Database Techniques for the World-Wide Web: A Survey

Daniela Florescu
Inria Roquencourt
dana@rodin.inria.fr

Alon Levy
Univ. of Washington
alon@cs.washington.edu

Alberto Mendelzon
Univ. of Toronto
mendel@cs.toronto.edu

1 Introduction

The popularity of the World-Wide Web (WWW) has made it a prime vehicle for disseminating information. The relevance of database concepts to the problems of managing and querying this information has led to a significant body of recent research addressing these problems. Even though the underlying challenge is the one that has been traditionally addressed by the database community – how to manage large volumes of data – the novel context of the WWW forces us to significantly extend previous techniques. The primary goal of this survey is to classify the different tasks to which database concepts have been applied, and to emphasize the technical innovations that were required to do so.

We do not claim that database technology is the magic bullet that will solve all web information management problems; other technologies, such as Information Retrieval, Artificial Intelligence, and Hypertext/Hypermedia, are likely to be just as important. However, surveying all the work going on in these areas, and the interactions between them and database ideas, would be far beyond our scope.

We focus on three classes of tasks related to information management on the WWW.

Modeling and querying the web: Suppose we view the web as a directed graph whose nodes are web pages and whose edges are the links between pages. A first task we consider is that of formulating queries for retrieving certain pages on the web. The queries can be based on the *content* of the desired pages and on the *link structure* connecting the pages. The simplest instance of this task, which is provided by search engines on the web is to locate pages based on the words they contain. A simple generalization of such a query is to apply more complex predicates on the contents of a page (e.g., find the pages that contain the word "Clinton" next to a link to an image). Finally, as an example of a query that involves the structure of the pages, consider the query asking for all images reachable from the root of the CNN web site within 5 links. The last type of queries are especially useful when detecting violations of integrity constraints on a web site or a collection of web sites.

Information extraction and integration: Certain web sites can be viewed at a finer granularity level than pages, as *containers* of structured data (e.g., sets of tuples, or sets of objects). For example, the Internet Movie Database (`http://www.imdb.com`) can be viewed as a front end interface to a database about movies. Given the rise in the number of such sites, there are two tasks we consider. The first task is to actually extract a structured representation of the data (e.g., a set of tuples) from the HTML pages containing them. This task is performed by a set of *wrapper* programs, whose creation and maintenance raises several challenges. Once we view these sites as autonomous heterogeneous databases, we can address the second task of posing queries that require the integration of data. The second task is addressed by *mediator* (or *data integration*) systems.

Web site construction and restructuring: A different aspect in which database concepts and technology can be applied is that of building, restructuring and managing web-sites. In contrast to the previous two classes which apply on *existing* web sites, here we consider the process of *creating* sites. Web sites can be constructed either by starting with some raw data (stored in databases or structured files) or by restructuring existing web sites. Peforming this task requires methods for *modeling* the structure of web site and languages for *restructing* data to conform to a desired structure.

Before we begin, we note that there are several topics concerning the application of database concepts to the WWW which are not covered in this survey, such as caching and replication (see [WWW98, GRC97] for recent works), transaction processing and security in web environments (see e.g. [Bil98]), performance, availability and scalability issues for web servers (e.g. [CS98]), or indexing techniques and crawler technology (e.g. [CGMP98]). (Furthermore, this is not meant to be a survey on existing products even in the areas on which we do focus. Finally, there are several tangential areas whose results are applicable to the systems we discuss, but we do not cover them here. Examples of such fields include systems for managing document collections and ranking of documents (e.g., Harvest [BDH+95], Gloss [GGMT99]) and flexible query answering systems [BT98]. Finally, the field of web/db is a very dynamic one; hence, there are undoubtedly some omissions in our coverage, for which we apologize in advance.

The survey is organized as follows. We begin in Section 2 by discussing the main issues that arise in designing data models for web/db applications. The following three sections consider each of the aforementioned tasks. Section 6 concludes with perspectives and directions for future research.

2 Data Representation for Web/DB Tasks

Building systems for solving any of the previous tasks requires that we choose a method for modeling the underlying domain. In particular, in these tasks, we need to model the web itself, structure of web sites, internal structure of web pages, and finally, contents of web sites in finer granularities. In this section we discuss the main distinguishing factors of the data models used in web applications.

Graph data models: As noted above, several of the applications we discuss require to model the set of web pages and the links between them. These pages can either be on several sites or within a single site. Hence, a natural way to model this data is based on a *labeled graph* data model. Specifically, in this model, nodes represent web pages (or internal components of web pages), and arcs represent links between pages. The labels on the arcs can be viewed as attribute names. Along with the labeled graph model, several query languages have been developed. One central feature that is common to these query languages is the ability to formulate *regular path expression* queries over the graph. Regular path expressions enable posing navigational queries over the graph structure.

Semistructured data models: The second aspect of modeling data for web applications is that in many cases the structure of the data is irregular. Specifically, when modeling the structure of a web site, we don't have a fixed schema which is given in advance. When modeling data coming from multiple sources, the representation of some attributes (e.g., addresses) may differ from source to source. Hence, several projects have considered models of *semistructured data*. The initial motivation for this work was the existence and relative success of permissive data models such as [TMD92] in the scientific community, the need for exchanging objects across heterogeneous sources [PGMW95], and the task of managing document collections [MP96].

Broadly speaking, semistructured data refers to data with some of the following characteristics:

- the schema is not given in advance and may be implicit in the data,
- the schema is relatively large (w.r.t. the size of the data) and may be changing frequently,
- the schema is *descriptive* rather than *prescriptive*, i.e., it describes the current state of the data, but violations of the schema are still tolerated,
- the data is not strongly typed, i.e., for different objects, the values of the same attribute may be of differing types.

Models for semistructured data have been based on labeled directed graphs [Abi97, Bun97].[1] In a semistructured data model, there is no restriction on the set of arcs that emanate from a given node in a graph, or on the types of the values of attributes. Because of the characteristics of semistructured data mentioned above, an additional feature that becomes important in this context is the ability to query the schema (i.e., the labels on the arcs in the graph). This feature is supported in languages for querying semistructured data by *arc variables* which get bound to labels on arcs, rather than nodes in the graph.

In addition to developing models and query languages for semistructured data, there has been considerable recent work on issues concerning the management of semistructured data, such as the extraction of structure from semistructured data [NAM98], view maintenance [ZGM98, AMR+98], summarization of semistructured data ([BDFS97, GW97]), and reasoning about semistructured data [CGL98, FFLS98]. Aside from the relevance of these works to the tasks mentioned in this survey, the systems based on these methods will be of special importance for the task of managing large volumes of XML data [XML98].

Other characteristics of web data models: Another distinguishing characteristic of models used in web/db applications is the presence of web-specific constructs in the data representation. For example, some models distinguish a unary relation identifying pages and a binary relation for links between pages. Furthermore, we may distinguish between links within a web site and external links. An important reason to distinguish a link relation is that it can generally only be traversed in the forward direction. Additional second order dimensions along which the data models we discuss differ are (1) the ability to model order among elements in the database, (2) modeling nested data structures, and (3) support for collection types (sets, bags, arrays). An example of a data model that incorporates explicit web-specific constructs (pages and page schemes), nesting, and collection types is ADM, the data model of the ARANEUS project [AMM97b]. We remark that all the models we mention in this paper represent only static structures. For example, the work on modeling the structure of web sites do not consider dynamic web pages created as a result of user inputs.

An important aspect of languages for querying data in web applications is the need to create complex structures as a result of a query. For example, the result of a query in a web site management system is the graph modeling the web site. Hence, a fundamental characteristic of many of the languages we discuss in this paper is that their query expressions contain a *structuring* component in addition to the traditional data filtering component.

Table 1 summarizes some of the web query systems covered in this paper. A more detailed version of this table, `http://www.cs.washington.edu/homes/alon/webdb.html` includes URLs for the systems where available. In subsequent sections we will illustrate in detail languages for querying data represented in these models.

3 Modeling and Querying the Web

If the web is viewed as a large, graph-like database, it is natural to pose queries that go beyond the basic information retrieval paradigm supported by today's search engines and take structure into account; both the internal structure of web pages and the external structure of links that interconnect them. In an often-cited paper on the limitations of hypertext systems, Halasz says: [Hal88]

[1] It should be noted that there is no inherent difficulty in translating these models into relational or object-oriented terms. In fact, the languages underlying Description Logics (e.g., Classic [BBMR89]) and FLORID [HLLS97] have some of the features mentioned above, and are described in non-graph models.

