Computer Science Review 36 (2020) 100231

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Survey

XML data manipulation in conventional and temporal XML databases: A survey

Zouhaier Brahmia *, Hind Hamrouni, Rafik Bouaziz

Department of Computer Science, Faculty of Economics & Management, University of Sfax, Tunisia

ARTICLE INFO

Article history:
Received 30 June 2019
Received in revised form 25 November 2019
Accepted 5 February 2020
Available online xxxx

Keywords:
XML
XML database
NoSQL database
Temporal XML database
XML data manipulation
Data manipulation operation

ABSTRACT

After more than two decades of its use, XML is not only a standard format for exchanging data between different (Web) applications but also a model for a family of some emerging or NoSQL databases, called XML databases. In addition to its efficiency in the management of conventional data (i.e., data without a temporal reference), XML is also an excellent support for storing, manipulating, and querying temporal data, due its hierarchical structure. Besides, although several survey papers have dealt with multiple aspects concerning XML data, like XML data modeling, storage, indexing, and querying, there is no survey on the XML data manipulation (i.e., XML data insertion, deletion and modification) topic that continues interesting researchers of the database community, both in conventional and temporal XML databases. For that reason, we think that it is interesting to have a paper that reviews and compares research contributions dealing with this topic. So, our present paper (i) provides an overview of the state-of-the-art of XML data manipulation, in conventional and temporal XML databases, (ii) studies the support of such functionality in mainstream commercial DBMSs and (iii) gives some remarks on possible future research directions related to this issue.

Contents

1. Introduction

Since the second half of the 1990s, the eXtensible Markup Language (XML) [1] has been proposed by the World Wide Web Consortium (W3C) as a standard format for exchanging data between different applications running on the Web, like e-commerce, e-government, and e-health applications. Nowadays, XML is also a model of a new generation of non-standard databases, called XML

* Correspondence to: Road of the Aerodrome, Km 4.5, P.O. Box 1088, 3018 Sfax, Tunisia.
E-mail addresses: zouhaier.brahmia@fsegs.rnu.tn (Z. Brahmia), hend.hamrouni@fsegs.rnu.tn (H. Hamrouni), rafik.bouaziz@usf.tn (R. Bouaziz).

https://doi.org/10.1016/j.cosrev.2020.100231
1574-0137/

databases [2–6]. Notice also that these databases are currently considered also as an example of NoSQL databases[1] [7–10], and precisely document-oriented ones.

Furthermore, since many of XML-based applications require keeping a full history of data evolution over time (which could be very useful e.g. for decision making), temporal XML databases [11–15] were proposed to respond to such a requirement. Notice that XML provides an excellent support for temporally grouped data models [16], which are considered as the most natural and effective representations of temporal information [17]. Notice also that a temporal XML database uses the valid time dimension and/or the transaction time dimension to manage the history of XML data, while valid time denotes the time when data are valid in the real world and transaction time denotes the time when data are current in the database. Hence, temporal XML databases could store valid-time XML data, transaction-time XML data, or bitemporal XML data.

In the XML database field, there are some functionalities that are fundamental, like XML data modeling, storage, manipulation (i.e., insertion, deletion, and modification), indexing, and querying. Several research works have been done on each one of these functionalities. Moreover, in the literature, there are multiple surveys dealing with some of XML aspects, like the following ones:

- [18,19], on XML data querying;
- [20,21], on conceptual modeling for XML;
- [22], on data storage and query processing in XML databases;
- [23], on XML data clustering;
- [24], on XML streaming;
- [25], on temporal and multi-versioned XML documents;
- [26], on transforming XML documents to OWL ontologies.

However, to the best of our knowledge, there is no survey on XML data manipulation, despite its importance. In fact, "manipulation of XML data", also called "XML data update", is continuing to be an interesting topic for researchers of the database community, both in conventional (i.e., non-temporal) XML databases [27–29] and temporal XML databases [30,31].

For all these reasons, we think that it is very important to fill this gap and to propose a survey on XML data manipulation, which will be helpful for forthcoming research work dealing with this issue. Thus, the main purpose of this survey is to present the state-of-the-art approaches for manipulating XML data in both conventional and temporal XML databases.

The rest of this paper is structured as follows. Section 2 gives some background on the considered topic. Section 3 presents the different research proposals dealing with XML data manipulation. Section 4 studies the support provided by existing DataBase Management Systems (DBMSs) for manipulating XML data. Section 5 provides future research directions. Finally, Section 6 summarizes this paper.

2. Background

In this section, first we define conventional XML databases and use an example to illustrate the manipulation of conventional XML data. Then, we deal with temporal XML databases and manipulation of temporal XML data.

[1] http://www.nosql-database.org/

2.1. Conventional XML databases and manipulation of conventional XML data

A conventional XML database consists of a set of XML documents/files, each one could be valid to an XML schema file. This latter is in general a Document Type Definition (DTD) [1] or an XML Schema Definition (XSD) [32] file that describes the structure of such an XML file.

Fig. 1 provides an example of a simple conventional XML database used by a company to manage the information of its employees: it includes (a) an XML file, named empData.xml, which stores data about one employee, called "Jamil" and earning 5000 TND per month, and (b) an XSD file, named empSchema.xsd, which describes the structure of this XML file; each employee is characterized by a name (of string type) and a salary (of float type).

Since the W3C has proposed a standard language for manipulating XML data, that is the XQuery Update Facility (XUF) language [33], we use it in the illustration of the XML data manipulation issue.

First, let us assume that the company would like to (i) add the information of a newly hired employee, named "Maria", with an initial salary of 5500 TND, and (ii) raise the Jamil's salary by 20%. These requirements could be satisfied through the execution of the two following XUF statements (insert and replace):

```
insert node
     <employee>
          <name>Maria</name>
          <salary>5500</salary>
     </employee>
as last into fn:doc("empData.xml")/employees

replace value of node doc("empData.xml")/employees/
                      employee[name="Jamil"]/salary
with fn:doc("empData.xml")/employees/
            employee[name="Jamil"]/salary * 1.2
```

Thus, the new state of the `empData.xml` file can be represented as displayed in Fig. 2; the `empSchema.xsd` file has not been modified.

After that, assume that we need removing the employee "Jamil" from the database since he has resigned. Therefore, the following XUF statement could be used to express such a change in the real world:

```
delete node fn:doc("empData.xml")/employees/employee
[name="Jamil"]
```

Consequently, the new state of the `empData.xml` file can be represented as displayed in Fig. 3; the corresponding XSD file has not been changed.

Notice that, in general, XML applications and XML DBMSs use XSD files to guarantee that the initial structure of each XML file has not been changed. In fact, an XML file that is declared as valid to an XSD file should remain conformant with respect to this latter, after the execution of each sequence of XML data manipulation operations (or sequence of XUF statements).

2.2. Temporal XML databases and manipulation of temporal XML data

A temporal XML database [11,13,14,30,34–36] consists of a set of temporal XML documents/files, each one could be valid to a temporal XML schema file. A temporal XML document is an XML document that stores temporal XML elements, i.e., valid-time, transaction-time or bitemporal XML elements.

```
<?xml  version="1.0"  encoding="UTF-8"?>
<employees  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:=noNamespaceSchemaLocation="empSchema.xsd">
  <employee>
    <name>Jamil</name>
    <salary>5000</salary>
  </employee>
</employees>
```

(a) The `empData.xml` file

```
<?xml  version="1.0"  encoding="UTF-8"?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="employees">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="employee" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="name" type=" xsd:string" />
       <xsd:element name="salary" type="xsd:float" />
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

(b) The `empSchema.xsd` file

Fig. 1. The initial state of the employee XML database.

```
<?xml  version="1.0"  encoding="UTF-8"?>
<employees  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:=noNamespaceSchemaLocation="empSchema.xsd">
  <employee>
    <name>Jamil</name>
    <salary>6000</salary>
  </employee>
  <employee>
    <name>Maria</name>
    <salary>5500</salary>
  </employee>
</employees>
```

Fig. 2. The state of the `empData.xml` file after inserting the employee "Maria" and raising the salary of the employee "Jamil".

