

Information Service Engineering

Lecture 8: Knowledge Graphs - 3



Leibniz Institute for Information Infrastructure

Prof. Dr. Harald Sack

IIZ Karlsruhe - Leibniz Institute for Information Infrastructure

KIIFB - Karlsruhe Institute of Technology

ummer Semester 2021

Last Lecture: Knowledge Graphs - 2

3.1 Knowledge Representations and Ontologies

3.2 Semantic Web and the Web of Data

3.3 Linked Data Principles

3.4 How to identify Things - URIs

**3.5 Resource Description Framework (RDF)
as simple Data Model**

3.6 Creating new Models with RDFS

3.7 Knowledge Graphs

3.8 Querying Knowledge Graphs with SPARQL

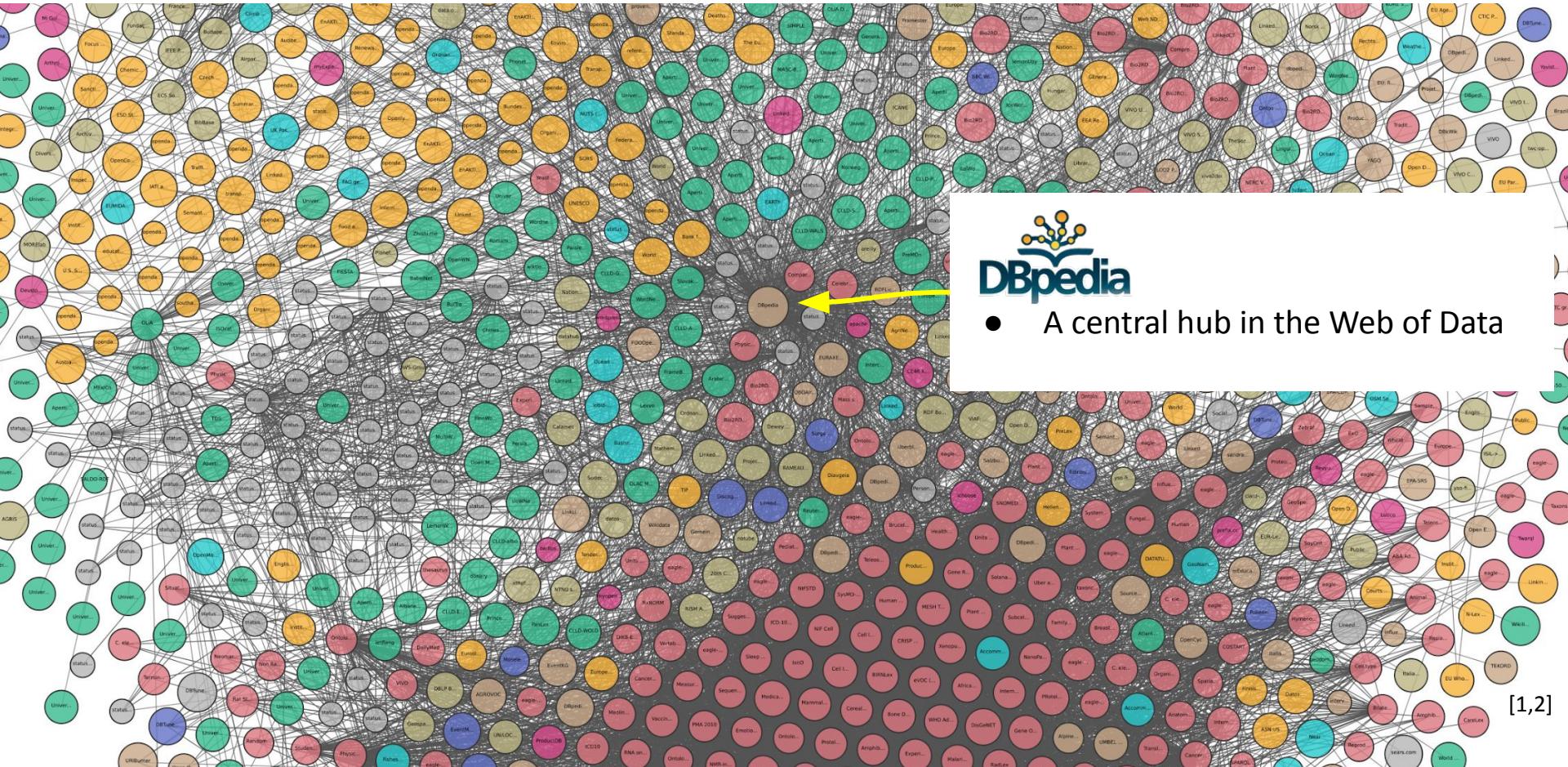
3.9 More Expressivity with Web Ontology Language (OWL)

3.10 Knowledge Graph Programming

- RDF Building Blocks
- RDF Blank Nodes
- RDF Schema
- RDF(S) Inference
- Knowledge Graphs
- The web of Data

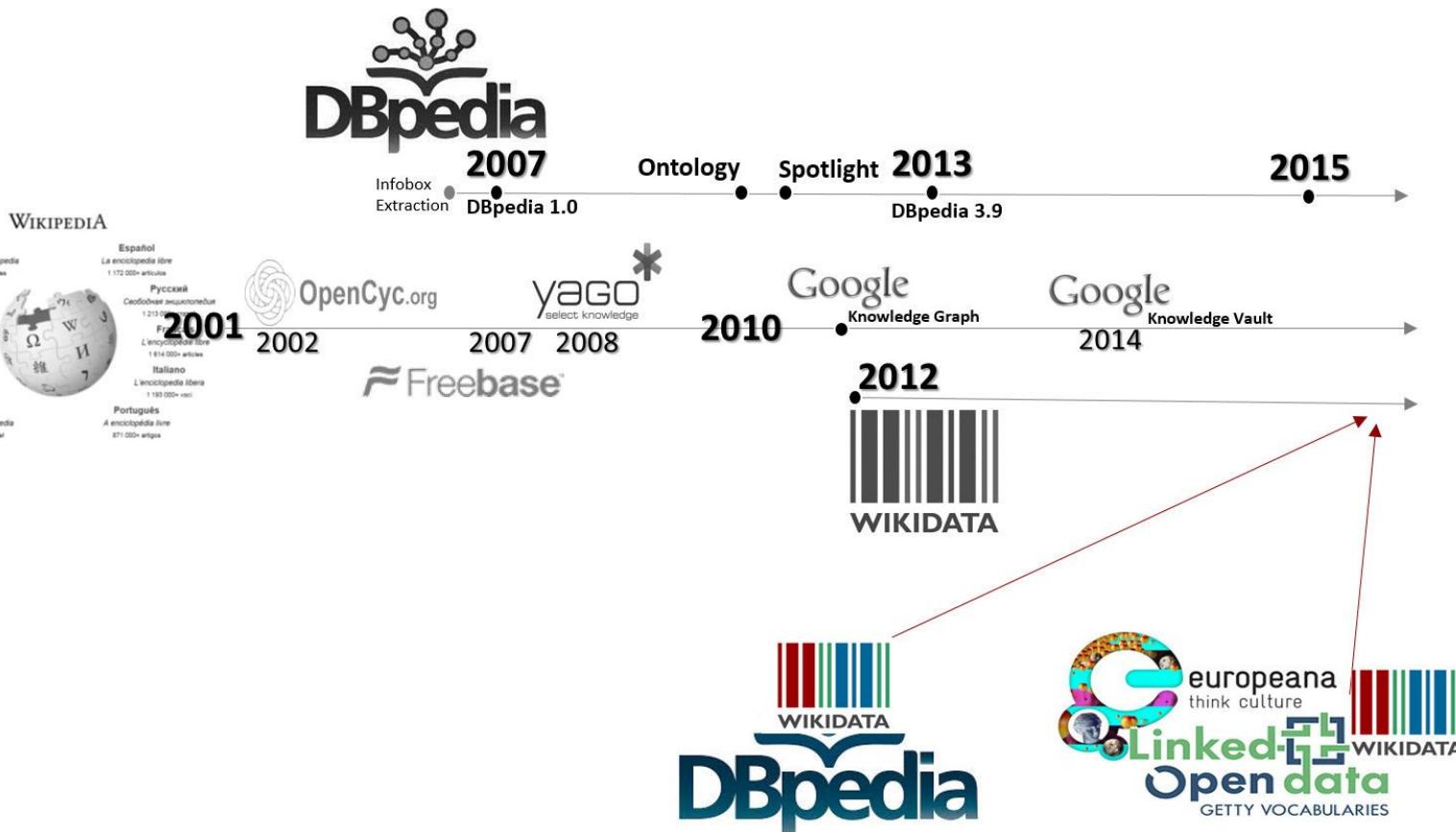
- 3.1 Knowledge Representations and Ontologies
- 3.2 Semantic Web and the Web of Data
- 3.3 Linked Data Principles
- 3.4 How to identify Things - URIs
- 3.5 Resource Description Framework (RDF) as simple Data Model
- 3.6 Creating new Model
- 3.7 Knowledge Graphs**  **Excursion: DBpedia Knowledge Graph**
- 3.8 Querying Knowledge Graphs with SPARQL
- 3.9 More Expressivity with Web Ontology Language (OWL)
- 3.10 Knowledge Graph Programming

DBpedia and the Web of Data



- A central hub in the Web of Data

Wikidata and the Web of Data



DBpedia

English version of the DBpedia Knowledge Graph (as by January 2020)

- describes 6.6 million things,
- of which 5.5 million are classified in a consistent [ontology](#)
- including 1.5 million persons,
- 840,000 places (including 513,000 populated places),
- 496,000 creative works
 - including 139,000 music albums,
 - 111,000 films and
 - 21,000 video games,
- 286,000 organizations
 - including 70,000 companies and 55,000 educational institutions,
- 306,000 species and
- 6,000 diseases.
- 125 localized DBpedia versions with overall 38.3 million things.



<https://wiki.dbpedia.org/develop/datasets>

DBpedia German Language Chapter

[HOME](#)[DATASET & QUERIES](#)[NEWS](#)[SPARQL ENDPOINT](#)

CONTACT

If you have questions or if you would like to collaborate, please contact

- [Fabian Hoppe](#) and
- [Tabea Tietz](#).

 Follow DBpedia Deutsch on Twitter



DATASET

The German DBpedia SPARQL Endpoint currently uses the dataset version of October 2016. We hope to update the endpoint soon, stay tuned! Until then, you can either use the Endpoint to query the dataset directly or download the dumpfile.

The datasets are made accessible on the Web under the terms of the Creative Commons Attribution-ShareAlike 3.0 License and the GNU Free Documentation License.

[Download](#)

Statistics

The dataset currently contains:

- 104.989.619 triples
- 1.374.894 entities
- 627.264 persons
- 62.054 organizations
- 406.943 locations

<http://de.dbpedia.org/>

From Wikipedia to DBpedia

http://en.wikipedia.org/wiki/Carbon_dioxide



WIKIPEDIA
The Free Encyclopedia

Not logged in | Talk | Contributions | Create account | Log in

Article | Talk | Read | Edit | View history | Search Wikipedia |

Carbon dioxide

From Wikipedia, the free encyclopedia

"CO₂" redirects here. For other uses, see *CO₂ (disambiguation)*.

Carbon dioxide (chemical formula CO₂) is a colorless gas with a density about 60% higher than that of dry air. Carbon dioxide consists of a carbon atom covalently double bonded to two oxygen atoms. It occurs naturally in Earth's atmosphere as a trace gas. The current concentration is about 0.04% (410 ppm) by volume, having risen from pre-industrial levels of 280 ppm.^[6] Natural sources include volcanoes, hot springs and geysers, and it is freed from carbonate rocks by dissolution in water and acids. Because carbon dioxide is soluble in water, it occurs naturally in groundwater, rivers and lakes, ice caps, glaciers and seawater. It is present in deposits of petroleum and natural gas. Carbon dioxide is odorless at normally encountered concentrations. However, at high concentrations, it has a sharp and acidic odor.^[1]

As the source of available carbon in the carbon cycle, atmospheric carbon dioxide is the primary carbon source for life on Earth and its concentration in Earth's pre-industrial atmosphere since late in the Precambrian has been regulated by photosynthetic organisms and geological phenomena. Plants, algae and cyanobacteria use light energy to photosynthesize carbohydrate from carbon dioxide and water, with oxygen produced as a waste product.^[7]

CO₂ is produced by all aerobic organisms when they metabolize carbohydrates and lipids to produce energy by respiration.^[8] It is returned to water via the gills of fish and to the air via the lungs of air-breathing land animals, including humans. Carbon dioxide is produced during the processes of decay of organic materials and the fermentation of sugars in bread, beer and wine making. It is produced by combustion of wood and other organic materials and fossil fuels such as coal, peat, petroleum and natural gas. It is an unwanted byproduct in many large scale oxidation processes, for example, in the production of acrylic acid (over 5 million tons/year).^{[9][10][11][12]}

It is a versatile industrial material, used, for example, as an inert gas in welding and fire extinguishers, as a pressurizing gas in air guns and oil recovery, as a chemical feedstock and as a supercritical fluid solvent in decaffeination of coffee^[13] and supercritical drying. It is added to drinking water and carbonated beverages including beer and sparkling wine to add effervescence. The frozen solid form of CO₂, known as *dry ice* is used as a refrigerant and as an abrasive in dry-ice blasting.

