Efficient and Correct Federated Queries for the QLever SPARQL engine

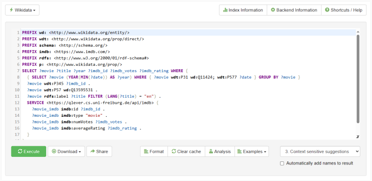
Moritz Dom

19.03.2025

Computer Science Department
University of Freiburg

What is QLever?

- SPARQL query engine
- operating on RDF knowledge bases



```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX schema: <http://schema.org/>
PREFIX imdb: <https://www.imdb.com/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX p: <http://www.wikidata.org/prop/>
SELECT ?movie ?title ?year ?imdb_id ?imdb_votes ?imdb_rating WHERE {
  { SELECT ?movie (YEAR(MIN(?date)) AS ?year) WHERE { ?movie wdt:P31 wd:Q11424; wdt:P577 ?date } GROUP BY ?movie }
  ?movie wdt:P345 ?imdb_id .
  ?movie wdt:P57 wd:Q13595531 .
  ?movie rdfs:label ?title FILTER (LANG(?title) = "en") .
  SERVICE <https://qlever.cs.uni-freiburg.de/api/imdb> {
    ?movie_imdb imdb:id ?imdb_id .
    ?movie_imdb imdb:type "movie" .
    ?movie_imdb imdb:numVotes ?imdb_votes .
    ?movie_imdb imdb:averageRating ?imdb_rating .
  }
```

https://qlever.cs.uni-freiburg.de

SPARQL Federated Queries

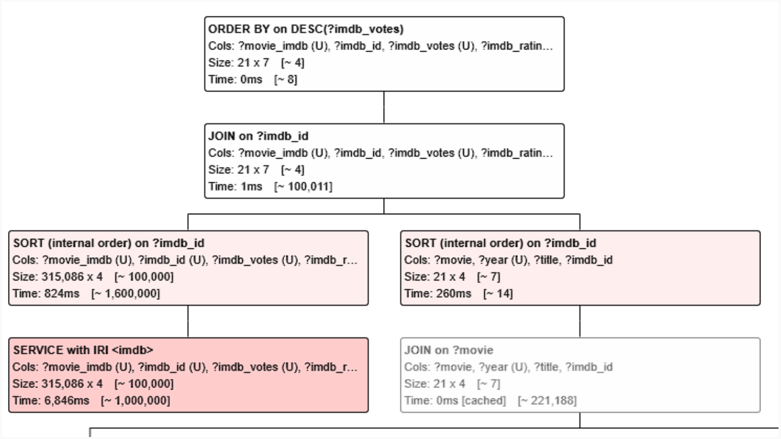
```
SELECT ?movie ?title ?year ?imdb_id ?imdb_votes ?imdb_rating WHERE {
  { SELECT ?movie (YEAR(MIN(?date)) AS ?year)
    WHERE { ?movie wdt:P31 wd:Q11424; wdt:P577 ?date } GROUP BY ?movie
  }
  ?movie wdt:P345 ?imdb_id .
  ?movie wdt:P57 wd:Q13595531 .
  ?movie rdfs:label ?title FILTER (LANG(?title) = "en") .

  SERVICE <https://qlever.cs.uni-freiburg.de/api/imdb> {
    ?movie_imdb imdb:id ?imdb_id .
    ?movie_imdb imdb:type "movie" .
    ?movie_imdb imdb:numVotes ?imdb_votes .
    ?movie_imdb imdb:averageRating ?imdb_rating .
  }
}
ORDER BY DESC(?imdb_votes)
```

Example query: Movies by Ethan Coen, with IMDb ratings and votes

Federated Query forwarded to the given endpoint:

```
SELECT * WHERE {
    ?movie_imdb imdb:id ?imdb_id .
    ?movie_imdb imdb:type "movie" .
    ?movie_imdb imdb:numVotes ?imdb_votes .
    ?movie_imdb imdb:averageRating ?imdb_rating .
}
```

Efficiency


ORDER BY on DESC(?imdb_votes)
Cols: ?movie_imdb (U), ?imdb_id, ?imdb_votes (U), ?imdb_ratin...
Size: 21 x 7 [~ 4]
Time: 0ms [~ 8]

