WDQS and SPARQL – Advanced Workshop

WikidataCon 2017

Lucas Werkmeister (@WikidataFacts)

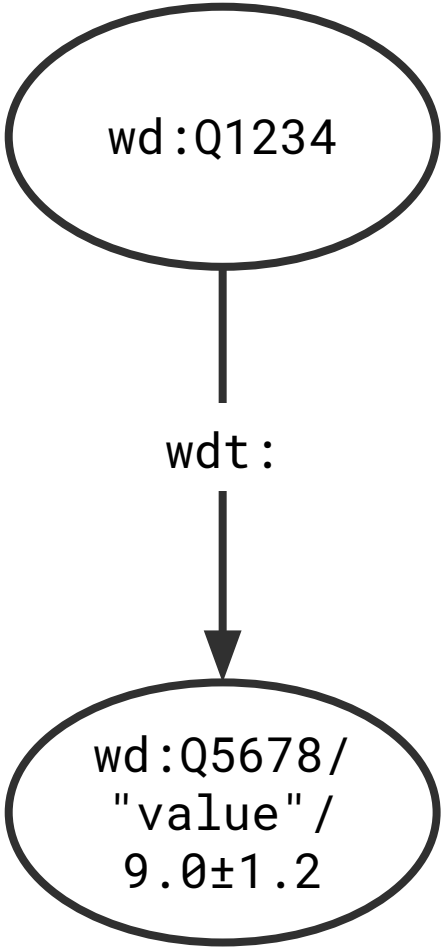
How this works

- ask questions at any time
- interrupt me if I’m going too fast
- try to follow along on your own laptops

