University of Alberta

A Comparison of XML Query Languages Based on W3C Requirements

by

Stanley R. M. Oliveira
Davood Rafiei

Technical Report TR 02-11
June 2002

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

A Comparison of XML Query Languages Based on W3C Requirements

Stanley R. M. Oliveira[1,2]
oliveira@cs.ualberta.ca

Davood Rafiei[2]
drafiei@cs.ualberta.ca

[1]Embrapa Information Technology, Campinas, Sao Paulo, Brazil
[2]Department of Computing Science, University of Alberta, Canada

Abstract

Various query languages have been proposed for XML, some inspired by database query languages (e.g. SQL and OQL) and others introduced by documents community. To fulfill its role of a standardization body, W3C has identified a number of requirements for XML query languages. In this paper, we examine three XML query languages and report how they conform to the query requirements set by the World Wide Web Consortium (W3C).

1 Introduction

A large fraction of data today is in the form of semi-structured data and XML has emerged as a standard to represent and exchange this data. Several query languages have been proposed for either semistructured data (in general) or XML (in particular) including Lorel [2], XML-QL [12], XQL [22], YATL [11], XML-GL [5], XSL [18], Quilt [9], and XQuery [8]. These languages are expected to provide functionalities for data extraction, transformation and integration.

In this paper, we study three XML Query Languages and evaluate their conformance to the requirements set by the W3C query working group [7]. For our study, we have selected Lorel, XML-QL, and an implementation of Quilt, namely Kweelt [23]. Our selection was affected by (1) both the simplicity and the generality of these languages and (2) the availability of free implementation prototypes. The evaluation is conducted in three different categories: a) general requirements, b) XML query data model, and c) XML query functionalities.

This article is organized as follows: related work is discussed in Section 2. An introduction to Lorel, XML-QL, and Kweelt is presented in Section 3. In Section 4, we present an analysis of these XML query languages based on the requirements of the W3C Query Working Group and report the results. Section 5 contains the conclusions.

2 Related Work

Several query languages have been developed for XML. For instance, XML-GL is a graphical query language, relying on a graphical representation of XML documents. YATL is a functional and declarative language for XML and it is not maintained anymore. Quilt is an XML query language for heterogeneous data source. Quilt borrowed features from several other query languages, such as XQL, XML-QL, SQL, and OQL. It was designed to meet the requirements identified by the W3C XML Query Working Group. XQuery [8], derived from Quilt, is the W3C working draft. Niagara [20] is a querying system providing limited functionalities for retrieving XML data and monitoring them for some interesting changes.

There is work on comparing XML query languages. Fernández et al. [15] identifies essential features of an XML query language by examining four existing query languages: XML-QL, YATL, Lorel, and XQL. The first three languages come from the database community and possess striking similarities. The fourth comes from the document community and lacks some key functionalities of the other three. Bonifati & Ceri [3] have published a comparative analysis of five XML query languages. They investigate common features among Lorel, XML-QL, XML-GL, XSL, and XQL.

Chamberlin et al. [6] specifies usage scenarios for the W3C XML query data model, query algebra, and query language. The use cases are prepared to illustrate important applications for an XML query language. The use cases along with the solutions are recently published for XQuery [8] and the W3C XML Query Algebra [14].

Our work differs from related work in several aspects. First, we do not take into account use cases because they are well defined and exploited by the W3C Query Working Group. In addition, some use cases are investigated in [15, 6]. Second, we devote special attention to the analysis of 3 XML query languages of general purpose: Lorel, XML-QL, and Kweelt. Our purpose is to investigate and compare similarities and differences of these query languages based on the requirements of the W3C Query Working Group. This approach has not been addressed in the literature.

