Scaling Wikidata Query Service

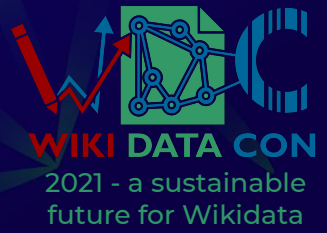
Unlimited access to all the world's knowledge for everyone is hard

Mike Pham
Product Manager, Search
Wikimedia Foundation

Guillaume Lederrey
Engineering Manager, Search
Wikimedia Foundation

Lydia Pintscher
Product Manager, Wikidata
Wikimedia Deutschland

Adam Shorland
Tech Lead, Wikidata
Wikimedia Deutschland



An introduction to:
WIkidata Query Service (WDQS)

The Wikidata Query Service

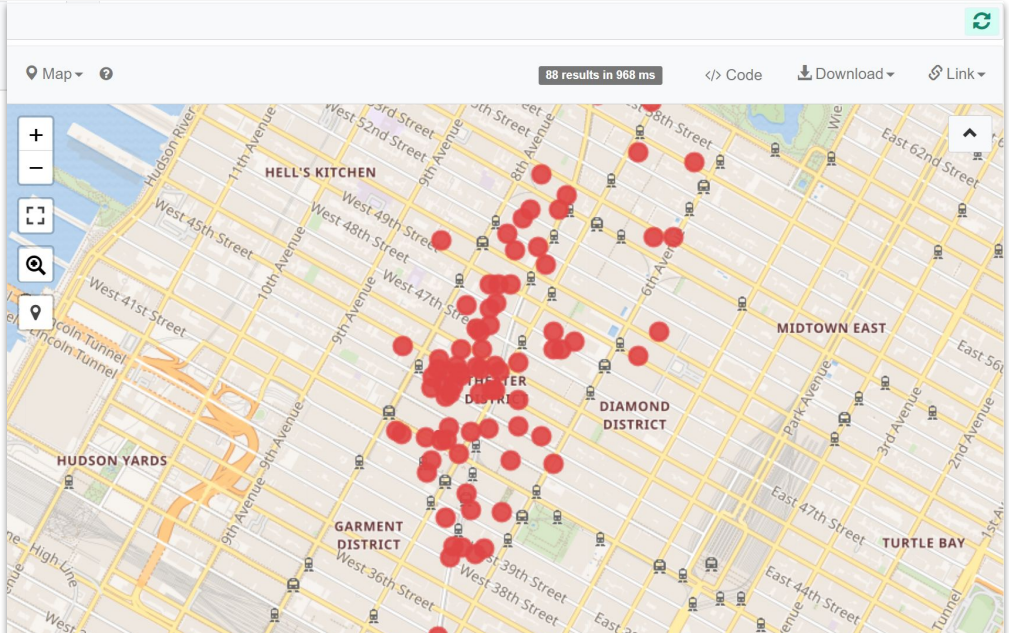
- Wikimedia SPARQL server deployed in 2015 (6 years)
- Enables users and scripts to access Wikidata data, and query entity graph connections, via a SPARQL API and User Interface

Wikidata Query Service | Examples | Query Builder

```
#Map of Broadway venues
#defaultView:Map
# Venues in Broadway
SELECT DISTINCT ?venue ?venueLabel ?coords {
    ?venue wdt:P1217 ?IDBDB .
    wd:Q235065 wdt:P625 ?broadWayLoc .

    SERVICE wikibase:around {
      ?venue wdt:P625 ?coords .
      bd:serviceParam wikibase:center ?broadWayLoc .
      bd:serviceParam wikibase:radius "1.5" .
    }

  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
```

