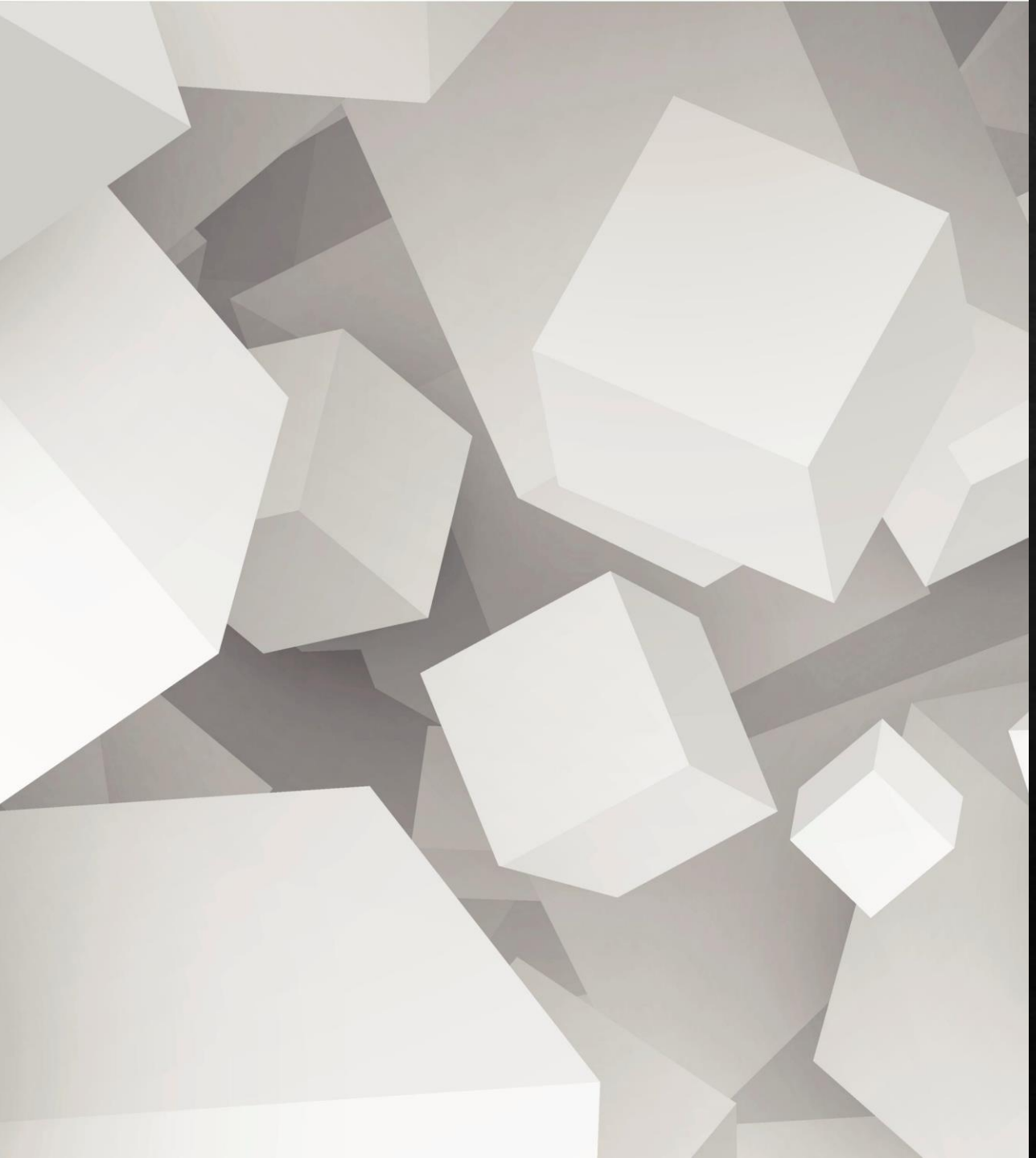
Querying RDF with SPARQL



Title Lorem Ipsum

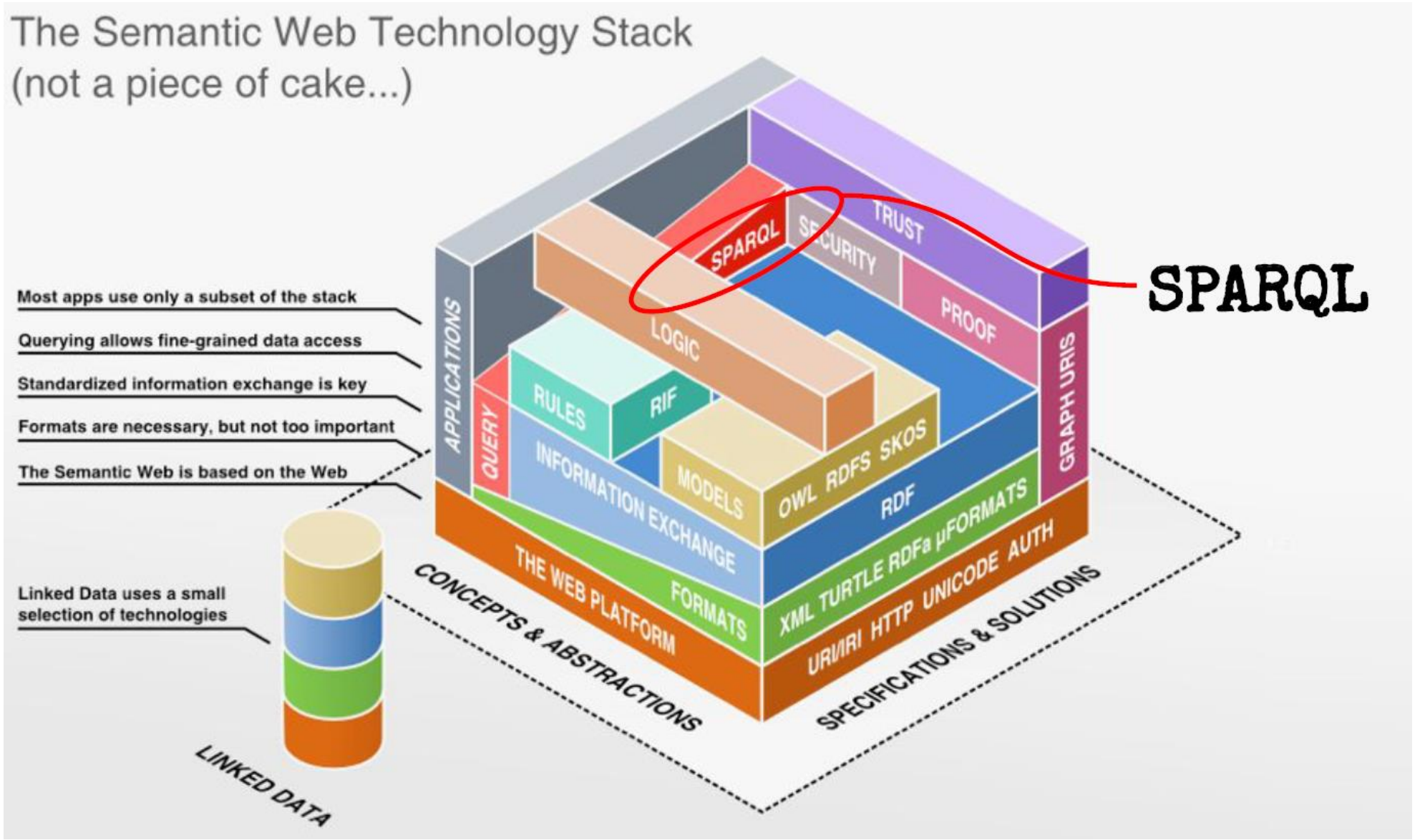
Dolor Sit Amet

Consectetuer Elit

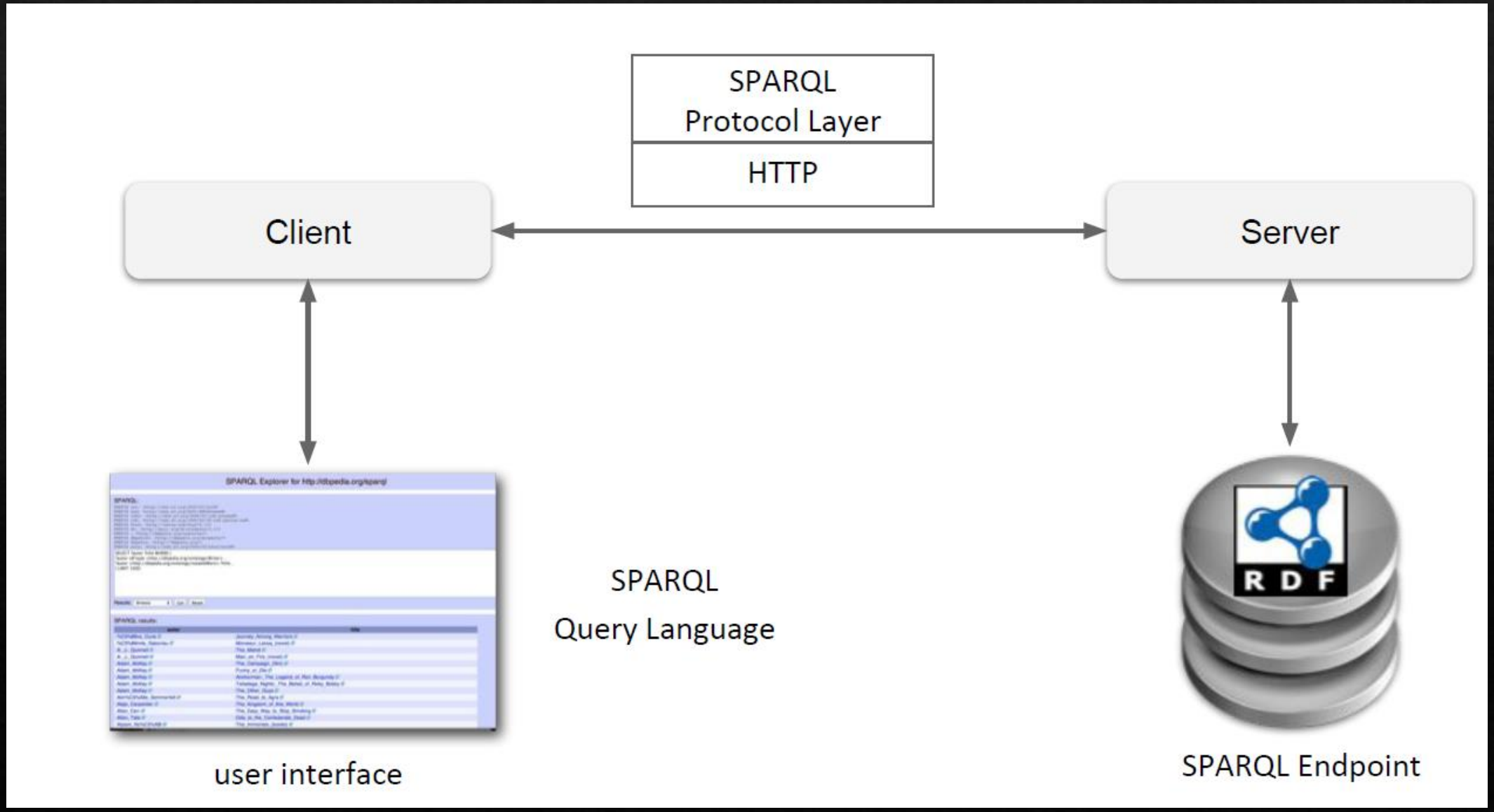
Nunc Viverra

Pellentesque Habitant

Lorem Ipsum


The Semantic Web Technology Stack
(not a piece of cake...)
Most apps use only a subset of the stack
Querying allows fine-grained data access
Standardized information exchange is key
Formats are necessary, but not too important
The Semantic Web is based on the Web
Linked Data uses a small selection of technologies
LINKED DATA
APPLICATIONS
QUERY
SPARQL
TRUST
SECURITY
PROOF
LOGIC
RULES
RIF
INFORMATION EXCHANGE
MODELS
OWL RDFS SKOS
RDF
GRAPH URIS
THE WEB PLATFORM
FORMATS
XML TURTLE RDFa μFORMATS
URI/IRI HTTP UNICODE AUTH
CONCEPTS & ABSTRACTIONS
SPECIFICATIONS & SOLUTIONS
SPARQL

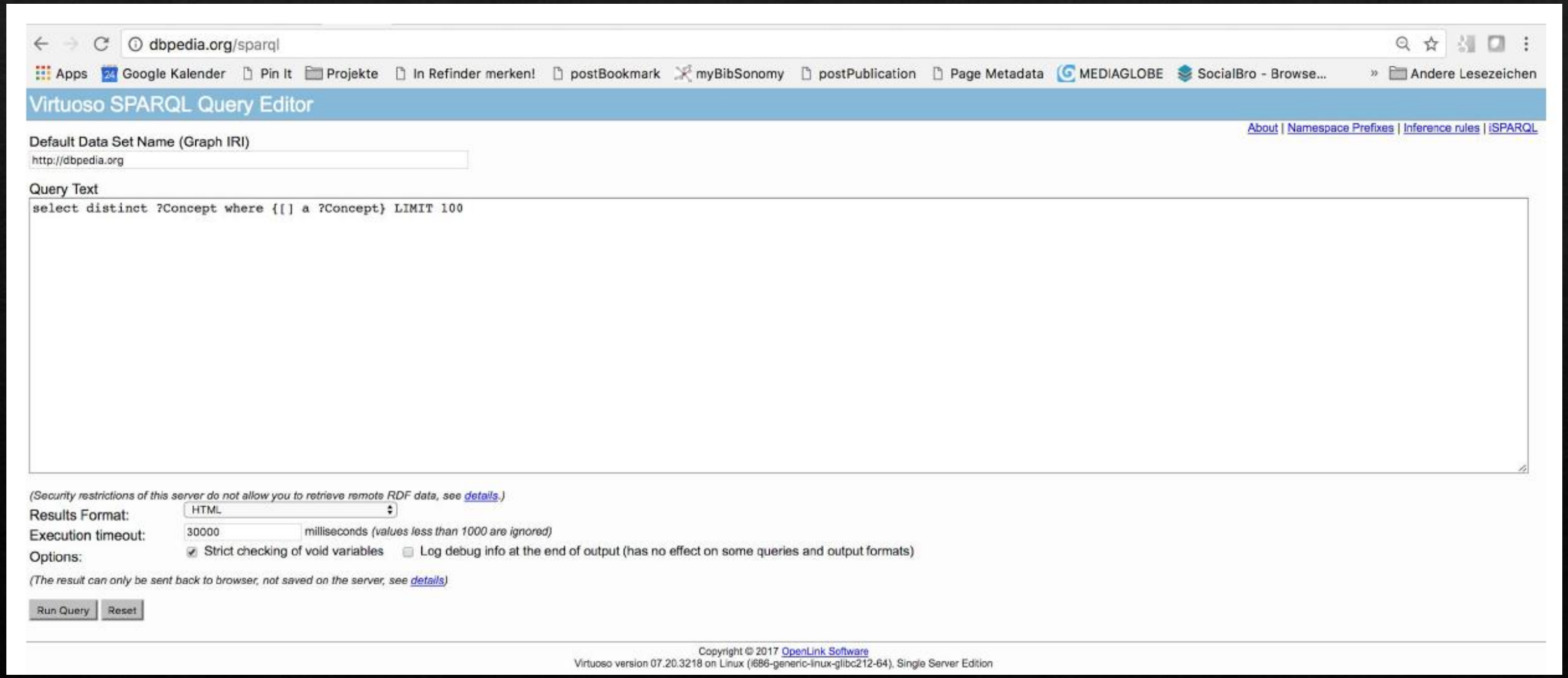
SPARQL – Querying Language



SPARQL – Querying Language

- **SPARQL P**rotocol **a**nd **R**DF **Q**uery **L**anguage is
  - a **Query Language** for RDF graph traversal (*SPARQL Query Language Specification*)
  - a **Protocol Layer**, to use SPARQL via http (*SPARQL Protocol for RDF Specification*)
  - an **XML Output Format Specification** for SPARQL queries (*SPARQL Query XML Results Format*)
  - W3C Standard (SPARQL 1.1, Mar 2013)
  - inspired by SQL

SPARQL – ENDPOINT



How to Query RDFs?

- SPARQL **variables** are bound to RDF terms
  - e.g. **?title, ?author, ?date**
- In the same way as in SQL,

  a **Query for variables** is performed via **SELECT statement**
  - e.g. **SELECT ?title ?author ?date**

SPARQL Query

- A SELECT statement returns query results as a **table**

?title	?author	?date
Nineteen Eighty-Four	George Orwell	1948
An Inconvenient Truth	Al Gore	2006
Make Room! Make Room!	Harry Harrison	1966

SPARQL Result

SPARQL Graph Pattern Matching

- SPARQL is based on
  (1) **RDF Turtle serialization** and (2) **basic graph pattern matching**.
