Ontology-based Semantic Metadata Validation

Vilho Raatikka[2,1] and Eero Hyvönen[1,2]

[1] University of Helsinki, Department of Computer Science
`firstname.lastname@cs.helsinki.fi`,
`http://www.cs.helsinki.fi/group/seco/`
[2] Helsinki Institute for Information Technology (HIIT)

Abstract. Much of the Semantic Web content is generated from databases, especially the instance data based on the ontology classes used in applications. A recurring problem is that the instance data does not always semantically conform to the ontology used. It may be ambiguous, incomplete, or partly erroneous. Validating the data is necessary when it is transformed to a more semantic format. This may be a difficult and laborious task given the large and complex ontologies and databases in use. This paper discusses the problem of validating existing database instance data according to an ontology. We show how Semantic Web standards can augment the syntactic data validation schemes of relational databases and XML towards semantic validation. A metadata editor for semantic validation is being implemented. Its idea is to provide an XML-based view to a relational database and transform the data into RDF instances conforming to an RDF Schema, the ontology. The tool will be applied in tranforming museum collection data into a semantically interoperable form in the Finnish Museums on the Semantic Web project.

1 Introduction

The goal of our work is to combine data from heterogeneous museum databases into a single open WWW service called *Finnish Museums on the Semantic Web* (FMS) [7]. With this system, people interested in museum collections are offered a single access point to a large national level collection of exhibits. To access the service, only a common PC with a WWW browser and an Internet connection is be needed. The idea of FMS is to combine collection data at the semantic level by using semantic web technologies. In this way the user can be provided with new semantic-based facilities for information retrieval, such as ontology-based information retrieval and semantic browsing [9].

In order to reach this goal, the collection data stored in heterogenous databases must first be harmonized. Combining data of heterogeneous data sources is a challenge, which has been widely studied [1, 14]. The problem can be addressed at syntactical and semantical levels.

On the syntactic level, naming conventions for the tables and fields of the databases may be different. Also storing formats may be different in expressing equal concepts and data. For instance, the age of a person can be either of a `numeric`, an `integer`, or a `number` type. A more difficult question is how to combine different database schemas. Syntactic and semantic harmonization entails that one is be able to tell how to find the corresponding data values in different databases, e.g., the name or the material of a collection item. This can be tricky, if the databases store semantically different data, e.g., salaries with or without taxes. In the museum case, databases record more or less the same data of similar collection objects. This is partially because museums catalog collections use standards, such as Spectrum[3] and the CIDOC Guidelines[4]. Roughly speaking, only the schema for storing the data is different, although some museums catalog more metadata than others and syntactical differences may still be remarkable. There is a great variety of different *database management systems* (DBMS) and different database *schemas* in use, and the systems run on different platforms.

Harmonization on the semantic level includes schema intergation and terminological interoperability. For example, different synonyms for the same object may be used in different museums and by different catalogers, data about artists and organisations may be cataloged by different naming conventions. There are some classification standards and controlled vocabularies addressing the terminology problem, such as OCM[5], SHIC[6], and ICONCLASS[7]. However, in practise the terminology used in the data records is herogeneous.

In this paper a solution for harmonizing museum collection data on the syntactic and semantic level is developed. We show how semantic web ontology technigues, especially RDF [10] and RDF Schema [3] (RDFS), can be used in making the database semantics explicit and for representing the database contents is a harmonized uniform way. However, during the transformation of the existing databases into a harmonized RDF repository, both syntactic and especially semantic validitation is needed. Valid semantic metadata is a prequisite for the semantic information retrieval services provided by the envisioned FMS system.

The paper is organized as follows. First representation and extraction of semantics in databases is discussed. After this the steps needed in transforming heterogeneous databases into a semantically harmonized RDF repository are discussed. In conclusion, the implementation of a tool for helping the user in this task is discussed.

2 Semantics in Databases

The design of a database starts typically by modeling a *conceptual schema* [17, Ch.7.2]. The conceptual schema is based on the entities of the data and relations between them. The relations are typically directed and labeled.

