LinkedLab: A Linked Data Platform for Research Communities

Fariz Darari and Ruli Manurung
Information Retrieval Laboratory, Faculty of Computer Science, Universitas Indonesia

E-mail: fadirra@gmail.com, maruli@cs.ui.ac.id

Abstract—**Although most research communities actively disseminate information about their projects, publications, and members, there is no well-established standard for doing so, and thus the value of this data is reduced, e.g. in terms of accessibility, discoverability, and reusability. Linked Data is a method of publishing and linking structured data on the Web using RDF (Resource Description Framework). We propose LinkedLab: a Linked Data based solution for data management regarding research communities. The use of Linked Data affords a more effective way to access, organize, and integrate this data. LinkedLab is organized into three major parts: the LinkedLab ontology, the LinkedLab nodes, and the LinkedLab integrator. The ontology presents a unified view of structured data stored across multiple LinkedLab nodes, enabling data integration by the integrator. The combination of these parts provides all necessary requirements to maximizing the potential benefits of Linked Data in the research environment.**

I. Introduction

INFORMATION concerning research communities typically consists of interlinked data, both internally within the scope of a research community itself, and externally with others. For example, data about publications and research outcomes are associated with the data of a project that produces them, and data about that project is also linked to the data of research members from various communities who contributed to the project. These data and their links, however, are often hidden away in the form of HTML pages on research community platforms. This is still common in practice today as we hardly access, relate, and process the data from various research community platforms. The data is locked in silos, making them difficult to disseminate and integrate with other data. Also, the model of the data is implicit and ambiguous. This implies the problem of understanding the domain knowledge.

One possible approach to address these problems is proposed in [1], which makes use of Semantic MediaWiki (SMW), a wiki application embedded with semantic functionalities, to build a semantic portal for the AIFB institute. The use of SMW combines the advantages of social and semantic applications. The content can be created and maintained collaboratively, while the semantic addition creates a flexible, extensible, and structured knowledge representation, enabling features such as semantic search and RDF data export.

Another related approach is the VIVO application [2]. VIVO can be populated with researcher interests, activities, and accomplishments, enabling the discovery of researchers and collaborators across institutions. It uses RDF and supports browsing and a search feature that returns faceted results of needed information. VIVO also includes an ontology that works as semantic glue among VIVO applications.

Both approaches are based on the concept of Linked Data. Linked Data is a technique to publish and link structured data on the Web [3]. This brings a web of data as an extension of the World Wide Web. There are four principles of Linked Data:
1) Use IRIs as identifiers for things
2) Use HTTP IRIs so that people can look them up
3) When someone dereferences an IRI, give useful standards-based information, e.g. RDF, SPARQL
4) Include links to other IRIs, so they can find more things.

The challenges in the data management of research communities as mentioned before can be tackled with Linked Data. The data are not held in silos anymore, which leads to easier reuse and integration by external applications. For instance, one can easily build a list of publications and projects from various research communities. Also, structured data enables machines to process the data more efficiently, affording the ability to handle complex queries, e.g. a query for a student who is looking for a supervisor on his/her specific research area, concerning a specific project funded by a particular organization. Communication among research members can also be facilitated unambiguously and more effectively due to the use of ontologies in Linked Data. The LinkedLab platform is built to accommodate such features.

The remainder of this paper is as follows. In Section

II, we provide an overview of the LinkedLab design. Section III explains the implementation of the platform, while Section IV reports some evaluations to its Linked Data features. We conclude and discuss future work in Section V.

II. Design

LinkedLab is a Linked Data platform to manage data of research communities. It is designed to publish, edit, consume, and integrate data between research communities. We divide LinkedLab into three components: the LinkedLab ontology, the LinkedLab nodes, and the LinkedLab integrator. The LinkedLab ontology is the schema part of the data, which defines the vocabulary for describing concepts and relationships in the research community domain. It is meant to provide a unified view of the data contained in all LinkedLab nodes. A LinkedLab node represents a single research community, which has its own specific data. The process of publishing, editing, and consuming data happens here. Finally, the LinkedLab integrator is responsible for merging and querying data across multiple LinkedLab nodes. The general relationship between these various parts is depicted in Fig. 1.

