Reaserch proposal submitted for approval as a plan for a Ph.D. thesis

Oblivious Querying over Data with Irregular Structure

Yaron Kanza

1 Introduction

In conventional data models (e.g., relational, object-oriented), it is common to separate the *schema* from the *data*. The schema defines the structure of the data, and consequently imposes constraints on the data that may appear in the database. New data can be added to a database only if these new data satisfy the constraints imposed by the schema of the database. Traditional database systems, such as relational database systems (RDBMS) and object-oriented database systems (ODBMS), were used to hold structured data, i.e., data that conform to a given schema (or to object definitions in the case of ODBMS).

In the World-Wide Web, data do not always conform to a strict schema for several reasons. First, data on the Web are stored not only in relations, but also in files according to many different formats. For example, the Web includes HTML files, XML files, image files (e.g., gif, jpeg), voice files (e.g., mp3, wav), movies (e.g., mpeg) and other formats. Secondly, the Web is an environment where many different users contribute data, and not all of these data conforms to the same schema definitions. Thirdly, in many Web sites, the data is changing continuously. The changes are not only in the data items, but also in the way that those items are formatted or structured. For these reasons, the management of data on the Web requires tools that are capable of dealing with unstructured data and with data whose structure is frequently changing.

Data that has no rigid structure and does not conform to a strict schema, such as data on the Web, is called *semistructured data* [Abi97, Bun97]. A semistructured database is essentially described as a rooted labeled directed graph. In the database, a node represents an object and a label edge links a node to another one if the second node is an attribute filler for the first node. In the graph, nodes that have no edge emerging from them are called *atomic nodes*. Atomic values are attached to the atomic nodes. In semistructured data, there is no distinction between the data and the schema. The structure of the data is described by the graph of the data.

The data model for semistructured data has also been known as the Object Exchange Data Model (OEM) [PGMW95]. Recently, XML [BPSMM00] has become an accepted format for representing semistructured data, and for exchanging semistructured as well as structured data.

The graph representing semistructured data is self describing in the sense that it contains information about the structure of the data. This information is sufficient, for example, for producing a logical view of and a common interface to data that are physically stored in different formats. In addition, a semistructured data model facilitates the representation of data that are the result of integrating many heterogeneous sources that lack a common structured schema.

Semistructured data can be incomplete in the sense that there can be missing values in the data. For example, a database that represents informations about cars for sale may include the requested price only for some, but not all of the cars.

Furthermore, semistructured data can be organized by different hierarchies in different parts of the database, as will be explained in this paragraph. A graph representing semistructured data has nodes that represent entities, and the label on an edge that enters a node is the *entity type* of that node. Two nodes are semantically related if they are connected by a directed path. However, the direction of the path is not predetermined by the entity types. For example, in the cars-for-sale database, the manufacturer-information node can be either an ancestor or a descendant of the car node. Moreover, it could also be that in the same database graph some pairs of manufacturer-information nodes are ancestors of car nodes while others are descendents of car nodes.

We say that a database has an *irregular structure* when either it is incomplete or has different hierarchies between elements. When a user queries a semistructured data, two major problems arise. The first problem is the irregular structure in the database graph and the second problem is the lack of knowledge about the structure of the database. Next, we explain these two problems.

In traditional query languages, when a user searches for an object in a semistructured database, she should state in the query where to find the requested object in the database graph. The user can state a path expression that expresses where to find the requested object, or she can supply a pattern that is matched to the database graph, and nodes from the database are mapped to nodes in the pattern according to the matching. Matchings of the query to the database produces assignments which are used to construct the result of the query.

Either when the user supplies a path or when she supplies a graph pattern there must be some knowledge about the paths in the database graph or at least about the patterns of those paths. However, if the semistructured database is irregular, then objects with the same type are reachable from the root of the database by different paths. In this case, stating merely one path or one pattern will not produce all the desired results. When the user does not have information about the structure of the database or when the information about the structure of the database is incomplete, using a wrong pattern or a wrong path expression can cause not to get the desired result, because the structure of the data graph does not match the supplied pattern.

When querying semistructured data, it is essential to be able to deal with incomplete information about the structure of the data and also to be able to cope with irregular structure in the database.

