Adding Valid Time to XPath

Shuohao Zhang and Curtis E. Dyreson

School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA, United State of America
(szhang2, cdyreson)@eecs.wsu.edu

Abstract. In this paper we extend the XPath data model and query language to include valid time. XPath is a language for specifying locations within an XML document. We extend XPath's data model by adding to each node a list of disjoint intervals or instants that represents the valid time. The valid time for a node is constrained to be a subset of the valid time for a node's parent. We extend the XPath query language with an *axis* to access nodes in a valid-time *view* of the valid time for a node. The view is a calendar-specific formatting of the valid time in XML. By rendering the time in XML, we can reuse non-temporal XPath to extract the desired information within a valid-time axis. The extension is fully backwards-compatible with XPath.

1 Background

The World Wide Web ("web") is the largest and most frequented network of information resources. The majority of the documents on the web conform to the HyperText Markup Language (HTML) [1] but the Extensible Markup Language (XML) [2] is expected to replace HTML as the next-generation markup language for web documents and data. There are several proposed query languages for XML such as Quilt [3], XQL [4], XML-QL [5], and XQuery [6], which is a W3C recommendation. XPath [7] is a subset of XQuery (and of several other recommendations by the W3C, e.g., XSLT [8]). XPath is a language to locate information in an XML document. Every XQuery query contains XPath expressions [9]. XPath operates on a tree-like data model, which is a logical-level data model of an XML document. XPath is a compact, expression-based grammar with a non-XML syntax.

XPath currently lacks temporal semantics. Time is pervasive in the real world. All events happen at some point of time, all objects exist for some interval of time, and most relationships among events and objects evolve over time. Interest in supporting and incorporating time in database systems led to the research area of temporal databases, which has been a quite active area for about twenty-five years [10]. Temporal database research has identified (at least) three dimensions of time: valid time, transaction time and user-defined time. *Valid time* concerns the time in the modeled reality. It is the time when an event occurs or an object is true in the real world. *Transaction*

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

time concerns the system time instead of the real world time. It is the database time when an object is stored in the database. These two dimensions are orthogonal. *User-defined* time is the time in the world that the user specifies. It is an un-interpreted attribute domain of time.

In this paper we extend the XPath data model to include valid time. There have been few papers on this research topic. Grandi and Mandreoli present techniques for adding explicit valid-time timestamps to XML documents [11]. Amagasa et al. propose a temporal extension of the XPath data model [12]. In their data model, valid-time timestamps are added to each edge. In contrast, we extend nodes with valid time. Dyreson was the first to establish a transaction-time XPath data model with special transaction-time axes, node tests and constructors [13].

There are two primary contributions of this paper. First, we propose that a valid time be presented in a *valid-time view*. The view is a calendar- and query-specific rendering of the valid time as a virtual XML document. We observe there are many different calendars each with their own representation of time, and even in a single calendar, the representation of time needs to be flexible. Second, we extend XPath with a *valid-time* axis. The role of the axis is both to provide users with a query language mechanism for accessing valid times and to keep wild-card queries from exploring the valid time. In our model, valid times are isolated in their own dataspace.

As an example, consider the XML document "`bib.xml`" shown in Figure 1. The document contains data about publishers and the books they publish. The valid times of the `<book>` elements are shown in Table 1. The valid times indicate when the book facts represented in the document are true. A user wants to select only the book elements that are valid before the year 2000, where 2000 is a year in the Gregorian calendar. In XPath, the book elements can be located with the following expression.

```
/descendant-or-self::book
```

But how are the valid times reached? In our model the valid-time axis serves to locate the valid time(s) formatted as a virtual XML document as sketched in Figure 3 (note that XPath's abbreviated syntax is used to find the books).

```
//book/valid::time
```

The `valid` axis contains a list of every element in the valid-time view. The `time` node test selects only the `<time>` elements. The times have been located but how can we view the year? We know that the Gregorian calendar view has a `<year>` element that contains the year data as shown in Figure 4, so we can locate that information as follows.

```
//book/valid::time/descendant-or-self::year
```

We could also shorten the expression to directly locate the year information.

```
//book/valid::year
```

Finally, we are interested in determining if the some year comes before 2000. The predicate can be expressed as follows.

```
//book/valid::year[text() &gt; 1999]
```

One important point to observe is that the query

```
//*/time
```

is very different from the following query.

```
//*/valid::time
```

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

The former locates all `<time>` child elements in the document whereas the latter locates only times in the valid-time views. The valid-time axis ensures that all nodes in the valid-time view are isolated from wild-card queries. Furthermore, (non-temporal) XPath can be reused to search within the valid-time view for calendar-specific information.

