CIT. Journal of Computing and Information Technology, Vol. 26, No. 2, June 2018, 115–129
doi: 10.20532/cit.2018.1004005

Transforming XML to RDF(S) with Temporal Information

Dan Yang and Li Yan

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

The Resource Description Framework (RDF) is a model for representing resources on the Web. With the widespread acceptance of RDF in various applications (e.g., knowledge graph), a huge amount of RDF data is being proliferated. Therefore, transforming legacy data resources into RDF data is of increasing importance. In addition, time information widely exists in various real-world applications and temporal Web data has been represented and managed in the context of temporal XML. In this paper, we concentrate on transformation of temporal XML (eXtensible Markup Language) to temporal RDF data. We propose the mapping rules and mapping algorithms which can transform the temporal XML Schema and document into temporal RDF Schema and temporal RDF triples, respectively. We illustrate our mapping approach with an example and implement a prototype system. It is demonstrated that our mapping approach is valid.

ACM CCS (2012) Classification: Information systems → World Wide Web → Web data description languages → Semantic web description languages → Resource Description Framework (RDF)

Information systems → World Wide Web → Web data description languages → Markup languages → Extensible Markup Language (XML)

Information systems → Data management systems → Database design and models → Data model extensions → Temporal data

Keywords: RDF, temporal RDF, temporal XML, mapping

1. Introduction

TRDF (Resource Description Framework) [1] as well as RDF Schema (RDFS for short) recommended by the W3C (World Wide Web Consortium) can be applied as a metadata model for describing the semantics and reasoning about resources on the Web. RDF and/or RDFS are written as RDF(S). Nowadays RDF(S) has been widely accepted and used by many fields (e.g., governments, organizations and companies) as the data model and representation format, either for semantic data search and processing [28], or for representation of data from information extraction and integration. With the increasing amount of RDF data which is becoming available, transforming legacy data resources into RDF data is of increasing importance. Some efforts have been devoted to transformation of various data resources (e.g., relational databases [2] and UML class diagrams [3]) to RDF(S). Among various legacy data resources, XML (eXtensible Markup Language) [4] recommended by the W3C is the de-facto standard for data representation and exchange over the Web. XML has been extensively applied in many applications and a large volume of data is managed today directly in XML format. For this reason, some efforts have been made to transform XML into RDF(S) [5] − [11].

Time information widely exists in various real-world applications, including many web-based applications. Actually, many web data are typically time-related [12]. Some data on the Web, for example, are valid only at a certain time point or in a given time interval. In addition, some historical web data may need to be recorded and managed [13], [14]. In order to model and manage temporal information on the Web, temporal XML model has been proposed, because of the wide utilization of XML for data

representation and exchange on the Web (e.g., [15] − [20]). Note that XML cannot represent semantic information of data, although XML has been extensively used and a large volume of XML data is available. With the widespread acceptance and utilization of RDF as a metadata model for semantic data representation and processing on the Web, temporal RDF modeling has recently received more attention [22], [24], [25].

In this paper, we concentrate on modeling temporal information both in XML and RDF(S). In particular, we introduce the formal definitions of temporal XML model, including temporal XML Schema and temporal XML document, and temporal RDF(S) model. On this basis, we propose the formal approach to mapping the temporal XML model to the temporal RDF(S). We present the mapping rules and mapping algorithm. We apply an example and implement a prototype system to demonstrate that our mapping approach is valid. With the mapping approach proposed in the paper, temporal XML can be transformed into temporal RDF(S) and this serves as automation construction of temporal RDF(S) and further as semantic temporal information extraction and integration.

The rest of this paper is organized as follows. We present related work in Section 2. In Section 3, we introduce temporal data models we use in the paper, which are temporal XML model and temporal RDF model. We propose our approach to mapping temporal XML model to temporal RDF model in Section 4, including the rules and algorithms of mapping temporal XML Schema to temporal RDF Schema and temporal XML document to temporal RDF triples, respectively. We design and implement a prototype in Section 5. Section 6 concludes this paper.

2. Related Work

The present work in this paper is closely related to two issues, which are classical XML transformation to RDF and temporal data modeling in XML and RDF.

2.1. Temporal Data Modeling

Temporal data modeling has been earlier investigated in the context of databases (e.g., [26]). With the wide utilization of XML for data representation and exchange on the Web, temporal XML modeling has received more attention. Time information is first considered in managing changes of semi-structured data [13], [14]. The time dimensions of temporal XML mainly include valid time and/or transaction time. In various temporal XML models proposed in literature, timestamps are explicitly added to nodes or/and edges of XML trees. In [15], two physical implementation models are presented for filling temporal data into XML documents. XML-based bitemporal data model (XBiT) proposed in [16] can manage valid time and transaction time histories. Without temporal information about XML Schema, the schemes for timestamps in XML documents are considered in [18]. A complex framework named τXSchema is presented in [17] to record the time-varying elements and physical implementation position. Temporal constraints are added to XML Schema in [19]. The issues of modeling, indexing, and querying temporal XML are investigated in [20].

Temporal RDF model is proposed to represent and manage temporal web data in a semantic way. In [21], [22], temporal reasoning is incorporated into RDF, yielding temporal RDF graphs. A temporal triple is defined as an RDF triple with a time interval. The syntax and deductive system for temporal RDF graphs are presented in [21], [22]. A very different temporal RDF model is proposed in [25], in which a temporal triple contains time information in its property component instead of a whole triple in [21], [22]. Several issues such as indexing [25], reasoning [23] and query [24] are investigated in the context of temporal RDF.

2.2. RDF Transformation from XML

Transformation of various data resources (e.g., relational databases [2] and UML class diagrams [3]) to classical RDF has been a crucial issue in RDF data management. In particular, some proposals establish the correspondences from XML to RDF(S). In [5], a procedure that transforms XML documents into RDF statements via an RDF-Schema specification is presented. A unified model for both XML and RDF by XML XQuery 1.0 and XPath 2.0 is provided in [6].

Transforming valid XML documents into RDF via RDF Schema is discussed by utilizing DTD (Document Type Definition) and XSD (XML Schema Definition) in [7] and [8], respectively. Considering the semantic similarity of duplicate elements in XML schema (XSD or DTD), in [10] a set of rules is presented, which derives classes, properties, and data types from XML schema and interprets XML data as RDF statements by using RDF schema vocabularies. An approach for automatic transformation from XML to RDF via XML Schema is proposed in [9].