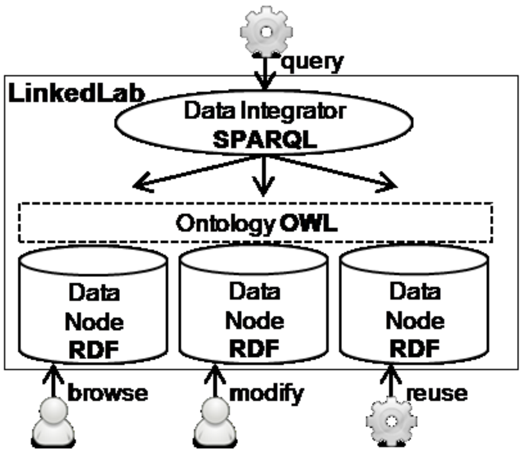


Fig. 1. LinkedLab Architecture

The architecture is based on the Semantic Web layer cake [4]. LinkedLab itself is on the topmost layer, i.e., the User Interface and Applications layer, and is constructed using components from the layers below, e.g. Query, Ontology, and Data Interchange. Each of these layers is standardized by the W3C (World Wide Web Consortium) and uses SPARQL, OWL, and RDF as its underlying technology respectively. By using these building blocks and standard technologies, LinkedLab can apply the benefits of Linked Data to its domain, i.e. to serve as a platform to manage data for research communities.

LinkedLab also combines both decentralization and centralization approaches. The decentralization notion in LinkedLab is due to the data sources that are distributed between research communities. Each research community has full control over its own data source. However, each data source is modeled using an ontology that is shared between research communities, i.e., the LinkedLab ontology. The use of a shared ontology makes data integration straightforward. This can be done because the data sources provide a similar view of the domain [5], i.e. the research community domain.

A. LinkedLab Ontology

Ontologies define a set of representational primitives, e.g., classes and properties, with which to model a domain of knowledge [6]. They can be encoded using OWL (Web Ontology Language) [7]. The LinkedLab ontology is designed to describe concepts and relationships related to the research community domain such as research outputs, research activities, and organizational profiles. It is also meant to be the semantic glue for data integration between LinkedLab nodes. We follow an ontology development methodology [8] to build the LinkedLab ontology. In order to determine the scope of the ontology, we started from the following competency questions, e.g. (1) Who are the members of a research community and what are their roles? (2) What papers were written by a particular researcher and who are his/her coauthors? (3) What projects were done by a research community and who funded them?

There are various existing ontologies, such as FOAF and BIBO, which can be reused. Each of these ontologies covers different specific concepts in the LinkedLab ontology; FOAF describes people and BIBO specifies documents. This reduces the costs related to the ontology development and gives an interoperability benefit as existing Linked Data applications can also consume LinkedLab data.

The research community domain consists of seven key terms: `Person`, `Document`, `Project`, `Topic`, `Product`, `Event`, and `Organization` [9]. We list the additional terms that are related to them, such as `Role`, `Membership`, and `Authorship`. We also list the properties for each term.

Next, we define the classes from the list of terms, and arrange them into a taxonomic hierarchy. We followed a top-down approach that defines classes from most general to most specific. For example, as for the term `Event`, we define a general event class, followed by stating its subclasses, such as `Conference`, `Workshop`, `Meeting`, and so on.

The remaining terms become OWL properties. OWL distinguishes between two kinds of properties: object properties, which link between individuals, and datatype properties, which link individuals to data values. We determined which properties should be included for each class. For example, the object properties for the class `Event` would be `organizer` and `based_near`, whereas `organizer` denotes an organization to be the organizer of an event, and `based_near` describes the location of an event.

Moreover, `Event` would also have datatype properties, such as `startDate`, `endDate`, and `description`. The characteristics of properties, such as domain, range, inverse, transitive, and functional, need to be described as well. In the LinkedLab ontology, properties like `isSubEventOf` and `isSuperEventOf` are the inverse of each other. They are also transitive and have `Organization` as their domain and range.

We faced an issue regarding the modelling of *n*-ary relations, as OWL properties are binary. Some relations such as authorship and membership link an individual to multiple values. Hence, we follow an approach that introduces a new class for the relation [10]. For example, to model the authorship relation, we create a class named `Authorship`, and then we define three new properties that have `Authorship` as their domain, such as `author`, `document`, and `rank`. By using instances of the class, we can describe the related author, the related document, and the author rank of a document authorship. The VIVO ontology [2], which is also reused in the LinkedLab ontology, also implements this model, which is illustrated in Fig. 2.