We propose to study oblivious querying of semistructured data with irregular structure. The concept "oblivious querying" means that the user can query the database without having a full knowledge of the following issues.

(a) The structure of the data.

(b) The attributes of each element in the database.

Sometimes, in addition to the above two points, oblivious querying also entails that there is no knowledge about the set of labels in the database graph ("ontology"). To deal with the lack of knowledge about the structure of the data, we have developed queries with flexible semantics. To deal with the lack of knowledge about the exact attributes of each element in the database, we have developed queries with incomplete answers. These two are explained later in this proposal. For an oblivious querying of a database, it is important to combine flexible queries with queries that return incomplete answers. Further research needs to be done on how to query when the ontology of the database is not fully known.

2 Related Work

In the next section, we will describe some of the work that has been done on querying semistructured data and querying databases with incomplete information.

Recently extensive work has been done on querying semistructured data [Abi97, Bun97, ABS00]. Initially, much of the work on semistructured data was motivated by, first, the need to query the World-Wide Web, and secondly, the need to deal with data that is integrated from various sources [GMQP+95, PGMW95, QRS+95]. For data integration, an architecture of *wrappers* and *mediators* was suggested in [Wie92].

Currently, there is a growing interest in techniques and languages for querying XML documents [BPSMM00]. XML documents are becoming a standard for representing semistructured data, and are used, in addition, for data exchange between applications.

2.1 Querying Semistructured Data

Several languages were developed for the management of semistructured data. Next, we will overview some of these languages.

Lorel [AQM+96, AQM+97, MAG+97, QWG+96] is one of the first languages devised for semistructured data. It is based on a method of binding variables in the query to objects in the database using navigation over the database graph. A different approach for querying semistructured data was of the query language UnQL [BDS95, BDHS96, BDFS97]. In UnQL, *structural recursion* was used to traverse the database graph and change it according to specified conditions.

In Strudel [FFK+98] and in Quest [BYKK+99], pattern matching was used to extract data from the database and Skolem functions were used for the creation of the result graph. In a pattern matchings of a query graph to a database graph, database objects are matched to query variables. From the matchings, the query result is constructed.

One motivation for querying semistructured data was to develop query languages for the World-Wide Web. Early languages for querying the Web were designed to extract information from HTML pages on the Web. Some of the early languages that were proposed for querying HTML pages on the Web were W3QL [KS95, KS97], WebLog [LSS96], WebOQL [AM98] and Araneus [AMM97, MAM+98]. Complexity of queries over the Web was investigated in [AV97a, MM97, MMM97].

2.2 Querying XML

In recent years, increasing amounts of data have been stored, exchanged, and presented using XML [BPSMM00]. XML has become a common format for representing semistructured data. The ability to easily and efficiently query XML data sources has become increasingly important. An extensive effort is being done for developing query languages for XML and tools to manage XML data. Two major research questions are as follows. First, how to store and query XML files efficiently. Secondly, how to give users the ability to easily formulate their queries.

One of the great strengths of XML is its flexibility in representing many different kinds of information from diverse sources. To exploit this flexibility, an XML query language must provide features for retrieving and interpreting information from these diverse sources.

Several languages were suggested for querying XML documents. XML-QL [DFF+99] is a query language that extracts information from the data by pattern matching. In XQL, the query describes a path expression and the result of the query consists of all elements in the document that are reachable by a path that matches the given path expression of the query. Recently, XQL has been replaced with XPath [DeR99], which is more a language for path expressions than a full-fledged query language. Currently, in an effort to reach a standard query language for XML, the World-Wide Web consortium is developing XQuery [CCF+01], which is an extension of a language called Quilt. Essentially, both XQuery and Quilt combine pattern matching as in XML-QL with path expressions of XPath.

In addition to storing data, XML has also been proposed for data integration and data exchange. MIX [BGL+99] and Yat [CDSS98] have been designed as tools for the support of information mediation with XML. Recently, a work on representing incomplete data with XML has been presented in [ASV01]. Dealing with incomplete information has a major importance when integrating data from heterogeneous sources.

