1 INTRODUCTION

A database Web service consists of a Web service interface with operations that provide access to a backend database. When a client sends a query to a database Web service, the backend engine submits the query to the backend database, collects the results, and delivers them to the client. The export schema describes the subset of the backend database schema that the database Web service makes visible to the clients (Sheth & Larson, 1990). Typically, a Web service announces its interfaces, including the export schema, using the Web Service Definition language – WSDL, a W3C standard.

A mediator is a software component that facilitates access to a set of data sources (Wiederhold, 1992). A mediator offers a mediated or global schema that represents an integrated view of the export schemas of the data sources.

In this paper, we focus on the design of a mediator that constructs the mediated schema adaptively from evidences elicited from user query responses. It precludes the a priori definition of the mediated schema and of the mappings between the export schemas and the mediated schema. The mediator assumes that the data sources are encapsulated by Web services, that is, they are database Web services. This assumption avoids the burden of interpreting HTML pages containing query results. Indeed, the mediator communicates with the database Web services by exchanging SOAP messages conforming to their WSDL descriptions.

The schema matching approach proposed in this paper is based on the following assumptions:
1. The database Web services accept keyword queries, that is, lists of terms as in a Web search engine.
2. The mediator accepts only keyword queries.
3. The database Web services return query results as flat XML documents. That is, each export schema consists of a single flat relational table (encoded as an XML Schema data type in the WSDL document of the service).
4. Attributes with similar domains are semantically equivalent.

This last assumption is rather strong, but the case studies described in the paper indicate that it is warranted in a number of application domains.

The remainder of this paper is organized as follows. Section 2 summarizes related works. Section 3 describes the instance-based schema matching approach. Section 4 presents the mediator architecture. Section 5 contains two case studies that illustrate the approach. Finally, section 6 contains the conclusions and directions for future work.

2 RELATED WORK

Schema matching is a fundamental issue in many database application domains, such as query mediation and Web-oriented data integration (Casanova et al., 2007; Rahm & Bernstein, 2001). By query mediation, we mean the problem of designing a mediator, a software service that is able to translate...
user queries, formulated in terms of a mediated schema, into queries that can be handled by local databases. The mediator must therefore match each export schema with the mediated schema. The problem of query mediation becomes a challenge in the context of the Web, where the number of local databases may be enormous and, moreover, the mediator does not have much control over the local databases, which may join or leave the mediated environment at will.

In general, the match operation takes two schemas as input and produces a mapping between elements of the two schemas that correspond to each other. Many techniques for schema and ontology matching have been proposed to automate the match operation. For a survey of several schema matching approaches, we refer the reader to (Rahm & Bernstein, 2001).

Schema matching approaches may be classified as syntactic vs. semantic and, orthogonally, as 

- a priori vs. a posteriori (Casanova et al., 2007). The syntactic approach consists of matching two schemas based on syntactical hints, such as attribute data types and naming similarities. The semantic approach uses semantic clues to generate hypotheses about schema matching. It generally tries to detect how the real world objects are represented in different databases and leverages on the information obtained to match the schemas. Both the syntactic and the semantic approaches work a posteriori, in the sense that they start with pre-existing databases and try to match their schemas. The a priori approach emphasizes that, whenever specifying databases that will interact with each other, the designer should start by selecting an appropriate standard (a common schema), if one exists, to guide the design of the export schemas.

An implementation of a mediator for heterogeneous gazetteers is presented in (Gazola et al., 2007). Gazetteers are catalogues of geographic objects, typically classified using terms taken from a thesaurus. Mediated access to several gazetteers requires the use of a technique to deal with the heterogeneity of different thesauri. The mediator incorporates an instance-based technique to align thesauri that uses the results of user queries as evidences (Brauner et al., 2006).

An instance-based schema matching technique, based on domain-specific query probing, applied to Web databases, is proposed in (Wang et al., 2004). A Web database is a backend database available on the Web and accessible through a Web site query interface. In particular, the interface exports query results as HTML pages. In particular, a Web database has two different schemas, the interface schema (IS) and the result schema (RS). The interface schema of an individual Web database consists of data attributes over which users can query, while the result schema consists of data attributes that describe the query results that users receive.

