RDF Transportable Organization Databases

Leonardo Maycotte
Center for Intelligent Systems
Instituto Tecnolgico y de Estudios Superiores de Monterrey

1 INTRODUCTION

The purpose of this work is to present a methodology of how to describe and transport your organization information across the Internet with the use of Protégé[1]. It can be applied to any type of organization from schools to businesses and government. In figure 1 we can see part of a city government organization in where our methodology can be used to manage employee's information between the Mayor's Office and it's departments (financial, recreation, police, etc).

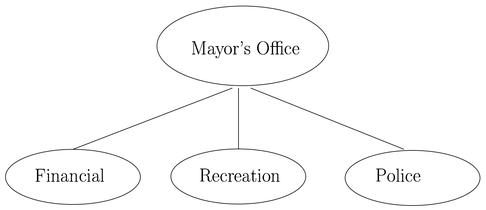

Figure 1: Organization example

2 Steps

First step: *Defining your organization schema.* We use the protégé classes as if they were database tables and the protégé slots as if they were the column names. In figure 2 we can see one employee record in the Payroll table and a query made to obtain the health benefits. The same information defined in protégé is in figure 3.

```
File  Edit  View  Terminal  Go  Help
mysql> select * from Payroll;
+--------+----------+-------------------+----------------------+
| ID     | Salary   | Benefits          | Health_Benefits      |
+--------+----------+-------------------+----------------------+
| 489812 | $35,000  | Paid vacations    | Dental Insurance     |
| 489812 | NULL     | Pension Plan      | Disability Insurance |
| 489812 | NULL     | Company Discounts | Life Insurance       |
| 489812 | NULL     | NULL              | Health Insurance     |
| 489812 | NULL     | NULL              | Paid Sick Leave      |
+--------+----------+-------------------+----------------------+
5 rows in set (0.00 sec)

mysql> select Health_benefits from Payroll where ID='489812' and Health_benefits
!='NULL';
+----------------------+
| Health_benefits      |
+----------------------+
| Dental Insurance     |
| Disability Insurance |
| Life Insurance       |
| Health Insurance     |
| Paid Sick Leave      |
+----------------------+
5 rows in set (0.00 sec)
```

Figure 2: MySQL database table

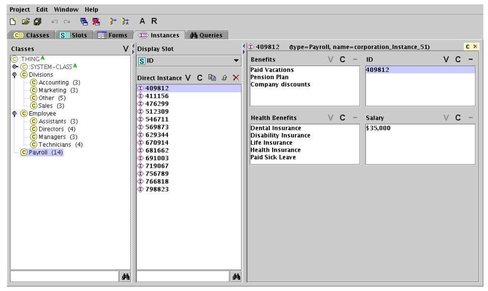

Figure 3: Database table in Protégé

Like this we define *n* protégé classes of our organization schema with one slot in common, the ID slot. This is our *key* slot and is used to define relational information in our other protégé classes, where each ID represents the unique personal number of an employee.

Second step: *Defining organization information.* Once we have defined our organization schema in protégé we save it and make it public as an RDFS file on the URL provided in our project namespace. Now the organization schema is available for use by anyone to import it to their local protégé installation and use it to define instances based on the organization schema. The instances defined by everyone are expected to be saved and sent as RDF files to an application that makes specific employee data queries.

Third step: *Parsing the rdf files.* The *rdf* & *rdfs* files are parsed using 4Suite, an open-source platform for XML and RDF processing[2]. In the next section we present an example of a web application that parses and makes queries on the *rdf* files. This application is programmed in python using 4Suite. The *rdf* documents are parsed as objects with bounded methods where the *rdf* statements (subject, predicate and object triplets) are stored in the document object.

Fourth step: *Making queries.* The queries are made with VERSA, an RDF query language[3]. First, before using VERSA, we make available the information that can be found on the *rdf* files thanks to the properties defined on our *rdf schema(rdfs file)*. This is done with the complete() method as seen in the next python function.

```
def getProperties():
  #d is the RDFS parsed doc
  resul = d.complete(None,
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#type",
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property")
  lista = []
  select = []
  for stmt in resul:
    lista.append(stmt.subject)
  for x in range(len(lista)):
    fi = lista[x].find('#')
    select.append(lista[x][fi+1:])
  return select
```

Sample of an RDFS file triplet parsed by the getProperties function:

```
<RDF Statement at 138810980:
[Subject: http://10.17.134.162/umbrella/corporation#email,
Predicate: http://www.w3.org/1999/02/22-rdf-syntax-ns#type,
Object: "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"]>
```

Now the information management is done by queries made with VERSA. The next example give us an idea of how this can be applied.