- A **Graph Pattern** (**Triple Pattern**) is a RDF Triple that contains variables at any arbitrary place (Subject, Property, Object).

  Graph Pattern (Triple Pattern) = Turtle + Variables
- Example:
  Look for books and their authors (via property dbo:author):

```
?book dbo:author ?author .
```

variables

SPARQL Graph Pattern Matching

- SPARQL Graph Pattern can be combined to form **complex (conjunctive) queries** for RDF graph traversal.

- *Find books, their authors, and their literary genres:*

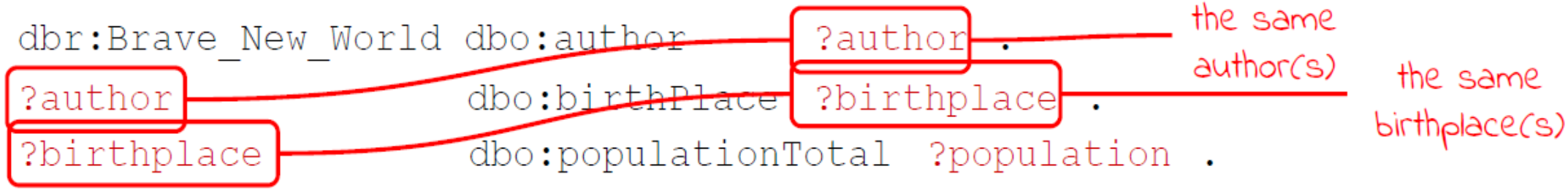
the same book(s)

```
?book dbo:author ?author .
?book dbo:literaryGenre ?genre .
```

SPARQL Graph Pattern Matching

- SPARQL Graph Pattern can be combined to form **complex (conjunctive) queries** for RDF graph traversal.

- *Given a book URI, find its author(s), the birthplace(s) of its author(s), including the number of population of the birthplace(s):*



SPARQL Complex Pattern Matching

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
    ?author rdf:type dbo:Writer .
    ?author rdfs:label ?author_name .
    ?author dbo:notableWork ?work .
    ?work rdfs:label ?title .
}
```

specifies namespaces

specifies output variables

specifies graph to be queried

specifies graph pattern to be matched

- Example: search for all **authors** and the **titles** of their **notable works**

query SPARQL endpoint

SPARQL Complex Pattern Matching

```
PREFIX :     <http://dbpedia.org/resource/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
      ?author rdf:type dbo:Writer .
      ?author rdfs:label ?author_name .
      ?author dbo:notableWork ?work .
      ?work rdfs:label ?title .
}
ORDER BY ASC (?author_name)
LIMIT 100
OFFSET 10
```

solution sequence modifiers

- Example:
  search for all **authors** and the **titles** of their **notable works**: **ordered by authors** in **ascending order** and **limit** the results to **the first 100 results** starting the list at **offset 10** position:

query SPARQL endpoint

SPARQL Filter Constraints


