Equal Time for Data on the Internet with WebSemantics[⋆]

George A. Mihaila[1], Louiqa Raschid[2] and Anthony Tomasic[3]

[1] Department of Computer Science, University of Toronto, Canada
[2] Maryland Business School and UMIACS, University of Maryland, USA
[3] INRIA Rocquencourt, 78153 Le Chesnay, France

Abstract. Many collections of scientific data in particular disciplines are available today around the world. Much of this data conforms to some agreed upon standard for data exchange, i.e., a standard schema and its semantics. However, sharing this data among a global community of users is still difficult because of a lack of standards for the following necessary functions: (i) data providers need a standard for describing or publishing available sources of data; (ii) data administrators need a standard for discovering the published data and (iii) users need a standard for accessing this discovered data. This paper describes a prototype implementation of a system, WebSemantics, that accomplishes the above tasks. We describe an architecture and protocols for the publication, discovery and access to scientific data. We define a language for discovering sources and querying the data in these sources, and we provide a formal semantics for this language.

1 Introduction

Recently, many standardized collections of scientific data, in specific disciplines, e.g., the environmental sciences, have become available. For example, governments have recognized that information about the environment has important consequences in the management of sustainable economic growth. As a result, many collections of environmental data, located around the world, are now available to scientists [Fra97]. Much of this data conforms to existing standards for the definition of data. These standards help scientists share information since each item of data is precisely defined. However, the sharing of information between scientists is still a very difficult process. Sharing is hindered by the lack of a standard system for *describing* the data that is available, the lack of a standard method for *discovering* the existence of data relevant to a problem, and the lack of a standard method for *accessing* discovered relevant data.

The World-Wide Web (WWW) provides a standard system for sharing documents. In this system there is a standard way to publish and access documents via the HTTP protocol and HTML document format standards. In addition,

⋆ This research was partially sponsored by the National Science Foundation grant IRI9630102 and the Defense Advanced Research Projects Agency grant 01-5-28838

there is a standard way to discover relevant documents, through the use of search engines. Because of the inherent nature of documents, only a loose standard is required for the definition of the meaning of a document – the language in which the document is written.

Typed data cannot be shared using such a system, and in this paper we describe the implementation of a prototype system which permits publication, discovery and access to data. Our goal is to make the access to structured data as easy as access to documents, that is, to give "equal-time" to data on the Internet. Our approach extends the WWW with a protocol for publishing the existence of data sources in documents, via a *metadata description*. We then provide a language for discovering relevant published data sources via the metadata. The language combines features for searching over the metadata with features for searching relevant documents that contain this metadata. This approach smoothly integrates the functionality already existing on the WWW for searching documents with our extensions for searching over the metadata. Once relevant data sources have been found, a query language provides access to this data.

Consider a scientist who samples water in the Seine river at Notre Dame in Paris. She analyzes the sample for hydro-biological material. The schema describing the sample and the result of the analysis has been standardized in France by the *Secrétariat d'Administration National des Données Relatives à l'Eau (SANDRE)* (the national administration of water related data) [SAN95]. The schema includes a standard taxonomy of fauna, the list of fauna found in the sample, the method (from a standard set of methods) used to analyze the sample, the date, time, and duration of the sampling operation, etc. Suppose that this scientist has stored the results of the analysis in an `Oracle` database. This data source can be published via the WebSemantics system with an HTML file like the one shown in Fig. 1.

This document pairs machine-readable information for accessing the database with a description of the data in the database. More precisely, the new `WSSCI` tag contains WebSemantics *source connection information* such as the address of a *wrapper* module (needed to access the data), the Internet address of the database server, authentication information, etc. We say that this document is an WS-MXF (WebSemantics*Metadata Exchange Format*) document that *describes* the data source.

Suppose a second scientist knows that the server `www.env.org` has links to documents that publish data sources. Then, he can execute the following query:

Query 1.
select s
from Document d **such that** "http://www.env.org/" $\rightarrow^*$ d,
Source s **such that** d **describes** s
where $d.text$ **contains** "water pollution";

This query finds any WebSemantics document that is linked to the given server and that mentions water pollution. The query then extracts the information

```
<HTML>
<HEAD>
<TITLE>Environmental Data for Paris</TITLE>
<WSSCI WRAPPER = "http://www.cs.toronto.edu/ws/WS-Oracle.class"
      ADDR  = "server.env.org:8001"
      USER  = "guest" PASSWD = "1234" >
</HEAD>
<BODY>
This repository contains daily measurements of water pollution
quality parameters in Paris for the year 1996.  The data conforms
to the SANDRE standard.  The samples of water were obtained using
the following equipment:

etc.

</BODY>
</HTML>
```

Fig. 1. Publishing a data source in an HTML file

about data sources that are published in that document. The scientist can then *register* these data sources to his *catalog* of data sources. Registering a data source means that the set of types of the data source and other additional information are made available to the catalog. Once a catalog has been constructed, other scientists can query this catalog, identify data sources in the catalog and access the data from these sources. The details of data access are described in Sect.3.4.

