A Temporal Data Model and Management System for Normative Texts in XML Format*

Fabio Grandi
DEIS, Alma Mater Studiorum - Univ. di Bologna
viale Risorgimento, 2
Bologna, Italy
fgrandi@deis.unibo.it

Federica Mandreoli
DII, Univ. di Modena e Reggio Emilia
via Vignolese, 905/b
Modena, Italy
mandreoli.federica@unimo.it

Paolo Tiberio
DII, Univ. di Modena e Reggio Emilia
via Vignolese, 905/b
Modena, Italy
tiberio.paolo@unimo.it

Marco Bergonzini
DII, Univ. di Modena e Reggio Emilia
via Vignolese, 905/b
Modena, Italy
bergonzini.marco@unimo.it

ABSTRACT

In this paper, we present the results of an on-going research activity concerning the temporal management of normative texts in XML format. In particular, four temporal dimensions (publication, validity, efficacy and transaction times) are used to correctly represent the evolution of norms in time and their resulting versioning. Hence, we introduce a multiversion data model based on XML schema and define basic mechanisms for the management of norm texts. Finally, we describe a prototype management system which has been implemented and evaluated.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Textual databases; H.3.5 [**Online Information Services**]: Web-based services; J.1 [**Administrative Data processing**]: Law

General Terms

Management, Design, Algorithms

Keywords

Temporal XML, Legal information systems

*This work has been partially supported by the MIUR-40% Project: "La dinamica della norma nel tempo: aspetti giuridici ed informatici".

WIDM'03, November 7–8, 2003, New Orleans, Louisiana, USA.

1. INTRODUCTION

Time is one of the main aspects characterizing several real world facets and phenomena. The ability to model the temporal dimension of the real world and to respond within time constraints to changes in the real world as well as to application-dependent operations is essential to many computer applications. The management of norms represents one of such applications as temporal concerns are ubiquitous in the law domain [14]. Time in normative systems has become a central topic of the cultural and political debate and is of fundamental concern to legal informatics. The law is under increasing pressure to keep pace with social change: normative texts and amendments follow one another in time and get overlapped. Moreover, the route to e-Government pushes government agencies for providing access to services for publication and exchange of norms on the Web.

In the context of database research, the management of time has been extensively studied in the last decades [17, 6]. In particular, many efforts have been devoted to add time support to database models and system functionalities. Temporal database systems provide special facilities for storing, querying, and updating historical and/or future data. In this context, two time dimensions are usually considered: valid time and transaction time [8]. Valid time is the time of the real world. It denotes the time a fact is true in reality. Transaction time is the time of the system and it denotes the time during which the fact is present in the database as stored data. In order to make a more complete picture, two other temporal dimensions have been considered useful for advanced applications: event time [9] (also called decision time in [11]), which is the occurrence time of events that initiate and/or terminate the validity of some fact in the real world, and availability time [4], which is the time some fact is available in the information system.

Moreover, in the database research community, there is a much current interest in representing and querying semi-structured data. For example, database-resident data can be published as static or dynamic XML documents, which can then be viewed on Web browsers and processed by var-

ious Web based applications, including queries written in languages such as XPath [22] and XQuery [18]. As a consequence, several works took into account change, versioning, evolution and also explicitly temporal aspects, in semistructured and XML-based data management, often applying conceptual tools and techniques developed by temporal database research. The main focus of some approaches was on the representation and management of changes, where different versions of data are produced by updates. In this approach, temporal attributes are often used to timestamp stored versions (e.g. [2, 1]). They represent the time the updates were applied and, thus, have the (implicit) semantics of transaction time with respect to the system where the changes are effected. Other approaches considered the classical notion of valid time (e.g. [7, 19]). For example, the "Valid Web" approach [7] is an XML/XSL infrastructure designed to represent and manage temporal Web documents (i.e. documents containing historical information, with timestamps explicitly encoded by the document authors to assign validity to information contents). Temporal documents can then be selectively browsed, in accordance with a user-supplied temporal period of interest. Some approaches also considered a bitemporal data model, that is supporting both valid and transaction time (e.g. [5, 10]). However, such approaches are not straightforwardly applicable to the legal application domain because of the specificity of the data semantics and operation requirements. In the context of legal computer science, previous approaches already dealt with the reconstruction of consolidated norm texts, consisting of their current temporal version [15]. One temporal dimension was considered in such approaches.

In this paper, we present the results of the research activity we carried out in the context of the multi-disciplinary project "The dynamics of norms in time: legal and informatics aspects" co-funded by the Italian Ministry of University. Such a project emphasizes, from a legal point of view, the necessity for a rigorous, effective and efficient management of time-varying norm texts. In this context, the main objective of our work has been the development of a computer system for the management of norms in time represented as XML documents and made available on the Web. To this end, we developed a temporal XML data model which uses four time dimensions to correctly represent the evolution of norms in time and their resulting versioning. The model, which will be presented in Section 3, is equipped with basic operators for the modification and retrieval of multiversion norm texts. For the efficient exploitation of the data model, a system prototype has been implemented and evaluated as described in Section 4.

Out of this context, our work covers a broader interest as we developed a temporal and text-centric application system with IR capabilities, which gave us the opportunity of studying the interaction of temporal aspects having a well founded semantics with the structural properties of XML documents. Hence, our approach can provide useful solutions also for other web-based advanced applications with similar requirements (e.g. management of clinical data).

