QueryPIE: Backward reasoning for OWL Horst over very large knowledge bases

Jacopo Urbani <jacopo@cs.vu.nl> - Vrije Universiteit Amsterdam
QueryPIE in one slide
QueryPIE

- QueryPIE is a hybrid reasoning engine that can scale to a billion triples

- It greatly reduces the costs of backward-chaining by performing a small precomputation
  - Precomputation: 8-300sec against 1-3hours of WebPIE
  - Query response time: in the order of few milliseconds both with and without reasoning
QueryPIE in 17 slides
Reasoning on large data

- Rule-based reasoning = process that applies a set of rules recursively

- Example of rule: 
  
  if <a type B> 
  and <B subclassOf C> 
  then <a type C>

- We can perform reasoning beforehand (forward-chaining inference) or at query time (backward-chaining inference)
Forward/Backward-chaining inference

- **Forward-chaining reasoning**
  - **Pros:**
    - Querying the data becomes a “database” problem
  - **Cons:**
    - Difficult to apply if data changes frequently

- **Backward-chaining reasoning**
  - **Pros:**
    - Reasoning performed only when necessary
  - **Cons:**
    - Even for small queries, it requires expensive reasoning
Currently, the state of the art is forward-chaining

Forward-chaining methods (one for all is WebPIE) can scale to very large extend

However, the intrinsic disadvantages of forward-chaining still remain!

QueryPIE:
  - Combine the advantages of both approaches through hybrid reasoning
  - Get closer to reasoning that is service-oriented and more “webby”
QueryPIE
Hybrid reasoning

- The input of our method is a generic triple pattern (e.g. "<s?p&o>"

- In QueryPIE we apply forward-chaining reasoning only to derive the schema (or terminological) triples. All the other triples are derived at query-time.

- **Terminological triples:**
  - Definition: triples that have a RDFS/OWL term as predicate and/or object and are used in the inference rules.
  - e.g:
    - (x rdfs:subClassOf ?y)
    - (x rdfs:subPropertyOf ?y)
    - (x rdf:type owl:TransitiveProperty)
    - etc.
Hybrid reasoning

- Why does it make sense to pre-compute the closure on the schema?
  - Schema does not change often
  - It is relatively small compared to the rest of the data
  - It is frequently used in the reasoning rules

- We define two algorithms:
  - *terminological-closure* (executed before query-time)
  - *terminology-independent reasoning* (executed during query-time)
Hybrid reasoning terminology-independent reasoning

**Terminology-independent reasoning**

- implements classical backward-chaining algorithm without performing inference on the schema
- some optimizations are possible only because the schema is pre-computed
  1) prune the reasoning tree by using the inferred schema (very important)
  2) replicate the schema and perform most of the joins where the data is
Terminological-closure

- **Problem**: cannot perform a partial closure in a forward-chaining fashion
- **Solution**: Exploit the terminology-independent reasoning to achieve this goal
- **Proposed algorithm**:
  1) Assume that the schema inference is already computed
  2) Query the engine asking for the schema
  3) Add the inferred triples to the input and repeat point 2 until fix point

In other words we perform forward reasoning with “backward steps”
Hybrid reasoning
Implementation

What about the implementation of QueryPIE?

- Java distributed prototype that uses the Ibis framework for the node communication
- Triples are indexed with 4 indexes (spo, sop, pos, ops) and partitioned across a set of nodes
- Algorithm:
  1) One node receives in input a generic triple pattern (?x ?y ?z)
  2) This node generates a reasoning tree that might derive new data and send it to the nodes where the data is
  3) The data is collected into end location and returned to the user
Evaluation
Hybrid reasoning Evaluation

- Our goal was to compare the performance against the current state of the art without external overhead
  - **Full materialization scenario:** calculated full materialization (with WebPIE) and loaded the data in QueryPIE *deactivating* reasoning
  - **Hybrid reasoning scenario:** loaded data in QueryPIE and calculated the partial closure and perform reasoning at query-time

- Used the same hardware infrastructure for both scenario’s: 8 machines, each 2 CPUs and 24G main memory

- Considered 3 datasets:
  - LDSR (860M triples),
  - LLD (700M triples)
  - LUBM (1.1B triples)
Terminological closure algorithm consistently reduces the complexity of pure backward-chaining:

<table>
<thead>
<tr>
<th>Input pattern (LUBM pattern queries)</th>
<th>#leaves hybrid/backward-chaining</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>?x ?y University</td>
<td>21/174</td>
<td>8.29</td>
</tr>
<tr>
<td>University0 hasAlumnus ?x</td>
<td>5/58</td>
<td>11.6</td>
</tr>
<tr>
<td>?x rdf:type ResearchGroup</td>
<td>2/3</td>
<td>1.5</td>
</tr>
<tr>
<td>UndergradStud0 rdf:type ?x</td>
<td>38/291</td>
<td>7.66</td>
</tr>
</tbody>
</table>
Hybrid reasoning Evaluation

- Terminological closure algorithm: Much faster than full materialization

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Terminologica l-closure</th>
<th>Full materialization (WebPIE)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUBM</td>
<td>8sec</td>
<td>1h15min</td>
<td>562</td>
</tr>
<tr>
<td>LLD</td>
<td>332sec</td>
<td>1h5min</td>
<td>11</td>
</tr>
<tr>
<td>FactForge (LDSR)</td>
<td>89sec</td>
<td>2h45min</td>
<td>111</td>
</tr>
</tbody>
</table>
Hybrid reasoning Evaluation

- terminology-independent algorithm: Overhead of reasoning not noticeable for the user

<table>
<thead>
<tr>
<th>Query</th>
<th># Results</th>
<th>Time w/reas.</th>
<th>Time w/o reas.</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>?x type CompactCar</td>
<td>182</td>
<td>3.38ms</td>
<td>3.32ms</td>
<td>1.02</td>
</tr>
<tr>
<td>?y ?x University</td>
<td>75613</td>
<td>55.12ms</td>
<td>35.75ms</td>
<td>1.54</td>
</tr>
<tr>
<td>?x uniprot:pathway… ?y</td>
<td>4</td>
<td>8.17ms</td>
<td>1.10ms</td>
<td>7.43</td>
</tr>
<tr>
<td>Arnold_Swart… ?x ?y</td>
<td>4937</td>
<td>8.57ms</td>
<td>2.86ms</td>
<td>2.99</td>
</tr>
<tr>
<td>?x rdf:type ResearchGroup</td>
<td>2400836</td>
<td>1166ms</td>
<td>1017ms</td>
<td>1.15</td>
</tr>
</tbody>
</table>
Take home message
Hybrid reasoning
Conclusions

- on the fly inference can be done up to very large scale, using hybrid reasoning

- hybrid reasoning is more convenient than full-materialization in case data changes frequently

- QueryPIE is still work in progress
  - Move from triple patterns to SPARQL
  - Move from OWL Horst to OWL 2 RL