To validate our research we have constructed WebSemantics with environmental information and a community of environmental scientists in mind. However, the underlying architecture and language are not specific to environmental data or scientific data in general. Given a standard semantic description for some data, WebSemantics can be utilized to share this data. For example, given a standardized semantic description of the want-ads of newspapers, (e.g., make, model, year, and price of a car for sale), users of the system can publish databases of want-ads and search for other databases of want-ads. WebSemantics provides the infrastructure for constructing this community of users.

The above example introduces several questions for which this paper gives preliminary answers. What metadata about data sources, in addition to the types, should be shared using the catalog? What metadata is obtained from the sources? What data model and language should be used to describe metadata of interest? How are text and type information combined in queries?

In summary, the contributions of this paper are as follows:

– A formal model and a physical architecture of a system which permits pub-

lication, discovery and access to data sources;
- A query language, WSQL, addressing the issue of resource discovery by context and allowing access to discovered relevant data;
- A formal calculus defining the semantics of the above query language.

The paper is organized as follows: Section 2 introduces a formal model of the objects manipulated by the system; Section 3 presents the WSQL language for the discovery, registration and access to data sources; Section 4 describes the semantics of the WSQL language; Section 5 describes the WebSemantics architecture; Section 6 describes related work; and Section 7 concludes the paper.

2 The WebSemantics Metadata Model

The WebSemantics system operates both at the data level (by providing uniform access to the tuples stored in repositories) and at the *metadata* level. In this section we introduce the metadata objects manipulated by the system: relational types, data sources, catalogs, active domains, WWW documents and links, which together form what we call the WebSemantics *universe*.

Definition 1. *The* WebSemantics *Universe is an 8-tuple $WSU = (\mathcal{T}, \mathcal{S}, \mathcal{N}, \mathcal{C}, \mathcal{D}, \mathcal{L}, \rho_{\mathcal{L}}, \rho_{\mathcal{S}})$*
where:

- *$\mathcal{T}$ is the set of all relational types exported by all sources;*
- *$\mathcal{S}$ is the set of all data sources;*
- *$\mathcal{N}$ is the set of all active domains for all sources.*
- *$\mathcal{C}$ is the set of all catalogs;*
- *$\mathcal{D}$ is the set of all the HTML documents in the WWW;*
- *$\mathcal{L}$ is the set of all links between documents;*
- *$\rho_{\mathcal{L}} : \mathcal{D} \to 2^{\mathcal{L}}$ is a mapping defining for each document the set of its outgoing links;*
- *$\rho_{\mathcal{S}} : \mathcal{D} \to 2^{\mathcal{S}}$ is a mapping defining for each document the set of sources it* describes*;*

The metadata model is represented by six *virtual relations*: *Type*, *Source*, *Domain*, *Catalog*, *Document* and *Link*. These relations are not materialized. Access to these virtual relations is distributed over various components in the WebSemantics architecture.

Definition 2. *A relational type is modeled by a tuple in the virtual relation* Type(name, attributes) *where:*

- name *is a string uniquely identifying the type;*
- attributes *is a set of pairs (*name, scalar_type*), where* scalar_type $\in$ {String, Int, Float, Boolean}*.*

Definition 3. *A data source is represented by a tuple in the virtual relation* Source(id, types, description, url, sci)*, where:*

- id *is a string identifying the source;*
- types *is the set of all relational types available is this source (represented by* Type *tuples);*
- description *is a textual description of the data present in this source;*
- url *is the location of the HTML document publishing this source;*
- sci *is the source connection information tuple: a nested tuple with variable structure, containing at least a field named* wrapper*, containing the location of the wrapper, and possibly other fields (eg.* addr*,* username*,* passwd*).*

One component of the metadata maintained by WebSemantics is a compact representation of the current contents of data sources, also known as the *active domain.*

Definition 4. *The active domain of a source is modeled by a set of tuples in a virtual relation* Domain(source_id, type, attribute, values) *where:*

- source_id *is the source's identifier;*
- type *is the name of a relational type;*
- attribute *is the name of an attribute of the type* type*;*
- values *is a textual representation of a set of values for the specified attribute;*

A catalog C stores information about a collection of data sources and the types and domains exported by them:

Definition 5. *A catalog is represented by a tuple in the virtual relation* Catalog(addr, sources, types, domains) *where:*

- addr *is the RMI address of the catalog service;*
- sources *is a set of Source tuples;*
- types *is the set of all types exported by the sources registered in the catalog;*
- domains *is the set of all domains exported by the sources;*

We identify a catalog with its RMI address.

