The QLever SPARQL Engine

Session @ Wikidata Modelling Days 2023
December 2, 2023   Online

Hannah Bast and Johannes Kalmbach
Department of Computer Science
University of Freiburg
What is QLever

QLever is a SPARQL engine for large knowledge graphs

- Very fast on standard hardware  slide 3, demo
- Efficient text and geospatial search  slide 4, demo
- Easy to set up yourself  slide 5, demo
- 100% open source, high-q codebase  slide 6, demo
- Clever query autocompletion  slide 7 - 9, demo

Questions welcome at any time
Example queries

- Typical queries that timeout on WDQS
  - ORDER BY with small result, but large intermediate data
    Ten movies with most sitelinks and their description
  - GROUP BY with small result, but large intermediate data
    Highest mountain per country
  - Simple queries with a large result
    All people and their name
  - Statistics over the complete data
    All predicates, with their name and frequency
  - Explorative queries
    Predicates attached to entities of type person
Special features

More example queries

- Federated queries (SERVICE)
  - All movies and their IMDb rating (Wikidata + IMDb)
  - The power network of the EU (Wikidata + OpenStreetMap)

- Geospatial queries
  - All entities with location in a 100 km ring around Freiburg
  - All streets contained in OpenStreetMap region X
  - Which countries contain river X how much

- SPARQL combined with text search
  - Movies where the Wikipedia abstract matches keywords X
  - Astronauts who walked on the moon
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 index start

  - Let's try it live for a small and a medium-sized dataset (on a 2000 € PC, a higher-end machine would be even faster)

    Olympics ca. 2 million triples ready in 2s
    DBLP ca. 400 million triples ready in 3min
    Wikidata (full) ca. 19 billion triples ready in 4h

    This is about as fast as just downloading the data

    QLever can manage over 100 billion triples on a single machine
Code quality and more features

- **Code quality**
  - Modern C++, very well-documented
  - Extensive unit tests, code reviews, static analysis, ...
  - Continuous integration on various platforms (Ubuntu, MacOS, ...)
  - Can be used with Docker or compiled natively
  - Meant to last

- **Some more features**
  - Individual query timeout
  - Individual query cancellation
  - Individual query *analysis* (also live while query runs)
Typing SPARQL queries is hard

- Consider the following simple search request
  
  Which Oscars did Meryl Streep win and for which movies?

- The result we are looking for is something like this:
  
  Academy Award for Best Supporting Actress  Kramer vs. Kramer
  Academy Award for Best Actress  Sophie's Choice
  Academy Award for Best Actress  The Iron Lady

- On the next slide, you see the correct SPARQL query on the Wikidata knowledge graph

- QLever's context-sensitive autocompletion let's you construct this complex query rather easily

  No chance to get this right fast without such help
Query Autocompletion

```sparql
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 {
  ?award_id wdt:P31 wd:Q19020 .
  ?award_id rdfs:label ?award . FILTER (LANG(?award) = "en")
  ?movie_id rdfs:label ?movie . FILTER (LANG(?movie) = "en")
}
```
Natural language questions

- Automatic translation to SPARQL queries
  - ChatGPT already does a good job guessing the right structure for a query, but it usually gets the identifiers wrong
    
    Example: birth places of all people with first name X
  - We are working on an approach to fix that