# SPANG: A SPARQL Client with Templates, One-Liners, and Application Programming Interfaces

Hirokazu Chiba

Database Center for Life Science, Chiba 277-0871, Japan
chiba@dbcls.rois.ac.jp

**Abstract.** SPARQL is a key component of the Semantic Web, and its reusability of is crucial for maximizing productivity on the Semantic Web. While an increasing number of datasets have been published in the Resource Description Framework (RDF) and the SPARQL standard allows for an interoperable framework for querying the RDF datasets, the reuse of SPARQL queries remains limited. Herein, we present SPANG, a framework used to facilitate the reuse of SPARQL queries by using a template engine, one-liner functionalities, and application programming interfaces. We demonstrate its use for making SPARQL queries more reusable; assign a set of metadata for annotating a SPARQL query, which enables the identification of the SPARQL query and provides useful information for the query; parameterize SPARQL queries, which makes each query reusable under different settings; assign a URI for each query; and construct a server for providing the indices of SPARQL queries. SPANG makes SPARQL queries more reusable in the local system or across the Web and facilitate the construction of an eco-system wherein SPARQL queries are shared to maximize the productivity of Semantic Web developers and users. SPANG is available at https://spang.dbcls.jp/.

**Keywords:** SPARQL · template engine · query library · one-liner · API

## 1 Introduction

An increasing number of datasets have been published in the Resource Description Framework (RDF). SPARQL is the standardized interface to RDF and contributes to the effective reuse of distributed RDF data. Although eco-systems have been developed for developing SPARQL queries, writing SPARQL codes is often a burden for Semantic Web developers and users. SPARQL queries have been written in many projects and are the result of developers' efforts. The value of the SPARQL queries accumulated thus far can be maximized by reusing SPARQL queries. For the effective construction of Semantic Web applications, the reuse of such queries is invaluable. However, owing to the limitation

```
Usage: spang2 [options] <SPARQL_TEMPLATE> [par1=val1 par2=val2 ...]

Options:
  -e, --endpoint <ENDPOINT>    target SPARQL endpoint
  -o, --outfmt <FORMAT>        tsv, json, n-triples (nt), turtle (ttl), rdf/xml (rdfxml), n3, xml, html
  -a, --abbr                   abbreviate results using predefined prefixes
  -v, --vars                   variable names are included in output (in the case of tsv format)
  -S, --subject <SUBJECT>      shortcut for specifying subject
  -P, --predicate <PREDICATE>  shortcut for specifying predicate
  -O, --object <OBJECT>        shortcut for specifying object
  -L, --limit <LIMIT>          LIMIT output (use by itself or with -[SPOF])
  -F, --from <FROM>            shortcut to search FROM specific graph (use alone or with -[SPOLN])
  -N, --number                 shortcut to COUNT results (use by itself or with -[SPO])
  -G, --graph                  shortcut for searching for graph names (use by itself or with -[SPO])
  -r, --prefix <PREFIX_FILES>  read prefix declarations (default: SPANG_DIR/etc/prefix,~/.spang/prefix)
  -n, --ignore                 ignore user-specific file (~/.spang/prefix) for test purpose
  -m, --method <METHOD>        GET or POST (default: "GET")
  -q, --show_query             show query and quit
  -f, --fmt                    format the query
  -i, --indent <DEPTH>         indent depth; use with --fmt (default: 2)
  -l, --list_nick_name         list available nicknames of endpoints and quit
  -d, --debug                  debug (output query embedded in URL, or output AST with --fmt)
  -V, --version                output version number
  -h, --help                   output usage information
```

**Fig. 1.** Usage of the SPANG command

of SPARQL specifications, such as the lack of a parameterization mechanism, the reuse of these queries remains limited.

We present SPANG and demonstrate its use for making SPARQL queries more reusable, thereby maximizing the value of the written SPARQL queries; we assign metadata to SPARQL queries, parameterize the SPARQL queries, assign a URI for each query, and construct a server for providing queries.

## 2 SPANG Overview

The SPANG project was commenced with the objective of maximizing the usability of SPARQL. SPANG was originally developed as a command-line SPARQL client [1], and it is now re-implemented in JavaScript using Node.js. Thus, it can not only be called in the command line but also in the JavaScript code used to construct a website.

## 3 Use Cases

The use cases illustrate the following issues involved in making SPARQL queries more reusable. An example query to the DisGeNET endpoint [2] is presented in Figure 2. This query retrieves genes associated with a specific disease. The example query includes metadata at the beginning of the query in the form of comment lines. The query also includes a parameter for a disease identifier with a default value of `C0751955` ("Brain Infarction").

```
# @title Get genes involved in a specific disease
# @endpoint http://rdf.disgenet.org/sparql/
# @prefix https://raw.githubusercontent.com/hchiba1/spang/master/prefix/bio
# @param arg1=C0751955

SELECT DISTINCT ?gene ?score ?gene_label ?source ?gda ?pmid ?description
WHERE {
    ?gda sio:SIO_000628 umls:{{arg1}} , ?gene ;
        a ?type ;
        sio:SIO_000253 ?source ;
        sio:SIO_000216/sio:SIO_000300 ?score .
    ?gene a ncit:C16612 ;
        rdfs:label ?gene_label .
    OPTIONAL {
        ?gda sio:SIO_000772 ?pmid ;
            dct:description ?description .
    }
}
ORDER BY DESC(?score) ?source ?pmid
```

