Presented at International Workshop on Efficient Web-based Information Systems (EWIS-2002), Montpellier, France, September 2nd, 2002.

Object-Oriented Mediator Queries to Internet Search Engines

Timour Katchaounov, Tore Risch, Simon Zürcher

Uppsala Database Laboratory, Department of Information Technology,
Uppsala University, Sweden

Abstract. A system is described where multiple Internet search engines (ISEs), e.g. Alta Vista or Google, are accessed from an Object-Relational mediator database system. The system makes it possible to express object-oriented (OO) queries to different ISEs in terms of a high level OO schema, the *ISE schema*. The OO ISE schema combined with the mediator database system provides a natural and extensible mechanism in which to express queries and OO views that combine data from several ISEs with data from other data sources (e.g. relational databases). High-level OO web queries are translated through query rewrite rules to specific search expressions sent to one or several wrapped ISEs. A generic ISE query function sends the translated queries to a wrapped ISE. The result of an ISE query is delivered as a stream of semantically enriched objects in terms of the ISE schema. The system leverages publicly available *wrapper toolkits* that facilitate extraction of structured data from web sources, and it is independent of the actual wrapper toolkit used. One such wrapper toolkit was used for generating HTML wrappers for a few well-known ISEs.

1. Introduction

To facilitate the combined access to data on the web with data from other databases, a system called ORWISE (Object-Relational Wrapper of Internet Search Engines) has been developed that can process queries combining data from different Internet search engines (ISEs) with data from regular databases and other data sources. The design of ORWISE leverages available *wrapper toolkits* to extract information from web pages. ORWISE has been implemented for three well-known search engines using a publicly available wrapper toolkit [31].

ORWISE is an extension to the database system Amos II [29], [30], that is based on the *wrapper-mediator approach* [34] for heterogeneous data integration. The core of Amos II is an extensible object-relational database engine having mediation primitives in a query language *AmosQL* similar to the OO parts of SQL-99 and

ORWISE thus permits SQL-99 like queries that combine ISE results with data from other types of sources such as relational databases [10] and XML [23]. Amos II is suitable for collecting and processing results from ISEs because its purpose is to act as a fast mediator database which can manage meta-data of heterogeneous and distributed data sources and efficiently process queries to the sources.

The generalized *ISE wrapper manager* ORWISE, described in this paper, makes it possible to easily access one or several ISEs from Amos II using different *ISE wrappers* for each engine. Combined with OO mediation facilities [4], [17], it allows to process OO database queries that combine data from several ISEs with data from conventional databases and other data sources. In difference to relational systems for web queries [14], the data produced by ORWISE is not just text strings but much more semantically rich object structures in terms of an OO schema for ORWISE, called the *ISE schema* (Internet search engine schema). The ISE schema describes capabilities and other properties of the search engines along with the structure of their results.

ISEs have some special problems compared to 'conventional' databases:

- *Semi-structured interfaces*: There are no standard interfaces to ISEs such as ODBC and JDBC. Web forms are used for specifying queries and other inputs to them. The result of an ISE query is a semi-structured web document containing not just the query result but also auxiliary text, banners, etc., which need to be filtered out from the query result.
- *Query languages*: ISEs do not have a standardized query language such as SQL but every ISE has its own query language with varying syntax and semantics.
- *Autonomy*: The content, structure and availability are totally controlled by the information supplier.
- *Evolution*: Internet sites tend to change very often. A system that accesses a site has to be very flexible.
- *Heterogeneity*: The data delivered by ISEs have varying structures and the system has to reconcile semantic differences.

In order to handle the above problems we need reliable and flexible interfaces to the ISEs, here termed *ISE wrappers,* which can programmatically fill and submit web forms and parse the structure of an ISE result document searching for predefined patterns. An ISE wrapper must be flexible enough to cope with small changes in the web sites.

To specify web source wrappers ORWISE utilizes wrapper toolkits to extract useful information from web pages. ORWISE is designed to be independent of the actual wrapper toolkits used. We investigated several of them to make sure that the system works with all of them. For our first implementation we chose W4F [31] to generate ISE wrappers for three search engines - Google (http://www.google.com), AltaVista (http://www.altavista.com), and Cora Research Paper Search (Cora) (http://cora.whizbang.com).

