Ontology-aware XML-Queries

Michael Erdmann[1], Stefan Decker[2]

[1]University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany
`erdmann@aifb.uni-karlsruhe.de`

[2]Department of Computer Science, Stanford University, Stanford, CA 94305
`stefan@db.stanford.edu`

Abstract.

The Extensible Markup Language is accepted as the emerging standard for data interchance on the Web. XML allows authors to create their own markup (e.g. `<Student>`), which seems to carry some semantics. However, from a computational perspective tags like `<Student>` carries as much semantics as a tag like `<H1>`. A query answering facility simply does not know, what an author is and how the concept author is related to e.g. a concept person. We investigate technologies to enrich query-answering with XML documents using background knowledge about concept-structures.

1 Introduction

The Extensible Markup Language (XML, [Bray et al. 98]) is accepted as the standard for data interchange on the Web. XML allows authors to create structured documents using their own markup (e.g. `<Student>`), which carries some semantics. However, from a computational perspective tags like `<Student>` carry as much semantics as tags like `<H1>` or `<XQ27>`, i.e. XML is often overestimated concerning its semantic capabilities. For example, an XML query asking for all `<Person>`-elements does not return any `<Student>` element, despite the fact, that every student semantically is also a person. The semantics of XML tags is typically encoded into the sending and receiving applications that handle these documents. That means, information interchange with XML needs specialized, case dependent applications, which are costly to built and to maintain.

In this paper we introduce a declarative method for defining the semantics of XML tags and thus, for supporting more cost effective creation of XML applications. For the definition of the semantics of XML tags we rely on *ontologies* that define the terminology of a domain, provide a sound semantics, and formalize relationships between the terms, i.e. provide rich background knowledge. Two different components have to be investigated:

1. **an XML query engine**, e.g. Lore [McHugh et al.99][Goldman et al. 99], XML-QL [Deutsch et al. 98][Florescu et al. 99], or XQL [Robie et al. 98][Robie 99]. This component accesses the information stored in XML documents based on query language expressions. The main purpose of an XML query engine is to provide a convenient way for data access for higher level applications.
2. **an ontology inference engine**, able to reason with the ontological background knowledge. The inference engine is used to compute the relationships between the terms defined in the ontology, and provides semantics about the domain.

We investigate different possibilities for the combination of an XML query engine and an ontology inference engine.

The rest of this paper is structured as follows: In section 2 we survey the notion of ontologies. Then we investigate the possibilities, that exist for coupling an XML query engine and an inference engine and list necessary conditions that both components must fulfill. We discuss an implementation of our approach, using XQL and a Frame-Logic inference engine. Finally we present our conclusions and describe future work.

2 Ontologies

2.1 Definition of Ontology

The notion of an ontology in computer science is based on work from the field of knowledge acquisition and reasoning. A common definition of this term stems from [Gruber 93]: "An ontology is a specification of a conceptualization." This definition stresses the importance of formality of the conceptual model. Equally important for the notion of ontology that we have is the agreement of a community of users about the terms of the ontology and their meaning. Only this agreement ensures knowledge sharing. An ontology typically contains the following components: a vocabulary of concepts or classes, often arranged in a taxonomic, tree-like structure; relationships between concepts; attributes of concepts; and a set of logical axioms that define the true assumptions about the domain, e.g. business rules.

There exist several representation languages for ontologies. Most of them are based on predicate logic, but differ significantly in the set of modeling primitives they provide, in expressiveness and computability. The view of ontologies in this paper is influenced by our experience with Frame-Logic [Kifer et al. 95], an object-oriented and logic-based language (cf. section 4.2). To give an impression of Frame-Logic and the expressiveness of ontologies in general we present an example describing (parts of) the domain of a research community in figure 1.

This ontology defines concepts like `Person`, with attributes like `name` and is-a-relationships that are indicated by

```
Person[name=>>STRING]
   Academic[university=>>University,
            publ=>Publication]
      Professor[supervises=>>PhDStudent]
      Researcher[field=>>ResearchTopic]
         PhDStudent[supervisedBy=>>Professor]
   NonAcademic
      Secretary[worksFor=>Professor]
Publication[author=>>Academic, title=>STRING]
   Book[isbn=>>NUMBER]
   Journal[article=>>Article]
   Article[journal=>>Journal, page=>>NUMBER]
   Report[org=>>Organization, no=>>NUMBER]
Organization[name=>>STRING]
   University
   Enterprise

supervises inverse to supervisedBy
author inverse to publ
article inverse to journal
```

Fig. 1 An example ontology

indentation. The axioms that hold in this domain are presented in pseudo-code for easy understanding.

