Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Electronic Notes in Theoretical Computer Science 150 (2006) 55–70

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Wrapping of Web Sources with restricted Query Interfaces by Query Tunneling

Thomas Kabisch[1] and Mattis Neiling[2]

Computation and Information Structures (CIS)
University of Technology Berlin
Germany

Abstract

Information sources in the World Wide Web usually offer two different schemes to their users, an *Interface Schema* which the user can query and a *Result Schema* which the user can browse. Often the Interface Schema is more restricted than the Result Schema, moreover many sources offer keyword-search interfaces only. Thus query capabilities of such sources are very small and a useful integration into a mediator-based information system using query capabilities is almost impossible. We propose the *Query Tunnelling* architecture for the wrapping of these restricted web sources. Wrapping of sources by Query Tunneling hides restrictive query interfaces and makes such sources fully queryable based on their result schema. The process of Query Tunneling is divided into two main steps, *Query Relaxation* to make a higher order query suitable to a restricted interface and *Result Restriction* in order to filter the results using the original query.

Keywords: Information Extraction, Information Integration, Information Filtering, Query Capabilities, Schema Mapping, Web Query Interface, Wrapping.

1 Introduction

In the context of federated information systems wrapper are used to encapsulate heterogeneous information sources [6]. The specific design of the certain wrappers is highly heterogeneous, it depends on the underlying sources, the desired query capabilities and the used query languages. In this context a wrapper provides a query interface for an information source in a predefined

[1] Email: tkabisch@cs.tu-berlin.de
[2] Email: mneiling@cs.tu-berlin.de

query language. Thus a wrapper needs to bridge various heterogeneities of the underlying sources. Web sources that are accessible through small query interfaces shall be wrapped in order to support higher level query capabilities. This aspect is the main focus of our contribution. Before presenting the approach we will discuss some types of information sources and give a short overview to RDF [23] and RDQL [3] which we use as infrastructure and interface to support complex queries.[3]

Classification of Information sources

Information sources may be categorized along many criteria. Here we will discuss two of them, the degree of source structuring and the degree of accessibility of sources.

Structuring The source-specific wrapper functionality depends on the kind of source which should be wrapped. Sources can be classified into structured, semi structured and unstructured ones.

- Structured sources (e.g. SQL databases) can be queried by a higher query language and provide a schema. The wrapper has to *transform queries and results between the mediator language and the language of the specific source.*
- Semi-structured sources (e.g. XML sources) do not necessarily provide a schema but have a less constrained structure. Higher level query languages (e.g. XQuery [24]) are under way and could be employed. So a wrapper also has to transform queries and results between languages.
- Unstructured sources (e.g. HTML pages) are more difficult to deal with. These sources have no schema and usually provide no higher level query language. They are designed for human interaction.

Accessibility In contrast to classical database information sources (e.g. relational databases with SQL-Query Interface), web databases generally allow a restricted access to the underlying data sources only. The bottlenecks in terms of accessibility of such sources are their query interfaces. Web databases with HTML-Frontend usually offer a form-based query interface. Such a query interface is restricted to some queryable attributes which may be requested by typing keywords and combine them by usually only one operator. Another restriction is the general lack of a typing system, each parameter is decoded into a string when submitting a query.

[3] RDF abbreviates Resource Description Framework,
while RDQL stands for Resource Description Query Language.

RDF/RDQL as Infrastructure

We use RDF [23] and its query language RDQL [3] as infrastructure. RDF is a standard for the description of resources. Thus our wrapper is queryable in RDQL and delivers RDF results.

RDF as common data model The RDF format and the related RDF schema are well suited for semi-structured web data and its describing metadata.

Query languages and RDQL Several proposals for RDF query languages exist. We decided to use RDQL because it is very readable — the notation is quite similar to SQL.

