# Database Optimization Techniques for Semantic Queries

Ioana Manolescu

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

https://team.inria.fr/oak

SEBD 2015, Gaeta

# Part I

## Motivation and outline

# Querying Semantic data

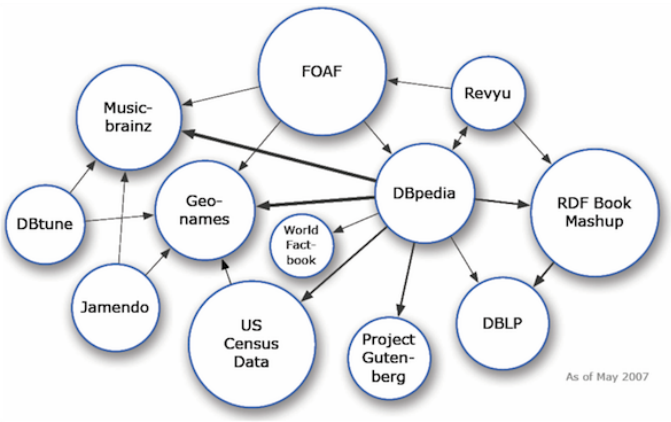
Most popular data model: **RDF** (W3C's Resource Description Framework)

- Famous application: the Linked Open Data cloud

# Querying Semantic data

Most popular data model: **RDF** (W3C's Resource Description Framework)

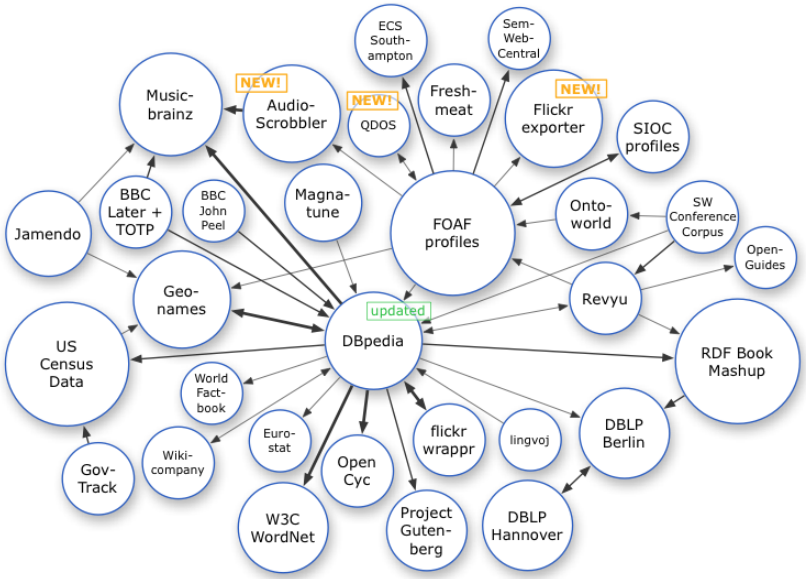
- Famous application: the Linked Open Data cloud (2007)



# Querying Semantic data

Most popular data model: **RDF** (W3C's Resource Description Framework)

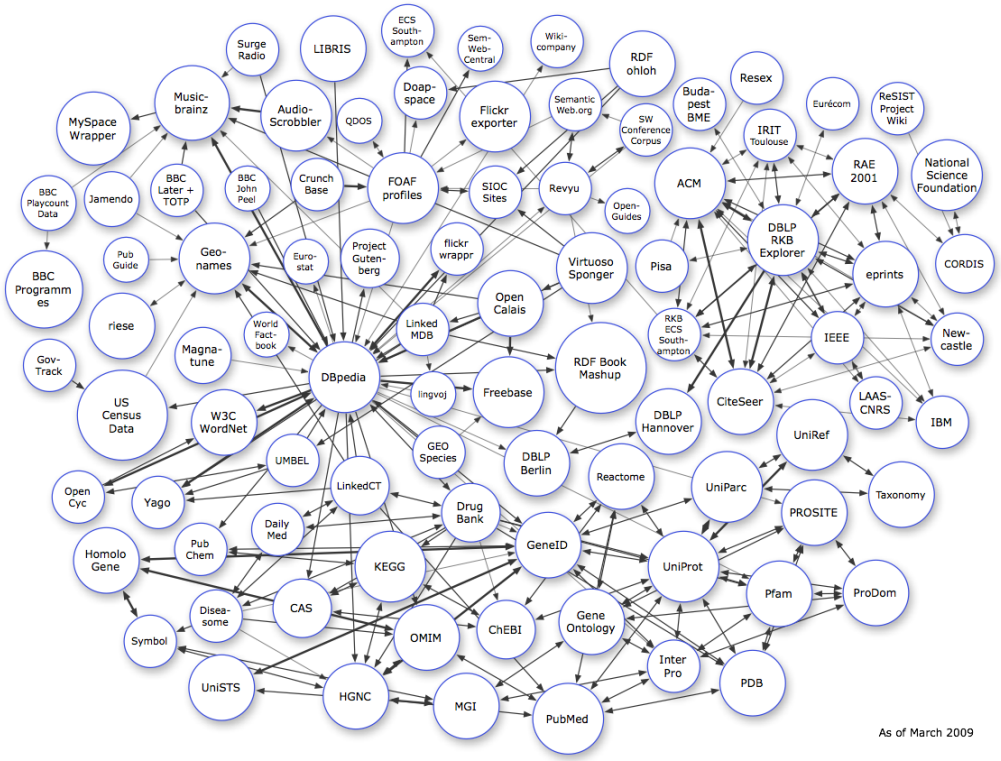
- Famous application: the Linked Open Data cloud (2008)



# Querying Semantic data

Most popular data model: **RDF** (W3C's Resource Description Framework)

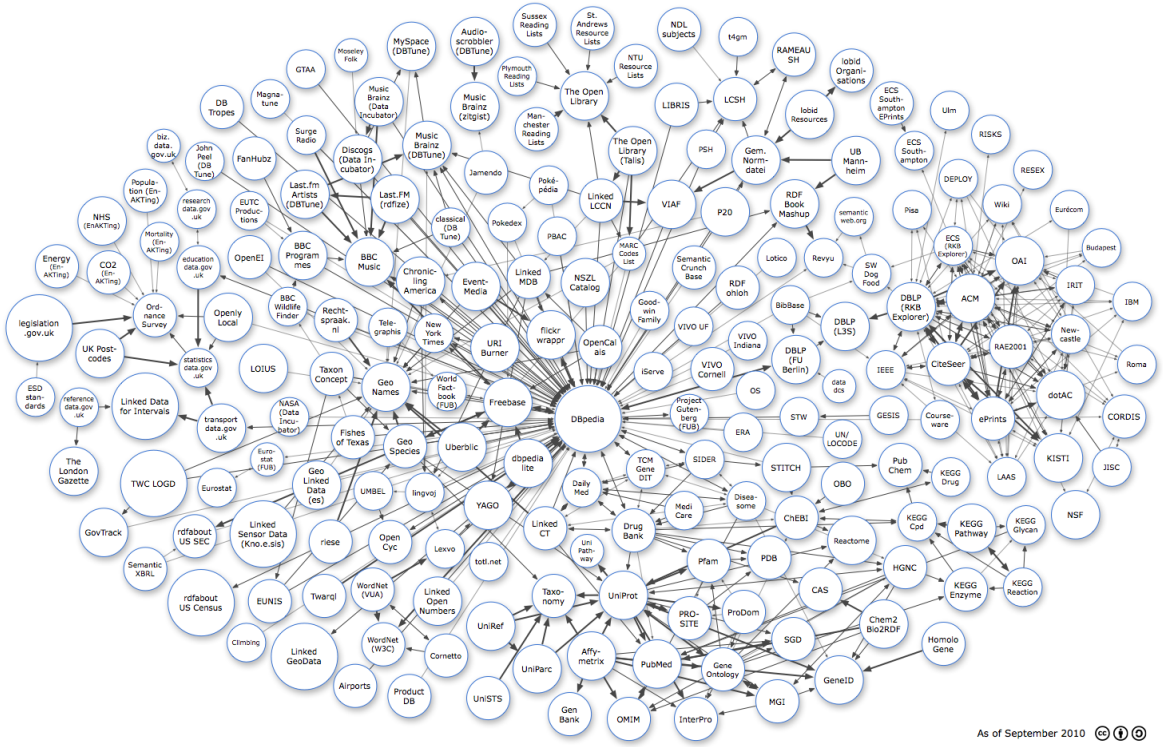
- Famous application: the Linked Open Data cloud (2009)



# Querying Semantic data

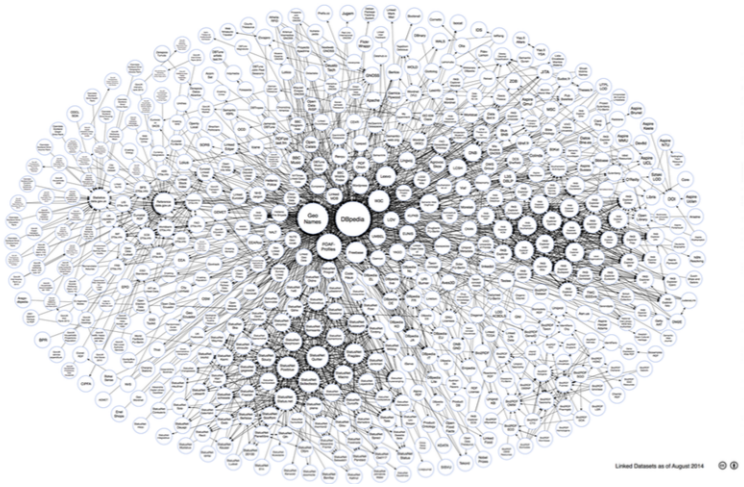
Most popular data model: **RDF** (W3C's Resource Description Framework)

- Famous application: the Linked Open Data cloud (2010)



# Linked Open Data cloud, 2014

900,129 documents describing 8,038,396 resources
(Schmachtenberg, Bizer, Paulheim, ISWC 2014)



| Topic | Datasets | % |
|---|---|---|
| Government | 183 | 18.05 |
| Publications | 96 | 9.47 |
| Life sciences | 83 | 8.19 |
| User-generated | 48 | 4.73 |
| Cross-domain | 41 | 4.04 |
| Media | 22 | 2.17 |
| Geographic | 21 | 2.07 |
| Social web | 520 | 51.28 |
| Total | 1014 | 100 |

There is more (Billion Triple Challenge etc.)

# Querying Semantic Data

1. RDF: three-attribute relation (subject, property, object)
   - The subject is the resource being described
   - The resource has the property property whose value is object
   - Resource type is a property, specified just like any other

# Querying Semantic Data

1. RDF: three-attribute relation (subject, property, object)
   - The subject is the resource being described
   - The resource has the property property whose value is object
   - Resource type is a property, specified just like any other
2. RDFS semantics providing information about the properties and classes (types) of resources:
   - Any undergraduateStudent is a Student
   - Anyone having a graduationDate is a Student (but may also be of other types) ...

# Querying Semantic Data

1. RDF: three-attribute relation (subject, property, object)
   - The subject is the resource being described
   - The resource has the property property whose value is object
   - Resource type is a property, specified just like any other
2. RDFS semantics providing information about the properties and classes (types) of resources:
   - Any undergraduateStudent is a Student
   - Anyone having a graduationDate is a Student (but may also be of other types) ...