[3] http://www.mda.org.uk/spectrum.htm
[4] http://www.willpowerinfo.myby.co.uk/cidoc/
[5] http://www.silverplatter.com/newFieldGuides/hraf/Outline_of_Cultural_Materia.htm
[6] http://www.holm.demon.co.uk/shic.htm
[7] http://www.iconclass.nl/

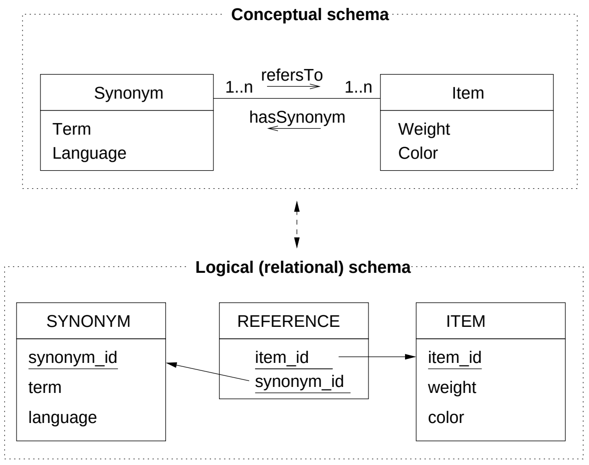


Fig. 1. The mapping between the conceptual schema and the logical schema. "Each synonym refers to at least one item and vice versa."

The modeling notation may be, for example, UML [17, Ch.7.3] and based on an *entity-relationship*-diagram (ER) [16, Ch.2]. The conceptual schema is mapped to a logical model, which usually is a *relational model* [16, Ch.3]. The entities are divided into tables as in Figure 1. Finally, the logical schema is transformed into a physical schema in which all necessary platform specific requirements are taken into account. The actual database is based on the physical model.

The semantics can be expressed in a relational schema in many ways. A table and its attributes (denoted as TABLE.attribute) are usually mapped to a class and its attributes. For example, the table ITEM can be mapped to the class with equal name and attributes ITEM.color and ITEM.weight. If an attribute refers to another table as in Figure 1, the meaning of the relation can easily be seen from the conceptual model. Referential integrity [4, Ch.6.2.4] in the relational schema is achieved by *foreign keys*. Say that there is a foreign key r in the table *T1* which refers to the attribute s in the table *T2*. The referential integrity means that for a given value of T1.r, T2.s having the equal value must exist. In Figure 1 the REFERENCE.item_id, for instance, refers to ITEM.item_id. Thus, any given value of REFERENCE.item_id must exist in the ITEM. Foreign keys in Figure 1 express that there cannot be an item without at least one synonym and vice versa. References of a database schema can also be explicitly named. The reference from REFERENCE to ITEM could be named *isSynonym*, for instance. Naming the references of a database further increases the expressiveness of a particular database schema.

A mechanism called *trigger* is used in ensuring cardinality constraints presented in the conceptual model. The trigger is launched as a consequence of a predefined action, for instance, if a row is included in the SYNONYM (see

Figure 1). The trigger performs an action, which checks that for the given SYNONYM.synonym_id, there is at least one REFERENCE.synonym_id with an equal value. That is, on the logical level the trigger may be used to force the given cardinality constraints defined on the conceptual level.

Usually the database semantics must be extracted from the logical schema because the up-to-date conceptual schema usually does not exist. Even if the conceptual schema exists, it seldom is synchronized with the logical schema after the logical schema is updated. The logical schema guarantees relatively efficient operations and good update properties. However, the semantic readability of the conceptual schema is lost in the logical model. In the *Karlsruhe Ontology and Semantic Web tool suite* [13] (KAON) the logical schema of the database is mapped straight to the ontology.

