Accessing RDF Knowledge Bases via LDAP Clients

Sebastian Dietzold  
(University of Leipzig, Germany  
dietzold@informatik.uni-leipzig.de)

Sören Auer  
(University of Pennsylvania, USA  
auer@seas.upenn.edu)

Abstract: LDAP based directory services constitute important parts of the IT infrastructure of many organizations and enterprises. They act as a central service for integrating new applications into an IT infrastructure and can be accessed by many different types of clients ranging from content management systems to e-mail clients. In order to make use of widely deployed tools for accessing directory services we present an approach which enables standard directory clients to obtain information from RDF stores (e.g., information about people, organizational structures and addresses). We describe how directory queries can be transformed into queries on an RDF knowledge base and give an overview of our implementation.

Key Words: knowledge management, directory services, LDAP, RDF, SPARQL

Category: H.2.7, H.2.8, H.3.5

1 Introduction

LDAP based directory services are an important part in the IT infrastructure of most organizations and enterprises. They act as a central service for integrating new applications into an IT infrastructure and can be accessed by many different types of clients ranging from content management systems to e-mail clients. The power of LDAP is its broad application support, its usage as an authorization and authentication instance and the flexible, object-oriented data model with a complex schema language, global unique object identifiers and sophisticated syntax matching rules. The most common operations on directory services are simple queries for testing the existence of an attribute of a given directory object, the search for an object based on a single attribute value and the authentication mechanism (bind) on a directory server.

The questions we try to tackle in this paper are: How can directory services benefit from the strengths of semantic web technologies? How can reasoning and rules enhance directory service data? What are the advantages of querying knowledge bases based on the resource description framework (RDF, see [Lassila and Swick1999]) with LDAP clients?

In Section 2 we motivate our work with a usage scenario. We describe in Section 3 how directory queries can be transformed into queries on an RDF
knowledge base. We give a short overview of our implementation also with regard to the execution of rules in Section 4 and conclude in Section 5 to future and related work.

2 Usage scenario

We expect that in the near future knowledge bases are the results of collaborative processes. These processes are supported by social, semantic web tools such as OntoWiki [Auer et al. 2006] or [Krötzsch et al. 2006]. An integral part of such knowledge bases is information about people, projects and organizations. The FOAF project [Brickley and Miller 2004] aims to provide an RDF schema for such information. By now, it is heavily used in the blogosphere and by many social networking services.

On the other hand, directory services based on Lightweight Directory Access Protocol [Wahl et al. 1997] are widely used within organizations and enterprises. The most common information in LDAP directories is user and contact information forming a shared address book. The primary directory schema for this use case is the inetOrgPerson schema [Smith 2000].

While inetOrgPerson focuses on a detailed and granular contact description, FOAF and its extensions describe relations between people, projects and organizations more elaborately.

In order to exploit the client support of LDAP directories with the semantic expressivity of RDF knowledge bases, we aim at integrating RDF knowledge bases with FOAF data into LDAP directories on the basis of the inetOrgPerson schema.

The use of Semantic Web technologies is in particular beneficial when new information can be inferred by processing rules. An example is when work phone information for a person can be inferred from her membership with regard to an organization or company. Such rules can be easily encoded in N3 [Berners-Lee et al. 2005] and executed e.g. by the rule processor of cwm\(^1\).

The ultimate goal is to use standard LDAP clients such as e-mail and address book software as well as software relying on LDAP authentication with RDF knowledge bases as backends.

3 Transformation and Mapping process

We analyze how LDAP queries can be translated into corresponding queries based on the query language SPARQL [Prud’hommeaux and Seaborne 2006]. The LDAP standard [Wahl et al. 1997] defines a syntax for query filters which

\(^1\) http://www.w3.org/2002/10/swap/doc/cwm.html
are used by LDAP clients to search for directory objects. This query filter together with some parameters for the search operation (e.g., the starting point of the search in the directory information tree) are the basis for the LDAP to SPARQL transformation process.

This process is divided into two parts.

1. **Query transformation**: The transformation of a given LDAP query into a SPARQL query is done in a straightforward way. The LDAP filter definition allows the compositions of more complex filters with conjunction (&), disjunction (|) and negation (!) operators. Any simple attribute filter consists of an attribute name, a filter type (equality, presence, order, ...), and a filter value. An example for a simple attribute filter is `(surname = "Mewes")`.

The corresponding SPARQL query uses group graph patterns for conjunction, `UNION` for disjunction and `OPTIONAL`, `FILTER` and `BOUND` for negation. A list of example queries for complex filter compositions is presented in Tab. 1. The namespace `ldap`, which is used in the table, describes vocabulary for the usage of LDAP schema as OWL ontologies [Dietzold2005a]. We use this namespace as the generic representation of all LDAP schema elements in RDF.

