Wrapper Generation for Semi-structured Internet Sources *

Naveen Ashish and Craig A. Knoblock
Information Sciences Institute and
Department of Computer Science
University of Southern California
4676 Admiralty Way Marina del Rey, CA 90292
{ashish,knoblock}@isi.edu
http://www.isi.edu/sims/{naveen,knoblock}

Abstract

With the current explosion of information on the World Wide Web (WWW) a wealth of information on many different subjects has become available on-line. Numerous sources contain information that can be classified as semi-structured. At present, however, the only way to access the information is by browsing individual pages. We cannot query web documents in a database-like fashion based on their underlying structure. However, we can provide database-like querying for semi-structured WWW sources by building wrappers around these sources. We present an approach for semi-automatically generating such wrappers. The key idea is to exploit the formatting information in pages from the source to hypothesize the underlying structure of a page. From this structure the system generates a wrapper that facilitates querying of a source and possibly integrating it with other sources. We demonstrate the ease with which we are able to build wrappers for a number of internet sources in different domains using our implemented wrapper generation toolkit.

1 Introduction

Numerous sources of information on the internet contain information that is semi-structured and presented in a highly formatted manner. Consider a source such as the CIA World Fact Book,[1] which provides information on the geography, economy, government, etc. of every country. A page for one country is shown in Figure 1. There are clearly identifiable sections such as Geography, Area, Land boundaries, etc. on each page. It would be useful to have the capability of issuing queries to this source which would allow us to query the source based on the structure in the pages. For instance, for each individual page we would like to be able to ask queries about one or more sections in the page, e.g. ``Find the *Land boundaries* and *Area* of France''. Or we could ask queries that span multiple pages (i.e., countries) such as ``Find the *National Product* and *Defense Expenditures* of all countries in Europe.'' We provide such a querying capability by building *wrappers* around WWW sources. A wrapper for a web source accepts queries[2] about information in the page(s) of that source, fetches relevant pages from that source, extracts the requested information and returns the results. There are two primary advantages to building wrappers around WWW sources:

- We can query these sources in a database-like fashion – thus our ability to obtain information from an individual source is enhanced.
- All sources around which we build wrappers can be queried using a common query language. We can then provide integrated access to several sources using a *mediator* that integrates information from several sources provided they can all be queried through a common language.

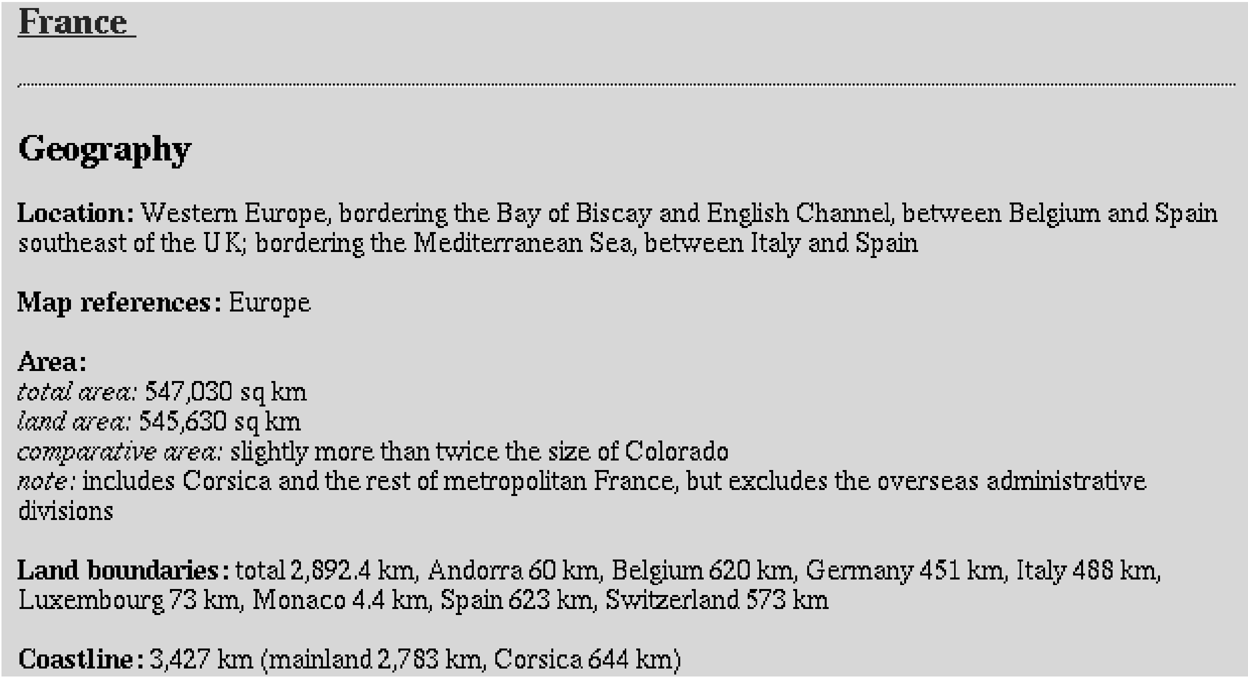
However, it is impractical to construct wrappers for Web sources by hand for a number of reasons:

- The number of information sources of interest is very large.
- Newer sources of interest are added quite frequently on the Web.
- The format of existing sources often changes.

*This work is supported in part by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-94-C-0210, by the National Science Foundation under grant number IRI-9313993, by the DARPA Fort Huachuca Contract DABT63-96-C-0066 and by the University of Southern California - Integrated Media Systems Center (IMSC), a National Science Foundation Engineering Research Center. The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, RL, NSF or any person or agency connected with them.

[1] http://www.odci.gov/cia/publications/nsolo/wfb-all.htm

[2] The queries are formulated in some query language appropriate for querying semi-structured data. For our purposes we are using the query language provided by Ariadne, the mediator we use to integrate several sources.


France

Geography

Location: Western Europe, bordering the Bay of Biscay and English Channel, between Belgium and Spain southeast of the U K; bordering the Mediterranean Sea, between Italy and Spain

Map references: Europe

Area:
total area: 547,030 sq km
land area: 545,630 sq km
comparative area: slightly more than twice the size of Colorado
note: includes Corsica and the rest of metropolitan France, but excludes the overseas administrative divisions

Land boundaries: total 2,892.4 km, Andorra 60 km, Belgium 620 km, Germany 451 km, Italy 488 km, Luxembourg 73 km, Monaco 4.4 km, Spain 623 km, Switzerland 573 km

Coastline: 3,427 km (mainland 2,783 km, Corsica 644 km)

Figure 1: Snapshot of a Page from the CIA World Fact Book

We present a semi-automatic approach to wrapper generation for WWW sources. We report on the development of an implemented wrapper generation toolkit that provides a semi-automatic, interactive wrapper generation facility. The rest of this paper is organized as follows. Section 2 describes how we semi-automatically generate wrappers. Section 3 discusses results that demonstrate the effectiveness of our techniques for wrapper generation. Section 4 describes related work. Section 5 presents future directions and conclusions.

2 Approach for Automated Wrapper Generation

This section describes our approach for generating wrappers for WWW sources. We have attempted to automate the process of building wrappers to the extent possible. The steps involved in generating a wrapper for a new WWW source are:

- *Structuring the source.* This involves identifying sections and sub-sections of interest on a page.
- *Building a parser for the source pages.* The parser is based on the structure derived in the first step.
- *Adding communication capabilities between the wrapper, mediator and web sources.* The mediator integrating several sources and the wrappers for these sources need to be able to communicate. Also, the wrappers need to communicate with Web sources to retrieve data to answer queries.

We describe these steps in detail below.

2.1 Structuring the Source

In specifying the structure of a page on the web there are two things that need to be clearly identified.

1. *Tokens of interest on a page.* By tokens we mean words or phrases that indicate the heading of a section, such as `Geography`, `Economy`, or `Total Area` on the CIA Fact Book page. A heading indicates the beginning of a new section, thus identifying headings identifies the sections in a page.
2. *The nesting hierarchy within sections.* Once a page has been decomposed into various sections, we have to identify the nesting structure of the sections. For instance a CIA fact book page is comprised of sections Geography, People, Economy, Government and Transportation. Geography in turn is comprised of the sections Area, Land boundaries, etc. while Area in turn is comprised of land area, total area etc.

