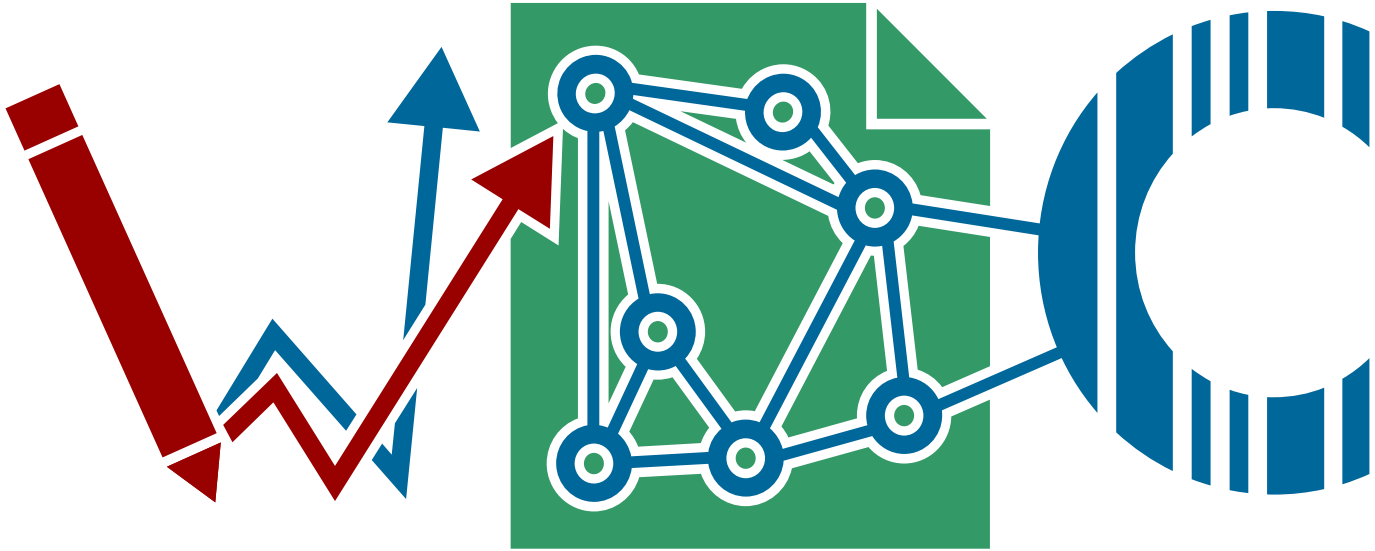


WIKI DATA CON

Wikidata & languages 2019

Wikidata Query Service Tutorial

Questions may be asked at any time 😊

WikidataCon 2019, 25-26 October, Berlin

Audience

Who here has already…

- looked at results of a query?
- edited a query using the query helper?
- edited a query in SPARQL?

First query

- We’re in room “Darwin”, probably named after Charles Darwin
- What else is named after him?
- Darwin, Australia, for example

First query

Darwin (Q11568)

capital city of the Northern Territory, Australia edit

▸ In more languages

Statements

instance of	city ▾ 0 references + add reference	edit
	+ add value	
named after	Charles Darwin ▾ 0 references + add reference	edit
	+ add value	

First query

Darwin, Australia (Q11568) is named after (P138) Charles Darwin (Q1035).

First query

For the query service, we write this as:

```
wd:Q11568 wdt:P138 wd:Q1035.
```

First query

To get a full query, we still need to add something:

```
SELECT * WHERE {
  wd:Q11568 wdt:P138 wd:Q1035.
}
```

Try it on query.wikidata.org!

Useful keyboard shortcuts

- Ctrl+Space: autocompletion – type `SEL` and it will suggest
  ```
  SELECT * WHERE {

  }
  ```
- Shift+Tab: automatically indent the current line(s)
- Ctrl+Space: type `wd:Darwin` or `wdt:named after`, then Ctrl+Space, and select the right entity instead of having to remember and type in entity IDs

First query

- Returns 1 empty result
- Means: yes – Darwin, Australia is indeed named after Charles Darwin (otherwise would have returned 0 results)
- Let’s make it more useful

Named after Darwin

We can take the same query, but replace Darwin, Australia with a *variable*:

```
SELECT * WHERE {
  ?item wdt:P138 wd:Q1035.
}
```

Now we get *everything* that's named after Charles Darwin.

