Integrating Diverse Information Management Systems: A Brief Survey

Sriram Raghavan Hector Garcia-Molina
Computer Science Department
Stanford University
Stanford, CA 94305, USA
{rsram, hector}@cs.stanford.edu

Abstract

Most current information management systems can be classified into text retrieval systems, relational/object database systems, or semistructured/XML database systems. However, in practice, many applications data sets involve a combination of free text, structured data, and semistructured data. Hence, integration of different types of information management systems has been, and continues to be, an active research topic. In this paper, we present a short survey of prior work on integrating and inter-operating between text, structured, and semistructured database systems. We classify existing literature based on the kinds of systems being integrated and the approach to integration. Based on this classification, we identify the challenges and the key themes underlying existing work in this area.

1 Introduction

Most information management systems (IMS) can usually be classified into one of three categories depending on the kind of data that they are primarily designed to handle.[1] Text retrieval systems are concerned with the management and query-based retrieval of collections of unstructured text documents. Relational or object-oriented database systems are concerned with the management of structured or strictly-typed data, i.e., data that conforms to a well-defined schema. Finally, semistructured databases are designed to efficiently manage data that only partially conforms to a schema, or whose schema can evolve rapidly [1]. Each of these systems employ different physical and logical data models, query languages, and query processing techniques appropriate to the type of data being managed (see Table 1 for a brief summary).

There is a substantial body of work that deals with the design of each of these classes of information management systems. In addition, there has been significant interest in combining, integrating, and inter-operating between information management systems that belong to different classes. There are two primary motivations for most of the work in this area. First, many applications require processing of data that belongs to more than one type. For instance, a medical information system at a hospital must process doctor reports (free text documents) as well as patient records (structured relational data). Similarly, an order processing application might

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] For the purposes of this paper, we will be ignoring non-text (audio, images, and video) information management systems as well as systems that are designed for specialized data types (e.g., geographical information systems).

	Text Retrieval Systems	*Relational & Object DBMS*	*Semistructured (XML) DBMS*
Data Models	Unstructured text documents possibly with structured fields	Relations, Objects grouped into classes	Directed edge-labeled graphs
Query Models	Extended boolean, Vector space, Probabilistic	Relational model	Relational model extended with graph operations and recursion
Query Operators	Boolean operators, Proximity operators, Pattern matching operators	Relational operators	Relational operators + (extended) path expressions + restructuring operators
Example Query Languages	Boolean, Natural language [5]	SQL, OQL	Lorel[2], UnQL [8]

Table 1: Comparing different information management systems based on data and query models[2]

need to handle inventory information in a relational database as well as purchase orders received as (semistructured) XML documents. Second, there are significant advantages in leveraging the facilities provided by one type of information management system to implement another. For instance, a text retrieval system that is built on top of a relational or object database system [7, 38] can benefit from the sophisticated concurrency control and recovery facilities of the latter, without having to implement these features from scratch.

However, fundamental differences in the data and query models of these systems, pose significant challenges to such integration efforts. Depending on the target application scenario, there are several ways of addressing these challenges. For instance, some integration approaches are based on *extending* one type of system, either natively or through the use of plugin modules [19, 18], to support operators, features, or data types normally found in another (e.g., support for keyword searches in a semistructured database system [36]). Other approaches employ a separate *middleware* layer that provides a uniform and common query interface to a collection of diverse information management systems (e.g., the Garlic system [9] at IBM Almaden).

In this paper, we present a short survey of some of the techniques for integrating different classes of information management systems. We present a classification system that enables us to group related work, based on the types of systems being integrated and the integration architecture. We emphasize that our survey is by no means comprehensive, nor is our classification the only way to organize the literature. Our aim is to use the classification to identify key integration challenges and describe some of the proposed solutions.

The rest of the paper is organized as follows. In Section 2, we describe our classification system. In Sections 3, 4, and 5, for each of the three possible pairs of information management systems, we identify the key issues in integration and briefly discuss some representative work from the literature. We conclude in Section 6.

