Querying XML Documents

John A. Miller
Sonali Sheth

Computer Science Department
415 GSRC
University of Georgia
Athens, GA 30602-7404

1 Introduction

Web-based simulation environments will need to maintain a large collection of documents. Information about simulation models and results from multiple sites will need to be stored. This information should be held in a widely utilized and highly viewable format. The eXtensible Markup Language (XML) is a new standard that supports data exchange on the World-Wide Web. It is sophisticated enough, so that complex real-world structures and relationships may be captured. Thus, it can be used as the universal format for data interchange. Large collections of XML documents may be stored for efficient retrieval using modern Database Management Systems (DBMSs).

The current state-of-the-art in database management is represented by the Object-Oriented (OO) and Object-Relational (OR) DBMSs each with their own query languages, Object Query Language (OQL, v.2) [CBB$^+$] and Structured Query Language (SQL3) [UW], respectively. These database systems are capable of storing Web documents such as those needed for Web-Based Simulation. However, even though Web documents (in this case XML documents) can be stored in these databases, they do not conform to either the relational or object-oriented model, rather they conform to the semistructured data model. Hence a few problems remain: (1) the OQL and SQL3 query languages are not ideal for accessing semi-structured documents, (2) they lack sophisticated information retrieval capabilities and (3) they require substantial learning.

Modern database query languages provide a high-level way to express queries to a database. For example, a concise query in SQL may be equivalent to hundreds of lines code in a programming language with a low-level (e.g., find/get record style) API. In addition, SQL in its original form was quite easy to learn. Unfortunately, the same cannot be said for SQL3.

The XML may provide an option to learning SQL3. As XML becomes the dominant language for client side Web applications and documents, the number of computer users that will be XML literate is expected be become very large. Because XML is object-oriented, real-world structures and relationships may be captured. Currently, there are a couple of query languages for XML based on the semistructured data model, viz. XML-QL [DFF+98], Lorel [AQM+] and some that are based on the XSL (Extensible Stylesheet Language) pattern syntax, viz. XQL [RLS].

This paper addresses the problem of storing and retrieving XML documents used in information intensive applications such as Web-based simulation. It surveys the current work being done on query languages and tools for XML, and then discusses our implementation of a simple graphical XML query tool to be used as a front end in the JSIM Web-Based Simulation Environment [MGT98].

2 XML as a Universal Form for Data Objects

Modern information systems consist of many applications and subsystems that must work together to support the information needs of organizations. In JSIM, several Java beans (software components) work together to provide useful simulation analyses. It is necessary to access information about simulation models, model agents and prior simulation results. It is also useful for JSIM beans to be able to interact with other software components from other products or packages.

In the past, it has been very difficult to get separately developed software to interoperate. Fortunately, recently developed applications and subsystems have one thing in common, they all utilize objects (e.g., OO Programming Languages such as Java and C++, OO and OR Databases, and middleware such as CORBA and DCOM). Unfortunately, the form of object varies. Rather than mapping an object to all other possible forms, it would be better to have one form as the *standard for data interchange*. It appears that several companies believe that XML will be that standard.

All core applications based on simple, structured ASCII text – e-mail, Usenet news, the Web, rely on fixed data-exchange formats. This requires that a new exchange format be specified each time a new application is used. Data integration solutions may use XML to automate the exchange of data. In general, with an integration solution, XML serves as an interface layer or wrapper for data being passed between data sources, making it possible for a wide variety of applications, legacy systems, and databases to exchange information. XML can thus be said to be Web-style Electronic Data Interchange (EDI) standard.

Let us look at some of the possibilities for data exchange with XML as the common format. The following figure depicts XML as the key ingredient through which disparate data systems interact.