Carbon dioxide is the most significant long-lived greenhouse gas in Earth's atmosphere. Since the Industrial Revolution anthropogenic emissions – primarily from use of fossil fuels and deforestation – have rapidly increased its concentration in the atmosphere, leading to global warming. Carbon dioxide also causes ocean acidification because it dissolves in water to form carbonic acid.^[14]

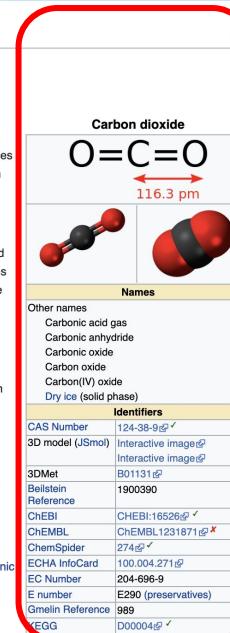
Chemical structure: O=C=O (bond length: 116.3 pm)

Names:

- Other names: Carbonic acid gas, Carbonic anhydride, Carbonic oxide, Carbon oxide, Carbon(V) oxide, Dry ice (solid phase)

Identifiers:

| | |
|------------------|--|
| CAS Number | 124-38-9 ✓ |
| 3D model (JSmol) | Interactive image ↗ Interactive image ↗ |
| 3DMet | B01131 ↗ |
| Beilstein | 1900390 |
| Reference | |
| ChEBI | CHEBI:16526 ✓ |
| ChEMBL | ChEMBL1231871 ↗ ✗ |
| ChemSpider | 274 ↗ ✓ |
| ECHA InfoCard | 100-004-271 ↗ |
| EC Number | 204-696-9 |
| E number | E290 (preservatives) |
| Gmelin Reference | 989 |
| EGG | D00004 ↗ ✓ |



http://dbpedia.org/resource/Carbon_dioxide

From Wikipedia to DBpedia

http://dbpedia.org/resource/Carbon_dioxide

 Browse using ▾ Formats ▾
 Faceted Browser Sparql Endpoint

About: Carbon dioxide

An Entity of Type : [chemical compound](#), from Named Graph : [http://dbpedia.org](#), within Data Space : [dbpedia.org](#)

Carbon dioxide (chemical formula CO₂) is a colorless and odorless gas vital to life on Earth. This naturally occurring chemical compound is composed of a carbon atom covalently double bonded to two oxygen atoms. Carbon dioxide exists in Earth's atmosphere as a trace gas at a concentration of about 0.04 percent (400 ppm) by volume. Natural sources include volcanoes, hot springs and geysers, and it is freed from carbonate rocks by dissolution in water and acids. Because carbon dioxide is soluble in water, it occurs naturally in groundwater, rivers and lakes, in ice caps and glaciers and also in seawater. It is present in deposits of petroleum and natural gas.

| Property | Value |
|------------------------------|---|
| dbo:abstract | <ul style="list-style-type: none"> ▪ Carbon dioxide (chemical formula CO₂) is a colorless and odorless gas vital to life on Earth. This naturally occurring chemical compound is composed of a carbon atom covalently double bonded to two oxygen atoms. Carbon dioxide exists in Earth's atmosphere as a trace gas at a concentration of about 0.04 percent (400 ppm) by volume. Natural sources include volcanoes, hot springs and geysers, and it is freed from carbonate rocks by dissolution in water and acids. Because carbon dioxide is soluble in water, it occurs naturally in groundwater, rivers and lakes, in ice caps and glaciers and also in seawater. It is present in deposits of petroleum and natural gas. ▪ Kohlenstoffdioxid oder Kohlendioxid ist eine chemische Verbindung aus Kohlenstoff und Sauerstoff mit der Summenformel CO₂. In Wasser gelöst wird es umgangssprachlich als „Kohlensäure“ bezeichnet. Kohlenstoffdioxid ist ein unbrennbares, saures, farb- und geruchloses Gas. |

DBpedia Naming Conventions

https://en.wikipedia.org/wiki/Carbon_dioxide



WIKIPEDIA
The Free Encyclopedia



Entity Identifier

HTML version

RDF/XML version



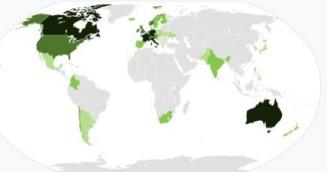
Wikipedia Infoboxes

| Carbon dioxide | |
|--|---|
| $O=C=O$ | 116.3 pm |
|  |  |
| Names | |
| Other names | |
| Carbonic acid gas | |
| Carbonic anhydride | |
| Carbonic oxide | |
| Carbon oxide | |
| Carbon(IV) oxide | |
| Dry ice (solid phase) | |
| Identifiers | |
| CAS Number | 124-38-9  |
| 3D model (JSmol) | Interactive image  Interactive image  |
| 3DMet | B01131  |
| Beilstein Reference | 1900390 |
| ChEBI | CHEBI:16526  |
| ChEMBL | ChEMBL1231871  |
| ChemSpider | 274  |
| ECHA InfoCard | 100-004-271  |
| EC Number | 204-696-9 |
| E number | E290 (preservatives) |
| Gmelin Reference | 989 |
| KEGG | D00004  |
| MeSH | Carbon-dioxide  |
| PubChem CID | 280  |
| RTECS number | FF6400000 |
| UNII | 142M471B3J  |
| UN number | 1013 (gas), 1845 (solid) |
| CompTox Dashboard (EPA) | DTXSID4027028  |
| InChI | [show] |
| SMILES | [show] |
| Properties | |

| |
|---|
| Greta Thunberg |
|  |
| Greta Thunberg in April 2019 |
| Born |
| Greta Ernman Thunberg 3 January 2003 (age 16) Stockholm, Sweden |
| Occupation |
| Student and climate activist |
| Movement |
| School strike for climate |
| Parent(s) |
| Svante Thunberg Malena Ernman |
| Relatives |
| Olof Thunberg (grandfather) |

| |
|---|
| Karlsruhe |
|  |
|  |
|  |
|  |
| Karlsruhe Palace, view over Karlsruhe, Schlossplatz, Konzerthaus, Crown of Baden |
|  |
|  |
| Location of Karlsruhe [show] |
|  |
| Karlsruhe |
| Show map of Germany |

| |
|---|
| An Inconvenient Truth |
|  |
| Theatrical release poster |
| Directed by |
| Davis Guggenheim |
| Produced by |
| Laurie David Lawrence Bender Scott Z. Burns |
| Written by |
| Al Gore |
| Starring |
| Al Gore |
| Music by |
| Michael Brook |
| Cinematography |
| Bob Richman Davis Guggenheim |
| Edited by |
| Jay Cassidy Dan Swietlik |
| Production company |
| Lawrence Bender Productions Participant Productions |
| Distributed by |
| Paramount Classics |
| Release date |
| May 24, 2006 |
| Running time |
| 97 minutes ^[1] |

| |
|---|
| School strike for climate |
| FridaysForFuture |
| Part of the climate movement |
|  |
| Maximum number of school strikers per country: |
|  100+  1000+  10 000+  100 000+ |
| Date |
| Since August 2018, mostly on Fridays, sometimes on Thursdays, Saturdays or Sundays |
| Location |
| International |
| Caused by |
| Political inaction against global warming |
| Goals |
| Climate change mitigation |
| Methods |
| Student strike |
| Status |
| Active |
| Parties to the civil conflict |
| Youth |
| Lead figures |
| Greta Thunberg |
| Number |
| estimated 1 400 000 (for 15 March 2019) ^[1] |

DBpedia Infobox Extraction - Infobox Properties

An Inconvenient Truth



Theatrical release poster

| | | |
|-----------------------|---|---|
| Directed by | Davis Guggenheim | http://dbpedia.org/resource/Davis_Guggenheim |
| Produced by | Laurie David Lawrence Bender Scott Z. Burns | |
| Written by | Al Gore | |
| Starring | Al Gore | |
| Music by | Michael Brook | |
| Cinematography | Bob Richman Davis Guggenheim | |

http://dbpedia.org/resource/An_Inconvenient_Truth

<http://dbpedia.org/property/director>

Directed by

Davis Guggenheim

Produced by

Laurie David

Lawrence Bender

Scott Z. Burns

Written by

Al Gore

Starring

Al Gore

Music by

Michael Brook

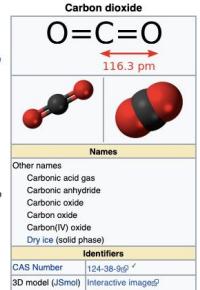
Cinematography

Bob Richman

Davis Guggenheim

DBpedia Category System - DBpedia Ontology Classes

dbr:Carbon_dioxide



rdf:type

dbo:Chemical_compound

rdfs:subClassOf

dbo:Chemical_substance

rdfs:subClassOf

owl:Thing

Prefixes:

rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>
 dbr: <<http://dbpedia.org/resource/>>
 dbo: <<http://dbpedia.org/ontology/>>
 owl: <<http://www.w3.org/2002/07/owl#>>
 rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

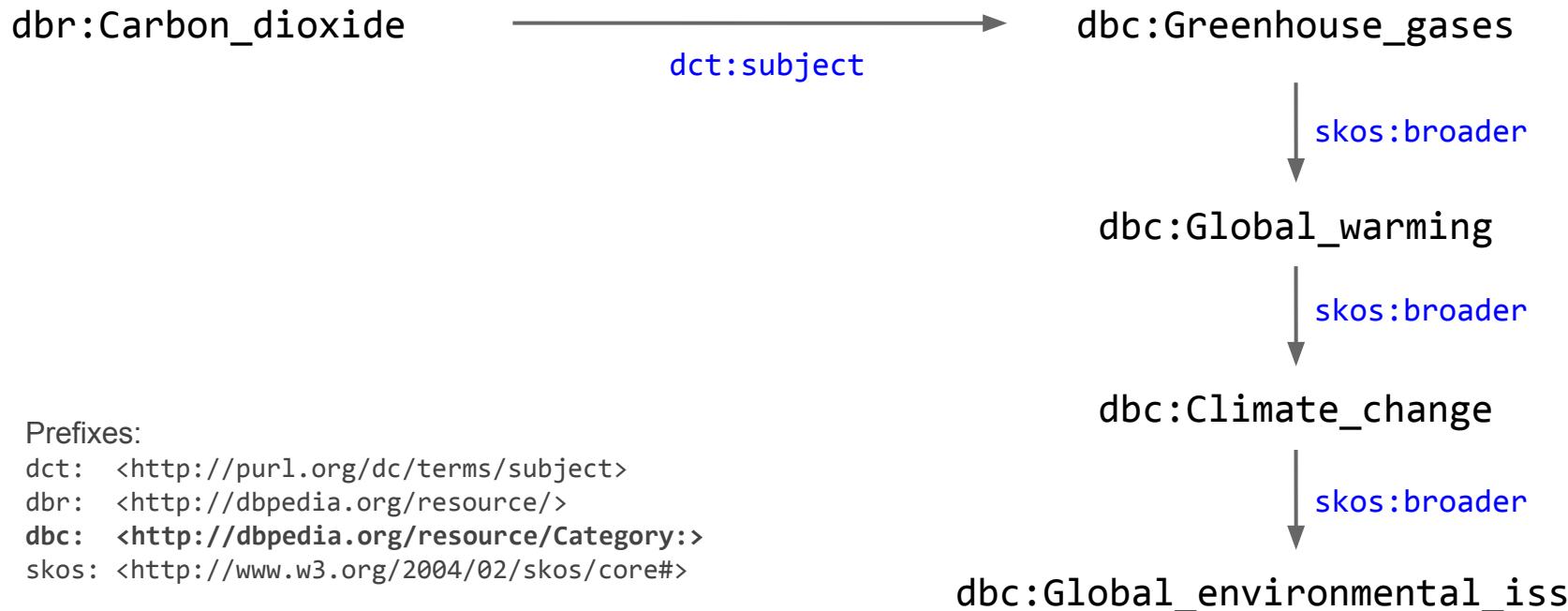
<http://mappings.dbpedia.org/server/ontology/classes/>

- Browser (edit)
- ChartsPlacements (edit)
- ChemicalSubstance (edit)
 - ChemicalCompound (edit)
 - ChemicalElement (edit)
 - Drug (edit)
 - CombinationDrug (edit)
 - MonoclonalAntibody (edit)
 - Vaccine (edit)
 - Mineral (edit)
- Cipher (edit)
- Colour (edit)
- Currency (edit)
- Demographics (edit)
- Depth (edit)
- Device (edit)
 - Battery (edit)
 - Camera (edit)
 - DigitalCamera (edit)
 - Engine (edit)
 - AutomobileEngine (edit)
 - RocketEngine (edit)
 - InformationAppliance (edit)
 - Instrument (edit)
 - Guitar (edit)
 - Organ (edit)
 - MobilePhone (edit)
 - Robot (edit)
 - Weapon (edit)
- Diploma (edit)

