Representing, Querying and Transforming Social Networks with RDF/SPARQL[⋆]

Mauro San Martín[1,2] and Claudio Gutierrez[2]

[1] Departamento de Matemáticas, Universidad de La Serena
[2] Computer Science Department, Universidad de Chile
{msanmart,cgutierr}@dcc.uchile.cl

Abstract. As *social networks* are becoming ubiquitous on the Web, the Semantic Web goals indicate that it is critical to have a standard model allowing exchange, interoperability, transformation, and querying of social network data.

In this paper we show that RDF/SPARQL meet this *desiderata*. Building on developments of *social network analysis*, *graph databases* and *Semantic Web*, we present a social networks data model based on RDF, and a query and transformation language based on SPARQL meeting the above requirements. We study its expressive power and complexity showing that it behaves well, and present an illustrative prototype.

1 Introduction

The recording of social interactions over the Web, and the tagging and annotation of data are creating networks of data with the structure of what is classically known as *social networks* [1,2,3]. Examples include Wikipedia, del.icio.us, YouTube, DBLP, Flickr, Facebook, and the blogosphere. In all these services, each time a user sends a message, chooses a tag or annotates a resource, one or multiple relations are recorded. Researchers, developers, and users have realized that there is useful information in these underlying networks [1], consequently some sort of structural analysis is performed over them [3,4]. Many of the techniques used come from the established area of social networks analysis (SNA), that Breiger [5] defines SNA as the disciplined inquiry into the patterning of relations among social actors of different kinds and at different levels (see also [6,7,8]).

Nevertheless, the processing and managing of this huge amount of data is still an unsolved problem for the common user and developer. The problem is not only the format (today ranging from spreadsheets to files in proprietary formats), but also the means of querying, mixing and transforming such data with standard models. Furthermore, most data formats currently in use do not have explicit semantics nor support for provenance [1]. Advances have been done in the field of Social Network Analysis, e.g. Pajek [9], Ucinet [10], and *network* and *sna* packages of R [11,12]. These tools are focused in implementing techniques and algorithms for data previously prepared and formated to fit their

[⋆] This research was supported by Fondecyt 1070348, RDF Databases.

L. Aroyo et al. (Eds.): ESWC 2009, LNCS 5554, pp. 293–307, 2009.

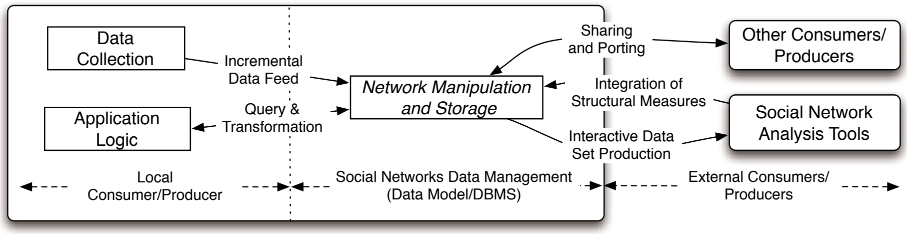
Data Collection
Incremental Data Feed
Application Logic
Query & Transformation
Network Manipulation and Storage
Sharing and Porting
Integration of Structural Measures
Interactive Data Set Production
Other Consumers/ Producers
Social Network Analysis Tools
Local Consumer/Producer
Social Networks Data Management (Data Model/DBMS)
External Consumers/ Producers

Fig. 1. *Data Management Needs for Social Networks.* Each social application is a consumer/producer of social networks, producing and/or collecting network data, and consuming data produced by other applications. The need for a standard representation and query language follows.

proprietary formats [10]. Another group of developments in these lines, proposals using RDF for covering different applications of the social Web, currently are focused only on the representational aspects of the problem (FOAF, XFN, RELATIONSHIP, etc.) This amounts to the definition of custom vocabularies and ontologies tailored for specific applications. The question remains how they may work together, and at what level of abstraction [13].

Thus a basic *desiderata* for managing social network data should include at least the following: 1) to be able to represent and store any kind of social network, including provenance information; 2) to be able to share and mix social networks (or parts of them) among users and applications; 3) to have a set of standard operations over these social networks, to query, transform and update them (a query language). We show in this paper that it is possible to fulfill such desiderata based on the experiences of social networks and database communities, and that of RDF/SPARQL.

The strategy to reach this goal is as follows. In the first place, we use the fact that all social networks of data have a stable common underlying model given by a characteristic practice of the SNA community, which is based in well developed methodologies and techniques (see Freeman [14] and [8, ch.1]). From here, and based on the Semantic Web practice, we abstract a model of data management for social networks based in consumers and producers of networks. It provides data management services to each consumer/producer: query, transform and update their own data; and between pair of consumers/producers: porting and sharing of data (see Fig. 1). For instance, SNA can be viewed as a consumer of social networks to be analyzed; annotators and users are producing and updating social networks; developers of applications are consuming and producing networks, etc. Finally, using the findings of database modeling [15], a graph structure emerges as the natural choice for such data model. In particular, RDF offers a standard model and SPARQL can be used as an off-the-shelf language. SPARQL fulfills almost all query and transformation requirements for social networks except for aggregation. Note also that relational model by itself does not intrinsically support a graph data structure, and consequently it is not straightforward to define relevant social network queries in SQL.