The remainder of this paper is organized as follows. In the next section a data model for valid-time XPath is developed. The data model extends the information set for a node with a list of times. XPath is then extended with the valid-time axis to query the times and valid-time views to support multiple calendars. The paper concludes with a brief discussion of future work.

2 Data Models

In this section, we briefly outline a data model for valid time. We first present some time-related concepts that serve as background for this research. We then discuss a data model for XML (XPath) and finally extend that data model to support valid time by adding valid time to the nodes in the data model. In the next section we extend XPath with the ability to query the valid-time information.

Time Background

A simple image of time is a directed line. The time-line could be isomorphic to the rational numbers, the real numbers, or the integers, resulting in a *continuous*, *dense* or *discrete* model, respectively [14]. Each point on the line represents a time *instant* and the segment between two points represents a time *interval*. A time *constant* is a pair of times, (b_i, e_i). In a time instant constant, $b_i = e_i$, whereas in an interval constant $b_i \leq e_i$. We assume, without loss of generality, that time is bounded in our model. The starting and ending instants are called *beginning* and *forever* [15].

The XPath Data Model

A well-formed XML document is a collection of nested *elements*. An element begins with a start tag and ends with a paired end tag. Between the tags, an element might contain *content* that is either a text string or other elements. The XPath data model is commonly assumed to be an ordered tree. The tree represents the nesting of elements within the document, with elements corresponding to nodes, and element content comprising the children for each node. Unlike a tree, the children for a node are *ordered* based on their physical position within the document. The XPath recommendation [7] does not provide a formal model.

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

Below we give one possible data model that omits details extraneous to the aims of this paper.

Definition [XPath data model] The *XPath data model*, *D*, for a well-formed XML, *X*, is a four-tuple $D(X) = (r, V, E, I)$ where,

- V is a set of nodes of the form (i, v) where v is the node identifier and i is an ordinal number such that for all $(i, v), (j, w) \in V$, v starts before w in the text of X if and only if $i < j$.
- E is a set of edges of the form (v, w) where $v, w \in V$. Edge (v, w) means that v is a parent of w. In terms of X, it represents that w is in the immediate content of v. There is an implied ordering among the edges; edge (v, y) is before edge (v, z) if $y < z$ in the node ordering.
- The graph (V, E) forms a tree.
- I is the information set function which maps a node identifier to an *information set*. An information set is a collection of properties that are generated during parsing of the document. For example, an element node has the following properties: *Value* (the element's name), *Type* (element), and *Attributes* (a set of name-value pairs, in XPath, attributes are unordered).
- $r \in V$ is a special node called the root. Note that r is the data model root rather than the document root. The document node is the first element node of the document. A document may also contain processing instruction nodes or comment nodes that precede the document root.

Figure 2 depicts a fragment of the data model for the XML document "bib.xml" shown in Figure 1. For brevity, not all the information is included in this figure. Each node is represented as an oval and the corresponding ordinal number is inside it. Each edge is represented as a straight line and the first three levels of edges are shown beside the corresponding lines. Note that the ordinal number of a node is unique in a document while that of an edge is not. This is because the node ordinal numbers reflect the order of all the nodes in the whole document, while the edge ordinal numbers only differentiate edges that emanate from the same node.

The information in the data model of "bib.xml" is sketched below (part of which is not displayed in Figure 2).