2.2 Rational for Ontology supported XML Querying

It must be clearly stated that XML is solely a representation language to specify the structure of documents and thus their *syntactic* dimension. The document structure can represent some semantic properties but it is not clear how this can be deployed outside of special purpose applications. To allow for real semantic interpretation and querying of XML documents XML must be complemented by a conceptual model that adequately describes the semantics of XML tags. This role cannot be filled by DTDs or XML-Schemas [Thompson et al. 99]. Although both represent a schematic view for XML documents they only cope with the *structure* of documents and do not define an actual model of a domain what would be needed for the semantics of and relationships between terms.

Everything said for XML in general also holds, of course, for XML query languages: True semantic queries are not possible with such languages, due to the lack of a conceptual model. Thus, XML query facilities must be complemented by such models, too.

Although, existing query languages are powerful tools to retrieve the contents of documents based on the document structure, the data models of all these approaches (e.g. XML-QL, Lore, XSL, and XQL) directly reflect the document structure, i.e. its syntax. When applying a more abstract (i.e. conceptual) approach we can abstract from this structure and refer to the contents as concepts and relationships, instead, and thus, answer queries on a more appropriate level. We propose to provide the conceptual model based on an *ontology* and respective means for inferencing. Since, conceptual terms can be used to retrieve facts, the ontology is a kind of mediator between an information seeker and the XML documents. It unifies the different syntaxes/structures of these documents and can add background knowledge to the process of answering a query.

To illustrate these statements, assume the ontology given in figure 1 and the XML document containing information about a research group:

```
<?xml version="1.0"?>

<Container>
   <Report OID="book1">
      <title>Ontobroker</title>
      <author OIDREF="dfe"/>
      <author OIDREF="sde"/>
      <author OIDREF="mer"/>
      <author OIDREF="rst"/>
   </Report>
   <Article OID="pub2">
      <title>On2broker</title>
      <author OIDREF="dfe"/>
      <author>
         <PhDStudent>
            <name>A. Witt</name>
         </PhDStudent>
      </author>
      <author OIDREF="rst"/>
      </Article>

   <Professor OID="rst">
      <name>R. Studer</name>
   </Professor>

   <Researcher OID="dfe">
      <name>D. Fensel</name>
   </Researcher>
</Container>
```

Fig. 2 XML document of a research group

As can be seen, the terms defined in the ontology appear as element-types in the XML-document and thus a semantic interpretation of these XML-tags in terms of the ontology can be made, i.e. queries to this kind of documents can be enhanced by adding the background knowledge encoded in the ontology to the query answering process. For example: a query for all publications with the names of all their authors is quite cumbersome to formulate in traditional XML query languages. If the query facility would interpret the terms person, author and name semantically it would be possible to simplify this query, and at the same time enhance the quality of the answer.

3 Ontology-aware XML Queries

3.1 Combination Approaches

For realizing ontology-aware XML queries we identified three possible approaches:

- [Welty, Ide 99] propose to transform an XML document into the representation language of the ontology. Queries on the XML document are written using the query language of the ontology representation system. Using this approach the highly optimized query-evaluation strategies for XML-queries can not be used, since evaluation and representation strategies for semistructured data are not build into usual ontology representation languages. Fur-

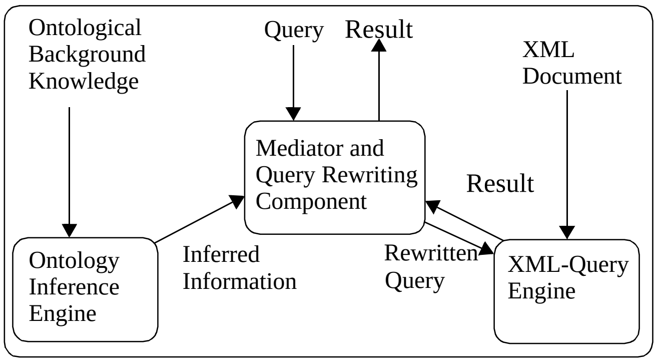
Ontological Background Knowledge
Query
Result
XML Document
Mediator and Query Rewriting Component
Result
Ontology Inference Engine
Inferred Information
Rewritten Query
XML-Query Engine

Fig. 3 Architecture of the loosely coupled approach

thermore this approach enforces a particular conceptual model for the XML document, which is not necessarily the intended one.