DBpedia Category System - Wikipedia Categories

https://en.wikipedia.org/wiki/Carbon_dioxide

Categories: Carbon dioxide | Acid anhydrides | Acidic oxides | Coolants | Fire suppression agents | Greenhouse gases | Household chemicals | Inorganic solvents | Laser gain media | Nuclear reactor coolants | Oxocarbons | Propellants | Refrigerants | Gaseous signaling molecules | Heterocumulenes | E-number additives



DBpedia SPARQL Endpoint

SPARQL Query Editor About Tables ▾

Conductor Facet Browser Permalink

Extensions: [cxml](#) [save to dav](#) [sponge](#) User: **SPARQL**

Default Data Set Name (Graph IRI)

Query Text

```
select distinct ?Concept where { [] a ?Concept} LIMIT 100
```

Results Format

Execution timeout milliseconds

Options

- Strict checking of void variables
- Strict checking of variable names used in multiple clauses but not logically connected to each other
- Suppress errors on wrong geometries and errors on geometrical operators (failed operations will return NULL)
- Log debug info at the end of output (has no effect on some queries and output formats)
- Generate SPARQL compilation report (instead of executing the query)



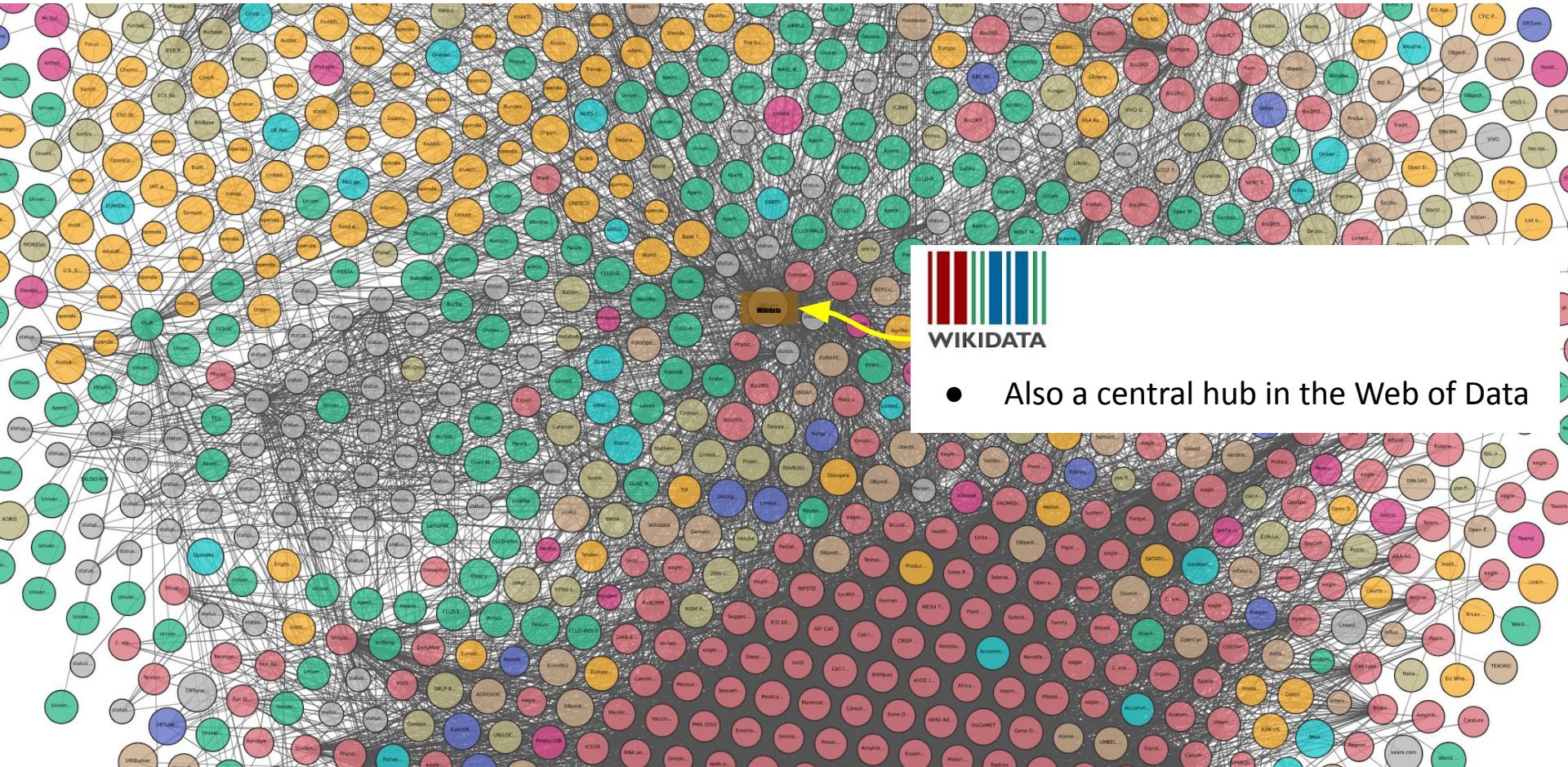
<http://dbpedia.org/sparql>

Information Service Engineering

Lecture 8: Knowledge Graphs - 3

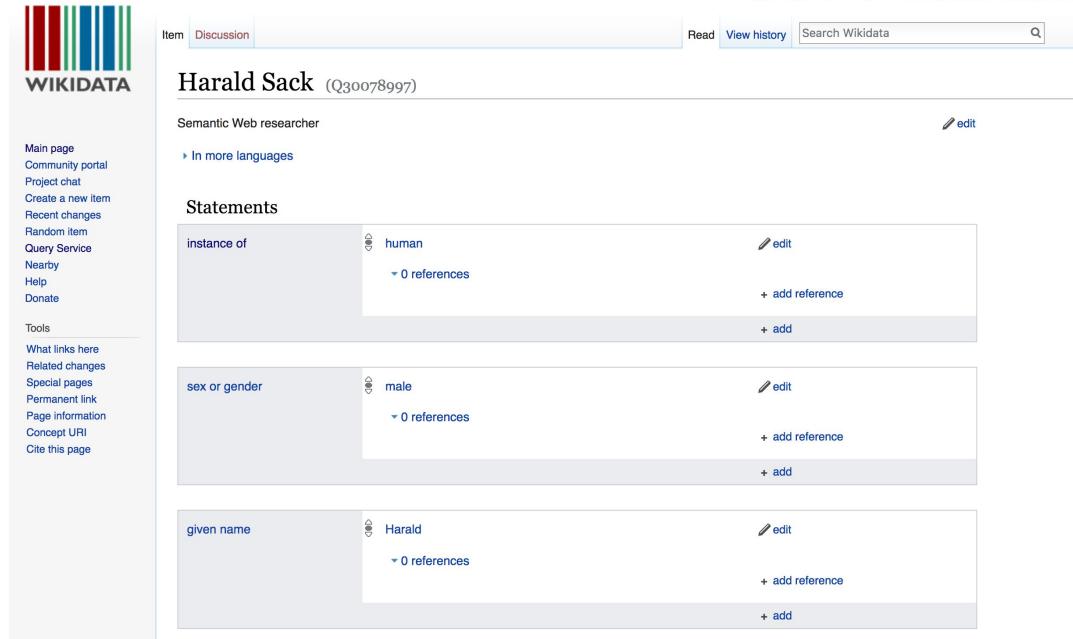
- 3.1 Knowledge Representations and Ontologies
- 3.2 Semantic Web and the Web of Data
- 3.3 Linked Data Principles
- 3.4 How to identify Things - URIs
- 3.5 Resource Description Framework (RDF) as simple Data Model
- 3.6 Creating new Model
- 3.7 Knowledge Graphs**  **Excursus: Wikidata Knowledge Graph**
- 3.8 Querying Knowledge Graphs with SPARQL
- 3.9 More Expressivity with Web Ontology Language (OWL)
- 3.10 Knowledge Graph Programming

Wikidata and the Web of Data



Wikidata

Collaboratively edited knowledge base operated by the Wikimedia Foundation (started in 2012)



The screenshot shows the Wikidata item page for Harald Sack (Q30078997). The page title is "Harald Sack (Q30078997)". Below the title, it says "Semantic Web researcher". There are links to "In more languages" and an "edit" button. The main section is titled "Statements". It lists three statements:

- instance of**: human (edit, + add reference, + add)
- sex or gender**: male (edit, + add reference, + add)
- given name**: Harald (edit, + add reference, + add)

Each statement row has a "0 references" link.

- > 93M entities (*May 2021*)
 - > 6.4M persons
 - > 1.9M populated places
 - > 3.1M architectural structures
 - > 3.9M events
 - > 1.2M chemical compounds
 - ~ 300K movies
 - > 4.6M astronomical objects
 - > 22.5M scholarly articles
- > 1.1B statements
- ~ 26K active users

<https://www.wikidata.org/>

Wikidata

Identifier: Joseph Fourier (Q8772)

French mathematician and physicist
Jean-Baptiste Joseph Fourier

Statements

- property**: instance of → **value**: human (with 2 references)
- property**: sex or gender → **value**: male (with 5 references)
- property**: country of citizenship → **value**: Kingdom of France
 - qualifiers**: start time → 21 March 1768 Gregorian
 - qualifiers**: end time → 21 September 1792 Gregorian
- property**: country of citizenship → **value**: French First Republic
 - qualifiers**: start time → 21 September 1792 Gregorian
 - qualifiers**: end time → 18 May 1804 Gregorian

<https://www.wikidata.org/wiki/Q8772>

Wikidata - SPARQL Query Service

Wikidata Query Service Examples Help More tools English

```

1 PREFIX wd: <http://www.wikidata.org/entity/>
2 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
3 PREFIX wikibase: <http://wikiba.se/ontology#>
4 PREFIX p: <http://www.wikidata.org/prop/>
5 PREFIX ps: <http://www.wikidata.org/prop/statement/>
6 PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
7 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
8 PREFIX bd: <http://www.bigdata.com/rdf#>
9
10 SELECT * WHERE {
11 wd:Q8772 ?p ?o .
12 }
13
14
  
```

502 results in 996 ms Code Download Link Search

| p | o |
|------------|--|
| p:P8947 | wds:Q8772-26286D22-0FCB-48FA-A9A0-EADDAC184651 |
| p:P9097 | wds:Q8772-42274727-31E6-4140-B297-F2DF84B5FB0F |
| rdfs:label | Joseph Fourier |
| rdfs:label | Joseph Fourier |
| rdfs:label | জোসেফ ফোরি |
| rdfs:label | Joseph Fourier |
| rdfs:label | Jean Baptiste Joseph Fourier |

[SPARQL query 1](#)
[SPARQL query 2](#)

Wikidata - More sophisticated SPARQL Queries

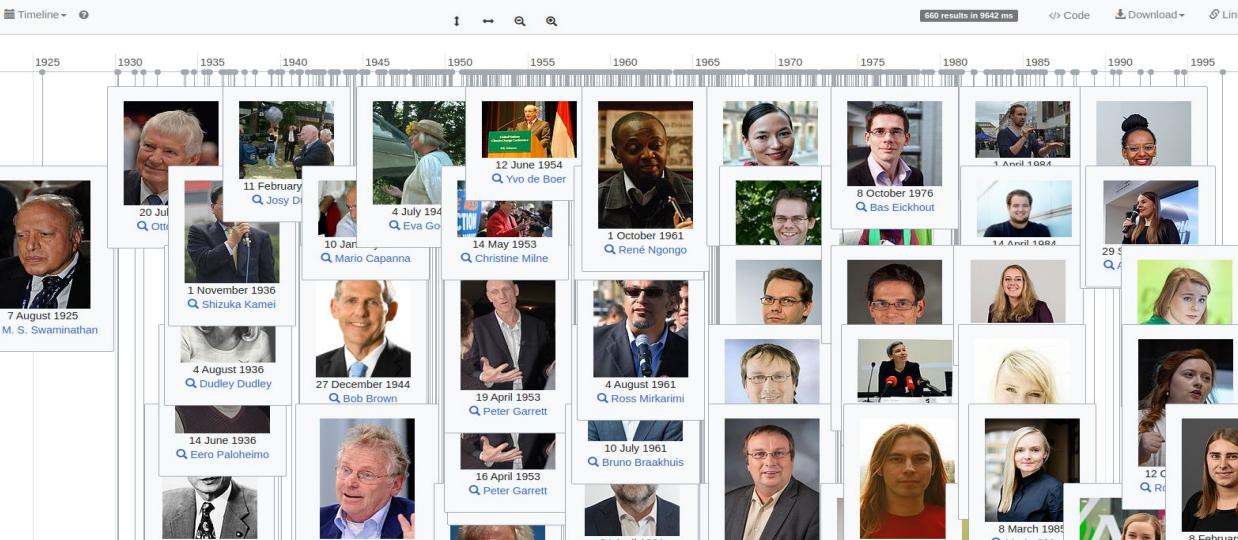
Wikidata Query Service Examples Help More tools English

```

1 #defaultView:Timeline
2 PREFIX dct: <http://purl.org/dc/terms/>
3 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
4 PREFIX dbc: <http://dbpedia.org/resource/Category:>
5 PREFIX dbo: <http://dbpedia.org/ontology/>
6
7 SELECT DISTINCT ?wditem ?wditemLabel ?date ?image WHERE {
8   SERVICE <http://dbpedia.org/sparql> { #Select from DBpedia
9     ?item dct:subject|dct:subject/skos:broader|dct:subject/skos:broader dbc:Environmentalists ; #Select all from Wikipedia Category "Environmentalist" or its subc
10    owl:sameAs ?wditem FILTER regex(?wditem, "wikidata.org") . #draw the connection to Wikidata
11  }
12  SERVICE <https://query.wikidata.org/sparql> { #Select from Wikidata
13    ?wditem wd:P106 wd:Q82955 . # the subject must have the occupation (P106) Politician (Q82955)
14    ?wditem wd:P569 ?date FILTER NOT EXISTS {?wditem wd:P570 ?date2} . # select the birthdate (P569) of still living (not P570) Politicians
15    ?wditem wd:P18 ?image . # Select an image (P18), if available
16    ?wditem rdfs:label FILTER (LANG(?wditemLabel)="en") . #Labels should be taken in English
17  }
18 } ORDER BY ?date

```

Timeline ▾ 660 results in 9642 ms Code Download Link



which (living)
politicians are
environmentalists ?