Semantics leads to implicit data

# Querying Semantic Data

Main RDFS constraints: rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range

Beyond RDFS

Many (richer) constraint (or ontology) languages

- W3C standards: OWL 2 profile family
- DL Lite family
  ▷ Calvanese, De Giacomo, Lembo, Lenzerini, Rosati, JAR 2007
- Datalog$^{\pm}$
  ▷ Cali, Gottlob Lukasiewick, Marnette, Pierris, LICS 2010

# 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
- The source and destination of a tripFromTo are either a streetAddress, or a cityAddress or a countryAddress

# 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
- The source and destination of a tripFromTo are either a streetAddress, or a cityAddress or a countryAddress

1. Without the semantics, we may miss query answers
2. Semantic contraints are a compact way of encoding information ("every ElectedOfficial is a Person" stated only once)

# Outline

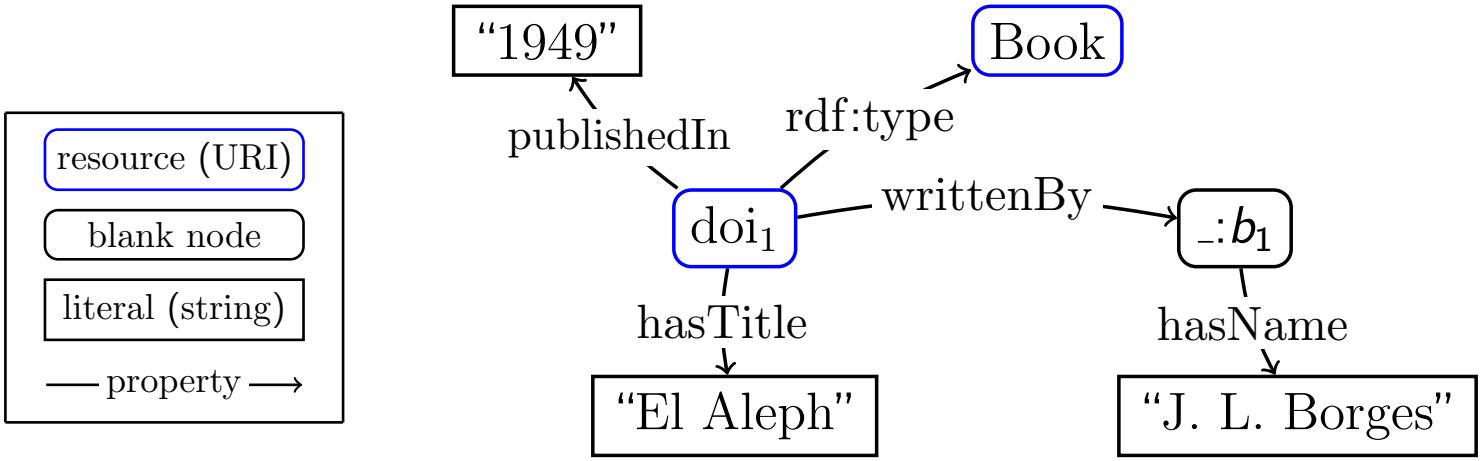
# Part II

# RDF data and queries

# The Resource Description Framework (RDF)

**RDF graph** – set of triples

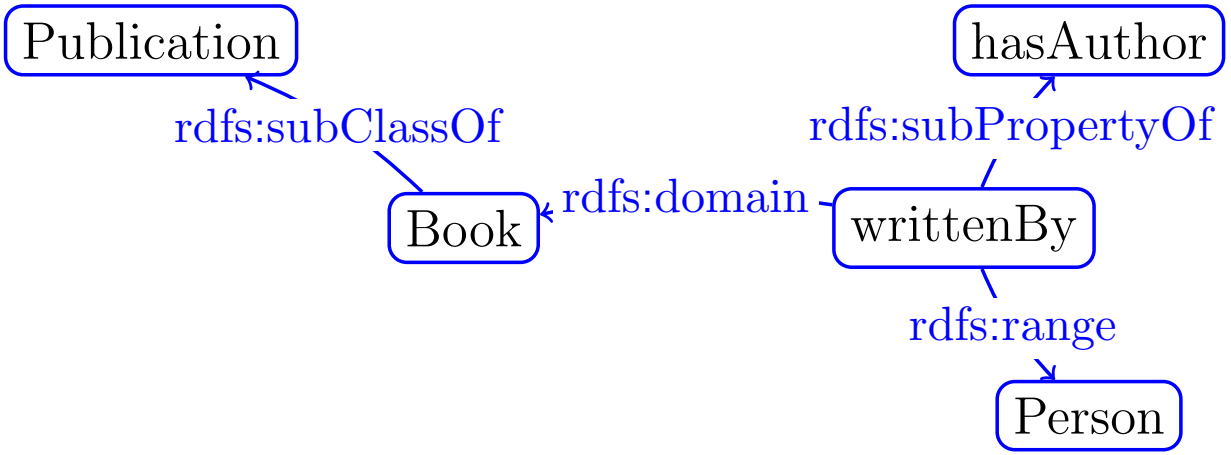
| Assertion | Triple | Relational notation |
|---|---|---|
| Class | s rdf:type o | o(s) |
| Property | s p o | p(s, o) |



# RDF Schema (RDFS)

Declare deductive constraints between classes and properties

| Constraint | Triple | OWA interpretation |
|---|---|---|
| Subclass | s rdfs:subClassOf o | $s \subseteq o$ |
| Subproperty | s rdfs:subPropertyOf o | $s \subseteq o$ |
| Domain typing | s rdfs:domain o | $\Pi_{\text{domain}}(s) \subseteq o$ |
| Range typing | s rdfs:range o | $\Pi_{\text{range}}(s) \subseteq o$ |



# Open-world assumption and RDF entailment

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

$\rightarrow$ deductive constraints – implicitly **propagate** triples

**Implicit triples:** part of the graph – not explicitly present

**Entailment** – reasoning mechanism

# Open-world assumption and RDF entailment

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

$\rightarrow$ deductive constraints – implicitly **propagate** triples

**Implicit triples:** part of the graph – not explicitly present

**Entailment** – reasoning mechanism

| | | |
|---|---|---|
| set of **explicit triples** | | |
| + | $\rightarrow$ | derive **implicit triples** |
| some **entailment rules** | | |

# Open-world assumption and RDF entailment

**RDF data model** – based on the **open-world assumption**.
→ deductive constraints – implicitly **propagate** triples
**Implicit triples:** part of the graph – not explicitly present

**Entailment** – reasoning mechanism

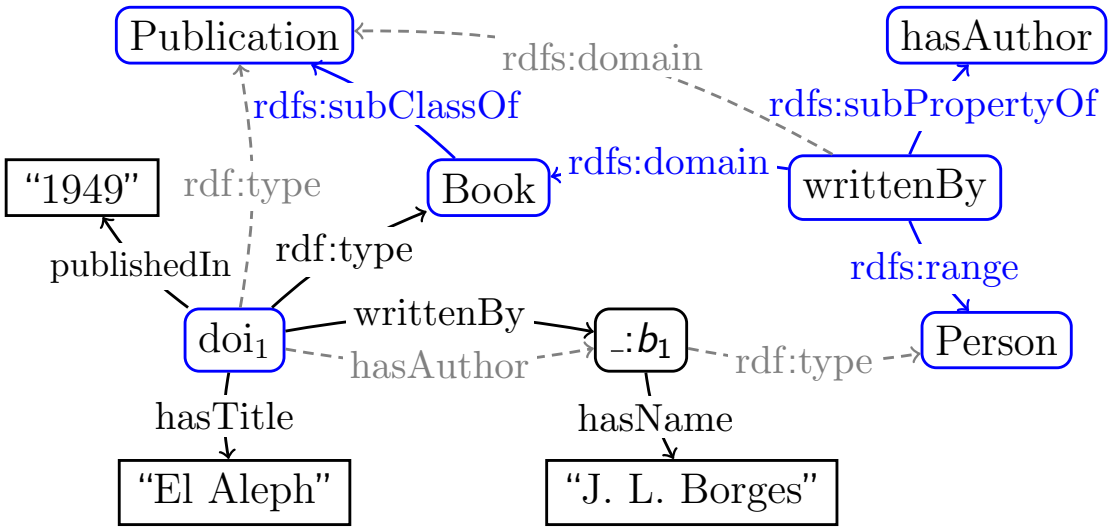
| | | |
|---|---|---|
| set of **explicit triples** + some **entailment rules** | → | derive **implicit triples** |

Exhaustive application of entailment → **saturation** (**closure**)

# RDF entailment

**The semantics of an RDF graph G is its saturation $G^{\infty}$.**

| Sample RDFS entailment rules | Instance entailment from combining schema and instance triples |
|---|---|
| **rdfs9** | $c_1$ rdfs:subClassOf $c_2$ $\wedge$ s rdf:type $c_1 \vdash^{\textbf{rdfs9}}_{\text{RDF}}$ s rdf:type $c_2$ |
| **rdfs7** | $p_1$ rdfs:subPropertyOf $p_2$ $\wedge$ s $p_1$ o $\vdash^{\textbf{rdfs7}}_{\text{RDF}}$ s $p_2$ o |
| **rdfs2** | p rdfs:domain c $\wedge$ s p o $\vdash^{\textbf{rdfs2}}_{\text{RDF}}$ s rdf:type c |
| **rdfs3** | p rdfs:range c $\wedge$ s p o $\vdash^{\textbf{rdfs3}}_{\text{RDF}}$ o rdf:type c |



# SPARQL query language and SPARQL conjunctive queries

SPARQL is the W3C query language for RDF.

SPARQL **conjunctive queries** = **Basic Graph Pattern (BGP)** queries

Sample BGP query:

$$q(a, t) \quad :- \quad (b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a), (b, \text{publishedIn}, \text{"1949"})$$

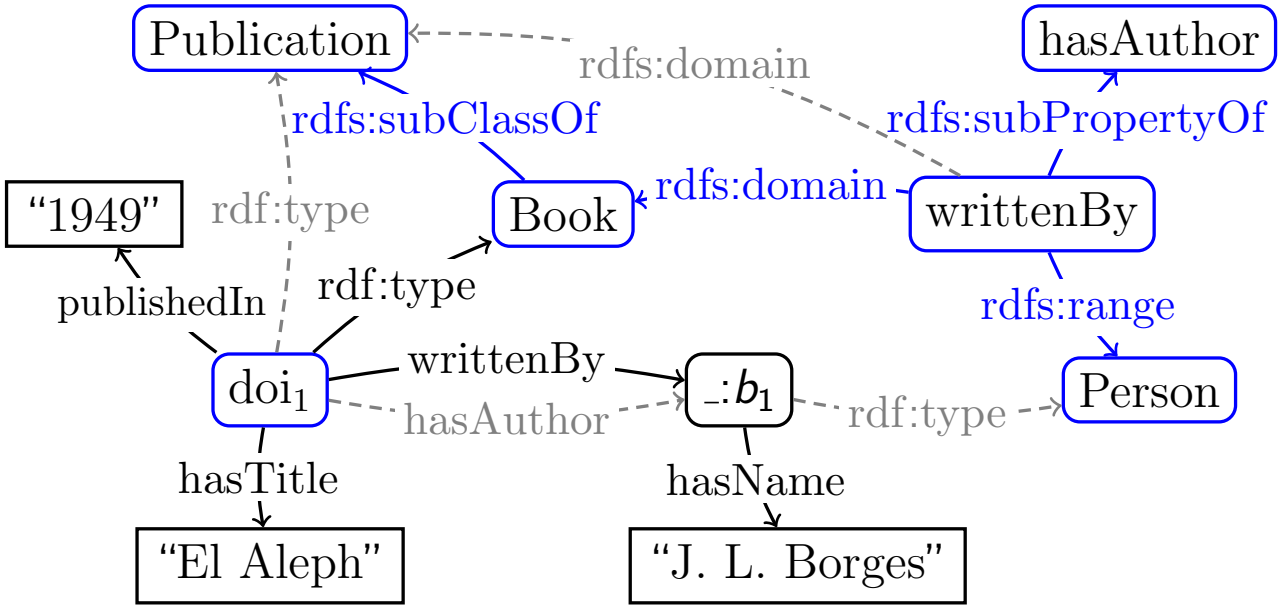
# Query semantics / Answer set

query **evaluation** $\neq$ query **answering**

- The evaluation of a query only uses the graph's **explicit triples**
- For the **(complete) answer set**, evaluate $q$ against the graph's saturation

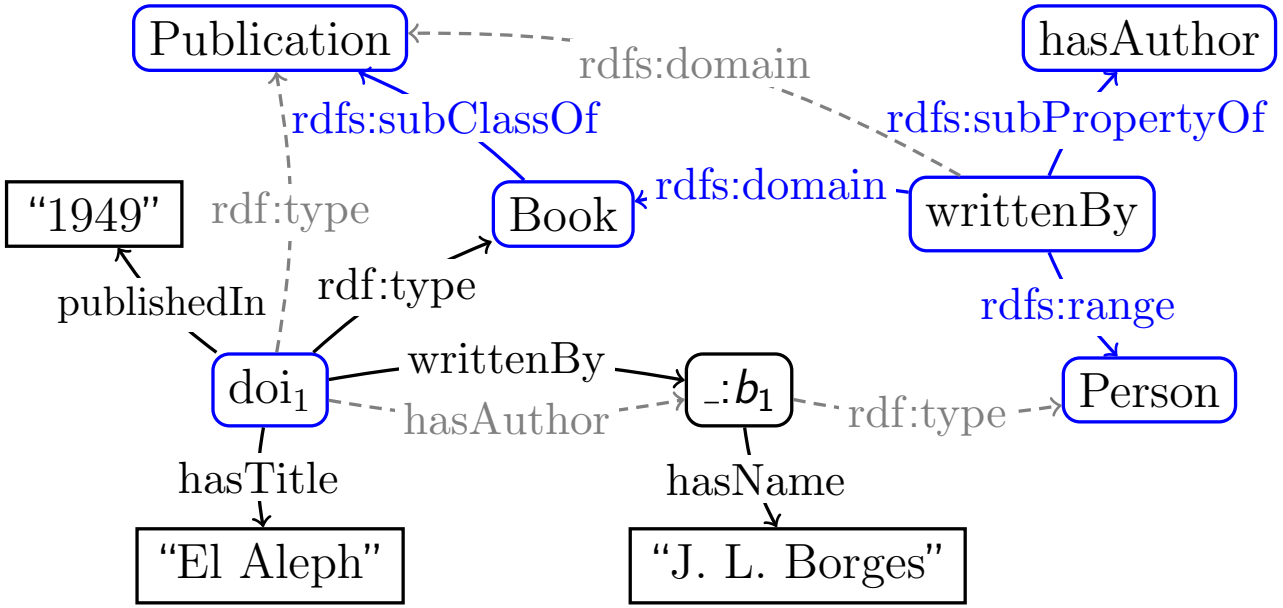
# Query answering example

Given the query **q**(**x**, **y**):- **x** rdf:type **y**:



# Query answering example

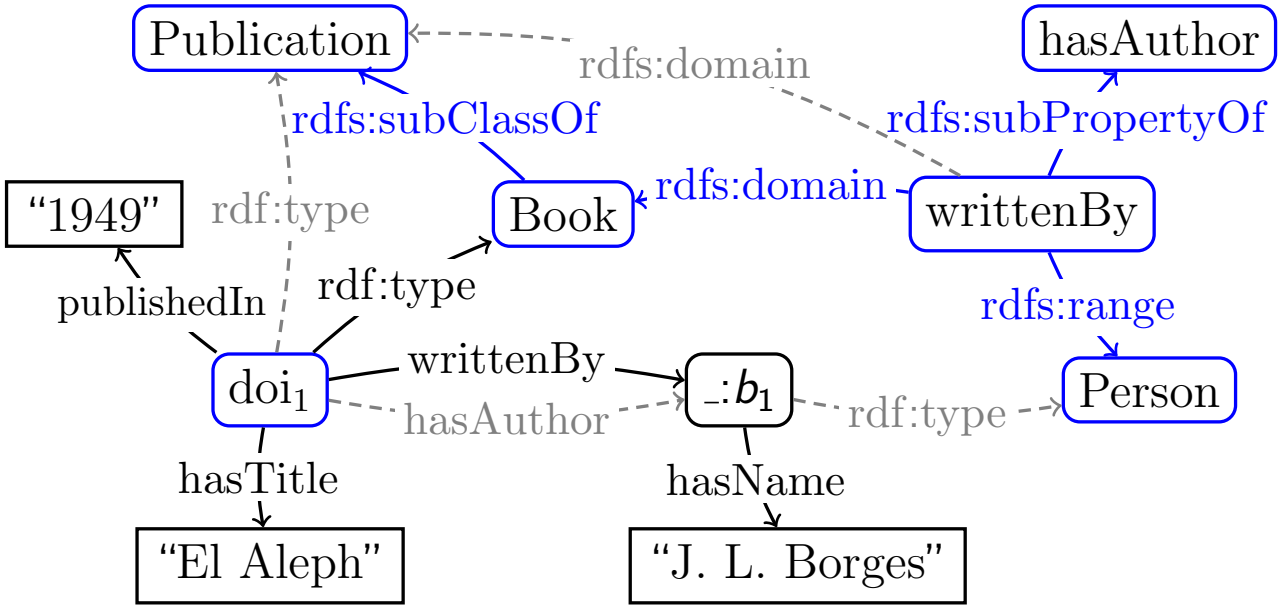
Given the query **q**(**x**, **y**):- **x** rdf:type **y**:




- $q(\mathtt{G}) = \{(\mathrm{doi}_1, \mathrm{Book})\}$

# Query answering example

Given the query **q**(**x**, **y**):- **x** rdf:type **y**:




- $q(\mathtt{G}) = \{(\text{doi}_1, \text{Book})\}$
- $q(\mathtt{G}^\infty) = \{(\text{doi}_1, \text{Book}),\ (\text{doi}_1, \text{Publication}),\ (\_{:}b_1, \text{Person})\}$

# Part III

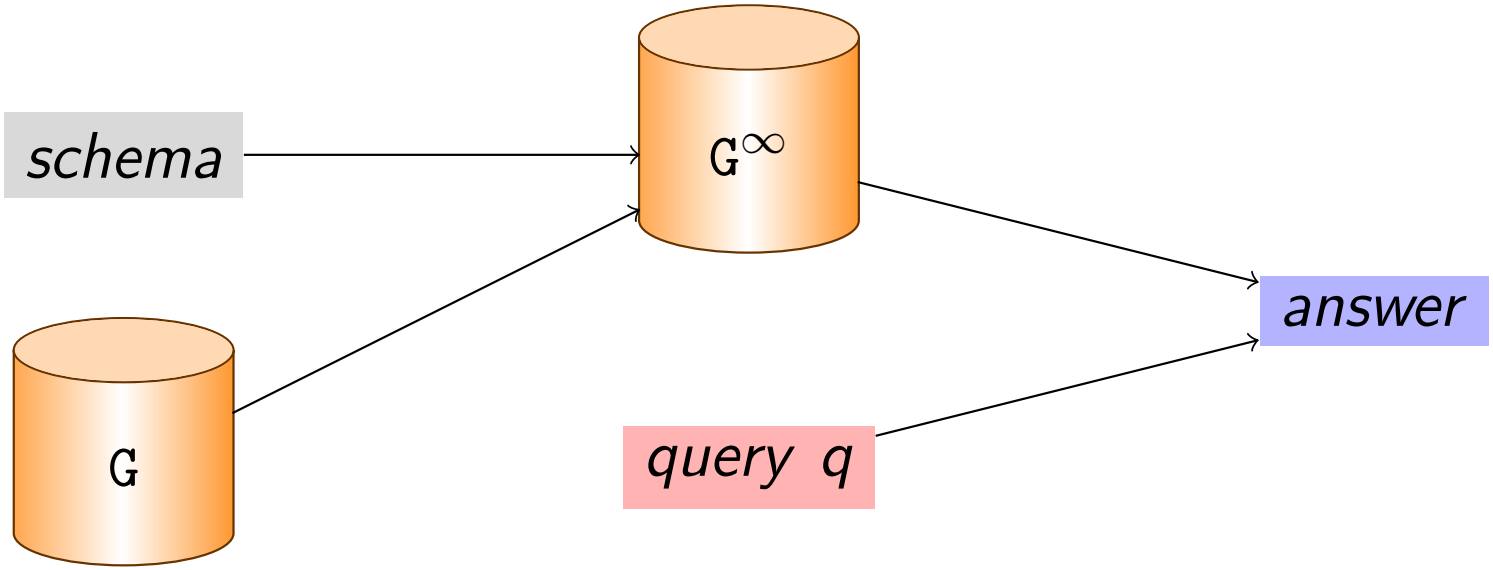
## Query answering techniques

# The need for reasoning

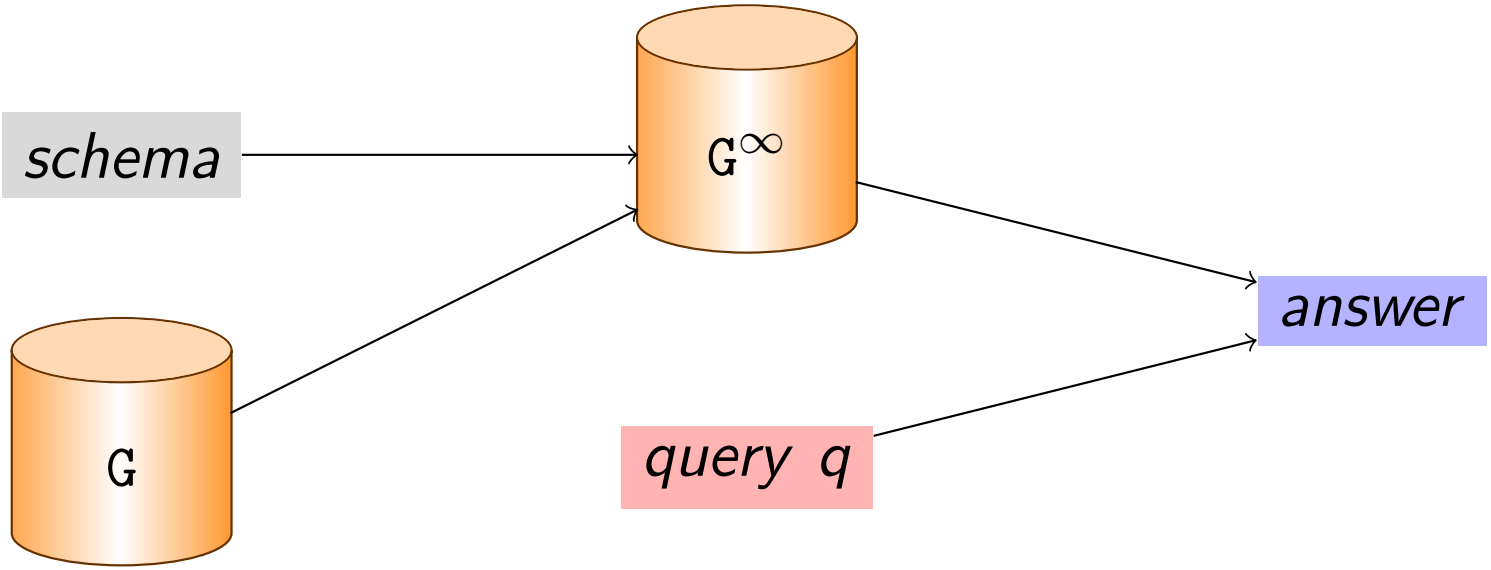
**Query answering needs explicit and implicit data!**

- Saturation-based query answering (when the saturation result is finite)
- Reformulation-based query answering
- Hybrids of the above
  ▷ J. Urbani, F. van Harmelen, S. Schlobach, and H. Bal, "QueryPIE: Backward reasoning for OWL Horst over very large knowledge bases", ISWC 2011

# Saturation-based query answering

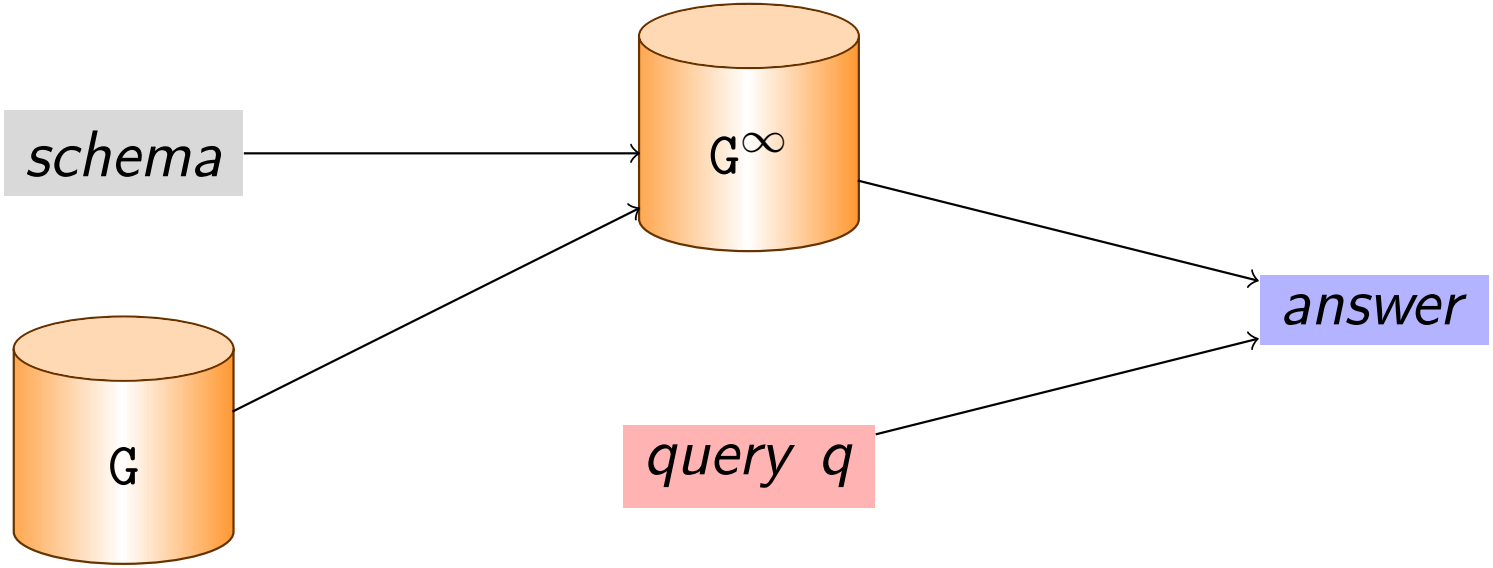


# Saturation-based query answering




- $q(G^\infty)$ 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)