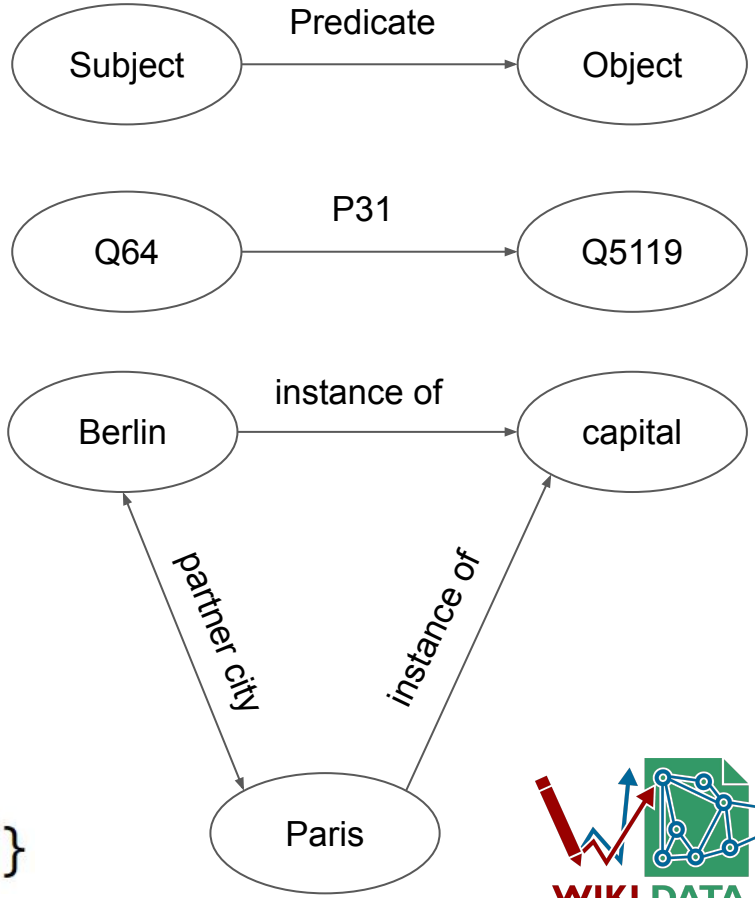


SPARQL, Graphs & Triples

SPARQL: A query language for graph content

Graph: Set of objects in which some object pairs are related

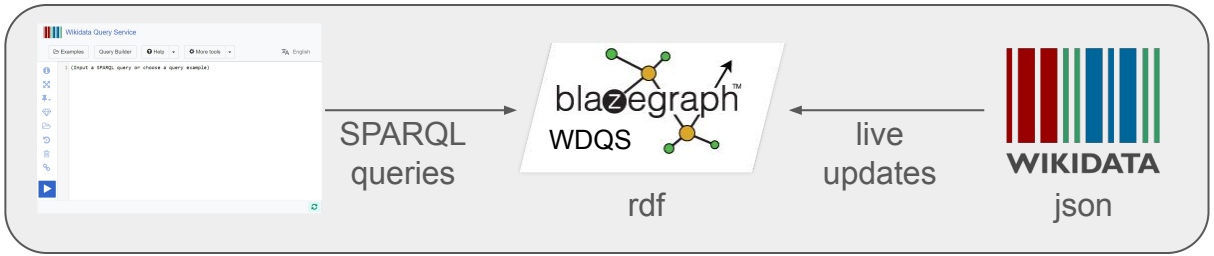
Triple: Set of 3 objects that codified a statement about semantic data




```
SELECT ?item
WHERE { ?item wdt:P31 wd:Q5119 }
```

Technology

- One of the largest public SPARQL endpoint on the internet
- Based on the Blazegraph engine



- 11 public servers (across 2 data centres), 6 internal servers [1]
- Specs similar to Dual Intel(R) Xeon(R) CPU E5-2620 v3, 1600GB raw raided space SSD, 128GB RAM

1 https://wikitech.wikimedia.org/wiki/Wikidata_Query_Service#Hardware

Scale

- 13.2 billion triples [1] from 95 million Wikidata entities [2]
  - (Averages 139 triples per entity, Large entities can have 10s of thousands of triples)
- Regular peaks of 700 Wikidata edits per minute
  - 700 per minute (12 per second) ≅ 20k triple updates per minute (330 per second) [1]
- 25.14 million peak requests per day (290 per second) [3]
  - 10-20% of these generally hit a cache

- So in a single second each backend server would generally peak handling ~58 SPARQL queries , ~12 Wikidata edits which makes up ~330 triple updates

1 https://grafana.wikimedia.org/d/000000489/wikidata-query-service
2 https://grafana.wikimedia.org/d/000000167/wikidata-datamodel
3 https://discovery.wmflabs.org/wdqs/#wdqs_usage

Scaling challenges

Current challenges of WDQS

- Keeping up with write load
  - WDQS lagging behind Wikidata
- Keeping up with data size
  - No sharding support => larger disks and memory size required over time
  - Internal Blazegraph limitations (see T213210)
- All those challenges are linked
- Keeping up with query load
  - System overloaded, leading to high response times and timeouts
- Keeping stable and secured
  - Servers crashing leading to limited capacity and overload
  - Blazegraph no longer actively developed / maintained

