XML query requirements

Joris Graaumans

November 13, 2002
institute of information & computing science, university utrecht
technical report UU-CS-2002-045
www.cs.uu.nl

XML Query Requirements

Joris Graaumans

Table of Contents

Abstract

We present an overview of requirements for XML query languages, gathered from three application area's: the database community, the traditional SGML/XML community, and the Information Retrieval community. Differences and similarities of these requirements are discussed, and we show to what extent XQuery and XSLT conform to these requirements.

1. Introduction

One of the promises of XML is to enhance the retrieval of information, or, as stated on the W3C website: “XML will (...) make it easier to provide metadata -- data about information -- that will help people find information and help information producers and consumers find each other. [W3C-AS]”
A query language for XML is needed in order to find information in XML documents. A large number of ad hoc query languages are proposed, for example XML-QL, XQL, and Lorel. The W3C has formed a working group to develop a query language for XML. The goal of this working group is: “to provide flexible query facilities to extract data from real and virtual documents on the Web. [W3C-AS]”
This report gives an overview of the requirements for XML query languages. With XML query language is meant an independent language with which one can query XML documents. Examples are Quilt, XQuery and XSLT. DOM applications are not considered as query languages. The requirements for XML query languages originate from three different communities: the database community, the traditional SGML/XML community, and the information retrieval community. In the next three sections a short description of each community is given.

1.1. The database community

The most well-known query language is SQL, the Structured Query language. This language works on the relational data model and was developed and standardized between 1972 and 1989. SQL is based on the relational algebra and adds functionality like update, arithmetic and aggregate operators. The most recent extension of the SQL standard, in 2000, consisted of support for full-text and images. The relational model is not always the best model to describe and preserve information. Other models used in the database community, are the object-oriented model, and, more important for the development of XML query languages, the models for semi-structured data. Semi-structured data is: “often irregular: some data is missing, similar concepts are represented using different types, heterogeneous sets are present, or object structure is not fully known. [AQM97]” Webpages and XML documents are usually considered as semi-structured data in the databases community. A number of query languages for XML have emerged from the semi-structured data community, for example, Lorel, YatL, and XML-QL. W3C's XQuery is based on this kind of languages. The database world is concerned with data retrieval. This is, according to Baeza-Yates and Ribeiro-Neto “the retrieval of items (tuples, objects, Web pages, documents) whose contents satisfy the conditions specified in a (regular expression like) user query.”[BR99]

1.2. The SGML/XML community

A query language for SGML has already been developed. This language, the Standard Document Query Language (SDQL), is a part of DSSSL, Document Style Semantics and Specification Language. DSSSL is a language that was meant to generate specific output from SGML documents. Another subset of DSSSL, DSSSL Lite, deals with the transformation of SGML documents, for example to merge documents, to generate an index or content, to perform data extraction, and so on. XML and XSLT are the simplified and broadly accepted successors of SGML and the transformation part of DSSSL. Although XSLT was never meant to be an XML query language, a large number of representative XML queries can be solved using XSLT [Lenz01-2]. Both the select part of XSLT and XQuery uses Xpath.

1.3. The Information Retrieval community

Information retrieval is: "The part of computer science which studies the retrieval of information (not data) from a collection of written documents. The retrieved documents aim at satisfying a user information need. usually expressed in natural language."[BR99] Information retrieval tries to find an answer to a certain user information need. This user need is expressed by a query. This is opposed to data retrieval. The central issue in data retrieval is to give the correct result of a query, the user information need is not considered. Relevance ranking, precision and recall are central issues of information retrieval, but are no issues in data retrieval.

2. Overview

This report gives an overview of all the XML Query requirements until February 2002. This overview of requirements can be used as a checklist for comparing query languages. Also, gaps, perplexities, and inconsistencies can be noticed. The structure of this report is as follows. First a number of general or common requirements are given. Secondly, the different points of view on XML and XML querying are discussed. What does one actually mean by XML and what are the targeted applications? Thirdly, an enumeration of the requirements of XML query languages is given. The latter is divided in six categories, namely:

- Data collection: input. What is valid input for an XML query?
- Functionality: What must the query language be capable of?
- Data collection: output. What is the result or the report of an XML query?
- Semantics: What is the meaning of a query?
- Syntax: What is the form of an query?
- Use of the query language: For whom is the query language intended?

Finally, the requirements are summarized in a table and it is indicated whether XSLT and XQuery conform to these requirements. Certain requirements are left out because they do not fall in the scope of a query language, for example:

- Transaction management;
- Security / authorization;
- XML Schema creation from non-XML data sources [Cot98];
- Conversion between XML data, relational databases and OO databases.

3. General Requirements

This part consists of requirements which are generally considered important. There is little discussion about the usefulness of the following requirements:

- The query language is non-procedural (declarative), which means that you ask what you want to have instead of how you want to gather the information;
- The query language is protocol independent [W3C-QR];
- The query language is as simple as possible and easy to implement. E.g. [Erw00][Rob98][CRF00];
- The query language supports nesting of queries;

- 'Find things in the structure, content, and linkage of hierarchical, linked document data.' [DSB98];
- The query language can be optimized.

4. Points of view

What is meant by 'XML'? What is the perspective on XML and for what purpose is XML used?

