Querying non-RDF Datasets using Triple Patterns

Benjamin Moreau[2], Patricia Serrano-Alvarado[1], Emmanuel Desmontils[1], and David Thoumas[2]

[1] GDD-LS2N – Nantes University, France {Name.LastName@}univ-nantes.fr
[2] OpenDataSoft {Name.Lastname}@opendatasoft.com

Abstract. Triple Pattern Fragments (TPF) interface allows to query Linked Data datasets with high data availability. But, data providers do not integrate large amounts of datasets as Linked Data due to expensive investments in terms of storage and maintenance. The problem we focus on is how to integrate non-RDF datasets on-demand as Linked Data simply and efficiently. In this demo, we present ODMTP, an On-Demand Mapping using Triple Patterns over non-RDF datasets. ODMTP is implemented over a TPF server. We showcase it with SPARQL queries over Twitter.

1 Introduction

Triple Pattern Fragments (TPF) interface [1] allows to query Linked Data datasets with high availability. One of the reasons of server availability is the simplicity of the server interface. TPF clients decompose SPARQL queries in *triple pattern queries* that are sent to a TPF server. Thus an important part of the query execution is on the client side.

Despite its benefits, large amounts of open datasets are not stored as Linked Data. Data providers continue storing in relational, document or column oriented formats. Replacing these datasets with triple-stores implies significant investment in time, storage, maintainability, etc.

The problem we focus on is how to integrate non-RDF datasets on-demand as Linked Data simply and efficiently using triple patterns.

Many works have been proposed to map relational databases to RDF [2]. Recent work focuses on accessing MongoDB documents on-the-fly [3]. The problem with this approach is that it proposes a translation of a complete SPARQL query and we need simply the translation of one triple pattern.

We propose ODMTP, an On-Demand Mapping using Triple Patterns over non-RDF datasets. To illustrate our approach, in this demo we show an implementation of ODMTP to query Twitter with SPARQL queries.

2 ODMTP: On-Demand Mapping using Triple Patterns

Figure 1 presents the global architecture of ODMTP. TPF clients receive SPARQL queries and decompose them into several http requests which are sent to a TPF server. Such requests, that we call triple pattern queries (TPQs), contain a triple pattern and a page number.

We propose to modify the TPF server such that, instead of evaluating TPQs over a RDF store, it sends TPQs to ODMTP.

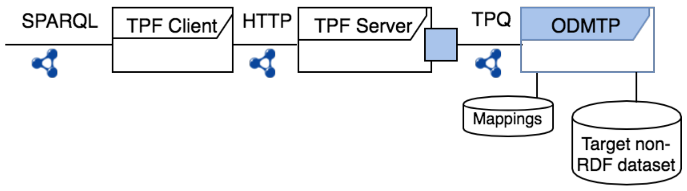

Fig. 1. ODMTP global architecture

When ODMTP receives a TPQ, it translates it into the target query language and sends it to the target non-RDF dataset. The query engine of the non-RDF dataset evaluates this query and returns the resultset to ODMTP. ODMTP then translates this answer in triples, it constructs fragments and sends them to the TPF server. Fragments are composed of triples, metadata and controls (total number of triples matching the triple pattern, number of triples per page, link to the next page, etc.).

ODMTP translations are possible thanks to a mapping file for the corresponding target dataset. Several mapping files can be defined for a target dataset depending on the needs of semantic applications.

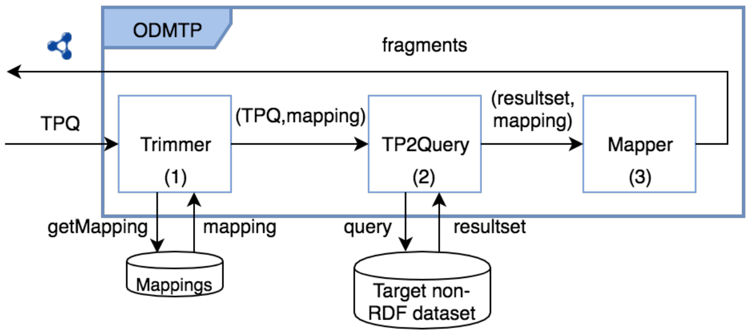

Fig. 2. ODMTP

Figure 2 shows more details about ODMTP. The trimmer component receives a TPQ and from a mapping file it extracts the mappings pertinent to the triple pattern. The mapping file maps triple patterns (i.e., subject, predicate, or object) with columns or attributes of the target non-RDF dataset. Any mapping language can be used to define this mapping file as long as it facilitates triple pattern matching and the mapping of heterogeneous data formats into RDF.

Then, the TPQ and its corresponding mapping is received by TP2Query that uses the mapping to translate the triple pattern into a query in the query language of the non-RDF dataset. TP2Query communicates with the query engine of the non-RDF dataset and obtains the resultset corresponding to TPQ. It also estimates the total number of triples matching the triple pattern. The implementation of this component depends on the non-RDF dataset.