2 Classification System

We classify existing integration literature along two axes, as shown in Table 2. The horizontal axis of Table 2 enumerates all possible pairs from among the three classes of information management systems (IMS) described earlier. The vertical axis lists the three most commonly employed architectures for tying such systems together. The cells of the table are populated with bibliographic references that indicate how each of the referenced works fit into our classification system.

The schematics in Figure 1 illustrate the differences between the three integration architectures. In systems with a *layered* architecture, an IMS of one type is implemented as an application that operates over an IMS

[2]Note that the entries in this table represent only the most common cases. Individual systems might use some variations or extensions of these models and languages.

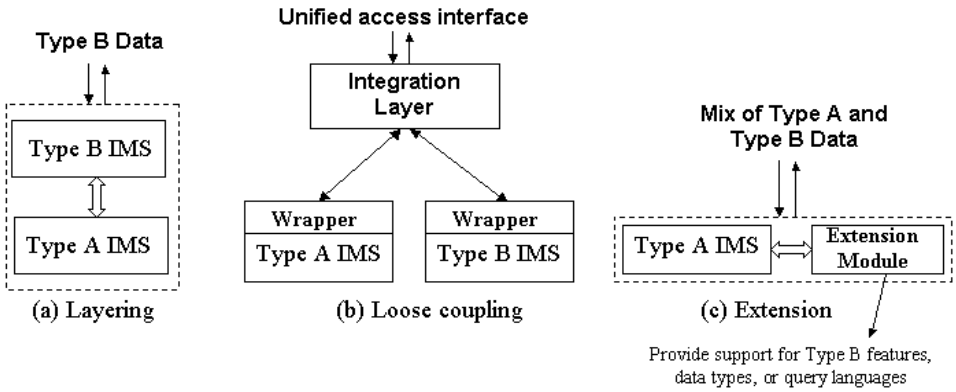

Figure 1: Common integration architectures

	Text Retrieval Relational/OO	*Text Retrieval Semistructured*	*Relational/OO Semistructured*
Layering	[6][16][41][46][7][21] [38][39][52][20]	[42]	[24][51][54][13] [30][49][28]
Loose Coupling	[40][56][9][11][12][26][27][17]	[3]	[45][14]
Extension	[44][32][34][19][18][35][21][23] [53][47][22][57][48][55][37]	[35][33][31][25] [4][36]	[29][15][50][43][10]

Table 2: Classification of literature on integrating different types of information management systems

of another type. The main advantage of this approach is that the top-level IMS can leverage the facilities of the underlying IMS (e.g., concurrency control, recovery, caching, index structures, etc.), without significant additional development time and effort. However, the challenge lies in efficiently mapping the data types and operators used by the top-level IMS in terms of the types and operators supported by the underlying IMS.

Loosely coupled architectures isolate the integration logic in a separate integration (or mediation) layer (Figure 1(b)). This layer provides a unified access interface to the integrated system using its own data and query languages. The fundamental challenge in this architecture is to design efficient mechanisms to translate queries expressed in the unified model in terms of the query capabilities of the individual IMSs. The advantage is that unlike the other two architectures, modifications to the individual IMSs are minimal or completely unnecessary.

Finally, *extension* architectures (Figure 1(c)) enhance the capabilities of a particular type of IMS by using an extension module that provides support for new data types, operators, or query languages usually available only in IMSs of another type. When extension interfaces are available in the original IMS (as is the case with most commercial relational systems [19, 18]), the extension module can be implemented using these interfaces. Otherwise, the original IMS is modified to natively support the new features.

Note that even though we have explicitly distinguished between these three architectures to help navigate the literature, actual implementations sometimes have flavors of more than one architecture. For instance, reference [21] proposes *extensions* to a relational DBMS to facilitate efficient implementations of inverted indexes and then *layers* a text retrieval system atop the enhanced RDBMS. Similarly, for efficiency reasons, some systems push the integration layer of the *loosely coupled* architecture into one of the individual systems [37].