```
<?xml  version="1.0"  encoding="UTF-8"?>
<employees  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:=noNamespaceSchemaLocation="empSchema.xsd">
  <employee>
    <name>Maria</name>
    <salary>5500</salary>
  </employee>
</employees>
```

Fig. 3. The state of the `empData.xml` file after deleting the details of the employee "Jamil".

- A transaction-time XML element has two timestamping attributes that denote the boundaries of a transaction time interval: TST (transaction start time) and TET (transaction end time).
- A valid-time XML element has two timestamping attributes: VST (validity start time) and VET (validity end time).
- A bitemporal XML element has four timestamping attributes: VST, VET, TST, and TET.

Notice that these timestamping attributes are automatically managed by the DBMS, transparently to the user.

Moreover, the evolution of a temporal XML element is grouped and represented in a complex XML element that contains subelements, each one representing a (temporal) version of that temporal XML element, as illustrated by Fig. 4 that shows an example of a temporal XML document storing information of the employees of the same company mentioned above. In this document, only the "salary" is temporal, precisely of valid-time format. Consequently, each <salary/> element, representing a temporal version of an employee's salary, is defined as a subelement of a <salaries/> complex element.

Notice that (i) the attribute "NUM_V" denotes the "number of a version" and identifies a timestamped version <element/> of a temporal "element"; it makes easy the access and manipulation of such a version; (ii) the VET attribute can have the special value "now" [37,38], which means that such a temporal version of an element continues to be valid in the real world until some change occurs.

Within this temporal setting, let us assume that on 2019-01-16, the company wants to (i) add an old forgotten salary (5500 TND) for the employee Jamil, valid from 2013-06-01 to 2013-12-31, (ii) update the current salary (i.e., that is valid at the current date: 2019-01-16) of the same employee (Jamil) with a new salary (7000 TND), valid from 2019-01-01 (i.e., with retroactive effect), and (iii) delete the current salary of the employee Maria, valid from 2019-02-01. These temporal requirements could be satisfied through the execution of the three following temporal XML update statements (insert, replace and delete) that are inspired by the statements of the TempoXUF (Temporal XUF) language proposed in [34,36]:

```
insert node <salary>5500</salary>
into fn:doc("empTemporalData.xml")/employees/
            employee[name="Jamil"]/salaries
as valid from '2013-06-01' to '2013-12-31'

replace content of node fn:doc("empTemporalData.xml")/
      employees/employee[name="Jamil"]/salaries/salary
                  valid at '2019-01-16'
with 7000
as valid from '2019-01-01'

delete node fn:doc("empTemporalData.xml")/employees/
             employee[name="Maria"]/salaries/salary
            valid at '2019-01-16'
as valid from '2019-02-01'
```

Thus, the new state of the empTemporalData.xml file can be represented as displayed in Fig. 5; the empTemporalSchema.xsd file has not been modified.

3. XML data manipulation in a conventional XML setting

In this section, we survey contributions which have dealt with XML data manipulation in a conventional setting. Indeed, first we present the research works concerning such a topic. Then, we study the support provided by existing DBMSs for manipulating conventional XML data.

3.1. State-of-the-art of manipulation of conventional XML data

In this subsection, first we present research contributions concerning conventional XML data manipulation. Then, we summarize and compare them.

3.1.1. Research proposals

Many works have dealt with manipulating XML data in a conventional XML setting, like [39–51].

Tatarinov et al. [52] present a set of seven basic operations for updating data stored in an XML document: Delete(child), Rename(child, name), Insert(content), InsertBefore(ref, content), InsertAfter(ref, content), Replace(child, content), and SubUpdate(patternMatch, predicates, updateOp). An XML update is then defined as a sequence of these primitive operations. After that, the authors propose extensions to the XQuery language [53], which is devoted to querying XML documents, in order to support also XML data updates. Next, the authors define strategies for implementing update operations in an environment where XML data are mapped into relational data stored and managed by a relational DBMS. Eventually, they provide an experimental evaluation of these update strategies.

Wong [51] extends XQL (XML Query Language) [54] to support updating XML data. The extended XQL includes five update constructs (insert, update, copy, move, and delete). It is implemented within the native XML DBMS SODA2 (Semistructured Object DAtabase system, version 2).

Klettke et al. [44] propose a set of four basic update operations (DELETE child, INSERT content [(Before| After) ref], RENAME child TO name, and REPLACE child WITH content) and define the effects of each one of them on the involved XML document and possibly on its XML schema. Moreover, since some XML update operations are not validity-preserving (i.e., when applying these operations on an XML document, this latter becomes invalid with respect to its schema), the authors introduce four approaches (IGNORE, REJECT, REDO, and EVOLVE) for processing these operations. (i) The IGNORE approach accepts all XML update operations, without checking the validity of the updated document with regard to its schema. It is supported by the native XML DBMS Extensible Information Server from the eXcelon Corporation (later bought by Sonic Software). (ii) The REJECT approach refuses update operations that produce an updated XML document which is not valid to its schema. It is implemented by Tamino from the Software AG Corporation. (iii) The REDO approach handles XML update operations that invalidate the XML schema as follows: (a) first, the update operation, which is specified by the user on an XML document, is rejected since it violates the schema of such a document; (b) then, the designer adapts the XML schema manually; (c) next, the designer also adapts all other XML documents, which are associated to the initial schema except the document that will be updated, so that they become valid with respect to the adapted schema; (d) finally, the end user redoes the update operation, which should normally be executed without problems. (iv) The EVOLVE approach supports all update operations, while forcing schema evolution if necessary. Indeed, when an update operation violates some constraints specified in the schema, the XML document is updated and its schema is automatically adapted without any intervention of the user or the designer.

Pardede et al. [47] propose a methodology which allows updating XML documents without violating the conceptual constraints of these documents. This methodology consists in a set of functions, which check before performing update operations and ensure that these operations preserve the conceptual constraints of the updated XML documents. The constraints are embedded in three structural relationships: (i) association, which involves constraints like number of participant type and referential integrity,

```
<?xml  version="1.0"  encoding="UTF-8"?>
<employees  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:=noNamespaceSchemaLocation="empTemporalSchema.xsd">
  <employee>
    <name>Jamil</name>
    <salaries>
      <salary Num_V="1" VST="2014-01-01" VET="2016-12-31">6000</salary>
      <salary Num_V="2" VST="2017-01-01" VET="now">6500</salary>
    </salaries>
  </employee>
  <employee>
    <name>Maria</name>
    <salaries>
      <salary Num_V="1" VST="2015-01-01" VET="2017-12-31">5500</salary>
      <salary Num_V="2" VST="2018-01-01" VET="now">6000</salary>
    </salaries>
  </employee>
</employees>
```

(a) The `empTemporalData.xml` file

```
<?xml  version="1.0"  encoding="UTF-8"?>
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="employees">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="employee" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="name" type=" xsd:string" />
       <xsd:element name="salaries">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="salary" maxOccurs="unbounded">
           <xsd:complexType>
            <xsd:simpleContent>
             <xsd:extension base="xsd:float">
              <xsd:attribute name="Num_V" type="xsd:integer"
                                          use="required"/>
              <xsd:attribute name="VST" type="xsd:date"
                                        use="required"/>
              <xsd:attribute name="VET" type="xsd:string"
                                        use="required"/>
             </xsd:extension>
            </xsd:simpleContent>
           </xsd:complexType>
          </xsd:element>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

(b) The `empTemporalSchema.xsd` file

Fig. 4. An example of a temporal XML database for the employees of a company.

```
<?xml version="1.0" encoding="UTF-8"?>
<employees xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:=noNamespaceSchemaLocation="empTemporalSchema.xsd">
  <employee>
    <name>Jamil</name>
    <salaries>
      <salary Num_V="1" VST="2013-06-01" VET="2013-12-31">5500</salary>
      <salary Num_V="2" VST="2014-01-01" VET="2016-12-31">6000</salary>
      <salary Num_V="3" VST="2017-01-01" VET="2018-12-31">6500</salary>
      <salary Num_V="4" VST="2019-01-01" VET="now">7000</salary>
    </salaries>
  </employee>
  <employee>
    <name>Maria</name>
    <salaries>
      <salary Num_V="1" VST="2015-01-01" VET="2017-12-31">5500</salary>
      <salary Num_V="2" VST="2018-01-01" VET="2019-01-31">6000</salary>
    </salaries>
  </employee>
</employees>
```

Fig. 5. The state of the `empTemporalData.xml` file on 2019-01-16, after executing the required temporal data manipulation operations.

(ii) aggregation, which involves constraints like ordering and homogeneity, and (iii) inheritance relationship, which involves only two constraints: the exclusive disjunction and the number of ancestors.

Wang and Zhang [50] propose XML-RL update language, a declarative XML update language that extends the XML-RL query language [55]. It is characterized by the following features: (i) it is based on a high-level data model; (ii) it allows specifying, in a flat way, complex update orders involving multiple levels in a document hierarchy; (iii) it supports updating complex objects; (iv) it provides a unified syntax for modifying tag names, values, and objects, by using three types of logical binding variables: name variables, value variables, and object variables.

In [41], the authors propose an approach which allows updating XML documents without respecting the constraint that the updated document must be (after the update) valid to its DTD. In fact, in such a case, the authors propose also to update the corresponding DTD so that it could describe the new structure of the modified XML document (it becomes a valid schema for this document).

Ni and Ling [45] propose GLASS^{U}, a graphical XML update language that extends the graphical XML query language GLASS (Graphical query Language for Semi-Structured data) [56]. GLASS^{U} requires that the user, who specifies graphically the XML update order, should know both schema and semantics of XML data. Notice that the considered XML schema model is ORA-SS (Object-Relationship-Attribute model for Semi-Structured data) [57]. A GLASS^{U} update expression is a graph that is constituted of three parts: the condition part, the view part, and the action part. Only the action part is required; the two other parts are optional. Furthermore, the condition and view parts are separated by a single vertical line; however, the view and action parts are separated by a double vertical line.

Dargham et al. [42] propose an approach for updating XML documents while preserving their validity to their XML schemas (defined as DTDs). This approach (i) is based on XUpdate, an XML update language defined by the authors, which extends the XQuery language with four update operators, i.e., insertAfter, insertBefore, delete, and modify, (ii) uses the object-oriented DBMS F2 [58] for storing XML documents, (iii) translates XUpdate orders, specified by the users, into update OQL (Object Query Language) [59] statements to be executed by the DBMS, and (iv) adopts a validation method that is both partial (i.e., it validates only the updated part of the XML document) and immediate (i.e., the validation is performed before executing the update, and the document is updated only if it will be valid to its DTD after the update operation).