The instance-based schema matching technique described in (Wang et al., 2004) is based on three observations about Web databases:

1. Improper queries often cause search failure, that is, return no results. For the authors, improperness means that the query keywords submitted to a particular interface schema element are not applicable values of the database attribute to which the element is associated. For instance, if you submit a string to query search element that is originally defined as an integer, you get an error. As an example, consider submitting a title value to the search element pages number.
2. The keywords of proper queries that return results very likely reappear in the returned result pages.
3. There is a global schema (GS) for Web databases of the same domain (He & Chang, 2003). The global schema consists of the representative attributes of the data objects in a specific domain.

The keyword probing technique consists of exhaustively sending keyword queries to the query interface of different Web databases, and collecting their results for further analysis. Based on the third observation, they assume, for a specific domain, the existence of a pre-defined global schema and a number of sample data objects under the global schema, called global instances. For Web databases, they deal with two kinds of schema matching: intra-site schema matching (that is, matching global with interface schemas, global with result schemas, and interface with result schemas) and inter-site schema matching (that is, matching two interface schemas or two result schemas).

The data analysis is based on the second observation. Given a proper query, the results will probably contain the reoccurrence of the submitted value (referring to the values of the attributes of the global instances). The results will be collected in the HTML sent to the Web browser. Thus, the reoccurrence of the query keywords in the returned results can be used as an indicator of which query submission is appropriate (i.e., to discover associated elements in the interface schema). In addition, the position of the submitted query keywords in the
result pages can be used to identify the associated attributes in the result schema.

Note that, differently from (Wang et al., 2004), we work with Web services that encapsulate databases. In particular, we assume that the service interface is specified by a WSDL document that describes the input attributes (interface schema) and the output attributes (export schema). This means that both the query definitions and query answers are encoded as SOAP messages. Therefore, we avoid the interpretation of query results encoded in HTML, which introduces a complication that distracts from the central problem of mediating access to databases.

Also, differently from (Brauner et al., 2007) and (Wang et al., 2004), we avoid the use of a global schema and a set of global instances. Defining a global schema and collecting a good set of global instances are hard tasks. The technique presented here instead uses an instance-based approach to align the export schemas using the results of user queries as evidences for the mappings, as in (Brauner et al., 2006).

3 THE SCHEMA MATCHING APPROACH

The schema matching approach proposed in this paper is based on the matching process illustrated in Figure 1. The matching process is the basis of a mediator that facilitates access to a collection of database Web services, assumed to cover the same application domain. In section 4, we discuss the complete mediator architecture.

The matching process starts with a client query submitted to the mediator and the WSDL descriptions of the database Web services accessed through the mediator engine (step 1 on Figure 1). Note that, through the WSDL documents, the mediator obtains the necessary information to encode queries to be sent to the database Web services, as well to interpret the results sent back.

Our current schema matching approach works under the following assumptions:

1. The database Web services accept keyword queries, that is, lists of terms as in a Web search engine.
2. The mediator accepts only keyword queries.
3. The database Web services return query results as flat XML documents. That is, each export schema consists of a single flat relational table (encoded as an XML Schema data type in the WSDL document of the service).
4. Attributes with similar domains are semantically equivalent.

This last assumption is rather strong, but the case studies described in the paper indicate that it is warranted in a number of application domains.

When a client submits a query, the mediator forwards it to the database Web services registered through the Query Manager Module (QMM) (step 2). The result sets are delivered to the user, and simultaneously, they are cached in a local cache database (step 3). Then, the mediator analyzes the cached instances by counting instance values that reoccurs in both result sets (step 4). This task is performed by the Mapping Rate Estimator Module (MREM).

The MREM analyzes the result sets by a probing technique. The probing technique consists of counting the reoccurrence of instances values from one set in the other.

After this analysis, the MREM generates the occurrence matrix (step 5). To enable the mediator to accumulate evidences, the MREM stores the occurrence matrix in the Mappings local database (step 6). If there is a previously stored occurrence matrix, the MREM must sum the existing matrix with the new matrix, generating an accumulated occurrence matrix. If new attributes are matched, the MREM must add rows and columns to the accumulated occurrence matrix. Finally, given an accumulated occurrence matrix, the MREM also generates the Estimated Mutual Information (EMI) matrix (step 7). The generation of the EMI is explained in detail at the end of this section.