System	Data Model	Language Style	Path Expressions	Graph Creation
WebSQL [MMM97]	relational	SQL	Yes	No
W3QS [KS95]	labeled multigraphs	SQL	Yes	No
WebLog [LSS96]	relational	Datalog	No	No
Lorel [AQM+97]	labeled graphs	OQL	Yes	No
WebOQL [AM98]	hypertrees	OQL	Yes	Yes
UnQL [BDHS96]	labeled graphs	structural recursion	Yes	Yes
Strudel [FFK+98, FFLS97]	labeled graphs	Datalog	Yes	Yes
Araneus (Ulixes) [AMM97b]	page schemes	SQL	Yes	Yes
Florid [HLLS97]	F-logic	Datalog	Yes	No

Table 1: Comparison of query systems

> Content search ignores the structure of a hypermedia network. In contrast, structure search specifically examines the hypermedia structure for subnetworks that match a given pattern.

and goes on to give examples where such queries are useful.

3.1 Structural Information Retrieval

The first tools developed for querying the web were the well-known search engines which are now widely deployed and used. These are based on searching indices of words and phrases appearing in documents discovered by web "crawlers." More recently, there have been efforts to overcome the limitations of this paradigm by exploiting link structure in queries. For example, [Kle98], [BH98] and [CDRR98], propose to use the web structure to analyze the many sites returned by a search engine as relevant to a topic in order to extract those that are likely to be authoritative sources on the topic. To support connectivity analysis for this and other applications, (such as efficient implementations of the query languages described below) the Connectivity Server [BBH+98] provides fast access to structural information. Google [BP98], a prototype next-generation web search engine, makes heavy use of web structure to improve crawling and indexing performance. Other methods for exploiting link structure are presented in [PPR96, CK98]. In these works, structural information is mostly used behind the scenes, to improve the answers to purely content-oriented queries. Spertus [Spe97] points out many useful applications of queries that take link structure into account explicitly.

3.2 Related query paradigms

In this section we briefly describe several families of query languages that were not developed specifically for querying the web. However, since the concepts on which they are based are similar in spirit to the web query languages we discuss, these languages can also be useful for web applications.

Hypertext/document query languages: A number of models and languages for querying structured documents and hypertexts were proposed in the pre-web era. For example, Abiteboul et al.[ACM93] and Christophides et al.[CACS94] map documents to object oriented database instances by means of semantic actions attached to a grammar. Then the database representation can be queried using the query language of the database. A novel aspect of this approach is the possibility of querying the structure by means of path variables. Guting et al.[GZC89] model documents using nested ordered relations and use a generalization of nested relational algebra as a query language. Beeri and Kornatzky [BK90] propose a logic whose formulas specify patterns over the hypertext graph.

Graph query languages: Work in using graphs to model databases, motivated by applications such as software engineering and computer network management, led to the G, G+ and GraphLog graph-based languages [CMW87, CMW88, CM90]. In particular, G and G+ are based on labeled graphs; they support regular path expressions and graph construction in queries. GraphLog, whose semantics is based on Datalog, was applied to Hypertext queries in [CM89]. Paredaens et al [PdBA+92] developed a graph query language for object-oriented databases.

Languages for querying semistructured data: Query languages for semistructured data such as Lorel [AQM+97], UnQL [BDHS96] and StruQL [FFLS97] also use labeled graphs as a flexible data model. In contrast to graph query languages, they emphasize the ability to query the schema of the data, and the ability to accommodate irregularities in the data, such as missing or repeated fields, heterogeneous records. Related work in the OO community [Har94] proposes "schema-shy" models and queries to handle information about software engineering artifacts.

These languages were not developed specifically for the web, and do not distinguish, for example, between graph edges that represent the connection between a document and one of its parts and edges that represent a hyperlink from one web document to another. Their data models, while elegant, are not very rich, lacking such basic comforts as records and ordered collections.

3.3 First generation web query languages

A first generation of web query languages aimed to combine the content-based queries of search engines with structure-based queries similar to what one would find in a database system. These languages, which include W3QL [KS95], WebSQL [MMM97, AMM97a], and WebLog [LSS96], combine conditions on text patterns appearing within documents with graph patterns describing link structure. We use WebSQL as an example of the kinds of queries that can be asked.

WebSQL WebSQL proposes to model the web as a relational database composed of two (virtual) relations: Document and Anchor. The Document relation has one tuple for each document in the web and the Anchor relation has one tuple for each anchor in each document in the web. This relational abstraction of the web allows us to use a query language similar to SQL to pose the queries.

If Document and Anchor were actual relations, we could simply use SQL to write queries on them. But since the Document and Anchor relations are completely virtual and there is no way to enumerate them, we cannot operate on them directly. The WebSQL semantics depends instead on materializing portions of them by specifying the documents of interest in the FROM clause of a query. The basic way of materializing a portion of the web is by navigating from known URL's. Path regular expressions are used to describe this navigation. An atom of such a regular expression can be of the form `d1 => d2`, meaning document d1 points to d2 and d2 is stored on a different server from d1; or `d1 -> d2`, meaning d1 points to d2 and d2 is stored on the same server as d1.

For example, suppose we want to find a list of triples of the form `(d1,d2,label)`, where `d1` is a document stored on our local site, `d2` is a document stored somewhere else, and `d1` points to `d2` by a link labeled `label`. Assume all our local documents are reachable from `www.mysite.start`.

```
SELECT d.url,e.url,a.label
FROM Document d SUCH THAT
         "www.mysite.start" ->* d,
     Document e SUCH THAT d => e,
     Anchor   a SUCH THAT a.base = d.url
WHERE a.href = e.url
```

The FROM clause instantiates two Document variables, d and e, and one Anchor variable a. The variable d is bound in turn to each local document reachable from the starting document, and e is bound to each non-local document reachable directly from d. The anchor variable a is instantiated to each link that originates in document d; the extra condition that the target of link `a` be document `e` is given in the WHERE clause. Another way of materializing part of the Document and Anchor relations is by content conditions: for example, if we were only interested in documents that contains the string "database" we could have added to the FROM clause the condition `d MENTIONS "database"`. The implementation uses search engines to generate candidate documents that satisfy the `MENTION` conditions.

Other Languages W3QL [KS95] is similar in flavour to WebSQL, with some notable differences: it uses external programs (similar to user defined functions in object-relational languages) for specifying content conditions on files rather than building conditions into the language syntax, and it provides mechanisms for handling forms encountered during navigation. In [KS98], Konopnicki and Shmueli describe planned extensions to move W3QL into what we call the second generation. These include modeling internal document structure, hierarchical web modeling that captures the notion of web site explicitly, and replacing the external program method of specifying conditions with a general extensible method based on the MIME standard.

WebLog [LSS96] differs from the above languages in using deductive rules instead of SQL-like syntax (see the description of Florid below).

WQL, the query language of the WebDB project [LSCH98], is similar to WebSQL but it supports more comprehensive SQL functionality such as aggregation and grouping, and provides limited support for querying intra-document structure, placing it closer to the class of languages discussed in the next subsection.

3.4 Second generation: Web Data Manipulation Languages

The languages above treat web pages as atomic objects with two properties: they contain or do not contain certain text patterns, and they point to other objects. Experience with their use suggests there are two main areas of application that they can be useful for: data wrapping, transformation, and restructuring, as described in Section 4; and web site construction and restructuring, as described in Section 5. In both application areas, it is often essential to have access to the internal structure of web pages from the query language, if we want declarative queries to capture a large part of the task at hand. For example, the task of extracting a set of tuples from the HTML pages of the Internet Movie Database requires parsing the HTML and selectively accessing certain subtrees in the parse tree.

In this section we describe the second-generation of web query languages that we call "Web data manipulation languages." These languages go beyond the first generation languages in two significant ways. First, they provide access to the structure of the web objects that they manipulate. Unlike the first-generation languages, they model internal structure of web documents as well as the external links that connect them. They support references to model hyperlinks, and some support ordered collections and records for more natural data representation. Second, these languages provide the ability to create new complex structures as a result of a query. Since the data on the web is commonly semistructured (or worse), these languages still emphasize the ability to support semistructured features. We briefly describe three languages in this class: WebOQL [AM98], StruQL[FFLS97] and Florid [HLLS97].

WebOQL

The main data structure provided by WebOQL is the hypertree. Hypertrees are ordered arc-labeled trees with two types of arcs, internal and external. Internal arcs are used to represent structured objects and external arcs are used to represent references (typically hyperlinks) among objects. Arcs are labeled with records. Figure 1, from [Aro97], shows a hypertree containing descriptions of publications from several research groups. Such a tree could easily be built, for example, from an HTML file, using a generic HTML wrapper.