3 Example

The example presented here is a web application (python cgi's) that parses and manages information defined in RDF using Protégé. The application searches employee information in the *rdf* files by name, middle name, last name or ID. The type of employee information available is presented to the user, before the search, by parsing the *rdfs* file. For example, if the properties of our *rdf schema* file are email and telephone (there can be many more), the search can be for an employee's email or phone number defined in the *rdf* files available. The queries are made with VERSA as relational searches using the ID slot value and the protégé instance values, with functions like:

```
#Gets the ID from Protege slot names(tipo) and values(dato)
def getID(dato,tipo):
  query = '("' +dato +'" <-' +schema +':' +tipo +' - *)
  - ' +schema +':ID -> *'
  resul = Util.VersaQuery(query, m, nsMap=NSS)
  resul = encode(resul)
  if type(resul).__name__!='list':
    return [resul]
  else:
    return resul

#From the ID gets the information(bus)
def busqa(ID,bus):
  query = '("' +ID +'" <- ' +schema +':ID - *)
  - ' +schema +':' +bus +' -> *'
  resul = Util.VersaQuery(query, m, nsMap=NSS)
  resul = encode(resul)
  return resul
```

An example of a protégé instance in rdf will be:

```
<corporation:Accounting
 rdf:about="&corporation;corporation_Instance_61"
          corporation:Department="Accounting"
          corporation:ID="409812"
          corporation:Position="Secretary"
          corporation:ReportsTo="Ashley Moore"
          rdfs:label="409812"/>
```

Here the instance value would be: `corporation_Instance_61`

Having an organization ontology in protégé as seen in figure 3 we generate or update the *rdf* & *rdfs* files of our organization employee information. Then the *rdf schema* file is parsed and the information available in the *rdf* file is presented to the user as seen on figure 4.

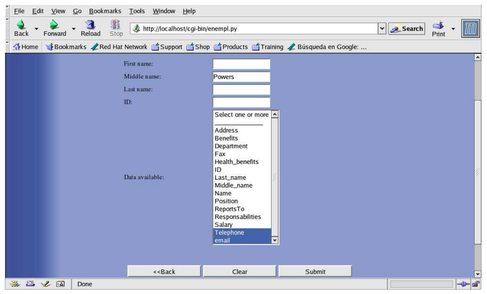

Figure 4: Organization search options

Later the user enters the ID or employee's name and selects the information he's interested in from the options available. Finally, the user gets the results from the search as we can see on figure 5.

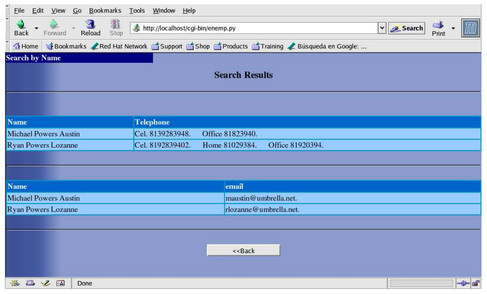

Figure 5: Organization search results

4 Conclusions

Given the increasingly complex requirements of applications, the need for rich, consistent, and reusable semantics, the growth of semantically interoperable enterprises into knowledge-based communities, and the evolution and adoption of semantic web technologies, ontologies represent the best answer to the demand for intelligent systems that operate closer to the human conceptual level[4].

References

[1] "The protégé ontology editor and knowledge acquisition system." Internet site. http://protege.stanford.edu/.

[2] "*4Suite: an open-source platform for XML and RDF processing.*" Internet site. http://4suite.org/index.xhtml.

[3] U. Ogbuji, "Thinking xml: Basic xml and rdf techniques for knowledge management, part 6 rdf query using versa." Internet site. http://www-106.ibm.com/developerworks/xml/library/x-think10/.

[4] L. Obrst, H. Liu, and R. Wray, "Ontologies for corporate web applications," *AI Magazine*, vol. 24, no. 3, pp. 49–62, 2003.