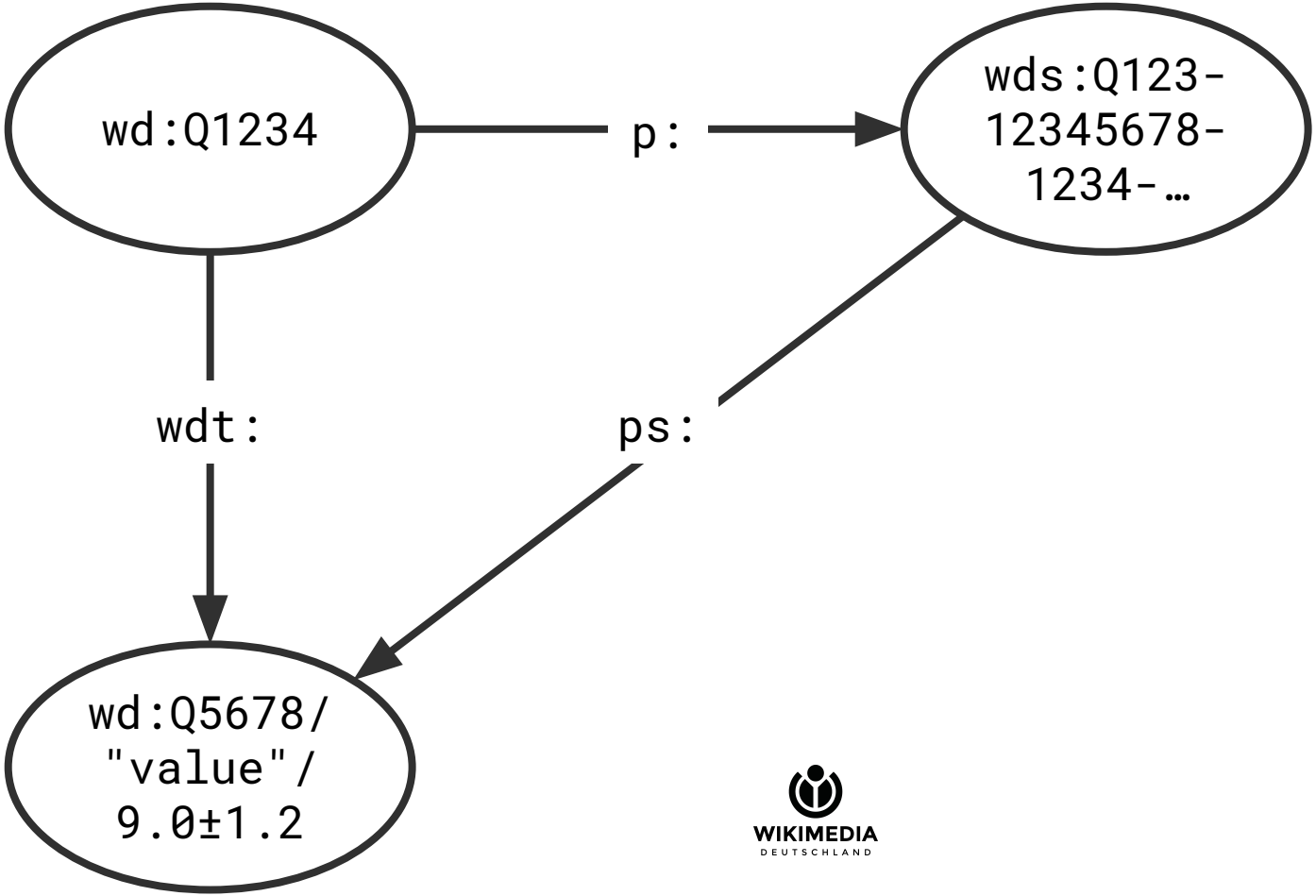
WIKIMEDIA
DEUTSCHLAND

RDF

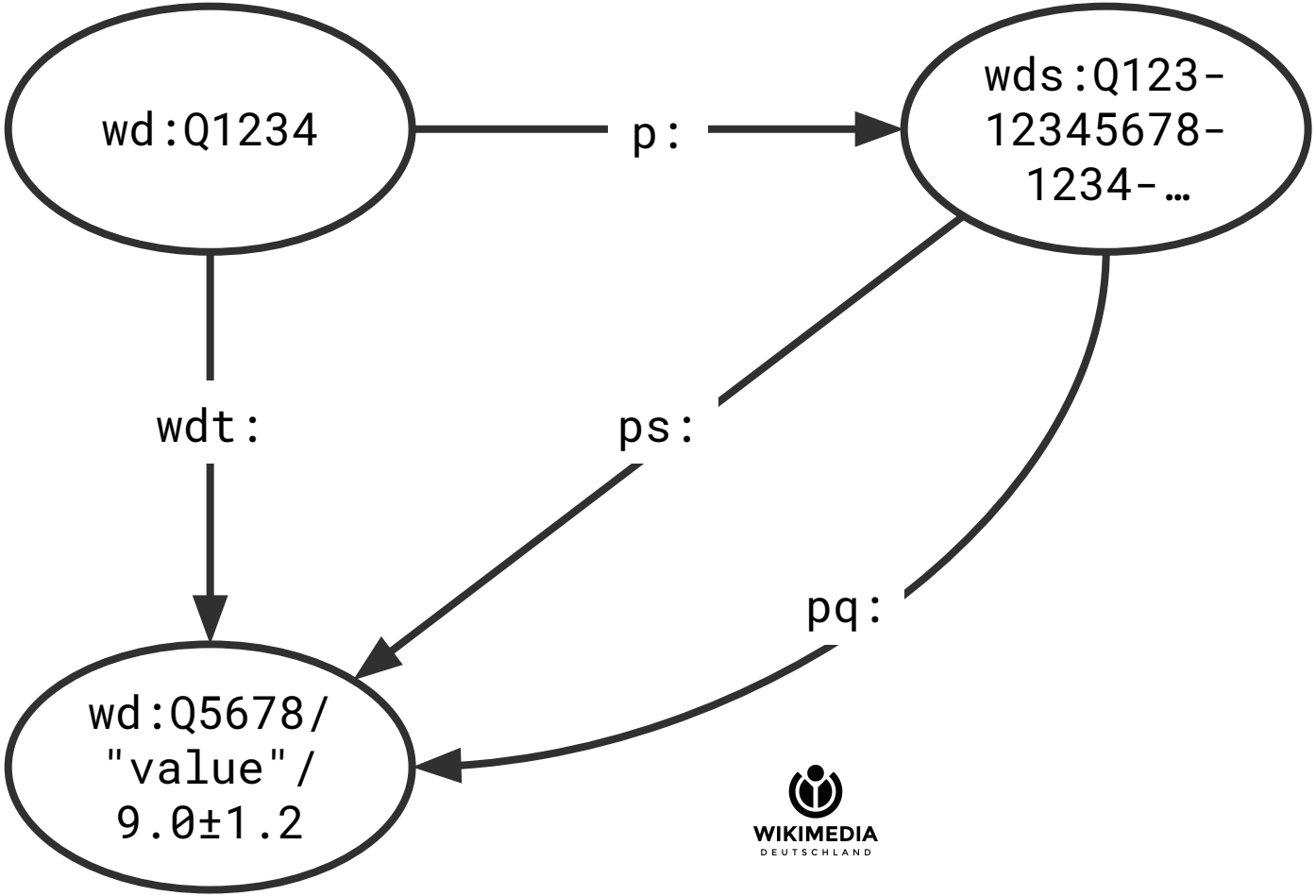
- WDQS and SPARQL operate on RDF data
- RDF knows only triples: subject, predicate, object
- other details of Wikidata (e. g. qualifiers, references) are mapped to triples



