Querying Bi-level Information

Sudarshan Murthy David Maier Lois Delcambre
Dept. of CSE, OGI School of Sc. & Eng. at OHSU
20000 NW Walker Road
Beaverton, OR 97006, USA
+1 503 748 7068
smurthy, maier, lmd @cse.ogi.edu

ABSTRACT

In our research on superimposed information management, we have developed applications where information elements in the superimposed layer serve to annotate, comment, restructure, and combine selections from one or more existing documents in the base layer. Base documents tend to be unstructured or semi-structured (HTML pages, Excel spreadsheets, and so on) with marks delimiting selections. Selections in the base layer can be programmatically accessed via marks to retrieve content and context. The applications we have built to date allow creation of new marks and new superimposed elements (that use marks), but they have been browse-oriented and tend to expose the line between superimposed and base layers. Here, we present a new access capability, called *bi-level queries*, that allows an application or user to query over both layers as a whole. Bi-level queries provide an alternative style of data integration where only relevant portions of a base document are mediated (not the whole document) and the superimposed layer can add information not present in the base layer. We discuss our framework for superimposed information management, an initial implementation of a bi-level query system with an XML Query interface, and suggest mechanisms to improve scalability and performance.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval---Information filtering, Retrieval models, H.2.5 [Database Management]: Heterogeneous Databases.

General Terms

Management, Performance, Design.

Keywords

Bi-level queries, SPARCE, Superimposed information management, Information integration.

Seventh International Workshop on the Web and Databases (WebDB 2004) June 17–18, 2004, Paris, France.

1. INTRODUCTION

You are conducting background research for a paper you are writing. You have found relevant information in a variety of sources: HTML pages on the web, PDF documents on the web and on your SIGMOD anthology of CDs, Excel spreadsheets and Word documents from your past work in a related area, and so on. You identify relevant portions of the documents and add annotations with clarifications, questions, and conclusions. As you collect information, you frequently reorganize the information you have collected thus far (and your added annotations) to reflect your perspective. You intentionally keep your information structure loose so you can easily move things around. When you have collected sufficient information, you import it, along with your comments, in to a word-processor document. As you write your paper in your word-processor, you revisit your sources to see information in its context. Also, as you write your paper you reorganize its contents, including the imported information, to suit the flow. Occasionally, you search the imported annotations, selections, and the context of the selections. You mix some of the imported information with other information in the paper and transform the mixture to suit presentation needs.

Most researchers will be familiar with manual approaches to the scenario we have just described. Providing computer support for this scenario requires a toolset with the following capabilities:

1. Select portions of documents of many kinds (PDF, HTML, etc.) in many locations (web, CD, local file system, etc.), and record the selections.
2. Create and associate annotations (of varying structure) with document selections.
3. Group and link document selections and annotations, reorganize them as needed, and possibly even maintain multiple organizations.
4. See a document selection in its context by opening the document and navigating to the selected region, or access the context of a selection without launching its original document.
5. Place document selections and annotations in traditional documents (such as the word-processor document that contains your paper).
6. Search and transform a mixture of document selections, annotations, and other information.

Systems that support some subset of these capabilities exist, but no one system supports the complete set. It is hard to use a collection of systems to get the full set of features because the systems do not interoperate well. Some hypertext systems can

create multiple organizations of the same information, but they tend to lack in the types of source, granularity of information, or the location of information consulted. For example, Dexter [6] requires all information consulted to be stored in its proprietary database. Compound document systems can address sub-documents, but they tend to have many display constraints. For example, OLE 2 [9] relies on original applications to render information. Neither type of system supports querying a mixture of document selections and annotations.

Superimposed information management is an alternative solution for organizing heterogeneous *in situ* information, at document and sub-document granularity. *Superimposed information* (such as annotations) refers to data placed over existing information sources (*base information*) to help organize, access, connect and reuse information elements in those sources [8]. In our previous work [12], we have described the *Superimposed Pluggable Architecture for Contexts and Excerpts (SPARCE)*, a middleware for superimposed information management, and presented some superimposed applications built using SPARCE. Together they support Capabilities 1 through 4. In this paper, we show how SPARCE can be used to support Capability 6. Details of support for Capability 5 are outside the scope of this paper.

