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 [52, 64]. 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 scheme [59, 34]. 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. Table 1 presents a brief summary of the models and languages employed by these systems[2].

There is a substantial body of work that deals with the design of each of these classes of information management systems [64, 34, 2]. 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 need to handle inventory information in a relational database as well as purchase orders received as

[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).

[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.

	Text Retrieval Systems	*Relational/Object DBMS*	*Semistructured (XML) DBMS*
Data Models	Set of unstructured text documents possibly with structured fields	Relational Model, Object Definition Language (ODL)	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 [6]	SQL, OQL [59]	Lorel[3], UnQL [10]

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

(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 [9, 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 [18, 17], 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 [25] 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 also 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 fundamental integration challenges and the common ideas behind 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 biliographic 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 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

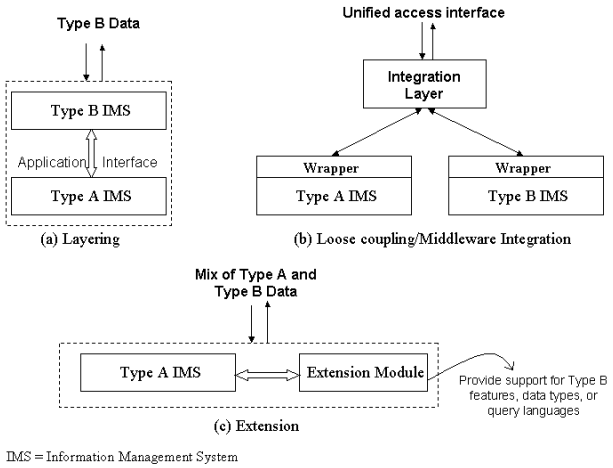

Figure 1: Common integration architectures

Layering	[8][15][41][46] [9][47][20][38] [39][57][19]	[42]	[23][56][60][12] [29][54][27]
Loose Coupling (or) Middleware Integration	[40][62][25][26][16]	[4]	[45][13]
Extension	[44][31][33][18] [17][35][20][22] [58][51][21][65] [53][61][37]	[35][32][30][24] [5][36]	[28][14][55][43][11]
	Text Retrieval Relational/OO	*Text Retrieval Semistructured*	*Relational/OO Semistructured*

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

additional development time and effort. However, the challenge lies in mapping the data types and operators used by the top-level IMS in terms of the types and operators supported by the underlying IMS, so as to maximize query performance.

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 [18, 17]), 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 [20] 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 information management systems. As an example, the WSQ system [37] *extends* a relational DBMS to support specialized virtual tables and a new technique for asynchronous query execution. This extended DBMS is then *loosely coupled* with Web search engines to enable Web-supported database queries.

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).

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 [58], 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 [21, 53, 51, 65] to capture hierarchical document structure and probabilistic models [31, 33] 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 [20] proposes the technique of cooperative indexing, a framework for scalable integration of IR and database systems. In this approach, the IR extension components define how documents are processed, how index terms are extracted and stemmed, and the kinds of information that are associated with each index entry. The database is responsible for index storage and for providing efficient access to the index.

References [37] and [22] address performance issues in implementing database extensions - specifically, extensions to integrate with external (i.e., outside the database system) text retrieval systems. In [22], Deßloch and Mattos discuss a query rewrite scheme based on table functions that is used to efficiently pass query results from standalone external text search systems into the database engine. In [37], Goldman and Widom propose a technique called *asynchronous iteration* that enables high concurrency between database query processing and high-latency calls to Web search engines. The SQL/MM Full-Text [58] standard attempts to standardize the integration of text retrieval with SQL database systems by providing definitions for text-related abstract data types.

In [35], Goldman et. al. propose an extension to relational and object databases by introducing a proximity operator, called the NEAR operator. They adapt the notion of textual proximity from text retrieval systems and apply it to proximity between nodes in a graph.[3]

[3]Note that OO models are inherently graph-based and that a relational schema with foreign-key constraints can be easily translated into a graph.

3.2 Layering IR systems atop 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 [8, 15, 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 [9, 19, 57]. 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.

More recently, Melnik et. al. [47] demonstrate the use of an embedded database system (such as Berkeley DB [50]) to store and manage inverted indexes. The performance results in [47] indicate that for application scenarios when a full-fledged heavyweight client server database is not necessary, an embedded DBMS can act as an efficient storage manager for inverted indexes.

3.3 Loosely-coupled IR-DB Systems

Even though layering and extension seem to be the favored architectures for integration of IR and Database systems, there has been some work in building loosely coupled IR-DB systems. For example, in [16], Croft et. al. describe their design for loosely coupling the INQUERY IR system and IRIS, a prototype object DBMS. The IRIS and INQUERY systems are coupled externally using a "control module" that corresponds to the integration layer shown in Figure 1(b).