The ISE wrappers are connected to the system through a generic query language function called *orwise*, which is a foreign function (implemented in Java) overloaded for each search engine. It returns objects of an ISE specific type[1] that describes the

[1] We use the terms 'type' and 'class' as synonyms.

retrieved query results. New ISE wrappers can dynamically be added to the system by creating a new subtype of the system type *SearchEngine* for each new ISE and then implementing some code (in Java) to interface its ISE wrapper. The overloading of the function *orwise* is used for facilitating the plug-in of new ISE wrappers.

Once a new search engine is connected to the *orwise* function it can be used in OO queries. Since the parameters for each implementation of *orwise* are search engine specific, such queries will be rather detailed with search engine specific parameters for, e.g., query strings, site names, etc. The system therefore provides high-level query functions that can be used for any ISE and where queries are specified uniformly. For example, the function *webSearch* is defined for every search engine to specify OO queries to it in a search engine independent form. The high-level OO query expressions need to be transformed before the actual call(s) to *orwise* is issued. The approach in Amos II is to implement a *translator* module for each kind of data source (search engine, relational database, etc.). In the case of ISEs, the translators rewrite the high-level query into search engine query specifications containing calls to *orwise*. Since different search engines have different ways of specifying searches, they have different rewrite rules.

In summary, ORWISE provides i) the ISE schema for describing and querying data from any ISE, ii) a mechanism to specify search engine specific translators, and iii) facilities to allow different wrapper toolkits to be easily plugged into the system.

2. Related work

Many projects (e.g. [11], [16], [21], [27], [33]) use the mediator approach to data integration in general. The work presented here describes how an object-relational mediation framework [29] leverages upon an available wrapper toolkit to provide access to ISEs.

The use of object-relational approach in querying the structure of XML Web documents has been done, e.g., in [3], [8], [12], [23]. A query language standard for XML, XQuery [35], is being developed with which the contents of XML documents can be queried and new XML documents constructed. All major ISEs use HTML, not XML. General Web query languages for HTML are proposed in [19], [25]. These are general languages for querying well-formed Web documents and not directly suitable for defining embedded interfaces to ISEs.

By contrast *wrapper toolkits* [9], [13], [15], [18], [20], [22], [24], [31] specify programmatic interfaces to web sources handling both sending commands and extracting structured data from responses. They often include some advanced pattern matching language to extract data from Web documents as regular expressions operating on varying levels of granularity. With a wrapper toolkit a web source wrapper is defined by processing *wrapper specifications*, consisting of statements to connect to web sources and to detect the parts of the text to be extracted. They allow new wrappers to be specified much easier than with manual programming and the developers need not master a complex query language. A good overview of projects related to wrapper construction for Web sources can be found in [31].

A wrapper toolkit can be a *wrapper-generator* that generates code (e.g. Java) implementing a web source wrapper [1], [2], [24], [31]. It can also be a *wrapper-interpreter* where the web source wrapper is specified as commands, which are interpreted at run time [18], [15]. ORWISE is designed to work with both wrapper-generators and wrapper-interpreters. Web source wrappers represent data differently and are not sufficient themselves to combine data from Web sources and conventional databases. Therefore there is need for data mediation facilities along the lines of this paper.

In [26] it is shown that an OO query language indeed is very useful for specifying queries to text engines. Our work differs in that we propose leveraging upon using external wrapper toolkits, OO query rewrites, and the ISE schema. Furthermore, we explicitly model the capabilities of the search engines in the ISE schema, rather than in the internals of the system. The WSQ/DSQ [14] project proposes an architecture where web searches are specified as SQL queries to two virtual relational tables. Their relational tables are inflexible for the purpose, compared to our ISE schema. The focus of the work in [5] is re-write rules and cost models for integrating text search with other queries. Those rewrite rules are applicable in our translator too.

To the best of our knowledge, no other project proposes a system that uses inheritance and overloading to model ISEs and their results on the conceptual level, while at the same time the implementation is independent of, and leverages existing wrapper toolkits. Another major difference to other projects is that our object-oriented ISE schema distinguishes between the search engine specific descriptions of documents and the actual documents. Furthermore, the ISE capabilities are modeled in the ISE schema too.

3. Scenario

We have implemented the scenario of Figure 1 to illustrate the functionality of the system. In the scenario, an Amos II mediator is used to process queries that combine data from a relational DB2 database through ODBC with three ISEs, AltaVista, Google and Cora. The access to the three Internet search engines uses the ORWISE wrapper, while the relational database is accessed through an ODBC-wrapper.

