How Caching Improves Efficiency and Result Completeness for Querying Linked Data

Olaf Hartig
http://olafhartig.de/foaf.rdf#olaf
@olafhartig

Database and Information Systems Research Group
Humboldt-Universität zu Berlin
Can we query the Web of Data as if it were a single, giant database?

Our approach: **Link Traversal Based Query Execution**

[ISWC'09]
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

Query:

- `http://bob.name`
- `?acq` knows `?project`
- `?name` ?prjName

Query-printing:

- `http://bob.name`

- `?acq` knows `?project`
- `?name` ?prjName

query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

Query

```
http://bob.name

?acq

knows

?prjName

?prj

name

project
```

"Descriptor object"
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - **Evaluate parts** of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

• Intertwine query evaluation with traversal of data links

• We alternate between:
  • Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  • Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

• Intertwine query evaluation with traversal of data links

• We alternate between:
  • Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  • Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

• Intertwine query evaluation with traversal of data links

• We alternate between:
  • Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  • Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links
- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Main Idea

- Intertwine query evaluation with traversal of data links

- We alternate between:
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset
Characteristics

- **Link traversal based query execution:**
  - Evaluation on a continuously augmented dataset
  - Discovery of potentially relevant data during execution
  - Discovery driven by intermediate solutions

- **Main advantage:**
  - No need to know all data sources in advance

- **Limitations:**
  - Query has to contain a URI as a starting point
  - Ignores data that is not *reachable* by the query execution

*formal definition in the paper*
The Issue

Query

?acq knows ?i

interest

http://bob.name

?iLabel

query-local dataset
The Issue

Query

http://bob.name

dataset

query-local

http://bob.name

Olaf Hartig - How Caching Improves Efficiency and Result Completeness for Querying Linked Data
The Issue

Query

?acq knows interest

?i knows label

http://bob.name

?iLabel

http://alice.name

http://bob.name

query local dataset
The Issue

Query

<table>
<thead>
<tr>
<th>?acq</th>
<th>?i</th>
<th>?iLabel</th>
</tr>
</thead>
</table>

The Issue

http://bob.name

http://alice.name

query-local dataset
The Issue
Reusing the Query-Local Dataset

Query

?acq knows interest

http://bob.name

?i

label

?iLabel

Query

http://bob.name

?prjName

?acq knows project

?prj

que local dataset

que local dataset
Reusing the Query-Local Dataset
Reusing the Query-Local Dataset

Query

?acq

http://bob.name

?i

?iLabel

http://alice.name

?acq

http://alice.name

http://bob.name

?acq

?prj

?prjName

project

name

knows

interest

label

knows

query-local dataset

http://bob.name

http://alice.name

http://alice.name

http://bob.name

Olaf Hartig - How Caching Improves Efficiency and Result Completeness for Querying Linked Data
Hypothesis

Re-using the query-local dataset (a.k.a. data caching) may benefit query performance + result completeness
Contributions

- Systematic analysis of the impact of data caching
  - Theoretical foundation
  - Conceptual analysis
  - Empirical evaluation of the potential impact

*see paper

- Out of scope: Caching strategies (replacement, invalidation)
Experiment – Scenario

FOAF Letter

Welcome, Olaf Hartig!

You have no new incoming contacts.

You claim to know these 3 people, but according to their FOAF profile, they do not seem to know you:

- Juan Sequeda
- Christian Bizer
- Michael Hausenblas

View potential new contacts (ALPHA warning: might take ages)

There are no upcoming birthdays.

Improve your profile by adding the following information:

- schoolHomepage: A homepage of a school attended by the person
- myersBriggs: A Myers Briggs (MBTI) personality classification

Information about the distributed social network of FOAF profiles

- 5 types of queries

Experiment Setup:

- 23 persons
- Sequential use
- 115 queries
Experiment – Complete Sequence

- **no reuse experiment:**
  - No data caching

- **given order experiment**
  - Reuse of the query-local dataset for the complete sequence of all 115 queries

- **Hit rate:**
  - look-ups answered from cache all look-up requests
Experiment – Complete Sequence