```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer .
        ?author rdfs:label ?author_name .
        ?author dbo:notableWork ?work .
        ?work dbo:numberOfPages ?pages
        FILTER (?pages > 500) .
        ?work rdfs:label ?title .
} LIMIT 100
```

specifies constraints for the result

- Example: search for all **authors** and the **titles** of their **notable works** that have **more than 500 pages** and **limit** the results to **the first 100**

- `FILTER` expressions contain operators and functions

query SPARQL endpoint

SPARQL Filter Constraints

Operator	Type(A)	Result Type
`!A`	xsd:boolean	xsd:boolean
`+A`	numeric	numeric
`-A`	numeric	numeric
`BOUND(A)`	variable	xsd:boolean
`isURI(A)`	RDF term	xsd:boolean
`isBLANK(A)`	RDF term	xsd:boolean
`isLITERAL(A)`	RDF Term	xsd:boolean
`STR(A)`	literal/URL	simple literal
`LANG(A)`	literal	simple literal
`DATATYPE(A)`	literal	URI

SPARQL Filter Constraints

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dct:  <http://purl.org/dc/terms/>
PREFIX dbc:  <http://dbpedia.org/resource/Category:>

SELECT ?author_name ?title
FROM <http://dbpedia.org/>
WHERE {
        ?author rdf:type dbo:Writer .
        ?author rdfs:label ?author_name
        FILTER (LANG(?author_name)="en").
        ?work dbo:author ?author .
                ?work rdfs:label ?title .
        FILTER (LANG(?title)="en")
        ?work dct:subject dbc:Environmental_fiction_books .
} LIMIT 100
```

- Example:
  Search for **authors** and their **books**, filter results for **English labels** and **Environmental fiction books** and **limit** the results to **the first 100**

query SPARQL endpoint

COMPLEX SPARQL QUERIES

WIKIDATA Label Language Filtering

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?authorLabel ?bookLabel ?date
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      ?book wdt:P921 wd:Q7942 .
      ?book wdt:P577 ?date .
      SERVICE wikibase:label
      { bd:serviceParam wikibase:language "en" }
}
```

Instance of written work

author

main subject global warming

publication date

- Example:
  Search for **authors and their books**, filter results for **English labels** and **Books on Global warming**

query SPARQL endpoint

WIKIDATA Label Language Filtering

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?authorLabel ?bookLabel ?date
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      ?book wdt:P921 wd:Q7942 .
      ?book wdt:P577 ?date .
      SERVICE wikibase:label
      { bd:serviceParam wikibase:language "en" }
}
```

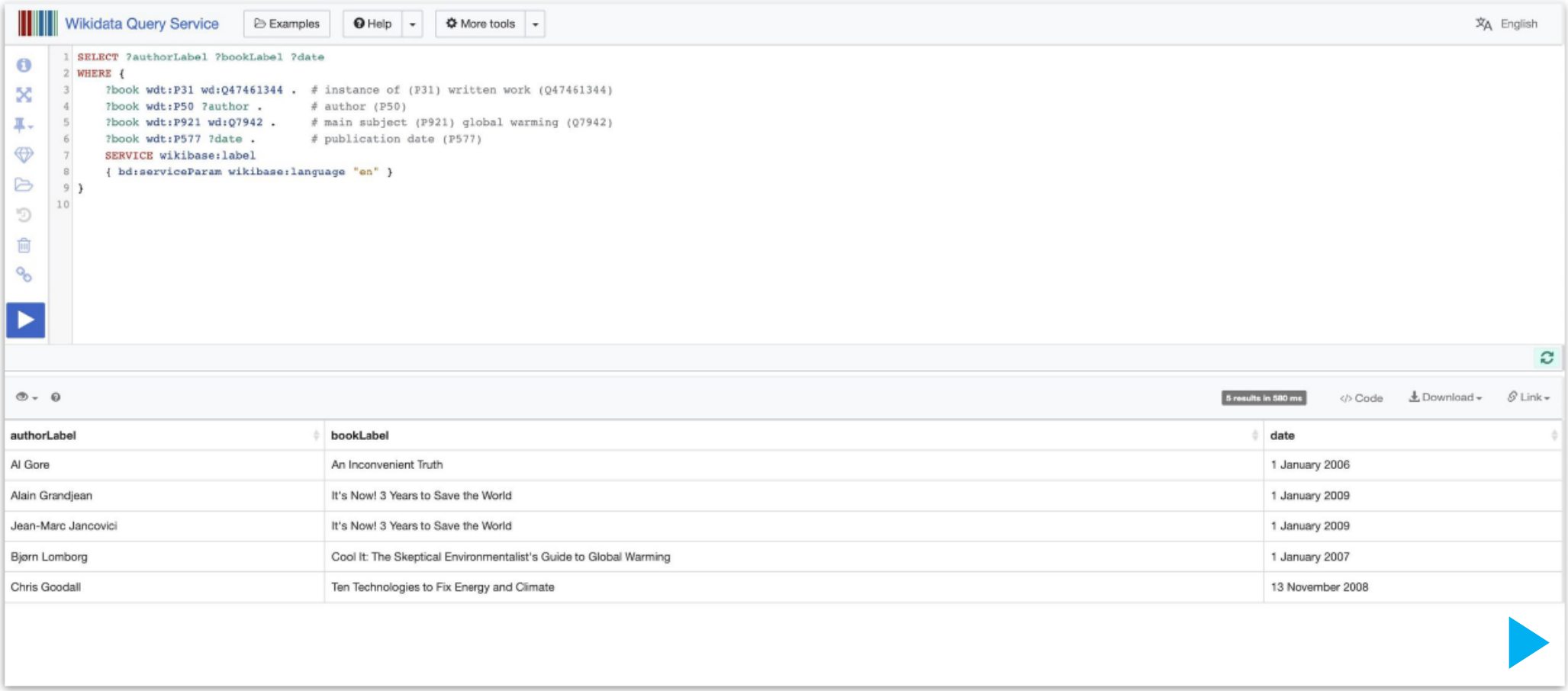
wikidata specific label service

- Example:
  Search for **authors and their books**, filter results for **English labels** and **Books on Global warming**

query SPARQL endpoint

WIKIDATA Label Language Filtering

Search for **authors and their books**, filter results for **English labels** and **Books on Global Warming**



authorLabel	bookLabel	date
Al Gore	An Inconvenient Truth	1 January 2006
Alain Grandjean	It's Now! 3 Years to Save the World	1 January 2009
Jean-Marc Jancovici	It's Now! 3 Years to Save the World	1 January 2009
Bjørn Lomborg	Cool It: The Skeptical Environmentalist's Guide to Global Warming	1 January 2007
Chris Goodall	Ten Technologies to Fix Energy and Climate	13 November 2008

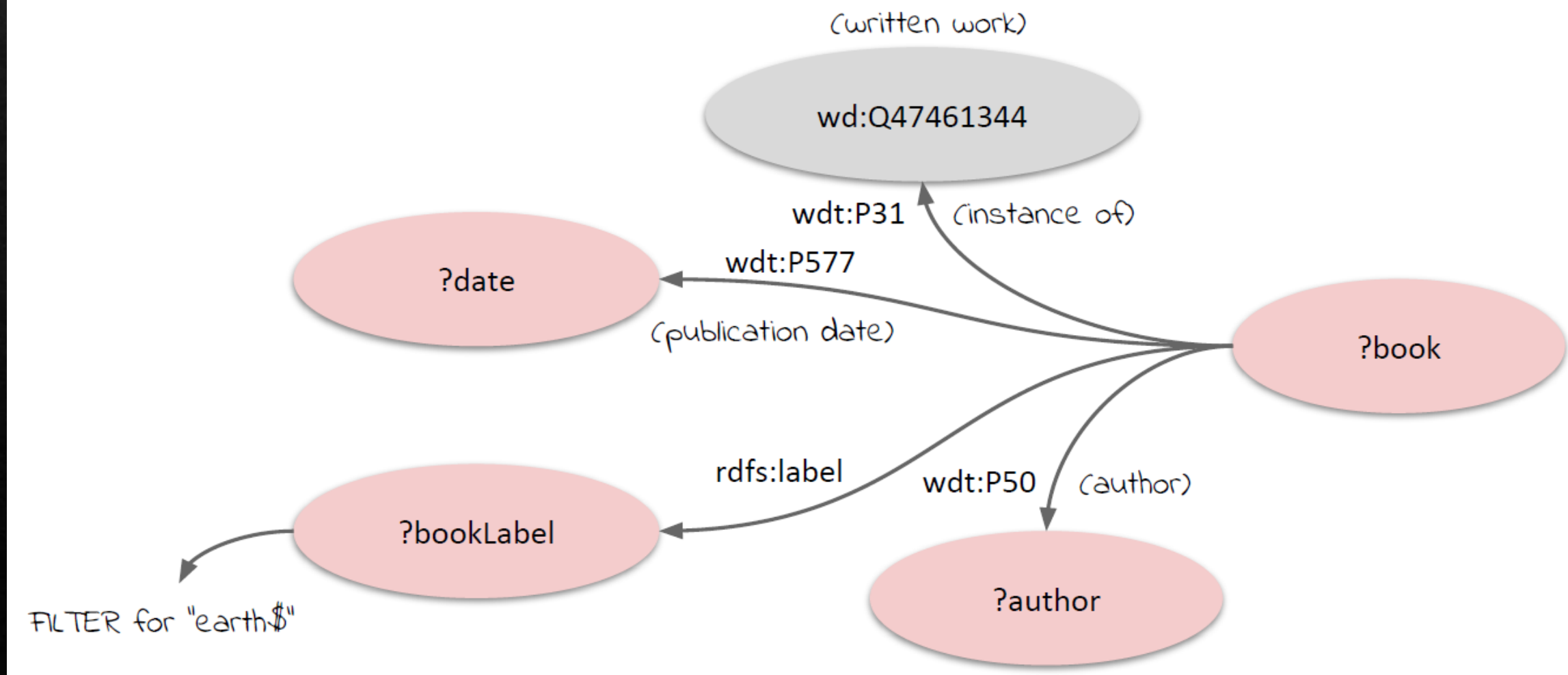
More SPARQL Operators

- Logical connectives `&&` (**AND**) and `||` (**OR**) for xsd:boolean
- Comparison operators `=`, `!=`, `<`, `>`, `<=`, and `>=` for numeric datatypes, xsd:dateTime, xsd:string, and xsd:boolean
- Comparison operators `=` and `!=` for other datatypes
- Arithmetic operators `+`, `-`, `*`, and `/` for numeric datatypes

- and in addition:
  - `REGEX(String,Pattern)` or `REGEX(String,Pattern,Flags)`
  - `sameTERM(A,B)`
  - `langMATCHES(A,B)`

SPARQL Filter Constraints

- what book titles end with the word "**earth**" sorted by publication date?



