrees: \( Dunit, Pdom, Tunit, Iunit, \) and \( Xunit \) trees, denoted as \( t^d, t^p, t^t, t^i \) and \( t^x \), respectively. We also denote the set of each trees as \( T^d, T^p, T^t, T^i, \) and \( T^x \).

- **Definition 6: Dunit Tree**
  A Dunit tree is defined as \( t^d = (p, c, e) \), where \( p \in N^d, c \in N^v \), and \( e = (p, c) \). Figure 2 shows a Dunit tree. A Dunit tree represents a time interval.

- **Definition 7: Pdom Tree**
  A Pdom tree is defined as \( t^p = (p, T^d_k, E') \), where \( p \in N^p, T^d_k \) is a set of Dunit trees such that \( T^d_k = \{t^d_1, t^d_2, \ldots, t^d_k\} \), and \( E' \subset E \).
  Let \( \text{root}(t^d_i) \) be a root node of \( t^d_i \). An edge \( (p, c) \) is defined as \( (p, \text{root}(t^d_i)) \) or simply \( (p, t^d_i) \) and \( E' = \{(p, t^d_1), (p, t^d_2), \ldots, (p, t^d_k)\} \). Therefore, a Pdom tree contains Dunit trees as children nodes. Figure 3 shows a Pdom tree. A Pdom tree represents a temporal element of an object. The domain is unions of Dunit trees.

- **Definition 8: Tunit Tree**
  A Tunit tree is defined as \( t^t = (p, N', E') \), where \( p \in N^t, N' \subset N \) and \( E' \subset E \). Let \( \text{root}(t^p) \) be a root node of \( t^p \). The set of nodes, \( N' \) is the union of a Pdom tree’s root and a value node, that is, \( N' = \{\text{root}(t^p)\} \cup \{v\} \). The set of edges in \( t^t \) is \( E' = \{(p, \text{root}(t^p), (p, v))\} \). Therefore, a Tunit tree contains a Pdom tree and a value node as children nodes. Figure 4 shows a Tunit tree. A Tunit tree represents a temporal value.

- **Definition 9: Iunit Tree**
  An Iunit tree is defined as \( t^i = (p, N', E') \), where \( p \in N^i, N' \subset N \) and \( E' \subset E \). Let \( \text{root}(t^p) \) be root node of \( t^p \). The set of nodes, \( N' \) is the union of a Pdom tree’s root and root nodes of Iunit trees, that is, \( N' = \{\text{root}(t^p) \cup \{\text{nodes}(\text{root}(T^i_k))\} \), where \( T^i_k = \{t^i_1, t^i_2, \ldots, t^i_k\} \). The set
Figure 4: Tunit Tree

of edges in \( t^i \) is defined as follows:

\[
E' = E_1 \cup E_2,
\]
where \( E_1 = \{(p, \text{root}(t^p))\} \),
and \( E_2 = \{(p, \text{root}(t^i_1), (p, \text{root}(t^i_2)), \ldots, (p, \text{root}(t^i_k))\} \)

Therefore, an Iunit tree contains a Para tree and a set of Iunit trees as children. Figure 5 shows an Iunit tree. As we will see in Section 4, Iunit trees are used to represent tuples in the parametric data model.

Figure 5: Iunit Tree

• Definition 10: Xunit Tree
  
  An Xunit tree is a special case of an Iunit tree and is defined as \( t^x = (p, N', E') \), where \( p \in N^i \), \( N' \subset N \) and \( E' \subset E \). Let \( \text{root}(t^p) \) and \( \text{root}(t^i) \) be root nodes of \( t^p \) and \( t^i \), respectively. The set of nodes, \( N' \) is the union of a Pdom tree’s root and root nodes of Tunit trees, that is, \( N' = \{\text{root}(t^p)\} \cup \{\text{nodes}(\text{root}(T^i_k))\} \), where \( T^i_k = \{t^i_1, t^i_2, \ldots, t^i_k\} \) The set of edges in \( t^x \) is defined as follows:

\[
E' = E_1 \cup E_2,
\]
where \( E_1 = \{(p, \text{root}(t^p))\} \),
and \( E_2 = \{(p, \text{root}(t^i_1), (p, \text{root}(t^i_2)), \ldots, (p, \text{root}(t^i_k))\} \)
Therefore, an Xunit tree contains a Pdom tree and set of Tunit trees as children. Figure 6 shows an Xunit Tree. An Xunit tree represents a temporal value.