- **Java**. Java and XML are very complementary. Java can dynamically produce XML documents. XML forms may send data to Java Servlets much as HTML forms now send data to CGI scripts. To facilitate the storage of Java objects in a form that other tools and systems

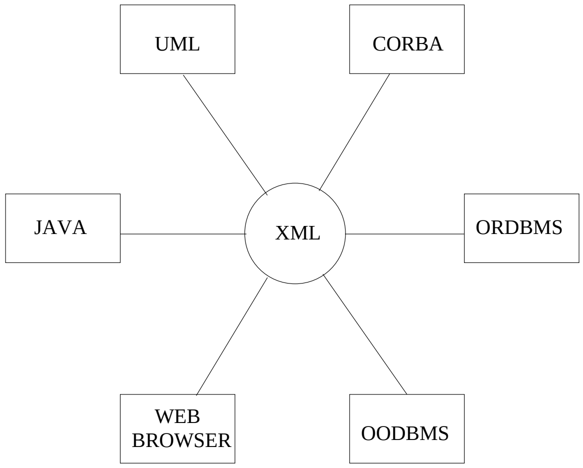

Figure 1: XML as a Universal Data Object

can understand, Java objects in main memory can be made persistent as XML documents (or parts of an XML document) rather than as binary files as is currently done when Java objects are serialized. The coins project has a prototype system to do just this [Coi]

- **Web Browsers**. An XML document should be thought of as data first and as a displayable document like an HTML page second. The display rendering information is given separately as an (eXtensible Style sheet Language) XSL specification. Currently, there are a few browsers that will display XML documents, but many more are expected to become available in the near future. Available browsers include Jumbo [MR98], Fujitso HyBrick [Lab98], Netscape Communicator 5 source code release [AW98], and Internet Explorer 5 preview release [Bra98]

- **Object-Relational Database Management Systems (ORDBMSs)**. Since XML documents may exhibit complex structures including deep nesting, traditional relational databases are not ideally suited for storing such documents. Object-relational databases augment relational databases with object-oriented capabilities making them more suitable for storing XML documents. Most object-relational database systems currently implement a subset of the ANSI/ISO draft SQL3 standard. *Oracle 8i* is Oracle's object-relational database for internet computing and it supports information definition and access through XML. Oracle sees XML as the means of providing a standard definition of all enterprise resources in one place, regardless of the type and location of the information. Oracle8i uses the structures in XML to build database structures and store data or links to data (through locators) according to the rules and constraints defined in XML. Individual XML instance objects are decomposed and stored according to the definitional structure. Upon access, the data is pulled and the XML object reconstructed and presented through the browser. XML provides the promise of establishing a coherent network of all kinds of information content not only within a system, but across systems. Oracle8i's support of XML means that this powerful capability will be supported, and its integrity ensured, by the Oracle DBMS [Ora].

- **Object-Oriented Database Management Systems (OODBMSs).** Object-oriented databases are designed to handle objects in their native forms. The objects then maintain their own data, methods, relationships and semantics of the whole model. This is ideal for the creation and management of hierarchical XML trees, while providing both hierarchical tree navigation and rich link traversal. XML also complements object-oriented databases in that, data from such a database, does not need to undergo transformation to be used at the client side; XML handles all of that. A couple of object-oriented databases that provide early support for XML are Object Store [Obj] and Poet [Poe].

  Object Store, an object-oriented database, can store a tree-like structures as-is, eliminating the mapping code required by traditional relational databases to convert the hierarchically structured XML object into rows and columns. With Object Store, navigating an XML tree is as simple as traversing a pointer in local memory. Object Store also has index and query capabilities which facilitate high-speed queries. Object Store also boosts performance, by supporting use of dynamic data. It can support a data abstraction model for XML, allowing developers to add a new field or attribute to a data structure without having to redefine the object model or evolve the schema. When used as a cache for application servers, the transactional and data consistency is managed by the Cache-Forward architecture. Object Store with its ability to manage, serve, index and query native XML data, thus provides a good foundation for many data integration solutions.

