The Valid Web: it's Time to Go...

Fabio Grandi and Federica Mandreoli

TR-46

A TimeCenter Technical Report

Title The Valid Web: it's Time to Go...

Copyright © 1999 Fabio Grandi and Federica Mandreoli. All rights reserved.

Author(s) Fabio Grandi and Federica Mandreoli

Publication History October 1999. Manuscript.
December 1999. A TIMECENTER Technical Report.

The Valid Web: it's Time to Go...

Abstract

The development of the World Wide Web technology on the Internet is a major achievement of computer research in recent years. The great availability of easy-to-access on-line hypermedial documents has been a real revolution in the information world, also for its important social, cultural and economical consequences. However, a scarce attention has been devoted so far to the temporal aspects of the World Wide Web definition and technology, although time undoubtedly plays a pervasive and fundamental role in the reality and in its information counterpart. Moreover, although the management of time-varying information is a consolidated research issue in the database field, no integrations between the World Wide Web and database worlds has been attempted in this respect yet. Notwithstanding, this happens while there is a substantial convergence between the two worlds, with a renewed interest on the XML emergence as the upcoming standard for the representation and exchange of semistructured data on the Internet.

In this work, we present a temporal extension of the World Wide Web by defining a complete XML/XSL infrastructure to support valid-time. The proposed valid-time versioning scheme allows the explicit definition of temporal information within HTML/XML documents, whose contents can then be selectively accessed on the basis of the validity coded within them. By acting on a navigation validity context, the proposed solution makes it possible to navigate through time in a given virtual environment with any XML-compliant browser; this allows, for instance, to cut personalized visit routes for a specific epoch in a virtual museum or a digital historical library, or to visualize the evolution of an archaeological site through successives ages, or to selectively access past issues of magazines, stock quote archives, etc. Moreover, the proposed temporal extension can also be applied in a straightforward way to generic XML-encoded semistructured data, laying the foundations for the management of temporal XML data and the development of temporal XML query languages. In particular, the realization of a temporal virtual museum and the management of temporal XML data (via TSQL2-like query facilities) with the proposed temporal extensions have been shown feasible through a prototype demonstration World Wide Web site, which will also be described in the paper.

1 Introduction and Related Work

A great deal of work has been done in recent years in the field of Temporal Databases (TDBs) [22, 11, 40, 9, 23]. Due to this effort, a large infrastructure (namely data models, query languages, index structures, etc.) has been developed for the management of data evolving in time, for which successive versions need to be maintained rather than being overwritten or discarded by destructive changes. Two well-known to date kinds of time are usually considered in the literature [9]: *transaction-time* and *valid-time.* The former concerns the evolution of data with respect to the system where data are stored and the latter concerns the evolution of data with respect to the application reality that data describe. By the way, research interests on temporal information have been almost focused on highly structured data (e.g. relational or object-oriented), and, for instance, no textual data or less structured multimedia documents have been diffusely considered for temporal extensions so far.

On the other hand, the World Wide Web (WWW, W3 or Web [3]) is a large distributed collection of hypertextual and multimedia irregular documents, available on-line on the Internet. Virtual navigation from site to site is made possible, in a transparent way, by following hypertextual links (Uniform Resource Locators [33]) found in browsing the documents. Almost every Web documents are currently written and formatted according to the HTML standard [27], which is a *markup* language based on the SGML formalism [35]. However, the eXtensible Markup Language (XML [25]), which has been designed to overcome the main limitations of HTML and has been recently recommended by the W3C Consortium [39], is becoming the new emerging standard for publishing documents over the Web. The browsing of XML documents can be best enjoyed by means of provided *stylesheets*. In particular, the eXtensible Stylesheet Language (XSL [26]) allows the definition of XML document transformation rules [46] and the specification of formatting semantics. Moreover, from a database research perspective, XML can also be considered as a data modeling language [16], suitable to describe *semistructured data*, that is data which can be irregular or incomplete and whose structure may rapidly or unexpectedly change. For this reason, XML is also likely to become the future dominant standard for data representation and exchange on the Internet.

The quickly growing popularity of the Internet, chiefly due to the Web itself, has gained momentum for a consistent investment of intellectual (and financial) resources on research and development of its technology. Although the functionalities of the Web, including markup language potentialities and browser capabilities, have been lately greatly increased, scarce attention has been so far devoted to the *temporal* aspects. The only necessity actually acknowledged in the Web community is for the support of resource *versioning*, particularly in a scenario of collaborative authoring and concurrent document editing [21, 41]. Hence, also standardization efforts have been made on these problems by the IETF WebDAV (Web Distributed Authoring and Versioning) Working Groups [38], involving formalization of requirements, HTTP protocol [28] extensions, searching facilities for DAV repositories, versioning and configuration management support. The kind of versioning considered there is on the system side: the maintenance of versions reflects the history of modifications applied to the documents, as committed by authors to the document store (remote server). Therefore, in TDB terminology, the temporal dimension involved is the *transaction time*, although the temporal aspects related to resource versioning have not explicitly been recognized as such.

However, Web documents may also contain intrinsically temporal (i.e. *historical*) information, where the validity is a constitutive part of the information itself. The temporal dimension involved is, this time, the *valid time*. This aspect has not been taken into account with the necessary worth yet and the main purpose of the present work is to fill this gap. Notice that one of the main problems related to the uncontrolled growth of the information amount available on the net is to provide some aid to the users not to “get lost”. The adoption and use of the validity temporal dimension could be a good tool to improve the selectivity of a network information search, which could also enhance the usefulness and functionality of already available navigation aids (e.g. Web search engines). The setting of a suitable validity coordinate in browsing a Web site containing historical information (e.g. a digital library or a virtual museum, but also an archive of newspaper issues or stock quotes) would improve the search quality: only relevant information would be displayed, instead of being immersed in a mass of temporally heterogeneous and non-pertinent stuff.

A third temporal dimension is also already of interest in the Web community, that is the time dimension intrinsic to multimedia data like music, movies, animated presentations and others, involving *streams* of information. The issue is not very central indeed in the context of TDBs (e.g. it has been dealt with in [7]), but has been taken into account in hypertext research. For instance, the HyTime proposal [14] mainly consists of an SGML extension to represent *time-based* documents. Also more recent proposals (e.g. based on SMIL 1.0 [36]) took multimedia synchronization problems into account. This third time dimension involves temporal aspects which can be considered from our perspective as “internal,” local to the representation and usage of some special types of data. Therefore, such time dimension is quite orthogonal to the two time dimensions usually considered in temporal database research (it could be labelled as a *user-defined time*, in TDB terminology), although possible interactions would deserve a thoroughly investigation in future work.

The contribution of this paper concerns the introduction of valid-time into the World Wide Web to support the management of historical information and temporal structured or semistructured data. Two reference applications will be introduced in Sec. 2 to be used in the rest of the paper. Section 3 is devoted to the technical core of our proposal: we will present a complete XML/XSL infrastructure to embody temporal information into Web documents and put temporal navigation and query facilities at Web user's disposal. The solution we propose does not require changes in the current Web technology as it is based on XML and related standards. The adoption of a suitable XML schema for document timestamping and a provided XSL stylesheet for temporally selective document processing will enable any XML-compliant browser, like Microsoft Internet Explorer 5 (Ie5 [32]), to support temporal documents.

The proposed Web extensions have also been tested on a prototype implementation which will be briefly described in Sec. 4. Our implementation also largely exploits another powerful Web technology supported by Ie5: the Document Object Model (DOM [24]), which is an application programming interface (API), which allows dynamic manipulation of HTML and XML documents (e.g. via scripting languages like JavaScript [31]).

Conclusions can finally be found in Sec. 5.

