Acta Cybernetica **21** (2014) 353–366.

VOSD: A General-Purpose Virtual Observatory over Semantic Databases[*]

Gergő Gombos[†], Tamás Matuszka[†], Balázs Pinczel[†], Gábor Rácz[†], and Attila Kiss[†]

Abstract

E-Science relies heavily on manipulating massive amounts of data for research purposes. Researchers should be able to contribute their own data and methods, thus making their results accessible and reproducible by others worldwide. They need an environment which they can use anytime and anywhere to perform data-intensive computations. Virtual observatories serve this purpose. With the advance of the Semantic Web, more and more data is available in Resource Description Framework based databases. It is often desirable to have the ability to link data from local sources to these public data sets. We present a prototype system, which satisfies the requirements of a virtual observatory over semantic databases, such as user roles, data import, query execution, visualization, exporting result, etc. The system has special features which facilitate working with semantic data: visual query editor, use of ontologies, knowledge inference, querying remote endpoints, linking remote data with local data, extracting data from web pages.

Keywords: virtual observatory, semantic web, e-Science, data sharing, linked data

1 Introduction

E-Science is based on the interconnection of enormous amounts of data collected from various scientific fields. These massive data sets can be used for conducting researches, during which it is often desirable that researchers can share their own data and methods, thus making the results of the research accessible and reproducible by anyone. The idea of virtual observatories coming from Jim Gray and Alex S. Szalay serves this purpose [8]. A system like this expands the possibilities of combining data coming from various different instruments. Virtual observatories

[*]This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). We are grateful to Zsófia Mészáros and Zoltán Vincellér for helpful discussion and comments.

[†]Eötvös Loránd University, Budapest, Hungary, E-mail: {`ggombos, tomintt, vic, gabee33, kiss`}`@inf.elte.hu`

DOI: 10.14232/actacyb.21.3.2014.5

can also be used to teach and demonstrate the basic research principles of various scientific fields (for example, astronomy or computer science). The researchers must have access to these constantly growing amounts of data, in order to be able to use them in various research projects. Another important requirement is to be able to publish the results. The Internet provides an excellent opportunity to satisfy the criteria mentioned above [8]. The primary motivation for creating virtual observatories is to facilitate making new discoveries, and to provide a solution for carrying out data-intensive computations remotely. To access remote data, web services can be used [19].

The basic principles of science have been extended with a fourth paradigm. A thousand years ago, experimental results and observations defined science. In the last few hundred years, it shifted towards a theoretical approach, focusing on creating and generalizing models. During the last few decades, simulating complex phenomena with computers were becoming more and more common. Nowadays, researchers have to deal with large amounts of data, usually coming from sensors, telescopes, particle accelerators, etc. The data is processed using software solutions, and the extracted knowledge is stored in databases. Analyzing or visualizing the results needs further software support [7, 11].

A possible way to manage the data available on the Internet is to use the Semantic Web [4]. The Semantic Web aims for creating a"web of data": a large distributed knowledge base, which contains the information of the World Wide Web in a format which is directly interpretable by computers. The goal of this web of linked data is to allow better, more sensible methods for information search, and knowledge inference. To achieve this, the Semantic Web provides a data model and its query language. The data model called the Resource Description Framework (RDF) [14] uses a simple conceptual description of the information: we represent our knowledge as statements in the form of subject-predicate-object (or entity-attribute-value) triples. This way our data can be seen as a directed graph, where a statement is an edge labeled with the predicate, pointing from the subject's node to the object's node. The query language called SPARQL [17] formulates the queries as graph patterns, thus the query results can be calculated by matching the pattern against the data graph. Furthermore, there are numerous databases which contain theoretical and experimental results of various scientific experiments in the field of computer science, biology, chemistry, etc. There is a quite complex collection of these kinds of data maintained by the Linked Data Community [5]. This collection contains datasets and ontologies which are at least 1000 lines in length, and which contain links to each other.

In this paper, we present a prototype system, which fulfills the standard requirements of a virtual observatory, such as handling user roles, bulk loading data, answering queries, visualization, and storing results. In addition, we extended the system with special semantic technologies. We use the SPARQL language to formulate queries, aided by a visual SPARQL editor. Ontologies can be used to describe the hierarchy of complex conceptual systems, and to carry out knowledge inference. The system implements a tool, which helps its users to convert the data found on the web to the formats of the Semantic Web. We also provide a SPARQL endpoint to enable remote querying of the knowledge base. The query results can be exported to various common semantic data formats. We demonstrated the flexibility of the system by implementing two different database backends.

