# Ontology-Based Querying of Linked XML Documents

Lule Ahmedi* Georg Lausen
Institut für Informatik, Universität Freiburg
Georges-Köhler Allee, Geb. 51
79110 Freiburg, Germany
{ahmedi,lausen}@informatik.uni-freiburg.de

## ABSTRACT

We investigate the Lightweight Directory Access Protocol (LDAP) that provides a powerful means to link distributed data collections into an entity searchable as a single collection. Besides the linking mechanism, LDAP offers a rich collection of modeling primitives required to express ontologies as a common searchable interface. Encouraged in addition by the fact that XML has become the de facto standard for information interchange on the Internet, LGAccess, our global querying system, combines ontologies and XML into a unified LDAP-based framework to improve the power for accessing related XML data in the network. We describe a global query evaluation strategy based on expressive links among entities that are spread across different local or remote servers. Also, the way ontologies are annotated to hold such network-aware links along with semantic annotations is shown. Built on grounds of the standard and well-established LDAP technology instead of the technologies that are still in the process of adoption by the community, our system is an ideal candidate for applications that need to interact with semistructured databases at a global extent.

## 1. INTRODUCTION

In recent years the proliferation of XML-based systems has dramatically increased to the point of requiring the redesign of existing software and even the creation of new paradigms to handle semistructured data. At the same time, since the conception of the LDAP protocol version 3 in 1997 [35], the use of lightweight directories to store a variety of information has been steadily gaining momentum. Today, many universities and research centers use LDAP servers as a means to manage information about their members, organizations, networks, etc., and companies like Netscape or Microsoft offer LDAP support even in their Internet browsers. Among a large number of directory servers [21], an implementation that supports running LDAP against relational databases is now available [15].

Recently, Tim Berners-Lee's vision of a "Semantic Web" [5] in which a new form of Web content, that is meaningful to computers and linked up to be easily processable by machines on a global scale, is attracting a great attention in the community. From the point of view of databases, the Semantic Web can be seen as a globally linked multidatabase, where links pointing from one database object to the other can readily carry out sophisticated tasks for users/applications when querying. To bring a meaning to the content of data, ontologies can be used to hold a kind of semantic annotations along with the outgoing links that serve to relate data. As regards the aspect of handling queries for the case of the aforementioned circumstances (interlinked documents), the Semantic Web technology presents yet an evolving infrastructure.

On the other side, the similarities between LDAP and the XML "native" model, i.e. DOM [16], makes the former an ideal candidate for querying XML-based sources without the need to incur in cumbersome transformations. Also, the development of XML technology, as well as the adoption of the still evolving standards, like XLink [12] or XPointer [11], is not as well established as LDAP.

Therefore, instead of using the technologies that are still in their infancies, we have opted to base our querying system on LDAP technology, and expect to leverage its strengths for the purpose of querying semistructured data networks at a global extent. Decisive thereby is its tight connection to network and distribution channels, that, by design, offers distribution capabilities far superior from those found in traditional multidatabase systems [35]. Links that relate remote data can make particular use of such capabilities, enabling efficient query processing of those data regardless of the breadth of their distribution.

Our main objective of combining ontologies and XML into a unified LDAP-based framework for global querying is to improve the power for accessing XML data, relieving the information searcher from considering the distributed allocation of data and their semantical and structural heterogeneity in distinct sources. This is not the case when using any of the query languages so far proposed (XQuery [7], XML-QL [13], etc.) for accessing XML documents. We illustrate the distinctions by an example, both in XQuery and in our approach, and outline the improvements we offer.

**Example 1** *Assume that we are interested in integrating data about provinces from two geographic databases, the* Terra *and GlobalStatistics databases which are available in XML form from [28]. Fragments of each of these XML*

*The work of this author is supported by the Deutsche Forschungsgemeinschaft, Aktenzeichen La 598/4-1.

*sources and their DTDs are shown in Fig.1.*

```
<gs>
<province name="Nordrhein Westfalen">
 <population>17816079</population>
 <area>34077</area></province>
<province name="Bayern">
 <population>11921944</population>
 <area>70546</area></province>
  :
</gs>
```

```
<terra>
<province name="Tirol"
 abbrev="TIR" pop="586700"/>
<province name="England"
 abbrev="ENG" pop="47100000"/>
<province name="Bayern"
 abbrev="BAY" pop="10973700"/>
 :
</terra>
```

**Figure 1: Excerpt of XML data**

*In XQuery syntax, the query would be the following:*

```
FOR $prov_gs
       IN document("gs-europe.xml")//province,
    $prov_terra
       IN document("terra-europe.xml")//province
WHERE $prov_terra/@name = $prov_gs/@name AND
      $prov_terra/@pop > 10000000
RETURN <province abbrev=$prov_terra/@abbrev>
         <name>{$prov_terra/@name}</name>,
         <population>{$prov_terra/@pop}
                           </population>,
         <area>{$prov_gs/area}</area>
       </province>
```

*and returns the* province *elements as above, with their* name, population, abbrev *taken from* TERRA, *and* area *taken from GS.*

*In our system, we achieve the same result if we submit a query to the ontology where we have to specify only the entities we want to have at the output, and not where they have to come from and how they should be combined, as in the following (using XPath):*

//StateGeopolitical[population $>$ 10000000]

*given that provinces are named as* StateGeopolitical *in the ontology.*

In this paper, we present the global query evaluation strategy behind this idea targeted to the ontologies model, putting the emphasis on the network features of the approach that works particularly well in the case of a highly distributed data topology. The rest of the paper is organized as follows: Section 2 gives a brief overview of the capabilities offered by LDAP servers. Section 3 provides an explanation of the overall architecture of our system, whereas Section 4 goes in short through the modeling issues used, leaving the discussion about links as the key constructs of this model for Section 5. A global query evaluation strategy based on links is elaborated in Section 6, and then continued with the emphasis on its efficiency in Section 7. Finally, Section 8 compares our approach with other systems, and concludes the paper.

## 2. LDAP OVERVIEW

An LDAP [35, 20] server can be viewed as a semistructured database system, specialized for the operations performed on the Internet, that is, simple and fast remote read operations that occur much more frequently than writes (that are in general local), and support for the classical client/server architecture.