2 Two Reference Sample Applications

The proposal outlined in this paper concerns the addition of the valid time dimension to Web documents. Typical class of documents we consider in this context are classical HTML-based hypermedia documents (Web pages) containing irregular data, or XML-based documents containing structured or semistructured data. In the former case, the proposed extension is finalized to support temporal navigation in virtual environments which are sources of historical information. An extremely appropriate example of such an environment is a Web museum, which will be used as a first reference application in the rest of the paper. In the latter case, the extension is aimed at supporting the management of temporal data, which are represented and exchanged as XML documents. As reference example of this second kind of use of our approach, we will consider the representation of relational data concerning employees, the very most common example used in the TDB literature. The support of queries involving all the temporal selection predicates available in a language like TSQL2 [19] will be considered in this case.

2.1 The Temporal Web Museum

The number of Internet museum sites available on the Web is increasing day by day. For instance, several famous real museums have got their official Web sites:

- The Louvre Museum in Paris, URL: `http://mistral.culture.fr/louvre/`
- The Uffizi Galleria in Florence, URL: `http://www.uffizi.firenze.it/`
- The Vatican Museums in Rome, URL: `http://www.christusrex.org/www1/vaticano/`
- The Guggenheim Museums, URL: `http://www.guggenheim.org/`
  (including Manhattan, Soho, Berlin, Bilbao and Venice sites)
- The Metropolitan Museum of New York, URL: `http://www.metmuseum.org/`
- The Museum of Modern Art in New York, URL: `http://www.moma.org/`

The main purpose of these sites is the presentation of the corresponding real institutions (providing general information, plans, samples from the main collections, etc.), the advertisement of current initiatives, till the merchandising of museum gadgets and publications. Moreover, private or corporate art collections may be found on the Web. Scientific journals (e.g. *Archives and Museum Informatics*) and international conferences devoted to Web museums (e.g. *Museums and the Web*) also exist [2].

EMPLOYEE

ID	*Name*	*D-Birth*	*Department*	*Salary*	*Valid Time*
T1	Tom	8/15/51	Marketing	27K	[1996-04-15 – 1998-09-19]
T1	Thomas	8/15/51	Accounting	28K	[1998-09-20 – 1999-05-31] ∪ [2000-01-01 – *now*]
A1	Ashley	5/25/59	Engineering	30K	[1994-01-01 – 1995-12-31] ∪ [1997-03-01 – 1999-11-30]
A1	Ashley	5/25/59	Finance	35K	[1996-01-01 – 1997-02-28]

Table I: The Employee Temporal Table.

On the other hand, purely *virtual* museums, that is without a physical counterpart, are also available on-line. For instance, one of the very first sites to appear, and one of the most popular now (mirrored on several servers throughout the world), is the "Web Museum" [15] authored and maintained by Nicholas Pioch, also known as the "Web Louvre" before the official Louvre site was opened. It is basically a collection of image data representing famous paintings, heterogeneous as to their origin, which can accessed, for example, via an artist or a theme index. The Web Museum has been used to test our proposal: we realized a temporal version of a subset of the Web Museum pages and developed a Web environment for the temporal browsing of its collections.

A related application is also the production of a virtual archaeological site, based on the finds, reliefs and reconstructions of an underlying real site. Virtual sightseeings can be made available by means of hypermedial plans (e.g. topographic, thematic, historical), or also through *virtual reality* environments, for instance based on VRML code [37]. As a virtual reality application makes it possible an interactive spatial trip in the modeled scenario, the integration of time into the Web would add also the temporal dimension to the navigation, with the site changing through the ages when the user moves along the time axis. This is the approach taken, for instance, in the NU.M.E. (New Museum Environment) project [34], where the time dimension has been added to a VRML environment. The integration between such approaches and the temporal extension we introduce here is beyond the scope of this work.

2.2 The Temporal XML Employee Data

As far as the management of employee data is concerned, we will consider the basic functionalities of a relational temporal database. To this aim, we start from a temporal table like the one in Tab. I, adopting tuple time-stamping with a *temporal element* (namely a disjoint union of time intervals), according to the BCDM temporal data model [10]. Each tuple represents a fact concerning an employee and the *Valid Time* attribute represents when the fact is (was or will be) true. The *ID* attribute is assumed to be a time-invariant system-defined identifier (a surrogate in the TSQL2 data model), which guarantees the identification of employees even when their key value is subject to change (as it happens for Tom's name in the example). Temporal queries can be issued on such data to extract temporal information, for instance, by selecting tuples on the basis of their validity (temporal selection). Without loss of generality, we adopt as reference language TSQL2, which is widely known, derives from a substantial consensus effort and, thus, summarizes most of the functionalities of other temporal query languages. The class of simple queries (on one relation) we will consider is of the following kind:

```
SELECT <target-list>
   FROM Employee
      WHERE VALID(Employee) <temporal-predicate> Validity_Context
```

where Validity_Context is a time interval defined by the user and <*temporal-predicate*> is one of the available operators for comparison between a temporal element and an interval (namely OVERLAPS, EQUALS, PRECEDES, CONTAINS, MEETS[1]). Obviously, we also consider dual queries with the reversed order of the operands of the temporal predicate, when this is not symmetric (i.e. PRECEDES, CONTAINS, MEETS). For instance, the query:

```
SELECT * FROM Employee
  WHERE PERIOD( DATE('1994-01-01'), DATE('1994-12-31') )
      PRECEDES VALID(Employee)
```

selects all the employee data whose validity follows the [1994] time interval (all the tuples but the third one in Tab. I).

If the VALID(Employee) timestamps are also intervals, the simple queries we consider allow to test any of the thirteen possible relative relationships with the Validity_Context (being TSQL2 *temporally complete* in this respect, as equivalent to Allen's algebra [1]).

3 Integrating Valid Time into the Web

In this Section we outline our proposal concerning an XML/XSL infrastructure for the definition and use of valid-time temporal Web documents. Generally speaking, the adoption of valid time is aimed at allowing the management of historical information (past, present or future). Historical information must explicitly be coded within the Web documents, to be selectively accessed during temporal browsing. To this purpose, distinct parts of a Web document can be timestamped with their own validity during the document creation.

In a Web museum, selective browsing would allow the definition of personalized visit paths through centuries and artistic or historical periods within the museum collections. In order to plan a visit (virtual or real), we could act on valid time selection to change the historical period of interest. For instance, we can choose the High Renaissance period, by selecting the validity range 1495–1520. Hence, we may start our virtual visit entering some virtual hall or gallery: only the temporally relevant paintings or sculptures would be present; by changing the validity context we could see some works vanish and some different works materialize. For example, in a hall dedicated to the Italian High Renaissance, we could view the evolution of the painting styles of Leonardo da Vinci, Raphael, Michelangelo and Titian and, say, have a look to works contemporaneous to the Mona Lisa picture.

3.1 Defining Valid Temporal Documents

Valid-time versioning is aimed at the production of *temporal* Web documents, that is containing (historical) information with different validities. To this purpose, Web resources need to be especially designed as temporal, and the timestamping of time-varying information has to be explicitly coded by the authors in the document creation phase. The granularity of temporal versioning we consider is finer than the Internet resource (identified by a URL/URI [33]), that is single pieces of Web documents (i.e. text paragraphs, images, links...) can be assigned different validities. The addition of valid time to Web documents we propose is based on the extensions of the XML markup language [25] with timestamping tags. The functionalities of the new tags can be fully specified by means of suitable XML schemas and stylesheets, without requiring modifications to the Web browsers supporting XML (like Ie5).

