Felix Heine et. al.: Processing Complex RDF Queries over P2P Networks
Seminar on “Peer-to-Peer Networks” - Summer Term 2009

Johannes Schwenk

Computer Networks and Telematics

Department of Computer Science
University of Freiburg

July 30, 2009
Goal & Motivation

- Evaluate large, *distributed* collections of information by machines (Grids, Semantic Web, ...)
- The presented approach aims to:
  - Allow to reason about the information
  - Dynamically integrate heterogeneous information
  - Provide highly expressive logic and sophisticated reasoning features
  - Scale efficiently on the amount of information and complexity of the query
Scenario

Figure: Scenario (adapted from [3])

- Union of local and network knowledge $\rightarrow$ model graph
Node Architecture

Figure: Node architecture (from [3])

Johannes Schwenk
CoNe - Dept. Computer Science - University of Freiburg
Felix Heine et. al.: Processing Complex RDF Queries over P2P Networks
Definitions

- $T_Q$: query graph
- $T_M$: model graph
- $V_Q$: variables in $T_Q$
- $C_T(t)$: candidate set for $t \in T_Q$
- $C_V(v)$: candidate set for $v \in V_Q$
- $C_V(v) := \Delta$: candidate set for $v$ is undefined
- $C_V(v) := \inf \iff C_V(v) = \Delta$
- $C_V(x) := \{x\} \iff x$ is a label
Specification grade of a triple $t \ (1)$

Simplest version in the papers:

$$sg_1(x) = |C_V(x)|$$

For the whole triple:

$$sg_1(\langle s, p, o \rangle) = \min(sg_1(s), sg_1(p), sg_1(o))$$

More on the optimized versions later!
Determination of candidate sets (1)

function candidates(\(T_Q, T_M\))

set each \(C_T(t)\) and \(C_V(v)\) to \(\Delta\)

while there is an undefined \(C_T(t)\)

determine a triple \(t = \langle s, p, o \rangle\) where

\[C_T(t) = \Delta, \text{ and } sg(t) \leq sg(t') \forall t' \text{ with } C_T(t') = \Delta\]

if \(sg(t) = sg(s)\)

\[C_T(t) := \bigcup_{x \in C_V(s)} \text{getBySubject}(x)\]

\[C_T(t) := \{\langle s, p, o \rangle | C_T(t) : p \in C_V(p), o \in C_V(o)\}\]

else (similar code for predicate and object)

if refine(\(C_T, C_V, \{t\}, \emptyset\)) = error

return error

\(T_Q\): query graph
\(T_M\): model graph
\(V_Q\): variables in \(T_Q\)
\(C_T(t)\): candidate set for \(t \in T_Q\)
\(C_V(v)\): candidate set for \(v \in V_Q\)
\(C_V(v) := \Delta\): undefined
\(C_V(v) := \inf \iff C_V(v) = \Delta\)
\(C_V(x) := \{x\} \iff x\text{ is a label}\)
Determination of candidate sets (2)

- $S_g$ in the first iteration as example:
- Chose e.g. $t^Q_1 = \langle \text{?res}, \text{pc2:hasProcessor}, \text{blank1} \rangle$
- $sg_1(t^Q_1) = min(sg_1(\text{?res}), sg_1(\text{pc2:hasProcessor}), sg_1(\text{blank1})) = min(\infty, 1, 1) = 1$

$T_Q$: query graph
$T_M$: model graph
$V_Q$: variables in $T_Q$
$C_T(t)$: candidate set for $t \in T_Q$
$C_V(v)$: candidate set for $v \in V_Q$
$C_V(v) := \Delta$: undefined
$C_V(v) := \inf \iff C_V(v) = \Delta$
$C_V(x) := \{x\} \iff x$ is a label
Determination of candidate sets

**getByPredicate**(pc2:hasProcessor):

- Set $C_T(t) := \bigcup_{x \in C_V(p)} \text{getByPredicate}(x)$
  - $C_T(t) := \{\text{getByPredicate}(pc2:hasProcessor)\}$
  - $C_T(t) := \{t_1^M = \langle pc2:sfb, pc2:hasProcessor, blank1 \rangle\}$

- Set $C_T(t) := \{\langle s, p, o \rangle \in C_T(t) : s \in C_V(s), o \in C_V(o)\}$
  - Nothing changes here!
Determination of candidate sets

**Function candidates** \( T_Q, T_M \)

set each \( C_T(t) \) and \( C_V(v) \) to \( \Delta \)