The structure of the paper is as follows. After the introductory Section 1, we outline preliminaries in Section 2. Afterwards, we present the high-level architecture of our virtual observatory in Section 3. Then, in Section 4, we describe the main functionality of the system. Then, we show some possible use cases of our system in Section 5, followed by the conclusion and our future plans in Section 6.

2 Preliminaries

As we mentioned in the introduction, the Semantic Web [4] provides various techniques to manage the data available on the Internet. This section gives insight into the basic concepts of Semantic Web that are necessary for understanding what our system is capable of and how it works. The main technologies that are used in our system are the following: Resource Description Framework (RDF), RDF Schema (RDFS), SPARQL query language, Web Ontology Language (OWL). In the formal discussion we follow the concepts and notations introduced in [16].

The Resource Description Framework is a description language, where the information is represented by RDF triples. Informally an RDF triple consists of a subject, a predicate, and an object; or alternatively it consists of an entity, a property, and the value of that property of the described entity. This representation form is similar to natural language sentences. For example the sentence *'Eötvös Loránd University is located in Budapest.'* can be translated into the triple *(Eötvös Loránd University, location, Budapest)*. Three kinds of terms are distinguished: IRIs represent entities (e.g. *http://dbpedia.org/resource/ELTE*) or relations (e.g. *http://dbpedia.org/ontology/location*); literals can only occur as value of a property; blank nodes are the terms that do not represent real world entities, they just help to construct complex values, for example, mail addresses which consist of multiple parts such as postal code, city, street and number. Below is the formal definition of RDF triples (Definition 1).

Definition 1. *Let I, B, and L (IRIs, Blank Nodes, Literals) be pairwise disjoint sets. An* RDF triple *is a $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$, where v_1 is the subject, v_2 is the predicate and v_3 is the object. A finite set of RDF triples is called an* RDF graph *or* RDF dataset*.*

The RDF Schema is a data-modeling vocabulary built on the top of RDF for defining concepts, properties and constraints which are essential for organizing the knowledge represented by triples. The Web Ontology Language also enables us to define concept and property hierarchies, however, it is a computational logic-based language. Therefore logical constraints and rules can be expressed in order to verify the consistency of that knowledge or to make implicit knowledge explicit. The formal definition of an ontology is presented in Definition 2, based on [20].

Definition 2. *An* ontology *is a structure $\mathcal{O} := (C, \leq_C, P, \sigma)$, where C and P are two disjoint sets. The elements of C and P are called* classes *and* properties, *respectively. A partial order $\leq_C$ on C is called class hierarchy and a function $\sigma \colon P \to C \times C$ is a signature of a property. For a property $p \in P$, its domain and its range can be defined in the following: $dom(p) := \pi_1(\sigma(p))$ and $range(p) := \pi_2(\sigma(p))$, where π is the projection operation. Let $c_1, c_2 \in C$ be two classes; if $c_1 \leq_C c_2$, then c_1 is a subclass of c_2 and c_2 is a superclass of c_1.*

SPARQL is a query language for retrieving and manipulating RDF data. It is an SQL-like declarative language; the queries are based on pattern matching, where the patterns are in the form of triples, though they can contain variables as well. Most of the keywords and their meanings are the same, such as *SELECT*, *WHERE*, *LIMIT*. However, there are some new keywords in SPARQL, for example, *OPTIONAL* means optional pattern matching, or *FILTER* that defines constraints for the variables. Definition 3 gives the abstract syntax of the filter conditions and Definition 4 presents the abstract syntax of the SPARQL expressions.