Designing a mediator-based information system one should consider that the chosen data model has effects on the domain-specific query capabilities and mapping rules as well as on the query executor and result integrator of the mediator component. In distributed information systems the semantics of the query language and the query capabilities should be precisely defined, especially the logic between selection and projection attributes. Using the *Jena API* [14] a tuple is only part of the RDF result of a RDQL query if it contains all of the selected attributes. In federated information systems this is not appropriate: The user usually wants *any* information that he could get, even if some of the selected attributes are missing.

Introducing examples

Web sources are mostly unstructured (e.g. HTML) and usually can be requested through a small query interface. Nevertheless, the generated HTML pages of queryable Web sources often carry some regular structure, that can be utilized. As an example we will discuss the scientific publication source "CiteSeer", cf. [19]. This data source supports keyword-queries only.

Fig. 1. CiteSeer interface

CiteSeer offers an interface schema $I_{CiteSeer}$, that allows keyword retrieval only: $I_{CiteSeer} = (keyword)$. The result schema $R_{CiteSeer}$ is more sophisticated, we will discuss the overview page here, whose schema may be denoted as $R_{CiteSeer} = (title, author, year, link, citations)$. Thus the only query capability of this source is $keyword \rightarrow (title, author, year, link, citations)$. More

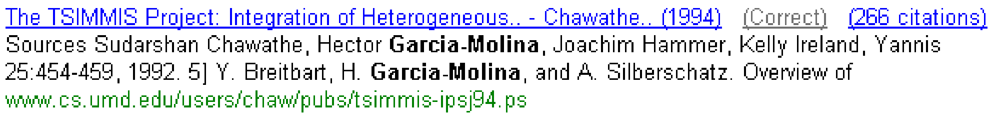
The TSIMMIS Project: Integration of Heterogeneous.. - Chawathe.. (1994) (Correct) (266 citations)
Sources Sudarshan Chawathe, Hector **Garcia-Molina**, Joachim Hammer, Kelly Ireland, Yannis
25:454-459, 1992. 5] Y. Breitbart, H. **Garcia-Molina**, and A. Silberschatz. Overview of
www.cs.umd.edu/users/chaw/pubs/tsimmis-ipsj94.ps

Fig. 2. CiteSeer result snippet

complex queries are generally not supported. Based on four different query examples we will discuss our approach.

Example 1.1 [Simple Query] `Return all papers of the Author "Garcia-Molina"`. This is an example for a simple query which cannot be issued against the interface directly. The attribute `author` is not a valid query attribute, but the value could be a selection criteria for a keyword query.

Example 1.2 [Non queryable Attributes] `Return all papers of the Author "Garcia-Molina" which have more than 100 Citations".`
This query is challenging because the number of *citations* is an element of the result schema but is not queryable — the attribute is not indexed in the underlying data source.

Example 1.3 [Range Query] `Return all papers of the Author "Garcia-Molina" which have been written between 1998 and 2004.`
Most sources support exact matches only — thus a range query cannot be issued. Even if this given query might be rewritten to certain exact queries, this is not a suitable way in general.

A complex query containing selection criteria for queryable attributes combined with Boolean operators (e.g. `AND`, `OR`, and brackets `(...)`) can be issued directly, if the query interface supports the respective operators. Otherwise, if the the source is not capable for such complex queries, an adequate handling is needed in order to split the query into its atoms. If the results of the different queries are received, an integration will be made.

Example 1.4 [Complex Query] `Return all publications written by "Garcia-Molina" or where the title contains "federation".`
If the source does not support disjunctive (`OR`) queries, the query has to be split into two queries:

- `Return all publications written by "Garcia-Molina"` and
- `Return all publications where the title contains "federation".`

Then the union of the results forms a superset of the original query results.

The rest of this paper is organized as follows. Section 2 introduces our general wrapping architecture, which is the basis for Query Tunneling. The 3rd and 4th section describes *Query Relaxation* and *Result Restriction* in detail, section 5 gives an overview about related work, section 6 describes briefly our *MiWeb* prototype system [5], which implements the Query Tunneling approach. Finally section 7 gives a conclusion and an outlook.