**no reuse experiment:**
- No data caching

**given order experiment**
- Reuse of the query-local dataset for the complete sequence of all 115 queries

**Hit rate:**
look-ups answered from cache all look-up requests
Experiment – Complete Sequence

- **ContactInfoPhillipe** (Query No. 36)
- **UnsetPropsPhillipe** (Query No. 37)
- **2ndDegree1Phillipe** (Query No. 38)
- **2ndDegree2Phillipe** (Query No. 39)
- **IncomingPhillipe** (Query No. 40)

**hit rate**

**number of query results**

<table>
<thead>
<tr>
<th>hit rate</th>
<th>0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactInfoPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnsetPropsPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree1Phillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree2Phillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IncomingPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**query execution time (in seconds)**

<table>
<thead>
<tr>
<th>number of query results</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactInfoPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnsetPropsPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree1Phillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree2Phillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IncomingPhillipe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experiment – Complete Sequence

- **ContactInfoPhillipe** (Query No. 36)
- **UnsetPropsPhillipe** (Query No. 37)
- **2ndDegree1Phillipe** (Query No. 38)
- **2ndDegree2Phillipe** (Query No. 39)
- **IncomingPhillipe** (Query No. 40)

- **hit rate**
- **number of query results**
- **query execution time (in seconds)**

- **no reuse**
- **given order**
Summary

• Contributions:
  • Theoretical foundation
  • Conceptual analysis
  • Empirical evaluation

• Main findings:
  • Additional results possible (for semantically similar queries)
  • Impact on performance may be positive but also negative

• Future work:
  • Analysis of caching strategies in our context
  • Main issue: invalidation
Backup Slides
Contributions

• Theoretical foundation (extension of the original definition)
  • Reachability by a $D_{\text{seed}}$-initialized execution of a BGP query $b$
  • $D_{\text{seed}}$-dependent solution for a BGP query $b$
  • Reachability $R(B)$ for a serial execution of $B = b_1, \ldots, b_n$
  → Each solution for $b_{\text{cur}}$ is also $R(B)$-dependent solution for $b_{\text{cur}}$

• Conceptual analysis of the impact of data caching
  • Performance factor: $\rho(b_{\text{cur}}, B) = c(b_{\text{cur}}, []) - c(b_{\text{cur}}, B)$
  • Serendipity factor: $s(b_{\text{cur}}, B) = b(b_{\text{cur}}, B) - b(b_{\text{cur}}, [])$

• Empirical verification of the potential impact

• Out of scope: Caching strategies (replacement, invalidation)
Query Template Contact

```
SELECT * WHERE {
  OPTIONAL { ?p foaf:name ?name }
  OPTIONAL { ?p foaf:firstName ?firstName }
  OPTIONAL { ?p foaf:givenName ?givenName }
  OPTIONAL { ?p foaf:givenname ?givenname }
  OPTIONAL { ?p foaf:familyName ?familyName }
  OPTIONAL { ?p foaf:family_name ?family_name }
  OPTIONAL { ?p foaf:lastName ?lastName }
  OPTIONAL { ?p foaf:surname ?surname }
  OPTIONAL { ?p foaf:birthday ?birthday }
  OPTIONAL { ?p foaf:img ?img }
  OPTIONAL { ?p foaf:phone ?phone }
  OPTIONAL { ?p foaf:aimChatID ?aimChatID }
  OPTIONAL { ?p foaf:icqChatID ?icqChatID }
  OPTIONAL { ?p foaf:jabberID ?jabberID }
  OPTIONAL { ?p foaf:msnChatID ?msnChatID }
  OPTIONAL { ?p foaf:skypeID ?skypeID }
  OPTIONAL { ?p foaf:yahooChatID ?yahooChatID }
}
```
SELECT DISTINCT ?result ?resultLabel WHERE 
{
  ?result rdfs:isDefinedBy <http://xmlns.com/foaf/0.1/> .
  ?result rdfs:domain foaf:Person .

  OPTIONAL { <PERSON> ?result ?var0 } 
  FILTER ( !bound(?var0) )

  <PERSON> foaf:knows ?var2 .
  ?result rdfs:label ?resultLabel .
}
ORDER BY ?var1
SELECT DISTINCT ?result WHERE {
    ?result foaf:knows <PERSON> .