Definition 3. *Let V be the set of distinct variables over $(I \cup B \cup L)$. The variables are distinguished by a question mark. Let $?X, ?Y \in V$ be variables and $c, d \in (L \cup I)$ be a literal and an IRI constant, respectively. We define the filter conditions recursively as follows. The $?X = c$, $?X = ?Y$, $c = d$, $bound(?X)$, $isIRI(?X)$, $isLiteral(?X)$, and $isBlank(?X)$ are atomic filter conditions. Thereafter, if R_1, R_2 are filter conditions, then $\neg R_1$, $R_1 \wedge R_2$ and $R_1 \vee R_2$ are filter conditions as well.*

Definition 4. *A SPARQL expression is built up recursively in the following way:*

1. *the triple $t \in (I \cup V) \times (I \cup V) \times (L \cup I \cup V)$ is a SPARQL expression,*
2. *if Q_1, Q_2 are SPARQL expressions, and R is a filter condition, then Q_1 FILTER R, Q_1 UNION Q_2, Q_1 OPT Q_2, and Q_1 AND Q_2 are SPARQL expressions as well.*

The discussion of formal semantics of SPARQL is out of the scope of this paper. Set and multiset semantics are described in [16].

3 Architecture of the Virtual Observatory over Semantic Databases

The VOSD system was built using the Java EE platform. Figure 3 summarizes the main architectural elements of the system. As typical Java EE applications, VOSD consists of three major parts: a frontend, a business logic, and a database layer. Frontend is an interface for users, while backend contains the business logic which operates over the database.

As the Figure 3 shows, the basis of the system is an application server which is an Oracle WebLogic Application Server in our case. On the frontend side, our system uses the Java Server Faces (JSF) technology, which is a complete framework for Java EE. This framework contains some basic elements, such as text boxes, message bars or pageable dynamic tables. In addition, it can handle file upload, even multiple files at once, error messages, user interactions. As JSF pages are supported by web browsers, the clients of the system can be various devices, for example, mobile phones, tablets, laptops. Besides the JSF pages, the system is also available via a REST webservice, to access the uploaded semantic models. This makes it possible to build different kind of applications over the system as you can see in Section 5 or in [13].

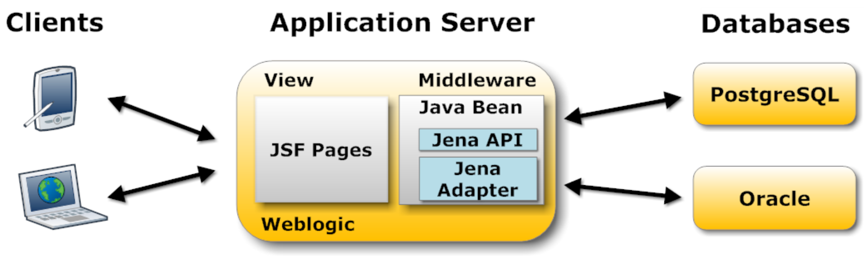


Figure 1: The architecture of the Virtual Observatory over Semantic Databases

On the backend side, two different databases are available by default. The first one is an Oracle 11g R2 database which supports the managing of semantic models and provides a Jena Adapter API for Java applications to use these features. Using the built-in semantic support, we can, for example, perform knowledge inference at the database level which can be much more faster than using a third-party tool. The second database is the PostgreSQL, which is a widely-used open-source relational database, however, it has no built-in semantic support. We chose this one to demonstrate how the already existing technologies can be applied to handle semantic data, and how efficient these two different solutions can be. On the top of PostrgeSQL, Jena is used to map RDF data model to the relational model.

4 Functionality

In this section, we present the main functions of our system. Users can upload their own data sets in various formats. Then they can browse and query the uploaded datasets. A visual query editor is provided to facilitate the construction of syntactically correct SPARQL queries and the queries can be saved and re-used. Two third-party visualizer tools are integrated into our system to help understand and explore the structure of data sets. In addition, we offer a tool, which is able to extract RDF triples from semi-structured web pages. Last but not least, to support the collaboration of researchers, our system provides user group management. Users can share their own data sets and their own saved queries within groups or they can make their work publicly available for every user.