**While** there is an undefined \( C_T(t) \)

determine a triple \( t = \langle s, p, o \rangle \) where

\[
C_T(t) = \Delta, \text{ and} \quad sg(t) \leq sg(t') \forall t' \text{ with } C_T(t') = \Delta
\]

**If** \( sg(t) = sg(s) \)

\[
C_T(t) := \bigcup_{x \in C_V(s)} \text{getBySubject}(x)
\]

\[
C_T(t) := \{ \langle s, p, o \rangle \in C_T(t) : p \in C_V(p), o \in C_V(o) \}
\]

**Else** *(similar code for predicate and object)*

**If** \( \text{refine}(C_T, C_V, \{ t \}, \emptyset) = \text{error} \)

return error
Refinement (1)

function refine($C_T, C_V, T, V$)

while $V \neq \emptyset$ or $T \neq \emptyset$

for each $t = \langle s, p, o \rangle \in T$

if $s \in V$

\[
C_V(s) := C_V(s) \cap \text{subject}(C_T(t))
\]

if $C_V(s)$ has been changed

\[
V := V \cup \{s\}
\]

end if

end if

similar code for predicate and object

$T := T - \{t\}$

end for
Refinement (2)

function refine($C_T, C_V, T, V$)

while $V \neq \emptyset$ or $T \neq \emptyset$

... 

for each $v \in V$

for each $t \in T$

if subject($t$) = $v$

$C_T(t) := \{ (s', p', o') \in C_T(t) : s' \in C_V(v) \}$

if $C_T(t)$ has been changed

$T := T \cup \{ t \}$

end if

end if

similar code for predicate and object

end for

$V := V - \{ v \}$
Refinement (3)

function refine($C_T$, $C_V$, $T$, $V$)

while $V \neq \emptyset$ or $T \neq \emptyset$

... 

end for

if some $C_V(v)$ or $C_T(t)$ is empty

return error

end if

end while

return ok

end function

$T_Q$: query graph
$T_M$: model graph
$V_Q$: variables in $T_Q$
$C_T(t)$: candidate set for $t \in T_Q$
$C_V(v)$: candidate set for $v \in V_Q$
$C_V(v) := \Delta$: undefined
$C_V(v) := \inf \Leftrightarrow C_V(v) = \Delta$
$C_V(x) := \{x\} \Leftrightarrow x$ is a label
$\mathcal{V}$: all variables
Determination . . . (again)

**function** candidates($T_Q, T_M$)

set each $C_T(t)$ and $C_V(v)$ to $\Delta$

**while** there is an undefined $C_T(t)$

determine a triple $t = \langle s, p, o \rangle$ where

$C_T(t) = \Delta$, and

$s g(t) \leq s g(t') \forall t'$ with $C_T(t') = \Delta$

**if** $s g(t) = s g(s)$

$C_T(t) := \bigcup_{x \in C_V(s)} \text{getBySubject}(x)$

$C_T(t) := \{ \langle s, p, o \rangle \in C_T(t) : p \in C_V(p), o \in C_V(o) \}$

**else** *(similar code for predicate and object)*

**if** refine($C_T, C_V, \{ t \}, \emptyset$) = error

**return** error
Evaluation

**function** evaluate($T_Q, C_T, V$)

if there is a $t \in T_Q$

remove $t$ from $T_Q$

for each $u \in C_T(t)$

if $u$ does not contradict $V$

$V' := V$

store the variable assignment of $u$ in $V'$

evaluate($T_Q, C_T, V'$)

end if

end for

else

the variable assignments in $V$ are a match

end if

end function

$T_Q$: query graph

$T_M$: model graph

$V_Q$: variables in $T_Q$

$C_T(t)$: candidate set for $t \in T_Q$

$C_V(v)$: candidate set for $v \in V_Q$

$C_V(v) := \Delta$: undefined

$C_V(v) := \inf \iff C_V(v) = \Delta$

$C_V(x) := \{x\} \iff x$ is a label
Specification grade of a triple \( t \) (1)

Different versions in the papers!