3. Temporal Data Models

We concentrate in this paper on the temporal XML and RDF data models containing two time-dimensions: valid time and transaction time. The valid time refers to the time when the data is true in the modeled reality and the transaction time denotes the time at which the information is edited. We apply two special time labels: now and UC, to indicate the current time for valid and transaction, respectively. In addition, we apply the point-based time domain and encode some continuous time points with time intervals. We introduce two operations for time intervals, which are interval union and interval intersection (union and interval intersection, for short), respectively. Let t_1, t_2, ... be time intervals. Interval union, denoted $\cup(t_1, t_2, \ldots)$, is applied to calculate the union of these time intervals and interval intersection, denoted $\cap(t_1, t_2, \ldots)$, is applied to calculate the intersection of these time intervals.

In this section, we present the temporal models of XML Schema, XML document, RDF, and RDFS, respectively. We introduce the formal definitions of these temporal data models. Note that, for simplicity, only one time-dimension is presented at a formal definition and the other can be addressed in an analogous way.

3.1. Temporal XML Schema

XML Schema is used to define the structure of XML documents shared between applications. It always contains various elements and attributes with different structures and types. Following the step of [22], we extend the XML Schema model to support temporal information.

Definition 1. (temporal XML Schema model). A temporal XML Schema model is a tuple (S, T, M), in which

1. $S = \{S_1, \ldots, S_n\}$ is a set of XML Schemas and an XML Schema (say $S_i \in S$) is a tuple $(E_i, A_i, D, \rho, \kappa, \tau)$ [11], in which E_i is a set of elements, A_i is a set of attributes, and D is a set of simple datatypes which contains the built-in datatypes of XML Schema. The set of attributes of an element $e \in E_i$ is defined by ρ: $E_i \rightarrow 2^{A_i}$ and κ: $E_i \cup A_i \rightarrow D \mid \epsilon$ specifies the simple types of elements and attributes (ϵ is for the empty type). Specifically, an element type may be more complex and can be presented by $\tau = \epsilon \mid Order\ [e_1 : \tau_1, \ldots, e_n : \tau_n\ ]$, $e_j \in E_i$, where *Order* is an order indicator for (*Sequence*, *Choice*, *All*) of XML Schema.
2. T is a set of times.
3. $M : S \rightarrow 2^T$ is a timestamp function that maps an XML Schema to a timestamp (a set of times).

Definition 2. (snapshot of temporal XML Schema). Let $TXS = (S, T, M)$ be a temporal XML Schema model. For $t \in T$, snapshot $(t, TXS) = S_i$, in which $S_i \in S$ and $t \in M(S_i)$.

```
<TemporalSchema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <SchemaVersion path="../TemporalBook.xsd" vstart="x" vend="x" tstart="x" tend="x">
  <element target="/xs:schema/xs:element[1]" vstart="x" vend="x" tstart="x" tend="x" />
  <element target="/xs:schema/xs:element[2]" vstart="x" vend="x" tstart="x" tend="x" />
  ……
 </SchemaVersion>
 <SchemaVersion ..>
  ……
 </SchemaVersion>
</TemporalSchema>
```

Figure 1. An example of temporal XML Schema.

The notion of *temporal XML Schema* is applied to record different versions of XML schemas. Figure 1 depicts an example of temporal XML Schema. Here element *SchemaVersion* is used to record the version information of XML Schema and its children can be elements or other standard notations of XML Schema. Each child contains an attribute *target* which is used to record the location of the corresponding XML Schema via XPath. Note that, in the temporal Schema, the specified paths are the time intervals.

In order to accommodate temporal information in XML Schema, it is necessary to modify the corresponding XML Schema so that it can support temporal XML documents. For this purpose, we introduce several predefined structures which can be quoted to mark time varying contents. Note that these predefined structures can be contained in other schemas.

The first structure is an attributeGroup named *temporalAttr*, which contains four temporal attributes vStart, vEnd, tStart, and tEnd, representing the starting times and ending times of a valid time and a transaction time, respectively (Figure 2).

The second structure is a complexType named *temporalAttrType*, which is used by time varying attribute element (Figure 3).

The third structure is an element named *temporalText*, which can be contained by such elements that their texts are changing with time.

3.2. Temporal XML Documents

An XML document tree contains four kinds of nodes, which are *root*, *element*, *attribute* and *text* nodes, respectively. Actually, a root node is a special element node, which represents the beginning of an XML document uniquely. Connecting nodes forms the edges of XML tree.

A temporal XML document is also represented as an XML tree. As shown in Figure 5, we attach bi-temporal annotations to tree nodes rather than to tree edges. Note that the valid or

```
<xs:attributeGroup name="temporalAttr">
  <xs:attribute name="vStart" type="xs:date"/>
  <xs:attribute name="vEnd">……</xs:attribute>
  <xs:attribute name="tStart" type="xs:date"/>
  <xs:attribute name="tEnd">……</xs:attribute>
</xs:attributeGroup>
```

Figure 2. temporalAttr structure.

```
<xs:complexType name="temporalAttrType">
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="isAttr">……</xs:attribute>
    <xs:attributeGroup ref="temporalAttr"/></xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Figure 3. temporalAttrType structure.

```
<xs:element name="temporalText">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attributeGroup ref="temporalAttr"/>
    </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

Figure 4. temporalText structure.

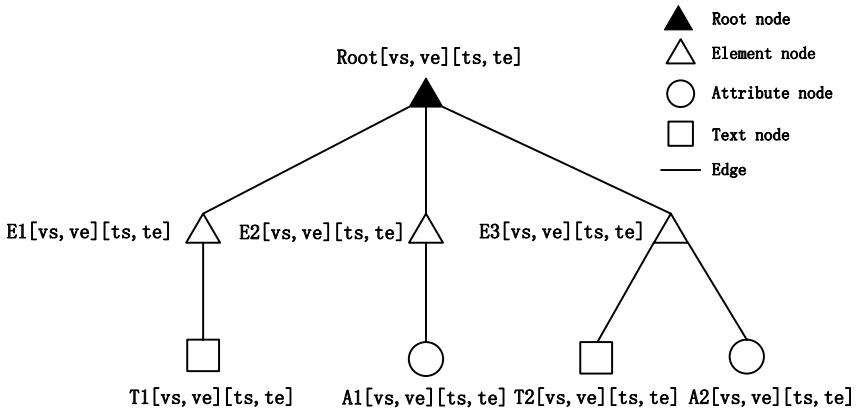

Figure 5. An example of temporal XML tree.

transaction time of a child node is contained by its parent node.