SPARQL query

Wikidata and the Web of Data

Caution: WIKIDATA is not a real Knowledge Base!

WIKIDATA is a Wiki based large **structured database**.

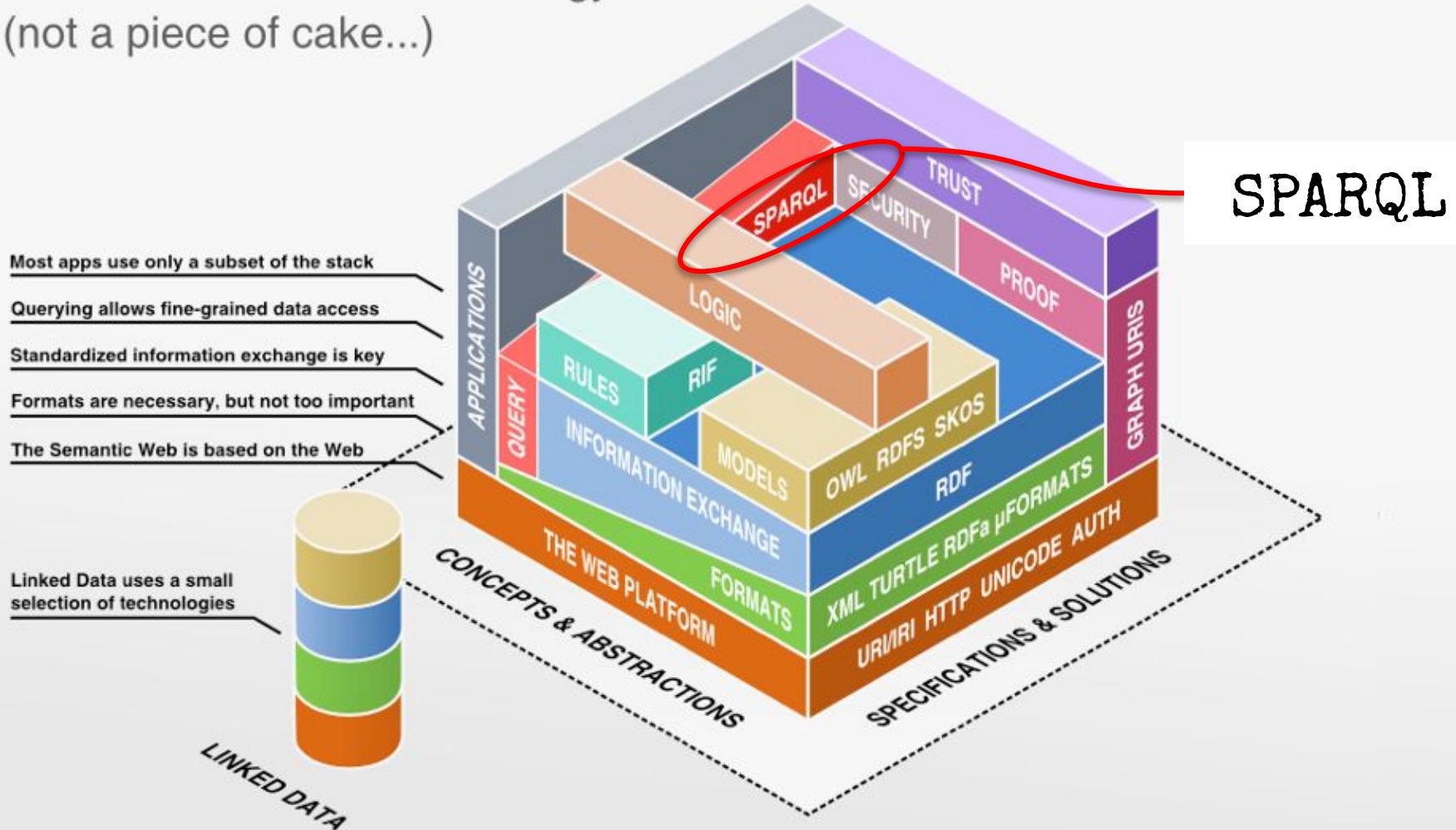
The available Triple Store and SPARQL query service is only an addendum.

WIKIDATA is **not** fully W3C compliant,

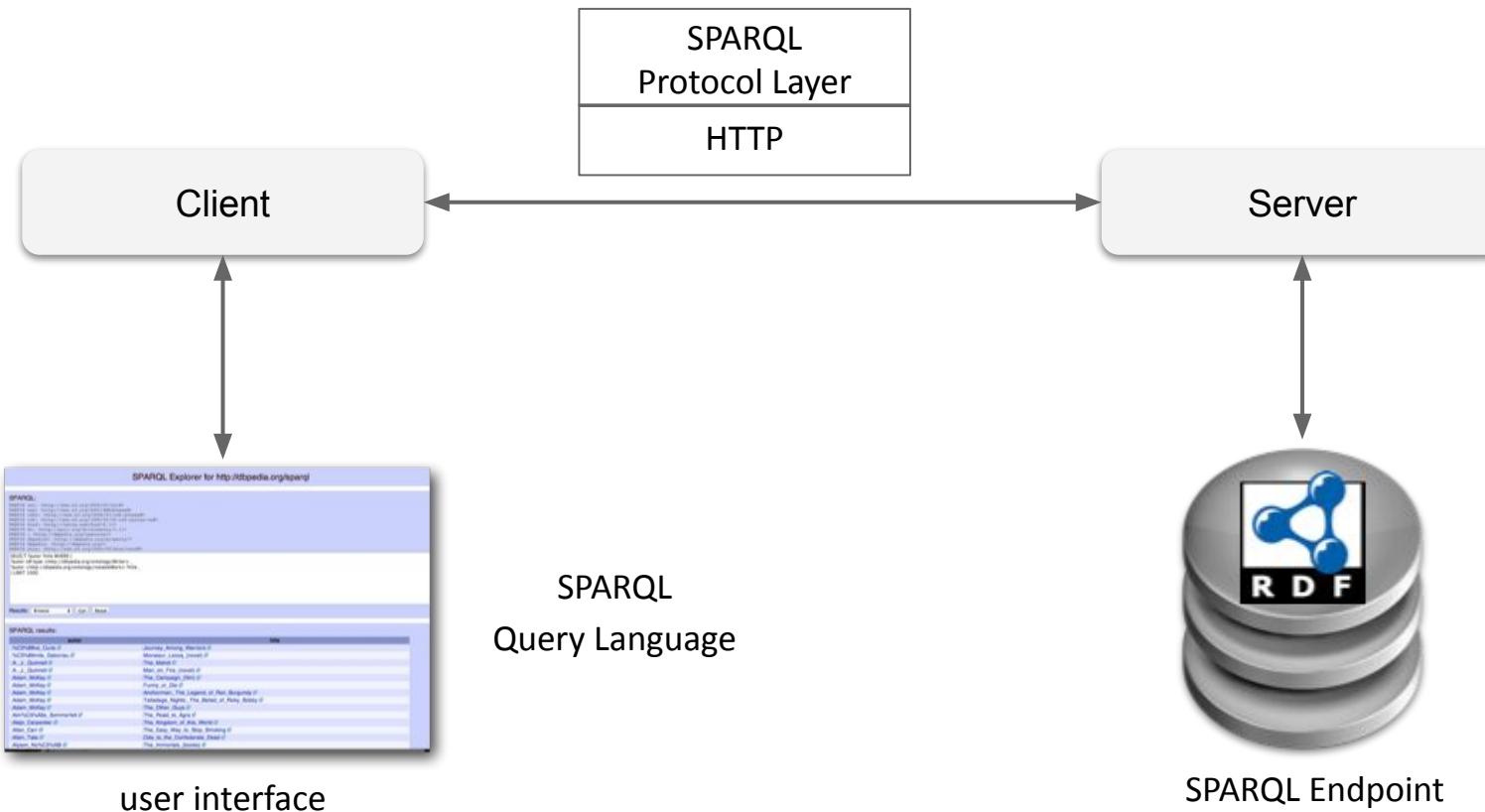
i.e. no W3C compliant vocabulary (RDF, RDFS, OWL) or semantics is used.

- 3.1 Knowledge Representations and Ontologies
- 3.2 Semantic Web and the Web of Data
- 3.3 Linked Data Principles
- 3.4 How to identify Things - URIs
- 3.5 Resource Description Framework (RDF) as simple Data Model
- 3.6 Creating new Models with RDFS
- 3.7 Knowledge Graphs
- 3.8 Querying Knowledge Graphs with SPARQL**
- 3.9 More Expressivity with Web Ontology Language (OWL)
- 3.10 Knowledge Graph Programming

The Semantic Web Technology Stack (not a piece of cake...)



SPARQL - A Query Language for RDF(S) Knowledge Graphs



SPARQL Endpoint Example

SPARQL Query Editor About Tables ▾

Conductor Facet Browser Permalink

Extensions: [cxml](#) [save to dav](#) [sponge](#) User: **SPARQL**

Default Data Set Name (Graph IRI)