Contributions. In this paper we show that RDF/SPARQL are excellent data model/query language candidates for data processing social networks on the Web and cope with their basic requirements. First, we gather and study SN requirements of representation and storage, manipulation (querying, transforming and updating), and sharing/porting. Based on them, we define a conceptual graph data model for social networks. Second, we show that RDF/SPARQL fulfill these requirements and model, by mapping the conceptual data model data structure to RDF, and the conceptual query language to a composition of SPARQL and basic SQL queries (essentially used to provide aggregation which is currently missing in SPARQL). Third, we study the expressive power and the complexity of the query language showing that the model scales in the database sense. Finally, we provide a formal and practical bridge between Semantic Web framework and Social networks applications (which are key consumers/producers of social network data). We sketch a prototype implementation of a DBMS for social networks over RDF/SPARQL.

Related Work. There are three main areas related to our research:

1. Networks and Social Network Analysis. The closest development to our work are SNA software tools. One of the most popular and complete is Pajek [9], which offers an extensive set of analysis algorithms and visualization options. It has basic elements of data manipulation, for instance to filter out nodes and edges. The *R* statistical software also has packages to deal with network data [11]. In all these SNA tools, data storage is based on text files in various formats, which limits the data managing options available to the user, and in the long term sensibly rises the costs of network data curation. Low level storage restrictions are exposed to the user complicating unnecessarily the data manipulation. None of these tools provide the required data management services. There are many custom made applications that solve specific problems, for instance Klink et al. [16] use explicit social network data to improve DBLP navigation.

2. Network Data Models and Databases. Although there is an increasing awareness of the need of data management support in several fields which deal with networks[17,18,19], to the best of our knowledge there are no works discussing systematically data models for social networks. Jensen and Neville [20] propose Proximity, a data mining tool that supports statistical models over network data, which in turn uses MonetDB, a database management system (DBMS) able to store network data at low level, but lacking the appropriate level of abstraction for SNA use. Proximity uses a graph query language called QGraph [21]. Tsvetovat et al. [22] propose a specific application based on the *relational data model*, and also propose DyNetML [23] , an XML based format, to store rich social network data. No query or transformation issues are addressed in these works.

3. Semantic Web. In this area representational aspects present in our work have been addressed for specific applications with several standards, like for instance FOAF. How to integrate these standards and at which abstraction level remains an open question [13]. Mika [1, ch. 5] discusses several possible representation models, tools and standards from database, SNA, and Semantic Web communities, and reaches the same conclusion as us: current tools and data formats do not solve data representation and aggregation requirements. However, he proposes a solution based on ontologies and automated reasoning, whereas we propose a data model and the corresponding querying and transformation language. We show that having explicit metadata and a query language suffices for most data management requirements. Erétéo et al. [4] present a framework to provide "semantic aware social network analysis". Jung and Euzenat [2] propose a three layer model to represent and extract information from social networks. In terms of data management as previously defined, none of these models cope with the requirements. Producer/consumer has been addressed in other works, for instance, Polleres et al. propose XSPARQL [24] as a standard query language for lifting and lowering between XML and RDF data sources. Finally there are many works that extract, model, manipulate, and analyze social networks in the Semantic Web for specific applications, see for example: Aleman-Meza et al. [25], Kinsella et al. [26], and Mika [27].

The paper is organized as follows: In Section 2 the data structure of the data model is presented; Section 3 studies the corresponding query language, and discusses implementation issues; Section 4 presents conclusions.

2 Representing Social Networks over RDF

Social Networks data management requirements have come to the foreground with the advent of the "Social Web" [13], particularly due to the needs of interoperability of SN data (weak or inexistent in applications like Flickr, Delicious, etc.) Some applications like DBLP and FOAF have data available in several formats (particularly XML and RDF). Nevertheless, they have different vocabularies and model design, making interoperability non automatic. The core of these applications are their social network characteristic. Thus a natural source to look for requirements is the work that the SN community has done.

Example 1 (The KHTM Network). To illustrate the ideas we will use the *Krackhardt's High-tech Managers* (KHTM) network, a well known data set in the field of SNA. We use the version provided by Wasserman and Faust [7] of the data gathered by Krackhardt's in 1987 in a small manufacturing organization on the west coast of the U.S. The KHTM data set is a one mode network, with three kinds of relations (friendship, advice, reports_to) collected among the twenty one managers of the organization. For each manager are also recorded four attributes: age, tenure, level in corporate hierarchy, and the department in which the manager works (see Fig. 2). A total of 624 ties were recorded.

Requirements for a Social Networks Data Model. An appropriate departure point for SNA model requirements is Freeman's *maximal structure experiment* [8, ch.1]: one or more kinds of relations; one or more types or levels of social units; structures that change through time; sets of social units that grow or shrink; attributes of social units, and attributes that change. Two elements should be added to this model: attributes of relations, and n-ary relations, i.e. relations linking more than two actors [28].