Definition 3. (Temporal XML document model). A temporal XML document model is a tuple (X, T, M) [19], where

1. $X = \{X_1, \ldots, X_n\}$ is a set of XML documents and an XML document (say $X_i \in X$) is tuple (N_i, ED_i, μ). Here N_i is a set of nodes (including all types of nodes in XML tree); $ED_i = \{(n_1, n_2)\}$ (n_1, $n_2 \in N_i$) is a set of edges1; μ: $N_i \rightarrow \epsilon \mid E_i \cup A_i$ maps the nodes of the XML document model to an element set E_i and an attribute set A_i of an XML Schema.
2. T is a set of times.
3. M: $X \rightarrow 2^T$ is a timestamp function that maps an XML document to a timestamp (a set of times).

Definition. (snapshot of temporal XML document). Let $TX = (X, T, M)$ be a temporal XML document model. For $t \in T$, snapshot (t, TX) = X_i, in where $X_i \in X$ and t $\in M(X_i)$.

An element of temporal XML documents is assigned with four special attributes *vstart*, *vend*, *tstart* and *tend*, which represent the valid and transaction interval of this element, respectively. When the attribute or text of an element is varying with time, we denote the element using a special attribute *isAttr* and represent the attribute or text with a subelement, in which the text node is denoted by an element named *TextNode*.

Let us look at an example. Suppose that we have a temporal element named `phone`, which has a text node which changes with time. Then we have a fragment of temporal XML document shown in Figure 6.

3.3. Temporal RDF(S)

RDF can be used to express propositions using precise formal vocabularies. W3C has given the semantics of RDF with model theory, which is regarded as "interpretation theory" in [27]. To represent time information in RDF, we need to add temporal information to this theory. In this paper, we adopt the temporal RDF model proposed in [22] with both valid and transaction dimensions by labeling.

Definition 4. (Temporal RDF model). A simple temporal interpretation of RDF is a tuple (I, T, M), in which

```
<phone isAttr =" True" vstart= "2008/03/23" vend="now"
                      tstart="2008/07/12" tend="UC">023-546887>
</phone>
<TextNode vstart= "2012/05/21" vend="now" tstart="2012/05/22" tend="UC">
  This is a text node changing with time.
</TextNode >
```

Figure 6. A fragment of temporal XML document.

1. $I = \{I_1, \ldots, I_n\}$ is a set of simple interpretations. In $I_i = (C_i, P_i, R_i, Ext, CExt) \in I$, C_i is a set of classes; P_i is a set of properties; R_i is a set of all resources, which is actually the universe of RDF(S), containing a distinguished subset L_i called literal values; Ext: $P_i \rightarrow R_i \times R_i$ is used to express the relationship between resources; $CExt$: $C_i \rightarrow 2^{R_i}$ maps a class $c \in C_i$ to a subset of R_i (i.e., $C_i = CExt$ (*rdfs:Class*), which means each element of C_i is an extension of *rdfs:-Class*).
2. T is a set of times.
3. M: $I \rightarrow 2^T$ is a timestamp function that maps an interpretation to a timestamp (a set of times).

Definition 5. (Snapshot of temporal RDF model). Let $TI = (I, T, M)$ be a simple interpretation. Then for $t \in T$, snapshot $(t, TI) = I_i$, where $I_i \in I$, $t \in M(I_i)$.

A temporal triple is an RDF triple with temporal labels and the notation, denoted (a, b, c): $[v]$ $[t]$. Here (a, b, c) denotes a triple with subject a, property b and object c, $[v]$ denotes the valid time of triple, and $[t]$ denotes the transaction time of triple. Actually, the expression (a, b, c): $[v_1, v_2]$ $[t_1, t_2]$ is a notation for $\{(a, b, c)$: $[v]$ $[t]$ $| v_1 \le v \le v_2, t_1 \le t \le t_2\}$. Note that the temporal labels are applied at the level of triples, not the level of the subject, property or object of triples.

We use the classical RDF graph with temporal vocabularies to represent temporal RDF. A temporal RDF graph is a set of temporal triples, which consists of a set of (a, b, c), $(X, tsubj, a)$, $(X, tpred, b)$, $(X, tobj, c)$ and $(X, valid, V)$, $(X, trans, T)$. Here, X denotes the statement of triple (a, b, c), and V and T represent the valid and transaction time, respectively.

Figure 7 shows an example of temporal RDF graph with temporal triples, in which the triple X is valid from time 3 to now. In other words, it exists in the knowledge base from time 3 to current time (UC). Here we use three labels of interval, initial and final to describe a period of time, the start time of the period and the end time of the period, respectively. In Figure 6, Y and Z represent the time intervals of valid time ([3, Now]) and transaction time ([3, UC]) of triple X, respectively.

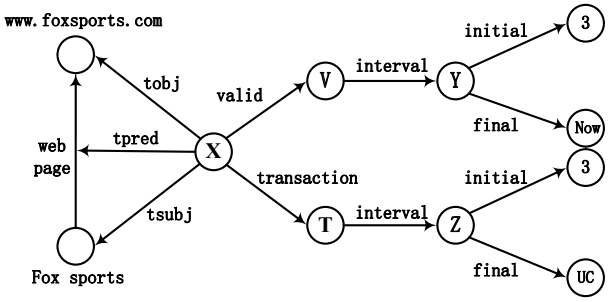

Figure 7. An example of temporal RDF graph.

4. Mapping Temporal XML to Temporal RDF(S)

As we know, XML Schema defines all elements and attributes with relevant structure and types of XML documents, which generally contain rich semantic information. So, transforming temporal XML to temporal RDF(S) is conducted at two levels. The first level is to extract temporal RDF Schema from temporal XML Schema and the second level is to transform temporal XML document to temporal RDF triples.

In this section, we propose the rules for mapping temporal XML to RDF(S) based on the formal definitions of two temporal data models. The detailed algorithm for mapping temporal XML document to RDF triples will be clarified. At last, we employ an example to illustrate the complete transforming procedure.