Query Text

```
select distinct ?Concept where { [] a ?Concept} LIMIT 100
```

Results Format

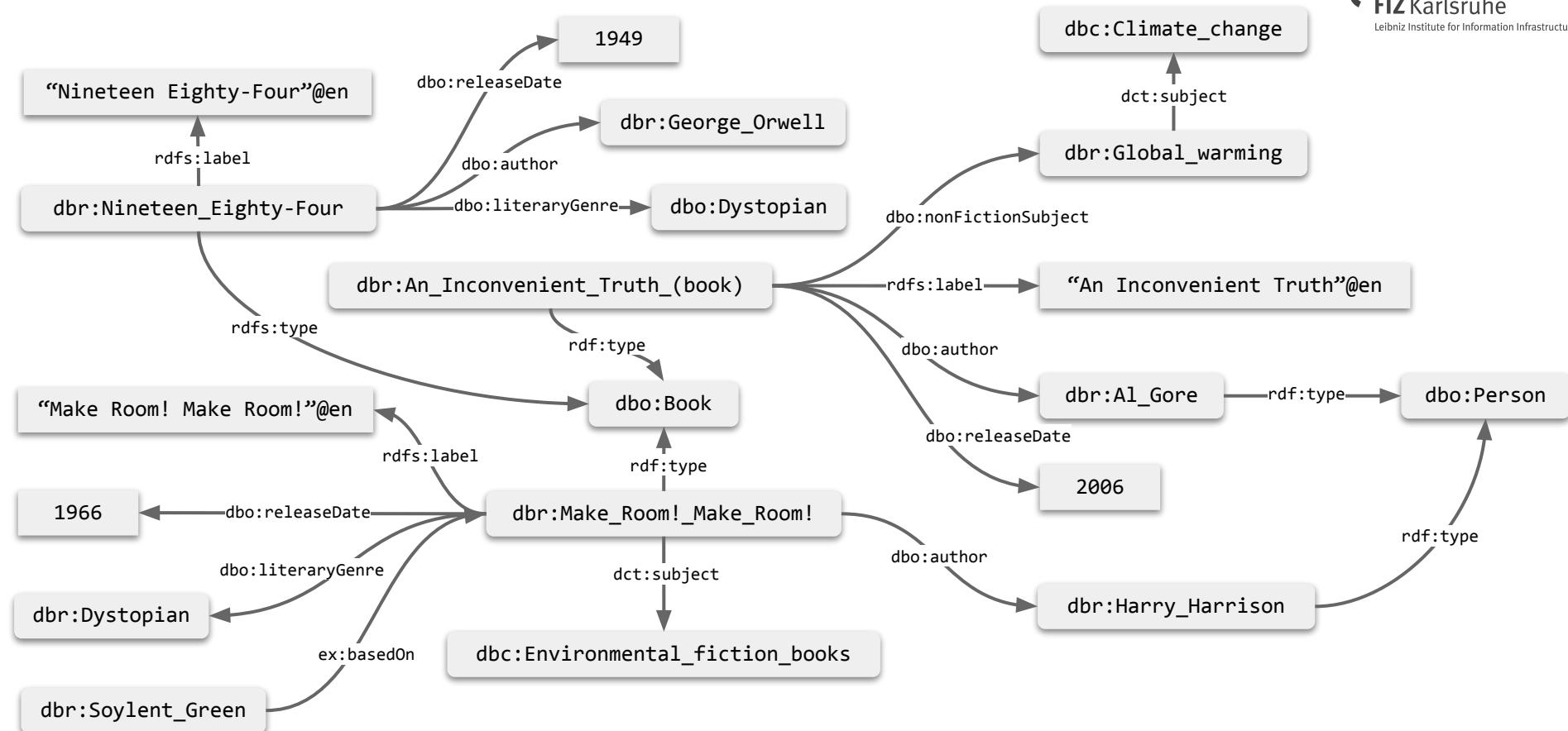
Execution timeout milliseconds

Options

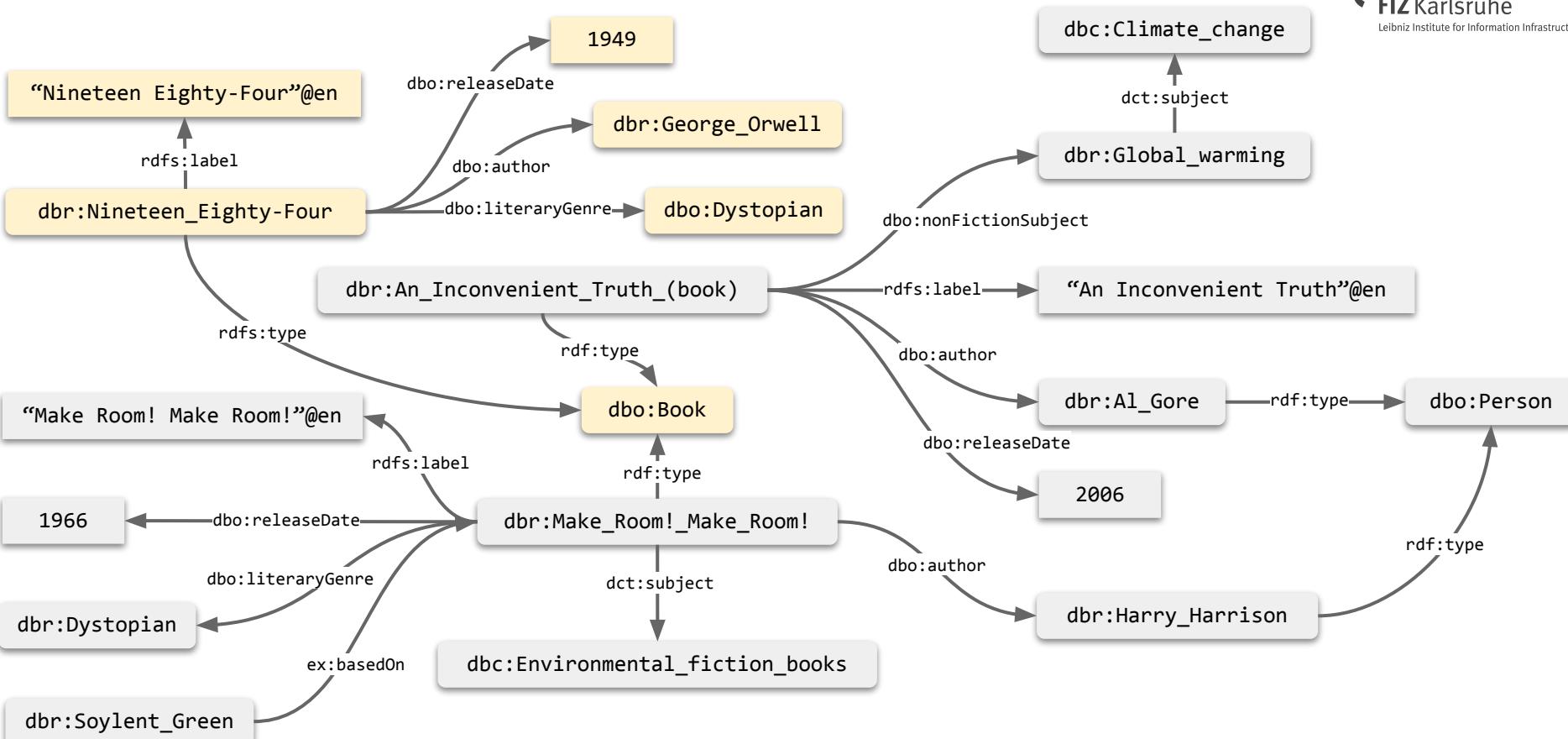
- Strict checking of void variables
- Strict checking of variable names used in multiple clauses but not logically connected to each other
- Suppress errors on wrong geometries and errors on geometrical operators (failed operations will return NULL)
- Log debug info at the end of output (has no effect on some queries and output formats)
- Generate SPARQL compilation report (instead of executing the query)


<http://dbpedia.org/sparql>

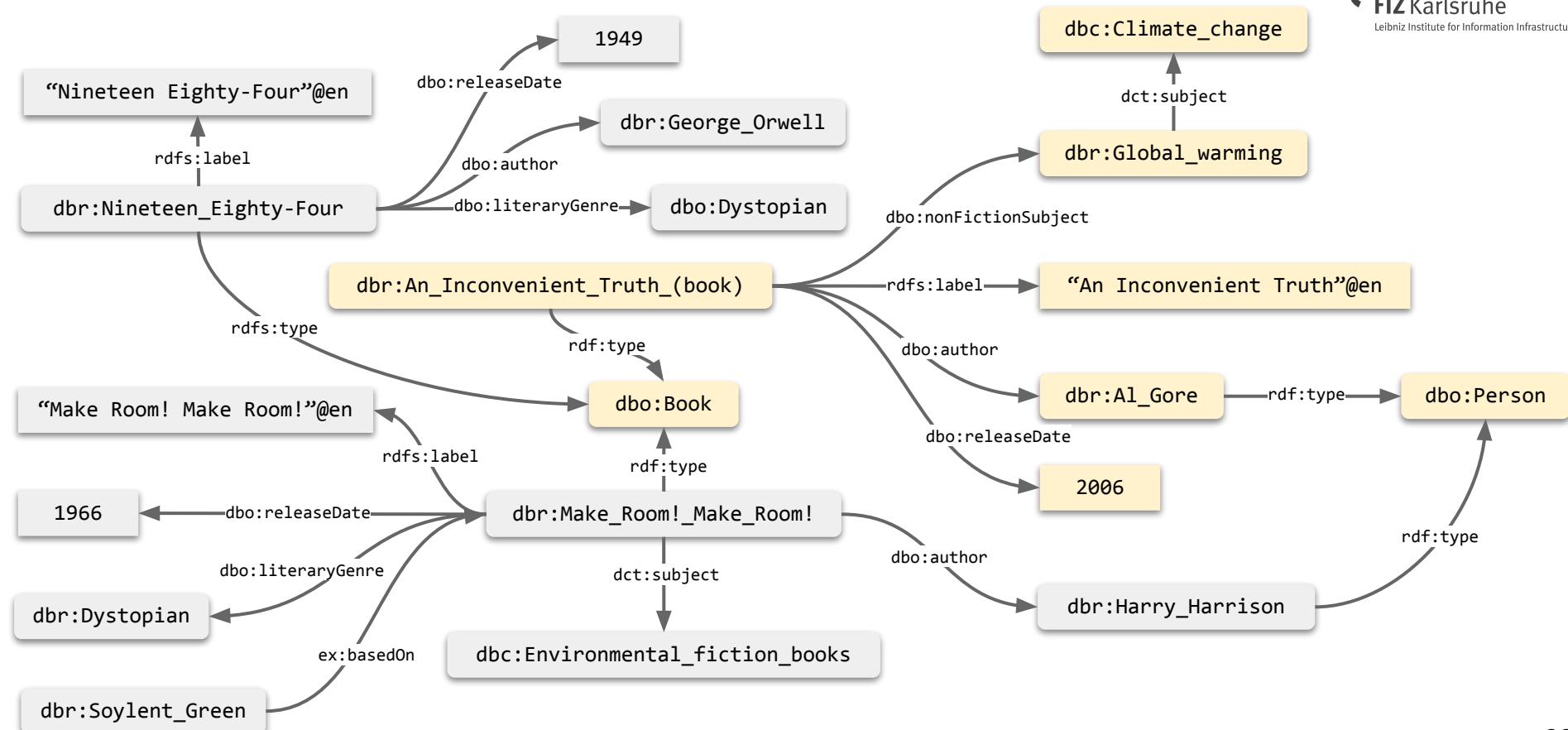
Querying an RDF-based Knowledge Graph



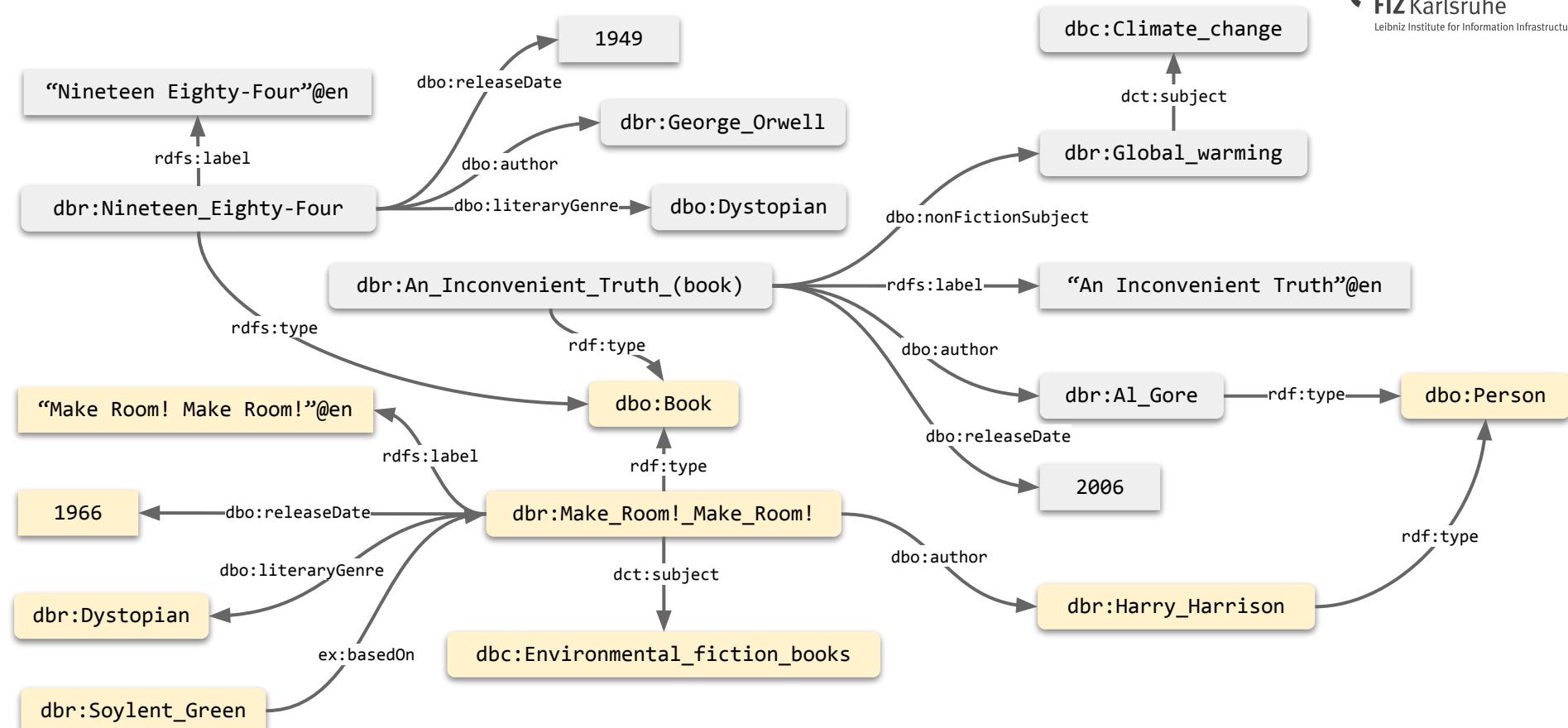
Querying an RDF-based Knowledge Graph



Querying an RDF-based Knowledge Graph



Querying an RDF-based Knowledge Graph



For Queries we need Variables

- 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**
- 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).