For instance, suppose that the mediator provides access to databases $D_A$ and $D_B$ with export schemas $S_A$ and $S_B$, respectively. Suppose that the mediator receives a keyword query $Q$ from the client application, which it resends to the database Web
services. Let $R_d$ and $R_b$ be the results received from $D_d$ and $D_b$, respectively. These results are analyzed to detect if there are XML elements $a_v$ in $R_d$ and $b_v$ in $R_b$ that have the same value $v$. If this is indeed the case, then $v$ establishes some evidence that $a_v$ and $b_v$ map to each other. This analysis generates an occurrence matrix $M$ between attributes from the export schemas of both databases.

As in (Wang et al., 2004), we assume that the attributes of $S_d$ and $S_b$ induce a partition of the result sets $R_d$ and $R_b$ returned by the database Web services. Suppose the attributes of $S_d$ partition $R_d$ into sets $A_1, A_2, \ldots, A_m$ and the attributes of $S_b$ partition $R_b$ into sets $B_1, B_2, \ldots, B_n$. The element $M_{ij}$ in the occurrence matrix $M$ actually indicates the content overlap between partitions $A_i$ and $B_j$ with respect to the reoccurrences of values in the two partitions. The schema matching problem now becomes that of finding pairs of partitions from different schemas with the best match. In what follows, we formalize what we mean by best match.

To address this point, we introduce the concept of mutual information (Wang et al., 2004), which interprets the overlap between two partitions $X$ and $Y$ of a random event set as the “information about $X$ contained in $Y$” or the “information about $Y$ contained in $X$” (Papoulis, 1984). In other words, the concept of mutual information aims at detecting the attributes that have similar domains, i.e., similar domain value sets.

Given an occurrence matrix $M$ of two export schemas $S_d$ and $S_b$, the estimated mutual information (EMI) between the $r^{th}$ attribute of $S_d$ (say $a_r$) and the $s^{th}$ attribute of $S_b$ (say $b_s$) is:

$$EMI(a_r, b_s) = \frac{m_{rs}}{\sum_{i,j} m_{ij}} \log \frac{\sum_{i,j} m_{ij}}{\sum_{i,j} m_{ij} \sum_{i} m_{is}}$$

Note that, if $m_{rs}$ is equal to 0, EMI is assumed to be 0 as well.

So, given an EMI matrix, the MREM derives the mappings between the export schemas elements (step 8 in Figure 1). Given an EMI matrix $[e_{ij}]$, the $r^{th}$ attribute of $S_d$ matches the $s^{th}$ attribute of $S_b$ if $e_{ij} \geq e_{ir}$ for all $j \in [1,n]$ with $j \neq s$, and $e_{ij} \geq e_{jr}$ for all $i \in [1,m]$ with $i \neq r$. Note that, by this definition, there might be no best match for an attribute of $S_d$.

4 THE MEDIATOR ARCHITECTURE

Figure 2 illustrates the architecture for a mediator implementing the schema matching approach.

The User Interface Module (UIM) is responsible for the communication between the user (clients) and the mediator. The UIM accepts two kinds of user interactions: query and registration. Through the UIM, a user could register a new data source providing the WSDL description of the database Web service. The Registration Module (RM) is responsible for accessing the WSDL and registering the new service, and creating a new wrapper to access the remote source. The UIM also accepts keyword queries. Every time the UIM receives a new query, it forwards it to the Query Manager Module (QMM). The QMM is responsible for submitting user queries to database Web services. The QMM communicates with the Local Sources Module (LSM) and Wrappers Module (WM) to access local and remote sources respectively. Moreover, the QMM communicates with the LSM to store query responses in the local cache database. The Mapping Rate Estimator Module (MREM) is an autonomous module that is responsible for accessing the local cache database to compute the occurrence matrix and the estimated mutual information matrix between export schemas and generate the mappings. The MREM communicates with the LSM to store mappings into the local mappings database. The Mediated Schema Module (MSM) is responsible for accessing the local database and used to induce the elements of the mediated schema (see Section 5 examples).
5 CASE STUDIES

5.1 Bookstore Databases Experiment

The first experiment we describe uses two bookstore databases, Amazon.com ($D_A$) and Barnes and Noble ($D_B$).