2. THE TEMPORAL DATA MODEL

In this section, we present the XML-based temporal data model we propose for the representation and management of versioned normative texts. The model supports multiple time dimensions, all involved in the law application lifecycle.

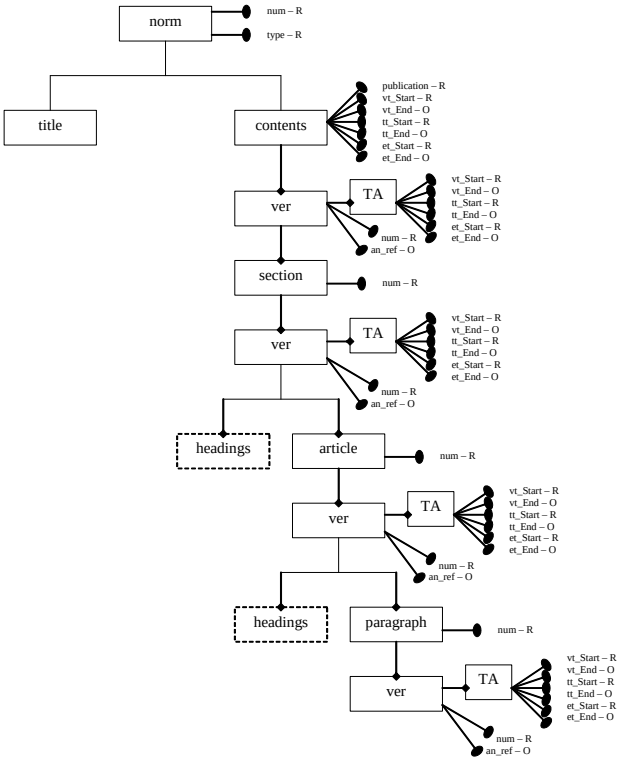


Figure 1: The XML-schema for the representation of norms in time

2.1 Representation of Time and Norms

The model encodes the hierarchical organization of normative texts (i.e. as in many other countries, also in Italy norms are based on a contents-section-article-paragraph hierarchy) in a tree-like inner structure by means of an XML schema [21]. Such an encoding is enriched with timestamping metadata modelling the temporal aspects of normative texts. The time dimensions we consider are the following:

Publication time. It is the time of publication of the norm on the Official Journal. It has the same semantics as event time (and availability time, as the two time dimensions, in such a context, coincide). It is a global and unchangeable property for the whole norm contents and, thus, it will be treated separately.

Validity time. It is the time (some part of) the norm is in force (in the Italian regulations, usually a norm is in force from the publication date plus 15 days on, until its validity is changed by a subsequent act). It has the semantics of valid time, as it represents the time the norm actually belongs to the regulations in the real world.

Efficacy time. It is the time (some part of) the norm can be applied to a concrete case. It usually corresponds to the validity of norms, but it can be the case that an abrogated norm continues to be applicable to a limited number of cases. Until such cases cease to exist, the norm continues its efficacy.

Transaction time. It is the time (some part of) the norm is stored in our computer system. Obviously, it has the same semantics of transaction time as in temporal databases.

Disregarding publication time, all the three dimensions above are independent and thus can be treated in an "orthogonal" way. The temporal model supports lossless updates at any level of the hierarchy by means of *temporal versions* representing the results of the changes normative texts undergo. The *temporal pertinence* of a version is a subset of the tridimensional space validity × efficacy × transaction and can be represented by a temporal element [8], that is the disjoint union of tridimensional time intervals, each obtained as the Cartesian product of one time interval for each of the supported temporal dimensions. The adoption of temporal elements avoids the duplication of version contents in the presence of a temporal pertinence with a complex shape. The fourth time dimension, publication time, is a global property of the document which cannot be changed after publication and, thus, is not included in the timestamping mechanism.

The full version of our XML schema is depicted in Figure 1 ("R" and "O" near attribute names stand for "required" and "optional", respectively, and one-to-many relationships are denoted with arrows ending with a diamond). It comes out as a quite unfaithful translation and extension of one of the DTDs published by the "Norma in Rete" (Norm on Network) [12] working group. As norms are identified by a (type, number) pair (e.g. type="Law", number="27/2003") the meta-level of normative texts is rooted by a `norm` top element, characterized by `type` and `num`(ber) attributes, which includes the `title` and `contents` elements. The `contents` element has attributes defining global temporal properties: an attribute `publication` storing the publication date of the norm and also temporal attributes which define a tridimensional bounding box for all the timestamps the document contains and which is used as a summary temporal pertinence of the whole norm for faster query processing. Then, at each level of the contents-section-article-paragraph hierarchy it is possible to represent one or more temporal versions by means of the `ver` elements, where attribute `num` represents the version number and `an_ref` is the reference to the so-called *active norm*, that is the norm whose enforcement caused the versioning. Each temporal version is characterized by a temporal element-type timestamp, which is defined by union of `TA` XML elements, each of which represent a tridimensional time interval whose boundaries are encoded as `TA` attributes: validity (`vt_start` and `vt_end`), efficacy (`et_start` and `et_end`), and transaction time (`tt_start` and `tt_end`). The "end" attribute associated to each dimension (e.g. `vt_end`) is optional. Its absence denotes a positive infinite time interval which, in the literature [8], is usually represented as $[vt, forever)$ in case of valid time intervals and $[tt, UC)$ in case of transaction time intervals, where vt and tt are valid and transaction time values, respectively, and UC means "Until Changed". Timestamps can thus occur at any level of the XML norm hierarchy and obey to an ancestor-descendant inheritance semantics. In accordance with the hierarchical structure of normative texts complying with our XML schema, the timestamps of any node are inherited by its descendants, unless redefined. Redefinitions are local definitions of timestamps which can only restrict the inherited values. In other words, the timestamps owned by each version must be contained in the timestamps of its parent. The temporal pertinence of each version is thus the temporal element given by the disjoint union of the inherited tridimensional time intervals, some of which can be locally redefined. Finally, the temporal pertinence of sibling versions must be disjoint, that is, at each level, at most one version is associated to each tridimensional time point (or *chronon* [8]).