**Fig. 2.** Example query in the SPANG template library

### 3.1 Use in command line

The Unix command-line environment is available as an interface for the SPANG template library. While the SPANG templates can be stored in the local system, users can also call templates that are publicly available across the Web if each query is assigned a dereferenceable URI. The following example command line gives a parameter to a local template query file for the DisGeNET endpoint.

```
spang2 disease_gene.rq disease=C0751955
```

The template can also be a URL that returns a SPANG template (`https://raw.githubusercontent.com/hchiba1/spang/master/library/disgenet/disease_gene.rq`). Thus, anyone can host the query libraries on GitHub. Other use cases include one-liner functionalities, such as

```
spang2 -e disgenet -N -O ncbigene:2247
```

where the query counts the number of triples that have `ncbigene:2247` as the object in the DisGeNET endpoint. The target endpoints and URI prefixes can be predefined via configuration files. The following command prints the internally generated SPARQL query according to the command-line options to the standard output.

```
spang2 -e disgenet -N -O ncbigene:2247 -q
```

Most of these one-liner functionalities are compatible with previous versions of SPANG [1]. The full usage of the SPANG command is illustrated in Figure 1.

### 3.2 Use through API

The SPARQL templates can also be called in the JavaScript code used to construct a website. The documentation is available at the project home page. Users can also execute templates that are predefined and provided by the SPANG server through the REST API. Accessing the following URI returns the result

of the query with the given parameter: `https://spang-portal.dbcls.jp/api/library/disgenet/disease_gene.rq?arg1=C0751955`. The following URI returns the list of queries in the library: `https://spang-portal.dbcls.jp/api/library/disgenet`.

Users can also obtain the results on the SPANG server by clicking on the web pages while browsing the templates on the server (`https://spang-portal.dbcls.jp/library/disgenet/disease_gene.rq`).

## 4 Implementation

The SPARQL templating syntax used in the SPANG framework is an extension of sparql-doc [3]. Although the use of sparql-doc was originally aimed at the documentation of SPARQL, this notation is suitable for annotating each SPARQL query with metadata to increase the usability of the query. Each query, with the metadata added as comment lines, can be used as a standard SPARQL query. Furthermore, if a parser is implemented for the comment lines, users can utilize the metadata to search for useful information about the query. Herein, we further parameterize the SPARQL queries and consider the parameters of the query as the metadata of SPARQL. As a syntax for parameterization in SPARQL code, we use mustache notation [4].

We implemented the SPANG template engine in JavaScript, where the template is transformed into an abstract syntax tree, processed using the given parameters, and then reconstructed into SPARQL to be submitted to an endpoint. We used Parsing Expression Grammar (PEG) [5] expression of the SPARQL grammar based on the EBNF of the SPARQL specifications [6]. PEG.js [7] is a parser generator, and thus, we could generate a custom parser based on a PEG description. The original implementation of PEG.js of the SPARQL grammar is in the rdfstore-js project (`https://github.com/antoniogarrote/rdfstore-js`). The PEG.js code was modified to fit the use in the case of the SPANG framework. While other efforts have been made to implement the parser and formatter of SPARQL [8], we developed a parser and formatter that stringify the generated abstract syntax tree based on the PEG expression.

We also developed a SPANG server, wherein the SPARQL queries can be executed in a browser or through a REST API (`https://spang.dbcls.jp/`). The SPANG server is developed using Ruby on Rails, and it can run on Docker. Thus, users can also run a SPANG server on their own.

## 5 Related Work

SPARQL templating mechanisms have been implemented in various projects. SPARQL libraries, such as Bioqueries [9], have been implemented based on templating mechanisms. Bioqueries comprise an effort to collect SPARQL queries as a community attempt, contributing to the increased findability of the queries. However, the users should still copy the queries into their own software in specific templating formats to reuse the queries. In this paper, we present SPANG

framework, where we attempt to build a software network with solid programming interfaces based on the Semantic Web platform.

## References


1. Chiba, H., Uchiyama, I. (2017). SPANG: a SPARQL client supporting generation and reuse of queries for distributed RDF databases. BMC bioinformatics, 18(1), 93.
2. Queralt-Rosinach, N., Pinero, J., Bravo, À., Sanz, F., Furlong, L. I. (2016). DisGeNET-RDF: harnessing the innovative power of the Semantic Web to explore the genetic basis of diseases. Bioinformatics, 32(14), 2236-2238.
3. sparql-doc: Generate HTML documentation from SPARQL queries. `https://github.com/ldodds/sparql-doc`
4. Logic-less templates. https://mustache.github.io/
5. Ford, B. (2004, January). Parsing expression grammars: a recognition-based syntactic foundation. In Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages (pp. 111-122).
6. SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013. `http://www.w3.org/TR/sparql11-query/`
7. PEG.js – Parser Generator for JavaScript. `https://pegjs.org/`
8. SPARQL.js – A SPARQL 1.1 parser for JavaScript. `https://github.com/RubenVerborgh/SPARQL.js`
9. García-Godoy, M. J., Navas-Delgado, I., Aldana-Montes, J. (2011, December). Bioqueries: a social community sharing experiences while querying biological linked data. In Proceedings of the 4th international workshop on semantic web applications and tools for the life sciences (pp. 24-31).