Explorator: a tool for exploring RDF data through direct manipulation.

Samur F. C. de Araújo
Catholic University of Rio de Janeiro
R. M. S. Vicente 225
Gávea, Rio de Janeiro, RJ, Brazil
+55 21 3527-1500
saraujo@inf.puc-rio.br

Daniel Schwabe
Catholic University of Rio de Janeiro
R. M. S. Vicente 225
Gávea, Rio de Janeiro, RJ, Brazil
+55 21 8241-4313
dschwabe@inf.puc-rio.br

ABSTRACT

In this paper we introduce Explorator, a tool for exploring the Semantic Web data by direct manipulation. Explorator implements a model of operations that is supported by a visual interface that enables the user, with minimal knowledge of RDF model, to explore an RDF database without a-priori knowledge of data domain. Consequently, it is well suited for tasks that involve information search, exploration and visualization.

Categories and Subject Descriptors

H.5.3 Web-based interaction; H.5.4 Hypertext/Hypermedia - Navigation, H.3.3 Information Search and Retrieval – search process, query formulation.

General Terms

Algorithms, Design, Experimentation, Human Factors, Languages, Theory, Verification.

Keywords

RDF, exploratory search, exploration, ontology, semantic web.

1. INTRODUCTION

As the volume of information on the Web increases considerably, we need better tools to help us discover and make sense of the available information, as well as to seek answers to specific questions we may have.

Currently, seeking information is a task that permeates most activities we develop in our day-to-day. Depending on the type of activity we perform, we use different strategies and tactics to search for information. In the web, these tactics are supported by computational tools such as keyword search, navigation and browsing [11]. But the process of seeking information is not simply finding it, we must keep in mind that the task of the user ranges from simply searching for a known item to activities such as knowledge acquisition, understanding of concepts, discovery, planning, transforming, etc. [11]

A more recent development has been the Semantic Web (SW), and the rapidly growing amount of semantically annotated data leads to the need to support not only for searching, but also for investigating and learning about a set of data without a-priori knowledge of its domain. This data is expressed in RDF[1], and is typically stored in very large interconnected databases, without a homogeneous schema. The exploration mechanisms currently available are not sufficient to accomplish the user tasks in the SW. Keyword search, e.g. Sindice[2], only addresses simple information lookup. Explicitly formulated queries, e.g. iSparql[3], requires schema and technical knowledge from the users. Semantic browsers, e.g. Tabulator [3], are not designed to explore huge datasets and semantic faceted browsing, e.g. BrowseRDF [12], is inefficient for fact-finding or known-item retrieval and some more complex exploratory tasks.

In this paper we will describe a model for representing information processing by users in exploratory tasks, and Explorator tool, which provides a browser interface supporting this model. Explorator is based on the metaphor of direct manipulation of information on the interface, with immediate feedback of user actions. The remainder of the paper is organized as follows. Section 2 defines more precisely the exploratory search itself; Section 3 presents the information processing model; describes Explorator tool and its interface; Section 4 we present some details of its implementation; Section 5 presents some conclusions and directions for further work.

2. EXPLORATORY SEARCH

In the hypertext field, we call information exploration the process of seeking, learning about and investigating a (potentially large) collection of information items through search, browsing or navigation, but not excluding other forms, in order to discover something new.

Research in the area called exploratory search [11] has tried to develop solutions that support information exploration. Exploratory search is applicable in situations where the user' task and the search environment have complex elements that require constant user interpretation during the exploration process. For example, how to support the user's search task when she is not familiar with the search domain, or she does not have sufficient knowledge about the domain to make a query; how to support the navigation in vast information spaces, or when the navigation, searching and browsing are not enough. In other words, how to

LDOW 2009, April 20-24, 2009, Madrid, Spain.

[1] RDF – Resource Description Framework

[2] http://sindice.com

[3] iSparql can be accessed at http://demo.openlinksw.com/isparql/

take into account all aspects [2, 7, 11] that influence the exploration process: the user's task, the user's context, the user's profile, the environment, the information provenance, etc.

Marchionini [11] made a distinction between exploratory search, lookup and search retrieval. According to him, exploratory search is based not only on lookup but also on investigation and learning. He argues that investigative search and learning search require more human iteration than a simple lookup, because these are exploratory processes that support tasks that require the cognitive and interpretative ability of user. These kinds of tasks are commonly found in the exploration of RDF databases, where the users need to identify classes and properties from the schema, in order to understand concepts, acquire knowledge and learn about the domain.

Berners-Lee et al. [3] argue that once the information sought is found, it may be necessary to analyze it. According to their description, exploration and analysis are distinct processes that are inter-related during the user's task. In our point of view, the process of exploration involves both finding a piece of information and investigating or learning about its domain, because it is guided by the need to perform a task. The cognitive process of analysis permeates the entire exploratory task, since while browsing, the user creates an expectation of what she will obtain, she sees what has been achieved and uses this information to guide her in the next step.

In order to provide to the user an exploratory search tool that supports learning and investigative search on SW, we focused on three fronts:

- Information search (how semantic data is found on the Semantic Web),
- Information usage (how semantic data is used on the Semantic Web),
- Information visualization (how semantic data is presented on the Semantic Web).

2.1 Information Search (in the SW)

Nowadays, we can access the SW data in three different manners: through a SPARQL Endpoint[4], through an URI, or by processing semantically annotated HTML pages (e.g. Microformats[5] or RDFa[6]). There are tools which can explore the SW directly, such as semantic web browsers, such as Tabulator [3], Disco[7], Zitgist data viewer[8], Marbles[9], ObjectViewer[10] and Openlink RDF Browser[11].

These tools all implement a similar exploration strategy, allowing the user to visualize an RDF sub-graph in a tabular fashion or in a more "visual" way (e.g., map views or timelines) when applicable. The sub-graph is obtained by dereferencing [4, 6] an URI and each tool uses a distinct approach for this. Tabulator is able to extract semantic annotations from HTML pages obtained from URIs that cannot be dereferenced as an RDF file, using GRDLL. In spite of distinct dereferencing processes being able to retrieve different amounts of information, the process itself does not improve the nature of tasks performed in these tools. In fact, the set of exploration tasks are limited to navigation between sub-graphs by clicking on the resources displayed in the interface and dereferencing the corresponding URIs.