4.1 Data Loading

There are two ways to load data into the system. One works by uploading a file containing the semantic data, the other requires a URL pointing to a resource on the Internet which contains the data. There are various RDF serialization formats for RDF which can be used with the system, such as RDF/XML, N3, Turtle, and N-Triples. The most wide-spread is the RDF/XML, which represents the RDF graph as an XML document. This format is easier for computers to read, since there are numerous tools available for processing and transforming XML. The other formats store the data using a more human-readable serialization. The simplest one is the N-Triples [1], which is simply the enumeration of the RDF triples (the edges of the RDF graph) separated with a dot. The Turtle [2] serialization allows more structures to simplify the expressions. For example, we can use prefix abbreviations to eliminate long, repeating IRIs, thus reducing the file size significantly. Furthermore, we have the option to group triples sharing the same subject, without repeating the common subject for all triples. This works similarly, if both the subject and the predicates are the same, and only the objects vary. This, too, helps to reduce the file size. Literals in Turtle can have language tags, or data type information added to them. Notation 3 [3] (or N3) allows further simplifications to make the serialization of complex statements easier.

4.2 Querying and Saving Results

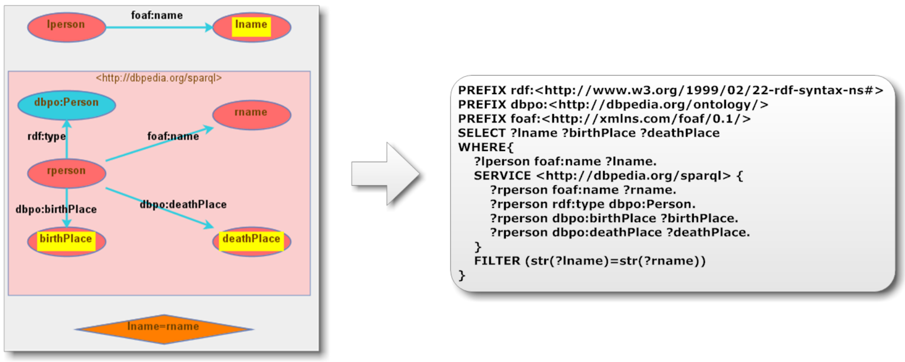
Another main function of the system is querying the already loaded data. The SPARQL [17] language is used to express queries over semantic data sets. The language is similar to the well-known SQL language. The *(*SELECT) clause defines a projection of the variables, the values for which we would like to see in the result set. The WHERE clause defines the criteria the data must satisfy in order to appear as a result. This is basically a graph pattern that has to match the data graph. The simplest queries contain only triples in the graph pattern. The *FILTER* clause lets us provide further filtering conditions for the nodes. For example, if we have numeric nodes, we can use arithmetic operators on them to restrict the values to a given range. If we have string nodes, we can filter for their values as well. IRIs, and string nodes can be filtered using regular expressions, too. By default, all edges in the graph pattern of the *WHERE* clause have to match the data. However, we have the option to define optional matching criteria with the *OPTIONAL* keyword. If parts of the graph pattern are optional, then we can have rows in the result set which satisfy only the non-optional parts, with null values for the variables appearing only in the optional parts. This is useful when some information is not given for all of our individuals. For example, if we have an address book with addresses for all contacts and phone numbers for some of them, we can ask the phone numbers in the optional part. Without the *OPTIONAL* keyword, we would only get the contacts with both an address and a phone number.

The advantage of the Semantic Web is that we can link our data with knowledge from other sources. In queries, the *SERVICE* keyword allows querying remote data sets. The keyword requires a URL to a SPARQL endpoint, and a graph pattern that has to match the remote data. The most well-known data set is the DBpedia [6], which contains a subset of the knowledge of Wikipedia in semantic form. Data sets linked with DBpedia can be found in the LOD cloud [5].

Another useful feature of the semantic web is knowledge inference, which lets us extract new information based on what we already know. Computing inferred data may take long time, thats why our system offers two options regarding inference. One option is to run the query using only the data already available to us as facts, or we can enable inference – meaning slower query execution. There are multiple ways to carry out inference. For example, we can use the relationship information given in ontologies to generate new information. Another option is to use user-specified rules. A rule consists of a head (a new triple holding the new information) and a body (a condition that has to be satisfied in order for the rule to activate). The simplest example is the grandparent relationship (if x is parent of y, and y is parent of z, then x is grandparent of z). We can save the query results using the already mentioned formats: RDF/XML, N3, TURTLE, and also CSV.