The LDAP data model consists of two components: The *directory schema* defines a finite set of classes, their organization in the schema hierarchy, together with their content, attributes and datatypes. The *directory instance* contains a finite set of entries organized in a forest, where each entry (1) has a non-empty set of (possibly) multi-valued attribute-value pairs $e(a, v)$ that conform to the schema definition; and (2) belongs to at least one class which is given by the (mandatory) attribute objectClass.

The naming model defines how to arrange entries into an instance hierarchy based on their names and how to refer to any particular entry. Giving a unique name to any entry in the instance hierarchy allows to refer to any entry unambiguously. The unique name of an LDAP entry is its *distinguished name* (dn). It is formed by enumerating all the individual names of the parent entries back to the root of the hierarchy. Reading the entry's dn, e.g., (cn=capital,cn=Country,o=Ontology), one can trace from the entry capital itself through the Country back to the root o=Ontology of the tree. In this work we are mainly interested on how to refer to individual entries in the hierarchy and how to handle referred entries when searching.

The functional model determines the operations that can be performed on the directory server. The search operation acts in a similar way a database query does. It is invoked by a search request defined as a combination of the following four components: The *base* denotes the distinguished name of the entry in the directory instance where the search will start. The *scope* can be base, if the search is to be restricted to just the first node, onelevel, if only the first level of nodes is to be searched, or subtree, if all nodes under the base should be considered by the filter expression. The *filter expression* is a boolean combination of atomic filters of the form ($a$ $op$ $v$), where $a$ is an attribute name, $op$ is a comparison operator, and $v$ is an attribute value. The *projection* defines the set of attributes to be returned by the query.

*Referral Mechanisms.* There are two kinds of referral mechanisms in LDAP we make use of in our framework, i.e., aliases and smart referrals.

An *alias* is an entry that contains the dn of another entry in the same server. If the search result contains an alias, that entry is *replaced* by the entry the alias points to through its dn, i.e., by its aliased entry.

**Example 2** *Consider the following alias entry that describes capitals at the meta-data level of the geographic data introduced in Example 1*

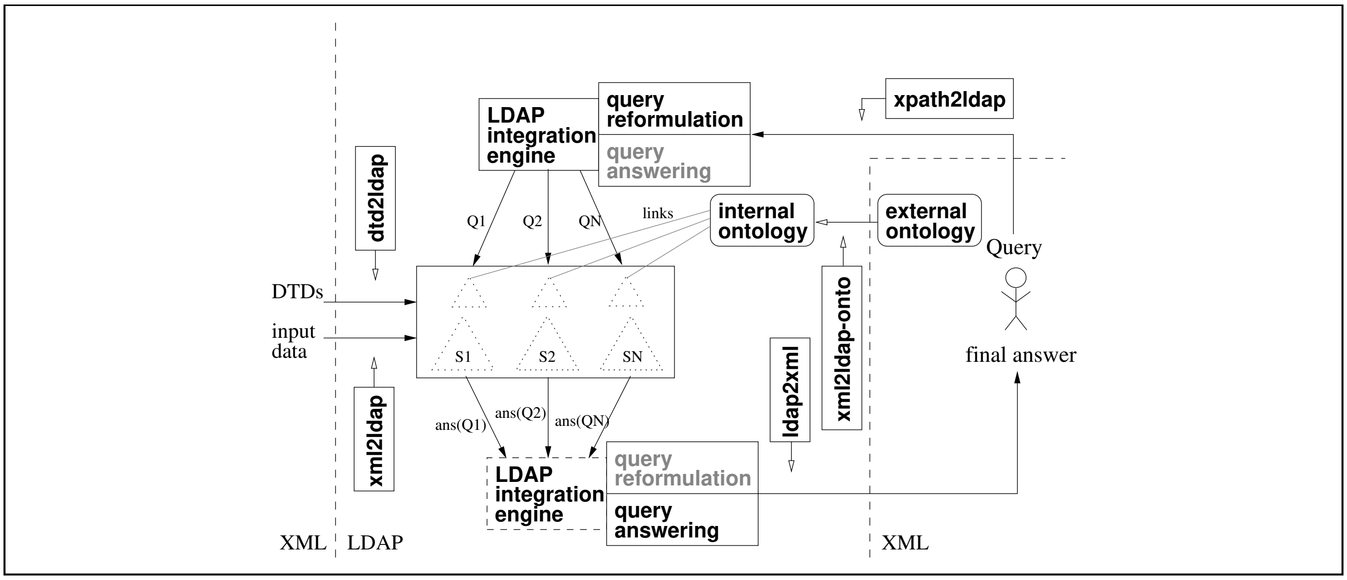


**Figure 2: The system architecture**

dn: cn=has_capital,cn=Country,o=Body,
ou=Ontology,o=LGAccess,dc=top
name: has_capital
objectClass: alias
aliasedObjectName: r=capital,c=country,c=terra,
ou=srcSchema,o=LGAccess,dc=top

*It states among others that if the entry appears in the answer set, it is to be replaced by its aliased entry, i.e., the entry whose dn is r=capital,c=country,c=terra,...,dc=top.*
*For example, the following query that looks for a concept describing capitals in a given ontology:*

Q = (cn=has_capital,cn=Country,o=Body,ou=Ontology,
o=LGAccess,dc=top ? subtree ?)

*would instead of the has_capital alias defined above result in its aliased entry*

dn: r=capital,c=country,c=terra,ou=srcSchema,
o=LGAccess,dc=top
name: capital

*that is a capital concept as described at the source schema level.*

A *smart referral* is, in a way, a generalization of an alias. It is an entry in the directory instance that refers to one or more entries in the same *or in another* LDAP server. References (ref attributes) are in the form of LDAP uniform resource locators (URLs). An LDAP URL [18] is of the form ldap://host/searchExpr, where searchExpr is the search request of the form base?projection?scope?filter to be sent to the contacted server. Smart referrals are handled in the same way as the aliases. E.g., the following reference points to the entries that satisfy a filter (indep_date > 1990) in a tree rooted at cn=Country,o=Ontology and located at the ServerA.LGAccess.org host:

ldap://ServerA.LGAccess.org/cn=Country,o=Ontology??
sub?(indep_date > 1990).

In XML terms, aliases are local, similar to the XML ID/IDREF mechanism, whereas smart referrals are global and resemble to the XLink/XPointer linking mechanism. The use of aliases and smart referrals for the global querying purposes is investigated in Section 5.

