Triple Pattern Fragments

Background

Triple Pattern Fragments is a form of Linked Data Fragments (see References) for querying a triple store to retrieve a set of triples matching a specified pattern. The pattern is always of the form subject predicate object, where any or all of the elements of the pattern may be unspecified, that is, wildcards.

SPARQL and Linked Data Fragments, alternatives to Triple Pattern Fragments, are very powerful, and as a result, can generate queries that take a long time to run, slowing the server, and in some cases, making the server unavailable. This results in sites shutting down their access points for fear of losing availability to long-running queries. Triple Pattern Fragments solves this problem by allowing only one kind of query, the pattern. Pattern matching is indexed and very fast, insuring the servers remain available while handling queries.

For example the pattern <uri> * * finds all the triples which have the specified URI as a subject. The pattern * <uri> * finds all the triples with the specified predicate, and * * <uri> finds all the triples with the specified object.

Triple pattern fragments is a very fast, very simple means for querying a triple store. The triple pattern fragments API in VIVO puts little load on the server, providing a simple means for getting data from the triple store. The API has a web interface for manual use, can be used from the command line via curl, and can be used by programs. Each mode of usage is described below.

Open API

⚠ Triple Pattern Fragments, as delivered in VIVO, is an open API. This means that anyone, and any software can access the Triple Pattern Fragments endpoint of your VIVO without logging on, that is, without authorization. All the data in your full graph is accessible to the API and to those who use it. VIVO is built for data sharing, and the Triple Pattern Fragments API makes it very easy for your VIVO to share data with others. **Please be sure your VIVO does not contain restricted data that should not be shared with others**

VIVO uses the LinkedDataFragments Server, available on GitHub here: https://github.com/LinkedDataFragments/Server.Java

Configuration

Since the TPF endpoint does not currently enforce visibility settings for classes or properties set in VIVO, the service is turned off by default. In order to enable the TPF endpoint, a "tpf.activeFlag" property must be added to the runtime.properties file with a value of "true". For this property to take effect, the Tomcat server must be restarted.

