Toward Analytics for RDF Graphs

Ioana Manolescu

INRIA and Ecole Polytechnique, France
ioana.manolescu@inria.fr
http://pages.saclay.inria.fr/Ioana.Manolescu

Joint work with D. Bursztyn, S. Cebiric (Inria),
F. Goasdoué (U. Rennes 1)

University of Copenhagen, August 2017
Outline

1. Background: semantic RDF graphs
2. Efficient RDF query answering in the presence of ontologies [EDBT2015, VLDB 2016]
3. Summarizing semantic-rich RDF graphs [VLDB 2015, TR2017]
Part I

Background: RDF graphs
Big Data needs semantics

AI Magazine, Spring 2015
Do we really need the semantics?

Yes. All the time.

Application knowledge / constraints:
- Every Senator is an ElectedOfficial which is a Person
- (On Wikipedia) being BornInAPlace means being a Person
Do we really need the semantics?

Yes. All the time.

Application knowledge / constraints:
- Every Senator is an ElectedOfficial which is a Person
- (On Wikipedia) being BornInAPlace means being a Person

Without the semantics, we may miss query answers

<table>
<thead>
<tr>
<th>Data</th>
<th>Constraints</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>John is a Senator</td>
<td>Every Senator is a Person</td>
<td>Who is a Person?</td>
</tr>
</tbody>
</table>
Do we really need the semantics?

Yes. All the time.

Application knowledge / constraints:

- Every Senator is an ElectedOfficial which is a Person
- (On Wikipedia) being BornInAPlace means being a Person

Semantic contraints are a compact way of encoding information

“Every ElectedOfficial is a Person” stated only once even if thousands of ElectedOfficials.
Semantics for Web data

Data and metadata on the Web is often structured in **graphs**, e.g., **RDF** (W3C’s Resource Description Framework)

- Famous application: the Linked Open Data cloud (**2017**)
**RDF graph:** set of **triples**

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Triple</th>
<th>Relational notation</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>s rdf:type o</td>
<td>o(s)</td>
<td>&quot;s is an o&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>s p o</td>
<td>p(s, o)</td>
<td>&quot;The p of s is o&quot;</td>
</tr>
</tbody>
</table>
The Resource Description Framework (RDF)

**RDF graph**: set of triples

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Triple</th>
<th>Relational notation</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>s rdf:type o</td>
<td>o(s)</td>
<td>”s is an o”</td>
</tr>
<tr>
<td>Property</td>
<td>s p o</td>
<td>p(s, o)</td>
<td>”The p of s is o”</td>
</tr>
</tbody>
</table>

**Example RDF graph**:

- Resource (URI)
- Blank node
- Literal (string)
- Property

- **Book**
  - publishedIn
  - rdf:type
  - writtenBy

- **doi1**
  - hasTitle
  - hasName

  "1949"

  "El Aleph"

  "J. L. Borges"
The Resource Description Framework (RDF)

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Triple</th>
<th>Relational notation</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>s rdf:type o</td>
<td>o(s)</td>
<td>&quot;s is an o&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>s p o</td>
<td>p(s, o)</td>
<td>&quot;The p of s is o&quot;</td>
</tr>
</tbody>
</table>
The Resource Description Framework (RDF)

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Triple</th>
<th>Relational notation</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>s rdf:type o</td>
<td>o(s)</td>
<td>&quot;s is an o&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>s p o</td>
<td>p(s, o)</td>
<td>&quot;The p of s is o&quot;</td>
</tr>
</tbody>
</table>

Example:

```
“1949”
publishedIn

doi1
hasTitle
“El Aleph”

Book

rdf:type

writtenBy

_:b1

hasName

“J. L. Borges”
```
RDF Schema (RDFS)