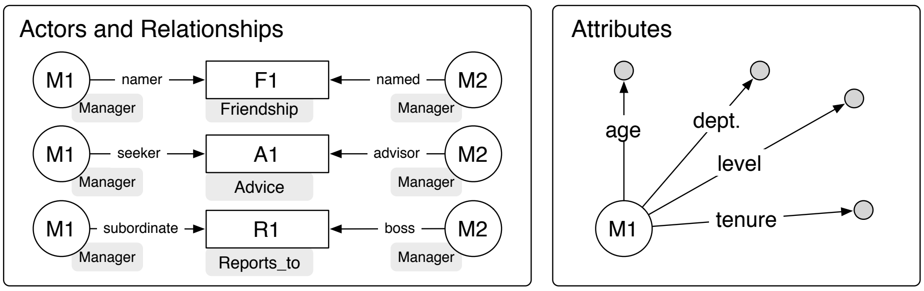

Fig. 2. *Schema of the KHTM data set.* Left hand side shows the three types of binary relations (rectangular nodes) between pairs of distinct managers (round nodes) in the original network data set. Note that relations are also represented as nodes, and arcs' labels represent the role that an actor perform in the relation. On the right hand side, attributes are represented as part of the graph.

In addition, the SN data model must fulfill the requirements that arise from current practices in the SN community (see [7], [6] and [29]), for instance, it must provide data management support to extract input data sets for popular SNA tools, e.g. Pajek [9], as a recurrent stage in the workflow depicted in the Fig. 1.

The implemented model must also address the problems that arise in the long term scientific data curation, e.g. provenance, reuse, and archiving [19]. Portability is another crucial requirement, particularly for the Semantic Web, from user profiles to more recently reuse of whole networks (e.g., in eScience). This is achieved via a standard underlying data structure and data manipulation language. Time is another important parameter to be included as desired. It can be added as metadata to actors or relations, although there are subtleties that have to be considered [30].

The requirements discussed above demand representations more elaborate than the simple graphs (or matrices) of the SNA classical modeling. This challenge is not exclusive of the domain of SN and has been explored extensively in other contexts [31,21,15]. From this background and trends in information exchange indicating that all the information should be in the same data structure and that it should support the addition of arbitrary metadata (e.g. provenance), basic requirements for such a model emerge. First, attributes values should be part of the graph, and second, relations should be represented as nodes instead of edges or arcs, allowing the seamlessly representation of n-ary relations and attributes on relations over the same data structure (see Fig. 2 and 3). Formally a *social network* can be represented by a special type of directed labeled graph.

Definition 1 (Social Network Data model). *A* social network *is a triple $S = (V, E, L)$ where (V, E) is a directed graph and L a set of labels (and labeling functions), specified as follows:*

- *The set of nodes $V = A \cup R \cup C$ is a disjoint union of the set A of actors, R of relations and C of attributes.*

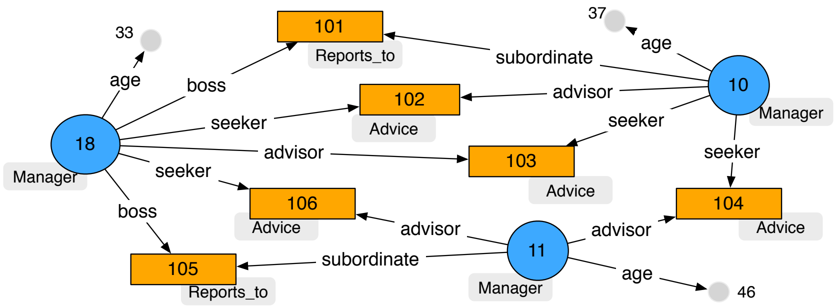

Fig. 3. *Subnetwork of KHTM data set: A view of a subnetwork of all relations among members of Department 3.* Actors are round nodes (black outline), and relations are square nodes. Attributes are small round nodes (grey outline). Labels indicate participation roles and meanings (attributes). To keep the diagram clean, only age attribute is shown and family belonging is depicted as a second label with grey background in each node (instead of the corresponding node and arc). Note that this is already an RDF graph. Arrows goes from subject to objects and are labeled with predicates.

Each of this set is partitioned in families (or types) as follows: Actor set $A = \bigcup_i A_i$, where A_i defines a family of actors. Relation set $R = \bigcup_i R_i$, where R_i defines a family of relations. Attribute set C is usually as well partitioned into different data types.

- *The set of edges $E = E_{AR} \cup E_{AC} \cup E_{RC}$ is a disjoint union of the following types of edges:*
  - *E_{AR} is a multiset of elements of $A \times R$, the* participation edges *(i.e. we allow multiple edges between an actor and a relation.)*
  - *$E_{AC} \subseteq A \times C$ and $E_{RC} \subseteq R \times C$ are the set of* meaning edges.
- *The set L of labels is the union of sets of labels for the different types of edges: L_P (participation) for labels on E_{AR}, and L_M (meanings) for E_{AC} and E_{RC}, with their corresponding labeling functions.*

This model can be naturally viewed as a set of triples in the style of RDF. The triples corresponding to the view in Fig. 3 are shown in the Fig. 4. The formal specification is as follows.