[1]Actually, TSQL2 defines MEETS as a comparison operator between *intervals* only. Therefore, the clause "VALID(Employee) MEETS Validity_Context" should more properly be written as "LAST(VALID(Employee)) MEETS Validity_Context", where the LAST operator extracts the last interval from a temporal element. Hence, we intend above a slightly extended MEETS operator to work in this way with an element. We apologize for the little abuse.

In particular, our proposal consists of the addition of a new XML tag, `<valid>`, to define a *validity context*. The validity context is used to timestamp a part of a document, to which it assigns a specific time pertinence which can then be used for temporally-selective document manipulation. The contents embraced in a validity context can be of any allowed kind (namely text, graphic elements, and any other XML structure including nested `<valid>` elements). The timestamps can be specified in a validity context by means of `<validity>` tags, which allow the definition of a temporal interval through its boundaries (i.e. the values of the `from` and `to` attributes of the `<validity>` XML element). In general, multiple intervals can be used: in this case, the timestamp is defined as the union of all the validity intervals specified; formally, the timestamp is a *temporal element* as defined in the BCDM temporal data model. For instance, the following code:

```
<!-- definition of a validity context -->
<valid>
  <!-- valid-time timestamping -->
  <validity from="1980-01-01" to="1985-12-31" />
  <validity from="1995-01-01" to="2000-12-31" />

  This is text <b>valid from 1980 to 1985</b>
    but also <b>valid from 1995 to 2000</b>...

</valid>
```

defines a validity context whose validity is $[1980\text{–}1985] \cup [1995\text{–}2000]$. The time constants are specified according to the ISO 8601 format [4], corresponding to the XML `date` data type. The introduction of the validity context, including the definition of the required new tags, is effected via the XML schema, named `ValidSchema.xml`, displayed in Fig. 1. Such a schema must be included in any XML temporal document to enable validity contexts, in the way usual for XML-Data [42] schemas:

```
<?xml version="1.0" ?>
<TemporalDoc xmlns:dt="x-schema:ValidSchema.xml">

      the rest of the document, with timestamped <valid> elements,
      including any XML and (well-formed) HTML mark-up

</TemporalDoc>
```

Notice that the use of an XML schema ensures syntactic checks on the well-formedness of temporal documents to be automatically effected by the XML-enabled browers[2]. The adoption of an XML schema instead of a Document Type Definition (DTD, [17]) is due to its flexibility and extensibility, in addition to the availability of predefined data types. Unlike a DTD, an XML schema is based on an open content model, and thus it can be applied to any kind of documents (irregular data), also containing XML elements defined by other schemas or even not defined anywhere. In this way, our proposal concerns a way of adding the timestamping facility to generic XML documents, also just containing plain HTML code. Any existing HTML-based Web site can thus easily be made temporal, by adding XML `<valid>` timestamps to its documents in accordance to the `ValidSchema.xml` schema above.

Therefore, as also exemplified by our demo prototype, the purpose of our proposed timestamping schema is (at least) twofold:

[2]This also applies to the date constants specified as `from`/`to` attribute values. However, the Ie5 parser and DOM do not currently support the `date` type and, thus, does not effect correctness checks on date values when applying the XML schema.

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
       xmlns:dt="urn:schemas-microsoft-com:datatypes">

   <AttributeType name="from" required="yes" dt:type="date" />
   <AttributeType name="to" required="yes" dt:type="date" />

   <ElementType name="validity">
      <attribute type="from" minOccurs="1" maxOccurs="1" />
      <attribute type="to" minOccurs="1" maxOccurs="1" />
   </ElementType>

   <ElementType name="valid" content="mixed">
      <element type="validity" minOccurs="1" maxOccurs="*" />
   </ElementType>

</Schema>
```

Figure 1: The `ValidSchema.xml` XML schema.

- it can be used to make temporal "traditional" Web sites (featuring HTML multimedia documents), in order to support the representation of historical information, and enabling a temporally selective navigation with respect to information validity; our first reference application corresponds to such an approach;
- it can be used to make temporal the emerging deployment of XML for data representation and exchange on the Web, in order to support the management of temporal structured or semistructured data, and enabling the utilization of functionalities as developed by temporal database research (e.g. TSQL2-like temporal query languages); our second reference application corresponds to such an approach.

Furthermore, any other kind of new XML-based application can be made temporal in the same way. Hence, they will benefit from the representation of historical information and temporal data management capabilities.

Referring to our first reference application, the legacy HTML pages composing the Web Museum must be converted to XML documents making reference to the `ValidSchema.xml` schema as described above. Moreover, their original HTML markup needs some checks for conversion into *well-formed* HTML code. This phase of Web site re-engineering can largely be automated. The human designer intervention is indeed required for the addition of timestamping, as the pieces of information to be enclosed in `<valid>` environments have to be carefully identified and appropriate time values have to be assigned in `<validity>` timestamps.

Referring to our second reference application, the employee data of Tab. I can be given the temporal XML representation shown in Fig. 2. The *ID* attribute, which is a time-invariant identifier, is transformed into a proper attribute of each `<employee>` element. For each employee, one `<valid>` environment is introduced for each of its tuples in Tab. I to represent a temporal version of the employee data. The

```
<?xml version="1.0" ?>
<emp xmlns:dt="x-schema:ValidSchema.xml">
  <employee ID="T1">
    <valid>
      <Name>Tom</Name>
      <D-Birth>8/15/51</D-Birth>
      <Department>Marketing</Department>
      <Salary>27K</Salary>
      <validity from="1996-04-15" to="1998-09-19" />
    </valid>
    <valid>
      <Name>Thomas</Name>
      <D-Birth>8/15/51</D-Birth>
      <Department>Accounting</Department>
      <Salary>28K</Salary>
      <validity from="1998-09-20" to="1999-05-31" />
      <validity from="2000-01-01" to="9999-99-99" />
    </valid>
  </employee>
  <employee ID="A1">
    <valid>
      <Name>Ashley</Name>
      <D-Birth>5/25/59</D-Birth>
      <Department>Engineering</Department>
      <Salary>30K</Salary>
      <validity from="1994-01-01" to="1995-12-31" />
      <validity from="1997-03-01" to="1999-11-30" />
    </valid>
    <valid>
      <Name>Ashley</Name>
      <D-Birth>5/25/59</D-Birth>
      <Department>Finance</Department>
      <Salary>35K</Salary>
      <validity from="1996-01-01" to="1997-02-28" />
    </valid>
  </employee>
</emp>
```

Figure 2: The `Employee.xml` XML data source.

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<!-- identity transformation template -->
<xsl:template>
  <xsl:copy>
    <xsl:apply-templates select="@*|*|comment()|pi()|text()" />
  </xsl:copy>
</xsl:template>

<!-- recursive valid-time selection template -->
<xsl:template match="valid">
  <xsl:choose>
    <xsl:when test="validity[condition on from and to attribute values]">
      <xsl:copy>
        <xsl:apply-templates select="@*|*|comment()|pi()|text()" />
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <xsl:apply-templates select="valid" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

</xsl:stylesheet>
```

Figure 3: The XSL `Valid.xsl` stylesheet.

`<validity>` timestamps correspond to the *Valid Time* attribute in Tab. I. The order in which the different `<validity>` tags composing a temporal-element timestamp appear (and their individual position within the `<valid>` environment) is irrelevant. Only conformance to the `ValidSchema.xml` schema is required.