SPARQL Filter Constraints

- what book titles end with the word "**earth**" sorted by publication date?


```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?authorLabel ?bookLabel ?date
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      ?book wdt:P577 ?date .
      ?book rdfs:label ?bookLabel
      FILTER (LANG(?bookLabel)="en")
      FILTER REGEX (?bookLabel,"earth$","i") .
      ?author rdfs:label ?authorLabel
      FILTER (LANG(?authorLabel)="en") .
} ORDER BY ?date
```

string

regular expression

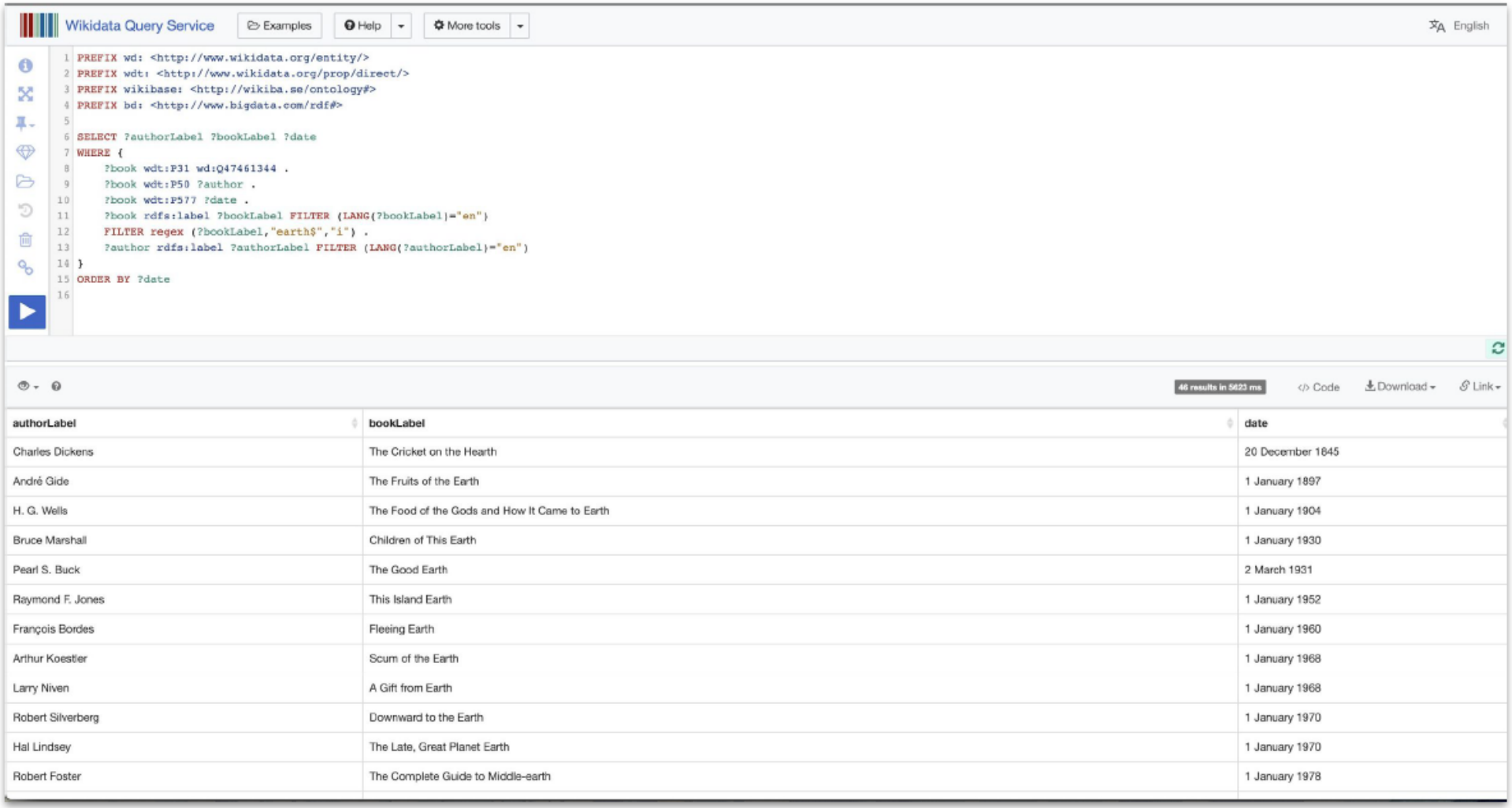
flags

- With **FILTER REGEX**, regular expressions can be filtered

query SPARQL endpoint

SPARQL Filter Constraints

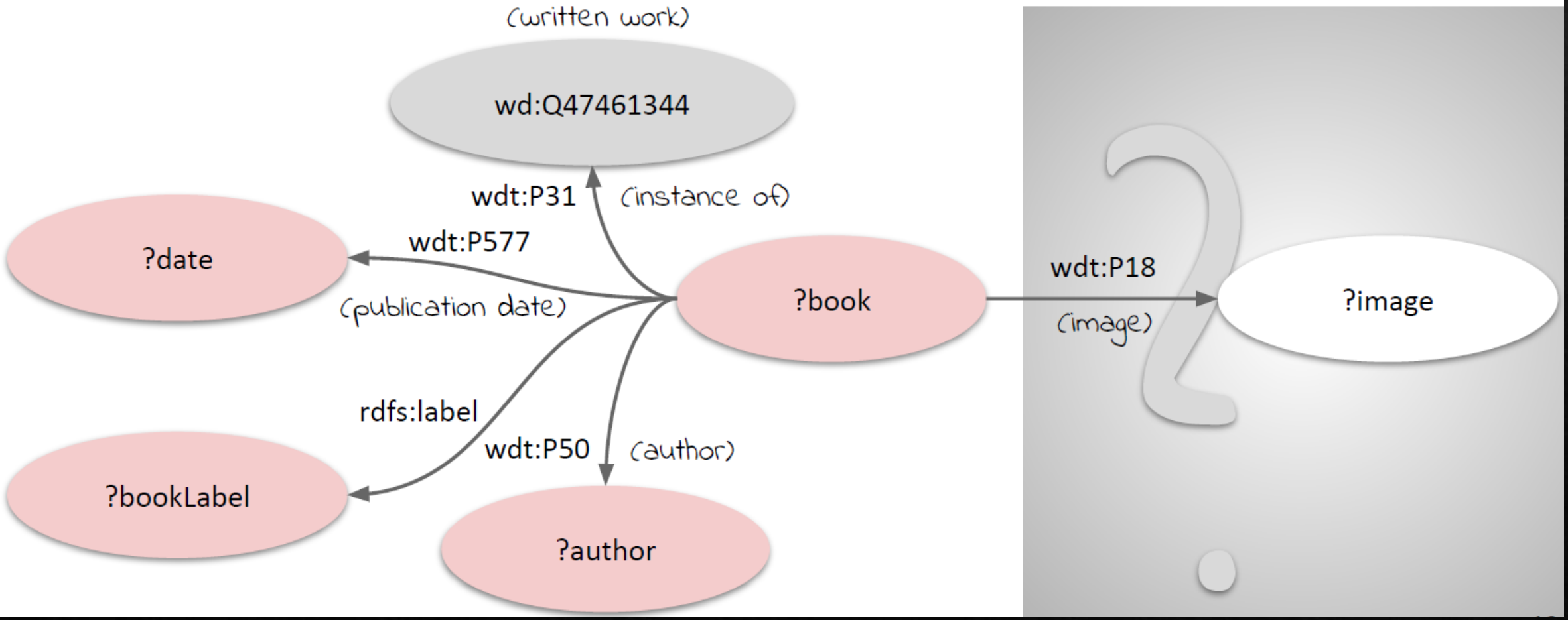
- what book titles end with the word "earth" sorted by publication date?



authorLabel	bookLabel	date
Charles Dickens	The Cricket on the Hearth	20 December 1845
André Gide	The Fruits of the Earth	1 January 1897
H. G. Wells	The Food of the Gods and How It Came to Earth	1 January 1904
Bruce Marshall	Children of This Earth	1 January 1930
Pearl S. Buck	The Good Earth	2 March 1931
Raymond F. Jones	This Island Earth	1 January 1952
François Bordes	Fleeing Earth	1 January 1960
Arthur Koestler	Scum of the Earth	1 January 1968
Larry Niven	A Gift from Earth	1 January 1968
Robert Silverberg	Downward to the Earth	1 January 1970
Hal Lindsey	The Late, Great Planet Earth	1 January 1970
Robert Foster	The Complete Guide to Middle-earth	1 January 1978

SPARQL Filter Constraints

- which book titles **end with the word "earth"**, **and, if available, do also have an image?**



SPARQL Filter Constraints

- which book titles **end with the word "earth"**, **and, if available, do also have an image?**


```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?authorLabel ?bookLabel ?date ?image
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      ?book wdt:P577 ?date .
      ?book rdfs:label ?bookLabel
      FILTER (LANG(?bookLabel)="en")
      FILTER regex (?bookLabel,"earth$","i") .
      ?author rdfs:label ?authorLabel
      FILTER (LANG(?authorLabel)="en") .
      OPTIONAL {?book wdt:P18 ?image .}
} ORDER BY ?date
```

optional constraint

- Optional selection of graph pattern via **OPTIONAL**

query SPARQL endpoint

SPARQL Filter Constraints

- which book titles **end with the word "earth"**, and, if available, do also have an image?