V = ((0,&0),(1,&1),…,(28,&28))
E = ((&0,&1),(&1,&2),(&1,&12),…,(&27,&28))
I = ((&0, (Value=root, Type =root, Attributes="")),
(&1, (Value=db, Type =element, Attributes="")),
(&2, (Value=publisher, Type =element, Attributes="")),
(&3, (Value=name, Type =element, Attributes="")),
(&4, (Value="ABC", Type =text, Attributes="")),
…
(&4, (Value="59.99", Type =text, Attributes="")))
r = (0,&0)

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

Valid-time XPath

We have established the data model for XPath. But this data model contains no support for time. To define the data model we first need a uniform representation scheme. Usually, a node is valid at a point of time or an interval of time. But more generally, a node could be valid at several points and (or) intervals of time, e.g., a *temporal element* [15]. A combination of valid-time constants should be allowed. We therefore define the valid time of a node to be a list of time constants.

```
<db>
  <publisher>
    <name>ABC</name>
    <book>
      <isbn>1234</isbn>
      <title>book1</title>
      <price>19.99</price>
    </book>
  </publisher>
  <publisher>
    <name>XYZ</name>
    <book>
      <isbn>2345</isbn>
      <title>book2</title>
      <price>29.99</price>
    </book>
    <book>
      <isbn>5678</isbn>
      <title>book5</title>
      <price>59.99</price>
    </book>
  </publisher>
</db>
```

Figure 1 The document "bib.xml"

Definition [node valid-time representation] The *valid time* of a node in an XML document is represented as a list of time constants, $[t_1, t_2,\ldots,t_n]$, where each t_i (i = 1,…,n) represents a time constant when the node is valid.

- Each time constant $t = (b_i, e_i)$ is either a time interval or a time point.
- All and only the time constants when the node is valid are included in this list.
- Any two time constants do not overlap, no matter if they are time intervals or time points, i.e., $(b_i , e_i) \cap (b_j, e_j) = \emptyset$, $\forall i \neq j$, $1 \leq i, j \leq n$.
- These time constants are ordered by the valid time contained in each of them, i.e., $b_{i+1} \leq e_i$, (i = 1,…,n-1).

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

It is required that the time constants in any valid-time list be disjoint. Otherwise, the constraint cannot be satisfied that these time constants are ordered. For example, time intervals (1,3) and (2,4) cannot be ordered; neither can time interval (1,3) and time point (2,2). Order is important in an XML document.

We are now in a position to define a valid-time XPath data model.

Definition [valid-time XPath data model] The *valid-time XPath data model*, D_{VT}, for a well-formed XML document, X, is a four-tuple $D_{VT}(X) = (r, V, E, I)$.

- V is a set of nodes of the form (i, v, t) where v is the node identifier, i is an ordinal number and t is a valid time such that the following two conditions hold.
  1. For all (i, v, t), $(j, w, s) \in V$, v starts before w in the text of X if and only if $i < j$.
  2. For all $(i, v, t) \in V$ and its children (i_1, v_1, t_1), (i_2, v_2, t_2),…, $(i_n, v_n, t_n) \in V$, $t_1 \cup t_2 \cup \ldots \cup t_n \subseteq t$.
- E, I, and r are the same as the non-temporal XPath data model.

Every node in the valid time data model is associated with the valid time that represents when the node is valid. A node's valid time constrains the valid time of other nodes in the model. No node can exist at a valid time when its parent node is not valid. The following things can be further inferred from the data model definition,

- The valid time of any node is a superset of the union of the valid times of all its children.
- The valid time of any node is the superset of the union of the valid times of all its descendants.
- The valid time of the root node is the superset of the union of the valid times of all the nodes in the document.