Another way to access SW data is by querying a SPARQL Endpoint that receives a SPARQL[12] query and returns a set of RDF resources described in XML notation. There are a few tools that allow us to explore a SPARQL Endpoint. NITELIGHT [15] and iSPARQL[13] are Visual Query Systems (VQS) [5] which allow visual construction of SPARQL queries, differing mainly in the visual notation employed. It is understood that to use these tools the user must have a full comprehension of the underlying RDF schema and the query language syntax, therefore leading to a high cognitive load for newcomers and less experienced users. Tabulator also provides a way to query its data using SPARQL by providing an interface in which the user can formulate a query based on the selection of the elements of the RDF graph displayed on the interface. However, more complex queries need to be edited manually, exposing the user to some of the issues cited before.

Some tools address a different goal in the process of accessing SW data. Instead of focusing on access to RDF data, they focus on how to consume RDF data. Exhibit [9] is a lightweight structured data publishing tool that can be used to export small collection of RDF data. This tool accomplishes an important role on the SW, by publishing content from different sources on the Web.

Taking all this into consideration, we can see there are no tools adequate to explore the semantic web as a whole. Currently, the browsers and SPARQL query builders are addressing different goals, and were designed for different kinds of users. In order to provide a complete and integrated exploratory search mechanism to access the SW data, we are proposing Explorator.

2.2 Information Usage (in the SW)

The RDF model provides a format for data, information, and knowledge exchange. However, the repositories of data are scattered on the SW, which demand a unified mechanism to access them. Many information-intensive human tasks demand the manipulation of multiple pieces of information. In a SW exploration tool, at a low level, the objects manipulated are RDF data (resources, triples, literals, properties, etc) and queries. These are the information items being manipulated when using an RDF browser.

Consider the SW user looking for all papers mentioning another paper; or all paper authors' phone numbers. The user may encounter different data architectures while performing such tasks. For example, the information sought may be stored in multiple RDF files or in a single large RDF repository, and expressed in distinct vocabularies. It is crucial that any exploratory tool be able to consolidate the information to be accessed in an integrated way. The user should be able to merge information described in different vocabularies, at least by directly manipulating each piece of information. For example,

[4] http://www.w3.org/TR/rdf-sparql-protocol/

[5] http://microformats.org/

[6] http://www.w3.org/TR/xhtml-rdfa-primer/

[7] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/

[8] http://dataviewer.zitgist.com/

[9] http://beckr.org/marbles

[10] http://objectviewer.semwebcentral.org/

[11] http://demo.openlinksw.com/rdfbrowser/index.html

[12] http://www.w3.org/TR/rdf-sparql-query/

[13] iSparql can be accessed at http://demo.openlinksw.com/isparql/

suppose she is looking for all email addresses by dereferencing four different URIs, each one returning triples expressed in a distinct vocabulary. Even if she could see all the data together, she would not be able to manipulate this set of information to obtain a unique final set of email addresses, only by using current RDF browsers' functionality.

Some of these browsers, like Openlink RDF Browser, cache all RDF data during the user's navigation. Therefore, the user can treat pieces of information from different sources as coming from a unique repository. However, the user cannot issue a query on the results, which limits the kinds of tasks supported. For example, it is very difficult to obtain the homepage address for all people known to someone, as reported in their FOAF profile, by using one of the RDF Browsers mentioned earlier.

From the user's task point of view, exploring the SW involves asking questions and getting answers about the schema and instances. Obviously, understanding what is presented, what and how it can be manipulated is essential for the user to be able to formulate her question. Thus, querying is an important way for the user to increase her knowledge about the schema and data contained in an RDF repository. Direct SPARQL query formulation, which is allowed in some browsers, still imposes a higher mental load from the user, even for the more advanced. In addition, the user often does not have enough knowledge about the domain to formulate a query. As seen in Cartaci et al. [5], the raw use of query languages induces the user to make mistakes during writing, considerably increasing the time for query formulation and usually being far from the mental model that the user has of the reality.

Ding at al. [7] argue that the object of interest is not only the domain schema and instances, but also the source of data, which is an import piece of information in the exploratory process. In fact, when we are exploring several repositories, we could want to know from where each piece of information comes from. Marbles and Disco are examples of RDF browsers that track the provenance of the information, helping the user in judging its credibility.

In summary, current tools allow the user to manipulate raw RDF data and do not provide a user friendly way to ask question. The user is limited to visualizing the result as aggregate data. Any processing is done manually, and the user has a limited way to rearrange, group or filter the data, and process it further. We will discuss later how Explorator can be a step forward in SW data manipulation.

2.3 Information Visualization (in the SW)

A SW browser navigates along relationships between concepts. At each step of navigation, in this unknown and semi-structured (in the sense of schema-less) space, a set of RDF triples is displayed in the interface.

Browsers such as Disco, Marbles, Zitgist data viewer, Openlink RDF Viewer, represent RDF data in a tabular fashion. In Disco's interface, each triple is a line in a two columns table, the navigation is done by clicking on the resources displayed in the interface. Marbles does the same, and groups the values of properties that occur more than once for the same resource. In addition to the tabular presentation, the user has a more refined view of the triples being displayed. As in Disco, for each navigation step, the whole content is replaced by a new set of triples retrieved from the dereferenced URI.

Tabulator's more general view represents the information in a tree structure. As the user selects a resource in the interface, a new node is added to the tree, thus recording user's navigation process in the interface. The authors argue that it is comfortable for the user to see the information in a tree-oriented interface, due to familiarity with other sources of data are also represented in a hierarchical structure. The authors also proposed a model of views to be applied when the domain is known. A view oriented towards a specific domain improves the understanding of the instances being explored. For example, it is better to see geographic coordinates on a map than in a table.