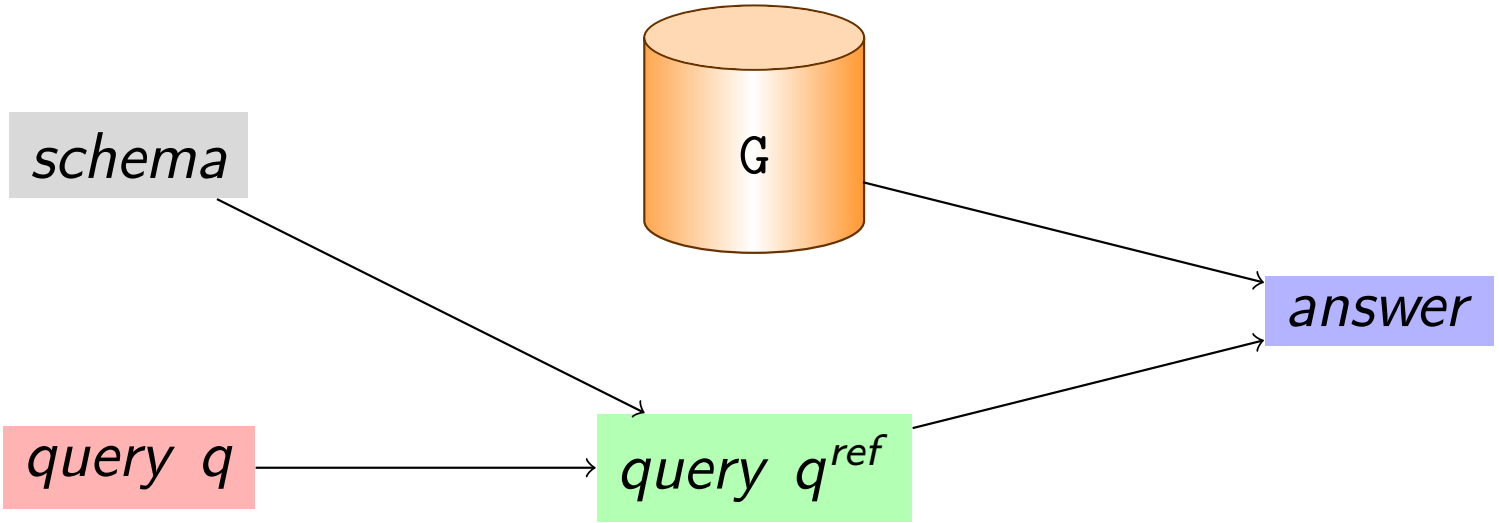
# Saturation maintenance



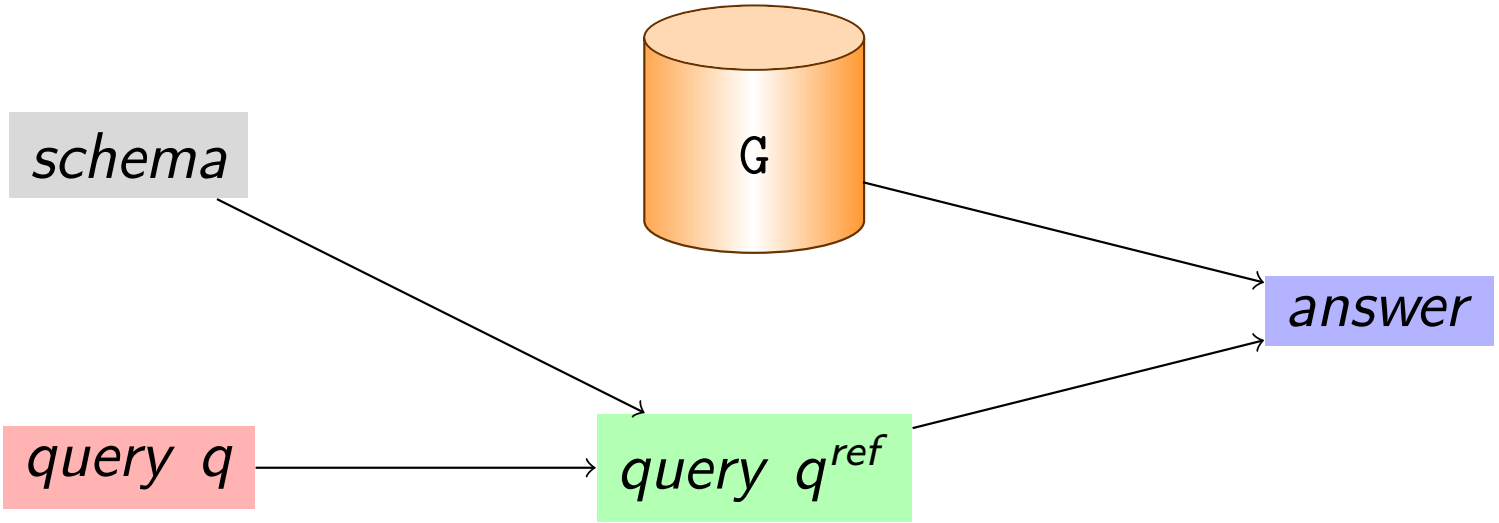

**Compute $\Delta$ for an update of an RDF graph $\mathtt{G}$ s.t.**

- $(\mu(\mathtt{G}))^\infty = \mathtt{G}^\infty \cup \Delta$ when $\mu$ an insertion
- $(\mu(\mathtt{G}))^\infty = \mathtt{G}^\infty \setminus \Delta$ when $\mu$ a deletion

# Reformulation-based query answering

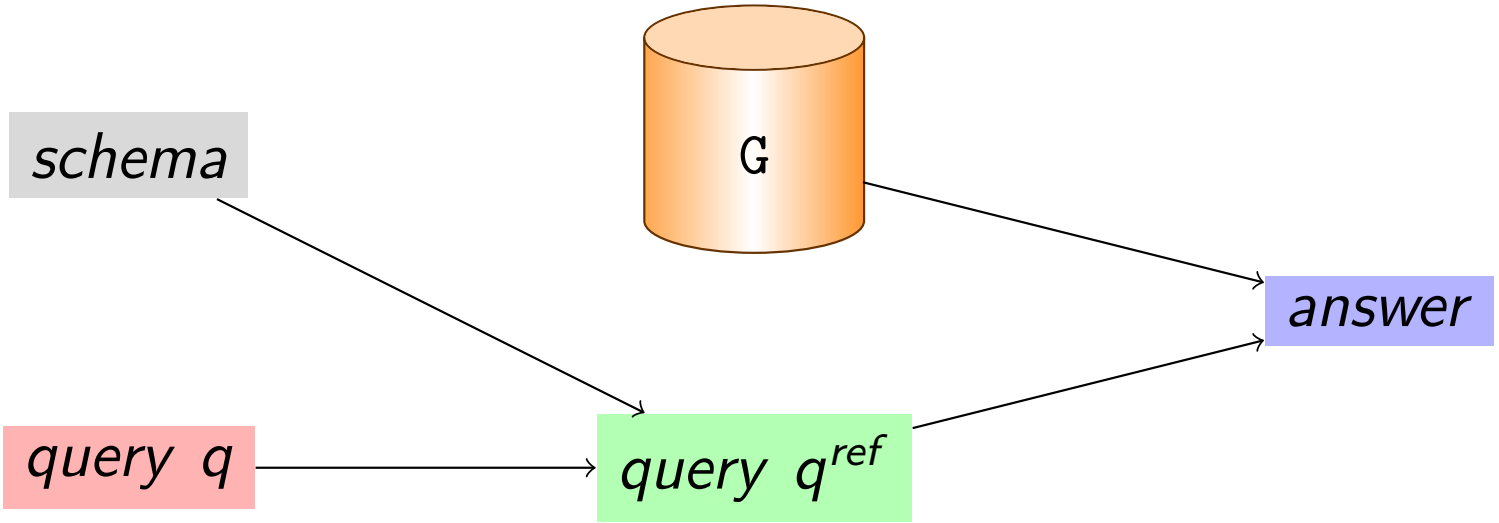


# Reformulation-based query answering

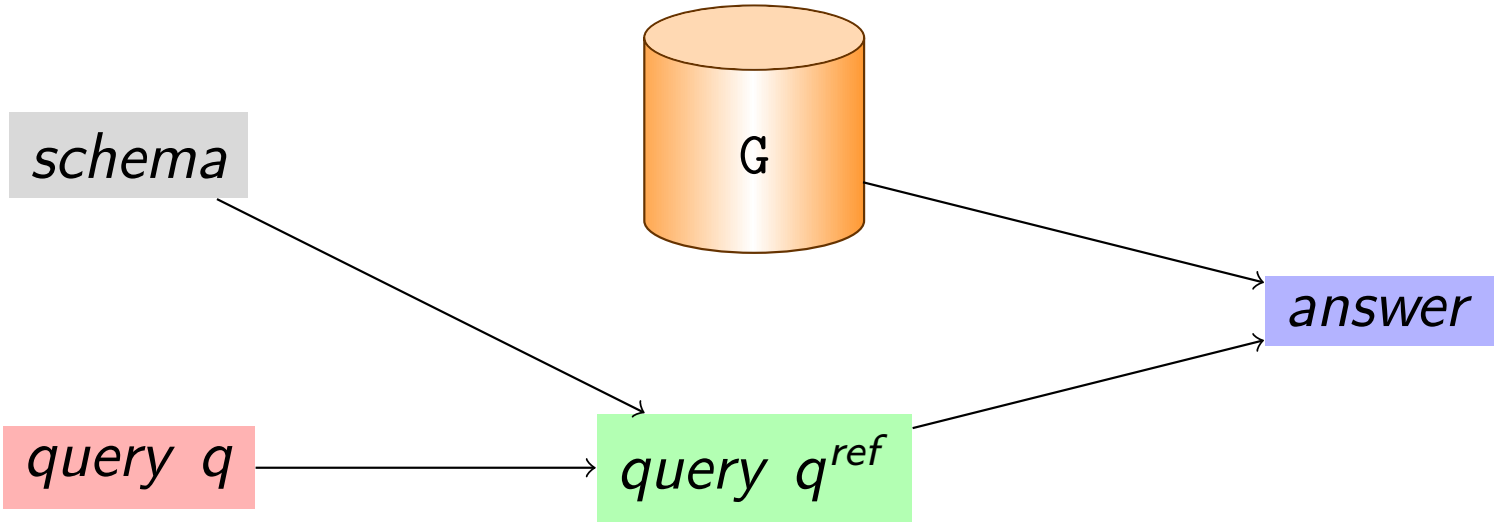



- $q^{ref}(\mathtt{G})$ can be evaluated using an RDBMS
- Robust to updates
- Reformulated queries are complex, thus costly to evaluate

# Reformulation-based query answering



# Reformulation-based query answering




- $q^{ref}(\mathtt{G})$ can be evaluated using an RDBMS
- Robust to updates
- Reformulated queries are complex, thus costly to evaluate

# Semantic query answering in the presence of mappings

▷ Di Pinto, Lembo, Lenzerini, Mancini, Poggi, Rosatti, Ruzzi, Savo, EDBT 2013

- The logical storage of the data is related to the ontology by means of mappings.
- A mapping states that a query on the database is included into a query on the ontology.
- Reformulation-based query answering in the presence of (GAV) mappings:
  ▷ Lanti, Rezk, Xiao and Calvanese, EDBT 2015
  1. Start-up phase (may perform some materialization)
  2. Query rewriting phase (based on the schema)
  3. Query translation phase (e.g., into SQL, based on the mappings)
  4. Query execution phase (e.g., through an RDBMS)

# Semantic query answering in the presence of mappings

- Reformulation-based query answering in the presence of (GAV) mappings:
  ▷ Lanti, Rezk, Xiao and Calvanese, EDBT 2015
  1. Start-up phase (may perform some materialization)
  2. Query rewriting phase (based on the schema)
  3. Query translation phase (e.g., into SQL, based on the mappings)
  4. Query execution phase (e.g., through an RDBMS)

# Semantic query answering in the presence of mappings

- Reformulation-based query answering in the presence of (GAV) mappings:
  ▷ Lanti, Rezk, Xiao and Calvanese, EDBT 2015
  1. Start-up phase (may perform some materialization)
  2. Query rewriting phase (based on the schema)
  3. Query translation phase (e.g., into SQL, based on the mappings)
  4. Query execution phase (e.g., through an RDBMS)

**This talk: Optimized reformulation technique based on considering translation and rewriting together**

Can be composed with mappings (and further mapping-based optimizations)

# Semantic query answering vs. Ontology-Based Data Access (OBDA)

Fundamentally the same

**Input** data (stored somehow), ontology, query

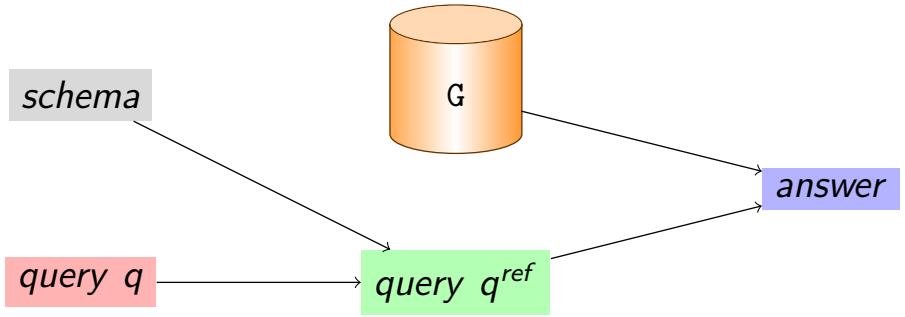
**Output** query answer

In general, mappings may be GAV, LAV, GLAV etc.
LAV can be very practical (see: RDBMS indexes!)