The WebSemantics system uses the World Wide Web for publishing data sources. The following definition refers to objects in the WWW: documents and links.

Definition 6. *A WWW document is modeled by a tuple in the virtual relation* Document(url, title, text, date)*, where the attributes are the document's URL, title, text, and last modification date, respectively. We identify a document with its URL.*

A hypertext link is modeled by a tuple in the virtual relation Link(base, href, label)*, where the* base *is the URL of the source document,* href *is the URL of the target document, and* label *is the link's labeling text.*

3 Locating, Registering and Querying Data Sources

In this section we introduce the WebSemantics Query Language (WSQL) which provides the following functions: 1) finding data sources published in WS-MXF documents on the WWW or registered in existing catalogs; 2) registering sources in a catalog; 3) selecting sources based on their contents; and 4) extracting data from sources. To accomplish these tasks, the WSQL language integrates constructs borrowed from WebSQL [MMM96] and OQL [C+96].

3.1 Finding Sources on the WWW

In Sect.1 we described the WebSemantics Metadata Exchange Format (WS-MXF) for publishing information about data sources in Web pages. In order to access this information, the WebSemantics system needs to locate these pages and extract the source connection information from them.

Consider the following scenario: researchers at an institute for environmental studies publish on the institute's Web server several HTML pages describing data sources containing measurements of various parameters. A scientist in a different location, interested in this data, knows only the home page of the institute, and would like to locate all data sources related to water pollution. Then, the following query would build the desired collection of sources:

Query 2.
select s
from Document d **such that** "http://www.env.org/" $\rightarrow^{*}$ d,
Source s **such that** d **describes** s
where $d.text$ **contains** "water pollution";

The first construct in the **from** clause sets the range of the variable d to the set of all documents on the "www.env.org" server which are reachable from the root page. The path regular expression '$\rightarrow^{*}$' means "traverse any number of local links starting from the specified URL". The set of documents of interest is restricted by a predicate in the **where** clause which specifies a string containment condition on $d.text$. The second construct in the **from** clause sets the range of the variable s to the set of sources described in the documents which satisfy the predicate.

Path regular expressions, a construct borrowed from WebSQL, are regular expressions over the alphabet of link types: $\rightarrow$ — *local* link between documents on the same Web server, $\Rightarrow$ — *global* link to a different Web server, $\mapsto$ — *interior* link to a position in the same document, and $\overset{u}{\rightarrow}$ — a user defined link type[4].

[4] For more examples of WSQL queries, including user defined link types, one can see [MRT97]

3.2 Selecting Sources from Catalogs

We have seen in the previous section how we can extract sources from the WWW. Once specialized catalogs have been built, however, it is reasonable to pick certain sources directly from these catalogs, instead of exploring the WWW.

For instance, going back to our example with environmental data, assuming one knows the RMI addresses of several catalogs registering relevant data sources, one can extract sources of interest from these catalogs with the following query:

Query 3. Find all data sources containing mean values of the UV index for Paris, according to a specific list of catalogs.

```
select  s
from    Catalog c in {"rmi://an.env.org/WSCat", "rmi://rep.env.fr/WSCat"},
        Source s in c.sources,
        Domain d in c.domains,
where   d.type = "UVIndex" and d.att = "city"
        and d.values contains "Paris"
        and d.source_id = s.id;
```

3.3 Registering Sources

We have presented example queries to extract and build sets of data sources based on their metadata. In order to make such a set of sources available to the WebSemantics system they have to be registered in a catalog. This is accomplished by the **register** command, whose syntax is specified below:

register into catalog *catalog_address* (**sources** | **types**) *WSQL_query*

Thus, the **register** command can be used to register either a set of sources or a set of types, as computed by a WSQL query.