Label service

Seeing only the item IDs isn't very useful, so let's add the *label service*:

```
SELECT ?item ?itemLabel ?itemDescription WHERE {
  ?item wdt:P138 wd:Q1035.
  SERVICE wikibase:label { bd:serviceParam wikibase:language
"[AUTO_LANGUAGE],en". }
}
```

Use autocompletion! Type SER, then Ctrl+Space.

Take a step back

What does this mean?

```
SELECT * WHERE {
  wd:Q11568 wdt:P138 wd:Q1035.
}
```

The query service uses the *RDF* data model. In RDF, all information is encoded in *triples*: subject – predicate – object.

RDF

- Normal statements become plain triples: subject (item) – predicate (property) – value (item or other)
- Labels, descriptions and aliases also become other triples, with special predicates such as `rfds:label` or `schema:description`
- Everything else (qualifiers, references, ranks, sitelinks, etc.) is also encoded in further triples – we'll get to that later

SPARQL

This query means:

```
SELECT * WHERE {
  wd:Q11568 wdt:P138 wd:Q1035.
}
```

Check if this triple exists.

SPARQL

This query means:

```
SELECT ?item WHERE {
  ?item wdt:P138 wd:Q1035.
}
```

Find triples where the predicate is `wdt:P138` and the object is `wd:Q1035`, and save the subject as `?item`.

SPARQL

We can also turn this around:

```
SELECT ?item WHERE {
  wd:Q11568 wdt:P138 ?item.
}
```

Find triples where the subject is `wd:Q11568` and the predicate is `wdt:P138`, and save the object as `?item`.

SPARQL

You can have more than one triple in a query:

```
SELECT ?item WHERE {
  ?item wdt:P138 wd:Q1035.
  ?item wdt:P625 ?coordinates.
}
```

Find triples where the predicate is `wdt:P138` and the object is `wd:Q1035`, and whose subject (`?item`) is also the subject of a triple with the predicate `wdt:P625`, and save the object of that in the variable `?coordinates`.

SPARQL

Find triples where the predicate is `wdt:P138` and the object is `wd:Q1035`, and whose subject (`?item`) is also the subject of a triple with the predicate `wdt:P625`, and save the object of that in the variable `?coordinates`.

Or, in other words: find items that are named after Charles Darwin and also have a coordinate location.

Some more query ideas

- Items named after Jane Austen (“Austen” is the room across the hall)
- What is Wikidata named after?
- Books written by Jane Austen
- Books written by anyone with the family name “Darwin”
- Books in the public domain on the topic of biology written by English-speaking authors who were buried in Westminster Abbey, with title and number of pages

If you get stuck, my solutions are on the next slides (though there’s more than one way to solve most of these).

Query solutions

Items named after Jane Austen:

```
SELECT ?item ?itemLabel ?itemDescription WHERE {
  ?item wdt:P138 wd:Q36322.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

Query solutions

What is Wikidata named after?

```
SELECT ?item ?itemLabel WHERE {
  wd:Q2013 wdt:P138 ?item.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

Side note: in theory, multiple results may be returned in any order (“wiki”, “data” or “data”, “wiki”).

Query solutions

Books written by Jane Austen:

```
SELECT ?item ?itemLabel ?itemDescription WHERE {
  ?item wdt:P50 wd:Q36322.
  ?item wdt:P136 wd:Q8261.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

There are different ways to distinguish between her books and other works; here, I've used "genre": "novel" statements.

Query solutions

Books written by Jane Austen:

```
SELECT ?item ?itemLabel ?itemDescription WHERE {
  ?item wdt:P50 wd:Q36322;
        wdt:P136 wd:Q8261.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

Instead of repeating `?item`, we can also finish the first triple with a semicolon instead of a full stop.

