PECULIARITIES OF LINKED DATA PROCESSING IN SEMANTIC APPLICATIONS

Sergey Shcherbak, Ilona Galushka, Sergey Soloshich, Valeriy Zavgorodniy

***Abstract**: Nowadays linked data popularity increases along with information description in the form of RDF-triplets. Efficient implementation of users' interaction with these kinds of data requires studying communication procedures with triple stores. One of the main difficulties, that are currently unsolved, is the complexity of dynamic querying procedures. We try to deal with this issue by creating query patterns and decreasing query complexity. A unified search interface is developed, which enables visual querying the triple stores implemented through OpenLink Virtuoso universal server. Visual queries are automatically converted into SPARQL query language, which is used for accessing the triple stores. After the query is executed, a user gets the desired context with triplets according to constraints for predicates and objects. Also the formal model is developed, which mathematically describes linked data based on partially defined object schemas. Existing search model for distributed environments is improved. The developed practical implementation for medical institution is under stage of manufacturing application.*

***Keywords**: linked data, RDF data management, query pattern, triple store, semantic application.*

***ACM Classification Keywords**: E.2 DATA STORAGE REPRESENTATIONS (Linked representations), I.2.4 Knowledge Representation Formalisms and Methods (Representation languages)*

Introduction

The development of linked data standards (like "RDF Primer", "RDF/XML Syntax Specification" "SPARQL Protocol for RDF", "SPARQL Query Language for RDF" and other W3C standards) and their support by authoritative software development companies determine trends of evolution for the global network as a huge data and knowledge storage, which provides means for accessing structured data through specialized search interfaces [Ma, 2009]. Linked data are defined using Resource Description Framework (RDF) [Bizer, 2007]. This definition takes place in a form of triplets (subject – predicate – object) or quads (named graph – subject – predicate – object). For simplicity we shall use the term "triplet" to define triplets as well as quads, if it does not lead to contradictions.

RDF model assumes distributed storage of objects along with their schemas (if such schemas exist) on different web-servers. Such servers include integrated or external triple stores. SPARQL (recursive acronym for SPARQL Protocol and RDF Query Language) is used for accessing triple stores. If we draw analogy with relational database, SPARQL is alike to SQL in a way [DuCharme, 2011]. To use such a language of querying effectively, one should possess some special knowledge and skills. It neither encourages SPARQL popularization, nor increases triple store expansion.

Our goal is to increase the efficiency of search based on triple stores by means of decrease in complexity of SPARQL dynamic querying and due to implementation of query patterns for further search interface development. On our way to it, we analyze current approaches and possibilities of linked data in the context of search, modern user interface development techniques for linked data storages. In the next sections, formalization of linked data and search procedures are provided, architecture and unified search interface based on SPARQL are described.

Search interfaces for triple stores

Linked data search is characterized by determination of objects that belong to some application domain. In case of using such an approach, the content of documents is presented as a collection of objects grouped together by certain context [Allemang, 2009]. To determine object context we shall use a term “graph” that belongs to a quad. Information search assumes identification of named graph triplets and context identification that match user specific criterions or restrictions.

Implementation patterns should be designed for efficient search implementation and friendly user interfaces [Beck, 2008]. The patterns should provide visual querying to named graphs of triple stores without requiring any knowledge or skills in SPARQL from users. Fig. 1 shows architecture of a typical application on triple stores and SPARQL [Kalfoglou, 2009].