- The second approach is the extension of existing XML query answering systems with ontological query primitives, that results in a *close coupling* between an XML query engine and an ontology query system. The ontology inference engine is directly involved in XML query answering at runtime. This is a high effort task, since the XML query engine and the ontology query engine have to be tightly integrated. The integration leads to deep alterations in both systems.
- A third possibility is the *loosely coupled approach*. Here, besides an XML query engine and the ontology inference engine a third component is required, that mediates between the other two components. The task of the mediation component is to perform query rewriting of the XML query based on the ontological information from the ontology inference engine.

The advantages of the loosely coupled approach is that it is possible to take standard "off-the-shelf" XML query engines and ontology inference systems that are highly optimized. Since they need not be modified, the implementation takes not much effort. Depending on the specific task it is furthermore possible to easily replace one component with another, e.g. an XQL [Robie 99] component with an XML-QL [Florescu et al. 99] component, or a Horn Logic-based ontology inference engine (eg. [Decker et al. 98]) with a Description Logic-based one (eg. [Horrocks 1999]). Therefore we have choosen the loosely coupled approach.

However, these components have to fulfill certain requirements that are investigated in the next section.

3.2 Loosely Coupled Approach

The basic architecture necessary for a loose coupling between the ontology inference engine and the XML query engine is depicted in figure 3. The principle mediator algorithm imposed by the architecture is as follows:

1. The mediator receives the XML query and parses it, looking for ontology terms (concept names).
2. Then the XML query is rewritten in a semantics preserving way using information from the ontology inference engine.
3. The rewritten XML query is passed to the XML query engine, which passes back the query results to the mediator.
4. In a final (optional, depending on the application) step the results are rewritten again using the terminology from the original query and passed to the calling application.

What exactly "semantics preserving rewriting" means in this context (step 2) will be clarified in section 4.3. The architecture is generic in the sense, that several ontology inference engines and XML query engines can be used. However, they have to fulfill certain requirements concerning expressivity. Using these requirements, it is possible to judge, whether a certain ontology inference engine can be used or a certain XML query mechanism is extensible.

1. The ontology inference engine must be able to answer queries that ask for all subclasses of a given class (basically, transitive closure of a binary relationship). The relationship can be defined explicitly, via Rules (as in Frame-Logic [Kifer et al. 95]) or implicitly (Description logics).
2. Further, the XML-query engine must fulfill one of the following requirements:
   - It has either to implement the possibility to ask for the set theoretic *union* of the results of separate subqueries. E.g. if *Q* is an XML query and *Q(R/S)* is the query, where the tag-name *R* is replaced by the tag name *S*, then the query language must be expressive enough for the expression: *Q(R/S1) union Q(R/S2)*
   - Or, the second possibility is the existence of a boolean operator for the disjunction of boolean expressions, e.g. for the use in filter expression for tag names. I.e. it must be possible to filter results from an expression based on a logical *or, e.g. expr[bExpr1 or bExpr2].*

Both possibilities enable *semantics preserving query rewriting*. A semantics preserving query rewriting of a query q is a query q’ that returns all answers using the subclass semantics of the ontology inference component. Using the first possibility of query rewriting, a query has to be rewritten by generating all possible queries by replacing a tag-name by all the subclasses and combining the queries together with the union operator.

Using the second possibility, a given tag-name is replaced with the disjunction of all subclass-expressions, including the original tag.

Looking at existing XML-Query languages, eg. XQL [Robie 99] implements both possibilities, whereas XML-QL [Florescu et al. 99] just implements the second one.

4 Implementation

In the following we present an example implementation of the architecture defined in the last two sections. After assessing the given requirements, XQL and the XQL engine from GMD- IPSI[1] for the XML query language component and our own inference engine [Decker et al. 98] for F-Logic [Kifer et

1. The XQL-Engine was used by kind permission of GMD-IPSI, Darmstadt, Germany.
`http://www.xml.ipsi.darmstadt.gmd.de/xql`

al. 95] for the ontological inferencing were chosen. After briefly presenting these existing components, the rewriting component is illustrated in this section.