Query solutions

Books written by anyone with the family name “Darwin”:

```
SELECT ?item ?itemLabel ?authorLabel WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

(This completely ignores the problem of books vs. other works.)

Query solutions

Books written by anyone with the family name “Darwin”:

```
SELECT ?item ?itemLabel WHERE {
  ?item wdt:P50 [
    wdt:P734 wd:Q1166891
  ].
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

If we don’t care who the author is, we can hide them inside a [ ] block.

Query solutions

Books in the public domain on the topic of biology written by English-speaking authors who were buried in Westminster Abbey, with title and number of pages:

```
SELECT ?item ?title ?pages WHERE {
  ?item wdt:P6216 wd:Q19652.
  ?item wdt:P921 wd:Q420.
  ?item wdt:P50 ?author.
  ?author wdt:P1412 wd:Q1860.
  ?author wdt:P119 wd:Q5933.
  ?item wdt:P1476 ?title.
  ?item wdt:P1104 ?pages.
}
```

Try it!

Query solutions

Books in the public domain on the topic of biology written by English-speaking authors who were buried in Westminster Abbey, with title and number of pages:

Try it!

```
SELECT ?item ?title ?pages WHERE {
  ?item wdt:P6216 wd:Q19652;
        wdt:P921 wd:Q420;
        wdt:P50 [
          wdt:P1412 wd:Q1860;
          wdt:P119 wd:Q5933
        ];
        wdt:P1476 ?title;
        wdt:P1104 ?pages.
}
```

Optional query parts

Coming back to the items named after Jane Austen:

```
SELECT ?item ?itemLabel ?itemDescription WHERE {
  ?item wdt:P138 wd:Q36322.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

What if we also want to see images for them?

Optional query parts

If we just add another triple for the image –

```
SELECT ?item ?itemLabel ?itemDescription ?image WHERE {
  ?item wdt:P138 wd:Q36322.
  ?item wdt:P18 ?image.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

– then we'll get less results, because this excludes items that don't have an image.

Optional query parts

Instead, we want to make that triple *optional*:

```
SELECT ?item ?itemLabel ?itemDescription ?image WHERE {
  ?item wdt:P138 wd:Q36322.
  OPTIONAL { ?item wdt:P18 ?image. }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it! If there is an image, show it, but otherwise don't discard the result.

Ordering and grouping

Coming back to the books written by anyone with the family name “Darwin”:

```
SELECT ?item ?itemLabel ?authorLabel WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Suppose we want to find out how many books each author wrote.

Ordering and grouping

We can sort the books by author, so that books by the same author go together:

```
SELECT ?authorLabel ?item ?itemLabel WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
ORDER BY ?authorLabel
```

Try it! Then we can count them by hand.

Ordering and grouping

Instead of *ordering*, we can also *group*:

```
SELECT ?authorLabel (COUNT(?item) AS ?count) WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
GROUP BY ?authorLabel
```

Try it! All the results with the same `?authorLabel` form a *group*, which we can *aggregate* with functions like `COUNT()`, `SUM()`, `MAX()`, ...

Ordering and grouping

We can also order again after grouping:

```
SELECT ?authorLabel (COUNT(?item) AS ?count) WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
GROUP BY ?authorLabel
ORDER BY DESC(?count)
```

Try it!

Filtering

Books written by anyone with the family name “Darwin”, except Charles Darwin:

```
SELECT ?item ?itemLabel ?authorLabel WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  FILTER(?author != wd:Q1035)
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

Filtering

Books written by anyone with the family name “Darwin”, except Charles Darwin:

```
SELECT ?item ?itemLabel ?authorLabel WHERE {
  ?item wdt:P50 ?author.
  ?author wdt:P734 wd:Q1166891.
  MINUS { ?item wdt:P50 wd:Q1035. }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Try it!

FILTER vs. MINUS

`FILTER(…)` and `MINUS { … }` both let you restrict the query results in different ways. `FILTER` uses expressions based on values you already have (e. g. equals, not equals, less than), while `MINUS` uses additional graph patterns.

The two previous example queries actually return different results: strictly speaking, the first finds books that have any author who is surnamed Darwin and not Charles Darwin, while the second finds books that have any author who is surnamed Darwin and that were not written by Charles Darwin. If a book was written by Charles Darwin and another Darwin, the first query will return it, but the second one won't.