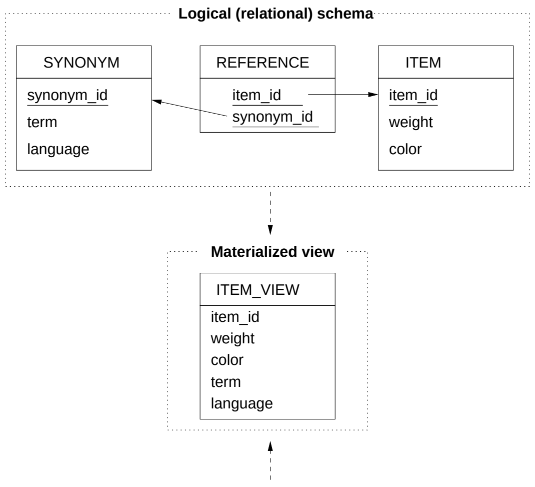
An alternative method is to re-create the conceptual schema from the logical schema. It might be useful for two reasons. Firstly, reading semantics is much easier from a conceptual schema than from a logical schema. Secondly, it is also easier to combine two conceptual models of different databases than two logical models.

It has been shown that any relational schema can be transformed into an ER-diagram if the relational schema is in an *entity-relationship normal form* (ERNF) [11, Ch.11.5]. By following certain rules in logical modeling, the database will always be in the ERNF. Transforming a relational schema to an ER-diagram does not result in a single ER-diagram but a restricted group of valid ER-diagrams. It is the user's task to choose the right one among the diagrams produced [11, Ch.11]. By using conceptual models of databases it may be possible to further automate the database mapping process.

All logical models cannot be mapped to a single concept schema. Instead, the mapping may result in a set of formally correct schemas. As an example, if the conceptual schema included a hierarchy of classes and was mapped to a relational schema, then the re-engineering of the conceptual schema would probably produce a set of models, all of which are not equal to the original schema. For this reason human interaction will be needed.

The semantics of a database application is typically embedded into both the business application and the database schema. The main goal of the database is to offer safe and efficient operations with the data. Parts of the system's semantics are stored into the database, such as entities, relations between entities and both cardinality and referential constraints. The semantics can be expressed in many ways in the database thus making the reliable extraction of the database semantics difficult.

The FMS system uses semantic web technologies in which the semantics are not embedded but explicitly represented by a set of ontologies. An *ontology* "describes a formal, shared conceptualization of a particular domain of interest" [6]. In FMS, ontologies are used for defining the semantics of stored museum data. An ontology can be described with an *RDF Schema* (RDFS) [3] specification. With RDFS one is able to define classes, class properties, basic property constraints, and inheritance between classes. For example, it is possible to state



```
<xsd:element name="item">
  <xsd:attribute name="itemId" type="xsd:positiveInteger"/>
  <xsd:complexType>
    <xsd:element name="weight">
      <xsd:complexType>
        <xsd:element="type" type="xsd:string"/>
        <xsd:element="quantity" type="xsd:positiveInteger"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="color" type="xsd:string"/>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="synonym" maxOccurs="unbounded" minOccurs="1">
          <xsd:complexType>
            <xsd:element name="term" type="xsd:string" />
            <xsd:element name="language" type="xsd:string" />
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:complexType>
</xsd:element>
```

Fig. 2. Constructing a materialized view including the data required in the XML Schema.

that the cat crossing the street is an instance of the class Cat, which in turn is a subclass of Mammal. The *Resource Description Framework* (RDF) [10] can be used to represent real-world instance data in terms of the ontology class(es) and properties. The databases to be combined are transformed into an RDF database, a knowledge base, conforming the ontologies in use.

3 Steps of Data Validation

Museum collections typically reside in protected databases to which only privileged applications have the read access. Alternatively, the data can be republished outside the database in a form that anybody can read and understand. The latter approach was chosen in the FMS system. The collection data to be published in FMS will reside in the ordinary public HTML directories of the participating museums. No privileged access rights are needed. The FMS system collects the data every now and then, and creates a central repository for information retrieval. In this way the museums can select, edit, and separate the information to be published from the database. In this way, there is no fear of the original database being corrupted by unauthorized users. An additional benefit is that the FMS system can be developed independently from the rest of the museum information systems. The public repository can be used by other applications as well.