3 A Glance at Lorel, XML-QL and Kweelt Languages

In this section, we present an introduction to Lorel, XML-QL and Kweelt. To illustrate the syntax, we use the following example: "*Selects all titles of books published by McGraw-Hill after 1995.*"

The query is posed to a set of bibliography entries with DTD described in Figure 1. Instances of this DTD can be found in `www.bn.com/bib.xml`.

```
⟨!ELEMENT bib (book* ) ⟩
⟨!ELEMENT book (title,(author+ | editor+),publisher,price)⟩
⟨!ATTLIST book year CDATA #REQUIRED⟩
⟨!ELEMENT author (last, first)⟩
⟨!ELEMENT editor (last, first, affiliation)⟩
⟨!ELEMENT title (#PCDATA)⟩
⟨!ELEMENT last (#PCDATA)⟩
⟨!ELEMENT first (#PCDATA)⟩
⟨!ELEMENT affiliation (#PCDATA)⟩
⟨!ELEMENT publisher (#PCDATA)⟩
⟨!ELEMENT price (#PCDATA)⟩
```

Figure 1. A DTD for XML documents

3.1 Lorel

General Idea: Lorel is designed for querying semistructured data and is extended to query XML documents [2]. It is a user-friendly language in the SQL/OQL style for querying such data effectively. For wide applicability, the simple object model underlying Lorel can be viewed as an extension of the ODMG (Object Database Management Group) data model and the language as an extension of OQL [4].

Main Characteristics: (a) extensive use of coercion to avoid the strict typing of OQL which is inappropriate for semistructured data [4, 1]; (b) support of powerful path expressions; (c) provision of a declarative update language and a query optimizer [19].

Syntax Example:

```
select xml(bib:{
  (select xml(book:{@year:y, title:t})
  from bib.book b, b.title t, b.year y
  where b.publisher = "McGraw-Hill" and  y ⟩ 1995)})
```

In a Lorel query, the constructor appears in the SELECT clause, patterns appear in the FROM clause, and both patterns and filters appear in the WHERE clause. In this query, bib is used as the entry point for the data in the XML document. The FROM clause binds variables to the element ids of elements denoted by the given pattern, and the WHERE clause selects those elements that satisfy the given filters.

3.2 XML-QL

General Idea: XML-QL is designed to express queries, which extract pieces of data from XML documents as well as transformations. For instance, it can map XML data between DTDs and can integrate XML data from different sources. XML-QL is declarative, like SQL, and is relational complete (e.g. it can express joins) [12].

Main Characteristics: (a) extracts data from large XML documents and constructs new XML documents; (b) supports data extraction, transformation, and integration of multiple XML sources.

Syntax Example:

```
CONSTRUCT ⟨bib⟩ {
  WHERE
    ⟨bib⟩
      ⟨book  year=$y⟩
        ⟨title⟩$t⟨/title⟩
        ⟨publisher⟩⟨name⟩McGraw-Hill⟨/name⟩⟨/publisher⟩
      ⟨/book⟩
    ⟨/bib⟩ IN "www.bn.com/bib.xml", $y ⟩ 1995
  CONSTRUCT ⟨bookyear=$y ⟩⟨title⟩$t⟨/title⟩⟨/book⟩
} ⟨/bib⟩
```

In an XML-QL query, patterns and filters appear in the WHERE clause, and the constructor appears in the CONSTRUCT clause. The result of the inner WHERE clause is a relation that maps variables to tuples of values that satisfy the clause. In this case, the result contains all pairs of year and title values bound to (y,t) that satisfy the clause. The result of the complete query is one ⟨bib⟩ element, constructed by the outer CONSTRUCT clause. It contains one ⟨book⟩ element for each book that satisfies the WHERE clause of the inner query, i.e., one for each pair (y,t).

3.3 Kweelt

General Idea: Kweelt is an adaptation of the Quilt proposal. It does try to offer an intuitive, powerful and extensible syntax to query (navigate, extract, compose, reconstruct) XML documents. Kweelt has been designed as a reference platform to perform all sort of research experiments related to XML: storage, query optimization, document output, etc [23].

Main Characteristics: (a) offers multiple XML back-ends; (b) offers an evaluation engine for the Quilt XML query language; (c) It is able to run on any Java platform and has a small footprint; (d) It is extensible, allowing one to create his/her own user-defined functions providing various template classes to make the creation of such functions very easy. More information about Kweelt can be found in [23].

Syntax Example:

```
⟨bib⟩
  FOR $book IN
    document("bib.xml")//book [@year ⟩ 1995 AND
    publisher = "McGraw-Hill" ]
  RETURN
    ⟨book year=$book/@year ⟩$book/title⟨/book⟩
⟨/bib⟩
```

In this example, the constructor appears in the FOR clause. The FOR clause binds variables to the element ids of elements denoted by the given pattern and selects those elements that satisfy the given filters. Only those tuples for which this condition is true are used to invoke the RETURN clause. In other words, the RETURN-clause is executed once for each tuple of bindings that is generated by the FOR clause and satisfies the given filters.