3 Text Retrieval and Relational/Object Database Systems

The integration of information retrieval (IR) and traditional database systems has long been recognized as a challenging and difficult task. Fundamental differences in the query and retrieval models (precisely defined declarative queries and exact answers in databases versus imprecise queries and approximate retrieval in IR systems) have resulted in vastly different query languages, index structures, storage formats, and query processing techniques. Both the IR and DB communities have attempted to address this problem, but with different goals, and by adopting different architectures. The database community has favored the *extension* architecture with the aim of efficiently providing IR features within the DBMS framework (lower left cell of Table 2). In contrast, the IR community has shown a preference for the *layered* architecture, with the aim of exploiting DBMS features (concurrency control, recovery, security, transaction semantics, robustness, etc.) to build more scalable and robust text retrieval systems (top left cell of Table 2). Though less popular, there has also been some work in building loosely coupled IR-DB systems [17]. In the interest of space, we do not discuss such loosely-coupled systems in this section.

3.1 Extensions to Database Systems

The extension-based approach of the DB community has led to work on extended relational models and algebras, extensions to database query languages, new index structures [44] and data types [53], and query execution strategies for optimizing IR-style text operations.

Extensions to the relational model fall into two categories - nested (non-first normal form or NF^2) relational models [22, 48, 47, 57] to capture hierarchical document structure and probabilistic models [32, 34] to incorporate uncertainty and imprecision into the DBMS framework. Such probabilistic extensions, though promising, require substantial changes to the core query processing algorithms of database systems and as a result, are not yet a part of actual implementations.

Reference [21] proposes the technique of cooperative indexing, where the database is responsible for inverted index storage and access but the IR extension defines the actual contents of the index. References [37] and [23] address performance issues in implementing database extensions - specifically, extensions to integrate with external (i.e., outside the database system) text retrieval systems. The SQL/MM Full-Text [53] standard attempts to standardize the integration of text retrieval with SQL database systems by providing definitions for text-related abstract data types. Finally, in [35], Goldman et al. propose an extension to relational and object databases by introducing a proximity operator, called the NEAR operator, adapted from the notion of textual proximity used in text retrieval systems.

3.2 Layering IR Systems atop Relational/Object Database Systems

Some of the earliest attempts at integrating IR and DB systems treated a text retrieval system as a database application that was implemented on top of a standard relational DBMS [6, 16, 41, 46, 39, 38]. The inverted index, the lexicon, and other term frequency statistics were stored in standard database tables. IR queries were translated into SQL queries over these tables and executed by the database. In addition, several prototype text retrieval systems have also been built using object database systems [7, 20, 52]. Since the object data model natively supports nesting, in addition to collection types and sets, we expect that systems for content-based retrieval of structured documents could be effectively implemented on top of OODB systems.

4 Text Retrieval and Semistructured Database Systems

XML has emerged as the de facto standard for representing semistructured information and for exchanging structured data between applications. Since XML shares the same graph-based data model as several other semistructured database query languages [2, 8], to date, most of the work on querying, indexing, and searching XML corpora has its origins in the database community (see Section 5). In [33], Fuhr and Grossjohann refer to this approach as the *data-centric* view of XML.

However, the alternative *document-centric* [33] view has only recently received the attention of the research community. This approach treats an XML corpus as a collection of logically structured text documents. By extending IR models and indexes to encode the structure and semantics of XML documents, it becomes possible to apply well-known IR techniques and support keyword searches, similarity-based retrieval, automatic classification, and clustering, of XML corpora.

In [31], the authors describe an approach for integrating keyword searches with XML query processing. They extend the XML-QL query language by introducing a new "contains" predicate for keyword-based search operations. They define the precise semantics of the extended query language and describe how to efficiently execute queries that involve keyword search as well as non-text operations. In the same vein, reference [33] describes XIRQL, an extension to the XQL query language to support IR-related features such as weighting, ranking, relevance-oriented search, and vague predicates. The Xyleme warehousing system [4] supports text search and pattern matching operations over XML documents.