4.1 XQL

The main idea behind XQL (XML Query Language, [Robie et al. 98], see [Robie 99] for an updated version) is the formulation of path expressions that typically locate a set of elements to be returned as the result of a query. The syntax of XQL queries resembles the syntax of URLs and indeed is intended to be usable as part of URLs (in a similar way as XPath expressions [Clark, DeRose 99]). A simple XQL query consists of a path expression, e.g.

```
/Book/author
```

An XQL query returns, if not otherwise specified, the rightmost element in the path. The basic path syntax has been extended to support a large number of operators to widen the expressive power, e.g.

- access to attributes: `/Book/@genre`
- filter expressions that reduce the set of located elements: `/Book[@genre="scifi"]/author`
- wildcard elements that match any element type or attribute: `/Book[@*]/*`
- unrestricted nesting depths: `//Person//lastName`
- boolean operators and set operations: `/Book[@genre="scifi"] $union$ /Toy[@type="sci-fi" $or$ @type="fantasy"]`

The proposed ’98 version of XQL does not provide means for transforming the structure of an input document nor the definition of variables, which is a prerequisite for formulating joins. These and other new features are considered in the ’99 version and are nearly completely implemented in the available GMD-IPSI XQL engine. These features include:

- variables and joins: `//Book[$x := isbn] {*, //customer[.//isbn = $x]/name}`
- return operators that add certain nodes to the set of returned elements although they are not the rightmost in a path: `//Book?//author??//degree`

There exist a number of implementations for XQL; among others an implementation in PERL, and the GMD-IPSI XQL engine, which has been chosen for the research presented in this paper. We made this choice because the GMD-IPSI engine is implemented in Java, collaborates with all XML processors and is thus very portable. It enriches the XQL core by a number of features such as access to multiple documents, extensibility, regular expressions and the possibility to query HTML documents. It also supports XML as a representation format for the query *results*, which makes it a useful component in processing chains.

4.2 F-Logic Inference Engine

Usually, ontologies are defined via concepts or classes, is-a relationships, attributes, further relationships, and axioms. Therefore an adequate language for defining the ontology has to provide modeling primitives for these concepts. Frame-Logic [Kifer et al. 95] provides such modeling primitives and integrates them into a logical framework providing a Horn logic subset. Furthermore, expressing the ontology in Frame-Logic allows for queries, that directly use parts of the ontology as first class citizens (which was the requirement an ontology inference engine has to provide). That is, not only instances and their values but also concept and attribute names can be provided as answers via variable substitutions.

We use a slightly modified variant of Frame-Logic, which suits our needs. Mainly the subclassing: $C_1 :: C_2$ mechanism, meaning that class C_1 is a subclass of C_2 is used. Please note that subclasses can also be result of a rule application, thus also implicit defined subclasses are possible.

4.3 Rewriting XQL Queries

In order to improve the semantic adequacy of the results of XQL queries, these queries are transformed with the help of an ontology. This rewriting step must make certain assumptions about the relationships between the terms defined in the ontology and the structure of XML documents. One way of defining these relationships has a prescribing character that enforces a certain XML structure by deriving a DTD from a given ontology (cf. [Erdmann, Studer 99]). When rewriting XQL queries, the XML structure of the source documents needs not necessarily be valid according to such a derived DTD. The rewriting procedure is rather liberal, i.e. it only transforms those parts of the query, that have counterparts in the ontology. The XML/XQL particles and their ontological counterparts are illustrated in the following list (cf. figure 1 and figure 2 for ontology and XML data):

1. *Element types* in the XML source are mapped to *ontological concepts* iff a concept exists with the same name as the name of the XML tag.
2. Other *element types* are mapped to *attributes of ontological concepts* iff such an attribute with the same name exists in the ontology.
3. The value of an ontological attribute is defined by the contents of a corresponding XML element:
    - as PCDATA for atomic values (i.e. strings, integer ...),
    - as a subelement indicating the concept, the attribute value is an instance of, or
    - as the value of the OIDREF-attribute expressed in XPath that points to an element representing the actual value.