3.2 Temporal Browsing, Navigation and Querying

The use of temporal documents is aimed at supporting selective browsing of Web documents, temporal Web navigation and querying of temporal data. The default valid time used for the Web navigation is usually the whole time range, enabling to view the full contents of documents, which corresponds to the basic use of non-temporal (standard) documents. For a selective temporal navigation, the time range can be reduced by the user (e.g. via some browser facility). It does not seem too restrictive to adopt a time *interval* as validity context, since also most queries in valid-time databases are commonly based on the comparison with an interval. Once set up by the user, such validity interval is known by the Web client as a *navigation validity context*. When temporal documents are processed, only the parts whose valid timestamp overlaps the navigation validity context are effectively taken into account and displayed. The same is automatically applied when new documents are retrieved by following a link, enabling a full-fledged temporal navigation. When dealing with XML temporal data, the navigation context assumes the meaning of a *query validity context*, which can be used by a Web-based application to select the data on the basis of a temporal comparison operator (not only the overlap one).

In our proposal, the valid-time selection relies on the adoption of an XSL stylesheet [26], named `Valid.xsl`, to perform a dynamic filtering of the document contents according to the navigation context. Such stylesheet embeds XSL transformations [46] and adopts the XML path language [44] facilities implemented in Ie5 (see [45] for reference). The definition of the stylesheet can be seen in Fig. 3: the first part consists of a simple identity-transformation template, whereas the second part is devoted to the temporal selection of the contents of valid contexts. The processing of the new XML `<valid>` element causes the output of the element contents when a validity selection condition (involving the document element timestamp) is verified. For instance, if the condition has the form:

```
@from[.$le$ '1999-12-31' ] and @to[.$ge$ '1999-01-01' ] ,
```

each `<valid>` element whose validity overlaps year 1999 is included in the stylesheet output: the selection condition matches any `<validity>` element where the `from` attribute value is $\leq$1999/12/31 and the `to` attribute value is $\geq$1999/1/1.

The particular structure of the selection template causes the execution of a test involving the navigation context and all the `<validity>` timestamps found in the current `<valid>` element. The conditional processing uses the `xsl:choose` instruction which provides for an `xsl:otherwise` case (not supported by the `xsl:if` XSL element), in order to recursively look for nested validity contexts. The `xsl:when` instruction is activated if at least one of the intervals (corresponding to a `validity` element) belonging to the timestamp satisfies the selection condition. The `xsl:otherwise` instruction is activated only when none of the timestamps of the current `<valid>` environment satisfies the selection condition.

Notice that, if the navigation validity context is changed by the user during his/her navigation, such a condition should dynamically be changed in the style sheet. If the stylesheet is then re-applied to the document, also the document visualization can be changed to reflect the user's action. For instance, in our prototype implementation, which is based on Ie5, the stylesheet is actually changed to update the selection condition and re-applied to the document by means of the DOM methods' functionalities. In this way, the change is actually effected on the stylesheet copy loaded (as an XML document object) in the main memory space managed by the browser or on a copy of the stylesheet cached on a local disk of the machine on which the browser is running. For example, the overlap with 1999 condition above can dynamically be applied to the displayed document by means of the JavaScript code shown in Fig. 4.

Notice also that more complex temporal selections than the simple overlap could be implemented by applying a different condition or even by defining a different filtering template in the stylesheet. Therefore, also sophisticated temporal query facilities could easily be added to a temporal Web site by means of appropriate direct management of the `Valid.xsl` temporal selection condition. As a matter of fact, as it is based on the Ie5 XSL support (with Microsoft XSLT [46] and XPath [44] extensions) our proposal represents a straightforward temporal extension of the XQL query language [18]; extension that has been designed as an application of XQL itself. For instance, we will show in the Section that follows how different XSL filters can be used to support TSQL2-like selection predicates over XML temporal data.

4 A Prototype Implementation

In this section we describe the software prototype we implemented for testing and validating our proposed valid-time XML/XSL temporal extensions. The prototype, named "The Valid Web" [5], consists of a Web site which can be browsed with Ie5 (see Fig. 6). The Web pages of the site are organized in two frames. A

```
// Find the validity test condition in the stylesheet
var sel = document.XSLDocument.selectSingleNode("//xsl:when/@test");

// Replace its value with the 1999 overlap condition
sel.value =
    "validity[ @from[.$le$’1999-12-31’] and @to[.$ge$’1999-01-01’] ]";

// Apply the modified stylesheet to the document, and update the display
document.body.innerHTML =
    document.XMLDocument.transformNode(document.XSLDocument)
```

Figure 4: A DOM script for dynamic change of the navigation validity context.

small service frame in the bottom part of the window contains all the required controls to deal with the user interactive specification of the validity context to be used for temporal navigation and querying, including the visualization of the current validity context. All the controls are implemented as JavaScript functions. A larger frame, occupying almost all the browser window space, is used to display temporal documents, that is the results of the temporally selective filtering effected by the `Valid.xsl` stylesheet on timestamped XML documents. The results of such filtering is, in our first example application, a plain HTML document which is then rendered by the browser in the “usual” way. In our second reference application, the results of the temporal filtering of the XML document containing employee data is then formatted via a second XSL stylesheet which converts them into an HTML tabular representation which is eventually dynamically included into the main page for visualization.

In general, the valid-time selection implies the choice of an interval. This can be done by an independent choice of the two time points representing the interval bounds. The selection of each interval bound can be based, for instance, on a graphic *scrollbar* or *slider* for analog fine selection of a time-point (at a given granularity level). In our prototype implementation, time-points are dates (i.e. the granularity level is the day) and the selection of an interval can be effected by means of a Java JFC/Swing applet [30, 8], which contains two graphic sliders: the former to select the year and the latter to select the day of the year (see Fig. 5). The former slider, for the user’s convenience, has a 500-year range, which can be changed (from 0–500 to 2000–2500) by means of a *multiple-choice menu* available next to the year slider. Assume we have to fix a date, say 1996/3/7. We can start by choosing the year 1996 with the former slider (with the default range 1500–2000 set) and then choose the March, 7 date with the latter slider. The chosen value can then be assigned to the From or To interval bound by means of the corresponding “Set” *button.* However, also editable input fields for direct typing of a valid time value (in the “`YYYY-MM-DD`” string format) are always available in the dialog window containing the running applet. The communication between the applet and the JavaScript control functions in the calling service frame (e.g. to return the selected validity context) is managed by means of the LiveConnect package [12] supported by the Java Plug-in 1.2.2 [29].

4.1 Visiting the Temporal Web Museum

In the Web Museum application, once the navigation validity context choice is confirmed by the user, the temporal selection over the currently displayed document is automatically re-executed by means of the DOM mechanism described in the previous Section. Furthermore, in order to enable a full-fledged temporal