**We optimize query rewriting and translation independently of the global $\leftrightarrow$ local (source) data mappings**

Integration/interactions are intriguing and current/future work

# Reformulation-based query answering landscape




Target reformulation languages for conjunctive queries (CQs):

- mainly **unions of CQs (UCQs)**

  ▷ F. Goasdoué, I. Manolescu, A. Roatiș: "Efficient query answering against dynamic RDF databases", EDBT 2013

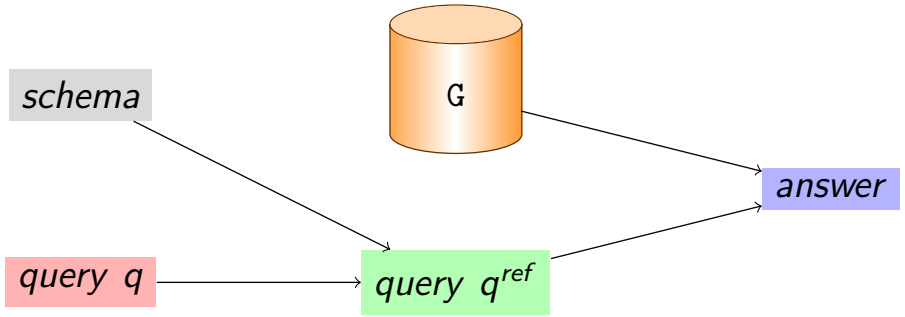
- **joins of single-atom CQs (SCQs)**

  ▷ M. Thomazo: "Compact Rewritings for Existential Rules", IJCAI 2013

- **joins of UCQs (JUCQs)**

  ▷ D. Bursztyn, F. Goasdoué, I. Manolescu: "Optimizing Reformulation-based Query Answering in RDF", EDBT 2015

# Reformulation-based query answering landscape




Target reformulation languages for conjunctive queries (CQs):

- mainly **unions of CQs (UCQs)**

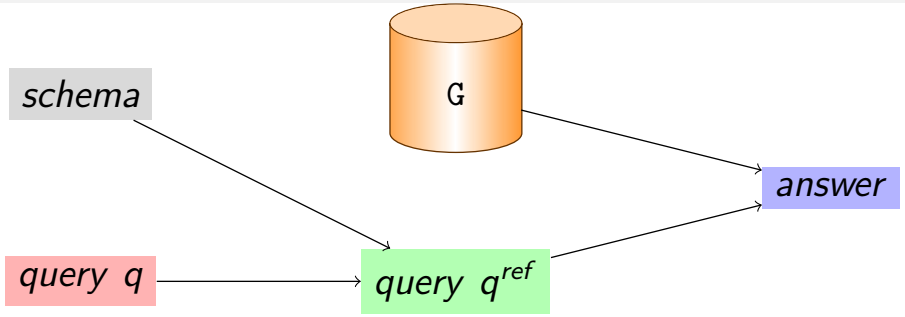
  ▷ F. Goasdoué, I. Manolescu, A. Roatiș: "Efficient query answering against dynamic RDF databases", EDBT 2013

Wait: is this about SQL syntax?!...

- **joins of UCQs (JUCQs)**

  ▷ D. Bursztyn, F. Goasdoué, I. Manolescu: "Optimizing Reformulation-based Query Answering in RDF", EDBT 2015

# Reformulation-based query answering landscape




Target reformulation languages for conjunctive queries (CQs):

Wait: is this about SQL syntax?!...

Yes. And it makes a big difference.

From failing to feasible, or 4 orders of magnitude speed-up on the 8 M triples DBLP dataset.

# Part IV

## Cover-based reformulation for the database fragment of RDF

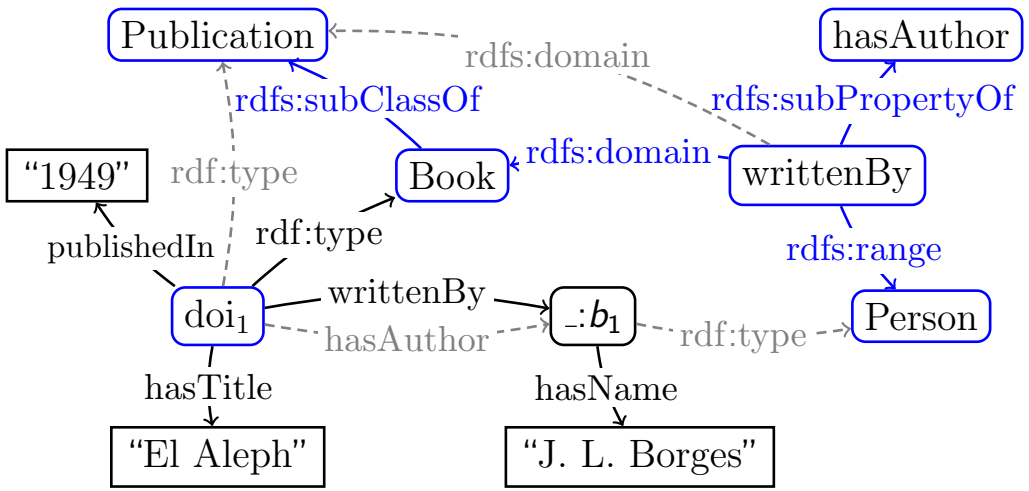
# The database fragment of RDF

Largest RDF fragment for which an FOL reformulation-based query answering technique is known

- Restrict RDF entailment to the rules dedicated to RDF Schema only (a.k.a. RDFS entailment).
- No restriction to graphs: any triple allowed by the RDF specification is also allowed in the DB fragment (e.g., blank nodes in data and schema)

Algorithm `Reformulate`$(q, \mathtt{db})$ for BGP queries over schema and data. Output size bound $O((6 * \#\mathtt{db}^2)^{\#q})$
▷ Goasdoué, Manolescu, Roatiș, EDBT 2013

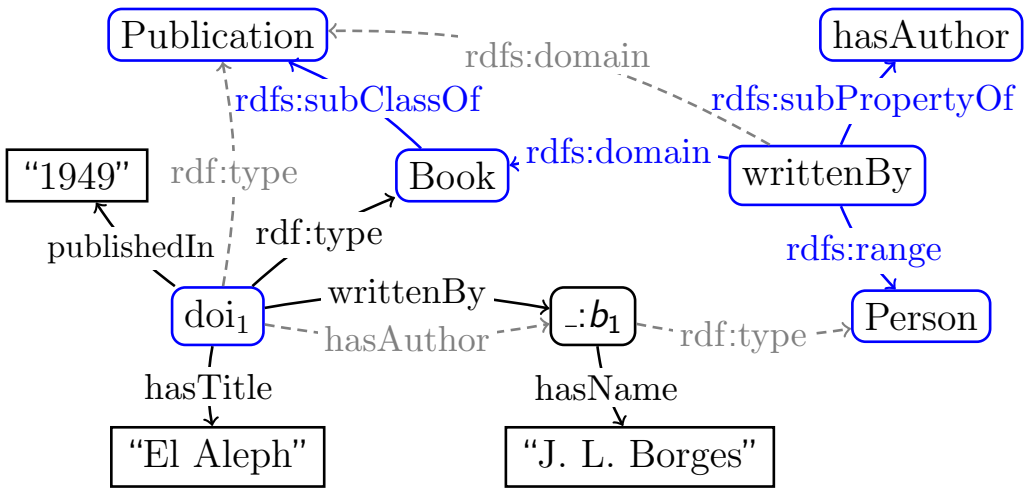
# CQ-to-UCQ query reformulation example




$q(a, t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

(0) $q(a, t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

# CQ-to-UCQ query reformulation example

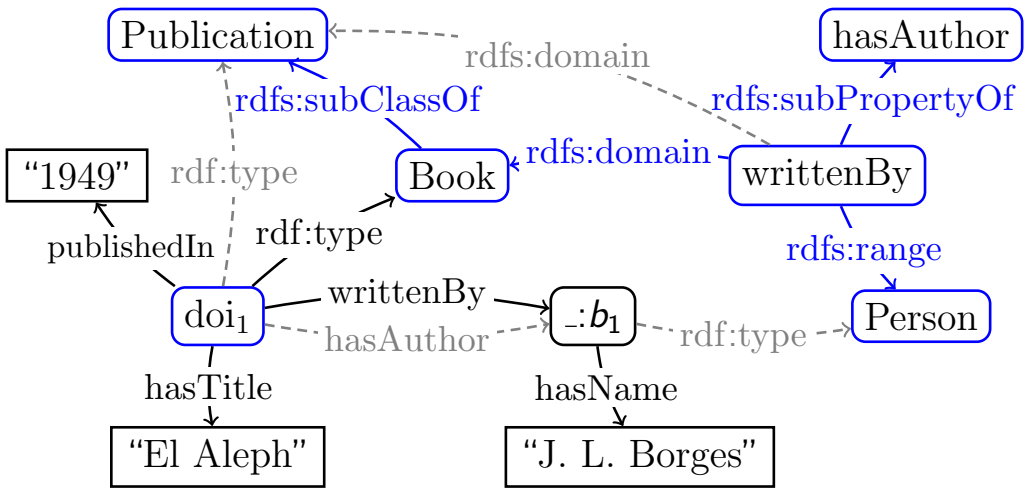



$q(a, t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

(0) $q(a, t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(1) $q(a, t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

# CQ-to-UCQ query reformulation example



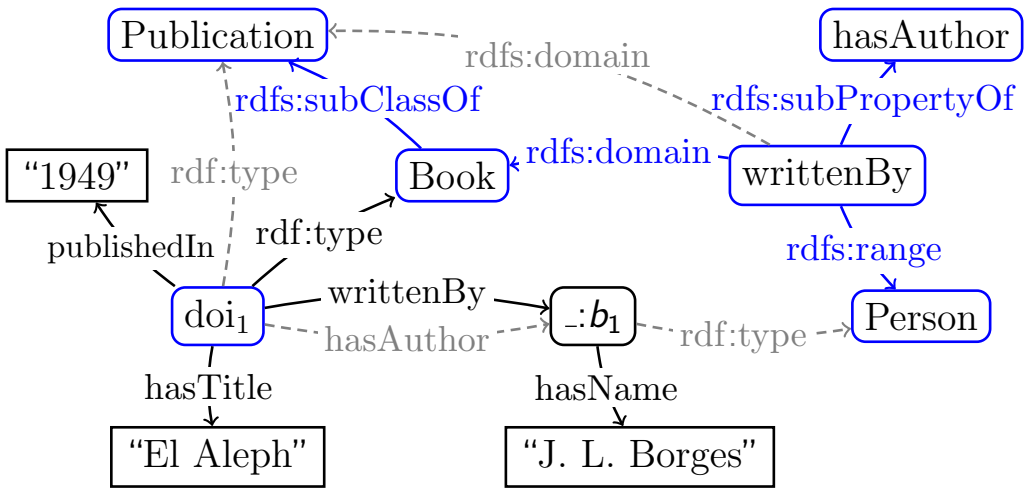

$q(a,t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

(0) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(1) $q(a,t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(2) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

# CQ-to-UCQ query reformulation example




$q(a,t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

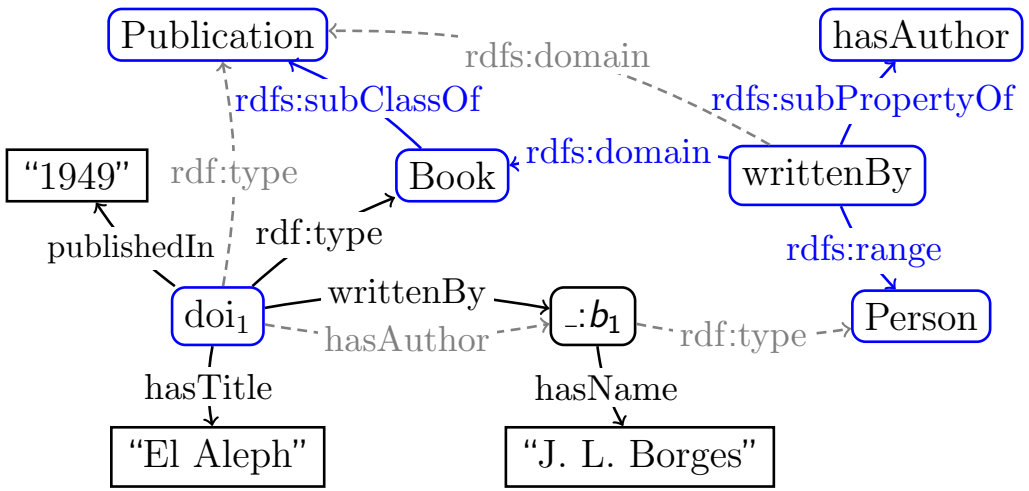
(0) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(1) $q(a,t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(2) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(3) $q(a,t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

# CQ-to-UCQ query reformulation example




$q(a,t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

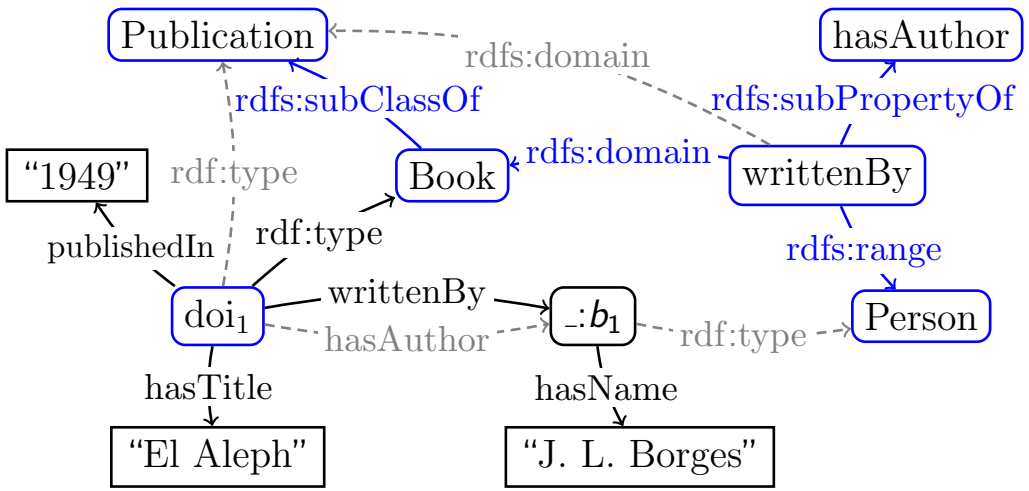
(0) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(1) $q(a,t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(2) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(3) $q(a,t)$ :- $\cancel{(b, \text{writtenBy}, x)}, (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