Declare **deductive constraints** between classes and properties

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Triple</th>
<th>OWA interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>$c_1$ rdfs:subClassOf $c_2$</td>
<td>$c_1 \subseteq c_2$</td>
</tr>
<tr>
<td>Subproperty</td>
<td>$p_1$ rdfs:subPropertyOf $p_2$</td>
<td>$p_1 \subseteq p_2$</td>
</tr>
<tr>
<td>Domain typing</td>
<td>$p$ rdfs:domain $c$</td>
<td>$\Pi_{\text{domain}}(p) \subseteq c$</td>
</tr>
<tr>
<td>Range typing</td>
<td>$p$ rdfs:range $c$</td>
<td>$\Pi_{\text{range}}(p) \subseteq c$</td>
</tr>
</tbody>
</table>

“Any $c_1$ is also a $c_2$”

Publication ➤ rdfs:subClassOf ➤ Book ➤ rdfs:domain ➤ writtenBy ➤ Person ➤ rdfs:range ➤ hasAuthor ➤ rdfs:subPropertyOf
RDF Schema (RDFS)

Declare deductive constraints between classes and properties

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Triple</th>
<th>OWA interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>c₁ rdfs:subClassOf c₂</td>
<td>c₁ ⊆ c₂</td>
</tr>
<tr>
<td>Subproperty</td>
<td>p₁ rdfs:subPropertyOf p₂</td>
<td>p₁ ⊆ p₂</td>
</tr>
<tr>
<td>Domain typing</td>
<td>p rdfs:domain c</td>
<td>∏_{domain}(p) ⊆ c</td>
</tr>
<tr>
<td>Range typing</td>
<td>p rdfs:range c</td>
<td>∏_{range}(p) ⊆ c</td>
</tr>
</tbody>
</table>

"If two resources are related by p₁, they are also related by p₂"
### RDF Schema (RDFS)

Declare **deductive constraints** between classes and properties

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Triple</th>
<th>OWA interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>$c_1 \text{ rdfs:subClassOf } c_2$</td>
<td>$c_1 \subseteq c_2$</td>
</tr>
<tr>
<td>Subproperty</td>
<td>$p_1 \text{ rdfs:subPropertyOf } p_2$</td>
<td>$p_1 \subseteq p_2$</td>
</tr>
<tr>
<td>Domain typing</td>
<td>$p \text{ rdfs:domain } c$</td>
<td>$\Pi_{\text{domain}}(p) \subseteq c$</td>
</tr>
<tr>
<td>Range typing</td>
<td>$p \text{ rdfs:range } c$</td>
<td>$\Pi_{\text{range}}(p) \subseteq c$</td>
</tr>
</tbody>
</table>

**Diagram:**

```
Publication
  rdfs:subClassOf
  Book
  rdfs:domain
  writtenBy
  Person
  rdfs:range
  hasAuthor
  rdfs:subPropertyOf
```

“Anyone having $p$ is a $c$”
### RDF Schema (RDFS)

Declare **deductive constraints between classes and properties**

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Triple</th>
<th>OWA interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td>$c_1$ rdfs:subClassOf $c_2$</td>
<td>$c_1 \subseteq c_2$</td>
</tr>
<tr>
<td>Subproperty</td>
<td>$p_1$ rdfs:subPropertyOf $p_2$</td>
<td>$p_1 \subseteq p_2$</td>
</tr>
<tr>
<td>Domain typing</td>
<td>$p$ rdfs:domain $c$</td>
<td>$\Pi_{\text{domain}}(p) \subseteq c$</td>
</tr>
<tr>
<td>Range typing</td>
<td>$p$ rdfs:range $c$</td>
<td>$\Pi_{\text{range}}(p) \subseteq c$</td>
</tr>
</tbody>
</table>

**Example:**

- **Publication**
  - subclass of **Book**
  - has author is a value of **writtenBy**

- **Book**
  - domain of **writtenBy**

- **Person**
  - range of **writtenBy**

“Anyone who is a value of $p$ is a $c$”
Open-world assumption and RDF entailment

RDF data model based on the open-world assumption.

Deductive constraints lead to implicit triples: part of the graph even though not explicitly present.
**Open-world assumption and RDF entailment**

**RDF data model** based on the **open-world assumption**.

Deductive constraints lead to **implicit triples**: part of the graph even though not explicitly present.

\[
\text{explicit triples} + \text{entailment rules} \rightarrow \text{implicit triples}
\]
Open-world assumption and RDF entailment

RDF data model based on the open-world assumption.