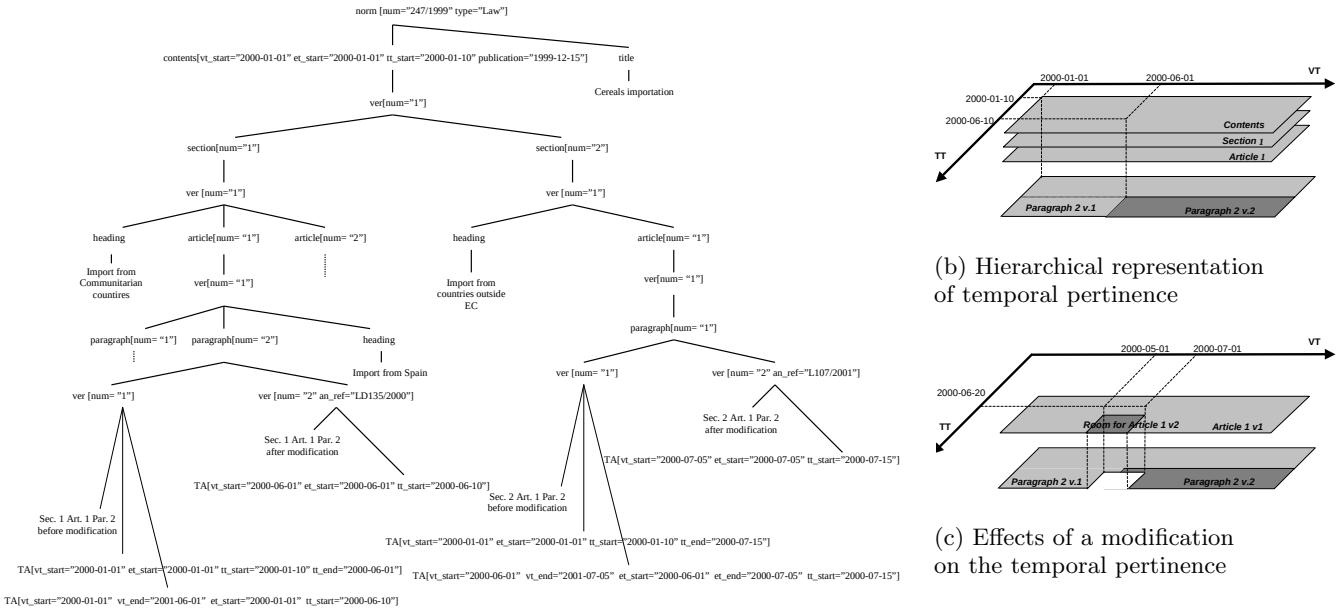
EXAMPLE 1. *Figure 2.a shows the tree-like structure of a document conforming to the temporal XML schema. The Law 247/1999 concerns the cereal importation and contains two sections, three articles and four paragraphs. It has been published on 1999/12/15 and is valid from 2000/1/1 (it has been recorded in the system on 2000/1/10). Only (two) paragraphs underwent punctual modifications and thus have more than one version. For this reason, all parts but paragraphs inherit the timestamps from the* `contents` *tag. For the paragraphs indeed it is necessary to explicit the temporal attributes since they are redefined by the corresponding versions. Paragraph 2 of Sec. 1, Art. 1 has been modified by the "LD135/2000" Legislative Decree, in force since 2000/6/1 (modification recorded on 2000/6/10). Paragraph 1 of Sec. 2, Art. 1 has been modified by the "L107/2001" Law, in force since 2001/7/5 (modification recorded on 2001/7/15).*

Notice that, in the former case the old version continues to be applicable (e.g. to the cases for which it was applicable before the modification), whereas in the latter case the modifying Law has stated that the old version is definitely no longer applicable (hence, efficacy time has been stopped to 2001/7/5 like validity). In both cases, the temporal elements correctly map the temporal pertinence of the text before the modification on a tridimensional space (validity × efficacy × transaction). In particular, Fig. 2.b shows the projection on the bidimensional space validity × transaction of the containment relationship (due to inheritance) between the temporal pertinence of the versions in the path to the second paragraph. In the figure, the different levels of the document hierarchy are represented on the z-axis, v. is used in place of version (e.g. Paragraph 2 v. 1 corresponds to the version number 1 of the Paragraph number 2) and the dotted line on the second version of the second paragraph shows how the temporal pertinence has been represented as union of disjoint intervals, each of which corresponds to a `TA` *element.* □

2.2 Management Aspects

The management of the dynamics of norms is quite complex. Normative texts often undergo modifications. Each modification is enforced by a norm, the active norm, containing a reference to the portion of the norm to be modified, the passive norm. Modifications usually concern the contents or the time frames of validity/applicability. The former case, known as textual modification, concerns either the deletion of (a part of) the passive norm (abrogation), or the introduction of a new part of the passive norm, or the substitution of (a part of) the passive norm. The latter case, known as temporal modification, includes either the time extension or the suspension of the passive norm.