In *relational database management system* (RDBMS) data is divided into rows. Rows are stored into tables defined in the database schema. When data is read from the database and copied for the application, it must be presented in such a format that the application can use it. In an open system such as FMS, the data transfer format must be such that it is commonly known and accepted, and that anyone can translate to it. *XML*[8] [2] fulfills these requirements and is used in the FMS project as a data transfer format. XML is an open well-known standard family with plenty of public domain and proprietary software available.

In order to publish a museum database in FMS, the following syntactic and semantic validation steps must be taken. On the syntactic level, there may be errors in the data, inconsistent usage of formats, or data missing. On the semantic level, the terminology must be disambigiated and semantic validity constraints taken into account.

Syntactic Validation

1. **Database to XML transformation** The data to be published is retrieved from the data source and transformed to an XML format locally in the museum. For this purpose, every joining museum is given the *FMS XML Schema* defined by the FMS authorities.
2. **XML Schema[9] based validation** The XML data is syntactically validated against the FMS XML Schema. The schema consists of both obligatory and optional elements. Fulfilling the obligatory requirements is the museum's own responsibility.

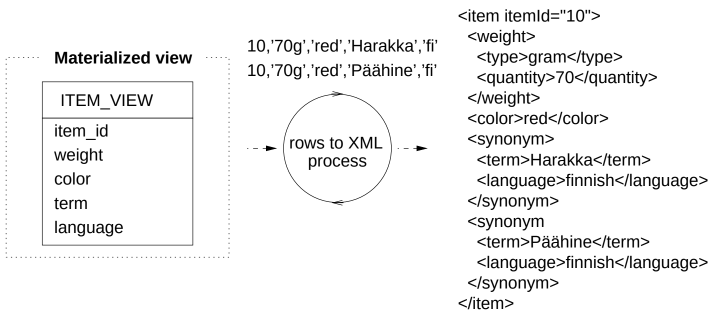
Semantic Validation

1. **XML to RDF transformation** After producing syntactically valid XML data, the records are transformed into RDF. This phase has two components: 1) The meaning of the XML elements is defined by mapping them to ontology concepts. 2) The terminology used in the XML element values is disambiguated by mapping them to RDF Schema ontology classes.
2. **RDF Schema based validation** The correctness of the RDF-statements is validated against the property constraints of the RDF Schema.

In the following these steps are discussed in more detail.

[8] http://www.w3c.org/XML/
[9] http://www.w3c.org/XML/Schema

3.1 Syntactic validation



```
10,'70g','red','Harakka','fi'
10,'70g','red','Päähine','fi'
```

```
<item itemId="10">
  <weight>
    <type>gram</type>
    <quantity>70</quantity>
  </weight>
  <color>red</color>
  <synonym>
    <term>Harakka</term>
    <language>finnish</language>
  </synonym>
  <synonym
    <term>Päähine</term>
    <language>finnish</language>
  </synonym>
</item>
```

Fig. 3. Transforming the data from the database view to the XML-document.

The data to be published in FMS must first be transformed to the XML format conforming to the FMS XML Schema. The data can be read from the database through a *view* (see Figure 2) [4, Ch.6.4], which helps in XML-transformation. A view is a virtual table, a query-able interface, which hides the physical and the logical details of the database from the user. The view constitutes of an SQL query, which may join multiple tables. The query is executed every time the view is queried. The remarkable difference to the database table is that updating a view is restricted. By rewriting the query, the database schema may be updated or the DBMS can be replaced with another without having to interfere with the application. The DBMS guarantees that the data of a view remains consistent if the original data is updated. In FMS the view should include at least all obligatory fields defined in the XML Schema.