# CQ-to-UCQ query reformulation example




$q(a,t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ leads to $q^{ref}$:

(0) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

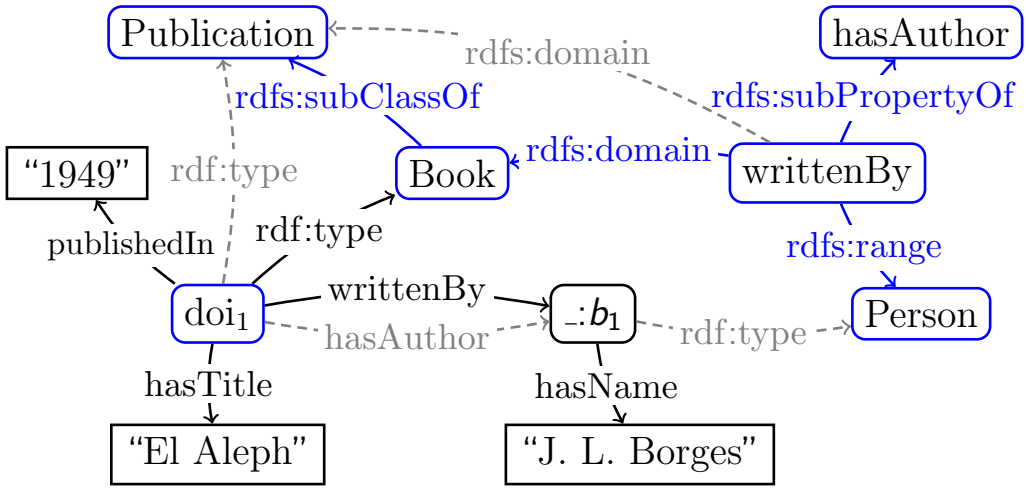
(1) $q(a,t)$ :- $(b, \text{writtenBy}, x), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(2) $q(a,t)$ :- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ $\cup$

(3) $q(a,t)$ :- $\cancel{(b, \text{writtenBy}, x)}, (b, \text{hasTitle}, t), (b, \text{writtenBy}, a),$
$(b, \text{publishedIn}, \text{"1949"})$

$q(\mathtt{G}^{\infty}) = q^{ref}(\mathtt{G}) = \{(\_{:}b_1, \text{"}El\ Aleph\text{"})\}.$

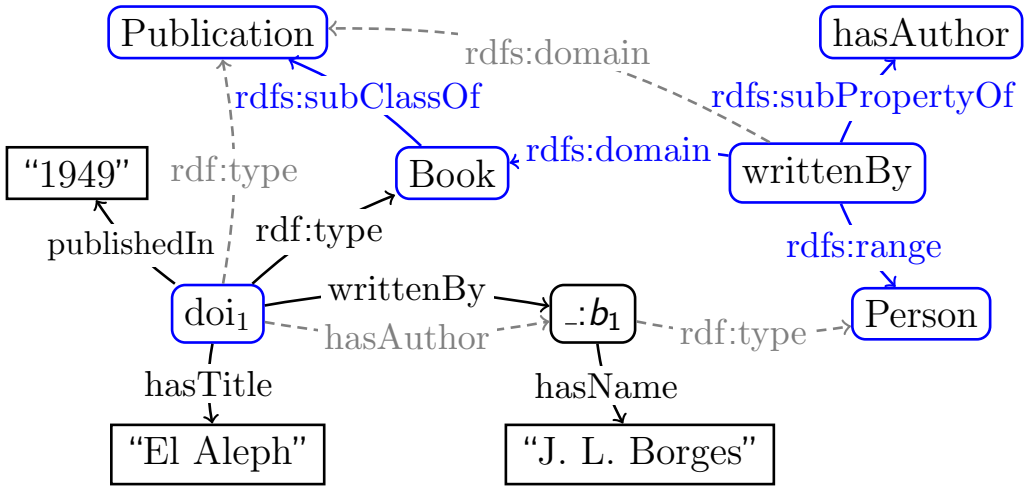
# CQ-to-SCQ query reformulation example




$q(a, t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ produces the $q^{ref}$:

| | | | | |
|---|---|---|---|---|
| (0) | $q(b)$ | :- | $(b, \text{rdf:type}, \text{Book}) \cup (b, \text{writtenBy}, x)$ | $\bowtie$ |
| (1) | $q(b, a)$ | :- | $(b, \text{hasAuthor}, a) \cup (b, \text{writtenBy}, a)$ | $\bowtie$ |
| (2) | $q(b)$ | :- | $(b, \text{publishedIn}, \text{"1949"})$ | |

# CQ-to-SCQ query reformulation example

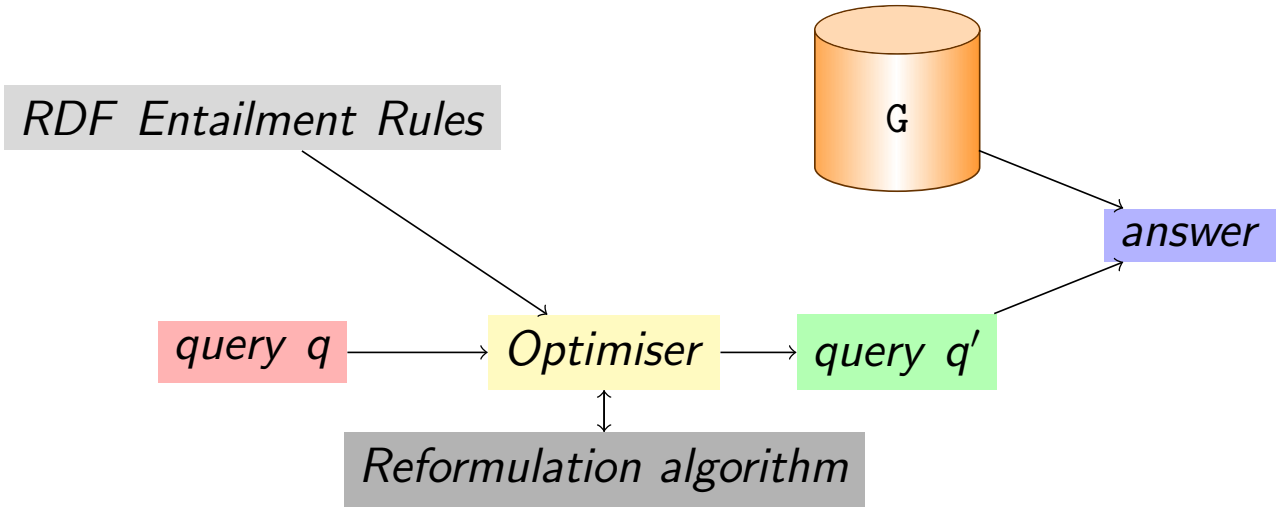



$q(a, t)$:- $(b, \text{rdf:type}, \text{Book}), (b, \text{hasTitle}, t), (b, \text{hasAuthor}, a),$
$(b, \text{publishedIn}, \text{"1949"})$ produces the $q^{ref}$:

| | | | | |
|---|---|---|---|---|
| (0) | $q(b)$ | :- | $(b, \text{rdf:type}, \text{Book}) \cup (b, \text{writtenBy}, x)$ | $\bowtie$ |
| (1) | $q(b, a)$ | :- | $(b, \text{hasAuthor}, a) \cup (b, \text{writtenBy}, a)$ | $\bowtie$ |
| (2) | $q(b)$ | :- | $(b, \text{publishedIn}, \text{"1949"})$ | |

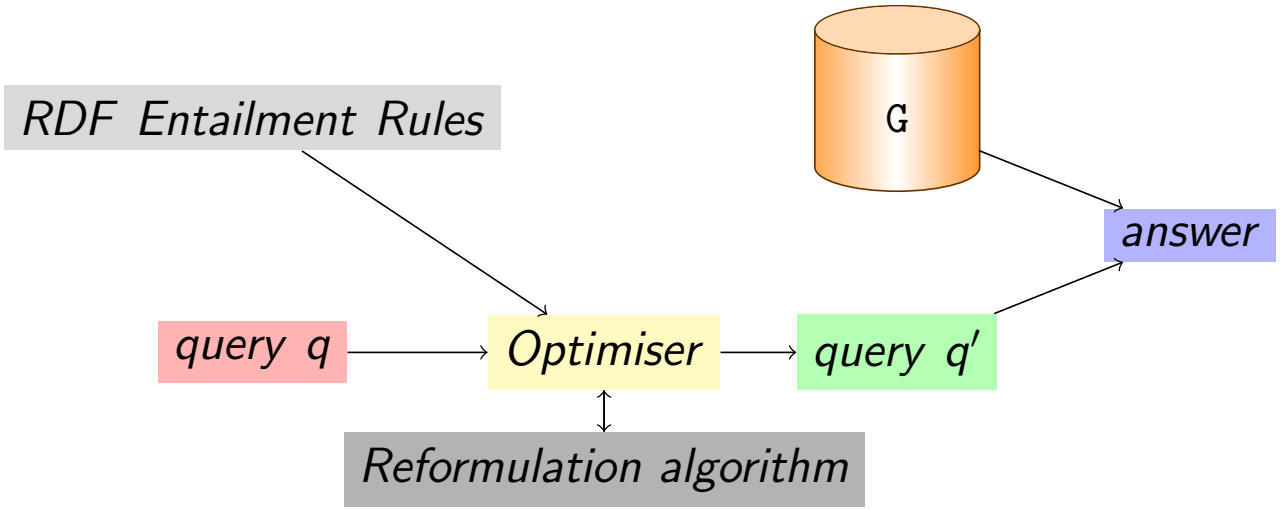
$q(\mathtt{G}^{\infty}) = q^{ref}(\mathtt{G}) = \{(\_{:}b_1, \text{"}El\ Aleph\text{"})\}.$

