A Nested Graph Model for Visualizing RDF Data

Renzo Angles

Department of Computer Science, Universidad de Chile
Department of Computer Science, Universidad de Talca
rangles@utalca.cl

Abstract. This paper presents an abstract data model for visualizing RDF data based on the notion of nested graphs. Our study gives theoretical results that shows directions to enhance the representation and visualization of RDF data.

1 Introduction

The term *data exploration* refers to the process by which the user is able to visualize, browse and query the data. *Data visualization* involves selecting, transforming and representing abstract data in a form that facilitates human interaction and understanding.

RDF data is typically very large, highly interconnected, and heterogeneous without following a fixed schema [5]. Any technique for exploring RDF data should therefore be scalable; should support graph-based navigation; should be generic, not depending on a fixed schema; and allow exploration of the RDF descriptions without a-priori knowledge of its structure [14].

Current applications for exploring RDF data use different interfaces, each one having its advantages and disadvantages. For example, circle-and-arrow diagrams (e.g., IsaViz [15]) are intuitive and useful to show the graph structure of RDF, but they are not proper to visualize large datasets. In another hand, an *object-based* interface (e.g., Tabulator [1,8]) provides a view where the user is able to see the data as a nested structure of resources.

The objective of this paper is to show that underlying object-based interfaces, there is a powerful data model which needs to be studied formally to take advantage of its properties. In this sense, we analyze object-based interfaces. We define a general framework based on the notion of *nested graphs*, a concept that was introduced in the area of database modeling by the hypernode model [6]. A nested graph extends the plain structure of a graph to a nested structure, allowing a simple and flexible representation of nested complex objects. We study the information capacity of a nested graph (i.e., the set of objects that can be modeled by it) in terms of its plain representation called a *graphset*.

The contribution of this paper is the formal definition of an abstract data model for visualizing RDF data based on nested graphs. We show that RDF graphs, nested graphs and graphsets are three equivalent representations for

RDF data. We propose the nested graph model as an abstract representation for object-based interfaces.

The paper is organized as follows. In Section 2 we present the RDF model and we study object-based interfaces. A framework for nested graphs and graphsets is defined in Section 3. The abstract model for visualizing RDF data is presented in Section 4. Finally, Section 5 presents some conclusions.

2 Preliminaries

2.1 The RDF Model [12,10]

Assume there are pairwise disjoint infinite sets **U**, **B**, **L** such that, **U** is the set of *RDF URI references*, **L** is the set of *RDF literals*, and **B** is the set of *Blank nodes*. We denote by **T** the union $\mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$.A tuple $(v_1, v_2, v_3) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times \mathbf{T}$ is called an *RDF triple*, where v_1 is the *subject*, v_2 the *predicate*[1], and v_3 the *object*. A set of RDF triples is called an *RDF Graph*.

2.2 Exploring RDF data

Nowadays, there are many interfaces for browsing RDF data, each one having its advantages and disadvantages. Among these interfaces we can mention the following types:

- *Keyword Search:* It is suffices for simple information lookup, but not for higher search activities such as browsing and querying (e.g., Swoogle [17]).
- *Explicit queries:* It consists in using a query language for querying the data. It has the advantage that the language can be powerful enough to express any query, however writing queries is difficult and requires schema knowledge.
- *Graph Visualization:* It is the most basic visualization model and is based on circle-and-arrow diagrams (e.g. IsaViz [15]). It provides an intuitive and useful interface when trying to understand the structure of the data (a graph in the case of RDF). However it is not an appropriate way to look at data when a big number of nodes are present or for comparing objects of the same class. Moreover, graph visualization does not scale to large datasets.
- *Faceted Browsing:* In this model the information is faceted, that is, composed of orthogonal sets of categories. Facets allow the user to restrict the information space to be visualized and to find information without an a-priori knowledge of its schema. As example of application using facets is the Flamenco Search Interface [13].
- *Object-based:* In this kind of interface, the user is able to see the description of a resource (i.e., its outgoing and incoming properties) as being an object which encapsulates its information. The basic approach shows the data of a unique resource in each moment (e.g., Marbles [3,7], Gruff [11], BrownSauce [16] and DAML Viewer [9]). A more complex approach consists to show a nested structure of resources (e.g., The Tabulator [1,8] and Zitgist [2]). An example of the complex approach is presented in Figure 1.

