Automated standard compliance testing and visualization for the QLever SPARQL engine

Rico Andris

University of Freiburg

26.03.2024
Problem - Background

- **Resource Description Framework (RDF)**
- Designed by World Wide Web Consortium (W3C)
- Directed graph using triples
- subject, predicate, object

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>isA</td>
<td>Mammal</td>
</tr>
<tr>
<td>Cat</td>
<td>isA</td>
<td>Mammal</td>
</tr>
<tr>
<td>Parrot</td>
<td>isA</td>
<td>Bird</td>
</tr>
</tbody>
</table>

Dog isA Mammal
Cat isA Mammal
Parrot isA Bird
Problem - Background

- **SPARQL Protocol and RDF Query Language (SPARQL)**
- Designed by the World Wide Web Consortium

```sparql
SELECT ?animal
WHERE {
  ?animal isA Bird .
}
```

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>isA</td>
<td>Mammal</td>
</tr>
<tr>
<td>Cat</td>
<td>isA</td>
<td>Mammal</td>
</tr>
<tr>
<td>Parrot</td>
<td>isA</td>
<td>Bird</td>
</tr>
</tbody>
</table>

```sparql
?animal
Parrot
```
Problem - Background

- **QLever** SPARQL engine
- Process and execute query
- Aims to fully support SPARQL 1.1
- Test the **standard compliance** of QLever using standardized tests provided by the W3C
Problem - Background

- SPARQL 1.1 test suite
- Consists of 603 tests
  1. Query/Update Evaluation Tests
  2. Syntax Tests
  3. Format Tests
  4. Protocol Tests
  5. Service Description Tests
Problem - Background

Live demo
1. Automate test suite execution
2. Visualize test result
3. Compare two executions
4. Integrate into workflow

Retaining and improving QLever’s standard compliance
Questions ?
Approach - Extracting tests

- Using QLever
- Tests specify action and result
- Stored in manifest file

```turtle
:agg01 rdf:type mf:QueryEvaluationTest ;
   mf:name "COUNT 1" ;
   mf:feature sparql:count ;
   rdfs:comment "Simple count" ;
   dawgt:approval dawgt:Approved ;
   dawgt:approvedBy <http://www.w3.org/2009/sparql/meeting/2012-01-31#resolution_3> ;
   mf:action
      [ qt:query <agg01.rq> ;
        qt:data <agg01.ttl> ] ;
   mf:result <agg01.srx>
.
```
Approach - Test execution

1. Index graph
2. Prepare and send the query
3. Evaluate response
   1. Check response status
   2. if successful: **compare result**
4. Generate a string, highlighting differences in the results
Approach - Test execution

:agg01 rdf:type mf:QueryEvaluationTest ;
  mf:name "COUNT 1";
  mf:feature sparql:count ;
  rdfs:comment "Simple count" ;
  dawgt:approval dawgt:Approved ;
  dawgt:approvedBy <http://www.w3.org/2009/sparql/meeting/2012-01-31#resolution_3> ;
  mf:action
    [ qt:query <agg01.rq> ;
      qt:data <agg01.ttl> ] ;
  mf:result <agg01.srx> .
Approach - Comparing results

• Result formats
  1. Turtle or RDF/XML
  2. SPARQL 1.1 Query Results XML Format
  3. SPARQL 1.1 Query Results CSV/TSV Format
  4. SPARQL 1.1 Query Results JSON Format

• Implement comparison
Approach - Comparing results

• Custom comparison

Method for SPARQL 1.1 Query Results

1. Iterate over the elements of one result
2. Remove matching elements in both results
3. If both results are empty the results are equivalent
Approach - Comparing results

• Custom comparison
• Matching elements:
  • Ignore order
  • Handle blank nodes
  • Consider QLever specifics
Protocol tests

:query_post_form rdf:type mf:ProtocolTest ;
  mf:name "query via URL-encoded POST" ;
  rdfs:comment ""

### Request

```plaintext
POST /sparql/ HTTP/1.1
Host: www.example
User-agent: sparql-client/0.1
Content-Type: application/x-www-url-form-urlencoded
Content-Length: XXX

query=ASK%20%7B%7D
```

### Response

2xx or 3xx response
Content-Type: application/sparql-results+xml, application/sparql-results+json

```
true
"" ;
```
1. Build a string representation of the result
2. Highlight parts that don’t have a match in the other result
   1. Construct an HTML element for a leftover part.
   2. Replace that part in the string representation with the HTML element

result1
<span class="error"> result2 </span>
result3

result1
result2
result3
Approach

Questions ?
Approach - Website

- HTML, CSS, JS
- No custom backend server
- Bootstrap
Approach - Workflow

- GitHub Action
  1. Build QLever
  2. Run tests
  3. Send results to the visualization website
  4. Fail Action if prev. passed tests now fail
  5. Link website comparing the run with the main branch
Questions ?
Results Test Suite Execution

• Using QLever 22.03.2024
• Executing 600 tests:
  282 Query Evaluation Tests
  3 Result Format Tests
  94 Update Evaluation Tests
  169 Syntax Tests
  52 Protocol Tests
• 23.83% successfully processed
• 69.33% fail
• 6.83% semi-successful
## Results Test Suite Execution

<table>
<thead>
<tr>
<th>Category</th>
<th>Tests</th>
<th>Passed</th>
<th>Semi-Passed</th>
<th>Failed</th>
<th>Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>301</td>
<td>97</td>
<td>39</td>
<td>165</td>
<td>32.22% / 45.18%</td>
</tr>
<tr>
<td>Update</td>
<td>157</td>
<td>21</td>
<td>0</td>
<td>136</td>
<td>13.38%</td>
</tr>
<tr>
<td>CSV/TSVFormat</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>100 %</td>
</tr>
<tr>
<td>JSON Format</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0 % / 50%</td>
</tr>
<tr>
<td>Protocol</td>
<td>34</td>
<td>13</td>
<td>0</td>
<td>21</td>
<td>38.24%</td>
</tr>
</tbody>
</table>
Results Test Suite Execution

<table>
<thead>
<tr>
<th>Category</th>
<th>Tests</th>
<th>Passed</th>
<th>Semi-Passed</th>
<th>Failed</th>
<th>Pass Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federated Extensions</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0%</td>
</tr>
<tr>
<td>Entailment Regimes</td>
<td>70</td>
<td>4</td>
<td>0</td>
<td>66</td>
<td>5.71%</td>
</tr>
<tr>
<td>Graphstore Protocol</td>
<td>18</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>11.11%</td>
</tr>
</tbody>
</table>
• Using QLever 22.03.2024
• Of the 416 failed tests:
  174 Query Exceptions (ex. 18 ASK)
  72 Error in result
  94 Update not supported
  60 Index error
Questions?