Short term: Disaster mitigations

Catastrophic failure scenarios

Two major modes of catastrophic failure:

- Maxing out on data size
  - Not able to ingest any new data
  - Only historical data stays available
  - Breaks most editing workflows

- Query load vs. update lag
  - Can’t keep up updates under load
  - To a point where we never catch up

Disaster mitigation goals

1. Prevent total collapse of WDQS
2. Preserve minimum level of functionality

Disaster mitigation != long term goals

- Disaster mitigation is not the same as ideal end state
- Playbook for how to keep WDQS alive in the event of an emergency
- Some of these actions may be applicable to long term scaling

Planning for catastrophic failure

Steps we identified:

1. Transparency with community about current state of WDQS
2. Blazegraph size: prioritize data to (temporarily) delete from Blazegraph
3. Determine acceptable compromise between query loads and update lag

Blazegraph size: Data prioritization

We may need to temporarily delete/split data from Blazegraph

- Data will still exist in Wikidata and dumps

Currently investigating shape of the Wikidata graph and queries:

- Remove descriptions/labels/etc (vertical slicing)
- Delete subgraphs (horizontal slicing)
  - E.g. ~50% of Wikidata triples (40% of entities) are scholarly articles; ~2% of queries[1]
  - Astronomical objects; Lexemes also appear to be distinct subgraphs
  - May lead to medium-long term graph splitting

1 https://wikitech.wikimedia.org/w/index.php?title=User:AKhatun/Wikidata_Scholarly_Articles_Subgraph_Analysis

Query Load vs. Update lag

In production:

- New Streaming Updater: ~10 edits/sec → ~88 edits/sec

Options:

- Determine compromise for WDQS lag vs. updates, max lag threshold, max rate limit tolerance
- More aggressively rate limit bots
  - we already do to some extent

Disaster Mitigation summary

Emergency playbook for catastrophic failure scenarios

We are working on:

- Prioritizing graph data deletion/splitting
- Balance query load and update lag

We look forward to hearing more from the community as more information about our options becomes clear

User survey:
Towards a long-term strategy

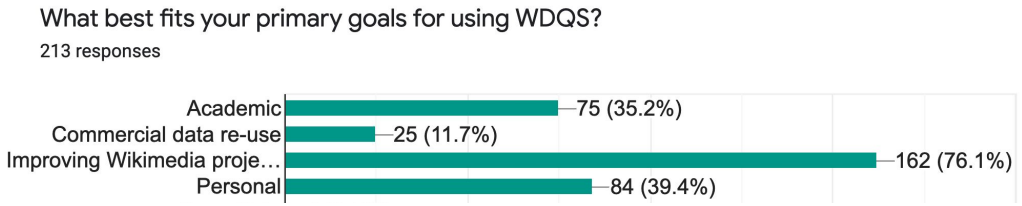
WDQS User Survey

- **Goal**: Learn more about WDQS user priorities, so we can optimize our scaling strategy with user needs in mind

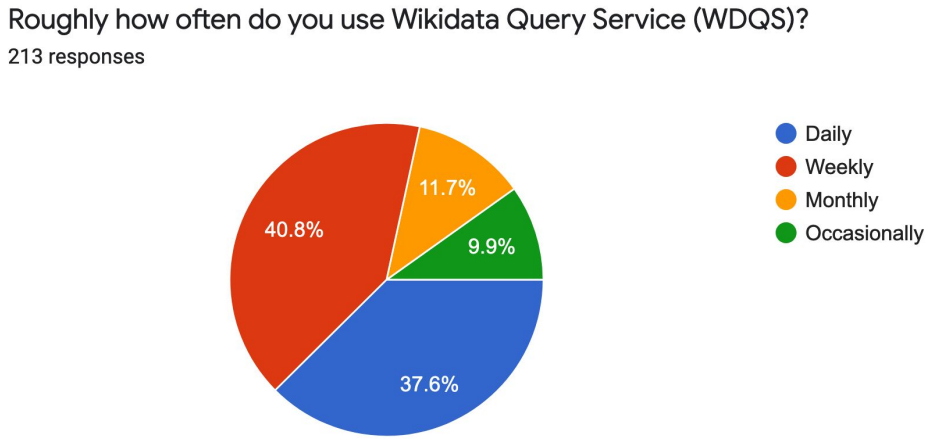
- Currently have 213 responses (12 Aug - 16 Sept)