4.1. Database perspective

Deutsch considers XML mainly as a language to exchange electronic messages, for example for EDI applications. A XML document is comparable to a database, the DTD as a database schema [DFF99], and: “From the database viewpoint , the enticing role of an XML query language is as a tool for structural and content-based query that allows an application to extract precisely the information it needs from one or several XML data sources ”[DFF99-2]. The query language is relational complete and simple enough to be optimized [DFF99]. Relational complete means: the selection mechanism of the query language can produce all relationships that can be computed using relational algebra. Ives and Lu argue for SQL-like operators because XML can serve as a container of relational data [IL00]. According to Goldmann et al XML can be considered as a form of semi-structured data. Concerning the structure: “is not as rigid, regular, or complete as the structure required by traditional database management systems (...)' Furthermore, even if the data is fairly well structured, the structure may evolve rapidly ”[AQM97].

4.2. Document perspective

Document perspective is the umbrella term of the database community for all non-database related applications of XML. Two application areas are denoted with this:

1. The 'traditional' SGML/XML community. The idea of SGML, the Standard Generalized Markup Language, is that the structure of documents can be encoded independent of platform or application. Documents can be processed by machines and treated uniformly. An example of an SGML application is TEI, the Text Encoding Initiative. This international project aims at the encoding of literary and historical texts for storing and publishing purposes [TEI]. The focus of the SGML community is on modeling, exchanging, and publishing of textual documents. Examples of this kind of material include technical manuals, legal documents and historical texts.
2. The Information Retrieval community. As stated above, IR deals with information extraction from, mainly, text collections. Production and structuring of the source documents is not an issue in IR, since the documents are given. Important issues in this community are: effective searching in very large text corpora, finding all relevant documents that fit a certain user need and ranking the results. Performance issues are very important in IR, the search process in large collectives must be quick and efficient.

4.3. Database vs. document community

Fernandez says the following about the differences between the database community and the document community: “The two communities address different application areas. The database community is concerned with large repositories of data, integrating data from heterogeneous sources, exporting new views of legacy data, and transforming data into common data-exchange formats. The document community is concerned with full-text and queries of structured documents, integrating full-text and structured queries, and deriving multiple presentations from a single underlying document”[FSW99].
The impression given of the 'document community' is rather over-simplified. Study of full-text searching and combined full-text and structured text is being done in the Information Retrieval community. The traditional SGML/XML community is not only concerned with generating multiple presentations from one source document, but also with modeling and storing textual data. Dealing with very large data collections is not only an important topic in the database community. Also in the document community, particularly in IR, very large amounts of data need to be processed. Performance issues are very important in this field. Quilt, XQuery's predecessor, is designed to work with material from both the document and database community: “We also want a language that is flexible enough to query a broad spectrum of XML Information sources, and we have used examples from the database and document communities as representative of these requirements.”[CRF00].
The communities differ in another aspect than the way data is structured. There is also a difference in the way structure and content is being used: “In many structured models, the text content is merely seen as a secondary source of information which is used only to restrict the matches of structural elements. In classic IR models, on the other side, information on the structure is the secondary element which is used only to restrict text matches. For an effective integration of queries on text content with queries on text structure, the query language must provide for full expressiveness of both types of queries and for effective means of combining them.”[BR99]
The goal of XQuery [W3C-QR] is to query documents, independently of application area. To illustrate this, a number of usage scenarios for XQuery are listed:

- Human-readable documents like technical manuals;
- Data-oriented documents, like database data, object data, or other traditional data sources;
- Filtering streams, perform queries on streams of XML data to process the data in a manner analogous to UNIX filters.

5. Data collection - input

This section describes the requirements related to the input of an XML Query language. The central issue is: what is acceptable input for an XML Query? An XML Query should at least be able to work on one or multiple XML documents. Multiple documents are generally regarded as a collection of XML documents. [Mai98][Cot98][CCD99]. Besides this rather trite requirement there are a lot of supplementary requirements for the possible input for an XML Query. One of the proposals is to make it possible to query smaller parts than an XML document, namely fragments of XML [CRF00]. The W3C adopted the above ideas and made them some more abstract. The W3C Query Group requires that both the input and the output are defined in terms of the XML Query Data Model. This is called closure. [W3C-QR]. This can be real, physical XML files, but also virtual representations or XML views, see also [BMR99].

Most other requirements are related to the possibility to query something different than one or more XML documents. Maier [Mai98] for example wants to query streams of XML data. Another proposal is to incorporate local variables, like time, date, place, and user in the query [Cot98]. These requirements are also adopted by the W3C XML Query Group.
An important issue is the support for other XML related standards. There are several requirements to use Schemas or DTDs for querying purposes. An often returning requirement is that it must be possible to query both well-formed and valid XML. [CCD99][Mai98][W3C-QR][Cot98]. Querying without schema means that XML queries must be usable without knowing the exact structure of the data [DFF99]. This is called untyped querying. One must be able to use the document schema to formulate [BC00], validate [Cot98][Mai98], or optimize [Erw00] a query. This is called typed querying.
Other standards which are often mentioned are Xlink and Xpointer [Cot98][IL00][Mai98][BC00][IL00]. Maier states that queries must be able to follow Xpointers and Xlinks and that the query language must be Xpointer and Xlink cognizant. Fernandez proposes special operators to make it easier to deal with references [FSW99]. It is remarkable that the W3C XML Query group does not mention Xlink or Xpointer: "The data model must include support for references"[W3C-QR]. The way this is done and whether Xlink and Xpointer will be supported is not discussed. Not every information item in an XML document is important for every application. For most database applications, for example the order of elements, does not really matter. The possibility to query an XML document both ordered (that is, the document order is important) and unordered (that is, the document order is left aside) is a requirement from the database community [CCD99][DFF99]. Optimization may be also easier when order is ignored.