As is evident from the descriptions in the previous paragraph, much of the work in XML-IR integration is based on the *extension* approach (Figure 1(c)), adding IR-style features to XML databases and query languages. As an example of a *layered* system, in [42], Hayashi et al. describe their implementation of a relevance ranking based XML search engine on top of a standard text retrieval system. Finally, in [3], Adar describes a personal information management system that is implemented by loosely coupling the Lore semistructured database system with the Haystack personal information retrieval system.

5 Relational/Object and Semistructured Database Systems

With the advent of XML as the dominant standard for information interchange, the integration of XML with relational and object database systems is an extremely active research area. Techniques for mapping the XML (semistructured) data model to the relational or object data model, for exporting relational data as XML documents, for providing XML views of relational data, and for extending relational query engines to process queries over XML data, are topics currently being investigated by the research community.

In the commercial arena, most major relational database vendors already provide support for an XML data type to natively store and manage XML documents, as well as some primitive programming APIs for importing and exporting XML documents to and from database tables [15].

In the interest of space, for the rest of the section, we shall discuss only extension and layered approaches to integrating relational/object and semistructured database systems. However, there has been some work in loosely coupling such systems [45, 14], motivated mainly by the use of XML as the integration language.

5.1 Extensions to Relational/Object Database Systems

Since XML is intended as a language for inter-enterprise information interchange, it is natural that techniques for publishing relational data as XML documents are in great demand. Several commercial tools already provide this functionality, but with some limitations. For instance, Oracle's XSQL [15] tool generates a fixed canonical mapping of the relational data into XML documents, by mapping each relation and attribute to an XML element, and nesting tuple elements within table elements. However, it is incapable of mapping to arbitrary XML DTDs. IBM's DB2 XML Extender supports a language for composing relational data into arbitrary XML as well as to decompose XML documents into relations.

In general, there are two parts to designing a system for publishing relations as XML documents. The first is the need for a language to specify the conversion/mapping from relations to XML documents. The second is an efficient implementation strategy to actually carry out the conversion. The SilkRoute system described in [29] was one of the earliest research prototypes that supported automatic XML generation from relational tables. SilkRoute used a language called RXL, based on a combination of SQL and XML query languages, for specifying mappings of relational tables to arbitrary XML DTDs. Using this language, it is possible to define XML views of the relational data. SilkRoute efficiently executes queries over these XML views by materializing only the portion of the XML that is required to answer the query. In [50], Shanmugasundaram et al. propose a simple language based on SQL with minor extensions for specifying the mappings. They compare different implementation alternatives and report significant performance gains from constructing XML documents as much as possible inside the relational engine.

In [43], the authors describe Ozone, an extension of an object database system to handle both structured and semistructured data. The authors extend the standard ODMG object model and the OQL query language, to handle semistructured data based on the OEM data model and Lorel query language [2].

5.2 Layering Semistructured Database Systems atop Relational/Object Database Systems

To design a database-backed XML repository, one must precisely define (i) a mapping from XML documents to tables or objects, (ii) an algorithm for translating queries over XML documents into SQL or OQL queries over the underlying database, and (iii) a mechanism for translating the result of database query execution into XML. There are several proposals for implementing XML repositories on top of relational [24, 30, 49, 51] and object [54, 13, 28] database systems, differing in their choices for (i),(ii), and (iii).

There are three basic alternatives for mapping XML documents into relational tables. The simplest, and least useful mapping, is to store an entire XML document as a *single database attribute*. Another possibility is to interpret XML documents as *graph structures* and supply a relational schema that can store such graphs [30, 49]. A third approach is to map the *structure* of the XML documents, (e.g., expressed as a DTD) into a corresponding relational schema and to store the documents based on these mappings [51, 24]. Only the last approach allows the repository to fully exploit the query processing and optimization capabilities of the underlying database system.

The STORED system described in [24] uses data mining to separate XML documents into structured and semistructured components. The structured component is stored in a relational database and the semistructured component is stored in a separate overflow semistructured database. In contrast, Shanmugasundaram et al. [51] store the repository entirely within the relational database. They assume that the input XML has an associated DTD and also impose restrictions on the set of DTDs that they can handle.