Wikidata Query Service | Examples | Help | More tools | English

```
#defaultView:Timeline
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?authorLabel ?bookLabel ?date ?image
WHERE {
    ?book wdt:P31 wd:Q47461344 . # instance of (P31) written work (Q47461344)
    ?book wdt:P50 ?author .      # author (P50)
    ?book wdt:P577 ?date .       # publication date (P577)
    ?book rdfs:label ?bookLabel FILTER (LANG(?bookLabel)="en")
    FILTER regex (?bookLabel,"earth$","i") .
    ?author rdfs:label ?authorLabel FILTER (LANG(?authorLabel)="en")
    OPTIONAL {?book wdt:P18 ?image}
} ORDER BY ?date
```

Timeline | 46 results in 4086 ms | Code | Download | Lin

1840 1850 1860 1870 1880 1890 1900 1910 1920 1930 1940 1950 1960 1970 1980 1990 2000 2010

Charles Dickens
The Cricket on the Hearth
20 December 1845

H. G. Wells
The Food of the Gods and How It Came to Earth
1 January 1904

Pearl S. Buck
The Good Earth
2 March 1931

Bruce Marshall
Children of This Earth
1 January 1930

André Gide
The Fruits of the Earth
1 January 1897

Robert Silverberg
Downward to the Earth
1 January 1970

Larry Niven
A Gift from Earth
1 January 1968

Hal Lindsey
The Late, Great Planet Earth
1 January 1970

Arthur Koestler
Scum of the Earth

Jack Vance
Ecce and Old Earth
1 January 1991

Arthur C. Clarke
Tales from Planet Earth
1 January 1990

David Brin
Earth
1 January 1990

Ken Follett
The Pillars of the Earth

Tim Winton
Down to Earth
1 January 2008

Peter Kennard
@earth
1 January 2011

Eckhart Tolle
A New Earth
1 January 2005

Richard Woldendorp
Down to Earth

SPARQL – UNION

- Example: which books mention "London" in their title or have London as their narrative location

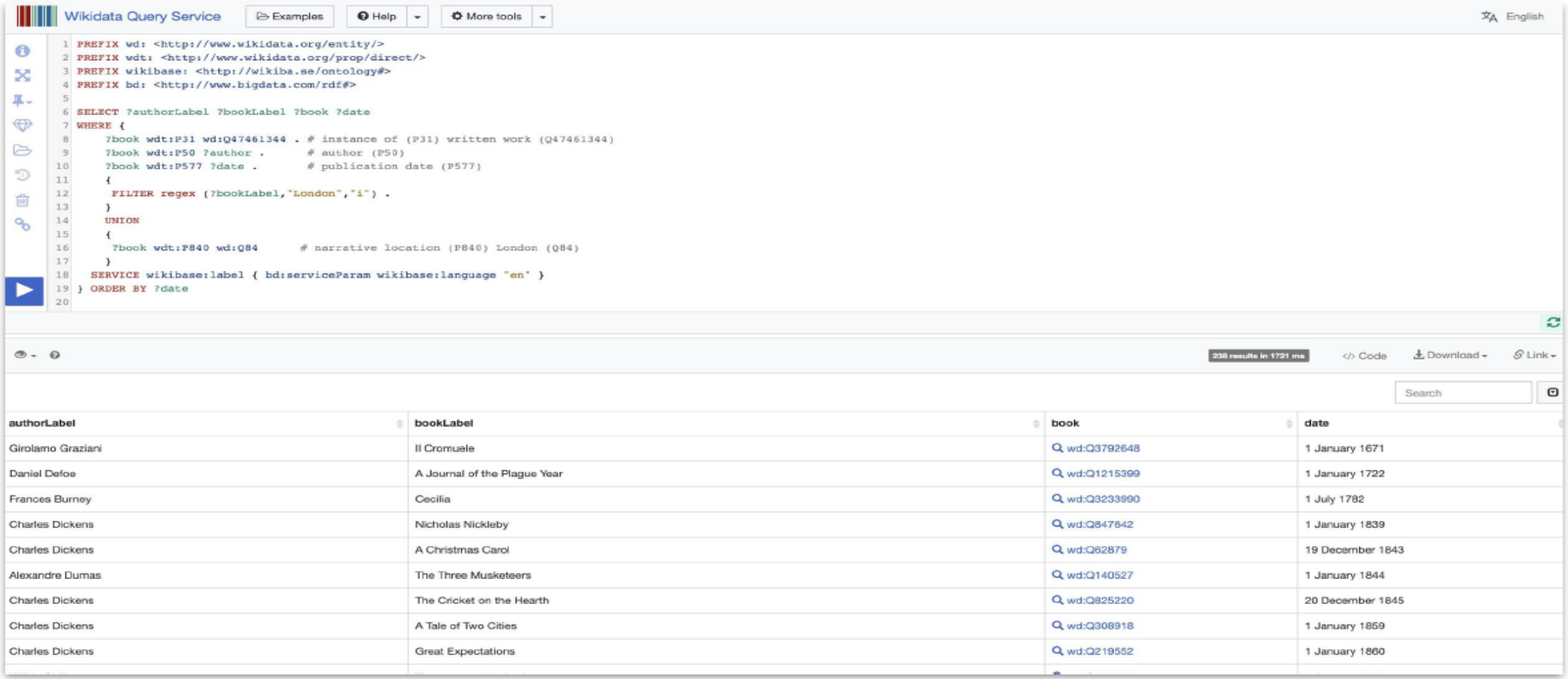

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?authorLabel ?bookLabel ?book ?date ?image
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      ?book wdt:P577 ?date .
      { FILTER regex (?bookLabel,"London","i") . }
     UNION
      { ?book wdt:P840 wd:Q84 .}
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} ORDER BY ?date
```

logical disjunction

- The keyword **UNION** allows for alternatives (logical disjunction)

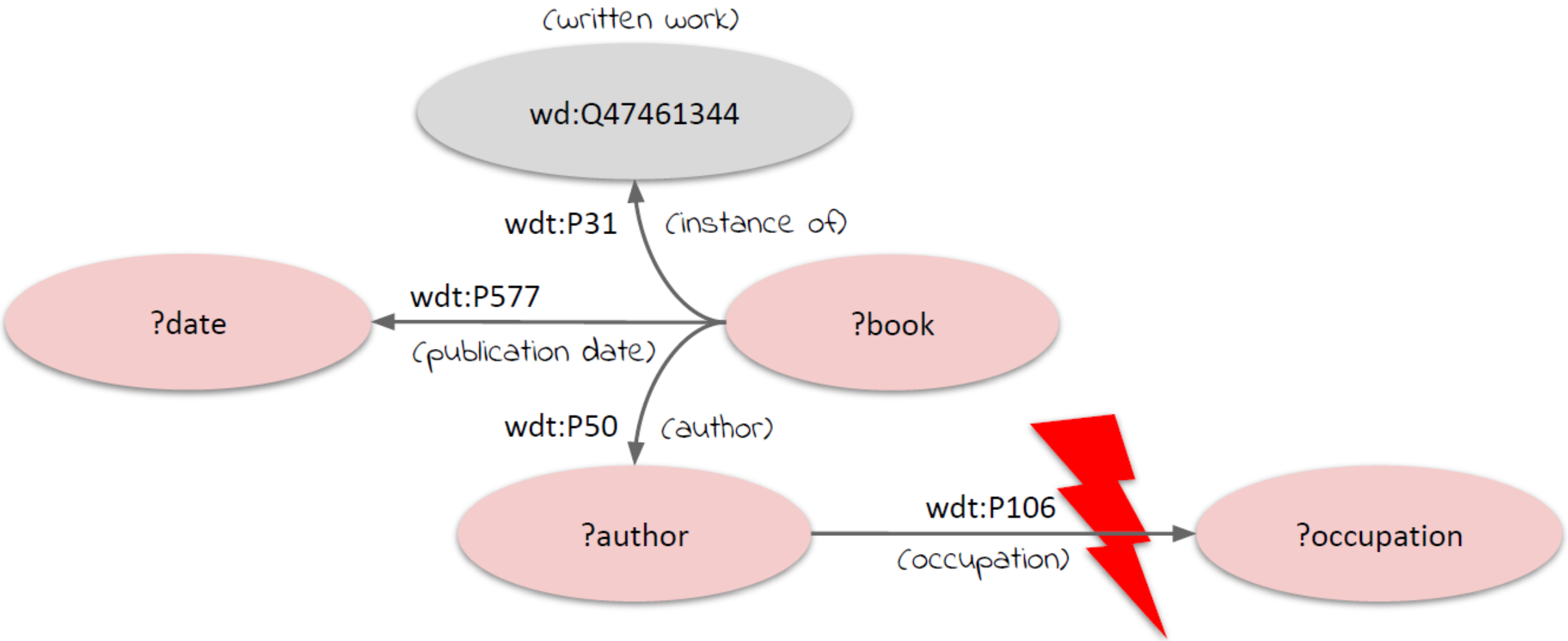
SPARQL – UNION

- Example: which books mention "London" in their title or have London as their narrative location



SPARQL – NEGATION

- Example: which books are written by authors who **don't have an occupation** ?



SPARQL – NEGATION

- Example: which books are written by authors who **don't have an occupation** ?