6. Functionality

Most XML Query requirements are related to the functionality of the language. The requirements are roughly divided in the same way Maier did in [Mai98], namely:

1. selection and extraction,
2. restructuring or transformation,
3. combination,
4. updates.

6.1. Selection and Extraction

Selecting a document or document part, based on certain properties, is the most important task for a query language. In IR the focus is on relevant documents within a certain collection, while in the database community the focus is on finding data within a database. This distinction is less obvious when one queries XML documents, for examples because of the possibilities of linking, the use of different schemas in a document collection and the textual nature of XML. Selection is described in the following way: "Choosing a document or document element based on content, structure or attributes"[Mai98]. Descriptions of the same kind can to be found in [DFF99][BMR99][DFF99-2][BC00][FSW99]. In the next section a discussion follows on what an XML Query language must be able to select based on content and structure.

6.1.1. Content

Content is not defined in the papers mentioned above and the concept is used ambiguously. The XML 1.0 recommendation gives the following definition:

- "Content: The text between the start-tag and end-tag is called the element's content."
- "Text: A parsed entity contains text, a sequence of characters, which may represent markup or character data. "

[W3C-XML] Content, text and data are often used as synonyms and most of the time these terms denote something like: the XML document without the markup. However, according to the definition of content in the XML 1.0 specification, is markup a part of content. 'Find information based on content and structure' is a requirement often stated in papers about XML Query languages. The concepts 'content' and 'structure' are used in a different way than in the XML 1.0 recommendation. *The Web Content Accessibility Guidelines 1.0* explains the distinction between content, structure and presentation in the following manner: "The content of a document refers to what it says to the user through natural language, images, sounds, movies, animations, etc. The structure of a document is how it is organized logically (e.g., by chapter, with an introduction and table of contents, etc.). An element (e.g., P, STRONG, BLOCKQUOTE in HTML) that specifies document structure is called a structural element. The presentation of a document is how the document is rendered (e.g., as print, as a two-dimensional graphical presentation, as an text-only presentation, as synthesized speech, as Braille, etc.) An element that specifies document presentation (e.g., B, FONT, CENTER) is called a presentation element."[W3C-WAC]
An example:
<h1>Sailboat</h1>
Here "Sailboat" is the content and the h1-tag the structural element.
In Information Retrieval the text is the primary source of information and this community raised the most requirements for text operations. The most basic manner to search in a text or a collection of texts is by means of full-text keyword search [FKM00][CFP00]. Full-text means that all words in the document are accessible, and stopword elimination is not performed. The use of wildcards are additional requirements in strings and proximity search. An example of proximity search combined with wildcards based on words: "[Find] elements in which "rose*" and "sweet*" occur within 10 words of each other"[MR98]. Proximity search is also mentioned in [Cot98], [Erw00] and [DSB98]. The latter distinguishes between proximity between words, sentences and elements. [CFP00] wants to be able to proximity search based on semantics and based on structure.
In *XML Query Use Cases from the Library of Congress*[A01] a lot of practical requirements related to text searching for XML are mentioned. Besides proximity, there is also the wish to find words in a certain order. Furthermore, the need for wildcards are further specified. Wildcards should not only be allowed as a suffix, but also as a prefix and within strings. It should be possible to restrict the number of characters that can be replaced by a wildcard. For example " sweet*<3" means that * can be replaced by at most three characters. Another required text operation is stemming. The following example is given: "Find all bills containing words stemmed from revoke"[A01] The "stem" operator must have access to specialized indexes. Both [DSB98] and [A01] propose the possibility to ignore certain information, which is about the possibility to perform case-insensitive search, and ignoring punctuation marks. Most of these requirements can also be found in [FG01]. In [A01] the requirements mentioned above are regarded as data retrieval, while others might want to use the term text retrieval. The requirements formulated by the W3C XML Query Group related

to text search are very general and less ambitious: “Queries must be able to express simple conditions on text. Including conditions on text that spans element boundaries.”[W3C-QR].

6.1.2. Structure

An XML document is structured as a labeled tree. Support for regular path expressions is very useful for searching in unknown structures [DFF99][CCD99][AQM97][IL00][Rob99]. The advantage of using regular path expressions is that it is not really necessary to know the whole structure of the document in advance. Fernandez et al. provides the following example [FSW99]: “[Select] all section or chapter titles containing the word ``XML", regardless of the nesting level at which it occurs.”
Bonifati and Ceri want to use wildcards in path expressions [BC00], and several people argue for pattern matching to find data [CCD99][Erw00][Ste00]. Other useful required operators are universal and existential quantifiers. In addition, Fernandez et al. plea for external defined functions [FSW99]. The following structural categories are distinguished: hierarchy, sequence and position, links, names, and datatypes.

6.1.2.1. Hierarchy

An XML query language needs to be able to express both parent/child and ancestor/descendant relationships, for example found in [Rob99] and [DSB98]. The latter gives the following example: “Get last-sibling-or-self”.

6.1.2.2. Sequence and position

An XML Query language should be able to use "before" and "after" conditions. That is, an XML query language should be able to express conditions on the relative document order of nodes. [Rob99][BC00]. These nodes can be siblings as well as nodes that precede in document order. Combined before and after conditions can be shortened to range qualifiers [GMW99]. An example of a condition on position is “Get nth-sibling”. Where n is a positive integer [DSB98].