![Xunit Tree](image)

By using the subtrees, XML(R) can be defined with set of subtrees.

**Definition 11: XML(R)**

Let XML(R) be an XML representation for the parametric data model. XML(R) is defined as follows:

\[
XML(R) = (T^d, T^p, T^t, T^x, T^i),
\]

where \(T^d, T^p, T^t, T^x, \) and \(T^i\) are set of Dunit, Pdom, Tunit, Xunit, and Iunit trees, respectively. If XML(R) is an XML representation of the parametric data model, then there exist trees such that \(t^d \subset t^p \subset t^t \subset t^x \subset t^i\), where \(t^d \in T^d, t^p \in T^p, t^x \in T^x, \) and \(t^i \in T^i\).

### 4 Transformation Scheme

In this section, we will discuss the transformation scheme from a parametric relation to XML(R). In order to transform a parametric relation to an XML, we need to generalize parametric relations. Suppose that we are given a parametric relation \(R\). This relation consists of \(n\) different attributes, denoted as \(A_1, \cdots, A_n\). A parametric value of \(A_k\) is denoted as \((\text{Dom}(a_{ij}^k) \ a_{ij}^k \cdots \text{Dom}(a_{ij}^k) \ a_{ij}^k)\), which is the value at the \(k\)th attribute in \(i\)th tuple. Unlike classical relation, a parametric value contains its domain, denoted as \(\text{Dom}(a_{ij}^k)\) which is a temporal element. Figure 7 shows a generalized table of a parametric relation.

We can map the parametric relation \(R\) to a tree representation as shown in Figure 8. In the figure, the name of \(R\) is mapped to the root node in the tree. Each tuple is mapped to \(tup\) node. Each \(tup\) node has attribute nodes, \(A_1, \cdots, A_n\). Each attribute node has a parametric value. We will transform the tree shown in Figure 8 to a complete XML document representing a parametric relation by tree definitions discussed in Section 3.

The nodes in Figure 8 will be changed to subtrees as follows:

- Root Node \(\rightarrow\) Iunit Tree
- Tuple Node \(\rightarrow\) Iunit Tree
- Attribute Node \(\rightarrow\) Xunit Tree
- Attribute Value Node \(\rightarrow\) Tunit Tree
By the mapping relationship, we can transform the tree to an XML representation, XML(R), as shown in Figure 9.

Root Node and Tuple Node have been changed to Iunit trees (or node). Attribute Node and Attribute Value Node are transformed to an Xunit tree and a Tunit tree, respectively. We have to note here that the XML representation, XML(R), has equivalent information with the relation R. If we define a function $Info$ which determines the amount of information that cannot be driven, then $Info(\text{XML}(R)) = Info(R)$. However, XML(R) is more informative in that it has driven domain information from tuple level and relation level. It reduces searching time for domain-related queries.

---

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$\cdots$</th>
<th>$A_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Dom}(a_{11}^1)$</td>
<td>$a_{11}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{11}^n)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\text{Dom}(a_{1a}^1)$</td>
<td>$a_{1b}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{1c}^n)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\text{Dom}(a_{21}^1)$</td>
<td>$a_{21}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{21}^n)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\text{Dom}(a_{2d}^1)$</td>
<td>$a_{2x}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{2f}^n)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\text{Dom}(a_{m1}^1)$</td>
<td>$a_{m1}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{m1}^n)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\text{Dom}(a_{mg}^1)$</td>
<td>$a_{mg}^2$</td>
<td>$\cdots$</td>
<td>$\text{Dom}(a_{mg}^n)$</td>
</tr>
</tbody>
</table>

*a, b, c, d, e, f, g, and i ≥ 1*
5 Properties

5.1 Temporal Domain

In XML(R), temporal domains are easily obtained at Iunit, and Tunit nodes. Table 1 shows the relationship between nodes and temporal domains that can be accessed.

