



# Linked Data Fragments

Ruben Verborgh



**Semantic Web:**

**SOLUTION**

**looking for**

**PROBLEM?**

**Semantic Web:**

~~**SOLUTION**~~

**looking for**

**PROBLEM?**

# ***PROBLEM***

**How to read and write  
data from different sources  
on the Web?**

# ***SOLUTION***

**A scalable way  
to query data  
on the Web.**

***Web interfaces to Linked Data***

***A different balance of trade-offs***

***Real-world federated queries***

# The **Resource Description Framework** captures facts as triples.

`</articles/www> a schema:ScholarlyArticle.`

`</articles/www> schema:name "The World-Wide Web".`

`</articles/www> schema:author </people/timbl>.`

`</articles/www> schema:author </people/cailliau>.`

`</articles/www> schema:author </people/groff>.`

**SPARQL** is a language (*and protocol*)  
to query RDF datasources.

```
SELECT * WHERE {
```

```
  ?article a schema:ScholarlyArticle.
```

```
  ?article schema:author ?author.
```

```
  ?author schema:name "Tim Berners-Lee".
```

```
}
```



**A SPARQL endpoint lets clients execute SPARQL queries over HTTP.**

**The server has a triple store.**

**The client sends a query to the server.**

**The server executes the query and sends back the results.**

## ***Web interfaces to Linked Data***

*A different balance of trade-offs*

*Real-world federated queries*

**Queryable Linked Data on the Web  
has a **two-sided availability problem.****

**There a few SPARQL endpoints  
because they are expensive to host.**

**Those endpoints that are on the Web  
suffer from frequent downtime.**

The average public SPARQL endpoint  
is down for 1.5 days *each month*.

**Data dumps** allow people to set up their own *private* SPARQL endpoint.

**Users need a technical background and the necessary infrastructure.**

**What about casual usage and mobile devices?**

**We are not really querying the Web...**

**It is not an all-or-nothing world.**  
**There is a spectrum of trade-offs.**

*out-of-date data*  
*high bandwidth*  
*high availability*  
*high client cost*  
*low server cost*

*live data*  
*low bandwidth*  
*low availability*  
*low client cost*  
*high server cost*



**Linked Data Fragments** are  
a uniform view on Linked Data interfaces.

*Every Linked Data interface  
offers **specific fragments**  
of a Linked Data set.*



# Each type of Linked Data Fragment is defined by three characteristics.

**data** *What triples does it contain?*

**metadata** *What do we know about it?*

**controls** *How to access more data?*

# Each type of Linked Data Fragment is defined by three characteristics.

***data dump***

**data**

*all dataset triples*

**metadata**

*number of triples, file size*

**controls**

*(none)*



# Each type of Linked Data Fragment is defined by three characteristics.

***SPARQL query result***

**data** *triples matching the query*

**metadata** *(none)*

**controls** *(none)*

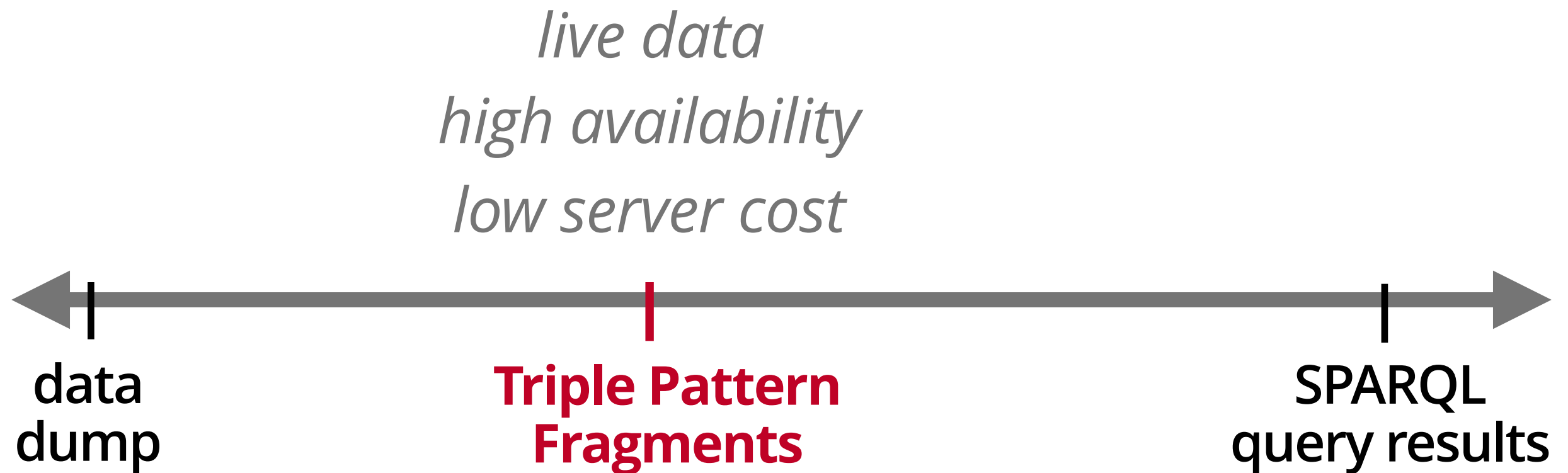
# We designed **a new trade-off mix** with low cost and high availability.