4.1. Temporal XML Schema Mapping

Some generic mapping rules from non-temporal XML Schema to RDFS have been proposed. But they cannot be directly applied to extract temporal RDFS from temporal XML Schema. For our purpose, we define a mapping function f: *XML Schema* $\rightarrow$ *RDFS*, which maps a snapshot $S_t = (E_t, A_t, D, \rho, \kappa, \tau)$ of temporal XML Schema at time point t to the corresponding RDF Schema $I_t = (C_t, P_t, R_t, Ext, CExt)$, that is,

$$f(S_t) = f(E_t, A_t, D, \rho, \kappa, \tau) = (C_t, P_t, R_t, Ext, CExt) = I_t.$$

The common properties of RDF Schema are presented as follows:

rdf: *type*, *rdfs*: *domain*, *rdfs*: *range*,
rdfs: *subClassOf*, *rdfs*: *subPropertyOf*.

Based on the formal definitions of temporal XML Schema and temporal RDF(S) above, we propose the following mapping rules, which establish the major correspondences of different components in these two temporal data models.

1. For attribute $@a \in A_t$, we have $f(@a) \in P_t$;
2. Simple datatypes D of XML Schema are mapped to resources of RDF. We have $dt \in D$, $\langle f(dt)$, *rdfs*: *Datatype* $\rangle \in Ext$ (*rdf*: *type*);
3. For attribute $@a \in A_t$ and $\kappa(@a) = dt$, $dt \in D$, we have $\langle f(@a), f(dt) \rangle \in Ext$ (*rdfs*: *range*);
4. For element $e \in E_i$ and $\kappa(e) = dt$, $dt \in D$, we have $f(e) \in P_t$, $\langle f(e), f(dt) \rangle$ *Ext* (*rdfs*: *range*);
5. For element $e \in E_i$ and $\tau(e) \neq \epsilon$, we have $f(e) \in C_t$.

Temporal information in XML Schema is directly mapped to the corresponding temporal RDFS. To ensure the correctness of RDFS, snapshot mapping based on time points is applied with the function *f*. As we know, XML Schema is made up of three main building blocks: *elements*, *attributes* and *type definitions* and it contains many built-in datatypes (e.g., *xs: string* and *xs: decimal*). First, RDFS does not have built-in datatypes. Rule 2 can map the built-in datatypes of XML Schema to the instances of *rdfs: Datatype*, which is a predefined class of RDFS. Second, the attributes of XML Schema are usually used to describe the features of elements which are similar to the properties of RDFS. We can interpret the attributes of XML Schema as the properties of RDFS with Rule 1 and get the range information with Rule 3. Note that, however, the domain of a property is related to the attribute's location in the XML Schema, which can be extracted by specific processing. Third, we identify two categories of elements: *simple elements* and *complex elements*. The simple elements of XML Schema contain only the text without any attributes and subelements and can be directly interpreted as the properties of RDFS, which range restrictions are obtained with Rule 4. The complex elements of XML Schema, however, are translated into the classes of RDFS with Rule 5, because they possess complex data structures with attributes and subelements.

Based on the mapping rules above, we present the following algorithm to map a temporal XML Schema model into a temporal RDFS model.

For a snapshot of temporal XML Schema, it is parsed in a depth-first order and then all elements, attributes and types are mapped with the mapping rules above. Note that two declaration types *global* and *local* in XML Schema cannot be mapped directly by the formal definitions. So, the algorithm proposed above does not deal with the references themselves. The references of *complexTypes* can be mapped to classes and the elements which refer to the *complexTypes* are further mapped to subclasses of the mapped classes. In addition, we can exploit some domain information of properties by using positional relationship between elements and attributes.

4.2. Temporal XML Document Mapping

With the RDF Schema mapped from the temporal XML Schema, we can map the corresponding temporal XML document to temporal RDF. We define a mapping function g : *XML document* $\rightarrow$ *RDF*, which maps a snapshot $X_t = (N_t, ED_t, \mu)$ of temporal XML document at time point e to the corresponding RDF $I_t = (C_t, P_t, R_t, Ext, CExt)$, that is,

$$g(X_t) = f(N_t, ED_t, \mu) = (C_t, P_t, R_t, Ext, CExt) = I_t.$$

Based on the formal definitions of temporal XML document and temporal RDF above, we propose the following mapping rules which establish the major correspondences among different components in these two temporal data models.

1. If node $n \in N_t$ and $\mu(n) = e \in E_t$ and $\tau(e) \neq \epsilon$, we have $g(n) \in R_t$, $\langle g(n), f(e) \rangle \in Ext$ (*rdf*: *type*);
2. If node $n \in N_t$ and $\mu(n) = e \in E_t$ and $\kappa(e) \neq \epsilon$, for pair $(p, n) \in ED_t$, we have $\langle g(p), g(l) \rangle \in Ext(f(e))$, where l is a string value;
3. If node $n \in N_t$ and $\mu(n) = @a \in A_t$, for pair $(p, n) \in ED_t$, we have $\langle g(p), g(l) \rangle \in Ext(f(@a))$, where l is a string value of node n;

4. If node $n \in N_t$ and $\mu(n) = \epsilon$ (it is a text node), there are two possibilities for pair $(p, n) \in ED_t$. First, if $\mu(p) = e \in E_t$ and $\tau(p) \neq \epsilon$, we have $\langle g(p), g(l)\rangle \in Ext(rdf: value)$, where l is the string value of node n. Second, if $\mu(p) = e \in E_t$ and $\kappa(q) \neq \epsilon$, we have $\langle g(ancestor), g(l)\rangle \in Ext(f(e))$, where *ancestor* is the parent of node p.

The temporal information of XML document can be directly mapped to the corresponding temporal RDF. We map the snapshot of temporal XML document with the function g. As we know, XML Schema is used to constrain the structure of XML documents. All elements and attributes in XML documents come from the corresponding XML Schema. The elements of XML document are the instances of the elements of XML Schema. As mentioned in Subsection 4.1, the elements of XML Schema may be parsed into classes or properties of the RDFS. For the former elements of XML Schema, the corresponding elements of XML document are mapped into the instances of the mapped classes with Rule 1. For the latter elements of XML Schema, the corresponding elements of XML document, which have concrete values (say, *l*), are mapped to the predicates of specific triples with object *l*. Rule 2 and Rule 3 are used to clarify the mappings of these elements and attributes. In an XML tree, there is a kind of nodes named text nodes, which are presented as formatted strings. In Rule 4, two situations in the parent nodes (say, *p*) of the text nodes are discussed. If *p* is converted to an instance of class, the text node is mapped to the object of the triple with subject *p* and predicate *rdf*: *value*. If *p* is parsed into a property, the text node is mapped to an object of the triple whose subject is the ancestor (the parent of *p*) resource and the predicate is the property that *p* is mapped to.