$D_A$ is available as a database Web service. Figure 3 shows a fragment of the Amazon.com WSDL. We selected the operation ItemSearch. Referring to Figure 3, this operation accepts an ItemSearchRequestMsg message as input and returns an ItemSearchResponseMsg message as output. We suppressed some elements from Figure 3 to preserve readability. Table 1 reproduces the parameter list returned by the ItemSearch operation, representing the export schema $S_A$ of $D_A$.

The second data source, Barnes and Noble ($D_B$), does not provide a Web service interface, which forced us to create one to run the experiment. Table 2 shows $D_B$ export schema $S_B$.

For the keyword query “Age”, $D_A$ returned the result set $R_A$, with 23,338 entries, and $D_B$ returned the result set $R_B$, with 6,168 entries. With both result sets in cache, the mediator applied the technique described in Section 3, generating the occurrence matrix between attributes of $D_A$ and $D_B$. Figure 4 shows the occurrence matrix and Figure 5 the estimated mutual information matrix.

Table 1: Amazon.com ($D_A$) Export Schema ($S_A$).

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>title (a1)</td>
<td>title</td>
<td>String</td>
</tr>
<tr>
<td>edition (a2)</td>
<td>edition</td>
<td>String</td>
</tr>
<tr>
<td>author (a3)</td>
<td>author name</td>
<td>String</td>
</tr>
<tr>
<td>publisher (a4)</td>
<td>publisher</td>
<td>String</td>
</tr>
<tr>
<td>isbn (a5)</td>
<td>International Standard Book Number (10 digit)</td>
<td>String</td>
</tr>
<tr>
<td>ean (a6a)</td>
<td>European Article Number - a barcoding standard – book id</td>
<td>String</td>
</tr>
<tr>
<td>pages (a7)</td>
<td>number of pages</td>
<td>Number</td>
</tr>
<tr>
<td>date (a8)</td>
<td>publication date</td>
<td>String</td>
</tr>
</tbody>
</table>

Table 2: Barnes and Noble ($D_B$) Export Schema ($S_B$).

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (b1)</td>
<td>title</td>
<td>String</td>
</tr>
<tr>
<td>by (b2)</td>
<td>author name</td>
<td>String</td>
</tr>
<tr>
<td>isbn (b3)</td>
<td>International Standard Book Number – book id</td>
<td>String</td>
</tr>
<tr>
<td>pub_date (b4)</td>
<td>publication date</td>
<td>String</td>
</tr>
<tr>
<td>sales_rank (b5)</td>
<td>number of times that other titles sold more than this book title</td>
<td>Number</td>
</tr>
<tr>
<td>number_of_pages (b6)</td>
<td>number of pages</td>
<td>Number</td>
</tr>
<tr>
<td>publ (b7)</td>
<td>publisher</td>
<td>String</td>
</tr>
</tbody>
</table>

Figure 3: Amazon.com WSDL fragment.

Figure 4: Occurrence matrix between $S_A$ and $S_B$.

Figure 5: Estimated Mutual Information matrix between $S_A$ and $S_B$.  

<?xml version="1.0" encoding="UTF-8" ?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://webservices.amazon.com/AWSECommerceService/2007-10-29"
>
<types>
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://webservices.amazon.com/AWSECommerceService/2007-10-29" elementFormDefault="qualified">
+<xs:element name="ItemSearch">
+<xs:complexType name="ItemSearchRequest">
+<xs:element name="ItemSearchResponse">
+<xs:complexType name="ItemSearchResponse">
+<xs:element name="ItemSearchRequest">
+<xs:element name="ItemSearchResponse">
</ts:complexType>
</xs:schema>
</types>
<message name="ItemSearchRequestMsg">
<part name="body" element="tns:ItemSearch" />
</message>
<message name="ItemSearchResponseMsg">
<part name="body" element="tns:ItemSearchResponse" />
</message>
<portType name="AWSECommerceServicePortType">
+<operation name="Help">
+<input message="tns:ItemSearchRequest" />
+<output message="tns:ItemSearchResponse" />
</operation>
</portType>
<binding name="AWSCECommerceServicePortType">
+<service name="AWSCECommerceService">
+<port name="AWSCECommerceServicePort" binding="tns:AWSCECommerceServicePortType">
+<soap:address location="http://soap.amazon.com/onca/soa?Service=AWSCECommerceService" />
</port>
</service>
</binding>
</definitions>
In this first experiment, we used simple comparison operators to identify reoccurred values. For textual attributes, we used the SQL statement “LIKE” and for numerical attributes, the “=” operator.