Deductive constraints lead to implicit triples: part of the graph even though not explicitly present

\[
\begin{array}{c}
\text{explicit triples} \\
+ \\
\rightarrow \\
\text{implicit triples} \\
\text{entailment rules}
\end{array}
\]

Exhaustive application of entailment leads to saturation (closure)
The semantics of an RDF graph $G$ is its saturation $G^\infty$.

Sample instance entailment rules from schema and instance triples:

1. $c_1 \text{rdfs:subClassOf } c_2 \land s \text{ rdf:type } c_1 \vdash_{\text{RDF}} s \text{ rdf:type } c_2$
2. $p_1 \text{rdfs:subPropertyOf } p_2 \land s \text{ } p_1 \text{ o } \vdash_{\text{RDF}} s \text{ } p_2 \text{ o}$
3. $p \text{ rdfs:domain } c \land s \text{ } p \text{ o } \vdash_{\text{RDF}} s \text{ rdf:type } c$
4. $p \text{ rdfs:range } c \land s \text{ } p \text{ o } \vdash_{\text{RDF}} o \text{ rdf:type } c$
Part II

Efficient query answering
Query answering through a relational database management system (RDBMS)

Query answering $\neq$ query evaluation!
Query answering through a relational database management system (RDBMS)

Query answering $\neq$ query evaluation!

**Advantage of using RDBMS**
- Highly efficient and scalable for *query evaluation* on the data stored in the database

**Limitation**
- Still mostly unaware of semantics (except: integrity constraints)

**Two main methods**
- *Saturation*-based query answering
- *Reformulation*-based query answering
Saturation-based query answering

- $q_\infty$ can be computed using an RDBMS
- $G_\infty$ needs time to be computed and space to be stored
- Not suitable for high update rate (data and/or schema triples)
Reformulation-based query answering

RDF Inference Rules

query $q$

query $q^{ref}$

$G$

answer

$G$ can be evaluated in the RDBMS

Robust to updates

Reformulated queries are complex, thus costly to evaluate
Reformulation-based query answering

- $q^{ref}(G)$ can be evaluated in the RDBMS
- Robust to updates
- Reformulated queries are complex, thus costly to evaluate
Reformulation-based query answering

Target reformulation languages for RDF conjunctive queries (CQs):

- **Unions of CQs (UCQs)**

<table>
<thead>
<tr>
<th>Data</th>
<th>Constraints</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>John is a <strong>Senator</strong></td>
<td>Every <strong>Senator</strong> is a <strong>Person</strong></td>
<td>Who is a <strong>Person</strong>?</td>
</tr>
</tbody>
</table>

Reformulated query: Who is a **Person** or a **Senator**?

**Reformulation introduces unions**
Each union term is an alternative reason or proof for a result.
Reformulation-based query answering

Target language for $q^{ref}$:
- **Unions of CQs (UCQs)**

<table>
<thead>
<tr>
<th>Query (CQ)</th>
<th>Reformulated query (UCQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who are the Professors and what are their Publications?</td>
<td>(Professors with their JournalPapers) or (Professors with their ConferencePapers) or (AssistantProfessors with their JournalPapers) or (AssistantProfessors with their ConferencePapers) or (FullProfessors with their ConferencePapers) or (FullProfessors with their JournalPapers)</td>
</tr>
</tbody>
</table>
Reformulation-based query answering

Target language for $q^{\text{ref}}$:

- **semi-conjunctive queries** (joins of unions of atoms) (**SCQs**)

<table>
<thead>
<tr>
<th>Query (CQ)</th>
<th>Reformulated query (SCQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who are the Professors and what are their Publications?</td>
<td>(Professors or AssistantProfessors or FullProfessors) with their (JournalPapers or ConferencePapers)</td>
</tr>
</tbody>
</table>
Reformulation-based query answering

Target language for $q^{ref}$:

- **Unions of CQs (UCQs)**
  - F. Goasdoué, I. Manolescu, A. Roatiş: “Efficient query answering against dynamic RDF databases”, EDBT 2013

- **Joins of single-atom UCQs (SCQs)**

- **Joins of UCQs (JUCQs)**
  - D. Bursztyn, F. Goasdoué, I. Manolescu: “Teaching an RDBMS about ontological constraints”, PVLDB 2016 (for DL-LITE\(_R\) ontologies)
Reformulation-based query answering

Target language for $q^{ref}$:

- **Unions of CQs (UCQs)**
  - F. Goasdoué, I. Manolescu, A. Pattis: “Efficient query answering

**Wait: does SQL syntax mater?!...**

- **Joins of UCQs (JUCQs)**
  - D. Bursztyn, F. Goasdoué, I. Manolescu: “Teaching an RDBMS about ontological constraints”, PVLDB 2016 (for DL-LiteR ontologies)
Reformulation-based query answering

Target language for $q^{ref}$:

- **Unions of CQs (UCQs)**

---

Wait: does SQL syntax matter?!...

Yes. A lot.

From failing to feasible, e.g., 4 orders of magnitude speed-up on 8M triples of DBLP data.
Reformulation into a join of UCQs (JUCQ)

1. Considers a set of reformulation alternatives (not a fixed one)
Reformulation into a join of UCQs (JUCQ)

1. Considers a set of reformulation alternatives (not a fixed one)
2. Uses a cost model for estimating the cost of evaluating $q^{ref}$ through an RDBMS
Reformulation into a join of UCQs (JUCQ)

1. Considers a set of reformulation alternatives (not a fixed one)
2. Uses a cost model for estimating the cost of evaluating $q^{\text{ref}}$ through an RDBMS
3. Chooses the cheapest alternative.
Optimized reformulation into JUCQs

Query answering techniques

Query reformulation

Query $q$

CQ-to-UCQ ref. algo.

Graph $G$

Result

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$

$\equiv$
Cover-based reformulation

Idea

1. Split the query into a cover of potentially overlapping query fragments
2. Each fragment determines a (conjunctive) sub-query
3. Reformulate each fragment into a UCQ
4. Join the reformulations.

This gets translated into an SQL query of the form:

```
WITH ... as refq1, ..., as refq2...
SELECT...
FROM refq1, refq1, ...
WHERE...
```
Given the query

\[ q_1(x, y) : - x \text{ rdf:type } y, \]
\[ \quad x \text{ ub:degreeFrom “http://www.University532.edu”}, \]
\[ \quad x \text{ ub:memberOf “http://www.Department1.University7.edu”} \]

on 100M LUBM data:

<table>
<thead>
<tr>
<th>JUCQ</th>
<th>#reformulations</th>
<th>exec. time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((t_1, t_2, t_3)^{\text{ref}})</td>
<td>2,256</td>
<td>6,387</td>
</tr>
<tr>
<td>((t_1)^{\text{ref}} \boxtimes (t_2)^{\text{ref}} \boxtimes (t_3)^{\text{ref}})</td>
<td>195</td>
<td>1,074,026</td>
</tr>
<tr>
<td>((t_1, t_2)^{\text{ref}} \boxtimes (t_3)^{\text{ref}})</td>
<td>755</td>
<td>1,968</td>
</tr>
<tr>
<td>((t_1)^{\text{ref}} \boxtimes (t_2, t_3)^{\text{ref}})</td>
<td>200</td>
<td>846,710</td>
</tr>
<tr>
<td>((t_1, t_3)^{\text{ref}} \boxtimes (t_2)^{\text{ref}})</td>
<td>568</td>
<td>554</td>
</tr>
<tr>
<td>((t_1, t_2)^{\text{ref}} \boxtimes (t_1, t_3)^{\text{ref}})</td>
<td>1,316</td>
<td>2,734</td>
</tr>
<tr>
<td>((t_1, t_2)^{\text{ref}} \boxtimes (t_2, t_3)^{\text{ref}})</td>
<td>764</td>
<td>2,289</td>
</tr>
<tr>
<td>((t_1, t_3)^{\text{ref}} \boxtimes (t_2, t_3)^{\text{ref}})</td>
<td>576</td>
<td>588</td>
</tr>
</tbody>
</table>
Experiments