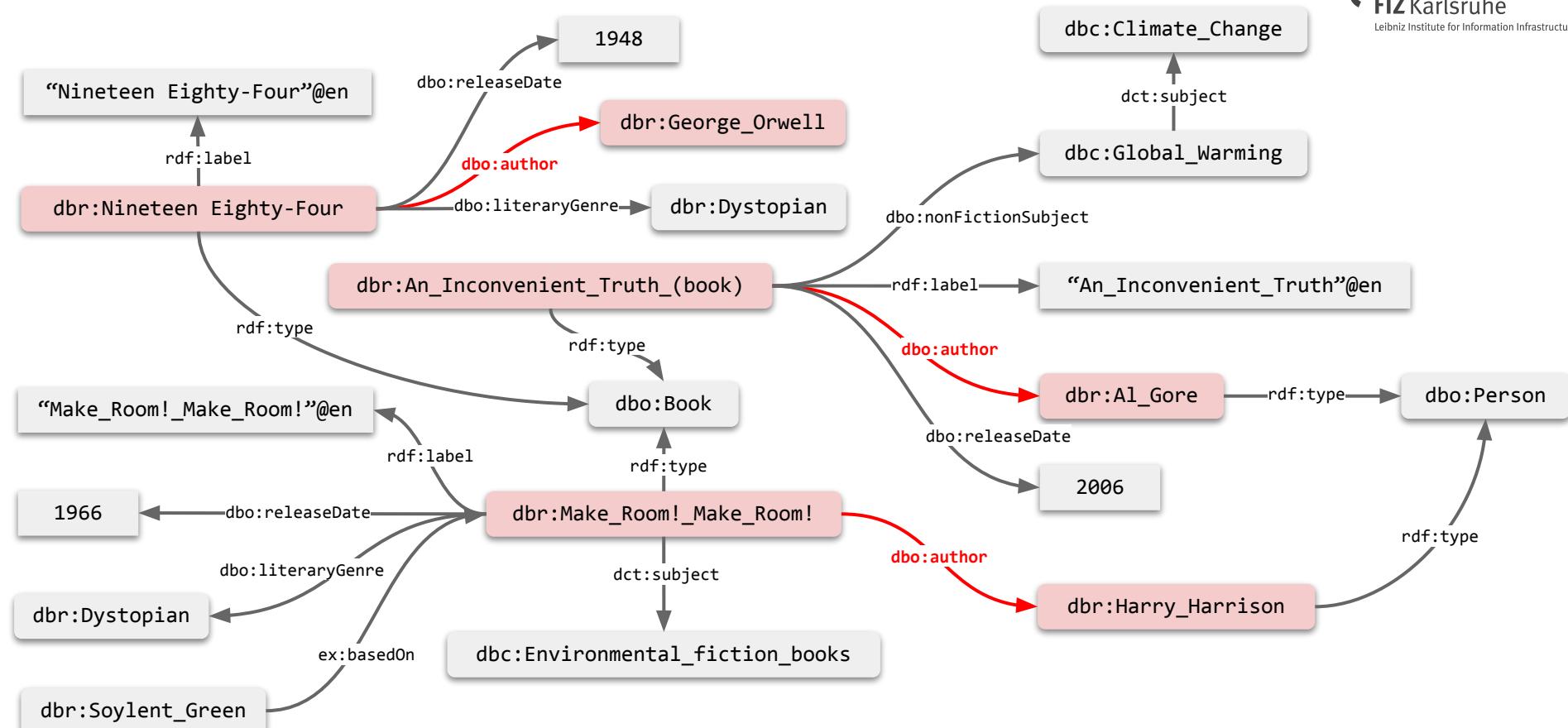
Graph Pattern (Triple Pattern) = Turtle + Variables

- Example:

*Look for **books** and their **authors** (via property **dbo:authors**):*



SPARQL Graph Pattern Matching



SPARQL Complex Query Pattern

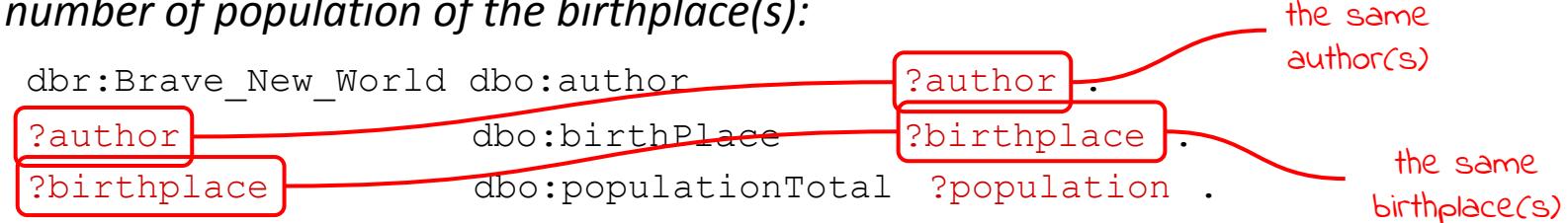
- SPARQL Graph Pattern can be combined to form **complex (conjunctive) queries** for RDF graph traversal.
- *Find books, their authors, and their genre:*

the same book(s)

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

SPARQL Complex Query Pattern

- 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 General Query Format

```
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 —— specifies output variables

FROM <http://dbpedia.org/> —— specifies graph to be queried

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

specifies namespaces

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

specifies graph pattern
to be matched



[query SPARQL endpoint](#)

SPARQL General Query Format

```

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) . specifies constraints  
for the result
    ?work rdfs:label ?title .
} LIMIT 100
  
```

- 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


[query SPARQL endpoint](#)

- FILTER expressions contain operators and functions

SPARQL Unary Operators

| 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](#)

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 . Instance of written work
    ?book wdt:P50 ?author . author
    ?book wdt:P921 wd:Q7942 . main subject global warming
    ?book wdt:P577 ?date . publication date
}
SERVICE wikibase:label
{ bd:serviceParam wikibase:language "en" }
```

- Example:
Search for authors and their books including publication date, 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**,
their books, and
their publication
dates, filter results
for **English labels** and
Books on Global
warming.



WIKIDATA

[query SPARQL endpoint](#)

- Example:

Search for authors, their books, and publication dates, filter results for English labels and Books on Global warming.

Wikidata Query Service Examples Help More tools

English

```

1 SELECT ?authorLabel ?bookLabel ?date
2 WHERE {
3   ?book wdt:P31 wd:Q47461344 . # instance of (P31) written work (Q47461344)
4   ?book wdt:P50 ?author .       # author (P50)
5   ?book wdt:P921 wd:Q7942 .     # main subject (P921) global warming (Q7942)
6   ?book wdt:P577 ?date .       # publication date (P577)
7   SERVICE wikibase:label
8   { bd:serviceParam wikibase:language "en" }
9 }
10

```



6 results in 749 ms Code Download Link

| 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 |
| Marcel Hänggi | Null Öl, null Gas, null Kohle | 1 January 2018 |

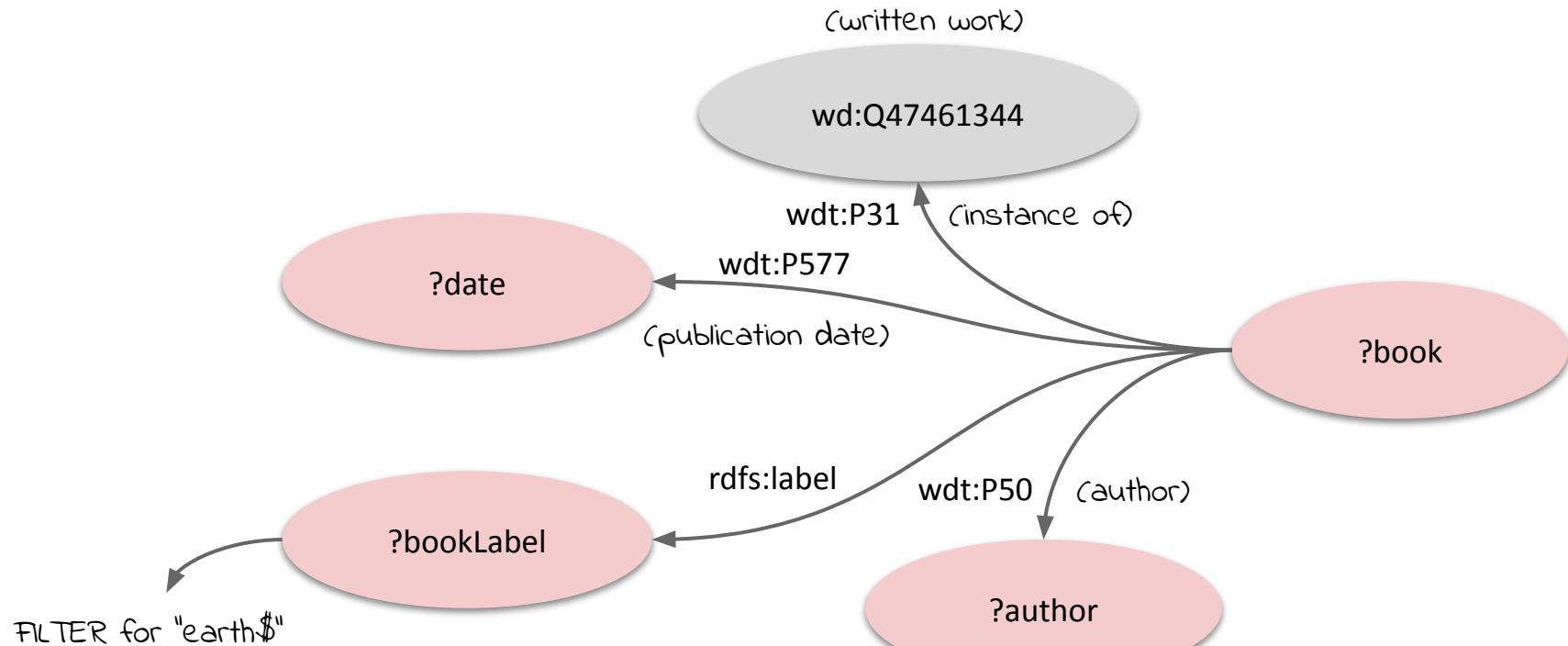
[query SPARQL endpoint](#)

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](#)

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

Wikidata Query Service Examples Help More tools English

```

1 PREFIX wd: <http://www.wikidata.org/entity/>
2 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
3 PREFIX wikibase: <http://wikiba.se/ontology#>
4 PREFIX bd: <http://www.bigdata.com/rdf#>
5
6 SELECT ?authorLabel ?bookLabel ?date
7 WHERE {
8   ?book wdt:P31 wd:Q4746134 .
9   ?book wdt:P50 ?author .
10  ?book wdt:P577 ?date .
11  ?book rdfs:label ?bookLabel FILTER (LANG(?bookLabel)="en")
12  FILTER regex (?bookLabel,"earth$","i") .
13  ?author rdfs:label ?authorLabel FILTER (LANG(?authorLabel)="en")
14 }
15 ORDER BY ?date
16

```

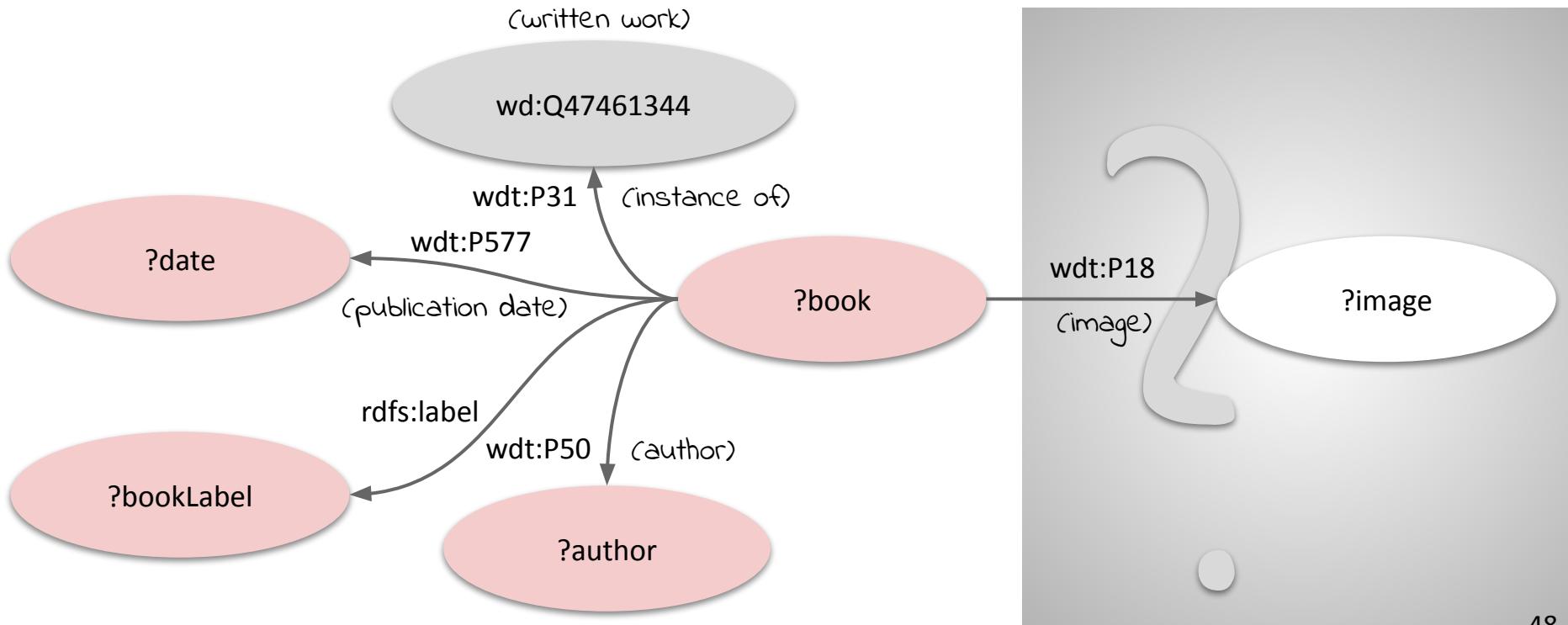
46 results in 5623 ms ↻ Code Download ⌂ Link

| 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 |

[query SPARQL endpoint](#)

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

[query SPARQL endpoint](#)

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

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

The screenshot shows a timeline visualization of famous books from 1840 to 2010. The timeline is represented by a horizontal axis with vertical markers for each year. Each book is shown as a box containing its title, author, and publication date. Some books also feature small thumbnail images.