The rewriting mechanism in general performs two steps. Firstly, it analyses the original XQL query. Then, it inductively converts the query into a richer query that embodies ontological background knowledge. As can be seen in the above list, the most important mapping point between ontology and XML structure (or rather XQL query) are element names. Three major cases must be distinguished:

1. The element name in the query is equal to the name of a concept of the ontology.
   The subquery representing one element is substituted by a union of elements. The set of elements consists of all sub-

concepts of the concept used in the original query and is inferred from the ontology by passing a `getAllSubClasses` query to the ontology inference engine.

```
//Publication/author/Academic
   =>
//(Publication | Book | Article | Report)/
author/(Academic | Professor | Researcher |
PhDStudent)
```

The converted query returns all `<Academic>`, `<Professor>`, `<Researcher>` and `<PhDStudent>`-Elements that are subelements of an `<author>`-tag inside one of the elements representing a kind of publication (i.e. **`<Publication>`**, **`<Book>`**, **`<Article>`**, or **`<Report>`**).

2. The element name is equal to the name of an attribute of the ontology.

   In this case the subquery is substituted by a more complex expression that reflects the possibility to refer to other elements in the document in the OIDREF-attribute to denote the destination of these links as attribute values.

```
//Publication/author
   =>
//Publication/
   (author[$not$ @OIDREF] |
   (author[$x := @OIDREF]
      {//Academic[@OID = $x]} ))
```

   Here, the `author`-subquery is substituted by a disjunction asking for all `<author>`-Elements with no `OIDREF`-attribute or for all `<Academic>`-Elements[2] that are contained in the document and whose `OID`-Attribute matches the value of `OIDREF`.

3. The element name neither matches a concept nor an attribute name of the ontology.

   The subquery is left unchanged, since this part of the query cannot be enriched with background knowledge from the ontology.

The presented query rewriting algorithm makes certain assumption about the mapping from ontological terms to the XML structure (i.e. correspondence of concept and attribute names with tag-names). Other mappings are possible, e.g. conforming to Open Catalog Protocol/Format (OCP/OCF, [Martsoft 99]), where each category and product in the catalog is marked with a `<category>` resp. `<product>`-tag and is further specified by a `name`-attribute. Here, the value of the `name`-attribute has to be mapped to the concept names of the ontology. When converting a query to an OCF catalog, the rewriting algorithm has to check whether the query contains references to the `name`-attribute and translate them according to the ontology, e.g.:

```
//category[@name="Publication"]//product
   =>
//category[@name="Publication" |
      @name="Book" | @name="Article" |
      @name="Report" | @name="Journal" ]//
   product
```

2. Actually, instead of `Academic` the union `(Academic | Professor | Researcher | PhDStudent)` is inserted in the converted query.

OCF-documents contain definitions of subcategory relations, that can be seen as a kind of ontology. Rewriting queries to these documents based on an ontology, allows different users of the catalog to impose different category-hierarchies and thus define different views on the information.

To adopt our approach in the OCP/OCF scenario, simply the rewriting rules have to be changed; the principled architecture stays the same.

5 Conclusion and Related Work

Several query-approaches for XML and XML-based languages are reported in the literature. We have already discussed XQL and XML-QL, and will now focus on other approaches.

LORE (Lightweight Object Repository) [Goldman et al. 99] is a DBMS designed specifically for querying data expressed in OEM (Object Exchange Model) and XML. It does not support ontology-aware queries, but is extensible using the sketched framework in our paper.

RQL (RDF Query Language) (http://www.ics.forth.gr/proj/isst/RDF/rdfquerying.pdf) does not support general XML, but RDF. It does support retrieval based on the subclass structure, but the subclass structure is given through an RDF schema, which defines subclassOf facts between terms. Our approach enables to use reasoning instead of a fix encoded subclass structure.

[Bar-Yossef et al. 99] queries documents based on the semantic tagging, but does not consider a specific ontology, and especially not a subclass structure and does not support XML.

Because the relationship between XML structure and ontologies may be manifold, it is necessary to define several rewriting algorithms, and thus extend the applicability of our approach. As a next step in our work we plan to parameterize the XML/ontology mapping in a way that the rewriting algorithm can be easily adopted in different application scenarios.