Techniques for mapping XML documents into object databases tend to be considerably simpler, because the object model naturally supports a hierarchical structure, collection types, and structured types such as sets and lists. Usually, a straightforward analysis of the DTD can be used to generate object type definitions (for example, in ODL) and IDREFs and IDs in the XML document can be mapped to object references and object IDs in the database system [54, 13]. However, there are two key challenges that need to be addressed. First is the fact that OODB systems are generally strongly typed whereas XML, being semistructured, is not. As a result, most often, the object model of the database must be extended, before it can be used for implementing the XML repository. Second, many OODB systems support only simple path expressions whereas most XML query languages include regular path expressions. References [54, 13, 28] address some of these challenges.

6 Conclusions

The design of integrated information management systems that can seamlessly handle unstructured, structured, and semistructured data, is a topic with significant research and commercial interest. In this paper, we presented a short survey of prior work on designing such integrated systems.

A visual inspection of Table 2 allows us to draw some conclusions about major research themes and focus areas. For instance, it is clear that so far, a significant portion of work on integrated information systems has involved layering other systems atop relational/object databases and on extending the capabilities of the latter (four corner cells of Table 2). However, we expect that with recent interest in middleware integration, loosely coupled systems will receive a lot of research attention in the future. Also, as mentioned in Section 4, integration of semistructured data in general, and XML in particular, with the relational/object database world has received a lot more attention than corresponding integration with text retrieval systems (comparing two rightmost columns of Table 2). We believe that this will change, once efforts to incorporate IR-style operators and structured text retrieval models into XML query languages, bear fruit.

In this paper, we have concentrated mainly on integration of pairs of systems. However, the Web provides us with a large and interesting data set that shares characteristics with all three types of data that we have dealt with in this paper. For instance, a repository of Web pages could be treated as a large collection of text documents. It could also be treated as a huge graph database since Web pages link to each other through hypertext links. Finally, each page in a Web repository can be associated with simple attributes (e.g., URL, page length, domain name, crawl date, etc.) that are easily managed in a relational database. Hence, complex queries over such repositories are likely to involve search terms connected by Boolean operators, navigational operators to refer to pages based on their interconnections, as well as predicates on page attributes. We believe that a query processing architecture that can efficiently execute such queries over huge Web repositories is an exciting research topic with applications in Web search and mining.

References

[1] S. Abiteboul. Querying semi-structured data. In *Proc. of the 6th Intl. Conf. on Database Theory*, pages 1–18, January 1997.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *Intl. Journal on Digital Libraries*, 1(1):68–88, April 1997.

[3] E. Adar. Hybrid-search and storage of semi-structured information. Master's thesis, MIT, May 1998.

[4] V. Aguilera, S. Cluet, P. Veltri, D. Vodislav, and F. Wattez. Querying XML documents in Xyleme. In *Proc. of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, July 2000.

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, New York, 1999.

[6] D. C. Blair. An extended relational document retrieval model. *Information Processing and Management*, 24(3):349–371, 1988.

[7] E. W. Brown, J. P. Callan, W. B. Croft, and J. E. B. Moss. Supporting full-text information retrieval with a persistent object store. In *4th Intl. Conf. on Extending Database Technology*, pages 365–378, March 1994.

[8] P. Buneman, S. B. Davidson, G. G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 505–516, June 1996.

[9] M. J. Carey et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. of the 5th Intl. Workshop on Research Issues in Data Engineering - Distributed Object Management*, pages 124–131, 1995.

[10] M. J. Carey, J. Kiernan, J. Shanmugasundaram, E. J. Shekita, and S. N. Subramanian. XPERANTO: Middleware for publishing object-relational data as XML documents. In *Proc. of 26th Intl. Conf. on Very Large Data Bases*, pages 646–648, September 2000.

[11] S. Chaudhuri, U. Dayal, and T. W. Yan. Join queries with external text sources: execution and optimization techniques. In *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pages 410–422, May 1995.