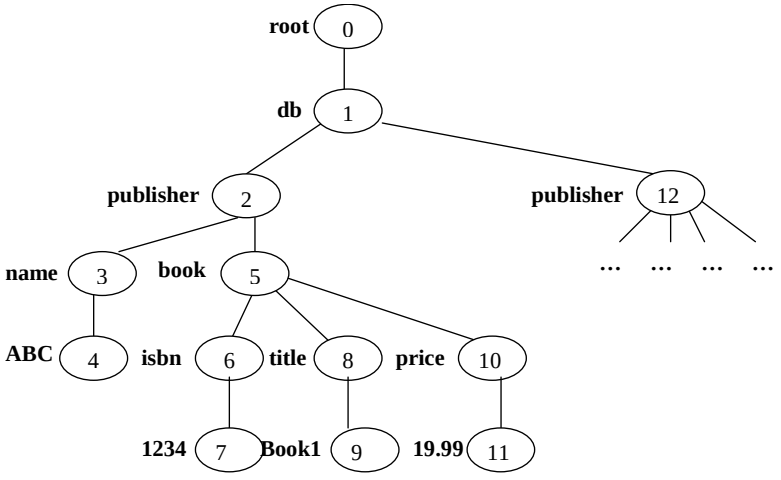

Figure 2 The data model for "bib.xml"

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

In the valid-time XPath data model, only nodes have valid time information. The edge set E is the same as it is in the non-temporal data model. It does not mean that edges are not associated with valid time information, rather the valid time information is implied since only nodes can have information in an XML data model. The valid time of an edge is in fact determined by the valid time of the nodes at the edge's two ends. An edge can only exist at the time when both nodes are valid. For an edge (n, v, w) $\in$ E that emanates from the parent node (i, v, t_i) to the child node (j, w, t_j), its valid time $t = t_i \cap t_j$. Since $t_j \subseteq t_i$ (as mentioned above in the data model definition), the valid time of an edge is always the same as the child node: $t = t_j$.

We again use the XML document "bib.xml" shown in Figure 1 as an example. The only difference here is the node set V and therefore only some nodes are shown (other components are identical to those described prevously). Table 1 shows the time information for the "book" nodes in "bib.xml". The node set, V, containing the three "book" nodes is given below.

V = ((0, &0, t_0),…,(5,&3, t_5),…,(15,&15, t_{15}),…,(25,&25, t_{22}),…)

t_5 = [("Jan 31,1999", now)]

t_{15} = [("Jan 31,2000", "Dec 31, 2000"), ("Jan 1, 2001", now)]

t_{22} = [("Jan 31,2002", now)]

The important thing to note is that the valid time information must satisfy the constraint in the data model definition. For example, the valid time of the root node, t_0, should be a superset of the union of the valid times of all the nodes in "bib.xml".

Book (ISBN)	Valid time
1234	[("Jan 31,1999", now)]
2345	[("Jan 31,2000", "Dec 31, 2000"), ("Jan 1, 2001", now)]
5678	[("Jan 31,2002", now)]

Table 1 Valid-time information for "book" elements in "bib.xml"

3 Querying Valid Time

In this section we present extensions to XPath to query the valid time information. The semantics is *unsequenced*. In a sequenced semantics a query would (logically) be evaluated in every valid-time snapshot simultaneously. Sequenced queries often do not have explicit valid-time constraints. We extend XPath with an axis to locate the valid time for a node. The valid time can be *viewed* as an XML document in any calendar that the user has defined.

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

Review of XPath

An XPath expression is a sequence of steps. Each step may have four portions, a *context*, an *axis*, a *node text* and some *predicates*. The *context* is the starting point of the evaluation. The *axis* specifies a certain relationship between the nodes selected by the axis step and the context node. For example, a `child` axis selects the nodes that have a "child" relationship with the context node, i.e., each selected node is a child of the context node. The *node test* specifies the node type of the nodes selected by the axis step. The symbol "`::`" is used to syntactically separate a node test from an axis. A *predicate*, enclosed in square brackets, further filters a node sequence, retaining some nodes and discarding others. There can be zero or more predicates in one step. The nodes selected after the axis and the node test are finally evaluated against the predicates in the order that they appear in the step. A simplified syntax for a step in XPath is given below.

$\texttt{axis::node test[predicate}_1\texttt{]}\ldots\texttt{[predicate}_n\texttt{]}$

Below is an example of an expression that locates all the `<book>` elements that are children of a `<publisher>` element.

```
/descendant-or-self::publisher/child::book
```

The first step starts at the data model root, and locates all publishers by following the descendant-or-self axis to every node, and then applying a node test to keep only publishers. The second step finds the children relative to each publisher and keeps only those children that happen to be books.