Pardede et al. [48] focus on updating XML documents stored in XML-enabled databases [4], while maintaining the semantics of these documents. The authors start by defining the set of XML conceptual constraints that can be specified on XML data. After that, they transform these conceptual constraints into a logical model represented as a schema specified in the SQL/XML standard [60]. Next, the authors use this schema to propose generic XML update methods (or algorithms) that preserve all identified XML constraints after any update operation (i.e., insertion, deletion, or replacement) performed on XML data. Finally, they implement the proposed update methods in one of the mainstream XML-enabled databases (Oracle 10g).

O'Connor and Roantree [46] define a set of ten desirable properties for dynamic labeling and encoding of XML documents to support efficient updating mechanisms. These properties are as follows: Document order, Encoding representation, Persistent labels, XPath evaluations, Level encoding, Overflow problem, Orthogonal labeling scheme, Compact encoding, Division computation, and Recursive labeling algorithm.

Baazizi et al. [39] develop a DTD-based optimization technique that allows reducing memory consumption of in-memory engines when processing main memory XML updates. Indeed, given an XML update specified on an XML document which is valid to DTD, the main idea consists in statically determining which parts of this document are involved by such an update. This technique is very useful specifically for updating very large XML documents (due to their big sizes).

Recently, Bidoit et al. [40] deal with management of Big XML documents and propose a system, named Andromeda, for processing updates and also queries on very large XML documents. As for updates, they are specified as XQuery Update Facility expressions. The technique implemented by Andromeda consists in

partitioning statically and/or dynamically the Big document to be updated, in order to distribute the update load on the machines of a Map/Reduce cluster.

Kircher et al. [43] propose an optimization technique, named structural bulk updates, which reduces the processing time of XML updates to the Pre/Dist/Size XML encoding [61].

Besides, Cavalieri et al. [62] propose PUL-Diff, an approximated diff algorithm that allows (i) comparing a source and a target XML documents, and (ii) producing the difference between them as XQuery Update Facility Pending Update Lists (PULs) which are unordered lists of atomic update expressions. These PULs can be applied to the source document in order to transform it into the target one.

Several conventional XML update languages were also proposed, like XUpdate [63], SiXDML [64], UpdateX [52], Flux [65], and XQuery Update Facility [33].

Kane et al. [66], Bouchou and Alves [67], Balmin et al. [68], and Barbosa et al. [69,70] study incremental validation of updates to XML documents. The incremental validation consists in checking only the validity of the portion of the document, which is directly concerned by the update. Thus, an XML update is effectively executed on a valid XML document only when the validity of the document which results from such an update is guaranteed. Notice that this technique is very useful in contexts where XML updates are frequent (e.g., collaborative editing environments) and/or XML documents are very large (e.g., bibliographic databases, genome databases).

Updates to XML documents are also present in environments which support XML schema evolution. In fact, when some changes are applied to an XML schema, XML documents that are valid to the original schema must be (incrementally) revalidated against the new schema, and, if they are no longer valid, adapted to this new schema, by updating them through a sequence of XML update operations; such a sequence is either provided by the designer or automatically generated by the schema evolution system. The adaptation of XML documents under schema evolution has been widely studied by Giovanna Guerrini's team, e.g., in [71,72], and [73]. Besides, Guerrini et al. [74] deal with incremental validation of XML documents that are being adapted, under schema evolution. Cavalieri et al. [75] propose a technique for reducing sequences of XML update operations that are applied to XML documents during their adaptation to the evolved XML schema. Notice here that, like Bouchou and Duarte [76], Cavalieri et al. [75] allow updates on an XML document to have an effect on the corresponding XML schema. Indeed, when an XML document becomes, after an update, not valid to its schema, the approach proposed by Cavalieri et al. [75] consists in updating this schema so that it describes the structure of the updated document.

3.1.2. Summary

In Table 1, we summarize the main approaches proposed by the research community for XML data manipulation in conventional XML databases, while considering the following six criteria:

- Data manipulation operations: the set of operations that are proposed (insert, update, delete, …).
- Data model: the model of XML data that are manipulated and on which the manipulation operations are effected (XML Query data model, XQL model, …).
- XML schema language: the language through which the structure of XML documents, that store XML data, is defined (DTD, XML Schema, …).
- Impact on XML schema: the execution of data manipulation operations has an impact on the XML schema of the updated XML document? (Yes, No).
- Implementation: the proposal has been implemented? (Yes, No).

Notice that:

- the granularity of each data manipulation operation, in all presented proposals, is "XML element";
- the value "ns" (in the fourth column of Table 1) means that the XML schema language has not been specified;
- the value "nc" (in the fifth column of Table 1) means that the impact of data manipulation operations on XML schema has not been considered/studied.

3.2. State of the practice: DBMS support for manipulation of conventional XML data

As for existing relational or native XML DBMSs, they provide a limited support for manipulating conventional XML data.

Relational DBMSs, like Microsoft SQL Server 2014 [78] or Microsoft SQL Server 2008 [79], IBM DB2 version 10 [80], version 9.5 [81] or version 8.1 [82], and Oracle 12c [83,84] or Oracle 10g [85], allow managing XML data, by storing, querying and processing XML documents as tables. Indeed, Microsoft SQL Server supports operations to manipulate XML data with relational engine. IBM DB2 supports XQuery Update Facility language. Oracle provides a set of functions, e.g., insertXML(), updateXML() and deleteXML(), to manipulate XML data; these functions are extensions of SQL/XML [86] functions.

Native XML DBMSs, like xDB, eXist [87], Timber [6], and Sedna [88,89], store, query and process XML data in their native type. In order to manipulate data, xDB supports XQuery Update Facility; eXist is based on the XUpdate language [63]; Timber uses XML Application Programming Interface (API); Sedna supports a declarative node-level update language, which is based on the XQuery update proposal of Lehti [90] with the number of improvements [91].

4. XML data manipulation in a temporal XML setting

Temporal data management has been studied deeply in the context of temporal relational databases (e.g., [92–95]) and temporal object-oriented databases (e.g., [96–98]). In the XML setting, only few works have dealt with this issue.

In this section, first we survey research contributions which have dealt with manipulation of temporal XML data. Then, we summarize and compare them.

4.1. Research proposals

Faisal and Sarwar [25] provide a survey on temporal and multi-versioned XML documents but do not study how temporal XML data are inserted, modified, and deleted. They deal with aspects like change detection, querying multi-versioned XML documents, and storage structures.

Nørvåg et al. [99] adopt a stratum approach in order to build a temporal XML DBMS, Nørvåg [100] uses an integrated approach to achieve the same purpose (i.e., modifying or extending the internal modules of the DBMS to support temporal data). The author proposes V2, a native temporal document DBMS which allows storing and searching temporal data versions, retrieving documents that were valid at a certain time, and querying changes to documents, and manipulating (i.e., inserting, updating, and deleting) documents. Nørvåg [100] deals with temporal documents under a single schema version (there is no support of schema versions) and considers the whole document as the granularity of any data management operation (i.e., insert, update, delete).

Table 1
Summary of conventional XML data manipulation research proposals.

	Data manipulation operations	Data model	XML schema language(s)	Impact on XML schema	Implementation
[49]	– Delete(child) – Rename(child, name) – Insert(content) – InsertBefore(ref, content) – InsertAfter(ref, content) – Replace(child, content) – SubUpdate(patternMatch, predicates, updateOp)	XML Query data model	DTD	No	Yes
[51]	– insert – update – copy – move – delete	XQL [54] model	DTD, XML Schema	Yes	Yes
[44]	–DELETE child –INSERT content [(Before|After) ref] – RENAME child TO name – REPLACE child WITH content	XML Query data model	DTD, XML Schema	No	No
[41]	– Insertion of an attribute – Insertion of an element – Delete operation	Unranked labeled tree	DTD	Yes	Yes
[42]	– insertAfter – insertBefore – delete – modify	Object-oriented model	DTD	No	No
[39]	XQuery Update Facility operations	XML store [77]	DTD	No	Yes
[40]	XQuery Update Facility operations	XQuery and XPath Data Model (XDM)	ns	nc	Yes
[43]	XQuery Update Facility operations	Pre/Dist/Size XML encoding [61]	ns	nc	Yes
[62]	XQuery Update Facility Pending Update Lists (PULs)	XQuery and XPath Data Model (XDM)	ns	nc	Yes

Grandi [101] has introduced an excellent annotated bibliography dealing with temporal and evolution aspects in the World Wide Web. But, he has not focused on temporal XML data manipulation.

Wang and Zaniolo [102] propose an approach which represents and publishes the contents of transaction-time, valid-time and bi-temporal relational databases as XML documents, using temporally-grouped data models [17]. Such an approach supports (i) complex temporal queries, which are difficult to be expressed in SQL on relational tables, written in XQuery, and (ii) temporal data insertions, deletions, and modifications, through user-defined functions.