4 Analysis of Lorel, XML-QL and Kweelt Based on the W3C's XML Query Requirements

In this section, we analyze Lorel, XML-QL and Kweelt, taking into account the requirements that are recommended by the W3C XML Query Working Group [7].

These requirements provide flexible query facilities for extracting data from real and virtual documents on the Web. They are related to the data model for XML documents, a set of query operators on that data model, and a query language based on these query operators.

The XML Query Working Group assumes that queries operate on single documents or fixed collections of documents. Moreover, they can select whole documents or subtrees of documents that match conditions defined in document content and structure, and can construct new documents based on what is selected.

The requirements can be split into three different groups as follows: General Requirements, the XML Query Data Model, and XML Query Functionality. The following key words are used in this Section to specify the extent to which an item is a requirement for the work of the XML Query Working Group

Must: this word means that the item is an absolute requirement.

Should: this word means that there may exist valid reasons not to treat this item as a requirement, but the full implications should be understood and the case carefully weighed before discarding this item.

4.1 General Requirements

a) *Query Language Syntax*: An XML Query Language may have more than one syntax binding. The query language syntax must be understandable for humans and also be expressed in XML in a way that reflects the underlying structure of the query. Lorel, XML-QL and Kweelt have their own syntax. KWEELT has some added features, such as: in-lined, a way to embed some small XML pieces inside a query itself, statements for external Java functions and dereference hist to be able to dereference Ids with no need for any DTD/Schema information. For instance, @spouse → {Person@pid, Student@sid}/@name. This is to be understood as follows: grab the Person elements for which attribute pid is equal to the value of attribute spouse or the Student elements for which attribute sid is equal to the value of attribute spouse. Examples of the syntax of these three languages are available in Section 3.

b) *Declarativity*: The user specifies what needs to be done, rather than how it needs to be done. In contrast to conventional programming languages and scripting languages, Lorel, XML-QL and Kweelt are full declarative languages.

c) *Protocol Independence*: XML Query Languages must be defined independently of any protocols with which it is used, such as DOM, XSL, etc. Lorel, XML-QL and Kweelt are designed taking this feature into account.

d) *Error Conditions*: An XML Query Language must define standard error conditions that can occur during the execution of a query, such as processing errors within expressions, unavailability of external functions to the query processor, or processing errors generated by external functions. Although Lorel, XML-QL and Kweelt do not implement this feature, Kweelt can simulate error conditions with IF THEN ELSE.

e) *Updates*: A query language for XML must allow the addition of update capabilities if they are not supported yet. Though Lorel provides update capabilities, the other languages may be extended to support update operations. In LOREL, it is possible to create and delete object names, and create a new atomic or complex object.

f) *Defined for Finite Instances*: XML instances must validate XML documents (sequences of elements and datatypes) against a DTD/Schema for a finite number of finite instances. All three languages, Lorel, XML-QL and Kweelt support finite instances.

The summary of the general requirements can be seen in Table 1.

Table 1: Summary of the General Requirements

Requirement	*W3C*	*Xmlql*	*Lorel*	*Kweelt*
Language Syntax	Must	Yes	Yes	Yes
Declarativity	Must	Yes	Yes	Yes
Protoc. Independence	Must	Yes	Yes	Yes
Error Conditions	Must	No	No	No
Updates	Must	No	Yes	No
Finite Instances	Must	Yes	Yes	Yes

4.2 XML Query Data Model

In XML-QL and Kweelt, an XML document is modeled by an XML graph where each node is associated with an object identifier (OID); edges are labeled with element tag identifiers; intermediate nodes are labeled with sets of attributes value pairs representing attributes; leaves are labeled with values (e.g., CDATA or PCDATA). Every graph has a distinguished node called root [12, 23].