From the user's task point of view, the representation of information helps its assimilation, but it does not expand the kinds of tasks that can be done. What we have observed so far is that without a proper model of exploration, involving well-defined operations, the user's exploration resumes to navigating between the nodes of an RDF graph, sequentially.

3. EXPLORATOR

Explorator [14]is an open-source exploratory search tool for RDF graphs, implemented in a direct manipulation interface metaphor. It implements a custom model of operations, and also provides a Query-by-example [18] interface. Additionally, it provides faceted navigation over any set obtained during the operations in the model that are exposed in the interface. It can be used to explore both a SPARQL endpoint as well as an RDF graph in the same way as "traditional" RDF browsers. Its general architecture is represented in the diagram below:

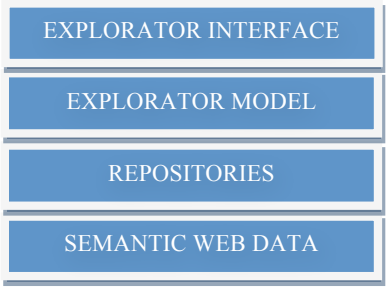


Figure 1. Explorator's general architecture.

At the most elementary level, the user's task resumes to dereferencing an URI or formulating and executing a SPARQL query against a SPARQL Endpoint. In Explorator, every SPARQL Endpoint is a repository, that can be enabled or disabled and can be manipulated individually or integrated into a single global source of RDF data. The dereferenced URIs are stored in a local SESAME [15] repository which can then be queried and manipulated as if it were a SPARQL Endpoint. In other words, the user always explores a federation of databases, containing SPARQL Endpoints and RDF triples obtained by dereferencing specific URIs.

[14] Explorator information, including a demo interface and the URL of the subversion repository can be accessed at http://.www.tecweb.inf.puc-rio.br/explorator

[15] http://www.openrdf.org/

The set of manipulation operations is limited to the operations defined in our processing information model which we will describe next.

3.1 The Information Processing Model

Exploring a set of information items in the SW is understood here as a process of transforming resources and triple by successive application of operations.

Our experience in Web application design methods [10, 16] has shown us that it useful to characterize the user information processing as set of manipulation operations, in what has been called "set based navigation" [14]. This view is also supported by more recent proposal such as Parallax[16]. Basically, the user is always processing (browsing) information items within a set of interest; if necessary, this set is further manipulated to either remove uninteresting elements or to add additional elements of interest.

Explorator's model is composed of two elements: the manipulated items and the manipulation operations. The items are primitive elements in the RDF model: triple, resources, literals, URIs, etc. The operations are grouped in two sets: set operations and search operations.

We will show in the following sub-sections that this model can encompass classical browsing, set-based navigation as found in SHDM [10], and faceted browsing, as well as keyword search.

3.1.1 Sets

The model manipulates two kinds of sets – sets of RDF triples and sets of RDF resources. When dealing with sets of RDF resources, the usual set operations, union, intersection and difference are available. Since RDF resources are treated as URIs, blank nodes will only be included if they are assigned URIs, as occurs in some data stores.

When operating on sets of triples, we interpret the set operations as applying to any of the triple components, namely, subjects (S), predicates (P) or objects (O). This is equivalent to projecting a set of triples along one of its three slots.

3.1.2 Search Operation

As previously stated, there are two ways to access the data in SW: dereferencing an URI or querying a SPARQL Endpoint. We define in our model general query operation, called SPO (S, P, O), to be applied to a SPARQL Endpoint. This operation allows the user to obtain a new set of interest, which can then be processed in the next step in the task.

The SPO operation has three parameters, all of which are sets: a set of subjects, predicates, and objects. This operation is a subset of general SPARQL queries, allowing the user to query an RDF database by providing an example pattern of the desired set of triples.

For example, the function `SPO(∅,∅,∅)` can be translated into the following SPARQL query:

```
SELECT ?s ?p ?o WHERE  {?s ?p ?o} .
```

For the following data:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

The query above should return all triples. On the other hand, the function `SPO(∅,{foaf:mbox}, ∅)` can be translated to:

```
SELECT ?s ?p ?o WHERE  { ?s ? p ?o. Filter (p =
  foaf:mbox)} .
```

This query returns all triples that have the property `foaf:mbox`.

Consider the more complex example of how this model could be used, to solve the task: "find all Russian lakes":

Let S be a function that returns all subjects from a set of triples.

SPO(

S(SPO(∅,{rdf:type}**,{**mondial:Lake**}))**,

{mondial:locatedIn},

{mondial:Russia}

)

The expression above returns all triples that have the property mondial:locateIn with value mondial:Russia.

It should be noted that, whereas these examples show single valued parameters, in general the parameters for SPO are sets.

3.1.3 Set Operations

The model allows the user to manipulate items of information within the RDF domain. Once the user has obtained a set of triples and resources, she can manipulate them individually, formulate new queries, or create new sets. To do so, the model supports the following set operations:

Let A be the set of all triples.

Union:

Given two sets M and N, each containing a triple, the union between M and N is the union of triples of M and N.

Intersection:

The intersection set I between M and N is the union of the triples in A such that the subject of the triples in I appear in triples in both M and N.

Difference:

The difference set D between M and N contains the triples in A such that their subjects appear in triples in M and do not appear in triples in N .

Note that, in this model, the result is always a set of triples, and the operations are always computed on the sets of subjects, predicates or objects of these triples.

3.2 Visualizing RDF data with Explorator

In existing RDF browsers, the data are expressed in one of the following metaphors: table, tree or graph. In our approach, the interface represents the elements of the underlying exploration model: resources, triples and sets.

