dbis
INSTITUT FÜR INFORMATIK
HUMBOLDT-UNIVERSITÄT ZU BERLIN

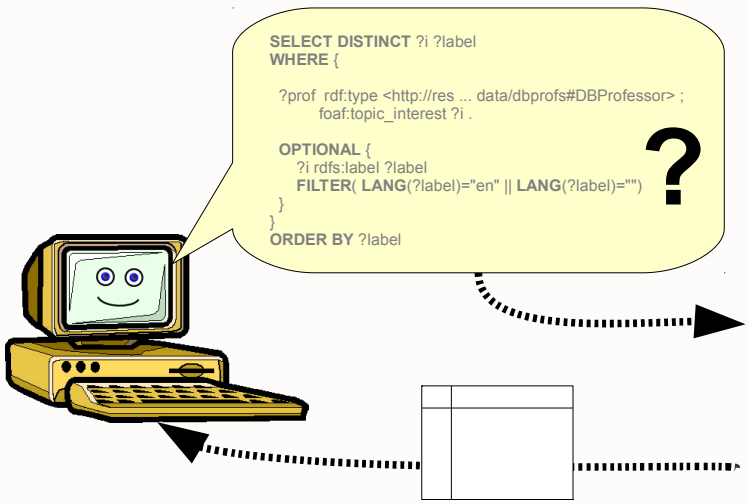
HUMBOLDT-UNIVERSITÄT ZU BERLIN

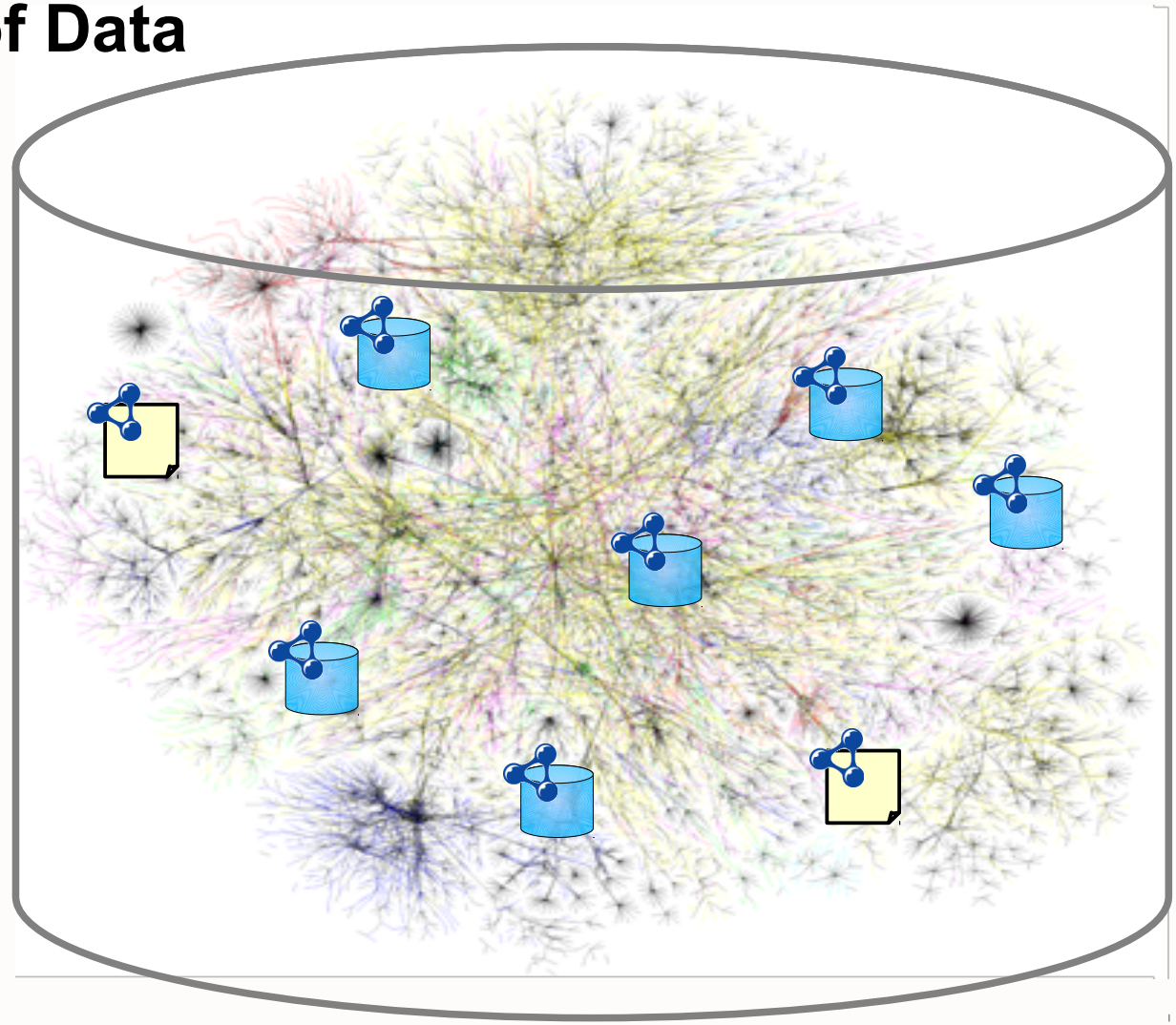
How Caching Improves Efficiency and Result Completeness for Querying Linked Data

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 of it were a single, giant database?



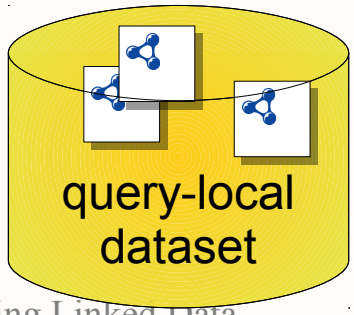


Our approach: Link Traversal Based Query Execution

[ISWC'09]

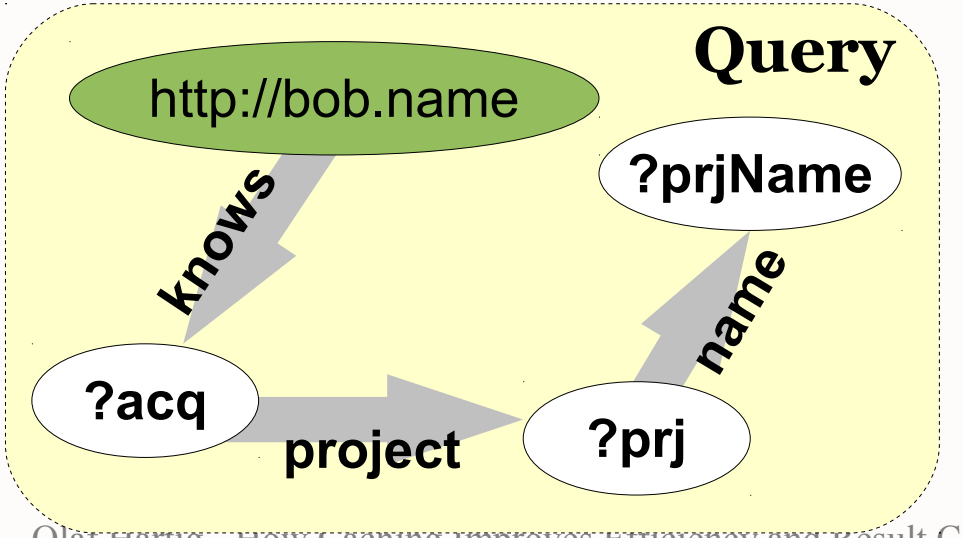
Main Idea

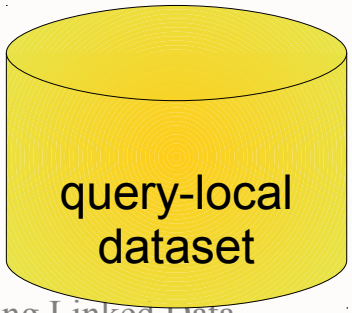
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset



Main Idea

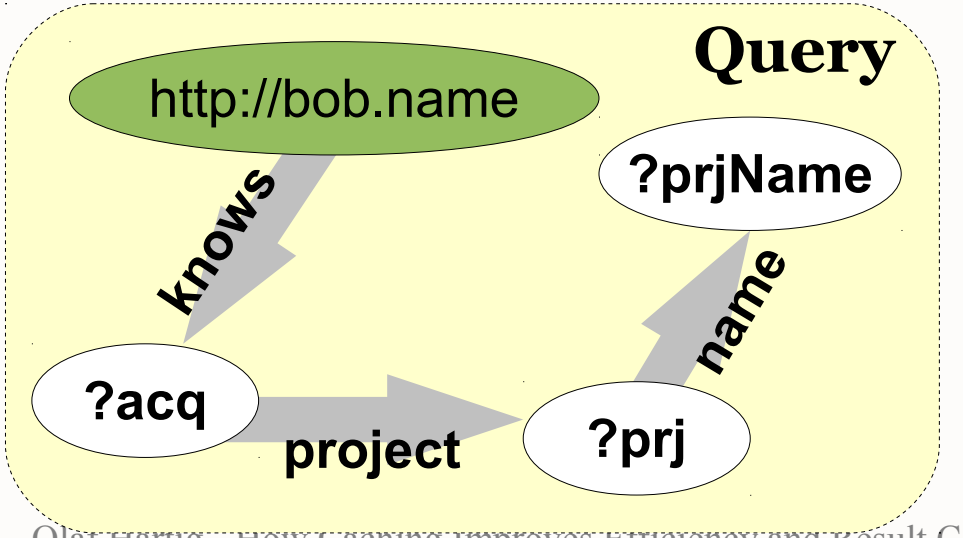
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

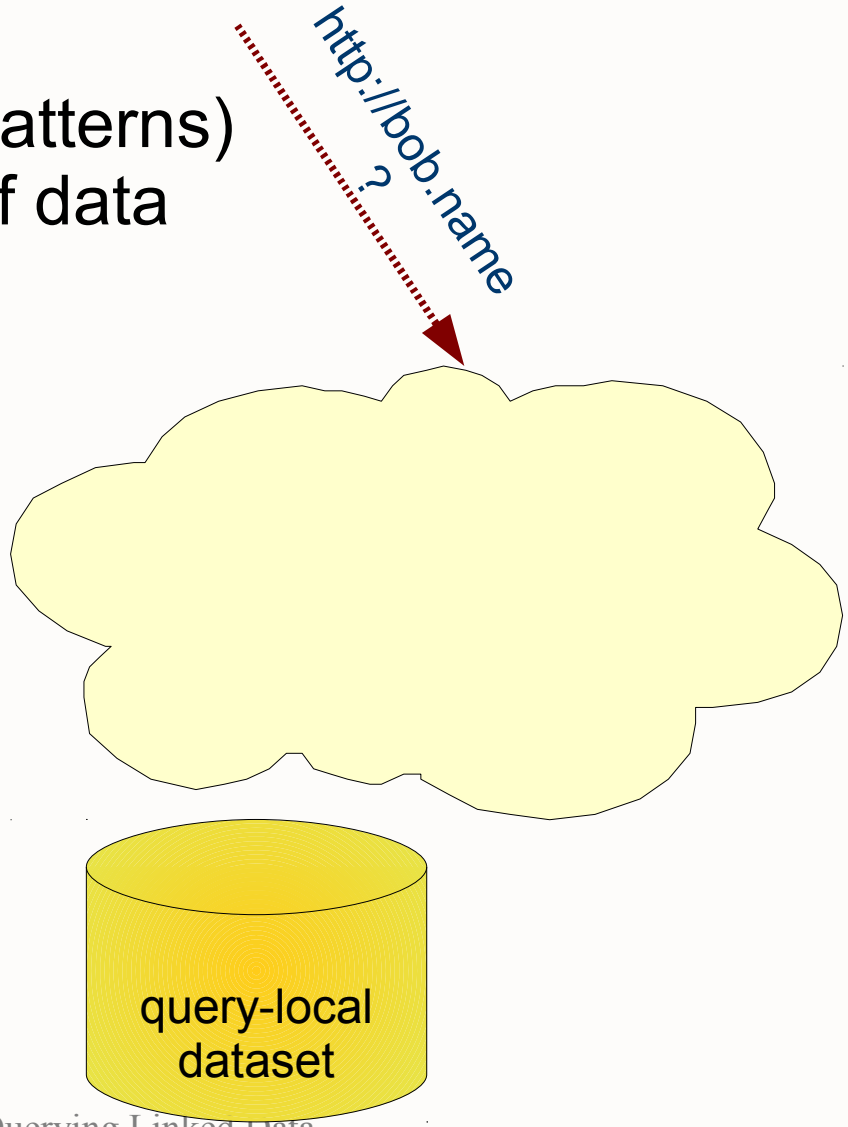




Main Idea

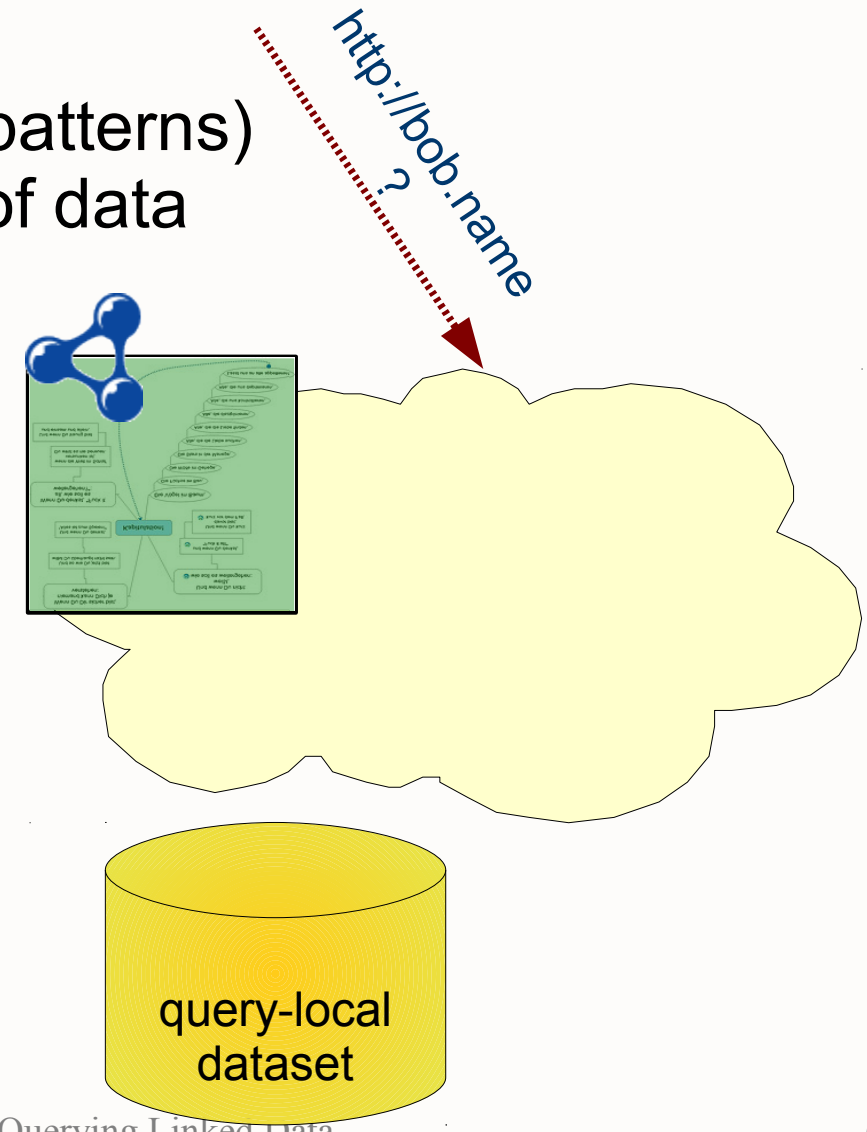
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

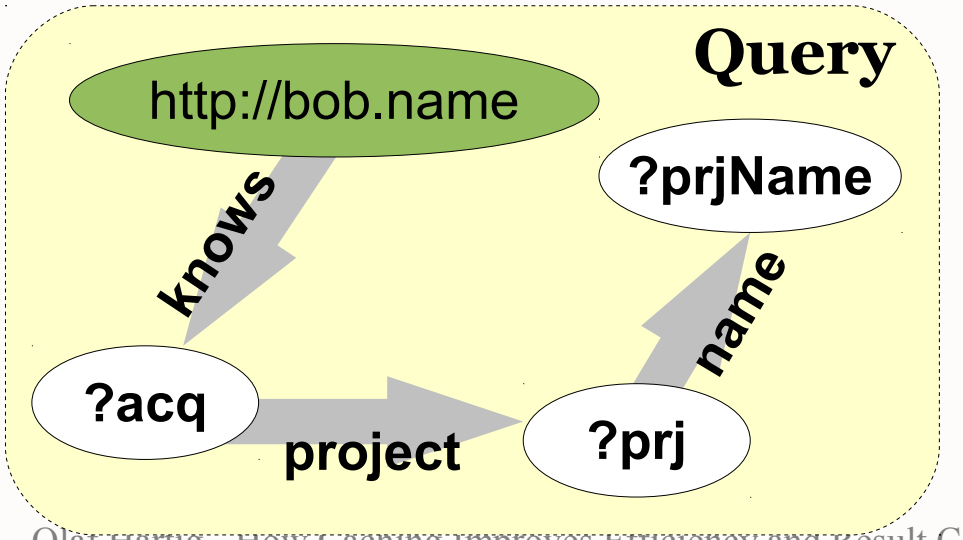




Main Idea

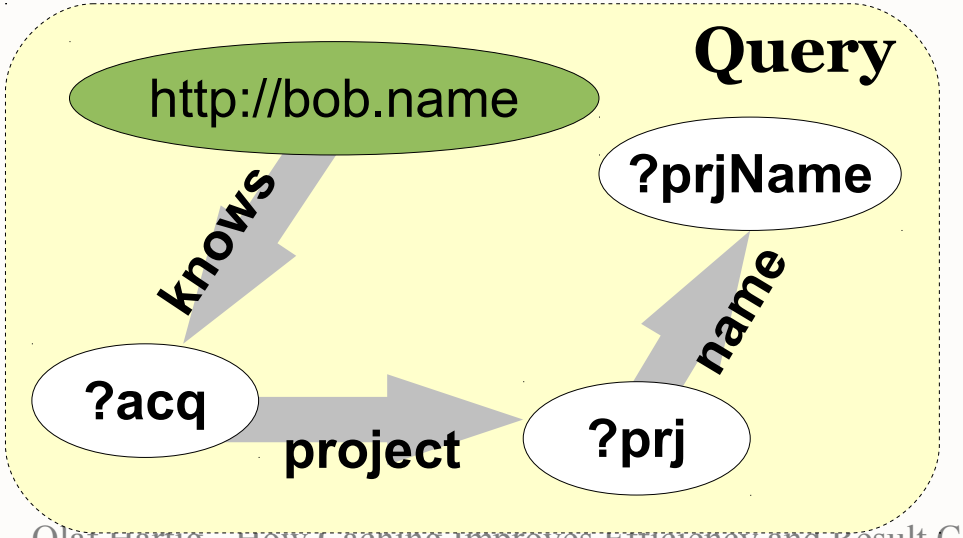
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