2 General Web Wrapping Architecture

Our general understanding of wrapping is that the wrapper offers a query interface, which supports a higher level query language and delivers structured results. At the time the architecture targets conjunctive queries only. The proposed general wrapper architecture is structured along different transformation tasks which need to be offered by a web wrapper in order to support an higer level query interface. Figure 3 draws a general picture, outlinging all components which a web wrapper may include. This section gives a short overview of the whole wrapping architecture and summarizes all tasks and transformations, which need to be supported. The main focus of this paper are the *Query Relaxation* and *Result Restriction* components. All other components are described in brief only.

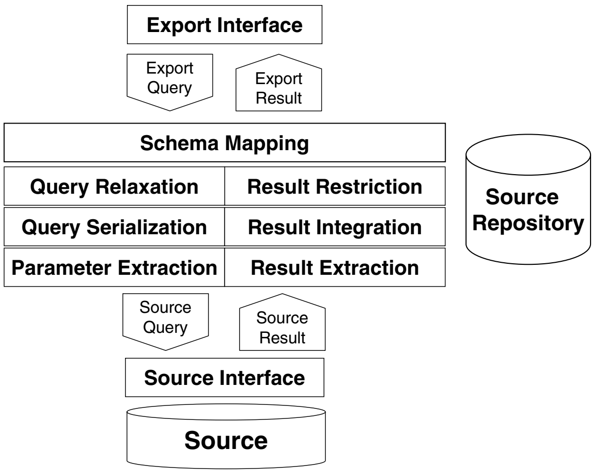

Fig. 3. General Wrapping Architecture for Web Sources

Components

Our general architecture supports a distributed handling of different kinds of heterogeneity — thus the architecture is segmented w.r.t this tasks, and each component is responsible for one specific transformation task.

Query Relaxation/Result Restriction Web-Pages often only offer a small query interface, which allows a very small set of queryable attributes, while the underlying source schema is of higher complexity. Thus a complex query needs to be relaxed in order to cope with the restricted interface. Query relaxation supports two main use cases:

- Attribute Substitution
- Attribute Elimination

In most cases query relaxation leads to a superset of the desired result set. Thus, in a second step, after the relaxed query has issued to the source, *Result Restriction* has to be applied. After the result set have been transformed back to an RDF representation the original query is issued again against the result set. This step reduces the result set to all valid elements w.r.t. this query.

Schema Mapping In a general wrapping architecture this component tackles schema heterogeneity. Mapping rules (mostly on attribute level) are specified in order to bridge this heterogeneity. In our approach this task is performed by an extra component which is not discussed here, but in [5].

Parameter/Result Extraction This component reduces complex query statements to a list of query parameters which need to be piped to the source. On the way back this component deals with extraction, which occurs if the source output is not database-like structured. The output of the result extraction step conforms to the so-called *result schema of a source* and is well structured (e.g. employing RDF or XML). The most web-sources are HTML-based and thus offer no structured output. Result extraction is focused by numerous related works, e.g. [10], [8], [7] or [25]. [16] gives a good overview to this issue. We follow a grammar-based paradigm [15]. At the time the grammar has been developed by hand, but in future we plan to adopt an automated solution, similar to [10],[20].

Query Serialization/Result Integration The Query Serialization/Result Integration component is responsible for splitting a higher level query into multiple queries and collecting results together if the source offers their content distributed over multiple pages. Two main cases are of interest:

Master-Detail Pages Many deep web sources return first an overview page which presents a list of suitable results. Detail information on each delivered result is stored on extra pages. In order to get all matching information this detail pages need to be considered as well. Having collected all relevant information an integration process is issued to transform data into instances of the wrapper exort schema. [11] investigates master-detail sources.

Restricted Number per Page Another challenge in web sources are restricted result sets. Many information sources deliver the $first - k$ or $top - k$ results at once only. In order the get the complete result sets more than one query need to be issued or (in HTML) a Hyperlink to the "next" k elements needs to be followed.