The model is equipped with two basic operators for the management of norm modifications: one is devoted to the explicit textual modification and the other allows temporal pertinence modifications. The two operators act on norms



(a) The tree-like structure of a sample document

(b) Hierarchical representation of temporal pertinence

(c) Effects of a modification on the temporal pertinence

Figure 2: An example of multiversion XML document

complying with the XML schema of Fig. 1 and, thus, having a tree representation similar to the example in Fig. 2. In both cases, updates are performed in a lossless fashion by means of an accurate and correct management of the versions representing the history of the document contents and of their timestamps.

The former operator, named *changeText*, has been included in the temporal XML model in order to support textual modifications occurring at any level of the tree-like inner structure of documents. *changeText* essentially adds a temporal version to the structural element of the normative text to be modified and, in order to comply with the inheritance constraints, it also accommodates the temporal pertinence of the sibling, ancestor and descendant versions. The efficacy and validity time pertinence of the newly added version is specified by means of parameters according to the provisions of the active norm, whereas the transaction time semantics forces the interval $[now, UC)$ (now is the current chronon [8]) to be assigned as the transaction time pertinence. The algorithm implementing *changeText* is shown in Fig. 3 where paths in the tree are specified *à la* XPath [22]. It makes use of the following node operations: `parent`: $Node \rightarrow Node$, `children`: $Node \rightarrow \{Node\}$, `addChild`: $Node \times Tree \rightarrow Tree$. `parent`$(q)$ and `children`(q) return the parent node and the set of children nodes of q, respectively, whereas `addChild`(q, t) returns the tree obtained by adding the tree t as child of q. *changeText* requires the path which starting form the root allows the identification of the portion of the law undergoing textual modifications (e.g. `/norm[@num=1224, @type='Law']/section[@num=1]` identifies the first section of the Law number 1224). The validity and efficacy temporal element to be assigned to the new version is specified by the parameter $\bigcup_{i=1,\ldots,n}[vt_{s_i}, vt_{e_i}] \times [et_{s_i}, et_{e_i}]$, whereas the new XML text is given in *txt*, the number of the version in *vnum* and the reference to the active norm in *an*. Starting from the root of the passive norm, the algorithm goes along the tree down to the node rooting the portion undergoing modification and it losslessly updates such a node by adding the new version (step 4) with temporal pertinence $\bigcup_{i=1,\ldots,n}[vt_{s_i}, vt_{e_i}] \times [et_{s_i}, et_{e_i}] \times [now, UC)$. Moreover, in order to comply with the inheritance semantics, the algorithm updates the temporal pertinence of the versions which are siblings (steps 1-2) and ancestors (step 3) of the newly added one, if necessary. More precisely, as the temporal pertinence of sibling versions must be disjoint, for each version which is sibling of the newly added one, the function `restrictTimestamps` determines the temporal pertinence of it (steps 2-4) and if the temporal pertinence overlaps that of the newly added one (step 5), the function updates it (steps 6-7) and recursively applies itself to their first descending versions (step 9). The algorithm thus avoids to access the descendants of such versions for which the pertinence does not need updates. Even in the worst case, descendants are visited only once as the function works in a depth-first manner. In order to update the temporal pertinence of any version, it is necessary to exclude the overlapping portion by splitting each tridimensional temporal interval into the union of two or more non-overlapping tridimensional temporal intervals covering the remaining temporal pertinence (e.g. see Fig. 2.c where the introduction of the new version - the version number 2 - causes the restriction of the temporal pertinence of the version number 1 and causes "propagation" with timestamp restiction to the underlying level(s)). To this end, we devised an algorithm which generalizes to three dimensions the covering methods used in [3]: it first splits the temporal interval of each dimension into union of intervals excluding the overlapping part and then products them to obtain tridimensional temporal intervals. The algorithm can also require a coalescing transformation [8] where overlapping or adja-

```
Algorithm changeText($p, text, vnum, an, ⋃_{i=1,...,n}[vt_{s_i}, vt_{e_i}] × [et_{s_i}, et_{e_i}])
(1) for each node $c in $p/ver
(2)    restrictTimestamps(⋃_{i=1,...,n}[vt_{s_i}, vt_{e_i}] × [et_{s_i}, et_{e_i}] × [now, UC], $c);
(3) extendTimestamps( ⋃_{i=1,...,n}[vt_{s_i}, vt_{e_i}] × [et_{s_i}, et_{e_i}] × [now, UC], parent($p));
(4) addChild($p,<ver num=vnum an_rif=an>
                    <ta vt_start=vt_{s_1} vt_end=vt_{e_1} et_start = et_{s_1} et_end = et_{e_1} tt_start=now> </ta>...
                    <ta vt_start=vt_{s_n} vt_end=vt_{e_n} et_start = et_{s_n} et_end = et_{e_n} tt_start=now> </ta>
                text </ver>);

function restrictTimestamps(tempElem, $m)
(1) recursion=false;
(2) te = ∅;
(3) for each $n in $m/ta
(4)    te = te ∪ [$n/@vt_start,$n/@vt_end] × [$n/@et_start,$n/@et_end] × [$n/@tt_start,$n/@tt_end]
(5) if (te ∩ tempElem ≠ ∅)
(6)   te = coalesce(te \ tempElem);
(7)   set timestamps of m to te;
(8)   recursion= true;
(9) if(recursion)
      for each $d in children(children($m))
         restrictTimestamps(tempElem,$d);

function extendTimestamps(tempElem, $m)
(1) te = ∅;
(2) for each $n in $m/ta
(3)    te = te ∪ [$n/@vt_start,$n/@vt_end] × [$n/@et_start,$n/@et_end] × [$n/@tt_start,$n/@tt_end]
(4) if not (tempElem ⊆ te)
(5)   changeTime(parent($m), (vt, et, tt) ∈ te, te \ tempElem);
```