  Poet Object Server addresses the needs of XML as well as the needs of associated applications. It stores the XML elements intact, along with the structural semantics, thus maintaining the integrity of the XML model. Poet provides an object-oriented database that can be scaled up as well as down while leveraging the same APIs, thus simplifying application development. It provides a language-independent object model which allows developers to mix and match C++ and Java even in a single application. It also offers simple standards-based APIs and embeddability. It also provides for a higher granularity locking, down to element level, which is critical for user scalability. In addition, Poet provides features like tightly integrated XML-specific tree navigation, versioning, management of arbitrary links, import/export, publishing of structured content on the Web and support of object-oriented programming languages.

- **Common Object Request Broker Architecture (CORBA).** CORBA is a major standard for distributed object computing. Most CORBA products provide mechanism for making CORBA objects persistent. For maximal interoperability, the states of CORBA objects may be saved as XML objects in documents. CORBA's Interface Definition Language (IDL) defines the interface to objects. The IDL class members can be converted to corresponding elements in an XML DTD (Document Type Definition). Also different data types can be mapped to data types in XML with some modifications.

- **Unified Modeling Language (UML)**. UML is positioned as a general-purpose, unified, graphical modeling language for software development, in particular object analysis and design. Its main components are a metamodel, and a notation for graphically representing the concepts from the metamodel. It provides one language to the software and system development community, ending the method wars of the past few years at least in the area of concepts used to model object-based systems [Met]. With XML gaining momentum throughout the industry, recently several efforts have been started to integrate existing metamodels and metamodeling with XML. Design tools based on UML can generate code or specifications in several languages and formats. Generating XML DTDs from a graphical design can be done much like database schema are generated today. Some work is being carried out by the CDIF Technical Committee [Cdi]. CDIF is a Family of Standards that lays out a single architecture for exchanging information between modeling tools, and between repositories, and defines the interfaces of the components to implement this architecture. The XML-based CDIF syntax allows the exchange of meta-models and models using the emerging XML standard.

JSIM, requires Java applets, Web Browsers, ORDBMSs/OODBMS to be interfaced. Interfacing to CORBA and UML could also be useful as future extensions. Therefore, for JSIM all persistent objects should be savable in XML format. This makes display of information particularly easy (e.g., a model's description and properties stored in a repository, can be retrieved and displayed as an XML page).

3 Converting XML to Database Schema

If XML documents are required to have a Document Type Definition (DTD), they are called valid XML documents. (When no DTD is given, the document is simply said to be well-formed.) It is straightforward to convert a DTD into an object-oriented schema specification. For example, the following DTD can be readily converted to OQL's Object Definition Language (ODL).

Consider XML DTDs for office MEMO and MEETING document types, adapted from an XML tutorial [Xml].

```
<!DOCTYPE MEMO [
    < !ELEMENT MEMO (TO+, FROM, SUBJECT, BODY) >
    < !ATTLIST MEMO MEMO_NUM ID #REQUIRED >
    < !ATTLIST MEMO MEETING_REFS IDREFS #IMPLIED >
    < !ELEMENT TO           (#PCDATA) >
    < !ELEMENT FROM         (#PCDATA) >
    < !ELEMENT SUBJECT      (#PCDATA) >
    < !ELEMENT BODY         (P+) >
    < !ELEMENT P            (#PCDATA) >
] >
```

Figure 2: Memo Document Type

```
<!DOCTYPE MEETING [
    < !ELEMENT MEETING (ATTENDEE, PLACE, CALLEDBY) >
    < !ATTLIST MEETING MEETING_NUM ID #REQUIRED >
    < !ATTLIST MEETING MEMO_REFS IDREFS #IMPLIED >
    < !ELEMENT ATTENDEE     (#PCDATA) >
    < !ELEMENT PLACE        (#PCDATA) >
    < !ELEMENT CALLEDBY     (#PCDATA) >
] >
```