If the query constituting the view is complicated or the view is frequently queried, the use of a *materialized views* is recommended. The materialized view is a physical structure rather than a virtual table. Using materialized views results in faster query execution than in logical views.

The data is queried through the view so that the rows are grouped by items. For each item there is a set of rows, which are combined to a single *XML card* (see Figure 3). After the translation, the document is validated against the XML Schema. If the card is valid, the process continues to the next phase, semantic validation.

3.2 Semantic Validation

The semantic validation phase has three components. Firstly, the meaning of XML *elements* is defined by mapping them to ontological structures. Secondly, the XML element *values* are mapped to the semantically corresponding ontology classes. This mapping is semi-automatic; human intervention may be needed to resolve semantic ambiguities. Thirdly, the RDF descriptions corresponging to an XML card is created based on the mappings and it is validated against the constraints expressed in the RDFS ontology.

Mapping XML Elements to Ontology Concepts The elements defined in the XML Schema must be mapped to the classes and properties of the RDFS ontology. Based on such a mapping, an XML card can be transformed into a set of RDF triplets. The mapping defines the meaning of the XML elements by a set of *mapping rules*. A mapping rule is a template of RDF triplets where *XPath*[10] expressions are used to identify the actual element values. As an example of XPath, assume the XML Schema below telling that the element `item` has a subelement `name` with a string value. The XPath referring to the element `name` is `/item/name`.

```
<xsd:element name="item">
  <xsd:element name="name" type="xsd:string"/>
  ...
</xsd:element>
```

When applying a template rule to an XML card, the XPath expressions are instantiated with matching element values. If some XPath in a rule cannot be matched, the rule cannot be applied to the card. If the rule matches, then the RDF template evaluates to a set of RDF triples where XPath expressions are substituted by the corresponding element values. For example, three template rules within the `fms` namespace are given below:

```
R1:   </item/id,        rdf:type,          /item/name>

R2:   </item/id,        fms:madeOf,        /item/material>

R3:   </item/id,        fms:hasLength,     /item/length/id>
      </item/length/id, rdf:type,          fms:Length>
      </item/length/id, fms:unitOfMeasure, /item/length/unit>
      </item/length/id, fms:lengthValue,   /item/length/value>
```

By applying them to the XML card

```
<item>
  <id>id74</id>
  <name>chip</name>
  <material>silicon</material>
  <lenght>
     <id>len7</id>
     <unit>mm</unit>
     <value>12</value>
  </length>
</item>
```

[10] http://www.w3.org/TR/xpath

the following result is obtained:

```
<id74, rdf:type,          'chip'>

<id74, fms:madeOf,        'silicon'>

<id74, fms:hasLength,     len7>
<len7, rdf:type,          fms:Length>
<len7, fms:unitOfMeasure, 'mm'>
<len7, fms:lengthValue,   '12'>
```

The rules R1, R2, and R3 describe the meaning of the elements `name`, `material`, and `length`, respectively.

Mapping Element Values to Classes The next step is *term mapping*. Here the literal subject and object values of the RDF-triples are mapped to the corresponding ontology classes. However, based on the special character of the `rdf:type` property, the literals in its domain position, id74 and len7 in the example, are unique identifiers of instances

In FMS, term mapping is based on synomyn sets, *synsets* that are attached to the ontology classes (concepts). They tell the possible classes that the literal names may refer to. For example, the class `SemiconductorChip` may have the synset {'chip'}. The mapping from names to concepts is not unique due to homonymy[11]. A syntactically valid literal may have several semantic interpretations. For example, the literal `chip` in our example may refer to a squirrel species, a semiconductor component, a kind of coin, and a piece of wood. As a result, the synset of a class `WoodChip` in addition to `SemiconductorChip` may contain the literal `chip` .