To map a temporal XML document to a temporal RDF, the temporal XML document tree is parsed in a depth-first order. Each node in the XML document tree is identified if it is a node for a property or a class. The validity of nodes is verified via temporal information according to the generated schema. In addition, the resources described in RDF have unique identifiers, denoted by URIs. So, we jointly utilize the namespaces of the temporal XML document and its location in the temporal XML document, which is presented by XPath, to represent resources while generating the instances of RDF classes. Based on the mapping rules above, we present the following algorithm to map a temporal XML document model into a temporal RDF model.

In Algorithm 2, we use *resource* to represent the current XML tree node and semi namespace-qualifier "*onto*:" to denote the objects in RDF Schema. In addition, we introduce new RDF Schema properties *rdfx*: *describes* and *rdfx*: *hasClass* to deal with two special cases. Here *rdfx*: *describes* connects the resource of the XML document itself (its URI) with the root class it describes, and *rdfx*: *hasClass* is defined to solve a problem when two directly connected labels in the XML tree are both identified as classes. It can be observed from Algorithm 2 that time dimensions in the temporal XML document are mapped to attributes of elements. The time that limits the relationship between two resources in RDF is presented as a label. So, we use the intersection of time intervals in resources to label the relevant triples.

Note that XML data is not always time varying and some data lack temporal information. To handle this problem, we propose three additional rules in order to apply the proposed algorithm.

1. If an element (including non-temporal attributes) lacks valid information, its temporal information follows the time of the schema.
2. The valid time of an element must be contained by the valid time of its ancestors.
3. If a transaction time is absent, the algorithm follows the transaction time of the parents.

4.3. An Example Illustration

To illustrate the proposed approach in translating temporal XML into temporal RDF(S), we present an example of temporal XML about e-bookstore. The temporal XML Schema of the example is presented in Figure 8.

According to the schema mapping rules and the corresponding mapping algorithm, the temporal XML Schema shown in Figure 8 is mapped to the classes and properties of temporal RDFS, which are shown in Table 1.

Algorithm 1. Extracting temporal RDF Schema.

```
Input: stemporal XML Schema
Output: temporal RDFS
  for each element SchemaVersion of the root element TemporalSchema {
      schemaPath = getSchemaVersionPath();
      (sv,st) = getAttributeTimeVT();
      Map<String, Time> childNodesMap = createEmptyNodesPath();
      for each child of SchemaVersion {
          xpath = getAttributePath();
          (V, T) = getAttributeTimeVT();
          insertChildNodesMap(childNodesMap, xpath, (V, T)); }
      parseSchema(schemaPath, childNodesMap, (sv,st)); }
  parseSchema(path, map, (SV, ST)) {
      root = getSchemaRootElement(path);
      activeTime (AV, AT) = (SV, ST);
      visitNode(root, activeTime); }
  visitNode(node, activeTime (AV, AT)) {
      qname = getNodeQName();
      nodeName = getAttributeName();
      xpath = getNodeXpath();
      (V, T) = check(xpath,map);
      if (V, T) != null {
          set activeTime=(∩ (AV, V), ∩ (AT, T)); }
      if checkIsExistInRDFS(nodeName) == true {
          (oldV, oldT) = getExistTime(nodeName);
          setNewTime(nodeName, (∪ (oldV, AV), ∪ (oldT, AT))); }
      else {
      switch(qname) {
      case "element":
          if getAttributeRef() != null {
              if isComplexType(getAttributeRef()) == true {
                  createClass(nodeName, activeTime);
                  setParentClass(nodeName, getAttributeRef()); }
              break; }      /*checked with mapping rules and map element to RDFS*/
          mapWithRules(xpath, activeTime);
          if checkIsClass(nodeName) == true {     /* activeClass is a global variable to record current valid class*/
              set activeClass = nodeName; }
          else {
              setPropertyDomain(nodeName, activeClass); }
          break;
      case "attribute":
          if getAttributeRef() != null {
          break; }      /* checked with mapping rules and map attribute to RDFS*/
          mapWithRules(xpath, activeTime);
          if isGlobal(xpath) == false {
              setPropertyDomain(nodeName, activeClass); }
          break;
      case "complexType":
          if isGlobal(xpath) == true & onlyTemporalAttribute(xpath) == false {
              createClass(nodeName, activeTime); }
          break;
  }}
  for each child of this node {
          visitNode(childNode, activeTime); }
  }
```

Algorithm 2. Interpreting temporal XML document to temporal RDF.

```
Input: doc_Uri, the uri of temporal XML document
Output: temporal RDF triples
  root = getRootElement(doc_Uri);
  activeRes = getNodeXpath();
  set activeTime(AV, AT) = (infinite, (doc_CreationTime, UC));
  visitNode(root, activeTime);
  visitNode(node, activeTime(AV, AT)) {
      nodeName = getNodeName();
      resource = getNodeXpath();
      (V, T) = getAttributeTimeVT();
      if (V, T) == null {
          set (V, T) = (infinite,infinite) }
      (XV, XT) = getSchemaTime(nodeName);
      switch(getNodeType) {
          case "class":
              if node == root {
                  createTriple(doc_Uri, rdfx:describes, resource, (∩(AV, XV, V), ∩(AT, T))); }
              if isfinished(getLastTriple()) == false {
                  (LV, LT) = getTime(getLastTriple());
                  finshTriple(getLastTriple(), resource, (∩(LV, XV, V), ∩(LT, T))); }
              else {
                  createTriple(activeRes, rdfx:hasClass, resource, (∩(AV, XV, V), ∩(AT, T)));
              }
              onto:Class = getClass(nodeName);
              createTriple(resource, rdf:type, onto:Class, (∩(AV, XV, V), ∩(AT, T)));
              set activeRes = resource;
              set activeTime = (∩(AV, XV, V), ∩(AT, T));
              break;
          case "property":
              if isfinished(getLastTriple()) == false {
                  (LV, LT) = getTime(getLastTriple());
                  finshTriple(getLastTriple(),activeRes, (∩(LV, AV), ∩(LT, AT))); }
              onto:Prop = getProperty(nodeName);
              createTriple(activeRes, onto:Prop, null, (∩(AV, XV, V), ∩(AT, T)));
              break;
          default:
              text = getText();
              if isfinished(getLastTriple()) == false {
                  (LV, LT) = getTime(getLastTriple());
                  finshTriple(getLastTriple(), text, (∩(LV, AV), ∩(LT, AT))); }
              else {
                  createTriple(activeRes, rdf:value, text, (∩(AV, V), ∩(AT, T))); }
      }
      for each child of this node {
              visitNode(childNode, activeTime); }
  }
```

Figure 9 presents a temporal XML document which has the temporal XML Schema shown in Figure 8.