LUBM 100 M data

- PostgreSQL 9.3.2
- System A
- System B

1. **UCQ** reformulation
2. **SCQ** reformulation
3. **Exhaustive JUCQ** reformulation
4. **Greedy JUCQ** reformulation (starting from SCQ, break into fragments)
Query answering using PostgreSQL (RDFS ontology)

LUBM 100M; 28 queries; 2 to 6 atoms; 1 to 318,089 union terms
Query answering using System A (RDFS ontology)

LUBM 100 M; 28 queries; 2 to 6 atoms;
1 to 318,089 union terms
Query answering using System B (RDFS ontology)

LUBM 100M; 28 queries; 2 to 6 atoms; 1 to 318,089 union terms
Query answering using PostgreSQL (DL-LITE\(_R\) ontology)

LUBM\(_{20}\) benchmark, 100 M
Some reformulations under RDFS are lossy under DL-LITE\(_R\)
Query answering techniques

Cover-based reformulation for RDFS

Query answering using DB2 (DL-Lite$_R$ ontology)

LUBM$_{20}^3$ benchmark, 100 M
Simple data layout vs. RDF-specific one
▷ M. Bornea, J. Dolby, A. Kementsietsidis et al., “Building an efficient RDF store over a Relational Database”, SIGMOD 2013
Why does DB2/RDF perform so poorly?

This query:
Why does DB2/RDF perform so poorly?

...becomes after reformulation on DB2's RDF-specific store:
RDF query answering through optimized reformulation

1. Equivalent SQL syntaxes are not equal from the RDBMS optimizer perspective (inside or outside well-supported dialect)

2. Chosing the reformulation based on cost model makes queries feasible or efficient when they were not
   - RDBMS optimizers have been tuned for conjunctive queries (historical use-case) ⇒
     The cost-based optimizer sees one CQ at a time!

3. This amounts to enlarging the optimizer’s well-supported dialect, at a very modest performance overhead.
Part III

RDF summarization
RDF summaries

Problem

RDF graph $G$ is large, heterogeneous, partially implicit. How to compactly represent all its structure?
Summary of DBLP data

150M triples
Summary of geographic data

French territory division in regions, departments, urban areas, cities, districts etc.
368K triples
RDF summaries

Solution

1. Define RDF node equivalence relation $\equiv$: equivalence relation such that class and property nodes are only equivalent to themselves.

2. Define RDF summary $G/\equiv$ of an RDF graph $G$ as the quotient of $G$ through $\equiv$.

Recall: quotient of a directed graph $G$ by $\equiv$

$G = (V, E)$, $\equiv$ equivalence relation on $V$

- $G/\equiv$ nodes: one for $\equiv$ equivalence class of $V$
- $G/\equiv$ edges: $n^1_{\equiv} \xrightarrow{a} n^2_{\equiv}$ iff $\exists n_1 \xrightarrow{a} n_2 \in G$ such that $n_1$ represented by $n^1_{/\equiv}$, $n_2$ represented by $n^2_{/\equiv}$
Properties

Formal summary properties

For any RDF equivalence relation \( \equiv \):

**Size limit**  The summary is at most as large as the graph.

**Schema preservation**  The schema of \( G/\equiv \) is the schema of \( G \).

**Representativeness**  Any conjunctive query \( q \) with answers on \( G \) also has answers on its summary:

\[
q(G^\infty) \neq \emptyset \implies q((G/\equiv)^\infty) \neq \emptyset
\]

This enables **query pruning (for query answering)** without saturating \( G \).
Which equivalence relations to use?