[16] http://mqlx.com/~david/parallax/index.html

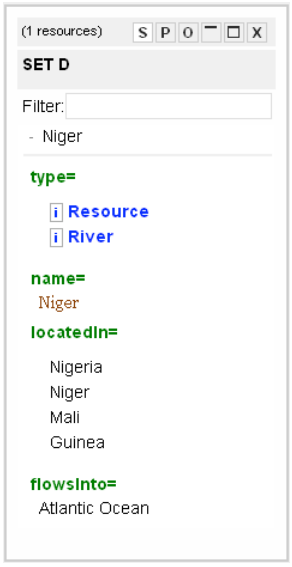


Figure 2. A set of triples displayed in Explorator. The subject is "Niger", the properties and values are listed under it.

Considering a generic exploration mechanism over the RDF model, the concept of triple, entity and resource are mixed. In Explorator's interface. The predicates and objects of the triples are nested and right aligned under the subject, thus evidencing the entity represented by the subject of the triple, as shown in the figure 2.

Explorator uses the following heuristic to render a resource (or URI) in the interface:

- If the resource has a label, name or title property, it renders its value.
- Otherwise the URI *localname* is rendered.

In this interface, each element can be manipulated individually. Sets of subjects, predicates and objects can be selected by the user and provided as parameters in the operations described in the model. Dereferencing an URI, or the result of an operation over the model always results in a new set in the interface. In this sense, Explorator incorporates elements of the Direct Manipulation paradigm [17], since the output of an operation may be used as input of another, as they are expressed in the same notation. Direct Manipulation is a user-system interaction paradigm that allows users to point at visual representations of objects and actions to carry out tasks rapidly and observe the results immediately. Explorator's interface follows this paradigm.

The interface has two main elements, the toolbar and the result sets. The toolbar has a menu giving access to repository configuration and additional functionalities; a search box; and a group of buttons representing the operations of the model.

Figure 3. Explorator toolbar.

The operations menu is divided in two groups, as shown in Figure 4. The first area (Fig. 4 - 1) has the set operations: To operate, the user must select the first set among the sets displayed, then click on the operation (union, intersection or difference), then select (click on) another set, and then click on '='. Specifically for union, the user can also click on multiple resources in the interface (ctrl-click) and then click on the union operation to form the corresponding set.

The second subdivision, marked as 2, includes the operands for the SPO operation. In this case, the user must select one set, and then click on one of S, P or O. She may also assign another set to one of the other operands (S, P, O). Clicking on "=" produces the result. Clicking on "clear" resets the operands previously selected.

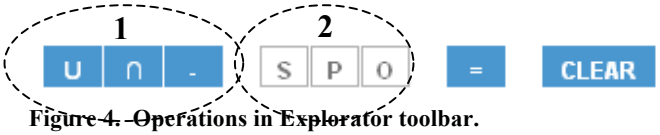
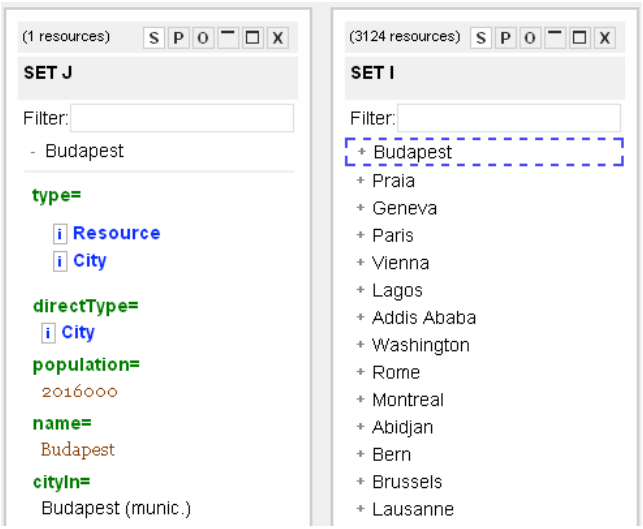


Figure 4. Operations in Explorator toolbar.

The sets are represented as boxes, and stand for both sets of triples or sets of resources. Strictly speaking, all boxes represent sets of triples which can be grouped by subject, property or object. Classes are shown in blue, and RDF properties are shown in green.

Figure 5. Sets of triples represented in Explorator's interface. On the left we have all triples with Budapest as subject. On the right we have some triples grouped by subject.



To select a triple the user simply clicks on the surrounding box, whose border becomes dashed to indicate the selection. If the user double-clicks on a triple, it is interpreted as a request for all triples with the same subject as the subject of the clicked triple.

3.3 Faceted Navigation

In addition to the operations already described, we have also defined a model for specifying tailor made facets. This model can be specified using a custom made vocabulary called FACETO, which we do not elaborate here for reasons of space.

While many tools implement faceted navigation (FacetMap[17], Longwell[18], BrowseRDF[19], Flamenco[20], Exhibit[21], /facet[22][8]), none allow the specification of facets using RDF.

[17] http://www.facetmap.com/

[18] http://simile.mit.edu/wiki/Longwell

[19] http://browserdf.org/

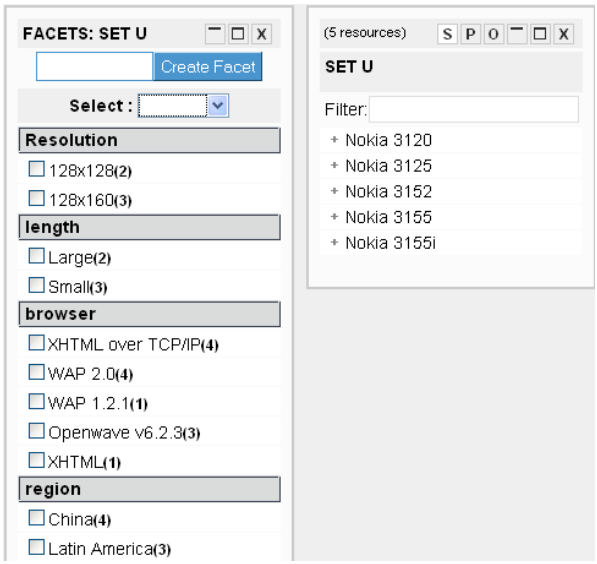


Figure 6: Explorator's faceted interface.

Using FACETO, the designer may.