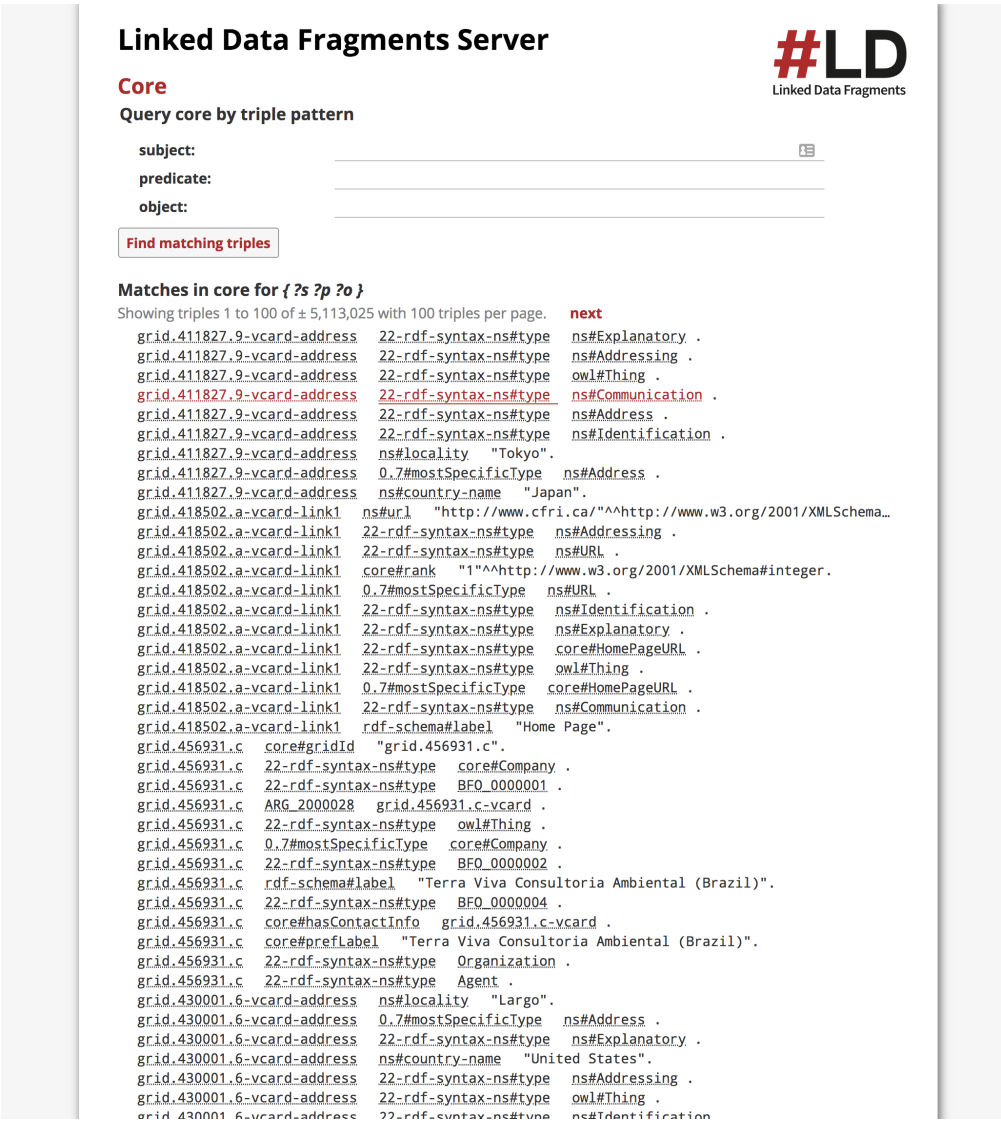
```
tpf.activeFlag = true
```

Manual Query

Manual query can be used to view triples, and to run SPARQL queries resolved as Triple Pattern Fragments. These methods are view data. To save data, use curl, or Programmatic Access.

View Triples using Triple Pattern Fragments

To use TPF manually, visit the TPF endpoint of a VIVO, http://yourvivo/tpf/core, where "yourvivo" is the web address of the VIVO of interest. In the example below we use http://openvivo.org/tpf/core. You will see:



Linked Data Fragments Server

#LD Linked Data Fragments

Core

Query core by triple pattern

subject:

predicate:

object:

Find matching triples

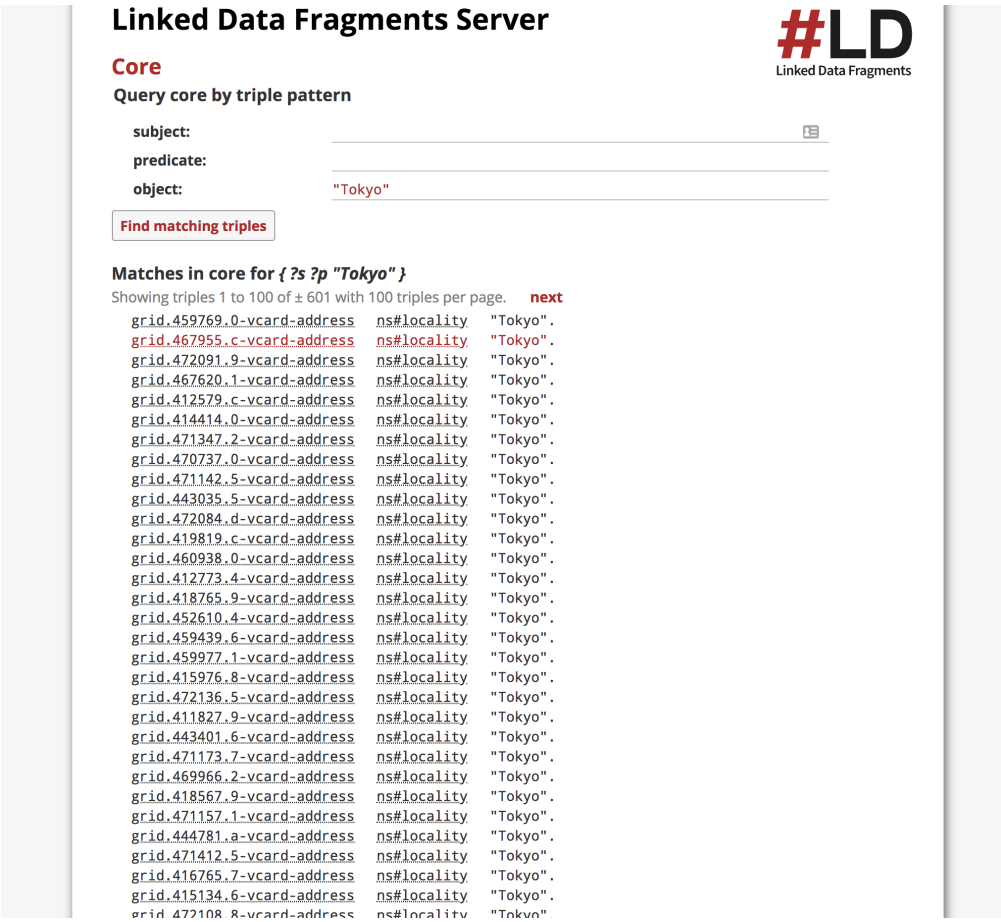
Matches in core for *{ ?s ?p ?o }*

Showing triples 1 to 100 of ± 5,113,025 with 100 triples per page. **next**

```
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   ns#Explanatory .
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   ns#Addressing .
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   owl#Thing .
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   ns#Communication .
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   ns#Address .
grid.411827.9-vcard-address   22-rdf-syntax-ns#type   ns#Identification .
grid.411827.9-vcard-address   ns#locality   "Tokyo".
grid.411827.9-vcard-address   0.7#mostSpecificType   ns#Address .
grid.411827.9-vcard-address   ns#country-name   "Japan".
grid.418502.a-vcard-link1   ns#url   "http://www.cfri.ca/"^^http://www.w3.org/2001/XMLSchema…
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   ns#Addressing .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   ns#URL .
grid.418502.a-vcard-link1   core#rank   "1"^^http://www.w3.org/2001/XMLSchema#integer.
grid.418502.a-vcard-link1   0.7#mostSpecificType   ns#URL .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   ns#Identification .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   ns#Explanatory .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   core#HomePageURL .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   owl#Thing .
grid.418502.a-vcard-link1   0.7#mostSpecificType   core#HomePageURL .
grid.418502.a-vcard-link1   22-rdf-syntax-ns#type   ns#Communication .
grid.418502.a-vcard-link1   rdf-schema#label   "Home Page".
grid.456931.c   core#gridId   "grid.456931.c".
grid.456931.c   22-rdf-syntax-ns#type   core#Company .
grid.456931.c   22-rdf-syntax-ns#type   BFO_0000001 .
grid.456931.c   ARG_2000028   grid.456931.c-vcard .
grid.456931.c   22-rdf-syntax-ns#type   owl#Thing .
grid.456931.c   0.7#mostSpecificType   core#Company .
grid.456931.c   22-rdf-syntax-ns#type   BFO_0000002 .
grid.456931.c   rdf-schema#label   "Terra Viva Consultoria Ambiental (Brazil)".
grid.456931.c   22-rdf-syntax-ns#type   BFO_0000004 .
grid.456931.c   core#hasContactInfo   grid.456931.c-vcard .
grid.456931.c   core#prefLabel   "Terra Viva Consultoria Ambiental (Brazil)".
grid.456931.c   22-rdf-syntax-ns#type   Organization .
grid.456931.c   22-rdf-syntax-ns#type   Agent .
grid.430001.6-vcard-address   ns#locality   "Largo".
grid.430001.6-vcard-address   0.7#mostSpecificType   ns#Address .
grid.430001.6-vcard-address   22-rdf-syntax-ns#type   ns#Explanatory .
grid.430001.6-vcard-address   ns#country-name   "United States".
grid.430001.6-vcard-address   22-rdf-syntax-ns#type   ns#Addressing .
grid.430001.6-vcard-address   22-rdf-syntax-ns#type   owl#Thing .
grid.430001.6-vcard-address   22-rdf-syntax-ns#type   ns#Identification
```