<table>
<thead>
<tr>
<th>Node</th>
<th>Accessible Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iunit</td>
<td>Attribute, Tuple, Relation Level</td>
</tr>
<tr>
<td>Tunit</td>
<td>Attribute Value Level</td>
</tr>
</tbody>
</table>

Let $N'$ be a set of nodes such that $N' = N^i \cup N^t$. Since every node in $N'$ has its domain node, let $p$ be a domain node of $n \in N'$. $p$ has children nodes which are Dunit nodes: $d_1, d_2, \cdots, d_k$. Let’s define a function $val$ as follows:

$$\text{val}(d_i) = \text{value of node } d_i$$

Since $d_i$ is a Dunit node, it is used as a parent node of a value node representing a time interval. By using a function $val$, we can define a temporal domain of $n$, denoted as $\text{Dom}(n)$ as shown in equation (1).

$$\text{Dom}(n) = \bigcup_{i=1}^{k} \left( \text{val}(d_i) \right)$$

(1)

Therefore, a domain of a relation, a tuple, an attribute, or an attribute value is easily obtained by traversing the domain node of Iunit or Tunit nodes. It provides an efficient mechanism to retrieve domains of objects without stepping through an entire XML(R).
5.2 Homogeneity

In the parametric data model, the homogeneity is that domains of attribute values should be the same if the attribute values are in the same tuple. In XML(R), it can be easily evaluated by comparing Xunit trees.

**Lemma 1:** Let \( t^x_i \) and \( t^x_j \) be Xunit trees, which are children of an Iunit tree \( t^i_k \). \( t^x_i \) and \( t^x_j \) satisfy homogeneity if equation (2) holds for all children root nodes of \( t^i_k \):

\[
\text{Dom}(t^x_i) = \text{Dom}(t^x_j), \text{where } i \neq j
\]  (2)

**Proof:** Since an Xunit tree represents an attribute value, its siblings are also Xunit trees in the same tuple whose parent is an Iunit tree, \( t^i_k \). Therefore, the domains of siblings are same.

5.3 Disjoint Property

The disjoint property is to guarantee that domains are not overlapped. Since an attribute is defined as function of time, a single time domain should not be mapped to multiple values of the attribute.

**Lemma 2:** Let \( t^t_i \) and \( t^t_j \) be Tunit trees. If \( t^t_i \) and \( t^t_j \) are siblings, then

\[
\text{Dom}(t^t_i) \cap \text{Dom}(t^t_j) = \emptyset
\]  (3)

**Proof:** Since two trees are Tunit trees, they represent a temporal value in the same attribute. If the domains are not disjoint, there is a common domain such that it is mapped to multiple values. It violates the definition of function. Therefore, if two trees are Tunit trees and siblings, then domains should be disjoint.

5.4 Union Property

Domains of a given node can be obtained by unioning domains of its children nodes except when the node is a Tunit node. Since a Tunit tree whose root is a Tunit node represents an attribute value, the domain of the node cannot be driven, but directly given.

**Lemma 3:** For \( n \in N^i \), let \( N^* \) be a set of children nodes of \( n \) such that \( N^* = \{ n^*_1, n^*_2, \cdots, n^*_p \} \) and \( N^* \subset N^i \cup N^t \). If \( n \) holds equation (4) with respect to \( N^* \), \( n \) satisfies the union property.

\[
\text{Dom}(n) = \bigcup_{i=1}^{p} \text{Dom}(n^*_i) \tag{4}
\]

**Proof:** Since \( n \) is a parent node of \( N^* \), the domain of \( n \) should contain all domains of its children nodes. Therefore, the domain node of \( n \) should be unions of all its children’s domains.

5.5 Reduced Union Property

A node’s domain in XML(R) can be obtained by unions of its children nodes’ domains. However, there is a special case, that is, without unioning domains, we determine the domain of a node if the node is an Iunit tree which has Xunit trees as children.
Lemma 4: Let $t^i_k$ be an Iunit tree. If $t^i_k$ has $t^j_x$ as child tree, then

$$\text{Dom}(t^i_k) = \text{Dom}(t^j_x)$$

(5)

Proof: Since $t^i_k$ has $t^j_x$ as a child node, $t^i_k$ represents a tuple. By homogeneity property, all attribute nodes which are all children nodes of $t^i_k$ have same domains. Therefore, unions of the same domains is equal to a domain of an attribute.

5.6 Consistency Property

In database systems, keeping consistency is important. It is required to provide a definition of consistency of XML(R).

Lemma 5: If XML(R) satisfies union property, homogeneity, reduced union property, and disjoint property, then XML(R) is consistent.