Definition 2 (Social Network Triple Representation). *Consider the vocabulary $\Sigma = A \cup R \cup C \cup \{isA, isR\} \cup L_P \cup L_M$. Then define the following:*

1. *List of Nodes and Family Belonging triple set:*
   $N_A \subseteq A \times \{isA\} \times \{A_1, \ldots, A_{m_A}\}$,
   $N_R \subseteq R \times \{isR\} \times \{R_1, \ldots, R_{m_R}\}$.
2. *Participation triple set:* $P \subseteq A \times L_P \times R$.
3. *Meanings (attributes) triple set:* $M \subseteq (A \cup R) \times L_M \times C$.

N_A
(10, isA, Manager)
(11, isA, Manager)
(18, isA, Manager)

N_R
(102, isR, Advice)
(103, isR, Advice)
(104, isR, Advice)
(106, isR, Advice)
(101, isR, Reports_to)
(105, isR, Reports_to)

P
(10, seeker, 103)
(10, seeker, 104)
(10, advisor, 102)
(10, subordinate, 101)
(11, advisor, 104)
(11, advisor, 106)
(11, subordinate, 105)
(18, advisor, 103)
(18, seeker, 102)
(18, seeker, 106)
(18, boss, 101)
(18, boss, 105)

M
(10, age, 37)
(11, age, 46)
(18, age, 33)

Fig. 4. *KHTM - Department 3 subnetwork (Fig. 3) as triples.* The separation in four groups (tables) of triples represents the social structure of the RDF graph. Tables P and M represent the node-edge-node triples shown in Fig. 3. Tables N_A and N_R represent the typing of each node (actors and relations respectively).

From the definitions above it is not difficult to show:

Lemma 1. *A social network can be represented by four sets of triples* (N_A, N_R, P, M).

In what follows, we will associate social networks with their triple representation. Regarding the RDF implementation, the vocabulary, that is, actors and relations and the sets L_P and L_M, are implemented as URIs, and attributes values as literals. The predicates isA and isR can be either implemented as predicates or replaced by the RDF keyword `rdf:type`. Then the set of triples is precisely the union of the sets N_A, N_R, P and M.

3 Transforming Social Networks with SPARQL

Query and Transformation Requirements. Querying and transforming social networks turns out to be a non-trivial task due to the intrinsic complexity of the networked data [15]. The good news is that the requirements are uniform, as shown by use cases collected from bibliographic sources [7,6,9], the papers from last three years of the Social Networks Journal and current software tools for SNA, particularly Pajek and its reference book *Exploratory Social Network Analysis with Pajek* [9] (see Table 1).

The range of queries that SN practitioners need are diverse, and can be roughly analyzed along two axis. First, queries that return values or measures of the whole input network or of some subnetwork (e.g. centrality, diameter, etc.) They fall into structural analysis and can be more properly treated by analysis-tools for final users. The second axis regards queries and transformations which output networks and constitutes properly the data management part of processing SN. (cf. Table 1). We focused on this latter group, which is the one currently missing from the applications and corresponds properly to data management needs discussed above.

Table 1. Archetypical social network data management operations from book *Exploratory Social Network Analysis with Pajek* [9], which also recurrently appear in literature (see [7,6,9])

Chapter Title	Use Case	Query Description (see Definition 3)
Looking for Social Structure	Directed to undirected Binary Relations (*arcs to edges* in Pajek)	In P_I both participation roles have different participation roles, in P_O both roles are the same.
	Remove relations (*remove edges* in Pajek)	Some relation nodes matched by P_I are not replicated in P_O
Attributes and Relations	Extract a subnetwork based on attributes	P_I include triples from M with fixed literals, or with filters on the corresponding variables.
	Group actors based on attributes (*shrink network* in Pajek)	P_I matches a subnetwork, including at least one actor attribute, P_O produces only one actor per value of the attribute. Aggregate functions may be required to count the size of the groups or summarize other attributes.
	Selective grouping of actors based on attributes (*contextual view* in Pajek)	A variant of the previous, here some portion of the subnetwork matched is not grouped
Cohesive Subgroups	Extract the subnetwork induced by cliques of size n	P_I is the desired clique for a fixed n, $P_O = P_I$
Sentiments and Friendship	Extract subnetwork by time	Time is represented by an attribute. M triples in P_I have fixed literals or filters on the corresponding variables.
Affiliations	Two-mode network to one-mode network	P_I represents a transitive relation, P_O eliminates the actor in the middle. It may need a function to generate ids for the new relation.
Center and Periphery	Group multiple binary relations (*remove multiple lines* in Pajek)	P_I selects all the required binary relations. P_O groups and typically counts them by pair of actors.
Brokers and Bridges	Extract egonetwork of an actor	P_I selects all relations and neighbors, and the relations involving those. $P_O = P_I$. Requires OPTIONAL or UNION. Each radius requires different set of patterns.
	Remove relations between groups (*remove lines between clusters* in Pajek)	Requires that the groups are represented as attributes first. P_I selects the network except the relations between actors of different groups.
Diffusion	Selective counting of neighbors	P_I selects the neighbors given certain condition. P_O requires aggregate functions.
	Operations between attribs. (*divide vectors* in Pajek)	Requires functions over literals in P_O.
	Change relation direction based on attributes	P_I selects the required network, P_O swaps labels of participation roles.
Prestige	Discretize an attribute	Requires functions over literals in P_O.
Ranking	Find triads by type	P_I is a three actor pattern, a triad. In each triad some relations are allowed and some are forbidden. Requires negation.
Genealogies and Citations	Loop removal	P_I selects all relations except those with arity equals to 1.