Source Repository

Generally the Source Repository contains the meta information for the configuration of the wrapper. Each wrapping component uses the Source Repository for its own tasks. Thus it contains four sections *Mapping Rules*, *Relaxation Rules* and *Serialization Information* and *Extraction Grammars.*

Relaxation Rules The task of the corresponding Relaxation Component is to ensure a fully queryable interface to the source which is based on the source result schema R. Thus Relaxation rules are formulated to answer the following questions:

- Which attributes of R are part of the Interface Schema I?
- Which attributes of R are queryable and which not?

Mapping Rules Whereas relaxation rules are based on the source result schema R, additional mapping rules have to be applied, if the desired wrapper export schema W is different from the result schema R. This is mostly the case if the integration into a mediator based information system with a common schema is targeted. In this case schema mapping rules are formulated which solve schema heterogeneity between R and W. In our prototype MiWeb [5], which is shortly introduced in section 6, we used an extra mapping component for schema mapping.

Extraction Grammars The repository entry for extraction grammars needs to be distinguished into *parameter extraction grammars* used for parameter extraction and *result extraction grammars.* The first is used in order to extract queryable fields from the relaxed and serialized query expression. An important point in this part of these grammar is the aggregation of multiple query conditions for one interface field to one expression. The result expression grammar needs to mine the result schema from the unstructured result document which is delivered by a HTML source. Some approaches use regular expression for this task, other derive their own wrapping grammars [7], [20], [11], [10].

Serialization Information Under this repository segment we sum up information required if queries need to be serialized because of result distribution over multiple pages. In case of master-detail web sources there has to be information, wether a specific attribute $attr_I$ can be found on master or detail pages or both in order to transform query and reintegrate results correctly.

To avoid partial result sets caused by source limitations additional information have to be provided about the behavior of the source: Either it delivers all results at once or it cuts after a predefined number of results. In that case information how to split a query into partial ones which deliver all valid results is stored. This part is subject of future research.

3 Query Relaxation

Relaxation Rules

Relaxation rules will be distinguished into *Attribute Substitution Rules* and *Attribute Elimination Rules.*

Attribute Substitution Rules Attribute Substitution Rules are formulated if an attribute of the source result schema $attr_R$ is not provided in the query interface schema I but an attribute $attr_I$ of the interface schema exists, which may be used to query $attr_R$ implicitly. In this case we call $attr_R$ an *Queryable Attribute*, because the extensions of $attr_R$ are indexed and can be queried. An Attribute Substitution Rule is denoted as: $attr_R \hookrightarrow attr_I$. The most common use-case for the application of an attribute substitution rule is a form-based data source which offers a "keyword search" only.

Attribute Elimination Rules In contrast Elimination Rules occur in case if an attribute of the result schema $attr_R$ does not exist in the interface schema I, and additionally, if there is no suitable attribute of the interface schema which allows an implicit querying of $attr_R$. In that case no useful mapping is possible, more over a mapping would produce incorrect query relaxations. We call this kind of attributes *Non-queryable Attributes*, meaning that the interface of the source does not provide any query opportunity for them. A relaxation rule for a non-queryable attribute $attr_R$ is written $attr_R \hookrightarrow \epsilon$, meaning that this attribute needs to be eliminated from the query before the query is issued against the source.

Notably, the approach provides an added value, because non-queryable attributes are getting queryable at the wrapper export interface, as we will elaborate in section 4.

While the query interface of CiteSeer provides keyword search only, our wrapper is able to query the attributes of the result schema, it provides the result schema $R_{CiteSeer}$. Thus the wrapper provides the query capability $(title*, author*, year*, link*, citations*) \rightarrow (title, author, year, link, citations)$. Internally, each of the left-hand attributes has to be either mapped to the interface schema of CiteSeer (i.e. to the *keyword*) or eliminated from the query: $title \hookrightarrow keyword$, $author \hookrightarrow keyword$, $year \hookrightarrow keyword$, $link \hookrightarrow \epsilon$, and $citations \hookrightarrow \epsilon$.