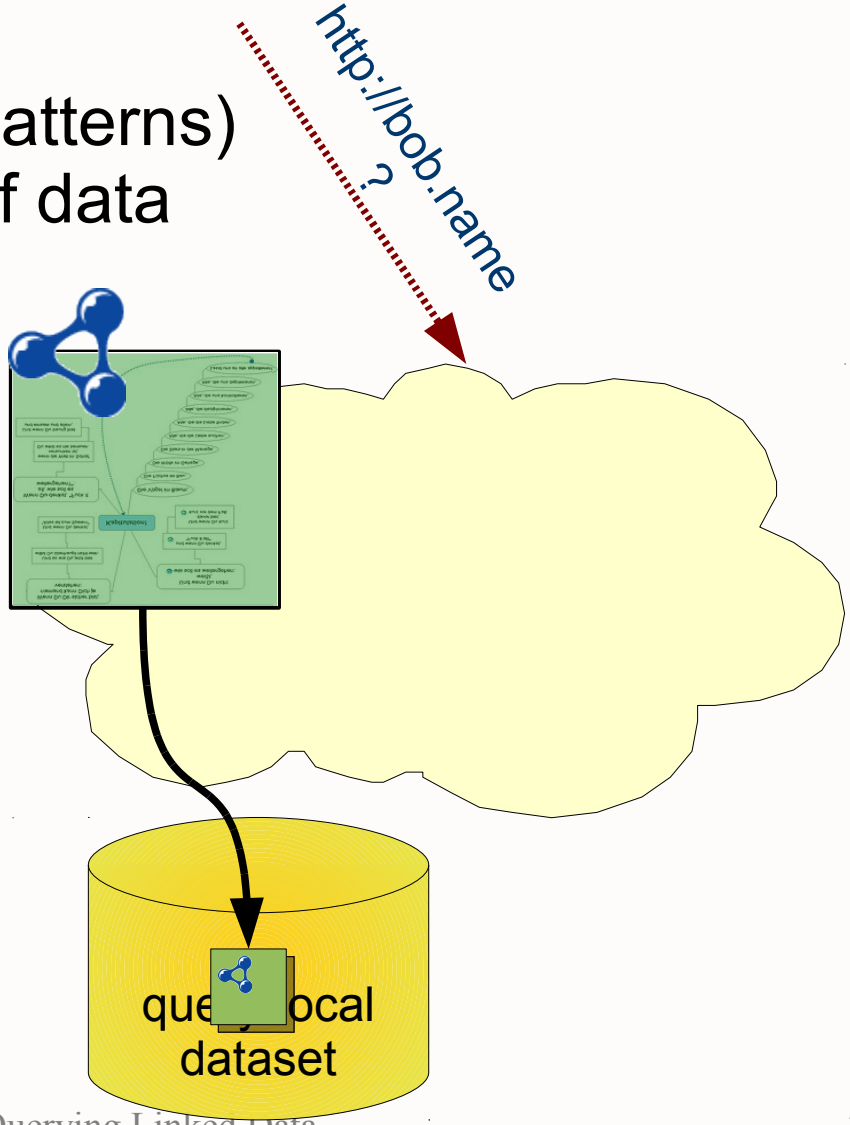




Main Idea

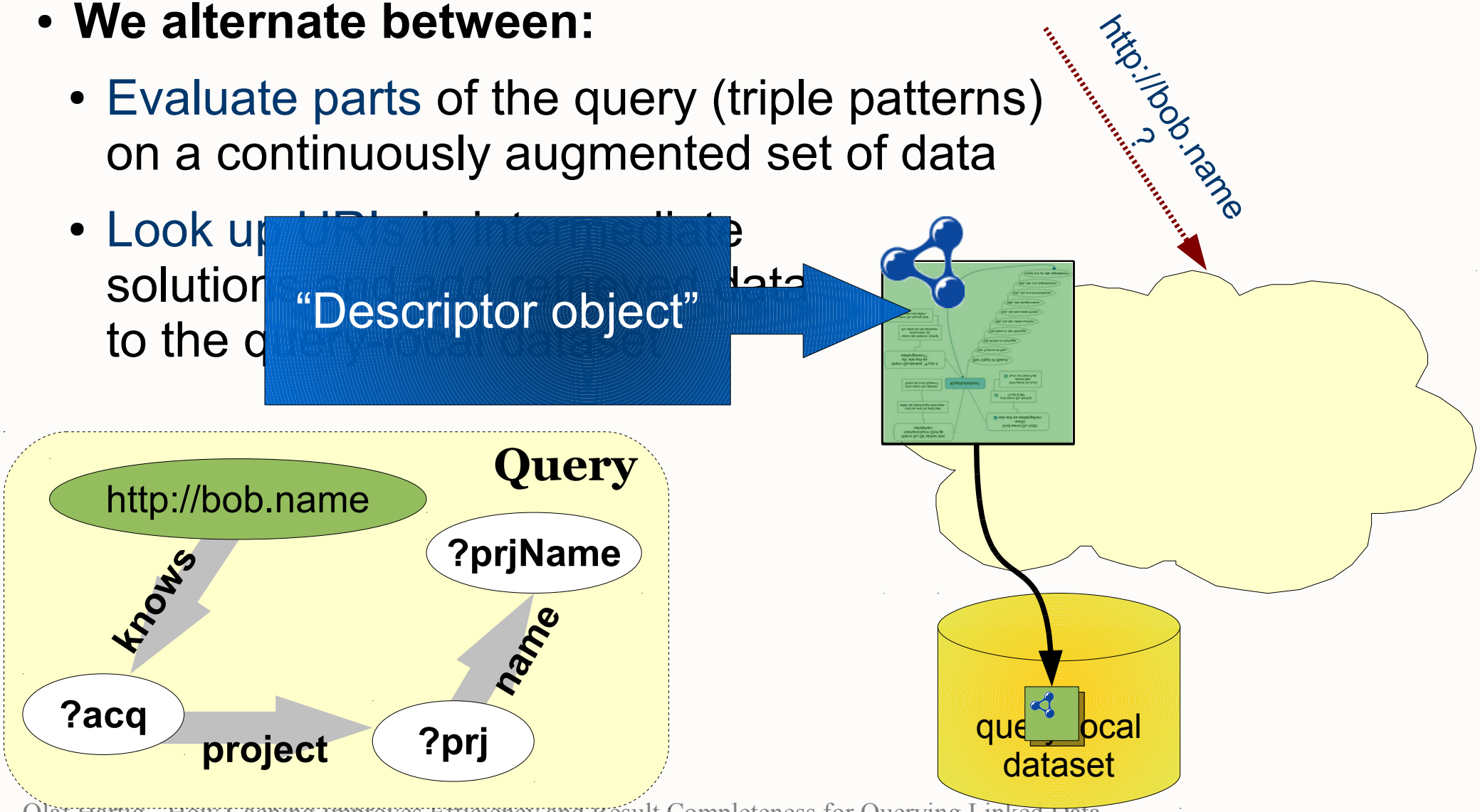
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset





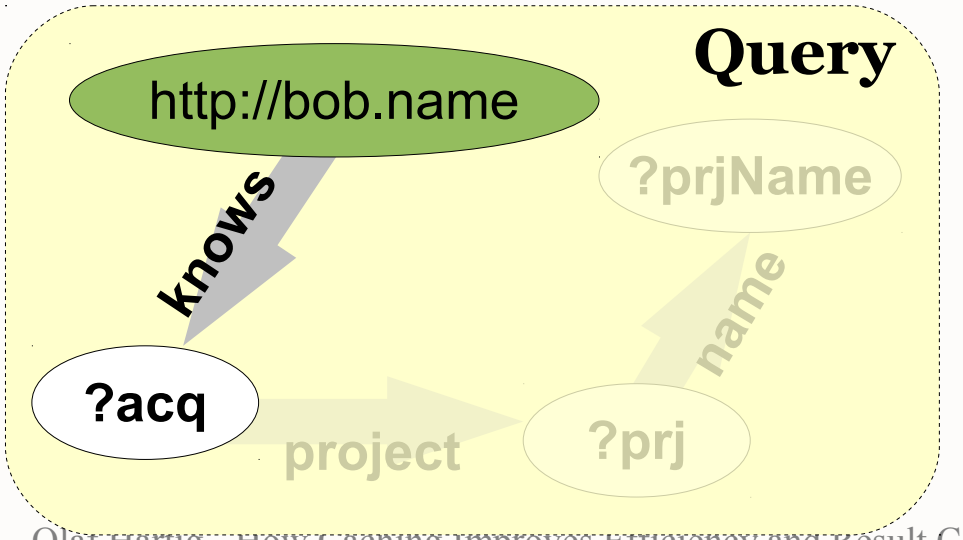
Main Idea

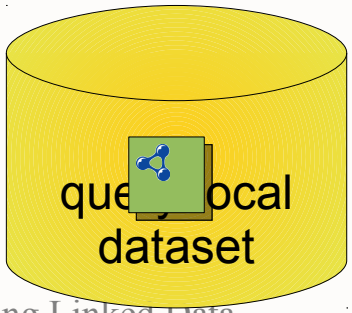
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solution [illegible] data to the q[illegible]



Main Idea

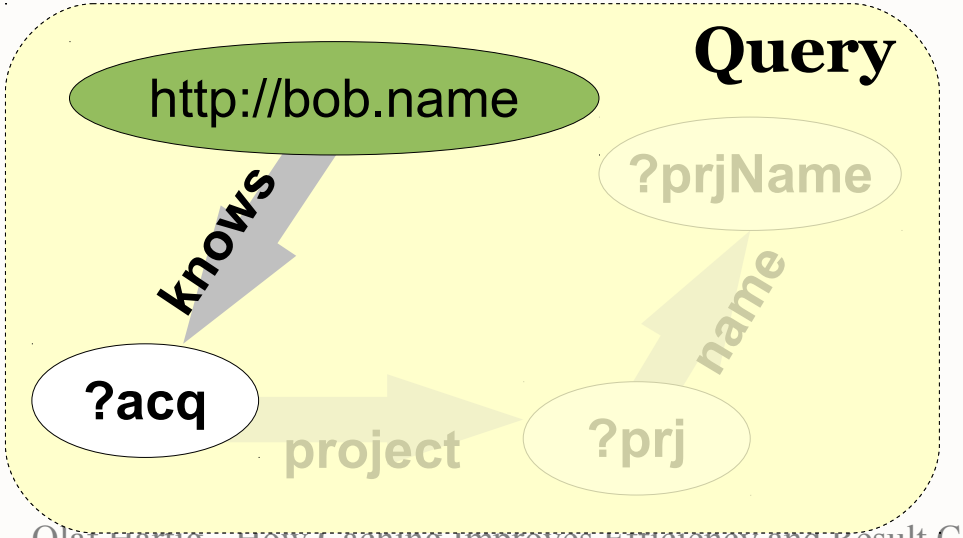
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

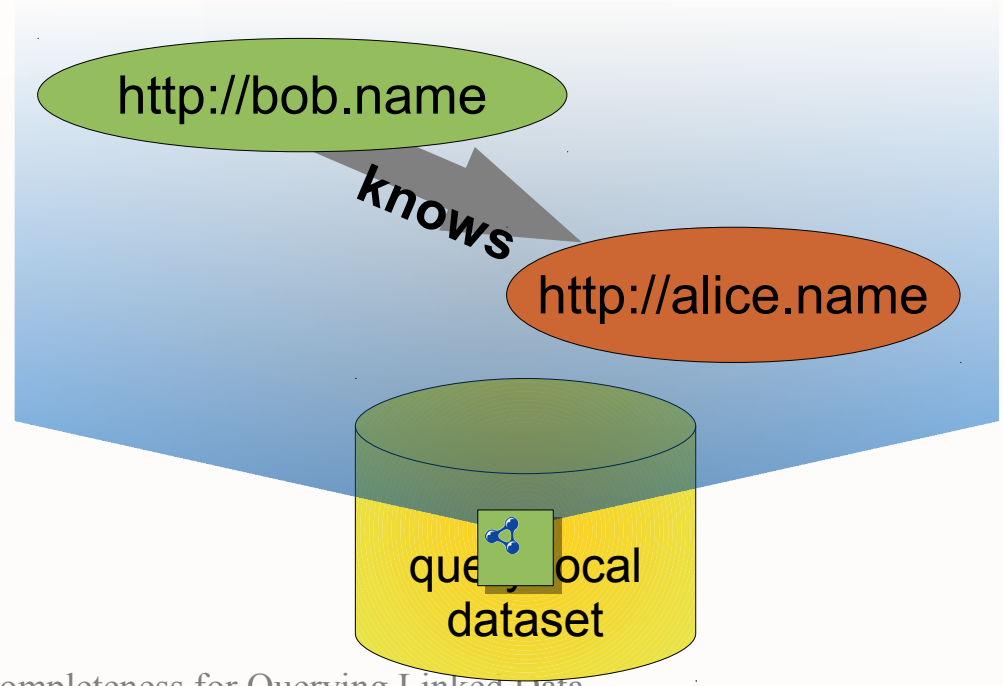




Main Idea

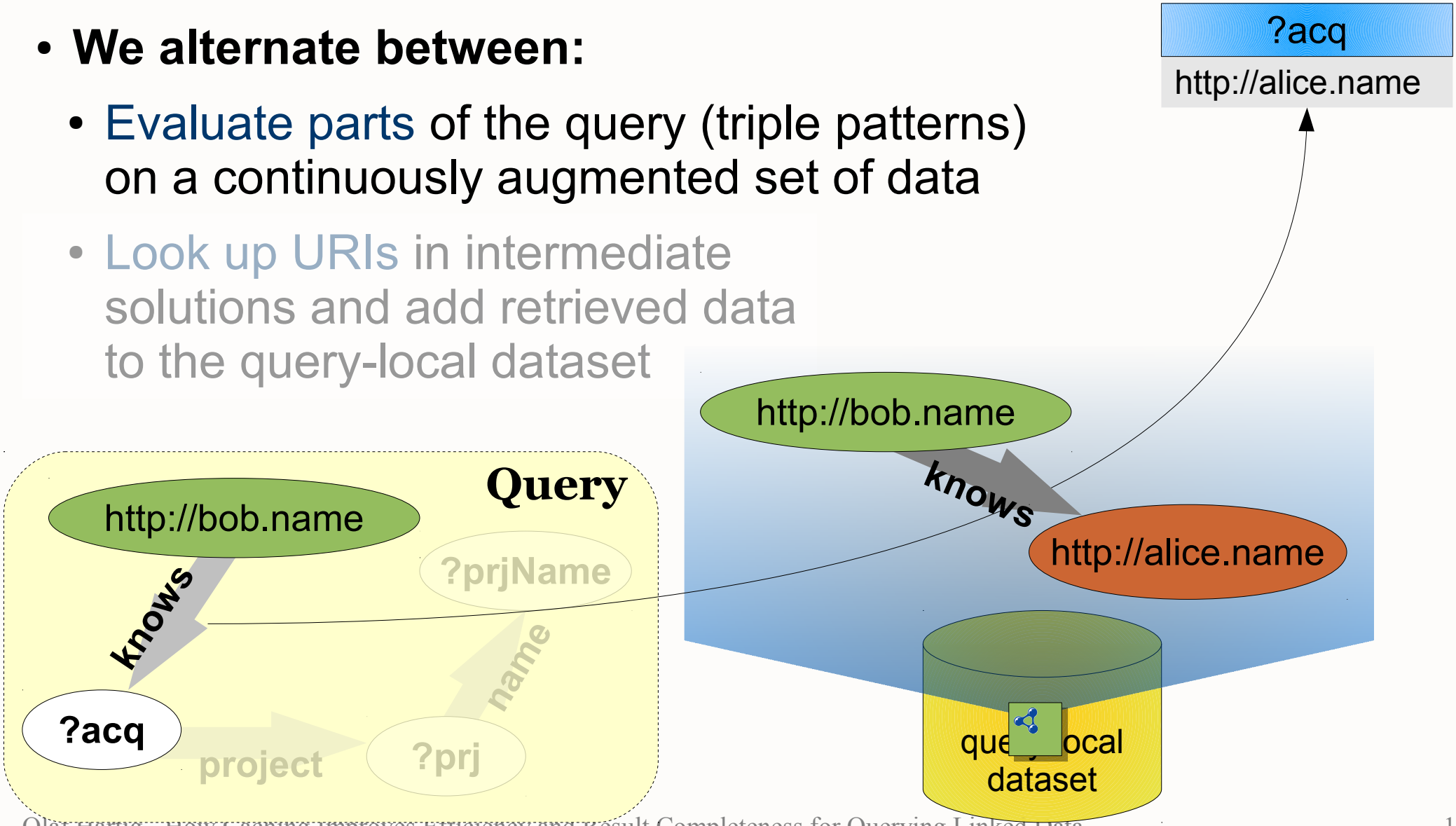
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset





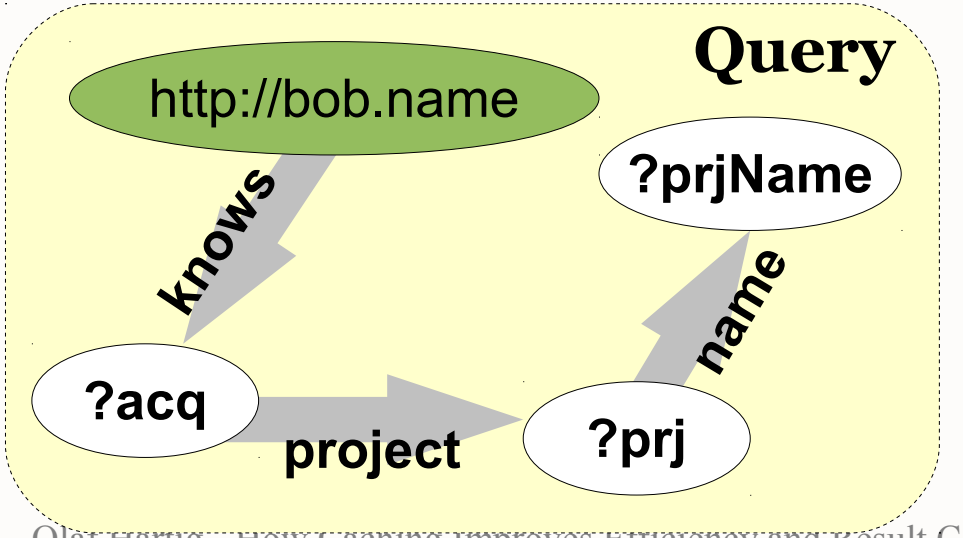
Main Idea

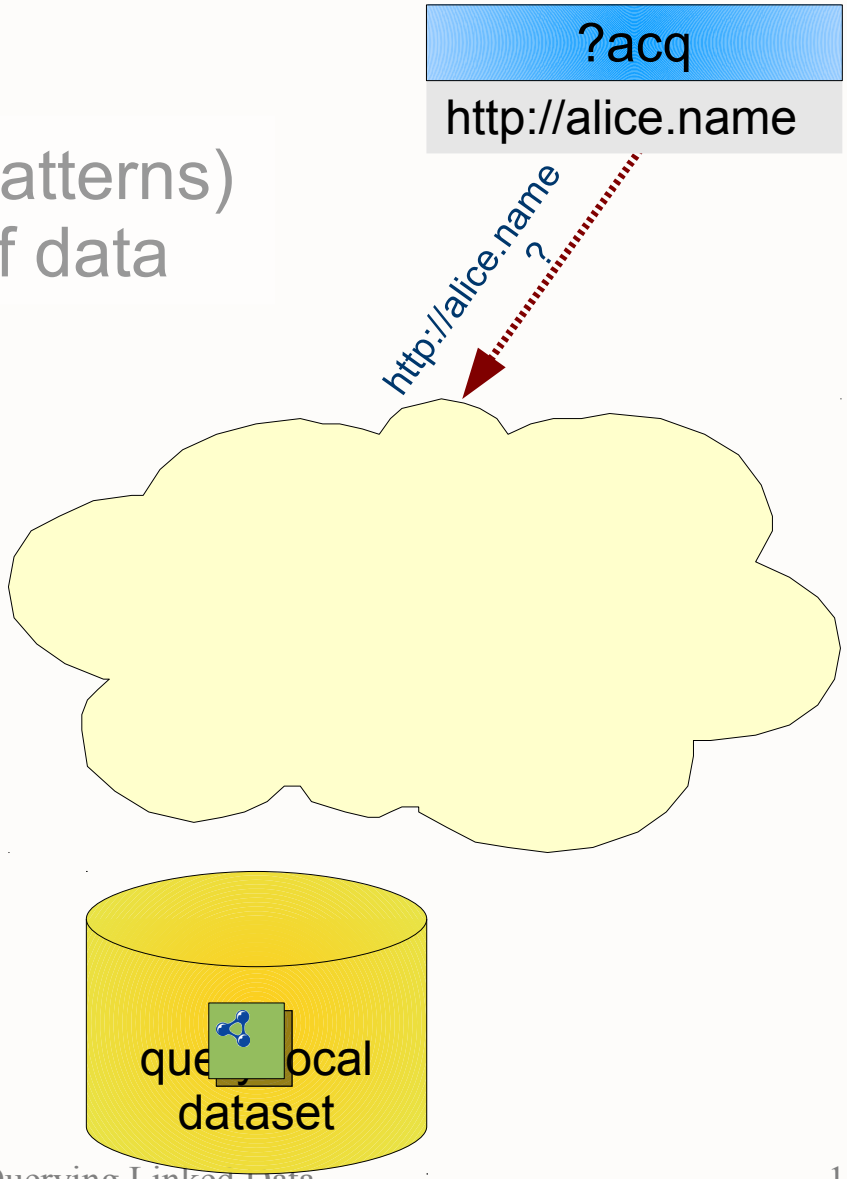
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset



Main Idea

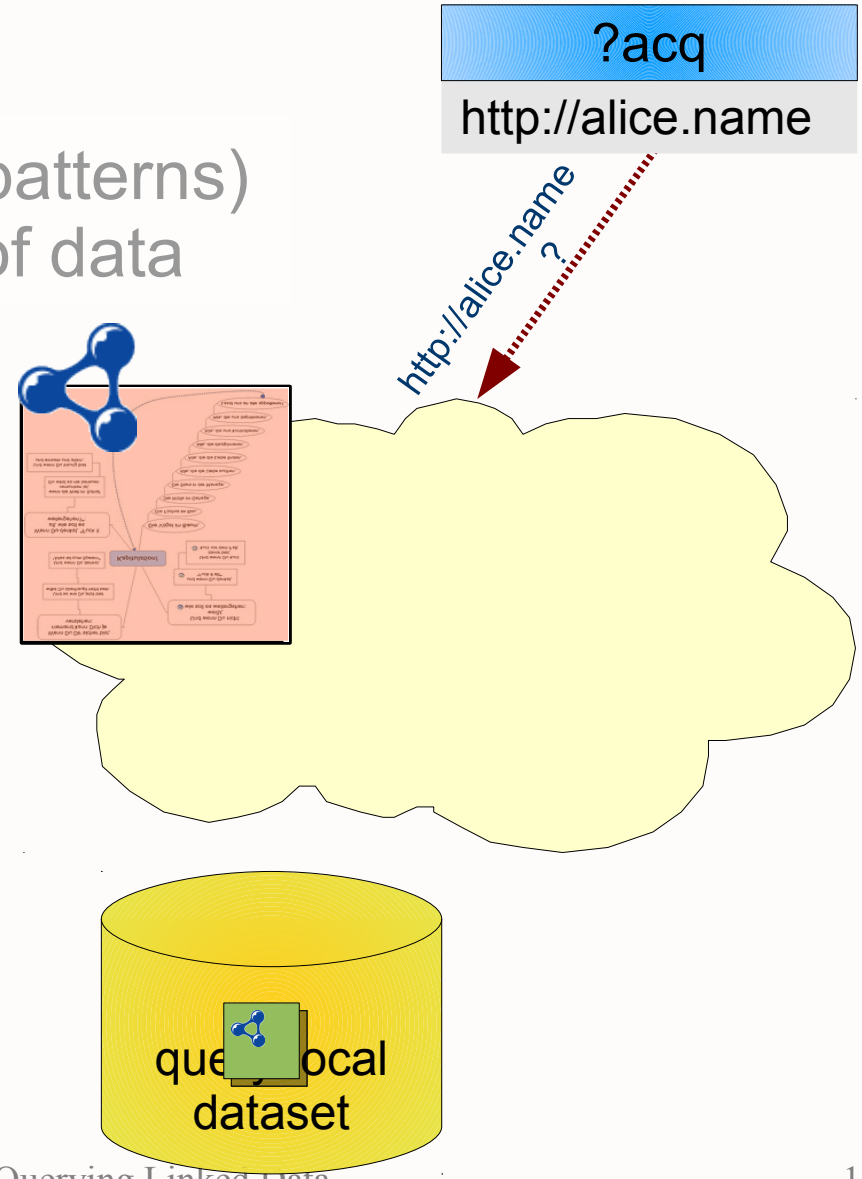
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

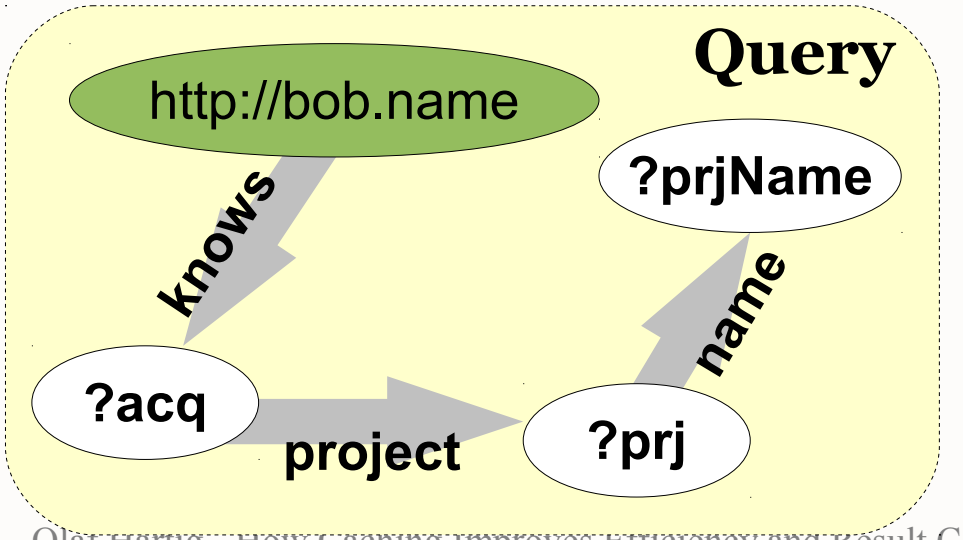




Main Idea

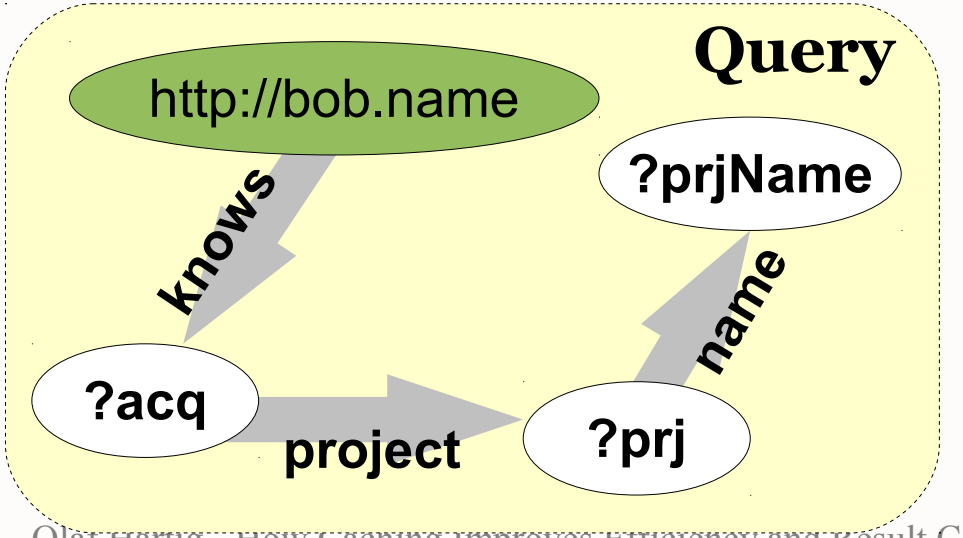
- **Intertwine query evaluation with traversal of data links**
- **We alternate between:**
  - Evaluate parts of the query (triple patterns) on a continuously augmented set of data
  - Look up URIs in intermediate solutions and add retrieved data to the query-local dataset

