SPARQL-LD

Overview

A constantly increasing number of data providers publish their data on the Web in the RDF format as Linked Data. SPARQL is the standard query language for retrieving and manipulating RDF data. However, the majority of SPARQL implementations requires the data to be available in advance (in main memory or in a repository), not exploiting thereby the real-time and dynamic nature of Linked Data.

SPARQL-LD is an extension (actually a generalization) of SPARQL 1.1 that allows to directly fetch and query RDF data from any Web source. Using **SPARQL-LD** one can even query a dataset coming from the partial results of a query (i.e., discovered at query execution time) or RDF data that is dynamically created by Web Services. Such a functionality motivates Web publishers to adopt the Linked Data principles and enrich their digital contents and services with RDF, since their data is made directly accessible and exploitable via SPARQL (without needing to set up and maintain an endpoint).

Using **SPARQL-LD** one can directly exploit and combine in the same SPARQL query:

- data stored in the (local) repository
- data coming from online RDF (in any standard format) or JSON-LD files (http://json-ld.org/)
- data embedded in Web pages as RDFa (https://rdfa.info/) or JSON-LD "islands"
- data coming from dereferenceable URIs
- data that is dynamically created by Web Services (returning RDF data)
- data coming by querying other SPARQL endpoints

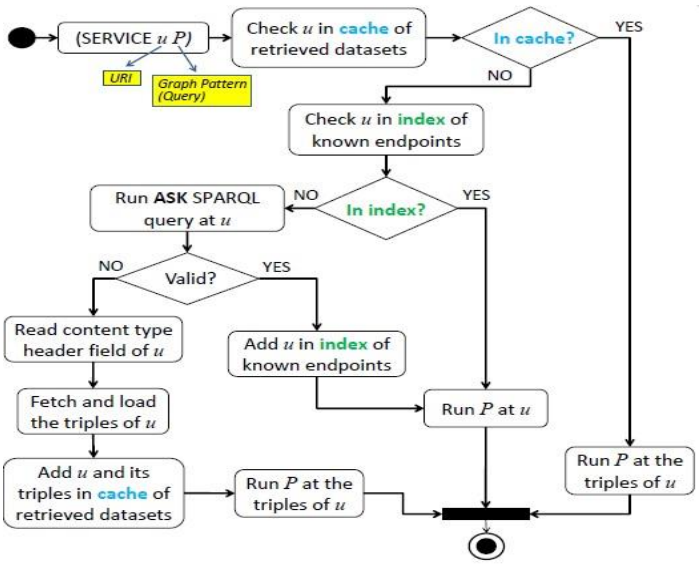
Target Domains

The functionality offered by **SPARQL-LD** can motivate **Web publishers** to enrich their documents and digital libraries with RDF since it makes their data directly accessible via SPARQL without needing to set up and maintain a costly SPARQL endpoint (e.g., they can just publish RDF files). For instance, a museum can enrich its Web page with RDFa or just put online an RDF dump, and thereby make its data directly accessible and queryable via SPARQL.

Description and Examples

SPARQL-LD extends the applicability of the SERVICE operator of SPARQL 1.1 Federated Query. Using the extended SERVICE operator, we can fetch and query RDF data from any HTTP web resource. This extension does not require the named graphs to have been declared, thus one can even fetch and query a dataset returned by a portion of the query (i.e., whose URI is derived at query execution time).

The query execution process is described in the following figure:



The execution process of as SERVICE graph pattern.

Examples

❖ Query RDFa, an external endpoint, and dereferenceable URIs derived at query-execution time:

```
SELECT DISTINCT ?authorURI (count(?paper) AS ?numOfPapers) (count(?series) AS ?numofDiffConfs) WHERE {
      SERVICE <http://users.ics.forth.gr/~fafalios> { ?p <http://purl.org/dc/terms/creator> ?authorURI }   <- RDFa Web page
      SERVICE <http://dblp.l3s.de/d2r/sparql> {   <- SPARQL endpoint
            ?p2 <http://purl.org/dc/elements/1.1/creator> ?authorURI; <http://swrc.ontoware.org/ontology#series> ?series . }
      SERVICE ?authorURI {  ?paper < http://purl.org/dc/elements/1.1/creator> ?authorURI }   <- Dereferenceable URIs derived at query-execution time
} GROUP BY ?authorURI ORDER BY DESC(?numOfPapers)
```

The query returns all co-authors of P. Fafalios together with the number of their publications and the number of distinct conferences in which they have a publication.

❖ Parameterize and call a named-entity recognition Web Service at query-execution time

```
SELECT DISTINCT ?detectedEntity ?categoryName (count(?position) AS ?numOfOccurences) WHERE {
      SERVICE <http://dbpedia.org/resource/Thunnus> {   <- Derefereceable URI
            dbpedia:Thunnus dbpedia-owl:wikiPageExternalLink ?page }
      VALUES ?temp1 {<http://139.91.183.72/x-link-marine/api?categories=fish;country&url=PAGE>}
      BIND(REPLACE(str(?temp1), "PAGE", str(?page), "i") as ?x) BIND(URI(?x) as serv)
      SERVICE ?serv { ?annot oa:hasBody ?ent. ?ent oae:regardsEtityName ?detectedEntity;   <- Named-entity recognition Web Service returning RDF
            oae:position ?position; oae:belongsTo ?category. ?category rdfs:label ?categoryName}
} GROUP BY ?detectedEntity ?categoryName ORDER BY DESC(?numOfOccurences)
```

The query first retrieves Web pages related to the fish genus Thunnus, then it calls a named-entity recognition service (X-Link) for identifying (at request time) names of fishes and countries in these Web pages, and for each detected entity the query retrieves (and shows) its name, its category and its number of occurrences in the Web pages.

Additional Information

More **information** about **SPARQL-LD** can be found at:
https://github.com/fafalios/sparql-ld

SPARQL-LD was partially **supported** by the H2020 EU project BlueBRIDGE.

Contact details: **Yannis Tzitzikas**
tzitzik@ics.forth.gr
www.ics.forth.gr/isl