Figure 3: Meeting Document Type

The MEMO and MEETING DTDs, each contain an ID type attribute to uniquely identify each memo/meeting. The attribute MEMO_REFS in MEETING is the list of memos discussed in that meeting while the attribute MEETING_REFS in MEMO is the list of meetings in which the memo was discussed. Expressed in ODL, the DTD for MEMO is shown in Figure 4.

```
interface MEMO
    (extent MEMOS
    key (FROM, SUBJECT))
{
    attribute set<string> TO;
    attribute set         FROM;
    attribute set         SUBJECT;
    attribute set<string> BODY;
}
```

Figure 4: DTD of Figure 2 as Expressed in ODL

4 Forming Queries from DTDs

Patterns may be substituted into a DTD to form the basis for a query. Such a query can be run against one or more databases, in which case all matching XML documents will be retrieved for which the conditions in the query are true. This form of querying a database is analogous to the approach used in Query-By-Example (QBE) [Zlo75]. With a graphical tool supporting such an approach, finding information about simulation models or results, becomes a process of rapid query formulation, followed by browsing the query result, leading query to reformulation. From a simulationist's point of view, it would be similar to surfing the Web, except that the query tool would provide much better precision than today's search engines.

4.1 Pattern matching in XML Documents

Pattern matching in XML documents is a straightforward process. Consider the memo document type as shown in Figure 2. Suppose that one wishes to obtain all documents of type MEMO which are from John Miller and only see the subject (not the body) of the memo. Instead of explicitly writing a where clause with a condition, as one would do in either SQL or OQL (select ... from ... where), the DTD itself can be used to specify the where condition, as in Figure 5. Note that a "!!" symbol before any attribute or element denotes a attribute or element to be shown (as in a select clause).

```
<!DOCTYPE MEMO [
    < !ELEMENT MEMO (TO+, FROM, SUBJECT, BODY) >
    < !ATTLIST MEMO MEMO_NUM ID #REQUIRED >
    < !ATTLIST MEMO MEETING_REFS IDREFS #IMPLIED >
    < !!ELEMENT TO          (#PCDATA) >
    < !!ELEMENT FROM        ("John Miller") >
    < !!ELEMENT SUBJECT     (#PCDATA) >
    < !ELEMENT BODY         (P+) >
    < !ELEMENT P            (#PCDATA) >
] >
```

Figure 5: Query to Find John Miller's Memos

The same query may be written in OQL or SQL.

```
select m.TO, m.FROM, m.SUBJECT
from   MEMO m
where  m.FROM = "John Miller";
```

Although simple in this case, complex queries may be hard to express in this form. To simplify a user's interaction a graphical query tool may be built. Our approach is to make writing a query as simple as modification of a DTD.

4.2 Pattern Matching using Regular Expressions

Simply finding XML documents that match constant strings is too crude of an approach to serve as a query language. XML includes several regular expressions operators that can be very useful in queries.

- | – alternation (match one of the cases)
- * – closure (repeat 0 or more times)
- + – closure (repeat 1 or more times)

The query in Figure 5 may miss certain memos since in some cases the middle initial may have been used. To handle both cases an alternation may be used. This is accomplished by changing the FROM element specification to

```
<!!ELEMENT FROM    ("John Miller" | "John A. Miller")>
```

If one is interested in memos from any Miller, the FROM of Figure 5 is changed to

```
<!!ELEMENT FROM    ("_*Miller")>
```

Suppose one is interested in memos from Miller and to Sheth. The query in Figure 6 introduces multiple conditions to constrain the values that the elements FROM and TO can take on.

```
<!DOCTYPE MEMO [
    < !ELEMENT MEMO (TO+, FROM, SUBJECT, BODY) >
    < !ATTLIST MEMO MEMO_NUM ID #REQUIRED >
    < !ATTLIST MEMO MEETING_REFS IDREFS #IMPLIED >
    < !!ELEMENT TO          ("_*Sheth") >
    < !!ELEMENT FROM        ("_*Miller") >
    < !!ELEMENT SUBJECT     (#PCDATA) >
    < !ELEMENT BODY         (P+) >
    < !ELEMENT P            (#PCDATA) >
] >
```

Figure 6: Query to Find Memos from Miller to Sheth

The conditions on elements are ANDed together. To handle more complex cases including those with ORed conditions, a graphical query tool may be used. In this case, conditions in the same color are ANDed together, while different colors represent disjuncts. Finally, elements to be displayed have their checkboxes checked. This approach is illustrated in the next subsection through a complex query.