Valid-time Axis

The valid time axis of a node contains the valid-time information of the node as if it had originated from an XML document. The goal is to provide *flexible* representation for the valid time of a node in XML, and to provide the means by which users can customize their view of the valid time. In this section, the concepts of a *valid-time view* and *default valid-time view* are introduced, followed by the definition and syntax of the valid-time axis. We also address a specific problem of how different calendars can be used in views desired by different users.

An axis returns a list of nodes relative to the context node. The valid-time axis returns a list of nodes that forms the valid-time information of the context node. The valid time of a node can be viewed as an XML document.

Definition [node valid-time view] A valid-time list can be viewed as an XML document. Let v be a node in the data model tree of an XML document. The valid-time view, V, is a mapping from the valid time for v to an XML data model, X, denoted $V(v) = X$.

Each node has a corresponding view containing its valid time information. A node valid-time view of a node x is shown in Figure 3. The root node in the view serves as a handle that the user can use to access the valid time information. Each time in the

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

valid-time list is denoted as a <time> element. The content of the <time> element is unique to the view (specific examples are given later in this chapter).

It is assumed that each node in an XML document has a corresponding valid-time view. These views are created and maintained by the system that stores the XML document. The user should be able to query the views to extract the desired valid-time information of the nodes.

```
<validTime>
  <time>
    view of t1
  </time>
  <time>
    view of t2
  </time>
    …
  <time>
    view of tn
  </time>
</validTime>
```

Figure 3 A node valid-time view document

Figure 3 does not show what is exactly contained in each view. To be a valid XML document, each view must contain XML. The user, often the creator of the valid-time view documents, can define specific views. For example, Figure 4, shows the valid-time view in the commonly used Gregorian calendar. “Year”, “month” and “day” element nodes are nested under “begin” and “end” of each view; text nodes representing the “year”, “month” and “day” information are further nested.

```
<time>
  <begin>
    <day>31</day>
    <month>1</month>
    <year>2000</year>
  </begin>
  <end>
    <day>31</day>
    <month>12</month>
    <year>2000</year>
  </end>
</time>
```

Figure 4 A valid-time view in Gregorian calendar

Though the valid-time view can be user-defined, it is useful to have a default valid-time view provided by the system. The default valid-time view uses the system time,

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

commonly seconds from the origin, to represent the “begin” and “end” time points. Therefore, a text node is nested under any “begin” or “end” node in the default valid-time view document. This default view has several advantages. Because the system time is commonly used in computer systems to represent time, a valid-time view can be more easily created by the computer system. It is also easy to compare since the system time is represented as a number. Time information can be easily extracted because there is no additional sub-tree under the “begin” and “end” nodes.

The default valid-time view for t_1 is shown in Figure 5, where b_1 and e_1 are represented with the system time. The default valid-time view is used when there is no user-defined view.

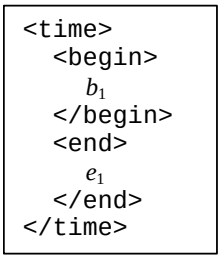

Figure 5 The valid-time view is used in a valid-time axis

Definition [valid-time axis] The *valid-time axis* selects the list of nodes that forms a document-order traversal of the *valid-time view*.

Syntax [valid time axis] `v/valid` specifies the valid-time axis of the node `v`.

From the definition, the nodes in the valid time axis is ordered in document order traversal of the valid-time view. 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.

Multi-calendar Support

The valid time is the real world time, which may be represented in various calendars used by people in different regions. It should not be restricted to any fixed type of representation. The most commonly used calendar in computer systems is Gregorian calendar. In this section, the study of Chinese calendar helps us understand the multi-calendar support in valid time axis.