3.4 Selecting and Querying Data Sources

In the previous sections, we described the process of locating and registering data sources in a catalog. Once catalogs have been built, they can be queried to identify sources for data access. In this section, we describe the features of the WSQL language that can be used to select sources in a catalog and then extract tuples from these sources.

Although the WSQL language allows the user to combine the source discovery features with data extraction, we expect that most user queries will be executed against sources registered in the default catalog. To support the typical user, we expect that a WebSemantics database administrator will use the Web navigation and catalog interrogation features of WSQL to maintain the default catalog.

The WSQL features support the following operations:

- Selecting data sources from the catalog based on the meta-data describing them, for example, the types that are supported by each source, the textual descriptions of the data, the domains of some types, etc.
- Comparing the meta-data contents of two sources or the data contents of two sources.
- Combining source selection with data extraction.

Once the sources are selected from the catalog, then an appropriate subquery against the selected type(s) that are supported by that source will be submitted and answer tuples will be obtained as the result to this subquery. Answers from multiple sources will be combined by evaluating those operations in the WS query processor.

For example, if we are interested in all the measured values of the UV index in Canada in a specific day, we can write the following query:

Query 4.
select *x.uv_index*
from Source *s*,
UVIndex *x* in *s*
where *s.types* $\supseteq$ {UVIndex} **and** *s.description* **contains** "Canada"
and *x.date* = "30.06.1997" **and** *x.country* = "Canada"

The set of sources that need to be contacted is restricted by explicitly including a content-dependent selection of the sources from the default catalog.

4 Formal Semantics

In this section we give a formal semantics for the WSQL language, by introducing a calculus and providing the translation rules from the language to this calculus.

We start by specifying the meaning of the various constructs in the **from** clause. Every variable in a query is defined by a *range*, i.e. a set of values. The query examples given so far illustrate several ways to specify a range. The following definition introduces domain calculus predicates for each of these syntactic constructs.

Definition 7. *Consider the* WebSemantics *universe* $WSU = (\mathcal{T}, \mathcal{S}, \mathcal{N}, \mathcal{C}, \mathcal{D}, \mathcal{L}, \rho_{\mathcal{L}}, \rho_{\mathcal{S}})$ *and a finite set* Λ *of link types. Consider also a multi-sorted collection of* metadata variables $V_{Meta} = \bigcup_{\mathcal{X} \in \{\mathcal{T},\mathcal{S},\mathcal{N},\mathcal{C},\mathcal{D}\}} V_{\mathcal{X}}$ *and a separate multi-sorted collection of* data variables $V_{Data} = \bigcup_{t \in \mathcal{T}} V_t$*, where the subscript specifies the sort of the variables. Also, denote by* $Dom_{Meta} = \bigcup_{\mathcal{X} \in \{\mathcal{T},\mathcal{S},\mathcal{N},\mathcal{C},\mathcal{D}\}} \mathcal{X}$*, the* meta-data domain*, and by* $Dom_{Data} = \bigcup_{t \in \mathcal{T}} Dom_t$*, the* data domain*, where for each type* $t \in \mathcal{T}$*,* Dom_t *is the set of all possible data tuples of type* t*.*

A range atom *is an expression of one of the following forms:*

- $Path(d, R, e)$*, where* $d \in \mathcal{D} \cup V_{\mathcal{D}}$*,* R *is a regular expression over* Λ*, and* $e \in V_{\mathcal{D}}$*;*

– *Describes*(d, s), *where* $d \in \mathcal{D} \cup V_{\mathcal{D}}$ *and* $s \in V_{\mathcal{S}}$*;*
– $x \in extent(s, t)$*, where* $s \in V_{\mathcal{S}}$*,* $t \in \mathcal{T} \cup V_{\mathcal{T}}$*, and* $x \in V_{\mathcal{T}}$*;*
– $x \in \{c_1, ..., c_k\}$*, where* $x \in V_{\mathcal{X}}$ *and* $c_1, ..., c_k \in \mathcal{X}$ *are constants* ($\mathcal{X} \in \{\mathcal{T}, \mathcal{S}, \mathcal{N}, \mathcal{C}, \mathcal{D}\}$)*;*
– $x \in u.attr$*, where* $u, x \in V_{Meta}$ *and* $attr$ *is a set-valued attribute;*

A range expression *is an expression of the form* $\{(x_1, ..., x_n) | A_1 \wedge ... \wedge A_m\}$*, where* $A_1, ..., A_m$ *are range atoms and* $x_1, ..., x_n$ *are all the variables occurring in them.*

Consider a valuation $\nu : V_{Meta} \cup V_{Data} \rightarrow Dom_{Meta} \cup Dom_{Data}$, such that for all variables $x \in V_{\mathcal{X}}$, $\nu(x) \in \mathcal{X}$. We extend ν to the identity function on $Dom_{Meta} \cup Dom_{Data}$. The following definition assigns semantics to range atoms.