Figure 3: The *changeText* algorithm

cent temporal intervals are collapsed into a single temporal interval. Coalescing may reduce the number of tridimensional temporal intervals for representing a temporal pertinence, and, as such, is a space optimization. Moreover, as each version can only redefine the temporal pertinence it inherits, the addition of a new version could also require the temporal pertinence of its ancestors to be extended. The `extendTimestamps` function determines the temporal pertinence te of the first ancestor version (steps 1-3) and checks whether it has to be extended in order to include the one of the newly added version (step 4). In this case, it calls the *changeTime* operator which is devoted to the extension of the temporal pertinence of an existing document portion. The *changeTime* call includes one chronon (vt, et, tt) of the temporal pertinence te of the ancestor version and the portion of the temporal pertinence of the newly added version to be added to te. The *changeTime* operator will be discussed in the following whereas, due to the lack of space, we do not present the code of the function `coalesce`. Notice that whenever the XML text is empty, the *changeText* operator performs an abrogation or a suspension.

The other operator, named *changeTime*, is devoted to the extension of the temporal pertinence of an existing document portion. Such operator too requires the path *path* allowing the identification of the portion of the passive norm undergoing temporal pertinence extension. Moreover, it also requires the temporal coordinate (one chronon (vt, et, tt)) of the version to be modified, a validity and efficacy temporal element $\bigcup_{i=1,...,n}[vt_{s_i}, vt_{e_i}] \times [et_{s_i}, et_{e_i}]$ and a reference to the active norm (an). As the previous operator, it acts by first identifying the node corresponding to the portion undergoing modification. Then it selects the version for which it is required a temporal extension. Such a version is the only one whose temporal pertinence contains the chronon (vt, et, tt). Then it adds the new tridimensional temporal element $\bigcup_{i=1,...,n}[vt_{s_i}, vt_{e_i}] \times [et_{s_i}, et_{e_i}] \times [now, UC]$ to the temporal pertinence of the selected version. Finally, it performs those operations required to comply with the inheritance semantics. As to the management of the sibling versions, the *changeTime* operator calls the `restrictTimestamps` function shown in Fig. 3. Moreover, the extension of the temporal pertinence of any version could also require the temporal pertinence of its ancestors to be extended. The algorithm then checks whether the ancestor version owns a temporal element which has to be extended. In this case, it applies the extension and then it continues upwards by recursively calling itself, else it stops (thanks to the inheritance semantics). Due to the lack of space, we do not show the code of the *changeTime* operator which is similar to the code of the *changeText* operator.

The worst-case complexity of both operators is linear in the number of nodes, since also recursive `restrictTimestamps` calls prevent the same subtree to be visited more than once. As a final remark, notice that, we did not include an operator performing the restriction of the temporal pertinence since it can be performed by means of the *changeText* operator whenever a suspension is required, or by means of the *changeTime* operator whenever the portion of the temporal pertinence to be deleted has to be substituted by the temporal pertinence of another version.

2.3 Querying aspects

Legal text repositories are usually managed by traditional information retrieval systems where users are allowed to access their contents by means of keyword-based queries expressing the subjects they are interested in. We have extended such a framework by offering to users the possibility of expressing temporal specifications for the reconstruction

of a consistent version of the retrieved normative acts (consolidated act). More precisely, we support queries having the following Xquery [18] form:

```
FOR $a IN path
WHERE constraints on $a
RETURN const-tree(document($a), temporal specs)
```