Grandi et al. [103] deal with the dynamics of norms (e.g., laws, acts, decrees, provisions, regulations) in time and propose a model to (i) capture the semantics of norm texts evolving over time, and (ii) represent their multiple versions with regard to four temporal dimensions: publication, validity, efficacy, and transaction times. Multi-version norms are represented as temporal versioned XML documents. Modifications on such norms are performed through two basic operators which implement lossless changes: the changeText operator for textual modifications that can occur at any level of the hierarchical structure of XML documents, and the changeTime operator for the extension of the temporal pertinence of an existing document portion (i.e., a portion of a norm). Grandi et al. [103] deal with temporal XML data management while the XML schema is static (one XML Schema file for all norms); a data update operation involves always one XML document. Furthermore, their approach is domain-dependent; it is specific to the legal domain and cannot be adopted in all contexts.

In [16], the authors propose briefly deal with updating XML documents that store valid-time data coming from a valid-time relational database (each document represents the history of a distinct valid-time table). More precisely, they define three primitives (Insert, Update, and Delete) for manipulating valid-time XML data in a single-schema context.

Rizzolo and Vaisman [104] deal with updates acting on a transaction-time XML document. They propose three data management operations: insertion of a new node, deletion (in the sense of transaction-time databases) of an existing node, and update of containment edges. For each one of these operations, the authors show how it is performed in the involved document. Furthermore, since only transaction-time data are considered, all operations are executed at the current time: only on-time updates are allowed.

Brahmia and Bouaziz [11] propose an approach for manipulating temporal XML data in an XML database which supports temporal schema versioning. The authors define a set of four basic temporal XML data management operations: three operations act on an XML element (i.e., addition, deletion, and modification of the content of an XML element) and one operation acts on an attribute of an XML element (i.e., modification of the value of an XML element's attribute).

In the field of temporal XML databases, τXSchema (Temporal XML Schema) [13,15,105] is a very well known and complete framework (i.e., a data model, a language, and a suite of tools) for creating and validating temporal XML data. These latter are generated from conventional XML data, by applying a set of logical and physical annotations on them. These annotations are specified on the conventional schema that describes the non-temporal structure of XML data.

- Logical annotations specify (i) whether an element or an attribute varies over valid time and/or transaction time,

(ii) whether its lifetime is described as a continuous state or a single event, (iii) whether the item itself may appear at certain times (and not at others), and (iv) whether its content changes.

- Physical annotations specify the timestamp representation options chosen by the designer, such as where the timestamps are placed and their kind (e.g., valid time or transaction time) and the kind of representation adopted. The location of timestamps is largely independent of which components vary over time. Timestamps can be located either on time-varying components (as specified by the logical annotations) or somewhere above such components. Two documents with the same logical information will look very different if we change the location of their physical timestamps. Changing an aspect of even one timestamp can make a big difference in the representation.

A temporal schema is created by combining a conventional schema and a set of logical and physical annotations corresponding to it. τXSchema has also been extended in [12,106] to support schema versioning [107,108]. However, such a framework does not provide a language for manipulating temporal XML data. To fill this gap, Brahmia et al. [30] have proposed a temporal extension of the W3C XUF language, named τXUF (Temporal XUF), to manipulate temporal XML data in the τXSchema framework. It extends XUF statements with a valid clause to specify the applicability period of the data manipulation operation. If we ignore its valid clause, a τXUF statement is identical to the corresponding XUF statement to be executed in a conventional context. Indeed, while taking advantage of the τXSchema logical and physical independence aspect, the authors aim at helping the programmer/user, by allowing him/her to focus only on the XML data structure, as defined in the conventional schema, and to ignore how XML data are structured in the temporal schema. Therefore, implementation aspects and their transparent management are left to the system. Moreover, τXUF has two other advantages: (i) A τXUF statement could be specified either on the temporal document or on its squashed version, and the system can correctly manage both methods, through the use of logical and physical annotations. (ii) τXUF supports a new revalidation declaration option, named "force", to be used by the τUpd:applyUpdates procedure for executing XML data updates that require implicit schema changes (i.e., the system transparently performs implicit schema changes, before executing data updates).

Hamrouni et al. [31,109,110] deal with performing retroactive updates to valid-time or bitemporal XML databases and managing their effects on the consistency of such databases; notice here that retroactive update is an update whose applicability period begins before the execution time of the update. Indeed, Hamrouni et al. [109] propose an approach that allows a temporal XML DBMS detecting data inconsistencies that result from retroactive updates and automatically and transparently repairing them. Hamrouni et al. [110] extends the proposal of Hamrouni et al. [109] with support of temporal XML schema versioning: they provide an approach for automatically detecting and repairing inconsistencies which happen due to retroactive updates, in the presence of multiple schema versions. In [31], the authors extend their approach described in [110] by (i) presenting in a detailed way the architecture of the proposed approach, (ii) providing a technique for extracting data dependencies from temporal XML update orders, recording them in the database, and using them to determine all data that are dependent on a retroactively updated datum and that should be consequently updated, (iii) defining a new structure for the transaction log which saves all transaction details that are useful for re-executing transactions in order to repair the detected data inconsistencies, and (iv) proposing a tool prototype which shows the feasibility of the proposed approach.

Dyreson [111] has dealt with a very special operation, named data vacuuming [112–114], in the context of a transaction-time XML database. Such an operation specifies which XML data versions to vacuum (or to physically be removed), in order to reclaim space and prevent data from exceeding storage capacity. The author has proposed two vacuuming policies: a micro policy and a macro one. (i) A micro vacuuming policy is applied on a single element type (e.g., vacuum versions of the <price> element that are more than 10 years old). (ii) A macro vacuuming policy is applied to the entire transaction-time XML database (e.g., the size of the database must not exceed 20 GB).

Notice that in the literature of temporal XML databases, there are four well known temporal XML query languages that have been proposed for only querying temporal XML instances in an environment which does not support schema versioning: (i) TTXPath [115], which is a temporal extension of XPath [116] and supports only transaction-time, (ii) τXQuery [117], which extends XQuery [53] to support both valid-time and transaction-time, (iii) ParaSQL [118], which is a user-friendly SQL-like language for querying time-varying parametric XML data, and (iv) TXPath [104], which is an extension of XPath and supports both valid-time and transaction-time.

Furthermore, to the best of our knowledge, existing native XML DBMSs (like eXist, xDB, and Sedna) and commercial DBMSs that support XML (like Oracle 12c, and DB2 ver. 10) do not provide any support for manipulating (inserting, deleting, and modifying) temporal XML data.

4.2. Summary

In Table 2, we summarize the main approaches proposed by the research community for the manipulation of temporal XML data, while considering the following six criteria:

- Under schema versioning aspect: the temporal data manipulation is performed in an environment that supports XML schema versioning? (Yes, No);
- Supported time dimension(s): the evolution of manipulated data is maintained along transaction time dimension and/or along valid time dimension;
- Implementation: is there an implementation of the proposal of the authors? (Yes, No);
- Impact on XML schema: the execution of manipulation operations has an impact on the XML schema of the manipulated XML data? (Yes, No, nc: not considered/studied);
- Granularity of data manipulation operations: XML element or XML document?

5. Future research directions

From our study, we have noticed that whereas a lot of research work has been done on manipulation of conventional XML data, temporal XML data manipulation has been considered only to a limited extent. Thus, a lot of work needs to be done in this area. In the following, we present several research issues that are related to the conventional and temporal XML data manipulation topic.

(a) Although a W3C language (i.e., XQuery Update Facility) has been recommended for updating conventional XML data, and implemented by several DBMSs and XML tools, there is yet no consensual/recommended language for updating time-varying XML data. Therefore, it would be interesting to propose such a language, e.g., as a TSQL2-like one. Notice that TSQL2 [95] is a consensual language designed for temporal relational databases.

Table 2
Summary of temporal XML data manipulation research proposals.

	Under schema versioning	Supported time dimension(s)		Implementation	Impact on XML schema	Granularity of data manipulation operations	
		Transaction time	Valid time			XML element	XML document
[100]	No	✔		Yes	nc		✔
[119]	No	✔	✔	No	nc	✔	
[103]	No	✔	✔	Yes	No	✔	
[16]	No	✔		Yes	No	✔	
[104]	No	✔		Yes	nc	✔	
[11]	Yes	✔	✔	Yes	No	✔	
[30]	Yes		✔	No	Yes	✔	

(b) Even though mainstream (XML) DBMSs provide a significant support for manipulating conventional XML data, there is no available support for manipulating time-varying XML data. Consequently, application programmers and database administrators proceed in an ad hoc manner when they have to deal with temporal XML data. Thus, it is interesting to develop tools for such a manipulation, in order to facilitate the task of these users.

(c) Manipulated data are obviously queried by applications and final users. However, whereas a great deal of work has been done on querying conventional XML data, only few research have dealt with queries to time-varying XML data [13,14,104]. So, in order to complete the figure, we think that a thorough investigation of temporal XML data querying [13,14,104] has to be made.

(d) Before manipulating (i.e., inserting, updating, or deleting) or querying XML data, whether they are conventional or temporal, the XML database that contains such data should be physically created under an XML DBMS. In general, the creation of such a database is actually performed after conceptually and logically modeling the corresponding database, according to a data modeling method (e.g., eXolution [120]) or language (e.g., UML, XUML [121], or UXS [122]), while using a conceptual/logical modeling tool (e.g., Altova XMLSpy). Thus, we think that conceptual and logical modeling of conventional [121,123] and temporal [104,124,125] XML data is a topic that should be well studied.