1. Specify a facet based on a given RDF property;
2. Specify a facet based on computed values. For example, she may define a "dimension" facet based on the combination of values of the "width" and "height" properties.
3. Define synonyms among different resources that represent the same information.
4. Define a facet as an arbitrary enumeration of values, or as a range. For example, "inexpensive" and "expensive".
5. Specify a facet based on a hierarchical relation, such as "located in".

Note also, none of the existing tools can be applied directly to an arbitrary SPARQL Endpoint. Using Explorator, the user can facet any set of triples retrieved during her navigation.

As an added convenience, we have also implemented an algorithm, based on entropy measures, that given a set of triples, determines the set of properties that is most discriminant for that set, and builds a set of facets based on these properties. Again, due to space limitations, we do not detail this algorithm here. This operator can be activated by clicking on the F* button in the interface of any set.

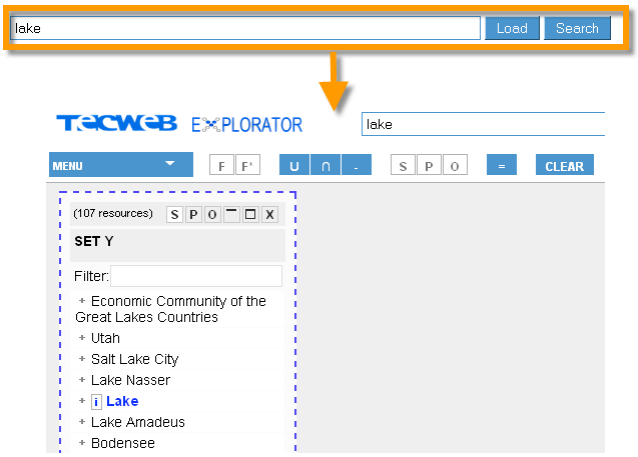
Due SPARQL language limitations (missing of aggregation functions), applying this operation over a SPARQL endpoint may be very time consuming.

3.4 An Example

Let us now illustrate the usage of Explorator. Suppose the user needs to find all the lakes contained exclusively in Russia. There are several possible ways to achieve this task; one possible way would be as follows:

1. Find all the lakes in the database;
2. Find Russia, the country;
3. Find all the lakes in Russia obtaining a set we will call LR;
4. Find the countries that share a boundary with Russia (Russia's neighbors);
5. Find all the lakes in Russia's neighbors, obtaining a set we will call LN; and
6. Build the set of the lakes contained exclusively in Russia by calculating the difference between the previous sets: LR-LN

To find all the lakes in the database, the user first searches for "lake":

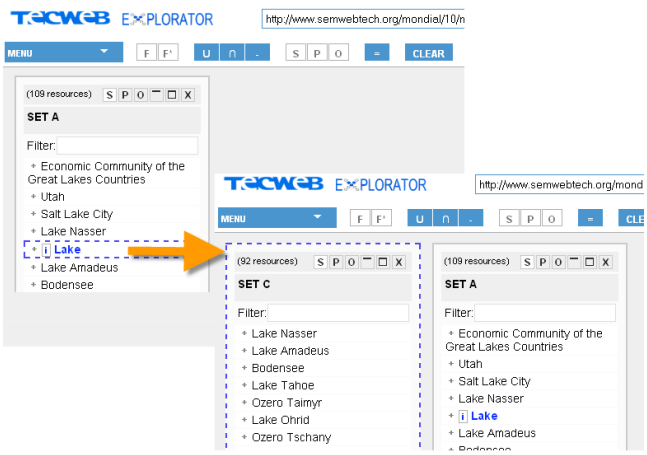


She locates the Lake class (in blue) in the resulting set, and gets the set of instances of the Lake class by clicking on it, to obtain all the lakes in the database:

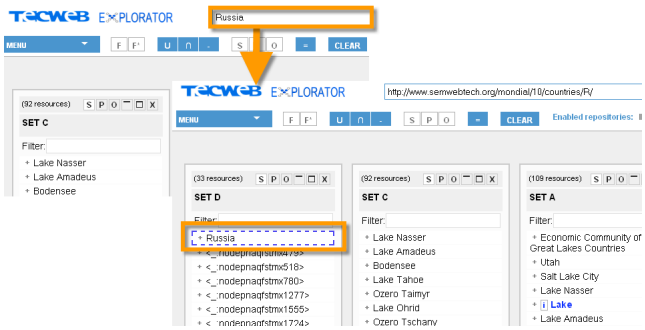
[20] http://flamenco.berkeley.edu/

[21] http://simile.mit.edu/exhibit/

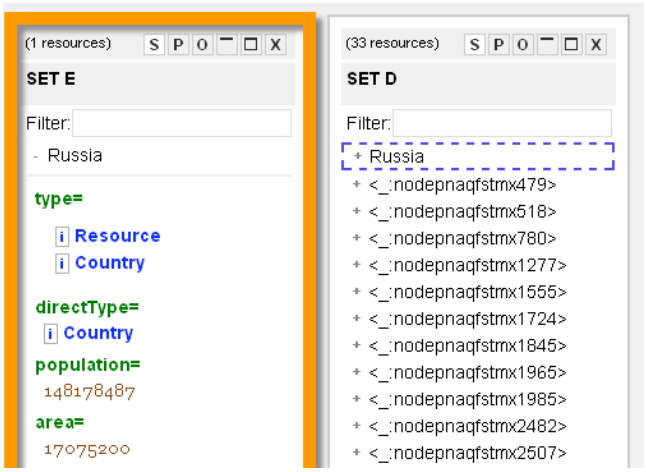
[22] http://slashfacet.semanticweb.org/



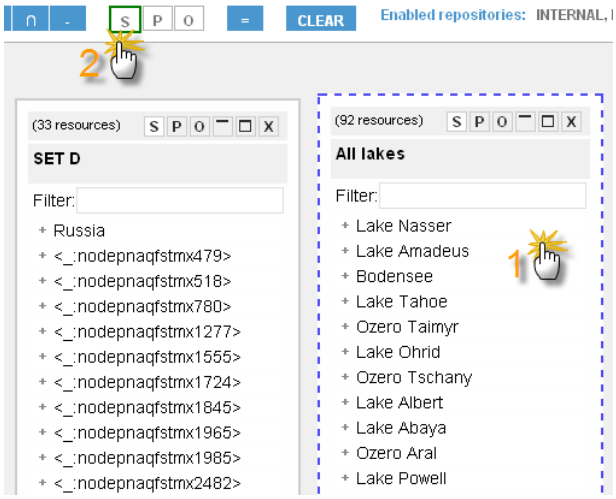
Next, to find Russia, she searches for "Russia" and locates the resource Russia in the resulting set:



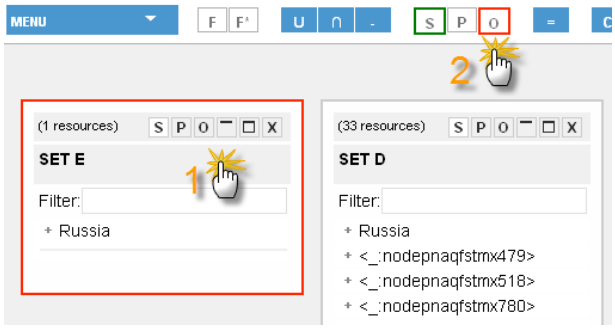
To make sure she has the right resource, she views the resource details:



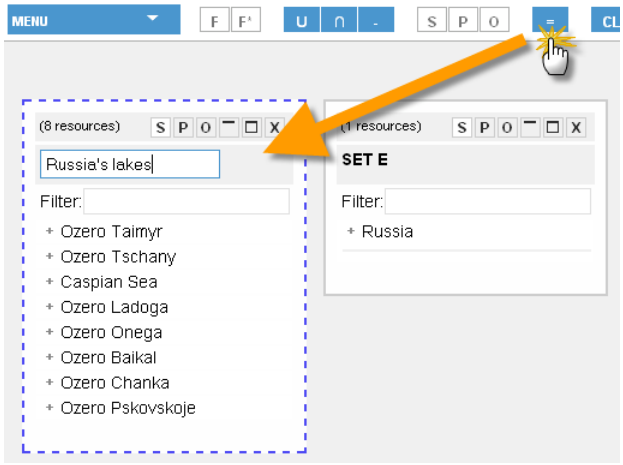
Next, to find all lakes LR in Russia, she selects the set of all lakes and sets it as the subject of her query by clicking on the [S] toolbar button:



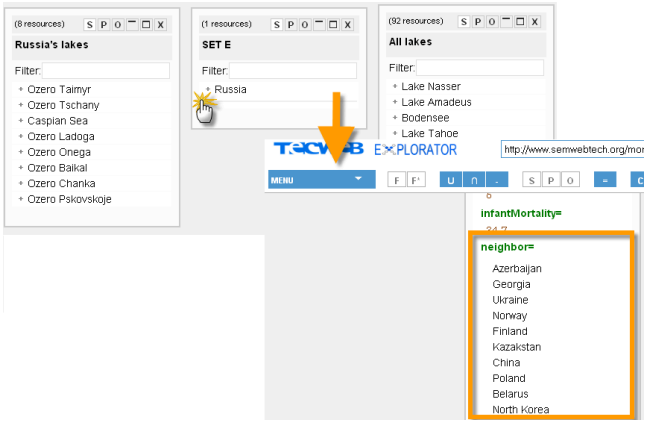
Continuing to build the query, she selects the resource Russia and sets it as the object of her query:



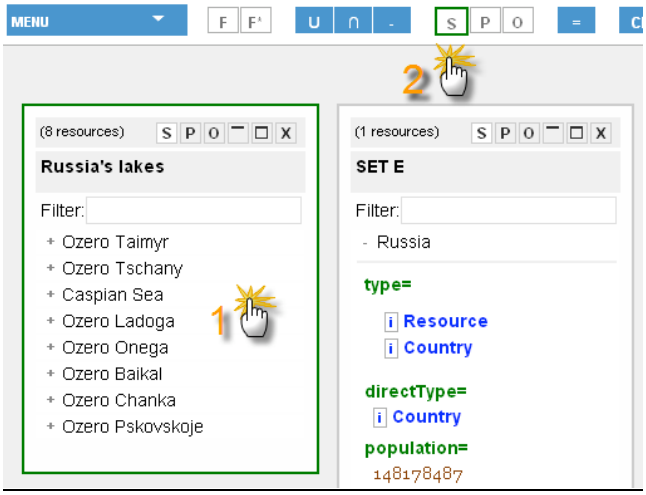
She executes the query to obtain the set of all lakes in Russia:



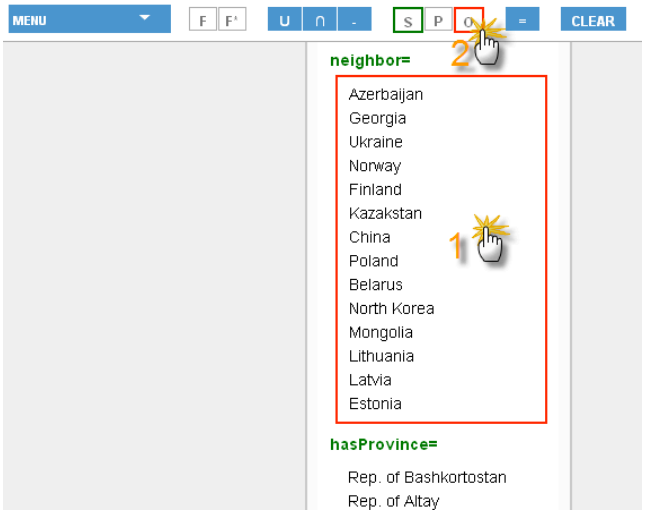
Next, to find the countries that share a boundary with Russia, she views the details of the Russia resource and locates the "neighbor" property for Russia, thereby finding its neighboring countries:



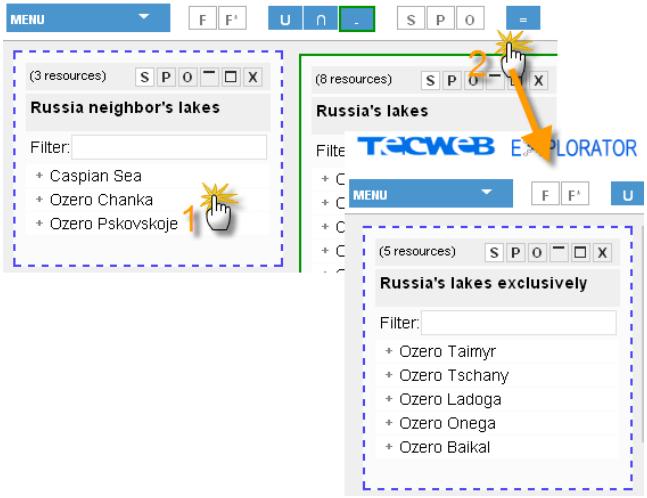
To find all the Russian lakes that are also in Russia's neighbors, she selects the set of Lakes in Russia and sets it as the subject of her next query:



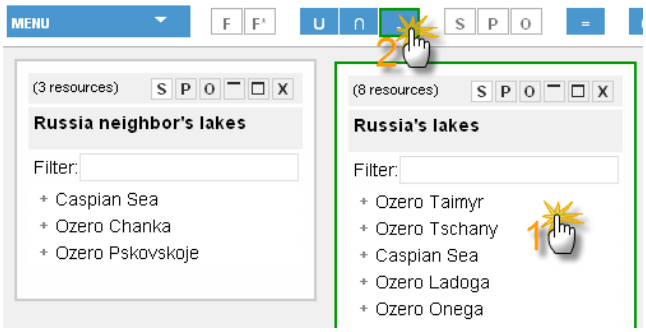
She selects the set of Russia's neighbors and sets it as the object of her query:



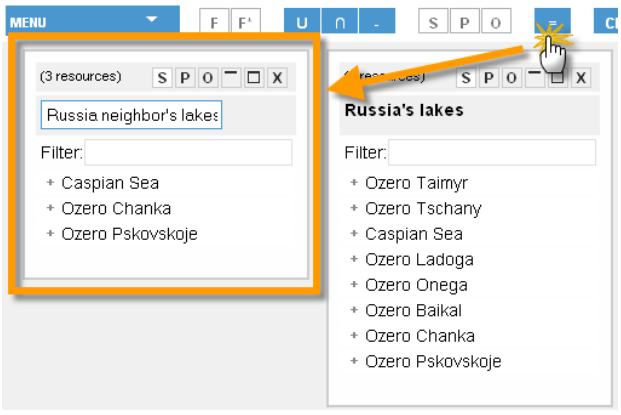
She then executes the query to find all lakes in Russia's neighboring countries:



Finally, to build the set of the lakes contained exclusively in Russia, she needs to calculate the difference between the set of lakes in Russia and the set of lakes in Russia's neighbors. To do this, she selects the first set and the difference operator:



Finally, she selects the second set (containing the lakes in Russia's neighbors) and executes the difference operation by clicking on the equal sign [=] toolbar button, thereby obtaining the desired result:



4. IMPLEMENTATION

In the following we outline our implementation architecture and some notable details. We decided to use a two layer architecture which separates the upper presentation layer from the lower model layer.

4.1 Presentation Layer

For the implementation of the proposed interface we adopted the approach of adding semantic annotations in the HTML code to define interface widgets behavior. To that end, we used the Prototype[23] library, which allows us to easily navigate the DOM tree, select elements by their class attribute values - using CSS - and link operations to interface events such as onclick, onmouseover, onkeyup, etc.. This technique enables us to create very dynamic interfaces for direct manipulation with continuous representation, incremental actions and feedback. Also, all users requests to the server are made using Ajax[24], allowing users to continue to explore data while their request are being processed.

4.2 Model Layer

The model layer can be summed up in the picture below:

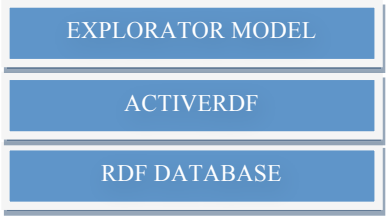


Figure 7. Explorator model architecture

We used the ActiveRDF [13] framework as a layer for translating the Explorator model to the RDF model. Basically, we used the ActiveRDF to generate SPARQL queries from our model. The set operations are performed on Ruby objects because the ActiveRDF and SPARQL do not support those operations natively. The query and cache mechanism of ActiveRDF were modified to better support integration with Explorator's model.

The default dereferencing mechanism implemented is quite simple: it simply retrieves and loads all triples retrieved from the URI into a SESAME repository. No inference or recursive dereferencing heuristic is applied. As a result of this approach, the user can explore the triples retrieved along the direct URI navigation as a SPARQL Endpoint.

5. CONCLUSION

Exploratory search is a data exploration technique that supports complex user's tasks involving lookup as well as learning and investigation. We have shown how this technique can be employed for arbitrary RDF databases. We have developed an information-processing model that supports the tasks in the Semantic Web that not only consist of a searching for a known item, but also consists of acquisition and assimilation of knowledge and concepts in an RDF database. This model has been implemented in a tool called Explorator. We use the direct manipulation metaphor in the construction of the interface, which was very effective in formulation of complex queries over an unknown domain.

Explorator also allows faceted navigation, and we developed an RDF vocabulary for faceted specification and an algorithm for automatic extraction of all facets of a set of triples.

We have conducted a preliminary study [1] that has shown encouraging results. Users with only basic knowledge of RDF were able to elaborate nontrivial queries with Explorator. We realized that Explorator's performance (query execution time) had a negative impact on the user experience, especially when accessing remote endpoints. It may be the case that users explored less because of the time it took to compute the queries. In fact, the time consumption is demanded by the SPARQL datastores, which are still in early stages, especially when compared to relational DBMSs. This issue is of the utmost importance and is being addressed for future versions.