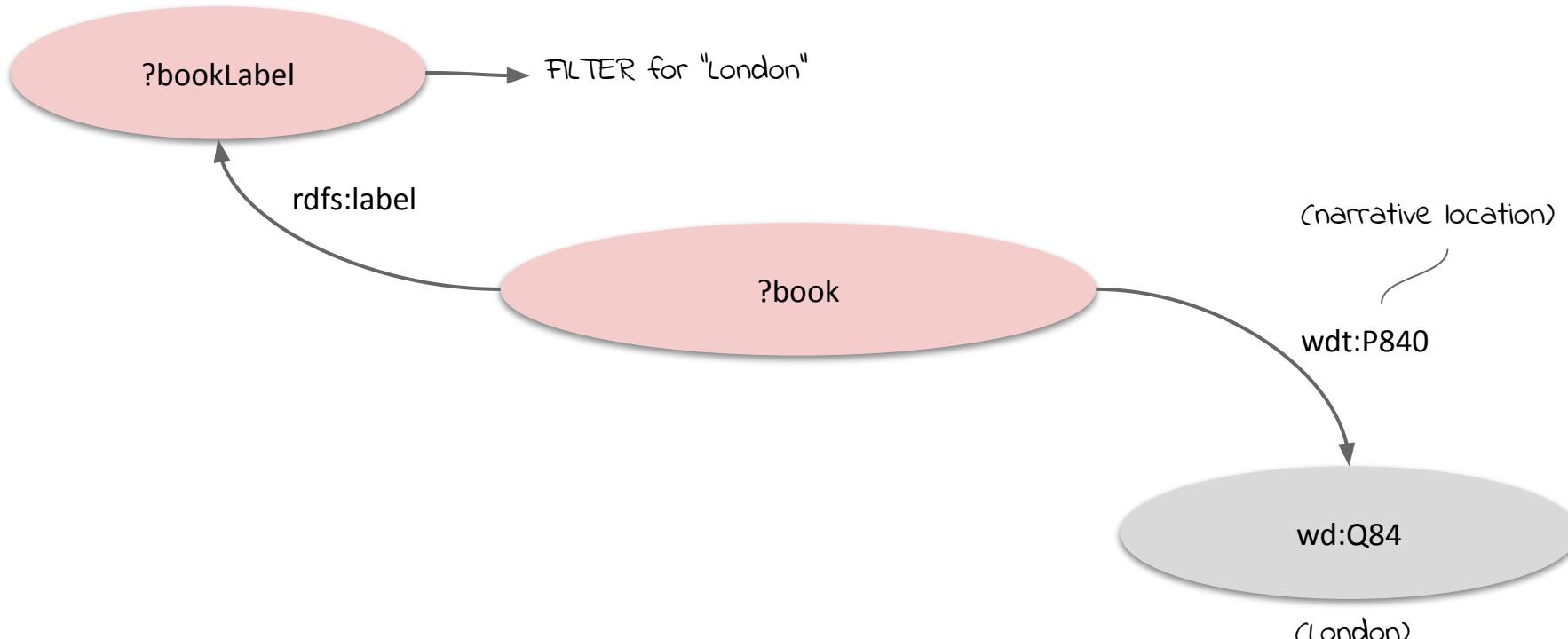
Books displayed on the timeline:

- 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)
- Hal Lindsey: The Late, Great Planet Earth (1 January 1970)
- Robert Silverberg: Downward to the Earth (1 January 1970)
- Ken Follett: The Pillars of the Earth (1 January 1989)
- David Brin: Earth (1 January 1990)
- Arthur Koestler: Scum of the Earth (1 January 1941)
- Richard Woldendorp: Down to Earth (1 January 2009)
- Eckhart Tolle: A New Earth (1 January 2005)
- Arthur C. Clarke: Tales from Planet Earth (1 January 1990)
- Jack Vance: Ecce and Old Earth (1 January 1991)
- Tim Winton: Down to Earth (1 January 2008)

query SPARQL endpoint

SPARQL Alternative Results via UNION

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



SPARQL Alternative Results via 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
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



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

*logical
disjunction*

[query SPARQL endpoint](#)

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

Wikidata Query Service Examples Help More tools

```

1 PREFIX wd: <http://www.wikidata.org/entity/>
2 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
3 PREFIX wikibase: <http://wikiba.se/ontology#>
4 PREFIX bd: <http://www.bigdata.com/rdf#>
5
6 SELECT ?authorLabel ?bookLabel ?book ?date
7 WHERE {
8   ?book wdt:P31 wd:Q47461344 . # instance of (P31) written work (047461344)
9   ?book wdt:P50 ?author . # author (P50)
10  ?book wdt:P577 ?date . # publication date (P577)
11  {
12    FILTER regex (?bookLabel,"London","i") .
13  }
14  UNION
15  {
16    ?book wdt:P840 wd:Q84 # narrative location (P840) London (Q84)
17  }
18 SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
19 } ORDER BY ?date
20

```

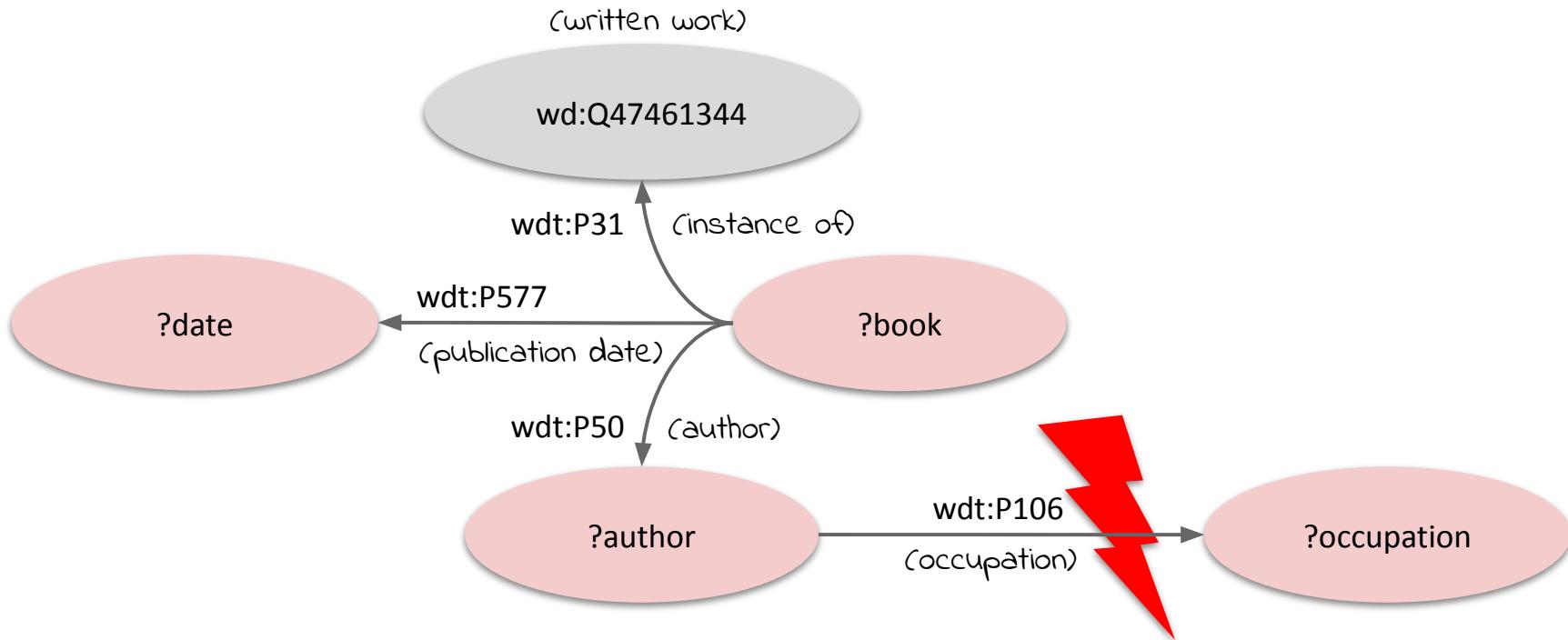
232 results in 6 ms ↻ Code ↻ Search

| authorLabel | bookLabel | book | date |
|--------------------|------------------------------|---------------|----------------|
| Girolamo Graziani | Il Cromuele | Q.wd:Q3792648 | 1 January 1671 |
| Daniel Defoe | A Journal of the Plague Year | Q.wd:Q1215399 | 1 January 1722 |
| Frances Burney | Cecilia | Q.wd:Q3233990 | 1 July 1782 |
| Alexandre Dumas | The Three Musketeers | Q.wd:Q140527 | 1 January 1844 |
| Charles Dickens | A Tale of Two Cities | Q.wd:Q308918 | 1 January 1859 |
| Wilkie Collins | The Law and the Lady | Q.wd:Q3480742 | 1 January 1875 |
| Arthur Conan Doyle | A Study in Scarlet | Q.wd:Q223131 | 1 January 1887 |
| George Gissing | The Nether World | Q.wd:Q7753436 | 1 January 1889 |
| Oscar Wilde | The Picture of Dorian Gray | Q.wd:Q82464 | 1 January 1890 |

[query SPARQL endpoint](#)

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** ?



WIKIDATA

SPARQL 1.1 offers several variants for negation:

- FILTER NOT EXISTS**
- MINUS**
- !BOUND()**

```
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

[query SPARQL endpoint](#)

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

Wikidata Query Service Examples Help More tools