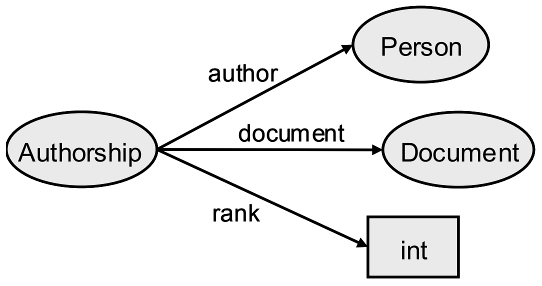


Fig. 2. Authorship Modeling

B. LinkedLab Node

The LinkedLab node represents a single research community and stores its data, which is identified by the LinkedLab ontology. The node serves three main functions: data publishing, data editing, and data consumption. All these functions must meet the Linked Data principles as mentioned earlier.

1) Data Publishing

The LinkedLab node publishes its data in a Linked Data manner. The instances contained in the node are identified by IRIs, and can be dereferenced using HTTP IRIs with 303 redirects and content negotiation [3]. For example, the identifier for a researcher, whose name is "John Doe", would be like "http://example.org/id/john-doe". When someone dereferences that IRI, they will get either the HTML page or the RDF description of John Doe, depending on the content type request. The RDF description describes the researcher, containing data such as his email, role, research areas, and publications. The description uses annotations from the LinkedLab ontology. In order to link the published data to external IRIs, we allow the RDF description to be annotated with `owl:sameAs`, which denotes the equivalence between two instances. LinkedLab nodes are also equipped with a SPARQL endpoint, so one can execute SPARQL queries over their data.

2) Data Editing

One of the core functionalities in LinkedLab nodes is data editing. The editing process covers creating, updating, and deleting data. The LinkedLab node must allow non-expert users, i.e., users with no prior experience of Linked Data, to edit its data easily. The solution is to use semantic forms as the interface for data editing. Such semantic forms look similar to regular forms which are familiar to Web users, but they are used to edit RDF data. Their behaviors are driven by underlying ontologies, e.g., the LinkedLab ontology. They are also instance-centric, i.e., an editing takes place on each instance. The fields shown in a semantic form are taken from the properties of the classes of an instance that is to be edited. Furthermore, the field values are obtained from the property values of the instance's properties. For example, when we edit data about an instance of `Organization`, the corresponding properties such as `name`, `homepage`, `mbox` or email address, and `subOrganizationOf`, will be displayed as form fields, filled with property values from the instance.

3) Data Consumption

A LinkedLab node must be able to consume both its own data and external data, e.g., the LOD cloud [11]. The consumption patterns can be varied such as browsing, searching, and visualizing data. There are two advantages of using Linked Data for data consumption: standardized data representation and access, and openness of the Web of Data [11]. Those advantages would provide features such as faceted browsing, semantic search, and rich data visualization in LinkedLab nodes, with little effort. Further, by importing external data, we can give additional description to instances on nodes. For example, to add extra description about `Information Retrieval` as a research topic instance in a LinkedLab node, one can import data from DBpedia, a part of the LOD cloud, at http://dbpedia.org/data/Information_retrieval.

C. LinkedLab Integrator

One of the benefits of Linked Data is easier data integration, as Linked Data relies on standardized data models and access mechanisms. In the LinkedLab platform, data concerning various research communities are distributed on their own LinkedLab nodes. The LinkedLab integrator is built to combine data among those nodes, so one can query and make use of the merged data.

The integrator is basically a SPARQL server that employs the SPARQL federated extension [12]. The extension enables the integrator to send parts of a SPARQL query into a set of remote endpoints of LinkedLab nodes and then merge the query results. The extension relies on the SERVICE keyword to invoking the federated query, in which one of its patterns is as in Fig. 3.

```
  #variables to appear
  SELECT ...
  #query pattern
  WHERE {
  ...
  #subquery pattern to be sent to remote
endpoints
      SERVICE ?endpoint {
      ...
      }
  }
```

Fig. 3. SERVICE clause

The SERVICE clause above involves a variable, which binds to the default graph that contains data about remote endpoints on LinkedLab nodes. The graph uses the voiD vocabulary [13]. The following example on Fig. 4 describes an endpoint for the default graph in the Turtle syntax:

```
  <http://example.org/> a void:Dataset ;
     void:sparqlEndpoint
     <http://example.org/sparql>.
```