6.1.2.3. Links

Links in XML documents can be realized in different ways:

1. Within a document using the attribute types id, idref and idrefs.
2. Between documents using DTD specific links. For example by means of an element <lnk>
3. By means of the Xlink standard.

The W3C XML Query Requirements says the following about linking: “Queries must be able to traverse intra- and inter-document references”[W3C-QR]. The W3C does not state the Query language should support the Xlink standard, as one would expect. The support for linking is often raised, for example in [IL00] and [Rob99]

6.1.2.4. Names

An XML document is structured like a labeled tree. Therefore, people want to perform operations on the labels. The W3C state: “tests for equality in element names, attribute names,

and processing instruction targets, and to perform simple operations on combinations of names and data. Queries may perform more powerful operations on names "[W3C-QR]. Using wildcards in tagnames is one example of a condition on the labels. Similar requirements can be found in [Erw00][DFF99][Rob99] and [CRF00].

6.1.2.5. Datatypes

The nodes of an XML tree are not only named, they can also be typed. The number of datatypes that can be used with DTDs is limited, but the W3C XML Schema standard provides a large number of datatypes. A requirement from the W3C XML Query Group is that the query language: "Support operations on all datatypes represented by the XML Query Data Model"[W3C-QR]. XSLT 2.0, XPath 2.0 and XQuery 1.0 share the same data model. The data model serves two purposes:
"First, it defines precisely the information contained in the input to an XSLT or XQuery processor. Second, it defines all permissible values of expressions in the XSLT, XQuery, and XPath languages. A language is closed with respect to a data model if the value of every expression in a language is guaranteed to be in the data model. XSLT 2.0, XQuery 1.0, and XPath 2.0 are all closed with respect to the data model. " The W3C XML Query Group states explicitly that null values should be supported. According to Ives and Lu, and Fernandez et al. problems can arise when relational tables with NULL values are mapped to XML [FSW99][IL00]. Another requirement is the possibility to extend the fixed set of datatypes: "XQuery should have an extension mechanism for conditions and operations specific to a particular datatypes. I am thinking mainly of specialized operations for selecting different kinds of multimedia content."[Mai98][DFF99-2][BC00].
When a query language supports an extension mechanism for datatypes, it is necessary to find out what datatypes can be compared with each other. The conversion from one datatype to another is called type casting. Explicit type casting is useful, but: "Implicit casts can, if carefully designed, make a language far easier to use, though they risk encouraging imprecision and uncertainty about the precise meaning of expressions. "[DSB98].
Lorel implements type coercion (that is some form of implicit type casting) to relieve the user from the 'strict typing' in OQL [AQM97] This can also be found in [BC00]. Fuhr and Grossjohann in their paper about XIRQL, XML Query language for Information Retrieval, take the implicit type casting one step further: "There should be a way to express vague searches for these datatypes". Examples of this kind of predicates are: "similarity with respect to sounding, classification (vague equality), additional comparison operators should be provided: near, broader, narrower and related for terms from a classification or thesaurus."[FG01]. In short, there is still a lot of discussion about the advantages and disadvantages of implicit and explicit type casting. Furthermore, there is a wish that a query language should be able to deal with new datatypes.

6.2. Transformation

Restructuring of data is both in the database community as well as in the traditional SGML/XML community of great importance. The result of a database query is a new instance of the data model. This instance can be serialized in many different ways, for example as a new table, as a new document or as an update of the existing data. In the traditional SGML/XML world transformation is necessary to produce one or more representations of one source document. In IR restructuring is not an important issue, since the result of a query is very often a pointer to the information source. Transformation is described as follows:
"Queries must be able to transform XML structures and must be able to create new

structures"[W3C-QR]. This is an often stated claim, see for example [Mai98][CCD99][GMW99][DFF99][CS00][Erw00].
Besides the construction of structures mentioned in the preceding paragraph, the following operations should be possible:

- *Sorting*: sort the result in alphabetical order, mentioned in [BMR99][W3C-QR][BC00][FSW99]. The concept of relevance ranking from the IR can be understood as a form of sorting.
- *Unorder*: transform a sequence to a non-ordered collection [BMR99].
- *Flattening*: create a collection of singletons from a collection of collections [BMR99][Rob99][Erw00].
- *Grouping*: it should be possible to group information items together [BMR99][BC00][FSW99]. Deutsch et al. and Ives and Lu propose to achieve this by means of skolem functions [DFF99][IL00].
- *Aggregate functions*: There is a need for typical SQL aggregate functions like min, max, sum, count, and avg [DFF99][CCD99][BC00]. Erwig observes that such functions are hard to define in XML, because of the lack of numeric datatypes in DTDs [Erw00]. This is no longer an issue when a query language supports a type system like W3C XML Schema Part 2: Datatypes.
- *Preserve order and association*: the result of a query should not always be transformed. It is also of importance that the result of a query can be a non-transformed document or document fragment [Mai98][Rob99][Rob99-2][DFF99-2][BC00][W3C-QR][FSW99].
- *Reduction*: By this is meant the removal of selected sub-elements of an element [Mai98][BC00]. Other terms that denote the same concept are filtering and pruning. Something similar is the DISTINCT operator, named in among others [BMR99]. This operator removes duplicates in a collection. Pruning is mentioned in [CS00] and [IL00].

