universität freiburg

# The QLever SPARQL Engine

**Talk @ Knowledge Graph Forum 2024**

**Roche Tower, Basel**

**May 30, 2024**

Prof. Dr. Hannah Bast
Faculty of Engineering
University of Freiburg

With credits to a great team, notably:

Johannes Kalmbach
Patrick Brosi
Robin Textor-Falconi
Julian Mundhahs

# Overview

- The QLever SPARQL Engine ... pronounce: "clever"
  - **Efficient** also for very large graphs, with moderate hardware
  - **Free** open-source software, high-quality codebase
  - **Easy** to set up yourself
  - Approaching full SPARQL 1.1 **conformance**
  - **Autocompletion** of SPARQL queries
  - Combined **SPARQL+Text** Search
  - **GeoSPARQL** support, in particular: very fast spatial joins
  - **Visualization** of very large results on a map

# Efficient and scalable on moderate hardware

- Selection of queries that are hard for other engines
  all run on a 2.000 € PC (AMD Ryzen 9, 16 Cores, 128 GB RAM)

  Ten people with most sitelinks on **Wikidata**

  Small final result, but has to read a lot of the data

  All object types on **Uniprot**, ranked by frequency

  Small final result (few types), but rdf:type is huge

  All predicates and their frequency, on **OpenStreetMap**

  Statistics over the complete dataset

  All movies with their **Wikipedia** abstract and **IMDb** rating

  SERVICE query involving large data exchange

# Performance evaluation

| SPARQL Engine | Code | Loading time | Loading speed | Index size | Query time | Ease of setup |
|---|---|---|---|---|---|---|
| Oxigraph | Rust | 640 s | 0.6 M/s | 67 GB | 93 s | very easy |
| Apache Jena | Java | 2392 s | 0.2 M/s | 32 GB | 69 s | very easy |
| Stardog | Java | 724 s | 0.5 M/s | 28 GB | 17 s | many hurdles |
| GraphDB | Java | 1066 s | 0.4 M/s | 28 GB | 16 s | some hurdles |
| Blazegraph | Java | 6326 s | < 0.1 M/s | 67 GB | 4.3 s | some hurdles |
| Virtuoso | C | 561 s | 0.7 M/s | 13 GB | 2.2 s | many hurdles |
| QLever | C++ | 231 s | 1.7 M/s | 8 GB | 0.7 s | very easy |

DBLP benchmark (400 M triples, average on spectrum of queries)
on larger datasets, the gap between the engines widens

# Setting it up yourself

- Running your own QLever instance is **easy**
  - For example, to run your own Wikidata server:

    pip install qlever
    qlever setup-config wikidata && qlever get-data
    qlever index && qlever start

  - Let's try it live for a small and a medium-sized dataset
    (on the kind of 2000 € PC mentioned on slide 3)

| | | |
|---|---|---|
| Olympic | ~ 2 million triples | ready in **2 s** |
| DBLP | ~ 400 million triples | ready in **4 min** |
| Wikidata | ~ 20 **b**illion triples | ready in **5 h** |
| UniProt | ~ 150 **b**illion triples | ready in **40 h** |

Performance **scales linearly** with the size of the dataset

# Free open-source, high code quality

- Codebase
  - Modern C++, very well documented
  - Extensive unit tests, code reviews, static analysis, ...
  - Continuous integration for various OSs and compilers
  - Runs with Docker or compiled natively
  - Bus factor > 1, meant to last
- Usability features
  - Individual query timeouts and cancellation
  - Individual query **analysis** (also **live** while query runs)
  - Powerful command-line interface

# SPARQL 1.1 Conformance

- We are almost there
  - The two biggest features still missing are

    SPARQL 1.1 **Named Graphs**

    SPARQL 1.1 **Update**

    We are actively working on these, already have prototype implementations, and hope to finish them this summer
  - Some of the more exotic features are still missing

    https://github.com/ad-freiburg/qlever/wiki/Current-deviations-from-the-SPARQL-1.1-standard

    We will add the missing features over time, trying to prioritize stuff that people need urgently (drop us a line)

# Context-Sensitive SPARQL Autocompletion

Query for: All Oscars of Meryl Streep and corresponding movies

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wdt:   <http://www.wikidata.org/prop/direct/>
PREFIX pq:    <http://www.wikidata.org/prop/qualifier/>
PREFIX ps:    <http://www.wikidata.org/prop/statement/>
PREFIX p:     <http://www.wikidata.org/prop/>
PREFIX wd:    <http://www.wikidata.org/entity/>
SELECT ?movie ?award WHERE {
  wd:Q873     p:P166     ?mediator .
  ?mediator   ps:P166    ?award_id .
  ?mediator   pq:P1686   ?movie_id .
  ?award_id   wdt:P31    wd:Q19020 .
  ?award_id   rdfs:label ?award . FILTER (LANG(?award) = "en")
  ?movie_id   rdfs:label ?movie . FILTER (LANG(?movie) = "en")
}
```

# SPARQL+Text

- SPARQL+Text search
  - Efficient combination of SPARQL queries and keyword search, for example

    Find all occurrences of an astronaut in Wikipedia, next to the words moon and walk*

    Note: in principle, such queries can also be processed with **FILTER CONTAINS** or **FILTER REGEX**, but not efficiently so

# GeoSPARQL

- Efficient support for geo-spatial queries
  - QLever can compute and handle basic **GeoSPARQL**, in particular the typical DE-9IM spatial predicates

    ogc:sfIntersects, ogc:sfCovers, ogc:sfContains, ogc:sfTouches, ogc:sfCrosses, ogc:sfEquals, ogc:sfOverlaps
  - Example queries

    All streets from OpenStreetMap contained in region X

    The power network of the European Union

    Which countries contain river X by how much
- Interactive visualization of large results on a map
  - No geo-spatial database we know can draw large numbers of geometric objects on a map in reasonable time … we can