The Chinese calendar is based on a combination of lunar and solar movements. The lunar cycle is about 29.5 days hence a lunar year consisting of 12 months is approximately 12 x 29.5 = 354 days. So a lunar year is about 11 days shorter than a solar year. In order to “catch up” with the solar calendar a leap month is inserted once every few years. To be precise, there are seven leap months, called an intercalary month, in each nineteen-year cycle. This is the same as adding an extra day on a leap year in Gregor-

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

ian calendar. This is why, according to the solar calendar, the Chinese New Year falls on a different date each year.

Let us consider the example document "bib.xml" (shown in Figure 1). The valid time in Table 1 is represented in Gregorian calendar. However, there are other calendars that are widely used by people in different regions. For instance, we can also represent the valid time in Chinese calendar as shown in Table 2. For the convenience of the readers to convert between the two calendars, Table 3 lists the recent Chinese new-year days (the first day of a year in Chinese calendar) in Gregorian calendar. A book element is valid from its publishing date to current time (denoted *now*). Therefore, if the specified calendar is Gregorian calendar, then the valid time of the book element with ISBN "1234" is [("Jan 31,1999", "*now*")]. Similarly, its valid time in Chinese calendar is [("Month 12, Day 19, Year Wu-yin (Cycle 78)", "*now*")]. They are lists both consisting of one time constant. The corresponding node valid-time views are shown in Figure 6 and Figure 7.

Book (ISBN)	**Publishing date**	
	Gregorian	**Chinese**
1234	Jan 31,1999	Cycle 78, Year Wu-yin, Month 12, Day 15
2345	Jan 31,2000	Cycle 78, Year Ji-mao, Month 12, Day 25
5678	Jan 31,2002	Cycle 78, Year Xin-si, Month 12, Day 19

Table 2 Publishing date in different calendars

Year in Chinese calendar		**Chinese new-year day in Gregorian calendar**
Chinese name	**Western name**	
癸未	Gui-wei	February 1, 2003
壬午	Ren-wu	February 12, 2002
辛巳	Xin-si	January 24, 2001
庚辰	Geng-chen	February 5, 2000
己卯	Ji-mao	February 16, 1999

Table 3 Recent Chinese new-year days in Gregorian calendar

The special symbol "*now*" should be interpreted into a form that conforms to the corresponding beginning time. For example, if the beginning node has three child

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

nodes, “year”, “month” and “day”, then the sub-tree starting from the end node should be like that shown in Figure 7.

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

Figure 6 A node valid-time view in Gregorian calendar

```
<validTime>
  <time>
    <begin>
      <day>19</day>
      <month>12</month>
      <year>Wu-yin(Cycle 78)</year>
    </begin>
    <end>
      now
    </end>
  </time>
</validTime>
```

Figure 7 A node valid-time view in the Chinese calendar

```
<end>
  <day>now</day>
  <month>now</month>
  <year>now</year>
</end>
```

Figure 8 A possible representation of “now”

Before a user queries an XML document, he or she should be informed of what type of calendar is supported. This may be specified in the document itself (as a processing instruction for example) or in the schema that the document that it must con-

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

form to. The user is then able to specify which calendar to use in the query. A proposed syntax for specifying calendar is given below.

Syntax [calendar in valid time axis] `v/valid("calendar")` specifies that the calendar to use in the valid time axis of `v` is `"calendar"`.

It is the responsibility of a system to provide the user with a list of supported calendars. For instance, it may support Gregorian calendar, Chinese calendar and Russian calendar. The system should verify if a calendar required by the user is available. It may also use a default calendar (which should probably be Gregorian calendar) since some users may not be interested in a specific calendar.

Examples

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

`v/valid::day` selects all the `day` nodes in the axis.

`v/valid::time[2]` selects the second `time` node in the axis. It contains the second time constant during which the context node was valid. Note that here `[2]` is an abbreviated syntax for `[position()=2]`.

`v/valid::begin[1]` selects the first `begin` node in the axis. It contains the time point when the context node first became valid.

If a user has knowledge of a valid-time view, such as that of the Gregorian calendar, then they can make node tests such as `v/valid("Gregorian")::year`, `and v/valid("Gregorian")::month`.