4.3 Visual SPARQL Editor

With the spreading of the Semantic Web technologies, using SPARQL becomes more and more inevitable, since this declarative language is the standard tool to express queries over RDF data sets. VisualQuery is a visual query editor program, which allows us to build a SPARQL query using graphs and supplementary forms.



```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpo:<http://dbpedia.org/ontology/>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?lname ?birthPlace ?deathPlace
WHERE{
   ?lperson foaf:name ?lname.
   SERVICE <http://dbpedia.org/sparql> {
      ?rperson foaf:name ?rname.
      ?rperson rdf:type dbpo:Person.
      ?rperson dbpo:birthPlace ?birthPlace.
      ?rperson dbpo:deathPlace ?deathPlace.
   }
   FILTER (str(?lname)=str(?rname))
}
```

Figure 2: An example SPARQL query both in graphic and textual form which finds additional information on DBpedia about locally stored famous people

Graphic representation has various advantages. Firstly, using this approach, it is easier to see and understand the relationship of the individual elements, thus, the meaning of the query can clearly be seen as demonstrated in Figure 2 where the graphic and textual representation of the same query are shown. Secondly, we can quickly and easily modify the components and parameters defining the query. This way, we can improve or refine the query step-by-step. Thirdly, because the visual representation is language-independent, the co-operative work of researchers speaking different languages is supported. Another advantage of the program is that it performs various checks during editing, which helps preventing syntactical errors, for example:

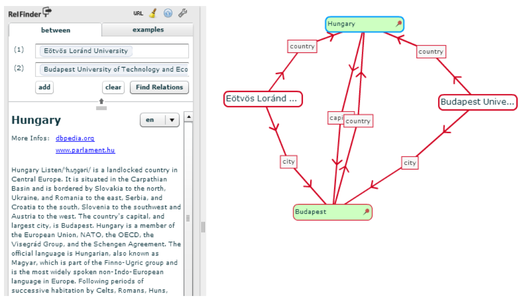
- literal nodes can not have outgoing edges – they can not be subjects in a triple,
- only variables or IRI nodes can be edges – blank nodes and literals can not,
- variables in the head of a CONSTRUCT-type query must appear at least once in the WHERE clause.

What makes this solution different from similar programs – like iSparql [15] or LuposDate [9] – is the distinction of visual elements by type, and the built-in checks based on this distinction.

4.4 Visualizations

We mentioned earlier that the semantic data can be seen as a directed graph. Subjects and objects are the nodes, and the predicates are the edges of the graph. Visualizing this graph helps us interpret the data. More graph visualization tools are available, and some can visualize the semantic data. We integrated two third-party visualizer tools into the system, that is seen on Figure 3.

One of them is Cytoscape Web [18], which allows us to display the semantic graph of locally stored models using various built-in layouts, such as a tree or circle. It is an open-source, interactive, customizable tool. It is a simple version of the Cytoscape for the web and it is reusable. The application uses Flex/Actionscript with JavaScript API, so rendering happens on the clients' computer. The user can visualize own models and the public models. However, the models can contain large amounts of data, and visualizing these models are resource-intensive, so we have to limit the edges.



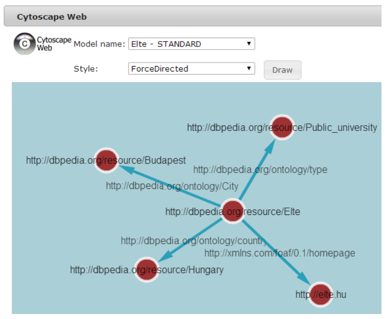
(1) Eötvös Loránd University
(2) Budapest University of Technology and Eco

Hungary

More Infos: dbpedia.org
www.parlament.hu

Hungary Listen/ˈhʌŋɡəri/ is a landlocked country in Central Europe. It is situated in the Carpathian Basin and is bordered by Slovakia to the north, Ukraine, and Romania to the east, Serbia, and Croatia to the south, Slovenia to the southwest and Austria to the west. The country's capital, and largest city, is Budapest. Hungary is a member of the European Union, NATO, the OECD, the Visegrád Group, and the Schengen Agreement. The official language is Hungarian, also known as Magyar, which is part of the Finno-Ugric group and is the most widely spoken non-Indo-European language in Europe. Following periods of successive habitation by Celts, Romans, Huns,

(a) RelFinder


Cytoscape Web
Cytoscape Web
Draw
http://dbpedia.org/resource/Public_university
http://dbpedia.org/resource/Budapest
http://dbpedia.org/ontology/type
http://dbpedia.org/ontology/City
http://dbpedia.org/resource/Elte
http://dbpedia.org/ontology/country
http://xmlns.com/foaf/0.1/homepage
http://dbpedia.org/resource/Hungary
http://elte.hu