## 3. THE OVERALL ARCHITECTURE

In order to be capable of processing XML data, any common global querying framework based on the ontologies model is required to support the definition and management of schema information (e.g., the Document Type Definition (DTD) that describes the structure of an XML document), and the actual integrated contents (i.e., the XML documents).
Our framework provides, via an ontology definition, the required features to seamlessly process both types of data, as detailed in Section 6. Figure 2 shows the architecture of LGAccess, our LDAP-based global querying system. LGAccess stands for Lightweight Global Access system, a non-intrusive extension of the traditional LDAP server modified to meet the needs of global querying in the Web. It proceeds as described in the following.

An (*external*) *ontology* of the domain of interest available in the Web is incorporated in our system such that it is taken in its original representation, namely XML, and transformed into the corresponding LDAP-modeled ontology by the *xml2ldap-onto* component. Then, for each available XML source and its DTD, a corresponding representation in LDAP is generated by the *xml2ldap* and *dtd2ldap* components, respectively (for details see [27], [1]). Further, as part of the application design, the correspondences between the entries in the ontology and those in the source schemata are established by means of the LDAP referral mechanisms. Global queries are evaluated by the *LDAP Integration Engine*, composed of the *Query Reformulation Module* and the *Query Answering Module*. The user formulates a query (in XPath) over the *external ontology* and poses it to the system. The *xpath2ldap* component transforms the query from XPath to LDAP. The *Query Reformulation Module* rewrites it into the terms of the source level queries by using the knowledge derived from the referrals and the semantical definitions found in the *internal*

```
<ontology><header> ... </header>
  <body-definition> ...
    <role-def><role name="name"/><range>STRING</range>
              <cardinality>1</cardinality></>
    <role-def><role name="has_capital"/>
              <subrole-of><role name="has_city"/></subrole-of>
              <range><concept name="City"/></range>
              <inverse>is_capital_of</inverse></>
    <concept-def><concept name="GeopoliticalEntity"/>
      <role-constraint><role name="name"/><key>YES</key></role-constraint>
      <role-constraint><role name="population"/></role-constraint>
      <role-constraint><role name="area"/></role-constraint></>
    <concept-def><concept name="Country"/>
      <subconcept-of><concept name="GeopoliticalEntity"/></subconcept-of>
      <role-constraint><role name="has_capital"/></role-constraint></>
    <concept-def><concept name="StateGeopolitical"/>
      <subconcept-of><concept name="GeopoliticalEntity"/></subconcept-of>
      <role-constraint><role name="abbrev"/></role-constraint></>
  </body-definition>
  <instances> <!-- here comes the derived XML view --> </instances>
</ontology>
```

**Figure 3: Geography ontology in XML**

*ontology.* These queries are evaluated separately for each source by the LDAP server. Then, the results are combined by the *Query Answering Module* taking into account the rewriting previously formulated by the *Query Reformulation Module*. Finally, *ldap2xml* transforms the result from LDAP into XML.

This paper focuses on the *LDAP Integration Engine*. These components use some modeling bases that are briefly described in the next section. Section 5 then concentrates on modeling constructs that particulary contribute to these component facilities, i.e. links.

## 4. MODELING BASICS

*Ontologies.* Ontologies play a crucial role in our framework representing a common conceptual model for all users of a domain (concepts, roles, concept taxonomy, etc.). Built upon LDAP, they offer support for global querying (data integration) characteristics we are interested about in this paper along with the support for the standard ontology design requirements. The modeling and technical grounds of LDAP, their tight connection to network and distribution channels offer capabilities not present in prevailing data integration approaches, as will be detailed in the next section. We partition the ontology component into an *internal* ontology to be represented in the LDAP middleware in LGAccess, and an *external* ontology which resides as an XML source at a given, accessible location on the Web. Also, the *xml2ldap-onto* component is introduced in the system to provide a mapping between the conceptual entities of the internal LDAP ontology and those marked up externally in XML.

A detailed description how to model ontologies, both in LDAP and in XML, as well as a mapping between them is given in [1]. An example XML geographic ontology is sketched out in Figure 3.

*Data Modeling.* In our system, an arbitrary XML document is represented in LDAP according to the definitions made in [27] for data representation. In this representation, *elements* and *attributes* are modeled by LDAP classes. The meaning of each of the attributes in these classes (Figure 10), the oc, oid, name, order, and value attributes, is also explained there.

**Example 3** *For our running example, we assume that the system already contains:*

- *the Geography ontology whose* StateGeopolitical *concept definition with* name, abbrev, population, *and* area *roles is depicted within a solid line framed box in Figure 4, and*
- *the* TERRA *and GlobalStatistics data sources and their DTDs whose* Province *element definitions are depicted within dashed framed boxes in the same figure.*

## 5. REFERRALS AS INTEGRATION MEANS

When the modeling of the existing domain ontology, the schemata of the sources that are subject of integration and their data is done, the substantial task of the modeling process, that of establishing the relationships between those models follows.

The LDAP referral mechanisms introduced in Section 2 offer a good basis for expressing links. Aliases are well-suited for specifying local links. But, in a real-world integration scenario, data sources and their schemata may be available locally or distributed across several directory services in the network. In addition, an entry in an ontology may relate to more than one entries in available data source schemata. Smart referrals are appropriate to define potentially multiple links in a distributed directory topology.

We annotate ontologies in the middleware (internal ontologies) of our system to hold such links. Network-enabled ontologies present the gist of the discussion concerning the