Definition 8. *Let* A *be a range atom. We say that* A *is validated by the valuation* ν *if:*

– *for* $A = Path(d, R, e)$*, if there exists a simple path from* $\nu(d)$ *to* $\nu(e)$ *matching the regular expression* R*;*
– *for* $A = Describes(d, s)$*, if* $\nu(s) \in \rho_{\mathcal{S}}(\nu(d))$ *(the document* $\nu(d)$ *contains connection information for the source* $\nu(s)$*);*
– *for* $A = x \in extent(s, t)$*, if* $\nu(x)$ *is a data tuple of type* $\nu(t)$ *in source* $\nu(s)$*;*
– *for* $A = x \in \{c_1, ..., c_k\}$*, if* $\nu(x) \in \{c_1, ..., c_k\}$*;*
– *for* $A = x \in u.attr$*, if* $\nu(x) \in \nu(u).attr$*;*

Now we can give semantics to range expressions.

Definition 9. *Let* $\mathcal{E} = \{(x_1, ..., x_n) | A_1 \wedge ... \wedge A_m\}$ *be a range expression. Then, the set of tuples* $\Psi(\mathcal{E}) = \{(\nu(x_1), ..., \nu(x_n)) | \nu$ *is a valuation s.t.* $A_1, ..., A_m$ *are all validated by* $\nu\}$ *is called the* range *of* $\mathcal{E}$*.*

Range expressions model the syntactic constructs in the **from** clause. In order to model the **select** and **where** clauses, we need to include in our calculus the traditional relational operators *project* (π) and *select* (σ), with the standard semantics:

Definition 10. *Let* $X = \{x_1, ..., x_n\} \subset V_{Meta} \cup V_{Data}$ *be a set of variables. Let* $T = \{(\nu_i(x_1), ..., \nu_i(x_n)) \mid \nu_i$ *valuation,* $i = 1, \cdots, m\}$ *be a set of tuples. Let* ϕ *be a Boolean expression over the variables in* X *(involving any combination of equality, inequality, set containment and string containment tests). Then, the result of applying the* select *and* project *operators on* T *is:*

$$\sigma_{\phi}(T) = \{t \in T | \phi(t) = true\}$$

$$\pi_{x_{i_1}.a_{j_1}, ..., x_{i_k}.a_{i_k}}(T) = \{(x_{i_1}.a_{j_1}, ..., x_{i_k}.a_{i_k}) | (x_1, ..., x_n) \in T\}$$

We are now ready to specify the semantics of WSQL in terms of the introduced calculus. We first consider un-nested WSQL queries. Thus, a query of the form:

```
select  L
from    C_1, C_2, ..., C_m
where   φ;
```

translates to the following calculus query:

$$\pi_L \sigma_\phi \{(x_1, ..., x_n) | A_1 \wedge ... \wedge A_m\}$$

where each range atom A_i is obtained from the corresponding condition in the **from** clause by applying the following transformation rules:

- if C_i = "Document e **such that** $d\ R\ e$", then $A_i = Path(d, R, e)$;
- if C_i = "Source s **such that** d **describes** s", then $A_i = Describes(d, s)$;
- if C_i = "$MetaType\ v$ **in** $\{c_1, ..., c_k\}$", then $A_i = v \in \{c_1, ..., c_k\}$ ($MetaType$ is any of the virtual relations introduced in Sect.2);
- if C_i = "$MetaType\ v = c$", then $A_i = v \in \{c\}$;
- if C_i = "$MetaType\ v$ **in** $u.attr$", then $A_i = v \in u.attr$;
- if C_i = "$t\ v$ **in** s", then $A_i = v \in extent(s, t)$;
- if C_i = "$t\ v$", then $A_i = v \in extent(s, t)$ and we introduce one extra atom $A_{m+1} = s \in C.sources$, where C is the default catalog and $s \in V_{\mathcal{S}}$ is a new variable;

For example, Query 1 translates to the following calculus query:

$$\pi_s \sigma_{d.text \supset \text{"water pollution"}} \{(d, s) | Path(www.env.org, \rightarrow^*, d) \wedge Describes(d, s)\}$$

For queries that contain subqueries, the semantics is defined recursively starting from the inner-most subquery.