Model name: Elte - STANDARD
Style: ForceDirected

(b) Cytoscape

Figure 3: Vizualization tools

Another visualization tool integrated into the system is RelFinder [10], which searches connections among IRIs. To find connections, it runs SPARQL queries on an endpoint. The relations among the IRIs can be paths via common predicates. RelFinder first finds the shortest path, and adds its nodes to the graph. After that it tries to find longer paths. We can specify the maximum depth of the search. The program uses Flex/ActionScript for the display that provides various tools to create animations. We configured this tool to work on the semantic data of the virtual observatory, and the users can search their own models.

4.5 Extracting Semantic Data from the Web

Nowadays, we can easily find all kinds of information using the web. There are numerous sites which specialize in collecting and organizing knowledge about one specific topic. For example, we can find websites collecting information about hardware components, reviews about movies, historical weather data, recipe collection, etc. These websites usually operate using a database of their own, and the web pages displayed to us are generated dynamically using the stored data. However, the databases are usually not using semantic technologies, moreover, they are often not public, so the only way for us to access the data is to visit the web pages. Fortunately, extracting data from the web pages does not always require complex text processing and text mining, because the consistent structure of the documents can be utilized to extract the information we are interested in. The structure is almost always consistent on all pages of a web site. For example, on a site collecting recipes, the structure can be the following: the name of the dish is always the title of a section, and it is followed by some additional information (always in the same order), such as the name of the uploader, the difficulty and the required time to prepare the meal. After this, we have a bullet-point list of the ingredients, and finally, there is a numbered list of the steps in the recipe. If we know this structure, we can utilize it to extract all recipes from all pages of the site.

To help users in extracting data from sites like these, we created a browser extension that allows them to define the structure using one example page of a web site. Based on the structure information created this way, our virtual observatory is capable to extract the required information from all pages that use the same document structure. The tool can be downloaded and installed from the web front end of our virtual observatory. Then, visiting the desired website, the user can mark the sections to be extracted using selection with the mouse. To extract information about all entities (e.g. all recipes) on the same page, the user only needs to annotate the first occurrence by binding variables to the parts of interest (e.g. the name of the dish and the list of ingredients). These variables can be used to formulate an RDF template, which – during the extraction phase – gets instantiated for each entity on all similar-structured pages of the website, with the appropriate extracted values in place of the variables.

The inner model for the annotation and the extraction is based on the DOM (Document Object Model) tree of the web page. When the user marks the first occurrence of an entity for extraction, the corresponding node in the DOM tree is marked as an anchor. Variables are defined relative to an anchor, by the (possibly empty) path that leads from the anchor to the node bound to the variable. During the extraction phase, occurrences of the repeating structure will be traversed by iterating over those sibling nodes of the anchor which have the same type (i.e. HTML tag). In each iteration, the appropriate variables are evaluated by following their defined path, staring from the current sibling. To handle repeating structure inside a repeating structure (e.g. ingredients as list items inside recipes), the anchors can be nested, resulting in a nested iteration during the extraction phase. Figure 4 shows an example model with two anchors (one of them nested), and two variables.