Query Language Definition. The ideal design is to have a set of simple primitives from where to compose by simple operations all needed queries. In our case, pattern matching fulfills this need. Informally speaking, the language consists of three parts: an *input pattern* P_I, which identifies (matches) all pieces of information needed and extracts the corresponding values; a *transformation* T of values, which updates and aggregates the values obtained in the previous step; and finally, an *output pattern* P_O, which defines the form of the data that will constitute the social network to be outputted.

Definition 3. *Let $\Sigma^X = \Sigma \cup X$ be the extension of the alphabet Σ with the set of variables X (disjoint of Σ).*

- *A* basic pattern *P is a social network over the vocabulary Σ^X.*
- *A* composite pattern *is defined by the following grammar:*

$$P ::= P\,\mathrm{AND}\,P \mid P\,\mathrm{OR}\,P \mid P\,\mathrm{MINUS}\,P \mid P(E),$$

where E is a Boolean expression of atoms of the form $(x\,\mathrm{op}\,y)$ or $(x\,\mathrm{op}\,c)$, where $x, y \in X$, c is a constant, and op *is in the set $\{=, \leq, \geq\}$.*

A query is a triple of the form $((P_I, X_I), T, (P_O, X_O))$, where

1. *(P_I, X_I) is a pattern over Σ^{X_I} (the* input *pattern). (Note that we write explicitly in the query the set of variables X_I occurring in the pattern.)*
2. *T is a* basic *SQL query from a table with attributes X_I which outputs a table with attributes X_O. A 'basic SQL' means here single table, no nesting; the selected attributes Aj could be functions of the attributes of X_I.*
3. *(P_O, X_O) is a basic pattern over Σ^{X_O} (the* output *pattern).*

Query Language Semantics. We will give an operational semantics described by the following query evaluation procedure using SPARQL and SQL. P_I and P_O will be SPARQL patterns, and P_O a basic graph pattern that may also contains filters. The procedure is composed by three stages:

Stage 1: Extraction. This corresponds to a SPARQL query that captures the values of the input pattern. The output is a solution sequence (a table).

```
SELECT X_I  FROM G  WHERE P_I
```

The result of this query is a table $R(X_I)$ containing all the bindings of the variables in X_I.

Stage 2: Transformation. The table obtained in the previous step is transformed by an SQL query (representing T) to output another table, whose attributes correspond to the values $f_i(X_I)$ for certain functions f_j.

```
SELECT f1(X_I) AS att1,...,fk(X_I) AS attk
FROM R(X_I) WHERE C_1
GROUP BY <...>  HAVING C_2
```

where C_1 and C_2 are conditions. From this table the final graph will be constructed in the next step. Note that this transformation is currently not expressible in SPARQL (even if there are no aggregate functions).

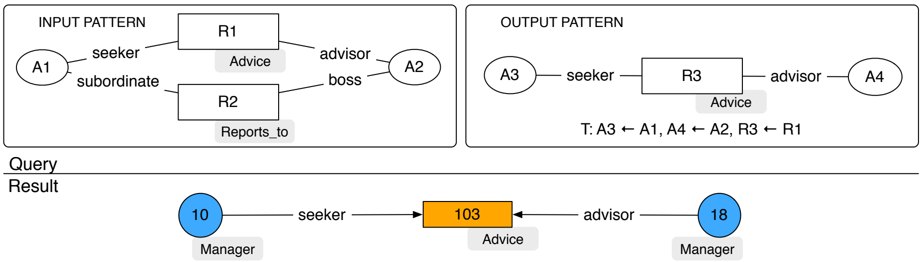

Fig. 5. *Query and Result: "Find all managers that seek advice with their bosses".* The query is composed by the input and output patterns, and the the transformation T. The result network when applied to network in Fig. 3 is depicted below.

Stage 3: Construction. Here the `CONSTRUCT` query form of SPARQL is used. The idea is to create an RDF graph which is the union of the instantiations of the output pattern P_O for each tuple of values of the table produced in the previous step.

Note that we cannot input a table to a SPARQL query. To handle this problem, we need to construct an auxiliary RDF graph H. Assuming $X_O = \{X_O^1, \ldots, X_O^k\}$, the specification of the final query is:

```
CONSTRUCT P_O  FROM H
WHERE {(att1, ?Tuple, ?X_O^1). ... (attk, ?Tuple, ?X_O^k)}
```

where the graph H is defined as follows: For each tuple $t_j = (v_{1j}, \ldots, v_{kj})$ in the output table in the previous step, the following set of triples T_j will be produced, $T_j = \bigcup_{i=1}^{k}\{(att_i, tuple_j, v_{ij})\}$. Then $H = \bigcup_j T_j$.

