# Data Acquisition by Traversing Class–Class Relationships over the Linked Open Data

Atsuko Yamaguchi[1], Kouji Kozaki[2], Kai Lenz[3], Yasunori Yamamoto[1], Hiroshi Masuya[4,3], and Norio Kobayashi[3,4,5]

[1] Database Center for Life Science (DBCLS),
Research Organization of Information and Systems,
178-4-4 Wakashiba, Kashiwa, Chiba, 277-0871 Japan
{atsuko,yy}@dbcls.rois.ac.jp
[2] The Institute of Scientific and Industrial Research (ISIR), Osaka University,
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
kozaki@ei.sanken.osaka-u.ac.jp
[3] Advanced Center for Computing and Communication (ACCC), RIKEN,
2-1 Hirosawa, Wako, Saitama, 351-0198 Japan
{kai.lenz, norio.kobayashi}@riken.jp
[4] RIKEN BioResource Center (BRC),
3-1-1, Koyadai,Tsukuba, Ibaraki, 305-0074 Japan
hmasuya@brc.riken.jp
[5] RIKEN CLST-JEOL Collaboration Center,
6-7-3 Minatojima-minamimachi, Chuo-ku, Kobe 650-0047, Japan

**Abstract.** Linked Open Data (LOD) is a powerful mechanism for linking different datasets published on the Web, which is expected to create new value of data through mash-up over such various datasets. One of the important needs to extract data from LOD is to find a path of resources connecting given two classes, each of which has an end resource of the path. Based on the concept, we have been developing data acquisition system named SPARQL Builder assisting users in semantic queries for LOD. Through the development, we introduced the two technologies for the approach: a labeled multigraph named class graph to compute class-class relationships and an RDF specification named SPARQL Builder Metadata to obtain and store required metadata for construction of a class graph.

**Keywords:** linked data, class–class relationships, data integration

## 1 Introduction

In order to efficiently use databases published as Linked Open Data (LOD), the users need to be allowed to obtain data in the flexible way according to their interests. An important case is to find paths of links between instances (resources) whose types are given two classes for integrative data analysis with semantics. These paths can be obtained by retrieving chains of properties (links)

which connect instances of classes. In other words, these paths can be obtained by traversing paths of class–class relationships over the LOD.

Therefore, based on class–class relationships, we have been developping a system named SPARQL Builder to obtain data from LOD flexibly, by assisting users in writing SPARQL queries to the SPARQL endpoints. To realize our approach, we should develop the following two techniques: 1) a method to collect profiles related to class–class relations through SPARQL endpoints of RDF datasets: This is implemented as SPARQL Builder Metadata (SBM), which describes comprehensive metadata including not only class definitions but also statistics such as the number of instances while it is not supported existing metadata. 2) a method to obtain chains of properties and classes by computing paths on labeled multigraph named class graph: This enables an efficient method to compute path and a measure to remove paths of classes with no instance path are proposed.

Related application includes Visor[1], which enables users to browse RDF datasets in the light of class–class relationships. However, Visor doesn't provide a method to find an end-to-end path through multiple resources. Although another related work is RelFinder [2] which computes paths between resources in LOD, it is not based on class–class relationships but on instance–instance relationships.

## 2 SPARQL Builder

We have been developing a practical LOD search tool named SPARQL Builder for the life-science data analysis (`http://www.sparqlbuilder.org/`). This tool provides an interactive GUI that allows users who are not familiar with SPARQL language to generate SPARQL queries without knowledge of SPARQL and RDF data schema [3]. Overview of system architecture is shown in Fig 1. SPARQL Builder manages SBM generated by accessing SPARQL endpoints in advance (1). When a user access to the SPARQL Builder system via a web browser as a GUI, SPARQL Builder obtains a list of classes by analysing SBM (2) and displays the list on the user's web browser (3). Then, when the user selects "input" and "output" classes, SPARQL Builder constructs class paths by traversing the class graph constructed using information described in SBM (4) and draw them on the web browser. Using this GUI, users can explore datasets as their interest by specifying classes. If a user interested in the interrelationships between molecular pathways and proteins, he should do at first is to select protein as input class and pathway as output class. Then, SPARQL Builder shows all possible paths in-

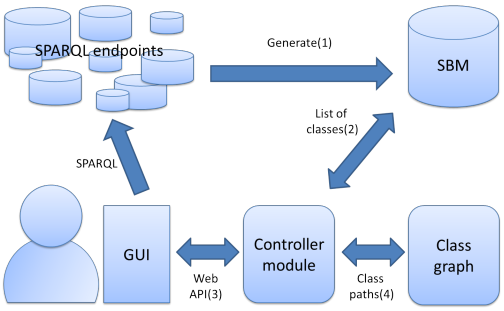


**Fig. 1.** Overview of the SPARQL Builder system.

volves pathways in which proteins that catalyses chemical reactions constitutes. These paths has sequentially connected two relationships as the form of "Protein -(left/right)- BiochemicalReaction -(pathwayComponnt)- Pathway". When he select one of the class paths, SPARQL Builder create a SPARQL query which can use to retrieve data his interest. SPARQL Builder is used for support service to generate SPARQL queries for 38 SPARQL endpoints as of July 2016.