In all the query languages described hitherto, the user must have quite a lot of information on the structure of the documents she wants to query. For example, knowing just the names of elements and attributes is not enough—one should also have some information about the hierarchy of elements in the documents in order to formulate a query.

It is not always easy for a user that queries XML documents to study the details of the structure of those documents. In the query language EquiX [CKK+99, CKK+00], querying is a process done by means of a graphical user interface. In this process, the document is explored using a query form, and the elements of the document are being shown to the user as she opens nodes in the form. The user marks nodes to be included in the output and adds quantifiers and constraints to the query form in order to formulate the query. EquiX provides a very expressive query language with negation and quantification, and yet allows to formulate easily queries without a full advance knowledge of the graph structure of the document being queried.

Another approach for querying XML documents without a knowledge of the hierarchy of elements is the *flexible semantics* used in SQL4X [CKS01]. In the flexible semantics, two elements in a document are deemed as semantically related if they are connected by a path in the document. When a user formulates a query under the flexible semantics, the user does not have to know the full hierarchy of the elements in the document. Instead, the query contains the requested elements and requested relationships between these elements (where a "relationships" means that the two elements are connected by a directed path in the database graph).

In several query languages for XML and for semistructured data, *regular path expressions* are being used in the query. A regular path expression describes a set of paths using a regular expression. Using a regular path expression in a query causes the evaluation runtime of the query to be exponential [AV97b]. Furthermore, using regular path expressions to cover a disjunction of possible structures in the database may cause the size of the query to be very large. These reasons make it hard to use regular path expression as a solution to the problem of querying with only partial knowledge about the structure of the database.

2.3 Incomplete Information

From an early stage in the development of relational databases, it was recognized that incomplete information should be dealt with. Codd noted that available data are sometimes incomplete and this led to the introduction of a special value, *null*, to indicate the absence of any value different from null. The null value was then used to represent partial information, i.e., information that is incomplete due to missing values [Cod75, Cod79].

Codd's observation motivated the development of models of partial information by Lipski [Lip79] and by others [Zan84, Rei86, Var86], defining semantics for queries applied to incomplete databases. More sophisticated formalisms for storing information in tables, which were based on the idea of annotating null values with various types of conditions were suggested in [IL84, AG85].

The semantics of incompleteness is usually viewed from two perspectives, either a *closed-world assumption* (CWA) or an *open-world assumption* (OWA). The CWA states, intuitively, that nothing holds unless explicitly stated in the database. The OWA, on the contrary, states that anything that is not ruled out by the information in the database is possible.

Dealing with incomplete information in the database changes the view of what a database is. A database that contains complete information about a subset of the world can be regarded as a *model* of (a fragment of) the world. However, a database with some form of incompleteness (e.g., missing facts, missing attribute values, etc.) cannot be regarded as a single model of the world, and instead can be seen as a state that complies with some models and does not comply with other models of the world. In the formal semantics expressed in terms of logical theories, this distinction is captured by the fact that a "complete" database is associated with a single model (or unique minimal model) whereas an "incomplete" database can have several models, and possibly even infinitely many.

Since there is no single model of the world, when dealing with incomplete information, it is necessary to revisit the notion of a query and of the answers to a query. If a database has more than one minimal model, one defines a *possible answer* as a tuple of values that satisfies the query in at least one minimal model whereas a *certain answer* is one that satisfies the query in all minimal models [Lip79]. Imielinski and Lipski introduced the notions of *strong representation systems* and *weak representation systems*. In the former, for every query over an incomplete database there exists a theory that represents all possible answers, while in the latter only certain answers can be represented [IL84]. The complexity of query answering over various types of databases with incomplete information was studied in [AKG91, Mey97].

Incomplete information is sometimes a result of data integration from heterogeneous sources. When information from several relations is integrated, it is not always clear what the result of the integration should be. In [GL94, RU96], integration of relations is described in terms of a *full disjunction* where outer joins are used for the computation of data integration.

In the context of querying data on the Web, where the data often tend to be irregular, queries that allow for incomplete answers have been investigated [KNS99]. In viewing the Web (or other semistructured data) as a graph, queries are essentially patterns that are matched to the graph to extract information. Different semantics for such queries were defined and the complexity of the query evaluation was examined. Also, in [KNS99], query containment for queries with incomplete answers has been characterized and analyzed.