The result of applying term mapping to an RDF template is a set of RDF triples where literal data values (other than identifiers and numbers) are replaced by the URI references of the corresponding RDFS classes of the ontology. In our example, the result is two alternative sets of triplets:

```
<id74, rdf:type,            fms:SemiconductorChip>
<id74, fms:madeOf,          fms:Silicon>
<id74, fms:hasLength,       len7>
<len7, rdf:type,            fms:Length>
<len7, fms:unitOfMeasure,   fms:Mm>
<len7, fms:lengthValue,     12>

<id74, rdf:type,            fms:WoodChip>
<id74, fms:madeOf,          fms:Silicon>
<id74, fms:hasLength,       len7>
<len7, rdf:type,            fms:Length>
```

[11] A homonymous word has several meanings.

```
<len7, fms:unitOfMeasure,    fms:Mm>
<len7, fms:lengthValue,      12>
```

Validation against RDF Schema constraints The final step is to validate the corresponding RDF triple sets with the RDF Schema constraints and, if multiple interpretations still remain, select the right one by asking the human user of the validator.

There are two property constraints defined in RDFS: The *domain* constraint restricts the set of classes whose instances may have a particular property attached to them. The *range* constraint defines the set of classes whose instances can be values of a particular property. For example, the following description can be used to tell that a `SemiconductorChip` is `madeOf Silicon`:

```
<rdf:Property rdf:ID="madeOf">
  <rdfs:domain rdf:resource="#SemiconductorChip"/>
  <rdfs:range rdf:resource="#Silicon"/>
</rdf:Property>
```

In our example, using this constraint would mean that the latter RDF triplet interpretation in which the `WoodChip` is made of `Silicon` can be ruled out and the semantic meaning of the XML card is disambiguated. However, using this constraint means that it would not be possible to make another valid statement, where a `WoodChip` is `madeOf Wood`. This is of course not the intention. A disjuntion of range-domain-constraints allowing either wooden `WoodChips` or `SiliconChips` made of silicon is needed. However, such a construct is not supported by RDFS. In more advanced ontology languages, such as DAML+OIL[12], mechanisms for defining more sophisticated constraints like this are available. Another possibility is to design a special RDF(S) representation for disjuctive constraints and use it for validation.

In addition to disambiguating term mapping, semantic validation based on RDFS constraints can be used to detect semantic errors in the XML data. For example, an XML card claiming that the chip is `madeOf Water` can be detected semantically invalid. Also typos in names can be found because then the corresponding classes cannot be found (unless the typo results in a another valid class name).

RDFS constraints allow only simple validation of binary constraints. It is not possible to validate the data in a larger context by n-ary constraints, where $n < 2$. For example, assume that the length of the chip is given in liters in our example, which would be a semantic mistake. However, this cannot be detected by considering the binary constraints on hasLength and unitOfMeasure. A tertiary constraint on the length instance, its class type, and the unit of measure is needed. Liter is a valid unit but not for lengths.

The result of term mapping and constraint validation is a unique set of RDF triplets, the *RDF card*. It represents the original XML card on the semantic

[12] http://www.daml.org/2001/03/daml+oil-index

level. The union of such RDF cards, generated from the XML cards of the heterogeneous databases, constitues a knowledge base. This knowledge base is the harmonized semantic representation of the underlying heterogeneous databases.

4 Meedio: A Metadata Editor for Semantic Validation

To evaluate the ideas presented in this paper, an ontology about museum textiles is being designed in the FMS project. The ontology describes the common concepts about the textile domain so that museums can map their data sources to it. This ontology will be used as the first test case for combining collection data in the FMS system. To start with, XML data from the Espoo City Museum[13] and the National Museum of Finland[14] museum will be used. For this purpose, an initial FMS XML Schema has been specified and a database to XML transformator for the database of the National Museum of Finland has been implemented. The next step is generate RDF data conforming to the ontology.