4.3 XML Query Tool

Consider expressing the following complex query using the graphical XML query tool as shown in Figure 7.

Query: Find the memos where the person who received the memo is the person who called the meeting for discussing the memo and whose name ends with "Miller".

This involves selecting and placing query boxes on a canvas and then linking them together. Clicking on a query box brings up a dialog box in which the name of a DTD is given. Pressing the get button will place the DTD in the dialog box. Text boxes may be modified to constrain the elements. Check boxes are provided for each element and checking an element, selects its value to be displayed in the output of the query result. Finally, conditions that apply to two different DTDs are specified in a join box associated with the link between the two query boxes.

5 Summary

XML is likely to play a central role in future information systems. It will be used for data interchange and XML documents will need to be stored in databases. Web-based simulation carried out at a variety of sites and even using different simulation products can exchange models and simulation results, if they agree to use XML. (Semantic issues in interoperability may be partially addressed through defining standard sets of DTDs like CML for Chemistry or BSML for Genetics). This paper has discussed the use of XML as a standard for data interchange. It also has presented a simple query tool for XML documents. With knowledge of XML, complex queries can be formulated without resorting to using traditional queries like SQL or OQL. Such a simple form of querying will be very useful to simulationists, many of whom are not inclined to learn database query languages.

References

[AQM+] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel Query Language for Semistructured Data. http://www-db.stanford.edu.

[AW98] Liora Alschuler and Mark Walter. Surprise! Netscape Puts XML Support in Mozilla. *The Bulletin: Seybold News and Views on Electronic Publishing*, 3(26), 1998. http://www.xml.com/xml/pub/SeyboldReport/bul032601.html.

[Bra98] Tim Bray. Microsoft IE 5.0 Preview. 1998. http://www.xml.com/xml/pub/98/06/ie5/ie5.html.

[CBB+] R.G.G. Cattell, Douglas K. Barry, Dirk Bartels, Mark Berler, Jeff Eastman, Sophie Gamerman, David Jordan, Adam Springer, Henry Strickland, and Drew Wade. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann Publishers, Inc., San Fransisco, California.

[Cdi] Introduction to cdif. http://www.eia.org/eig/cdif/intro.html.

[Coi] Coins: Connecting XML Elements and JavaBeans. http://www.jxml.com/coins/index.html.

[DFF+98] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL: A Query Language for XML. 1998. http://www.w3.org/TR/1998/NOTE-xml-ql-19980819.

[Lab98] Fujitsu Laboratories. HyBrick SGML/XML Browser. 1998. http://collie.fujitsu.com/hybrick.

[Met] Metamodel information. http://www.metamodel.com.

[MGT98] J.A. Miller, Y. Ge, and Z. Tao. Component-Based Simulation Environments: JSIM as a Case Study Using Java Beans. pages 786–793, December 1998.

[MR98] Peter Murray-Rust. Jumbo2. 1998. http://www.xml-cml.org/jumbo.

[Obj] Server-side xml: Taming the tower of babel. http://www.odi.com/ODILIVE/contents/happenings/tech_trends/whitepapers/ServerSideXML.pdf.

[Ora] Internet database solutions with oracle 8i. http://www.oracle.com/html/webdb.html.

[Poe] Xml - the foundation for the future. http://www.poet.com/PDF/xml.pdf.

[RLS] Jonathan Robie, Joe Lapp, and David Schach. Xml query language (xql).

[UW] Jeffrey D. Ullman and Jennifer Widom. *A First Course in Database Systems.* Prentice Hall, Upper Saddle River, New Jersey.

[Xml] A tutorial in xml and xsl authoring. http://pdbeam.uwaterloo.ca/ rlander/XML_Tutorial/.