The structuring task can be done automatically or with minimal user interaction. The key idea here is that a program analyses the HTML and other formatting information in a sample page from the source and makes a guess of the what are the interesting tokens on that page. Also, using the formatting information, the system makes a guess about the nesting structure of the page. The heuristics used for identifying important tokens in a page and the algorithm used to organize sections into a nested hierarchy are an important contribution of this work. We describe them in more detail below.

2.1.1 Identifying Tokens

Tokens identifying the beginning of a section are often either presented in bold font in HTML, comprised of all upper case words, or end with a colon. We can generate a lexical analyzer that searches a page for such tokens using LEX [Les75], a lexical analyzer generator. In Table 1 we list the regular expressions given as specifications to LEX to identify tokens indicating headings on a page. From these specifications we generate a lexical analyzer that identifies words or phrases conforming to the regular expressions. When structuring any page, the system is able to identify headings that are formatted in any of the ways listed in Table 1. For instance, given a page from the CIA Fact Book the system is able to identify the tokens of interest such as `Geography`, `Land boundaries`, `Area` etc. Since each token marks the beginning of a section on a page, at the end of the above tokenizing step all the different sections on a page have been identified.

2.1.2 Determining the Hierarchical Structure

The next step is to obtain the nesting hierarchy of sections on the page, i.e., what sections comprise the page at the top level, what sub-sections, if any, comprise other sections in the page etc. As with the tokenizing step, obtaining the nesting hierarchy too can be obtained in a semi-automatic fashion for a large number of pages. We have developed an algorithm that given a page with all sections and headings correctly identified, outputs a hierarchy of sections. The following two simple heuristics are used:

1. Font Size - Sub-sections in general have headings with smaller font than the heading of the section they are part of.
2. Indentation - Indentation spaces (which can be detected from raw text or HTML tags) are often used to indicate that one section is a subsection of another.

Using the procedure shown in Figure 2, the system outputs a grammar describing the nesting hierarchy of sections in a page. This procedure first builds a tree that reflects the nesting hierarchy of sections. We construct a node for each heading that identifies a new section and make this node a child of the section that should be its immediate parent based on the font size and indentation of the section headings. The children of each node are ordered i.e., they appear in the same order in which the corresponding sections appear on the page. When all nodes for all sections have been placed in the tree the procedure outputs grammar rules for each node in the tree, essentially stating that the section at each node has as sub-sections all its immediate children in the tree (and in the order in which they appear in the tree). For instance for pages from the CIA fact book the grammar output is shown in Figure 3.

It is possible that the system makes mistakes when trying to identify the structure of a new page. Based on the heuristics listed in Table 1 the system can highlight tokens erroneously i.e., identify some words or phrases to be tokens when they are not, or fail to identify phrases that are headings or tokens, but do not conform to any of the regular expressions in Table 1. We have provided a facility for the user to interactively correct the system's guesses. Through a graphical interface the user can highlight tokens that the system misses, or delete tokens that the system erroneously chooses. Also the user can correct the grammar output by the system for the structure of the page if it is incorrect.

2.2 Building a Parser for the Source Pages

The next step is to generate a parser for pages from the source. Given a page from the source, such a parser can extract any selected section(s) from the page. For instance a parser for pages from the CIA Fact Book can extract sections such as Geography.Area (the "." indicates that Area is a subsection of Geography in the spirit of complex objects) i.e., the Area sub-section within the Geography section from the page for any country. Such a parser can be automatically generated since all the grammatical and lexical information needed to parse the page is obtained in the structuring step. The compiler generator YACC [Joh78] and the tool LEX are used for this purpose. Given tokens identified in the structuring step we provide specifications to LEX to generate a lexical analyzer that identifies exactly those tokens from a page. The tool YACC can generate a parser for a language given grammar rules that specify valid sentences in the language. We directly translate the grammar describing the overall structure of the page, which is obtained at the end of the structuring step, into grammar rules as specification to YACC. A parser generated from these specifications can parse any page from the source according to the structure determined during the structuring step.