(There is also `FILTER NOT EXISTS { … }`, similar to `MINUS { … }` but subtly different and usually slower. I don't recommend it, but you can try it.)

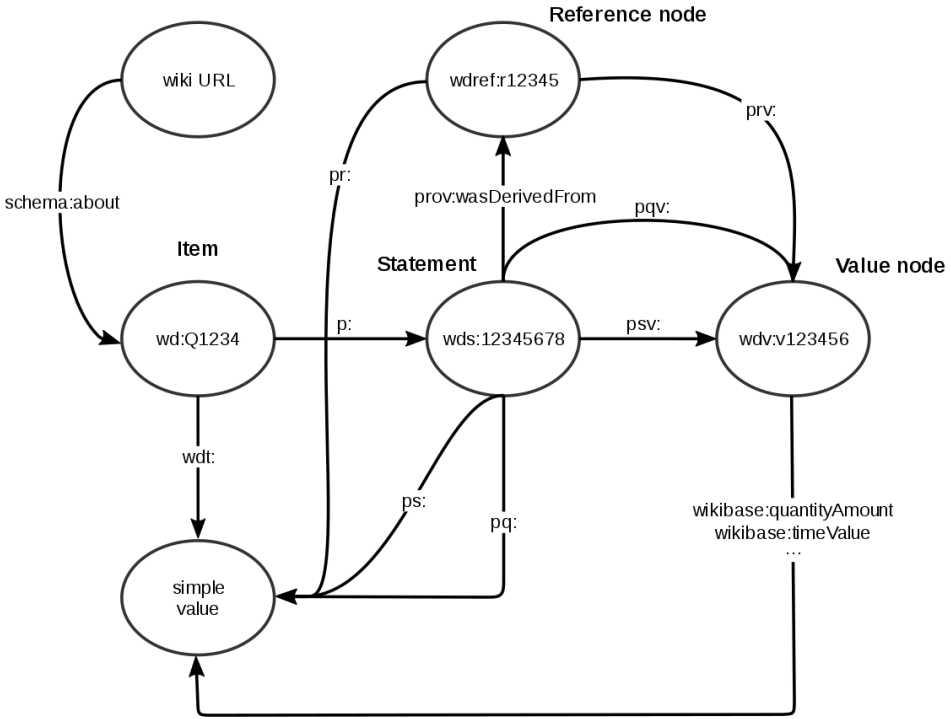
Full statements

So far, we saw statements exported as a single triple:

```
wd:Q11568 wdt:P138 wd:Q1035.
```

That only gives you the main value – what about qualifiers, references, ranks?
Those are exported as additional triples, with some auxiliary nodes in between.

Full statements



Full statements

```
wd:Q39 wdt:P36 wd:Q70;
       p:P36 [
         ps:P36 wd:Q70; # main value
         pq:P580 "1848-11-28T00:00:00Z"^^xsd:dateTime;
         pq:P459 wd:Q712144; # qualifier
         prov:wasDerivedFrom [ # reference
           pr:P854 <https://www.bk.admin.ch/…>;
           pr:P1476 "Bundesstadtstatus Stadt Bern"@de;
           pr:P813 "2018-10-16T00:00:00Z"^^xsd:dateTime
         ];
         wikibase:rank wikibase:PreferredRank # rank
       ].
```

Full statements

`wdt:` triples only give you “truthy” statements: best rank without being deprecated. `p:` triples give you access to all statements.

Value nodes

Similarly, for some data types the usual prefixes only gives you a simple value, and you can get the full value with all the details via another set of prefixes.

This is mostly important for time values (full value has the precision) and quantity values (full value has the bounds and unit).

Value nodes

```
wd:Q39 p:P36 [
  pq:P580 "1848-11-28T00:00:00Z"^^xsd:dateTime;
  pqv:P580 [ # time value
    wikibase:timeValue "1848-11-28T00:00:00Z"^^xsd:dateTime;
    wikibase:timePrecision 11;
    wikibase:timeZone 0;
    wikibase:timeCalendarModel wd:Q1985727
  ]
].
```

More information

- Request a query (WD:RAQ)
- SPARQL tutorial
- Query portal (Help:SPARQL)