Performance Comparison of select and construct queries of triplestores
-
on the example of the JVMG project

Tobias Malmsheimer Stuttgart Media University
Japanese Visual Media Graph

- Research project at Stuttgart Media University
- Goal: Build a knowledge graph on the domain of Japanese visual media (manga, anime, computer games)
- Data source: Enthusiast communities on the web
Our Workflow

- **Ingestion workflow**
  - Retrieve data dump / load data from API in proprietary data format
  - Assess data structure and organisation and create an OWL ontology
  - Convert into RDF triples
  - Load into Fuseki database

- **Integration workflow**
  - Match semantically equivalent entities into clusters
  - Merge information from all sources (create a merged entity)
  - Match semantically equivalent entity properties
  - Merge property values (reduce redundancy)
Why we need a Database

- Need to store and access our knowledge graph

- Our frontend expects a sparql-endpoint
  - User can explore our knowledge graph

- Tiny use cases (TUCs)
  - Media researcher use the knowledge graph for their research questions
  - In this way, we can improve the knowledge graph and the supporting tools
vndb abstract data model

Source: https://zenodo.org-record/5506936
How does vndb look like

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>90077</td>
</tr>
<tr>
<td>Trait</td>
<td>2237</td>
</tr>
<tr>
<td>Tag</td>
<td>2053</td>
</tr>
<tr>
<td>Producer</td>
<td>10394</td>
</tr>
<tr>
<td>Staff</td>
<td>21164</td>
</tr>
<tr>
<td>Visual Novel</td>
<td>28190</td>
</tr>
<tr>
<td>Visual Novel Release</td>
<td>71349</td>
</tr>
</tbody>
</table>
Disclaimer

- Just my opinion
- I did not optimize everything for each database
- Interested in the “out of the box” experience
- Therefore critique and bad numbers do not mean the database is bad
Small Query - First Evaluation
Medium Query - First Evaluation

- **Fuseki**
  - select - first: 2.94
  - select - cached: 2.34
  - construct - first: 10.71
  - construct - cached: 9.65

- **Blazegraph**
  - select - first: 10.26
  - select - cached: 10.83
  - construct - first: 13.7
  - construct - cached: 12.51

- **Virtuoso**
  - select - first: 0.7
  - select - cached: 0.72
  - construct - first: 13.3
  - construct - cached: 13.8

- **GraphDB**
  - select - first: 1.35
  - select - cached: 1.6
  - construct - first: 14.07
  - construct - cached: 11.66
Performance - Queries

- 2 different queries
  - Usual Queries we often use
- Small Query
  - Character page from Phorni
  - 96 triple
  - Small character page
- Medium Query
  - Overview of all vndb Characters
  - ~180k triple
  - Name of every entity which has type Character
Performance - CPU

- Python
  - Sparqlwrapper and rdflib for sparql endpoints
- Complete round trip
  - Executing sparql query
  - Serializing result
  - Average of 5 Queries, after warm-up queries
- Two kinds of queries
  - Select, assumption: better optimized
  - Construct
- Serialization methods
  - XML, JSON, CSV, JSONLD, Turtle, n-triples
Query - Select

```sparql
PREFIX label: <http://www.w3.org/2000/01/rdf-schema#label>
PREFIX graph_label: <http://mediagraph.link/jvmg/ont/shortLabel>
SELECT
WHERE {
  GRAPH ?graph {
  }
  OPTIONAL { ?graph graph_label: ?graphLabel }
  OPTIONAL { ?object label: ?objectLabel }
  OPTIONAL { ?predicate label: ?predicateLabel }
  UNION {
    GRAPH ?graph {
    }
    OPTIONAL { ?graph graph_label: ?graphLabel }
    OPTIONAL { ?subject label: ?subjectLabel }
    OPTIONAL { ?predicate label: ?predicateLabel }
  }
}
```
```turtle
PREFIX label: <http://www.w3.org/2000/01/rdf-schema#label>
PREFIX graph label: <http://mediagraph.link/jvmg/ont/shortLabel>

CONSTRUCT {
}
WHERE {

  GRAPH ?graph {
  }
  OPTIONAL { ?graph graph_label: ?graphLabel }
  OPTIONAL { ?object label: ?objectLabel }
  OPTIONAL { ?predicate label: ?predicateLabel }
  UNION {
    GRAPH ?graph {
    }
    OPTIONAL { ?graph graph_label: ?graphLabel }
    OPTIONAL { ?subject label: ?subjectLabel }
    OPTIONAL { ?predicate label: ?predicateLabel }
  }
```