Sets of related hypertrees are collected into *webs*. Both hypertrees and webs can be manipulated using WebOQL and created as the result of a query.

WebOQL is a functional language, but queries are couched in the familiar select-from-where form. For example, suppose that the name csPapers denotes the papers database in Figure 1, and that we want to extract from it the title and URL of the full version of papers authored by "Smith".

```
select [y.Title,y'.Url]
from x in csPapers, y in x'
where y.Authors ~ "Smith"
```

In this query, x iterates over the simple trees of csPapers (i.e., over the research groups) and, given a value for x, y

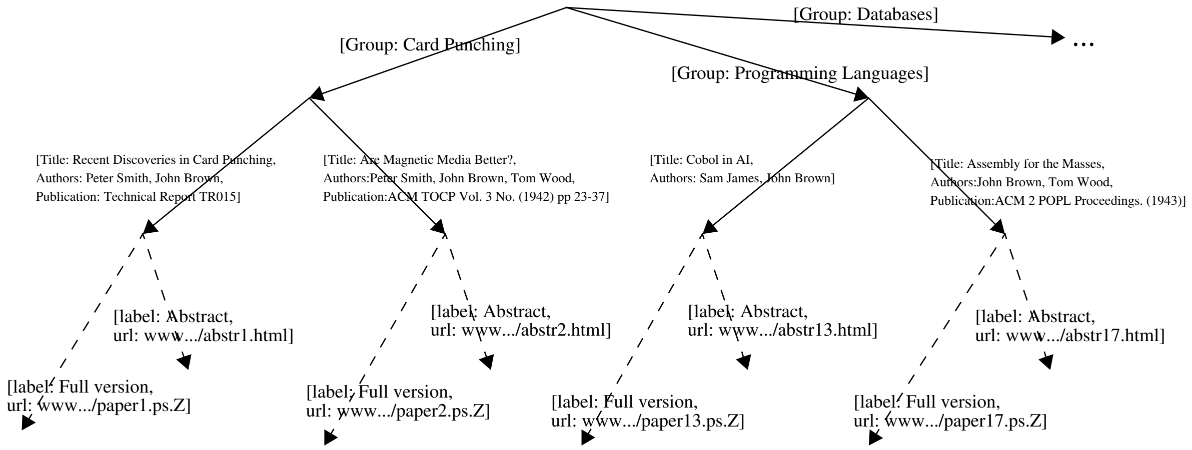

Figure 1: Example of a hypertree

iterates over the simple trees of x'. The primed variable `x'` denotes the result of applying to tree `x` the Prime operator, which returns the first subtree of its argument. The same operator is used to extract from tree `y` its first subtree in `y'.Url`. The square brackets denote the Hang operator, which builds an arc labeled with a record formed with the arguments (in this example, the field names are inferred.) Finally, the tilde represents the string pattern matching predicate: its left argument is a string and its right argument is a pattern.

Web Creation The query above maps a hypertree into another hypertree; more generally, a query is a function that maps a web into another. For example, the following query creates a new page for each research group (using the group name as URL). Each page contains the publications of the corresponding group.

```
select x' as x.Group
from x in csPapers
```

In general, the select clause has the form ' select q_1 as s_1, q_2 as s_2, ... , q_m as s_m', where the q_i's are queries and each of the s_i's is either a string query or the keyword schema. The "as" clauses create the URL's s_1, s_2, ... , s_m, which are assigned to the new pages resulting from each query q_i.

Navigation Patterns Navigation patterns are regular expressions over an alphabet of record predicates; they allow us to specify the structure of the paths that must be followed in order to find the instances for variables.

Navigation patterns are mainly useful for two purposes. The first reason is for extracting subtrees from trees whose structure we do not know in detail or whose structure presents irregularities, and the second is for iterating over trees connected by external arcs. In fact, the distinction between internal and external arcs in hypertrees becomes really useful when we use navigation patterns that traverse external arcs. Suppose that we have a software product whose documentation is provided in HTML format and we want to build a full-text index for it. These documents form a complex hypertext, but it is possible to browse them sequentially by following links having the string "Next" as label. For building the full-text index we need to feed the indexer with the text and the URL of each document. We can obtain this information using the following query:

```
select [ x.Url, x.Text ]
from x in browse("root.html")
      via (^*[Text ~ "Next"]>)*
```

StruQL

StruQL is the query language of the Strudel web site management system, described below in Section 5. Even though StruQL was developed in the context of a specific web application, it is a general purpose query language for semistructured data, based on a data model of labeled directed graphs. In addition, the Strudel data model includes named collections, and supports several atomic types that commonly appear in web pages, such as URLs, and Postscript, text, image, and HTML files. The result of a StruQL query is a graph in the same data model as the input graphs. In Strudel, StruQL was used for two tasks: querying heterogeneous sources to integrate them into a site *data graph*, and for querying this data graph to produce a *site graph*.

A StruQL query is a set of possibly nested blocks, each of the form:

```
[where C1,...,Ck]
[create N1,...,Nn]
[link L1,...,Lp]
[collect G1,...,Gq].
```

The `where` clause can include either membership conditions or conditions on pairs of nodes expressed using regular path expressions. The `where` clause produces all bindings of node and arc variables to values in the input graph, and the remaining clauses use Skolem functions to construct a new graph from these bindings.

We illustrate StruQL with a query defining a web site, starting with a Bibtex bibliography file, modeled as a labeled graph. The web site will consist of three kinds of pages: a PaperPresentation page for each bibliography entry, a Year page for each year, pointing to all papers published in that year, and a Root page pointing to all the Year pages. After showing the query in StruQL, we show it in WebOQL to give a feel for the differences between the languages.

```
// Create Root
create RootPage()
// Create a presentation for every publication x
where  Publications(x), x->l->v
create PaperPresentation(x)
link PaperPresentation(x) -> l -> v
 { // Create a page  for every year
  where  l = "year"
  create YearPage(v)
  link
   YearPage(v) -> "Year" -> v
   YearPage(v)->"Paper"->PaperPresentation(x),
   // Link root page to each year page
   RootPage() -> "YearPage" -> YearPage(v)
 }
```

In the `where` clause, the notation `Publications(x)` means that `x` belongs to the collection Publications, and the atom `x` → `l` → `v` denotes that there is a link in the graph from `x` to `v` and the label on the arc is `l`. The same notation is used in the `link` clause to specify the newly created edges in the resulting graph. After creating the Root page, the first CREATE generates a page for each publication (denoted by the Skolem function, PaperPresentation). The second CREATE, nested within the outer query, generates a Year page for each year, and links it to the Root page and to the PaperPresentation pages of the publications published in that year. Note the Skolem function YearPage ensures that a Year page for a particular year is only created once, no matter how many papers were published in that year.

Below is the same query in WebOQL.

```
select unique [Url: x.year, Label:"YearPage"]
                                as "RootPage",
              [ label: "Paper" / x ] as x.year
from x in browse("bibtex: myfile.bib")

|

select [year: y.url] + y as y.url
from y in "browse(RootPage)"
```

The WebOQL query consists of two subqueries, with the web resulting from the first one "piped" into the second one using the "|" operator. The first subquery builds the Root, Paper, and Year pages, and the second one redefines each Year page by adding the "year" field to it.

Florid

Florid [HLLS97, LHL+98] is a prototype implementation of the deductive and object-oriented formalism F-logic [KLW95]. To use Florid as a web query engine, a web document is modeled by the following two classes:

```
url::string[get => webdoc].

webdoc::string[url => url; author => string;
                            modif => string;
       type => string; hrefs@(string) =>> url;
                       error =>> string].
```

The first declaration introduces a class `url`, subclass of `string`, with the only method `get`. The notation `get => webdoc` means that `get` is a single-valued method that returns an object of type `webdoc`. The method `get` is system-defined; the effect of invoking `u.get` for a url `u` in the head of a deductive rule is to retrieve from the web the document with that URL and cache it in the local Florid database as a `webdoc` object with object identifier `u.get`.

The class `webdoc` with methods `self, author, modif, type, hrefs` and `error` models the basic information common to all web documents. The notation `hrefs@(string) =>> url` means that the multi-valued method `hrefs` takes a `string` as argument and returns a set of objects of type `url`. The idea is that, if `d` is a `webdoc`, then `d.hrefs@(aLabel)` returns all URL's of documents pointed to by links labeled `aLabel` within document `d`.