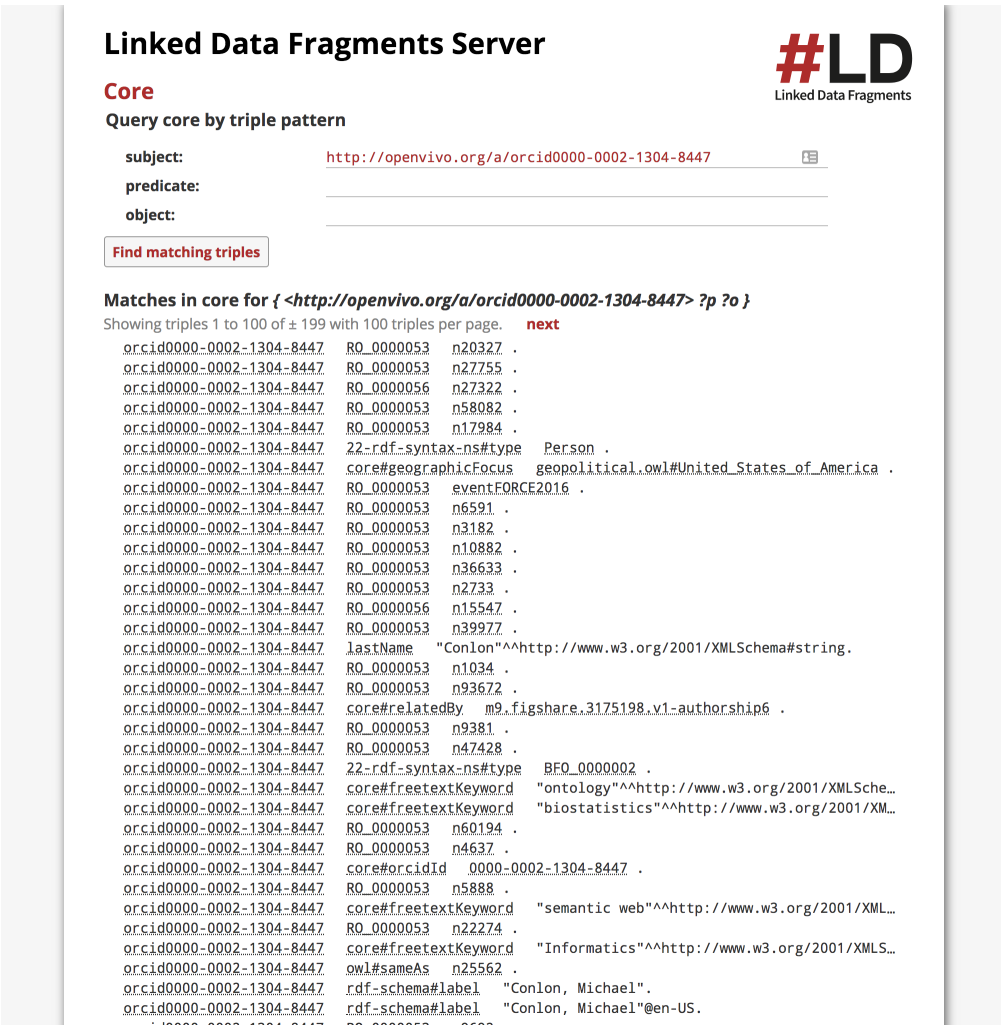
Notice that results are returned for the triple pattern fragment * * *. More than 5 million triples were returned, with the first hundred being displayed on the web page. Pressing "next" at the top of the list of triples will display the next hundred triples. Each of the rows in the display is a triple in the full graph. The TPF web page uses a display with simplifies the presentation of URI. Many of the URI are shortened, not with the use of prefixes, but merely by intelligently truncating the URI.

Each of the elements in each of the triples is a link. Clicking a link will issue a TPF query with the selected element as the specified value in a triple. For example, clicking on "Tokyo" in the example above, generates a TPF request of the form * * "Tokyo" – that is, find all the triples that have the text string "Tokyo" as an object. The result is shown below:



We see that OpenVIVO has 601 triples with "Tokyo" as an object. The displayed triples show Tokyo as a locality in an address. Putting double quotes around literal values is required. TPF supports the use of language tags to select literal values with specific language tags.