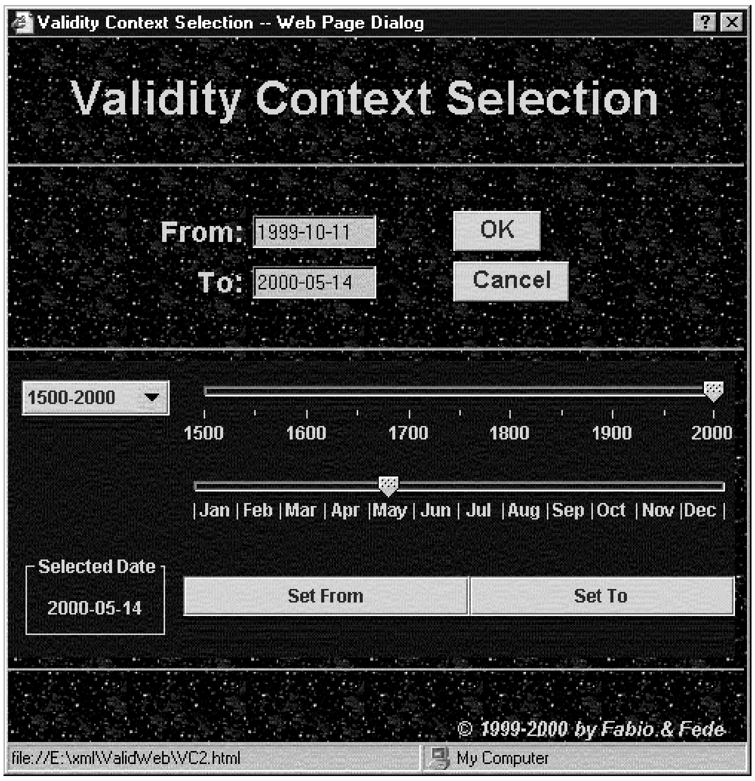
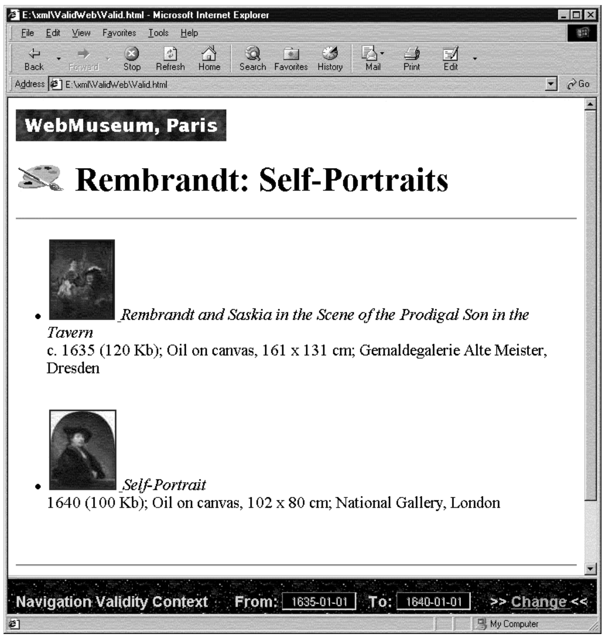

Validity Context Selection -- Web Page Dialog
Validity Context Selection
From: 1999-10-11
To: 2000-05-14
OK
Cancel
1500-2000
1500
1600
1700
1800
1900
2000
|Jan |Feb |Mar |Apr |May |Jun |Jul |Aug |Sep |Oct |Nov |Dec |
Selected Date
2000-05-14
Set From
Set To
© 1999-2000 by Fabio & Fede
file://E:\xml\ValidWeb\VC2.html
My Computer

Figure 5: The Java Applet for the selection of a Validity Context.

navigation, each time the user changes the displayed document in the usual way (e.g. by following a link), the current validity context is automatically "inherited" by the newly loaded page, if also the new document is a temporal XML one. This behaviour is forced in our prototype thanks to the dynamic HTML facilities supported by Ie5. In fact, we used a slightly modified `Valid.xsl` stylesheet with respect to Fig. 3. The actual stylesheet implements a dynamic callback mechanism by inserting some JavaScript code in the preamble of the processed document. Such a script provokes, on load of the document by the browser, the immediate activation of the temporal selection functionality: as when the validity context is changed, the `Valid.xsl` filter is updated on the fly (to include the overlap with the current validity context as selection condition) and then re-applied to the displayed document by means of DOM method calls similar to the ones in Fig. 4. In this way, the valid-time selection is automatically propagated, in a transparent way, also to the newly loaded temporal documents.

For example the top half of Fig. 6 shows two snapshots of the navigation of a sample page containing Rembrandt's self-portraits. The complete page contains seven pictures, dating from 1629 to 1669. Fig. 6.a shows the page when the validity context has been set to [1635–1640] and only two pictures are visible (the third and the fourth one). Fig. 6.b shows the same page with the validity context changed to [1660–1665] and two other pictures have been displayed (the fifth and the sixth one). The current navigation context is always visible in the bottom service frame of the window. The "Change" command on the right is a link that activates the applet of Fig. 5.

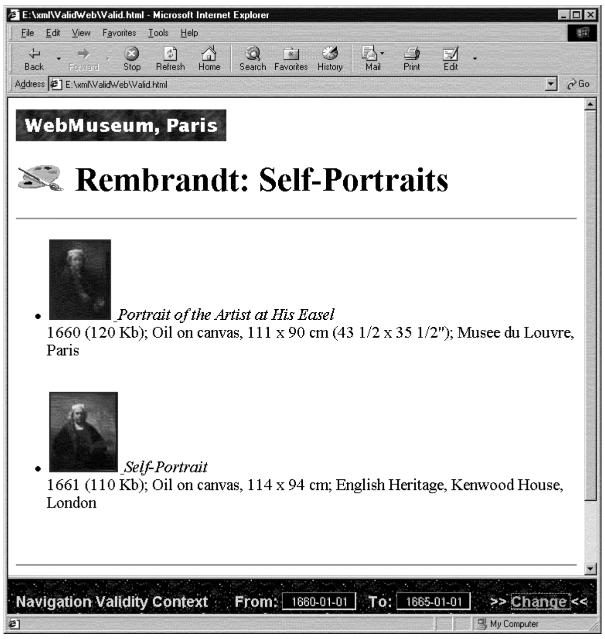

WebMuseum, Paris

Rembrandt: Self-Portraits

- *Rembrandt and Saskia in the Scene of the Prodigal Son in the Tavern*
  c. 1635 (120 Kb); Oil on canvas, 161 x 131 cm; Gemaldegalerie Alte Meister, Dresden
- *Self-Portrait*
  1640 (100 Kb); Oil on canvas, 102 x 80 cm; National Gallery, London

Navigation Validity Context From: 1635-01-01 To: 1640-01-01 >> Change <<

(a)

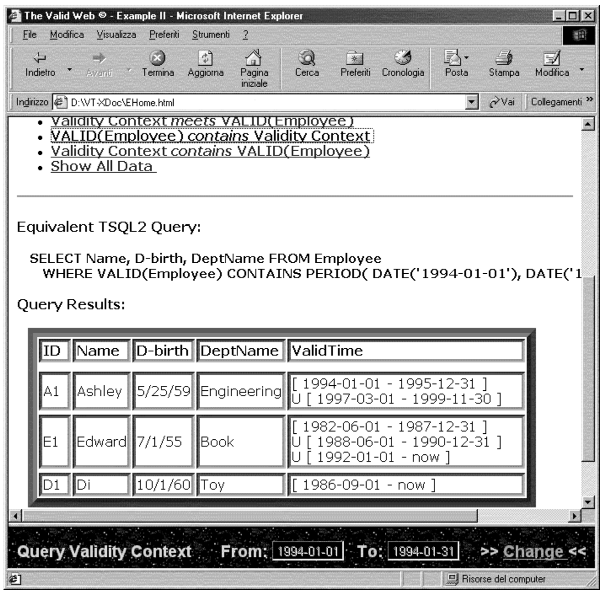

WebMuseum, Paris

Rembrandt: Self-Portraits

- *Portrait of the Artist at His Easel*
  1660 (120 Kb); Oil on canvas, 111 x 90 cm (43 1/2 x 35 1/2"); Musee du Louvre, Paris
- *Self-Portrait*
  1661 (110 Kb); Oil on canvas, 114 x 94 cm; English Heritage, Kenwood House, London

Navigation Validity Context From: 1660-01-01 To: 1665-01-01 >> Change <<

(b)


- Validity Context *meets* VALID(Employee)
- VALID(Employee) *contains* Validity Context
- Validity Context *contains* VALID(Employee)
- Show All Data

Equivalent TSQL2 Query:

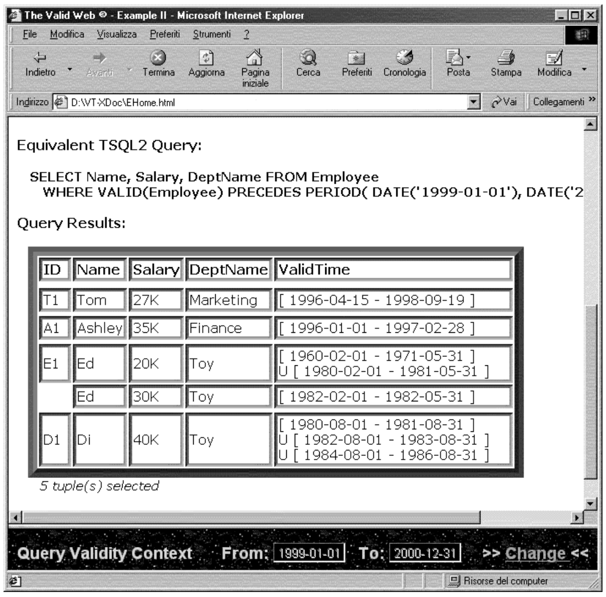
```
SELECT Name, D-birth, DeptName FROM Employee
  WHERE VALID(Employee) CONTAINS PERIOD( DATE('1994-01-01'), DATE('1
```

Query Results:

ID	Name	D-birth	DeptName	ValidTime
A1	Ashley	5/25/59	Engineering	[1994-01-01 - 1995-12-31] U [1997-03-01 - 1999-11-30]
E1	Edward	7/1/55	Book	[1982-06-01 - 1987-12-31] U [1988-06-01 - 1990-12-31] U [1992-01-01 - now]
D1	Di	10/1/60	Toy	[1986-09-01 - now]

Query Validity Context From: 1994-01-01 To: 1994-01-31 >> Change <<

(c)


Equivalent TSQL2 Query:

```
SELECT Name, Salary, DeptName FROM Employee
  WHERE VALID(Employee) PRECEDES PERIOD( DATE('1999-01-01'), DATE('2
```

Query Results:

ID	Name	Salary	DeptName	ValidTime
T1	Tom	27K	Marketing	[1996-04-15 - 1998-09-19]
A1	Ashley	35K	Finance	[1996-01-01 - 1997-02-28]
E1	Ed	20K	Toy	[1960-02-01 - 1971-05-31] U [1980-02-01 - 1981-05-31]
	Ed	30K	Toy	[1982-02-01 - 1982-05-31]
D1	Di	40K	Toy	[1980-08-01 - 1981-08-31] U [1982-08-01 - 1983-08-31] U [1984-08-01 - 1986-08-31]

5 tuple(s) selected

Query Validity Context From: 1999-01-01 To: 2000-12-31 >> Change <<

(d)

Figure 6: The Valid Web demo prototype: (a),(b) temporal navigation of the Web Museum; (c),(d) temporal querying of XML employee data.

TSQL2 predicate	XSL filter (“`xsl:when/@test`” node value in `Valid.xsl`)
VE OVERLAPS VC / VC OVERLAPS VE	`validity[ @from[.$le$` $VCto$ `] and @to[.$ge$` $VCfrom$ `] ]`
VE EQUALS VC / VC EQUALS VE	`validity[ index()=0 and end() and @from[.=` $VCfrom$ `] and @to[.=` $VCto$ `] ]`
VE PRECEDES VC	`validity[ end() and @to[.$lt$` $VCfrom$ `] ]`
VC PRECEDES VE	`validity[ index()=0 and @from[.$gt$` $VCto$ `] ]`
VE MEETS VC	`validity[ end() and @to[.=` $prevDate(VCfrom)$ `] ]`
VC MEETS VE	`validity[ index()=0 and @from[.=` $nextDate(VCto)$ `] ]`
VE CONTAINS VC	`validity[ @from[.$le$` $VCfrom$ `] and @to[.$ge$` $VCto$ `] ]`
VC CONTAINS VE	`validity[ index()=0 and @from[.$ge$` $VCfrom$ `] ]`
	`validity[ end() and @to[.$le$` $VCto$ `] ]`

Table II: XSL filters to support TSQL2 temporal selection predicates. VE represents the employee validity, where $VC = [VCfrom, VCto]$ represents the query Validity Context.

4.2 Temporal Querying the Employee Data

In the XML data management application, the validity context is used as argument of TSQL2-like temporal selection predicates for querying employee data. The main window of the application contains a set of links which allows the user to choose one of the possible predicates. Each link activates a JavaScript function which dynamically modifies the filtering `Valid.xsl` stylesheet and executes the query. Also the target list can be defined by the user, who can choose the desired attributes by means of *checkboxes* in a provided form. The form is then read by the application before query execution in order to define the projection specifications. The projection is implemented via a sort of parametric XSL stylesheet for translating the query results into HTML table format. Also this stylesheet is dynamically changed by the application through the DOM mechanism to enable the visualization of the only attributes present in the target list. The final results are eventually dynamically inserted into the page (in a `<PRE>` HTML element using the `innerHTML` property) and displayed as a table. The correspondence between the query temporal selection predicates and the applied XSL filters is shown in Tab. II. Notice that the `OVERLAPS` and `EQUALS` predicates are symmetric and, thus, the same filter can be used for them also when the arguments are exchanged.

In Ie5 proprietary XSL, “`end()`” is a boolean function that returns true when the current node is the last child of its father, whereas the “`index()`” function returns the current child number, being 0 the number of the first one. If we suppose that the non-overlapping intervals composing the timestamps are declared in ascending order (along the time axis) within `<validity>` elements, the XSL pattern “`validity[ index()=0 ]`” is thus matched when processing the *first* interval and “`validity[ end() ]`” does when processing the *last* interval. Hence, the TSQL2 predicates can easily be translated as reported in Tab. II. For instance, the equality predicate corresponds to an XSL filter which checks whether the `<validity>` interval under consideration is at the same time the first and the last one of the containing `<valid>` environment (i.e. the timestamp is also an interval), and whether its boundaries coincide with those of the query validity context. However, since we explicitly made no assumption on the order in which `<validity>` elements can be specified in XML documents, they must be sorted (e.g. in “`@from`” order) before the application of the `Valid.xsl` stylesheet in order to have temporal predicates correctly checked. In our prototype, this is done via another XSL stylesheet, which transforms the XML data source into a copy with sorted `<validity>` elements. Furthermore, the last predicate in the Table corresponds to two XSL filters which must be both applied to select the data: the first one ensures that the query interval starts before the beginning of the first validity interval of the selected data, whereas the second one ensures that the query interval ends after the end of the last validity interval (so that it completely contains the data validity). This is effected, in our prototype, by two successive applications of the `Valid.xsl` stylesheet.

For example, the bottom half of Fig. 6 shows two snapshots of the running prototype. Fig. 6.c and Fig. 6.d show the results of two queries on XML employee data. The TSQL2-like form of the queries is also (partially) visible in the captured images. The validity contexts used by the queries can be seen in the bottom frame of the windows.

As exemplified by the demo application, our proposed infrastructure can already be used, as it is, to support TSQL2-like temporal selection queries over *one* data source (i.e. one "relation" stored as an XML file). However, its query power is the same as the underlying XQL language supported by Ie5. Therefore, it is not relationally complete, as it does not support the join operation (i.e. queries over multiple XML sources, as XML-QL does [20]).