Recently, [ASV01] has studied queries over an XML data warehouse that represents incomplete information. Essentially, in this work, a representation system as in [IL84] was defined and applied to XML documents instead of applying it to relations.

3 My M.Sc. Thesis

In my M.Sc. thesis, I have studied the theory of queries with incomplete answers over semistructured data. The theoretical results of this research have been published in [KNS99]. In the work that was presented in [KNS99], we defined different semantics for queries with incomplete answers over semistructured data and we studied the principles of such queries. As a data model we used a labeled directed graph, as in the OEM data model [PGMW95]. Our queries, too, were defined by graphs, which were matched to the database.

In an abstract view of [KNS99], database queries consists of a set of variables and constraints on the variables. A *solution* to a query is a binding of the variables in the query to objects in the database, such that the constraints in the query are satisfied. In order to accept as a solution also assignments that do not bind all the variables of the query to database objects, the solutions are refined. The constraint in the query are divided to *search constraints* and to *filter constraints*. The search constraints form a labeled directed graph whose nodes are the variables of the query. The graph is the pattern that has to be matched by some part of the database. We are only interested in *maximal assignments*, because they contain maximal information. The maximal assignments are filtered and those that satisfy the filter constraints are the solutions. We distinguish between *strong* and *weak* filter constraints. A strong constraint is satisfied by an assignment if all variables of the constraint are bound and their values comply with the constraint, while a weak constraint is already satisfied if one of its variables is not bound by the assignment. Hence, it is important that only maximal matchings are filtered, since this enhances the applicability of the weak constraints. Finally, solutions are restricted to a subset of their variables, the *output variables*. Those restrictions are called *answers*. The roles of the different components of a query in our model can be illustrated by setting up an analogy with SQL-queries. The FROM-clause is the analog of the search constraints, the WHERE-clause the analog of the filter constraints, while the SELECT-clause specifies the output variables. Thus, with its search and filter constraints, a query is conceptually evaluated in two phases: the first being structural matching for the retrieval, and the second filtering. Note that the fuller an assignment is, the more filter constraints it has to satisfy.

In [KNS99], we have defined several semantics for queries that return incomplete answers. We have developed algorithms for evaluating such queries. These algorithms produce the set of maximal assignments to a query according to the semantics of the query. In addition, in this work the complexity of the algorithms was analyzed and soundness and completeness of the algorithms were shown.

We have also studied the complexity of evaluating queries that include *filtering constraints*. The filtering constraints are constraints that are being applied to the assignments produced by the graphs pattern matchings. We have shown that in many cases adding filtering constraints to the queries makes the evaluation of the queries NP-Complete.

For the purpose of optimization, it is important to be able to tell whether two queries are equivalent. For this reason, we have studied in [KNS99] containment of queries with incomplete answers. For queries in the different semantics, characterization of query containment was given. Furthermore, the complexity of deciding containment was analyzed.

During the time I was working on My master thesis, I was involved in a project named Quest to develop a system that enables querying of HTML documents over the Internet. The implementation of Quest was a joint work with Yakov Kogan and Ziv Bar-Yossef. In Quest we have implemented part of the theory that is presented in [KNS99]. Quest is one of the earliest projects on query languages that were designed to query documents on the World-Wide Web [BYKK+99]. Note that both [KNS99] and [BYKK+99] were the first papers that were published on queries with incomplete answers over semistructured data, and they preceded the more recent work of [ASV01].

The main concept in Quest is to add *semantic tags* to HTML documents. The semantic tags are used to mark the semantic meaning of different parts of the documents. By adding semantic tags to the HTML documents, objects and links between the objects are defined in the documents. As a result of the tagging, a semistructured database is created from the documents.

