

Integration and interoperability of data sources: forward into the new century

Jaroslav POKORNÝ

Charles University, Czech Republic
e-mail: pokorny@ksi.ms.mff.cuni.cz

Abstract: The goal of the next years is to publish both human-readable and machine-processable information on the Web. E-commerce, intelligent services, search agents, and information filters require integrating a huge amount of heterogeneous data sources. A form of data integration on various levels of its processing is necessary. In addition to syntactic and structural data description, a part of its semantics must be formally expressed. Today's attempts with XML are only the first step to data integration and, consequently, to data interoperability. In the paper we summarise the roles of XML in this development. Then, we show some deficiencies of XML with respect to integration of data sources and present RDF language as a tool, which improves many of these issues. However, in the current state neither XML nor RDF provides sufficient support for the integration of heterogeneous structures or different meanings of terms. Shared vocabularies and conceptualizations are needed. Although there are no universal tools for conceptualisations, some restricted approaches to so called ontologies help to resolve these problems. A number of such techniques are mentioned in the paper and some conclusions concerning their usability are formulated.

1. Introduction

A motivation for data integration and interoperability is very old and reflects activities from 80s, when various databases were integrated. Companies have been very interested in the decentralization of processing (at the system level) while achieving an integration of the data sources (at the logical level) within their geographically distributed systems of databases, applications, and users. With the advent of the Web, users formulated new requirements on distributed data processing. After simple "surfing" in the early 90s, more advanced approaches are needed. On the most general level, the goal is to formulate a query and obtain an answer without any previous knowledge about data sources participated during the query evaluation.

It is well known that any interoperability would not have been possible without standardization. The standardization concerns both hardware and software. Unfortunately, there are many standards. In [Ke20], J. Ketchell mentions around 200 separate open organizations producing "standards", all too often not talking to one other, and certainly invisible to many users.

The Extensible Markup Language (XML) developed by the WWW Consortium (W3C) [W3C98] is recognized today as the current standard for establishing interoperability on the Web. Since it provides a standard syntax for representing data, it is assumed to be a key enabling technology for exchange of information

on the Web and within corporations. As a consequence, integration of XML data from multiple external data sources is becoming a critical task.

On the other hand, it is well known that XML is not more than a basis for interoperability on the Web. In fact, it provides only one member of the family of various tools that are adepts for fulfilling requirements of interoperability. Being only syntax, XML even only partially supports these requirements. However, XML without agreed upon Document Type Definitions (DTD) does nothing to support integration at the semantic level. The names and meanings of the tags used in XML documents are arbitrary. As a result, the emergence of XML implies significant activities in various communities to agree on DTDs or, more recently, XML schemes [W3C00a].

To enable interoperability on the Web, however, a number of issues should be considered:

- *Languages for resource descriptions.* A key element of a data integration system is a language for describing the contents and capabilities of data sources. These descriptions provide the semantic mapping between the data in the source and the relations in the mediated schema.
- *Query reformulation algorithms.* We need to develop algorithms for efficient reformulating user queries (posed on a mediated schema) to queries that refer to the data sources. The known techniques for reformulation from the relational case do not extend easily to languages for querying, e.g., XML data.
- *Translation among metadata schemes.* There will be a significant need for tools that translate e.g. XML data conforming to one DTD into an XML document conforming to a different DTD (presumably with semantically related content).
- *Obtaining source descriptions:* When the number of data sources grows, we can no longer manually write their content descriptions. We must develop methods for automatically (or at least, semi-automatically) computing source descriptions for newly introduced data sources.

Any data must be made visible in a way that allows people or systems to tell whether the sources are likely to be useful to them. In the past we developed tools for describing data sources based mainly on relational DBMSs. It is often cited that classical database techniques that support interoperability are not usable for the same purpose in Web environment. Consequently, we need new models and languages for describing e.g. XML sources, full text databases, video catalogues, etc. The main novel challenges that arise include (1) the kinds of data restructuring that appear in these applications are richer than in relational data, (2) we need to scale up to a very large number of data sources, and (3) exploit the knowledge conveyed by accompanying metadata.

In the paper, we restrict the notion of metadata the structured data, which describes the characteristics of a resource. Metadata is also a systematic method for describing resources and thereby improving access to them. Remind the relational case. Relational metadata is, e.g., a relational schema derived from

CREATE TABLE statements, or a relational schema equipped with a conceptual domain description given by an E-R schema.

The approach with XML provides similar issues. First, a common domain model is necessary. Although some E-R schemes or UML specifications are usually at disposal, DTDs as well as XML schemes are often constructed in an ad hoc way or, vice versa, the domain model is reengineered from DTDs and XML schemes. Second, using XML in situations when new data sources or communication partners are added requires more effort than necessary. Thus, semantic interoperability is gained only partially.

In other words, we need more intelligent approaches than XML to guarantee intelligent services such as information brokers, search agents, information filters, etc. For example, other W3C standard, RDF [W3C99], better facilitates interoperability because it provides unambiguous methods of expressing data semantics, and a data model that can be extended to address sophisticated ontology¹ representation techniques.