[12] S. Chaudhuri and L. Gravano. Optimizing queries over multimedia repositories. In *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pages 91–102, June 1996.

[13] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pages 313–324, May 1994.

[14] V. Christophides, S. Cluet, and J. Siméon. On wrapping query languages and efficient XML integration. In *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, pages 141–152, May 2000.

[15] Oracle Corporation. XML support in Oracle 8 and beyond. http://www.oracle.com/xml/.

[16] W. B. Croft and T. J. Parenty. A comparison of a network structure and a database system used for information retrieval. *Information Systems*, 10(4):377–390, 1985.

[17] W. B. Croft, L. A. Smith, and H. R. Turtle. A loosely-coupled integration of a text retrieval system and an object-oriented database system. In *Proc. of the 15th Intl. ACM SIGIR Conf.*, pages 223–232, June 1992.

[18] IBM Informix DataBlade. http://www-4.ibm.com/software/data/informix/blades/, 2000.

[19] DB2 Universal Database Extenders. http://www-4.ibm.com/software/data/db2/extenders/, 2000.

[20] A. P. de Vries and A. N. Wilschut. On the integration of IR and databases. In *Proc. of the 8th IFIP 2.6 Working Conf. on Database Semantics*, January 1999.

[21] S. DeFazio, A. M. Daoud, L. A. Smith, J. Srinivasan, W. B. Croft, and J. P. Callan. Integrating IR and RDBMS using cooperative indexing. In *Proc. of the 18th Intl. ACM SIGIR Conf.*, pages 84–92, July 1995.

[22] B. Desai, P. Goyal, and F. Sadri. Non first normal form universal relations: an application to information retrieval systems. *Information Systems*, 12(1):49–55, 1987.

[23] S. Deßloch and N. M. Mattos. Integrating SQL databases with content-specific search engines. In *Proc. of 23rd Intl. Conf. on Very Large Data Bases*, pages 528–537, August 1997.

[24] A. Deutsch, M. F. Fernandez, and D. Suciu. Storing semistructured data with STORED. In *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, pages 431–442, June 1999.

[25] D. Egnor and R. Lord. Structured information retrieval using XML. In *Proc. of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, July 2000.

[26] R. Fagin. Fuzzy queries in multimedia database systems. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–10, June 1998.

[27] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proc. of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, May 2001.

[28] L. Fegaras and R. Elmasri. Query engines for Web-accessible XML data. In *Proc. of the 27th Intl. Conf. on Very Large Data Bases*, pages 251–260, September 2001.

[29] M. Fernandez, W.-C. Tan, and D. Suciu. SilkRoute: Trading between relations and XML. In *Proc. of the 9th Intl. World Wide Web Conf.*, pages 723–745, May 2000.

[30] D. Florescu and D. Kossman. Storing and querying XML data using a RDBMS. *Bulletin of the Technical Committee on Data Engineering*, 22(3), 1999.

[31] D. Florescu, I. Manolescu, and D. Kossmann. Integrating keyword search into XML query processing. In *Proc. of the 9th Intl. World Wide Web Conf.*, pages 119–136, May 2000.

[32] N. Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proc. of the 16th Intl. Conf. on Very Large Data Bases*, pages 696–707, August 1990.

[33] N. Fuhr and K. Grossjohann. XIRQL: A query language for information retrieval in XML documents. In *Proc. of the 24th ACM SIGIR Conf.*, pages 172–180, September 2001.

[34] N. Fuhr and T. Rolleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 15(1):32–66, 1997.

[35] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *Proc. of 24th Intl. Conf. on Very Large Data Bases*, pages 26–37, August 1998.

[36] R. Goldman and J. Widom. Interactive query and search in semistructured databases. In *Proc. of the First Intl. Workshop on the Web and Databases (WebDB)*, pages 42–48, March 1998.

[37] R. Goldman and J. Widom. WSQ/DSQ: A practical approach for combined querying of databases and the web. In *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, pages 285–296, May 2000.