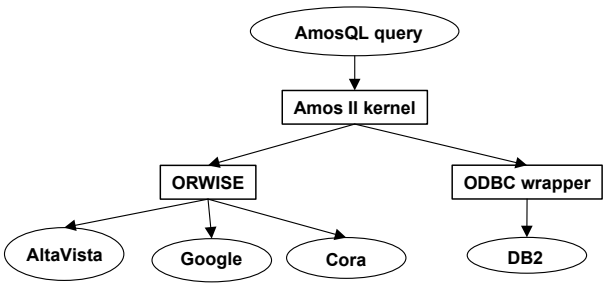

Fig. 1. Mediator scenario.

The relational database stores a table of employees that is mapped to the mediator type *Employee*, using the techniques for defining OO views of relational databases [10]. The following AmosQL query uses Google to find the names of those employees who are mentioned in some web page in the web site `'www.csd.uu.se'`:

```
SELECT DISTINCT given_name(e), family_name(e)
FROM Employee e, DocumentView d, Google ise
WHERE d = webSearch(ise, given_name(e)) AND
      d = webSearch(ise, family_name(e)) AND
      host(url(d))=' www.csd.uu.se');
```

The first two lines of the 'where' clause in the query retrieve the documents that contain given names and family names of employees in the relational database, while the last line restricts the search to only those persons whose names are found by Google in web pages on the host `'www.csd.uu.se'`. Other text-related predicates such as `'near'` can also be added to refine the search. The type *DocumentView* represents descriptions of documents returned by an ISE and the type *Google* represents the wrapper for Google. The same query can be specified for Alta Vista by replacing the type *Google* with *AltaVista*. It is also possible to specify queries over several search engines by using the generic supertype *SearchEngine* instead of *AltaVista* or *Google*.

4. The ISE Schema

Queries to ISEs are posed in terms of the OO database schema on Figure 2. Inheritance and overloading are used to model heterogeneity of both ISEs and their results. Furthermore, we separate the description of results returned by ISEs from the documents themselves. Since Amos II has a functional data model [32], both type attributes and relationships between types are modeled by functions shown as think lines on Figure 2. For clarity, the overloaded function *orwise* is represented as an attribute of the subtypes of type *SearchEngine*. The core of the ISE Schema consists of three base types:

- *SearchEngine* – this type is used to categorize ISEs. It reflects the fact that search engines have different query capabilities and parameters. It has a subtype for each specific ISE normally with only one instance. The generic function *orwise* is overridden for each ISE to reflect their different semantics. Analogously each of them has a specific query rewrite function.
- *DocumentView* – objects of this type describe the results of a query to different ISEs. By introducing this type of objects we can distinguish between the documents themselves and the description of a document by an ISE. Document views often contain information about a document that is not part of the document itself and is imprecise or outdated. They may use different formats from the document itself; e.g. the Cora ISE returns HTML descriptors of PostScript

documents. Differentiating between documents and views over documents allows for more precise queries.

- *Document* – describes document objects themselves. Subtypes of *Document* may describe document objects with different structure. The problem of querying structured documents is outside the scope of this work and has been addressed by other researchers [6], [28]. All this work can be easily reused in our system due to the flexibility of our OO data model.