Example 2 (Simple Case). Figure 5 graphically represents the query: "extract the subnetwork of relations seeker-advisor, where the actor A, seeking advise with B, have also the relation subordinate-boss with B." In this query $X_I = \{A1, A2, R1, R2\}$, $P_I = \{(A1, seeker, R1), (A2, advisor, R1), (A1, subordinate, R2), (A2, boss, R2)\}$, $T = \{A3 \leftarrow A1,\ A4 \leftarrow A2,\ R3 \leftarrow R1\}$, $X_O = \{A3, R3, A4\}$, and $P_O = \{(A3, seeker, R3), (A4, advisor, R3)\}$. In this and the following examples we assume, for the sake of clarity, that the types of the actors and relations are implicit in the pattern (grey labels in Fig. 5), i.e., all the triples of the form (a, isA, X) and (r, isR, Y) –for a and r in the pattern– are implicit in the pattern and are carried as needed to the output following the transformation. The SPARQL query is:

```
Construct {?A1 KHTM:seeker ?R1. ?A2 KHTM:advisor ?R1}
From G
Where     {?A1 KHTM:seeker ?R1. ?A2 KHTM:advisor ?R1.
           ?A1 KHTM:subordinate ?R2. ?A2 KHTM:boss ?R2 }
```

If G is the network data set in Fig. 3, the result is the following set of triples (namespace `KHTM` omitted): $\{(10, seeker, 103), (18, advisor, 103)\}$ (See Fig. 5).

However, in the general case T is not a simple renaming mapping, the variables X_O are functions of X_I, and T may include aggregate functions. The following example cover this.

Example 3 (Filtering and Counting). "Extract the subnetwork of relations seeker-advisor for those advisors which have at least 2 advisees and add the number of advisees as an attribute to the advisors". G is again the data in Fig. 3.

Stage 1. First SPARQL query (SELECT)	Result: table R		
`SELECT ?A1, ?R1, ?A2`	A1	R1	A2
`FROM G`	10	104	11
`WHERE {?A1 KHTM:seeker ?R1.`	18	106	11
`       ?A2 KHTM:advisor ?R1}`	10	103	18
	18	102	10

Stage 2. SQL query	Result: table S			
`SELECT  A1 as att1, R1 as att2,`	att1	att2	att3	att4
`        A2 as att3, COUNT(*) as att4`	10	104	11	2
`FROM R`	18	106	11	2
`GROUP BY A2  HAVING COUNT(*) >= 2`				

Stage 3. Second SPARQL query (CONSTRUCT)

```
CONSTRUCT {?A1 KHTM:seeker ?R1. ?A2 KHTM:advisor ?R1.
           ?A2 KHTM:numAdv ?N}
FROM H
WHERE  {att1 ?T ?A1. att2 ?T ?R1. att3 ?T ?A2. att4 ?T ?N }
```

Where H is the set of triples { (att1, t1, 10), (att2, t1, 104), (att3, t1, 11), (att4, t1, 2), (att1, t2, 18), (att2, t2, 106), (att3, t2, 11), (att4, t2, 2) }, which is simply a coding of Table S as a set of triples.

Example 4 (Queries expressible in the language). Table 1 describes typical data management queries in SNA. All of them are expressible in the language (indications given in the table). Note that some of them require negation of patterns, which can be expressed in SPARQL, despite not being explicit in its syntax [32].

3.1 Expressive Power and Complexity of the language

Is the language rich enough for performing the usual queries in the area? Does the language have an efficient evaluation? Are two key questions we must answer to prove that the language works in general. We will see that while the latter can be answer in the positive, the answer for the former is a yes/no. Using the result in Angles [32] which shows that SPARQL has the same expressive power as Relational Algebra, it is not difficult to prove the following result:

Theorem 1 (Expressive Power). *The expressive power of the query language defined above in this section (see Definition 3) contains Relational Algebra and is contained in Relational Algebra plus aggregation.*

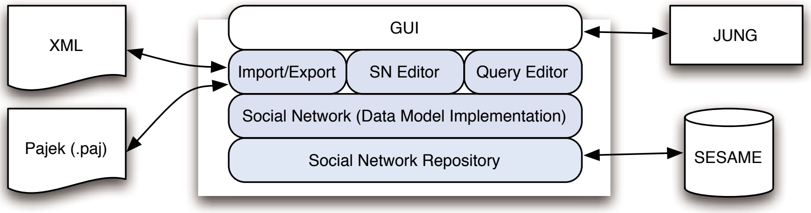
XML
Pajek (.paj)
GUI
Import/Export
SN Editor
Query Editor
Social Network (Data Model Implementation)
Social Network Repository
JUNG
SESAME

Fig. 6. *Prototype Architecture.* The figure shows the packages that forms the SNDB.

This result shows that the language is expressive enough to make all sensible transformations of networks (see Table 1).

On the negative aspects, it shows that "global" queries cannot be expressed in it. These are queries found in SNA practice ([7,6,9]) corresponding to analytical measures of the network as a *whole* (i.e. they are attributes of the entire network): density, balance theory (based in paths and cycles), betweenness centrality (depends on paths), exposure/ threshold, critical mass (diffusion), etc. Technically they are not expressible in relational algebra, see e.g. [33]. There is work both on trying to incorporate some features for retrieving paths in SPARQL [34,35], and also on retrieving paths in the spirit of SNA [4].