In this paper, we focus mainly on the languages that seem to be appropriate for describing data in various heterogeneous data sources. Their main feature should be a support of *semantic interoperability*. To achieve it, systems must be able to exchange data in such way that the precise meaning of the data is readily accessible and data itself can be translated by any system into a form that it understands.

We briefly summarize key elements of XML, DTDs and XML Schema (Section 2), and some roles of these tools in recent approaches to data interoperability. The notion of metadata will be discussed in Section 3. As an example of one usable metadata schema we briefly mention the Dublin Core standard [DCMI99a]. Further we continue with RDF, which is able to express metadata. It includes a basic model, RDM-graphs, and RDF schema. We will also compare RDF and XML. In conclusions we draw some trends towards so called semantic Web.

2. XML basics

Due to its modelling power, XML language is considered to be the most appropriate for integration of various heterogeneous data today. Moreover, XML provides richer possibilities for modelling data than, e.g., the relational data model. Typically, XML offers a straightforward representation of hierarchical structures, such as in Fig. 1, which would be represented as, e.g., relations fairly clumsily.

XML is used to serve a range of purposes:

- *Uniform data-exchange format*. This is a primary goal for XML.
- *Semantic markup of Web pages*.

¹ AI (Artificial Intelligence) researchers mean by an ontology a formal specification of the concepts and relations of some domain.

- *Serialization syntax for other markup languages* (e.g. XHTML, Chemical Markup Language, Open Financial Exchange, Math Markup Language, etc.).
- *Tool for data integration*. We can view heterogeneous data as XML data. We can access, manipulate, and query this data.
- *New database model*. XML data can be stored and approached in a database way (see, e.g. [Pok00a]).

Last two items show that XML data can be conceived both on the physical and logical level. First, we shortly describe basic XML notions - elements, attributes, then DTDs, and a new language called XML Schema

2.1 Elements, attributes

XML models data as a tree of *elements* that contain character data and have *attributes* composed of name-value pairs. For example, consider an XML representation of information for a book in Fig. 2. In an element, inside `<...>` is markup, while the rest is character data. The `Title` element contains character data denoting the book title and is contained in the `Book` element. Similarly, the `Price` element contains character data denoting the book's price. This element also has an attribute named `currency` with value `EGP`, represented using the syntax `attribute-name="attribute-value"` within the element's start tag. In general, element names are not unique; e.g., the `Book` element in our example contains two `Review` elements. On the other hand, attribute names are unique within an element. For example, the `Price` element cannot have another attribute named `currency`. The syntax permits an empty element `<Bestseller></Bestseller>` to be represented more concisely as `<Bestseller/>`. Elements may contain a mix of character data and other elements, i.e. so-called *mixed content*.

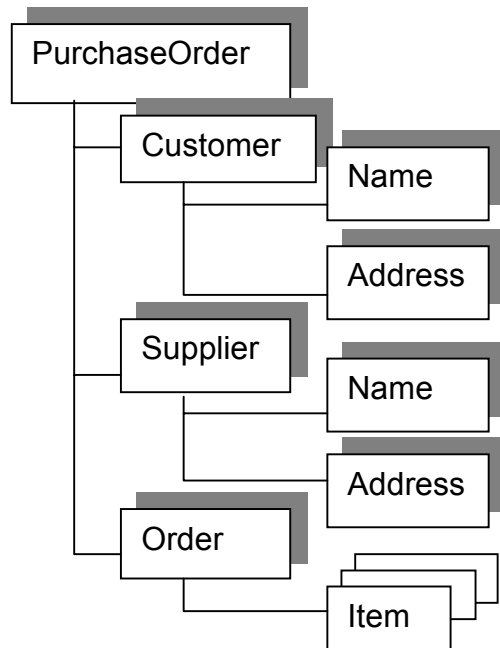


Fig. 1. Conceptual schema

```

<?xml version="1.0"?>
<Book>
  <Title> XML Black Book </Title>
  <Author> <Firstname>Nathanya</Firstname>
    <Lastname> Pitts</Lastname>
</Author>
  <Year> 2000 </Year>
  <Price currency="EGP">155</Price>
  <Review><Reviewer>Jim</Reviewer>
    <Rtext>Very good... </Rtext>
</Review>
  <Review><Reviewer>John</Reviewer>
    <Rtext> The book is... </Rtext>
</Review>
  <Bestseller authority = "Web quarterly"/>
</Book>
  
```

Fig. 2. Book description in XML

XML documents are called *well-formed* if they satisfy simple syntactic constraints, such as proper delimiting of element names and attributes and proper

nesting of start and end tags. Although technically not required, XML documents should begin with an XML declaration, similar to the one on the first line of our example, identifying the version of XML used.