WDQS User demographic

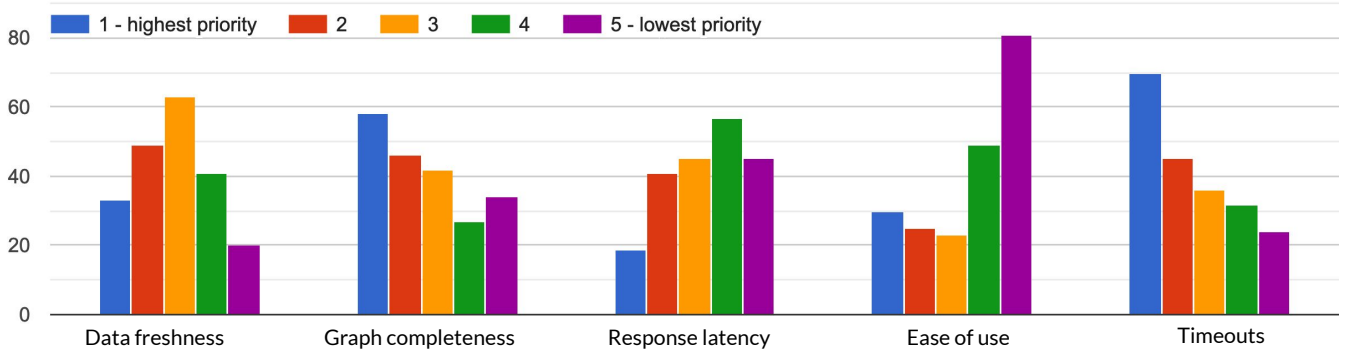
- Most users are interested in improving Wiki projects
  - Relatively low commercial use



- Most users use WDQS weekly or daily
  - Roughly equal number of weekly and daily users



WDQS User priorities



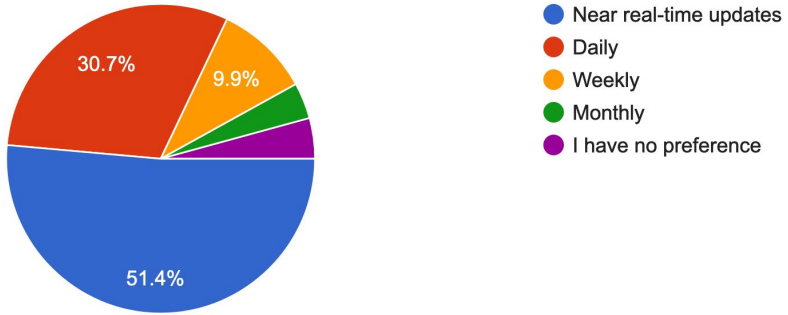
Eyeballing:

- Timeouts are top priority
- Graph completeness & data freshness next priorities
- Response latency relatively low priority
- Ease of use is lowest priority

WDQS User survey: data freshness

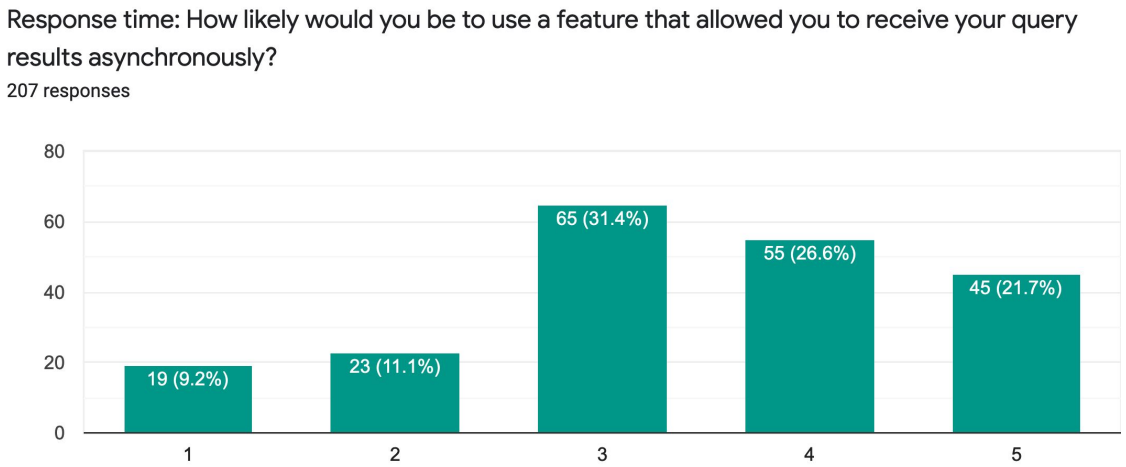
Data freshness: Generally speaking, how quickly do you need Wikidata edits be reflected in WDQS results?