Using the document mapping rules and the corresponding mapping algorithm, the temporal XML document shown in Figure 8 is mapped to the temporal RDF triples shown in Table 2, which have the temporal RDFS shown in Table 1.

```
BookStore_TemporalSchema.xml:
<TemporalSchema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <SchemaVersion path="../BookStore.xsd" vstart="2010-07-18" vend="now"
                                        tstart="2010-08-01" tend="UC">
  <element target="//xs:element[@name='book']//xs:element[@name='cost']"
                vstart="2010-07-18" vend="2013-03-21" tstart="2013-04-01" tend="UC"/>
  <element target="//xs:element[@name='book']//xs:element[@name='price']"
                vstart="2013-03-22" vend="now" tstart="2013-04-01" tend="UC"/>
 </SchemaVersion>
</TemporalSchema>

BookStore.xsd:
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="bookStore">
  <xs:complexType><xs:sequence>
   <xs:element name="owner" type="xs:string"/>
   <xs:element name="book" maxOccurs="unbounded">
    <xs:complexType><xs:sequence>
     <xs:element ref="temporalText" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element name="name" type="temporalAttrType" maxOccurs="unbounded"/>
     <xs:element name="cost" type="xs:decimal" minOccurs="0"/>
     <xs:element name="price" type="priceType" maxOccurs="unbounded"/>
    </xs:sequence>
     <xs:attribute name=" isbn " type="xs:string"/>
     <xs:attributeGroup ref="temporalAttr"/></xs:complexType></xs:element>
   </xs:sequence>
  </xs:element>
  <xs:complexType name="priceType">
   <xs:simpleContent><xs:extension base="xs:decimal">
    <xs:attributeGroup ref="temporalAttr"/></xs:extension>
   </xs:simpleContent>
  </xs:complexType>
  <xs:attributeGroup name="temporalAttr">……</xs:attributeGroup>
  <xs:element name="temporalText">……</xs:element>
  <xs:complexType name="temporalAttrType">……</xs:complexType>
</xs:schema>
```

Figure 8. A fragment of temporal XML Schema.

```
BookStore.xml (2011-03-11)
<bookStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                      xsi:noNamespaceSchemaLocation="BookStore.xsd">
 This bookstore supplies books about computer.
 <owner>July</owner>
 <book isbn="978-7-302-40864-2" vStart="2011-06-10"
                               vEnd="now" tStart="2011-06-12" tEnd="UC">
  <name isAttr="true" vStart="2011-06-10" vEnd="now" tStart="2011-06-12"
                               tEnd="2013-08-19">Computer Organization </name>
  <name isAttr="true" vStart="2011-06-10" vEnd="now" tStart="2013-08-20"
                               tEnd="UC">Computer Organization and Architecture</name>
  <cost>38</cost>
   <price vStart="2011-06-10" vEnd="now" tStart="2013-08-20"
                               tEnd="2014-02-27">38</price>
  <price vStart="2014-02-28" vEnd="now" tStart="2014-02-28" tEnd="UC">42.5</price>
 </book>
</bookStore>
```

Figure 9. An example of temporal RDF.

Table 1. An example of classes and properties in temporal RDFS.

property	domain	range	validTime	transactionTime
owner	bookStore	xs:string	(2010-7-18, now)	(2010-08-01, UC)
name	book	xs:string	(2010-7-18, now)	(2010-08-01, UC)
cost	book	xs:decimal	(2010-07-18, 2013-03-21)	(2013-04-01, UC)
price	book	xs:decimal	(2013-03-22, now)	(2013-04-01, UC)
isbn	book	xs:string	(2010-7-18, now)	(2010-08-01, UC)

Table 2. An example of temporal XML.

sub	pred	obj	validTime	transactionTime
../BookStore.xml	rdfx:describes	/bookStore	(2010-7-18, now)	(2011-03-11, UC)
/bookStore	rdf:type	onto:bookStore	(2010-7-18, now)	(2011-03-11, UC)
/bookStore	rdfx:value	This bookstore supplie….	(2010-7-18, now)	(2011-03-11, UC)
/bookStore	onto:owner	July	(2010-7-18, now)	(2011-03-11, UC)
/bookStore	rdfx:hasClass	/bookStore/book	(2011-06-10, now)	(2011-06-12, UC)
/bookStore/book	rdf:type	onto:book	(2011-06-10, now)	(2011-06-12, UC)
/bookStore/book	onto:isbn	978-7-302-40864-2	(2011-06-10, now)	(2011-06-12, UC)
/bookStore/book	onto:name	Computer Organization	(2011-06-10, now)	(2011-06-12, 2013-08-19)
/bookStore/book	onto:name	Computer Organization and...	(2011-06-10, now)	(2013-08-20, UC)
/bookStore/book	onto:cost	38	(2011-06-10, 2013-03-21)	(2011-06-12, UC)
/bookStore/book	onto:price	38	(2013-03-22, now)	(2013-08-20, 2014-02-27)
/bookStore/book	onto:price	42.5	(2014-02-28, now)	(2014-02-28, UC)

5. Prototype Implementation

Based on the mapping rules and algorithms proposed above, we design and implement a prototype system called *TX-TR* as proof-of-concept, which can map temporal XML Schema and documents into temporal RDF(S). In the following, we briefly discuss the design and implementation of the prototype system.

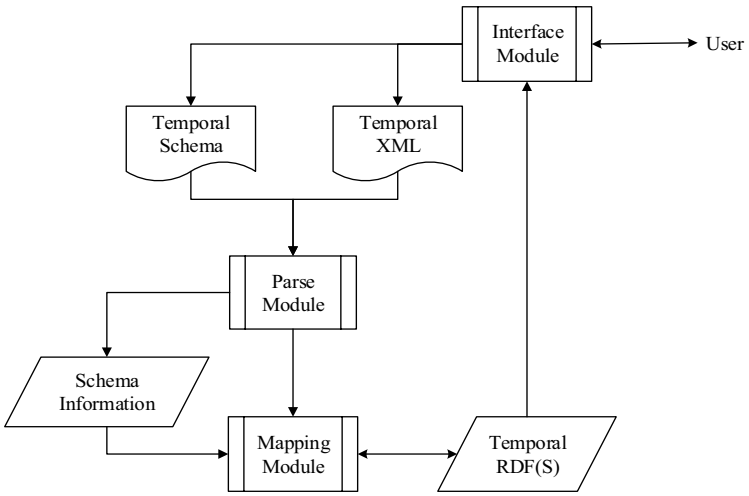

Figure 10. Architecture of *TX-TR*.