5 Architecture

In this section we introduce the WebSemantics architecture and components, and describe the functionality and interactions of these components.

We assume that a community of users in a specific discipline have agreed upon the data model, data exchange format, and semantics of the data to be shared, and have provided a set $\mathcal{T}$ of tuple types (in the relational data model). In other words, we assume *semantic homogeneity* within a well-defined application domain. Each type in $\mathcal{T}$ is defined by a set of named attributes having scalar types (*string, int, float*, etc.).

5.1 The WebSemantics Components

The WebSemantics system has a layered architecture of interdependent components (see Fig.2). We begin by introducing each layer, and leave the description of the way the components interact to the next section.

The *Data Source Layer* has two components, *data sources* and *wrappers*. Data providers create and manage a set $\mathcal{S}$ of autonomous *data sources*, that can be accessed over the Internet. These data sources can provide query capability ranging from full DBMS to simple scaning of files. WebSemantics provides uniform access to both kinds of sources, independent of the capability of the sources. This is accomplished using *wrapper* components. These components provide two kinds of functionality. First, they provide the metadata (types, domains, etc.) needed to register a data source in a *catalog* (the catalog component is described later). This functionality is defined by a *Metadata Protocol*. Second, wrappers accept a WebSemantics query and provide answers. This functionality is supported by the *Query and Answer Protocol*[5].

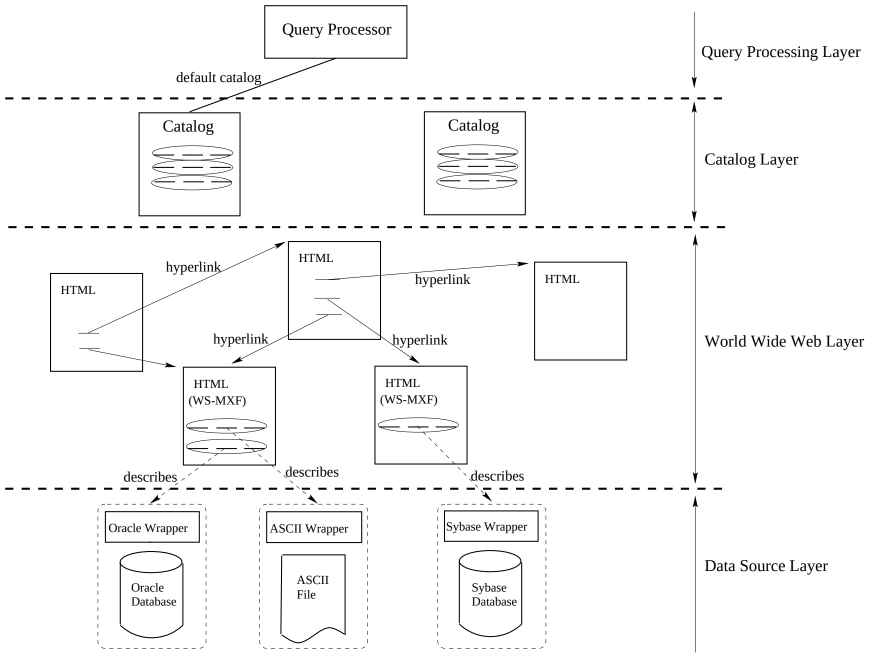

Fig. 2. The WebSemantics layers

The next layer is the *World Wide Web Layer*. This layer includes WebSemantics *documents* that publish data sources as well as other documents and data on the WWW. To publish a source, the provider needs to identify the location of the wrapper, and some wrapper-specific connection parameters, such as the address of the source, user name, password., etc. This information is made available by publishing it in a WWW-accessible HTML document using the WebSemantics metadata exchange format (WS-MXF), described in

[5] A more detailed description of WebSemantics wrappers and their supported protocols is given in [MRT97]

Sect.1. The collection of all HTML documents, including these WebSemantics documents, connected through hyperlinks, constitutes the *World Wide Web Layer*.

The next layer is the *Catalog Layer*. A *catalog* is a specialized repository storing collections of SCI tuples describing the source. In addition to this connection information, a catalog stores additional metadata characterizing the data source, such as the set of types exported by the source, the domain for a subset of attributes of some types, a textual description of the data source obtained from the WebSemantics document describing the source, the URL of the WebSemantics document that published the source, etc. Individual users can select sources and register them in a *catalog*.

Finally, the WebSemantics *query processor* component, bound to a specific catalog, gives the user integrated access to the collection of sources registered in the catalog and to the data in these sources.