Table 3 shows the alignments found between the attributes of \( S_A \) and \( S_B \). Note that the only wrong alignment was between \( edition \) from \( S_A \) and \( sales\_rank \) from \( S_B \). These attributes are not semantically similar. However, a false matching was found due to the reoccurrence of 4974 values from \( D_A \) in \( D_B \). For instance, in the Amazon.com database, several first book editions had the \( edition \) value equal to “1” and, in the Barnes and Noble database, several books had \( sales\_rank \) also equal to “1”.

An interesting observation can be made regarding book identifiers. Starting in 2007, the 13-digit ISBN began to replace the 10-digit ISBN. \( D_A \) stores both numbers, with the attribute ISBN holding the old 10-digit ISBN and the attribute EAN, the new 13-digit ISBN. \( D_B \) stores only the new 13-digit ISBN. Our instance-based technique correctly matched attribute EAN from \( S_A \) with attribute ISBN from \( S_B \). Differently from a syntactical approach, which would wrongly match attribute ISBN from \( S_A \) with attribute ISBN from \( S_B \), our instance-based technique correctly matched attribute EAN from \( S_A \) with attribute ISBN from \( S_B \).

The date attributes would never reoccur due to format differences. For instance, Amazon.com database stores dates in the format “YYYY-MM-DD”, while the Barnes and Noble database stores the publication date as “Month, YEAR”. To solve this problem, the algorithm would have to be enhanced with a type-based filter. In our experiments, attribute \( date \) was modelled as a string in both sources.

Assume that attributes with similar domains are semantically equivalent. Based on this assumption, by analyzing the values of the estimated mutual information matrix, the mediator has some evidence of which attributes should be part of a mediated schema. For instance, by observing Table 3, we note that \( title \) from the Amazon.com database aligns with \( name \) from the Barnes and Noble database, and so on. Table 4 shows the complete mediated schema, where attribute names were chosen from the first export schema, in our case, that of Amazon.com.

### 5.2 Gazetteers Experiment

Our second experiment used two geographical gazetteers available as database Web services, the Alexandria Digital Library (ADL) Gazetteer (\( D_C \)), and Geonames (\( D_D \)). In our experiments, we accessed both gazetteers through their search-by-place-name operations.

The ADL Gazetteer contains worldwide geographic place names. The ADL Gazetteer can be accessed through XML- and HTTP-based requests (Janée & Hill, 2004). Table 5 contains the ADL export schema (\( S_C \)) and Figure 6 shows a fragment of the XML response of this service.

Geonames is a gazetteer that contains over six million features categorized into one of nine classes and further subcategorized into one out of 645 feature codes. Geonames was created using data from the National Geospatial Intelligence Agency (NGA) and the U.S Geological Survey Geographic Names Information System (GNIS). Geonames services are available through Web services. Table 6 presents the Geonames export schema (\( S_D \)) and Figure 7 shows a fragment of the XML response of this service.
Table 6: Geonames \((D_\text{G})\) Database Web Service Export Schema \((\mathcal{S}_\text{G})\).