Not surprisingly, the experiments showed us that Explorator is better suited to advanced users who have solid knowledge about RDF. Nevertheless, the experiments were brief, so we cannot yet draw any conclusions about Explorator's learning curve. Preliminary evidence indicates that once the initial difficulty is overcome, users can become quite proficient with the system.

The next step in our study will be to investigate the use of Explorator as an epistemic tool, for users to understand more about the represented data domain, as opposed to performing predefined tasks and answering specific questions. In particular, an open hypothesis is the adequacy of the RDF model to match the user's mental models – some of the collected evidence suggests that it might be too low level, which means suitable abstractions might have to be introduced. Exposing Explorator's operation model to naïve users is still a challenge which is the subject of ongoing research.

Additional larger-scale experiments should be conducted to compare different user interface alternatives and interaction paradigms to better support both novice and expert users in exploring the semantic web. To do so, Explorator can be instrumented to remotely capture the users' actions at the user interface and on the underlying processing model.

As future work, we will extend the model to support the definition of parameterized sets, i.e., sets derived from parameterized operations. Following the QBE paradigm, the user will be able to select any set in the interface, and indicate which should be the parameters. Once this has been done, the user can then plug the output of a box as the input of another box (set), thus establishing a graph of inter-related operations, much like a spreadsheet. Such parameterized sets can be saved to libraries, to be later reused by any user.

Explorator needs some improvements related to the dereferencing heuristics. Also, we are working on some mechanisms to enable exporting RDF, and for enabling alternative views to allow the user to visualize the resources and triples in table, timetables and maps, as well as in customized domain-dependent formats.

In summary, Explorator's contributions are:

- An information exploration model for RDF based on facet and set navigation;
- An exploration environment that allows query formulation by direct manipulation, allowing remote and local SPARQL endpoints exploration;
- Automatic facet generation for given sets of RDF triples;

[23] http://www.prototypejs.org/

[24] http://ajaxpatterns.org/

- A facet specification vocabulary and corresponding implementation within the tool (not shown in this paper).

Explorator is an open source project and can be accessed at http://www.tecweb.inf.puc-rio.br/explorator.

ACKNOWLEDGMENT. Daniel Schwabe was partially supported by a grant from CNPq.

6. REFERENCES

[1] Araújo F. C. S.; Schwabe D.; Barbosa D. J. S. Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. Visual Interfaces to the Social and the Semantic Web. VISSW 2009. Sanibel Island, Florida February 2009 (http://www.smart-ui.org/events/vissw2009/index.html)

[2] Baldonado M. Q. W., Winograd T. SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests. 1996

[3] Berners-Lee T., Chen Y., Chilton L., Connolly D., Dhanaraj R.,Hollenbach J, Lerer A., and Sheets D. Tabulator: Exploring and Analyzing linked data on the Semantic Web. Decentralized Information Group. Computer Science and Artificial, Intelligence Laboratory. Massachusetts Institute of Technology. Cambridge, MA, USA. 2006.

[4] Best Practice Recipes for Publishing RDF Vocabularies. http://www.w3.org/TR/swbp-vocab-pub/

[5] Catarci, T., Costabile, M. F., Levialdi, S., Batini, C., 1997. Visual Query Systems for Databases: A Survey. Journal of Visual Languages and Computing, 8(2), 215-260, 1997.

[6] Dereferencing a URI to RDF. http://esw.w3.org/topic/DereferenceURI

[7] Ding L., Zhou L., Finin T., Joshi A. How the Semantic Web is Being Used: An Analysis of FOAF Documents. Proceedings of the 38th Hawaii International Conference on System Sciences – 2005

[8] Hildebrand M., Ossenbruggen J. v. and Hardman L. /facet: A Browser for Heterogeneous Semantic Web Repositories. The 5th International Semantic Web Conference (ISWC). Athens, GA, USA. 2005

[9] Huynh D. F., Karger D. R., Miller R. C.. Exhibit: lightweight structured data publishing. International World Wide Web Conference. Proceedings of the 16th international conference on World Wide Web (WWW). Banff, Alberta, Canada. 2007

[10] Lima, F.; Schwabe, D.: "Application Modeling for the Semantic Web", Proceedings of LA-Web 2003, Santiago, Chile, Nov. 2003. IEEE Press, pp. 93-102, ISBN (available at http://www.la-web.org).

[11] Marchionini G. Exploratory search: From finding to understanding. Comm. Of the ACM, 49(4), 2006.

[12] OREN, E.; Delbru, R.; Decker, S. Extending faceted navigation for RDF data. 5th International Semantic Web Conference, Athens, GA, USA, LNCS 4273, p. 5-9. 2006

[13] Oren E.,Delbru R., Gerke S., Haller A., Decker S. ActiveRDF: ObjectOriented Semantic Web Programming. Digital Enterprise Research Institute National University of Ireland, Galway Galway, Ireland. 2007

[14] ROSSI, G.; SCHWABE, D.; LYARDET, F.; "Patterns for Designing Navigable Spaces", Proceedings of PLoP98 (Tech Report TR #WUCS-98-25, Washington University, St. Louis, MO, USA), Monticello, Illinois, USA, August 1998.

[15] Russell, A., Smart, P. R., Braines, D. and Shadbolt, N. R. (2008). NITELIGHT: A Graphical Tool for Semantic Query Construction. In: Semantic Web User Interaction Workshop (SWUI 2008), 5th April, Florence, Italy. 2008.

[16] Schwabe, D., Rossi, G.: An object-oriented approach to web-based application design. Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, October, 1998, 207-225.

[17] Shneiderman, Ben, Direct manipulation: a step beyond programming languages. IEEE Computer 16,8 (August 1983), 57-69.

[18] Zloof, M. M., 1977. Query-by-example: a database language. IBM System Journal 16, 324-343, 1977.