2.2 DTD

Applications that operate on XML data often need additional guarantees on the structure and content of such data. For example, a program that calculates the tax on the sale of a book may need to assume that each **Book** element in its XML input includes a **Price** subelement with a **currency** attribute and a numeric content. Such constraints on document structure can be expressed using a DTD. A DTD defines a class of XML documents using a language that is essentially a context-free grammar with several restrictions. For example, one may use the DTD declaration in Fig. 3 to constrain XML documents in our example.

```
<!DOCTYPE Book[
  <!ELEMENT Book (Title, Author+, Year?, Price, Review*,
                 Bestseller?)>
  <!ELEMENT Title (#PCDATA)>
  <!ELEMENT Author (Firstname, Lastname)>
  <!ELEMENT Year (#PCDATA)>
  <!ELEMENT Price (#PCDATA)>
  <!ATTLIST Price currency CDATA „EGP“
              source (list | regular | sale) list
              taxed CDATA #FIXED "yes">
  <!ELEMENT Review (Reviewer, Rtext)>
  <!ELEMENT Bestseller EMPTY>
  <!ATTLIST Bestseller authority CDATA #IMPLIED>
]>
```

Fig. 3. Example of DTD for books

The first line of this declaration is an element type declaration that constrains the contents of the **Book** element. The declaration syntax uses commas for sequencing, parentheses for grouping, and the operators **?**, *****, and **+** to denote, respectively, zero or one, zero or more, and one or more occurrences of the preceding construct. Our DTD requires every **Book** element to have **Title** and **Price** subelements, and at least one author.

The second line declares the type for the **Title** element to be *parsed character data* (PCDATA) implying an XML processor will parse the contents looking for markup. The use of some element names (e.g., **Rtext**, **Lastname**) without a corresponding declaration is allowed in XML. Such elements are simply not constrained by this DTD. The last two lines declare **Bestseller** as empty with the optional **authority** attribute of type character data. The declaration also indicates that the **Price** element may have attributes **currency**, of type character data and default value **EGP**, **source**, with one of the three values shown (an enumerated type) and default value **list**; and **taxed**, with the fixed value **yes**. The fixed attribute type is a special case of the default attribute type; it determines that the specified default value must not be changed by an XML document conforming to the DTD. Fixed-value attributes are convenient for ensuring that data critical to

processing an element type is available with the desired value without requiring it to be explicitly specified for each element of that type.

An XML document that satisfies the constraints of a DTD is said to be *valid* with respect to that DTD. The DTD associated with an XML document may be specified using several methods, one of which is the inclusion of a document type declaration `!DOCTYPE Book`. This declaration indicates that the XML document claims validity with respect to the `Book`.

Finally, we try to model partially the conceptual world depicted in Fig. 1. Some items can be ordered are books. DTD covering both the items and the purchase orders is in Fig. 4. Books are structured into subelements, other items (artefacts) are expressed only as strings.

2.3 XML Schema

Also the data modelling facilities provided by DTDs are insufficient for ensuring interoperability of more data sources. We cannot, e.g., use DTDs to require that the value of the element `Price` be a fixed precision real number with the range `<0,2000>` with two digits after the point. Thus our tax calculation cannot rely on XML validity with respect to its DTD for such simple error-checking. Another deficiency is that DTDs are not written in XML. The XML Schema [W3C00b] defines facilities that address these needs.

XML Schema is the new language that is currently being drawn up by W3C to describe the content and structure of XML documents in XML. XML Schema reproduces the full capabilities of DTD, so existing DTD document schemes can be translated to XML Schema without problems. However, it goes beyond these capabilities, allowing additional types of constraints to be specified.

Built-in and user-defined data types

One of the main weaknesses of DTD was its lack of support for data types beyond character strings. XML Schema provides a wider range of primitive data types. For example, in the Fig. 3, the `Price` element is described as character data, which makes elements such as

```
<!DOCTYPE PurchaseOrder [  
  <!ELEMENT PurchaseOrder (Customer,Supplier,Order)>  
  <!ELEMENT Customer(Name, Address)>  
  <!ELEMENT Supplier(Name, Address)>  
  <!ELEMENT Order(Item*)>  
  <!ELEMENT Name (#PCDATA)  
  <!ELEMENT Address (#PCDATA)>  
  <!ELEMENT Item (Book | Artefact)  
  <!ATTLIST Item amount CDATA>  
  <!ELEMENT Book (Title, Author+, Year?, Price, Review*,  
    Bestseller?)>  
  <!ELEMENT Title (#PCDATA)>  
  <!ELEMENT Author (Firstname, Lastname)>  
  <!ELEMENT Year (#PCDATA)>  
  <!ELEMENT Price (#PCDATA)>  
  <!ATTLIST Price currency CDATA „EGP“  
    source (list | regular | sale) list
```

```

        taxed CDATA #FIXED "yes">
    <!ELEMENT Review (Reviewer, Rtext)>
    <!ELEMENT Bestseller EMPTY>
    <!ATTLIST Bestseller authority CDATA #REQUIRED>
    <!ELEMENT Artefact(#PCDATA)>
  ]>

```

Fig. 4. DTD for purchase orders

<Price>Hi!</Price> legal. Using XML Schema, we can state that the Price element is of positive integer data type, which limits possible data values to positive whole numbers.