[38] D. A. Grossman and J. R. Driscoll. Structuring text within a relation system. In *Proc. of the 3rd Intl. Conf. on Database and Expert System Applications*, pages 72–77, September 1992.

[39] D. A. Grossman, O. Frieder, D. O. Holmes, and D. C. Roberts. Integrating structured data and text: A relational approach. *Journal of the American Society for Information Sciences*, 48(2):122–132, 1997.

[40] J. Gu, U. Thiel, and J. Zhao. Efficient retrieval of complex objects: Query processing in a hybrid DB and IR system. In *Proc. of the 1st German National Conf. on Information Retrieval*, 1993.

[41] D. J. Harper and A. D. M. Walker. ECLAIR: An extensible class library for information retrieval. *Computer Journal*, 35(3):256–267, 1992.

[42] Y. Hayashi, J. Tomita, and G. Kikui. Searching text-rich XML documents with relevance ranking. In *Proc. of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*, July 2000.

[43] T. Lahiri, S. Abiteboul, and J. Widom. Ozone: Integrating structured and semistructured data. In *Research Issues in Structured and Semistructured Database Programming, 7th Intl. Workshop on Database Programming Languages*, September 2000.

[44] C. A. Lynch and M. Stonebraker. Extended user-defined indexing with application to textual databases. In *Proc. of the Fourteenth Intl. Conf. on Very Large Data Bases*, pages 306–317, August 1988.

[45] I. Manolescu, D. Florescu, and D. Kossman. Answering XML queries on heterogeneous data sources. In *Proc. of the 27th Intl. Conf. on Very Large Data Bases*, pages 241–250, September 2001.

[46] I. A. McLeod and R. G. Crawford. Document retrieval as a database application. *Information Technology: Research and Development*, 2:43–60, 1983.

[47] R. Sacks-Davis, A. J. Kent, K. Ramamohanarao, J. A. Thom, and J. Zobel. Atlas: A nested relational database system for text applications. *Transactions on Knowledge and Data Engineering*, 7(3):454–470, 1995.

[48] H.-J. Schek and P. Pistor. Data structures for an integrated data base management and information retrieval system. In *Proc. of the 8th Intl. Conf. on Very Large Data Bases*, pages 197–207, September 1982.

[49] A. Schmidt, M. L. Kersten, M. Windhouwer, and F. Waas. Efficient relational storage and retrieval of XML documents. In *WebDB (Informal Proceedings)*, pages 47–52, 2000.

[50] J. Shanmugasundaram, E. J. Shekita, R. Barr, M. J. Carey, B. G. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently publishing relational data as XML documents. In *Proc. of 26th Intl. Conf. on Very Large Data Bases*, pages 65–76, September 2000.

[51] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *Proc. of 25th Intl. Conf. on Very Large Data Bases*, pages 302–314, September 1999.

[52] G. Sonnenberger. Exploiting the functionality of object-oriented database management systems for information retrieval. *Bulletin of the Technical Committee on Data Engineering*, 19(1):14–23, 1996.

[53] SQL/MM Full-text - ISO/IEC 13249. http://www.wiscorp.com/sql/sqlfulltext.zip.

[54] R. van Zwol, P. M. G. Apers, and A. N. Wilschut. Modeling and querying semistructured data with MOA. In *Proc. of the Workshop on Semistructured Data and Non-Standard Data Types*, 1999.

[55] S. R. Vasanthkumar, J. P. Callan, and W. B. Croft. Integrating INQUERY with an RDBMS to support text retrieval. *Bulletin of the Technical Committee on Data Engineering*, 19(1):24–33, 1996.

[56] M. Volz, K. Aberer, and K. Bohm. An OODBMS-IR coupling for structured documents. *Bulletin of the Technical Committee on Data Engineering*, 19(1):34–42, 1996.

[57] J. Zobel, J. A. Thom, and R. Sacks-Davis. Efficiency of nested relational document database systems. In *Proc. of the 17th Intl. Conf. on Very Large Data Bases*, pages 91–102, September 1991.