The `FOR...WHERE` construct follows the Xquery syntax and specifies selection constraints on the variable `$a` iterating over the nodes returned by the path expression `path`. Search keywords can be specified by means of the function `contains` [20] in the `WHERE` clause (e.g. `contains($a,'sea')`). In the `RETURN` clause, the operator `const-tree` is devoted to the reconstruction of the temporally consistent versions of the XML documents containing the selected nodes. The `temporal specs` expression is the conjunction of elementary constraints specified on the time values of the four supported temporal dimensions. Each elementary constraint has the form `DIMENSION [NOT] OP VALUE` where `DIMENSION` can be `PUBLICATION`, `VALIDITY`, `EFFICACY`, `TRANSACTION` in order to specify a constraint on the publication, validity, efficacy or transaction time pertinence of a normative text, respectively. The publication time of a normative text is a chronon which can be compared against a date (i.e. `VALUE` is a date) by means of the `OP` operators `=`, `PRECEDES`, and `FOLLOWS`, or which can be compared against a date interval (i.e. `VALUE` is a date interval) by means of the `OP` operator `CONTAINS`. The optional `NOT` keyword negates the operator which follows. The other temporal dimensions support temporal intervals and, thus, interval-based `OP` operators `PRECEDES`, `=`, `OVERLAPS`, `MEET` and `CONTAINS` can be used. Note that, since events may be perceived as special cases of intervals, events may occur as arguments where intervals are allowed (i.e. `VALUE` can be a date or a date interval). They are the standard operators which are usually adopted for the temporal selection in temporal databases [16]. Due to the lack of space we do not present the formal semantics which can be found, for instance, in [16].

EXAMPLE 2. *The following query asks for the version whose validity contains the date 01-01-1999 of the normative acts published before 01-01-2001 and containing the word "sea" in their paragraphs:*

```
FOR     $a IN //article/paragraph
WHERE   contains ($a, 'sea')
RETURN  const-tree(document($a),
        VALID CONTAINS '01-01-1999' and
        PUBLICATION PRECEDES '01-01-2001')
```

□

The `const-tree` operator takes as input the XML document (`document($a)`) containing a qualifying node and the temporal specifications. It reconstructs the document by selecting –at each level of the hierarchy– the portions whose temporal pertinence satisfies the specifications. The default value for those temporal dimensions not involved is `now`. The inheritance semantics allows us to prune out portions of the XML trees which may not be "interesting". Indeed, while visiting any XML tree, whenever we come across a node whose temporal pertinence does not satisfy the temporal specifications, then the sub-tree rooted by such a node can be pruned out as the temporal pertinence of its nodes is contained in that of the root.

3. PROTOTYPE IMPLEMENTATION AND EVALUATION

The model described in the previous section has been implemented in a prototype system. At present, the prototype manages large collections of norms in XML format with the aid of the XML document management facilities offered by Oracle 9i [13]. The system is easily accessible through a Web interface and efficiently supports queries involving temporal constraints and keywords.

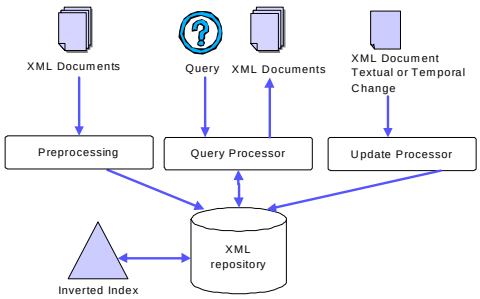


Figure 4: The overall system architecture

The overall architecture is depicted in Fig. 4. The temporal aspects of the XML model are managed on top of the DBMS where XML normative texts complying with the XML-Schema in Fig. 1 are stored as CLOBs into a table equipped with additional columns storing timestamping metadata. More precisely, the table `tnorms` has the following schema:

```
tnorms( ID, XML-DOC, TYPE, PUBLICATION,
        VT-START, VT-END, ET-START, ET-END,
        TT-START, TT-END )
```

Every tuple in this relation represents a temporal XML document. The timestamp attributes have the same values of the timestamping tags associated with the `contents` tag. The inheritance semantics of our model guarantees that they represent the summary time values of the whole norm text. The addition of such metadata is aimed at improving the efficiency of query execution by introducing a preliminary document filtering phase on the temporal predicates specified in the query. Moreover, as XML does not support the inheritance semantics of our model, all the timestamps which could be derived owing the semantics of inheritance are explicited at every level of the document hierarchy. In this way, by fully exploiting the potentialities of the XML query engine provided by Oracle 9i, we further speed up the processing of queries when conditions on temporal values are specified at different levels of the document tree structure. The extraction of document metadata and the explicitation of timestamps are performed once by the preprocessing module shown in Fig. 4 when new documents are inserted in the system.

When a query having the form shown in Subsec. 2.3 is issued in the system, the query processor first translates the request into a SQL query to be submitted to the ORACLE query engine and then applies the temporal reconstruction to the qualifying XML documents. In the first phase, the "static" part of the query (i.e. the `FOR...WHERE...` part) is straightforwardly translated into SQL calls. In order to support norm retrieval by keywords, an inverted index has been

built on the contents of selected XML elements (`heading` and `paragraph`). Moreover, in this phase we also exploit the fact that the outcome of the temporal reconstruction process is empty for any normative act having no portion which satisfies the temporal conditions. The temporal conditions are thus translated into SQL calls involving the temporal metadata columns in order to quickly discard tuples of the `tnorms` table representing normative acts which cannot be temporally qualifying.