Further constraints can be placed on the range of possible data values by creating new data types that extend built-in data types. For example, if our books cover only current literature, with XML Schema, we can limit the values of the Year element to be between 1995 and 2000, as shown in Fig. 5. Prices of the books can also be limited to decimal numbers greater than zero that have two digits after the decimal point.

Occurrence indicators

DTD gives some control over the number of subelements that can be contained in an element. For example, in Fig. 4, we allowed more than one Author element in a Book element, while the Year element has been made optional.

XML Schema is more flexible in specifying the number of allowable subelement instances. The minimum and maximum number of occurrences can be any non-negative whole number, not just zero, one, or infinity. In our example, we might want to limit the number of authors of a book to three.

Refinement mechanism

Another new feature of XML Schema is refinement, where elements can inherit the schema constraints (i.e. element content and attribute list) of other elements, and extend or modify these constraints for their particular purposes. This mechanism supports code reuse. Common properties can be shared, so that there is no need to redeclare them again.

For example, we could modify reviewers in Fig. 6 in such way, that both Author and Reviewer elements may store first names and last names. Instead of defining these common properties in both elements, they can be declared once in the supertype, Person element.

Extensibility Mechanism

In DTD, an instance of an element cannot have additional subelements and attributes that have not been specified in the element declaration of the schema. XML Schema relaxes this rule by providing three types of element models with regards to extensibility.

- In the *open model*, the content and attributes that have been declared for the element are required, but other content and attributes can be present.
- The *refinable model* requires the content and attributes that have been declared for the element, and allows for those that have been explicitly declared in the refined subtypes.

- The *closed model* reflects the status quo of DTD, where additional child elements and attributes not in the element declaration are not allowed to be present.

```

<datatype name="YearType">
  <basetype name="positive-integer"/>
  <minInclusive>1995</minInclusive>
  <maxInclusive>2000</maxInclusive>
</datatype>
<element name="year"
  type="YearType"></element>
<datatype name="PriceType">
  <basetype name="decimal"/>
  <minExclusive>0.00</minExclusive>
  <scale>2</scale>
</datatype>
<element name="Price"
  type="PriceType"></element>
<element name='Author' minOccurs='1'
  maxOccurs='3'>
...
</element>
<element name='Person'>
...
</element>
<element name='Author'>
<refines name='Person'!>
...
</element>
<element name='Reviewer'>
<refines name='Person'!>
...
</element>

```

Fig. 6. Type hierarchy in XML schema

Fig. 5. Integrity constraints by XML Schema

These changes to the element model in XML Schema allow for both extensibility and control. For example, an open model permits extensibility, which was not possible in DTD. Old document schemes will be able to validate new documents that have additional features, which are not known to them.

There are tendencies to replace DTDs with XML schema definitions. Although XML schema offers some advantages over DTDs, its role is essentially the same: to define a grammar for XML documents.

3. Metadata and its encoding

Metadata is a description of objects, documents or services that may contain data about their form and content. It may be part of the data sources themselves or kept separately from them. The reason for creating metadata, from the provider perspective, is to improve the possibilities of data retrieval (especially the search precision) as well as to support control and management of data sources.

Each metadata schema includes a limited number of *elements*, the *name* of each element, and the *meaning* of each element. In context of data sources, the most popular metadata schema is now Dublin Core (CD) [DCMI99a]. Since each metadata schema is also a vocabulary, we can speak about a namespace in our context. A *namespace* is a vocabulary that has been formally published, usually on the Web; it describes objects with natural language labels, definitions, and other relevant documentation. Namespaces are, e.g., an important addition to the base XML recommendation because they permit distributed autonomous development of XML schemes without fear of name clashes.

While the syntax is not strictly part of the metadata schema, the data will be unusable, unless the encoding schema understands the semantics of the metadata schema. Important encoding schemes include e.g. HTML, SGML, XML, RDF, MARC, and Z39.50.

Metadata interoperability is a fundamental requirement for access to information on the Internet. In particular there are three scenarios in which interoperability between metadata descriptions is essential:

- to apply a single query syntax over descriptions expressed in multiple descriptive formats,
- to express the relationship between multiple descriptions in terms of a "core" or "canonical" description,
- to project community or individual specific descriptions out of a single canonical description.

The main problem of various metadata schemes is either insufficient description of its semantics or insufficient choice of elements. There is a number of possibilities how to solve these problems. The term "application profile" has recently become highly topical. Heery and Patel [HP00] define *application profiles* as metadata schemes that consist of metadata elements drawn from one or more namespaces, combined together by implementors and optimized for a particular local application. Other approach supports developing domain-specific profiles (see, e.g. various application instances of XML). More advanced solutions include building a dictionary (registry, ontology) for a particular application domain.

In this chapter, we mention a simple approach to metadata interoperability given by DC standard. Then we introduce a more advanced approach - the framework RDF.