Before we proceed with the details of how we support Capability 6, we introduce a superimposed application called *RIDPad* [12]. Figure 1 shows a RIDPad document that contains information selections and annotations related to the topic of information integration. The document shown contains eight items: CLIO, Definition, SchemaSQL, Related Systems, Goal, Model, Query Optimizer, and Press. These items are associated with six distinct base documents of three kinds—PDF, Excel, and HTML. An *item* has a name, a descriptive text, and a reference (called a *mark*) to a selection in a base document. For example, the item labeled 'Goal' contains a mark into a PDF document. The boxes labeled Schematic Heterogeneity and Garlic are groups. A *group* is a named collection of items and other groups. A *RIDPad document* is a collection of items and groups.

RIDPad affords many operations for items and groups. A user can create new items and groups, and move items between groups. The user can also rename, resize, and change visual characteristics such as color and font for items and groups. With the mark associated with an item, the user can navigate to the base layer if necessary, or examine the mark's properties and browse context information (such as containing paragraph) from within RIDPad via a reusable *Context Browser* we have built.

The operations RIDPAD affords are at the level of items and groups. However, we have seen the need to query and manipulate a RIDPad document and its base documents as a whole. For example, possible queries over the RIDPad document in Figure 1 include:

Q1: List base documents used in this RIDPad document.
Q2: Show abstracts of papers related to Garlic.
Q3: Create an HTML table of contents from the groups and items.

Query Q1 examines the paths to base documents of marks associated with items in the RIDPad document. Q2 examines the context of marks of items in the group labeled 'Schematic Heterogeneity.' Q3 transforms the contents of the RIDPad document to another form (table of contents). In general, queries such as these operate on both superimposed information and base information. Consequently, we call them *bi-level queries*.

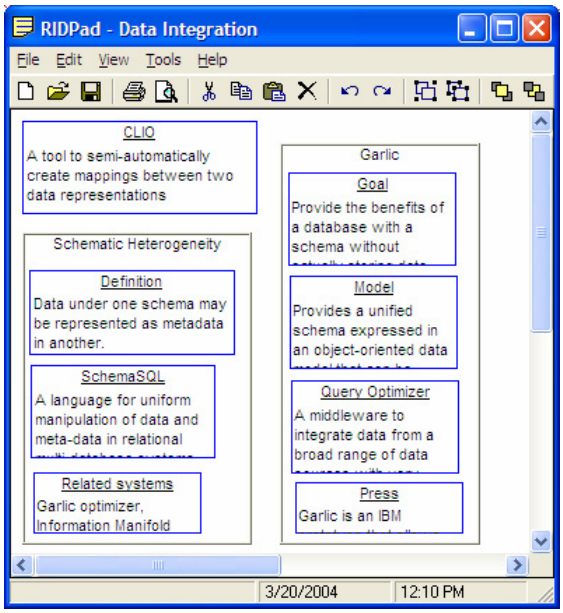

Figure 1: A RIDPad document.

There are many possible choices on how to present the contents of superimposed documents (such as the RIDPad document in Figure 1) and base documents for querying. We could make the division between the superimposed and base documents obvious and let the user explicitly follow marks from superimposed information to base information. Instead, our approach is to integrate a superimposed document's contents and related base information to present a uniform representation of the integrated information for querying.

The rest of this paper is organized as follows: Section 2 provides an overview of SPARCE. Section 3 provides an overview of bi-level query systems and describes a naïve implementation of a bi-level query system along with some example bi-level queries. Section 4 discusses some applications and implementation alternatives for bi-level query systems. Section 5 briefly reviews related work. Section 6 summarizes the paper.

We use the RIDPad document in Figure 1 for all examples in this paper.

2. SPARCE OVERVIEW

The Superimposed Pluggable Architecture for Contexts and Excerpts (SPARCE) facilitates management of marks and context information in the setting of superimposed information management [12]. A *mark* is an abstraction of a selection in a base document. Several mark implementations exist, typically one per base type (PDF, HTML, Excel, and so on). A mark implementation chooses an addressing scheme appropriate for the base type it supports. For example, an MS Word mark implementation uses the starting and ending character index of a text selection, whereas an MS Excel mark uses the row and column names of the first and last cell in the selection. All mark implementations provide a common interface to address base information, regardless of base type or access protocol they

support. A superimposed application can work uniformly with any base type due to this common interface.