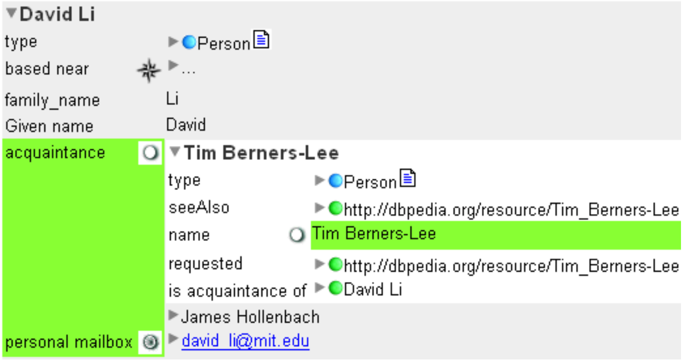

Fig. 1. The Tabulator: Generic data browser [1,8]

We concentrate our effort in to study object-based interfaces. The following lists some characteristics of this kind of interface.

- The resource's description includes outgoing and (possibly) incoming properties, which are presented as a list of property-value pairs (recall that an outgoing / incoming property comes from a statement where the resource occurs as the subject / object respectively). An incoming property is commonly represented by an expression "`is` *property* `of`". For example see property `acquaintance` in Figure 1.
- The nesting of resources is not necessarily hierarchical and cyclic references are possible. For example in Figure 1, the node `Tim Berners-Lee` contains the property `is acquaintance of` which introduces a cyclic reference to the root resource `David Li`.
- Encapsulation of information is allowed because a non-expanded resource hides its properties.
- There is redundant data, in that whenever an object is expanded more than one level, for and outer property we will also found a "dual inner property" in the opposite direction. As example, see the properties `acquaintance` and `is acquaintance of` in Figure 1.
- The exploration begins in a root node (a resource that acts as the container), and browsing is achieved by selecting a property-value which either shows a new description (in the basic approach) or expands a node (in the complex approach).
- Queries can be defined by either writing a query expression or constructing graphically a graph pattern in a query-by-example style (e.g. The Tabulator [8]).

From the above features, we consider that object-based interfaces are a good approach for visualizing RDF data. In fact, we will show that underlying these interfaces there is a powerful data model which deserves to be studied formally to take advantage of its properties and features.

[1] The predicate is also known as the *property* of the triple.

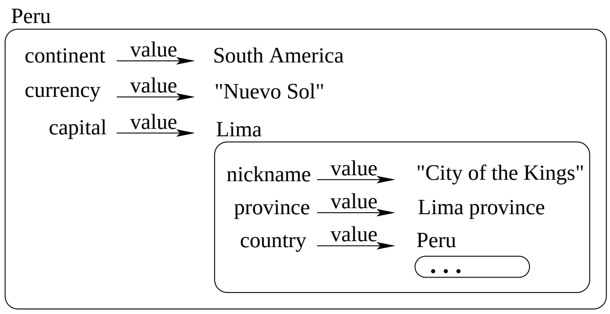

Fig. 2. An example of nested graph

3 Nested Graph Model

In this section, we will define an abstract data model based on the notion of *nested graphs*, a concept that was introduced in the area of graph database modeling by the Hypernode Model [6].