Queries in Quest are essentially built from three different components—a graph, a set of constraints and a set of templates. When querying, the graph of the query is matched to the database using a graphs pattern matching. Matching the query graph to the database graph produces a set of assignments. Each assignment is a partial function that maps database objects to the query variables, such that the constraints that the query graph imposes are satisfied. To suit semistructured data, Quest has the ability to deal with partial matchings (i.e., matchings in which some of the variables of the Query are assigned a null value instead of a database object). Intuitively, a matching is *maximal* if there is no other matching that *subsumes* it, i.e., one cannot replace some null values with database objects and still get a matching. When matching a query graph of Quest to a database, we get a set that includes all maximal assignments of the query with respect to the database.

In the second phase of a Quest query, the filter constraints of the query are applied to the assignment from the matching phase. The third phase of a Quest query creates HTML document as the query result, using the templates in the Query. One of the novelties of Quest with respect to other query languages was in the ability to deal with incomplete information.

The work an querying databases with incomplete information was submitted by me as a thesis for M.Sc. degree. I intend to include this work in my Ph.D. thesis and to further extend the ideas and techniques that are presented in this work. By integrating queries with incomplete answers and flexible queries that will be discussed next, we intend to achieve a powerful query language that is suitable for oblivious querying of semistructured data with irregular structure.

4 Preliminary Results

So far, we have only studied one aspect of oblivious querying over data with irregular structure—the aspect of querying incomplete information. However, databases that have different hierarchies between elements are another aspect of irregularity in the data. This aspect must also be dealt with when querying irregular databases, or data for which it is unknown whether it has regular or irregular structure.

In the introduction, we have discussed some of the problems that may arise in querying irregular semistructured data. In summary, in semistructured data the following problems may cause querying be difficult. First, the structure of the data can be unknown to the user. Secondly, the structure of the data can change from time to time. Thirdly, the schema of the database can be very large and full with details. Queries that are based on a schema require that the user will know many details about the schema. Another problem is when there is no schema to the data and there are different hierarchies between elements in different parts of the database. Thus, queries that allow for incomplete answers are not always powerful enough to query databases with irregular structure.

Due to the reasons mentioned above, we have developed *flexible* and *semiflexible* queries. In [KS01], we have presented a work on flexible and semiflexible queries. The usage of flexible (semiflexible) semantics allows a user to query a semistructured database when there is incomplete information about the structure of the data. Intuitively, for writing a query with flexible semantics the user is required to know the names of elements in the document (*ontology*), whereas non-flexible (rigid) queries require a user to know the path to a requested element in the database, or at least to know the hierarchy between elements in the database, otherwise, the query will not give all the desired results.

The work in [KS01] first defined queries with flexible semantics and queries with semi-flexible semantics. These two semantics are suitable to ontology-based querying of semistructured data and we believe that they capture the intended meaning of many common queries. In this work, queries are represented as labeled directed graphs. However, a user only needs to know the ontology of the database in order to phrase a query. Hence query formulation is more intuitive and simpler in comparison to query languages that *rigidly* match queries to databases, or query languages that use regular expressions.

In a *rigid matching* semantics, the implicit relationships between objects are expressed by labeled edges. Complex relationships have to be constructed explicitly by formulating queries in terms of labeled directed graphs. The semiflexible semantics, in comparison, also uses labeled directed graphs to exhibit relationships between objects, but assumes that database objects are implicitly related if they are connected by a directed path (and not just by an edge). Thus, a path π of the query can match a path ϕ of the database if ϕ includes all the labels of π. However, the inclusion needs not be continuous or in the same order as in π. The flexible semantics further extends this idea by applying transitivity to the implicit relationship that holds under the semiflexible semantics.

In the work, we presented the essentials of the data model. We also defined the new semantics and discussed some basic properties of semiflexible semantics. For query evaluation under semiflexible and flexible semantics, we provided algorithms and analyzed the runtime complexity of these algorithms. In addition, we characterized containment and equivalence of queries.

Under the flexible and the semiflexible semantics, two databases could be equivalent (even in non trivial cases) in the sense that they give the same results for all queries. We have characterized equivalence of databases with respect to flexible and semi-flexible queries. In addition, we have characterized when a database has an equivalent tree database. Algorithms for transforming a database that has an equivalent tree database to its equivalent tree were given for both the flexible and the semi-flexible semantics. Transforming a database to a tree is important for several reasons. First, query evaluation over a tree database is more efficient than query evaluation over databases that are not trees. Secondly, when the database is an XML document we may not want to have references in the document, because not all tools for XML support references in XML documents. Thirdly, it is natural, when using a graphical tool, to present a tree while presenting graphs that are not trees can be much more cumbersome.