Context is information concerning a base-layer element. Presentation information such as font name, containment information such as enclosing paragraph and section, and placement information such as line number are examples of context information. An *Excerpt* is the content of a marked base-layer element. (We treat an excerpt also as a context element.) Figure 2 shows the PDF mark corresponding to the item 'Goal' (of the RIDPad document in Figure 1) activated. The highlighted portion is the marked region. Table 1 shows some of the context elements for this mark.

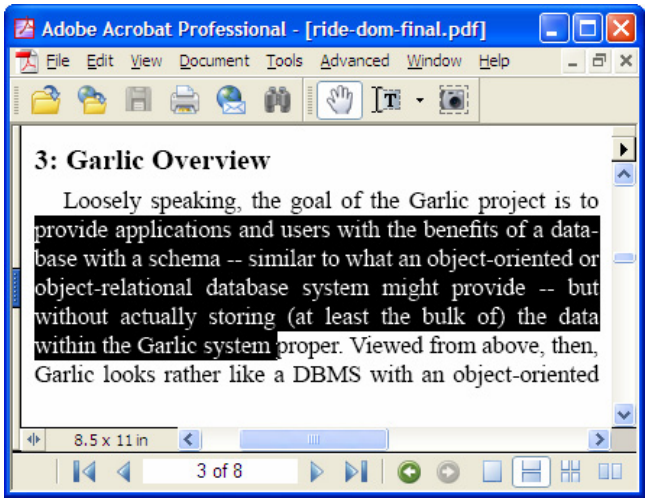

Figure 2: A PDF mark activated.

Figure 3 shows the SPARCE architecture reference model. The Mark Management module is responsible for operations on marks (such as creating and storing marks). The Context Management module retrieves context information. The Superimposed Information Management module provides storage service to superimposed applications. The Clipboard is used for interprocess communication.

Table 1: Some context elements of a PDF mark.

Element name	Value
Excerpt	provide applications and users with … Garlic system
Font name	Times New Roman
Enclosing paragraph	Loosely speaking, the goal …
Section Heading	Garlic Overview

SPARCE uses mediators [13] called *context agents* to interact with different base types. A context agent is responsible for resolving a mark and returning the set of context elements appropriate to that mark. A context agent is different from mediators used in other systems because it only mediates portions of base document a mark refers to. For example, if a mark refers to the first three lines of a PDF document, the mark's context agent mediates those three lines and other regions immediately around the lines. A user could retrieve broader context information for this mark, but the agent will not do so by default.

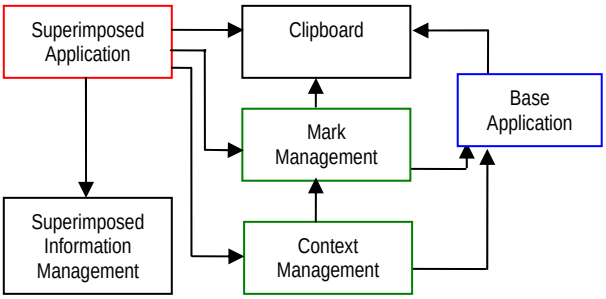

Figure 3: SPARCE architecture reference model.

A superimposed application allows creation of information elements (such as annotations) associated with marks. It can use an information model of its choice (SPARCE does not impose a model) and the model may vary from one application to another. For example, RIDPad uses a group-item model (simple nesting), whereas the Schematics Browser, another application we have built, uses an ER model [2, 12]. The superimposed model may be different from any of the base models. A detailed description of SPARCE is available in our previous work [12].

3. BI-LEVEL QUERY SYSTEM

A *bi-level query system* allows a superimposed application and its user to query the superimposed information and base information as a whole. User queries are in a language appropriate to the superimposed model. For example, XQuery may be the query language if the superimposed model is XML (or a model that can be mapped to XML), whereas SQL may be the query language if superimposed information is in the relational model.

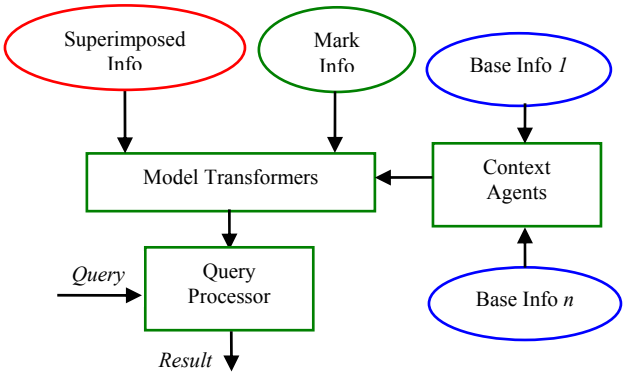

Figure 4: Overview of a bi-level query system.