Subclasses of documents can be declared as needed using F-logic inheritance, e.g.:

```
htmldoc::webdoc[title => string; text => string].
```

Computation in Florid is expressed by sets of deductive rules. For example, the program below extracts from the web the set of all documents reachable directly or indirectly from the URL `www.cs.toronto.edu` by links whose labels contain the string "database."

```
("www.cs.toronto.edu":url).get.
(Y:url).get   <-
            (X:url).get[hrefs@(L)=>>{Y}],
            substr("database",L).
```

Florid provides a powerful formalism for manipulating semi-structured data in a web context. However, it does not currently support the construction of new webs as results of computation; the result is always a set of F-logic objects in the local store.

Ulixes and Penelope

In the Araneus project [AMM97b], the query and restructuring process is split into two phases. In the first phase,the Ulixes language is used to build relational views over the web. These views can then be analyzed and integrated using standard database techniques. Ulixes queries extract relational data from instances of page schemes defined in the ADM model, making heavy use of (star-free) path expressions. The second phase consists of generating hypertextual views of the data using the Penelope language. Query optimization for relational views over sets of web pages, such as those constructed by Ulixes, is discussed in [MMM98].

Interactive query interfaces

All the languages in the previous two subsections are too complex to be used directly by interactive users, just as SQL is; like SQL, they are meant to be used mostly as programming tools. There has however been work in the design of interactive query interfaces suitable for casual users. For example, Dataguides [GW97] is an interactive query tool for semistructured databases based on hierarchical summaries of the data graph; extensions to support querying single complex web sites are described in [GW98]. The system described in [HML+98] supports queries that combine multimedia features, such as similarity to a given sketch or image, textual features such as keywords, and domain semantics.

Theory of web queries

In defining the semantics of first-generation web query languages, it was immediately observed that certain easily stated queries, such as "list all web documents that no other document points to," could be rather hard to execute. This leads naturally to questions of query computability in this context. Abiteboul and Vianu [AV97a] and Mendelzon and Milo [MM97] propose formal ways of categorizing web queries according to whether they can in principle be computed or not; the key idea being that essentially, the only possible way to access the web is to navigate links from known starting points. (Note this includes a special case navigating links from the large collections of starting points known as index servers or search engines.) Abiteboul and Vianu [AV97b] also discuss fundamental issues posed by query optimization in path traversal queries. Mihaila, Milo and Mendelzon [MMM97] show how to analyze WebSQL queries in terms of the maximum number of web sites. Florescu, Levy and Suciu [FLS98] describe an algorithm for query containment for queries with regular path expressions, which is then used for verifying integrity constraints on the structure of web sites [FFLS98].

4 Information Integration

As stated earlier, the WWW contains a growing number of information sources that can be viewed as *containers* of sets of tuples. These "tuples" can either be embedded in HTML pages, or be hidden behind form interfaces. By writing specialized programs called *wrappers*, one can give the illusion that the web site is serving sets of tuples. We refer to the combination of the underlying web site and the wrapper associated with it as a *web source*.

The task of a web information integration system is to answer queries that may require extracting and combining data from multiple web sources. As an example, consider the domain of movies. The Internet Movie Database contains comprehensive data about movies, their casts, genres and directors. Reviews of movies can be found in multiple other web sources (e.g., web sites of major newspapers), and several web sources provide schedules of movie showings. By combining the data from these sources we can answer queries such as: *give me a movie, playing time and a review of movies starring Frank Sinatra, playing tonight in Paris.*

Several systems have been built with the goal of answering queries using a multitude of web sources [GMPQ+97, EW94, WBJ+95, LRO96, FW97, DG97b, AKS96, Coh98, AAB+98, BEM+98]. Many of the problems encountered in building these systems are similar to those addressed in building heterogeneous database systems [ACPS96, WAC+93, HZ96, TRV98, FRV96, Bla96, HKWY97]. Web data integration systems have, in addition, to deal with (1) large and evolving number of web sources, (2) little meta-data about the characteristics of the source, and (3) larger degree of source autonomy.

An important distinction in building data integration systems, and therefore in building web data integration systems, is whether to take a warehousing or a virtual approach (see [HZ96, Hul97] for a comparison). In the warehousing approach, data from multiple web sources is loaded into a warehouse, and all queries are applied to the warehoused data; this requires that the warehouse be updated when data changes, but the advantage is that adequate performance can be guaranteed at query time. In the virtual approach, the data remains in the web sources, and queries to the data integration system are decomposed at run time into queries on the sources. In this approach, data is not replicated, and is guaranteed to be fresh at query time. On the other hand, because the web sources are autonomous, more sophisticated query optimization and execution methods are needed to guarantee adequate performance. The virtual approach is more appropriate for building systems where the number of sources is large, the data is changing frequently, and there is little control over the web sources. For these reasons, most of the recent research has focused on the virtual approach, and therefore, so will our discussion. We emphasize that many of the issues that arise in the virtual approach also arise in the warehousing approach (often in a slightly different form), and hence our discussion is relevant to both cases. Finally, we refer the reader to two commercial applications of web data integration, one that takes the warehousing approach [Jun98] and the other that takes the virtual approach [Jan98].

A prototypical architecture of a virtual data integration system is shown in Figure 3. There are two main features distinguishing such a system from a traditional database system:

- As stated earlier, the system does not communicate directly with a local storage manager. Instead, in order to obtain data, the query execution engine communicates with a set of wrappers. A wrapper is a program which is specific to every web site, and whose task is to translate the data in the web site to a form that can be further processed by the data integration system. For example, the wrapper may extract from an HTML file a set of tuples. It should be emphasized that the wrapper provides only an interface to the data served by the web site, and hence, if the web site provides only limited access to the data (e.g., through a form interface that requires certain inputs), then the wrapper can only support limited access patterns to the data.
- The second difference from traditional systems is that the user does not pose queries directly in the schema in which the data is stored. The reason for this is that one of the principal goals of a data integration system is to free the user from having to know about the specific data sources and interact with each one. Instead, the user poses queries on a *mediated schema*. A mediated schema is a set of *virtual* relations, which are designed for a particular data integration application. The relations in the mediated schema are not actually stored anywhere. As a consequence, the data integration system must first *reformulate* a user query into a query that refers directly to the schemas in the sources. In order to perform the reformulation step, the data integration system requires a set of *source descriptions*. A description of an information source specifies the contents of the source (e.g., contains movies), the at-

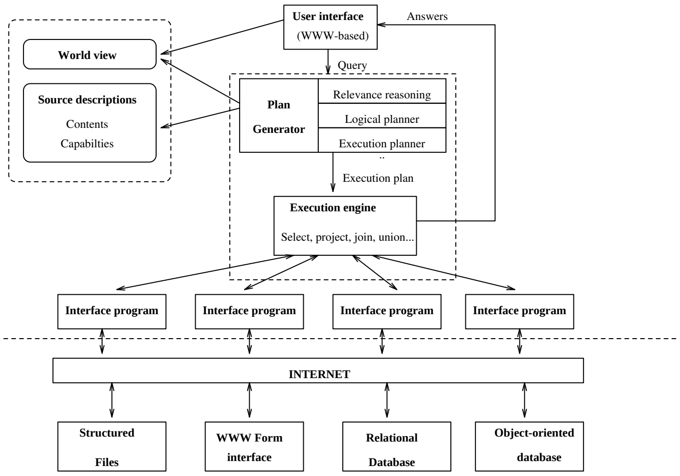

Figure 2: Architecture of a data integration system

tributes that can be found in the source (e.g., genre, cast), constraints on the contents of the source (e.g., contains only American movies), completeness and reliability of the source, and finally, the query processing capabilities of the source (e.g., can perform selections, or can answer arbitrary SQL queries).

The following are the main issues addressed in the work on building web data integration systems.

Specification of mediated schema and reformulation: The mediated schema in a data integration system is the set of collection and attribute names that are used to formulate queries. To evaluate a query, the data integration system must translate the query on the mediated schema into a query on the data sources, that have their own local schemas. In order to do so, the system requires a set of source descriptions. Several recent research works addressed the problem of how to specify source descriptions and how to use them for query reformulation. Broadly speaking, two general approaches have been proposed: *Global as view* (GAV) [GMPQ$^+$97, PAGM96, ACPS96, HKWY97, FRV96, TRV98] and *Local as view* (LAV) [LRO96, KW96, DG97a, DG97b, FW97] (see [Ull97] for a detailed comparison).