## 3 SPARQL Builder Metadata

SPARQL Builder Metadata (SBM), is a summary of RDF datasets provided via a SPARQL endpoint. SBM is defined as an extension of VoID (`https://www.w3.org/TR/void/`) and SPARQL 1.1 service description (`https://www.w3.org/TR/sparql11-service-description/`) with our original vocabulary whose name space is `sbm:`. SBM contains statistic summary data called "graph summary" for default graph and each named graph provided by the SPARQL endpoint. Graph summary is an extension of VoID vocabulary related to `void:Dataset` class with detailed statistical parameters as follows: *A property partition* is a subset of RDF dataset associated with a property. In addition to original VoID properties, three properties `sbm:subjectClasses`, `sbm:objectClasses`, and `sbm:objectDatatypes` to describe numbers of classes and datatypes are used. *A class relation* is a distinct pair of subject class and object class/datatype, where subject class and object class/datatype are the class of subject instance and class/datatype of object instance/literal in all triples associated with the concerned property partition. `sbm:classRelation` property is introduced to describe for each class relation having properties `sbm:subjectClass`, `sbm:objectClass`, and `sbm:objectDatatype` as our original extension and properties VoID vocabulary.

## 4 Class Graph

To compute paths between two classes efficiently, we employed a specialized graph whose nodes and edges correspond to classes and the class–class relations with predicates, respectively. We call the graph *class graph*. A class graph can be constructed from SBM efficiently because SBM includes a list of all the classes and a list of all the class–class relationships. Given a class graph, an undirected path on the graph is called as a *class path*. Note that a class path is not always simple path because the same classes may appear twice or more in the path with different properties. Class paths between two classes can be found in practically short time using algorithm written in [4] although a class graph is a labeled multi-edge graph and a class path is not simple.

Too many paths to select by a user may be found for relatively large datasets [4]. For example, for Reactome of EBI RDF Platform [5] as of December 2014, the average number of paths between classes with maximum length four was 609. In addition, we found that some class paths have no sequence of instances obtained by traversing triples along the class paths by our preliminary

investigation. We call such a class path *an empty path*. Because a user can not obtain any data using an empty path, it is important to present a method to remove such class paths as many as possible. To do so, we employed a measure to remove empty paths by using statistic values describing in SBM. The probability that a path is not an empty path is estimated by using SBM and used to remove empty paths and used to extract non-empty paths from all the paths. For example, for Reactome, although the average nonempty path ratios for all the class paths with maximum length four is 0.393, the average ratio of non-empty paths for the highest 10 class paths of the probability is 0.893.

## 5 Conclusion

In this study, we discussed novel LOD data exploring methodology and its application SPARQL Builder which enables practical LOD data searching in a SPARQL endpoint. Although the system originally was designed for biological databases, the technologies used in the system including SBM and class graphs are applicable to another domain. Therefore, our future work includes expanding our application into multiple domains and evaluate the generalities of our approach. In addition, we will consider to expand class paths into more general types of subgraphs on class graph, to support more styles of SPARQL queries.

**Acknowledgments** This work was supported by JSPS KAKENHI Grant Number 25280081, 24120002 and the National Bioscience Database Center (NBDC) of the Japan Science and Technology Agency (JST).

## References

1. Popov, IO., Schraefel, M., Hall, W., Shadbolt, N.: Connecting the dots: a multi-pivot approach to data exploration. In The Semantic WebISWC 2011, 553-568 (2011)
2. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: Revealing Relationships in RDF Knowledge Bases. 4th International Conference on Semantic and Digital Media Technologies, SAMT 2009, LNCS 5887, 182–187 (2009)
3. Yamaguchi A., Kozaki K., Lenz K., Wu H, Kobayashi N.: An Intelligent SPARQL Query Builder for Exploration of Various Life-science Databases, CEUR Workshop Proceedings 1279, The 3rd International Workshop on Intelligent Exploration of Semantic Data (IESD 2014), Riva del Garda, Italy.
4. Yamaguchi, A., Kozaki, K., Lenz, K., Wu, H., Yamamoto, Y., Kobayashi, N.: Efficiently finding paths between classes to build a SPARQL query for life-science databases. 5th Joint International Conference (JIST 2015), LNCS 9544, 321–330 (2015)
5. Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., Gaulton A., Gehant, S., Laibe, C., Redaschi, N., Wimalaratne, S. M., Martin, M., Le Novére, N., Parkinson, H., Birney, E., Jenkinson, A. M.: The EBI RDF platform: linked open data for the life sciences. Bioinformatics 30(9), 1338–1339 (2014)