6.3. Combination

A XML Query language must: "Combine related information from different parts of a given document or from multiple documents"[W3C-QR] Similar requirements are stated in among others [Mai98] and [BC00]. This is a typical database requirement. Querying different sources is also important in IR, but combining information from multiple sources to one result is not an issue in IR. This kind of operation can be understood as a form of transformation or restructuring. Deutsch et al. state that an XML query language should be relational complete [DFF99]. Therefore the language must, among others, be able to express relational joins. Joins are demanded by [Cot98][Rob99][Erw00][IL00][FSW99]. More specific are the following requirements: outer joins by [DFF99], inner join and left outer join by [IL00], the equi-join by [BC00] and by [BMR99] one-to-one, one-to-many, many-to-many.

6.4. Update

The possibility to update data is a database requirement. An update can be interpreted as a specific form of serialization of a query result. Instead of making a new XML document (or view, or whatever kind of instance), the source document is adjusted. Commonly required operations are delete and insert [Cot98][CCD99][AQM97][BC00][CFP00]. The W3C does not exclude an update operator, but it will not be added to XQuery 1.0. At this moment

serialization of the result is a task of the software system, not of the query language. Tatarinov et al. propose to extend XQuery with the following operators:

- Delete (child);
- Rename (child, name);
- Insert(content), InsertBefore(ref, content) /InsertAfter. Also mentioned in [AQM97];
- Replace(child, content);
- Sub-Update (patternMatch, predicates, updateOp) [TIH01].

7. Data collection - output

The most often, and most obvious, stated requirement is that the result of a query must be an XML document [Mai98][DFF99-2][BC00]. The W3C gives a more formal definition, namely the result of an XQuery must be an "instance of the XML Query Data Model". Deutsch and Ceri want to compute the DTD of the result [CCD99]. DeRose is more detailed about the report: “In interactive query languages, a query against a document database must locate the set of locations satisfying the query and may return a document, a set of documents, a list of documents, a list of locations within documents, a set of document fragments, or perhaps even an <XLink> element with a list of Xpointers to the locations satisfying the predicate.”[DSB98]
Robie states that (XQL) queries “may return any number of results, including 0.”[Rob98]. As appears from most examples of XML queries and from the W3C's XML Query Use Cases, the result of a query is mainly considered as one well-formed XML document. There are very little examples of queries where the result consists of multiple documents or list of documents. The IR community offers a few additions to the requirements mentioned above. These requirements are related to relevance ranking, precision and recall. Schlieder says: “An XML query engine should retrieve the best results possible. If no exact matching documents are found, results similar to the query should be retrieved and ranked according to their similarity.”[Sch01] Similar requirements are expressed by [CCD99][CFP00][FG01].

8. Semantics

A query language should have precise, formalized semantics. This can be achieved by mapping the language to an existing, well-defined programming language, like ML [Mai98]. The XML Query Group defines the semantics of XQuery in the XML Query Data Model. This model is based on the XML Information Set and is namespace aware [W3C-QR].

9. Syntax

A lot of different query languages are proposed, and a lot of variation exists between these proposals. The choice for a certain syntax seems to be based on the personal preferences and background of the designers. In this section the most important proposals are discussed.

9.1. SQL/OQL-like syntax

XML-QL uses a syntax that combines characteristics from traditional query languages like SQL and OQL, and XML syntax. This form is adopted by Quilt [CRF00] and via Quilt by

XQuery. An SQL/OQL like syntax is handy, because it is well-known: “(...) Lorel, has a familiar select-from-where syntax and is based on OQL”[GMW99]
Fernandez describes in [FSW99] the common structure of a database query as follows.

The query consists of three parts:

1. a pattern clause;
2. a filter clause;
3. a constructor clause.

The information between these three clauses can be modeled as a relation with a flat and unordered structure.

9.2. XML syntax

There is little discussion about the usefulness of an XML syntax for an XML query language. An advantage of using XML syntax is that the query language and XML will be mutually embeddable [DFF99-2][Mai98]. Another advantage is that queries can be processed by XML processors. The XML Recommendation states as 6th design goal: “XML documents should be human-legible and reasonably clear.”[W3C-XML]
The argument that XML syntax enhances the readability of queries does not hold. XML representations of typical programming language constructions, like "if then else", tend to be very verbose and laborious. There are a number of proposals to create multiple syntaxes of a query language. One syntax that can be easily read and written by human users and one XML representation of this syntax to be processed by machine users. See also [Mai98][DFF99][CRF00][W3C-QR]. A number of proposals were made to base the query syntax on an XML related syntax. DeRose proposes to base the syntax on Xpointer [DSB98]. Reasons for this is that this syntax is well-known and stable. Furthermore deals Xpointer very well with ordered trees and a number of implementations already exist. Another example is XSL. XSL is mentioned in [Cot98]. Lenz takes this one step further: not only the syntax of XSL should be used, XSLT in itself can be used as a query language. He claims that XSLT is an excellent query language for certain applications. Another advantage is that XSLT is a proved and stable standard which is used very much in practice. See also [Lenz01-2] and [Lenz01]. Van der Steen proposes to use a pattern grammar of which the syntax is an extension of XML [Ste00]. For the selection part of a query language Xpath is an important candidate. XQL is an extension of Xpath [Rob98]. Quilt [CRF00] and XQuery are using the syntax Xpath for the selection part.

9.3. Graphical syntax

A number of people advocate a graphical query language for XML. An example is XML-GL [CCD99]. Another representative can be found in [Erw00]. In his paper Erwig pleads for a form-based representation of XML. He claims such a syntax to be the most user friendly and intuitive.