In addition to the described prototype, we also developed a test application where temporal joins [9] between temporal XML data sources can be executed, by explicitly implementing a nested-loops algorithm in JavaScript. The JavaScript "`do_Tjoin()`" function, whose listing can be seen in Appendix A.1, executes the following TSQL2 query[3]:

```
SELECT E.Name AS EmpName, D.Name AS DeptName, E.Salary, D.Budget
   [ VALID INTERSECT(E,D) ]
      FROM Employee AS E, Department AS D
         WHERE E.Department = D.Name
            [ AND VALID(E) OVERLAPS VALID(D) ]
```

where the Employee table is the same as in Tab. I and the Department table has relational schema **DEPARTMENT**(*Name*, *Budget* | *Valid Time*). The "`do_Tjoin()`" function accesses the two XML data sources (stored in separate files but which could also be contained in the same file) by means of DOM methods for manipulating XML document objects. DOM methods are also used by the function to explicitly build from scratch the temporal XML document containing the join results. The so built XML object, which also conforms to the `ValidSchema.xml` schema, can then be temporally filtered with the `Valid.xsl` stylesheet (but also the source documents can be filtered before the join execution) and eventually formatted with another XSL stylesheet to be displayed as a table in an HTML page.

Indeed, there is currently no way of performing a temporal join between two XML temporal data collections (even both stored in the same XML file) with pure XSL transformations. However, the JavaScript code we developed could be used as starting point for an XSL extension supporting temporal join processing. Obviously, since it is based on the manipulation of XML document objects, its performance heavily relies on their efficient management. Therefore, we could expect a breakdown in performances when source XML objects cannot be entirely maintained in the main memory space managed by the browser executing scripts and XSL-driven transformations, as it is the case of joining large temporal data sources. The introduction of temporal XML query optimization techniques [13] with efficient temporal join methods would be needed in order to overcome the performance problem.

5 Conclusions and Future Work

In this paper we outlined an approach for the integration of valid time into the Web. We addressed the issues concerning the extensions required to document structure and format, document processing and temporal browser usage. We basically sketched some feasible solutions and gave an idea of their potentialities by means of the application to a Web Museum and to the temporal management of XML data. The solution

[3]The parts in square brackets can be omitted, due to the default TSQL2 semantics. They have been reported here for clarity.

we proposed relies on simple XML-based extensions: the introduction of a new `<valid>` markup tag with the definition of an XML schema for the creation of temporal documents, and the use of an XSL stylesheet for selective filtering of temporal documents according to a user-defined navigation validity context. We also developed a prototype temporal Web site, accessible with Ie5, which implements the proposed solution and provides tools for the management of the validity context with a friendly user interface. However, the functionalities for the validity context management could directly be embedded in future XML-compliant browsers, without the need for a special frame provided for the purpose. Nevertheless, the use of valid-time temporal documents is also perfectly compatible with the nowadays XML Ie5 technology, as our prototype Web site demonstrates.

Moreover, the proposed solutions are also completely compatible with temporally unaware Web clients, in the sense that all the information contained in the temporal HTML documents would still be accessible (being the new temporal markup tags simply ignored), although temporal navigation and selective visualization facilities would clearly not be available. Legacy HTML-based Web sites can easily be made temporal by adding the required timestamps to the historical multimedia information they possibly contain.

Future work will be devoted to the extension of the proposed infrastructure. In particular we would like to allow the selection of any kind of validity context, relaxing the constraint of an interval to be used. For example, the valid-time context could also be set by capturing the valid time connected with a selected object within the document under visualization. Notice that, in general, the captured validity context is a temporal element rather than a single interval. Such mechanism may be used to extend, contract and/or shift the current navigation validity context to better suit virtual time-travel desires by means of an adaptive self-tuning process. When querying XML data, this would mean to extend the class of supported TSQL2-like queries, by allowing both arguments of selection predicates to be temporal elements, possibly extracted from the underlying data (like in a generic temporal θ-join).

Moreover, a complete query language for XML temporal documents could be introduced to support complex temporal queries over the Web. To this purpose, in our future work, we will consider the extension of an XML query language like XML-QL [43] (or the XQL developed by Fujitsu [6]) to support general temporal selection conditions over multiple timestamped XML documents. The required temporal extensions will be based on all the features of the TSQL2 temporal query language and, thus, will have the same temporal expressiveness.

Acknowledgments

This work has been in part supported by the M.U.R.S.T. co-funded "INTERDATA" project (subproject: Design methodologies for Web sites).

References

[1] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol. 26, No. 11, 1983.

[2] Archives & Museums Informatics, URL: `http://www.archimuse.com`.

[3] T. Berners-Lee, R. Cailliau, A. Lautonen, H.F. Nielsen, A. Secret, "The World Wide Web," *Communications of the ACM*, Vol. 37, No. 8, 1994.

[4] Date and Time Formats, W3C Consortium Note,
URL: `http://www.w3.org/TR/NOTE-datetime`.

[5] F. Grandi, F. Mandreoli, “The Valid Web ©”, *Proc. of Software Demo Track at the EDBT’2000 Intl. Conference*, Konstanz Lake, Germany (*to be held*).

[6] H. Ishikawa, K. Kubota, Y. Kanemasa, Y. Noguchi, “The Design of a Query Language for XML Data,” *Proc. DEXA Workshop on Query Processing in Multimedia Information Systems (QPMIDS)*, Florence, Italy, 1999.

[7] R. Hjelsvold, R. Midtstraum, O. Sandstå, “A Temporal Foundation of Video Databases,” *Proc. Intl. Workshop on Temporal Databases*, Zurich, 1995.

[8] Java Foundation Classes (JFC), Sun Microsystems,
URL: `http://java.sun.com/products/jfc/`.

[9] C.S. Jensen, J. Clifford, R. Elmasri, S.K. Gadia, P. Hayes, S. Jajodia (eds.) et al., C. Dyreson, F. Grandi, W. Käfer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J.F. Roddick, N.L. Sarda, M.R. Scalas, A. Segev, R.T. Snodgrass, M.D. Soo, A. Tansel, P. Tiberio, G. Wiederhold, “A Consensus Glossary of Temporal Database Concepts - February 1998 Version,” in O. Etzion, S. Jajodia and S. Sripada (eds.), *Temporal Databases - Research and practice*, LNCS N. 1399, Springer-Verlag, 1998.

[10] C.S. Jensen, M.D. Soo, R.T. Snodgrass, “Unifying Temporal Data Models via a Conceptual Model,” *Information Systems*, Vol. 19, N. 7, 1994.

[11] N. Kline, “An Update of the Temporal Database Bibliography,” *ACM SIGMOD Record*, Vol. 22, N. 4, 1993.

[12] LiveConnect, in *JavaScript Guide*, Ch. 5, Netscape Communications,
URL: `http://developer.netscape.com/docs/manuals/communicator/jsguide4-/livecon.htm`.

[13] J. McHugh, J. Widom, “Query Optimization for XML”, Proc. 25th VLDB Conference, Edinburgh, Scotland, 1999.

[14] S.R. Newcombe, N.A. Kipp, V.T. Newcombe, “The “HyTime” Hypermedia/Time-based Document Structuring Language,” *Communications of the ACM*, Vol. 34, No. 11, 1991.

[15] N. Pioch, “The Web Museum,” URL: `http://www.cnam.fr/wm/`.

[16] *Proc. of QL’98 - The W3C Query Languages Workshop*, Boston, MA, Dec. 1998,
URL: `http://www.w3.org/TandS/QL/QL98/`.

[17] Prolog and Document Type Declaration, in *Extensible Markup Language (XML) 1.0*, W3C Consortium Recommendation,
URL: `http://www.w3.org/TR/REC-xml#sec-prolog-dtd`.

[18] J. Robie, J. Lapp, D. Schach, “XML Query Language (XQL),” in [16],
URL: `http://www.w3.org/TandS/QL/QL98/pp/xql.html`.

[19] R.T. Snodgrass (ed.), I. Ahn, G. Ariav, D. Batory, J. Clifford, C.E. Dyreson, R. Elmasri, F. Grandi, C.S. Jensen, W. Käfer, N. Kline, K. Kulkarni, T.Y. Cliff Leung, N. Lorentzos, R. Ramakrishnan, J.F. Roddick, A. Segev, M.D. Soo, S.M. Sripada, *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers, Boston, Massachussets, 1995.