WIKIMEDIA
DEUTSCHLAND



WIKIMEDIA
DEUTSCHLAND



WIKIMEDIA
DEUTSCHLAND

RDF syntax

```
wd:Q1234 wdt:P12 wd:Q5678.

wd:Q1234 p:P12 wds:Q1234-12345678-1234-….
wds:Q1234-12345678-1234-… ps:P12 wd:Q5678.

wds:Q1234-12345678-1234-… pq:P34 "a qualifier".
```

WIKIMEDIA
DEUTSCHLAND

SPARQL syntax

```
wd:Q1234 wdt:P12 wd:Q5678;
         p:P12 wds:Q1234-12345678-1234-….

wds:Q1234-12345678-1234-… p:P12 wd:Q5678;
                          pq:P34 "a qualifier".
```

WIKIMEDIA
DEUTSCHLAND

SPARQL syntax

```
wd:Q1234 wdt:P12 wd:Q5678;
         p:P12 ?statement.

?statement p:P12 wd:Q5678;
           pq:P34 "a qualifier".
```

WIKIMEDIA
DEUTSCHLAND

SPARQL syntax

```
wd:Q1234 wdt:P12 wd:Q5678;
         p:P12 [
           ps:P12 wd:Q5678;
           pq:P34 "a qualifier"
         ].
```

WIKIMEDIA
DEUTSCHLAND

Population of Berlin

Let’s write a query that returns the population of Berlin at different times (i. e. all *population* statements with the *point in time* qualifier).

WIKIMEDIA
DEUTSCHLAND

Population of Berlin

Current population of Berlin:

```
SELECT ?population WHERE {
  wd:Q64 wdt:P1082 ?population.
}
```

Returns only current population due to truthy triple (`wdt:`).

WIKIMEDIA
DEUTSCHLAND

Population of Berlin

All population statements of Berlin:

```
SELECT ?population WHERE {
  wd:Q64 p:P1082 [
    ps:P1082 ?population
  ].
}
```

WIKIMEDIA
DEUTSCHLAND

Population of Berlin

With point in time:

```
SELECT ?population ?pointInTime WHERE {
  wd:Q64 p:P1082 [
    ps:P1082 ?population;
    pq:P585 ?pointInTime
  ].
}
```

WIKIMEDIA
DEUTSCHLAND

Premiere locations

Another possible qualifier is the *location* of a *first performance*.

WIKIMEDIA
DEUTSCHLAND

Premiere locations

```
SELECT ?workLabel ?locationLabel WHERE {
  ?work p:P1191 ?premiereStatement.
  ?premiereStatement pq:P276 ?location.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

WIKIMEDIA
DEUTSCHLAND

Premiere locations

Instead of just listing all the locations, let’s count how many works were first performed at each location.

Premiere locations

```
SELECT ?locLabel ?workLabel WHERE {
  ?work p:P1191 ?premiereStatement.
  ?premiereStatement pq:P276 ?loc.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
ORDER BY ?locLabel
```

WIKIMEDIA
DEUTSCHLAND

Premiere locations