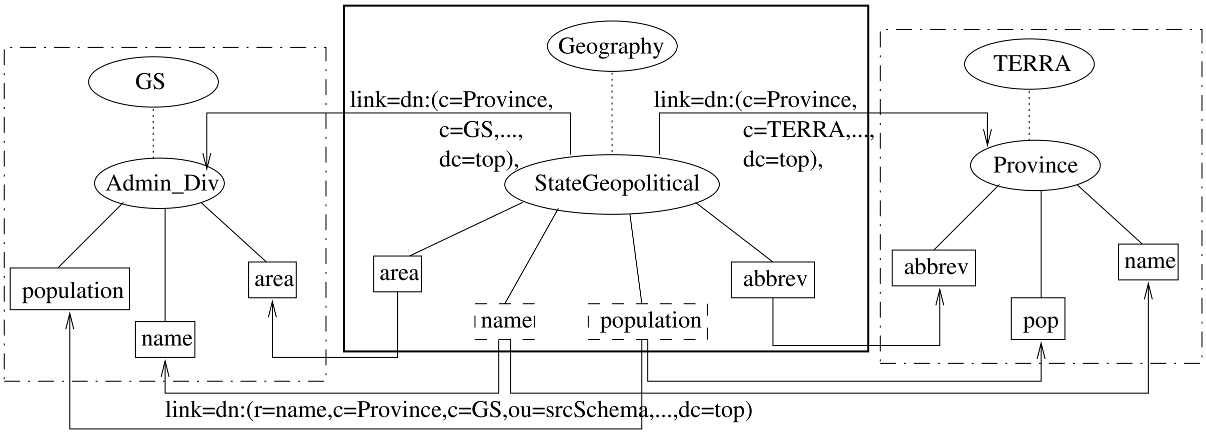


**Figure 4: The GS and TERRA source schemata embedded in the Geography ontology**

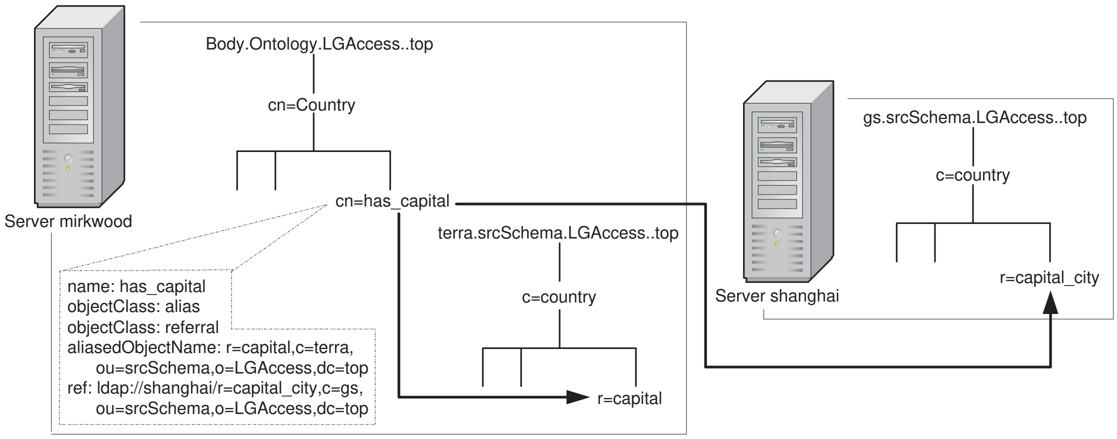


**Figure 5: An example of specifying local and remote links in LDAP**

modeling framework. Thanks to them, separate partitions of the whole model (source schemata and their data) in the network can be easily hooked together into a single conceptual view through the use of links.

As regards the interpretation of links when a query is issued to the system, LDAP's approach of handling aliases and smart referrals is quite basic. During the query evaluation in our system, the assigned links allow to relate any global queries performed over the ontology with their corresponding source level queries in a relatively easy fashion.

Figure 4 provides a graphical representation of the way our example ontology and the corresponding source schemata are related by means of the link attributes. The exact specification of these attributes in the LDAP instance of another fragment of our geographic example (Example 3) to be used later in the paper is given in Figure 5.

## 6. QUERY PROCESSING

XPath [9] is a W3C recommendation for addressing node sets of *an XML document*. Most XML querying languages are built on top of XPath as a core language. The LGAccess system supports querying in XPath by deploying a slightly modified version of the *xpath2ldap* translator developed as a part of the HLCaches project [27].

Furthermore, the meta-information of LDAP ontologies regarding a global semantical and structural definition over several distinct XML documents - their schemata - can be used by the system for evaluating XPath queries to extract relevant data from *the related XML documents*.

### 6.1 XPath into LDAP Query Translation

There are two groups that can make use of ontology definitions in the LGAccess system. One are information searchers who do not substantially care about the semantic descriptions of entities as described by the ontology and would rather be satisfied with a kind of a *view* containing only structural elements of the domain and keeping the internal layout of the view across several different sources and/or ontologies transparent to them. The other group includes users/applications who are interested in the meta-data definitions in the *ontology*. It would be nice to query related XML documents and the global meta-information about them in one format, but this looks difficult unless a standard XML ontology model and a proper language for accessing it is conceived by the Semantic Web community. See [22, 33, 31, 34] for the ongoing work on consolidating a proper query language for meta-data.

XPath as a path-oriented language isn't the proper way to query an ontology description. The XML ontology tree cannot directly be used to derive an XPath query against an XML instance. Instead, we propose to use an *XML view* of the ontology which represents a higher level abstraction of the domain of interest released from accurate semantic descriptions [1]. Hence, the ontology model does not explicitly include instance data from XML documents being integrated. It provides a pattern for picking up those data, i.e. XML view, which is enclosed within `<instances>` tags in the ontology specification (c.f. Figure 3). In this way, for *accessing the data* in LGACCESS, the information searcher formulates a query by using the view as a pattern. For an example (reduced) XML view, see Figure 6.

```
<Geography><GeopoliticalEntity>
   <Country> ...
      <car_code/><has_capital/><indep_date/></>
   <StateGeopolitical>
      <name/><population/><area/><abbrev/></>
</GeopoliticalEntity></Geography>
```

**Figure 6: Geography view in XML**

On the other side, for *accessing the meta-data* in LGACCESS using XPath, a straightforward way is to query the underlying model (LDAP), regardless of the surface syntax.

An XPath query that derives both, the underlying model of the related instance data and their meta-data in LDAP ontologies requires a slightly different algorithm when translated into LDAP from the one used in [27] that is at best illustrated through an example.

**Example 4** *If we apply the algorithm to the following XPath query (see Example 1 for the abbreviated syntax)*

$$\begin{aligned} Q_{XPath} = \; & /descendant :: StateGeopolitical \\ & [child :: population > 10000000] \end{aligned}$$

*our* xpath2ldap *convertor produces the translated LDAP query* $Q = (M, R)$ *where* $M$ *is the main query,* $R$ *is the refinement query [27], and:*

$$\begin{aligned} M &= (dn(root)?subtree? \\ &\quad (name = "StateGeopolitical")), \\ R &= (dn(StateGeopolitical)?onelevel? \\ &\quad (\&(name = "population")(value > 10000000))). \end{aligned}$$

## 6.2 Query Reformulation

Armed with the modeling formalisms and with the XPath into LDAP query translator, we are now ready to tackle the problem of handling global queries. In general terms, a query performed on the ontology must be reformulated into source-level queries. See [25, 26, 32] for other data frameworks that also support query reformulation.