JOIN on ?imdb_id
Cols: ?movie_imdb (U), ?imdb_id, ?imdb_votes (U), ?imdb_ratin...
Size: 21 x 7 [~ 4]
Time: 1ms [~ 100,011]

SORT (internal order) on ?imdb_id
Cols: ?movie_imdb (U), ?imdb_id (U), ?imdb_votes (U), ?imdb_r...
Size: 315,086 x 4 [~ 100,000]
Time: 824ms [~ 1,600,000]

SORT (internal order) on ?imdb_id
Cols: ?movie, ?year (U), ?title, ?imdb_id
Size: 21 x 4 [~ 7]
Time: 260ms [~ 14]

SERVICE with IRI <imdb>
Cols: ?movie_imdb (U), ?imdb_id (U), ?imdb_votes (U), ?imdb_r...
Size: 315,086 x 4 [~ 100,000]
Time: 6,846ms [~ 1,000,000]

JOIN on ?movie
Cols: ?movie, ?year (U), ?title, ?imdb_id
Size: 21 x 4 [~ 7]
Time: 0ms [cached] [~ 221,188]

Previous Runtime information of the example query.

Efficiency

- reduce federated query result to relevant rows
- JOIN: an operands row is relevant if the others result set contains a row with matching values in the joined on columns
- Solution: VALUES clause with the values of the common variables of the local result
- Similar for MINUS and OPTIONAL JOIN operations

```
SELECT * {
    VALUES (?imdb_id) {("tt0477348") ("tt0118715") ... } .
    ?movie_imdb imdb:id ?imdb_id .
    ?movie_imdb imdb:type "movie" .
    ?movie_imdb imdb:numVotes ?imdb_votes .
    ?movie_imdb imdb:averageRating ?imdb_rating .
  }
```

Service query constrained with VALUES clause.

Correctness

- result format TSV, with each binding encoded in the TURTLE language
- TURTLE parser produced malformed literals, e.g. "Abraham \"Bram\" Wiertz" $\rightarrow$ "Abraham \"
- instead use more robust/verbose JSON format

```
?imdb_id   ?movie
"tt0116282"  "Fargo"@en
"tt0118715"  "The Big Lebowski"@en
```

SPARQL TSV result format

LazyJsonParser

- compute and export results in chunks
- Service operation imports a result
- additional memory cost for the imported string
- lazy processing of a JSON result is not as trivial as with TSV

LazyJsonParser

- goal: reconstruct partial json object
- no data retrieval
- input characteristic: one array with many elements

LazyJsonParser

```
{
  "head": {
    "vars": ["counter"]
  },
  "results": {
    "bindings": [
      {
        "counter": {
          "type": "literal",
          "value": "one"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "two"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "three"
        }
      }
    ]
  }
}
```

- No output
- add chunk to input buffer

LazyJsonParser

```
{
  "head": {
    "vars": ["counter"]
  },
  "results": {
    "bindings": [
      {
        "counter": {
          "type": "literal",
          "value": "one"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "two"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "three"
        }
      }
    ]
  }
}
```

- first bindings read
- reconstruct suffix of the object
- write partially read binding to input buffer

Output:

```
{
  "head": {
    "vars": ["counter"]
  },
  "results": {
    "bindings": [
      {
        "counter": {
          "type": "literal",
          "value": "one"
        }
      }
    ]
  }
}
```

LazyJsonParser

```
{
  "head": {
    "vars": ["counter"]
  },
  "results": {
    "bindings": [
      {
        "counter": {
          "type": "literal",
          "value": "one"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "two"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "three"
        }
      }
    ]
  }
}
```

- end of the object reached
- reconstruct only the prefix

Output:

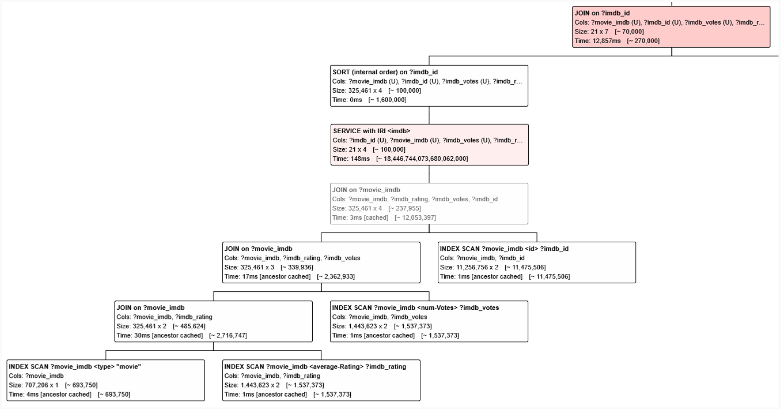
```
{
  "results": {
    "bindings": [
      {
        "counter": {
          "type": "literal",
          "value": "two"
        }
      },
      {
        "counter": {
          "type": "literal",
          "value": "three"
        }
      }
    ]
  }
}
```