```
SELECT ?locLabel (COUNT(*) AS ?count) WHERE {
  ?work p:P1191 ?premiereStatement.
  ?premiereStatement pq:P276 ?loc.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
GROUP BY ?locLabel
ORDER BY DESC(?count)
```

WIKIMEDIA
DEUTSCHLAND

Grouping

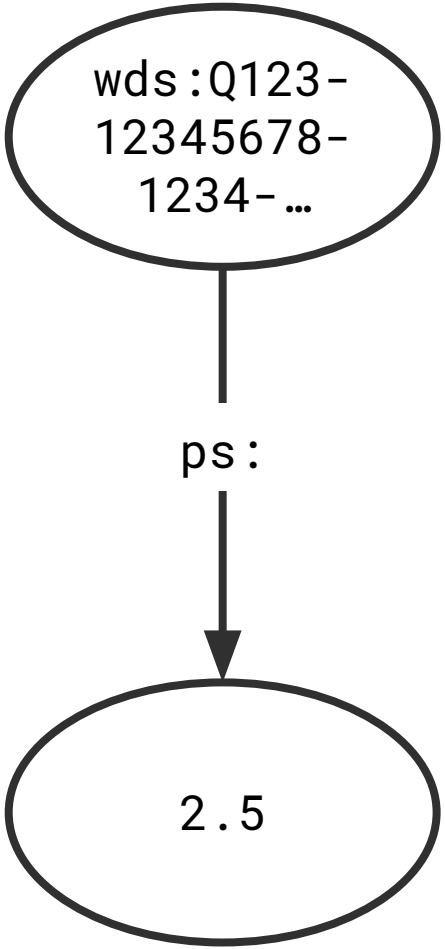
- `GROUP BY` every variable that you want to select per group (e. g. `?location ?locationLabel`)
- for everything else you `SELECT`, add an aggregate function (e. g. `COUNT`, `MIN`/`MAX`) and select it under a new name with `(FUNCTION(?variable) AS ?newName)`

WIKIMEDIA
DEUTSCHLAND

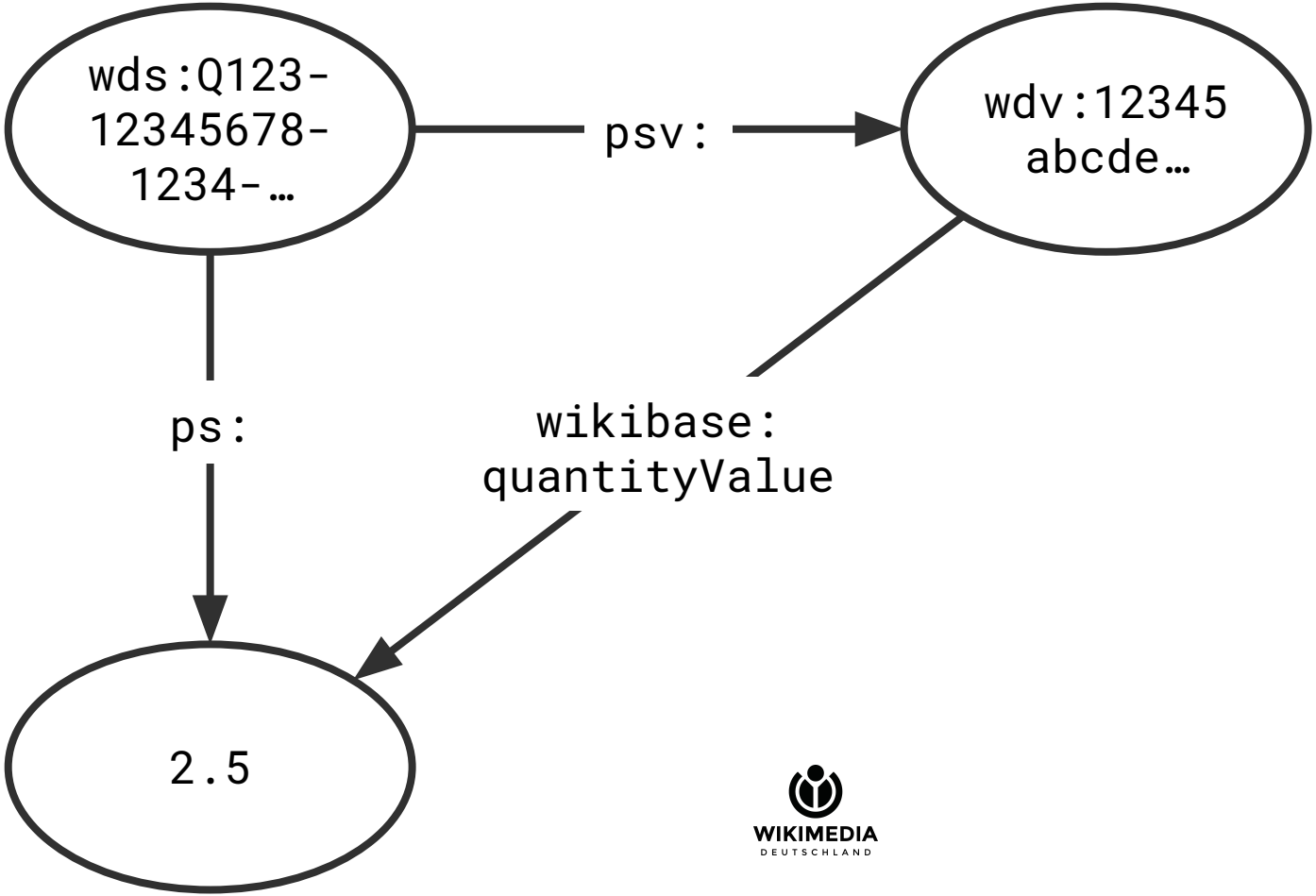
Quantities and units