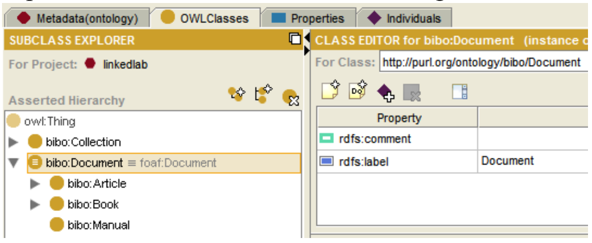
Fig. 4. Endpoint in the Turtle Syntax

III. Implementation

The implementation efforts of LinkedLab can be separated into five different steps: ontology implementation, Semantic MediaWiki reuse, ontology integration, data population, and data integration. The ontology implementation produces `lab.owl` as the serialization of the LinkedLab ontology. As for ontology integration and data population, we employ Semantic MediaWiki as the basis for the LinkedLab node. Then, we integrate the LinkedLab ontology and populate data into the node. Finally, we build the LinkedLab integrator by reusing Fuseki, a SPARQL server that already supports the federated query extension.

A. Ontology Implementation

The LinkedLab ontology is created using the Protégé-OWL ontology editor [14]. The implementation steps follow the LinkedLab ontology design as mentioned earlier. As for the terms that are not defined in the reused external ontologies, we define them in a separate namespace with the prefix `lab:`. The consistency checker of Protégé-OWL is utilized during the implementation process to guarantee the consistency of the resulting ontology. The LinkedLab ontology is serialized using the RDF/XML syntax and is saved as `lab.owl`. It uses the OWL DL expressivity. Fig. 5 shows the implementation of OWL classes in Protégé-OWL.



Property	
rdfs:comment	
rdfs:label	Document

Fig. 5. Protégé-OWL Class Editor

B. Semantic MediaWiki Reuse

Semantic MediaWiki (SMW) [1] is a semantic wiki application that operates based on Linked Data principles in a mature way. The semantic functionalities supported by SMW have correspondences with requirements for implementing the LinkedLab node. All data created within SMW can be published via Linked Data. SMW IRIs' pattern to acquire the RDF description of an instance is `http://{{sitename}}/wiki/index.php/Special:URIResolver/{{instancename}}`. In order to model *n*-ary relations, SMW can be extended with the Semantic Internal Objects extension. As for the editing function, there is the Semantic Forms extension, which provides a forms-based interface to edit SMW data easily. As for the data consumption, SMW has various features, such as browsing its own semantic data with the `Special:Browse` page, semantic search on `Special:Ask`, faceted browsing using the Semantic DrillDown extension, data visualization with the Semantic Result Formats extension, and external data reuse by the External Data extension. SMW data can also be synchronized with a triple store, e.g., Joseki, thus enabling data sharing via SPARQL endpoint. Based on all of the above considerations, we decided to employ Semantic MediaWiki as the foundation for LinkedLab nodes.

C. Ontology Integration

Ontology integration is the process to integrate the LinkedLab ontology into a LinkedLab node. This must be done so that LinkedLab nodes can annotate their data with the LinkedLab ontology. We developed a program that supports this process, called OWL2SMW. The program reuses Protégé-OWL API and Pywikipedia bot. There are two steps of running OWL2SMW. First, the program transforms the OWL ontology into the SMW syntax serialization using Protégé-OWL API. As for this process, we follow the conversion guidelines from [15]. Next, the Pywikipedia bot creates vocabulary reuse pages and generates categories and properties as the transformations from OWL classes and properties respectively, on the LinkedLab node.

D. Data Population

There are two ways of populating data into LinkedLab nodes, either by using OWL2SMW or the Semantic Forms extension. The first method can only be taken if there are already data available in OWL format that conforms to the LinkedLab ontology. The program converts the input data into the SMW syntax serialization. We use `[[Category:{{class name}}]]` and `[[{{property name}}::{{property values}}]]` to denote classes and properties of an instance respectively. Subsequently, the Pywikipedia bot will generate those instances as wiki pages on the node. Furthermore, one can also manually input data using Semantic Forms on the node, which are generated automatically by OWL2SMW.

E. Data Integration

The LinkedLab integrator is implemented to support data integration between LinkedLab nodes. We reuse and configure Fuseki to serve as the LinkedLab integrator. It also loads an RDF file as the default graph which contains the description about endpoints in LinkedLab nodes by specifying the parameter `file`. The query in Fuseki can be done using the provided query form or an HTTP GET request by putting the encoded query as the value of the parameter `query`. The results of the query can be formatted as XML, JSON, or CSV.