9.4. Other proposals

The requirement to embed queries in URL's can be found in [Cot98][Rob98]. Strangely enough, this requirement can also be found in the W3C Query requirements. It is impossible

to embed and XQuery in an URL, and there no indications that this will become possible in the future. James Clark proposes to use the same element constructors for both XSLT 2.0 and XQuery 1.0. The very few differences that exists between both syntax variants can be very confusing for users of both languages. [Cla01]

9.5. Remarks

All design decisions about syntax seem to be made for 'the best interest of the user'. In [FSW99] for instance ten essential queries are being solved with four different query languages. An indication is given of the most natural and simple solutions. The criteria for these indications remain implicit. In seven of the ten cases the most natural solution is made by XML-QL and this is the language developed by the authors of the paper. The best interest for the user seems to be in most cases rather the best interest for the developer. Research of the usability of query syntaxes could solve this matter.

10. Use of the query language

In this section the requirements which refer to the future users of an XML Query Language are stated. There are two main points of view concerning the future users of XML Query languages:

1. Users are end-users, that is, the query language will be directly used by people
2. People will use the query language by means of an GUI or application.

Within several query language proposals it is not always clear which point of view is taken or that both kind of usage is possible. Robie states in the design goals of XQL that “XQL shall be easy to type and read”[Rob98]. Something comparable can be found in the requirements of the W3C XML Query Group: “the XML Query syntax must be convenient for humans to read and write”[W3C-QR]. These requirements are very vague. It is unclear what is meant by 'easy' and 'convenient' and furthermore it is not clear what kind of users those 'humans' are. Are these information experts, programmers, database experts, librarians, or naive users? More explicit points of view can be found in [CCD99][Erw00][A01]. They state that a query language also is intended for end-users, and that both novices and experts should be able to use the language. The opposed point of view is taken by Maier and Deutsch. They think that a query language is not suitable for end users, and that an intermediate layer is necessary to use the language: “XQueries should be amenable to creation and manipulation by programs. Most queries will not be written directly by users or programmers. Rather, they will be constructed through user-interfaces or tools in application development environments”[Mai98][DFF99-2]. The same view about the interface can be found by ???. Another option is to create different query languages for different user groups. This is mentioned in [Cot98]. Robie says it is not desirable to develop two or more different query languages, for example one for data and one for documents. “In a very real sense, traditional documents are data found in a format that users can use, but which is difficult for programs to process in any meaningful way; similarly, traditional data is data found in a format convenient for computers, but less convenient for human beings. When human-readable documents are marked up using XML, the data in those documents is made available for processing by both programs and query languages”[Rob99-2]. This is partly true, but the question remains if people who are interested in 'human-readable' documents benefit from an database-like type system, syntax and database-oriented optimization. Fuhr and Grossjohann for example note: “Most datatypes defined in XML Schema are useless for the document-oriented view. In order to support IR in XML

documents, there should be a core set of appropriate datatypes (text, classification schemes, thesauri and person names)"??? The idea to derive a domain specific query language from a more universal query language might be very useful.

11. Comparing XQuery and XSLT

In the introduction of this report it was stated that the collection of requirements gathered here could serve as an checklist to compare different query languages. In the following table a short start is made by comparing XQuery and XSLT. The requirements are sorted in the order of this report.legend:

- Y = Yes
- N = No
- U = Undecided
- P = Partly

Table 1. Comparing XQuery and XSLT

XML query requirements	**XQuery**	**XSLT**
General Requirements		
non-procedural, declarative	Y	Y
query a broad spectrum of XML Information sources	U	U
protocol independent	Y	Y
suitable for server-side processing	Y	Y
small, implementable, as simple as possible	U	U
nested queries	Y	Y
find things in structure, content, and linkage of hierarchical, linked document data	Y	Y
Optimizability	U	U
Database perspective		
SQL-like operators	Y	N
Relational complete	Y	Y
IR perspective		
Relevance ranking	N	N
Full-text searching	P	P
Searching in very large corpora	U	U
Input		
Collection of XML documents	Y	Y
Access to document fragment	N, only via document root	N, only via document root
Stream	N	N
Local variables, like time, user, etc.	N	N
Well-formed XML	Y	Y
Valid XML	Y	Y

XML query requirements	**XQuery**	**XSLT**
Input - support XML standards		
Xlink	N	N
Xpointer	N	N
References	P, id, idref, idrefs	P, id and idref
Input - data specific		
Ordered and unordered XML input	Y	Y
Output		
XML document	Y	Y, other formats possible
Calculate DTD/ Schema of result	N	N
Set or list of documents	N	N
Set of document fragments	Y	Y
List of locations	Y	Y
Xlink element plus Xpointers	Y, if element is explicitly constructed	Y, if element is explicitly constructed
Retrieve the best possible results	N	N
Similar results	N	N
Relevance ranking	N	N
Functionality - selection		
Choosing a document (or part) based on content and/or structure	Y	Y
Regular path expressions	N	N
Wildcards in path expressions	Y	Y
Universal and existential universal quantification	Y	N
Support for externally defined functions	N	N (req. XSLT 2.0)
pattern matching	Y	Y
Functionality - selection - content		
Keyword search	Y	Y
Wildcards in strings (prefix, suffix, restricted number of characters)	N	N
Stemming	N	N
Proximity search	N	N
Word order	N	N
Case insensitive search	N	N
Ignoring punctuation marks	N	N
Ignoring accent marks	N	N
Find text that spans element boundaries	Y	Y