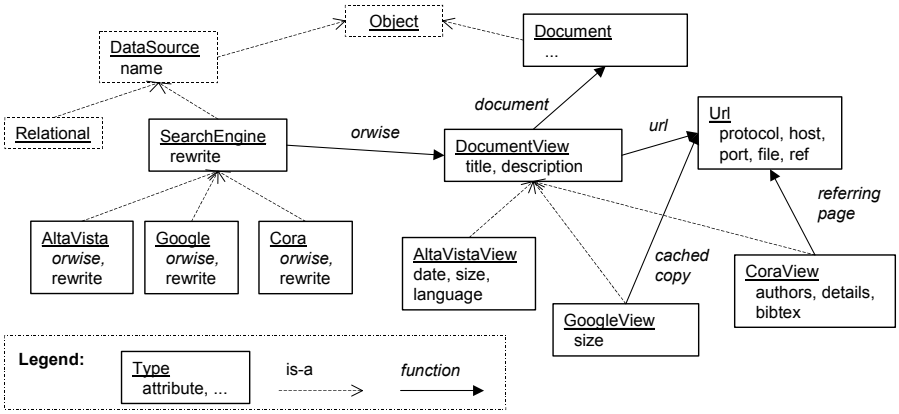

Fig. 2. The ISE schema

The two classes *DataSource* and *Relational* are part of the general Amos II meta-type hierarchy. The type *DataSource* serves as the base type of all meta-types for different kinds of data sources accessible through the mediator system. One such meta-type is *Relational*, which describes relational data sources. It has the function *sql*, analogous to the *orwise* function of *SearchEngine*. In our current implementation, the type *SearchEngine* has three subtypes for each of the wrapped search engines AltaVista, Google and Cora. Each of them defines its own version of the *orwise* function and specific rewrite rules. Correspondingly the type *DocumentView* has three subtypes: *AltaVistaView, GoogleView and CoraView*, where each of them has additional properties. For example, of the three ISEs only AltaVista returns the language of a document, while only Google may provide a locally cached copy of it's indexed documents, accessible through the function *cached_copy*. Finally, *Document* objects may be accessed and queried further through the *document* function of the type *DocumentView*. The type *Url* is an example of semantic enrichment of the ISE query results, as they return URLs as strings.

5. The ORWISE Architecture

Figure 3 shows the layered architecture of the system. The left part shows how ORWISE is interfaced with the Amos II kernel, while the right part shows the layers of ORWISE itself.
The architecture is designed to fulfill several requirements:

- It provides a uniform interface from the Amos II query processor to any ISE.
- It can use any existing general wrapper toolkits.
- It is independent of the wrapper toolkits used.
- It is possible to easily add a new ISE wrapper without any changes to the rest of the system.
- There is no need to modify the definitions of wrappers generated by wrapper-generators.

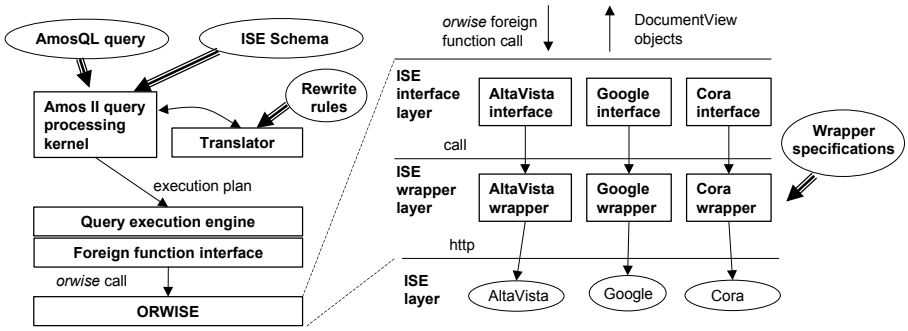

Fig. 3. System architecture.

The two layers *ISE interface* and *ISE wrapper* fulfill these requirements. This architecture permits any wrapper toolkit to be used and different kinds of wrapper toolkits can even be combined.

The ISE interface layer defines an interface between the Amos II kernel and the underlying ISE wrapper layer used for interfacing each search engine. The functionality common for every ISE wrapper, such as instantiating ISE specific *DocumentView* objects and emitting the result stream, is encapsulated in this layer. It is called by the query processor and it calls the ISE wrapper for the chosen search engine. The basic foreign function interface of Amos II allows new ISE interfaces to be dynamically added to a running system. The ISE wrappers are specified by some external wrapper toolkit(s) chosen for each particular search engine. Therefore, the functionality they expose can be very different and cannot be directly used by ORWISE. The ISE interface therefore must instantiate objects, convert strings to URL objects or numbers, etc.

The ISE wrapper layer consists of the modules specified through the wrapper toolkit. It forms and sends HTTP requests to an ISE server and then extracts the results from the so received HTML page. The input to a wrapper toolkit is a

specification of request submission and data extraction rules for a web source. The chosen W4F [31] toolkit is a wrapper-generator, which generates Java classes per wrapped data source. In this case the layer consists of the generated code. For wrapper-interpreters the interpreter together with the specifications is the layer.

With this layered architecture, the following steps are needed to add a new search engine to ORWISE:

1. Design an ISE wrapper for the specific search engine by using a chosen wrapper toolkit. For example in the case of W4F this involves specifying the extraction rules in terms of the HEL extraction language from which a Java class is generated per each wrapped web source. By contrast, wrapper-interpreters are directly called from the ISE interface layer using the wrapper specifications as parameters.
2. Create types in the mediator database as subtypes of *SearchEngine* and *DocumentView*.
3. Design an ISE interface module as the overloaded Amos II foreign function, *orwise*, calling the ISE wrapper module from step 1.

Once step 1-3 are completed the ISE is already queryable directly through *orwise*. However, the queries can be complex and very ISE dependent. Efficient and transparent queries to an ISE therefore requires an additional step:

4. Design the rewrite rules needed for the ISE to translate between, e.g., *webSearch* calls and the particular *orwise* calls.