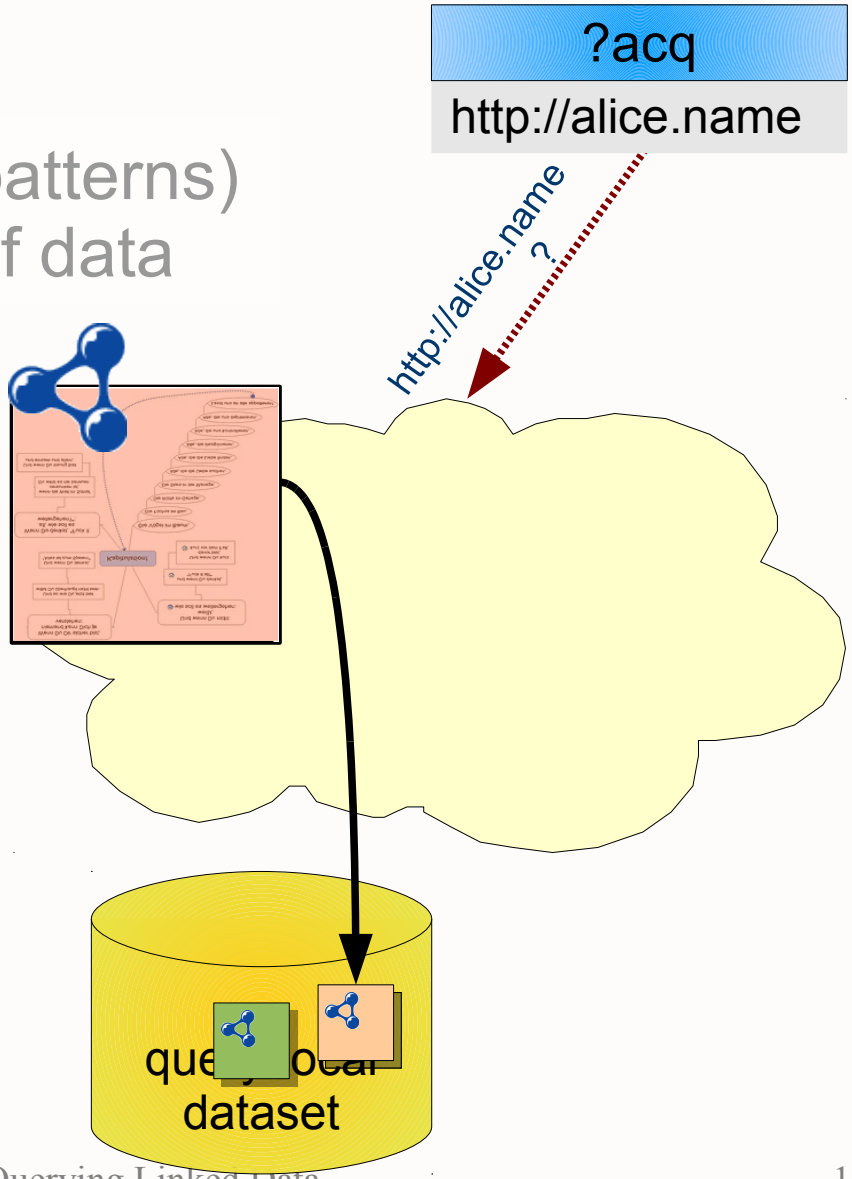




Main Idea

- **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

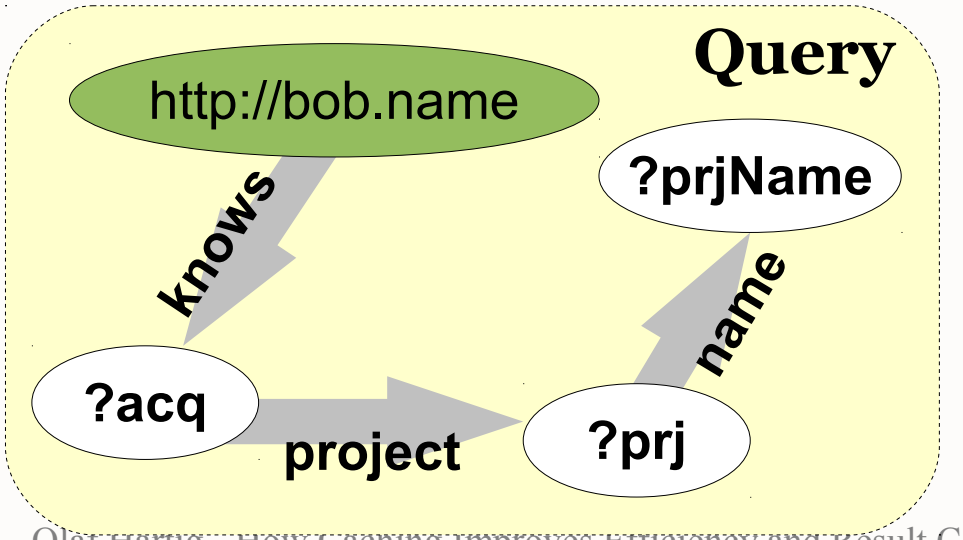


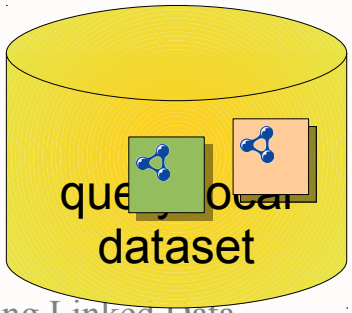


Main Idea

- **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

?acq
http://alice.name

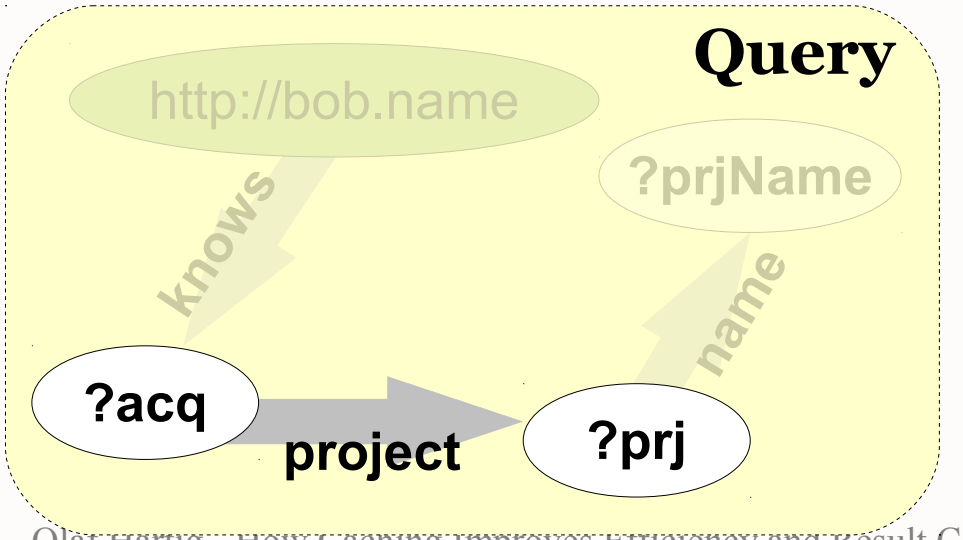


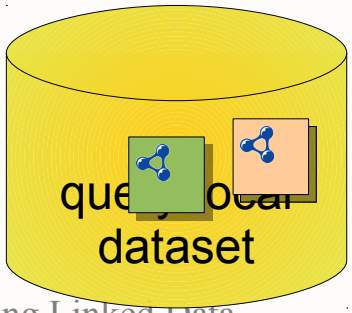


Main Idea

- **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

?acq
http://alice.name

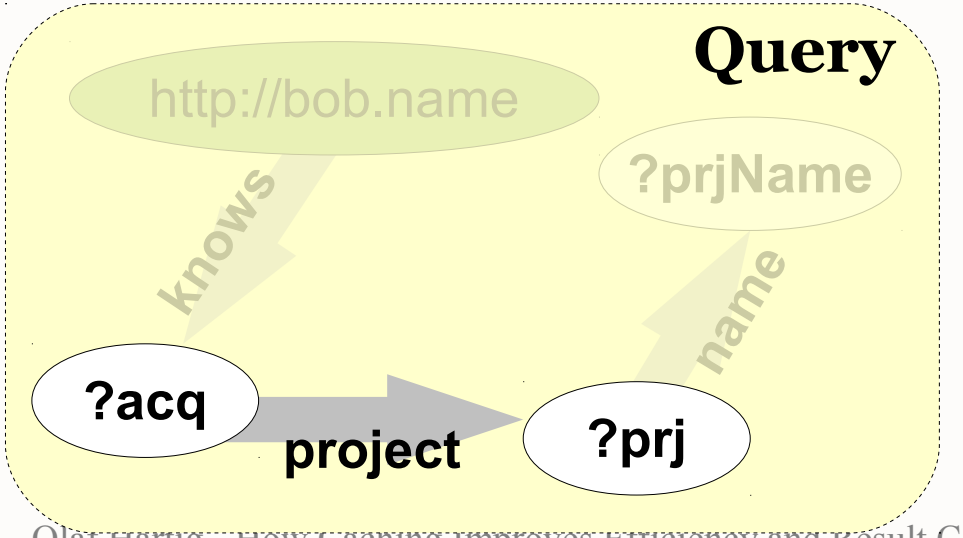


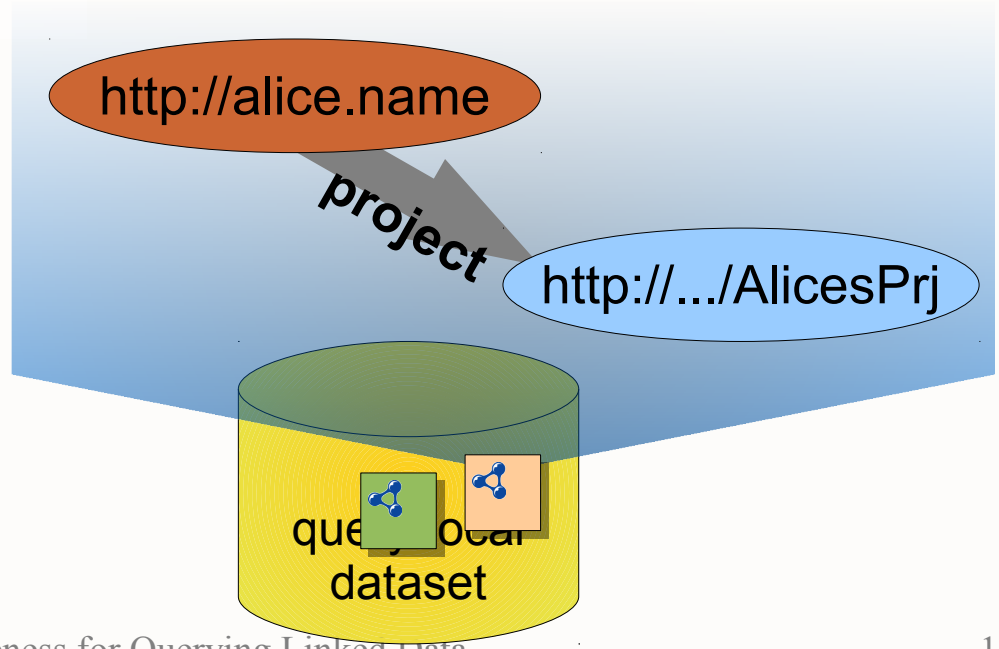


Main Idea

- **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

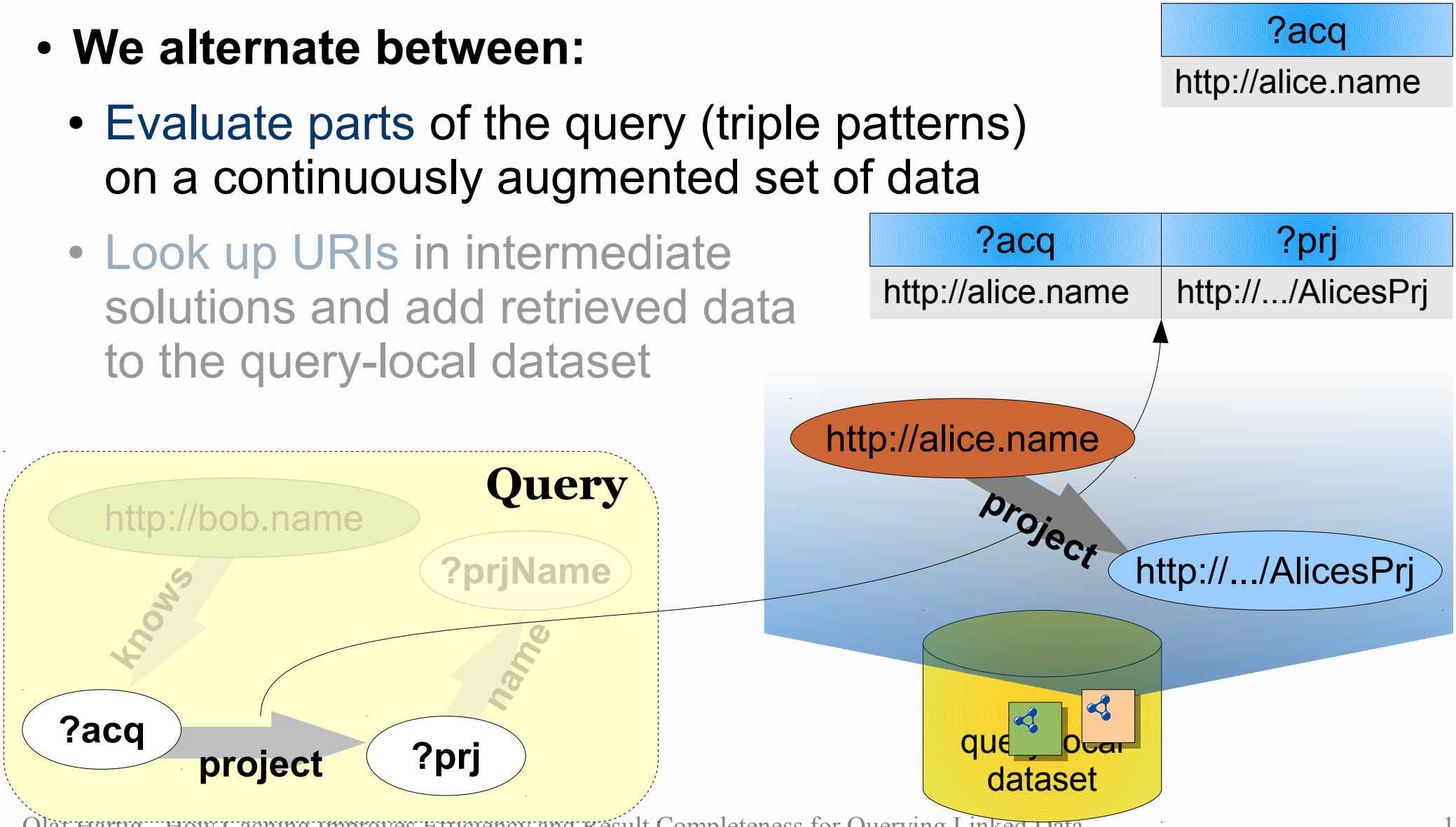
?acq
http://alice.name





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

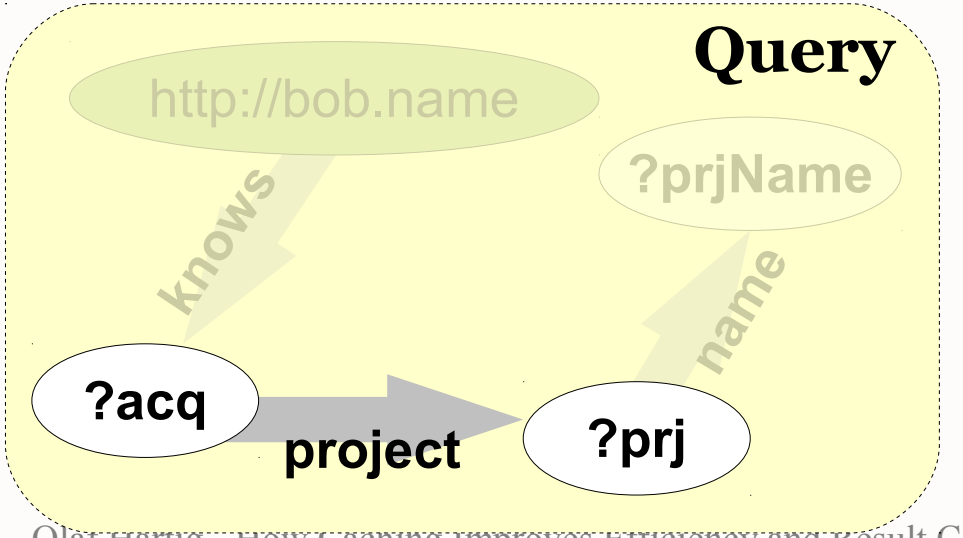


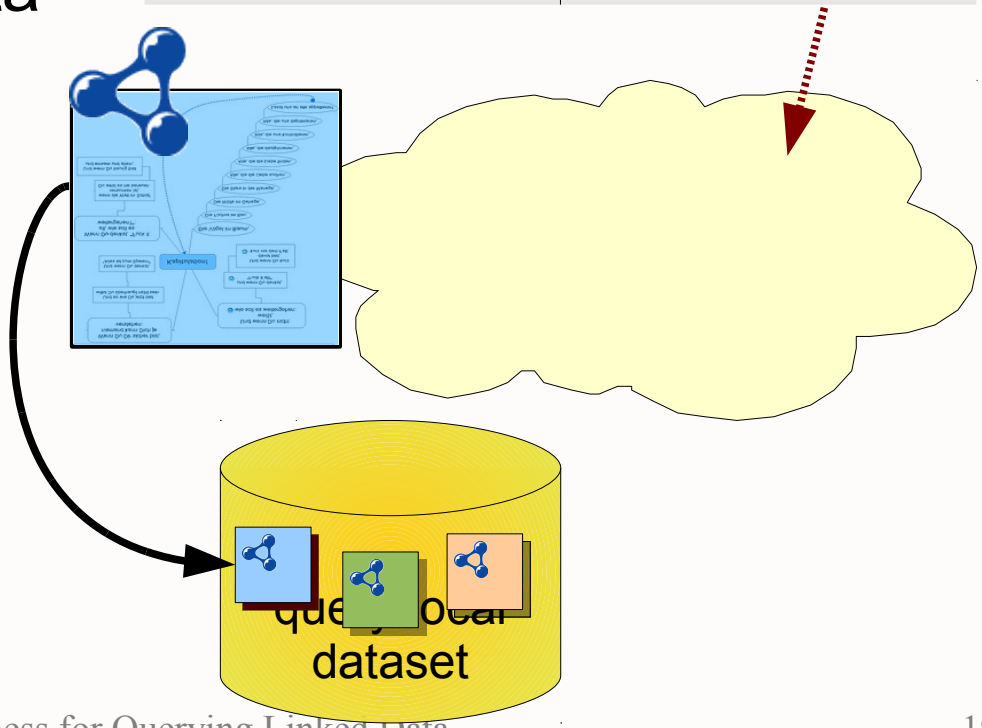
Main Idea

- **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

?acq
http://alice.name

?acq	?prj
http://alice.name	http://.../AlicesPrj





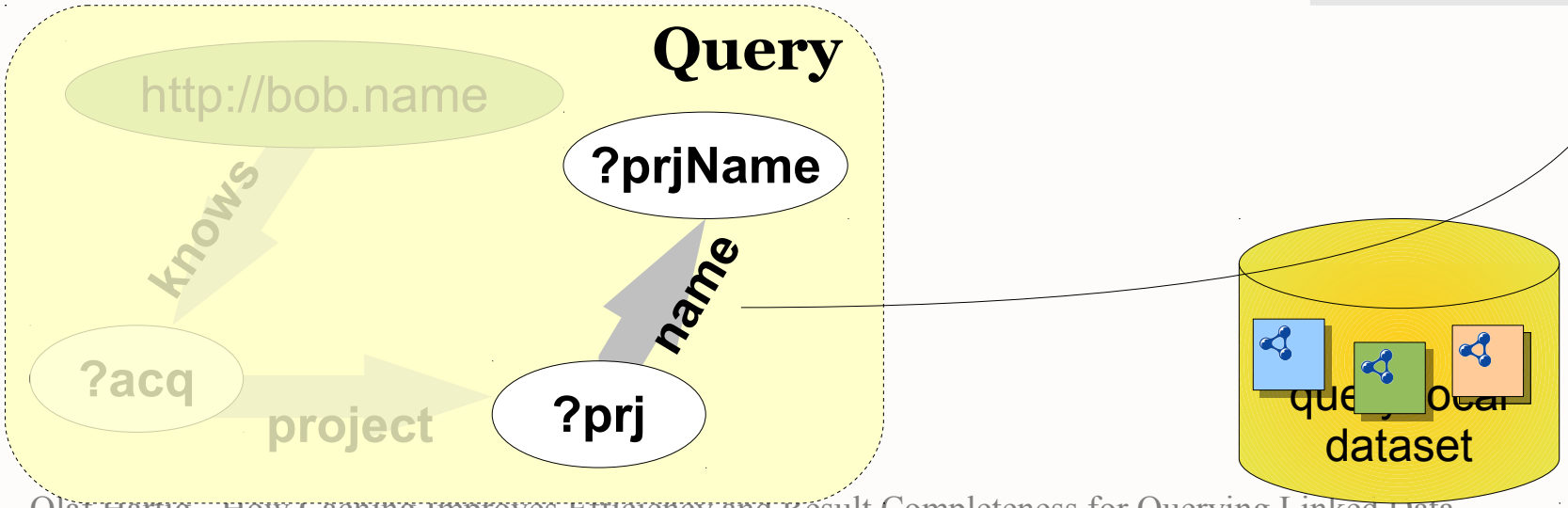
Main Idea