The core of *TX-TR* is to automatically map temporal XML into temporal RDF(S). Before the mapping is carried out, *TX-TR* needs to read in temporal XML Schema as its input, which is used to extract temporal RDFS. Then, temporal XML document needs to be read in and mapped, which is done to obtain temporal RDF triples. *TX-TR* displays temporal RDF triples as its output after the mapping is completed. The overall architecture of *TX-TR* is shown in Figure 10, which includes three main modules, namely, Parse module, Mapping module and Interface module.

1. Parse module. This module parses inputs of temporal XML Schema, temporal XML document and stores the parsed schema information. Schema information (e.g., filePath, temporal information, element constraints) of XML Schema can be extracted and represented.
2. Mapping module. This module transforms the parsed results into the corresponding temporal RDF(S) by using the mapping rules and algorithms proposed above.
3. Interface module. This module provides an interface to users, which mainly produces and finally displays the resulting RDF.

Implementation of *TX-TR* is based on Java with the API of dom4j and XPath over Eclipse platform, and the Graphical User Interface (GUI) is exploited over eclipse platform. *TX-TR* is implemented and run with a PC (CPU 2.5GHz, RAM 8GB and Windows 10 system).

The screen snapshot of *TX-TR* running one of the case studies is shown in Figure 11 and Figure 12. It can be seen from Figure 11 and Figure 12 that the graphical user interface of *TX-TR* contains two parts: "Temporal Schema → RDFS" and "Temporal XML document → RDF". Figure 10 shows the first part that maps temporal XML Schema into temporal RDFS, in which "Location" is used to import a file of temporal XML Schema and "Export RDFS" is used to export the corresponding temporal RDFS transformed the imported temporal XML Schema. Figure 11 shows the second part that maps temporal XML document into temporal RDF triples, in which "Location" is used to import a file of temporal XML document and "Export RDF" is used to export the corresponding temporal RDF transformed the imported temporal XML document.

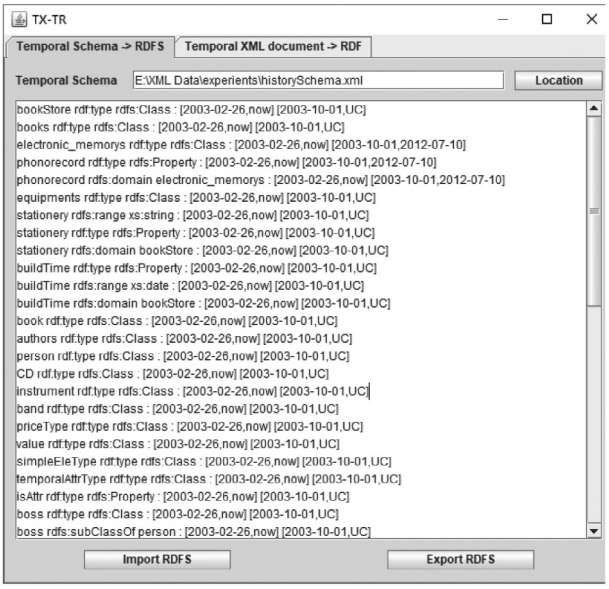

Figure 11. Screen snapshot of temporal RDF schema mapping.

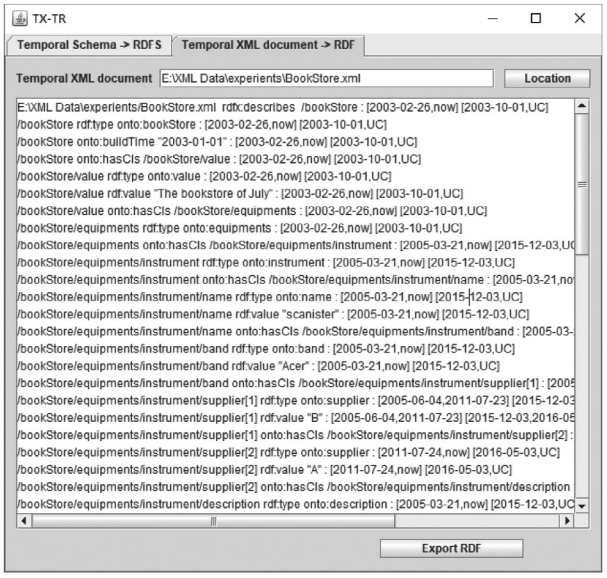

Figure 12. Screen snapshot of temporal RDF triple mapping.

6. Conclusions

In this paper, we present how to map temporal XML document with Schema into temporal RDF(S). The mapping is conducted at two levels. The first level of mapping is to map the temporal XML Schema into temporal RDF Schema and the second level of mapping is to map the temporal XML document into temporal RDF triples. For each level of mapping, we

propose the mapping rules and mapping algorithm. We illustrate our mapping approach with an example and implement a prototype system.

In our mapping approach, we apply the interval operations for time calculation. We also apply snapshots to describe the relation between theoretical models. Note that obtaining a snapshot is time-consuming. So, we propose some rules and constraints to ensure the effectiveness of extracting time information. In addition, in our mapping approach, we fully consider the semantics of the temporal XML and try to preserve the semantics in the mapping as much as possible. However, the cardinality constraint in XML cannot be directly represented by RDF(S). Also, the order indicators in XML Schema are mapped into the containers or collections of RDF in literature, but the containers and collections are only used to describe instances and cannot describe the inclusion relationship between classes of RDFS. We will dedicate ourselves to deal with their mapping in our future work.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China (61772269).