Lorel's original OEM model is modified for XML, allowing the representation of attributes and order. An XML document in the new data model is a directed, labeled, ordered graph where nodes represent data elements and edges describe both element-subelement and ID-IDREFF relationships. A node is either *atomic* containing a string value or *complex* containing a tag and a list of attribute name and atomic value pairs. There are two kinds of edges: *subelement edges* labeled with the tag of the subelement and *crosslink edges* labeled with the attribute name of an IDREFF or IDREFFS [16].

a) *Reliance on XML Information Set*: The XML query data model relies on information provided by XML processors and schema processors, such as datatypes, so that it does not require information that is not made available by such processors. Kweelt does not assume the existence of schemas or DTDs for the documents it processes. On the other hand, Lorel and XML-QL predate the XML Query Requirements and XML Infoset so that these languages did not consider such standard issues. Therefore, none of these languages support this feature.

b) *Datatypes*: The XML Query Data Model must represent both XML 1.0 character dataype indicates that the contents of an element can be interpreted as both a string and also, more specifically, as an object that can be interpreted more specifically as a number, date, etc. The datatype indicates that the element's contents can be parsed or interpreted to yield an object more specific than a string. Lorel supports coercion, a feature with the ability of implicit data casting among different types, as well as compare values represented with different type constructors, for instance scalars, singletons set, and list with only one element. XML-QL datatype is based on XML Schema specification, that is, it supports primitive types. Kweelt, on the other hand, makes use of XPath data types [10]. XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. The primary syntactic construct in XPath is the expression. An expression is evaluated to yield an object, which has one of the following four basic types: node-set (an unordered collection of nodes without duplicates), boolean, number (a floating-point number), and string.

c) *Collections*: One XML query language must represent collections of documents and collections of simple and complex values. XML-QL and Lorel do not support this feature, whereas in Kweelt, collections are based on node lists which are ordered Xpath nodesets. More details can be found in [10].

d) *References*: This includes both references within an XML document and references from one XML document to another. XML-QL, Lorel and Kweelt languages support references.

e) *Schema Availability*: Queries must be possible whe-ther or not a schema is available. If a schema is present and applied to validate a document, the data model must represent any items that they define for their instances, such as default attributes, entity expansions, or data types. These items will not be present if a schema is not present [7]. The current version of these three XML languages do not support Schema Availability.

f) *Namespace Awareness*: An XML namespace is a collection of names that can be used as element or attribute names in an XML document with the purpose of qualifying element names uniquely on the Web. Qualified names (QNames) which are element names with an understanding of the prefix-to-URI mapping and the separation of the prefix from the local part of the element name. Processors can rely on the URI for uniqueness, and ignore the prefix entirely, if appropriate to their needs. This schema mechanism is able to validate a document using multiple such URIs with respect to the schemas for all of the URIs, while DTDs cannot support this feature. XML-QL, Lorel and Kweelt are not Namespace Aware.

The summary of the XML query data model requirements can be seen in Table 2.

Table 2: Summary of the Query Data Model Requirements

Requirement	*W3C*	*Xmlql*	*Lorel*	*Kweelt*
Information Set	Should	No	No	No
Datatypes	Must	Yes	Yes	Yes
Collections	Must	No	No	Yes
References	Must	Yes	Yes	Yes
Schema availability	Must	No	No	No
Namespace aware	Must	No	No	No

4.3 XML Query Functionality

In this section, we compare Lorel, XML-QL and Kweelt based on the third group of requirements proposed by the W3C XML Query Working Group. More examples showing how these three languages support query functionalities can be found in [15, 6].

a) *Supported Operations*: As in SQL, an XML query language must support binary queries composed of union, intersection, or difference of subqueries. XML-QL supports this feature partially. It means that

XML-QL only supports union and intersection, while Lorel supports union, difference and intersection. Kweelt models an XML document as a tree of nodes. The nodes include element nodes, attribute nodes and text nodes. So, nodeset operations support union, difference, before and after [9, 23].

b) *Text and Element Boundaries*: This feature expresses simple conditions on text and requests essentially that element bouderies can be ignored in queries. A simple example of this feature is tags and white spaces. All three languages support this feature.

c) *Universal and Existential Quantifiers*: In some queries it might be useful to check whether a property holds for all elements of a collection. A universal predicate over a set of instances is satisfied if all the instances satisfy the predicate. Lorel and Kweelt support universal quantifiers while XML-QL does not. On the other hand, an existential predicate over a set of instances is satisfied if at least one of the instances satisfies the predicate. The three languages support existential quantifiers.

