

# Support for Reusable Explorations of Linked Data in the Semantic Web

Marcelo Cohen and Daniel Schwabe

Pontifical Catholic University of Rio de Janeiro

R. M. S. Vicente 225

Gávea, Rio de Janeiro, RJ, Brazil

mcohen21@gmail.com, dschwabe@inf.puc-rio.br

**Abstract.** The Linked Data cloud growth is changing current Web application development. One of the first steps is to determine whether there is information already available that can be immediately reused. We provide an environment which allows non-technically savvy users, but who understand the problem domain, to accomplish these tasks. They employ a combination of search, query and faceted navigation in a direct manipulation, query-by-example style interface. In this process, users can reuse solutions previously found by other users, which may accomplish sub-tasks of the problem at hand. It is also possible to create an end-user friendly interface to allow them to access the information. Once a solution has been found, it can be generalized, and optionally made available for reuse by other users.

**Keywords:** RDF, exploratory search, exploration, ontology, semantic web, reuse, interface, set-based navigation.

## 1 Introduction

The availability of Linked Open Data in the WWW has increased tremendously<sup>1</sup>. Currently, when building a new application, it is becoming increasingly common to first explore available data that can be leveraged to enhance and complete one's own data to provide the desired functionality. The BBC Music website<sup>2</sup> is one visible example of this approach, combining MusicBrainz and DBPedia with their own data.

Even though it is engineered to be processed by programs, it is still common that human beings need to explore these datasets, especially when they are previously unknown. In such cases, experts typically explore the repository to make sense out of the available data, to eventually be able to formulate queries that will support their tasks. Existing interfaces range from basic RDF browsers such as Tabulator<sup>3</sup>, Zitgist

---

<sup>1</sup> <http://linkeddata.org>

<sup>2</sup> <http://www.bbc.co.uk/music>

<sup>3</sup> <http://www.tabulator.org/>

data viewer<sup>4</sup>, Marbles<sup>5</sup>, ObjectViewer<sup>6</sup> and Openlink RDF Browser<sup>7</sup>, to query generators such as NITELIGHT [9] and iSPARQL<sup>8</sup>, to faceted browsers [8][3] and set-based interfaces [4].

In previous work [2], we presented Explorator, a model for representing information processing by users in exploratory tasks, and its associated tool, which provides a browser interface supporting this model. Explorator is based on the metaphor of direct manipulation of information in the interface, with immediate feedback of user actions.

Our experience with Explorator [1] has shown that to be effectively used, it is necessary for users to understand the RDF model. Even for these users, once a solution was found, it was not possible to generalize it, and to save it for reuse later. These two mechanisms are essential to enable a community of users around datasets of interest, so that more experienced users can find and share solutions with less experienced ones. Furthermore, it is desirable to provide an end-user facing interface that hides the underlying data and operations, and has the look-and-feel of a traditional web application.

In this paper we present RExplorator<sup>9</sup>, a significant extension of Explorator that allows

1. Parameterized interlinked operations, forming a graph of operations;
2. Saving these graphs for reuse;
3. The user to define new operators;
4. The user to define end-user friendly interfaces.

In the remainder of this paper, section 2 provides a running example, section 3 describes RExplorator, section 4 discusses evaluation, and section 5 draws some conclusions.

## 2 Summary of Explorator and a Running Example

### 2.1 Summary of Explorator

Explorator is an environment that allows users to explore a set of RDF repositories by direct manipulation of its contents, following a set-based metaphor. The user starts by either executing a full-text search, or by executing pre-defined queries (e.g., “All RDF Classes” or “All RDF Properties”). It is also possible to simply take a URI and de-reference it. In all cases, the results are always sets of triples.

The user explores the repositories by executing operations that take as operands sets of resources, and return new sets. The usual set operations, union, intersection and difference are available. In addition, there is the SPO operator, which corresponds to a match operation over  $\langle s, p, o \rangle$  triple patterns (e.g.,  $\langle s, *, * \rangle$ ,  $\langle s, p, * \rangle$ , for given  $s$  and  $p$  values, which are URIs). This match is executed against all enabled RDF triple repositories. Thus,  $\langle s, *, * \rangle$  corresponds to the SPARQL query

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

<sup>4</sup> <http://dataviewer.zitgist.com/>

<sup>5</sup> <http://beckr.org/marbles>

<sup>6</sup> <http://objectviewer.semwebcentral.org/>

<sup>7</sup> <http://demo.openlinksw.com/rdfbrowser/index.html>

<sup>8</sup> iSparql can be accessed at <http://demo.openlinksw.com/isparql/>

<sup>9</sup> Available at <http://www.tecweb.inf.puc-rio.br/rexplorator>

In reality, the SPO operator has been defined to operate on sets of resources instead of individual ones, by taking the union of the triples resulting from individual match operations as described above.