- **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

?acq
http://alice.name

?acq	?prj
http://alice.name	http://.../AlicesPrj

?prj	?prjName
http://.../AlicesPrj	“ … “



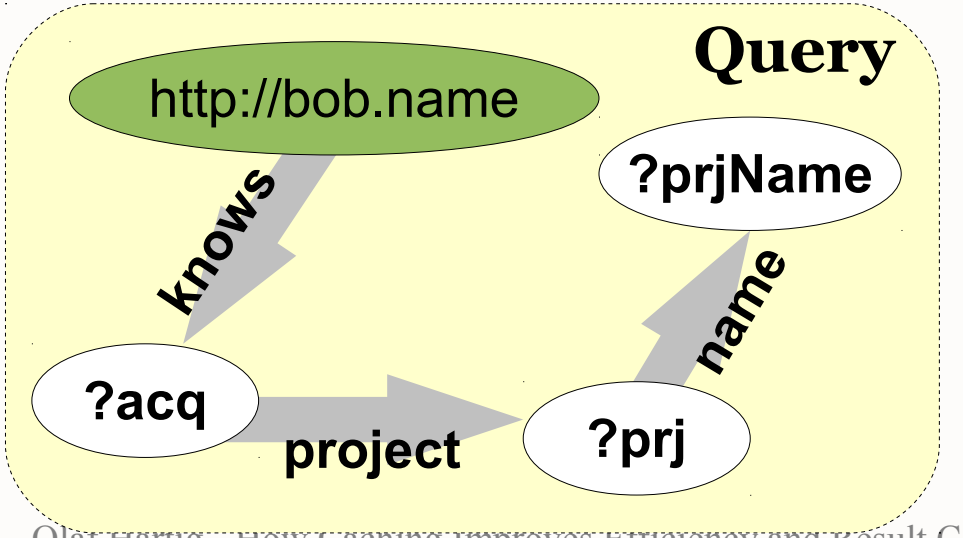
Main Idea

- **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

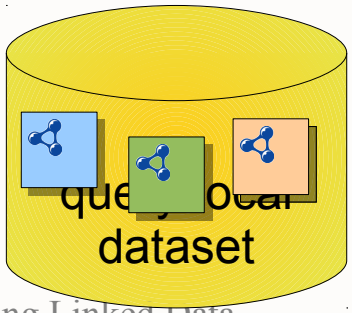
?acq
http://alice.name

?acq	?prj
http://alice.name	http://.../AlicesPrj

?prj	?prjName
http://.../AlicesPrj	“ … “



?acq	?prj	?prjName
http://alice.name	http://.../AlicesPrj	“ … “

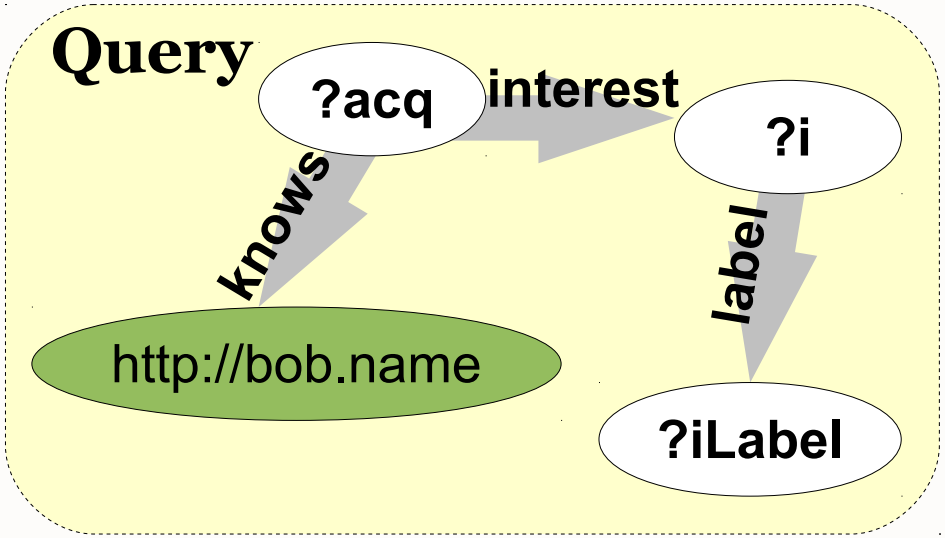


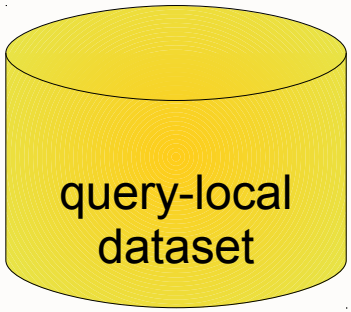
Characteristics

- **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

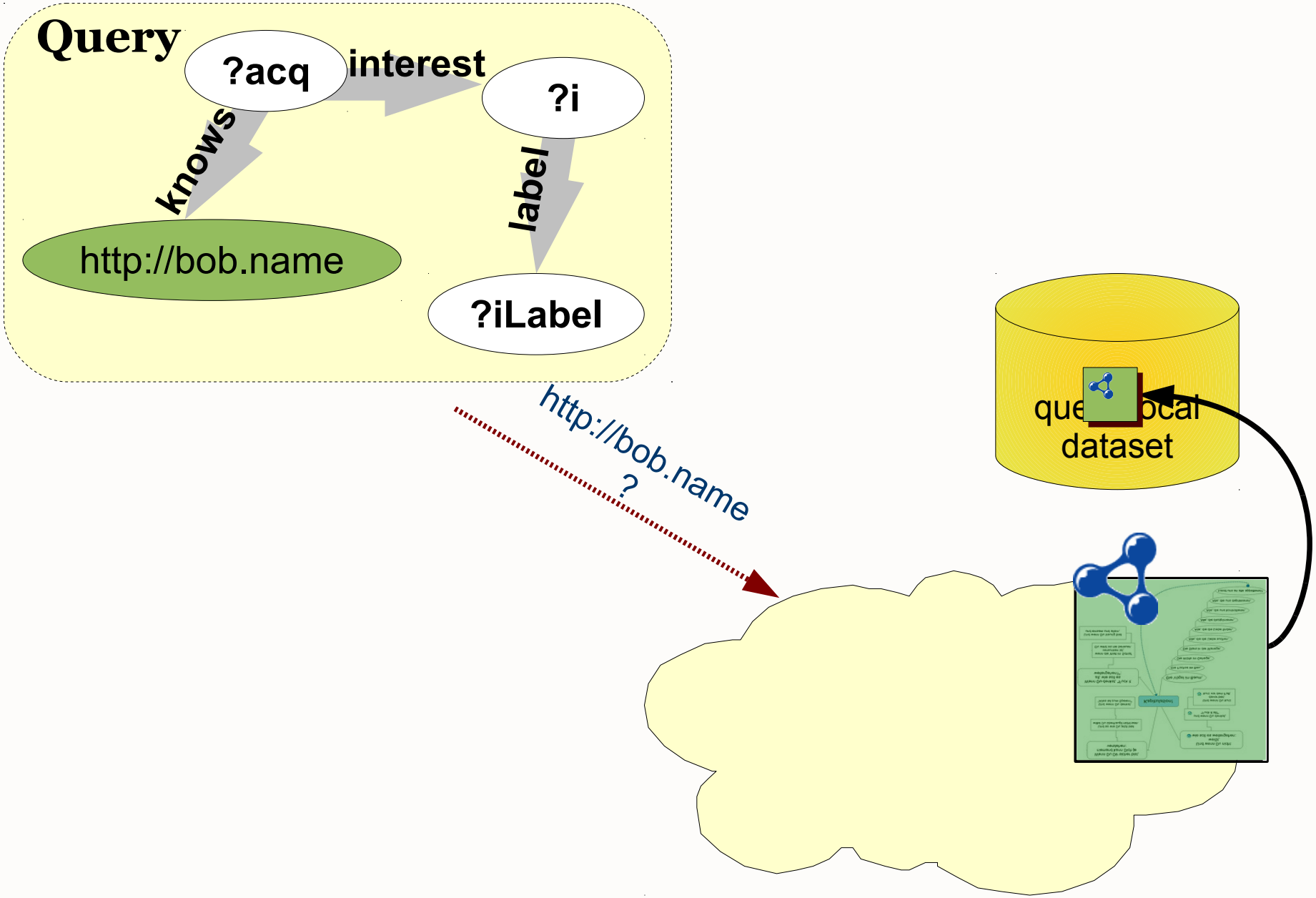
[*] formal definition in the paper

The Issue

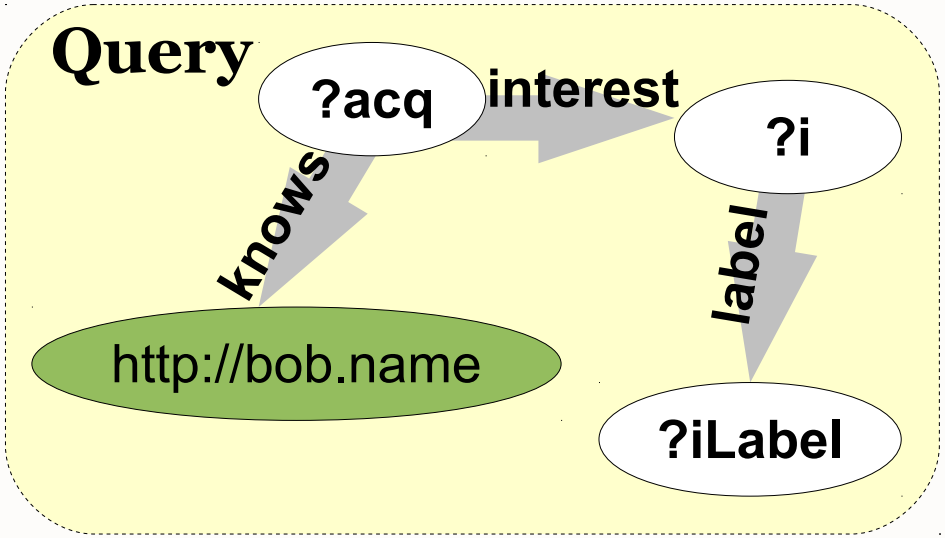

Query
?acq
interest
?i
knows
label
http://bob.name
?iLabel

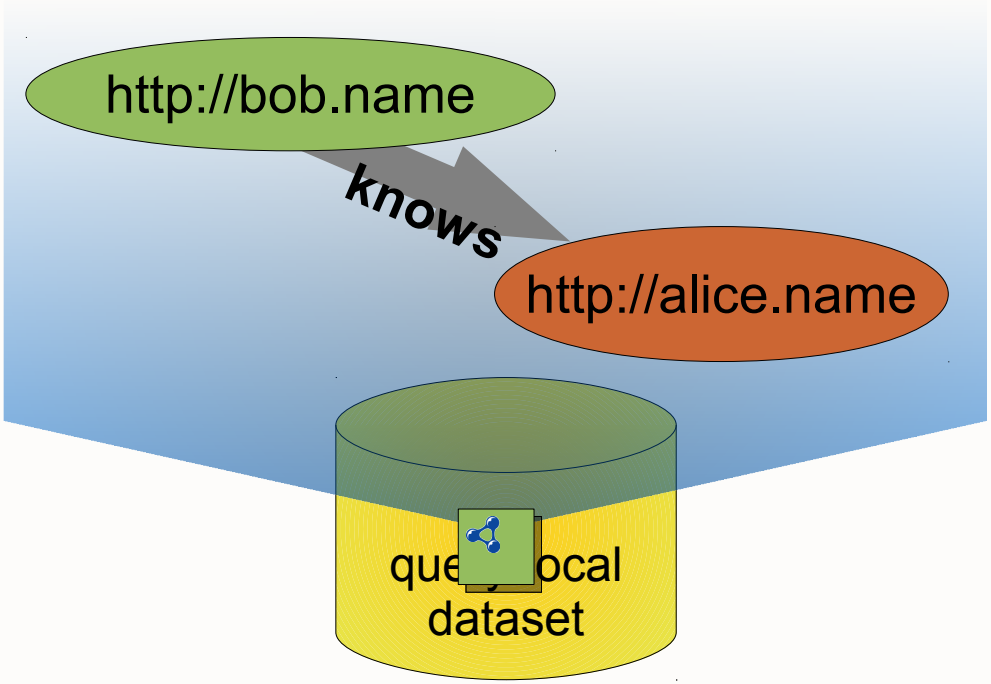

query-local
dataset

The Issue

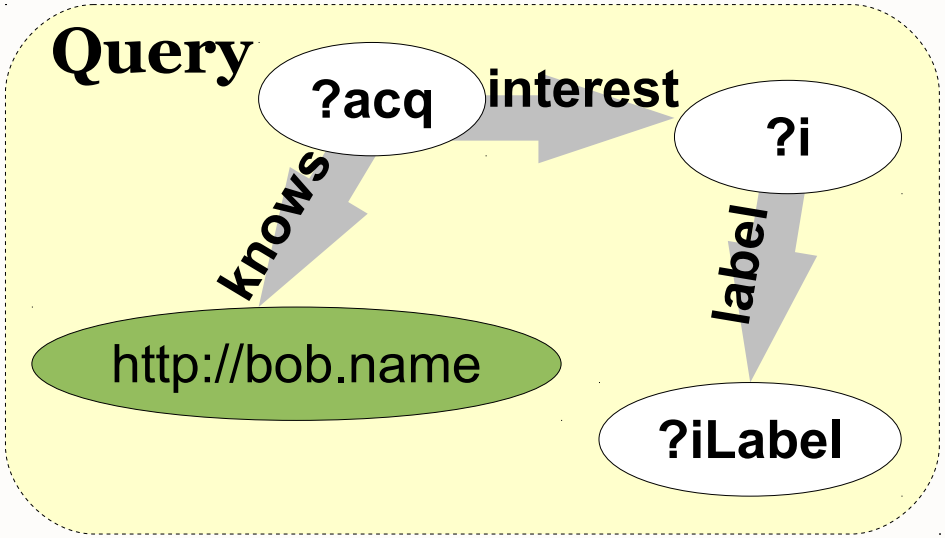

Query
?acq
interest
?i
knows
label
http://bob.name
?iLabel
http://bob.name
?
dataset