3.1 Nested Graphs

Definition 1. *(Nested Graph) Assume that Σ is an infinite set of labels. A nested graph is defined recursively as follows:*

(i) a triple (u, N, E) such that $u \in \Sigma$, $N \subset \Sigma$ and $E \subseteq N \times N \times N$ is a nested graph;
(ii) let $\boldsymbol{NG}$ be a set of nested graphs, then any triple (u, N, E) for which $u \in \Sigma$, $N \subset \Sigma \cup \boldsymbol{NG}$ and $E \subseteq N \times N \times N$ is also a nested graph.

Given a nested graph $G = (u, N, E)$, u is called the *name* of G, N is the set of *nodes* of G, and E is the set of *edges* of G. Given a node $n \in N$, if $n \in \Sigma$ then n is called a *primitive node* and $\text{name}(n) = n$; otherwise, if $n = (u', N', E') \in \mathbf{NG}$ then n is called a *complex node* and $\text{name}(n) = u'$.

Figure 2 presents graphically a nested graph named `Peru`. It contains primitive nodes (e.g. `continent` and `"Nuevo Sol"`), complex nodes (e.g. `Lima`) and edges (e.g. `continent` $\overset{value}{\rightarrow}$ `South America`). The complex node `Lima` is a nested graph which is nested inside the nested graph `Peru`.

Definition 2. *Let $G = (u, N, E)$ be a nested graph and $k > 1$. The operator $\text{nodes}^k(G)$ is defined recursively as follows:*

(i) $\text{nodes}^1(G) = N$; and
(ii) $\text{nodes}^k(G) = \text{nodes}^1(G) \cup \bigcup_{G' \in N \cap \boldsymbol{NG}} \text{nodes}^{k-1}(G')$.

Then, $\text{nodes}^k(G)$ extracts nodes from G by examining recursively each nested graph in G until reaching the k^{th} level of nesting.
Additionally, we define $\text{nodes}^*(G) = \bigcup_{j \geq 1} \text{nodes}^j(G)$.

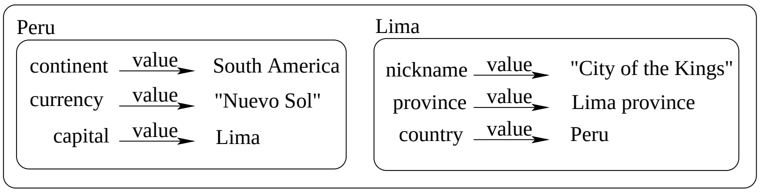

Fig. 3. An example of graphset which contains two plain graphs called `Peru` and `Lima` respectively.

A nested graph G is *cyclic* if $G \in \text{nodes}^*(G)$ or if there exists $G' \in \text{nodes}^*(G)$ such that G' is cyclic. If G is not cyclic then it is a *hierarchical* nested graph. A node n is *encapsulated* in G if $n \in \text{nodes}^*(G)$. For example, the nested graph of Figure 2 is cyclic because the nested graph `Peru` occurs as node in the nested graph `Lima`, i.e., the nested graph `Peru` is encapsulated in itself.

3.2 Graphsets

In this section we present the notion of *graphset*, a plain (or unnested) representation for a nested graph. The complexity of working with a nested structure is usually reduced by transforming it to a plain structure. In this sense, we define additional operators and properties for nested graphs in terms of graphsets.

Definition 3. *(Plain Graph) A* Plain Graph *is a nested graph* (u, N, E) *satisfying that* $N \subset \Sigma$*, i.e., a plain graph has no nested graphs as nodes.*

Given two plain graphs $G_1 = (u_1, N_1, E_1)$ and $G_2 = (u_2, N_2, E_2)$, we say that G_1 is a subgraph of G_2, denoted $G_1 \subseteq G_2$, iff $N_1 \subseteq N_2$ and $E_1 \subseteq E_2$. We say that G_1 and G_2 are *isomorphic*, denoted $G_1 \approx G_2$, if and only if $G_1 \subseteq G_2$ and $G_2 \subseteq G_1$. Additionally, consider the following operations between G_1 and G_2:

Union: $G_1 \cup G_2 = (u_3, N_1 \cup N_2, E_1 \cup E_2)$
Intersection: $G_1 \cap G_2 = (u_3, N_1 \cap N_2, E_1 \cap E_2)$
Difference: $G_1 - G_2 = (u_3, N_1 \setminus N_2, \{(n_1, n_2, n_3) \in N_1 \mid n_i \in N_1 \setminus N_2\})$

where $u_3 \in \Sigma$ is a fresh label. If $\text{name}(G_1) = \text{name}(G_2)$ then $u_3 = \text{name}(G_1)$.