Since each new operation takes its parameter from existing sets, the end result is a graph of inter-related operations, where the inputs of one are outputs of others. This is analogous to an Excel spreadsheet, where each cell has formulas that reference the value of other cells, forming a graph of interdependent formulas.

## 2.2 A Running Example

Consider the simple task of finding all publications of a given author. to be carried out over the “Dogfood” data server<sup>10</sup>, containing collected publication information for several conferences related to the Semantic Web. We assume the user has no prior knowledge about the contents of this repository. The user has to

1. Find a class that represents persons
2. Find the desired person, “a”.
3. Find a property “p” that relates a person to publications,
4. Find all triples of the form <a p ?pub> and collect all objects from these triples.

The screenshot shows the RExplorer interface with three main panels. At the top, there is a 'MENU' dropdown, a search bar with 'F P', a navigation bar with 'S P O', and a 'CLEAR' button. The 'Enabled repositories:' field shows 'IN'.

The first panel, 'Selected Person Details', shows the details for 'Steffen Staab'. It lists properties such as 'type' (Resource, Person), 'seeAlso' (Steffen\_Staab), 'label' (Steffen Staab), 'based\_near' (Germany), 'homepage' (isweb.uni-koblenz.de), 'name' (Steffen Staab), 'affiliation' (University of Koblenz-Landau), and 'family\_name' (Staab).

The second panel, 'Publication by Person', shows a list of publications made by 'Steffen Staab'. The titles include 'SAIQL Query Engine - Querying OWL Theories for Ontology Extraction', 'On How to Perform a Gold Standard Based Evaluation of Ontology Learning', 'COMM: Designing a Well-Founded Multimedia Ontology for the Web', 'Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web', 'Querying for Meta Knowledge', and 'Logging in Distributed Workflows'.

The third panel, 'All Persons', shows a list of all persons in the repository. The names include Alexander Kubias, Simon Schenk, Steffen Staab (highlighted with a dashed blue border), Tim Furche, Alina Hang, Benedikt Linse, Francois Bry, Hugh Glaser, Ian Millard, Benedicto Rodriguez, Afraz Jaffri, Eetu Mäkelä, Jouni Tuominen, Kim Viljanen, Osmo Suominen, Eero Hyvönen, Antti Vehviläinen, Olli Alm, Tomi Kauppinen, Christine Deichstetter, Andrea Bonomi, and Glauro Mantovani.

**Fig. 1.** All Persons, Details of a selected Person, and Publications of selected Person, in RExplorer.

<sup>10</sup> <http://data.semanticweb.org>

In Explorer, this is achieved by first clicking on “Menu”-> All RDF Classes”, noticing class Person, mousing over it to click on “All Instances”, which reveals a set of all Persons. Double-clicking on a Person (e.g. “Steffen Staab”), a new box appears with all details for this resource (i.e., all triples with this resource as subject). Looking at the details, one notices the property “made”, which relates Person to Publications.

To get all publications by a Person, one may click on the “Selected Person Details” box, then click on the “S” operand position at the top; click on the “made” box and click on the “P” operand position at the top, and finally click on the “=” (“compute”) operator at the top. Figure 1 shows the results after these steps.

### 3 REplorer

REplorer extends Explorer by

1. Allowing operations to be parameterized;
2. Allowing the results of a query to be fed as input of another query, thus forming graphs of interconnected operations;
3. Allowing to keep such graphs as separate workbenches, while enabling interconnection of graphs across workbenches;
4. Allowing the designer to import previously defined query graphs into the current workbench;
5. Allowing the designer to define additional operators beyond the builtin set and query operations provided;
6. Allowing the designer to define interfaces oriented towards end users, hiding details and customizing the look-and-feel.

REplorer’s metamodel is shown in Figure 2, which supports the implementation of these features. Some of its aspects will be elaborated as we explain these added functionalities in the coming sub-sections.