*out-of-date data*  
*high bandwidth*  
*high availability*  
*high client cost*  
*low server cost*

*live data*  
*low bandwidth*  
*low availability*  
*low client cost*  
*high server cost*



**A Triple Pattern Fragments interface**  
**is low-cost and enables clients to query.**



**A Triple Pattern Fragments interface**  
**is low-cost and enables clients to query.**

<b>data</b>	<i>matches of a triple pattern (paged)</i>
<b>metadata</b>	<i>total number of matches</i>
<b>controls</b>	<i>access to all other fragments</i>

# DBpedia – Linked Data Fragments



Query DBpedia 2014 by triple pattern

subject:

predicate: `dbpedia-owl:birthPlace`

object: `dbpedia:Italy`

Find matching triples

**controls** (*other fragments*)

Showing triples 1 to 101 of ±8141

**metadata** (*total count*)

```
%C3%81ivaro_Crespi birthPlace Italy.  
%C3%89douard_Fachleitner birthPlace Italy.  
108_(artist) birthPlace Italy.  
A._F._K._Organski birthPlace Italy.  
Aaron_March birthPlace Italy.  
Abdon_Sgarbi birthPlace Italy.  
Abel_Gigli birthPlace Italy.  
Abelardo_Olivier birthPlace Italy.  
Abele_Blanc birthPlace Italy.  
Achille_Compagnoni birthPlace Italy.
```

**data** (*first 100*)

**Triple patterns are not the final answer.**  
**No interface ever will be.**

**Triple patterns show how far we can get**  
**with simple servers and smart clients.**



**Experience the trade-offs yourself  
on the **official DBpedia interfaces.****

**DBpedia data dump**



**DBpedia Linked Data documents**

**DBpedia SPARQL endpoint**

**DBpedia Triple Pattern Fragments**  
***[fragments.dbpedia.org](http://fragments.dbpedia.org)***

*Web interfaces to Linked Data*

***A different balance of trade-offs***

*Real-world federated queries*



**Triple Pattern Fragments publication  
is very straightforward.**

**Servers only need to implement  
a simple API.**

**A SPARQL endpoint as backend  
is not a necessity.**

**The compressed HDT format  
is very fast for triple patterns.**

# **The LOD Laundromat hosts 650.000 Triple Pattern Fragment APIs.**

**Datasets are crawled from the Web,  
cleaned, and compressed to HDT.**

**This shows the potential  
of a very light-weight interface.**

**Centralization is not a goal though:  
we aim for distributed interfaces.**

# How can intelligent clients solve SPARQL queries over fragments?

Give them a SPARQL query.

Give them a URL of any dataset fragment.

They look inside the fragment  
to see how to access the dataset

and use the metadata  
to decide how to plan the query.

**Suppose a client needs to evaluate this query against a **TPF interface**.**

```
SELECT ?person ?city WHERE {  
    ?person rdf:type dbpedia-owl:Scientist.  
    ?person dbpedia-owl:birthPlace ?city.  
    ?city      foaf:name "Geneva"@en.  
}
```

***Fragment: <http://fragments.dbpedia.org/2014/en>***

# Triple Pattern Fragment servers *enable clients to be intelligent.*

Query DBpedia 2014 by triple pattern

subject:

predicate:

`dbpedia-owl:birthPlace`

object:

`dbpedia:Italy`

Find matching triples

**controls** *The HTML representation explains:  
“you can query by triple pattern”.*

# Triple Pattern Fragment servers

## *enable clients to be intelligent.*

```
<http://fragments.dbpedia.org/2014/en#dataset> hydra:search [
  hydra:template "http://fragments.dbpedia.org/2014/en
    {?subject,predicate,object}";
  hydra:mapping
    [ hydra:variable "subject";   hydra:property rdf:subject ],
    [ hydra:variable "predicate"; hydra:property rdf:predicate ],
    [ hydra:variable "object";    hydra:property rdf:object ]
].
```

**controls** *The RDF representation explains:  
“you can query by triple pattern”.*

# Triple Pattern Fragment servers *enable clients to be intelligent.*

Showing triples 1 to 101 of ±8141

%C3%81lvaro\_Crespi birthPlace Italy.