Definition 4. *(Graphset) A* Graphset S *is a set of plain graphs satisfying that, for each two plain graphs* G_1 *and* G_2 *in* S*,* $\text{name}(G_1) \neq \text{name}(G_2)$ *, i.e., a label* $u \in \Sigma$ *identifies at most one plain graph in* S*. We denote by* ***GS*** *the set of graphsets.*

Figure 3 shows an example of a graphset.

Given two graphsets S_1 and S_2, we say that S_1 is a *sub-graphset* of S_2, denoted $S_1 \subseteq S_2$, if and only if for each $G_1 \in S_1$ there exists $G_2 \in S_2$ satisfying that $\text{name}(G_1) = \text{name}(G_2)$ and $G_1 \subseteq G_2$. Then, S_1 and S_2 are *equal*, denoted $S_1 = S_2$, if and only if $S_1 \subseteq S_2$ and $S_2 \subseteq S_1$.

Additionally, define the *maximal-intersection* and *union* between two graphsets S_1 and S_2:

$$S_1 \sqcap S_2 = \{G_1 \cup G_2 \mid G_1 \in S_1, G_2 \in S_2 \text{ and } \mathrm{name}(G_1) = \mathrm{name}(G_2)\}$$
$$S_1 \cup S_2 = (S_1 \sqcap S_2) \cup \{G_1 \in S_1 \mid \text{for all } G_2 \in S_2, \mathrm{name}(G_1) \neq \mathrm{name}(G_2)\}$$
$$\cup \; \{G_2 \in S_2 \mid \text{for all } G_1 \in S_1, \mathrm{name}(G_2) \neq \mathrm{name}(G_1)\}$$

3.3 Information Capacity of Nested Graphs

The *information capacity* of a representation is given by the set of objects modeled by such representation. Additionally, two complex object types are absolutely equivalent if and only if they can both be reduced to a normal form complex object types, which is based on some natural restructuring operators [4]. In this direction, the representation capacity of a nested graph will be defined in terms of graphsets.

First, we introduce operations for transforming nested graphs into graphsets.

Definition 5. *(Flattening of Nested Graphs) Let $G = (u, N, E)$ be a nested graph. We define the operators* flat *and* flat* *as follows:*

- $\mathrm{flat}(G) = (u, N', E')$ *where* $N' = \{\mathrm{name}(n) \mid n \in N\}$ *and* $E' = \{(\mathrm{name}(n_1), \mathrm{name}(n_2), \mathrm{name}(n_3)) \mid (n_1, n_2, n_3) \in E\}$
- $\mathrm{flat}^*(G) = \mathrm{flat}(G) \cup \{\mathrm{flat}^*(G') \mid G' \in N \cap \boldsymbol{NG}\}$

Then, $\mathrm{flat}(G)$ transforms the nested graph G into a plain graph by flattening its first level of nesting. Additionally, $\mathrm{flat}^*(G)$ flattens G and each nested graph G' in G until a fixpoint is reached. For example, Figure 3 shows the graphset obtained by flattening the nested graph in Figure 2.