In our framework, the query reformulation task involves rewriting the original query formulated over the ontology into single-source queries by taking into account the ontological constructs:

- the link attributes in both concepts and roles,
- the semantic descriptions (subconceptOf, inverseOf, ...), and
- the inter-ontology relationships (synonymOf, hypernymOf, hyponymOf)

The link attributes carry the information that relates the data residing in multiple XML documents and are hence *the* constructs that are investigated for query reformulation in the sequel. For example, the query $Q_{XPath}$ in Example 4 is formulated relative to the XML view shown in Figure 6. It is translated into LDAP yielding the query $Q$. In order to process such a query in our system, the Reformulate algorithm depicted in Figure 7 is used. The details of each step of the algorithm are given below.

*Relevant Source Discovery.* Since the cost of accessing an information source over the network is significant, the main optimization offered is the minimization of the number of external information sources that need to be accessed in order to answer the query.

The link property of pointing to the namespace contained on the same or a different server is a suitable means to delegate the query context expressed by $base$ to the new context nodes $base_{sch_i}$ - one per each local/external relevant source. This category of links is referred to as baseLinks. Thus, $baseLink_i(base_i, base_{sch_i})$ is used to denote the existence of a connection between entities $baseLink_i$ from the ontology and $base_{sch_i}$ from a source level schema (DTD). Step 1 does the identification of the relevant sources from a given data repository. It involves examining the baselinks associated with the $base$ entity appearing in $Q$ and extracting the corresponding host information $baseLink_i.host_i$ for each of them $(i = 1, M)$.

For our example query and the geographic model depicted in Figure 4, the resulting set of baseLinks is as follows:

$$\begin{aligned} L_{base} \;=\; & \{baseLink(StateGeopolitical, Province), \\ & baseLink(StateGeopolitical, Admin\_Div)\}, \end{aligned}$$

giving $\{terra, gs\}$ as the set of relevant sources.

*Query Decomposition.* The query decomposition involves finding the set of source level search requests

$$Q_{src} = \{Q_i : Q_i = (base_i, scope_i, filter_i), i = 1, M\}$$

such that each of the components, $base$, $scope$, and $filter$ in the original query $Q$ is substituted by its corresponding associate, $base_i$, $scope_i$, and $filter_i$ in the source level schemata respectivelly. The process of substituting original $base$ and $scope$ is trivial thanks to the assigned semantics to the baseLink attributes being suitable for this kind of mapping and the easy of managing and processing those links in LDAP.

The major part of the algorithm deals with finding the filter component of its dedicated query $Q_i$, i.e., $filter_i$. Step 2 reduces the filter of the original query to $filter_{ont}$ by omitting all predicates dealing with non-ontological attributes, i.e., value checking predicates. Then it evaluates this query over the ontology, ignoring for a while the existence of the underlying sources and their schemata. The result is a set $absTargets$ of so-called *absolute target entries.* The link attributes assigned on these entries, called targetLinks are parsed and their aliased and/or referred entries are extracted in Step 3 of the algorithm. We call such entries *potential targets* since they do still not represent the final targets we were looking for originally. They are entries found in the general context, i.e. without (yet) taking into consideration the context as restricted by the $base$ specification in the original query; they belong to the source schemata, not to the source

**Algorithm (Reformulate( Q, Ont, $S_{sch}$, $S_{src}$ ))**

Let $Q = (base, scope, filter)$ be a global query posed over the ontology $Ont.$, and
$S_{sch}, S_{src}$ be the set of source schemata, source data respectively.

/* STEP 1. Discovering relevant sources */
for $base$, find the set of links $L_{base}$, such that:
$L_{base} = \{baseLink_i(base, base_{sch_i}) : base \in \{Q \cap Ont\}, base_{sch_i} \in S_{sch_i}, i = 1, m\}$

/* STEP 2. Finding the set of absolute targets */
/* reduce the filter by discarding any value constraints, e.g. $(value > 1000000)$ */
$filter_{ont} = Reduce(filter)$
$Q_{ont} = (base, scope, filter_{ont})$
$absTargets = FindTargets(Q_{ont}, Ont) = \{absTarget_1, ..., absTarget_j, ..., absTarget_m\}$.

/* STEP 3. Finding the set of potential targets */
/* assume that for each absTarget, there exists at most one potTarget per rel. source */
for each $absTarget_j \in absTargets$, find the set of links $L_{target}$, such that:
$L_{target} = \{targetLink_i(absTarget_j, potTarget_{sch_{ij}}) : absTarget_j \in Ont,$
$potTarget_{sch_{ij}} \in S_{sch_i}, potTarget_{sch_{ij}}(a_{pt_j}, v_{pt_j})\}$.

/* STEP 4. Building the source level search requests */
$Q_{src} = \{\}$
for each $baseLink_i \in L_{base}$ {
/* group baseLinks & targetLinks by the host they belong to */
$potTarget_{sch_i} = \{\}$
for each $potTarget_{sch_{ij}} \in L_{target}$ {
if $(potTarget_{sch_{ij}}.host_{pt} == base_{sch_i}.host_b)$ {
$potTarget_{sch_i} = potTarget_{sch_i} \cup potTarget_{sch_{ij}}\}\}$
/* generate a new source level search request and add it to $Q_{src}$ */
$Q_i = BuildSourceQuery(Q, baseLink_i, potTarget_{sch_i})$ }
$Q_{src} = Q_{src} \cup Q_i$

return $Q_{src} = \{Q_1, Q_2, ..., Q_n\}$

**Figure 7: Reformulation algorithm**

**Algorithm (BuildSourceQuery(Q, $baseLink_i$, $potTarget_{sch_i}$))**

Let $Q_i = (base_i, scope_i, filter_i)$ be the source level query to be found.
$base_i = base_{sch_i}$
$scope_i = scope$ // simply overwrite it
$filter_i = \{\}$
$potTarget_{sch_i} = \{potTarget_{sch_{ij}} : j = 1, n\}$
for each $(a_f\ op\ v_f) \in filter$ of $Q$ {
/* replace $v_f$ with the corresponding one from the potTargets */
j=1
do {
find $(a_{pt_j}, v_{pt_j}) \in potTarget_{sch_{ij}}$ where $a_{ptj} = a_f$
if $((a_f\ op\ v_{pt_j}) \notin filter_i)$ {
$filter_i = filter_i \cup (a_f\ op\ v_{pt})$ }
j++
} while $(potTarget_{sch_{ij}})$ }