EXAMPLE 3. *In the first phase, the query shown in Ex. 2 is translated into the following XML/SQL query complying with the Oracle query language syntax:*

```
SELECT L.XML-DOC.extract('//article').getStringVal()
FROM   tnorms L
WHERE  (VT-START <= '01-JAN-1999')
AND (VT-END is null OR VT-END > '01-JAN-1999')
AND (PUBLICATION <= '01-JAN-2001')
AND CONTAINS (L.XML-DOC, 'sea WITHIN paragraph') > 0
```

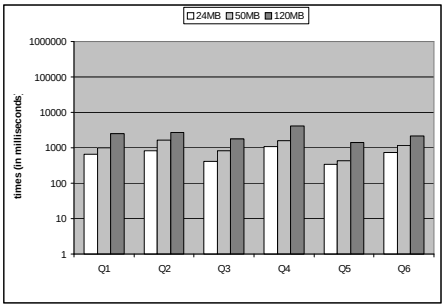
□

For each qualifying tuple, a norm reconstruction phase follows, during which the only parts which satisfy the temporal constraints are extracted. In this phase, the `const-tree` operator is executed by the query processor module.

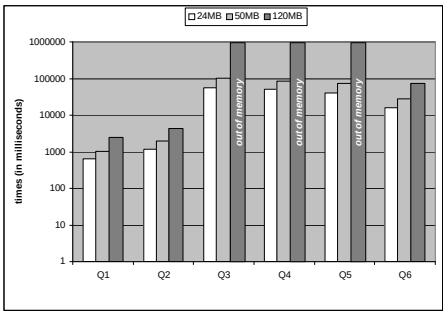
Finally, as to the management of changes that normative texts undergo during their life-cycle, the maintenance of the XML repository is managed by the update processor (implementing the *changeText* and *changeTime* operators).

To evaluate the effectiveness of the system we conducted a number of exploratory experiments by running the prototype above described on a 600Mhz Intel Pentium III processor with 256MB of main memory and a SCSI disk. In this paper, we report and discuss the most meaningful tests performed on three XML document sets of increasing size: 24MB (1000 documents), 50MB (2000 documents) and 120MB (5000 documents). The average, minimum and maximum document sizes are 24KB, 2KB and 125KB, respectively.

We first tested the system performance in query processing by separately evaluating the two phases: the retrieval of the qualifying normative texts and their temporal reconstruction. In the first phase, only those normative texts satisfying the static part and the temporal conditions specified in the submitted query are selected. Experiments were conducted by submitting queries of six different types: types q1 and q2 represent searches by keywords, in particular in q1 keywords are only specified on the `contents` subtree whereas in q2 keywords are specified both on the `type` and on the `contents` attribute, type q3 contains temporal conditions on the three dimensions, transaction, valid and publication time, and types q4, q5, and q6 mix the previous ones and contain both keywords and temporal conditions involving different parts and temporal dimensions. Fig. 5.a shows the average computing time in milliseconds (logarithmic scale) required to process four queries for each of the six types on the three document sets. For the smallest document set, the computing time ranges form 343 milliseconds up to 1.096 seconds whereas for the biggest document set, the computing time ranges from 1.432 seconds up to 4.064 seconds. From our experiments we can state that the presence of temporal constraints does not disrupt the system performance, even when a large number of documents is selected, as it happens for the queries of type q3 where the average number of selected documents is 1568 (over 5000). Moreover, the computing time grows sub-linearly with the



(a) Computing time using the `tnorm` table



(b) Computing time using the `Ptnorm` table

Figure 5: Selection of normative acts (log. scale)

number of documents thus showing the scalability of the system. In order to evaluate the improvement given by the temporal metadata columns of the `tnorm` table, we compared our approach with a "naive" approach consisting in the direct access to the XML documents. To this end, we recorded the normative acts into a simple table without metadata columns `Ptnorm(ID,XML-DOC)` and we relied for the selection of the normative texts on the Oracle XML engine by translating the temporal conditions into XML conditions on the temporal attributes in the `contents` element. The average computing time in this case is shown in Fig. 5.b. As we expected, the computing time for the first two query types, which do not involve temporal attributes, is the same, which means that the addition of the metadata columns does not affect the system efficiency. On the other hand, efficiency improvement given by the exploitation of the metadata columns is evident for the other four query types where our approach is even 127 times faster than the naive approach for the two smallest collections whereas the naive approach fails in processing queries involving temporal constraints on the biggest collection (types q3, q4, and q5). This is due to the fact that in processing pure temporal queries (type q3) or mixed queries involving keywords non very selective (types q4 and q5), the inverted indexes built on the XML column are very little useful or even useless. On the other hand, we also evaluated the time taken to insert XML documents in the two tables `tnorm` and `Ptnorm` and to update the related indexes. The overhead required for the extraction of the information to be inserted in the metadata columns of the table `tnorm` is about 35% in the worst case corresponding to the insertion of one document at a time (from 278 to 376 milliseconds) and it decreases up to 12% as the number of inserted documents increases for the insertion of about 30 documents (from 6376 to 7163

milliseconds).