Due to lack of space, we omit further details of our prototype implementation. Our prototype is available at `http://www.cs.toronto.edu/~georgem/ws/`.

5.2 Interactions between Components

WebSemantics supports the tasks of resource discovery and access to data in the discovered sources. Thus, interactions between components occur in two ways: updates to the catalogs as sources are discovered, and query processing where sources in the catalog are identified and queries are submitted to these sources.

Sources can be registered in the catalog in two ways. First, a data provider may explicitly register a source via a HTML forms based interface. In this case, all metadata that is stored in the catalog, including the types, domains, etc., may be provided via the form. Alternately, the wrapper that is identified during registration may be contacted to obtain this metadata.

The second method reflects resource discovery and is an interaction between all the components. The query processor evaluates a query in the WSQL language (described in Sect.3) and constructs a set of sources which are then registered in the catalog. The catalog contacts the wrapper associated with each of these sources via the Metadata Protocol to extract metadata about the sources, e.g., type and domain information.

Query processing proceeds as follows: a query, either generated by an application program, or entered through a parameterized user interface, is sent to the query processor. There are two phases to query processing. In the first phase, the set of data sources on which the query is to be evaluated must be bound. The query can declaratively specify the set of data sources to be used, in terms of selections on the default catalog, on other catalogs, or even by utilizing the resource discovery features of the language (that was used to discover sources). If unspecified, the set of sources is bound to the set of all sources registered in the default catalog (bound to the query processor).

After the set of sources has been determined, the query processor issues subqueries to each wrapper associated with some source. This time, the interaction with the wrapper is done via the Query and Answer protocol. The query processor combines the results, which are then returned to the user.

6 Related Work

Current research in mediation technology has proposed several techniques for describing, integrating or accessing structured data on the Internet. However, little research has been done for the problem of *locating* data sources on-the-fly and querying their contents. This is the main contribution of WebSemantics. However, WebSemantics is not a stand-alone technology, and it depends on the existence of other technologies. In this section, we review this research.

There are currently numerous wrapper mediator architectures, as proposed in [ACPS96, HKWY97, G$^+$96, KLSS95, R$^+$89, P$^+$96, TRV96, Wie92]. These systems differ widely in the capabilities of mediators and in the capabilities of wrappers. However, we believe that interoperation between WebSemantics and these systems is possible. WebSemantics differs in two distinct ways from these systems. Wrappers developed for Garlic [HKWY97] and the Information Manifold [KLSS95] assume the location of the data sources, types, and wrapper capability, are embedded within the wrappers. Wrappers for DISCO [TRV96] and TSIMMIS [GM$^+$95, VP97] use declarative languages to export types and query capabilities. The location of components is embedded within the mediators. Our main contribution is an architecture that uses WWW documents to publish the location of components (wrappers and data sources) and a uniform query language to locate data sources (based on metadata) and to access data from the sources.

Distributed information retrieval systems, for example the Harvest/Essence information retrieval based system [B$^+$95] are indirectly related to our work. Essence is a customizable information extraction system that is used to extract and structure mostly textual information from documents in Harvest. It exploits the formats of common file types and extracts contents of files. The result is a summary object (a SOIF record). Collections of SOIF records are indexed and organized into *brokers*. Brokers provide information retrieval search on their associated SOIF records. The information stored in SOIF records is similar to the metadata about sources maintained by WebSemantics catalogs. However, these systems focus on passive documents and information retrieval search, whereas we focus on querying strictly typed data in both active and passive data repositories.

The importance of the World Wide Web as a repository of information has generated an increasing interest in the research community for the design of high-level, declarative languages for querying it. WebSQL [MMM96] integrates textual retrieval with structure and topology-based queries. There, a minimalist relational approach is taken: each Web document is associated with a tuple in a virtual `Document` relation and each hypertext link with a tuple in a virtual `Anchor` relation. In order to query these virtual tables, one must first define computable sub-domains, either using keyword matching or through controlled navigation starting from known URLs. Another Web query language, W3QS [KS95] includes the specification of syntax and semantics of a SQL-like query language (W3QL) that provides simple access to external Unix programs, advanced display facilities for gathered information, and view maintenance facilities.