In the GAV approach, for each relation R in the mediated schema, we write a query over the source relations specifying how to obtain R's tuples from the sources. The LAV approach takes the opposite approach. For every information source S, we write a query over the relations in the mediated schema that describes which tuples are found in S. The main advantage of the GAV approach is that query reformulation is very simple, because it reduces to view unfolding. In contrast, in the LAV approach it is simpler to add or delete sources because the descriptions of the sources do not have to take into account the possible interactions with other sources, as in the GAV approach, and it is also easier to describe constraints on the contents of sources. The problem of reformulation becomes a variant on the of the problem of answering queries using views [YL87, TSI96, LMSS95, CKPS95, RSU95, DG97b].

Completeness of data in web sources: In general, sources that we find on the WWW are not necessarily complete for the domain they are covering. For example, a bibliography source is unlikely to be complete for the field of Computer Science. However, in some cases, we can assert completeness statements about sources. For example, the DB&LP Database[2] has the complete set of papers published in most major database conferences. Knowledge of completeness of a web source can help a data integration system in several ways. Most importantly, since a *negative* answer from a complete source is meaningful, the data integration system can prune access to other sources. The problem of describing completeness of web sources and using this information for query processing is addressed in [Mot89, EGW94, Lev96, Dus97, AD98, FW97]. The work described in [FKL97] describes a probabilistic formalism for describing the contents and overlaps among information sources, and presents algorithms for choosing optimally between sources.

Differing query processing capabilities: From the perspective of the web data integration system, the web sources appear to have vastly differing query processing capabilities. The main reasons for the different appearance are (1) the underlying data may actually be stored in a structured file or legacy system and in this case the interface to this data is naturally limited, and (2) even if the data is stored in a traditional database system, the web site may provide only limited access capabilities for reasons of security or performance.

To build an effective data integration system, these capabilities need to be explicitly described to the system, adhered to, and exploited as much as possible to improve performance. We distinguish two types of capabilities: negative capabilities that limit the access patterns to the data, and positive capabilities, where a source is able to perform additional algebraic operations in addition to simple data fetches.

The main form of negative capabilities is limitations on the binding patterns that can be used in queries sent to the source. For example, it is not possible to send a query to the Internet Movie Database asking for *all* the movies in the database and their casts. Instead, it is only possible to ask for the cast of *given* movie, or to ask for the set of movies in which a particular actor appears. Several works have con-

[2] `http://www.informatik.uni-trier.de/ ley/db/`

sidered the problem of answering queries in the presence of binding pattern limitations [RSU95, KW96, LRO96, FW97].

Positive capabilities pose another challenge to a data integration system. If a data source has the ability to perform operations such as selections and joins, we would like to push as much as possible of the processing to the source, thereby hopefully reducing the amount of local processing and the amount of data transmitted over the network. The problem of describing the computing capabilities of data sources and exploiting them to create query execution plans is considered in [PGGMU95, TRV98, LRU96, HKWY97, VP97a]

Query optimization: Many works on web data integration systems have focused on the problem of selecting a *minimal* set of web sources to access, and on determining the minimal query that needs to be sent to each one. However, the issue of choosing an optimal query execution plan to access the web sources has received relatively little attention in the data integration literature [HKWY97], and remains an active area of research. The additional challenges that are faced in query optimization over sources on the WWW is that we have few statistics on the data in the sources, and hence little information to evaluate the cost of query execution plans. The work in [NGT98] considers the problem of calibrating the cost model for query execution plans in this context. The work in [YPAGM98] discusses the problem of query optimization for *fusion queries*, which are a special class of integration queries that focus on retrieving various attributes of a given object from multiple sources. Moreover, we believe that query processing in data integration systems is one area which would benefit from ideas such as interleaving of planning and execution and of computing conditional plans [GC94, KD98].

Query execution engines: Even less attention has been paid to the problem of building query execution engines targeted for web data integration. The challenges in building such engines are caused by the autonomy of the data sources and the unpredictability of the performance of the network. In particular, when accessing web sources we may experience initial delays before data is transmitted, and even when it is, the arrival of the data may be bursty. The work described in [AFT98, UFA98] has considered the problem of adapting a query execution plans to initial delays in the arrival of the data.

Wrapper construction: Recall that the role of a wrapper is to extract the data out of a web site into a form that can be manipulated by the data integration system. For example, the task of a wrapper could be to pose a query to a web source using a form interface, and to extract a set of answer tuples out of the resulting HTML page. The difficulty in building wrappers is that the HTML page is usually designed for human viewing, rather than for extracting data by programs. Hence, the data is often embedded in natural language text or hidden within graphical presentation primitives. Moreover, the form of the HTML pages changes frequently, making it hard to maintain the wrappers. Several works have considered the problem of building tools for rapid creation of wrappers. One class of tools (e.g., [HGMN+98, GRVB98]) is based on developing specialized grammars for specifying how the data is laid out in an HTML page, and therefore how to extract the required data. A second class of techniques is based on developing inductive learning techniques for automatically learning a wrapper. Using these algorithms, we provide the system with a set of HTML pages where the data in the page is *labeled*. The algorithm uses the labeled examples to automatically output a grammar by which the data can be extracted from subsequent pages. Naturally, the more examples we give the system, the more accurate the resulting grammar can be, and the challenge is to discover wrapper languages that can be learned with a small number of examples. The first formulation of wrapper construction as inductive learning and a set of algorithms for learning simple classes of wrappers are given in [KDW97]. The algorithm described in [AK97] exploits heuristics specific to the common uses of HTML in order to obtain faster learning. It should be noted that Machine Learning methods have also been used to learn the mapping between the source schemas and the mediated schemas [PE95, DEW97]. The work described [CDF+98] is a first step in bridging the gap between the approaches of Machine Learning and of Natural Language Processing to the problem of wrapper construction. Finally, we note that the emergence of XML may lead web site builders to export the data underlying their sites in a machine readable form, thereby greatly simplifying the construction of wrappers.

Matching objects across sources: One of the hardest problems in answering queries over a multitude of sources is deciding that two objects mentioned in two different sources refer to the same entity in the world. This problem arises because each source employs its own naming conventions and shorthands. Most systems deal with this problem using domain specific heuristics (as in [FHM94]). In the WHIRL system [Coh98], matching of objects across sources is done by using techniques from Information Retrieval. Furthermore, the matching of the objects is elegantly integrated in a novel query execution algorithm.

5 Web site construction and restructuring

The previous two sections discussed tasks that concerned querying *existing* web sites and their content. However, given the fact that web sites essentially provide access to complex structures of information, it is natural to apply techniques from Database Systems to the process of *building* and *maintaining* web sites. One can distinguish two general classes of web site building tasks: one in which web sites are created from a collection of underlying data sources, and another in which they are created by restructuring existing web sites. As it turns out, the same techniques are required for both of these classes. Furthermore, we note that the task of providing a web interface to data that exists in a single database system [NS96] is a simple instance of the problem of creating web sites.[3]

To understand the problem of building web sites and the possible import of database technology, consider the tasks that a web site builder must address: (1) choosing and accessing the data that will be displayed at the site, (2) designing the site's structure, i.e., specifying the data contained within each page and the links between pages, and (3) designing the graphical presentation of pages. In existing web site management tools, these tasks are, for the most part, interdependent. Without any site-creation tools, a site builder writes HTML files by hand or writes programs to produce them and must focus simultaneously on a page's content, its relationship to other pages, and its graphical presentation. As a result, several important tasks, such as automatically

[3] Most database vendors were quick to provide commercial tools for performing this task.

updating a site, restructuring a site, or enforcing integrity constraints on a site's structure, are tedious to perform.

Web sites as declaratively defined structures: Several systems have been developed with the goal of applying database techniques to the problem of web site creation [FFK+98, AMM98, AM98, CDSS98, PF98, JB97, LSB+98, TN98]. The common theme to these systems is that they provide an explicit declarative representation of the structure of a web site. The structure of the web site is defined as a *view* over existing data. However, we emphasize that the languages used to create these views result in *graphs* of web pages with hypertext links, rather than simple tables. The systems differ on the data model they use, the query language they use, and whether they have an intermediate logical representation of the web site, rather than having only a representation of the final HTML.