- the `wdt:` or `ps:` value contains just the numeric part of the original quantity (e. g. `2.5` for `2.5 minutes`)
- `psv:` points to a *value node*, which contains all the parts (value, unit, uncertainty)
- `psn:` points to a similar value node, converted to standard units (e. g. `150` [seconds] for `2.5 minutes`)

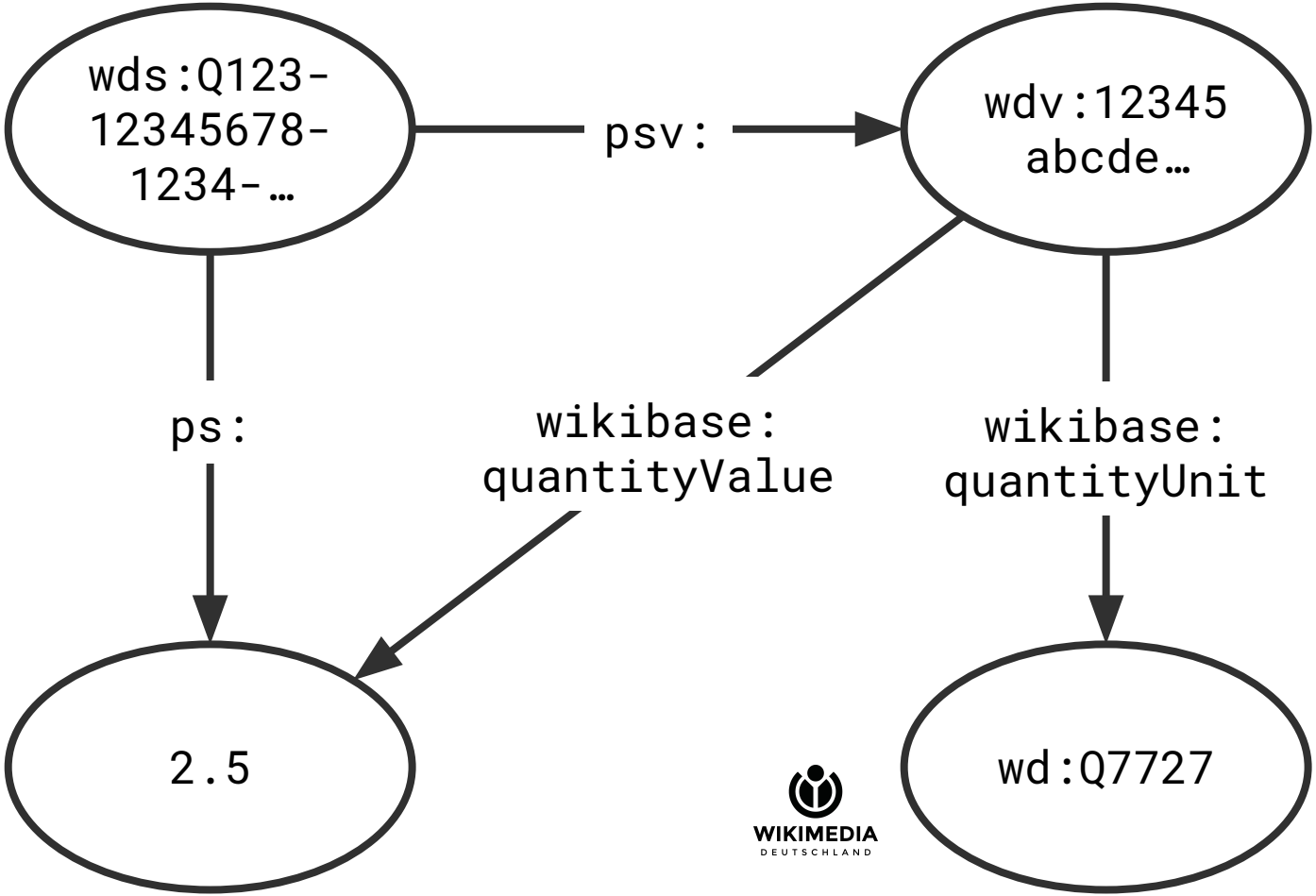
WIKIMEDIA
DEUTSCHLAND



WIKIMEDIA
DEUTSCHLAND



WIKIMEDIA
DEUTSCHLAND


wds:Q123-
12345678-
1234-…
psv:
wdv:12345
abcde…
ps:
wikibase:
quantityValue
wikibase:
quantityUnit
2.5
wd:Q7727
WIKIMEDIA
DEUTSCHLAND

Larger queries

- Named Subqueries can be useful
- extract small, auxiliary parts of a query (example)
- have a tree of subqueries (example)

Larger Queries