return $(base_i, scope_i, filter_i)$

**Figure 8: Reformulation algorithm (BuildSourceQuery procedure)**

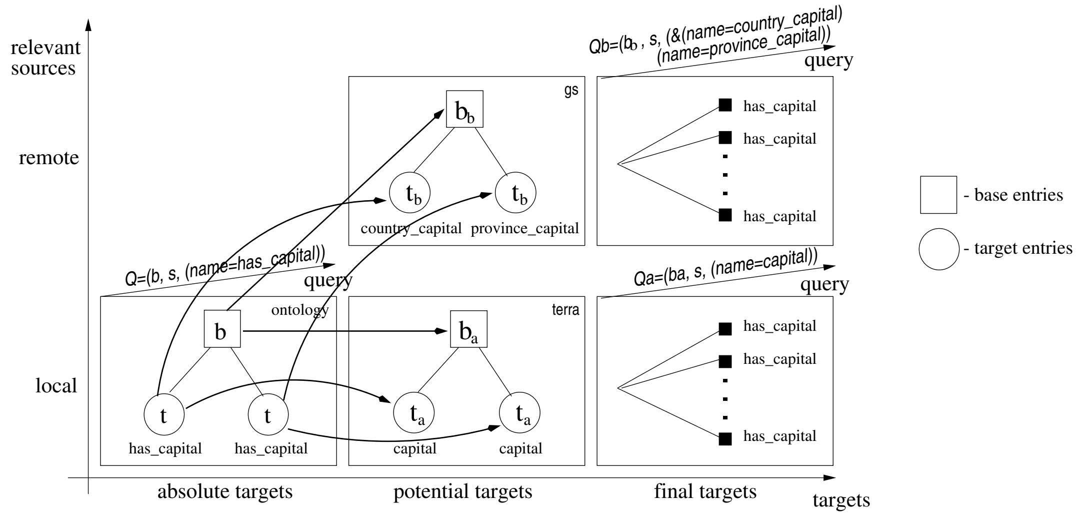


**Figure 9: The proceeding of an reformulation scenario**

data. Step 4 is the gist of the algorithm and the one that builds the source level search requests. After the baseLinks and the targetLinks are grouped by the baseLinks according to the host they belong to, they are sent as the arguments to the *BuildSourceQuery* function (c.f. Figure 8) which in turn generates a reformulated query definition in terms of its entries for each baseLink. Interesting is the way this function reformulates the filter component: For each filter predicate $(a_f \; op \; v_f)$ in $Q$, it generates a corresponding predicate $(a_f \; op \; v_{pt})$. If the set $potTarget_{sch_i}$ of potential targets contains more than one $potTarget_{sch_{ij}}$ entries, then for each $potTarget_{sch_{ij}} \in potTarget_{sch_i}$ a corresponding predicate $(a_f \; op \; v_{pt_{ij}})$ is generated if it differs from the ones generated from the previous $potTarget_{sch_{i1}}, \ldots, potTarget_{sch_{i(j-1)}}$ potential targets. At the end, the resulting predicates for a given $potTarget_{sch_i}$ are related with the boolean "&" operator.
In our case, applying Steps 2, 3, and 4 of the algorithm for the TERRA and the GS sources which are identified as relevant in Step 1 results in decomposing the original (global) query $Q$ into the following source level queries respectively:

$$\begin{aligned} Q_{Terra} &= (dn(terra.root)?subtree?(name = Province)), \\ &\quad (dn(Province)?onelevel? \\ &\quad (\&(name = pop)(value > 10000000)) \\ Q_{GS} &= (dn(gs.root)?subtree?(name = Admin\_Div)), \\ &\quad (dn(Province)?onelevel? \\ &\quad (\&(name = population)(value > 10000000))) \end{aligned}$$

Remind (Section 5) that links refer to the entries contained in the source schemata, not to the data source entries. Therefore, the task of redirecting the generated source-level queries from the source schemata to the respective data sources is also part of the reformulation process.

**Example 5** *Consider the query*

$$Q_{dist} = (cn = Country, o = Body, ou = Ontology, \\ o = LGAccess, dc = top \\ ?\; subtree \;?\; (name = has\_capital))$$

*given that the has_capital fragment of our model is as in Figure 5. The (de-)composition trace of this query wrt. different target trees and relevant sources is illustrated in Figure 9, giving an insight into the reformulation algorithm in general at the same time.*

## 6.3 Query Answering

A set of nodes obtained from evaluating an XPath expression in our system is an unordered collection of nodes *with duplicates.* Remind that the XPath Recommendation specification defined by the W3C expects an unoredered collection of nodes *without* duplicates at the result set. As regards the ordering, the LGACCESS system goes beyond the original XPath specification, and can be asked to return a sorted collection of nodes at the output based on some attribute value [19]. Duplicate nodes arise when more than one sources contain the requested data. To eliminate them, the result nodes are exposed to the following modifications:

**Renaming:** The attributes that also appear in the query use `targetLinks` to map back from the source level namespace into the ontology vocabulary. In the case of Example 5, result nodes stemming from the TERRA source replace their actual name attribute values, country_capital and province_capital, with has_capital, those stemming from the GS source replace capital with has_capital as well.

**Detection/Rejection:** Duplicates, i.e., entries with the same value(s) for the value attribute are identified. Among them, those that come from the source believed to be more reliable (up-to-date) are chosen. The rest is discarded, avoiding redundant result data.

A *flat* answer set (list) of nodes does not fit the idea of hierarchically organized collections of XML data in the World Wide Web. Due to its hierarchical namespace, LDAP is ideally suitable for managing the tree-granular answer sets. Given the aforementioned arguments, we address next the problem of obtaining trees of depth 1 at the output of the system that mirror to a certain degree the concept-role relationships of ontologies. That means, if the user asks for a concept, its children (roles) are also attached to the result, as well as the other way round. This format of returning results contributes considerably to enriching the answer set with multisource information, preserving at the same time the intended simplicity (lightweight) of our approach, ensured among others by the easy of "reading" results from the end-users. These pretensions become perceivable in the example below.