Building a web site using a declarative representation of the structure of the site has several significant advantages. Since a web site's structure and content are defined declaratively by a query, not procedurally by a program, it is easy to create multiple *versions* of a site. For example, it is possible to easily build internal and external views of an organization's site or to build sites tailored to different classes of users. Currently, creating multiple versions requires writing multiple sets of programs or manually creating different sets of HTML files. Building multiple versions of a site can be done by either writing different site definition queries, or by changing the graphical representation independently of the underlying structure. Furthermore, a declarative representation of the web site's structure also supports easy evolution of a web site's structure. For example, to reorganize pages based on frequent usage patterns [PE97], or to extend the site's content, simply rewrite the site-definition query, as opposed to rewriting a set of programs or a set of HTML files. Declarative specification of web sites can offer other advantages. For example, it becomes possible to express and enforce integrity constraints on the site [FFLS98], and to update a site incrementally when changes occur in the underlying data. Moreover, a declarative specification provides a platform for developing optimization algorithms for run-time management of data intensive web sites. The challenge in run-time management of a web site is to automatically find an optimal tradeoff between precomputation of parts of the web site and click-time computation. Finally, we remark that building web sites using this paradigm will also facilitate the tasks of querying web sites and integrating data from multiple web sources.

A prototypical architecture of such a system is shown in Figure 3. At the bottom level, the system accesses a set of data sources containing the data that will be served on the web site. The data may be stored in databases, in structured files, or in existing web sites. The data is represented in the system in some data model, and the system provides a uniform interface to these data sources using techniques similar to the ones described in the previous section. The main step in building a web site is to write an expression that declaratively represents the structure of the web site. The expression is written in a specific query language provided by the system. The result of applying this query to the underlying data is the logical representation of the web site in the data model of the system (e.g., a labeled directed graph). Finally, to actually create a browsable web site, the system contains a method (e.g., HTML templates) for translating the logical structure into a set of HTML files.

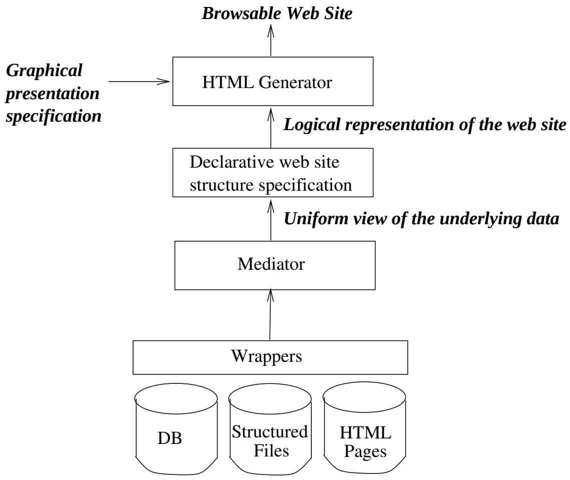

Figure 3: Architecture for Web Site Management Systems

Some of the salient characteristics of the different systems are the following. Strudel [FFK+98] uses a semistructured data model of labeled directed graphs for modeling both the underlying data and for modeling the web site. It uses a single query language, StruQL, throughout the system, both for integrating the raw data and for defining the structure of the web site. Araneus [AMM97b] uses a more structured data model, ADM, and provides a language for transforming data into ADM and a language for creating web sites from data modeled in ADM. In addition, Araneus uses web-specific constructs in the data model. The Autoweb [PF98] system is based on the hypermedia design model (HDM), a design tool for hypermedia applications. Its data model is based on the entity-relationship model; the "access schema" specifies how the hyperbase is navigated and accessed in a browsable site; and the "presentation schema" specifies how objects and paths in the hyperbase and access schemas are rendered. All three systems mentioned above provide a clear separation between the creation of the logical structure of the web site and the specification of the graphical presentation of the site. The YAT system [CDSS98] is an application of a data conversion language to the problem of building web sites. Using YAT, a web site designer writes a set of rules converting from the raw data into an abstract syntax tree of the resulting HTML, without going through an intermediate logical representation phase. In fact, in a similar way, other languages for data conversation (such as [MZ98, MPP+93, PMSL94]) can also be used to build web sites. The WIRM system [JB97] is similar in spirit to the above systems in that it enables users to build web sites in which in which the pages can be viewed context-sensitive views of the underlying data. The major focus of WIRM is on integrating medical research data for the national Human Brain Project.

6 Conclusions, Perspectives and Future Work

An overarching question regarding the topic of this survey is whether the World-Wide Web presents novel problems to the database community. In many ways, the WWW is not similar to a database. For example, there is no uniform structure, no integrity constraints, no transactions, no standard query language or data model. And yet, as the survey has showed, the powerful abstractions developed in the database community may prove to be key in taming the web's complexity and providing valuable services.

Of particular importance is the view of a large web site as being not just a database, but an information system built around one or more databases with an accompanying complex navigation structure. In that view, a web site has many similarities to non-web information systems. Designing such a web site requires extending information systems design methodologies [AMM98, PF98]. Using these principles to build web sites will also impact the way we query the web and the way we integrate data from multiple web sources.

Several trends will have significant impact on the use of database technology for web applications. The first is, of course, XML. The considerable momentum behind XML and related metadata initiatives can only help the applicability of database concepts to the web by providing the much needed structure in a widely accepted format. While the availability of data in XML format will reduce the need to focus on wrappers converting human readable data to machine readable data, the challenges of *semantic* integration of data from web sources still remains. Building on our experience in developing methods for manipulating semistructured data, our community is in a unique position to develop tools for manipulating data in XML format. In fact, some of the concepts developed in this community are already being adapted to the XML context [DFF+98, GMW98]. Other projects under way in the database community in the area of metadata architectures and languages (e.g. [MRT98, KMSS98]) are likely to take advantage of and merge with the XML framework.

A second trend that will affect the applicability of database techniques for querying the web is the growth of the so-called *hidden web*. The hidden web refers to the web pages that are generated by programs given user inputs, and are therefore not accessible to web crawlers for indexing. A recent article [LG98] claims that close to 80% of the web is already in the hidden web. If our tools are to be able to benefit from data in the hidden web, we must develop techniques for identifying sites that generate web pages, classify them and automatically create query interfaces to them.

There is no shortage in possible directions for future research in this area. In the past, the bulk of the work has focused on the logical level, developing appropriate data models, query languages and methods for describing different aspects of web sources. In contrast, problems of query optimization and query execution have received relatively little attention in the database community, and pose some of the more important challenges for future work. Some of the important directions in which to enrich our data models and query languages include the incorporation of various forms of meta data about sources (e.g., probabilistic information) and the principled combination of querying structured and unstructured data sources on the WWW.

Finally, in this article we tried to provide a representative list of references on the topic of web and databases. In addition to these references, readers can get a more detailed account from recent workshops related to the topic of the survey [SSD97, Web98, AII98].

Acknowledgements

The authors are grateful to the following colleagues who provided suggestions and comments on earlier versions of this paper: Serge Abiteboul, Gustavo Arocena, Paolo Atzeni, José Blakeley, Nick Kushmerick, Bertram Ludäscher, C. Mohan, Yannis Papakonstantinou, Oded Shmueli, Anthony Tomasic and Dan Weld.

References

[AAB+98] Jos Luis Ambite, Naveen Ashish, Greg Barish, Craig A. Knoblock, Steven Minton, Pragnesh J. Modi, Ion Muslea, Andrew Philpot, and Sheila Tejada. ARIADNE: A system for constructing mediators for internet sources (system demonstration). In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[Abi97] Serge Abiteboul. Querying semi-structured data. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

[ACM93] Serge Abiteboul, Sophie Cluet, and Tova Milo. Querying and updating the file. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Dublin, Ireland, 1993.

[ACPS96] S. Adali, K. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Montreal, Canada, 1996.

[AD98] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Seattle, WA, 1998.

[AFT98] Laurent Amsaleg, Michael Franklin, and Anthony Tomasic. Dynamic query operator scheduling for wide-area remote access. *Distributed and Parallel Databases*, 6(3):217–246, July 1998.

[AII98] *Proceedings of the AAAI Workshop on Intelligent Data Integration*, Madison, Wisconsin, July 1998.

[AK97] Naveen Ashish and Craig A. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.

[AKS96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *International Journal on Intelligent and Cooperative Information Systems*, (6) 2/3:99–130, June 1996.

[AM98] Gustavo Arocena and Alberto Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Orlando, Florida, 1998.

[AMM97a] Gustavo O. Arocena, Alberto O. Mendelzon, and George A. Mihaila. Applications of a web query language. In *Proc. of the Int. WWW Conf.*, April 1997.

[AMM97b] Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo. To weave the web. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, 1997.

