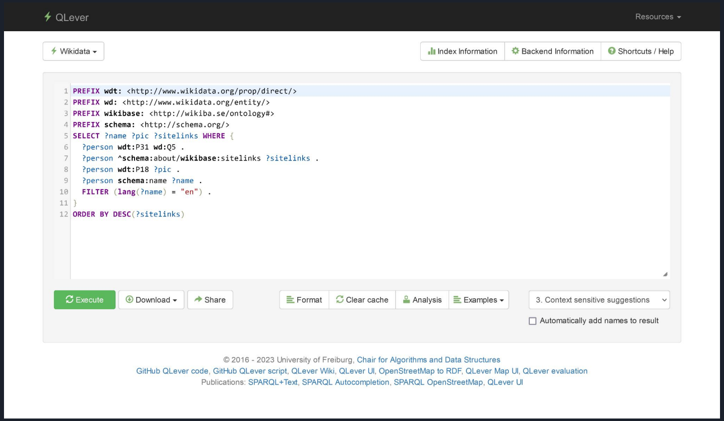
Lazy Evaluation of SPARQL Queries with Caching

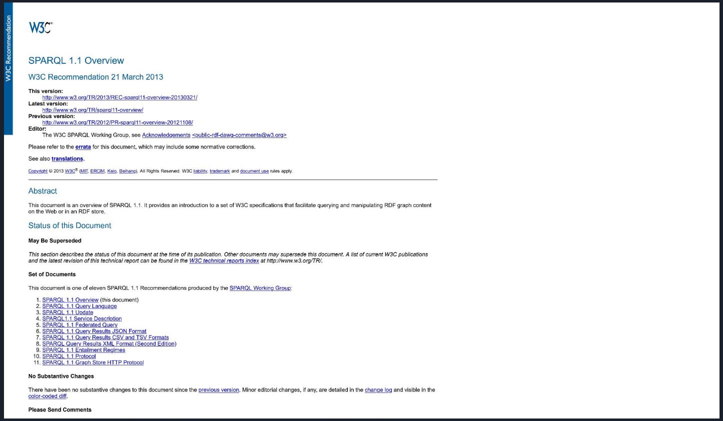
Robin Textor-Falconi

QLever & SPARQL



```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX schema: <http://schema.org/>
SELECT ?name ?pic ?sitelinks WHERE {
  ?person wdt:P31 wd:Q5 .
  ?person ^schema:about/wikibase:sitelinks ?sitelinks .
  ?person wdt:P18 ?pic .
  ?person schema:name ?name .
  FILTER (lang(?name) = "en") .
}
ORDER BY DESC(?sitelinks)
```

© 2016 - 2023 University of Freiburg, Chair for Algorithms and Data Structures
GitHub QLever code, GitHub QLever script, QLever Wiki, QLever UI, OpenStreetMap to RDF, QLever Map UI, QLever evaluation
Publications: SPARQL+Text, SPARQL Autocompletion, SPARQL OpenStreetMap, QLever UI



SPARQL 1.1 Overview

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
Latest version:
http://www.w3.org/TR/sparql11-overview/
Previous version:
http://www.w3.org/TR/2012/PR-sparql11-overview-20121108/
Editor:
The W3C SPARQL Working Group, see Acknowledgements <public-rdf-dawg-comments@w3.org>

Please refer to the **errata** for this document, which may include some normative corrections.

See also **translations**.

Abstract

This document is an overview of SPARQL 1.1. It provides an introduction to a set of W3C specifications that facilitate querying and manipulating RDF graph content on the Web or in an RDF store.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Set of Documents

This document is one of eleven SPARQL 1.1 Recommendations produced by the SPARQL Working Group:

1. SPARQL 1.1 Overview (this document)
2. SPARQL 1.1 Query Language
3. SPARQL 1.1 Update
4. SPARQL1.1 Service Description
5. SPARQL 1.1 Federated Query
6. SPARQL 1.1 Query Results JSON Format
7. SPARQL 1.1 Query Results CSV and TSV Formats
8. SPARQL Query Results XML Format (Second Edition)
9. SPARQL 1.1 Entailment Regimes
10. SPARQL 1.1 Protocol
11. SPARQL 1.1 Graph Store HTTP Protocol

No Substantive Changes

There have been no substantive changes to this document since the previous version. Minor editorial changes, if any, are detailed in the change log and visible in the color-coded diff.