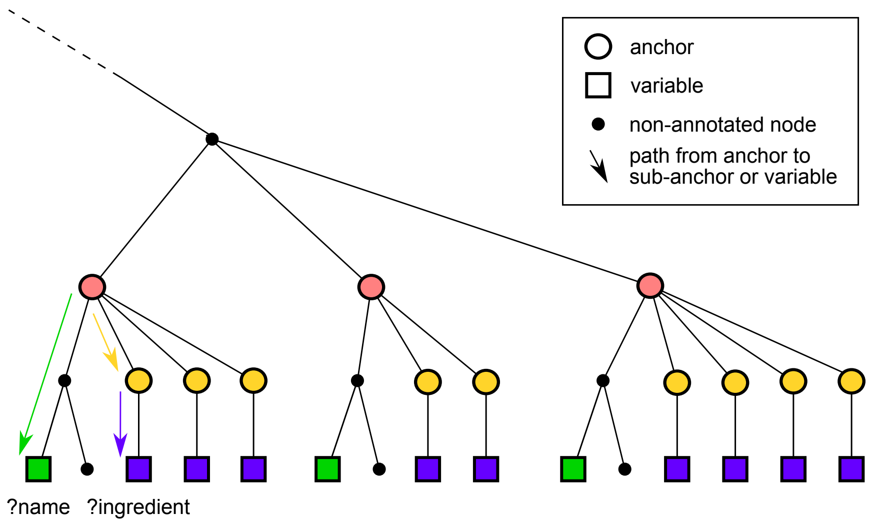


Figure 4: Part of an annotated DOM tree with two anchors and two variables.

The model described in the previous paragraph is automatically created and updated in the background, whenever the user marks an area for extraction or binds a variable to an element on the page. This way, no knowledge of DOM trees and paths are required to use the tool: the model for the data extraction can be created in a user-friendly way, using selection by mouse and a few clicks in the drop-down menu of the browser extension. The created model is saved as an XML file, which contains the structure information (anchors, variables, and paths between them) and the RDF template. The virtual observatory takes this file and a list of URLs as input, extracts the information from the specified web pages, and saves the extracted data to a semantic model.

4.6 Collaboration of Researchers

One of the most important purposes for virtual observatories is to collect information originating from various different sources, and to support their integration.

Our system allows users to upload their own data and share it with others. We applied a multi-level permission system based on user groups. Every user can create groups, and invite other users to them. This way, research groups can be organized. Then, we have two possibilities to share the models containing our data. We can make the model publicly available to every other user, or we can give right to one or more groups to access our model. While the first possibility gives read-only access, in the latter case the group members can have write rights, too. In this case, they can load their own data into the model.

It is also possible to publish queries. This can be useful in several cases: if other researchers would like to use our data, we can help their work by providing example queries, which illustrate the inner structure and relationships of the data. We can formulate basic queries, which can be further refined or specialized later.

5 Use Cases

In this section we describe two use cases that show the advantages of our system. The first one sums up how an application can be built on the top of data that is collected from heterogeneous sources, and how our system can be used to develop and manage such applications. The second example presents how we use the system in the education, how the functions and tools help the students to get familiar with the basic principles of the Semantic Web.

5.1 OCR Application

The first application is useful in the field of tourism. The main function of the program is to recognize text on street signs with OCR methods, based on pictures taken with mobile phones. Its purpose is to provide extra information about the famous people whose name can be found in the extracted texts. The extra information comes from various data sources converted to semantic format (Hungarian Electronic Library, various online encyclopedias [12]), joined with other public data sets (DBpedia, GeoNames). A user group created for this purpose allows the collaboration between the users. The group has access to the data sets described above. One member of the group was given the task to collect information about the famous people appearing in street names, and then upload them to a model. He then shared the model inside the group. Another member had the same task, but he had to use an online encyclopedia as the data source. He added his data to the shared model. Meanwhile, a third member worked on linking the data in the model to data available in DBPedia, using SPARQL queries. He stored the results in a new, local model, to make it faster to access. (His work was not influenced by the fact that in the meantime, new data has been added to the model.) He also published the queries and the new model to the group. The members of the group created a virtual model over the models mentioned. (A virtual model is not materialized, but it contains the union of the data found in other models, and it is supported by an index structure.) This step was important, because it allowed us to access the data as a single model. Then, using the REST API of our virtual observatory, we were able to run queries from a mobile application.

5.2 Use in Education

We use the virtual observatory during teaching the basic principles of the Semantic Web, within the Modern Databases course. The students of the course are added to a new group, and we share previously loaded models and queries with them. The models contain small data sets, so they could be viewed with the visualization tools, and the students could easily understand their structure. From week to week, they are introduced to the features of the SPARQL language, by solving typical tasks together. The new features can easily be demonstrated with the visual SPARQL editor, since the graphical representation speaks for itself. In some cases, the results of the exercises can be used in practical scenarios. For example, the family tree of a royal family can be created, if each student creates a model with the family tree of a selected king. During their work, they get to know the basic semantic serialization formats (RDF/XML, N3, etc.) and the results can be published to a common group.