XML query requirements	**XQuery**	**XSLT**
Functionality - selection - structure		
Hierarchy: parent-child relationships	Y	Y
Hierarchy: ancestor - descendant relationships	P, descendant axis	Y
Sequence and position: before - after on nodes	Y	Y
Sequence and position: range qualifiers	Y	N, not explicit
Links: intra document, id, idref, and idrefs	Y (idrefs: N)	Y (idrefs: N)
Links: DTD / Schema specific	Y	Y
Links: Xlink	N	N
Names: test equality in element names, attribute names and PI targets	Y	Y
Perform simple operations on combinations of names and data	U	U
Tag variables	Y	Y
Datatypes: all datatypes defined in XML Schema	Y	N, requirement for XSLT 2.0
NULL values	N	N, requirement for XSLT 2.0
Extension mechanisms for operations specific to particular datatypes	N	N
Explicit type casting	Y	N
Implicit type casting	N	N
Vague search on datatypes	N	N
Support for new datatypes	N	N
Functionality - transformation		
transform XML structures	Y	Y
create new XML structures	Y	Y
Sorting	Y	Y
Unorder	Y	Y
Flattening	Y	Y
Grouping	Y	N, requirement XSLT 2.0
Aggregate functions	Y	N, just a count() function
Preserving order and association	Y	Y
Reduction	Y	Y
Distinct operator	Y	N, requirement XSLT 2.0

XML query requirements	**XQuery**	**XSLT**
Functionality: combination		
Combine information from different parts of the document or from multiple documents	Y	Y
Joins	Y	Y
Functionality: updates	N	N
Formal semantics	Y	Y
Based on another language	N	N
Based on the XML Information set	Y	Y
Support for namespaces	Y	Y
Syntax		
SQL/OQL like syntax	Y	N
XML syntax	Y	Y
Based on XML related standard	P on Xpath (version 2.0)	P on Xpath
Graphical syntax	N	N
Embeddable in URL	N	N
Syntax is easy to write and read by humans	U	U
Use of the language		
Support for both novice and expert users	U	U
Accessible via GUI or other program	Y	Y

12. Conclusions

A number of separated communities have become more closely related thanks to the development of XML. Because of this, the differences as well as the similarities of the database community, the information retrieval community, and the traditional SGML/XML community have become more explicit. Every application community states different and sometimes opposite requirements to XML Query languages. Also, every community uses its own terminology and sometimes this causes confusion of tongues when the same terms are used for different concepts. The terms content, data and text are often used as synonyms, but these terms denote not always the same meaning. Baeza-Yates differs between data and information retrieval [BR99], while Deutsch uses these terms for the same concept [DFF99-2].
The traditional SGML/XML community, as well as the database community, as well as the IR, may profit from an universal XML Query language. The W3C assigned itself to develop an universal applicable XML query language. Based on this report the following conclusions can be drawn about the XQuery proposal of the W3C.

1. The XML query requirements originating from the Information Retrieval community can not, or can hardly, be found in XQuery. There is no possibility for relevance ranking and approximately search. Support for full-text searching are very feeble. The claim that XQuery is suitable for IR applications of XML documents is therefore futile.
2. XQuery does not provide full support for XML related standards like Xlink and Xpointer. In particular, XQuery doesn't have operators to follow Xpointers and Xlinks.

3. It is hard to decide whether XQuery is suitable to query a 'broad spectrum of XML Information resources', because of the following reasons:
    - There is not an exact definition given what 'different XML information sources' are.
    - The W3C XML Query Use Cases are examples to demonstrate XQuery. Not all of these examples are representative of practical, real life search problems. The fact that every signal XML Query Use Case is solvable with XQuery revealing enough.
4. The syntax of XQuery is based on typical database query languages. This is motivated by the claim that this syntax is the easiest for users. Interesting enough, exactly the same argument is given by developers of other query languages: this syntax is chosen because it is in the best interest for the user. Shortly: this kind of design decisions are very poorly motivated and it is interesting to investigate if preferences for a certain syntax exists in certain user groups.
5. It is often unclear what is meant by users. Are end-users or expert users meant?
6. XQuery and XSLT share basically the same functionality.

References

[A01] "XML Query Use Cases from the library of congress". Caroline Arms, et al.. 31 May 2001. http://www.loc.gov/crsinfo/xml/lc_usecases.html.

[AQM97] "The Lorel Query Language for Semistructured Data". Serge Abiteboul. Dallan Quass. Jason McHugh. Jennifer Widow. Janet L. Wiener. 68-88. 1 (1). April 1997. *Journal of digital libraries.* Springer. 1432-5012 (printed edition).

[BC00] "Comparative Analysis of Five XML Query Languages". Angela Bonifati. Stefano Ceri. March 2000. *SIGMOD Record*. 68-79. 29.

[BMR99] "A formal Data model and Algebra for XML". David Beech. Ashok Malhotra. Michael Rys. 9-10-1999. http://elib.cs.berkeley.edu/seminar/2000/20000207.pdf.

[BR99] Ricardo Baeze-Yates, Berthier Ribiero-Neto, and Jeffrey D. Ullman. *Modern Information Retrieval.* Addison Wesley. 1999. ISBN 0-201-39829-X.

[CCD99] "XML-GL: a graphical language for querying and restructuring XML documents". Stefano Ceri. Sara Comai. Ernesto Damiani. Piero Fraternali. Stefano Paraboschi. Letizia Tanca. . . *Computer Networks.* 1171-1187. 31.