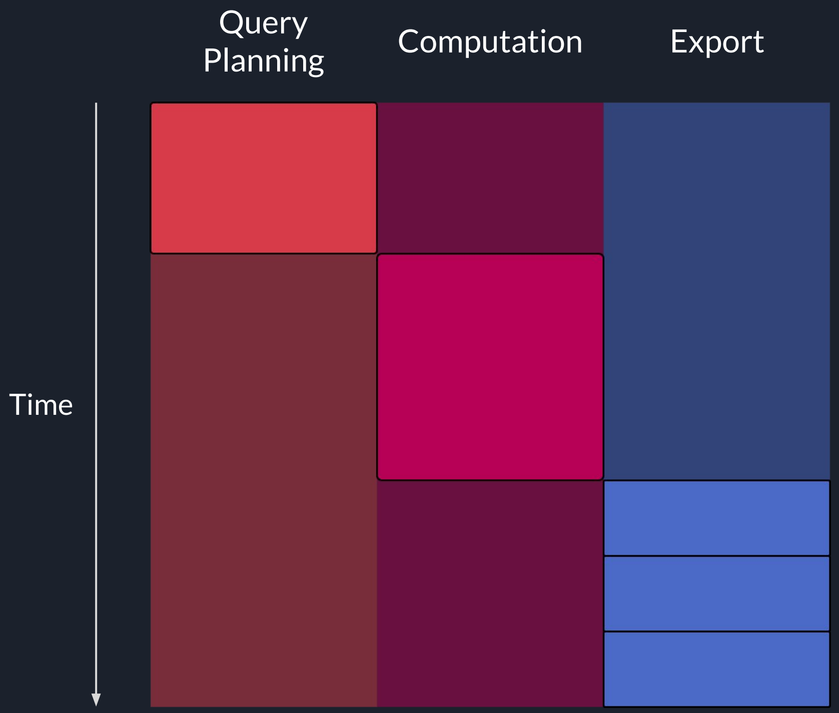
Please Send Comments

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

https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/

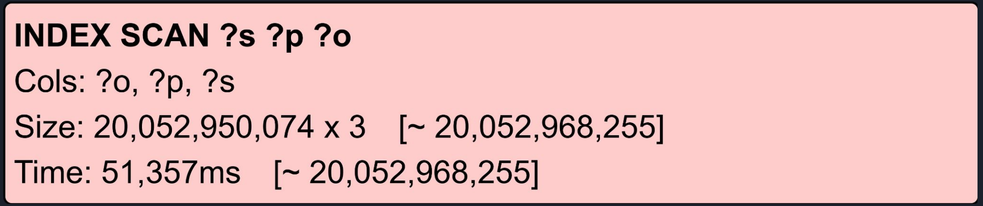
Problem Definition

Current Processing Model



Practical Example

```
SELECT * WHERE { ?s ?p ?o }
```



INDEX SCAN ?s ?p ?o

Cols: ?o, ?p, ?s

Size: 20,052,950,074 x 3   [~ 20,052,968,255]

Time: 51,357ms   [~ 20,052,968,255]

Practical Example

INDEX SCAN ?s ?p ?o
Cols: ?o, ?p, ?s
Size: 20,052,950,074 x 3 [~ 20,052,968,255]
Time: 51,357ms [~ 20,052,968,255]

$$\begin{aligned} & 3 \cdot 8 \cdot 20{,}052{,}950{,}074 \text{ bytes} \\ = \; & 481{,}270{,}801{,}776 \text{ bytes} \\ \approx \; & \textbf{481 GB !!} \end{aligned}$$

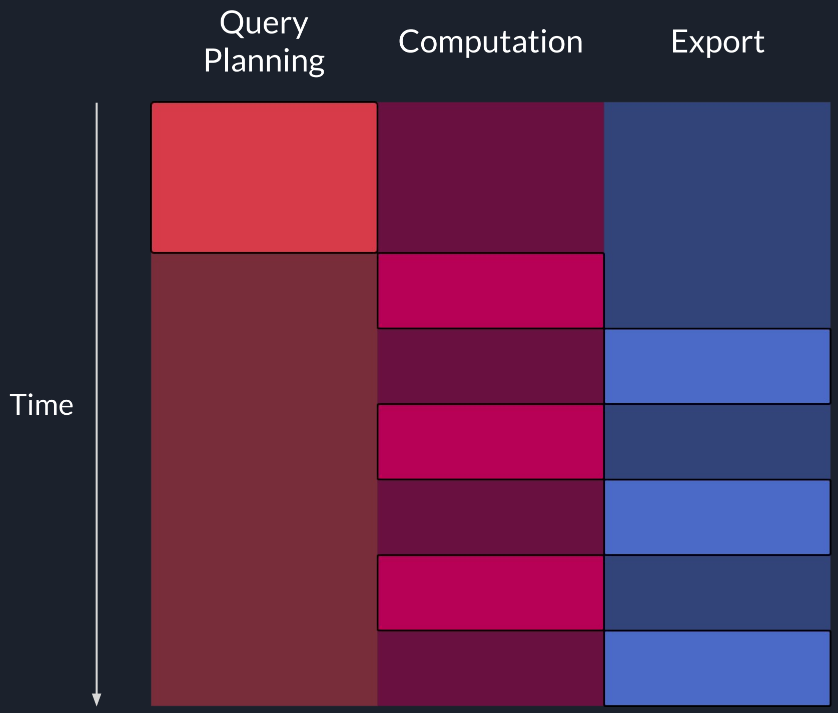
High memory requirement!