Figure 4 provides an overview of a bi-level query system. An oval in the figure represents an information source. A rectangle denotes a process that manipulates information. Arrows indicate data flow. The query processor accepts three kinds of information—superimposed, mark, and context. Model transformers transform information from the three sources in to model(s) appropriate for querying. One of these transformers, the *context transformer*, is responsible for transforming context information. We restrict bi-level query systems to use only one superimposed model at a time, for practical reasons. Choosing a query language and the model for the result can be hard if superimposed models are mixed.

3.1 Implementation

We have implemented a naïve bi-level query system for the XML superimposed model. We have developed a transformer to convert RIDPad information to XML. We have developed a context transformer to convert context information to XML. We are able to use mark information without any transformation since SPARCE already represents that information in XML. User queries can be in XPath, XSLT, and XQuery. We use Microsoft's XML SDK 4.0 [10] and XQuery demo implementation [11] to process queries.

We use three XML elements to represent RIDPad information in XML—`<RIDPadDocument>` for the document, `<Group>` for a group, and `<Item>` for an item. For each RIDPad item, the system creates four children nodes in the corresponding `<Item>` element. These children nodes correspond to the mark, container (base document where the mark is made), application, and context. We currently transform the entire context of the mark. The XML data is regenerated if the RIDPad document changes.

```
<?xml version="1.0" ?>
- <RIDPadDocument name="Data Integration">
  - <Group name="Garlic" index="1" left="2955" top="360">
    + <Item name="Press" index="6" left="3000" top="3990">
    - <Item name="Goal" index="4" left="2970" top="675">
      + <Mark ID="AcrobatPDFTextMark20040320105004SURYASMurthy">
      + <Container ID="CClassesCSE606INIride-dom-finalpdf">
      + <Application ID="Acrobat5">
      + <Context objecttype="Mark"
          objectid="AcrobatPDFTextMark20040320105004SURYASMurthy">
        <![CDATA[ Provide the benefits of a database with a schema
        without actually storing data within the Garlic system proper. ]]>
      </Item>
    + <Item name="Query Optimizer" index="3" left="3000" top="2910">
    + <Item name="Model" index="2" left="2985" top="1785">
    </Group>
  + <Group name="Schematic Heterogeneity" index="7" left="120" top="1320">
  + <Item name="CLIO" index="13" left="120" top="120">
  </RIDPadDocument>
```

Figure 5: Partial XML data from a RIDPad document.

Figure 5 shows partial XML data generated from the RIDPad document in Figure 1. It contains two `<Group>` elements (corresponding to the two groups in Figure 1). The 'Garlic' element contains four `<Item>` elements (one for each item in that group in Figure 1). There is also an `<Item>` element for the group-less item CLIO. The `<Item>` element for 'Goal' is partially expanded to reveal the `<Mark>`, `<Container>`, `<Application>`, and `<Context>` elements it contains. Contents of these elements are not shown.

3.2 Example Bi-level Queries

We now provide bi-level query expressions for the queries Q1 to Q3 listed in Section 1.

Q1: List base documents used in this RIDPad document.

This query must retrieve the path to the base document of the mark associated with each item in a RIDPad document. The following XQuery expression does just that. The Location element in the Container element contains the path to the document corresponding to the mark associated with an item.

```
<Paths> {FOR $l IN
document("source")//Item/Container/Location
RETURN <Path>{$l/text()}</Path>
} </Paths>
```

Q2: Show abstracts of papers related to Garlic.

This query must examine the context of items in the group labeled 'Garlic.' The following XPath expression suffices. This expression returns the text of a context element whose name attribute is 'Abstract', but only for items in the required group.

```
//Group[@name='Garlic']/Item/Context//Elemen
t[@name='Abstract']/text()
```

Q3: Create an HTML table of contents from the groups and items.