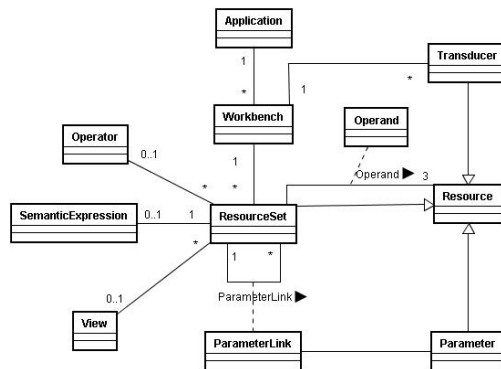



Fig. 2. REplorer’s meta-model

### 3.1 Parameterized Queries

The original Explorator metaphor lets users compose operations incrementally, seeing the results at each composition step. Each new query takes its operands from existing query results. In the end, one may regard this set of inter-related operations as a graph, similar to an Excel spreadsheet. However, the operations are all grounded, which would be akin to not having any variables in the formulas of the analogous spreadsheet. Thus, the first generalization made was to allow operations to have their operands parameterized, and to propagate values through the graph of operations when the value of the parameter is changed. This is equivalent to introducing variables in the expression that denotes the operation.

Consider step 4 in the example. In Explorator, this is achieved by selecting an instance of Person (e.g., “Steffen Staab in box “All Persons”) in Figure 1, setting it as the subject parameter, selecting the relation “make” as the property parameter, and clicking on the “=” operator to find all triples of the form  $\langle\langle\text{url for Steffen Staab}\rangle\text{made ?o}\rangle$ . Clicking on the  icon in each box, as shown in Figure 3 reveals the actual operations and their dependencies.

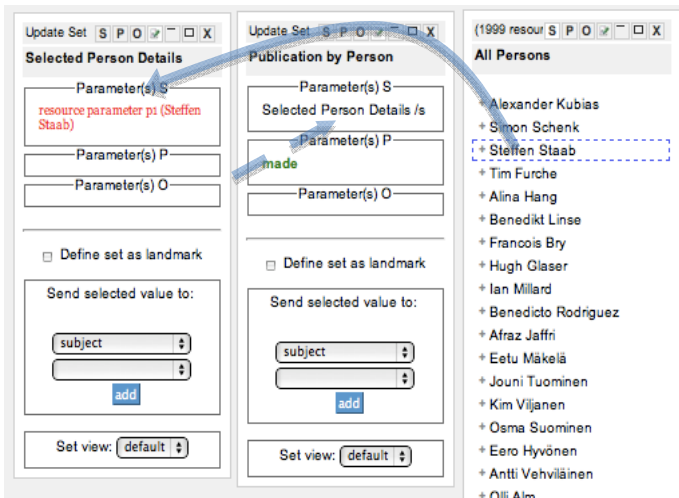


Fig. 3. Query structure and dependencies

The first box, **Selected Person Details**, represents the query that finds out all triples with a given Person as subject. Notice that the first position, “S”, has been parameterized, and the current parameter value is (the URI for) **Steffen Staab**. If we drag any person from the rightmost box (**All Persons**) onto the “S” position in the **Selected Person Details** box, the value is replaced and the query re-evaluated.

The **Publication by Person** query (middle box) is defined as taking its “subject” parameter from the “subject” position of the **Selected Person Details** query. Therefore, if a new value is plugged into the “S” position in the **Selected Person Details** query, it is automatically propagated to this query, triggering its reevaluation.

### 3.2 Workspace Organization

RExplorator organizes the workspace into workbenches. The idea is that each workbench represents a task, or a use case in traditional Software Engineering methods. A user may save workbenches for later reuse, and share it with other users as well.

In RExplorator a workspace contains several workbenches, similar to the way an Excel a workspace contains several worksheets, where there may be cross-references between operations within separate workbenches. For example, workbench **Co Workers by Person** contains the **Co Workers** query, which can be interconnected to the “Publication by Person” query in the similarly named workbench.

### 3.3 End-user Interfaces

The development interface of RExplorator is best suited to allow users to explore RDF repositories, and requires understanding the RDF model. For non-technical end users, RExplorator allows expert users to provide end-user friendly interfaces – called the Application Interface - to solutions found while exploring datasets. For reasons of space, we refer the reader to <http://www.tecweb.inf.puc-rio.br/rexplorator> to visualize the Application interface.

Views make full use of CSS, which is also defined in a separate view that can be customized to change the look-and-feel of the generated interface.

### 3.4 User-Defined Operators

The original Explorator tool provides built-in set operators to manipulate the resource (triple) sets, besides the SPO query operator. Besides this, RExplorator provides a mechanism for the designer to define new operators.

Since operators work on sets of triples, a natural kind of function is the “list”, “iterator” or “map” function commonly found in functional languages such as Lisp, Python, and Ruby, among others. In RExplorator, operators take two sets of triples as input and produce a set of triples as output.