---

**Query - Construct**

1. PREFIX label: <http://www.w3.org/2000/01/rdf-schema#label>
2. PREFIX graph label: <http://mediagraph.link/jvmg/ont/shortLabel>
3. CONSTRUCT {
}
   WHERE {
     GRAPH ?graph {
     }
     OPTIONAL { ?graph graph_label: ?graphLabel }
     OPTIONAL { ?object label: ?objectLabel }
     OPTIONAL { ?predicate label: ?predicateLabel }
     UNION {
       GRAPH ?graph {
       }
       OPTIONAL { ?graph graph_label: ?graphLabel }
       OPTIONAL { ?subject label: ?subjectLabel }
       OPTIONAL { ?predicate label: ?predicateLabel }
     }
   }
Fuseki - Small Query

- XML - Select: 0.011
- JSON - Select: 0.005
- CSV - Select: 0.006
- Turtle - Select
- n-triples - Select
- XML - Construct: 0.012
- JSON - Construct
- CSV - Construct
- JSONLD - Construct: 0.011
- Turtle - Construct: 0.008
- n-triples - Construct
Fuseki - Medium Query

- XML - Select: 11.88
- JSON - Select: 1.68
- CSV - Select: 2.14
- Turtle - Select:  
- n-triples - Select:  
- XML - Construct: 13.83
- JSON - Construct:  
- CSV - Construct:  
- JSONLD - Construct: 8.61
- Turtle - Construct: 1.2
- n-triples - Construct:  

Bar chart showing performance metrics for different query types.
Blazegraph - Small Query

- XML - Select: 1.58
- JSON - Select: 1.69
- CSV - Select: 1.74
- Turtle - Select: 1.68
- n-triples - Select: 1.68
- XML - Construct: 1.74
- JSON - Construct: 1.68
- CSV - Construct: 1.68
- JSONLD - Construct: 1.68
- Turtle - Construct: 1.68
- n-triples - Construct: 1.68
Virtuoso - Small Query

- XML - Select: 0.0094
- JSON - Select: 0.003
- CSV - Select: 0.0028
- Turtle - Select: 0.0024
- n-triples - Select: 0.0035
- XML - Construct: 0.012
- JSON - Construct: 0.004
- CSV - Construct: 0.0039
- JSONLD - Construct: 0.0065
- Turtle - Construct: 0.0036
- n-triples - Construct: 0.0032
Virtuoso - Medium Query

- XML - Select: 11.82
- JSON - Select: 2.32
- CSV - Select: 0.81
- Turtle - Select: 0.54
- n-triples - Select: 0.54
- XML - Construct: 13
- JSON - Construct: 1.56
- CSV - Construct: 1.59
- JSONLD - Construct: 5.8
- Turtle - Construct: 1.33
- n-triples - Construct: 1.34
GraphDB - Small Query
GraphDB - Medium Query

- XML - Select: 14.4
- JSON - Select: 1.78
- CSV - Select: 1.31
- Turtle - Select: 
- n-triples - Select: 
- XML - Construct: 11.33
- JSON - Construct: 
- CSV - Construct: 
- JSONLD - Construct: 6.26
- Turtle - Construct: 1.8
- n-triples - Construct: 1.68
Small Query

![Bar Chart]

- **Fuseki - Select**: 0.005
- **Fuseki - Construct**: 0.008
- **Virtuoso - Select**: 0.0024
- **Virtuoso - Construct**: 0.0032
- **GraphDB - Select**: 0.005
- **GraphDB - Construct**: 0.006
Medium Query
Conclusion

- Select queries tend to be a bit faster
- Huge performance difference when using different serialization formats
  - But: not all serialization formats allow all features (quads in fuseki)
- Your mileage may vary: test combinations of query types and serialization formats
Questions?