2.3 Adding Communication Capabilities

After building a parser for pages from the source we need to add the following communication function-

Regular Expression given as LEX Specification	*Description*	*Example Heading*
‘‘<’’[bB][^<>]*‘‘>’’[^\n]+‘‘<’’/[bB]‘‘>’’	Headings in bold tags	<a href=‘‘’’>Chair
‘‘<’’[hH][0-6][^<>]*‘‘>’’[^\n]+‘‘<’’/[hH][0-6]‘‘>’’	Headings with font size	Geography
‘‘<STRONG>’’[^\n]+‘‘</STRONG>’’	Headings in Strong Tags	<STRONG>Area</STRONG>
‘‘<Strong>’’[^\n]+ ‘‘</Strong>’’	Strong tags in different case	<Strong>Population</Strong>
‘‘’’[^\n]+‘‘’’	Strong tags in lower case	Deadlines
[A-Za-z0-9\-_]+[:]	Word sequence ending in colon	IRS NUMBER:
‘‘<’’[iI][^<>]*‘‘>’’[^\n]+‘‘</’’[iI]‘‘>’’	Italicized group of words	<i>total area:</i>

Table 1: Heuristics for Identifying Tokens when Structuring a Page

```
current_node= make_new_tree(); /* returns a  node that is the root of a new tree */
while(more_headings){

    new_node=construct_node(heading); /* makes a new node for the new section */
    while ((size_of(current_node) <= size_of(new_node)) or
       (indentation_of(current_node) >= indentation_of(new_node))){
                     /* search for the immediate parent section  of the new section */
           current_node=parent_of(current_node);
        }
    make_rightmost_child(current_node,new_node);
                     /* make the new section the rightmost child of its immediate parent */
    current_node=new_node;
}

generate_grammar();
/ * a procedure that from the tree constructed above, for each node N with ordered children
C1, C2 ... , Cm outputs a grammar rule of the form N --> C1 C2 ..... Cm */
```

Figure 2: Algorithm to Obtain Nesting Hierarchy

CIApage –> Geography People Government Economy Transportation

Geography –> Location Map_references Area Land_boundaries Coastline ..

Area –> total_area land_area comparative_area
...

Figure 3: Nesting Hierarchy for CIA Page

ality to the wrapper.

1. *Information about the network locations of pages from that source.* Given a query the wrapper should be able to determine the network locations (URLs) of the page(s) that need to be fetched from the source to answer the query. For instance for the CIA Fact Book there is a one to one mapping between the country name and the URL of the page for that country. This can be obtained from the index page for the CIA source. The user specifies a *mapping function* which takes necessary arguments from a query (e.g., *country name* from a query on the CIA source) and constructs a URL pointing to the page to be fetched.

2. *Capability to fetch pages over a network.* Currently we are using PERL scripts for the purpose of making HTTP connections to the information sources and retrieving data from them.

3. *Communication between the wrapper and a mediator.* We add code for KQML [FLM97], an agent communication language, to facilitate communication between the wrapper and a mediator.

3 Results

We have applied the wrapper generator to the task of generating wrappers for a variety of internet sources. We present experimental results to provide an idea of the amount of effort required to generate a wrapper for a new source. The step that is most difficult to automate when generating a wrapper is the first step where we obtain the structure of a page or sample pages from the source. Generating the parser is then done automatically and defining a mapping function from queries to URLs of relevant for sources with multiple pages requires comparatively little effort on part of the user. It is thus the structuring step that dominates the time and effort needed to build a wrapper for a new source.

We used the wrapper generator to build wrappers for several internet sources and to evaluate the effectiveness of the heuristics we use for structuring a new page automatically. To provide a quantitative measure of the effectiveness of the heuristics, we define what we call *correction steps*. During structuring a page, each time the user has to manually correct a token (i.e., add or delete a token) or correct a rule in the grammar describing the nesting hierarchy of sections, it is counted as one correction step. The total number of correction steps made before the page is completely structured provides an estimate of how hard it is to automatically structure that page. We also provide the time taken to generate the wrapper for each source. This would of course vary from user to user. Nevertheless, the results give a sense for approximately how long it might take to generate wrappers using this toolkit.