Finally, the Mapper component uses the mapping to translate the resultset in triples. And it produces the fragment that is sent to the TPF server.

The challenge is having a mapping file expressive enough for the semantic application purposes. Besides mapping triple patterns into the target columns or attributes, this mapping should semantically enrich the target dataset.

The overhead produced by ODMTP is tightly related to the performance of the query engine of the target non-RDF dataset.

We evaluated a Python implementation of ODMTP over several datasets stored in JSON and queried through Elasticsearch. Cardinality of datasets goes from 100,000 to 500,000 triples. Experiments were executed locally on a computer with a processor intel i7 2,5 GHz with 16 GB of RAM.

We measured the time to evaluate the last page (i.e., the potentially most expensive result page) of the 8 types of triple patterns, i.e., {?s ?p ?o}, {?s URI ?o}, {?s ?p URI}, {?s URI URI}, {URI URI URI}, etc. We compared these performances with those of a traditional TPF server using HDTs. ODMTP produced a constant overhead (on average +0,250 seconds). The reason of this limited overhead is the set of efficient indexes produced by Lucene[3] that is used by Elasticsearch.

3 Demo

In this demo we show a Python implementation of ODMTP over Twitter. Twitter, as Elasticsearch uses Lucene for real-time search. This implementation is available at `https://github.com/benjimor/odmtp-tpf`.

We used the TwitterApi package and xR2RML [4] as mapping language. The TPF server is setup locally as a django application. Once installed, it can receive requests from TPF clients (e.g., `http://client.linkeddatafragments.org/`).

Our implementation is a proof of concept that shows how it is possible to evaluate SPARQL queries using triple patterns. For the moment we consider only triple patterns with a variable in the subject. Predicates in our mapping consider `schema:author, schema:dateCreated, it:includedHashtag, it:includedUrl`. Objects must have corresponding data types.

Consider TPQ1= `<?s it:includedHashtag "iswc2017", page=1>`. When ODMTP receives TPQ1 the Trimmer identifies the corresponding triples from the mapping file, in this case the mappings shown in Listing 1.1. Then, TP2Query translates the triple pattern into the Twitter query `https://api.twitter.com/1.1/search/tweets.json?q=%23iswc2017&count=15` and sends it to the Twitter API (here it asks for 15 entries). Twitter returns results in JSON containing

[3] `https://lucene.apache.org/`

a link to the next result page. TP2Query follows this link until it retrieves the pertinent page. Then the Mapper constructs the triples and the fragment.

A more expressive mapping would consider other triple pattern types and more semantics for Twitter data.

During the demo, users will be able to test several SPARQL queries taken into account by our mapping. A video is available at `https://youtu.be/wruH8teK9tU`.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix xrr: <http://www.i3s.unice.fr/ns/xr2rml#> .
@prefix schema: <http://schema.org/> .
@prefix it: <http://www.influencetracker.com/ontology#> .

<#Tweets>
xrr:logicalSource [
  xrr:query "https://api.twitter.com/1.1/";
  rml:iterator "$.statuses.*";
];
rr:subjectMap [
  rr:template "https://twitter.com/statuses/{$.id_str}";
  rr:class schema:SocialMediaPosting;
];
rr:predicateObjectMap [
  rr:predicate it:includedHashtag;
  rr:objectMap [xrr:reference "$.entities.hashtags.[*].text";];
]
...
```

Listing 1.1. Excerpt of the mapping file for Twitter

4 Conclusion and perspectives

ODMTP shows how to integrate non-RDF datasets as Linked Data through triple patterns. It is an adaptable and modular framework that does not pretend to make non-RDF datasets compliant to Linked Data principles. Aspects like dereferencable URIs should complete our approach for instance via dynamic html pages.

We will use our implementation of ODMTP over Elasticsearch to integrate datasets provided by the OpenDataSoft platform. OpenDataSoft provides hundreds of open datasets that will benefit of the Web of Data without changing the storage format.

References

1. R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert, “Triple Pattern Fragments: A Low-cost Knowledge Graph Interface for the Web,” *Journal of Web Semantics*, vol. 37–38, 2016.
2. S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. Thibodeau Jr, S. Auer, J. Sequeda, and A. Ezzat, “A Survey of Current Approaches for Mapping of Relational Databases to RDF,” *W3C RDB2RDF Incubator Group Report*, vol. 1, pp. 113–130, 2009.
3. F. Michel, C. Faron-Zucker, and J. Montagnat, “A Mapping-based Method to Query MongoDB Documents with SPARQL,” in *Int. Conference on Database and Expert Systems Applications (DEXA)*, pp. 52–67, Springer, 2016.
4. F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat, “Translation of Relational and Non-relationalDdatabases into RDF with xR2RML,” in *Int. Conference on Web Information Systems and Technologies (WEBIST)*, pp. 443–454, 2015.