Query rewriting using relaxation rules

Outgoing from the original RDQL query, the wrapper relaxes it according to the source description. For each of the selection attributes contained in a query $attr_R$ it has to be checked, whether it is queryable, i.e. whether a substitution rule exists. In this case, the selection condition is rewritten accordingly. Otherwise an elimination rule exists, and thus the attribute is removed from the query. Then this selection attribute $attr_R$ has to be later applied in the result restriction phase. We discuss the query relaxation along the examples introduced in section 1.

Query with Queryable Attributes Only (cf. Examples 1.1 and 1.4). The simple query of example 1.1 can be written in RDQL as

```
(Q1) SELECT *
       WHERE  (?resource <cs:author> ?author)
       AND    ?author =~ "Garcia-Molina"
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

Using the substitution rule $author \hookrightarrow keyword$ it is simply relaxed to

```
(Q2) SELECT *
       WHERE  (?resource <cs:keyword> ?keyword)
       AND     ?keyword =~ "Garcia-Molina"
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

The complex query given in example 1.4 is represented in RDQL as:

```
(Q3) SELECT *
       WHERE  (?resource <cs:author> ?author)
              (?resource <cs:title>  ?title)
       AND    (?author =~ "Garcia-Molina" ||
               ?title=~ /Federation/)
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

Since both `author` and `title` are queryable via the `keyword` field, it is relaxed to

```
(Q4) SELECT *
       WHERE  (?resource <cs:keyword> ?keyword)
       AND    (?keyword =~ "Garcia-Molina" ||
               ?keyword =~ "Federation")
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

Query having Non-Queryable Attributes (cf. Example 1.2). Elimination rules have to be applied for non-queryable attributes.

Outgoing from the following RDQL query:

```
(Q5) SELECT *
       WHERE  (?resource <cs:author>    ?author),
              (?resource <cs:citations> ?citation)
       AND     ?author =~ "Garcia-Molina" &&
               ?citations >= 100
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

the elimination rule *citation* $\hookrightarrow \epsilon$ is applied — the `citations` attribute can not be queried through the query interface. Eventually it is also relaxed to the query `(Q2)` using a substitution rule.

Range Query (cf. Example 1.3). If range queries can not be applied through a query interface, the respective selection predicates have to be eliminated from the query like non-queryable attributes and applied to the result thereafter. Notably, the keyword query `(Q2)` is also yielded if we apply to the equality-queryable attribute `year` a range query as follows:

```
(Q6) SELECT *
       WHERE  (?resource <cs:author> ?author),
              (?resource <cs:year>   ?year)
       AND    (?author =~ "Garcia-Molina" &&
               ?year >= 1998 &&
               ?year <= 2004)
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

Alternatively, one could relax this query with several disjunctive selection conditions:

```
(Q7) SELECT *
       WHERE  (?resource <cs:keyword> ?keyword)
       AND    (?keyword =~ "Garcia-Molina" &&
              (?keyword =~ "1998" || ?keyword =~ "1999" ||
               ?keyword =~ "2000" || ?keyword =~ "2001" ||
               ?keyword =~ "2002" || ?keyword =~ "2003" ||
               ?keyword =~ "2004")
       USING  cs FOR <http://cis.cs.tu-berlin.de/citeseer-rdf/>
```

But in general, range queries can not be rewritten to equality-based queries, e.g. for the condition `?year <= 2000` or for continuous range intervals.

4 Result Restriction

Since we pose relaxed queries against the source, the result set might contain superfluous records. For instance, if a full-text index is accessed for the keyword search (as for CiteSeer), the results might contain a selection criteria anywhere in the respective document and not necessarily in the author or title attributes. Moreover, given the relaxed range query `(Q7)` above, several records in the result set might contain the respective year in their references section and do not necessarily be published between 1998 and 2004. Thus the results has to be filtered w.r.t. the previously relaxed selection criteria. In order to get the results fulfilling all the selection criteria stated in the original RDQL query, we execute the original query against the intermediate RDF result set. We give an example.