<table>
<thead>
<tr>
<th>LDAP:</th>
<th>(&amp;(gn=&quot;Jason&quot;)&amp;(sn=&quot;Mewes&quot;))</th>
<th>(conjunction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP:</td>
<td>(</td>
<td>(cn=&quot;Jay&quot;)&amp;(sn=&quot;Mewes&quot;))</td>
</tr>
<tr>
<td>SPARQL:</td>
<td>(?Entry ldap:cn &quot;Jay&quot;) UNION (?Entry ldap:sn &quot;Mewes&quot;)</td>
<td></td>
</tr>
<tr>
<td>LDAP:</td>
<td>(!((cn=&quot;Silent Bob&quot;)))</td>
<td>(negation)</td>
</tr>
<tr>
<td>SPARQL:</td>
<td>OPTIONAL { ?Entry ldap:cn ?Var1} FILTER {!bound(?Var1)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1: Query transformation: complex filter compositions**

In most cases, the queried RDF model is not an exact representation of a directory information tree (including the object hierarchy as described in [Dietzold2005b]). Instead, the RDF model is based on a semantic web ontology like FOAF so the different schemata have to be mapped as in a second step.

2. **Mapping of RDF schema identifiers to LDAP schema identifiers**: In this step of the transformation process we map the properties and classes, which are used in the LDAP directory, to properties and classes which are used in the
RDF model. For example, we would map inetOrgPerson to foaf:Person, organization to foaf:organization and givenName to foaf:givenname.

Additionally, we need some data transformation functions which are executed on the fly on the data. This is required because some attributes match semantically but have different encoding rules. An example is the mail attribute, which is in FOAF encoded as a resource with an IRI (mailto:) and in LDAP as an string attribute. These data transformation functions can be attached to RDF properties to allow a customized handling of these values.

We illustrate the transformation with an example. The following RDF statements are encoded in N3 and form the RDF model we want to query:

@prefix : <#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
:sd a foaf:Person;
 foaf:firstName "Sebastian";
 foaf:surname "Dietzold";
 foaf:phone <tel:+493419732366>;
 foaf:mbox <mailto:dietzold@informatik.uni-leipzig.de>.

The query for entries with first name Sebastian, last name Dietzold and a saved telephone number will looks as follows:

(&{gn="Sebastian"}{sn="Dietzold"}{telephoneNumber=+})

Note that this query filter string is only one part of the LDAP query. Another part is the list of attributes which are to be returned. In this example we want only the telephone numbers.

The resulting SPARQL query with mapped schema elements is

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
SELECT ?foaf WHERE { ?foaf foaf:firstName "Sebastian".
 ?foaf foaf:surname "Dietzold".
}

The SPARQL result set contains a list of identifiers with a corresponding telephone number. It has to be transformed to LDAP Data Interchange Format [Good2000] which is a intermediate syntax for the directory server.

4 Implementation

The transformation component is implemented as a backend for the widely used OpenLDAP server².

²[http://www.openldap.org]
The source code is available as open source software. The implementation works in conjunction with OntoWiki or any web accessible SPARQL endpoint such as Joseki.

Fig. 1 shows screenshots of an OntoWiki knowledge base containing information adhering to the FOAF schema. The screenshot on the right side shows the evolution address book accessing the OntoWiki knowledge base via LDAP.

Figure 1: "Same same but different": Visualization of a common RDF model in OntoWiki and over LDAP in the email reader evolution

3 http://aksw.org/Projects/LDAP/Backend
4 http://www.joseki.org
The example rule described in Section 2 can be expressed in N3 as follows:

\[
\text{\url{prefix foaf: <http://xmlns.com/foaf/0.1/> .}} \\
\text{\url{prefix log: <http://www.w3.org/2000/10/sw/swap/log#> .}} \\
\text{\url{forAll :person :org :phone.}} \\
\text{\url{:org a foaf:Organization.}} \\
\text{\url{:person a foaf:Person.}} \\
\text{\url{:org foaf:member :person.}} \\
\text{\url{:org foaf:phone :phone.}} \}
\]

\[
\text{\url{log:implies}} \\
\text{\url{:person foaf:phone :phone .}}
\]

Our implementation uses cwm to execute such rules. The inferred triples are fed back into the RDF store and are subsequently accessible for all clients.

5 Related and future work

SquirrelRDF\(^3\) is an implementation which allows LDAP servers and relational databases to be queried using SPARQL. It is part on the Jena Semantic Web Framework [Carroll et al., 2004] and allows the manually mapping of directory attributes to RDF properties. This project implements the opposite direction transforming SPARQL queries into LDAP queries.

Due to the direct mapping of LDAP queries into SPARQL queries, the performance largely depends on the performance of the SPARQL endpoint. In order to make the implementation more scalable it is furthermore desirable to perform the mapping between the LDAP and the RDF schemes directly within the RDF store and not by the query translator.

Acknowledgments

This research was supported in part by the following grants: BMBF (SE2006 #01ISF02D).

References


\(^3\)http://jena.sourceforge.net/SquirrelRDF/