To specify a URI in a pattern, give the full URI (no prefix, no truncation) with no brackets. For example, to find all the triples with a subject of http://openvivo.org/a/orcid0000-0002-1304-8447, put the URI in the subject field and leave predicate field and the object field empty. See below:



Linked Data Fragments Server

#LD Linked Data Fragments

Core

Query core by triple pattern

subject: http://openvivo.org/a/orcid0000-0002-1304-8447

predicate:

object:

Find matching triples

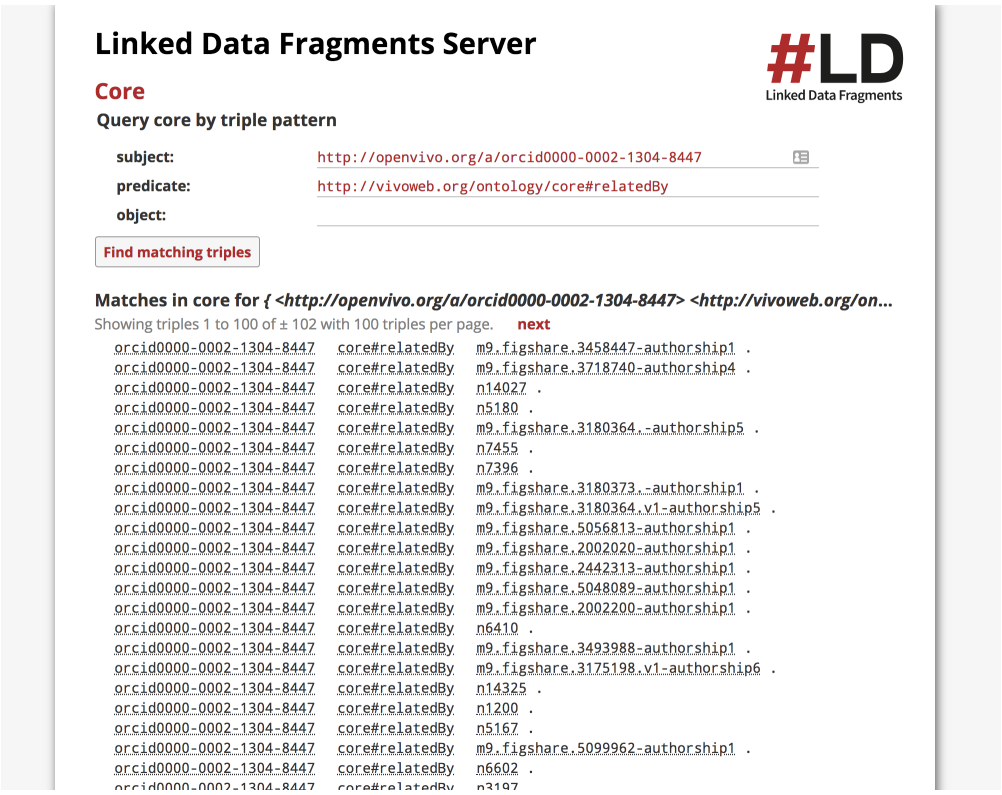
Matches in core for ***{ <http://openvivo.org/a/orcid0000-0002-1304-8447> ?p ?o }***