Table 2 demonstrates the ease with which we built sources for a dozen internet sites, from both the multiple instance and single instance categories.[3] We provide the number of correction steps to structure a sample page from each source as well as the total time (in minutes) taken to build a wrapper for that source. The results are extremely encouraging. Several sources require almost no or very few correction steps to structure them, thus showing that the heuristics for structuring pages are quite successful. Also it takes only a few minutes to generate a wrapper for most new sources. Such a toolkit is thus extremely useful as it provides a convenient and quick way to generate wrappers for new sources of information on the web and then integrate them via a mediator. We successfully integrated several sources in the countries information domain, such as the CIA Fact Book, Yahoo listings of countries by region etc. using the Ariadne system, which is a descendant of the SIMS [AKS96] information mediator and which addresses the problem of integrating web sources. We were then able to pose queries to Ariadne such as " Find the *External debt* and *Defense expenditures* of all countries in EEC.". The answer given by the mediator is shown in Figure 4.

4 Related work

Several researchers are investigating the topic of querying semi-structured data. The work of [Abi97] focuses on isolating the essential aspects of semi-structured data, and also surveys some proposals for models and query languages for such data. The work in [BDS96] proposes a language for querying data that has a tree like structure and also discusses optimization issues for such a language. Lorel [AQM+97] is a query language for semi-structured data. It is based on the OEM data model and has been implemented on top of the O2 object-oriented database system. Another effort on using an object-oriented database to manage SGML data is described in [VSSM94].

W3QS [KS95] is a system for SQL like querying for the WWW. The system interfaces to user programs and UNIX services for analyzing and filtering semi-structured information from WWW servers.

[3] An internet source is in the multiple instance category if it provides information in several pages, all in the same format and in the single instance category if it provides all information on a single page

Multiple instance sources	*Correction steps*	*Time in min*	*Single instance sources*	*Correction steps*	*Time in min*
1. The CIA World Fact Book.	0	2	1. CoopIS 96 Proceedings page	1	3
2. GSA On-line Shopping database.	2	5	2. AAAI-97 conference homepage	3	3
3. The NSF database.	0	5	3. List of US Universities by state	6	4
4. The OMIM Genetics database	4	4	4. List of AAAI Fellows by year	0	1
5. Hoover Company Profiles	2	4	5. Computer Science Job Listings	6	5
6. The Internet Movie Database	3	6	6. SIGMOD Record page	4	3
7. The Air Force Library Fact Sheets	0	2	7. US Air Force Organization page	0	1

Table 2: Experimental Results showing the effort and time to build wrappers for different sources

Country	**EXTERNAL_DEBT**	**DEFENSE_EXPENDITURES**
AUSTRIA	$21.5 billion (1994 est.)	exchange rate conversion – about $1.8 billion, 0.9% of GDP (1994)
BELGIUM	$31.3 billion (1992 est.)	exchange rate conversion – $3.9 billion, 1.8% of GDP (1994)
DENMARK	$40.9 billion (1994 est.)	exchange rate conversion – $2.7 billion, 1.9% of GDP (1994)
FINLAND	$30 billion (December 1993)	exchange rate conversion – $1.86 billion, about 1.9% of GDP (1994)
FRANCE	$300 billion (1993 est.)	exchange rate conversion – $47.1 billion, 3.1% of GDP (1995)
GERMANY	$NA	exchange rate conversion – $40 billion, 1.8% of GNP (1995)
GREECE	$26.9 billion (1993)	exchange rate conversion – $4.1 billion, 5.4% of GDP (1994)
IRELAND	$20 billion (1994 est.)	exchange rate conversion – $500 million, 1.3% of GDP (1994)

Figure 4: Answer to database-like query involving multiple WWW sources

The work in [MMM96] describes a query language WebSQL for querying Web sources by exploiting the structure and topology of the document networks. Most of the above efforts are concerned with issues such as the development of data models and query languages for semi-structured data, defining formal semantics for the proposed languages and implementation issues. However, the Web is the largest and ever growing source of semi-structured data. The focus of our work is on the generation of wrappers which can facilitate database-like querying of semi-structured data retrieved directly from Web servers as opposed to efforts that address the management of semi-structured data stored locally.