d) *Hierarchy and Sequence*: Queries must support operations on hierarchy and sequence of document structures. This feature is essential for every XML query language, and it is supported by these three languages.

e) *Combination*: An XML Query Language must be able to combine related information from different parts of a given document or from multiple documents. These languages support combination. For instance, in Lorel, XML-QL and Kweelt, a user can combine information collected from different portions of documents which are necessary to merge information from multiple data sources.

f) *Aggregation*: Aggregate functions compute summary information from a group of related document elements. For instance, the classical aggregates functions supported by SQL are min, max, sum, avg, and count. Lorel and Kweelt supports all aggregate functions. The aggregates functions are present in the new version of XML-QL.

g) *Sorting*: This feature is very important when one has to present sorted query results. In XML-QL this feature is specified but not implemented yet, whereas it is fully implemented in Lorel and Kweelt. For instance, a good application of sorting is the example available in section 3: Selects all titles of books published by McGraw-Hill after 1995, listed alphabetically.

h) *Composition of Operations*: This feature is able to support expressions in which operations can be composed, including the use of queries as operands. All three XML query languages studied in this work support composition of operations.

i) *NULL Values*: Every XML query language must support NULL values over all operators, including logical operators. However, none of the three languages, Lorel, XML-QL and Kweelt, support it. On the other hand, in some cases, this feature can be simulated by converting NULL values into empty strings to avoid errors.

j) *Structural Preservation*: This feature consists of ordering the resulting elements in the same way as the original document preserving the relative hierarchy and sequence of input document structures in query results. Kweelt and XML-QL produce ordered XML documents and the order can be specified to be the same as the source document. In Lorel, the retrieval documents are ordered in the same way as the original document, and the new ones are placed at the end of the document with an unspecified order among them.

l) *Structural Transformation*: As XML languages proliferate, translating XML documents between the many different sources is a vital enabler for application integration and effectiveness. XML transformation is the key to interoperability, both at enterprise and global market level. XML documents are transformed using the W3C standard, XSLT [13]. This feature is able to transform XML structures and must be able to create new structures. Lorel, XML-QL and Kweelt support structural transformation.

k) *References*: This feature is essential for every XML query language. Queries must be able to traverse intra- and inter-document references. References are present in Lorel, Kweelt and XML-QL and they are able to traverse document references in order to access element and attribute values.

m) *Identity Preservation*: Queries must be able to preserve the identity of items in the XML query data model. Identity preservation simplifies the representation of XML reference values, for example, IDREF, Xpointer, and URI values. Lorel, Kweelt and XML-QL support this feature.

n) *Operations on Literal Data*: An example of literal fragments of an XML document is:
⟨name⟩⟨first⟩Stanley⟨/first⟩⟨last⟩Oliveira⟨/last⟩⟨/name⟩,
which may be used for comparison in a query. Such literals are regarded as convenient but not essential, because such fragments may be expressed in different concrete syntaxes. Literals can be seen in the sense of constants (of a simple type). Operations on literal data are basic and all three languages support them.

o) *Operations on Names*: Queries must be able to perform simple operations on names, such as tests for equality in element names, attribute names, and processing instruction targets, and to perform simple operations on combinations of names and data. This feature is undefined in XML-QL but is present in Lorel and Kweelt.

p) *Operations on Schemas*: Schemas are emerging standards for representation of metadata related to XML documents. XML Schema improves upon DTDs in several ways, including the use of XML syntax and support for datatyping and namespaces. For example, an XML Schema allows one to specify an element as an integer, a float, a Boolean, a URL, and so on. None of the three languages support operations on schemas.

q) *Operations on Schema PSV Infoset*: According to the W3C XML Query Working Group, queries must be able to operate on information items provided by the post-schema-validation information set defined by XML Schema. As discussed in the previous item, Lorel, XML-QL and Kweelt do not support schemas. Hence, this feature is not present in these languages.