(e) Similarly to conceptual and logical aspects of XML data, we think also that engineering and physical aspects of conventional [126] and temporal [127] XML data (e.g., indexing, storage, query processing) require more attention by researchers as they have not been well studied in the current literature.

(f) In XML databases, not only XML data are evolving over time but also their XML schemas (which describe XML data structures). In fact, XML schema changes are unavoidable during the lifecycle of an XML database, in order to reflect changes in the real world or in the applications' requirements. Therefore, we think that studying changes to conventional [120,128] and temporal [12,15,106] XML schemas is an interesting issue, as an extension, to the schema level, of the work done in this survey at instance (or data) level. Notice that, in the database literature, there are two techniques already proposed for managing schema changes: (i) schema evolution [129–131], which keeps only the last schema version and adapts previous data to the new schema, after each schema change, and (ii) schema versioning [107,108,132], which keeps all schema versions with their corresponding data.

(g) Management of conventional and temporal big data [133,134] and, in general, temporal NoSQL data [135–141], as they are widely used in today's applications (online social networks, Internet of Things, smart cities, …) and XML is among the most popular data formats (like JSON) that are being used to store and exchange NoSQL data (although JSON seems to be more used than XML).

(h) Management of spatiotemporal XML data [102,142], as an extension of our present work to the spatiotemporal environment [143] in which both temporal and spatial dimensions are taken into account.

(i) Management of data evolution over time in multi-model databases [144,145] which support multiple data models, like relational, key–value, document (e.g., XML, JSON), and graph models. Thus, in such databases, XML data could coexist with other data that are stored in various data formats.

6. Summary

Since the beginning of 2000s, XML databases have been used by several applications, in particular those that are running on the Web (like e-commerce, e-health, and e-government applications). Consequently, a lot of work has been done in order to deal with (conventional) XML data management. Moreover, since most of these Web applications require keeping track of data evolution over time, in order to satisfy users' requirements concerning, e.g., temporal queries and audit missions, both researchers and practitioners have thought to models, techniques and approaches for handling time-varying XML data. In this paper, we have focused on XML data manipulation and have surveyed the different research contributions concerning this topic, in conventional XML databases as well as in temporal XML ones. We have also studied the support provided by current DBMSs for manipulating XML data.

Finally, we have given several open research directions in the area of conventional and temporal XML data management.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] W3C, Extensible markup language (XML) 1.0 (5th edition), W3C recommendation, 2008, http://www.w3.org/TR/2008/REC-xml-20081126/. (Accessed 25 November 2019).
[2] R. Bourret, XML and Databases, 2005, http://www.rpbourret.com/xml/XMLAndDatabases.html. (Accessed 25 November 2019).
[3] R. Bourret, XML database products, 2010, http://www.rpbourret.com/xml/XMLDatabaseProds.html. (Accessed 25 November 2019).
[4] A. Chaudhri, R. Zicari, A. Rashid, XML Data Management: Native XML and XML Enabled DataBase Systems, Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 2003.
[5] R. Eito-Brun, Chapter 4: Databases for XML data, in: R. Eito-Brun (Ed.), XML-Based Content Management – Integration, Methodologies and Tools, first ed., Chandos Publishing, Oxford, UK, 2017, pp. 117–153.

[6] H.V. Jagadish, S. Al-Khalifa, A. Chapman, L.V.S. Lakhsmanan, A. Nierman, S. Paprizos, J.M. Patel, D. Srivastava, N. Wiwattana, Y. Wu, C. Yu, TIMBER: A native XML database, VLDB J. 11 (4) (2002) 279–291.

[7] R. Cattell, Scalable SQL and NoSQL data stores, ACM SIGMOD Rec. 39 (4) (2010) 2–27.

[8] A. Davoudian, L. Chen, M. Liu, A survey on NoSQL stores, ACM Comput. Surv. 51 (2) (2018) 43, Article 40.

[9] J. Pokorný, NoSQL databases: a step to database scalability in web environment, Int. J. Web Inf. Syst. 9 (1) (2013) 69–82.

[10] S. Tiwari, Professional NoSQL, John Wiley & Sons, Inc., Indianapolis, Indiana, USA, 2011.

[11] Z. Brahmia, R. Bouaziz, Data manipulation in multi-temporal XML databases supporting schema versioning, in: Proceedings of the 4th International EDBT Workshop on Database Technologies for Handling XML Information on the Web, DaTaX'09, Saint-Petersburg, Russia, 22 March, 2009, paper 14, http://www.edbt.org/Proceedings/2009-StPetersburg/workshops/DataX09/pdf/datax09_14.pdf. (Accessed 25 November 2019).

[12] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Schema change operations for full support of schema versioning in the τXschema framework, Int. J. Inf. Technol. Web Eng. 9 (2) (2014) 20–46.

[13] F. Currim, S. Currim, C.E. Dyreson, R.T. Snodgrass, A tale of two schemas: creating a temporal XML schema from a snapshot schema with τXSchema, in: Proceedings of the 9th International Conference on Extending Database Technology, EDBT'04, Heraklion, Crete, Greece, 14–18 March, 2004, pp. 348–365.

[14] C.E. Dyreson, F. Grandi, Temporal XML, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, second ed., Springer-Verlag, New York, USA, 2018, http://dx.doi.org/10.1007/978-1-4614-8265-9_411.

[15] R.T. Snodgrass, C.E. Dyreson, F. Currim, S. Currim, S. Joshi, Validating quicksand: Schema versioning in τXschema, Data Knowl. Eng. 65 (2) (2008) 223–242.

[16] F. Wang, C. Zaniolo, Temporal queries and version management in XML-based document archives, Data Knowl. Eng. 65 (2) (2008) 304–324.

[17] J. Clifford, A. Croker, F. Grandi, A. Tuzhilin, On Temporal grouping, in: Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17–18 September, 1995, pp. 194–213.

[18] T.S. Chung, H.J. Kim, Techniques for the evaluation of XML queries: a survey, Data Knowl. Eng. 46 (2) (2003) 225–246.

[19] G. Gou, R. Chirkova, Efficiently querying large XML data repositories: A survey, IEEE Trans. Knowl. Data Eng. 19 (10) (2007) 1381–1403.

[20] H. Chen, H. Liao, A survey to conceptual modeling for XML, in: Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT'10, Chengdu, China, 9–11 July, 2010, pp. 473–477.

[21] M. Necaský, Conceptual Modeling for XML: A Survey, in: Proceedings of the DATESO 2006 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Desna, Czech Republic, 26–28 April, 2006.

[22] S.C. Haw, C.S. Lee, Data storage practices and query processing in XML databases: A survey, Knowl.-Based Syst. 24 (8) (2011) 1317–1340.

[23] A. Algergawy, M. Mesiti, R. Nayak, G. Saake, XML Data clustering: An overview, ACM Comput. Surv. 43 (4) (2011) 41, Article 25.

[24] X. Wu, D. Theodoratos, A survey on XML streaming evaluation techniques, VLDB J. 22 (2) (2013) 177–202.

[25] S. Faisal, M. Sarwar, Temporal and multi-versioned XML documents: A survey, Inf. Process. Manag. 50 (1) (2014) 113–131.

[26] M. Hacherouf, S.N. Bahloul, C. Cruz, Transforming XML documents to OWL ontologies: A survey, J. Inf. Sci. 41 (2) (2015) 242–259.

[27] N. Bidoit, D. Colazzo, N. Malla, C. Sartiani, Evaluating queries and updates on big XML documents, Inf. Syst. Front. 20 (1) (2018) 63–90.

[28] A. Grinberg, Modifying XML, in: XML and JSON Recipes for SQL Server, Apress, Berkeley, CA, USA, 2018, pp. 135–156.

[29] J. Liu, X.X. Zhang, Dynamic labeling scheme for XML updates, Knowl.-Based Syst. 106 (2016) 135–149.

[30] Z. Brahmia, F. Grandi, R. Bouaziz, τXuf: A temporal extension of the XQuery update facility language for the τXSchema framework, in: Proceedings of the 23rd International Symposium on Temporal Representation and Reasoning, TIME'16, 17–19 October, Technical University of Denmark, Copenhagen, Denmark, 2016, pp. 140–148.

[31] H. Hamrouni, Z. Brahmia, R. Bouaziz, A systematic approach to efficiently managing the effects of retroactive updates of time-varying data in multiversion XML databases, Int. J. Intell. Inf. Database Syst. 11 (1) (2018) 1–26.

[32] XML schema part 0: Primer second edition, W3C recommendation, 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/. (Accessed 25 November 2019).

[33] W3C, Xquery update facility 1.0, w3c Candidate recommendation, 2011, http://www.w3.org/TR/2011/REC-xquery-update-10-20110317/. (Accessed 25 November 2019).

[34] H. Hamrouni, F. Grandi, Z. Brahmia, Deferred repair of inconsistencies resulting from retroactive updates of temporal XML monetary data, Int. J. Web Inf. Syst. 13 (4) (2017) 485–519.

[35] F. Wang, C. Zaniolo, X. Zhou, ArchIS: an XML-based approach to transaction-time temporal database systems, VLDB J. 17 (6) (2008) 1445–1463.