212 responses



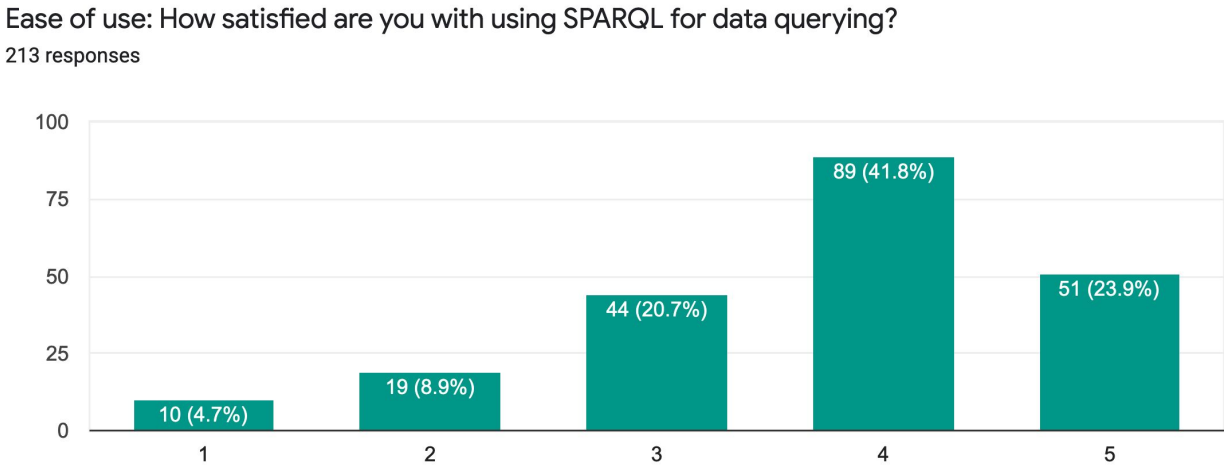
- ~50% want real-time data freshness
- May be skewed when considered independently
  - Who doesn't want freshest data all things being equal?

WDQS User survey: async results



- Most people would use a service with async results
- ...presumably that allowed for lenient timeout restrictions

WDQS User survey: SPARQL ease of use



- Most people are satisfied with using SPARQL
  - Caveat: most daily/weekly users are likely comfortable with SPARQL

WDQS survey summary

- Most users use WDQS at least once a week, and are interested in improving Wiki projects
- Timeouts are the biggest priority
- Half+ users want near real time data freshness
- Most would use an async querying service

Takeaway:

- Users are likely to use async results if they can fully run longer queries
- Optimizing for timeouts may require sacrificing data freshness and/or graph completeness

Long term: Where we go from here

Steps we are taking

- Identify and evaluate possible new backends
- Take pressure off the system
  - Create additional alternative services (e.g. REST API, search, ...)
  - Make it easier to run your own instance
  - Make Wikibase installation and maintenance easier so some content can find a better home

We need your help

How you can help

- Provide feedback on the plans we presented (-> unconference session, Wikidata:Query Service scaling update Aug 2021)
- Fill out our survey about your current use of WDQS (tinyurl.com/WDQSsurvey)
- Give input on selection criteria for new backend (-> T291207)
- Research and trial new backends (-> T206560)
- Contribute ideas for specialized additional services (-> T291340)

WIKI DATA CON
2021 - a sustainable future for Wikidata

Thanks for your attention!

Get in touch with us:

Mike Pham
mpham@wikimedia.org

Guillaume Lederrey
glederrey@wikimedia.org

Wikidata Mailing List
wikidata@lists.wikimedia.org

Lydia Pintscher
lydia.pintscher@wikimedia.de

Adam Shorland
adam.shorland@wikimedia.de