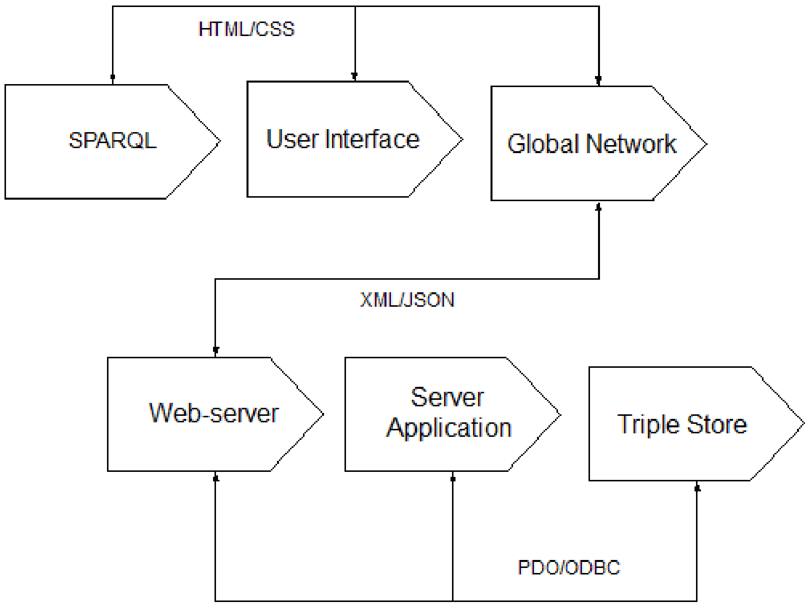


Figure 1. Architecture of a typical application with a triple store

User’s query on SPARQL is transmitted from web-application (with front end on HTML and CSS) to the triple store, where queries are processed using interfaces PDO and ODBC. Processing results are sent back to users in XML or JSON format. SPARQL queries are traditionally written by users. This requires some specialized knowledge about SPARQL and structure of objects residing in triple store. One of applications, that realize such a functionality, is ISQL web-interface from OpenLink used in Dbpedia. This application possesses universal data accessing techniques, and it is characterized by operation stability.

Let take a look at the proposed solution which defines user interaction with triple stores through SPARQL (see fig. 2). A user gains access to named graphs of triple store through visual SPARQL query builder. It gives an opportunity to achieve information about graph topology automatically. This information includes predicates and object data types. It can be used to create search request through determination of user restrictions (filters) on data returned from a triple store.

Denote user restrictions as a set of rules with conditions for objects returned from a triple store. Object type defines semantics of implemented comparison operation, i.e. the way query builder reacts on operation signs.

The following basic operations can be used to return objects: "$=$" (equality), "$>$" (more), and "$<$" (less). Equality sign for integer (xsd: integer) objects and float (xsd: float) objects, as well as equality sign for string objects (xsd: string), means that the analyzed objects are equal. Signs "$>$" and "$<$" for integer and float objects have their direct meaning also, whereas "$>$" and "$<$" signs for string objects mean that one object is a substring of another.

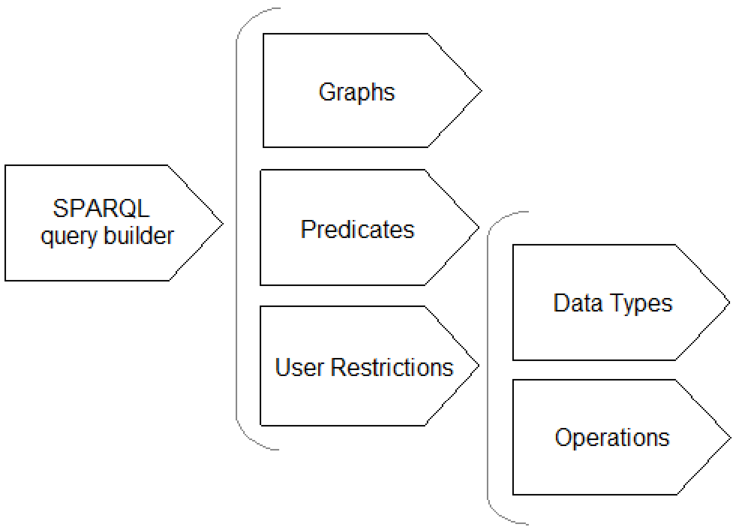


Figure 2. Schema of visual querying to triple stores

Consequently, a user may create rules to return content from a named graph of triple store. Typical SPARQL query to triple stores can be created automatically according to obtained rules. Practical aspects of such querying are observed in a section, and in the next section a formal presentation of triplets and linked data is given.

Formal model of linked data on partially defined schemas

Linked data can be presented in a following way:

$$t = < g, s, p, o >, \qquad (1)$$

where t is a triplet, g – named graph, s – subject, p – predicate, and o – object.

A collection of structures t, defined by formula (1), we denote as a triple store:

$$T = \{t_i\}, i = \overline{(1, n)}, \qquad (2)$$

where t_i is the i-th triplet and n – the number of triplets in a store.

Context information is needed to be assumed for search operation. The context is considered to be identical for all t_i with the same g. The following formula can be used to define context:

$$G = \{g_j\}, j = \overline{(1, m)}, \qquad (3)$$

where G is a set of all contexts from the store, g_j – j-th context of triple store, and m – number of contexts from the store.

Associate each context with the collection of tree elements ($< s, p, o >$) from formula (1). Consequently, the context can be defined as follows:

$$\forall g \in G:\ g = < S, P, O >, \tag{4}$$

where S is a set of subjects, P – a set of predicates, O – a set of objects.

An object can be defined as follows:

$$\forall o \in O:\ o = < T, L, V >, \tag{5}$$

where T – data type, L –language of presentation, V – value.