Existing Workaround

```
SELECT * WHERE { ?s ?p ?o } OFFSET           0 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    10000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    20000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    30000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    40000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    50000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    60000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    70000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    80000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET    90000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET   100000000 LIMIT 10000000
SELECT * WHERE { ?s ?p ?o } OFFSET   110000000 LIMIT 10000000
                                   …
SELECT * WHERE { ?s ?p ?o } OFFSET 20050000000 LIMIT 10000000
```

Solution

Proposed Processing Model



Code Examples in Different Languages

```cpp
#include <iostream>
#include <string>
#include <ranges>

int main() {
  std::ranges::istream_view<std::string> input{std::cin};
  for (const auto& line : input) {
    std::cout << line << std::endl;
  }
  return 0;
}
```

```csharp
using System;

class Cat
{
  static void Main()
  {
    string? line;
    while ((line = Console.ReadLine()) is not null)
    {
      Console.WriteLine(line);
    }
  }
}
```

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.stream.Stream;

public class Cat {
  public static void main(String[] args) throws IOException {
    try (var reader = new InputStreamReader(
           System.in, StandardCharsets.UTF_8);
         var bufferedReader = new BufferedReader(reader)) {
      bufferedReader.lines().forEach(System.out::println);
    }
  }
}
```

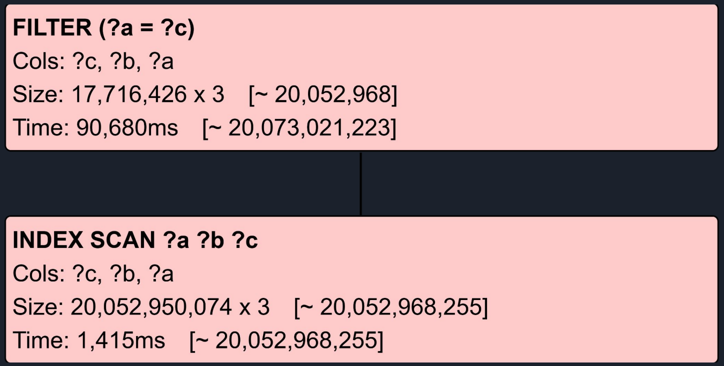
```haskell
import System.IO (isEOF)