[36] Z. Brahmia, H. Hamrouni, R. Bouaziz, Tempox: A disciplined approach for data management in multi-temporal and multi-schema-version XML databases, J. King Saud Univ. Comput. Inf. Sci. (2019) http://dx.doi.org/10.1016/j.jksuci.2019.08.009, in press, corrected proof, available online 25 August 2019.

[37] L. Anselma, L. Piovesan, A. Sattar, B. Stantic, P. Terenziani, Representing and querying now-relative relational medical data, Artif. Intell. Med. 86 (2018) 33–52.

[38] J. Clifford, C.E. Dyreson, T. Isakowitz, C.S. Jensen, R.T. Snodgrass, On the semantics of now in databases, ACM Trans. Database Syst. 22 (2), 171–214.

[39] M.-A. Baazizi, N. Bidoit, D. Colazzo, N. Malla, M. Sahakyan, Projection for XML update optimization, in: Proeedings of the 14th International Conference on Extending Database Technology, EDBT'11, Uppsala, Sweden, 21–24 March, 2011, pp. 307–318.

[40] N. Bidoit, D. Colazzo, C. Sartiani, A. Solimando, F. Ulliana, Andromeda: A system for processing queries and updates on big XML documents, in: T. Morzy, P. Valduriez, L. Bellatreche (Eds.), New Trends in Databases and Information Systems: ADBIS 2015 Short Papers and Workshops BigDap, DCSA, GID, MEBIS, OAIS, SW4CH, WISARD, in: Communications in Computer and Information Science, vol. 539, Springer International Publishing, Switzerland, 2015, pp. 218–228.

[41] B. Bouchou, D. Duarte, M.H.F. Alves, D. Laurent, M.A. Musicante, Schema evolution for XML: A consistency-preserving approach, in: Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science, MFCS'04, Prague, Czech Republic, 22–27 August, 2004, pp. 876–888.

[42] J. Dargham, Z. Alti, M. Karam, Updating XML documents without breaking their validity, in: Proceedings of the 3th International Conference on Internet and Web Applications and Services, ICIW'08, Athens, Greece, 8–13 June, 2008, pp. 342–347.

[43] L. Kircher, M. Grossniklaus, C. Grün, M.H. Scholl, Efficient structural bulk updates on the Pre/Dist/Size XML encoding, in: Proceedings of the 31st IEEE International Conference on Data Engineering, ICDE'15, Seoul, South Korea, 13–17 April, 2015, pp. 447–458.

[44] M. Klettke, H. Meyer, B. Hänsel, Evolution: The other side of the XML update coin, in: Proceedings of the ICDE Workshops 2005, Tokyo, Japan, 5–8 April, 2005, paper 1279.

[45] W. Ni, T.W. Ling, Update XML data by using graphical languages, in: Proceedings of the tutorials, posters, panels and industrial contributions at the 26th International Conference on Conceptual Modeling, ER'2007, Auckland, New Zealand, 5–9 November, 2007, pp. 209–214.

[46] M.F. O'Connor, M. Roantree, Desirable properties for XML update mechanisms, in: Proceedings of the 2010 EDBT/ICDT Workshops, Lausanne, Switzerland, 22–26 March, 2010, Article No. 23.

[47] E. Pardede, J. W.Rahayu, D. Taniar, Preserving conceptual constraints during XML updates, Int. J. Web Inf. Syst. 1 (2) (2005) 65–82.

[48] E. Pardede, J.W. Rahayu, D. Taniar, XML data update management in XML-enabled database, J. Comput. System Sci. 74 (2) (2008) 170–195.

[49] I. Tatarinov, Z.G. Ives, A.Y. Halevy, D.S. Weld, Updating XML, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM SIGMOD'01 Conference, Santa Barbara, CA, USA, 21–24 May, 2001, pp. 413–424.

[50] G.-R. Wang, X.-L. Zhang, Declarative XML update language based on a higher data model, J. Comput. Sci. Technol. 20 (3) (2005) 373–377.

[51] R.K. Wong, The extended XQL for querying and updating large XML databases, in: Proceedings of the 2001 ACM Symposium on Document Engineering, DocEng'01, Atlanta, Georgia, USA, 09–10 November, 2001, pp. 95–104.

[52] G.M. Sur, J. Hammer, J. Siméon, An xquery-based language for processing updates in XML, in: Informal proceedings of the 2nd workshop on Programming Language Technologies for XML, PLAN-X'2004, Venice, Italy, 13 January, 2004, pp. 40-53. http://www.brics.dk/NS/03/4/BRICS-NS-03-4.pdf. (Accessed 25 November 2019).

[53] W3C, XQuery 1.0: An XML query language (second edition), W3C recommendation, 2010, http://www.w3.org/TR/2010/REC-xquery-20101214/. (Accessed 25 November 2019).

[54] J. Robie, J. Lapp, D. Schach, (199 XML query language (XQL), in: The XSL Working Group, World Wide Web Consortium, 1998, http://www.w3.org/TandS/QL/QL98/pp/xql.html. (Accessed 25 November 2019).

[55] M. Liu, A logical foundation for XML, in: Proceedings of the 14th International Conference on Advanced Information Systems Engineering, CAiSE'02, Toronto, Ontario, Canada, 27–31 May, 2002, pp. 568–583.

[56] W. Ni, T.W. Ling, GLASS: A graphical query language for semi-structured data, in: Proceedings of the 8th International Conference on Database Systems for Advanced Applications, DASFAA'03, Kyoto, Japan, 26–28 March, 2003, pp. 363–370.