In the opposite direction, a graphset S can be transformed into a set of nested graphs by expanding the structure of each plain graph in S, i.e, by replacing simple nodes by complex nodes. The following definition formalizes this notion:

Definition 6. *(Expansion of Graphsets) Given a plain graph G and a graphset S, we define function* $\mathrm{integrate}(G, S)$ *recursively as follows: for each primitive node $n \in \mathrm{nodes}^1(G)$, if there exists $G' \in S$ such that $\mathrm{name}(G') = n$, then replace n by $\mathrm{integrate}(G', S)$. This procedure is applied until a fixpoint is reached. Additionally, we define $\mathrm{integrate}(S) = \bigcup_{G \in S} \mathrm{integrate}(G, S)$.*

Note that $\mathrm{integrate}(S)$ returns a set of nested graphs, i.e, one for each plain graph in S. For example, if we expand the graphset in Figure 3, we will obtain the nested graph in Figure 2 plus the nested graph in Figure 4.

The following lemma defines the equivalence, in terms of representation, between nested graphs and graphsets.

Lemma 1. *Let $\boldsymbol{NG}$ be the set of nested graphs and $\boldsymbol{GS}$ be the set of graphsets. Then:*

(i) for each nested graph $G \in \boldsymbol{NG}$, $G \in \mathrm{integrate}^*(\mathrm{flat}^*(G))$; and
(ii) for each graph set $S \in \boldsymbol{GS}$, $S = \bigcup_{G \in \mathrm{integrate}^*(S)} \mathrm{flat}^*(G)$.

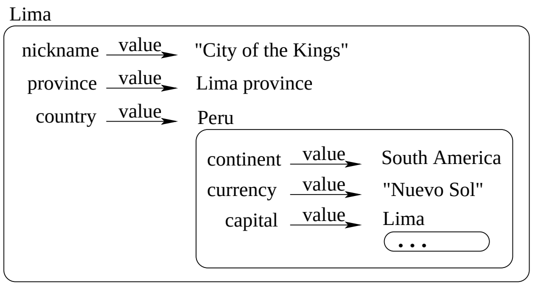

Fig. 4. A nested graph obtained by changing the structure of the nested graph presented in Figure 2

Consider a set of nested graphs M starting from a graphset S (i.e., $M = \text{integrate}^*(S)$), Then it holds that M models the same set of objects that S (because $S = \bigcup_{G \in \text{integrate}^*(S)} \text{flat}^*(G)$). However, in some cases it can occur that a minimal subset M' of M is enough for modeling all the data modeled by the graphset S (i.e., we can obtain S by flattening each nested node in M'). This minimal subset will be called the *core* of a set of nested graphs.

Definition 7. *Let M be a set of nested graphs. A* Core *of M is a minimal subset M' of M satisfying that $\bigcup_{G' \in M'} \text{flat}^*(G') = \bigcup_{G \in M} \text{flat}^*(G)$.*

For example, consider the nested graph presented in Figure 2 and call it G. If we flatten G, we have that $S = \text{flat}^*(G)$ is the graphset presented in Figure 3. Now, we have that $\text{integrate}^*(S)$ will contain the nested graphs of Figure 2 and Figure 4. Note that both nested graphs are a core of the graphset S, because any of them contains all the data modeled by S.

As we see, the core is not necessarily unique. Now, if the core of a set of nested graphs M is a single nested graph G, then G is called the *single-source* of M. In the context of graphsets, the above property introduces the notion of a *single-source graphset*.

Definition 8. *(Single-source graphset) A graphset S is called a* single-source graphset *if the core of S is unique.*

The notion of a core is useful for visualizing nested graphs. Consider the problem of selecting a good start point for navigation. If we use the core as a minimal set of navigation, we can reduce the number of visible or active nested graphs without losing data, such that all the data could be accessed by navigating through the nested graphs of the core. Clearly this problem could have a direct solution if we have a single-source graphset, then all the data can be accessed from a single nested graph.

Lemma 2. *Determining if a graphset S is single-source can be computed in polynomial time.*

Proof. Let S be a graphset. To construct a digraph $G = (N, E)$, where the set of nodes N is the set of names of the graphs in S, and there is an arc from u_1 to u_2 in E if the name u_2 occurs as node in the graph named u_1. We say that S is single-source if G is *connected* and there is a *unique spanning tree* for G (i.e., there is a tree which connects all the nodes together).