Acknowledgments: We are grateful to Ingo Macherius and Peter Fankhauser from GMD-IPSI for fruitful discussions and support**.**

6 References

[Abiteboul et al. 97] **S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener**: The Lorel query language for semi-structured data. in: *Journal of Digital Libraries*. Volume 1, No. 1, 1997

[Bar-Yossef et al. 99] **Z. Bar-Yossef, Y. Kanza, Y. Kogan, W. Nutt, Y. Sagiv**: Querying Semantically Tagged Documents on the World-Wide Web. in: *Proc. Fourth Workshop on Next Generation Information Technologies and Systems*, NGITS'99, Zikhron-Yaakov (Israel), July 1999.

[Bray et al. 98] **T. Bray, J. Paoli, and C.M. Sperberg-McQueen (eds.):** Extensible Markup Language (XML)

1.0. *W3C Recommendation*, February 10, 1998. `http://www.w3.org/TR/1998/REC-xml-19980210`

[Clark, DeRose 99] **J. Clark, S. DeRose (eds.)**: XML Path Language (XPath) 1.0. *W3C Recommendation*, November 16, 1999. `http://www.w3.org/TR/xpath`

[Decker et al. 98] **S Decker, D. Brickley, J. Saarela, and J. Angele**: A Query and Inference Service for RDF. In: ***Proceedings of the W3C Query Languages Workshop (QL'98)***, http://www.w3.org/TandS/QL/QL98/pp.html, 1998

[Decker et al. 99] **S. Decker, M. Erdmann, D. Fensel, and R. Studer**: Ontobroker: Ontology based Access to Distributed and Semi-Structured Information. in: **R. Meersman et al. (eds.)**: *Semantic Issues in Multimedia Systems*, Kluwer Academic Publisher, Boston 1999.

[Deutsch et al. 98] **A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu**: XML-QL: A Query Language for XML. *W3C Note*, August 19, 1998. `http://www.w3.org/TR/NOTE-xml-ql/`

[Erdmann, Studer 99] **M. Erdmann and R. Studer**: Ontologies as Conceptual Models for XML Documents. in: *Proceedings of the 12th Workshop on Knowledge Acquisition, Modelling and Management (KAW'99)*, Banff, Canada, October 1999. 1999.

[Florescu et al. 99] **D. Florescu, A. Deutsch, A. Levy, M. Fernandez, D. Suciu**: A Query Language for XML. in: *Proceedings of Eighth International World Wide Web Conference* (WWW'8), 1999.

[Goldman et al. 99] **R. Goldman, J. McHugh, and J. Widom**: From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. in: *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, June 1999. `ftp://db.stanford.edu/pub/papers/xml.ps`

[Gruber 93] **T. R. Gruber**: A Translation Approach to Portable Ontology Specifications. in: *Knowledge Acquisition*. vol. 6, no. 2, 1993. pp199-221

[Horrocks 1999] [**I. Horrocks:** FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, ***Proceedings of the International Workshop on Description Logics (DL'99)***, pages 133-135, 1999.

[Ide et al. 97] **N. Ide, T. McGraw, and C. Welty**: Representing TEI Documents in the CLASSIC Knowledge Representation System. *Proceedings of the Tenth workshop of the Text-Encoding Initiative*. November, 1997.

[Kifer et al. 95] **M. Kifer, G. Lausen, and J. Wu**: Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM*, 42, 1995.

[Martsoft 99] Open Catalog Format. `http://www.martsoft.com/ocp`

[McHugh et al.99]**J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom**: Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):54-66, September 1997.

[Robie et al. 98] **J. Robie, J. Lapp, and D. Schach**: XML Query Language (XQL). in: *Proceedings of the W3C Query Language Workshop (QL-98)*, Boston, MA, December 3-4, 1998. `http://www.w3.org/TandS/QL/QL98/pp/xql.html`

[Robie 99] **J. Robie (ed.)**: XQL (XML Query Language). Working draft. August 1999. `http://metalab.unc.edu/xql/xql-proposal.html`

[Thompson et al. 99] **H.S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn (eds.)**: XML Schema Part 1: Structures. *W3C Working Draft.* December 17, 1999 `http://www.w3.org/TR/1999/WD-xmlschema-1-19991217`

[Welty, Ide 99] **C. Welty and N. Ide**: Using the right tools: enhancing retrieval from marked-up documents. in: *Journal Computers and the Humanities*. 33(10):59-84. April, 1999.