main :: IO ()
main = do
    eof <- isEOF
    if eof
        then return ()
        else do
            line <- getLine
            putStrLn line
            main
```

Challenges of the Thesis

- Implement general mechanism to existing codebase
  - Keep backwards compatibility
  - Keep code architecture largely untouched
- Keep Caching Mechanism
- Identify operations benefiting from new mechanism
- Add actual implementations for suited operations
  - Performance should ideally not regress by much

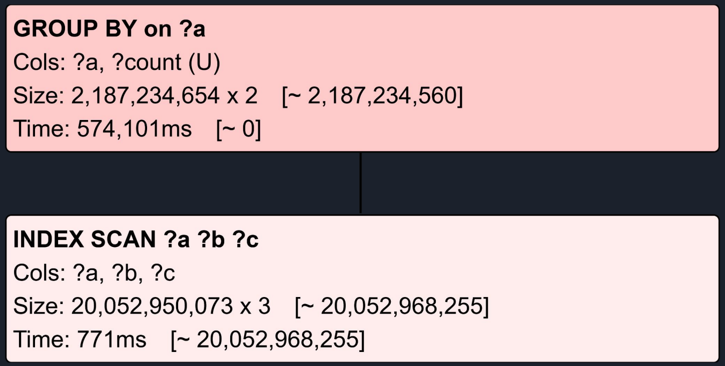
Demo



FILTER (?a = ?c)
Cols: ?c, ?b, ?a
Size: 17,716,426 x 3 [~ 20,052,968]
Time: 90,680ms [~ 20,073,021,223]

INDEX SCAN ?a ?b ?c
Cols: ?c, ?b, ?a
Size: 20,052,950,074 x 3 [~ 20,052,968,255]
Time: 1,415ms [~ 20,052,968,255]

qlever.cs.uni-freiburg.de/wikidata/CkGEt4

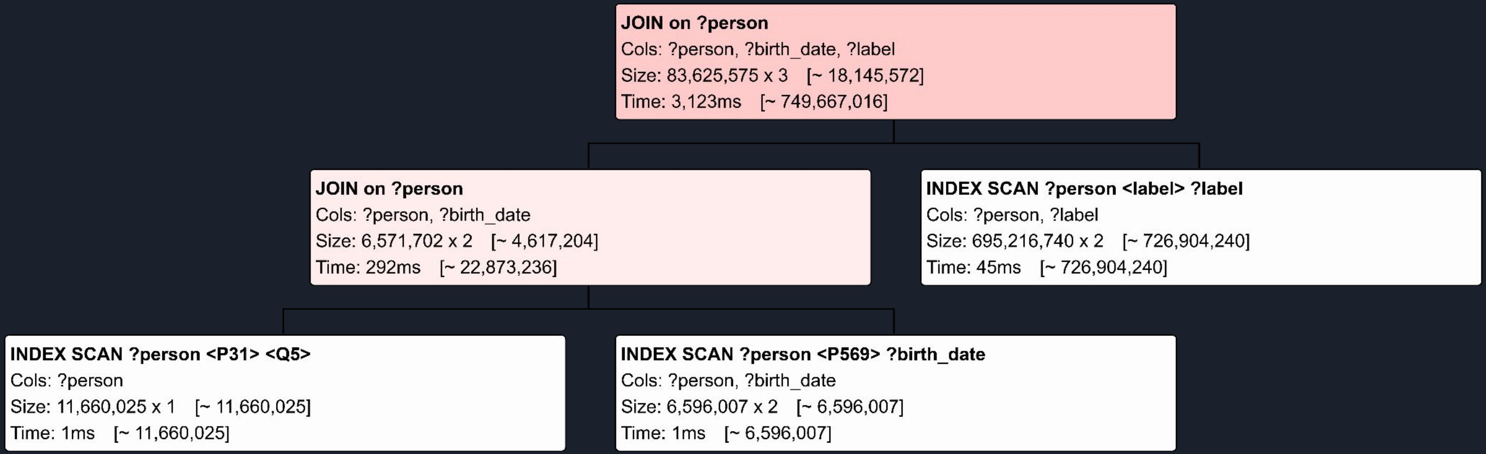


GROUP BY on ?a
Cols: ?a, ?count (U)
Size: 2,187,234,654 x 2 [~ 2,187,234,560]
Time: 574,101ms [~ 0]

INDEX SCAN ?a ?b ?c
Cols: ?a, ?b, ?c
Size: 20,052,950,073 x 3 [~ 20,052,968,255]
Time: 771ms [~ 20,052,968,255]

qlever.cs.uni-freiburg.de/wikidata/msUuJN

Demo



JOIN on ?person
Cols: ?person, ?birth_date, ?label
Size: 83,625,575 x 3 [~ 18,145,572]
Time: 3,123ms [~ 749,667,016]

JOIN on ?person
Cols: ?person, ?birth_date