3.1 Dublin Core

There are many different sources of metadata today that need vast resources of time, and qualified staff. The DC Set has been developed to provide a simpler solution. A small set of metadata elements such as the DC would be valuable for at least four reasons.

- it would encourage authors and publishers to provide metadata, in a form that automated resource discovery tools could collect it, when they make data available,
- it would encourage the creation of network publishing tools that contain a template for metadata elements, further simplifying the task of creating metadata records,
- a record created with the DC could serve as the basis for a more detailed cataloguing record if the need arises,
- if something like the DC became a standard, metadata records could be understood across user communities.

DC includes the following 15 elements:

Title	The name of the object
Subject	The topic addressed by the work
Description	A textual description of the content of the resource
Source	Objects, either print or electronic, from which this object is derived, if applicable
Language	Language of the intellectual content
Relation	Relationship to other resources, i.e. images in a document, chapters in books etc.
Coverage	The spatial locations and temporal durations characteristic of the object
Author or Creator	The person(s) primarily responsible for the intellectual content of the object
Publisher	The agent or agency responsible for making the object available
Contributor	The person(s), such as editors and transcribers, who have made other significant intellectual contributions to the work
Rights	A rights management statement or an identifier that links to a rights management statement
Date	The date of publication or availability
Type	The kind of the object, such as novel, poem, or dictionary
Format	The data representation of the object, such as Postscript file or Windows executable file
Identifier	String or number used to uniquely identify the object

The basic principle of usage DC says that every element is optional and repeatable. In additions to the elements, qualifiers [DCMI00] that modify the properties of DC statements can be used. In July 2000, 52 qualifiers have been recommended. For example, `dcq: iso8016` qualifies `dc:date` to specify the string "2001-06-04" is formatted according to the international standard. Notice that `dc` and `dcq` are conventional abbreviations of namespaces DC elements and DC qualifiers, respectively. Fig. 7 shows a DC metadata record given for this paper:

Subject = interoperability
Subject = data sources
Subject = XML
Subject = RDF
Subject = Metadata
Title = Integration and interoperability of data sources: forward into the new century
Author = Pokorny, Jaroslav
Publisher = American University in Cairo
Date = June, 2001

Type = conference paper
Format = text
Identifier = http:// text www.riti.org/bitworld2001/index.htm
Relation = http://www.kocour.cuni.cz/paper1
Language = English

Fig. 7. DC metadata

DC is by design independent of any particular encoding format. We will try to encode a part of DC metadata in Fig 7 using the <META> tag from HTML.

```
<META NAME=" DC.Title" CONTENT="Integration and interoperability of  
data sources: forward into the new century ">  
<META NAME=" DC.Date" CONTENT="June, 2000">
```

Notice that XML/RDF offers more advanced methods how to encode DC metadata. Concerning its expressive power, everybody will agree that DC is relatively weak tool for ensuring interoperability of data sources. Particularly, "typing" of metadata is weak. For example, `dc:rights` can be free text or a URL, `dc:coverage` can be a place name, the name of a time period, etc.

3.2 RDF

Both DTD and XML schema allow to describe the structure and to some extent semantics of document. On the other hand, XML makes no commitment on:

- domain specific ontological vocabulary,
- ontological modelling primitives.

This requires pre-arranged agreement in a choice of the tag collection and natural-language comments. This observation reduces essentially possibilities of XML in area of data interoperability. XML is feasible only for closed collaboration in the case of

- agents in a small and stable community,
- Web pages on a small and stable Intranet.

In consequence, XML is not sufficient for sharable Web resources.

The *Resource Description Framework (RDF)* [W3C99c] provides a means for adding semantics to a document without making any assumptions about its structure. More specifically, RDF is an infrastructure that enables the encoding, exchange, and reuse of structured metadata. RDF describes so called *resources*. According to W3C, all things being described by RDF expressions are called resources. A resource may be an entire Web page; such as the HTML document "http:// text www.riti.org/bitworld2001/index.htm" for example. A resource may be a part of a Web page; e.g. a specific HTML or XML element within the document source. A resource may also be a whole collection of pages; e.g. an entire Web site. A resource may also be an object that is not directly accessible via the Web; e.g. a printed book. Resources are always named by URIs² plus optional anchor ids. Anything can have a URI; the extensibility of URIs allows

² Short for Uniform Resource Identifier, the generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI.

the introduction of identifiers for any entity imaginable. With RDF descriptions search engines, intelligent agents, information broker, browsers and human users can make use of *semantic* information. However, RDF is equally well suited to representing data.

RDF is an XML application (i.e., its syntax is defined in XML) customized for adding meta-information to Web documents. Resources are described by *properties* (or *attributes*). *RDF statements* associate a property-value pair with a resource; they are thus triples composed of a subject (resource), a predicate (property), and an object (property value).

- A *subject* is an entity that can be referred to by an address in the Web (i.e., by an URI). Resources are the elements that are described by RDF statements.
- A *predicate* defines a binary relation between resources and/or atomic values provided by primitive data type definitions in XML.
- An *object* specifies for a subject a value for a predicate. That is, objects provide the actual characterizations of the Web documents.