Recently there is an increasing interest within the World Wide Web Consortium (W3C) for providing standard formats for machine-readable resource content descriptions, to allow for the automation of Web information retrieval and processing. Among the currently evolving standards we can mention the Resource Description Framework [RDF] and the Meta Content Framework [MCF]. Both standards use the Extensible Markup Language [XML] as a common data encoding format and provide a framework for defining and publishing a shared vocabulary of entities and properties of these entities. Clearly, a system like WebSemantics would benefit from such standardization efforts.

7 Conclusion

We have presented a system, WebSemantics, that provides a multi-layered infrastructure for publishing, discovering and accessing structured data on the Internet. Thus, the *Data Source Layer* contains *sources*, which contain data stored in either an active repository (such as a DBMS) or a passive collection of files and *wrappers*, which are software components that isolate the differences in query capabilities and data exchange formats between data sources and also provide metadata information about the contents of sources. To allow for dynamic location of data sources we proposed a special type of HTML document pairing data source connection information with a textual description of the data source content. This provides an easy way to publish sources and allows the use of information retrieval techniques for the location of relevant data sources. Thus, the second layer is the *World Wide Web Layer* containing all the HTML documents together with the hyperlinks between them. The third layer, the *Catalog Layer*, consists of *catalogs*, which are specialized repositories storing connection information and metadata about data sources (such as types, active domain, and natural language description). Finally, the *Query Processing Layer* provides content-dependent selection of sources and integrated data access.

We introduced a formal model, meant to capture the concepts manipulated by the system and the relationships between them. Furthermore, we introduced a declarative query language, WSQL whose purpose is to facilitate discovery and registration of data sources, content-dependent selection of sources and data access. We specified the semantics of this language by defining a domain calculus over the previously introduced model.

References

[ACPS96] S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmaniam. Query caching and optimization in distributed mediator systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 137–148, 1996.

[B$^+$95] C. Bowman et al. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28:119–125, 1995.

[C$^+$96] R.G.G. Cattell et al. *The Object Database Standard - ODMG 93, Release 1.2.* Morgan Kaufmann, 1996.

[Fra97] Michael J. Franklin, editor. *SIGMOD Record*, volume 26, March 1997. Special Section on Environmental Information Systems.

[G$^+$96] G. Gardarin et al. IRO-DB: A distributed system federating object and relational databases. In O.A. Bukhres and A.K. Elmagarmid, editors, *Object-Oriented Multidatabase Systems : A solution for Advanced Applications.* Prentice Hall, 1996.

[GM$^+$95] H. Garcia-Molina et al. Integrating and accessing heterogeneous information sources in TSIMMIS. In *Proceedings of the AAAI Symposium on Information Gathering*, pages 61–64, Stanford, California, March 1995.

[HKWY97] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proceedings of VLDB'97*, pages 276–285, 1997.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proc. of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments*, Stanford, California, March 1995.

[KS95] D. Konopnicki and O. Shmueli. W3QS: A query system for the World Wide Web. In *Proceedings of VLDB'95*, pages 54–65, 1995.

[MCF] Meta Content Framework using XML. http://www.w3.org/TR/NOTE-MCF-XML-970624.

[MMM96] Alberto O. Mendelzon, George A. Mihaila, and Tova Milo. Querying the World Wide Web. In *Proceedings of PDIS 96*, pages 80–91, 1996.

[MRT97] George A. Mihaila, Louiqa Raschid, and Anthony Tomasic. Equal Time for Data on the Internet with WebSemantics. Technical report, University of Toronto, 1997. http://www.cs.toronto.edu/~georgem/ws/ws.ps.

[P$^+$96] Y. Papakonstantinou et al. Capabilities-based query rewriting in mediator systems. In *Proceedings of the Intl. Conference on Parallel and Distributed Information Systems*, 1996.

[R$^+$89] M. Rusinkiewicz et al. Query processing in a heterogeneous multidatabase environment. In *Proceedings of the IEEE Symposium on Parallel and Distributed Processing*, 1989.

[RDF] Resource Description Framework (RDF). http://www.w3.org/RDF/.

[SAN95] *Secrétariat d'Administration National des Données Relatives à l'Eau.* Sandre, Rue Edouard Chamberland, 87065 Limoges, France, 1995.

[TRV96] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of DISCO. In *Proceeding of the International Conference on Distributed Computing Systems (ICDCS)*, 1996.

[VP97] Vasilis Vassalos and Yannis Papakonstantinou. Describing and using query capabilities of heterogeneous sources. In *Proceedings of VLDB'97*, pages 256–265, 1997.

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

[XML] Extensible Markup Language (XML). http://www.w3.org/XML.

This article was processed using the LaTeX macro package with LLNCS style