Assuming linked data peculiarities (namely optionality to define schemas, data types, languages used for object value presentation), we consider object elements optional, and object schema (according to formula (5)) is considered as partially defined by formulas (1)–(4).

To implement search tasks on G, we shall modify formula (4) by adding new element F for a function set. These functions may be performed on sets of context elements G.

$$\forall g \in G:\ g = < S, P, O, F >, \tag{6}$$

Practical aspects of function set implementation on SPARQL are described below. If it does not lead to contradictions, terms “context” and “named graph” are further used as synonyms.

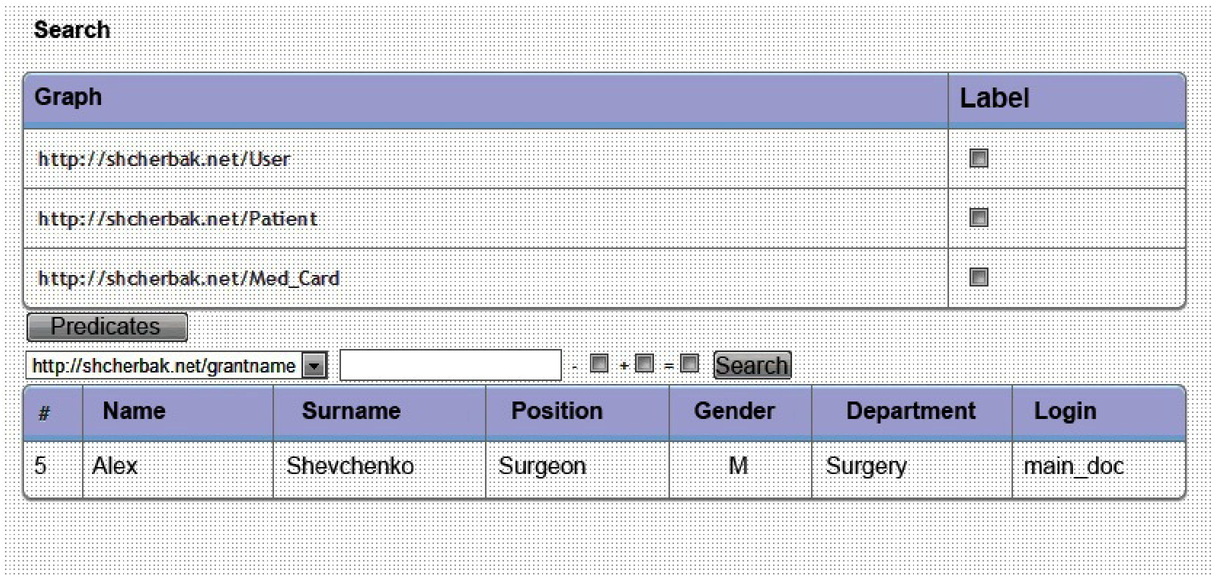
Practical aspects of search interface implementation

Typical application with a triple store (showed in fig. 1) and visual querying (showed in fig. 2) assume several actions to be performed on SPARQL for dynamic return of named graph topology. Such actions include: returning a list of named graphs from a triple store, returning a list of named graph predicates from a triple store, returning predicate data type. These SPARQL queries may be more complicated, for instance queries that return some named graph or group, which name equals (or partially corresponds) to a predefined criterion. Language tags can also be used as user restrictions. In this case, a filter can be set to return objects from a graph in Russian. Several restrictions can be used simultaneously to search for an object [DuCharme, 2011].

The proposed example of user interface is interesting as ontology integration on the user interface layer is quite a novel field of research [Paulheim, 2011]. Fig. 3 shows search interface for the queries mentioned above. It is implemented on PHP and tested on OpenLink Virtuoso server. OpenLink Virtuoso is chosen as a triple store, reasoner, RDF generator and SPARQL endpoint. It supports direct mapping and many programming languages, including C, C++, Python, PHP, Java, Javascript, C#, ActionScript, Tcl, Perl, Ruby, Obj-C [Segaran, 2009, Hitzler, 2009].

The proposed search interface provides users with additional information that can ease query building and decrease time required for it. Query building procedure reduces to selection of named graphs (which are interesting for users) and setting restrictions on predicates. Search results are grouped according to the subject in a tabular style. Consequently, data are presented at a clients’ front end in a friendly way, and users may know nothing about SPARQL query existence.

Although RDF language gives an opportunity to create graph structures of arbitrary level of complexity [Powers, 2003], there are several restrictions for communication with a triple store through ODBC (Open Database Connectivity). Data in a triple store are structured according to object-oriented principles, i.e. named graph is a container (class) for a set of objects belonging to it. The objects are uniquely identified by a triplet subject. This restriction provides more possibilities for software developers that use relational databases. If we do not take these restrictions into account, the proposed method will work, but data generation sense will be changed. To consider object restrictions, named graphs based on typical SPARQL queries are proposed to be created.