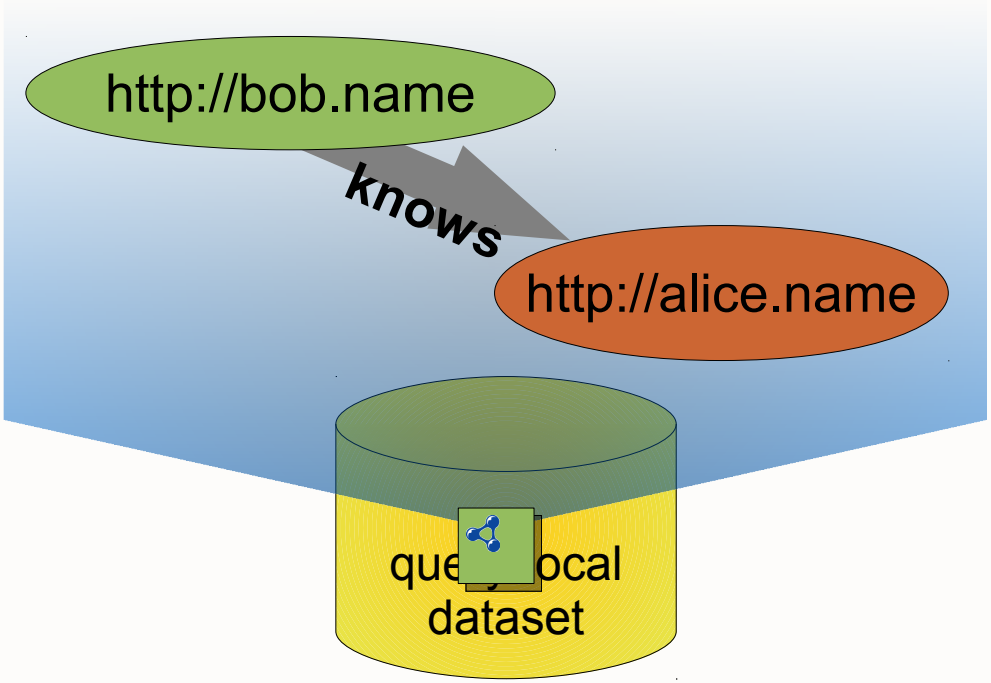
The Issue


Query
?acq
interest
?i
knows
label
http://bob.name
?iLabel


http://bob.name
knows
http://alice.name
dataset

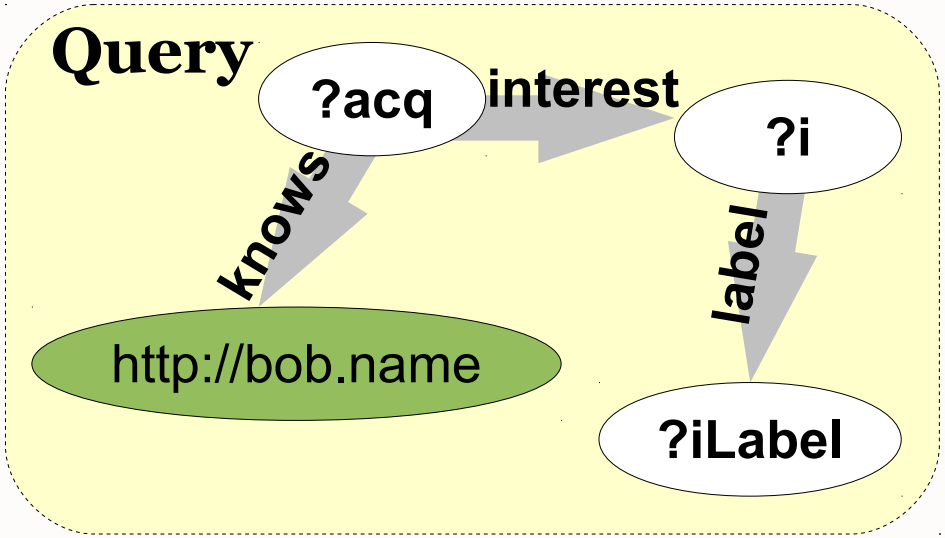
The Issue

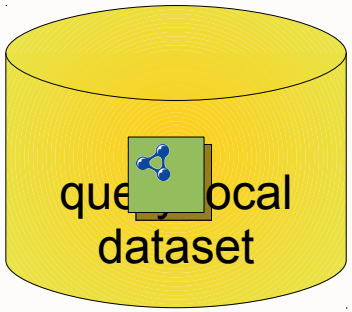


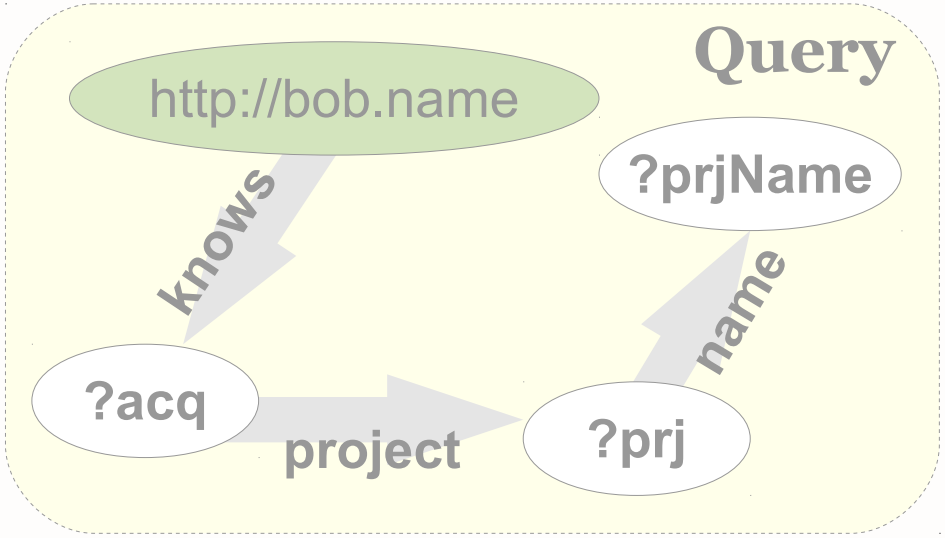


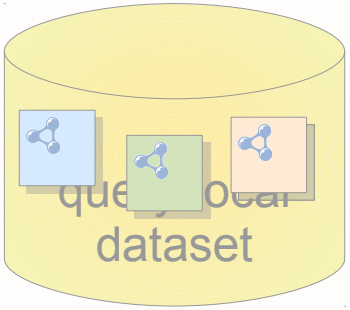
?acq	?i	?iLabel

The Issue

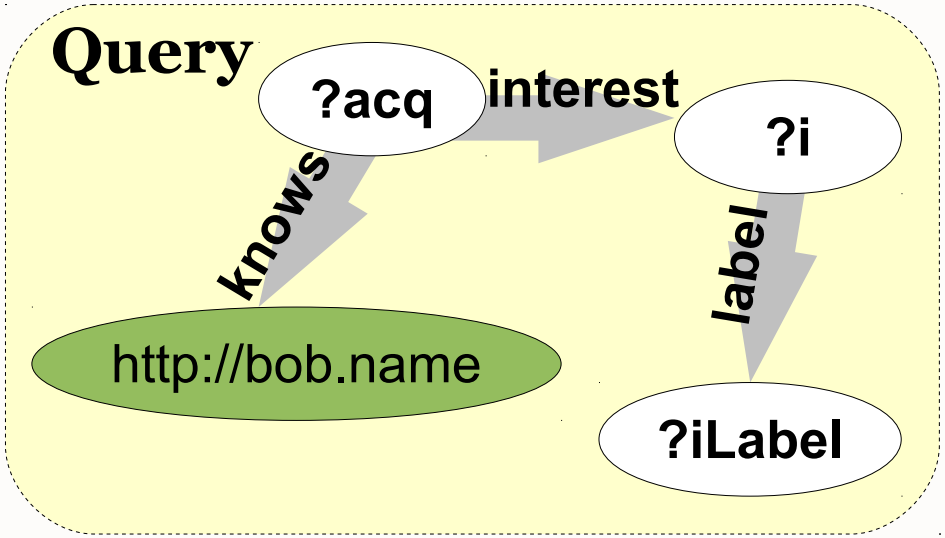

Query
?acq
interest
?i
knows
label
http://bob.name
?iLabel

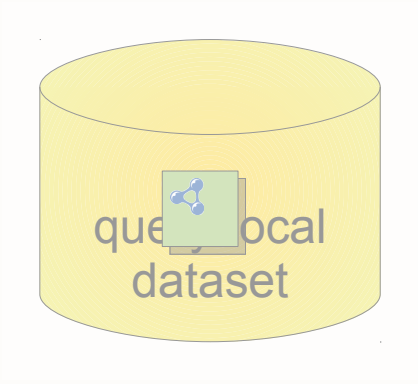

dataset

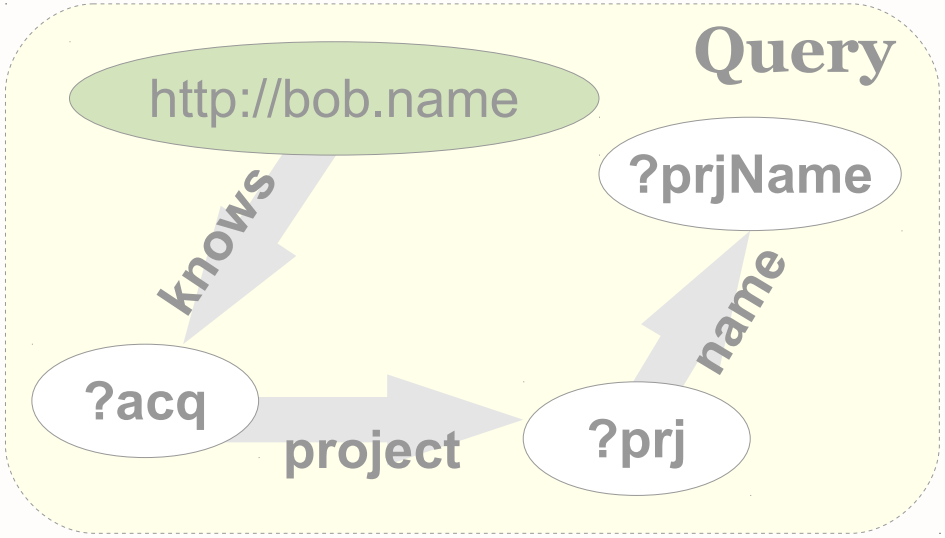

Query
http://bob.name
?prjName
knows
name
?acq
project
?prj

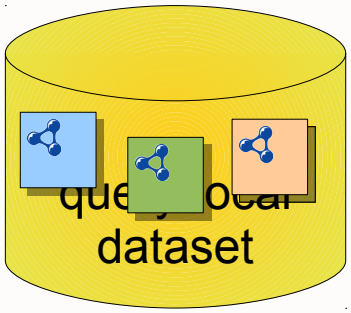

dataset

Reusing the Query-Local Dataset

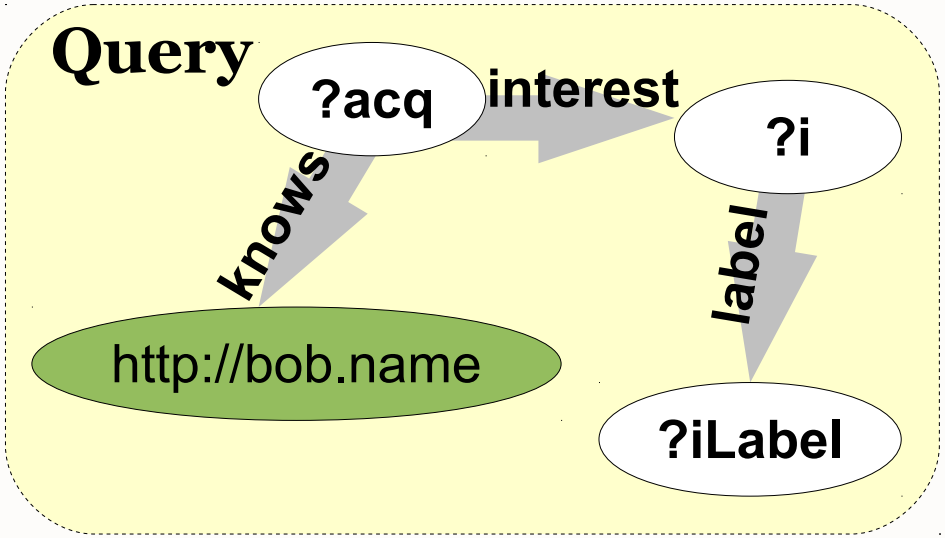


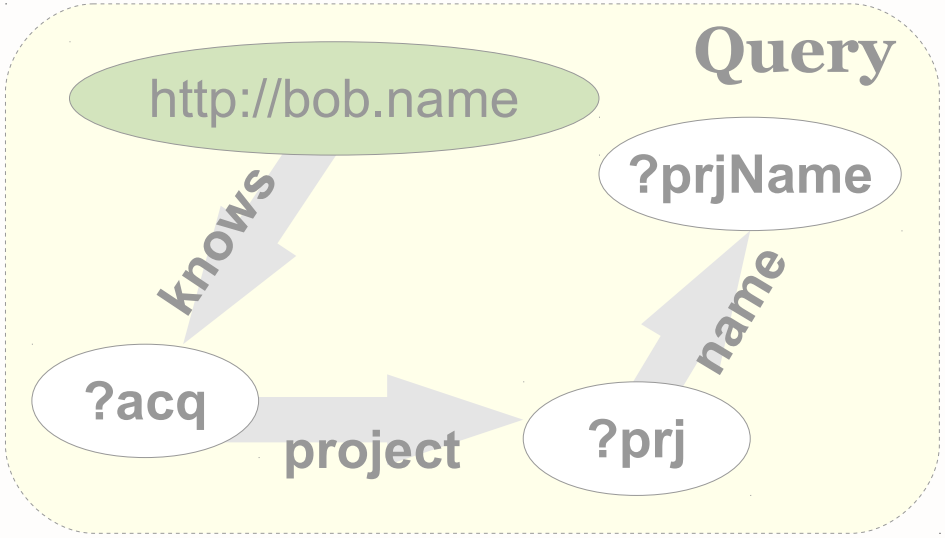


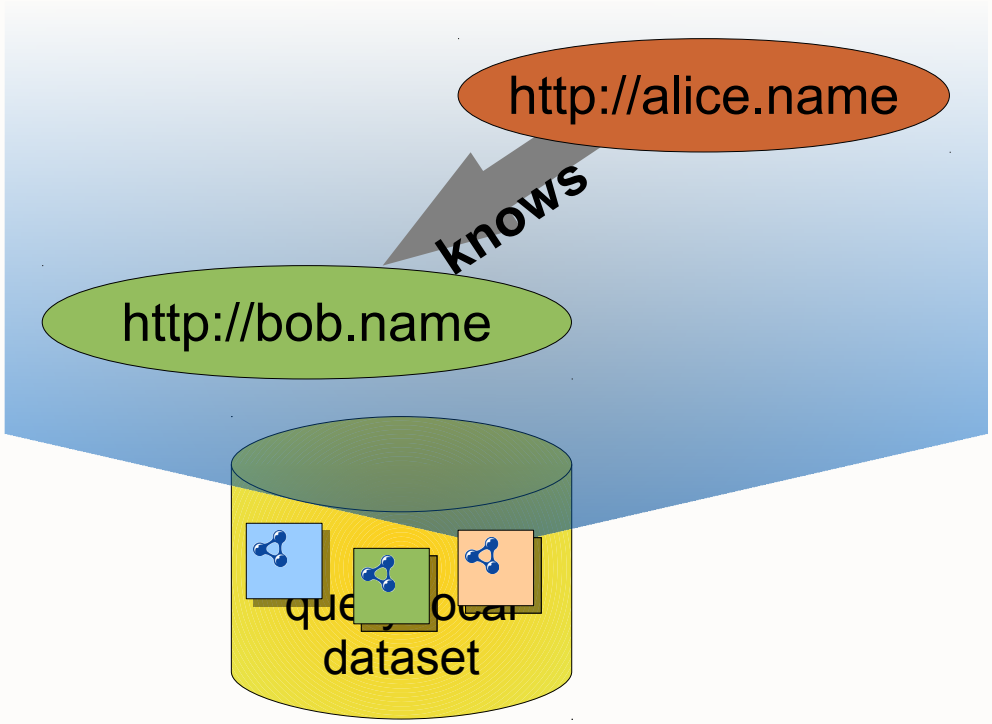




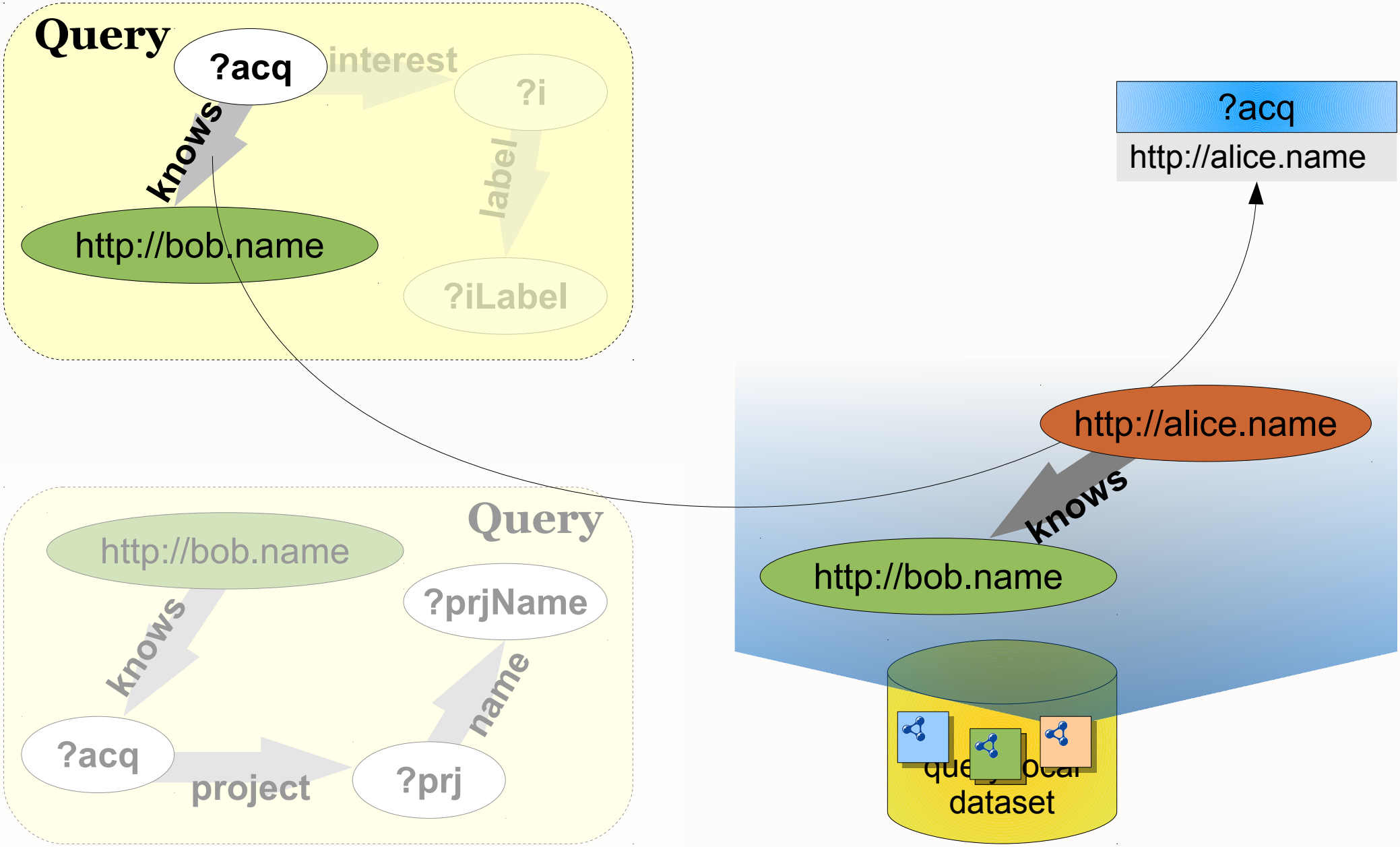
Reusing the Query-Local Dataset







Reusing the Query-Local Dataset



Hypothesis

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

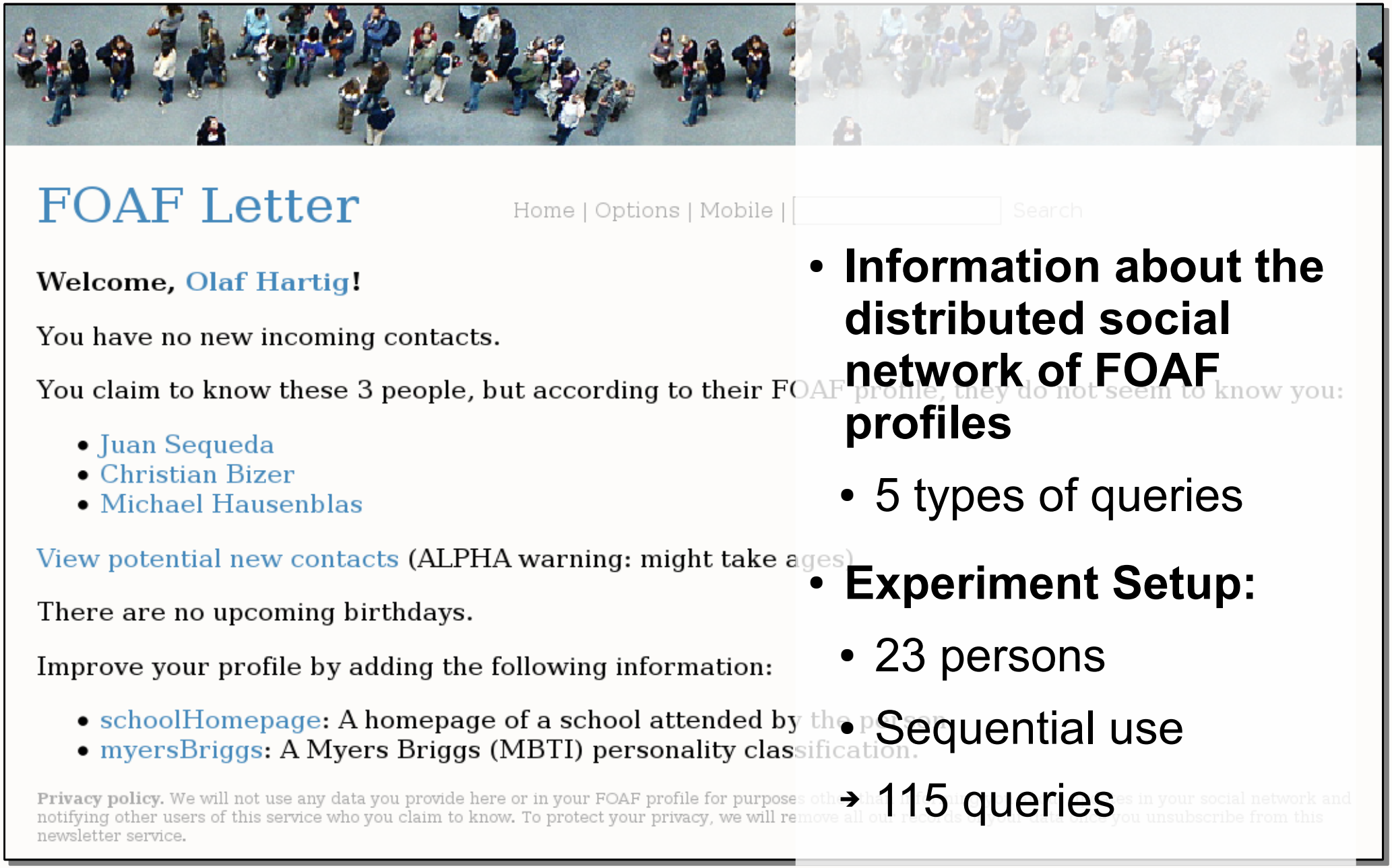
Contributions

- **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



- **Information about the distributed social network of FOAF profiles**
  - 5 types of queries
- **Experiment Setup:**
  - 23 persons
  - Sequential use
  - → 115 queries

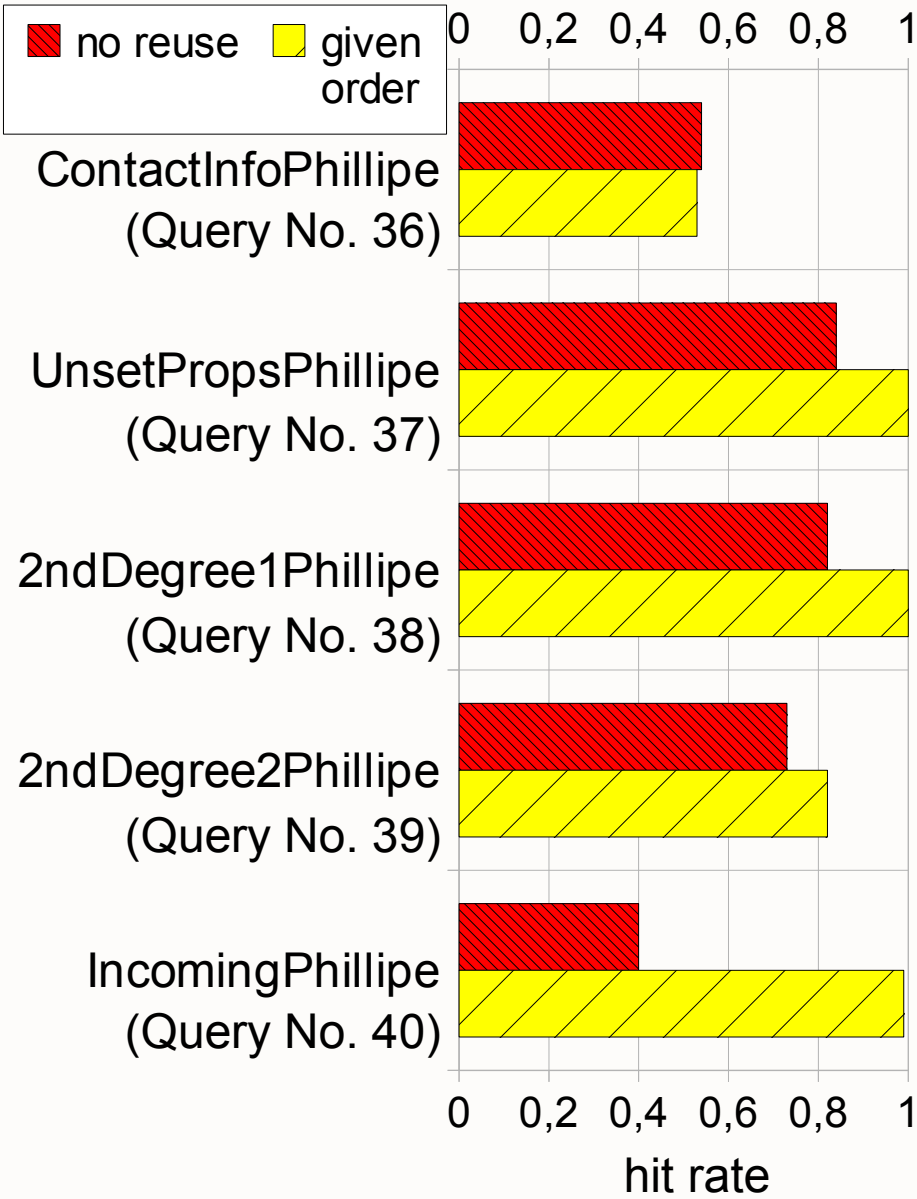
Experiment – Complete Sequence

- ***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:**

$$\frac{\text{look-ups answered from cache}}{\text{all look-up requests}}$$

Experiment – Complete Sequence

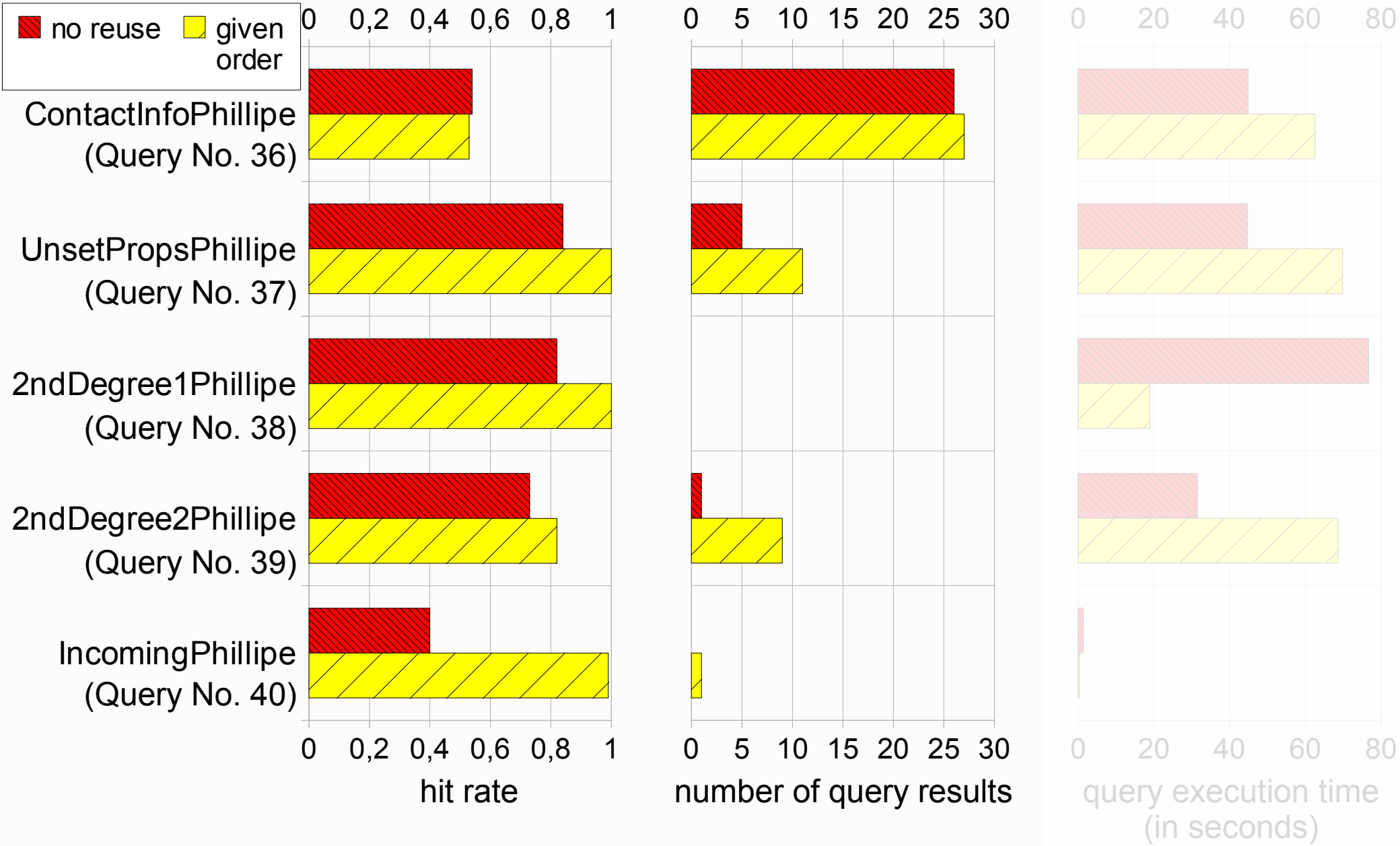


- ***no reuse* experiment:**
  - No data caching
- ***given order* experiment**
  - Reuse of the query-local dataset for the complete sequence of all 115 queries