[57] T.W. Ling, M.L. Lee, G. Dobbie (Eds.), Semistructured Database Design, Springer Science+Business media, Inc., New York, USA, 2005.
[58] L. Al-Jadir, F. El-Moukaddem, F2/XML: Storing XML documents in object databases, in: Proceedings of the 8th International Conference on Object-Oriented Information Systems, OOIS'02, Montpellier, France, 2–5 September, 2002, pp. 108–116.
[59] R.G.G. Cattell, D.K. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda, F. Velez (Eds.), The Object Database Standard : ODMG 3.0, Morgan Kaufmann Publishers, San Francisco, California, USA, 2000.
[60] A. Eisenberg, J. Melton, SQL/XML is making good progress, SIGMOD Rec. 30 (2) (2002) 101–108.
[61] C. Grün, Storing and Querying Large XML Instances (Ph.D. thesis), University of Konstanz, Germany, 2010.
[62] F. Cavalieri, A. Solimando, G. Guerrini, Synthetising changes in XML documents as PULs, Proc. VLDB Endow. 6 (13) (2013) 1630–1641.
[63] A. Laux, L. Martin, XUpdate – XML update language, XML:DB working draft, 2000, http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html. (Accessed 25 November 2019).
[64] D. Obasanjo, S.B. Navathe, A proposal for an XML data definition and manipulation language, in: Proceedings of the 1st VLDB Workshop on Efficiency and Effectiveness of XML Tools, and Techniques, EEXTT'02, Hong Kong, China, 19 August, 2002, pp. 1–21.
[65] J. Cheney, FLUX: Functional Updates for XML, in: Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming, ICFP'08, Victoria, British Columbia, Canada, 22-24 September, 2008, pp. 3–14.
[66] B. Kane, H. Su, E.A. Rundensteiner, Consistently updating XML documents using incremental constraint check queries, in: Proceedings of the 4th ACM CIKM International Workshop on Web Information and Data Management, WIDM'02, SAIC Headquaters, McLean, Virginia, USA, 2002, pp. 1–8.
[67] B. Bouchou, M.H.F. Alves, Updates and incremental validation of XML documents, in: Proceedings of the 9th International Workshop on Database Programming Languages, DBPL'03, Potsdam, Germany, 6–8 September, 2003, Revised Papers, pp. 216–232.
[68] A. Balmin, Y. Papakonstantinou, V. Vianu, Incremental validation of XML documents, ACM Trans. Database Syst. 29 (4) (2004) 710–751.
[69] D. Barbosa, A.O. Mendelzon, L. Libkin, L. Mignet, M. Arenas, Efficient incremental validation of XML documents, in: Proceedings of the 20th International Conference on Data Engineering, ICDE'04, Boston, MA, USA, 30 March – 2 April, 2004, pp. 671–682.
[70] D. Barbosa, G. Leighton, A. Smith, Efficient incremental validation of XML documents after composite updates, in: Proceedings of the 4th International XML Database Symposium, XSym'06, Seoul, Korea, 10–11 September, 2006, pp. 107–121.
[71] F. Cavalieri, G. Guerrini, M. Mesiti, Updating XML schemas and associated documents through Exup, in: Proceedings of the 27th International Conference on Data Engineering, ICDE'11, Hannover, Germany, 11–16 April, 2011, pp. 1320–1323.
[72] F. Cavalieri, G. Guerrini, M. Mesiti, Updates on XML documents and schemas, in: Workshops Proceedings of the 27th International Conference on Data Engineering, ICDE Workshops'11, Hannover, Germany, 11–16 April, 2011, pp. 308–311.
[73] A. Solimando, G. Delzanno, G. Guerrini, Static analysis of XML document adaptations, in: Proceedings of the ER 2012 Workshops CMS, ECDM-NoCoDA, MoDIC, MORE-BI, RIGiM, SeCoGIS, WISM, Florence, Italy, 15–18 October, 2012, pp. 57–66.
[74] G. Guerrini, M. Mesiti, M.A. Sorrenti, XML schema evolution: Incremental validation and efficient document adaptation, in: Proceedings of the 5th International XML Database Symposium, XSym'07, Vienna, Austria, 23–24 September, 2007, pp. 92–106.
[75] F. Cavalieri, G. Guerrini, M. Mesiti, B. Oliboni, On the reduction of sequences of XML document and schema update operations, in: Workshops Proceedings of the 27th International Conference on Data Engineering, ICDE'11 Workshops, Hannover, Germany, 11–16 April, 2011, pp. 77–86.
[76] B. Bouchou, D. Duarte, Assisting XML schema evolution that preserves validity, in: Proceedings of the 22nd Brazilian Symposium on Databases, SBBD'07, João Pessoa, Paraíba, Brasil, 15–19 October, 2007, pp. 270–284.
[77] M. Benedikt, J. Cheney, Semantics, types and effects for XML updates, in: Proceedings of the 12th International Symposium Database Programming Languages, DBPL'09, Lyon, France, 24 August, 2009, pp. 1–17.
[78] M. Cebollero, J. Natarajan, M. Coles, Chapter 13: Xquery and Xpath, in: M. Cebollero, M. Coles, J. Natarajan (Eds.), Pro T-SQL Programmer'S Guide, fourth ed., Apress, 2015, pp. 387–431.
[79] P. Nielsen, M. White, U. Parui, Chapter 18: Manipulating XML data, in: P. Nielsen, M. White, U. Parui (Eds.), Microsoft® SQL Server® 2008 Bible, Wiley Publishing, Inc., Indianapolis, Indiana, USA, 2009, pp. 435–490.
[80] IBM DB2, DB2 PureXml – Intelligent XML database management, in: Features and Benefits of DB2 for Linux, UNIX and Windows, 2015, http://www-01.ibm.com/software/data/db2/linux-unix-windows/xml/index.html. (Accessed 25 November 2019).
[81] M. Nicola, U. Jain, Update XML in DB2 9.5. IBM developerworks, 2007, http://www.ibm.com/developerworks/db2/library/techarticle/dm-0710nicola/. (Accessed 25 Nov 2019).
[82] B. Steegmans, R. Bourret, O. Cline, O. Guyennet, S. Kulkarni, S. Priestley, V. Sylenko, U. Wahli, XML for DB2 information integration, in: IBM Redbook Series, Redmond, WA, USA, 2004, http://www.redbooks.ibm.com/redbooks/pdfs/sg246994.pdf. (Accessed 25 November 2019).
[83] Oracle, Chapter 4: Xquery and oracle XML DB, in: Oracle XML DB Developer'S Guide, 2016, https://docs.oracle.com/database/121/ADXDB/xdb_xquery.htm#ADXDB1700. (Accessed 25 November 2019).
[84] Oracle, Chapter 5: Query and update of XML data, in: Oracle XML DB Developer'S Guide, 2016, https://docs.oracle.com/database/121/ADXDB/xdb04cre.htm#ADXDB0400. (Accessed 25 November 2019).
[85] Z.H. Liu, M. Krishnaprasad, J.W. Warner, R. Angrish, V. Arora, Effective and efficient update of XML in RDBMS, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM SIGMOD'07 Conference, Beijing, China, 12–14 June, pp. 925–936.
[86] A. Eisenberg, J. Melton, Advancements in SQL/XML, SIGMOD Rec. 33 (3) (2004) 79–86.
[87] W.M. Meier, Exist native XML database, in: A.B. Chaudhri, A. Rawais, R. Zicari (Eds.), XML Data Management: Native XML and XML-Enabled Database System, Addison–Wesley, 2003, pp. 43–68.
[88] A. Fomichev, M. Grinev, S.D. Kuznetsov, Sedna: A Native XML DBMS, in: Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM'06, Merín, Czech Republic, 21–27 January, 2006, pp. 272–281.
[89] I. Taranov, I. Shcheklein, A. Kalinin, L. Novak, S.D. Kuznetsov, R. Pastukhov, A. Boldakov, D. Turdakov, K. Antipin, A. Fomichev, P. Pleshachkov, P. Velikhov, N. Zavaritski, M. Grinev, M.P. Grineva, D. Lizorkin, Sedna: native XML database management system (internals overview), in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'2010, Indianapolis, Indiana, USA, 6–10 June, 2010, pp. 1037–1046.
[90] P. Lehti, Design and Implementation of a Data Manipulation Processor for an XML Query Language (Master thesis), Technical University of Darmstadt, Germany, 2001, http://www.lehti.de/beruf/diplomarbeit.pdf. (Accessed 25 November 2019).
[91] Sedna, update language, in: Sedna Programmer'S Guide, 2012, http://www.sedna.org/progguide/ProgGuidesu6.html. (Accessed 25 Nov 2019).
[92] T.Y. Cliff Leung, C.S. Jensen, R.T. Snodgrass, Chapter 16: Modification (Ph.D. thesis), 2000, pp. 445–448, http://people.cs.aau.dk/~csj/Thesis/pdf/chapter16.pdf. (Accessed 25 November 2019).
[93] C. De Castro, F. Grandi, M.R. Scalas, Extensional data management in multitemporal relational databases supporting schema versioning, in: Proceedings of the 3rd National Conference on Advanced Database Systems, SEBD'95, Ravello, Italy, 28–30 June, 1995, pp. 365–384.
[94] C.S. Jensen, Temporal Database Management (Ph.D. thesis), Department of Computer Science, Aalborg University, Aalborg, Denmark, 2000.
[95] R.T. Snodgrass, I. Ahn, G. Ariav, D. Batory, J. Clifford, C.E. Dyreson, R. Elmasri, F. Grandi, C.S. Jensen, W. Käfer, N. Kline, K. Kulkarni, T.Y.C. Leung, N. Lorentzos, R. Ramakrishnan, J.F. Roddick, A. Segev, M.D. Soo, S.M. Sripada, TSQL2 language specification, ACM SIGMOD Rec. 23 (1) (1994) 65–86.
[96] A. Gal, O. Etzion, A multiagent update process in a database with temporal data dependencies and schema versioning, IEEE Trans. Knowl. Data Eng. 10 (1) (1998) 21–37.
[97] R.M. Galante, N. Edelweiss, C.S. dos Santos, Á.F. Moreira, Data modification language for full support of temporal schema versioning, in: Proceeding of the 18th Brazilian Symposium on Databases, SBBD'03, Manaus, Amazonas, Brazil, 6–8 October, 2003, pp. 114–128.
[98] E. Rose, A. Segev, A temporal object-oriented query language, in: Proceedings of the 12th International Conference on the Entity-Relationship Approach, ER'93, Arlington, Texas, USA, 15–17 December, 1993, pp. 122–136.
[99] K. Nørvåag, M. Limstrand, L. Myklebust, TexOR: Temporal XML database on an object-relational database system, in: Proceedings of the 5th Andrei Ershov Memorial Conference, PSI'2003, Akademgorodok, Novosibirsk, Russia, 9–12 July, in: LNCS, vol. 2890, 2003, pp. 520–530, Revised papers.
[100] K. Nørvåag, The design, implementation, and performance of the V2 temporal document database system, Inf. Softw. Technol. 46 (9) (2004) 557–574.
[101] F. Grandi, Introducing an annotated bibliography on temporal and evolution aspects in the World Wide Web, ACM SIGMOD Rec. 33 (2) (2004) 84–86.
[102] L. Bai, C. Xu, Spatiotemporal query algebra based on native XML, in: Handbook of Research on Innovative Database Query Processing Techniques, IGI Global, 2016, pp. 275–293.

[103] F. Grandi, F. Mandreoli, P. Tiberio, Temporal modelling and management of normative documents in XML format, Data Knowl. Eng. 54 (3) (2005) 327–354.

[104] F. Rizzolo, A.A. Vaisman, Temporal XML: Modeling, indexing, and query processing, VLDB J. 17 (5) (2008) 1179–1212.

[105] F. Currim, S. Currim, C.E. Dyreson, S. Joshi, R.T. Snodgrass, S.W. Thomas, E. Roeder, τXSchema: Support for Data- and Schema-Versioned XML Documents, Technical Report TR-91, TimeCenter, 2009, http://timecenter.cs.aau.dk/TimeCenterPublications/TR-91.pdf. (Accessed on 25 November 2019).

[106] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Supporting structural evolution of data in web-based systems via schema versioning in the τXschema framework, in: A. Elçi (Ed.), Handbook of Research on Contemporary Perspectives on Web-Based Systems, IGI Global, Hershey, PA, USA, 2018, pp. 271–307.

[107] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Schema versioning, in: M. Khosrow-Pour (Ed.), Encyclopedia of Information Science and Technology, third ed., IGI Global, Hershey, Pennsylvania, USA, 2015, pp. 7651–7661.

[108] J.F. Roddick, in: L. Liu, M.T. Özsu (Eds.), Schema Versioning, second ed., Encyclopedia of Database Systems, New York, NY, USA, 2018, http://dx.doi.org/10.1007/978-1-4614-8265-9.