    OPTIONAL {
        FILTER ( <PERSON> = ?var1 )
        <PERSON> foaf:knows ?result .
    }
    FILTER ( !bound(?var1) )
}
SELECT DISTINCT ?result WHERE {
  FILTER ( ?p1 != ?p2 )

  FILTER ( <PERSON> != ?result )

  OPTIONAL {
    <PERSON> ?knows ?result .
    FILTER ( ?knows = foaf:knows )
  }
  FILTER ( !bound(?knows) )
}
Query Template 2ndDegree2

```sparql
SELECT DISTINCT ?result WHERE {
  FILTER ( ?p1 != ?p2 )

  FILTER ( <PERSON> != ?result )

  OPTIONAL {
    <PERSON> ?knows ?result .
    FILTER ( ?knows = foaf:knows )
  }
  FILTER ( !bound(?knows) )
}
```
Experiment – Single Query

- **no reuse experiment:**
  - No data caching

- **upper bound experiment**
  - Reuse of query-local dataset for 3 executions of each query
  - Third execution measured

- Hit rate:
  - look-ups answered from cache
  - all look-up requests
Experiment – Single Query

- **no reuse experiment:**
  - No data caching

- **upper bound experiment**
  - Reuse of query-local dataset for 3 executions of each query
  - Third execution measured

**Hit rate:**
look-ups answered from cache
all look-up requests
Experiment – Single Query

<table>
<thead>
<tr>
<th>Query</th>
<th>Hit Rate</th>
<th>Upper Bound</th>
<th>Number of Query Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactInfoPhillipe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Query No. 36)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnsetPropsPhillipe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Query No. 37)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree1Phillipe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Query No. 38)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2ndDegree2Phillipe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Query No. 39)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IncomingPhillipe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Query No. 40)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query execution time (in seconds)
Experiment – Single Query

<table>
<thead>
<tr>
<th>Query</th>
<th>Hit Rate</th>
<th>Number of Query Results</th>
<th>Query Execution Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactInfoPhillipe (Query No. 36)</td>
<td>0.4</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>UnsetPropsPhillipe (Query No. 37)</td>
<td>0.6</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>2ndDegree1Phillipe (Query No. 38)</td>
<td>0.8</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>2ndDegree2Phillipe (Query No. 39)</td>
<td>0.2</td>
<td>15</td>
<td>60</td>
</tr>
<tr>
<td>IncomingPhillipe (Query No. 40)</td>
<td>0.0</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>
# Experiment – Single Query

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Avg.(^1) number of Query Results (std.dev.)</th>
<th>Average(^1) Hit Rate (std.dev.)</th>
<th>Avg.(^1) query Execution Time (std.dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no reuse</td>
<td>4.983 (11.658)</td>
<td>0.576 (0.182)</td>
<td>30.036 s (46.708)</td>
</tr>
<tr>
<td>upper bound</td>
<td>5.070 (11.813)</td>
<td>0.996 (0.017)</td>
<td>1.943 s (11.375)</td>
</tr>
</tbody>
</table>

\(^1\) Averaged over all 115 queries

- **In the ideal case for** \( B_{upper} = [ b_{cur}, b_{cur} ] \):
  - \( p_{upper}( b_{cur}, B_{upper} ) = c( b_{cur}, [ ] ) - c( b_{cur}, B_{upper} ) = c( b_{cur}, [ ] ) \)
  - \( s_{upper}( b_{cur}, B_{upper} ) = b( b_{cur}, B_{upper} ) - b( b_{cur}, [ ] ) = 0 \)
## Experiment – Single Query

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Avg.¹ number of Query Results (std.dev.)</th>
<th>Average¹ Hit Rate (std.dev.)</th>
<th>Avg.¹ query Execution Time (std.dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no reuse</td>
<td>4.983 (11.658)</td>
<td>0.576 (0.182)</td>
<td>30.036 s (46.708)</td>
</tr>
<tr>
<td>upper bound</td>
<td>5.070 (11.813)</td>
<td>0.996 (0.017)</td>
<td>1.943 s (11.375)</td>
</tr>
</tbody>
</table>

¹ Averaged over all 115 queries

- **Summary (measurement errors aside):**
  - Same number of query results
  - Significant improvements in query performance
Experiment – Complete Sequence

- **given order experiment:**
- Reuse of the query-local dataset for the complete sequence of all 115 queries
Experiment – Complete Sequence

- **given order** experiment:
  - Reuse of the query-local dataset for the complete sequence of all 115 queries
Experiment – Complete Sequence

- ContactInfoPhillipe
- UnsetPropsPhillipe
- 2ndDegree1Phillipe
- 2ndDegree2Phillipe
- IncomingPhillipe

![Bar charts showing hit rate and number of query results for different queries.]