```

1 SELECT ?authorLabel ?bookLabel ?date
2 WHERE {
3   ?book wdt:P31 wd:Q47461344 . # instance of (P31) written work (Q47461344)
4   ?book wdt:P50 ?author .       # author (P50)
5   FILTER NOT EXISTS {?author wdt:P106 ?occupation}
6   ?book wdt:P577 ?date .       # publication date (P577)
7   SERVICE wikibase:label
8   { bd:serviceParam wikibase:language "en, de, es, it" }
9 }
10

```

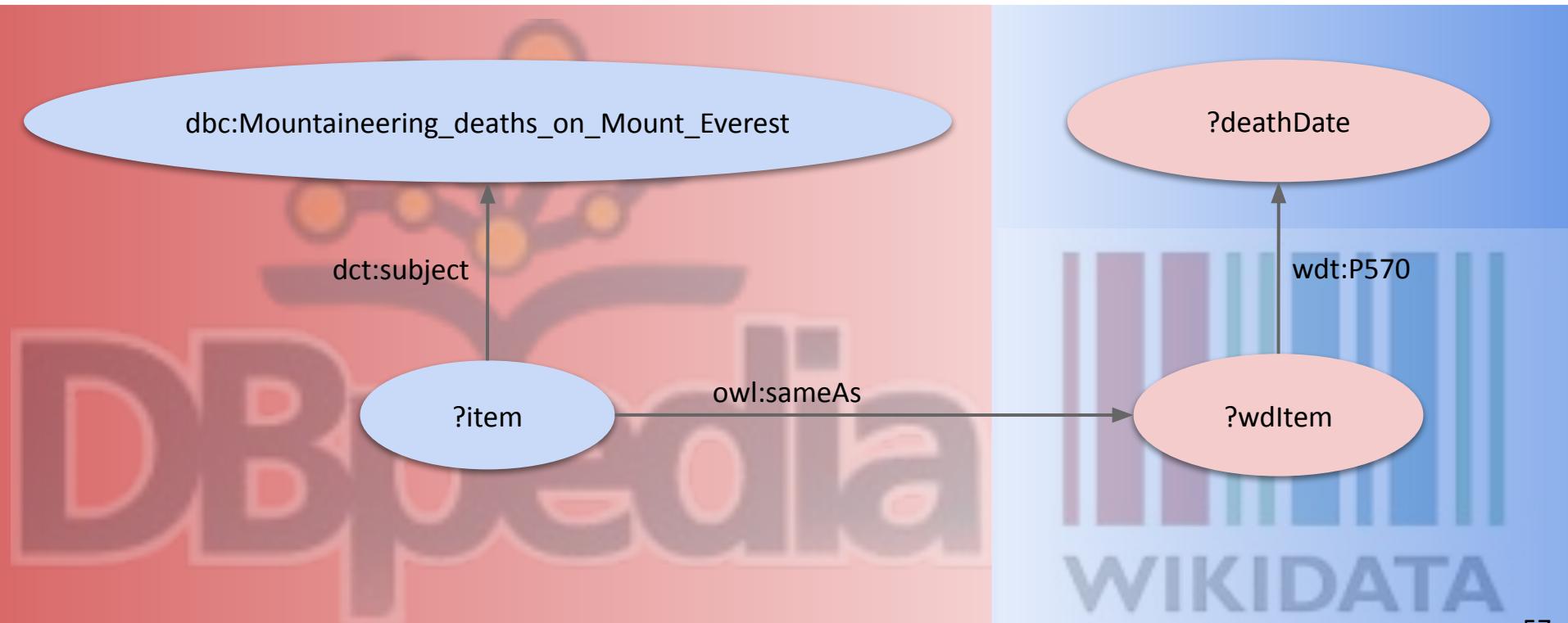
904 results in 4059 ms [Code](#) [Download](#) [Link](#)

| authorLabel | bookLabel | date |
|------------------------------|---|----------------|
| Fruttero & Lucentini | The Sunday Woman | 1 January 1972 |
| Monaldi & Sorti | Imprimatur | 1 January 2002 |
| Sjöwall and Wahlöö | The Laughing Policeman | 1 January 1968 |
| Sjöwall and Wahlöö | The Abominable Man | 1 January 1971 |
| Project Management Institute | A Guide to the Project Management Body of Knowledge | 1 January 2013 |
| Sjöwall and Wahlöö | The Terrorists | 1 January 1975 |
| Sjöwall and Wahlöö | The Man Who Went Up in Smoke | 1 January 1966 |
| Sjöwall and Wahlöö | The Man on the Balcony | 1 January 1967 |
| Sjöwall and Wahlöö | Cop Killer | 1 January 1973 |
| Sjöwall and Wahlöö | Roseanna | 1 January 1965 |
| Fruttero & Lucentini | A che punto è la notte | 1 January 1979 |

[query SPARQL endpoint](#)

SPARQL Federated Query

- 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#>

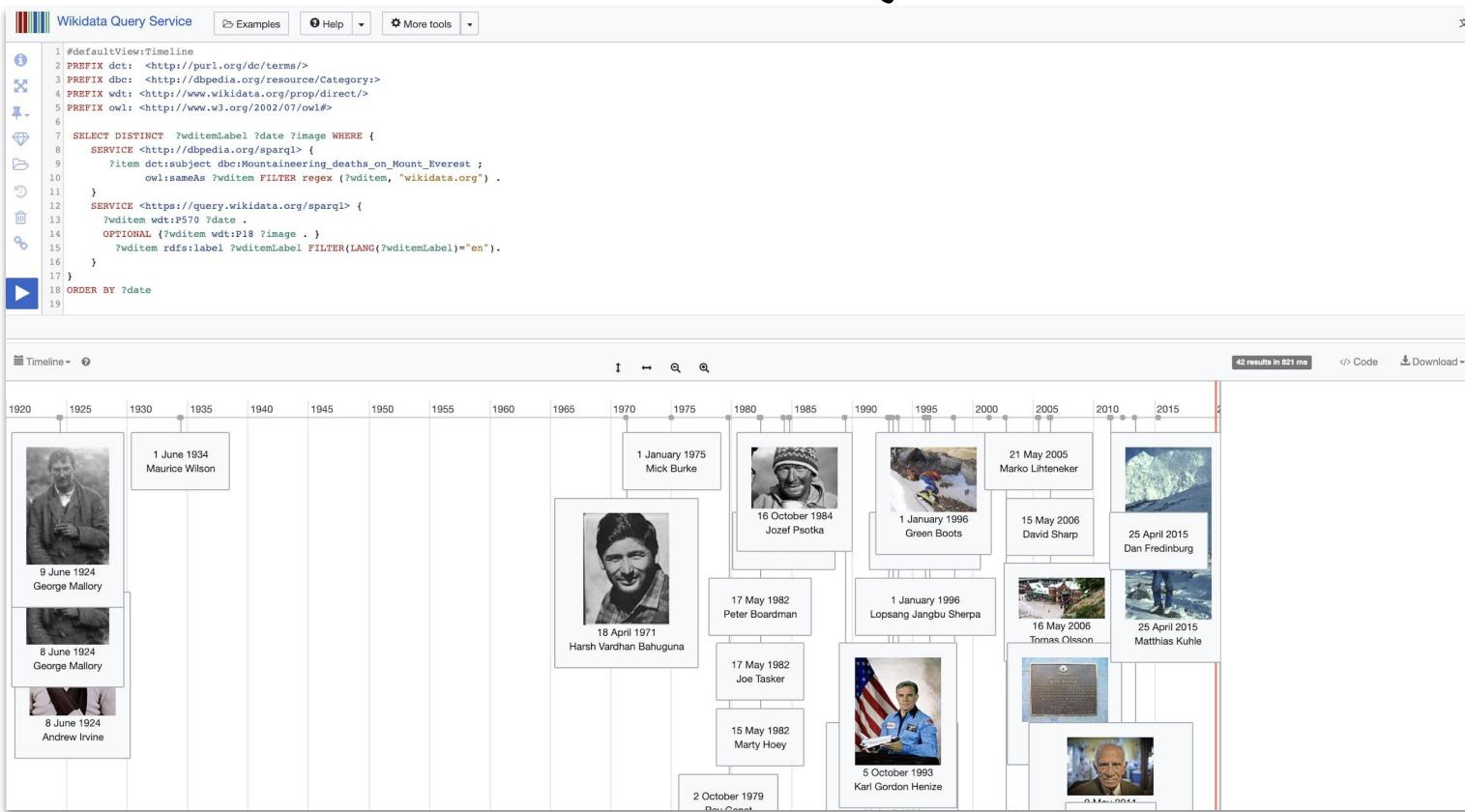
SELECT DISTINCT ?wditemLabel ?date WHERE {
  SERVICE <http://dbpedia.org/sparql> {
    ?item dct:subject dbc:Mountaineering_deaths_on_Mount_Everest ;
           owl:sameAs ?wditem FILTER regex (?wditem, "wikidata.org") .
  }
  SERVICE <https://query.wikidata.org/sparql>{
    ?wditem wdt:P570 ?date .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



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

[query SPARQL endpoint](#)

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


[query SPARQL endpoint](#)

SPARQL Variable Assignment

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



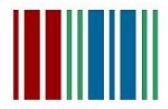
```

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



[query SPARQL endpoint](#)

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

Wikidata Query Service Examples Help More tools English

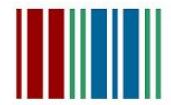
```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX wd: <http://www.wikidata.org/entity/>
3 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
4 PREFIX wikibase: <http://wikiba.se/ontology#>
5 PREFIX bd: <http://www.bigdata.com/rdf#>
6
7 SELECT ?authorLabel ?bookLabel ?book ?author ?year
8 WHERE {
9   ?author wdt:P106 wd:Q36180 ;
10   wdt:P800 ?book .
11   ?book wdt:P577 ?date .
12   BIND (YEAR(?date) AS ?year) FILTER (BOUND(?year))
13   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
14 } ORDER BY ?year
15

```

9030 results in 13210 ms </> Code Download Link Search

| authorLabel | bookLabel | book | author | year |
|-------------------------|----------------------------------|----------------|--------------|------|
| Hesiod | Theogony | Q wd:Q156498 | Q wd:Q44233 | -700 |
| Antimachus of Teos | Epigoni (epic) | Q wd:Q2067424 | Q wd:Q577773 | -600 |
| Euclid | Elements | Q wd:Q172891 | Q wd:Q8747 | -300 |
| Cato the Elder | De Agri Cultura | Q wd:Q1180565 | Q wd:Q180081 | -160 |
| Cicero | De re publica | Q wd:Q656161 | Q wd:Q1541 | -52 |
| Cicero | De Officiis | Q wd:Q1180721 | Q wd:Q1541 | -43 |
| Sappho | Ode to Aphrodite | Q wd:Q21070481 | Q wd:Q17892 | -5 |
| Titus Livius | Ab urbe condita libri | Q wd:Q1155892 | Q wd:Q2039 | 10 |
| Seneca | De Vita Beata | Q wd:Q1180753 | Q wd:Q2054 | 58 |
| Pliny the Elder | Natural History | Q wd:Q442 | Q wd:Q82778 | 74 |
| Ovidius Cornelius Rufus | Historiae of Alexander the Great | Q wd:Q27960221 | Q wd:Q5050 | 100 |


[query SPARQL endpoint](#)

SPARQL Aggregate Functions

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



aggregate
functions

```

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" }
}
  
```

- 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

[query SPARQL endpoint](#)

SPARQL Aggregate Functions

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

Wikidata Query Service Examples Help More tools

English

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX wd: <http://www.wikidata.org/entity/>
3 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
4 PREFIX wikibase: <http://wikiba.se/ontology#>
5 PREFIX bd: <http://www.bigdata.com/rdf#>
6
7 SELECT (COUNT(?book) AS ?bookcount) (COUNT(DISTINCT(?author)) AS ?authorcount)
8 WHERE {
9   ?author wdt:P106 wd:Q36180 ;
10   wdt:P800 ?book .
11   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
12 }
13
  
```



1 result in 1827 ms /> Code Download Link

| bookcount | authorcount |
|-----------|-------------|
| 18077 | 9489 |



WIKIDATA
query SPARQL endpoint

SPARQL Aggregate Functions

- Example: which author wrote 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 ?authorLabel (COUNT(?book) AS ?bookcount)
WHERE {
  ?author wdt:P106 wd:Q36180 ;
          wdt:P800 ?book .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} GROUP BY ?authorLabel
ORDER BY DESC (?bookcount)
  
```

aggregate function

- The solution can be divided into groups via **GROUP BY**.
- The aggregate function is then calculated for each group.


[query SPARQL endpoint](#)

- Example: which author wrote how many notable works?

Wikidata Query Service Examples Help More tools English

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX wd: <http://www.wikidata.org/entity/>
3 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
4 PREFIX wikibase: <http://wikiba.se/ontology#>
5 PREFIX bd: <http://www.bigdata.com/rdf#>
6
7 SELECT ?authorLabel (COUNT(?book) AS ?bookcount)
8 WHERE {
9   ?author wdt:P106 wd:Q36180 ;
10   wdt:P800 ?book .
11   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
12 } GROUP BY ?authorLabel
13 ORDER BY DESC (?bookcount)
14

```



9480 results in 12357 ms Code Download Link 

| authorLabel | bookcount |
|-----------------------|-----------|
| Thomas Mann | 57 |
| Enyd Blyton | 55 |
| Woody Allen | 49 |
| Marion Zimmer Bradley | 46 |
| Elmore Leonard | 46 |
| Stephen King | 36 |
| Kenji Miyazawa | 36 |
| Karel Čapek | 36 |

 Search 

[query SPARQL endpoint](#)



WIKIDATA

SPARQL Aggregate Functions

- 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

More SPARQL

- More SPARQL query functions
 - **ASK** - Check whether there is at least one result
 - **CONSTRUCT** - construct an RDF graph from a template
 - **DESCRIBE** - return all facts (RDF triples) for resources
- More sophisticated SELECT queries with
 - Subqueries
 - Property paths
- SPARQL UPDATE
 - **INSERT / DELETE** RDF triples
 - **CREATE / DROP / COPY / MOVE** RDF graph
- SPARQL RDF(S)/OWL entailment

- 3.1 Knowledge Representations and Ontologies
- 3.2 Semantic Web and the Web of Data
- 3.3 Linked Data Principles
- 3.4 How to identify Things - URIs
- 3.5 Resource Description Framework (RDF) as simple Data Model
- 3.6 Creating new Models with RDFS
- 3.7 Knowledge Graphs
- 3.8 Querying Knowledge Graphs with SPARQL
- 3.9 More Expressivity with Web Ontology Language (OWL)**
- 3.10 Knowledge Graph Programming

3. Linked Data Engineering - 3

Bibliography

- Ch. Bizer et al:
[DBpedia - A crystallization point for the Web of Data](#), Web Semantics: Science, Services and Agents on the World Wide Web. 7 (3): 154–165.
- D. Vrandečić, M. Krötzsch:
[Wikidata: A Free Collaborative Knowledge Base](#). Communications of the ACM. ACM. 2014.
- S. Harris, A. Seabourne (eds.)
SPARQL 1.1 Query Language, W3C Recommendation, 21 March 2013,
<https://www.w3.org/TR/sparql11-query/>.
- A. Hogan,
[The Web of Data](#), Springer, 2020. (*available via KIT network*)

3. Linked Data Engineering - 3

Syllabus Questions

- What are the **benefits and drawbacks** when comparing DBpedia and Wikidata?
- What is **SPARQL**?
- What is a **Graph Pattern**?
- Explain the basic principle of how a SPARQL query is carried out.
- What is the general **SPARQL query format**?
- Explain the communication with a SPARQL endpoint via the **SPARQL Protocol**.
- How can **Regular Expressions** be used in SPARQL queries?
- What is the use of a SPARQL **optional** query?
- How can **conjunctive** and **disjunctive queries** be expressed in SPARQL?
- What is a **federated query**?