Suppose we associate the URI `http://www.orbispictus.cz/orders` with the XML document in our PurchaseOrder example above. We may indicate that the XML document is owned by David using an RDF statement with the following triple: (Subject: `http:// www.orbispictus.cz/orders`; Predicate: Owner; Object: "David"). In the functional notation we write

$$\text{Owner}(\text{http://www.orbispictus.cz/orders}) = \text{David}$$

Such RDF statements are graphically represented using ovals for resources, rectangles for literal values, and labeled directed edges for properties. RDF-graph representing our example is in Fig. 8.

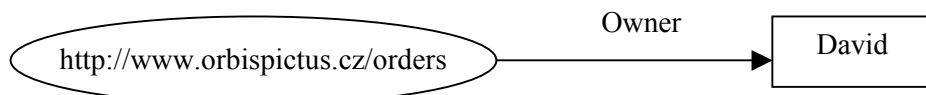


Fig. 8. RDF-graph

The value of a property is not required to be a literal such as above. It may be another resource. For example, the RDF graph in Fig. 9 indicates that the owner of the orders data is the resource identified by URI `http://datakon.cz/yan`, which has the name Yan and title Possessor.

In addition, RDF provides bags, sequence, and alternatives to express collections of Web sources. The Of property of the orders resource has the bag {books, artefacts} as its value. Symbols `_1`, `_2` denote properties of the membership in a bag. RDF also provides types sequence and alternative.

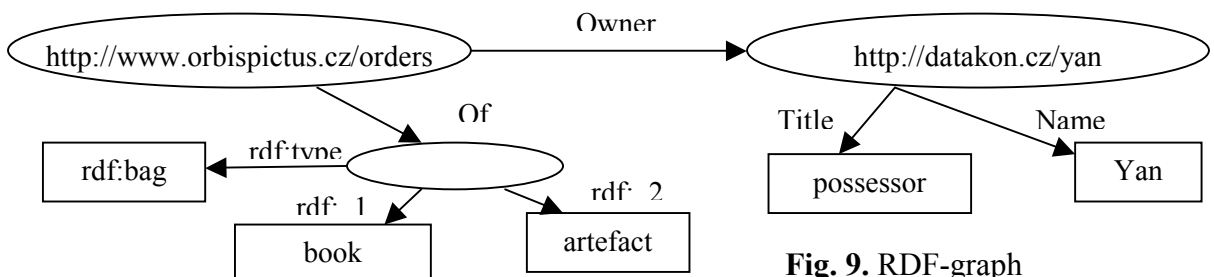


Fig. 9. RDF-graph

RDF also specifies a concrete syntax based on XML for expressing RDF statements. Fig. 10 contains a complete XML document representing the RDF graph in Fig. 9.

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax/#"
  xmlns:po="http://www.myschema.cz/#">
  <rdf:Description about="http://www.orbispictus.cz/orders">
    <po:Owner rdf:resource="http://datakon.cz/yan"/>
  </rdf:Description>
  <rdf:Description about="http://datakon.cz/yan">
    <po:Name>Yan</po:Name>
    <po:Title>possessor</po:Title>
  </rdf:Description>
  <rdf:Description about="http://www.orbispictus.cz/orders">
    <po:Of>
      <rdf:Bag><rdf:li>book</rdf:li>
      <rdf:li>artefact</rdf:li></rdf:Bag>
    </po:Of>
  </rdf:Description>
</rdf:RDF>

```

Fig. 10. XML document representing RDF-graph

In the XML document in Fig. 10 two namespaces are used: `rdf` (by RDF standard) and `po` (by our PurchaseOrder schema). RDF also allows statements to be made about other statements. This is useful when there is a need to express the properties of metadata. For example, we can describe the time period that the metadata about the ownership in Fig. 9 should be valid for etc.

The DC Initiative, an international resource description community focusing on simple resource description for discovery, has adopted RDF. An example, how to encode DC Set in RDF is described in [DCMI99b].

We conclude with encoding a fragment of DC metadata from Fig. 10 in RDF (Fig. 11). Concerning multiple subjects, instead of repeating the PropertyType, an RDF Bag is more convenient. Notice that `rdf` is the default name space.