Showing triples 1 to 100 of ± 199 with 100 triples per page. **next**

```
orcid0000-0002-1304-8447  RO_0000053  n20327 .
orcid0000-0002-1304-8447  RO_0000053  n27755 .
orcid0000-0002-1304-8447  RO_0000056  n27322 .
orcid0000-0002-1304-8447  RO_0000053  n58082 .
orcid0000-0002-1304-8447  RO_0000053  n17984 .
orcid0000-0002-1304-8447  22-rdf-syntax-ns#type  Person .
orcid0000-0002-1304-8447  core#geographicFocus  geopolitical.owl#United_States_of_America .
orcid0000-0002-1304-8447  RO_0000053  eventFORCE2016 .
orcid0000-0002-1304-8447  RO_0000053  n6591 .
orcid0000-0002-1304-8447  RO_0000053  n3182 .
orcid0000-0002-1304-8447  RO_0000053  n10882 .
orcid0000-0002-1304-8447  RO_0000053  n36633 .
orcid0000-0002-1304-8447  RO_0000053  n2733 .
orcid0000-0002-1304-8447  RO_0000056  n15547 .
orcid0000-0002-1304-8447  RO_0000053  n39977 .
orcid0000-0002-1304-8447  lastName  "Conlon"^^http://www.w3.org/2001/XMLSchema#string.
orcid0000-0002-1304-8447  RO_0000053  n1034 .
orcid0000-0002-1304-8447  RO_0000053  n93672 .
orcid0000-0002-1304-8447  core#relatedBy  m9.figshare.3175198.v1-authorship6 .
orcid0000-0002-1304-8447  RO_0000053  n9381 .
orcid0000-0002-1304-8447  RO_0000053  n47428 .
orcid0000-0002-1304-8447  22-rdf-syntax-ns#type  BFO_0000002 .
orcid0000-0002-1304-8447  core#freetextKeyword  "ontology"^^http://www.w3.org/2001/XMLSche…
orcid0000-0002-1304-8447  core#freetextKeyword  "biostatistics"^^http://www.w3.org/2001/XM…
orcid0000-0002-1304-8447  RO_0000053  n60194 .
orcid0000-0002-1304-8447  RO_0000053  n4637 .
orcid0000-0002-1304-8447  core#orcidId  0000-0002-1304-8447 .
orcid0000-0002-1304-8447  RO_0000053  n5888 .
orcid0000-0002-1304-8447  core#freetextKeyword  "semantic web"^^http://www.w3.org/2001/XML…
orcid0000-0002-1304-8447  RO_0000053  n22274 .
orcid0000-0002-1304-8447  core#freetextKeyword  "Informatics"^^http://www.w3.org/2001/XMLS…
orcid0000-0002-1304-8447  owl#sameAs  n25562 .
orcid0000-0002-1304-8447  rdf-schema#label  "Conlon, Michael".
orcid0000-0002-1304-8447  rdf-schema#label  "Conlon, Michael"@en-US.
```

199 triples are returned. Each has the specified subject. Data managers might note:

1. There are many RO_0000053 predicates. These are "bearer_of" role assertions. See Ontology Reference for diagrams showing how VIVO uses roles and represents information. TPF can be a very good tools for exploring the data and learning about information representation.
2. The sixth triple is an assertion that the subject in question is a person. At the bottom of the screen shot we see that the person has two labels. "Michael Conlon" and "Michael Conlon"@en-US. Is this something that is expected, or something that should be changed? TPF makes a good tool for discussing VIVO data practices with others.
3. We see that the person is relatedBy an authorship. How many relatedBy assertions does this person have? We could issue a TPF query for the subject and for related by as a predicate. The results are shown below:



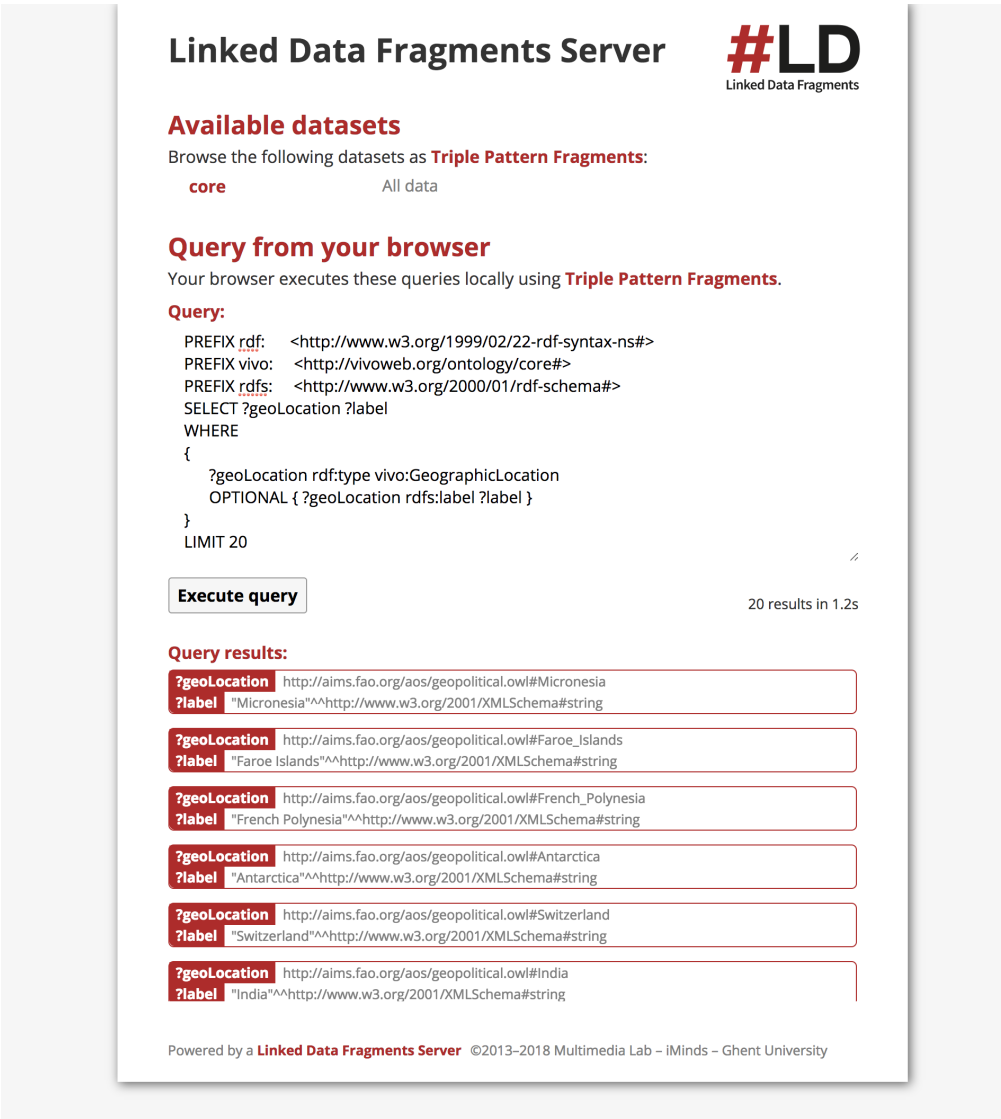
We see 102 triples are returned. Each indicates that the person is relatedBy to something else. We see some of the objects appear to be figshare related, others appear to be authorships, while still others non informative. Additional exploration might help us understand how the relatedBy assertions are used.

SPARQL Queries Resolved as Triple Pattern Fragments

The Linked Data Fragments server can also resolve full SPARQL queries. The queries are decomposed into a series of TPF requests behind the scenes in the browser. The VIVO server sees only TPF requests. Each TPF request is handled quickly as previously described. To issue a SPARQL query using Triple Pattern Fragments, go to http://yourvivo/tpf where "yourvivo" is the web address of your VIVO. You will see a screen such as that below. Type in your query. You will need to provide the prefixes used in your query, as shown below. Press Execute, and the query is resolved a series of TPF queries. Results are presented dynamically.

To try this yourself, you can use the Linked Data Fragments Server of OpenVIVO, available here: http://openvivo.org/tpf

Note in the example that prefixes are supplied as part of the query. The Triple Pattern Fragments server has no knowledge of VIVO prefixes. These must be supplied with the query.



Curl

curl can be used to issue triple pattern fragment queries and return RDF/XML. For example:

```
curl http://openvivo.org/tpf/core
```