One of the key challenges in building loosely coupled IR-DB systems is the design of efficient algorithms to merge ranked results (from the IR system) with the unranked result sets returned by a the database system. In [26], Fagin proposes the use of fuzzy sets as a unified model for representing merged result sets and describes efficient algorithms to perform the merge operation.

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 [3, 10], to date, most of the work on querying, indexing, and searching XML corpora has its origins in the database community (see Section 5). In [32], Fuhr and Grossjohann refer to this approach as the *data-centric* view of XML.

However, the alternative *document-centric* [32] 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 [30], 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 [32] describes XIRQL, an extension to the XQL query language to support IR-related features such as weighting, ranking, relevance-oriented search, and vague predicates. In [5], the authors describe the query processing architecture of their Xyleme XML warehousing system, which includes support for text search and pattern matching.

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 [4], 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.

In addition to the systems level integration work referenced in Table 2 and described so far in this section, there are distinct similarities between semistructured query models and models for structured text retrieval [7]. In particular, Ricardo Baeza-Yates and Gonzalo Navarro [7, 49, 48] demonstrate that the *proximal nodes* model for structured text retrieval addresses all of the complex operations of the XQL query language and can provide useful techniques for efficiently executing XQL queries.

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 [63]. 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 [14].

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, 13], motivated mainly by the use of XML as the middleware 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 [14] 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 [28] 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 [55], 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 [3].

5.2 Layered systems

To design a database-backed XML repository, one must precisely define (i) a mapping from XML documents to tables or objects, (ii) 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 executiong into XML. There are several proposals for implementing XML repositories on top of relational [23, 29, 54, 56] and object [60, 12, 27] 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 [29, 54]. A third approach is to map the *structure* of the XML documents, (for e.g., expressed as a DTD) into a corresponding relational schema and to store the documents based on these mappings [56, 23]. 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 [23] 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. [56] 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 [60, 12]. 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 [60, 12] and [27] describe potential solutions to 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 *Proceedings of the 6th Intl. Conferene on Database Theory*, pages 1–18, January 1997.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kauffman Publishing, San Francisco, 1st edition, 1999.

[3] 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.

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

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

[6] R. Baeza-Yates and B. Riberio-Neto. *Modern Information Retrieval*. Addison-Wesley, New York, 1 edition, 1999.

[7] R. A. Baeza-Yates and G. Navarro. Integrating contents and structure in text retrieval. *SIGMOD Record*, 25(1):67–79, 1996.

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

[9] 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.

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

[11] 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 *Proceedings of 26th Intl. Conf. on Very Large Data Bases*, pages 646–648, September 2000.

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

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

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

[15] 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.

[16] 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 *Proceedings of the 15th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 223–232, June 1992.

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

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

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

[20] 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 *Proceedings of the 18th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 84–92, July 1995.

[21] 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.

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

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

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

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

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

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

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

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

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

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

[32] N. Fuhr and K. Grossjohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of the 24th ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 172–180, September 2001.

[33] 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.

[34] H. Garcia-Molina, J. Ullman, and J. Widom. *Database System Implementation*. Prentice-Hall, New Jersey, 1st edition, 2000.

[35] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *Proceedings 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 *Proceedings 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 *Proceedings of the 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 *Proceedings 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 *Proceedings 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 *Proceedings 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 *Proceedings 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] S. Melnik, S. Raghavan, B. Yang, and H. Garcia-Molina. Building a distributed full-text index for the web. In *Proceedings of the 10th Intl. World Wide Web Conf. (WWW10)*, pages 396–406, May 2001.

[48] G. Navarro and R. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[49] G. Navarro and R. A. Baeza-Yates. A language for queries on structure and contents of textual databases. In *Proceedings of the 18th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 93–101, July 1995.

[50] M. Olson, K. Bostic, and M. Seltzer. Berkeley DB. In *Proc. of the 1999 Summer Usenix Technical Conf.*, June 1999.

[51] 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.

[52] G. Salton. *Information Retrieval: Data Structures and Algorithms*. Addison-Wesley, Reading, Massachussetts, 1989.

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

[54] 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.

[55] 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 *Proceedings of 26th Intl. Conf. on Very Large Data Bases*, pages 65–76, September 2000.

[56] 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 *Proceedings of 25th Intl. Conf. on Very Large Data Bases*, pages 302–314, September 1999.

[57] 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.

[58] SQL multimedia and application packages: Part2 (full-text) - ISO/IEC 13249. http://www.wiscorp.com/sql/sqlfulltext.zip, 2000.

[59] J. Ullman and J. Widom. *A First Course in Database Systems*. Prentice-Hall, New Jersey, 1st edition, 1997.

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

[61] 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.

[62] 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.

[63] J. Widom. Data management for XML: Research directions. *IEEE Data Engineering Bulletin*, 22(3):44–52, 1999.

[64] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Morgan Kauffman Publishing, San Francisco, 2nd edition, 1999.

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