```

<? xml version="1.0" ?>
<RDF xmlns:rdf="http://w3.org/TR/REC-rdf-syntax/#"
  xmlns:dc="http://purl.org/DC#"
  <Description about = "http://www.kocour.cuni.cz/paper1" >
    <dc:Title> Integration and interoperability of data sources: forward into
the new
      century
    </dc:Title>
    <dc:Author> Pokorny, Jaroslav </dc:Author>
    <dc:Date> June, 2001 </dc:Date>
    <dc:Subject>
      <Bag>
        <li>interoperability</li>
        <li>data sources</li>
        <li>XML</li>

```

```

    <li>RDF</li>
  </Bag>
</dc:Subject>
</Description>
</RDF>

```

Fig. 11. Encoding DC in RDF

3.3 RDF Schema

To support an interoperability of a data source, we need to define a vocabulary for RDF data. A developer should be able to specify a usage of "Owner", etc. *RDF Schema* [W3C99] provides a basic type schema for RDF based on core classes, core property types and core constraints. Three core classes are at disposal.

- *Resource* (i.e., the class of all subjects),
- *Property Type* (i.e., the class of all predicates), and
- *Class* (i.e., the class of all values of predicates).

Core property types of RDF schema are:

- *instanceOf* and *subClassOf*: *instanceOf* defines a relationship between a *resource* and an element of *Class*, and *subClassOf* defines a relationship between two elements of *Class*³. *subClassOf* is assumed to be transitive.
- *Constraint* is a subclass of *PropertyType*. It has the two core instances *range* and *domain* applicable to property types having a class as value. *Range* and *domain* define the range and domain of property types respectively.

Constraints on properties can be specified using domain and range constructs.

Fig. 12 contains our RDF data as an instance of an application RDF schema. XML serialization of RDF schema is the same as for RDF data. RDF schema expressions are valid RDF expressions.

Notice that RDF does not possess any direct mechanism for dealing general axioms, i.e. rules that allow additional reasoning. For example one could specify that *ParentOf* and *ChildOf* are inverse relations.

3.4 Comparison of XML and RDF

RDF is a framework for metadata interchange, providing a model and syntax to specify properties of resources. It does *not state* which properties need to be specified to achieve a common understanding between disparate systems. This task is left for metadata standards body like, e.g., DC. Note that the RDF Schema is similar to XML Schema in that it is also a schema specification language and can be used in validating XML documents. However, RDF Schema and XML Schema serve different purposes.

³ These properties are well-known in AI and conceptual modelling.

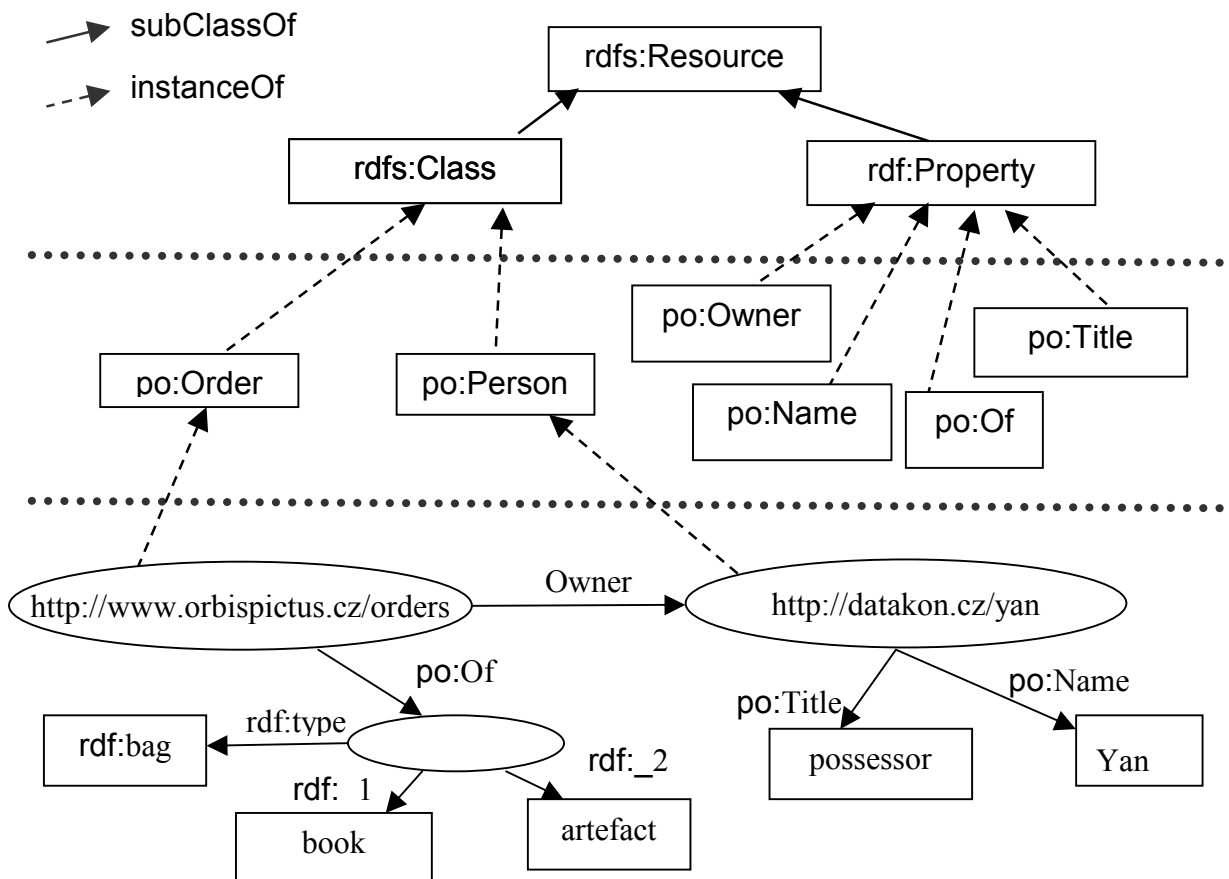


Fig. 12. RDF data, application RDF schema, and RDF Schema namespace

XML Schema places constraints on the content and structure of XML documents, whereas RDF Schema is concerned with properties of RDF descriptions, and their meanings and relationships. The main point commonly adopted to explain the difference between the two languages is that XML Schema is more a "syntactic" language concerned with how elements appear in a document, while RDF Schema is more a "semantic" language concerned with the meaning of relationships between objects and properties.

XML Schema simply sees data as XML elements and attributes, while RDF Schema provides a more structured model of data, viewing it as made up of related objects and properties.