```
SELECT ?authorLabel ?bookLabel ?date
WHERE {
      ?book wdt:P31 wd:Q47461344 .
      ?book wdt:P50 ?author .
      FILTER NOT EXISTS {?author wdt:P106 ?occupation }
?book wdt:P577 ?date .
      SERVICE wikibase:label
      { bd:serviceParam wikibase:language "en, de, es, it" }
}
```

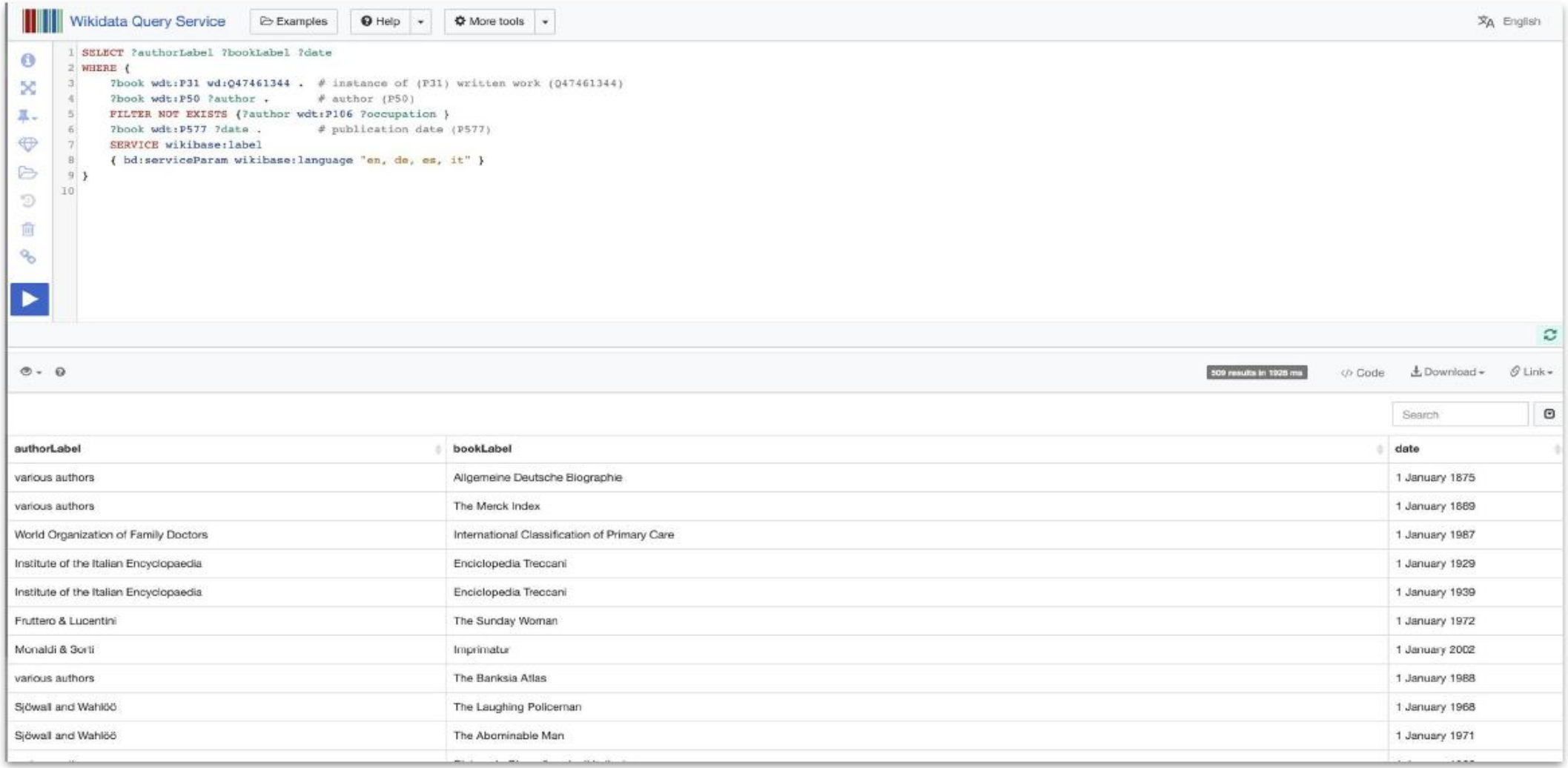
filter query result for existence

WIKIDATA

- SPARQL 1.1 offers several variants for negation:
  - **FILTER NOT EXISTS**
  - **MINUS**
  - **!BOUND()**

SPARQL – NEGATION

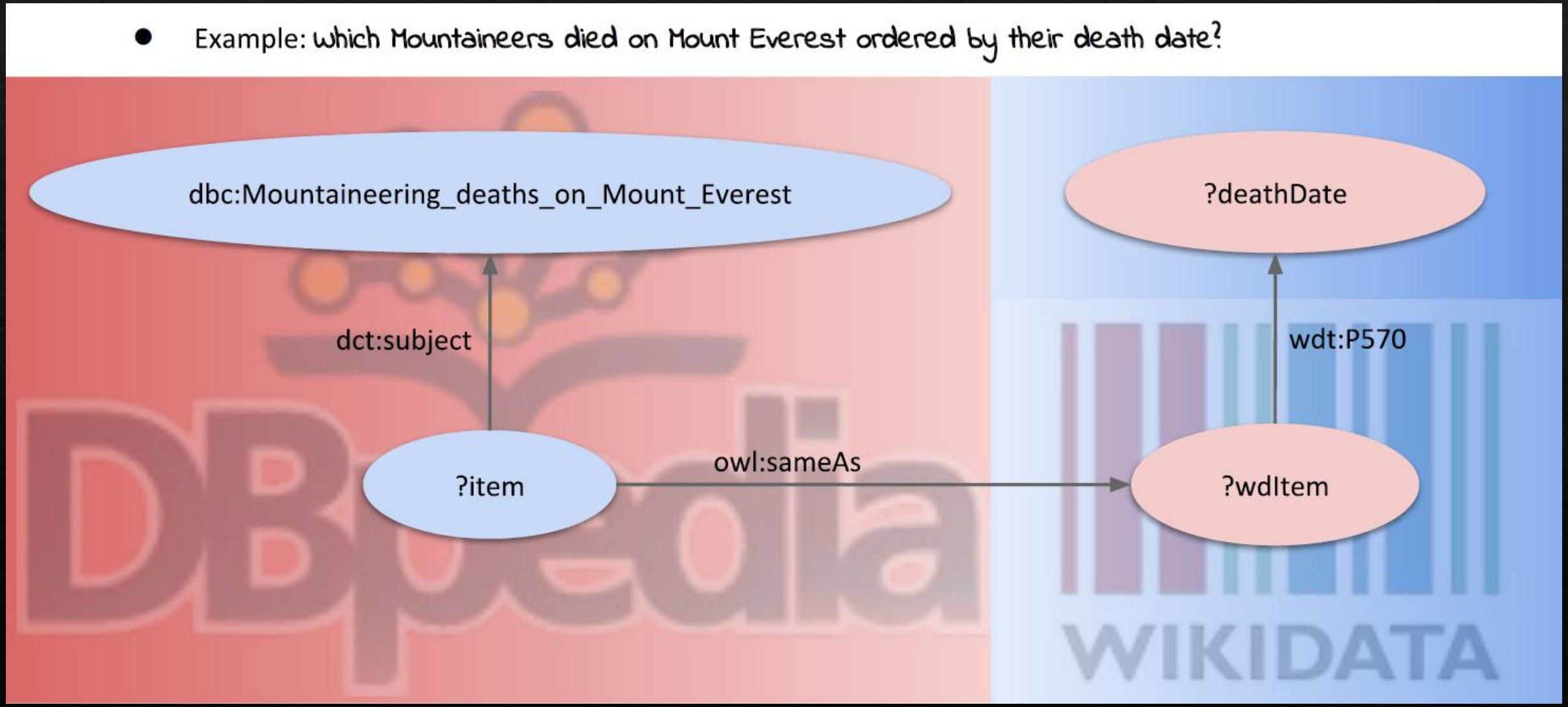
- Example: which books are written by authors who **don't have an occupation** ?



authorLabel	bookLabel	date
various authors	Allgemeine Deutsche Biographie	1 January 1875
various authors	The Merck Index	1 January 1889
World Organization of Family Doctors	International Classification of Primary Care	1 January 1987
Institute of the Italian Encyclopaedia	Enciclopedia Treccani	1 January 1929
Institute of the Italian Encyclopaedia	Enciclopedia Treccani	1 January 1939
Fruttero & Lucentini	The Sunday Woman	1 January 1972
Monaldi & Sorti	Imprimatur	1 January 2002
various authors	The Banksia Atlas	1 January 1988
Sjöwall and Wahlöö	The Laughing Policeman	1 January 1968
Sjöwall and Wahlöö	The Abominable Man	1 January 1971

SPARQL Federated Queries

- Example: which Mountaineers died on Mount Everest ordered by their death date?



SPARQL Federated Queries

- SPARQL enables federated queries over several RDF datasets or SPARQL endpoints via the **SERVICE** objective


```
PREFIX dct:  <http://purl.org/dc/terms/>
PREFIX dbc:  <http://dbpedia.org/resource/Category:>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

 SELECT DISTINCT  ?wditemLabel ?date WHERE {
    SERVICE <http://dbpedia.org/sparql> {
       ?item dct:subject dbc:Mountaineering_deaths_on_Mount_Everest ;
             owl:sameAs ?wditemFILTER regex (?wditem, "wikidata.org") .
    }
    SERVICE <https://query.wikidata.org/sparql>{
        ?wditem wdt:P570 ?date .
        OPTIONAL {?wditem wdt:P18 ?image . }
        ?wditem rdfs:label ?wditemLabelFILTER(LANG(?wditemLabel)="en").
    }
}
ORDER BY ?date
```

- Example: connect the DBpedia with Wikidata "which Mountaineers died on Mount Everest ordered by their death date?"
- only possible, if SPARQL endpoints permit federation

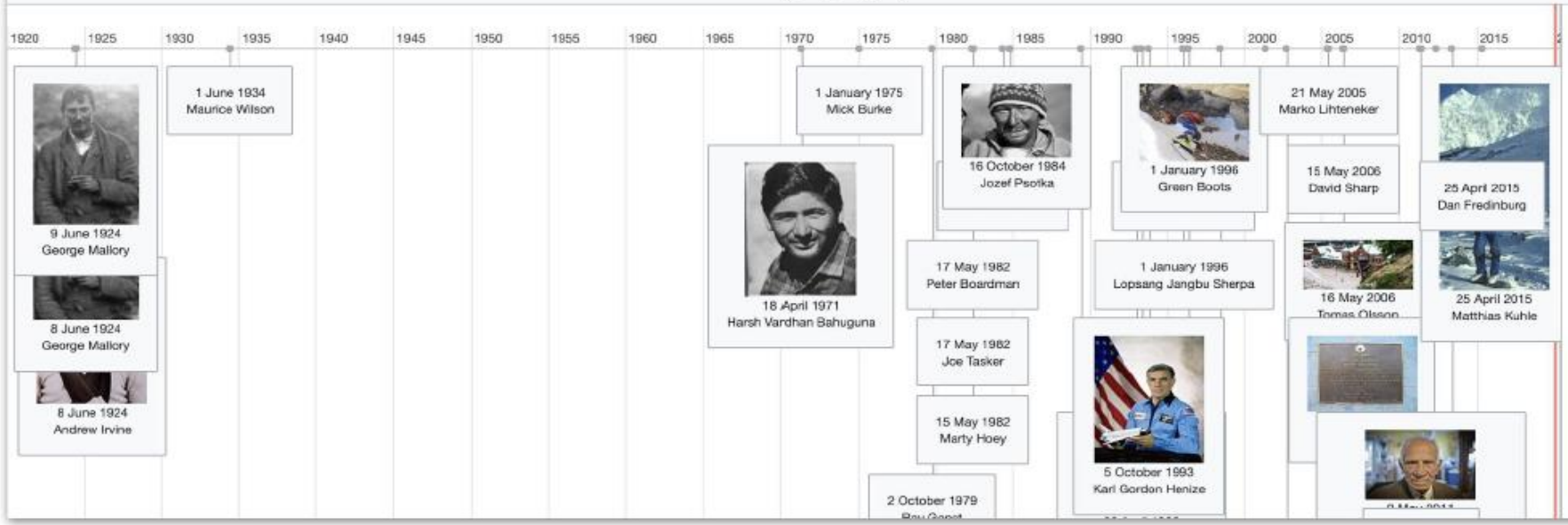
SPARQL Federated Queries

- which Mountaineers died on Mount Everest ordered by their death date?

Wikidata Query Service | Examples | Help | More tools