We use an XSLT style-sheet to generate a table of contents (TOC) from a RIDPad document. Figure 6 shows the query in the left panel and its results in the right panel. The right panel embeds an instance of MS Internet Explorer. The result contains one list item (HTML LI tag) for each group in the RIDPad document. There is also one list sub-item (also an HTML LI tag) for each item in a group. The group-less item CLIO is in the list titled 'Other Items.' A user can save the HTML results, and open it in any browser outside our system.

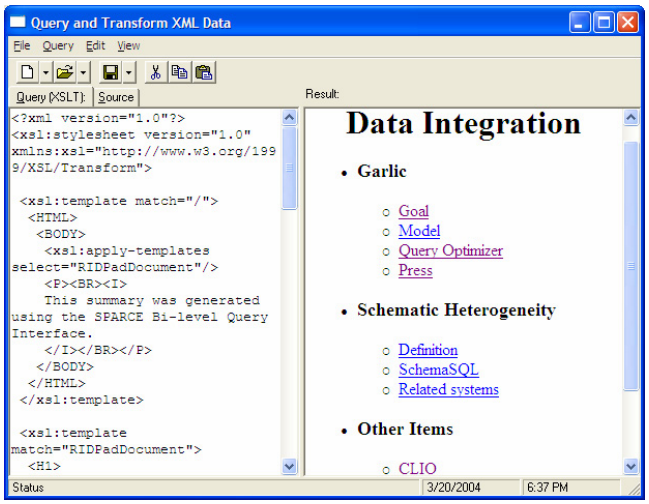

Figure 6: RIDPAD document transformed to HTML TOC.

The HTML TOC in Figure 6 shows that each item has a hyperlink (HTML A tag) attached to it. A hyperlink is constructed using a custom URL naming scheme and handled using a custom handler. Custom URLs are one means of implementing Capability 5 identified in Section 1.

4. DISCUSSION

The strength of the current implementation is that it retrieves context information for only those parts of base documents that the superimposed document refers to (via marks). Interestingly, the same is also its weakness: it retrieves context information for all parts of the base documents the superimposed document refers to, regardless of whether executing a query requires those elements. For example, only Query Q2 looks at context information (Q1 looks only at container information, Q3 looks at superimposed information and mark information). However, the XML data generated includes context information for all queries. Generating data in this manner is both inefficient and unnecessary—information may be replicated (different items may use the same mark), and context information can be rather large (the size of the complete context of a mark could exceed the size of its docu-

ment), depending on what context elements a context agent provides. It is possible to get the same results by separating RIDPad data from the rest and joining the various information sources. Doing so preserves the layers, and potentially reduces the size of data generated. Also, it is possible to execute a query incrementally and only generate or transform data that qualifies in each stage of execution.

Figure 7 gives an idea of the proposed change to the schema of the XML data generated. Comparing with the Goal Item element of Figure 5, we see that mark, container, application, and context information are no longer nested inside the Item element. Instead, an `<Item>` element has a new attribute called `markID`. In the revised schema, the RIDPad data, mark, container, application, and context information exist independently in separate documents, with references linking them. With the revised schema, no context information would be retrieved for Query Q1. Context information would be retrieved only for items in the 'Schematic Heterogeneity' group when Q2 is executed.

```
<?xml version="1.0" ?>
- <RIDPadDocument name="Data Integration">
  - <Group name="Garlic" index="1" left="2955" top="360">
    + <Item name="Press" index="6" left="3000" top="3990"
      markID="HTMLMark2004Mar20120711SURYASMurthy">
    - <Item name="Goal" index="4" left="2970" top="675"
      markID="AcrobatPDFTextMark20040320105004SURYASMurthy">
        <![CDATA[ Provide the benefits of a database with a schema
        without actually storing data within the Garlic system proper. ]]>
      </Item>
    + <Item name="Query Optimizer" index="3" left="3000" top="2910"
      markID="AcrobatPDFTextMark20031207193423TYEEsmurthy">
    + <Item name="Model" index="2" left="2985" top="1785"
      markID="AcrobatPDFTextMark20031207193126TYEEsmurthy">
    </Group>
  + <Group name="Schematic Heterogeneity" index="7" left="120" top="1320">
    <Item name="CLIO" index="13" left="120" top="120"
      markID="AcrobatPDFTextMark20031207194031TYEEsmurthy" />
  </RIDPadDocument>
```

Figure 7: XML data in the revised schema.