Complexity. The cost in time of the evaluation of a query of the language, based on the translation to relational algebra we provided is as follows:

1. First part: relational algebra. Essentially has the same cost as evaluating relational algebra. Note that the size of the output table $R_I(X_I)$ is, in the worst case, $O(|SN|^{X_I})$, where X_I is the number of variables of the input pattern of the query.
2. Second part, the SQL transformation T. The complexity of evaluating this transformation is linear in the number of rows of $R_I(X_I)$ once it is ordered by the `GROUP-BY` condition.
3. Third part: construction of the output network. The cost is asymptotically linear in the size of (P_O, X_O) times the size of the output of the table in (2).

3.2 A Prototype – Implementation Remarks

To illustrate the model, we built a proof of concept prototype of DBMS for the social network data model: SNDB. This prototype has two versions, one maps the conceptual model to the relational model, using as data back-end an RDBMS; and the other maps the model to RDF/SPARQL. The latter is described in this section.

The central module is *Social Network* (see Fig. 6) which implements the model and the query language, and handles main memory data structures needed to interact with upper layers. *SocialNetwork Repository* module works as an interface between *SocialNetwork* and the data services provider, in this case Sesame. Here model structures are translated to RDF, and model operations and queries are translated to Sesame update methods and SPARQL queries. *Network Import*

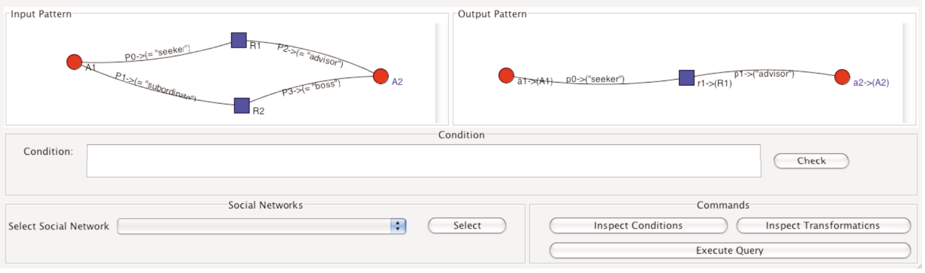

Fig. 7. *Protoype Query Editor.* Query editor snapshot showing query in *Example 2.*

and Export is composed by several modules, each one translating to and from an external format. Currently there are two modules implemented: XML, which handles the serialization and reading of social networks and queries to and from XML files (under a predefined schema); and Pajek, which imports and exports networks from and to Pajek files. The *Social Networks Editor* is a basic graphic social network editor which provides an interface to insert, delete and update individual social network components. Finally, the *Query Editor* (see Fig. 7) implements a graphical query editor, where it is also possible to execute the query over a given social network, and inspect the result.

The SNDB prototype was implemented using Java 1.5. It uses JUNG 1.7.6 network library to implement the GUI (analysis features are not used). Sesame 2.2.2 is used to implement the triple store. The choice of using SPARQL as the query language limits the queries that can be executed in the triple store to those that do not require aggregated functions. Figure 7 shows the query editor with the query in the *Example 2.*

The prototype has shown the advantages of having an abstract SNA model intuitive language, and a friendly query editor and processor. The problems we faced deal with the current lack of aggregation in SPARQL, and scalability, both at the visualization and response time level for big RDF graphs. There is plenty of optimizations to be done. For example, preliminary trials over a data set containing part of current DBLP (aprox. 100.000 records) in an average PC shows great flexibility for querying, however response times for complex queries are far from being satisfactory..

Overall, the prototype pinpoint several advantages of the approach: to interactively build networks and queries, to import and export data sets, and to execute queries and inspect results. Certainly it is mandatory to address the issues of optimization and aggregation in SPARQL.

4 Conclusions

We presented a data model and a query language for social networks data over RDF/SPARQL. This model provides a common data structure which support data interoperation, and the query language support the separation between data management tasks and application logic.

This query language formalizes requirements hence it may work as an implementation guide. These requirements where collected from SNA community practice, including published research, SNA software tools, and well known bibliographic sources, and are coincident with the main ideas suggested by other authors in the context of Semantic Web [1]. Finally we show that the RDF/SPARQL technology presents excellent features (except for the missing one of aggregation in SPARQL) for processing social networks.

Given that the implementation of the model may rest over the RDF/SPARQL stack, it can be almost seamlessly integrated with existing and new vocabularies, ontologies and their applications, leveraging the efforts to handle the enormous amount of heterogeneous social network data becoming available. Having a transformation and query language that has the flexibility and expressive power of relational algebra, clearly the main efforts should be directed to optimizations for big data sets.