Finally, the *information capacity of a nested graph* G is defined by the graphset $\text{flat}^*(G)$. Additionally, the equivalence of two nested graphs is given by the equivalence in their information capacities, i.e., they can be reduced to the same graphset.

Definition 9. *(Equivalence of nested graphs) Two nested graphs* G_1 *and* G_2 *are* equivalent*, denoted* $G_1 \approx G_2$*, if and only if* $\text{flat}^*(G_1) = \text{flat}^*(G_2)$.

For example, the nested graphs presented in Figure 2 and Figure 4 are equivalent. Additionally, we say that G_1 is a *sub-nested graph* of G_2, denoted $G_1 \subseteq G_2$, if it holds that $\text{flat}^*(G_1) \subseteq \text{flat}^*(G_2)$.

4 Abstract Data Model for RDF Data

In this section, we define the abstract model for visualizing RDF data based on nested graphs.

Assume that $\mathbf{NG}^* \subset \mathbf{NG}$ is the set of nested graphs whose set of labels is given by $\mathbf{U} \cup \mathbf{L}$ (RDF URIs and labels), and each nested graph (u, N, E) in $\mathbf{NG}^*$ satisfies: (i) $u \in \mathbf{U}$; (ii) $N \subset \mathbf{U} \cup \mathbf{L} \cup \mathbf{NG}^*$; (iii) $E \subseteq ((N \setminus \mathbf{L}) \times \{+\} \times N) \cup ((N \setminus \mathbf{L}) \times \{-\} \times (N \setminus \mathbf{L}))$; and (iv) if $n \in N$ then n occurs in some edge of E. We denote by $\mathbf{GS}^*$ the set of graphsets having only plain graphs from $\mathbf{NG}^*$.

Given an RDF graph[2] G_{RDF}, the following algorithm returns a graphset $S \in \mathbf{GS}^*$ which models the same data as G_{RDF}:

Let S be an empty graphset
for each triple (v_1, v_2, v_3) in G_{RDF} **do**
  $G_1 = (v_1, N_1, E_1)$ where $N_1 = \{v_2, v_3\}$ and $E_1 = \{(v_2, +, v_3)\}$
  $S \leftarrow S \cup G_1$
  if $v_3 \in \mathbf{U}$ **then**
    $G_2 = (v_3, N_2, E_2)$ where $N_1 = \{v_1, v_2\}$ and $E_2 = \{(v_2, -, v_1)\}$
    $S \leftarrow S \cup G_2$
  end if
end for

It means that for each resource $u \in \mathbf{U}$ occurring either as subject or object in some triple of G_{RDF}, there exists a plain graph (u, N, E) where E contains: (i) an edge $(n_1, +, n_2)$ for each triple (u, n_1, n_2) where u occurs as the subject; and (ii) an edge $(n_1, -, n_2)$ for each triple (n_2, n_1, u) where u occurs as the object. In this sense, a plain graph (u, N, E) in S interprets a resource identified u whose *outgoing* and *incoming properties* are encapsulated by edges labeled "$+''$ and "$-''$ respectively.

[2] We restrict our study to RDF graphs having no blank nodes.

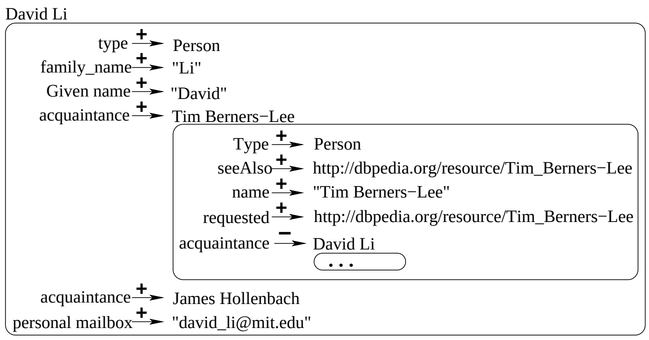

Fig. 5. An abstract representation to the interface in Figure 1.