%C3%89douard\_Fachleitner birthPlace Italy.

108\_(artist) birthPlace Italy.

**metadata** *The HTML representation explains:  
“this is the number of matches”.*

**Triple Pattern Fragment servers**  
*enable clients to be intelligent.*

**<#fragment> void:triples 8141.**

**metadata** *The RDF representation explains:  
“this is the number of matches”.*



**The server has triple-pattern access,  
so **the client splits a query** that way.**

```
SELECT ?person ?city WHERE {  
    ?person rdf:type dbpedia-owl:Scientist.  
    ?person dbpedia-owl:birthPlace ?city.  
    ?city      foaf:name "Geneva"@en.  
}
```

***Fragment: <http://fragments.dbpedia.org/2014/en>***

# The client gets the fragments and inspects their metadata.

?person rdf:type dbpedia-owl:Scientist **18.000**

*first 100 triples*

?person dbpedia-owl:birthPlace ?city. **625.000**

*first 100 triples*

?city foaf:name "Geneva"@en. **12**

*first 100 triples*

# Execution continues recursively using metadata and controls.

?person rdf:type dbpedia-owl:Scientist

?person dbpedia-owl:birthPlace ?city.

**?city foaf:name "Geneva"@en.**

**12**

dbpedia:Geneva foaf:name "Geneva"@en.

dbpedia:Geneva,\_Alabama foaf:name "Geneva"@en.

dbpedia:Geneva,\_Idaho foaf:name "Geneva"@en.

...

**Executing this query with TPFs  
takes 3 seconds—consistently.**

```
SELECT ?person ?city WHERE {  
  ?person rdf:type dbpedia-owl:Scientist.  
  ?person dbpedia-owl:birthPlace ?city.  
  ?city      foaf:name "Geneva"@en.  
}
```

*Results arrive in a streaming way,  
already after 0.5 seconds.*

*Web interfaces to Linked Data*

*A different balance of trade-offs*

***Real-world federated queries***

**For a Triple Pattern Fragments client,  
federated queries are not special.**

**The client's algorithm is based  
on fetching fragments from a server.**

**The client does not care about  
what server it asks fragments from.**

**Federation is *native* to  
Triple Pattern Fragments clients.**

# **We will use three large datasets as a demo of federated querying.**

**DBpedia**

**377.000.000 triples  
7.1GB HDT file**

**VIAF**

**684.000.000 triples  
6.8GB HDT file**

**Harvard Library**

**183.000.000 triples  
2.4GB HDT file**

***Web interfaces to Linked Data***

***A different balance of trade-offs***

***Real-world federated queries***



We will use three large datasets  
as a demo of federated querying.

DBpedia

***This*** 375.000.000 triples

7.1GB HDT file

***could be***

VIAF

***your data!*** 654.000.000 triples

6.8GB HDT file

**Harvard Library**

183.000.000 triples

2.4GB HDT file



We will use three large datasets  
as a demo of federated querying.

DBpedia

***This*** 375.000.000 triples

7.1GB HDT file

***should be***

VIAF

***your data!*** 654.000.000 triples

6.8GB HDT file



**Harvard Library**

183.000.000 triples

2.4GB HDT file

**Triple pattern fragments are easy:**  
**all software is available as open source.**

**Software**

***[github.com/LinkedDataFragments](https://github.com/LinkedDataFragments)***

**Documentation and specification**

***[linkeddatafragments.org](http://linkeddatafragments.org)***

**There are no excuses left  
to *not* publish Linked Data.**

**The Triple Pattern Fragments interface  
is inexpensive to host.**

**It allows people to query your dataset.**

**It allows federated queries with Web data,  
which includes your dataset.**

**Semantic Web:**

**SOLUTION**

**looking for**

**PROBLEM?**

**Linked Data:**

**SOLUTION**

**looking for**

**PROBLEM?**

**Linked Data:**

**SOLUTION**

**ready to tackle**

**data PROBLEMs**

**on the Web.**



# Linked Data Fragments

Ruben Verborgh