returns 169 lines of RDF/XML, output truncated below

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    ...
    xmlns:vivo="http://vivoweb.org/ontology/core#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:vitro-public="http://vitro.mannlib.cornell.edu/ns/vitro/public#">
  <void:Dataset rdf:about="http://openvivo.org/tpf/core#dataset">
    <hydra:search rdf:parseType="Resource">
      <hydra:template>http://openvivo.org/tpf/core{?subject,predicate,object}</hydra:template>
      <hydra:mapping rdf:parseType="Resource">
        <hydra:variable>subject</hydra:variable>
        <hydra:property rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#subject"/>
      </hydra:mapping>
      <hydra:mapping rdf:parseType="Resource">
        <hydra:variable>predicate</hydra:variable>
        <hydra:property rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate"/>
      </hydra:mapping>
      <hydra:mapping rdf:parseType="Resource">
        <hydra:variable>object</hydra:variable>
        <hydra:property rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#object"/>
      </hydra:mapping>
    </hydra:search>
    <rdf:type rdf:resource="http://www.w3.org/ns/hydra/core#Collection"/>
    <void:subset>
      <hydra:Collection rdf:about="http://openvivo.org/tpf/core">
        <hydra:firstPage rdf:resource="http://openvivo.org/tpf/core?page=1"/>
        <hydra:nextPage rdf:resource="http://openvivo.org/tpf/core?page=2"/>
        <rdf:type rdf:resource="http://www.w3.org/ns/hydra/core#PagedCollection"/>
        <void:triples rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
        >5113025</void:triples>
        <hydra:totalItems rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
        >5113025</hydra:totalItems>
        <hydra:itemsPerPage rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
        >100</hydra:itemsPerPage>
      </hydra:Collection>
    </void:subset>
    <hydra:itemsPerPage rdf:datatype="http://www.w3.org/2001/XMLSchema#long"
    >100</hydra:itemsPerPage>
  </void:Dataset>
  <vivo:Company rdf:about="http://openvivo.org/a/grid.456931.c">
    <vivo:gridId>grid.456931.c</vivo:gridId>
    <rdf:type rdf:resource="http://purl.obolibrary.org/obo/BFO_0000001"/>
    <obo:ARG_2000028 rdf:resource="http://openvivo.org/a/grid.456931.c-vcard"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <j.0:mostSpecificType rdf:resource="http://vivoweb.org/ontology/core#Company"/>
    <rdf:type rdf:resource="http://purl.obolibrary.org/obo/BFO_0000002"/>
    <rdfs:label>Terra Viva Consultoria Ambiental (Brazil)</rdfs:label>
    <rdf:type rdf:resource="http://purl.obolibrary.org/obo/BFO_0000004"/>
    <vivo:hasContactInfo rdf:resource="http://openvivo.org/a/grid.456931.c-vcard"/>
    <skos:prefLabel>Terra Viva Consultoria Ambiental (Brazil)</skos:prefLabel>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Organization"/>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
  </vivo:Company>
  ...
</rdf:RDF>
```


Notes:

1. The return is RDF/XML containing the first 100 triples of the result set. We see a company being described with a series of assertions.
2. The return contains a description of the returned data. There is a void:Dataset description containing information about the query, its size (5,113,025 triples), and that 100 are included. The description also contains URL than can be used to navigate to the next page and first page in the result set.

IRI Patterns

The following IRI patterns are valid for making TPF requests from curl or from software (see below).

```
http://example.org/tpf/core?subject={subject}&predicate={predicate}&object={object}
http://example.org/tpf/core?s={subject}&p={predicate}&o={object}
```

For example:

```
curl http://openvivo.org/tpf/core?subject=http://openvivo.org/a/orcid0000-0002-1304-8447
```

Returns an RD/XML document containing the first 100 triples regarding the specified subject.

Headers

Headers can be used to specify the output format.

```
curl -H "Accept: application/n-triples; charset=utf-8" http://openvivo.org/tpf/core?subject=http://openvivo.org
/a/orcid0000-0002-1304-8447
```

Which returns 166 triples as of this writing.

Programmatic Access

Programmatic access follows the same approach as curl. Issue an HTTP request for the specified data. Here's a simple Javascript JQuery Ajax code fragment making a TPF request. The code expects a siteUri, a subjectUri and a predicateUri, and returns all triples with the specified subjectUri and predicateUri. A dataFilterFunction is called on the return prior to the successFunction being called.

```
        $.ajax({
            headers: {Accept : "application/n-triples; charset=utf-8"},
            url: siteUri,
            data: {subject: subjectUri, predicate: predicateUri, object: "", page: "1"},
            dataFilter: function(data) { dataFilterFunction(data); },
            success: function(data) { successFunction(data); }
            });
```

References

1. Linked Data Fragments In-depth http://linkeddatafragments.org/in-depth/
2. Triple Pattern Fragments specification http://www.hydra-cg.com/spec/latest/triple-pattern-fragments/
3. Verborgh, R. et al. Triple Pattern Fragments: A low cost knowledge graph interface for the web https://www.sciencedirect.com/science/article/pii/S1570826816000214?via%3Dihub
4. Verborgh, R. The Future is Federated. Invited presentation at 2016 VIVO Conference, Denver, Colorado. http://openvivo.org/display/doi10.6084/m9.figshare.3680310
5. LinkedDataFragments Server. Github. https://github.com/LinkedDataFragments/Server.Java