

# SPARQL-Proxy: a generic proxy server for SPARQL endpoints

Shuichi Kawashima<sup>1</sup> and Toshiaki Katayama<sup>1</sup>

<sup>1</sup> Database Center for Life Science, Wakashiba 178-4-4, Kashiwa-shi, Chiba 277-0871, Japan  
kws@dbcls.rois.ac.jp, ktym@dbcls.jp

**Abstract.** SPARQL-Proxy is a portable Web application that works as a proxy server of a SPARQL endpoint. It provides several functions such as job scheduling for SPARQL queries, validating the safety of query statements, and caching of SPARQL search results to improve response time performance.

**Keywords:** SPARQL endpoint, proxy server, RDF.

## 1 Introduction

Providing a SPARQL endpoint is one of the most effective ways that users can easily utilize RDF data. Various SPARQL endpoints are available for major RDF datasets in life sciences such as UniProt and the EBI RDF platform. We are also providing SPARQL endpoints for our RDF data services including TogoGenome [1] and the NBDC RDF portal [2]. When providing a SPARQL endpoint, it is demanded to properly control the submitted queries so that the RDF data management system will not be down due to heavy queries. The function of filtering unsafe queries is also needed. In order to easily make use of such functionalities for any SPARQL endpoint running on the various environments and variety of RDF stores, we have developed a portable web application named SPARQL-proxy.

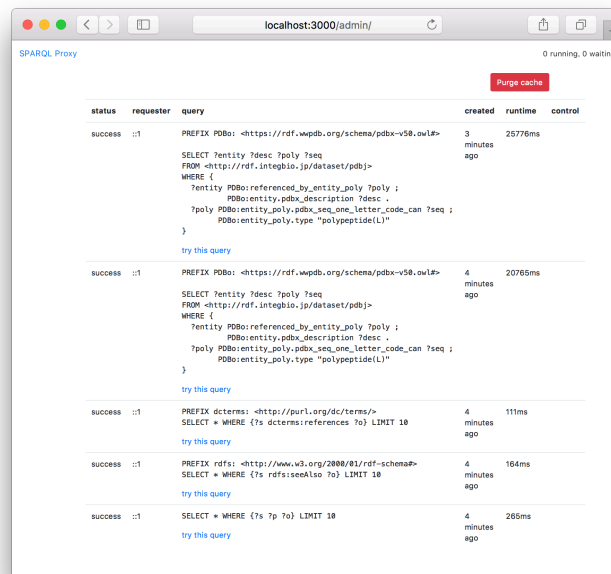
## 2 Methods and Results

SPARQL-proxy is implemented in Node.js. To start it, the user just executes the following command from the directory where it built.

```
$ PORT=3000 SPARQL_BACKEND=<url> npm start
```

It works as a proxy server for the SPARQL endpoint of which the URL is specified via the SPARQL\_BACKEND environment variable. The provider of the SPARQL endpoint can expose the proxy URL instead of the original endpoint URL. In the above case, port 3000 is assigned but it can be 80 or the provider can configure an HTTP reverse proxy to point that port. All other options such as a cache system of choice can also be set via the environment variables. SPARQL-proxy provides two web interfaces: one is the dashboard for administrators to monitor the execution of

jobs (Fig. 1) and the other is the query submission form for the debugging use. Administrators can see the execution logs, cancel running/queued jobs and remove cached results. Submitted queries are validated if it includes unsafe instructions such as a SPARQL Update query prior to passing them to the backend triplestore. The job timeout and the number of concurrent requests can also be specified.



The screenshot shows the SPARQL Proxy dashboard at localhost:3000/admin/. It features a 'Purge cache' button and a table with columns: status, requester, query, created, runtime, and control. The table contains five rows of query execution logs, each with a 'try this query' link below it.

status	requester	query	created	runtime	control
success	:::1	PREFIX PDBo: <https://rdf.wwpdb.org/schema/pdbx-v58.owl#> SELECT ?entity ?desc ?poly ?seq FROM <http://rdf.integbio.jp/dataset/pdbj> WHERE { ?entity PDBo:referenced_by_entity_poly ?poly ; PDBo:entity_pdbx_seq_one_letter_code_can ?seq ; ?poly PDBo:entity_poly_pdbx_seq_one_letter_code_can ?seq ; PDBo:entity_poly_type "polypeptide(L)" }	3 minutes ago	25776ms	
success	:::1	PREFIX PDBo: <https://rdf.wwpdb.org/schema/pdbx-v58.owl#> SELECT ?entity ?desc ?poly ?seq FROM <http://rdf.integbio.jp/dataset/pdbj> WHERE { ?entity PDBo:referenced_by_entity_poly ?poly ; PDBo:entity_pdbx_seq_one_letter_code_can ?seq ; ?poly PDBo:entity_poly_pdbx_seq_one_letter_code_can ?seq ; PDBo:entity_poly_type "polypeptide(L)" }	4 minutes ago	20765ms	
success	:::1	PREFIX dcterms: <http://purl.org/dc/terms/> SELECT * WHERE {?s dcterms:references ?o} LIMIT 10	4 minutes ago	111ms	
success	:::1	PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT * WHERE {?s rdfs:seeAlso ?o} LIMIT 10	4 minutes ago	164ms	
success	:::1	SELECT * WHERE {?s ?p ?o} LIMIT 10	4 minutes ago	285ms	

**Fig. 1.** The dashboard interface of the SPARQL-proxy.

In order to improve the response time of the requested query, SPARQL-proxy provides a function that caches each SPARQL result and returns a cached result when the same query is submitted. The provider of the service can select the caching mechanism from a local file, memory, Redis, and Memcached. To reduce the size of the cache, cached results can be compressed by using snappy.js which is a JavaScript implementation of Google's Snappy compression library. SPARQL-proxy is freely available and the source code is provided in the GitHub repository [3].

## References

1. <http://togogenome.org/>
2. <https://integbio.jp/rdf/>
3. <https://github.com/dbcls/sparql-proxy>