Example 4.1 For the RDQL query `(Q5)` at page 10 we issue the relaxed query `(Q2)` from page 9 against CiteSeer, whereby only the author name can be taken for the keyword search. Consequently, the results of this query has to be post-processed as follows:

- Only results that were reported as at least hundred time cited shall be filtered, and
- The string ”Garcia-Molina” has to be a part of the reported authors, and not only contained elsewhere. The latter results will be removed from the results.

Thanks to our architecture, the intermediate result is represented in RDF, such that we can execute RDQL queries on it by means of the Jena API. The execution of the original RDQL query against the result of the relaxed query delivers the correct result. For instance, given the three results displayed in figure 4 at page 12, only the first result fulfills the original query and will be returned by the wrapper. In detail, for the second result the string ”Garcia-Molina” is not contained in the `<cs:author>` element, while it is more then hundred times cited. In the third result neither of both selection criteria is fulfilled — it is less then hundred time cited and the `<cs:author>` element does not match the criteria. In fact, publications of ”Garcia-Molina” are referenced in both articles and therefore these are results of the keyword search for `"Garcia-Molina"`.

```
<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cs="http://http://www.cis.cs.tu-berlin.de/citeseer-rdf#">
 <rdf:Description rdf:about=
  "http://citeseer.ist.psu.edu/papakonstantinou95object.html">
  <cs:title>Object Exchange Across
            Heterogeneous Information Sources</cs:title>
  <cs:author> ... Hector Garcia-Molina ... </cs:author>
  <cs:year>1995</cs:year>
  <cs:link>www-db.stanford.edu/pub/papers/icde95.ps</cs:link>
  <cs:citations>243</cs:citations>
 </rdf:Description>
 <rdf:Description rdf:about="...">
  <cs:title>Zebra: A Striped Network File System</cs:title>
  <cs:author>John H. Hartman, John K. Ousterhout</cs:author>
  <cs:year>1993</cs:year><cs:link>...</cs:link>
  <cs:citations>157</cs:citations>
 </rdf:Description>
 <rdf:Description rdf:about="...">
  <cs:title>Managing Semantic ...  </cs:title>
  <cs:author>Richard Hull</cs:author>
  <cs:year>1997</cs:year><cs:link>...</cs:link>
  <cs:citations>88</cs:citations>
 </rdf:Description>
</rdf:RDF>
```

Fig. 4. An example RDF result

5 Related Work

Wrapping of information sources is an important issue of recent research. On the one hand wrapping is discussed in the specific focus of a component in a mediator system. Examples are Garlic [21] or TSIMMIS [17]. Garlic enriches wrapping functionality with planning support. On the other hand there are many systems which focus on wrapping of Web sources. Some frameworks have been developed for this task [13] or [25].
Many prior works describe the content extraction task in detail [10], [18] or [7]. Some approaches of recent research focus on web forms [28], [12] or [27]. A newer approach is the distinction between interface and a result schema and the identification of extensional overlaps between them for wrapping tasks [26]. Only a few of these papers discuss filtering issues. To best of our knowledge no of them uses RDF and RDQL for this task.

6 Experiments

We tested the approach in the *MiWeb-System* [5], which is a mediator-based information system for the integration of Web sources. The *MiWeb* system integrates metadata sources describing different types of web documents: the search engine Google [1] [4], the scientific citation index Citeseer [19], and specific resources for e-learning developed in the NewEconomy (NE) project [2]. *MiWeb* consists of three main components (see Figure 5): mediator, wrapper, and mapper. Inside the MiWeb-System Query Tunneling is used in the wrapper components of the Roodolf and CiteSeer data source.