- No reuse
- Upper bound
- Given order

![Graph showing query execution time in seconds.]

Olaf Hartig - How Caching Improves Efficiency and Result Completeness for Querying Linked Data
Experiment – Complete Sequence

- **no reuse**
- **upper bound**
- **given order**

<table>
<thead>
<tr>
<th>Query</th>
<th>Hit Rate</th>
<th>Number of Query Results</th>
<th>Query Execution Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactInfoPhillipe</td>
<td>0.6</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>UnsetPropsPhillipe</td>
<td>0.8</td>
<td>25</td>
<td>60</td>
</tr>
<tr>
<td>2ndDegree1Phillipe</td>
<td>0.4</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>2ndDegree2Phillipe</td>
<td>0.6</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>IncomingPhillipe</td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

---

Olaf Hartig - How Caching Improves Efficiency and Result Completeness for Querying Linked Data
Experiment - Complete Sequence

\[ B_{\text{given order}} = [q_1, \ldots, q_{38}] \]

\[ s(q_{39}, B_{\text{given order}}) = b(q_{39}, B_{\text{given order}}) - b(q_{39}, []) \]

\[ = 9 - 1 \]

\[ = 8 \]
$B_{\text{given order}} = [q_1, \ldots, q_{38}]$

$p'(q_{39}, B_{\text{given order}}) = c'(q_{39}, []) - c'(q_{39}, B_{\text{given order}})$

$= 31.48 \text{ s} \ - \ 68.64 \text{ s}

= -37.16 \text{ s}$
### Experiment – Complete Sequence

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Avg.(^1) number of Query Results (std.dev.)</th>
<th>Average(^1) Hit Rate (std.dev.)</th>
<th>Avg.(^1) query Execution Time (std.dev.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>no reuse</td>
<td>4.983 (11.658)</td>
<td>0.576 (0.182)</td>
<td>30.036 s (46.708)</td>
</tr>
<tr>
<td>upper bound</td>
<td>5.070 (11.813)</td>
<td>0.996 (0.017)</td>
<td>1.943 s (11.375)</td>
</tr>
<tr>
<td>given order</td>
<td>6.878 (12.158)</td>
<td>0.932 (0.139)</td>
<td>39.845 s (145.898)</td>
</tr>
</tbody>
</table>

\(^1\) Averaged over all 115 queries

- **Summary:**
  - Data cache may provide for additional query results
  - Impact on performance may be positive but also negative

Olaf Hartig - How Caching Improves Efficiency and Result Completeness for Querying Linked Data
### Experiment – Complete Sequence

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Avg.(^1) number of Query Results</th>
<th>Average(^1) Hit Rate</th>
<th>Avg.(^1) query Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(std.dev.)</td>
<td>(std.dev.)</td>
<td>(std.dev.)</td>
</tr>
<tr>
<td>no reuse</td>
<td>4.983</td>
<td>0.576</td>
<td>30.036 s</td>
</tr>
<tr>
<td></td>
<td>(11.658)</td>
<td>(0.182)</td>
<td>(46.708)</td>
</tr>
<tr>
<td>upper bound</td>
<td>5.070</td>
<td>0.996</td>
<td>1.943 s</td>
</tr>
<tr>
<td></td>
<td>(11.813)</td>
<td>(0.017)</td>
<td>(11.375)</td>
</tr>
<tr>
<td>given order</td>
<td>6.878</td>
<td>0.932</td>
<td>39.845 s</td>
</tr>
<tr>
<td></td>
<td>(12.158)</td>
<td>(0.139)</td>
<td>(145.898)</td>
</tr>
<tr>
<td>random orders</td>
<td>6.652</td>
<td>0.954</td>
<td>36.994 s</td>
</tr>
<tr>
<td></td>
<td>(11.966)</td>
<td>(0.036)</td>
<td>(118.700)</td>
</tr>
</tbody>
</table>

- Executing the query sequence in a random order results in measurements similar to the given order.
These slides have been created by
Olaf Hartig

http://olafhartig.de

This work is licensed under a
Creative Commons Attribution-Share Alike 3.0 License
(http://creativecommons.org/licenses/by-sa/3.0/)