6. Translating ORWISE Queries

Queries calling the *webSearch* function combined with other Web document related predicates are translated to an equivalent but more efficient query containing optimized calls to the function *orwise* overloaded for specific ISEs. The function *webSearch* could be defined as a query calling *orwise* without any translation. However such untranslated execution may be significantly less efficient. In our example, the Google query is translated to the following *orwise* query:

```
SELECT given_name(e), family_name(e)
FROM Employee e, DocumentView d, Google gse
WHERE d = orwise(gse, given_name(e) + ' ' +
family_name(e), 20, 'www.csd.uu.se', 'english');
```

where the signature of *orwise* is Google specific. Here *orwise* for Google takes the parameters *query*, *result size*, *language restriction*, and *host*. The function is defined as a foreign AmosQL function that calls the underlying ISE wrapper for Google. The example illustrates the semantic rewrite of the original query by the translator, where several calls to *webSearch* and *host* are combined into one call to Google's *orwise*. The translator also added the default specifications of '*english'* as language and that only the first 20 results should be returned. The result of *orwise* is a stream of *GoogleView* objects. The translator for each ISE knows how to generate optimized *orwise* calls with specific parameters expressing ISE supported capabilities.

As shown in the example, queries to a search engine will contain subqueries expressed using the specific query language of the ISE, which is usually different for different ISEs. In the example above the string "`given_name(e) + ' ' + family_name(e)`" is an example of the construction of a conjunctive query to Google (it uses AND by default). During query translation, there are possible query transformations that can dramatically improve performance and result quality. We have implemented some translator rules to show the usefulness of the system and can utilize other results in related areas [5], [6].

7. Summary

A flexible system for querying Internet search engines through an OO mediator database system was presented. The system has the following unique combination of features:

1. Data about both the search engine capabilities and the results they return were modeled in an OO *ISE schema* in a mediator database.
2. The ISE schema permits transparent queries to ISEs with different capabilities and result structures. The mediation facilities provide for processing heterogeneous queries that combine data from ISEs with data from other data sources.
3. New kinds of ISEs can be easily plugged in. The system assumes the ISEs are autonomous and outside the control of the query processor.
4. The system is designed to be independent of the wrapper toolkits used for specifying the ISE wrappers. Several such publicly available toolkits were evaluated to choose one for the implementation.
5. The query processor provides a mechanism to plug in OO search engine specific rewrite rules for translating OO queries into the parameterized *orwise* calls. The system is independent of the actual rewrite rules to utilize previous work in this area.

References

1. B. Adelberg: NoDoSe – A Tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents, *SIGMOD 1998 Conference*: 283:294, 1998.
2. N. Ashish, C. Knoblock: Semi-automatic Wrapper Generation for Internet Information Sources. *CoopIS'97 Conference:* 160-169, 1997.
3. G. Arocena, A. Mendelzon: WebOQL: Restructuring Documents, Databases, and Webs. *In Proc. ICDE'98*, Orlando, 1998.
4. O. Bukhres, A. Elmagarmid (eds.): *Object-Oriented Multidatabase Systems.* Prentice Hall, 1996.
5. V. Christophides, S. Abiteboul, S. Cluet, M Scholl: From Structured Documents to Novel Query Facilities. *SIGMOD 1994 Conference*: 313-324, 1994.
6. S. Chaudhuri, U. Dayal, T. W. Yan: Join Queries with External Text Sources: Execution and Optimization Techniques. *SIGMOD 1995 Conference*: 410-422, 1995.
7. C. Chang, H. Garcia-Molina, A. Paepcke: Predicate rewriting for translating Boolean queries in a heterogeneous information system. *ACM Trans. on Information Systems*, 17(1), 1999.