IV. Evaluation

We evaluated the platform by performing some use cases regarding Linked Data features such as data publishing, data editing, data consumption, and data integration. We used research community data from the Information Retrieval Laboratory at Universitas Indonesia as testing data to build the LinkedLab node. The data was extracted from the lab website using Perl scripts. We also built two other LinkedLab nodes using dummy data for evaluating data integration.

A. Data Publishing

The publishing function of LinkedLab nodes was evaluated by checking if the data is already published according to RDF publishing best practices [11], employing 303 redirects and content negotiation correctly. We used cURL to get the data published in our platform, as it can simulate how a Linked Data browser works.

One of the scenarios is that we want to get the machine readable description of ICACSIS 2010, an instance of the class `Conference`. First, we sent an HTTP request to the IRI of the instance to get its RDF description, as in Fig. 6.

```
  curl -I -H "Accept: application/rdf+xml"
http://localhost/wiki/index.php/Special:URI
Resolver/ICACSIS2010
```

Fig. 6. HTTP request example

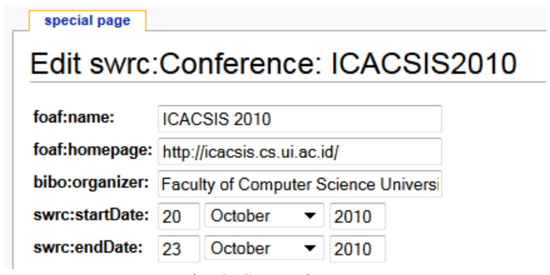
The node returns the request with the header `303 See Other` and the parameter `Location: http://localhost/wiki/index.php?title=Special:ExportRDF/ICACSIS2010&xmlmime=rdf`. This would redirect the request to the location of the RDF document of the instance. The received RDF document describes the conference, such as its homepage, start date, end date, and organizer. The snippet of the document in RDF/XML can be seen as in Fig. 7.

```
  <rdf:type
rdf:resource="&swrc;Conference"/>
  <swivt:wikiNamespace
rdf:datatype="http://www.w3.org/2001/XMLSch
ema#integer">0</swivt:wikiNamespace>
  <foaf:name
rdf:datatype="http://www.w3.org/2001/XMLSch
ema#string">ICACSIS 2010</foaf:name>
  <foaf:homepage
rdf:resource="http://icacsis.cs.ui.ac.id/"/
>
  <bibo:organizer
rdf:resource="&wiki;Faculty_Of_Computer_Sci
ence_UI"/>
  <swrc:startDate
rdf:datatype="http://www.w3.org/2001/XMLSch
ema#dateTime">2010-10-
20T00:00:00</swrc:startDate>
```

Fig. 7. Snippet of RDF/XML document

B. Data Editing

The editing process on the LinkedLab node is done through Semantic Forms. For example, if we want to edit the `ICACSIS2010` instance, we could access `http://localhost/wiki/index.php/Special:FormEdit/swrc:Conference/ICACSIS2010`. The displayed form in Fig. 8 shows the properties coupled with their current values of `ICACSIS2010` as the instance of the class `Conference`. One can edit the values and save the changes in the form to update the RDF description of the instance.



Edit swrc:Conference: ICACSIS2010

foaf:name:	ICACSIS 2010		
foaf:homepage:	http://icacsis.cs.ui.ac.id/		
bibo:organizer:	Faculty of Computer Science Universi		
swrc:startDate:	20	October	2010
swrc:endDate:	23	October	2010

Fig. 8. Semantic Forms

C. Data Consumption

Data consumption is a feature of the LinkedLab node to consume both its own data and external data. Here, we evaluate the Semantic Search feature. SMW provides Semantic Search as a way to query its own data. The query is built using the SMW QL syntax [15]. The form for executing queries is located on the `Special:Ask` page. We execute queries that utilize data structure on the LinkedLab node, which is derived from the LinkedLab ontology. For example, if we want to find all people who have written a paper presented at ICACSIS 2010, we execute query as seen in Fig. 9.

```
  [[Category:foaf:Person]][[-
vivo:linkedAuthor::<q>[[vivo:linkedInformat
ionResource::<q>[[bibo:presentedAt::ICACSIS
2010]]</q>]]</q>]]
```