XML and RDF are complementary technological means that will enable ontological support in knowledge management, e-business, and e-commerce. XML provides a standard serial syntax for exchanging data. In consequence, ontology-based data and information exchange can abstract from these aspects. A DTD allows us to define the structure and element tags of an XML document. The goal of the future research will be how DTD can be generated from an ontology and vice versa how an ontology can be derived from a DTD. Finally, the RDF provides a standard for describing machine-processable semantics of data. In consequence, it can be used for representing ontologies (i.e., as syntax for ontology specification).

3.4 Comparison of XML and RDF

Concerning the power of RDF model, anybody from database community can recognize its conceptual fundamentals - the well-known object-attribute-value model. Thus the problem of representing ontology will be reduced to mapping it into this conceptual framework. Initial attempts in this direction have been shown in the work of Decker et al [Dec⁺00], who used RDF schema for mapping of OIL (Ontology Interchange Language).

4. Conclusions

Without doubt the goal to integrate data from a large number of data sources on the Web is becoming a possible reality. XML revolutionizes the Web, but semantic interoperability is needed to achieve the Web's true potential. Having tools like RDF and RDF Schema for capturing more semantics about data sources, we will also need more powerful languages for describing transformations among various metadata. Fortunately, all developed languages have XML syntax. That makes it possible to use XML query languages and, obviously, all associated XML data models.

References

- [DCMI99a] The Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1. <http://purl.oclc.org/dc/documents/rec-desc-19990702>, 1999.
- [DCMI99b] The Dublin Core Metadata Initiative: Guidance on expressing the Dublin Core within the Resource Description Framework (RDF). Draft proposal, <http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/>, 1999.
- [DCMI00] The Dublin Core Metadata Initiative: Dublin Core Qualifiers. <http://purl.oclc.org/dc/documents/dcmi-qualifiers>, 2000.
- [Dec⁺00] Decker, S. et al: The Semantic Web: The Roles of XML and RDF. IEE Internet Computing, Sep.-Oct. 2000, pp. 2-13.
- [Flo⁺99] Florescu, D. Deutsch, A. Levy, A. Fernandez, M. and Suci D. A Query Language for XML. In: Proceedings of Eighth International World Wide Web Conference, 1999.
- [Gol⁺00] Goldman, R. McHugh, J. and Widom, J. Lore: A database Management Systems for XML. Dr. Dobbs' Journal, April. 2000.
- [HL01] Hunter, J., Lagoze, C.: Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles. To be published WWW10, HongKong, May 2001.
- [HP00] Heery, H., Patel M.: Application profiles: mixing and matching metadata schemas. Ariadne Issue 25, September 2000.
- [Ke00] Ketchell: Standardization in the information society – or how to avoid market anarchy. In: Proceedings of ECDL 2000, Lisbon, Portugal, 18-20 September 2000.

- [Pok00a] Pokorny, J.: XML: a challenge for databases?. 8th Int. Conf. on Information Systems, Christiansand, Norway, 2000 (to appear in Plenum Press, 2001).
- [Pok00b] Pokorny, J.: XML Functionally. In: Proceedings of IDEAS2000, B.C.Desai, Y. Kioki, M. Toyama (Eds.), IEEE Comp. Society, 2000, pp. 266-274.
- [RP00] Heery, R. Patel, M.: Application profiles: mixing and matching metadata schemas. Ariadne Issue 25, September 2000.
- [St⁺00] Staab, S. et al: An Extensible Approach for Modeling Ontologies in RDF(S). In: Proc. of ECDL 2000 Workshop on the Semantic Web. Lisbon, Portugal, September 2000,
- [W3C98] W3C Extensible Markup Language (XML) 1.0. <http://www.w3.org/TR/REC-xml>, 1998.
- [W3C99] W3C Resource Description Framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/REC-rdf-syntax/>, 1999
- [W3C00a] XML Schema Part 0: Primer W3C Candidate Recommendation 24, 2000.
- [W3C00b] W3C Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000, <http://www.w3.org/TR/rdf-schema/>, 2000