For this work, a semantic metadata validator is needed. A first version of such a system, called *Meedio* has been implemented in a student project in the summer 2002. The goal of this project was to implement a tool by which a museum cataloger without background knowledge of XML or RDF could edit and transform XML cards into semantically valid RDF form. Meedio allows the user to manually edit the metadata, complete missing data, and add new information. The user can browse the ontologies and attach RDF statements to items or remove statements. This allows a convenient way to find and pick correct instance values for a particular property. When the user saves the edited results, the semantics of the RDF descriptions is validated against the range and domain constraints of the ontology. If conflicts occur, the user is notified about the problem. The user is responsible for any more refined semantic validation of the data. For example, to notice that the length of a `chip` cannot be 12m. The resulting valid RDF is stored to a location publicly available for the FMS system.

Since the Meedio is an instance editor, ontologies cannot be updated. This guarantees that users cannot corrupt the ontology by mistake. The management of ontologies is centralized. If one realizes that the ontology is incomplete or erroneous, one must request an update from the administrative staff of the FMS.

At the moment, Meedio is still in its infancy. For example, the term mapping function described in this paper is not finished yet. The first version of the system was implemented in Java with the help of Java Server Page technologies [5], JSP Tag libraries [15], the Apache Tomcat servlet engine[15], and HP Lab's Jena toolkit[16] [12]. Meedio is used by an ordinary web browser.

[13] http://www.espoo.fi/museo
[14] http://www.nda.fi
[15] http://www.apache.org/
[16] http://www.hpl.hp.com/semweb/

Acknowledgements

Thanks to Samppa Saarela and Kim Viljanen for fruitful comments concerning the paper. Mikko Apiola, Ari Inkovaara, Miikka Junnila, Justus Karekallas, and Pekko Parikka participated in the team that implemented the demonstrational system. Our work was partly funded by the National Technology Agency Tekes, Nokia, TietoEnator, the Espoo City Museum, and the Foundation of the Helsinki University Museum, and was supported by the National Board of Antiquities.

References

1. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, 1986.
2. N. Bradley. *The XML Companion.* Addison-Wesley, 2002.
3. D. Brickley and R. V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 2000-03-27*, February 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
4. T. Connolly and C. Begg. *Database Systems - A Practical Approach to Design, Implementation, and Management, Third Edition.* Addison-Wesley, New York, 2002.
5. D. K. Fields, M. A. Kolb, and S. Bayern. *Java Server Pages.* Manning Publications Co., 2002.
6. T. R. Gruber. A translation approach to portable ontology spesifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
7. E. Hyvönen, S. Kettula, V. Raatikka, S. Saarela, and Kim Viljanen. Semantic interoperability on the web. Case Finnish Museums Online. In Hyvönen and Klemettinen [8], pages 16–29. http://www.hiit.fi.
8. E. Hyvönen and M. Klemettinen, editors. *Towards the semantic web and web services. Proceedings of the XML Finland 2002 conference. Helsinki, Finland*, number 2002-03 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi.
9. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In Hyvönen and Klemettinen [8], pages 43–45. http://www.hiit.fi.
10. O. Lassila and R. R. Swick (editors). Resource description framework (RDF): Model and syntax specification. Technical report, W3C, February 1999. W3C Recommendation 1999-02-22, http://www.w3.org/TR/REC-rdf-syntax/.
11. H. Mannila and J.-P. Räihä. *Design of Relational Databases.* Addison-Wesley, New York, 1992.
12. B. McBride, A. Seaborne, and J. Carroll. Jena tutorial for release 1.4.0. Technical report, Hewlett-Packard Laboratories, Bristol, UK, April 2002. http://www.hpl.hp.com/semweb/doc/tutorial/index.html.
13. B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for semantics-driven enterprise applications. Technical report, University of Karlshure, 2002.
14. E. Rahm and P. A. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Microsoft Corporation, Redmond, WA, USA, 2001.
15. G. Shachor, A. Chase, and Magnus Rydin. *JSP Tag Libraries.* Manning Publications Co., 2001.

16. A. Silbershcatz, H. F. Korth, and S. Sudarshan. *Database System Concepts, third edition*. McGraw-Hill, New York.
17. J. Sowa. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.