5 Future Work

Our goal in to further develop queries with flexible and semi-flexible semantics in order to facilitate oblivious querying of XML documents. We are working on techniques that will allow a user to query an XML document even when the user's knowledge of the document structure is incomplete. For facilitating such an oblivious querying, flexibility in matching the query to the data should be combined with the ability to return incomplete answers. Research should include the followings. First, definitions for queries that combines flexibility and incomplete answers and the semantics of these queries. Secondly, algorithms for the evaluation of such queries and an analysis of the runtime of these algorithms. Thirdly, optimization techniques for efficient evaluation of the queries are needed. Finally, implementing these ideas will help us in testing their benefits.

Recently, using cellular phones as a method to access the World-Wide Web gained popularity. It could be valuable if a cellular device could supply information from databases on the Web when such information is needed. For example, a person who waits for the bus may wish to query the bus timetable in order to know when the next bus is supposed to arrive. For that, a query language suitable for a cellular device is needed. As for now, there is still no query language that is suitable for the limited space and the limited view that a cellular device offers. However, techniques such as flexible queries, queries with incomplete answers and schema exploration in a query are tools that can help in designing a query language in which queries are concise and suitable for cellular devices. We intend to further examine the difficulties that wireless devices impose on using them for querying data. Then we intend to work on a query language or other tools that will allow users to query data with wireless application, in general, and cellular phones, in particular.

Another problem to which we are trying to apply the techniques of flexible queries is the data integration problem. In data integration, information from various sources is combined. The pieces of information from the different data sources do not always have a common structure. Since flexible and semiflexible queries are capable of dealing with varying structures of data, they provide a major part of the solution for how to integrate information. However, flexible and semiflexible queries rely on ontologies. For querying documents from heterogeneous sources that do not have a common ontology, a transformation that creates a uniform ontology in all the documents must be applied to the documents. Then flexible queries that allow for incomplete answers are capable of extracting information from the sources and integrating the data. Further research is still needed to be done on how to transform a set of documents with different ontologies to a set of documents that share one common ontology.

References

[Abi97] S. Abiteboul. Querying semi-structured data. In F.N. Afrati and Ph. Kolaitis, editors, *International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 1–18, Delphi (Greece), January 1997. Springer-Verlag.

[ABS00] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kaufmann, 2000.

[AG85] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *VLDB 1985*, pages 1–12, 1985.

[AKG91] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, 78:159–187, 1991.

[AM98] G.O. Arocena and A.O. Mendelzon. WebOQL: Restructuring documents, databases, and webs. In *Proc. 14th International Conference on Data Engineering*, pages 24–33, Orlando (Florida, USA), February 1998. IEEE Computer Society.

[AMM97] P. Atzeni, G. Mecca, and P. Merialdo. To weave the Web. In *Proc. 23nd International Conference on Very Large Data Bases*, pages 206–215, Athens (Greece), August 1997. Morgan Kaufmann Publishers.

[AQM+96] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. Technical report, The Stanford University Database Group, 1996.

[AQM+97] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[ASV01] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and querying XML with incomplete information. In *Proc. 20th Symposium on Principles of Database Systems*, pages 150–161, Santa Barbara (California, USA), May 2001. ACM Press.

[AV97a] S. Abiteboul and V. Vianu. Queries and computation on the web. In F.N. Afrati and Ph. Kolaitis, editors, *International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*, pages 122–133, Delphi (Greece), January 1997. Springer-Verlag.

[AV97b] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proc. 16th Symposium on Principles of Database Systems*, pages 122–133, Tucson (Arizona, USA), May 1997. ACM Press.

[BDFS97] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. Adding structure to unstructured data. In F.N. Afrati and Ph. Kolaitis, editors, *International Conference on Database Theory*, pages 336–350, Delphi (Greece), January 1997. Springer-Verlag.

[BDHS96] P. Buneman, S.B. Davidson, G.G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. 1996 ACM SIGMOD International Conference on Management of Data*, pages 505–516, Montreal (Canada), June 1996.