References

1. Mika, P.: Social Networks and the Semantic Web. Semantic Web And Beyond Computing for Human Experience, vol. 5. Springer, Heidelberg (2007)
2. Jung, J.J., Euzenat, J.: Towards semantic social networks. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 267–280. Springer, Heidelberg (2007)
3. Finin, T., Ding, L., Zou, L.: Social networking on the semantic web. Learning Organization Journal Ubiquitous Business Intelligence (2005)
4. Erétéo, G., et al.: A state of the art on social network analysis and its applications on a semantic web. In: SDoW 2008 at ISWC 2008 (2008)
5. Breiger, R.L.: The analysis of social networks. In: Hardy, M., Bryman, A. (eds.) Handbook of Data Analysis, pp. 505–526. Sage Publications, Thousand Oaks (2004)
6. Scott, J.: Social Network Analysis, 2nd edn. SAGE Publications, Thousand Oaks (2000)
7. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Structural Analysis in the Social Sciences. Cambridge University Press, Cambridge (1994)
8. Freeman, L., Romney, A.K., White, D.R. (eds.): Research Methods in Social Network Analysis. Transaction Publishers (1992)
9. de Nooy, W., Mrvar, A., Batagelj, V.: Exploratory Social Network Analysis with Pajek. Cambridge University Press, Cambridge (2005)
10. Huisman, M., van Duijn, M.: Software for Social Network Analysis. In: [29], pp. 270–316
11. Handcock, M.S., Hunter, D.R., Butts, C.T., Goodreau, S.M., Morris, M.: Statnet: Software tools for the representation, visualization, analysis and simulation of network data. Journal of Statistical Software 24(1), 1–11 (2008)
12. Butts, C.T.: Network: A package for managing relational data in R. Journal of Statistical Software 24(2), 1–36 (2008)
13. Halpin, H.: Beyond walled gardens: Open standards for the social web. In: SDoW 2008 at ISWC 2008 (2008)
14. Freeman, L.: The Development of Social Network Analysis. Empirical Press (2004)

15. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Computing Surveys (CSUR) 40(1), 1–39 (2008)
16. Klink, S., Reuther, P., Weber, A., Walter, B., Ley, M.: Analysing social networks within bibliographical data. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 234–243. Springer, Heidelberg (2006)
17. Carley, K.M.: Linking capabilities to needs. In: [36], pp. 363–370
18. Jagadish, H.V., Olken, F.: Database management for life science research: Summary report of the workshop on data management for molecular and cell biology. OMICS 7(1), 131–137 (2003)
19. Gray, J., Liu, D., Nieto-Santisteban, M., Szalay, A., DeWitt, D., Heber, G.: Scientific data management in the coming decade. SIGMOD Record 34(4), 34–41 (2005)
20. Jensen, D., Neville, J.: Data mining in social networks. In: [36], pp. 289–302
21. Blau, H., Immerman, N., Jensen, D.: A visual language for querying and updating graphs. CS Technical Report 2002-037, University of Massachusetts (2002)
22. Tsvetovat, M., Diesner, J., Carley, K.M.: Netintel: A database for manipulation of rich social network data. CMU-ISRI-04-135 (March 2005)
23. Tsvetovat, M., Reminga, J., Carley, K.M.: Dynetml: Interchange format for rich social network data. CMU-ISRI-04-105 (2004)
24. Polleres, A., Krennwallner, T., Lopes, N., Kopecký, J., Drecker, S.: XSPARQL language specification (2009), http://xsparql.deri.org/spec/
25. Aleman-Meza, B., Nagarajan, M., Ramakrishnan, C., Ding, L., Kolari, P., Sheth, A.P., Arpinar, I., Joshi, A., Finin, T.: Semantic analytics on social networks: Experiences in addressing the problem of conflict of interest detection. In: WWW 2006, pp. 407–416 (2006)
26. Kinsella, S., Harth, A., Troussov, A., Sogrin, M., Judge, J., Hayes, C., Breslin, J.G.: Navigating and annotating semantically-enabled networks of people and associated objects. In: ASNA 2007 (2007)
27. Mika, P.: Social networks and the semantic web. In: Web Intelligence, pp. 285–291. IEEE Computer Society, Los Alamitos (2004)
28. Bonacich, P., Cody Holdren, A., Johnston, M.: Hyper-edges and multidimensional centrality. Social Networks 26, 189–203 (2004)
29. Carrington, P.J., Scott, J., Wasserman, S. (eds.): Models and Methods in Social Network Analysis. Cambridge (2005)
30. Gutierrez, C., Hurtado, C.A., Vaisman, A.: Introducing time into RDF. IEEE Transactions on Knowledge and Data Engineering 19(2), 207–218 (2007)
31. Guting, R.: Graphdb: modeling and querying graphs in databases. In: 20th VLDB Conference, pp. 297–308 (1994)
32. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: International Semantic Web Conference, ISWC 2008, pp. 114–129 (2008)
33. Courcelle, B.: Graph Rewriting: an Algebraic and Logic Approach. In: Handbook of Theoretical Computer Science. Formal Models and Sematics, vol. B, pp. 193–242. Elsevier and MIT Press (1990)
34. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: A navigational language for RDF. In: ISWC 2008, pp. 66–81 (2008)
35. Alkhateeb, F., Baget, J., Euzenat, J.: Constrained regular expressions in SPARQL. In: SWWS 2008, pp. 91–99 (2008)
36. Breiger, R., Carley, K., Pattison, P. (eds.): Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers. The National Academies Press, Washington (2003)