8. Donald D. Chamberlin, Jonathan Robie, Daniela Florescu: Quilt: An XML Query Language for Heterogeneous Data Sources. *WebDB'2000*: 53-62, 2000.
9. A. Firat, S. Madnick, M. Siegel: The Caméléon Web Wrapper Engine, *First Workshop on Technologies for E-Services*, Cairo, 2000.
10. G. Fahl, T. Risch: Query Processing over Object Views of Relational Data*, The VLDB Journal* , 6(4), 261-281, 1997.
11. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom: The TSIMMIS Approach to Mediation: Data Models and Languages. *Intelligent Information Systems (JIIS),* Kluwer, 8(2), 117-132, 1997
12. R. Goldman, J. McHugh, J. Widom: From Semisturctured Data to XML: Migrating the Lore Data Model and Query Language, *WebDB'99*, 1999.
13. J. Gruser, L. Raschid, M. Vidal, L. Bright: Wrapper Generation for Web Accessible Data Sources. *CoopIS'98 Conference*: 14-23, 1998
14. R. Goldman, J. Widom: WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. *SIGMOD 2000 Conference*: 285-296, 2000.
15. G. Huck, P. Fankhauser, K. Aberer, Erich J. Neuhold: Jedi: Extracting and Synthesizing Information from the Web. *CoopIS'98 Conference*: 32-43, 1998.
16. L. Haas, D. Kossmann, E. L. Wimmers, J. Yang: Optimizing Queries across Diverse Data Sources. *23rd Intl. Conf. on Very Large Databases (VLDB'97),* 276-285, 1997
17. V. Josifovski, T. Risch: Integrating Heterogeneous Overlapping Databases through Object-Oriented Transformations, *25th Conference on Very Large Databases (VLDB'99),* 435-446, 1999.
18. T. Kistlera, H. Marais: WebL: a programming language for the Web. In *WWW7*, Brisbane, Australia, *http://www.research.digital.com/SRC/WebL/,* 1998.
19. D. Konopnicki, O. Shmueli. W3QS: A query system for the World Wide Web. *21st Conference on Very Large Databases (VLDB'95)*, 54–65, Zurich, Switzerland, 1995.
20. N. Kushmerick, D. Weld, R. Doorenbos: Wrapper Induction for Information Extraction. *IJCAI'97* Vol. 1: 729-737**,** 1997**.**
21. L. Liu, C. Pu: An Adaptive Object-Oriented Approach to Integration and Access of Heterogeneous Information Sources. *Distributed and Parallel Databases,* Kluwer, 5(2), 167-205, 1997.
22. L. Liu, C. Pu, W. Han: XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. *ICDE 2000:* 611-621, 2000.
23. H. Lin, T. Risch, T. Katchanounov: Adaptive data mediation over XML data. To be published in special issue on "Web Information Systems Applications" of *Journal of Applied System Studies (JASS)*, Cambridge International Science Publishing, 2001.
24. G. Mecca, P. Merialdo, P. Atzeni: ARANEUS in the Era of XML. *IEEE Data Engineering Bulletin*, Special Issue on XML, September, 1999.
25. A. O. Mendelzon, G. Mihaila, T. Milo: Querying the World Wide Web*. International Journal on Digital Libraries*, 1(1), 54-67, April 1997.
26. A. Paepcke: An Object-Oriented View Onto Public, Heterogeneous Text Databases*. Proceedings of the Ninth International Conference on Data Engineering (ICDE'93),* 1993.
27. D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, J. Widom: Querying Semistructured Heterogeneous Information. In *Deductive and Object-Oriented Databases, Proceedings of the DOOD'95 conference*, 1995, LNCS Vol. 1013, 319-344, Springer 1995.
28. D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, J. Widom: Querying Semistructured Heterogeneous Information. In *Deductive and Object-Oriented Databases, Proceedings of the DOOD'95 conference*, 1995, LNCS Vol. 1013, 319-344, Springer 1995.
29. T. Risch, V. Josifovski: Distributed Data Integration by Object-Oriented Mediator Servers, To be published in *Concurrency – Practice and Experience J.,* John Wiley & Sons, *http://www.csd.uu.se/~udbl/publ/concur00.pdf*, 2001.

30. T. Risch, V. Josifovski, T. Katchaounov: *Amos II Concepts*, *http://www.csd.uu.se/~udbl/amos/doc/amos_concepts.html*, 2000.
31. A. Sahuguet, F. Azavant: Building Intelligent Web Applications Using Lightweight Wrappers, *Data and Knowledge Engineering*, 36(3), 283-316, March, 2001.
32. D. W. Shipman: The Functional Data Model and the Data Language DAPLEX, *TODS,* 6(1), 140-173, 1981.
33. A. Tomasic, L. Raschid, P. Valduriez: Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE Transactions on Knowledge and Date Engineering*, 10(5), 808-823, 1998
34. G. Wiederhold: Mediators in the architecture of future information systems, *IEEE Computer,* 25(3), 38–49, 1992.
35. XQuery: A Query Language for XML, W3C Working Draft, 15 February 2001, *http://www.w3.org/TR/xquery/*.