References

[1] F. Manola and E. Miller, "RDF Primer", *W3C Recommendation*, 2004. https://www.w3.org/TR/rdf-primer/

[2] J. F. Sequeda *et al*., "On Directly Mapping Relational Databases to RDF and OWL", in *Proc. World Wide Web Conference*, 2012, pp. 649–658 https://doi.org/10.1145/2187836.2187924

[3] Q. Tong *et al*., "Construction of RDF(S) from UML Class Diagrams", *Journal of Computing and Information Technology*, vol. 22, no. 4, pp. 237–250, 2014. https://doi.org/10.2498/cit.1002459

[4] T. Bray *et al*., "Extensible Markup Language (XML) 1.0", *W3C Recommendation*, 1998. http://www.w3.org/TR/1998/REC-xml-19980210

[5] M. Klein, "Interpreting XML Documents via an RDF Schema Ontology", in *Proc. International Workshop on Database and Expert Systems Applications*, 2002, pp. 889–894. https://doi.org/10.1109/DEXA.2002.1046008

[6] P. F. Patel-Schneider and A. Simeon, "The Yin/Yang Web: a Unified Model for XML Syntax and RDF Semantics", *IEEE Transactions on Knowledge & Data Engineering*, vol. 15, no. 4, pp. 797–812, 2003. https://doi.org/10.1109/TKDE.2003.1209000

[7] P. T. T. Thuy *et. al*., "Transforming Valid XML Documents into RDF via RDF Schema", in *Proc. International Conference on Next Generation Web Services Practices*, 2007, pp. 35–40. https://doi.org/10.1109/NWESP.2007.23

[8] P. T. T. Thuy *et. al*., "Exploiting XML Schema for Interpreting XML Documents as RDF", in *Proc. IEEE International Conference on Services Computing*, 2008, pp. 555–558. https://doi.org/10.1109/SCC.2008.93

[9] Y. Q. Yang *et al*., "An Automatic Semantic Extraction Algorithm for XML Document", in *Proc. International Conference on Machine Vision and Human-machine Interface*, 2010, pp. 41–44. https://doi.org/10.1109/MVHI.2010.82

[10] P. T. T. Thuy *et al*., "A Semantic Approach for Transforming XML Data into RDF Ontology", *Wireless Personal Communications*, vol. 73, no. 4, pp. 1387–1402, 2013. https://doi.org/10.1007/s11277-013-1256-z

[11] Y. An *et al*., "Constructing Complex Semantic Mappings between XML Data and Ontologies", in *Proc. International Semantic Web Conference*, 2005, pp. 6–20. https://doi.org/10.1007/11574620_4

[12] F. Grandi and F. Mandreoli, "The Valid Web: an XML/XSL Infrastructure for Temporal Management of Web Documents", in *Proc. Advances in Information Systems*, 2000, pp. 294–303. https://doi.org/10.1007/3-540-40888-6_28

[13] S. S. Chawathe *et al*., "Representing and Querying Changes in Semistructured Data", in *Proc. International Conference on Data Engineering*, 1998, pp. 4–13. https://doi.org/10.1109/ICDE.1998.655752

[14] M. Gergatsoulis and Y. Stavrakas, "Representing Changes in XML Documents using Dimensions", in *Proc. International XML Database Symposium*, 2003, pp. 208–222 https://doi.org/10.1007/978-3-540-39429-7_14

[15] T. Amagasa *et al*., "A Data Model for Temporal XML Documents", in *Proc. International Conference on Database and Expert Systems Applications*, 2000, pp. 334–344. https://doi.org/10.1007/3-540-44469-6_31

[16] F. Wang and C. Zaniolo, "XBiT: an XML-based Btemporal Data Model", in *Proc. International Conference on Conceptual Modeling*, 2004, pp. 810–824. https://doi.org/10.1007/978-3-540-30464-7_60

[17] F. Currim *et al*., "A Tale of Two Schemas: Creating a Temporal XML Schema from a Snapshot Schema with τXSchema", in *Proc. International Conference on Extending Database Technology*, 2004, pp. 348–365. https://doi.org/10.1007/978-3-540-24741-8_21

[18] F. Grandi *et al*., "Temporal Modelling and Management of Normative Documents in XML Format", *Data & Knowledge Engineering*, vol. 54, no. 3, pp. 327–354, 2005. https://doi.org/10.1016/j.datak.2004.11.002

[19] F. A. Currim *et. al*., "Adding Temporal Constraints to XML Schema", *IEEE Transactions on Knowledge & Data Engineering*, vol. 24, no. 8, pp. 1361–1377, 2012. https://doi.org/10.1109/TKDE.2011.74

[20] F. Rizzolo and A. A. Vaisman, "Temporal XML: Modeling, Indexing, and Query Processing", *The VLDB Journal*, vol. 17, no. 5, pp. 1179–1212, 2008. https://doi.org/10.1007/s00778-007-0058-x

[21] C. Gutierrez *et al*., "Temporal RDF", in *Proc. European Semantic Web Conference*, 2005, pp. 93–107. https://doi.org/10.1007/11431053_7

[22] C. Gutierrez *et al*., "Introducing Time into RDF", *IEEE Transactions on Knowledge & Data Engineering*, vol. 19, no. 2, pp. 207–218, 2007. https://doi.org/10.1109/TKDE.2007.34

[23] C. Hurtado and A. Vaisman, "Reasoning with Temporal Constraints in RDF", in *Proc. International Conference on Principles and Practice of Semantic Web Reasoning*, 2006, pp. 164–178. https://doi.org/10.1007/11853107_12

[24] J. Tappolet and A. Bernstein, "Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL", in *Proc. European Semantic Web Conference*, 2009, pp. 308–322. https://doi.org/10.1007/978-3-642-02121-3_25

[25] A. Pugliese *et al*., "Scaling RDF with Time", in *Proc. International Conference on World Wide Web*, 2008, pp. 605–614. https://doi.org/10.1145/1367497.1367579

[26] A. Tansel *et al*., "Temporal Databases: Theory, Design and Implementation", Benjamin/Cummings, 1993.

[27] P. Hayes, "RDF Semantics", *W3C Recommendation*, 2004. https://www.w3.org/TR/2004/REC-rdf-mt-20040210/

[28] R. Ma *et al*., "SPARQL Queries on RDF with Fuzzy Constraints and Preferences", *Journal of Intelligent and Fuzzy Systems*, vol. 30, no. 1, pp. 183–195, 2016. http://dx.doi.org/10.3233/IFS-151745

Received: February 2017
Revised: June 2018
Accepted: June 2018

Contact addresses:
Dan Yang
College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106
China
e-mail: july283629790@163.com

Li Yan*
College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing, 211106
China
e-mail: yanli@nuaa.edu.cn
*Corresponding author

Dan Yang is currently a master candidate in the College of Computer Science and Technology at the Nanjing University of Aeronautics and Astronautics, China. Her research interests include RDF and XML data management.

Li Yan received her PhD degree from the Northeastern University, China and is a full professor in the College of Computer Science and Technology at the Nanjing University of Aeronautics and Astronautics, China. Her current research interests include XML, RDF and the Semantic Web.