Preserving the layers of data has some disadvantages. A major disadvantage is that a user will need to use joins to connect data across layers. Such queries tend to be error-prone, and writing them can take too much time and effort. A solution would be to allow a user to write bi-level queries as they currently do (against a schema corresponding to the data in Figure 5), and have the system rewrite the query to match the underlying XML schema (as in Figure 7). That is, user queries would actually be expressed against a view of the actual data. We are currently pursuing this approach to bi-level querying.

Our current approach of grabbing context information for all marks could be helpful in some cases. For example, if a query workload ends up retrieving context of all (or most) marks, the current approach is similar to materializing views, and could lead to faster overall query execution.

The current implementation does not exploit relationships between superimposed information elements. For example, Figure 8 shows the RIDPad document in Figure 1 enhanced with two relationships 'Uses' and 'Addresses' from the item CLIO. A user may exploit these relationships, to pose richer queries and possibly recall more information. For example, with the RIDPad document in Figure 8, a user could now pose the following queries: What system does CLIO use? How is CLIO related to SchemaSQL?

Our initial use anticipated for bi-level queries was to query superimposed and base information as a whole, but we have noticed that superimposed application developers and users could use the capability to construct and format (on demand) superimposed information elements themselves. For example, a RIDPad item's name may be a section heading. Such a representation of an item could be expressed as the result of a query or a transformation.

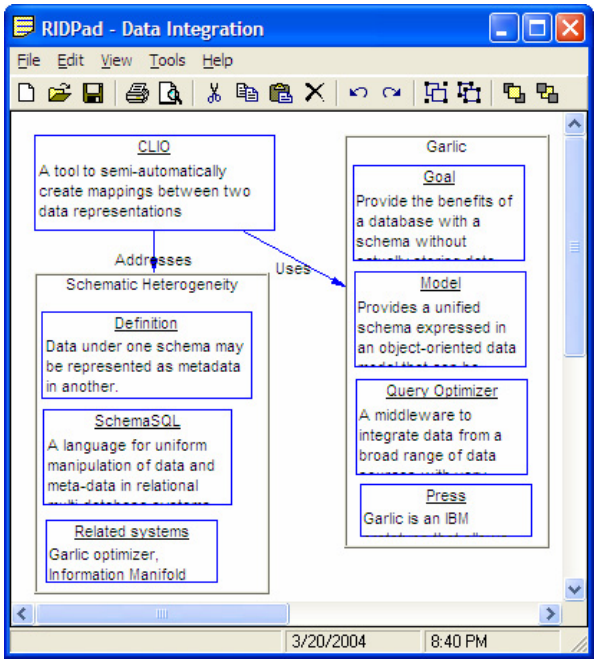

Figure 8: A RIDPad document with relationships.

Bi-level queries could also be used for repurposing information. For example, Query Q3 could be extended to include the contents of items (instead of just names) and transform the entire RIDPad document to HTML (like in Figure 6). The HTML version can then be published on the web.

We have demonstrated bi-level queries using XML query languages, but superimposed applications might benefit from other query languages. The choice of the query language depends largely on the superimposed information model (which in turn depends on the task at hand). More than one query language may be appropriate for some superimposed information models, in some superimposed applications. For example, both CXPath [3] and XQuery may be appropriate for some applications that use the XML superimposed model.

The base applications we have worked with so far do not themselves have query capabilities. If access to context or a selection over context elements can be posed as a query in a base application, we might benefit from applying distributed query-processing techniques. Finally, the scope of a bi-level query is currently the superimposed layer and the base information accessible via the marks used. Some applications might benefit from including marks generated automatically (for example, using IR techniques) in the scope of a query.

5. RELATED WORK

SPARCE differs from mediated systems such as Garlic [4] and MIX [1]. Sources are registered with SPARCE simply by the act of mark creation in those sources. Unlike in Garlic there is no need to register a source and define its schema. Unlike MIX, SPARCE does not require a DTD for a source.

METAXPath [5] allows a user to attach metadata to XML elements. It enhances XPath with an 'up-shift' operator to navigate from data to metadata (and metadata to meta-metadata, and so on). A user can start at any level, but only cross between levels in an upwards direction. In our system, it is possible to move both upwards and downwards between levels. METAXPath is designed to attach only metadata to data. A superimposed information element can be used to represent metadata about a base-layer element, but it has many other uses.