[20] D. Schach, J. Lapp, J. Robie, “Querying and Transforming XML,” in [16],
URL: `http://www.w3.org/TandS/QL/QL98/pp/query-transform.html`.

[21] I. Sommerville, T. Rodden, P. Raysin, A Kirby, A. Dix, “Supporting Information Evolution on the WWW,” *World Wide Web*, Vol. 1, No. 1, 1998.

[22] M.D. Soo, “Bibliography on Temporal Databases,” *ACM SIGMOD Record*, Vol. 20, N. 1, 1991.

[23] A. Tansel, J. Clifford, V. Gadia, S. Jajodia, A. Segev, R.T. Snodgrass (eds.), *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings Publishing Company, Redwood City, California, 1993.

[24] The Document Object Model (DOM) Home Page, W3C Consortium,
URL: `http://www.w3.org/DOM/`.

[25] The Extensible Markup Language (XML) Resource Page, W3C Consortium,
URL: `http://www.w3.org/XML/`.

[26] The Extensible Stylesheet Language (XSL) Resource Page, W3C Consortium,
URL: `http://www.w3.org/Style/XSL/`.

[27] The HyperText Markup Language (HTML) Home Page, W3C Consortium,
URL: `http://www.w3.org/MarkUp/`.

[28] The HyperText Transfer Protocol (HTTP) Home Page, W3C Consortium,
URL: `http://www.w3.org/Protocols/`.

[29] The Java Plug-in Home Page, Sun Microsystems,
URL: `http://java.sun.com/products/plugin/`

[30] The Java Technology Resource Page, Sun Microsystems,
URL: `http://java.sun.com`.

[31] The JavaScript Resource Page, Netscape Communications,
URL: `http://developer.netscape.com/tech/javascript/`.

[32] The Microsoft Internet Explorer Home Page, Microsoft,
URL: `http://microsoft.com/windows/Ie/`.

[33] The Naming and Addressing (URIs, URLs,...) Resource Page, W3C Consortium,
URL: `http://www.w3.org/Addressing/`.

[34] The NU.M.E. Project Home Page, CINECA,
URL: `http://www.cineca.it/visit/NUME/`.

[35] The Standard Generalized Markup Language (SGML) Resource Page, W3C Consortium,
URL: `http://www.w3.org/MarkUp/SGML/`.

[36] The Synchronized Multimedia Integration Language (SMIL) 1.0 Specification, W3C Consortium Recommendation,
URL: `http://www.w3.org/TR/REC-smil/`.

[37] The Virtual Reality Markup Language (VRML) Resource Page, Web3D Consortium,
URL: `http://www.vrml.org/`.

[38] The Web-based Distributed Authoring and Versioning (WebDAV) Resource Page,
URL: `hhtp://www.webdav.org/`.

[39] The World Wide Web Consortium (W3C) Home Page, URL: `http://www.w3.org/`.

[40] V.J. Tsostras, A. Kumar, “Temporal Database Bibliography Update,” *ACM SIGMOD Record*, Vol. 25, N. 1, 1996.

[41] F. Vitali, D.G. Durand, “Using Versioning to Support Collaboration on the WWW,” *World Wide Web Journal*, Vol. 1, No. 1, Winter 1996.

[42] XML-Data, W3C Consortium Note,
URL: `http://www.w3.org/TR/1998/NOTE-XML-data/`.

[43] XML-QL: A Query Language for XML, W3C Consortium Note,
URL: `http://www.w3.org/TR/NOTE-xml-ql/`.

[44] XML Path Language (XPath) Version 1.0, W3C Consortium Working Draft,
URL: `http://www.w3.org/TR/xpath`.

[45] XSL Developer’s Guide, The Microsoft Developer Network,
URL: `http://msdn.microsoft.com/xml/xslguide/`.

[46] XSL Transformations (XSLT) Version 1.0, W3C Consortium Working Draft,
URL: `http://www.w3.org/TR/xslt`.

A.1 JavaScript Code to Perform a Temporal Join

```
var saveint = new Array(); var numint = 0;

function copyNode( father, source, name ) {
   father.appendChild( source.selectSingleNode( name ).cloneNode( true ) );
}

// creates a new "<name>value</name>" node
function newTNode( docobj, father, name, value ) {
   father.appendChild( docobj.createElement( name ) );
   father.lastChild.appendChild( docobj.createTextNode( value ) );
}

// creates a new <validity> timestamp
function newVNode( docobj, father, interval ) {
   father.appendChild( docobj.createElement( "validity" ) );
   father.lastChild.setAttribute( "from", interval.from );
   father.lastChild.setAttribute( "to", interval.to );
}

// tests for the overlap between two temporal elements
// and stores the resulting intersection (if any) in "saveint";
// "numint" counts the number of intervals in the intersection
function overlap( val1, val2 ) {
   numint = 0;
   // nested loops over validity intervals
   for ( v1 = val1.nextNode(); v1 != null; v1 = val1.nextNode() )
   { for ( v2 = val2.nextNode(); v2 != null; v2 = val2.nextNode() )
      {  from1 = v1.selectSingleNode("@from").text;
         to1 = v1.selectSingleNode("@to").text;
         from2 = v2.selectSingleNode("@from").text;
         to2 = v2.selectSingleNode("@to").text;
         if (( from1 <= to2 ) && ( from2 <= to1 ) ) // overlap test
         {  var interval = new Object(); // compute the intersection
            interval.from = from1 > from2 ?  from1 :  from2 ;
            interval.to = to1 < to2 ?  to1 :  to2 ;
            saveint[numint++] = interval;
         }
      }
      val2.reset();
   }
   return ( numint > 0 );
}
```

```
function do_Tjoin() {
  var results = new ActiveXObject("Microsoft.XMLDOM");
  results.async = false;
  results.loadXML('<?xml version="1.0" ?>
                 <join xmlns:schema="ValidSchema.xml" />');
  var root = results.documentElement;
  var emp = empl.documentElement.selectNodes("//emp/employee/valid");
  var dep = dept.documentElement.selectNodes("//dept/department/valid");

  // nested loops
  var tup = 0;
  for ( var e = emp.nextNode(); e != null; e = emp.nextNode() )
  {  for ( var d = dep.nextNode(); d != null; d = dep.nextNode() )
    {  var en = e.selectSingleNode("Name");
      var ed = e.selectSingleNode("Department");
      var ve = e.selectNodes("validity");
      var dn = d.selectSingleNode("Name");
      var vd = d.selectNodes("validity");

      // test the join condition
      if ( ( ed.text == dn.text ) && overlap( ve, vd ) )
      {  root.appendChild( join.createElement("valid") );
        newTNode( join, root.childNodes.item(tup), "EmpName", en.text );
        newTNode( join, root.childNodes.item(tup), "DeptName", dn.text );
        copyNode( root.childNodes.item(tup), e, "Salary" );
        copyNode( root.childNodes.item(tup), d, "Budget" );
        for ( i=0; i<numint; i++ )
             newVNode( join, root.childNodes.item(tup), saveint[i] );
        tup++;
      }
    }
    dep.reset();
  }
  return results;
}

// load the data sources and execute the join
var empl = new ActiveXObject("Microsoft.XMLDOM"); empl.async = false;
var dept = new ActiveXObject("Microsoft.XMLDOM"); dept.async = false;
var join = new ActiveXObject("Microsoft.XMLDOM"); join.async = false;
emp.load("Employee.xml"); dep.load("Department.xml");
join = do_Tjoin();
```