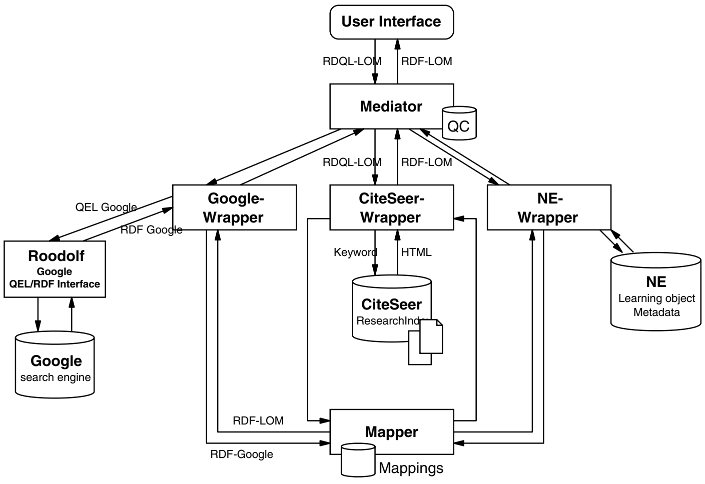

Fig. 5. Architecture of the MIWeb system

In *MiWeb* metadata is represented as a RDF model, that means the Resource Description Framework RDF [23] is used as the common data model [22]. The mediator schema adheres to the Learning Object Metadata standard (LOM) [9] that is used to describe e-learning resources. Users can query the system with RDQL [3].

The mediator is responsible for answering queries against the mediator schema. This includes

- to generate *plans* for querying the integrated wrappers so that the global query can be answered (query rewriting),
- to *execute* these queries by communicating with the wrappers and

[4] In detail, we use the QEL/RDF wrapper Roodolf ([4]) that already exists.

- to *integrate* the results by eliminating redundancies and identifying data conflicts.

Query planning is based on the descriptions of the wrappers' interfaces – the *query capabilities*. Therefore the mediator component also includes a manager for registering, changing, and deleting query capabilities. It is used to dynamically integrate data sources into the system.

In the *MiWeb* system most of the wrapping tasks are done with a grammar-based approach, which is suitable for many kinds of sources.

A user entered query is processed as follows. First a plan how to divide the query into sequences (plans) of subqueries to registered sources is generated. When these queries are sent to these sources the wrapper transforms the result into a source-specific representation in RDF. The mapper component translates specific RDF into RDF compliant to the LOM-specification, which is used by the mediator. The mediator component collects all pieces of information delivered by the sources, and integrates them to a result, which is sent back to the user interface.

7 Summary and Outlook

We presented an approach to overcome with restrictive query capabilities of web sources. More detailed, with Query Tunneling queries can be posed which might contain selection criteria against attributes that are present in the result schema only — and not in the query interface of the web source.

Wrapper employing Query Tunneling can be easily adapted, since the configuration is metadata-driven — the source specific information is provided in a descriptive manner.

Since web sources usually provide only a restricted number of results per page, query serialization plays an important role for wrappers, i.e. either the splitting of one query into several or of repetitive processing of several result pages. This is particularly relevant for Query Tunneling, because the number of results increases regularly for relaxed queries. Thus the processing of several result pages for one query needs sometimes to be done. Consequently, we are going to improve query serialization. The efficient processing of relaxed queries that lead to unmanageable large result sets (i.e. if only non-queryable attributes are used for selection criteria) will be another direction of our future research. In order to manage these issue statistical metadata about the selectivity of attributes will be investigated.

Another point of further research will focus to a method on how to infer relaxation rules automatically by using recognition technics among query and result interface as described in [26].

While we discussed the approach based on the RDF/RDQL framework, it is applicable independently of the chosen data representation and query language.

Summarizing, by means of Query Tunneling higher-order queries can be issued to web sources without to breach their autonomy towards semantic enriched querying.

References

[1] Google. http://www.google.de/.

[2] New Economy - Homepage. http://www.dialekt.cedis.fu-berlin.de/neweconomy/. Project founded of the bmb+f within the program 'Neue Medien in der Bildung'.

[3] RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.