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (d1)</td>
<td>primary name</td>
<td>String</td>
</tr>
<tr>
<td>lat (d2)</td>
<td>latitude</td>
<td>Number</td>
</tr>
<tr>
<td>lng (d3)</td>
<td>longitude</td>
<td>Number</td>
</tr>
<tr>
<td>geonameId (d4)</td>
<td>identifier</td>
<td>String</td>
</tr>
<tr>
<td>countryCode (d5)</td>
<td>country code (ISO-3166 2-letter code)</td>
<td>String</td>
</tr>
<tr>
<td>countryName (d6)</td>
<td>country name</td>
<td>String</td>
</tr>
<tr>
<td>fcl(d7)</td>
<td>Feature type super class code</td>
<td>String</td>
</tr>
<tr>
<td>fcode (d8)</td>
<td>Feature type classification code</td>
<td>String</td>
</tr>
<tr>
<td>fclName (d9)</td>
<td>Feature type super class name</td>
<td>String</td>
</tr>
<tr>
<td>fcodeName (d10)</td>
<td>Feature type classification name</td>
<td>String</td>
</tr>
<tr>
<td>population (d11)</td>
<td>population</td>
<td>Number</td>
</tr>
<tr>
<td>alternateNames (d12)</td>
<td>alternative names</td>
<td>String</td>
</tr>
<tr>
<td>elevation (d13)</td>
<td>elevation, in meters</td>
<td>Number</td>
</tr>
<tr>
<td>adminCode1 (d14)</td>
<td>Code for 1st adm. division</td>
<td>String</td>
</tr>
<tr>
<td>adminName1 (d15)</td>
<td>Name for 1st adm. division</td>
<td>String</td>
</tr>
<tr>
<td>adminCode2 (d16)</td>
<td>Code for 2nd adm. division</td>
<td>String</td>
</tr>
<tr>
<td>adminName2 (d17)</td>
<td>Name for 2nd adm. division</td>
<td>String</td>
</tr>
<tr>
<td>timezone (d18)</td>
<td>Timezone description</td>
<td>String</td>
</tr>
</tbody>
</table>

In this experiment, we submitted the keyword query “alps”. \(D_C\) returned 71 entries \((R_C)\) and \(D_D\) returned 77 entries \((R_D)\). With both result sets in cache, the mediator generated the occurrence matrix between attributes of the schemas of \(D_C\) and \(D_D\). Figure 8 shows the occurrence matrix and Figure 9 shows the EMI matrix between \(S_C\) and \(S_D\).
the estimated mutual information matrix. Based on the estimated mutual information matrix, Table 7 shows the alignments between attributes of $S_C$ and $S_D$.

By analyzing the values of the estimated mutual information matrix, the mediator has some evidence of which attributes could be part of a mediated schema. For instance, by observing Table 7, we note that attribute names from ADL aligns with name from Geonames, and so on for the other attributes in Table 7. Based on these observations, the mediator suggested the mediated schema in Table 8.

Table 7: The aligned attributes ($S_C$ - $S_D$).

<table>
<thead>
<tr>
<th>SC attributes</th>
<th>SD attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>names (c5)</td>
<td>name (d1)</td>
</tr>
<tr>
<td>bounding-box_X (c6)</td>
<td>lng (d3)</td>
</tr>
<tr>
<td>bounding-box_Y (c7)</td>
<td>lat (d2)</td>
</tr>
<tr>
<td>ftt_class (c8)</td>
<td>fcodeName (d10)</td>
</tr>
<tr>
<td>gnis_class(c9)</td>
<td>fcode (d8)</td>
</tr>
</tbody>
</table>

Table 8: The mediated schema ($S_C$ - $S_D$).

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>names</td>
<td>entry name</td>
<td>String</td>
</tr>
<tr>
<td>bounding-box_X (c6)</td>
<td>entry longitude</td>
<td>Number</td>
</tr>
<tr>
<td>bounding-box_Y (c7)</td>
<td>entry latitude</td>
<td>Number</td>
</tr>
<tr>
<td>ftt_class (c8)</td>
<td>entry class of FTT</td>
<td>String</td>
</tr>
<tr>
<td>gnis_class(c9)</td>
<td>entry class of GNIS</td>
<td>String</td>
</tr>
</tbody>
</table>

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an instance-based approach for matching export schemas of databases available through Web services. We also described a technique to construct the mediated schema and to discover schema mappings on the fly, based on matching query results. To validate the approach, we discussed an experiment using bookstore databases and gazetteers.

As future work, we intend to extend the case studies to include several data sources. In this context, we plan to investigate the alignment of the included export schemas with the mediated schema and its implications, as the association of the mediated schema with a global instance set derived from existing sources.

ACKNOWLEDGEMENTS

This work was partly supported by CNPq under grants 550250/05-0, 301497/06-0, and 140417/05-2, and FAPERJ under grant E-26/100.128/2007.

REFERENCES