CXPath [3] is an XPath-like query language to query concepts, not elements. The names used in query expressions are concept names, not element names. In the CXPath model there is no document root—all concepts are accessible from anywhere. For example, the CXPath expression '/Item' and 'Item' are equivalent. They both return all Item elements when applied to the XML data in Figure 5. The '/' used for navigation in XPath follows a relationship (possibly named) in CXPath. For example, the expression "/Item/{Uses}Group" returns all groups that are related to an item by the 'Uses' relationship when applied to an XML representation of the RIDPad in Figure 8. CXPath uses predefined mappings to translate CXPath expressions to XPath expressions. There is one mapping for each concept name and for each direction of every relationship of every XML source. In our system, we intend to support multiple sources without predefined mappings, but we would like our query system to operate at a conceptual level like CXPath does.

As discussed in Section 4, preserving the layers of data, yet allowing a user to express queries as if all data is in one layer means queries are expressed against views. Information Manifold [7] provides useful insight in to how heterogeneous source may be queried via views. That system associates a *capability record* with each source to describe its inputs, outputs, and selection capabilities. We currently do not have such a notion in our system, but we expect to consider source descriptions in the context of distributed query processing mentioned in Section 4.

6. SUMMARY

Our existing framework for superimposed applications supports examination and manipulation of individual superimposed and base information elements. More global ways to search and manipulate information become necessary as the size and number of documents gets larger. A bi-level query system is a first step in that direction. We have an initial implementation of a query system, but still have a large space of design options to explore.

7. ACKNOWLEDGMENTS

This work was supported in part by US NSF Grant IIS-0086002. We thank all reviewers.

8. REFERENCES

[1] Baru, C., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P., and Chu, V. XML-Based Information Mediation with MIX. In *Proceedings of the SIGMOD conference on Management of Data* (Philadelphia, June, 1999). ACM Press, New York, NY, 1999, 597-599.

[2] Bowers, S., Delcambre, L. and Maier, D. Superimposed Schematics: Introducing E-R Structure for In-Situ Information Selections. In *Proceedings of ER 2002* (Tampere, Finland, October 7-11, 2002). Springer LNCS 2503, 2002. 90–104.

[3] Camillo, S.D., Heuser, C.A., and Mello, R. Querying Heterogeneous XML Sources through a Conceptual Schema. In *Proceedings of ER 2003* (Chicago, October 13-16, 2003). Springer LNCS 2813, 2003. 186–199.

[4] Carey, M.J., Haas, L.M., Schwarz, P.M., Arya, M., Cody, W.F., Fagin, R., Flickner, M., Luniewski, A.W., Niblack, W., Petkovic, D., Thomas, J., Williams, J.H., and Wimmers, E.L. *Towards heterogeneous multimedia information systems: The Garlic approach.* IBM Technical Report RJ 9911, 1994.

[5] Dyreson, C.E., Bohlen, M.H., and Jensen, C.S. METAXPath. In *Proceedings of the International Conference on Dublin Core and Metadata Applications* (Tokyo, Japan, October 2001). 2001, 17-23.

[6] Halasz, F.G., and Schwartz, F. The Dexter Hypertext Reference Model. *Communications of the ACM*, *37*, 2, 30-39.

[7] Levy, A.Y., Rajaraman, A., and Ordille, J.J. Querying heterogeneous information sources using source descriptions. In *Proceedings of VLDB* (Bombay, India 1996). 251-262.

[8] Maier, D., and Delcambre, L. Superimposed Information for the Internet. In Informal Proceedings of WebDB '99 (Philadelphia, June 3-4, 1999). 1-9.

[9] Microsoft. COM: *The Component Object Model Specification*, Microsoft Corporation. 1995.

[10] Microsoft. *MS XML 4.0 Software Development Kit.* Microsoft Corporation. Available online at http://msdn.microsoft.com/

[11] Microsoft. *XQuery Demo.* Microsoft Corporation. Available online at http://xqueryservices.com/

[12] Murthy, S., Maier, D., Delcambre, L., and Bowers, S. Putting Integrated Information in Context: Superimposing Conceptual Models with SPARCE. In *Proceedings of the First Asia-Pacific Conference of Conceptual Modeling* (Dunedin, New Zealand, Jan. 22, 2004). 71-80.

[13] Wiederhold, G. Mediators in the architecture of future information systems. *IEEE Computer, 25*, 3 (March 1992). 38–49.