```
#defaultView:Timeline
PREFIX dct:  <http://purl.org/dc/terms/>
PREFIX dbc:  <http://dbpedia.org/resource/Category:>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

 SELECT DISTINCT  ?wditemLabel ?date ?image WHERE {
    SERVICE <http://dbpedia.org/sparql> {
       ?item dct:subject dbc:Mountaineering_deaths_on_Mount_Everest ;
             owl:sameAs ?wditem FILTER regex (?wditem, "wikidata.org") .
    }
    SERVICE <https://query.wikidata.org/sparql> {
      ?wditem wdt:P570 ?date .
      OPTIONAL {?wditem wdt:P18 ?image . }
        ?wditem rdfs:label ?wditemLabel FILTER(LANG(?wditemLabel)="en").
    }
}
ORDER BY ?date
```

Timeline | 42 results in 821 ms | Code | Download



SPARQL Variable Assignment

- Example: Select authors with their notable works and **date of publication ordered by year**

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?authorLabel ?bookLabel ?book ?author ?year
WHERE {
     ?author wdt:P106 wd:Q36180 ;
             wdt:P800 ?book .
     ?book   wdt:P577 ?date .
  BIND (YEAR(?date) AS ?year) FILTER (BOUND(?year))
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en"
}
} ORDER BY ?year
```

Binding a new variable

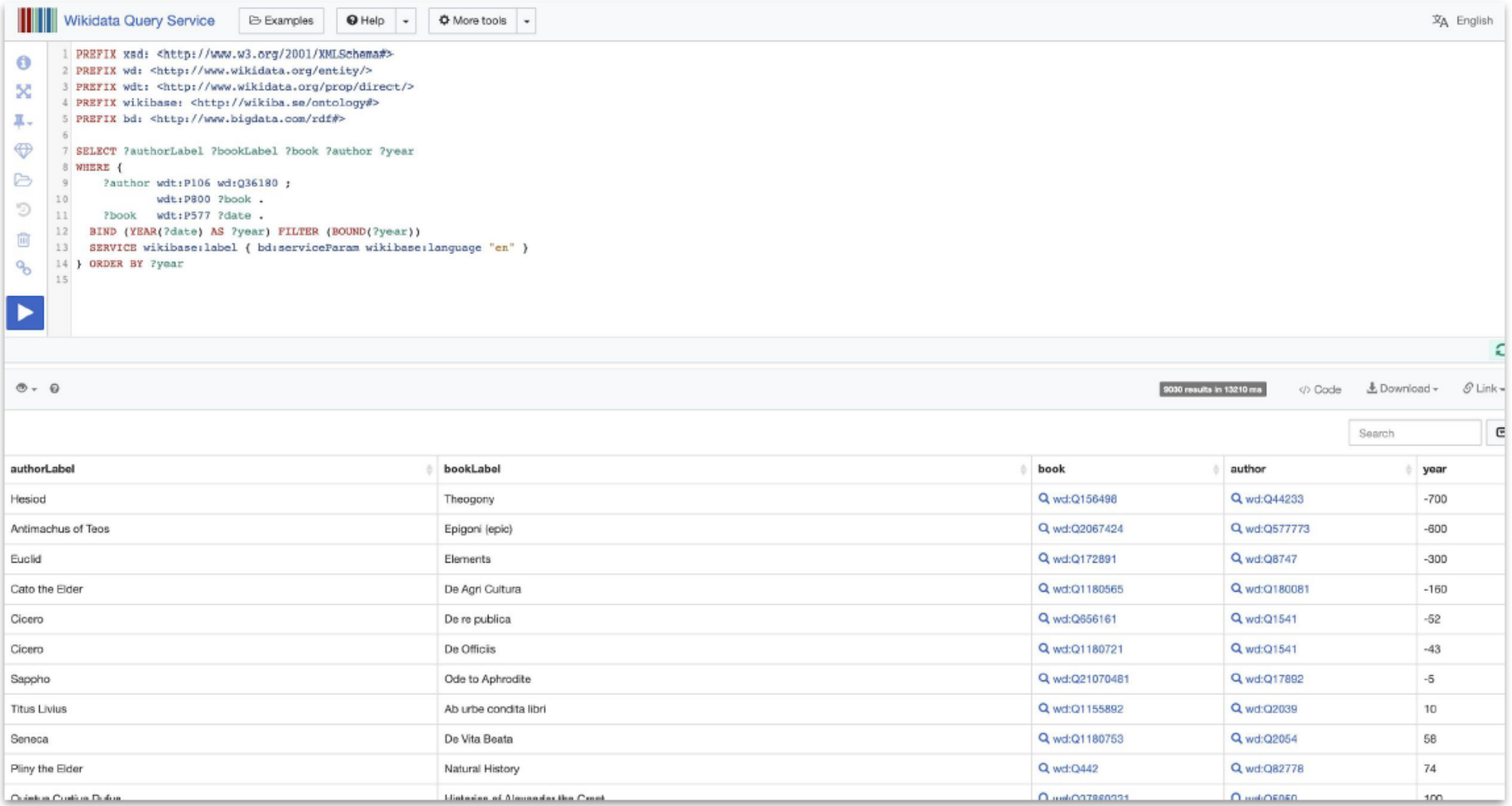
- The **BIND** form allows a value to be assigned to a variable

WIKIDATA

query SPARQL endpoint

SPARQL Variable Assignment

- Example: Select authors with their notable works and **date of publication ordered by year**



SPARQL Aggregate Function

- Example: How many authors are there and how many notable works?


```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT (COUNT(?book) AS ?bookcount)
       (COUNT(DISTINCT(?author)) AS ?authorcount)
WHERE {
     ?author wdt:P106 wd:Q36180 ;
             wdt:P800 ?book .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
}
```

aggregate functions

- **COUNT** is a SPARQL **aggregate function** which counts the number of times a given expression has a bound
- More aggregate functions:
  - **SUM**
  - **AVG**
  - **MIN / MAX**
  - **SAMPLE**

SPARQL Aggregate Function

- SPARQL 1.1 provides more aggregate functions
  - `SUM`
  - `AVG`
  - `MIN`
  - `MAX`
  - `SAMPLE` – „pick“ one non-deterministically
  - `GROUP_CONCAT` – concatenate values with a designated string separator

SPARQL Subqueries and Property Paths

SPARQL Subqueries

- Example: what books were written by the 30 most influential authors?

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?influencerLabel ?bookLabel ?authorCount
WHERE {
  { SELECT ?influencer (COUNT(?author) AS ?authorCount)
    WHERE {
      ?author wdt:P106 wd:Q36180 ;
              wdt:P737 ?influencer .
      ?influencer wdt:P106 wd:Q36180 .
    } GROUP BY ?influencer ORDER BY DESC(?authorCount)
      LIMIT 30
  }
  ?influencer wdt:P800 ?book .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} ORDER BY DESC(?authorCount) ?influencerLabel
```

inner Subquery

- Subqueries are a way to embed SPARQL queries within other queries
- Result is achieved by first evaluating the inner query

WIKIDATA

query SPARQL endpoint

SPARQL Subqueries

- Example: what books were written by the 30 most influential authors?

Wikidata Query Service | Examples | Help | More tools | English

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT  ?influencerLabel  ?bookLabel ?authorCount
WHERE {
  {
    SELECT ?influencer (COUNT(?author) as ?authorCount)
    WHERE {
      ?author wdt:P106 wd:Q36180 ;
              wdt:P737 ?influencer .
      ?influencer wdt:P106 wd:Q36180 ;
    }  GROUP BY ?influencer
       ORDER BY DESC(?authorCount)
       LIMIT 30
  }
  ?influencer wdt:P800 ?book .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} ORDER BY DESC(?authorCount) ?influencerLabel
```

185 results in 1944 ms | Code | Download | Link

influencerLabel	bookLabel	authorCount
William Faulkner	The Sound and the Fury	42
William Faulkner	As I Lay Dying	42
William Faulkner	Light in August	42
William Faulkner	Absalom, Absalom!	42
William Faulkner	A Rose for Emily	42
Vladimir Nabokov	Lolita	35
Vladimir Nabokov	The Defense	35
Vladimir Nabokov	Pale Fire	35
Vladimir Nabokov	Speak, Memory	35
Vladimir Nabokov	The Real Life of Sebastian Knight	35

SPARQL Subqueries

- Example: what books were written by the 30 most influential authors?