As far as the temporal reconstruction of normative texts is concerned, we evaluated the computing time required to process 30 normative texts of different sizes (from 2KB to 140 KB) and with different number of versions (from 3 to over 50). For these documents, `cons-tree` takes 1 second on average. Such an outcome is also due to the fact that the temporal pertinence is represented by temporal elements. If we adopted temporal intervals, any version whose pertinence is the union of intervals would be divided into different redundant versions with the same content. In such a case, documents would be larger and the time taken for the temporal reconstruction would obviously increase. We also implemented such an alternative approach and compared it with the temporal element-based approach. We noticed that any normative text undergoing modifications (about 3) at different levels is three times larger on average when temporal intervals are adopted in place of temporal elements. In such a way, the average size of the documents managed by the system reaches 250KB. The difference between the two alternatives for the temporal reconstruction is not particularly remarkable (passing from 1 second to 1.2 second). The most important aspect to be considered is the main memory space the `cons-tree` function takes during its execution: It is even 3.5-4 times the dimension of the document (we used JDOM to navigate XML documents) and thus processing one XML document of 250KB requires about 1MB. It follows that in order to process many requests, the system might recur to virtual memory, with an unavoidable performance degradation. As far as modifications are concerned, the *changeText* and *changeTime* operations take more or less the same computing time as `cons-tree`. This is due to the fact that most of the time is used for the XML tree navigation which is approximately the same in the two cases.

4. CONCLUSIONS

The management of norms and their dynamics requires the adoption of temporally enhanced data models and systems. In this paper, we introduced a temporal XML data model which is able to capture the semantics of norms evolving in time and represent their multiple versions with respect to publication, validity, efficacy and transaction times. The model is based on an XML schema which allows the introduction of timestamping metadata at each level of the structure of the documents which are subject to change, up to the granularity of a single paragraph. A well-defined inheritance semantics rules the interaction between the different levels of the norm hierarchy and the temporal pertinence of the versions. Norm text modifications are dealt with by means of two basic operators which implement lossless changes through a careful management of the versions. Moreover, the model extends conventional searches by keyword with the possibility of specifying additional temporal constraints the retrieved normative acts must satisfy. Finally, a prototype implementing the model has been evaluated. The preliminary experimental results on query performance which have been reported in the paper are encouraging.

In future work, we will try to improve the performance of our system by considering alternative solutions for physical storage of temporal XML documents, indexing and query processing. For instance, solutions based on XML document decomposition and execution of structural joins seem promising starting points for extensions of our approach.

5. REFERENCES

[1] T. Amagasa, M. Yoshikawa, and S. Uemura. A data model for temporal XML documents. *Proc. of DEXA 2000*, England, 2000.

[2] S. S. Chawathe, S. Abiteboul, and J. Widom. Managing historical semistructured data. *Theory and Practice of Object Systems*, 5(3):143–162, 1999.

[3] R. T. S. Christian S. Jensen, Michael D. Soo. Unifying Temporal Data Models via a Conceptual Model. *Information Systems*, 19(7):513–547, 1994.

[4] C. Combi and A. Montanari. Data models with multiple temporal dimensions: Completing the picture. *Proc. of CAiSE 2001*, Switzerland, 2001.

[5] C. E. Dyreson, M. H. Böhlen, and C. S. Jensen. Capturing and querying multiple aspects of semistructured data. *Proc. of VLDB '99*, Scotland, 1999.

[6] O. Etzion, S. Jajodia, and S. M. Sripada, editors. *Temporal Databases - Research and practice*, volume 1399 of *LNCS*. Springer-Verlag.

[7] F. Grandi and F. Mandreoli. The valid web: an xml/xsl infrastructure for temporal management of web document. *Proc. of ADVIS'2000*, Turkey, 2000. LNCS 1909.

[8] C. S. Jensen, C. E. Dyreson (eds.) et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In [6].

[9] S.-K. Kim and S. Chakravarthy. Modeling time: Adequacy of three distinct time concepts for temporal data. *Proc. of ER' 93*, Texas, 1993.

[10] L. A. K. Manuk G. Manukyan. Temporal XML. *Proc. of ADBIS '01 – Vol. 1*, Lithuania, 2001.

[11] M. Nascimento and M. Eich. Decision time for temporal databases. *Proc. of TIME'95*, Florida, 1995.

[12] Norma in rete. http://www.normainrete.it.

[13] The Oracle 9i database. `otn.oracle.com/products/oracle9i/content.html`.

[14] F. Ost and M. Van Hoecke (eds.). *Time and Law. Is the Nature of Law to Last*. Bruylant, Belgium, 1998.

[15] M. Palmirani and R. Brighi. Norma-system: A legal document system for managing consolidated acts. *Proc. of DEXA 2002*, France, 2002.

[16] R. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer, New York, 1995.

[17] A. Tansel, J. Clifford, V. Gadia, S. Jajodia, A. Segev, R. Snodgrass (eds.). *Temporal Databases: Theory, Design and Implementation*. Benjamin/Cummings.

[18] W3C. XQuery 1.0: An XML Query Language. `http://www.w3c.org/TR/xquery/`, 2002.

[19] F. Wang and C. Zaniolo. Preserving and querying histories of xml-published relational databases. *Proc. of ECDM 2002*, Finland, 2002.

[20] W3C XQuery 1.0 and XPath 2.0 Functions and Operators. `http://www.w3.org/TR/xquery-operators/`.

[21] W3C XML Schema. `www.w3.org/XML/Schema`.

[22] W3C XML path language (XPath) 2.0. `http://www.w3.org/TR/xpath20/`.