[BDS95] P. Buneman, S.B. Davidson, and D. Suciu. Programming constructs for unstructured data. In Paolo Atzeni and Val Tannen, editors, *6th International Workshop on Database Programming Languages*, Gubbio (Umbria, Italy), September 1995. Springer-Verlag.

[BGL+99] C. Baru, A. Gupta, B. Ludacher, R. Marciano, Y. Papakonst antinou, and P. Valikhov. Xml-base information mediation with mix. In *SIGMOD System Demonstration*, 1999.

[BPSMM00] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, editors. *Extensible Markup Language (XML) 1.0.* `http://www.w3c.org/TR/2000/REC-xml-20001006`, 2000. W3C Recommendation.

[Bun97] P. Buneman. Semistructured data. In *Proc. 16th Symposium on Principles of Database Systems*, pages 117–121, Tucson (Arizona, USA), May 1997. ACM Press.

[BYKK+99] Z. Bar-Yossef, Y. Kanza, Y. Kogan, W. Nutt, and Y. Sagiv. Querying semantically tagged documents on the world-wide web. In *Proc. 4th International Workshop on Next Generation Information Technologies and Systems*, Zichron Yaakov (Israel), July 1999. Springer-Verlag.

[CAW98] S. Chawathe, S. Abiteboul, and J. Widom. Representing and querying changes in semistructured data. In *Proc. 14th International Conference on Data Engineering*, pages 4–13, Orlando (Florida, USA), February 1998. IEEE Computer Society.

[CCF+01] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu, editors. *XQuery 1.0: an XML Query Language.* `http://www.w3c.org/TR/XQuery`, 2001. W3C Working Draft.

[CDSS98] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion! In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, Seattle (Washington, USA), June 1998. ACM Press.

[CKK⁺99] S. Cohen, Y. Kanza, Y. Kogan, W. Nutt, Y. Sagiv, and A. Serebrenik. Equix - easy querying in xml databases. In *ACM Workshop on The Web and Databases (WebDB)*, 1999.

[CKK⁺00] S. Cohen, Y. Kanza, Y. Kogan, W. Nutt, Y. Sagiv, and A. Serebrenik. Combining the power of searching and querying. In *Proc. of 7th International Conference on Cooperative Information Systems (CoopIS 2000)*, Eilat (Israel), September 2000. Springer-Verlag.

[CKS01] S. Cohen, Y. Kanza, and Y. Sagiv. SQL4X: A flexible query language for XML and relational databases. In *Informal Proc. of 8th Workshop on Database Programming Languages (DBPL)*, Frascati, Rome (Italy), September 2001.

[Cod75] E. F. Codd. Understanding relations (installment #7). *FDT - Bulletin of ACM SIGMOD*, 7(3):23–28, 1975.

[Cod79] E. F. Codd. Extending the database relational model to capture more meaning. *TODS*, 4(4):397–434, 1979.

[DeR99] J. Clarkand S. DeRose, editor. *XML Path Language (XPath) Version 1.0.* `http://www.w3c.org/TR/xpath`, 1999. W3C Recommendation.

[DFF⁺99] A. Deutsch, M. Fernandez, D. Florescu, A. Levi, and D. Suciu. A query language for xml. In *Proc. of the World Wide Web Conference*, 1999.

[FFK⁺98] M.F. Fernandez, D. Florescu, J. Kang, A.Y. Levy, and D. Suciu. Catching the boat with Strudel: Experiences with a web-site management system. In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, pages 414–425, Seattle (Washington, USA), June 1998. ACM Press.

[GL94] C.A. Galindo-Legaria. Outerjoins as disjunctions. In *Proc. 1994 ACM SIGMOD International Conference on Management of Data*, pages 348–358, Minneapolis (Minnesota, USA), May 1994. ACM Press.

[GMQP⁺95] H. Garcia-Molina, D. Quass, Y. Papakonstantinou, A. Rajaraman, Y. Sagiv, J.D. Ullman, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. In *Proc. 2nd International Workshop on Next Generation Information Technologies and Systems*, Naharia (Israel), June 1995.