[AMM98] Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo. Design and maintenance of data-intensive web sites. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[AMR+98] Serge Abiteboul, Jason McHugh, Michael Rys, Vasilis Vassalos, and Janet Weiner. Incremental maintenance for materialized views over semistructured data. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, New York City, USA, 1998.

[AQM+97] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[Aro97] Gustavo Arocena. WebOQL: Exploiting document structure in web queries. Master's thesis, University of Toronto, 1997.

[AV97a] S. Abiteboul and V. Vianu. Queries and computation on the Web. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

[AV97b] Serge Abiteboul and Victor Vianu. Regular path queries with constraints. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Tucson, Arizona, May 1997.

[BBH+98] Krisha Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: fast access to linkage information on the web. In *Proc. of the Int. WWW Conf.*, April 1998.

[BBMR89] Alex Borgida, Ronald Brachman, Deborah McGuinness, and Lori Resnick. CLASSIC: A structural data model for objects. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 59–67, Portland, Oregon, 1989.

[BDFS97] Peter Buneman, Susan Davidson, Mary Fernandez, and Dan Suciu. Adding structure to unstructured data. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

[BDH+95] C. M. Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1-2):119–125, December 1995.

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

[BEM+98] C. Beeri, G. Elber, T. Milo, Y. Sagiv, O.Shmueli, N.Tishby, Y.Kogan, D.Konopnicki, P. Mogilevski, and N.Slonim. Websuite-a tool suite for harnessing web data. In *Proceedings of the International Workshop on the Web and Databases*, Valencia, Spain, 1998.

[BH98] Krishna Bharat and Monika Henzinger. Improved algorithms for topic distillation in hyperlinked environments. In *Proc. 21st Int'l ACM SIGIR Conference*, 1998.

[Bil98] David Billard. Transactional services for the web. In *Proceedings of the International Workshop on the Web and Databases*, pages 11–17, Valencia, Spain, 1998.

[BK90] C. Beeri and Y. Kornatzky. A logical query language for hypertext systems. In *Proc. of the European Conference on Hypertext*, pages 67–80. Cambridge University Press, 1990.

[Bla96] Jose A. Blakeley. Data access for the masses through OLE DB. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 161–172, Montreal, Canada, 1996.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of the Int. WWW Conf.*, April 1998.

[BT98] Philippe Bonnet and Anthony Tomasic. Partial answers for unavailable data sources. In *Proceedings of the 1998 Workshop on Flexible Query-Answering Systems (FQAS'98)*, pages 43–54. Department of Computer Science, Roskilde University, 1998.

[Bun97] Peter Buneman. Semistructured data. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 117–121, Tucson, Arizona, 1997.

[CACS94] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 313–324, Minneapolis, Minnesota, 1994.

[CDF+98] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the world-wide web. In *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence*, 1998.

[CDRR98] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, and Sridhar Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. of the Int. WWW Conf.*, April 1998.

[CDSS98] Sophie Cluet, Claude Delobel, Jerome Simeon, and Katarzyna Smaga. Your mediators need data conversion. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[CGL98] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. What can knowledge representation do for semi-structured data? In *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence*, 1998.

[CGMP98] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through url ordering. In *Proc. of the Int. WWW Conf.*, April 1998.

[CK98] J. Carriere and R. Kazman. Webquery: Searching and visualizing the web through connectivity. In *Proc. of the Int. WWW Conf.*, April 1998.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Taipei, Taiwan, 1995.

[CM89] Mariano P. Consens and Alberto O. Mendelzon. Expressing structural hypertext queries in graphlog. In *Proc. 2nd. ACM Conference on Hypertext*, pages 269–292, Pittsburgh, November 1989.

[CM90] Mariano Consens and Alberto Mendelzon. GraphLog: a visual formalism for real life recursion. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 404–416, Atlantic City, NJ, 1990.

[CMW87] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A graphical query language supporting recursion. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 323–330, San Francisco, CA, 1987.

[CMW88] I.F. Cruz, A.O. Mendelzon, and P.T. Wood. G+: Recursive queries without recursion. In *Proceedings of the Second International Conference on Exper t Database Systems*, pages 355–368, 1988.

[Coh98] William Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[CS98] Pei Cao and Sekhar Sarukkai, editors. *Proceedings of Workshop on Internet Server Performance*, Madison, Wisconsin, 1998. http://www.cs.wisc.edu/ cao/WISP98-program.html.

[DEW97] B. Doorenbos, O. Etzioni, and D. Weld. Scalable comparison-shopping agent for the world-wide web. In *Proceedings of the International Conference on Autonomous Agents*, February 1997.

[DFF+98] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for XML. http://www.research.att.com/ mff/xml/w3c-note.html, 1998.

[DG97a] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Tucson, Arizona., 1997.

[DG97b] Oliver M. Duschka and Michael R. Genesereth. Query planning in infomaster. In *Proceedings of the ACM Symposium on Applied Computing*, San Jose, CA, 1997.

[Dus97] Oliver Duschka. Query optimization using local completeness. In *Proceedings of the AAAI Fourteenth National Conference on Artificial Intelligence*, 1997.

[EGW94] Oren Etzioni, Keith Golden, and Daniel Weld. Tractable closed world reasoning with updates. In *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning, KR-94.*, 1994. Extended version to appear in *Artificial Intelligence*.

[EW94] Oren Etzioni and Dan Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, 1994.

[FFK+98] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with Strudel: Experiences with a web-site management system. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[FFLS97] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for a web-site management system. *SIGMOD Record*, 26(3):4–11, September 1997.

[FFLS98] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. Reasoning about web-sites. In *Working notes of the AAAI-98 Workshop on Artificial Intelligence and Data Integration. American Association of Artificial Intelligence.*, 1998.

[FHM94] Douglas Fang, Joachim Hammer, and Dennis McLeod. The identification and resolution of semantic heterogeneity in multidatabase systems. In *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, pages 52–60. IEEE Computer Society Press, Los Alamitos, California, 1994.

[FKL97] Daniela Florescu, Daphne Koller, and Alon Levy. Using probabilistic information in data integration. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 216–225, Athens, Greece, 1997.

[FLS98] Daniela Florescu, Alon Levy, and Dan Suciu. Query containment for conjunctive queries with regular expressions. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Seattle,WA, 1998.

[FRV96] Daniela Florescu, Louiqa Raschid, and Patrick Valduriez. A methodology for query reformulation in cis using semantic knowledge. *Int. Journal of Intelligent & Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems*, 5(4), 1996.

[FW97] M. Friedman and D. Weld. Efficient execution of information gathering plans. In *Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan*, 1997.

[GC94] G. Graefe and R. Cole. Optimization of dynamic query evaluation plans. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Minneapolis, Minnesota, 1994.

[GGMT99] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. *GlOSS*: Text-source discovery over the internet. *ACM Transactions on Database Systems (to appear)*, 1999.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogenous information sources, March 1997.

[GMW98] R. Goldman, J. McHugh, and J. Widom. Lore: A database management system for XML. Presentation, Stanford University Database Group, 1998.

[GRC97] S. Gadde, M. Rabinovich, and J. Chase. Reduce, reuse, recycle: An approach to building large internet caches. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, May 1997.

[GRVB98] Jean-Robert Gruser, Louiqa Raschid, María Esther Vidal, and Laura Bright. Wrapper generation for web accessible data sources. In *Proceedings of the CoopIS*, 1998.

[GW97] Roy Goldman and Jennifer Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.

[GW98] Roy Goldman and Jennifer Widom. Interactive query and search in semistructured databases. In *Proceedings of the International Workshop on the Web and Databases*, pages 42–48, Valencia, Spain, 1998.

[GZC89] Ralf Hartmut Güting, Roberto Zicari, and David M. Choy. An algebra for structured office documents. *ACM TOIS*, 7(2):123–157, 1989.

[Hal88] Frank G. Halasz. Reflections on Notecards: Seven issues for the next generation of hypermedia systems. *Comm. of the ACM*, 31(7):836–852, 1988.

[Har94] Coleman Harrison. Aql: An adaptive query language. Technical Report NU-CCS-94-19, Northeastern University, October 1994. Master's Thesis.

[HGMN+98] Joachim Hammer, Hector Garcia-Molina, Svetlozar Nestorov, Ramana Yerneni, Markus M. Breunig, and Vasilis Vassalos. Template-based wrappers in the TSIMMIS system (system demonstration). In *Proc. of ACM SIGMOD Conf. on Management of Data*, Tucson, Arizona, 1998.

[HKWY97] Laura Haas, Donald Kossmann, Edward Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.