[109] H. Hamrouni, Z. Brahmia, R. Bouaziz, Automatic and graceful repairing of data inconsistencies resulting from retroactive updates in temporal XML databases, in: Proceedings of the 3rd International Conference on Data Management Technologies and Applications, DATA'2014, Vienna, Austria, 29–31 August, 2014, pp. 243–250.

[110] H. Hamrouni, Z. Brahmia, R. Bouaziz, An efficient approach for detecting and repairing data inconsistencies resulting from retroactive updates in multi-temporal and multi-version XML databases, in: Proceedings II of the 18th East-European Conference on Advances in Databases and Information Systems, ADBIS'14, Ohrid, Republic of Macedonia, 7–10 September, 2014, pp. 135–146.

[111] C.E. Dyreson, Vacuuming XML, in: Proceedings of the 2014 IEEE 11th International Conference on Ubiquitous Intelligence and Computing, and 2014 IEEE 11th International Conference on Autonomic and Trusted Computing, and IEEE 14th International Conference on Scalable Computing and Communications and Its Associated Workshops, UTC-ATC-ScalCom, Bali, Indonesia, 9–12 December, 2014, pp. 625–628.

[112] C.S. Jensen, Chapter 19: Vacuuming (Ph.D. thesis), 2000, pp. 465–475, http://people.cs.aau.dk/~csj/Thesis/pdf/chapter19.pdf. (Accessed 25 November 2019).

[113] J.F. Roddick, D. Toman, Temporal vacuuming, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, Springer US, 2009, pp. 3023–3027.

[114] J. Skyt, C.S. Jensen, L. Mark, A foundation for vacuuming temporal databases, Data Knowl. Eng. 44 (1) (2003) 1–29.

[115] C.E. Dyreson, Observing transaction-time semantics with TTXPath, in: Proceedings of the 2nd International Conference on Web Information Systems Engineering, WISE'01, Kyoto, Japan, 3–6 December, 2001, pp. 193–202.

[116] W3C, XML path language (xpath) 3.0, W3C recommendation, 2014, http://www.w3.org/TR/2014/REC-xpath-30-20140408/. (Accessed 25 November 2019).

[117] D. Gao, R.T. Snodgrass, Temporal slicing in the evaluation of XML documents, in: Proceedings of the 29th International Conference on Very Large Data Bases, VLDB'03, Berlin, Germany, 9–12 September, 2003, pp. 632–643.

[118] S.-Y. Noh, S.K. Gadia, A comparison of two approaches to utilizing XML in parametric databases for temporal data, Inf. Softw. Technol. 48 (9) (2006) 807–819.

[119] F. Wang, C. Zaniolo, XBiT: An XML-based bitemporal data model, in: Proceedings of the 23rd International Conference on Conceptual Modeling, ER'04, Shanghai, China, 8–12 November, 2004, pp. 810–824.

[120] J. Klímek, J. Malý, M. Nečaský, I. Holubová, EXolutio: Methodology for design and evolution of XML schemas using conceptual modeling, Informatica 26 (3) (2015) 453–472.

[121] H. Liu, Y. Lu, Q. Yang, XML conceptual modeling with XUML, in: Proceedings of the 28th International Conference on Software Engineering, ICSE'06, Shanghai, China, 20–28 May, 2006, pp. 973–976.

[122] C. Combi, B. Oliboni, Conceptual modeling of XML data, in: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC'2006, Dijon, France, 23–27 April, 2006, pp. 467–473.

[123] F. Massimo, G. Donatella, M. Angelo, P. Carla, A graph-theoretic approach to map conceptual designs to XML schemas, ACM Trans. Database Syst. 38 (1) (2013) Article 6.

[124] Q. Hoang, T. Van Nguyen, H.L.M. Vo, T.T.N. Thuy, A method for transforming timeER model-based specification into temporal XML, in: Advanced Computational Methods for Knowledge Engineering, 2016, pp. 59–73.

[125] K.A. Ali, J. Pokorný, A comparison of XML-based temporal models, in: E. Damiani, K. Yetongnon, R. Chbeir, A. Dipanda (Eds.), Advanced Internet Based Systems and Applications, SITIS'06, in: LNCS, vol. 4879, Springer, Berlin, Heidelberg, 2009, pp. 339–350, (Revised Selected Papers).

[126] L.I. Rusu, W. Rahayu, D. Taniar, Storage techniques for multi-versioned XML documents, in: Proceedings of 13th International Conference on Database Systems for Advanced Applications, DASFAA'08, New Delhi, India, 19–21 Mars, 2008, pp. 538–545.

[127] A.O. Mendelzon, F. Rizzolo, A. Vaisman, Indexing temporal XML documents, in: Proceedings of the 30th International Conference on Very Large Data Bases, VLDB'04, Toronto, Ontario, Canada, 31 August – 3 September, 2004, pp. 216–227.

[128] A. Baqasah, E. Pardede, W. Rahayu, XSM - A tracking system for XML schema versions, in: Proceedings of the 28th IEEE International Conference on Advanced Information Networking and Applications, AINA'2014, Victoria, BC, Canada, 13–16 May, 2014, pp. 1081–1088.

[129] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Schema evolution, in: M. Khosrow-Pour (Ed.), Encyclopedia of Information Science and Technology, third ed., IGI Global, Hershey, Pennsylvania, USA, 2015, pp. 7641–7650.

[130] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Schema evolution in conventional and emerging databases, in: M. Khosrow-Pour (Ed.), Encyclopedia of Information Science and Technology, fourth ed., IGI Global, Hershey, PA, USA, 2018, pp. 2043–2053.

[131] J.F. Roddick, Schema evolution, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, second ed., Springer, New York, NY, USA, 2018, http://dx.doi.org/10.1007/978-1-4614-8265-9_1532.

[132] Z. Brahmia, F. Grandi, B. Oliboni, R. Bouaziz, Schema versioning in conventional and emerging databases, in: M. Khosrow-Pour (Ed.), Encyclopedia of Information Science and Technology, fourth ed., IGI Global, Hershey, PA, USA, 2018, pp. 2054–2063.

[133] S. Brahmia, Z. Brahmia, F. Grandi, R. Bouaziz, Managing temporal and versioning aspects of JSON-based big data via the τJschema framework, in: Proceedings of the International Conference on Big Data and Smart Digital Environment, ICBDSDE'18, Casablanca, Morocco, Studies in Big Data (53), Springer Nature Switzerland AG, pp. 27–39.

[134] A. Cuzzocrea, Temporal aspects of big data management: state-of-the-art analysis and future research directions, in: Proceedings. of the 22nd International Symposium on Temporal Representation and Reasoning, TIME'15, Kassel, Germany, 23–25 September, 2015, pp. 180–185.

[135] S. Brahmia, Z. Brahmia, F. Grandi, R. Bouaziz, τJSchema: A framework for managing temporal JSON-based nosql databases, in: Proceedings of the 27th International Conference on Database and Expert Systems Applications, DEXA'16, Porto, Portugal, 5–8 September, 2016, Part 2, pp. 167–181.

[136] N.Q. Mehmood, R. Culmone, L. Mostarda, Modeling temporal aspects of sensor data for MongoDB NoSQL database, J. Big Data 4 (2017) paper 8.

[137] M. Eshtay, A. Sleit, M. Aldwairi, Implementing bi-temporal properties into various NoSQL database categories, Int. J. Comput. 18 (1) (2019) 45–52.

[138] S. Brahmia, Z. Brahmia, F. Grandi, R. Bouaziz, Temporal JSON schema versioning in the τJschema framework, J. Digit. Inf. Manag. 15 (4) (2017).

[139] S. Brahmia, Z. Brahmia, F. Grandi, R. Bouaziz, A disciplined approach to temporal evolution and versioning support in JSON data stores, in: Emerging Technologies and Applications in Data Processing and Management, IGI Global, 2019, pp. 114–133.

[140] Z. Brahmia, S. Brahmia, F. Grandi, R. Bouaziz, Implicit JSON schema versioning driven by big data evolution in the τJSchema Framework, in: Proceedings of the International Conference on Big Data and Networks Technologies, BDNT'2019, Leuven, Belgium, 29 April – 2 May, 2019, pp. 23–35.

[141] A. Goyal, Temporal JSON, (Master's thesis), Utah State University, Logan, Utah, USA, 2019, https://digitalcommons.usu.edu/etd/7653/. (Accessed 25 November 2019).

[142] Y. Chen, P. Revesz, Querying spatiotemporal XML using DataFox, in: Proceedings of the 2nd ACM International Conference on Web Intelligence, 2003, pp. 301–309.

[143] X. Xiong, M.F. Mokbel, W.G. Aref, Spatiotemporal database, in: S. Shekhar, H. Xiong, X. Zhou (Eds.), Encyclopedia of GIS, Springer International Publishing AG, Cham, Switzerland, 2017, pp. 2150–2151.

[144] I. Holubová, M. Klettke, U. Störl, Evolution management of multi-model data, in: Heterogeneous Data Management, Polystores and Analytics for Healthcare, Springer, Cham, 2019, pp. 139–153.

[145] M. Vavrek, I. Holubová, S. Scherzinger, MM-evolver: A multi-model evolution management tool, in: Proceedings of 22nd International Conference on Extending Database Technology, EDBT'2019, Lisbon, Portugal, March 2019, 26–29, pp. 586–589.