Again, for the case of the StateGeopolitical example, according to Figure 4, the set of links $L$ found in Terra and GS sources determines that the answer set of $Q_{Terra}$ and that of $Q_{GS}$ are to be merged on the name node (see the line in Figure 3 that defines name as key). As a result of such a merge operation, the StateGeopolitical one-depth trees are composed as depicted in Figure 10. Comparing the individual Province answer set of $Q_{Terra}$ together with that of $Q_{GS}$ with the combined StateGeopolitical answer set, it is obvious that the latter is richer for the attributes abbrev and/or area, for at least some of the trees in the result set.

## 6.4 Discourse

Given the current state of the system, we assumed that each link, be it baseLink or targetLink, can be specified in the ldap://host/dn form, allowing the entries to be referenced by their distinguished names. Using links in the extended ldap://host/dn?projection?scope?filter form would allow for a much richer way of referencing entries and remains as future work. These can then be used as a means for resolving the structural discrepancies across sources.

Important to note is the time complexity of our reformulation algorithm. The overhead of the algorithm is mainly dictated by the task of reformulating the filter. Due to the assumption that for each absTarget entry, say $M$ of them, there exists at most one corresponding potTarget per relevant source, say $N$ of them, it is obvious that the algorithm is linear in $M \times N$. Note that composed filters in a query do not cause an increase in complexity because they are solely a matter of no other but the LDAP query engine itself.

Besides decomposition, further reformulation procedures not considered here are generalization, specialization, inversion, etc. based on the ontological primitives like subconceptOf, inverseOf, synonymOf, etc. See [1] for a detailed description of specifying these contructs in LGAccess.

# 7. EXPERIMENTAL RESULTS

A preliminary version of the system is already available and providing very promising results.

Consider again the Example 5. We used an excerpt of the freely available CYC Geography ontology on the Web at [24] that contains among others information about geopolitical entities, i.e., countries, states, cities, etc. For the instance data, we used the Terra and the GlobalStatistics data sources. We located the Geography ontology as well as the Terra data source at the local mirkwood server, and the GlobalStatistics data source at the remote shanghai server. We first submitted one query per source separately as if we knew the naming contexts of each of the sources and their relationship to our final global intension. Then, we posed a global query against interlinked sources such that a result set equal to the union of the previous result sets is obtained by following the local and remote links as depicted in Figure 5. Surprisingly, the time needed to evaluate the global query nearly equals the sum of the times needed to evaluate each of the single source queries separately, one after the other in series without taking advantages of possible parallelism (see Figure 11). Notice that, when performing the former, the user's expertise on a given domain and his familiarity with the available data repository is no longer required.

Another interesting measure is the time spent when searching within the network in cases where not all servers contain information relevant to answer a given query. A gain in efficiency is achieved when querying the entire network without broadcasting the query to every repository, i.e., without spending time to contact irrelevant servers.

These results, combined with the fact that the system is able to implicitly glue nodes belonging to different answer trees without incurring in any additional work from the user side, makes our system an ideal candidate for the usage in practice for global querying scenario of XML data in the Web.

| Host Server(s) | Result Entries | Evaluation Time |
|---|---|---|
| mirkwood (local) | 183 | 8.926s |
| shanghai (Intranet) | 510 | 10.984s |
| mirkwood & shanghai | 693 | 20.879s |

**Figure 11: The run-time requirement analyses**

# 8. RELATED WORK AND CONCLUSION

For those familiar with XML, the question of using XLink/XPointer for specifying the links and adapt an existing XML query language to handle them in a proper way for any given distributed topology of XML documents comes natural. Actually, there is not yet an official definition about the interpretation of those links when queries occur, nor a system implementation that supports link specifications for querying in any way. [29] proposes a model where XML links are thought to serve to delegate a subtree at the point the link is defined in a document, but the approach aims not on freeing the user from the usually tremendous query definition task in case of a multisource environment (see Section 1). These facts, as well as the efficiency of the LDAP-based system to store and process XML data [27] and internet-wide pointing links (Section 7) argues for the deployment of LDAP instead of the emerging XML technologies in a middleware in our system.

Although extensive work has been done in the area of modeling ontologies for data integration, most researchers use logic-based languages like description logics [6], or F-Logic [23] to describe them and benefit from the reasoning abilities of logic systems. Our approach, on the other hand, uses a relatively simple model with a clear syntax that allows us to provide a unified formal framework to be used for both, ontology definition and information integration using the standard and well-established LDAP technology in the middleware.

Furthermore, due to the hierarchical and flexible structure of LDAP, we can very easily integrate not only structured data, but also semistructured data, as opposed to systems like Sims [2] or Observer [30] that are limited to the expressive power of their ontology description languages and are only able to deal with relational and flat file databases. Not even systems like Florid [17], Ontobroker [10], or Momis [4], which combine the reasoning capabilities of logic systems with the expressive power of an object-oriented model are able to describe XML data, as naturally as we do in our system. The lack of a tree-like modeling structure forces them to map XML into an artificial structure not particularly well-suited for such graphs, whereas our model is based