r) *Extensibility*: The basic idea of extensibility is the capacity to extend the language without touching the implementation itself. This feature allows an XML query language to support the use of externally defined functions on all datatypes of the XML query data model. The interface to such functions should be defined by the query language, and should distinguish these functions from functions defined in the query language. Neither Lorel nor XML-QL supports extensibility in this sense. On the contrary, in Kweelt users can write their own Java code (functions) and call it from the query. Such functions have to implement the right interface or extend some template (abstract) classes and must be registered inside the query using the construct: import ... as ...;

s) *Environment Information*: Through this feature, an XML query language must provide access to information derived from the environment in which the query is executed, such as the current date, time, local, time zone, or user. Neither Lorel nor XML-QL supports environment information. However, Kweelt does.

t) *Closure*: Both the input to a query and the output of a query must be defined purely in terms of the XML query data model. Similarly, query results are defined purely in terms of the XML query data model. In software systems these results may be instantiated in any convenient representation such as DOM (Document Object Models) nodes, hyperlinks, XML text, or various data formats [7]. All three languages were designed to support closure.

The summary of the XML query functionality can be seen in Table 3.

Table 3: Summary of the XML Query Functionality

Requirement	*W3C*	*Xmlql*	*Lorel*	*Kweelt*
Supported Operations	Must	Part	Yes	Yes
Element Boundaries	Must	Yes	Yes	Yes
Univ/Exist Quantifiers	Must	Yes	Yes	Yes
Hierarchy & Sequence	Must	Yes	Yes	Yes
Combination	Must	Yes	Yes	Yes
Aggregation	Must	No	Yes	Yes
Sorting	Must	No	Yes	Yes
Compos. of Operations	Must	Yes	Yes	Yes
NULL Values	Must	No	No	No
Structural Preservation	Must	Yes	Yes	Yes
Struct. Transformation	Must	Yes	Yes	Yes
References	Must	Yes	Yes	Yes
Identity Preservation	Must	Yes	Yes	Yes
Literal Operations	Should	Yes	Yes	Yes
Name Operations	Must	No	Yes	Yes
Schema Operations	Should	No	No	No
Schema PSV Infoset	Must	No	No	No
Extensibility	Should	No	No	Yes
Environment Info	Must	No	No	Yes
Closure	Must	Yes	Yes	Yes

5 Conclusions and Further Improvements

In this work we have analyzed three XML query languages, Lorel, XML-QL, and Kweelt, against the requirements of the W3C XML Query Working Group. This study revealed that none of these languages fully satisfies the requirements investigated. However, most of the requirements are available in these languages.

Lorel was developed at the Stanford University and comes from traditional database query languages (e.g. SQL and OQL). XML-QL was developed at AT&T Labs, and Kweelt was developed at the University of Pennsylvania. Unlike Lorel, XML-QL and Kweelt were inspired by XML documents.

This work also showed that although these languages were developed independently by research groups in different Universities/Centers, these three languages come from database community and possess many common features. For instance, these languages provide an excellent integration of expressive power, simplicity and performance.

XML-QL is not only a good example of simplicity, but also of expressive power as it supports querying, constructing, transforming, and integrating XML documents. Lorel provides expressive power and performance. It is the only language, among these three, that has a query optimizer for XML which provides a better performance for querying XML data efficiently. In addition, Lorel is the only language which supports update. It may be because Lorel was developed on traditional database concept. Kweelt is more complete than Lorel and XML-QL. In other words, Kweelt meets almost all the requirements proposed by the W3C XML Query Working Group. Kweelt is open-source [21] and is made available through the GNU General Public License [17].

We believe that Lorel, XML-QL and Kweelt may be applied in important domains, such as education, database research, and Web integration. All these languages are offered to the database community as an expressive contribution for those who are interested in data management on the Web area.

The similarities among Lorel, XML-QL and Kweelt are a result of a collaboration that is helpful for influencing and designing powerful general purpose query and transformation mechanisms for XML.

There are some directions for further improvement as an extension of this work. For instance, we particularly believe that it is worth comparing other languages such as Quilt, XQuery, and Niagara based on the requirements of the W3C Query Working Group, because they are also able to query XML documents from different sources, besides providing integration of expressive power, simplicity and performance. In contrast, it investigates languages such as XQL, YATL, and XML-GL may be not an expressive work because these languages do no meet many key requirements of the W3C Query Working Group.