Example: version with caching and Bloom filters

\[
sgs(x) = \begin{cases} 
\sum_{s \in CV(x)} cntBySubject(s) : CV(x) \neq \Delta \\
\infty : CV(x) = \Delta 
\end{cases}
\]

Similar for predicate and object

\[
sgA/ b(\langle s, p, o \rangle) = \min(sgs(s), sgp(p), sgo(o))
\]
Bloom filters in `getBy[...]` and `cntBy[...]` functions

**Figure:** Bloom filter (source: Wikipedia, public domain)
Results

- Compared to Sesame (Semantic Web Toolkit): about 10-20 times as fast
- Simulations and measurements with FreePastry based prototype
- Simulations: 60,000 RDF triples, 100 queries
- Measurements on cluster: 60,000 RDF triples, 100 queries, different node counts
- Optimizing caching of candidate sets → tradeoff between network lookups and message size
# Results

<table>
<thead>
<tr>
<th>Variant</th>
<th>Message Size</th>
<th>Lookups</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>sg1</td>
<td>461062</td>
<td>1043</td>
<td>0</td>
</tr>
<tr>
<td>sg1/b</td>
<td>71138</td>
<td>1043</td>
<td>1144</td>
</tr>
<tr>
<td>sg2</td>
<td>20716</td>
<td>12395</td>
<td>0</td>
</tr>
<tr>
<td>sg2/b</td>
<td>16700</td>
<td>12395</td>
<td>1065</td>
</tr>
<tr>
<td>sg3</td>
<td>11324</td>
<td>3003</td>
<td>0</td>
</tr>
<tr>
<td>sg3/b</td>
<td>7308</td>
<td>3003</td>
<td>1065</td>
</tr>
<tr>
<td>sg4</td>
<td>98373</td>
<td>2450</td>
<td>1643</td>
</tr>
<tr>
<td>sg4/b</td>
<td>6602</td>
<td>2447</td>
<td>2702</td>
</tr>
</tbody>
</table>

**Table:** Comparison of the sg versions (from [2])
Figure: Results for prototype architecture on ARMINUS cluster (from [3])
Figure: Empirical analysis: Ratio of triples sent by peers for Top 10 search divided by triples sent for exhaustive search (from [1])
Thank you for your attention!
Top k rdf query evaluation in structured p2p networks.

[2] Heine, F.
Scalable p2p based rdf querying.
In InfoScale ’06: Proceedings of the 1st international conference on Scalable information systems (New York, NY, USA, 2006), ACM.

Processing complex rdf queries over p2p networks.
<table>
<thead>
<tr>
<th>References</th>
</tr>
</thead>
</table>

Johannes Schwenk  
CoNe - Dept. Computer Science - University of Freiburg

Felix Heine et. al.: Processing Complex RDF Queries over P2P Networks
RDF Graph

Figure: RDF graph for the sfb cluster (from [3])

Johannes Schwenk
CoNe - Dept. Computer Science - University of Freiburg

Felix Heine et. al.: Processing Complex RDF Queries over P2P Networks
RDFS Graph

Figure: Graph for schema knowledge (from [3])
Model Graph, $T_M$

Figure: Union of local and schema knowledge (from [3])
Triples stored in the DHT

- \( t_1^M = \langle \text{pc2:sfb, pc2:hasProcessor, blank1} \rangle \)
- \( t_2^M = \langle \text{pc2:sfb, rdf:type, pc2:Cluster} \rangle \)
- \( t_3^M = \langle \text{blank1, rdf:type, pc2:Itanium2} \rangle \)
- \( t_4^M = \langle \text{pc2:sfb, pc2:hasOS, blank2} \rangle \)
- \( t_5^M = \langle \text{blank2, pc2:kernelVersion, 2.4} \rangle \)
- \( t_6^M = \langle \text{blank2, rdf:type, pc2:Debian} \rangle \)

- Distributed by hash function to \( n = 6 \) nodes:
- node\( _i \) stores triples \( t_i^M, t_{i-1}^M, t_{i+1}^M, \ i = mod(0 \ldots 6, 6) \)
Query Graph, $T_Q$

Figure: RDF query graph (from [3])
Example data

Query Graph, $T_Q$

The query graph as written triples:

- $t_1^Q = \langle ?\text{res}, \text{pc2:hasProcessor}, \text{blank1} \rangle$
- $t_2^Q = \langle ?\text{res}, \text{rdf:type}, \text{pc2:Cluster} \rangle$
- $t_3^Q = \langle \text{blank1}, \text{rdf:type}, \text{pc2:64Bit} \rangle$
- $t_4^Q = \langle ?\text{res}, \text{pc2:hasOS}, \text{blank2} \rangle$
- $t_5^Q = \langle \text{blank2}, \text{rdf:type}, \text{pc2:Linux} \rangle$
- $t_6^Q = \langle \text{blank1}, \text{rdf:type}, \text{pc2:Intel} \rangle$

In plain english: “All clusters that have 64Bit Intel CPUs and run Linux as operating system.”