[Zlo75] Query By Example. *NCC, AFIPS*, page 44, 1975.

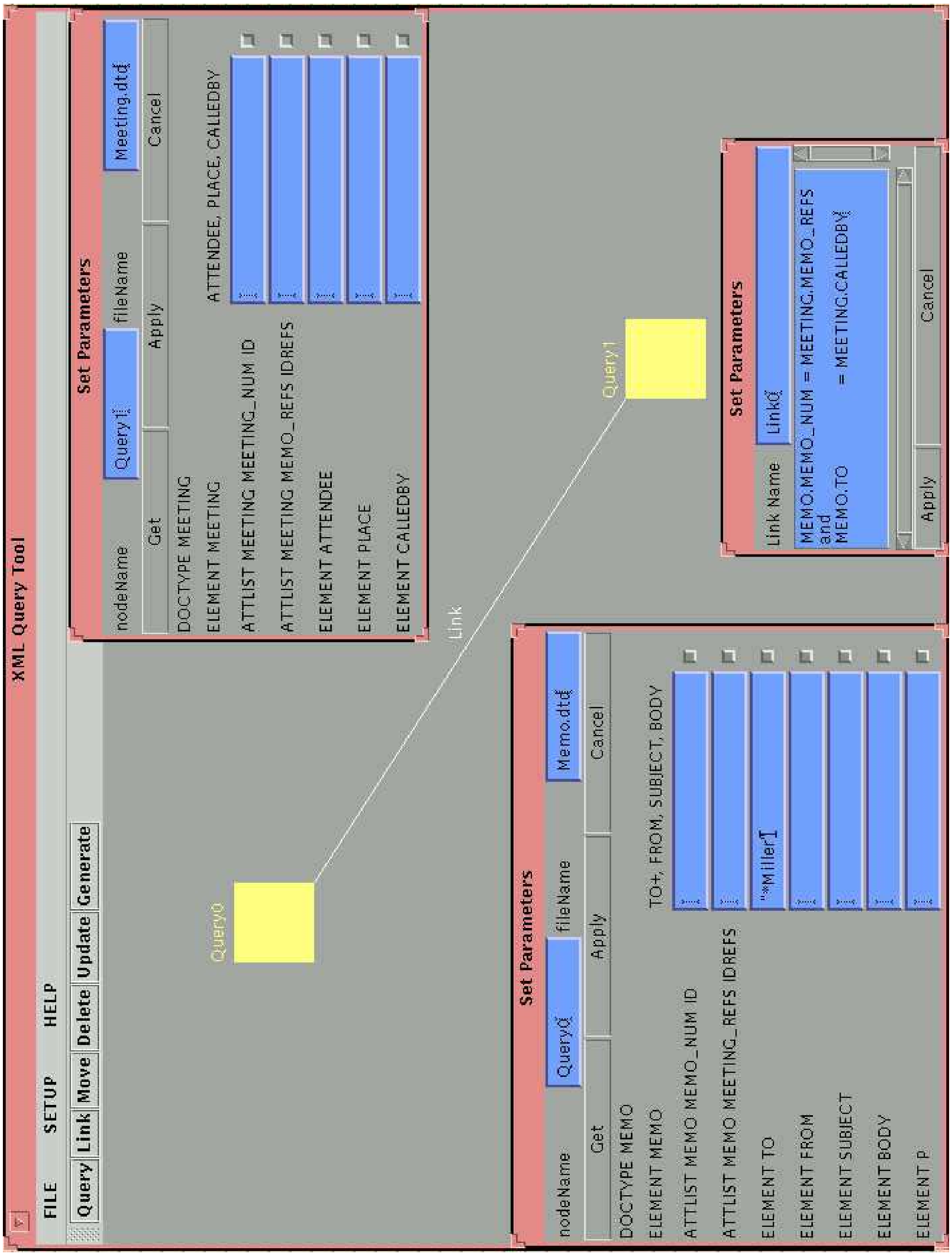

Figure 7: XML Query Tool