As an example, one may want to filter a result set that contains datatype properties (e.g., `rdf:label`) according to a string value passed as a parameter. The Ruby code snippet below shows the definition of an operator that takes a resource set and a string as input parameters, and selects those triples whose object position matches the string.

```
param_a.select { |triple| triple[2].to_s.strip.downcase ==
param_b[0].to_s.strip.downcase }
```

## 4 Evaluation

We conducted a small qualitative study to have a preliminary evaluation of RExplorator. We asked 5 persons with basic RDF knowledge to build simple applications using a repository describing cellular phone models. The tasks consisted of

1. Exhibiting all available models
2. Showing models that support MP3
3. Showing models grouped by supported band

First they were shown a short video with RExplorator's basic functionalities. Then they were allowed to experiment with RExplorator for a short time and have basic questions about its functioning answered, after which they were given one hour to accomplish the tasks.

Of the five people, three were able to successfully accomplish the tasks in less the allotted time; one completed the tasks but with a slightly incorrect solution; and one could not accomplish the task.

We consider these results to be positive, showing that the tool can be effective. The test subjects were given minimal instructions, and yet most were able to accomplish the tasks. It is clear that this interface is not for beginners, but once the developer has become familiar with it, it is quite effective.

Nevertheless, the experiments indicate that the authoring interface should be improved, for example using graphics to better represent the dependencies between sets.

## 5 Conclusions

The environment that has the closest functionality to RExplorator is DERI Pipes [3], which allows the definition of mash-ups by creating networks of interconnected operators, with strings, XML or RDF data flowing through them. The desired result is obtained by the composition of the operators.

By analogy, RExplorator can be seen as a network of interconnected operators, which can be queries, set operations or customized functions. The data that flows in this network are sets of triples. Thus, the major difference is that it is oriented towards mash-up development, and as such its operators work at a lower abstraction level. In addition, DERI Pipes does not provide an interface layer, and is not meant to be used together with an exploration environment.

One of the major focuses for future work is providing a graphical authoring interface that makes it easier to visually identify the inter-dependence of the various operations. We are also investigating the reuse of solutions within communities that share solutions over a specific set of repositories.

**Acknowledgment.** Daniel Schwabe was partially supported by a grant from CNPq.

## References

1. Araújo, F.C.S., Schwabe, D., Barbosa, D.J.S.: Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. In: 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. Araújo F. C. S., Schwabe D.: Explorator A tool for exploring RDF data through direct manipulation. In: Proceedings of the Linked Data on the Web Workshop (LDOW 2009), Madrid, Spain, April 20. CEUR Workshop Proceedings, pp. 1613–1673 (2009), [http://CEUR-WS.org/Vol-538/ldow2009\\_paper2.pdf](http://CEUR-WS.org/Vol-538/ldow2009_paper2.pdf) ISSN 1613-0073
3. Hildebrand, M., Ossenbruggen, J.v., Hardman, L.: /facet: A Browser for Heterogeneous Semantic Web Repositories. In: The 5th International Semantic Web Conference (ISWC), Athens, GA, USA (2005)

4. Huynh, D., Karger, D.: Parallax and companion: Set- based browsing for the data web, <http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf>
5. Le Phuoc, D., Polleres, A., Morbidoni, C., Manfred Hauswirth, M., Tummarello, G.: Rapid semantic web mashup development through semantic web pipes. In: Proceedings of the 18th World Wide Web Conference (WWW 2009), Madrid, Spain (April 2009)
6. Luna, A.M., Schwabe, D.: Ontology Driven Dynamic Web Interface Generation. In: Proceedings of the 8th International Workshop on Web Oriented Technologies (IWWOST 2009), San Sebastian, Spain. CEUR, vol. 493, pp. 16–27 (2009), [http://ceur-  
ws.org/Vol-493/iwwost2009-luna.pdf](http://ceur-ws.org/Vol-493/iwwost2009-luna.pdf) ISSN 1613-0073
7. Moura, S.S., Schwabe, D.: Interface Development for Hypermedia Applications in the Semantic Web. In: Proc. of LA Web 2004, pp. 106–113. IEEE CS Pres, Ribeirão Preto (2004) ISBN 0-7695-2237-8
8. Oren, E., Delbru, R., Decker, S.: Extending Faceted Navigation for RDF Data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006)
9. Russell, A., Smart, P.R., Braines, D., Shadbolt, N.R.: NITELIGHT: A Graphical Tool for Semantic Query Construction. In: Semantic Web User Interaction Workshop (SWUI 2008), Florence, Italy (April 5, 2008)