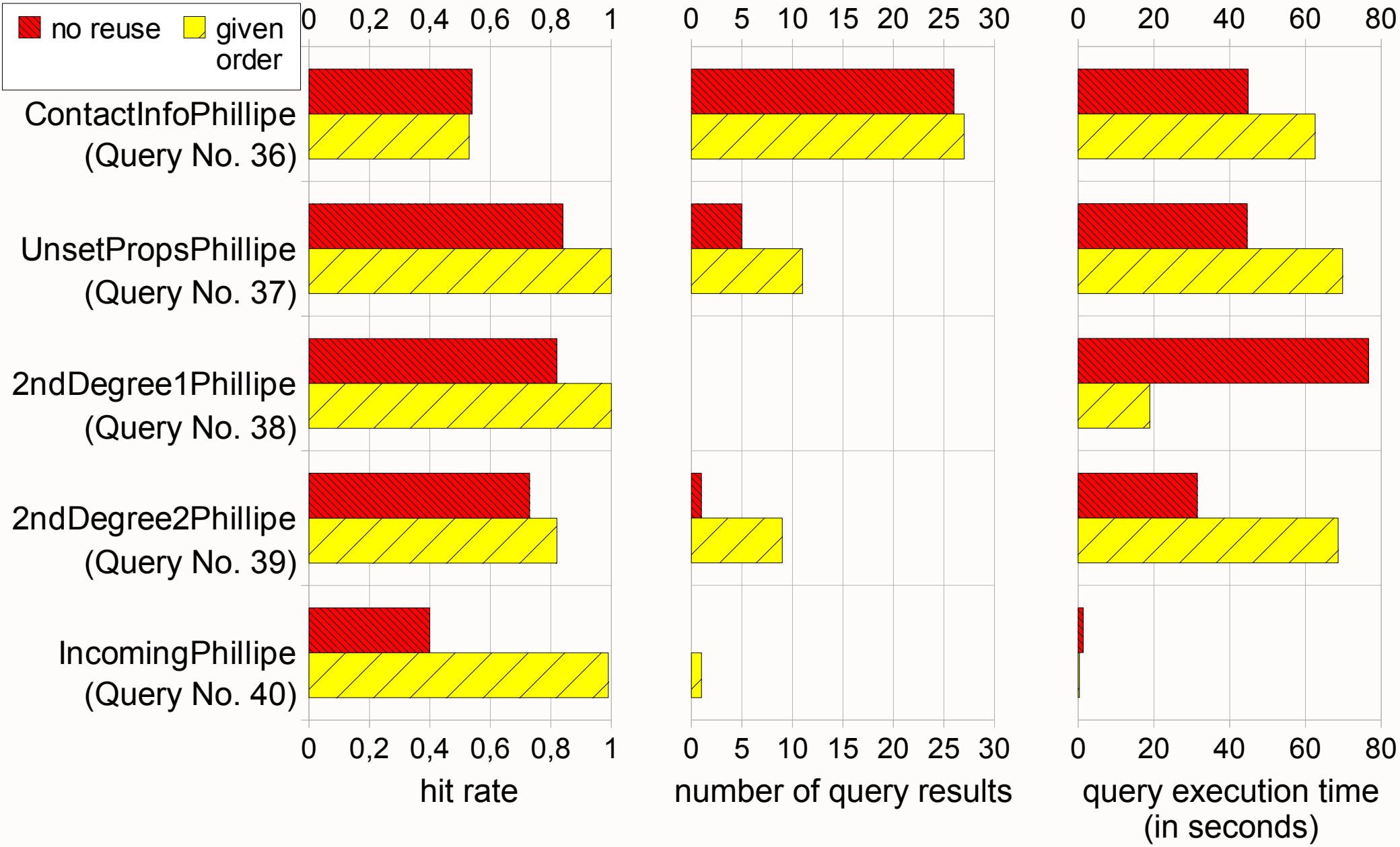
- **Hit rate:**

$$\frac{\text{look-ups answered from cache}}{\text{all look-up requests}}$$

Experiment – Complete Sequence



Experiment – Complete Sequence



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_{seed}-initialized execution of a BGP query b
  - D_{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_{cur} is also $R(B)$-dependent solution for b_{cur}
- **Conceptual analysis of the impact of data caching**
  - Performance factor: $p(b_{cur}, B) = c(b_{cur}, [\,]) - c(b_{cur}, B)$
  - Serendipity factor: $s(b_{cur}, B) = b(b_{cur}, B) - b(b_{cur}, [\,])$
- **Empirical verification of the potential impact**
- **Out of scope:** Caching strategies (replacement, invalidation)

Query Template Contact


```
SELECT * WHERE {  <PERSON> foaf:knows ?p .

                  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 }
}
```

Query Template UnsetProps

```
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 .
   ?var2 ?result ?var3 .
   ?result rdfs:label ?resultLabel .
   ?result vs:term_status ?var1 .
}
ORDER BY ?var1
```

Query Template Incoming


```
SELECT DISTINCT ?result WHERE
{
  ?result foaf:knows <PERSON> .

  OPTIONAL
  {
    ?result foaf:knows ?var1 .
    FILTER ( <PERSON> = ?var1 )
    <PERSON> foaf:knows ?result .
  }
  FILTER ( !bound(?var1) )
}
```

Query Template 2ndDegree1


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

   ?p1 foaf:knows ?result .
   FILTER ( <PERSON> != ?result )
   ?p2 foaf:knows ?result .

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

Query Template 2ndDegree2


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

   ?result foaf:knows ?p1 .
   FILTER ( <PERSON> != ?result )
   ?result foaf:knows ?p2 .

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

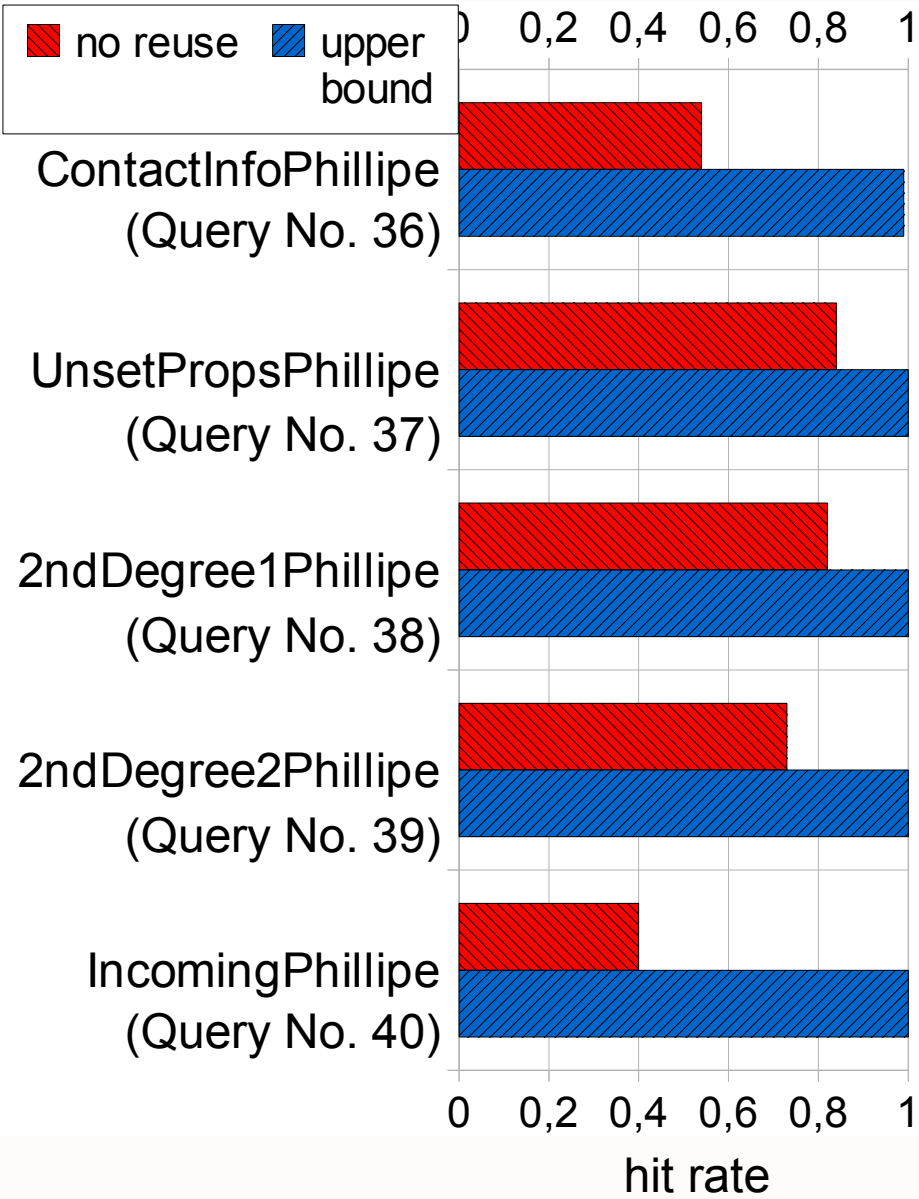
Experiment – Single Query

- ***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:**

$$\frac{\text{look-ups answered from cache}}{\text{all look-up requests}}$$

Experiment – Single Query

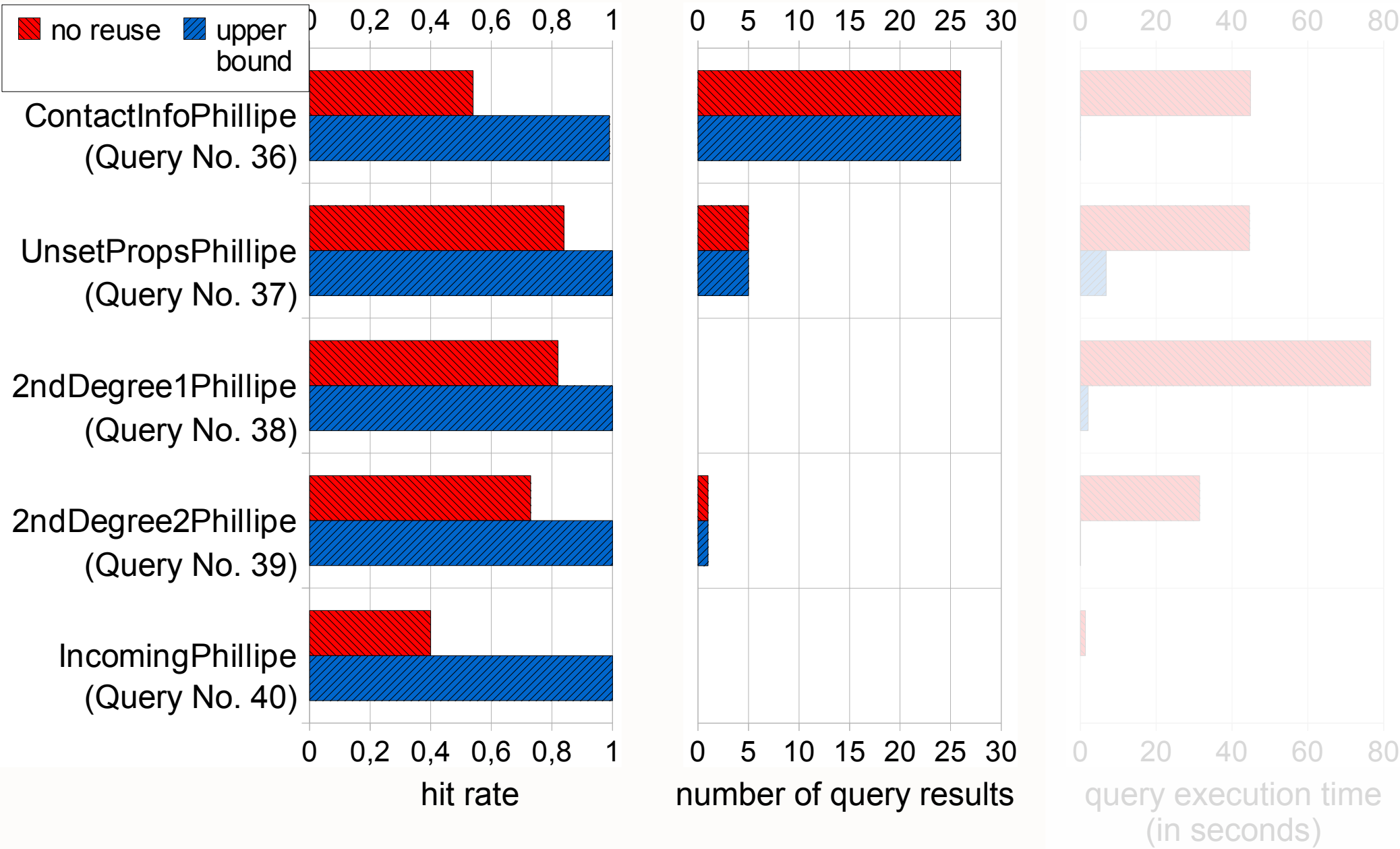


- ***no reuse* experiment:**
  - No data caching
- ***upper bound* experiment**
  - Reuse of query-local dataset for 3 executions of each query
  - Third execution measured

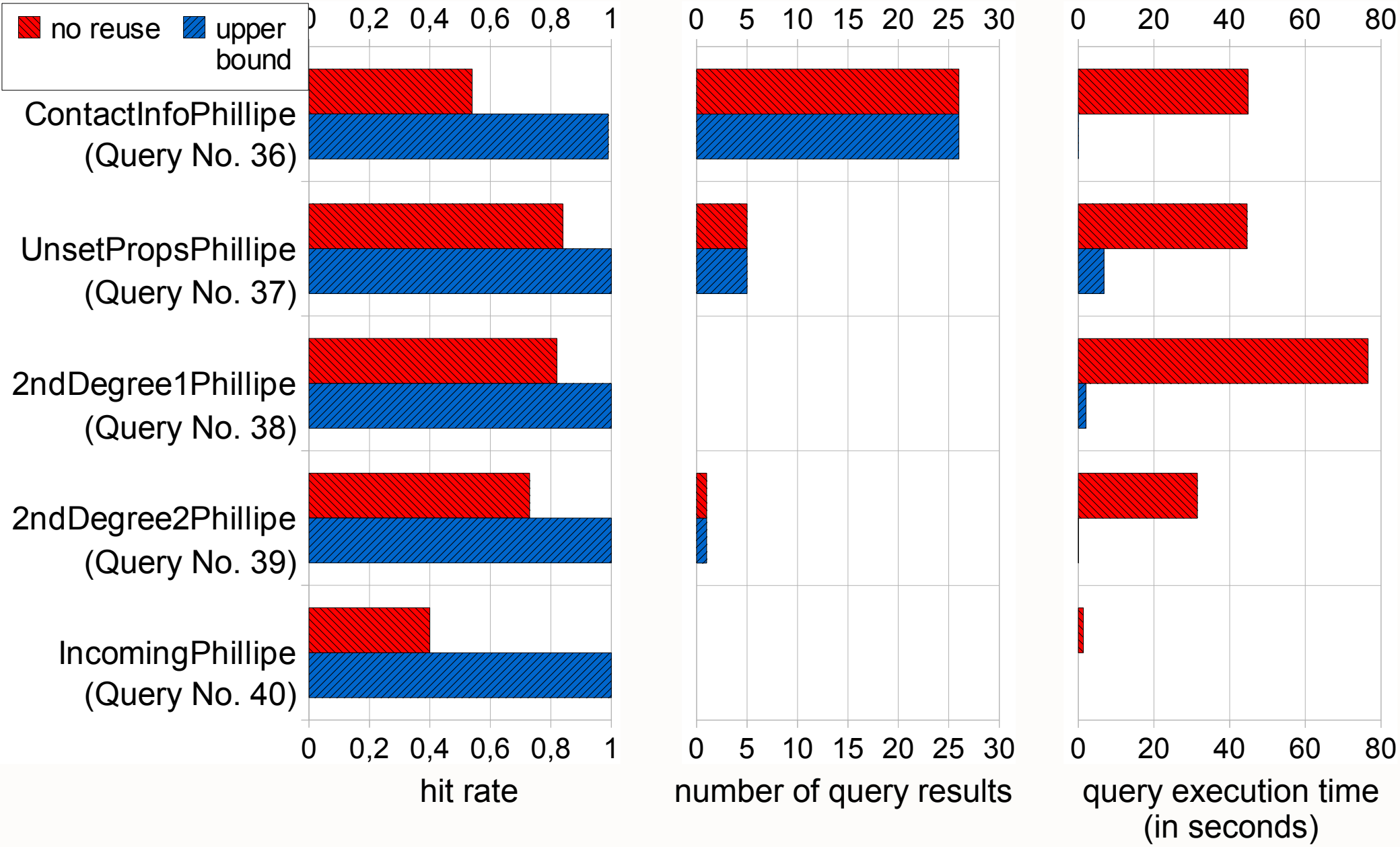
- **Hit rate:**

$$\frac{\text{look-ups answered from cache}}{\text{all look-up requests}}$$

Experiment – Single Query



Experiment – Single Query



Experiment – Single Query

Experiment	Avg.[1] number of **Query Results** (std.dev.)	Average[1] **Hit Rate** (std.dev.)	Avg.[1] query **Execution Time** (std.dev.)
no reuse	4.983 (11.658)	0.576 (0.182)	30.036 s (46.708)
upper bound	5.070 (11.813)	0.996 (0.017)	1.943 s (11.375)

[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

Experiment	Avg.[1] number of **Query Results** (std.dev.)	Average[1] **Hit Rate** (std.dev.)	Avg.[1] query **Execution Time** (std.dev.)
no reuse	4.983 (11.658)	0.576 (0.182)	30.036 s (46.708)
upper bound	5.070 (11.813)	0.996 (0.017)	1.943 s (11.375)