The work of [HBGM+97] is on wrapper generation, although the techniques discussed are for specifying wrappers for various kinds of sources such as relational databases and legacy systems besides WWW sources. A *template based* approach is used where a user provides actions that execute when a query matches a certain template or format. Wrapper generation for WWW sources is also discussed in [KWD97]. The focus of their work is very similar to ours i.e., building wrappers for WWW sources to be integrated by a software agent. However, they follow a very different approach since they are working on building an Internet shopping agent and they focus on pages that contain items for sale. As a result they make much stronger assumptions about the type of information they are looking for and use that information to hypothesize the underlying structure. Their approach cannot generate wrappers for more general types of pages, such as the CIA World Fact Book. The main contribution of our work is the development of heuristics by which the system can hypothesize the structure implicit from the formatting information in the source pages. Once the correct structure is obtained a wrapper for the source can be generated without much effort or time.

5 Future work and conclusions

We have presented the ideas and results of our approach to automatically generating wrappers for Web sources. The main contribution of our work is automating the structuring step through the use of heuristics for determining the structure by exploiting formatting information in pages from the source. Our ideas appear to be effective for many semi-structured types of sources. However we need more advanced wrappers to be able to broaden the scope of kinds of sources we can generate wrappers for and also to be able to be able to handle finer grained queries. Currently we are working on enhancing the wrappers with the following capabilities:

- Learning new tokens by examples - It is possible that while structuring a page, the system is unable to identify tokens on the page if the lexical analyzer for identifying tokens cannot identify tokens in a new and unexpected format. We are working on adding capabilities to the system to quickly *learn* the structure of a new kind of token from a few user examples. We are applying techniques for inducing Hidden Markov models (HMMs) describing the new kind of token from corpora of positive examples. The system can then identify the remaining tokens in the page automatically.

- Handling Tables - We are working on automatically building parsers to extract information in a tabular form.

- Handling finer grained queries - Consider the `Land boundaries` section on a CIA fact book page. Currently the wrapper cannot handle queries such as `''Find the names of all countries bordering France"` as the parser does not have enough knowledge of the structure *within* the Land boundaries field to decompose that field into pairs of countries and corresponding border lengths. We are currently investigating using machine learning techniques where the user gives a few examples highlighting items of interest within a field from which the system is able to learn the structure within that field.

Acknowledgements

We would like to thank Steve Minton and the other members of the SIMS and Ariadne projects for their contributions to this work.

References

[Abi97] Serge Abiteboul. Querying semi-structured data. In *ICDT (invited talk)*, 1997.

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

[AQM+97] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet Weiner. The Lorel query language for semistructured data. *Journal on Digital Libraries*, 1997.

[BDS96] Peter Buneman, Susan Davidson, and Gerd Hillebrand Dan Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996.

[FLM97] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. AAAI/MIT Press, Menlo Park, CA, 1997.

[HBGM+97] Joachim Hammer, Marcus Brennig, Hector Garcia-Molina, Svetlozar Nesterov, Vasilis Vassalos, and Ramana Yerneni. Template-based wrappers in the tsimmis system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (Demonstration Track)*, Tucson, AZ, 1997.

[Joh78] Stephen C. Johnson. Yacc: Yet another compiler compiler. Technical Report CSTR 32, AT&T Bell Laboratories, 1978.

[KS95] D. Konopnicki and O. Shemueli. W3QS: A query system for the World Wide Web. In *Proceedings of the 21st International Conference on Very Large Databases*, Zurich, Switzerland, 1995.

[KWD97] Nicholas Kushmerick, Daniel S. Weld, and Robert Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, Japan, 1997.

[Les75] M. E. Lesk. Lex - a lexical analyzer generator. Technical Report CSTR 39, AT&T Bell Laboratories, Murray Hill, New Jersey, 1975.

[MMM96] Albero O. Mendelzon, George A. Mihaila, and Tova Milo. Querying the world wide web. In *Symposium on Parallel and Distributed Information Systems*, Miami, Florida, 1996.

[VSSM94] V.Christophides, S.Abiteboul, S.Cluet, and M.Scholl. From structured documents to novel query facilities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, 1994.