From Lemma 1, we have that an RDF graph G_{RDF} can also be represented by the set of nested graphs integrate$^*(S)$, where S is the graphset constructed from G_{RDF} (as defined above). It allow us to present the following proposition.

Proposition 1. *The nested graph model is an abstract representation for Object-based interfaces.*

For example, consider the object-based interface presented in Figure 1. The expanded node `David Li` can be represented, from an abstract point of view, by the nested graph presented in Figure 5. If we compare both figures, we can see that an expanded node in the interface can be represented by a nested graph, and outgoing and incoming properties are represented by edges labeled "+" and "-" respectively. For example, the incoming link `is acquaintance of` $\rightarrow$ `David Li` of Figure 1 is represented by the edge `acquaintance`$\overset{-}{\rightarrow}$ `David Li` in Figure 5.

This abstract representation based on nested graphs enable us to study formally object-based interfaces. For example, the notion of single-source graphset can be used to select an object as the start point of navigation. A language for querying object-based interfaces could be based on the operators defined for nested graphs and graphsets. Exploration states can be stored, serialized or shared in the form of nested graphs or graphsets.

5 Conclusions

In this paper, we studied interfaces for visualizing RDF data, in particular object-based interfaces. We present a formal framework for representing and querying of nested graphs. We showed that an RDF graph can be modeled in terms of nested graphs and graphsets (a plain representation for nested graphs). We show that studying formal properties of RDF graphs viewed as nested graphs (or graphsets) is possible to optimize some visualization parameters. Further work includes to analyze properties of the model and the definition of a query language.

Acknowledgments. R. Angles was supported by Mecesup project No. UCH0109, and by FDI VAC650006, Faculty of Engineering, Universidad de Talca. The author wish to thank the reviewers for their comments.

References

1. The tabulator. http://www.w3.org/2005/ajar/tab.
2. Zitgist. http://zitgist.com/products/dataviewer/dataviewer.html, 2007.
3. Marbles linked data browser. http://beckr.org/marbles, 2008.
4. S. Abiteboul and R. Hull. Restructuring Hierarchical Database Objects. *Theoretical Computer Science*, 62(1-2):3–38, 1988.
5. R. Angles and C. Gutierrez. Querying RDF Data from a Graph Database Perspective. In *Proceedings of the 2nd European Semantic Web Conference (ESWC)*, number 3532 in LNCS, pages 346–360, 2005.
6. R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.
7. C. Becker and C. Bizer. Dbpedia mobile: A location-enabled linked data browser. In *1st Workshop about Linked Data on the Web (LDOW)*, April 2008.
8. T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and Analyzing linked data on the Semantic Web. In *Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06)*, 2006.
9. M. Dean and K. Barber. DAML Viewer. http://www.daml.org/viewer/, 2001.
10. P. Hayes. RDF Semantics. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/, February 2004.
11. F. Inc. Gruff: A Grapher-Based Triple-Store Browser. http://agraph.franz.com/gruff/, June 2008.
12. G. Klyne and J. Carroll. Resource Description Framework (RDF) Concepts and Abstract Syntax. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, February 2004.
13. The Flamenco Search Interface Project. http://flamenco.berkeley.edu/.
14. E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for RDF data. In *Proceedings of the 5th International Semantic Web Conference (ISWC)*, 2006.
15. E. Pietriga. IsaViz: A Visual Authoring Tool for RDF. http://www.w3.org/2001/11/IsaViz/.
16. D. Steer. Brownsauce RDF Browser. http://brownsauce.sourceforge.net/, 2002.
17. Swoogle. http://swoogle.umbc.edu/, 2004.