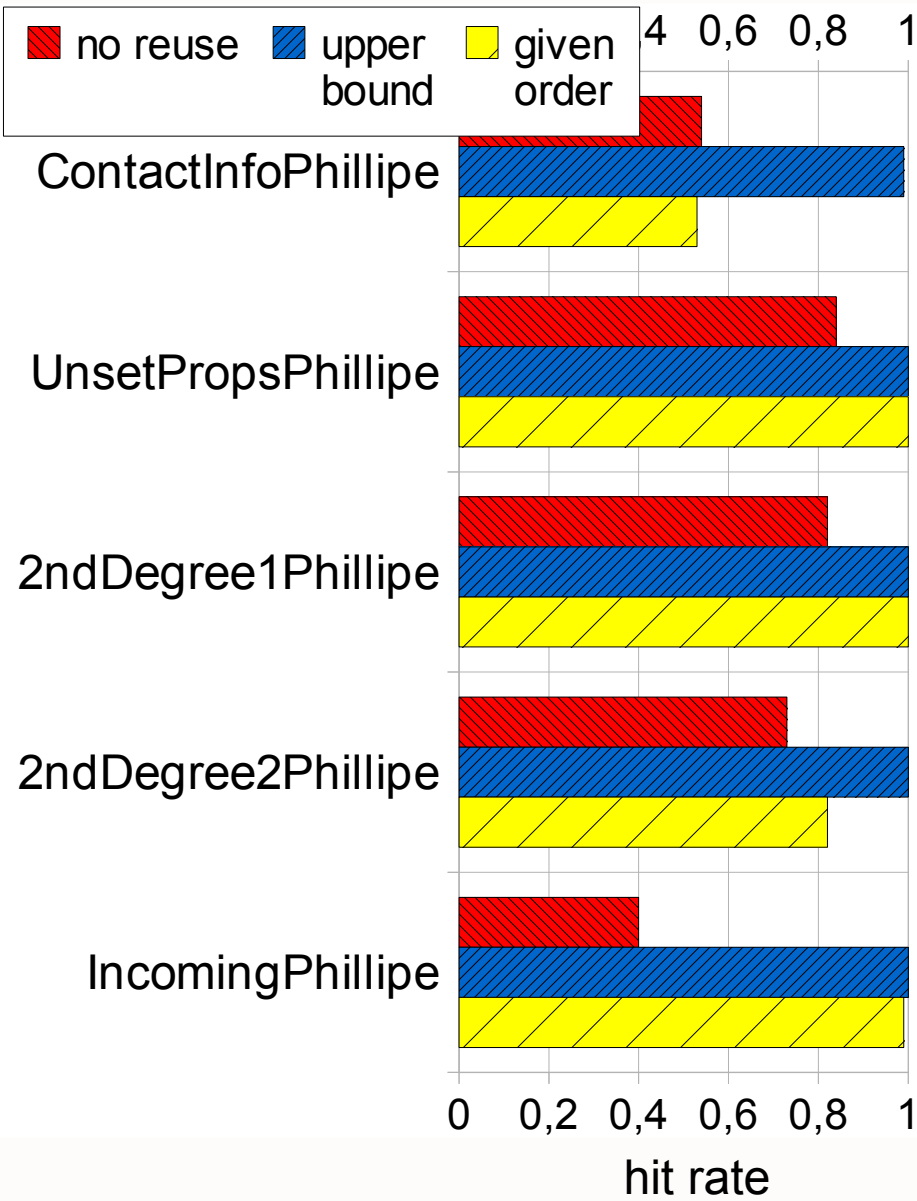
[1] Averaged over all 115 queries

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

Experiment – Complete Sequence

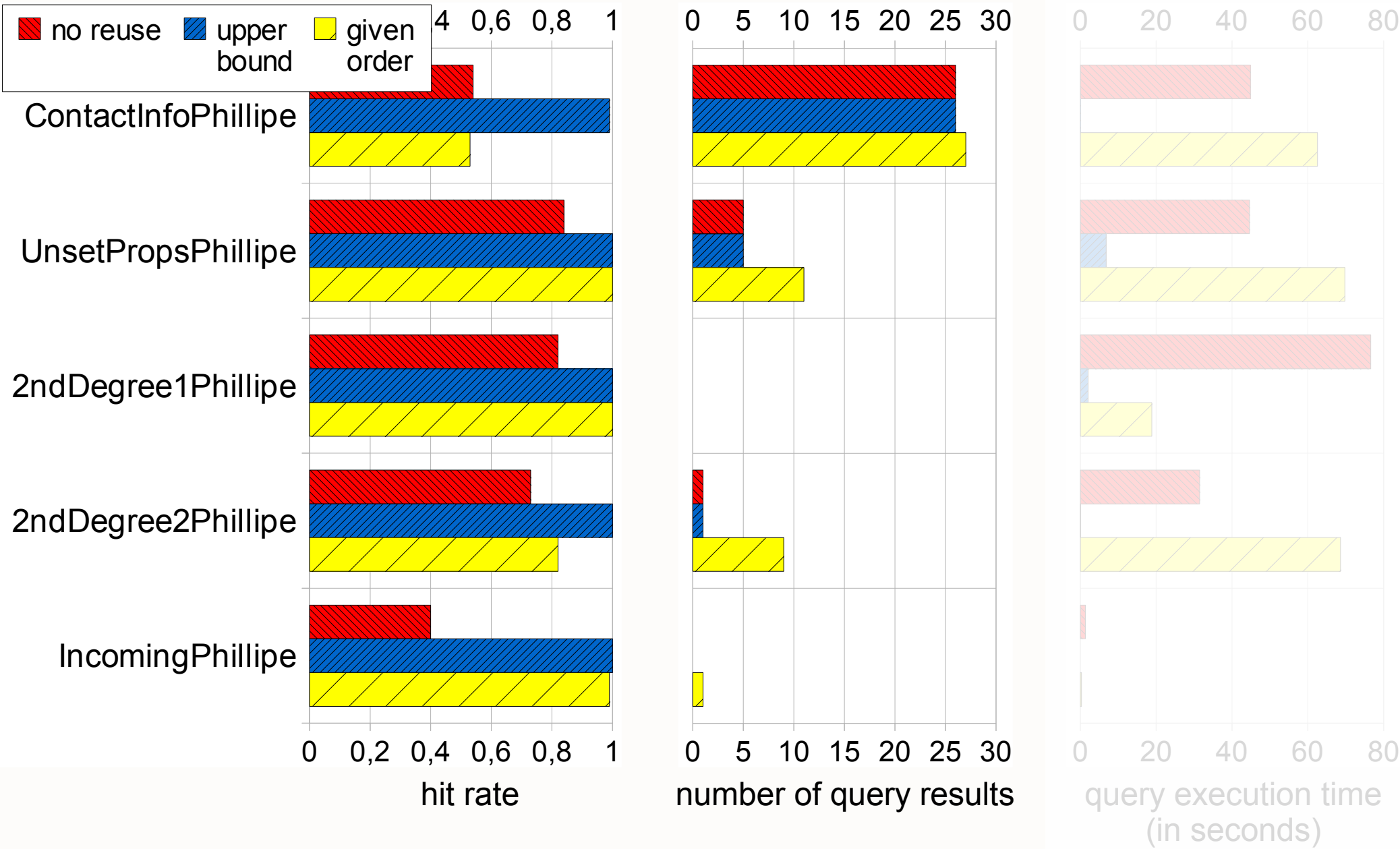
- ***given order* experiment:**
  - Reuse of the query-local dataset for the complete sequence of all 115 queries

Experiment – Complete Sequence

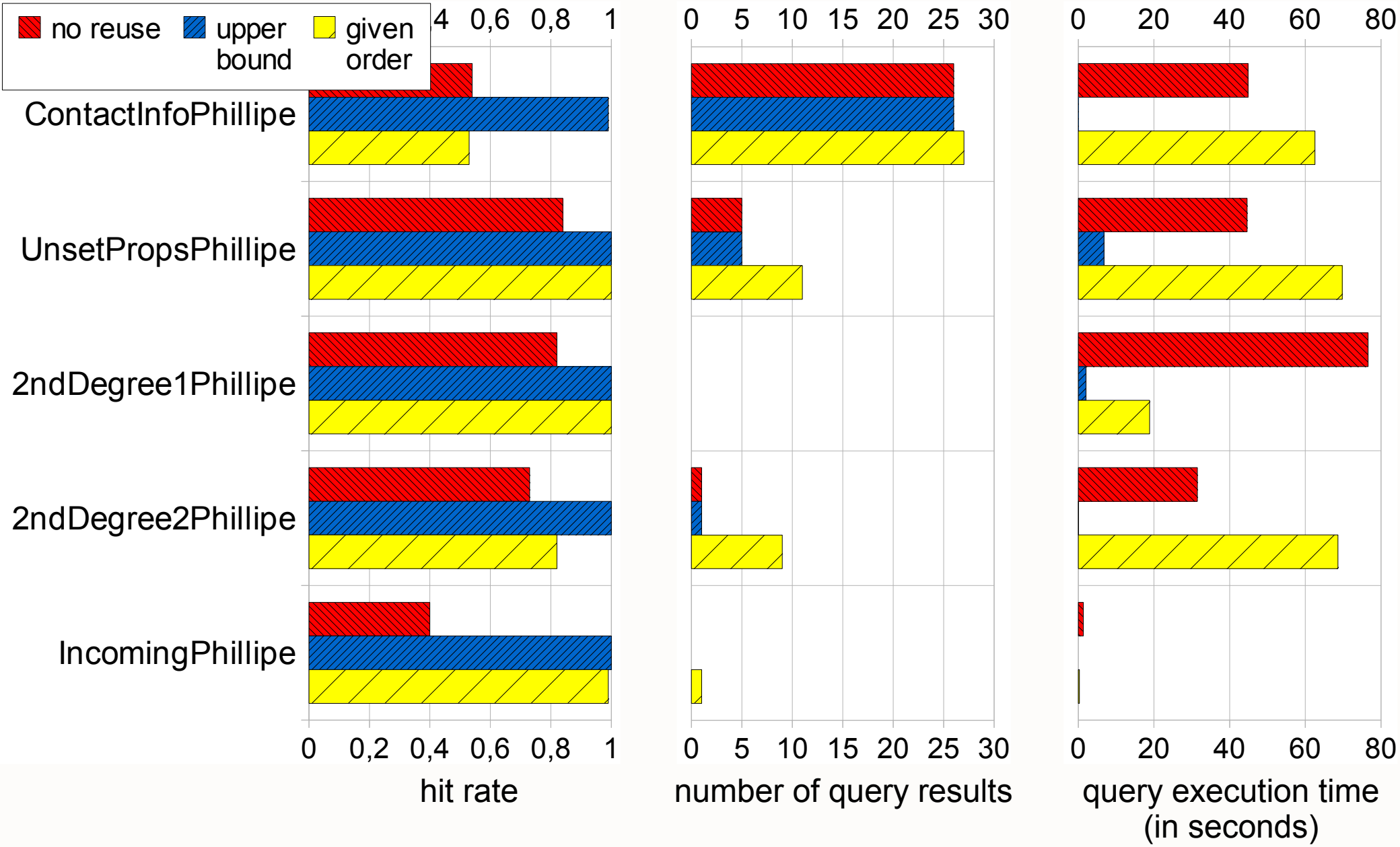


- ***given order* experiment:**
  - Reuse of the query-local dataset for the complete sequence of all 115 queries

Experiment – Complete Sequence



Experiment – Complete Sequence



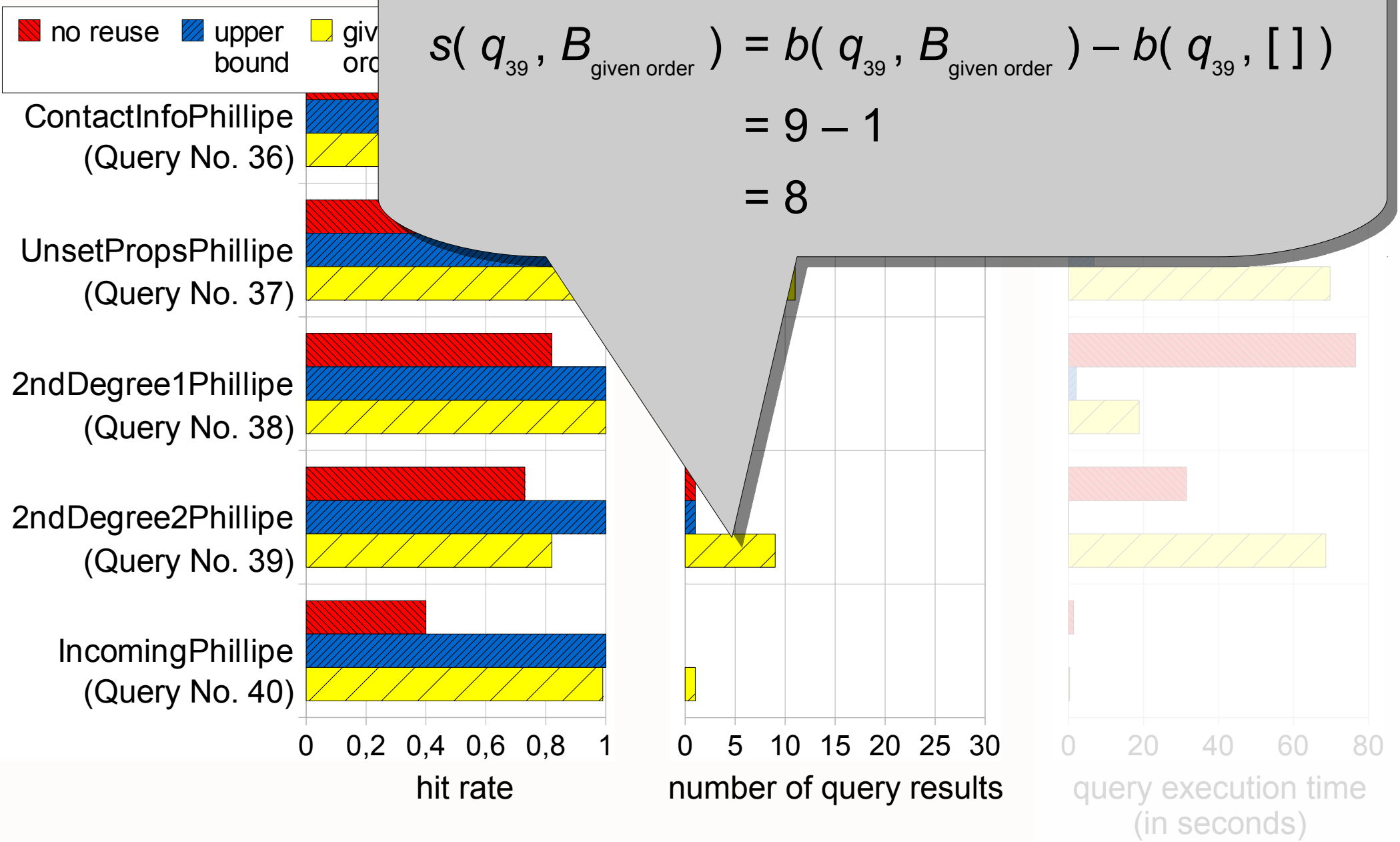
Experiment – Complete Sequence

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

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

$$= 9 - 1$$

$$= 8$$



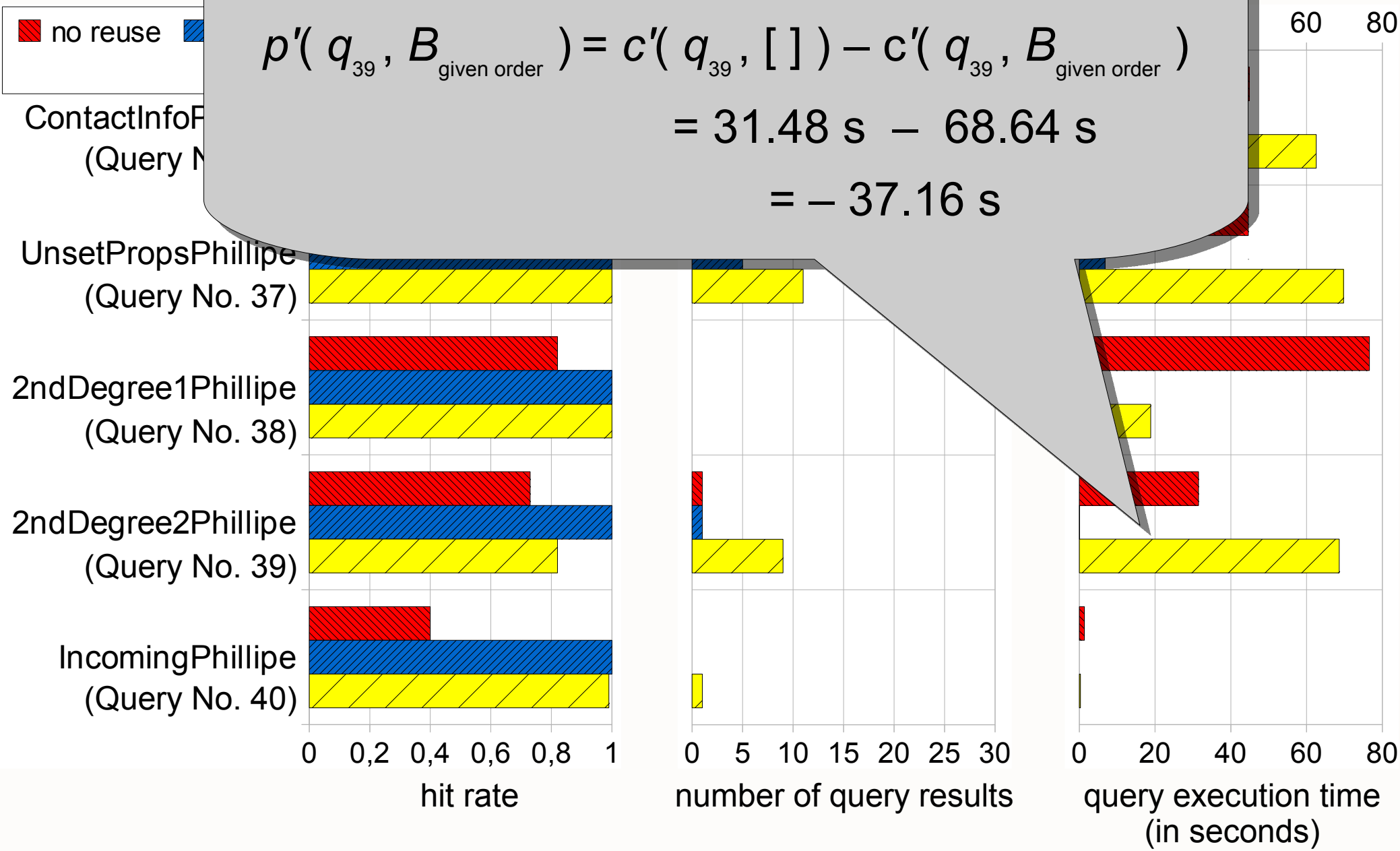
Experiment – Complete Sequence

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

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

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

$$= -37.16 \text{ s}$$



Experiment – Complete Sequence

Experiment	Avg.[1] number of **Query Results** (std.dev.)	Average[1] **Hit Rate** (std.dev.)	Avg.[1] query **Execution Time** (std.dev.)
no reuse	4.983 (11.658)	0.576 (0.182)	30.036 s (46.708)
upper bound	5.070 (11.813)	0.996 (0.017)	1.943 s (11.375)
given order	6.878 (12.158)	0.932 (0.139)	39.845 s (145.898)

[1] Averaged over all 115 queries

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

Experiment – Complete Sequence

Experiment	Avg.[1] number of **Query Results** (std.dev.)	Average[1] **Hit Rate** (std.dev.)	Avg.[1] query **Execution Time** (std.dev.)
no reuse	4.983 (11.658)	0.576 (0.182)	30.036 s (46.708)
upper bound	5.070 (11.813)	0.996 (0.017)	1.943 s (11.375)
given order	6.878 (12.158)	0.932 (0.139)	39.845 s (145.898)
random orders	6.652 (11.966)	0.954 (0.036)	36.994 s (118.700)

- **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