[CFP00] "XML: Current Developments and Future Challenges for the Database Community". Lecture Notes in Computer Science & Advances in Database Technology. Stefano Ceri. Piero Fraternali. Stefano Paraboschi. 2000. *EDBT 2000, 6th International Conference on Extending Database Technology.* 3-17. 1777.

[Cla01] "Unifying XSLT and XQuery element construction". James Clark. 2001. http://www.jclark.com/xml/construct.html.

[Cot98] *Candidate requirements for XML Query*. W3C QL'98 Position Paper. Paul Cotton. Ashok Malhotra. http://www.transquery.org. november 1998.

[CRF00] “Quilt: An XML Query Language for Heterogeneous Data Sources”. Jonathan Robie. Don Chamberlin. Daniela Florescu. 31 March 2000. http://www.almaden.ibm.com/cs/people/chamberlin/quilt_euro.html.

[CS00] “Yatl: a functional and declarative language for XML”. Sophie Cluet. Jérome Siméon. 2000.

[DFF99] “A Query Language for XML”. Alin Deutsch. Mary Fernandez. Daniela Florescu. Alon Levy. Dan Suciu. 1155-1169. 31. 1999. *Computer Networks*.

[DFF99-2] “Querying XML Data”. Alin Deutsch. Mary Fernandez. Daniela Florescu. Alon Levy. David Maier. Dan Suciu. 27-34. 22 (3). 1999. *Data Engineering Bulletin*.

[DSB98] “Queries on Links and Hierarchies”. W3C QL'98 Position Paper. Steven DeRose. C.M. Sperberg-McQueen. Bill Smith. 18-11-1998. http://www.w3.org/TandS/QL/QL98/pp/linkhier.html.

[Erw00] “A visual language for XML”. Martin Erwig. 47-54. 2000. *16th IEEE Symp. on Visual languages*.

[FG01] “XIRQL: A Query Language for Information Retrieval in XML Documents”. Norbert Fuhr. Kai Grossjohann. 2001. *Sigir 2001*.

[FKM00] “Integrating keyword search into XML query processing”. Daniela Florescu. Donald Kossmann. Ioana Monalescu. 119-135. 33. 2000. *Computer Networks*.

[FSW99] “XML Query languages: Experiences and Examples”. Mary Fernandez. Jérome Siméon. Philip Wadler. 1999. http://www.w3.org/1999/09/ql/docs/xquery.html.

[GMW99] “From Semistructured Data to XML:Migrating the Lore Data Model and Query Language”. Roy Goldman. Jason Mc Hugh. Jennifer Widow. June 1999. . *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99)*.

[IL00] “XML Query Languages in Practice: An evaluation”. Paper presented at Web Age Information Management 2000. Zachary G. Ives. Ying Lu. 2000.

[Lenz01] “XQuery: Reinventing the wheel? ”. Evan Lenz. http://xmlportfolio.com/xquery.

[Lenz01-2] *TransQuery*. Evan Lenz. http://www.transquery.org.
[Mai98] “Database Desiderata for an XML Query Language”. W3C QL'98 Position Paper. David Maier. 1998. http://www.w3.org/TandS/QL/QL98/pp/maier.html.

[MR98] “Observations on Structured Document Query Languages”. W3C QL'98 Position Paper. Makoto Murata. Jonathan Robie. 1998. http://www.w3.org/TandS/QL/QL98/pp/murata-san.html.

[Rob98] “XML Query language (XQL)”. W3C QL'98 Position Paper. Jonathan Robie. Joe Lapp. David Schach. September 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html

[Rob99] *The Tree Structure of XML Queries*. Jonathan Robie. 13-09-1999. http://www.w3.org/1999/10/xquery-tree.html.
[Rob99-2] "Combining and Querying XML Data with XQL". GCA Conference Paper XML 99 (Philadelphia). Jonathan Robie. 1999.

[Sch01] "ApproXQL: Design and Implementation of an Approximate Pattern matching Language for XML". Technical Report B 01-02. Torsten Schlieder. May 2001. .

[Ste00] "A canonical query language and its efficient implementation". GCA XML Europe 2000. Gert Van der Steen. 2000.

[TEI] *Text Encoding Initiative.* http://www.tei-c.org/.

[TIH01] "Updating XML". Igor Tatarinov. Zachary G. Ives. Alon Y. Halevy. Daniel S. Weld. 2001. . *SIGMOD 2001.*

[W3C-AS] *Extensible Markup Language (XML) Activity Statement.* Liam Quin. http://www.w3.org/XML/Activity.html.

[W3C-QR] *XML Query requirements.* W3C Technical Report / Working Draft. 15-02-2001. Don Chamberlin. Peter Fankhauser. Massimo Marchiori. Jonathan Robie. http://www.w3.org/TR/xmlquery-req.

[W3C-WAC] "Web Content Accessibility Guidelines 1.0". W3C Recommendation. Wendy Chisholm. Gregg Vanderheiden. Ian Jacobs. 5-May-1999. http://www.w3.org/TR/WAI-WEBCONTENT/.

[W3C-XML] "Extensible Markup Language (XML) 1.0 (Second Edition)". W3C Recommendation. Tim Bray. Jean Paoli. C. M. Sperberg-McQueen. Eve Maler. 6 October 2000. http://www.w3.org/TR/REC-xml.html.