Search

Graph	Label
http://shcherbak.net/User	▨
http://shcherbak.net/Patient	▨
http://shcherbak.net/Med_Card	▨

#	Name	Surname	Position	Gender	Department	Login
5	Alex	Shevchenko	Surgeon	M	Surgery	main_doc

Figure 3. User interface with search results in a tabular style

Conclusion

The proposed architecture and example of visual SPARQL querying implementation is oriented on fast input of queries to data stores and search quality perfection. It permits searching in conditions of partially defined object schemas. Linked data model is offered for partially defined schemas. Search model for distributed environments with partially defined schemas is extended. Technological recommendations are suggested for implementation of user interfaces to triple stores with their automatic generation. After a query is executed, a user obtains context with triplets that match to restrictions on predicates and objects set by this user.

Practical implementation is performed on PHP and tested on multi-model data server OpenLink Virtuoso. This server is selected because of several reasons. First of all, it is cross-platform and can be used for relational data management as well as RDF and XML data management, free text content management and full text indexing. Secondly, it supports lots of programming languages and semantic web technologies, and thirdly, it is available for free and commercial use. The proposed models and technologies are highly efficient in a sense of enterprise market appeal from the point of basic principles suggested in [Wood, 2010]. They are also interesting from the point of modern effective solutions for semantic applications.

Acknowledgements

The paper is published with partial support by the project ITHEA XXI of the ITHEA ISS (www.ithea.org) and the ADUIS (www.aduis.com.ua).

Bibliography

[Allemang, 2009] D. Allemang, J. Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann, 2009, 352 p.

[Beck, 2008] K. Beck. Implementation Patterns. Addison-Wesley, 2008, 157 p.

[Bizer, 2007] C. Bizer, R. Cyganiak, T. Gauß. The RDF Book Mashup: From Web APIs to a Web of Data. In: 3rd Workshop on Scripting for the Semantic Web, Vol. 248, 2007, 6 p.

[DuCharme, 2011] B. DuCharme. Learning SPARQL. O'ReillyMedia, 2011, 258 p.

[Hitzler, 2009] P. Hitzler, M. Krötzsch, S. Rudolph. Foundations of Semantic Web Technologies. Chapman and Hall/CRC, 2009, 456 p.

[Kalfoglou, 2009] Y. Kalfoglou. Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications. IGI Global, 2009, 350 p.

[Ma, 2009] Z. Ma, H. Wang. The Semantic Web for Knowledge and Data Management: Technologies and Practices. IGI Global, 2009, 367 p.

[Paulheim, 2011] H. Paulheim. Ontology-Based Application Integration. Springer, 2011, 270 p.

[Powers, 2003] S. Powers. Practical RDF. O'ReillyMedia, 2003, 352 p.

[Segaran, 2009] T. Segaran, C. Evans, J. Taylor. Programming the Semantic Web. O'Reilly Media, 2009, 302 p.

[Wood, 2010] D. Wood. Linking Enterprise Data. Springer, 2010, 291 p.

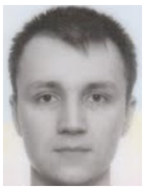
Authors' Information

***Ilona Galushka** – postgraduate student of Information and Control Systems department in Kremenchuk Mykhailo Ostrohradskyi National University, P.O. Box: 39600, Ukraine, Kremenchuk, Pershotravneva Street, 20; e-mail: anoli@gmail.com*

Major Fields of Scientific Research: Linked data, agent technologies

***Sergey Shcherbak** – PhD of Information and Control Systems department in Kremenchuk Mykhailo Ostrohradskyi National University, P.O. Box: 39600, Ukraine, Kremenchuk, Pershotravneva Street, 20; e-mail: ontolog@gmail.com*

Major Fields of Scientific Research: Semantic web, web services, RDF data management



***Sergey Soloshich** – postgraduate student of Information and Control Systems department in Kremenchuk Mykhailo Ostrohradskyi National University, P.O. Box: 39600, Ukraine, Kremenchuk, Pershotravneva Street, 20; e-mail: soloshich@gmail.com*

Major Fields of Scientific Research: Linked data, agent technologies

***Valeriy Zavgorodniy** – senior lecturer of System Software department in Dneprodzerzhinsk state technical university, P.O. Box: 51900, Ukraine, Dneprodzerzhinsk, Dneprostroevska Street, 2; e-mail: valera_ddtu@i.ua*

Major Fields of Scientific Research: Linked data, agent technologies