6 Conclusion and Future Work

In the paper, we presented a prototype system, which fulfills the requirements of a virtual observatory, and helps the collaboration of researchers by letting them work using the same shared data and queries. We used the data model of the Semantic Web, thus the data sets in the virtual observatory can easily be linked to each other and to public data sets. We provided several features which can facilitate the use of the system, such as advanced data and query sharing, visual query building and editing, data visualization, and web data extraction. The system can run on top of any standard relational database system, but if the underlying database has some support for storing and handling semantic data (like Oracle databases), it can make use of those functions as well. We also presented real world use cases, where the existence of the system helped our work on other projects and in education. We are currently working on incorporating the ability to build and maintain bisimulation-based structure indexes, and utilize them in query evaluation to achieve better performance. Another feature in development is the visualization of SPARQL query plans. During further work, we would like to extend the system to be able to work using a Hadoop cluster as backend. In this solution, data storage and query execution would be distributed, thus the efficiency of the data-intensive computations would increase. Our other plans include enhanced visualization, such as the ability to plot geographic locations on a map, and to create charts and diagrams to help the better understanding of the data.

References

[1] Beckett, D. RDF 1.1 N-triples. W3C Recommendation, 2014. http://www.w3.org/TR/n-triples/

[2] Beckett, D., Berners-Lee, T., Prud'hommeaux, E. and Carothers, G. RDF 1.1 Turtle - Terse RDF Triple Language W3C Recommendation, 2014. http://www.w3.org/TR/turtle/

[3] Berners-Lee, T. and Connolly, D. Notation3 (N3): A readable RDF syntax W3c Team Submission, 2011. http://www.w3.org/TeamSubmission/n3/

[4] Berners-Lee, T., Hendler, J. and Lassila, O. The semantic web. Scientific American, 284(5): 28–37, 2001.

[5] Bizer, C., Heath, T. and Berners-Lee, T. Linked data-the story so far. International journal on semantic web and information systems 5(3): 1–22, 2009.

[6] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S. DBpedia – A crystallization point for the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3): 154–165, 2009.

[7] Brase, J. and Blümel., I. Information supply beyond text: non-textual information at the German National Library of Science and Technology (TIB) – challenges and planning. Interlending & Document Supply, 38(2): 108–117, 2010.

[8] Gray, J. and Szalay, A. The world-wide telescope. Communications of the ACM, 45(11): 50–55, 2002.

[9] Groppe, J., Groppe, S., Schleifer, A. and Linnemann, V. LuposDate: A semantic web database system. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2083–2084. ACM, 2009.

[10] Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T. Relfinder: Revealing relationships in rdf knowledge bases. In *Semantic Multimedia*, pages 182–187. Springer, 2009.

[11] Hey, T., Tansley, S. and Tolle, K. M. *The fourth paradigm: data-intensive scientific discovery.* Microsoft Research, Redmond, 2009.

[12] Hungarian Electronic Library http://mek.oszk.hu/indexeng.phtml

[13] Gombos, G., Matuszka, T., Pinczel, B., Rácz, G., Kiss, A. and Gaizer, T. A Semantic Browser for Enterprise Information Systems on Mobile Platform In *Proceedings of the 12th International Scientific Conference on Informatics'2013*, pages 246–251. 2013.

[14] Manola, F., Miller, E. and McBride, B. RDF 1.1 Primer. W3C Recommendation, 2014. http://www.w3.org/TR/rdf11-primer/

[15] OAT Interactive SPARQL (iSPARQL) Query Builder http://oat.openlinksw.com/isparql/index.html

[16] Pérez, J., Arenas, M. and Gutierrez, C. Semantics and Complexity of SPARQL. In *The Semantic Web-ISWC 2006*, pages 30–43. Springer, 2006.

[17] Prud'hommeaux, E. SPARQL 1.1 Query Language W3C Recommendation, 2013. http://www.w3.org/TR/sparql11-query/

[18] Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Ideker, T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research, 13(11): 2498–2504, 2003.

[19] Szalay, A. S., Budavári, T., Malik, T., Gray, J. and Thakar, A. R. Web services for the virtual observatory. In *Astronomical Telescopes and Instrumentation*, pages 124–132. International Society for Optics and Photonics, 2002.

[20] Volz, R., Kleb, J. and Mueller, W. Towards Ontology-based Disambiguation of Geographical Identifiers. In *I3*, 2007.