Size: 6,571,702 x 2 [~ 4,617,204]
Time: 292ms [~ 22,873,236]

INDEX SCAN ?person <label> ?label
Cols: ?person, ?label
Size: 695,216,740 x 2 [~ 726,904,240]
Time: 45ms [~ 726,904,240]

INDEX SCAN ?person <P31> <Q5>
Cols: ?person
Size: 11,660,025 x 1 [~ 11,660,025]
Time: 1ms [~ 11,660,025]

INDEX SCAN ?person <P569> ?birth_date
Cols: ?person, ?birth_date
Size: 6,596,007 x 2 [~ 6,596,007]
Time: 1ms [~ 6,596,007]

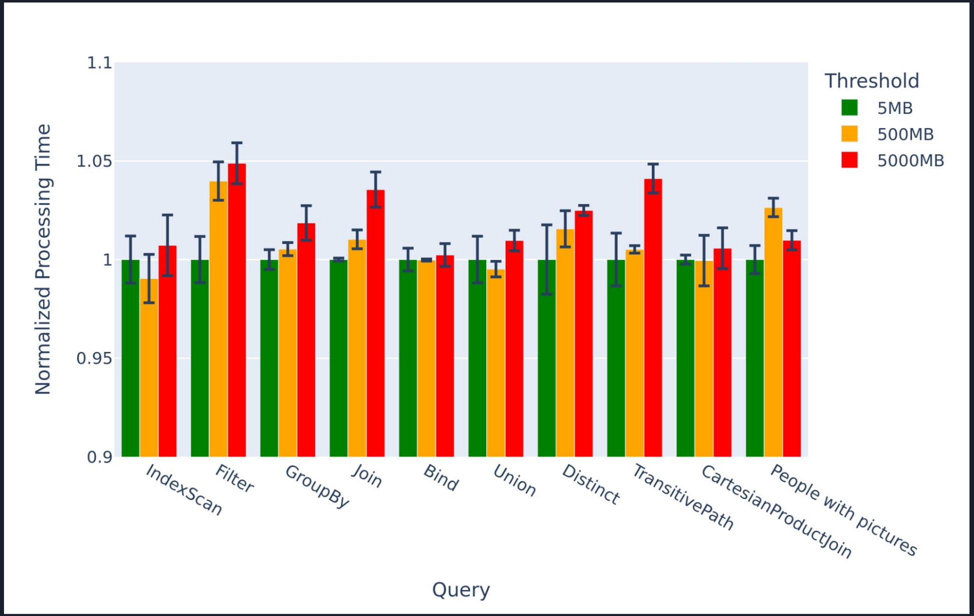
qlever.cs.uni-freiburg.de/wikidata/Zg778r

Performance Analysis

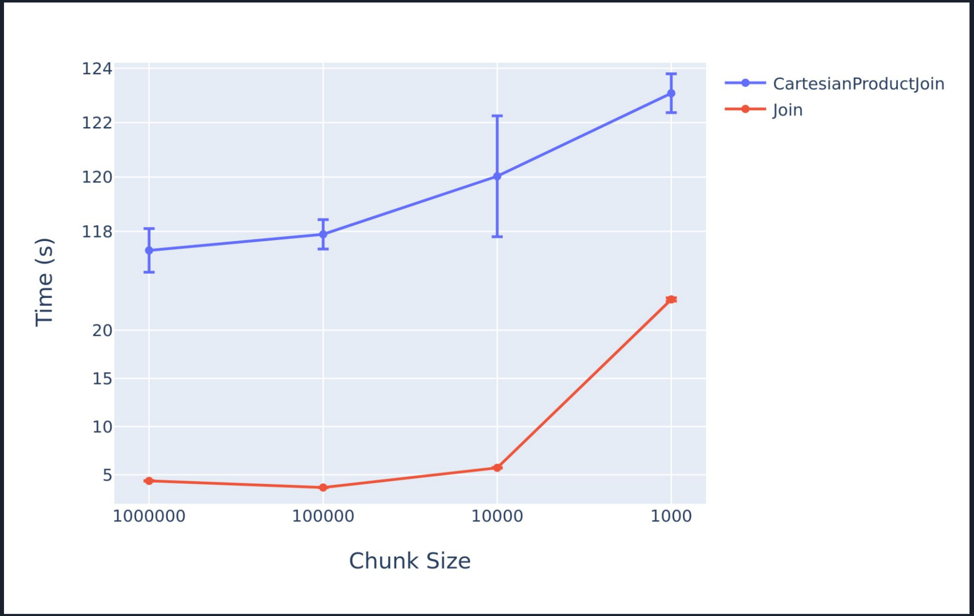
Laziness Performance Overhead

Query	Processing Time		Memory Delta	
	Lazy	Non-Lazy	Lazy	Non-Lazy
IndexScan example	20 s	-	702 MB	> 48 GB
Filter example	127 s	-	702 MB	> 48 GB
GroupBy example	608 s	-	702 MB	> 48 GB
Join example	3715 ms	4833 ms	702 MB	2151 MB
Bind example	447 min	-	1130 MB	> 48 GB
Union example	39 s	-	702 MB	> 48 GB
Distinct example	82 s	-	702 MB	> 48 GB
TransitivePath example	13 s	15 s	702 MB	2244 MB
CartesianProductJoin example	111 s	-	702 MB	> 48 GB
GroupBy example variant	385 ms	-	705 MB	> 48 GB
CartesianProductJoin example variant	144 ms	122 ms	702 MB	702 MB
People with pictures	2342 ms	2222 ms	851 MB	996 MB

Caching Overhead



Chunk Size Impact



Lazy Evaluation of SPARQL Queries with Caching

Robin Textor-Falconi