[HLLS97] Rainer Himmeröder, Georg Lausen, Bertram Ludäscher, and Christian Schlepphorst. On a declarative semantics for web queries. In *Proc. of the Int. Conf. on Deductive and Object-Oriented Databases (DOOD)*, pages 386–398, Singapore, December 1997. Springer.

[HML+98] Kyoji Hirata, Sougata Mukherjea, Wen-Syan Li, Yusaku Okamura, and Yoshinori Hara. Facilitating object-based navigation through multimedia web databases. *TAPOS*, 4(4), 1998. In press.

[Hul97] Richard Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61, Tucson, Arizona, 1997.

[HZ96] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 481–492, Montreal, Canada, 1996.

[Jan98] http://www.jango.excite.com, 1998.

[JB97] R. Jakobovits and J. F. Brinkley. Managing medical research data with a web-interfacing repository manager. In *American Medical Informatics Association Fall Symposium*, pages 454–458, Nashville, Oct 1997.

[Jun98] http://www.junglee.com, 1998.

[KD98] Navin Kabra and David J. DeWitt. Efficient mid-query re-optimization of sub-optimal query execution plans. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 106–117, Seattle, WA, 1998.

[KDW97] N. Kushmerick, R. Doorenbos, and D. Weld. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[Kle98] Jon Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[KMSS98] Y. Kogan, D. Michaeli, Y. Sagiv, and O. Shmueli. Utilizing the multiple facets of www contents. *Data and Knowledge Engineering*, 1998. In press.

[KS95] D. Konopnicki and O. Shmueli. W3QS: A query system for the World Wide Web. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 54–65, Zurich, Switzerland, 1995.

[KS98] David Konopnicki and Oded Shmueli. Bringing database functionality to the WWW. In *Proceedings of the International Workshop on the Web and Databases, Valencia, Spain*, pages 49–55, 1998.

[KW96] Chung T. Kwok and Daniel S. Weld. Planning to gather information. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[Lev96] Alon Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[LG98] S. Lawrence and C.L. Giles. Searching the world wide web. *Science*, 280(4):98–100, 1998.

[LHL+98] Bertram Ludäscher, Rainer Himmeröder, Georg Lausen, Wolfgang May, and Christian Schlepphorst. Managing semistructured data with *FLORID*: A deductive object-oriented perspective. *Information Systems*, 23(8), 1998. to appear.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, San Jose, CA, 1995.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[LRU96] Alon Y. Levy, Anand Rajaraman, and Jeffrey D. Ullman. Answering queries using limited external processors. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Montreal, Canada, 1996.

[LSB+98] T. Lee, L. Sheng, T. Bozkaya, N.H. Balkir, Z.M. Ozsoyoglu, and G. Ozsoyoglu. Querying multimedia presentations based on content. *to appear in IEEE Trans. on Know. and Data Engineering*, 1998.

[LSCH98] Wen-Syan Li, Junho Shim, K. Selcuk Canadan, and Yoshinori Hara. WebDB: A web query system and its modeling, language, and implementation. In *Proc. IEEE Advances in Digital Libraries '98*, 1998.

[LSS96] Laks V. S. Lakshmanan, Fereidoon Sadri, and Iyer N. Subramanian. A declarative language for querying and restructuring the Web. In *Proc. of 6th. International Workshop on Research Issues in Data Engineering, RIDE '96*, New Orleans, February 1996.

[MM97] Alberto O. Mendelzon and Tova Milo. Formal models of web queries. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 134–143, Tucson, Arizona, May 1997.

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

[MMM98] Giansalvatore Mecca, Alberto O. Mendelzon, and Paolo Merialdo. Efficient queries over web views. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[Mot89] Amihai Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, December 1989.

[MP96] Kenneth Moore and Michelle Peterson. A groupware benchmark based on lotus notes. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, 1996.

[MPP+93] Bernhard Mitschang, Hamid Pirahesh, Peter Pistor, Bruce G. Lindsay, and Norbert Sdkamp. Sql/xnf - processing composite objects as abstractions over relational data. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, pages 272–282, Vienna, Austria, 1993.

[MRT98] George A. Mihaila, Louiqa Raschid, and Anthony Tomasic. Equal time for data on the internet with websemantics. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[MZ98] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, New York City, USA, 1998.

[NAM98] Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Extracting schema from semistructured data. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[NGT98] Hubert Naacke, Georges Gardarin, and Anthony Tomasic:. Leveraging mediator cost models with heterogeneous data sources. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Orlando, Florida, 1998.

[NS96] Tam Nguyen and V. Srinivasan. Accessing relational databases from the world wide web. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Montreal, Canada, 1996.

[PAGM96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[PdBA+92] Jan Paredaens, Jan Van den Bussche, Marc Andries, Marc Gemis a nd Marc Gyssens, Inge Thyssens, Dirk Van Gucht, Vijay Sarathy, and Lawre nce V. Saxton. An overview of GOOD. *SIGMOD Record*, 21(1):25–31, 1992.

[PE95] Mike Perkowitz and Oren Etzioni. Category translation: Learning to understand information on the internet. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*. American Association for Artificial Intelligence., 1995.

[PE97] Mike Perkowitz and Oren Etzioni. Adaptive web sites: an AI challenge. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[PF98] P. Paolini and P. Fraternali. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[PGGMU95] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molina, and Jeffrey Ullman. A query translation scheme for rapid implementation of wrappers. In *Proc. of the Int. Conf. on Deductive and Object-Oriented Databases (DOOD)*, 1995.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, pages 251–260, Taipei, Taiwan, 1995.

[PMSL94] Hamid Pirahesh, Bernhard Mitschang, Norbert Sdkamp, and Bruce G. Lindsay. Composite-object views in relational dbms: an implementation perspective. *Information Systems*, IS 19(1):69–88, 1994.

[PPR96] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proc. ACM SIGCHI Conference*, 1996.

[RSU95] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, San Jose, CA, 1995.

[Spe97] Ellen Spertus. ParaSite: Mining structural information on the web. In *Proc. of the Int. WWW Conf.*, April 1997.

[SSD97] *Proceedings of Workshop on Management of Semistructured Data*, Tucson, Arizona, May 1997.

[TMD92] J. Thierry-Mieg and R. Durbin. A C.elegans database: syntactic definitions for the ACEDB data base manager, 1992.

[TN98] Motomichi Toyama and T. Nagafuji. Dynamic and structured presentation of database contents on the web. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[TRV98] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to distributed heterogeneous data sources with Disco. IEEE Transactions On Knowledge and Data Engineering *(to appear)*, 1998.

[TSI96] Odysseas G. Tsatalos, Marvin H. Solomon, and Yannis E. Ioannidis. The GMAP: A versatile tool for physical data independence. *VLDB Journal*, 5(2):101–118, 1996.

[UFA98] Tolga Urhan, Michael J. Franklin, and Laurent Amsaleg. Cost based query scrambling for initial delays. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 130–141, Seattle, WA, 1998.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

[VP97a] Vasilis Vassalos and Yannis Papakonstantinou. Describing and using the query capabilities of heterogeneous sources. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.

[VP97b] Anne-Marie Vercoustre and François Paradis. A descriptive language for information object reuse through virtual documents. In *Proceedings of the 4th International Conference on Object-Oriented Information Systems (OOIS'97)*, Brisbane, Australia, November 1997.

[WAC+93] Darrel Woelk, Paul Attie, Phil Cannata, Greg Meredith, Amit Seth, Munindar Sing, and Christine Tomlinson. Task scheduling using intertask dependencies in Carnot. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 491–494, 1993.

[WBJ+95] Darrell Woelk, Bill Bohrer, Nigel Jacobs, K. Ong, Christine Tomlinson, and C. Unnikrishnan. Carnot and infosleuth: Database technology and the world wide web. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 443–444, San Jose, CA, 1995.

[Web98] *Proceedings of the International Workshop on the Web and Databases*, Valencia, Spain, March 1998. http://poincare.dia.uniroma3.it:8080/webdb98/papers.html.

[WWW98] *Proceedings of 3d WWW Caching Workshop (Manchester, England)*, June 1998.

[XML98] http://w3c.org/XML/, 1998.

[YL87] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 245–254, Brighton, England, 1987.

[YPAGM98] Ramana Yerneni, Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina. Fusion queries over internet databases. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, pages 57–71, Valencia, Spain, 1998.

[ZGM98] Yue Zhuge and Hector Garcia-Molina. Graph structured views and their incremental maintenance. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Orlando, Florida, February 1998.