6 Acknowledgments

Stanley Oliveira was partially supported by CNPq of Ministry for Science and Technology of Brazil, under Grant No. 200077/00-7. Davood Rafiei was partially supported by a Research Grant from NSERC, Canada. Mary Fernández and Arnaud Sahuguet answered many of our questions on the details of XML-QL and Kweelt and their conformance to the W3C query requirements. We would like to thank both for their help.

References

[1] S. Abiteboul, R. Goldman, J. McHugh, V. Vassalos, and Y. Zhuge. Views for Semistructured Data. In *The Workshop on Management of Semistructured Data, Tucson, Arizona, May*, pages 83–90, 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The LOREL Query Language for Semistructured Data. *Int. Journal on Digital Libraries*, 1(1):68–88, 1997.

[3] A. Bonifati and S. Ceri. Comparative Analysis of Five XML Query Languages. *SIGMOD Record*, 29(1):68–79, 2000.

[4] R.G.G. Cattell and D. K. Barry (eds.). The Object database standard: ODMG 2.0. In *Morgan Kaufmann, San Francisco, California, USA*, 1997.

[5] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: a Graphical Language for Querying and Restructuring WWW data. In *Proc. of 8th Int. World Wide Web Conference, WWW8, Toronto, ON, May*, 1999.

[6] D. D. Chamberlin, P. Fankhauser, M. Marchiori, and J. Robie (eds.). XML Query Use Cases. W3C Working Draft 08 June, 2001. http://www.w3.org/TR/xmlquery-use-cases.

[7] D. D. Chamberlin, P. Fankhauser, M. Marchiori, and J. Robie. XML Query Requirements. W3C Working Draft 15 February, 2001. http://www.w3.org/TR/xmlquery-req.

[8] D. D. Chamberlin, J. Clark D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery 1.0: An XML Query Language. W3C Working Draft 07 June, 2001. http://www.w3.org/TR/xquery/.

[9] D. D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. *WebDB (Informal Proceedings)*, pages 53–62, 2000.

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

[11] S. Cluet and J. Siméon. YATL: a functional and declarative language for XML. Draft March 2000. http://citeseer.nj.nec.com/cluet00yatl.html.

[12] A. Deutsch, M. Fernández, D. Florescu, A. Levy, and D. Suciu. XML-QL: a query language for XML. In *Proc. of 8th Int. World Wide Web Conference, WWW8, Toronto, ON, May*, 1999.

[13] J. Clark (editor). XSL Transformations (XSLT) version 1.0. W3C Recommendation 16 November, 1999. http://www.w3.org/TR/xslt.

[14] P. Fankhauser, M. Fernández, A. Malhotra, M. Rys, J. Simoén, and P. Wadler. The XML Query Algebra. W3C Working Draft 07 June, 2001. http://www.w3.org/TR/query-algebra/.

[15] M. Fernández, J. Siméon, and P. Wadler (eds.). XML Query Languages: Experiences and Exemplars. Draft manuscript, September. http://citeseer.nj.nec.com/254041.html, 1999.

[16] R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *Workshop on the Web and Databases*, pages 25–30, June 1999.

[17] The General Public License (GPL). Version 2, June, 1991. Available at. www.opensource.org/licenses/gpl-license.html.

[18] W3C XSL Working Group. The Query Language Position Paper of the XSL Working Group. In *Proc. of the Query Languages Workshop, Cambridge, Mass. December*, 1998.

[19] J. McHugh and J. Widom. Query Optimization for XML. In *The 25th Int. Conference on VLDB, Edinburgh, Sept.*, pages 315–326, 1999.

[20] J. Naughton, D. DeWitt, and D. Maier. The Niagara Internet Query System. Submitted for publication in 2000. Available at. http://www.cs.wisc.edu/niagara/papers/ NIAGRAVLDB00.v4.pdf.

[21] Open Source Initiative (OSI). Copyright 2001 by the Open Source Initiative. Last updated 8 May, 2001. http://www.opensource.org/.

[22] J. Robie. The design of XQL, 1999. Available at. www.texcel.no/whitepapers/xql-design.html.

[23] A. Sahuguet. KWEELT, the Making-of: Mistakes Made and Lessons Learned. Technical report, Dept. of Computer and Information Science, University of Pennsylvania, USA, November 2000.