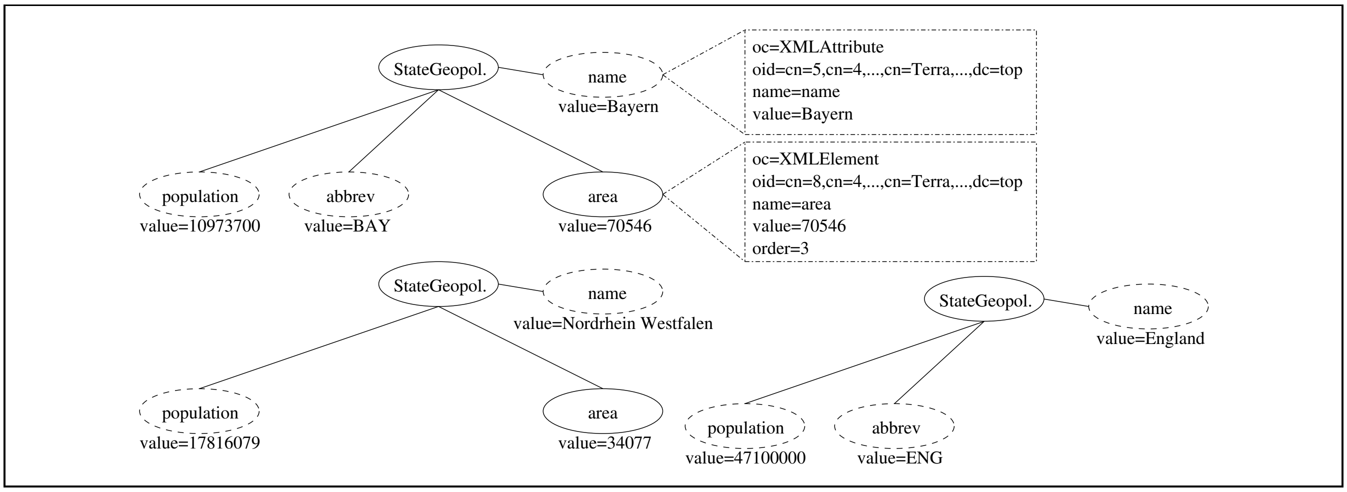


**Figure 10: The result data**

on a tree structure that resembles that of XML. TSIMMIS [14] addresses the above deficiencies by taking a similar approach to ours, namely using the OEM model to describe mediated views and sources, as opposed to our LDAP-based data model, which offers, by design, additional benefits derived by the nature of the LDAP directory services.

Some other approaches, like [3, 8] use XML as their common model for data exchange and integration, but donot consider the use of ontologies as an integral part of the system. Our LDAP-based model, on the other hand, provides seamless integration with ontologies and DTDs, which allows us to scale our system both, at the source level and at the user level.

*Conclusion.* In this paper, we have presented an approach for combining XML and ontologies into a unified LDAP-based framework that provides a common querying interface to XML data by means of powerful network-aware links that annotate ontologies and point to the instance data to be retrieved. A query evaluation strategy that supports this idea is explained in detail. Given the ubiquity of XML on the Web, our system will be useful for applications that need to interact with semistructured databases in a global sense. It can be applied to accommodate the namespace changes and mergers that are inevitable as organizations evolve, and to allow application designers to set up "search paths" for collecting results from multiple servers. It provides obvious advantages with respect to more classical integration approaches because of the simplicity, coherence, and uniformity of the LDAP model.

# 9. REFERENCES

[1] L. Ahmedi. Directory-Based Ontologies for Integrating XML Data. Intl. Workshop on Foundations of Models and Languages for Data and Objects (FMLDO'01), Viterbo, Italy, Sept. 2001.

[2] Y. Arens and C. Knoblock. SIMS: retrieving and integrating information from multiple sources. *SIGMOD Record*, 22(2):562–563, June 1993.

[3] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. *SIGMOD Record*, 28(2):597–599, 1999.

[4] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[6] A. Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[7] D. Chamberlin, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu (Eds). XQuery 1.0: An XML Query Language. W3C Working Draft, June 2001. `http://www.w3.org/XML/Query/`.

[8] V. Christophides, S. Cluet, and J. Simeon. On wrapping query languages and efficient XML integration. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):141–152, 2000.

[9] J. Clark and S. DeRose. XML path language (XPath) version 1.0. `http://www.w3c.org/tr/xpath`, 1999.

[10] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In DS-8: Semantic Issues in Multimedia Systems. Kluwer Acad. Publisher, 1999.

[11] S. DeRose, E. Maler, and R. D. J. (Eds). XML Pointer Language (XPointer) Version 1.0. `http://www.w3.org/TR/xptr/`, Sept. 2001.

[12] S. DeRose, E. Maler, and D. Orchard. XML Linking Language (XLink) Version 1.0. `http://www.w3.org/TR/xlink/`, June 2001.

[13] A. Deutsch, M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. XML-QL: A Query Language for XML. In *WWW The Query Language Workshop (QL)*, Cambridge, MA, Dec. 1998. `http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/`.

[14] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[15] O. Group. Openldap server. LDAP implementation available from `http://www.openldap.org/`.

[16] W. D. W. Group. Document object model

specification. `http://www.w3.org/DOM/`, Dec. 1997.

[17] R. Himmeröder, P. Kandzia, B. Ludäscher, W. May, and G. Lausen. Search, analysis, and integration of Web documents: A case study with FLORID. In *Proc. Intl. Workshop on Deductive Databases and Logic Programming (DDLP)*, pages 47–57, UK, 1998.

[18] T. Howes and M. Smith. RFC 2255: The LDAP URL format. Available from `ftp://ftp.isi.edu/in-notes/rfc2255.txt`, Dec. 1997. Status: Proposed Standard.

[19] T. Howes, M. Wahl, and A. Anantha. RFC 2891: LDAP control extension for server side sorting of search results.

[20] T. A. Howes, M. C. Smith, and G. S. Good. *Understanding and Deploying LDAP Directory Services*. Macmillan Technical Publishing, 1999.

[21] Innosoft. LDAP world implementation survey. Available from `http://www.innosoft.com/ldap_survey/lisurvey.html`.

[22] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF. In *The 11th Intl. World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, May 7–11 2002.

[23] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[24] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley, 1990.

[25] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering Queries Using Views. In *PODS*, 1995.

[26] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB*, pages 251–262, 1996.

[27] P. J. Marrón and G. Lausen. On processing XML in LDAP. In *VLDB*, 2001.

[28] W. May. Mondial database. `http://www.informatik.uni-freiburg.de/~may/Mondial`, 2000.

[29] W. May. Querying linked XML document networks in the Web. In *The 11th Intl. World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA, May 7–11 2002.

[30] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases*, 8(2):223–271, 2000.

[31] L. Miller. RDF query using SquishQL. Available at `http://swordfish.rdfweb.org/rdfquery/`, 2001.

[32] T. Millstein, A. Levy, and M. Friedman. Query containment for data integration systems. In *PODS*, 2000.

[33] A. Seaborne. RDQL: A data oriented query language for RDF models. Available at `http://www.hpl.hp.com/semweb/rdql.html`, 2001.

[34] M. Sintek and S. Decker. RDF query and transformation language. Available at `http://www.dfki.uni-kl.de/frodo/triple/`, 2001.

[35] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251, Dec. 1997.