Usability

- QLever provides Runtime information useful for debugging
- extended Service operation with a WebSocket client
- only supports QLever endpoints, not part of SPARQL standard

Usability

- QLever provides Runtime information useful for debugging
- extended Service operation with a WebSocket client
- only supports QLever endpoints, not part of SPARQL standard



```
JOIN on ?imdb_id
Cols: ?movie_imdb (U), ?imdb_id (U), ?imdb_votes (U), ?imdb_r...
Size: 21 x 7   [~ 70,000]
Time: 12,857ms   [~ 270,000]

SORT (internal order) on ?imdb_id
Cols: ?movie_imdb (U), ?imdb_id (U), ?imdb_votes (U), ?imdb_r...
Size: 325,461 x 4   [~ 100,000]
Time: 0ms   [~ 1,600,000]

SERVICE with IRI <imdb>
Cols: ?imdb_id (U), ?movie_imdb (U), ?imdb_votes (U), ?imdb_r...
Size: 21 x 4   [~ 100,000]
Time: 148ms   [~ 18,446,744,073,680,062,000]

JOIN on ?movie_imdb
Cols: ?movie_imdb, ?imdb_rating, ?imdb_votes, ?imdb_id
Size: 325,461 x 4   [~ 237,955]
Time: 3ms [cached]   [~ 12,053,397]

JOIN on ?movie_imdb
Cols: ?movie_imdb, ?imdb_rating, ?imdb_votes
Size: 325,461 x 3   [~ 339,936]
Time: 17ms [ancestor cached]   [~ 2,362,933]

INDEX SCAN ?movie_imdb <id> ?imdb_id
Cols: ?movie_imdb, ?imdb_id
Size: 11,256,756 x 2   [~ 11,475,506]
Time: 1ms [ancestor cached]   [~ 11,475,506]

JOIN on ?movie_imdb
Cols: ?movie_imdb, ?imdb_rating
Size: 325,461 x 2   [~ 485,624]
Time: 30ms [ancestor cached]   [~ 2,716,747]

INDEX SCAN ?movie_imdb <num-Votes> ?imdb_votes
Cols: ?movie_imdb, ?imdb_votes
Size: 1,443,623 x 2   [~ 1,537,373]
Time: 1ms [ancestor cached]   [~ 1,537,373]

INDEX SCAN ?movie_imdb <type> "movie"
Cols: ?movie_imdb
Size: 707,206 x 1   [~ 693,750]
Time: 4ms [ancestor cached]   [~ 693,750]

INDEX SCAN ?movie_imdb <average-Rating> ?imdb_rating
Cols: ?movie_imdb, ?imdb_rating
Size: 1,443,623 x 2   [~ 1,537,373]
Time: 1ms [ancestor cached]   [~ 1,537,373]
```

Benchmark

Query	unconstraint	constraint
Movies directed by Ethan Coen	13,150ms	103ms
JOIN of small Service with large sibling result	253ms	-
VALUES clause size: 10000	-	2522ms
VALUES clause size: 20000	-	5203ms
VALUES clause size: 30000	-	7701ms
VALUES clause size: 40000	-	10200ms

Benchmark of the Service operation

Benchmark

Parser	chunk size (bytes)	execution time
nlohmann/json	-	19,935ms
LazyJsonParser	100	37,490ms
	1000	22,173ms
	1500	21,421ms
	2000	21,084ms
	2500	21,321ms
	3000	21,858ms
	4000	23,065ms
	8000	23,034ms
	16000	22,986ms
	32000	23,051ms
	64000	23,222ms

Benchmark of the LazyJsonParser for different chunk sizes, operating on a SPARQL JSON result with 10 million rows / 1GB size.

Efficient and Correct Federated Queries for the QLever SPARQL engine

Moritz Dom

19.03.2025

Computer Science Department
University of Freiburg