Conclusions

This paper presents a simple scheme to add valid-time support to XPath. The support results in an extended data model and query language. Adding a list of valid times to each node extends the data model. A valid-time axis is added to the query language to retrieve nodes in a view of the valid time for a node. The benefit of adding an axis to reach valid-time information is that the nodes cannot be reached with existing axis (such as the descendant axis). Hence, the extension is fully backwards compatible with XPath. The valid-time view is a formatting of the valid time in XML. This allows XPath to be reused to query within a valid-time axis. Multiple calendars are supported through different valid-time views.

In future, we plan to extend XPath with temporal predicates and aggregates. We can add temporal predicates to permit some reasoning about the valid-time information, e.g., the complete set of predicates found in *Allen's temporal logic* [16]. We are also exploring operations to perform temporal aggregates, e.g., moving-window averages. Finally, implementation is an immediate goal. We plan to extend Xalan, which is Apache's XPath processing engine, with the valid-time axis and view.

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.

References

[1] Dave Raggett, Arnaud Le Hors, and Ian Jacobs, *HTML 4.01 Specification*, W3C Recommendation, 24 December 1999, <http://www.w3.org/TR/html4/> (30 April 2002).

[2] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6 October 2000, <http://www.w3.org/TR/REC-xml> (30 April 2002).

[3] Don Chamberlin, Jonathan Robie, and Daniela Florescu, *Quilt: an XML Query Language for Heterogeneous Data Sources*, Proceedings of WebDB 2000, Available at <http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html>(30 April 2002).

[4] J. Robie, J. Lapp, and D. Schach, *XML Query Language (XQL)*, <http://www.w3.org/TandS/QL/QL98/pp/xql.html> (30 April 2002).

[5] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu, *A Query Language for XML*, <http://www.research.att.com/~mff/files/final.html> (30 April 2002).

[6] Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu, *XQuery 1.0: An XML Query Language*, W3C Working Draft, 20 December 2001, <http://www.w3.org/TR/xquery> (30 April 2002).

[7] James Clark, and Steve DeRose, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 December 1999, <http://www.w3.org/TR/xpath> (30 April 2002).

[8] Michael Kay, *XSL Transformations (XSLT) Version 2.0*, W3C Working Draft, 20 December 2001, <http://www.w3.org/TR/xslt20> (30 April 2002).

[9] Ashok Malhotra, Jim Melton, Jonathan Robie, and Norman Walsh, *XQuery 1.0 and XPath 2.0 Functions and Operators*, W3C Working Draft, 20 December 2001, <http://www.w3.org/TR/xquery-operators/ > (30 April 2002).

[10] Vassilis J. Tsotras and Anil Kumar. Temporal database bibliography update. *ACM SIGMOD Record*, 25(1):41-51, March 1996.

[11] F. Grandi and F. Mandreoli, The Valid Web: it's Time to Go, Technical Report 46, Time-Center, Aalborg, Denmark, December 1999.

[12] T. Amagasa, Y. Masatoshi, and S. Uemura, A Data Model for Temporal XML Documents, Database and Expert Systems Applications, 11th International Conference, DEXA 2000, pages 334-344, London, UK, September 2000.

[13] Curtis E. Dyreson, Observing Transaction-time Semantics with TTXPath, *Proceedings of the Second International Conference on Web Information Systems Engineering (WISE2001)*, December 2001, Kyoto, Japan.

[14] Richard T. Snodgrass, *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers, 1995.

[15] Christian S. Jensen, Curtis E. Dyreson (editors), et al, *The Consensus Glossary of Temporal Database Concepts—February 1998 Version*, February 1998, <http://www.cs.auc.dk/~csj/Glossary/download/1399ch52.ps> (30 April 2002).

[16] J. F. Allen, Maintaining Knowledge about Temporal Intervals, *Communications of the Association of Computing Machinery*, 26, No. 11, November 1983, page 832-843.

Lecture Notes in Computer Science XXXX, Database and Network Information Systems, Proceedings of DNIS 2002, Aizu, Japan, December 2002, pp. XX-XX.

Copyright © Springer-Verlag. All rights reserved.