Distributed Query Processing in the Presence of Blank Nodes

Audun Stolpe, Senior Scientist
Jonas Halvorsen, Scientist

ISWC 2018, 12.10.2018
Background: NNEC

JOC — Joint Operations Center

- Analyst is monitoring extraction points for evacuation flights.
- Some flights are threatened by hostile activity.

So….

- Any friendly forces near enough to respond?
- Are they engaged or available?
- What kind of capabilities do they have?
Query: Find extraction points/planned evac missions s. t.
Query: Find extraction points/planned evac missions s. t.
1. the mission is threatened by hostile forces
Query: Find extraction points/planned evac missions s. t.
1. the mission is threatened by hostile forces
2. friendly forces are within responding distance
Query: Find extraction points/planned evac missions s. t.
1. the mission is threatened by hostile forces
2. friendly forces are within responding distance
3. with capabilities to counter the threat
Essentially an OBDI+ problem:

- how to process selection as if it were one source?
Essentially an OBDI+ problem:

- how to process selection as if it were one source?
- + how to adapt to a dynamic coalition topology.
OBDI+

• Hide the sources behind a domain theory/ontology
  – We used weakly recursive datalog

• Rewrite the global query into a union of conjunctive queries

• Distribute each CQ over the sources with a FedEx-style federator:
  – FedEx-style:
    • use the properties of triple patterns to route them to sources
    • keep exclusive groups intact
    • break everything else into singleton triples
    • evaluate every subpattern against every source that might answer it
DNS Service Discovery
DNS Service Discovery
DNS Service Discovery

ontology

rewriter

federator
Discoveries

• Worked fine with one instance of each type of source
• When multiple instances were added, we started to leak results:
  – and not just cross-site joins, but solutions derived from a single source
  – i.e. federation did worse than simple round robin query evaluation.

• Why?
  – with >1 source in the same vocab, there are no exclusive groups
  – hence, the global query is decomposed into singletons
  – each singleton is evaluated in a separate execution context
  – which acts as a sealed scope for blank nodes
  – hence joins on blank nodes will be standardized apart
The problem in a nutshell

Source A

Source B
The problem in a nutshell

Source A

Source B

Query decomposition 1:

FFI
The problem in a nutshell

- Misses cross-site join!

Source A

- :location

Source B

- :time

Query decomposition 1:

- :location
- :time

Matches b-connected set
The problem in a nutshell

Captures cross-site join

Misses b-connected sets!

Query decomposition 1:
The two horns of federation

If blank nodes occur, decomposition is delicate:

Too fine;

Too coarse;

single-source solutions may be lost
cross-site joins may be lost
Plausible responses

1. Don’t use blank nodes:
   – A bit late for that
   – Incurs serious expressive limitations
     • T-Box concepts
     • lists and containers
     • reification
     • n-ary relations

2. Replace blank nodes with Skolem constants:
   – may break isomorphism and entailment (canonical naming a possibility)
   – potentially problematic if the Skolem constants are reused elsewhere

3. Formalize the semantics of federated SPARQL processing, and see what can be done.
Taking option 3: Federation semantics.

- what is this the general situation?
- what is the semantics of sound and complete query federation?
- what does soundness and completeness even mean?

**Def. completeness:**
Fed. result = eval against merge of the sources.
Recap

In this example, both partitions must be evaluated to extract the two solutions.

Does it generalize?
It does: There is an exhaustive (wrt. solutions) map from a subset of all partitions (the productive partitions) to the powerset of solutions.
Uh oh!

- productive partitions are *characteristic* of sound and complete fed.
- productive partitions are unrecognizable in the 0-knowledge case
- thus completeness requires evaluation of all partitions
- but alas, \#partitions of size \( n \) set = Bell number of \( n \)

\[
B(10) = \sum_{k=0}^{9} \binom{9}{k} B(k) = 115,975
\]
Countermeasures

1. Leverage exclusive groups.

where \( k = \text{#remainder} \)
2. Interleave partition generation algorithm with probes for blanks.

Observation: every element in a productive partition is a b-component.
Contrapositively:

• not a b-component, not an element of a productive partition
• so computing a partition that contains it is wasted resources

Now,

• non-b-components can be excluded by cheap ASK probes
• Which in effect disqualify entire partitions

Heuristic idea:

• employ a restricted growth algorithm for partition generation
• probe the fringe
• propagate constraints
Probe rejects subset
All partitions below are disqualified
| Query | n  | Bell | Templ. | $|E(P)|$ | $|P \setminus \bigcup E(P)|$ | Bell,g | $\Delta$-checks / $|\text{hits}|$ | $\subseteq$-hits | $\Delta$-checks / hits | $\subseteq$-hits | Partitions |
|-------|----|------|--------|------|----------------|--------|----------------|-------------|----------------|-------------|-----------|
| F1    | 6  | 203  | T1     | 0    | 6            | 203    | 31/5 / 19     | 33/7 / 41   | 53 / 5       | 52 / 5      | 52        |
|       |    |      | T2     | 2    | 2            | 15     | 7/3 / 1       | 8/4 / 2     | 13 / 2       | 13 / 2      | 13        |
|       |    |      | T3     | 1    | 5            | 203    | 31/5 / 19     | 33/7 / 38   | 50 / 5       | 50 / 5      | 50        |
| F3    | 6  | 203  | T1     | 0    | 6            | 203    | 20/9 / 10     | 23/12 / 30  | 40 / 10      | 40 / 10     | 40        |
|       |    |      | T2     | 2    | 0            | 2      | 1/1 / -       | 1/1 / -     | 1 / -        | 1 / -       | 1         |
|       |    |      | T3     | 1    | 4            | 52     | 16/5 / 5      | 16/5 / 10   | 21 / 10      | 21 / 10     | 21        |
| F5    | 6  | 203  | T1     | 0    | 6            | 203    | 21/10 / 11    | 23/12 / 29  | 38 / 11      | 38 / 11     | 38        |
|       |    |      | T2     | 2    | 0            | 2      | 1/1 / -       | 1/1 / -     | 2 / -        | 2 / -       | 2         |
|       |    |      | T3     | 1    | 4            | 52     | 15/4 / 5      | 16/5 / 11   | 19 / 11      | 19 / 11     | 19        |
| C1    | 8  | 4140 | T1     | 0    | 8            | 4140   | 33/22 / 20    | 38/27 / 81  | 94 / 15      | 94 / 15     | 94        |
|       |    |      | T2     | 2    | 0            | 2      | 1/1 / -       | 1/1 / -     | 2 / -        | 2 / -       | 2         |
|       |    |      | T3     | 1    | 4            | 52     | 15/4 / 5      | 16/5 / 12   | 18 / 12      | 18 / 12     | 18        |
| C2    | 10 | 115975 | T1    | 0    | 10           | 115975 | 54/50 / 95    | 54/50 / 95  | 54 / 50      | 54 / 50     | 54        |
|       |    |      | T2     | 2    | 0            | 2      | 1/1 / -       | 1/1 / -     | 1 / -        | 1 / -       | 1         |
|       |    |      | T3     | 1    | 3            | 15     | 7/3 / 1       | 8/4 / 2     | 7 / 2        | 7 / 2       | 7         |
Possibilities

• #partitions drops steeply with knowledge, exploit
  – SHACL schemas
  – VOID descriptions
  – Resource shapes
  – Summaries

• Improve the heuristics
  – Ordering of triples has a tangible effect
  – ........

Semantic fidelity and federation may be compatible in realistic cases.