```
SELECT … WITH {
  SELECT … WHERE {
    …
  }
} AS %foo WHERE {
  INCLUDE %foo.
  …
}
```

WIKIMEDIA
DEUTSCHLAND

Larger Queries

- create auxiliary variables with `BIND(?c/?a AS ?var)`
- for example:

```
BIND(?count/?total AS ?ratio)
BIND(?ratio * 100 AS ?percent)
BIND(CONCAT(STR(?percent), "%") AS ?pctStr)
```

WIKIMEDIA
DEUTSCHLAND

Query Optimization

Query Optimization

Query Optimization (symbolic picture)
Vassil (https://commons.wikimedia.org/wiki/File:Watain_27_03_2014_08.jpg), „Watain 27 03 2014 08“, https://creativecommons.org/licenses/by-sa/3.0/legalcode

Query Optimization

- matching fixed values is super cheap
- matching ranges is also cheap

```
FILTER("2017-01-01"^^xsd:dateTime <= ?date &&
  ?date < "2018-01-01"^^xsd:dateTime)
```

is much cheaper than

```
FILTER(2017 <= YEAR(?date) &&
  YEAR(?date) < 2018)
```

WIKIMEDIA
DEUTSCHLAND

Query Optimization

- property paths (e. g. `wdt:P131*`, `wdt:P31/wdt:P279*`) can be expensive
- try to match some fixed value first so the property paths don’t have to run on as many potential results

WIKIMEDIA
DEUTSCHLAND

Query Optimization

- remove the label service
- if that helped, try to add it back, with a subquery
  - inner `SELECT` is the original one without `?*Label`
  - outer `SELECT` is the original one's variables (no expressions)
  - `GROUP BY`, `HAVING`, `LIMIT`, `OFFSET` on the subquery
  - `ORDER BY` on the outer query (if `LIMIT` present: both)

WIKIMEDIA
DEUTSCHLAND

Query Optimization

```
SELECT ?x ?xLabel ?count WHERE {
  {
    SELECT ?x (COUNT(*) AS ?count) WHERE {
      # …
    }
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

WIKIMEDIA
DEUTSCHLAND

Query Optimization

- if nothing else helps: optimizer hand-holding
- `hint:Prior hint:runFirst true.`
- `hint:Prior hint:runLast true.`
- these hints apply to the prior *join*, i. e. the dot *between* two triples (e. g. add `runFirst` after the second triple)
- it may be necessary to desugar syntactic constructs like `[ ]` to be able to apply the hint to a join

WIKIMEDIA
DEUTSCHLAND

Query Optimization

- `hint:Prior hint:gearing "forward".`
- `hint:Prior hint:gearing "reverse".`
- In a triple like `?item wdt:P131* wd:Q183`, where `?item` is already bound, start at `?item` and walk `wdt:P131` forward until finding `wd:Q183`, or vice versa?
- usually the optimizer chooses the right gearing, but you can explicitly specify one if necessary

WIKIMEDIA
DEUTSCHLAND

Query Optimization

- last resort: `hint:Query hint:optimizer "none".`
- (this only controls the *join order optimizer*, other optimizations still apply – basically, your query now runs in exactly the order it specifies)

WIKIMEDIA
DEUTSCHLAND

Questions?

WIKIMEDIA
DEUTSCHLAND