[4] RooDolF 2.0. http://nutria.cs.tu-berlin.de:8080/roodolf2/index.html.

[5] Susanne Busse, Thomas Kabisch, and Ralf Petzschmann. MiWeb: Mediatorbased integration of web sources. Technical report, University of Technology Berlin, 2005.

[6] Susanne Busse, Ralf Kutsche, Ulf Leser, and Herbert Weber. Federated informations systems: Concepts, terminology and architectures. Technical report, Technical University of Berlin, 1999.

[7] Chia-Hui Chang. IEPAD: Information extraction based on pattern discovery. In *Tenth International World Wide Web Conference*, pages 681–687, 2001.

[8] W. Cohen, M. Hurst, and L. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *The Eleventh International World Wide Web Conference WWW-2002*, 2002.

[9] IEEE Learning Technology Standards Committee. Standard for information technology – education and training systems – learning objects and metadata. Technical report, IEEE, 2002.

[10] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *The VLDB Journal*, pages 109–118, 2001.

[11] Christoph Göldner, Thomas Kabisch, and Jörn Guy Süß. Developing robust wrapper-systems with content based recognition. In *WRAP 2004: Proceedings of the First International Workshop on Wrapper Techniques for Legacy Systems, Computer Science Reports.*

[12] Bin He, Kevin Chen-Chuan Chang, and Jiawei Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 148–157. ACM Press, 2004.

[13] Kevin Chen Chang Bin He and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR*, 2005.

[14] HP labs. Jena Java RDF API and toolkit. http://www.hpl.hp.com/semweb/.

[15] Thomas Kabisch. Grammatikbasiertes semantisches wrapping für föderierte informationssysteme. In *Tagungsband zum 15. GI-Workshop Grundlagen von Datenbanken*, pages 62–66. Fakultät fuer Informatik,Otto-von-Guericke-Universität Magdeburg, 2003.

[16] A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2), June 2002.

[17] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 564–566. ACM Press, 1998.

[18] I. Muslea, S. Minton, and C. Knoblock. Stalker: Learning extraction rules for semistructured. In *Proceedings of AAAI-98 Workshop on AI and Information Integration, Technical Report WS-98-01, AAAI Press, Menlo Park, CA (1998).*

[19] NEC Research Institute. CiteSeer Scientific Literature Digital Library. http://citeseer.nj.nec.com/cs.

[20] Mattis Neiling, Markus Schaal, and Martin Schumann. Wrapit: Automated integration of web databases with extensional overlaps., 2003.

[21] Mary Tork Roth and Peter M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 266–275. Morgan Kaufmann Publishers Inc., 1997.

[22] F. Saltor, M. Castellanos, and M. Garcia-Solaco. Suitability of data models as canonical models for federated databases. *ACM SIGMOD Record*, 20(4):44–48, 1991.

[23] W3C World Wide Web Consortium. Resource Description Framework (RDF) Model and Syntax. W3C Recommendation 22 Feb 1999, REC-rdf-syntax-19990222, Feb. 1999.

[24] W3C World Wide Web Consortium.
XQuery 1.0: An XML query language. W3C Working Draft 11 February 2005, 2005. http://www.w3.org/TR/2005/WD-xquery-20050211/.

[25] Jiying Wang and Fred H. Lochovsky. Data extraction and label assignment for web databases. In *Twelft International World Wide Web Conference*, pages 470–480, 2003.

[26] Jiying Wang, Ji-Rong Wen, Fred Lochovsky, and Wei-Ying Ma. Instance-based schema matching for web databases by domain-specific query probing. In *VLDB '04: Proceedings of the 2004 Conference on Very Large Databases*, 2004.

[27] Wensheng Wu, Clement Yu, AnHai Doan, and Weiyi Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 95–106. ACM Press, 2004.

[28] Zhen Zhang, Bin He, and Kevin Chen-Chuan Chang. Understanding web query interfaces: best-effort parsing with hidden syntax. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 107–118. ACM Press, 2004.