[IL84] T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31:761–791, 1984.

[KNS] Y. Kanza, W. Nutt, and Y. Sagiv. Querying incomplete information in semistructured data. *Journal of Computer and System Sciences.* to appear.

[KNS99] Y. Kanza, W. Nutt, and Y. Sagiv. Queries with incomplete answers over semistructured data. In *pods99*, pages 227–236, Philadelphia (Pennsylvania, USA), June 1999. ACM Press.

[KS95] D. Konopnicki and O. Shmueli. W3QS: A query system for the world-wide web. In *Proc. 21st International Conference on Very Large Data Bases*, pages 54–65. Morgan Kaufmann Publishers, August 1995.

[KS97] D. Konopnicki and O. Shmueli. W3QS—A system for WWW querying. In *Proc. 13th International Conference on Data Engineering*, page 586, Birmingham (United Kingdom), April 1997. IEEE Computer Society.

[KS01] Y. Kanza and Y. Sagiv. Flexible queries over semistructured data. In *Proc. 20th Symposium on Principles of Database Systems*, Santa Barbara (California, USA), May 2001.

[Lip79] W. Lipski. On semantic issues connected with incomplete information in relational databases. *ACM Transactions on Database Systems*, 4:262–296, 1979.

[LSS96] L.V.S. Lakshmanan, F. Sadri, and I.N. Subramanian. A declarative language for querying and restructuring the web. In *Proc. 6th International Workshop on Research Issues on Data Engineering - Interoperability of Nontraditional Database Systems*, pages 12–21, New Orleans (Louisiana, USA), February 1996. IEEE Computer Society.

[MAG⁺97] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 3(26):54–66, 1997.

[MAM⁺98] G. Mecca, P. Atzeni, A. Masci, P. Merialdo, and G. Sindoni. The Araneus web-base management system. In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, pages 544–546, Seattle (Washington, USA), June 1998. ACM Press.

[Mey97] R. Van Der Meyden. The complexity of querying indefinite data about liniarly ordered domains. *Journal of Computer and System Sciences*, 54:113–135, 1997.

[MM97] A.O. Mendelzon and T. Milo. Formal models of web queries. In *Proc. 16th Symposium on Principles of Database Systems*, pages 134–143, Tucson (Arizona, USA), May 1997. ACM Press.

[MMM97] A.O. Mendelzon, G.A. Mihaila, and T. Milo. Querying the world wide web. *International Journal on Digital Libraries*, 1(1):54–67, 1997.

[NS00] F. Neven and T. Schwentick. Expressive and efficient pattern languages for tree-structured data. In *Proc. 19th Symposium on Principles of Database Systems*, pages 145–156, Dallas (Texas, USA), May 2000. ACM Press.

[PGMW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In P.S.Yu and A.L.P. Chen, editors, *Proc. 11th International Conference on Data Engineering*, pages 251–260, Taipei, March 1995. IEEE Computer Society.

[QRS+95] D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. In Tok Wang Ling, Alberto O. Mendelzon, and Laurent Vieille, editors, *Proc. 4th International Conference on Deductive and Object-Oriented Databases*, volume 1013 of *Lecture Notes in Computer Science*, pages 319–344, Singapore, December 1995. Springer-Verlag.

[QWG+96] D. Quass, J. Widom, R. Goldman, K. Haas, Q. Luo, J. McHugh, S. Nestorov, A. Rajaraman, H. Rivero, S. Abiteboul, J.D. Ullman, and J.L. Wiener. Lore: A lightweight object repository for semistructured data. In *Proc. 1996 ACM SIGMOD International Conference on Management of Data*, page 549, Montreal (Canada), June 1996.

[Rei86] R. Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. ACM*, 33(2):349–370, 1986.

[RU96] A. Rajaraman and J.D. Ullman. Integrating information by outerjoins and full disjunctions. In *Proc. 15th Symposium on Principles of Database Systems*, pages 238–248, Montreal (Canada), June 1996. ACM Press.

[Var86] M. Y. Vardi. On the integrity of databases with incomplete information. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*, pages 252–266, Cambridge (Massachusetts, USA), 1986.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[Zan84] C. Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, 28(1):142–166, 1984.