Fig. 9. Query example

D. Data Integration

Data integration between LinkedLab nodes is implemented using SPARQL with its federated extension [12]. Here, we execute queries in the LinkedLab integrator that use the federated query pattern as mentioned before. For example, in order to get all publications on the topic of knowledge representation, including the name of its first author, we invoke query as seen in Fig. 10.

```
  SELECT ?dataset ?title ?author_name
WHERE {
  ?dataset void:sparqlEndpoint ?ep .
  SERVICE ?ep {
    ?authorship vivo:linkedAuthor ?person
    ?authorship vivo:authorRank 1 .
    ?person foaf:name ?author_name .
    ?authorship
vivo:linkedInformationResource ?pub .
    ?pub dcterms:title ?title .
    ?pub dcterms:subject ?topic .
    ?topic owl:sameAs
<http://dbpedia.org/resource/Knowledge_repr
esentation>
    }
}
```

Fig. 10. Example query to gel all publication on the knowledge representation topic

The first subquery just before the SERVICE block retrieves all endpoints of LinkedLab nodes listed on the default graph to be bound on the variable `?ep`. Next, the SERVICE keyword invokes the subquery inside the block to the remote endpoints. The subquery would get publications on knowledge representation with its first author on each LinkedLab node. Then, the results are merged and are returned according to the requested format.

V. Conclusion and Future Work

The LinkedLab platform is built as a solution for data management in the research community domain. The platform conforms to the Linked Data principles on its operations such as data publishing, data editing, data consumption, and data integration. The use of Linked Data brings easier reuse and integration, more sophisticated data processing by machines, and a common understanding that can improve the communication among research members.

On the next development, we plan to extend the LinkedLab ontology with other relevant ontologies such as resume and CV ontologies. Furthermore, there is a need to implement more use cases for data consumption, such as faceted browsing and visualization of merged data from the LinkedLab integrator.

Acknowledgments

We would like to thank Evi Yulianti, S.Kom., for her help providing data during the evaluation phase.

References

[1] D. M. Herzig and B. Ell, "Semantic MediaWiki in Operation: Experiences with Building a Semantic Portal," Lecture Notes in Computer Science, vol. 6497/2010, pp. 114-128, 2010.
[2] (2011, August) VIVO. [Online]. http://vivoweb.org/
[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," International Journal on Semantic Web and Information Systems (IJSWIS), pp. 1-22, 2009.
[4] World Wide Web Consortium (W3C). (2011, November) Semantic Web Layer Cake. [Online]. http://www.w3.org/2007/03/layerCake.png
[5] H. Stuckenschmidt and F. v. Harmelen, Information Sharing on the Semantic Web. Germany: Springer-Verlag Berlin Heidelberg, 2005.
[6] T. Gruber, "Ontology," Encyclopedia of Database Systems, 2009.
[7] J. Bao and D. Calvanese. (2009, October) OWL 2 Web Ontology Language Overview. [Online]. http://www.w3.org/TR/owl2-overview/
[8] N. F. Noy and McGuinness D. L. (2001) Ontology Development 101: A Guide to Creating Your First Ontology. [Online]. http://protege.stanford.edu/publications/ontology_development/ontology101.html
[9] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle, "The SWRC Ontology - Semantic Web for Research Communities," in Progress in Artificial Intelligence 12th Portuguese Conference on Artificial Intelligence (EPIA 2005), Covilha, 2005, pp. 218-231.
[10] N. F. Noy and A. Rector. (2006, April) Defining N-ary Relations on the Semantic Web. [Online]. http://www.w3.org/TR/swbp-n-aryRelations/
[11] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space (1st edition), 111136th ed.: Morgan & Claypool, 2011.
[12] E. Prud'hommeaux and C. Buil-Aranda. (2011, June) SPARQL 1.1 Federated Query. [Online]. http://www.w3.org/2009/sparql/docs/fed/service
[13] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. (2011, March) Describing Linked Datasets with the VoID Vocabulary. [Online]. http://www.w3.org/2001/sw/interest/void/
[14] (2011, August) What is Protégé-OWL? [Online]. http://protege.stanford.edu/overview/protege-owl.html
[15] J. Bao, L. Ding, and J. Hendler, "Knowledge Representation and Query in Semantic MediaWiki: A Formal Study," Tetherless World Constellation (RPI), New York, Technical Report 2008.