Proof: Proving this property is straightforward. We use contradictions to prove it.

1. Suppose a node in XML(R) does not satisfy the union property. Since the domain of the node does not hold the union property, there is a portion of the node’s domain such that the portion is not in domains of children nodes or the node’s domain does not have a portion of unions of its children domains. Therefore, XML(R) has missing information.

2. Suppose a node in XML(R) does not satisfy the homogeneity property. Since there exists at least two children nodes such that their domains are different, it does not hold the reduced union property. That is, there is no way to choose a domain for a tuple. Therefore, XML(R) is not consistent.

3. Suppose a node in XML(R) does not satisfy the reduced union property. If it is, there exists at least two children nodes such that their domains are different in the same tuple. It violates homogeneity property. Therefore, XML(R) is not consistent.

4. Suppose a node in XML(R) does not satisfy the disjoint property. If it is, two children nodes have common domain that is non-empty. It violates the definition of attribute, that is, a function of time. Therefore, XML(R) is not consistent.

6 An Example

Suppose we are given a parametric relation, MANAGEMENT as shown in Figure 1 in Section 1. By using transformation scheme, we can transform the relation to an XML representation as shown in Figure 10.

In Figure 10, Subtree $T_A$ represents a temporal domain of “Clothing” department. $T_A$ consists of two Dunit trees and a Pdom element. Each Dunit tree shows a time interval. Therefore, $T_A$ represents a temporal element as unioning two Dunit trees. Subtree $T_B$ in Figure 10 shows an attribute value such that the name of a department is “Toys” and its domain is time interval $[11, 49]$. Subtree $T_C$ shows the disjoint property. Two attribute values are represented as Tunit trees and they are siblings, the domains, $[11, 44]$ and $[45, 49]$, are disjoint. Subtree $T_D$ shows an attribute, Manager. $T_D$ is an Xunit tree and consists of a Para tree and a Tunit tree, which are connected by an Iunit node. Subtree $T_E$ shows a domain of a tuple. In Figure 10, a tuple consists of two attributes-Dept and Manager, the domain of the tuple is the same as one of any attribute’s domain. As we can see in $T_E$, the domain of the tuple is $[11, 49]$, which is the same as the domain of
either Dept or Manager. \( T_E \) shows the reduced union property. In subtree \( T_B \) and \( T_C \), their parent node, Dept and Manager, have the same domain, \([11, 49]\). It shows the homogeneity property. Subtree \( T_F \) shows the domain of XML(R). The domain is the union of domains of tuples as follows:

\[
[11, 49] \cup [41, 47] \cup [71, \text{NOW}] \cup [45, 60] = [11, 60] \cup [71, \text{NOW}]
\]

From this example, we can expect that domain related query can be easily processed without traversing an entire XML(R). For example, if we want to retrieve the domain of a manager whose department is “Toy”, we have to look inside a tuple whose department value is “Toy” and gather domain information for the manager. However, if we use XML(R), retrieving domain of a specific Manager node is enough, that is, it does not need to go through the children nodes of the Manager node.

7 Conclusions

In this paper, we provided an XML representation of the parametric data model handling temporal data. XML is an emerging technology that is used in many different areas. Current trends in the database systems have moved their focus to XML databases that can provide seamless solutions for modeling complex object characteristics.

In the parametric data model, attribute values are stored with their domains, which is the main difference found among other interval-based temporal data models. Because of the nature of the parametric data model, a size-prefixed tuple format is not suitable. This property causes system complexity at the designing level. However, XML can provide an elegant solution to the problem. In order to use XML for the parametric data model, a transformation scheme from the parametric data model to an XML representation, XML(R), is required as well as XML(R) should have the same properties of the parametric data model.

In section 3, we defined several tree types that are constructed by element and value nodes. By using those trees, a transformation scheme was introduced based on the mapping relationship of the parametric data model.

We also discussed that XML(R) has homogeneity, disjoint, union, reduced union, and consistency properties. These properties are required to guarantee that XML(R) is consistent and follows
the properties in the parametric data model.

XML(R) was designed to provide fast answers to domain-specific queries. Since subtrees representing attribute values, attributes, tuples, and relations have their domain nodes, the domains can be retrieved without stepping through an entire XML(R).

References