Equivalence notions previously studied
- Forward / backward / forward and backward simulation
- Forward / backward / forward and backward bisimulation

Adapted to semantic RDF graphs

Novel equivalence notions we introduce (see next)
- Flexible similarity suited to heterogeneous graphs
- Based on property cliques and possibly on RDF types
Intuition: $a_1, a_2$ are similar; $r_1, r_2, r_3, r_4, r_5$ are similar
RDF node equivalence based on property cliques

Output property cliques: \{a, t, e, c\}; \{r\}; \{p\}; \emptyset
Input property cliques: \{a\}; \{t\}; \{e\}; \{c\}; \{r, p\}; \emptyset
Two nodes are weakly equivalent ($\equiv_W$) iff they have the same input clique or the same output clique.

Weak summary $G/\equiv_W$ of the sample RDF graph $G$: 

[Diagram of RDF graph with nodes labeled Book, Journal, Spec and edges labeled with relations such as r, a, t, c, e, p.]
Clique-based summaries

Two nodes are strongly equivalent ($\equiv_S$) iff they have the same input clique and the same output clique.

Strong summary $G_{\equiv S}$ of the sample RDF graph $G$:
Using types for summarization

Group nodes **first by their types**; then group untyped nodes by their property cliques.

Typed weak summary $G/\equiv_{TW}$ of the sample RDF graph $G$:

On this example, this is also the typed strong summary $G/\equiv_{TS}$.
## Summary sizes

| Graph G | |G| | Summary G/≡ | |G/≡| | cf≡ |
|---------|---|---|---|---|---|---|
| DBLP    | 150,787,464 | G/W | 71 | 2,123,767 |
| DBLP    | 150,787,464 | G/S | 206 | 731,978 |
| DBLP    | 150,787,464 | G/fw | 262,695 | 574 |
| LUBM1M  | 1,227,868 | G/W | 161 | 7,579 |
| LUBM1M  | 1,227,868 | G/S | 207 | 5,903 |
| LUBM1M  | 1,227,868 | G/fw | 1982 | 617 |
| LUBM10M | 11,990,183 | G/W | 162 | 74,013 |
| LUBM10M | 11,990,183 | G/S | 206 | 58,204 |
| LUBM10M | 11,990,183 | G/fw | 24,958 | 480 |
| LUBM10M | 11,990,183 | G/bw | 6,162 | 1,944 |
| LUBM10M | 11,990,183 | G/fb | 11,990,076 | 1 |
Can we summarize $G^{\infty}$ without saturating $G$?

**Shortcut theorem**

For the summaries $G/W$, $G/S$, $G/fw$, $G/bw$, $G/fb$:

$$(G^{\infty})/\equiv \text{ strongly isomorphic to } ((G/\equiv)^{\infty})/\equiv$$

(isomorphic, and identical class and property nodes)

Also: **sufficient condition** for any $\equiv$ to admit the shortcut.

Shortcut may **accelerate summarization** of $G^{\infty}$ by up to $20\times$
Shortcut example
Shortcut counter-example

\[ G \xrightarrow{a} x \]
\[ r_1 \xrightarrow{b} y_1 \]
\[ r_2 \xrightarrow{b} y_2 \]
\[ a \leftrightarrow_d c \]

\[ \frac{G}{TW} \]
\[ \frac{a}{b} \]
\[ \frac{a}{b} \]
\[ a \leftrightarrow_d c \]

\[ (G/TW)_{\infty} = \left( \left( G/TW \right)_{\infty} \right)/TW \]

\[ G^\infty \]
\[ \left[ c \right] \]
\[ \tau \]
\[ r_1 \xrightarrow{a} x \]
\[ r_2 \xrightarrow{b} y_2 \]
\[ a \leftrightarrow_d c \]

\[ \left( G^\infty \right)/TW \]
\[ \left[ c \right] \]
\[ \tau \]
\[ a \leftrightarrow_d c \]
Part IV

Conclusion
RDF graphs with semantics

Semantic rules lead to **implicit data**
Reformulated queries can be very complex ($\cup$, $\Join$)

- Two-stage processing: let the RDBMS handle (just) join fragments
- Ontology-aware (as opposed to “query rewrites”)

Quotient-based summaries represent explicit and implicit graph structure; shortcut for efficiently building $\left(G^\infty\right)_{/\equiv}$. Clique-based summaries much more compact than (bi)simulation-based