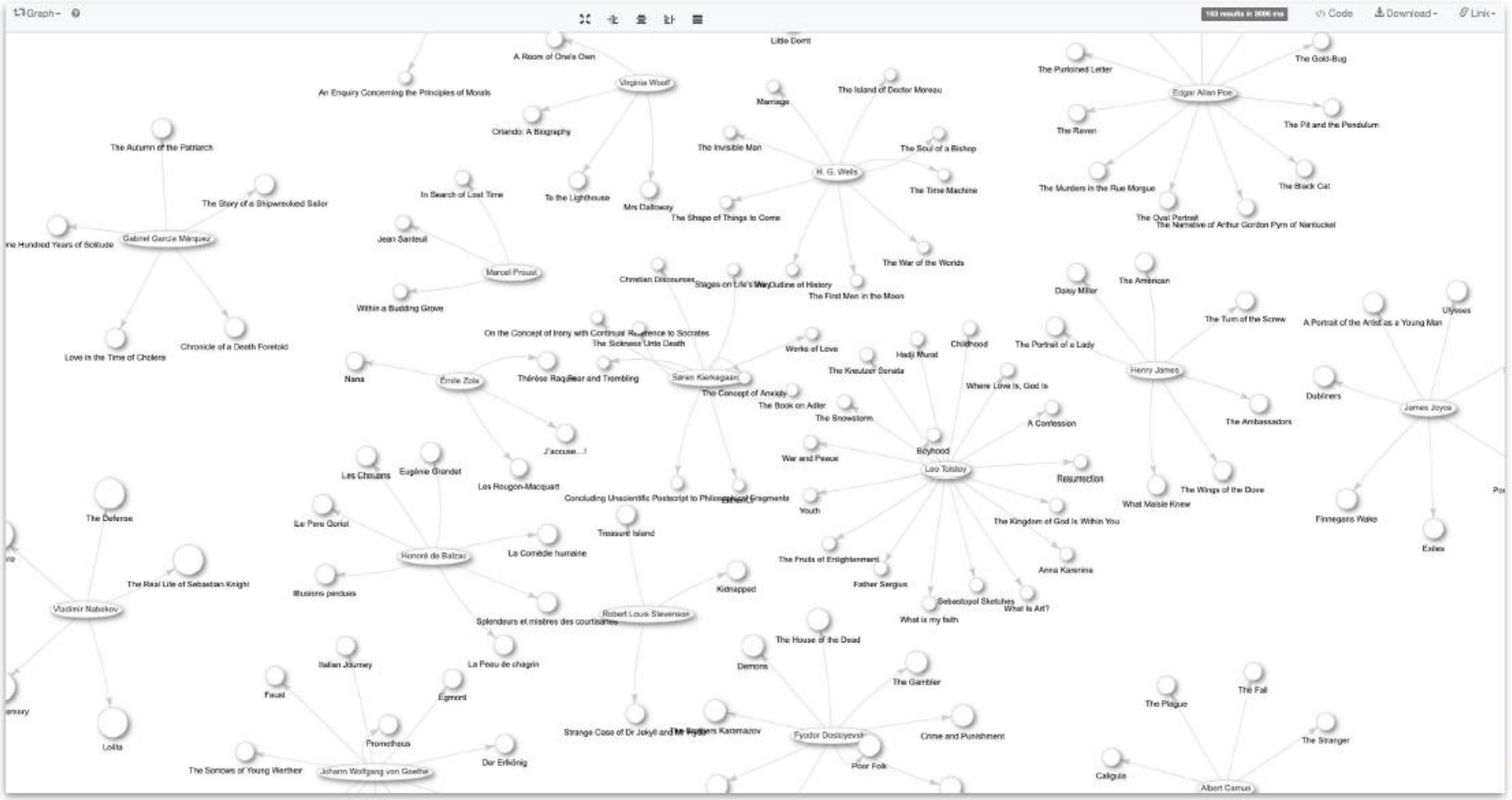
```
#defaultView:Graph
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT ?influencer ?influencerLabel ?book ?bookLabel ?authorCount
WHERE {
  { SELECT ?influencer (COUNT(?author) AS ?authorCount)
    WHERE {
      ?author wdt:P106 wd:Q36180 ;
              wdt:P737 ?influencer .
      ?influencer wdt:P106 wd:Q36180 .
    } GROUP BY ?influencer ORDER BY DESC(?authorCount)
      LIMIT 30
  }
  ?influencer wdt:P800 ?book .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} ORDER BY DESC(?authorCount) ?influencerLabel
```

- With the Wikidata SPARQL endpoint, we are able to display the result as a graph

query SPARQL endpoint

SPARQL Subqueries

- Example: **what books** were written by the **30 most influential authors**?



- With the Wikidata SPARQL endpoint, we are able to display the result as a **graph**

WIKIDATA

query SPARQL endpoint

SPARQL Property Paths

- A **property path** is a possible route through an RDF graph between two graph nodes.
  - trivial case: property path of length 1, i.e. a triple pattern
  - **alternatives**: match one or both possibilities

```
{ :book1 dc:title|rdfs:label ?displayString . }
```

  - **sequence**: property path of length >1

```
{ :alice foaf:knows/foaf:knows/foaf:name ?name . }
```

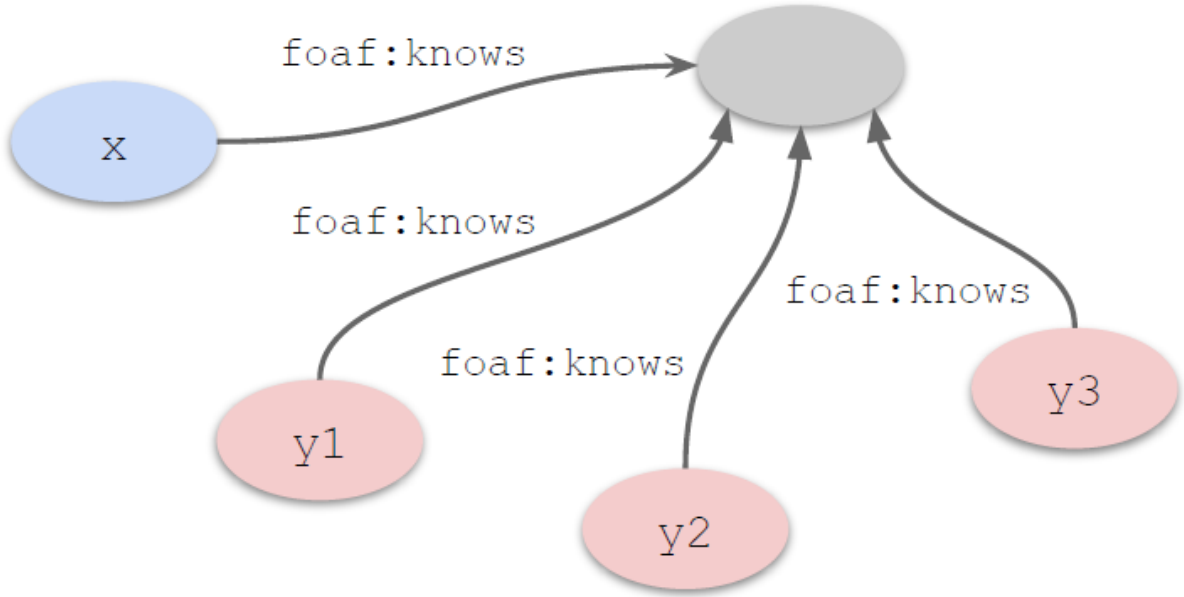
  - **inverse property paths**: reversing the direction of the triple

```
{ ?x foaf:mbox <mailto:alice@example> . }
                    =
{ <mailto:alice@example> ^foaf:mbox ?x . }
```

SPARQL Property Paths

- **inverse path sequences**

```
{ ?x foaf:knows/^foaf:knows ?y .
    FILTER (?x != ?y) }
```



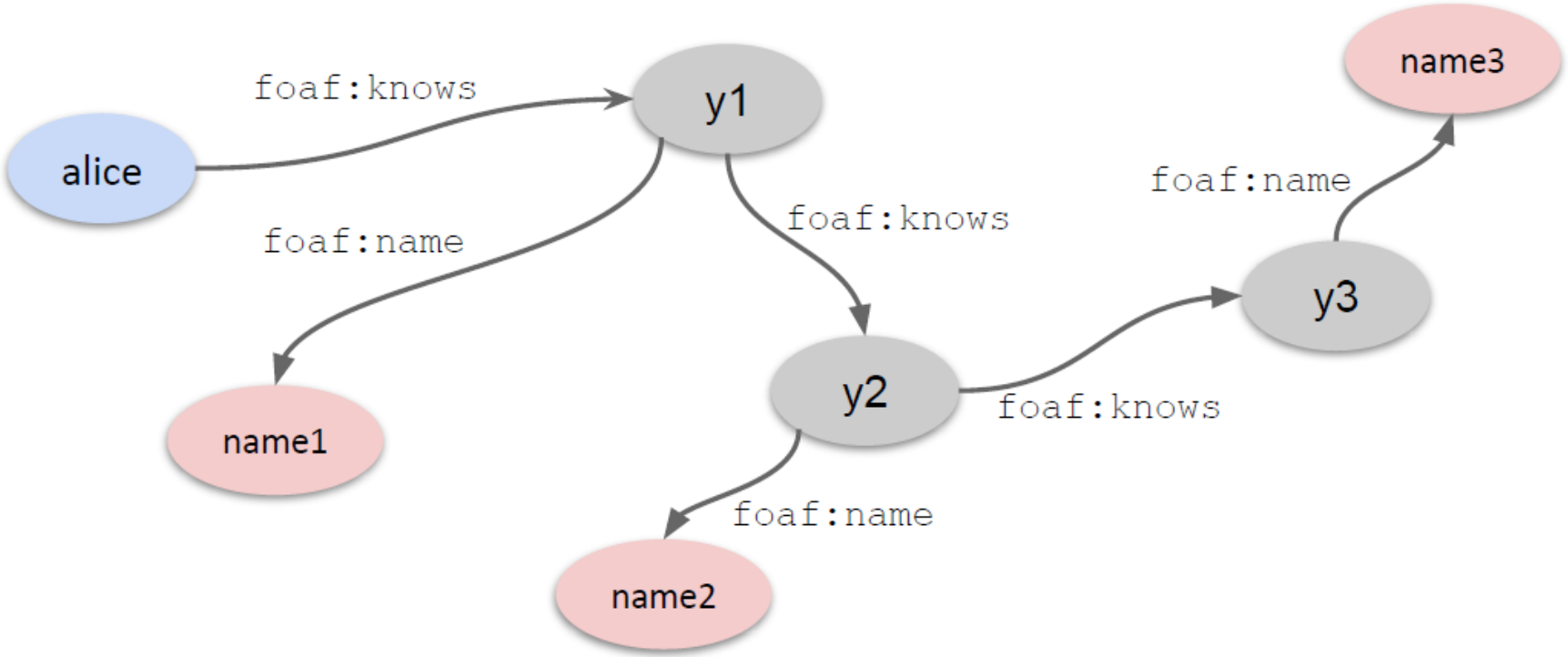
- Example:
  who else besides me knows the people I know?

SPARQL Property Paths

- **arbitrary length match**

```
{ :alice foaf:knows+/foaf:name ?name . }
```

- Example: what are the names of all persons I know and those who they know (and so on...) ?



SPARQL Property Paths

- Example: who else was influenced by the influencers of George Orwell?

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX bd: <http://www.bigdata.com/rdf#>

SELECT  ?influencedByInfluencersLabel
WHERE {
  wd:Q3335 wdt:P737|^wdt:P737 ?influencedByInfluencers
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
}
```

SPARQL Property Paths

- Example: who else was influenced by the influencers of George Orwell?