# CQ-to-JUCQ query reformulation



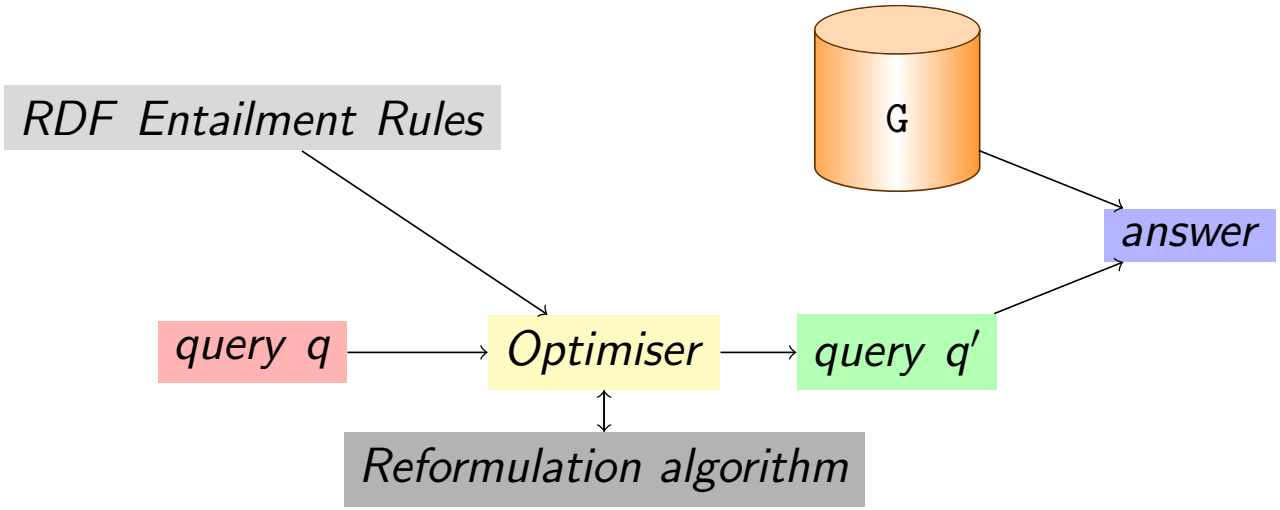

**1. Enlarges the query reformulation language** w.r.t. UCQ/SCQ to have more than one reformulation alternative

# CQ-to-JUCQ query reformulation



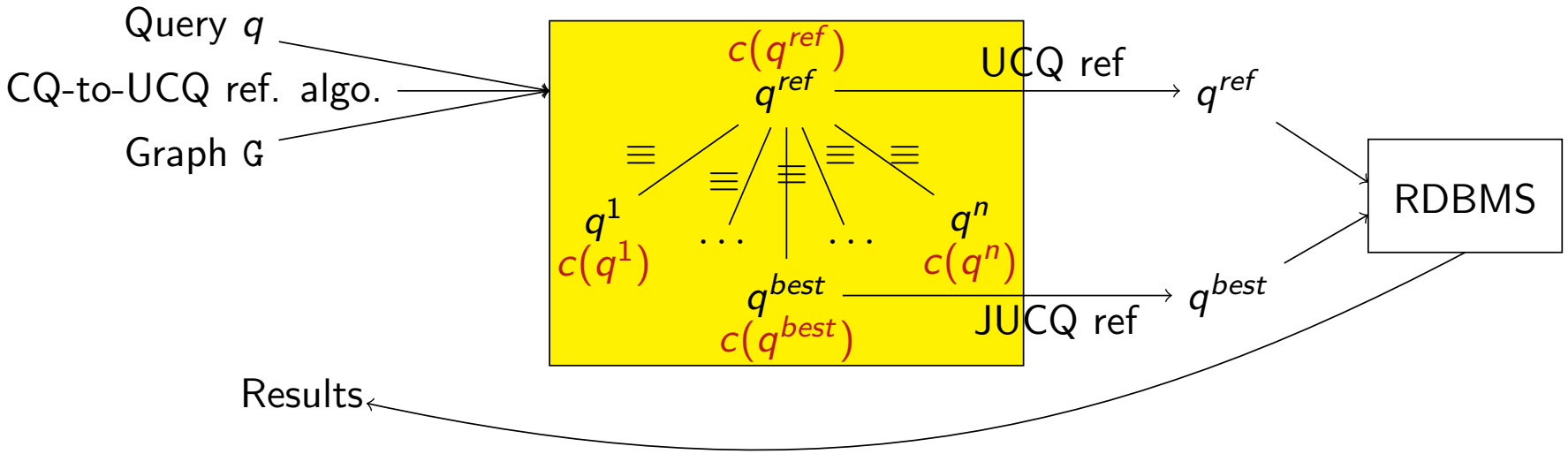

1. **Enlarges the query reformulation language** w.r.t. UCQ/SCQ to have more than one reformulation alternative
2. **Uses a cost model** for estimating the cost of evaluating $q^{\mathrm{ref}}$ through an RDBMS

# CQ-to-JUCQ query reformulation

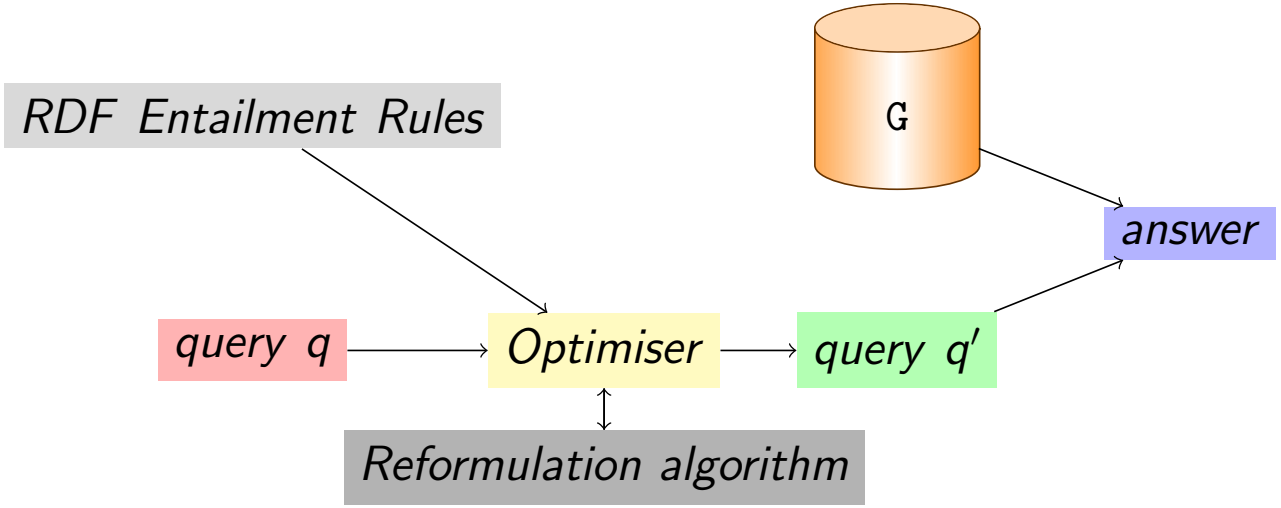



1. **Enlarges the query reformulation language** w.r.t. UCQ/SCQ to have more than one reformulation alternative
2. **Uses a cost model** for estimating the cost of evaluating $q^{\mathrm{ref}}$ through an RDBMS
3. **Chooses the cheapest alternative** from the search space.

# Optimized reformulation into JUCQs



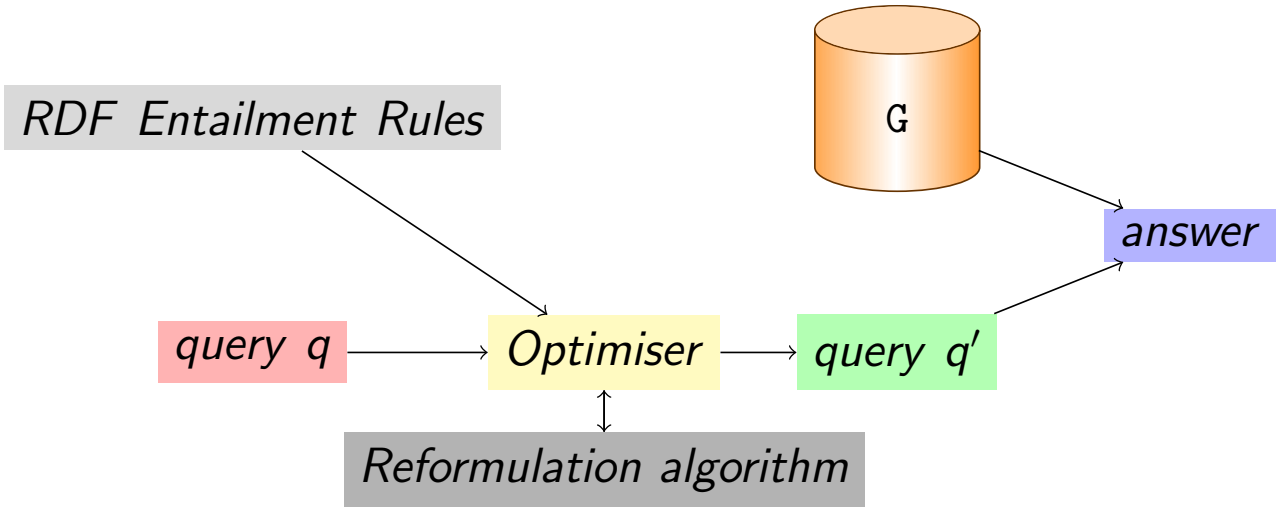
# CQ-to-JUCQ query reformulation




Given the query

$$
\begin{array}{lll}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, & (\mathbf{t_1}) \\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, & (\mathbf{t_2}) \\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} & (\mathbf{t_3})
\end{array}
$$

# CQ-to-JUCQ query reformulation




Given the query

$$
\begin{array}{lll}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, & (\mathbf{t_1}) \\
& x \text{ ub:degreeFrom } \text{“}http://www.University532.edu\text{”}, & (\mathbf{t_2}) \\
& x \text{ ub:memberOf } \text{“}http://www.Department1.University7.edu\text{”} & (\mathbf{t_3})
\end{array}
$$

and a state-of-the-art CQ-to-UCQ reformulation algorithm $.^{ref}$:

- the UCQ reformulation is: $(t_1, t_2, t_3)^{ref}$
- the SCQ reformulation is: $(t_1)^{ref} \bowtie (t_2)^{ref} \bowtie (t_3)^{ref}$

# CQ-to-JUCQ query reformulation

Given the query

$$
\begin{array}{lll}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, & (\mathbf{t_1}) \\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, & (\mathbf{t_2}) \\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} & (\mathbf{t_3})
\end{array}
$$

and a state-of-the-art CQ-to-UCQ reformulation algorithm $.^{ref}$, the space of JUCQs is:

1. $(t_1, t_2, t_3)^{ref}$
2. $(t_1)^{ref} \bowtie (t_2)^{ref} \bowtie (t_3)^{ref}$
3. $(t_1, t_2)^{ref} \bowtie (t_3)^{ref}$
4. $(t_1)^{ref} \bowtie (t_2, t_3)^{ref}$
5. $(t_1, t_3)^{ref} \bowtie (t_2)^{ref}$
6. $(t_1, t_2)^{ref} \bowtie (t_1, t_3)^{ref}$
7. $(t_1, t_2)^{ref} \bowtie (t_2, t_3)^{ref}$
8. $(t_1, t_3)^{ref} \bowtie (t_2, t_3)^{ref}$

# CQ-to-JUCQ query reformulation algorithms

## Exhaustive algorithm

Impractical since search space size is the number of minimal covers of a set of $n$ elements

## Greedy algorithm

Driven by cost; starting from 1-atom fragments and "growing" them

Given the query

$$
\begin{array}{lll}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, & (\mathbf{t_1}) \\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, & (\mathbf{t_2}) \\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} & (\mathbf{t_3})
\end{array}
$$

a cost-based greedy exploration is:

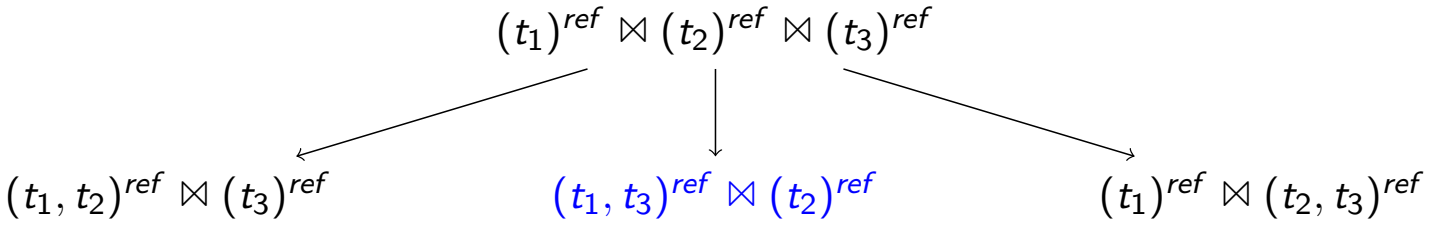
$$(t_1)^{ref} \bowtie (t_2)^{ref} \bowtie (t_3)^{ref}$$

# CQ-to-JUCQ query reformulation: greedy algorithm example

Given the query

$$
\begin{array}{lll}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, & (\mathbf{t_1}) \\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, & (\mathbf{t_2}) \\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} & (\mathbf{t_3})
\end{array}
$$

a cost-based greedy exploration is:

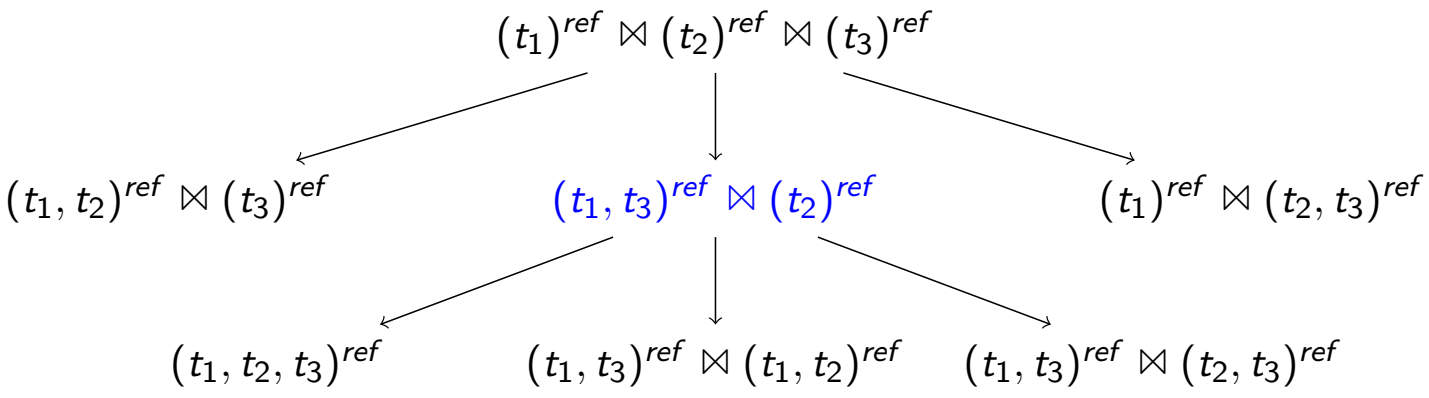


# CQ-to-JUCQ query reformulation algorithm

Given the query

$$
\begin{aligned}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, && (\mathbf{t_1})\\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, && (\mathbf{t_2})\\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} && (\mathbf{t_3})
\end{aligned}
$$

a cost-based greedy exploration is:



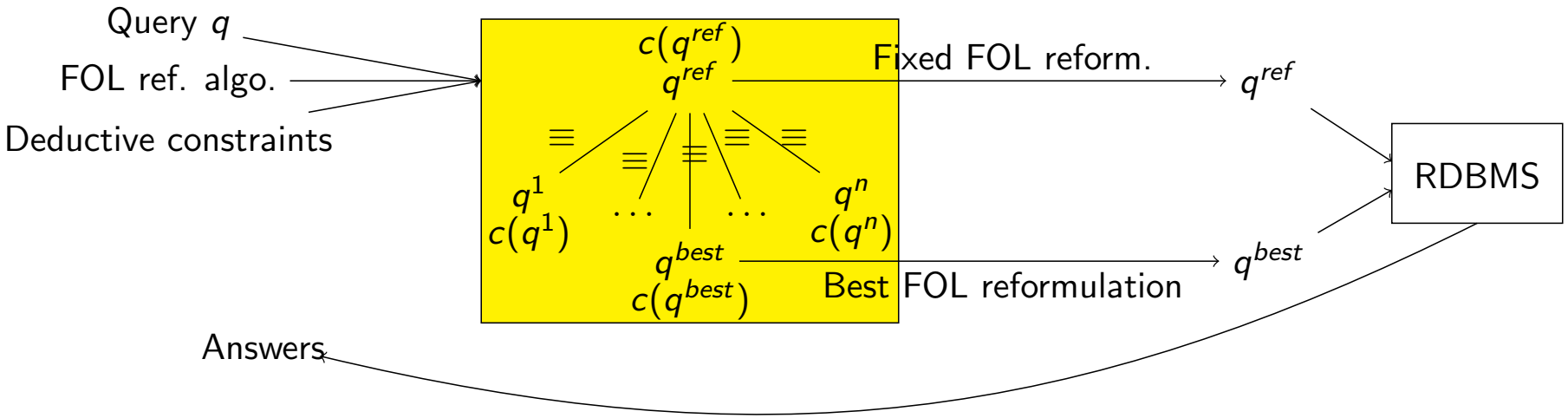
# Part V

## Cover-based reformulation in FOL-reducible settings

# Optimised reformulation-based query answering

For any setting where query answering is reducible to FOL query evaluation

The same overall approach:



# DL-Lite$_{\mathcal{R}}$ facts (a.k.a. ABox)

Given:

- $N_C$**:** set of **concept names** (unary predicates),
- $N_R$**:** set of **role names** (binary predicates),
- $N_I$**:** set of **individuals** (constants).

# DL-Lite$_{\mathcal{R}}$ facts (a.k.a. ABox)

Given:

- $N_C$**:** set of **concept names** (unary predicates),
- $N_R$**:** set of **role names** (binary predicates),
- $N_I$**:** set of **individuals** (constants).

Finite number of:

- **Concept assertions:** $A(a)$ with $A \in N_C$ and $a \in N_I$
- **Role assertions:** $R(a, b)$ with $R \in N_R$ and $a, b \in N_I$

# DL-Lite$_{\mathcal{R}}$ constraints (a.k.a. TBox)

Given:

- $N_C$**:** set of **concept names** (unary predicates),
- $N_R$**:** set of **role names** (binary predicates),

# DL-Lite$_{\mathcal{R}}$ constraints (a.k.a. TBox)

Given:

- $N_C$**:** set of **concept names** (unary predicates),
- $N_R$**:** set of **role names** (binary predicates),

Set of:

- **Concept inclusions:** $B \sqsubseteq C$
- **Role inclusions:** $Q \sqsubseteq S$

using the following grammar (where $A \in N_C$ and $R \in N_R$):

$$B := A \mid \exists Q,\ C := B \mid \neg B,\ Q := R \mid R^-,\ S := Q \mid \neg Q$$

# FOL Reformulation

Consider the query $q(x)$:- $\texttt{PhDStudent}(x) \wedge \texttt{worksWith}(y, x)$ and KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the TBox $\mathcal{T}$ and ABox $\mathcal{A}$ below:

$$
\begin{array}{lrcl}
(T1) & \texttt{PhDStudent} & \sqsubseteq & \texttt{Researcher} \\
(T2) & \exists\texttt{worksWith} & \sqsubseteq & \texttt{Researcher} \\
(T3) & \exists\texttt{worksWith}^{-} & \sqsubseteq & \texttt{Researcher} \\
(T4) & \texttt{worksWith} & \sqsubseteq & \texttt{worksWith}^{-} \\
(T5) & \texttt{supervisedBy} & \sqsubseteq & \texttt{worksWith} \\
(T6) & \exists\texttt{supervisedBy} & \sqsubseteq & \texttt{PhDStudent} \\
(T7) & \texttt{PhDStudent} & \sqsubseteq & \neg\exists\texttt{supervisedBy}^{-}
\end{array}
$$

$$
\begin{array}{ll}
(A1) & \texttt{worksWith}(\mathit{Ioana}, \mathit{Francois}) \\
(A2) & \texttt{supervisedBy}(\mathit{Damian}, \mathit{Ioana}) \\
(A3) & \texttt{supervisedBy}(\mathit{Damian}, \mathit{Francois})
\end{array}
$$

# FOL Reformulation

The UCQ reformulation of $q$ is:

$$
\begin{aligned}
q^{\mathtt{UCQ}}(x) \text{:-}\ & q^1(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{worksWith}(y, x) \\
& \vee q^2(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{worksWith}(x, y) \\
& \vee q^3(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{supervisedBy}(y, x) \\
& \vee q^4(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{supervisedBy}(x, y) \\
& \vee q^5(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{worksWith}(y, x) \\
& \vee q^6(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{worksWith}(x, y) \\
& \vee q^7(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{supervisedBy}(y, x) \\
& \vee q^8(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{supervisedBy}(x, y) \\
& \vee q^9(x) \text{:-}\ \mathtt{supervisedBy}(x, x) \\
& \vee q^{10}(x) \text{:-}\ \mathtt{supervisedBy}(x, y)
\end{aligned}
$$

# FOL Reformulation

The UCQ reformulation of $q$ is:

$$
\begin{aligned}
q^{\mathtt{UCQ}}(x) \text{:-}\ & q^1(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{worksWith}(y, x) \\
& \vee q^2(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{worksWith}(x, y) \\
& \vee q^3(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{supervisedBy}(y, x) \\
& \vee q^4(x) \text{:-}\ \mathtt{PhDStudent}(x) \wedge \mathtt{supervisedBy}(x, y) \\
& \vee q^5(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{worksWith}(y, x) \\
& \vee q^6(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{worksWith}(x, y) \\
& \vee q^7(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{supervisedBy}(y, x) \\
& \vee q^8(x) \text{:-}\ \mathtt{supervisedBy}(x, z) \wedge \mathtt{supervisedBy}(x, y) \\
& \vee q^9(x) \text{:-}\ \mathtt{supervisedBy}(x, x) \\
& \vee q^{10}(x) \text{:-}\ \mathtt{supervisedBy}(x, y)
\end{aligned}
$$

Naive exhaustive application of specialization steps leads, in general, to highly redundant reformulations.

# FOL Reformulation minimization

Minimization of $q^{\texttt{UCQ}}$ by eliminating disjuncts contained in another leads to:

$$
\begin{aligned}
q^{\texttt{UCQ}}_{min}(x) \text{:-}\ & q^1(x) \text{:-}\ \texttt{PhDStudent}(x) \wedge \texttt{worksWith}(y, x) \\
& \vee q^2(x) \text{:-}\ \texttt{PhDStudent}(x) \wedge \texttt{worksWith}(x, y) \\
& \vee q^3(x) \text{:-}\ \texttt{PhDStudent}(x) \wedge \texttt{supervisedBy}(y, x) \\
& \vee q^9(x) \text{:-}\ \texttt{supervisedBy}(x, y)
\end{aligned}
$$

# Cover-based FOL reformulation

▷ Bursztyn, Goasdoué, Manolescu (DL 2015)
Consider the query $q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$ and KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the TBox $\mathcal{T}$ and ABox $\mathcal{A}$ below:

$$\begin{array}{ll} (T1) & B \sqsubseteq \exists R' \\ (T2) & R' \sqsubseteq R \end{array}$$

$$\begin{array}{ll} (A1) & A(a) \\ (A2) & B(a) \end{array}$$

# Cover-based FOL reformulation

▷ Bursztyn, Goasdoué, Manolescu (DL 2015)
Consider the query $q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$ and KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ with the TBox $\mathcal{T}$ and ABox $\mathcal{A}$ below:

$$\begin{array}{ll} (T1) & B \sqsubseteq \exists R' \\ (T2) & R' \sqsubseteq R \end{array}$$

$$\begin{array}{ll} (A1) & A(a) \\ (A2) & B(a) \end{array}$$

The only answer to $q(x)$ is $a$.

# Cover-based FOL reformulation

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_1 = \{\{A(x), R(x, y)\}, \quad \{R'(z, y)\}\}$.

# Cover-based FOL reformulation

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_1 = \{\{A(x), R(x, y)\}, \quad \{R'(z, y)\}\}$.

The cover-based FOL reformulation of $q$ w.r.t. $C_1$ is:

$$
\begin{array}{lll}
q^{\mathtt{JUCQ}}(\mathbf{x}) \text{:-} & q_1^{\mathtt{UCQ}}(\mathbf{x}, \mathbf{y}) \text{:-} & (A(\mathbf{x}) \wedge R(\mathbf{x}, \mathbf{y})) \\
 & & \vee (A(\mathbf{x}) \wedge R'(\mathbf{x}, \mathbf{y})) \wedge \\
 & q_2^{\mathtt{UCQ}}(\mathbf{y}) \text{:-} & R'(z, \mathbf{y})
\end{array}
$$

# Cover-based FOL reformulation

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_1 = \{\{A(x), R(x, y)\}, \quad \{R'(z, y)\}\}$.

The cover-based FOL reformulation of $q$ w.r.t. $C_1$ is:

$$
\begin{array}{lll}
q^{\mathtt{JUCQ}}(\mathbf{x}) \text{:-} & q_1^{\mathtt{UCQ}}(\mathbf{x}, \mathbf{y}) \text{:-} & (A(\mathbf{x}) \wedge R(\mathbf{x}, \mathbf{y})) \\
 & & \vee (A(\mathbf{x}) \wedge R'(\mathbf{x}, \mathbf{y})) \wedge \\
 & q_2^{\mathtt{UCQ}}(\mathbf{y}) \text{:-} & R'(z, \mathbf{y})
\end{array}
$$

**Cover $C_1$ prevents unification** $\Rightarrow$ $q^{\mathtt{JUCQ}}(x)$ is <u>not</u> a FOL reformulation of $q$ w.r.t. $\mathcal{T}$.

# Safe cover

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\}\rangle$

Consider the cover $C_2 = \{\{A(x)\}, \quad \{R(x, y), R'(z, y)\}\}$

# Safe cover

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_2 = \{\{A(x)\}, \quad \{R(x, y), R'(z, y)\}\}$

The cover-based FOL reformulation of $q$ w.r.t. $C_1$ is:

$$q^{\mathtt{JUCQ}}(\mathbf{x}) \text{:-} \quad q_1^{\mathtt{UCQ}}(\mathbf{x}) \text{:-} \quad (A(\mathbf{x})$$

$$q_2^{\mathtt{UCQ}}(\mathbf{x}) \text{:-} \quad (R(\mathbf{x}, y) \wedge R'(z, y)) \vee (R'(\mathbf{x}, y) \wedge R'(z, y)) \vee (R'(\mathbf{x}, y)) \vee (B(\mathbf{x}))$$

# Safe cover

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_2 = \{\{A(x)\}, \quad \{R(x, y), R'(z, y)\}\}$

The cover-based FOL reformulation of $q$ w.r.t. $C_1$ is:

$$q^{\mathtt{JUCQ}}(\mathbf{x}) \text{:-} \quad q_1^{\mathtt{UCQ}}(\mathbf{x}) \text{:-} \quad (A(\mathbf{x})$$

$$q_2^{\mathtt{UCQ}}(\mathbf{x}) \text{:-} \quad (R(\mathbf{x}, y) \wedge R'(z, y)) \vee (R'(\mathbf{x}, y) \wedge R'(z, y)) \vee (R'(\mathbf{x}, y)) \vee (B(\mathbf{x}))$$

**Cover $C_2$ preserves unification** $\Rightarrow q^{\mathtt{JUCQ}}(x)$ is a FOL reformulation of $q$ w.r.t. $\mathcal{T}$ and $C_2$ a **safe cover**.

# Root cover

**We call root cover the minimal safe cover.**

# Root cover

**We call root cover the minimal safe cover.**

$C_2 = \{\{A(x)\}, \quad \{R(x, y), R'(z, y)\}\}$ is a **root cover** for $q$ w.r.t. $\mathcal{T}$.

# Extended cover and extended fragment queries

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\}\rangle$

Consider the cover $C_3 = \{f_1 \| f_1, \quad f_2 \| f_0\}$, with:

- $f_0 = \{A(x)\}$
- $f_1 = \{R(x, y), R'(z, y)\}$
- $f_2 = \{A(x), R(x, y)\}$

# Extended cover and extended fragment queries

$q(x)$:- $A(x) \wedge R(x, y) \wedge R'(z, y)$
$\mathcal{K} = \langle \mathcal{T} = \{B \sqsubseteq \exists R', R' \sqsubseteq R\}, \mathcal{A} = \{A(a), B(a)\} \rangle$

Consider the cover $C_3 = \{f_1 \| f_1, \quad f_2 \| f_0\}$, with:

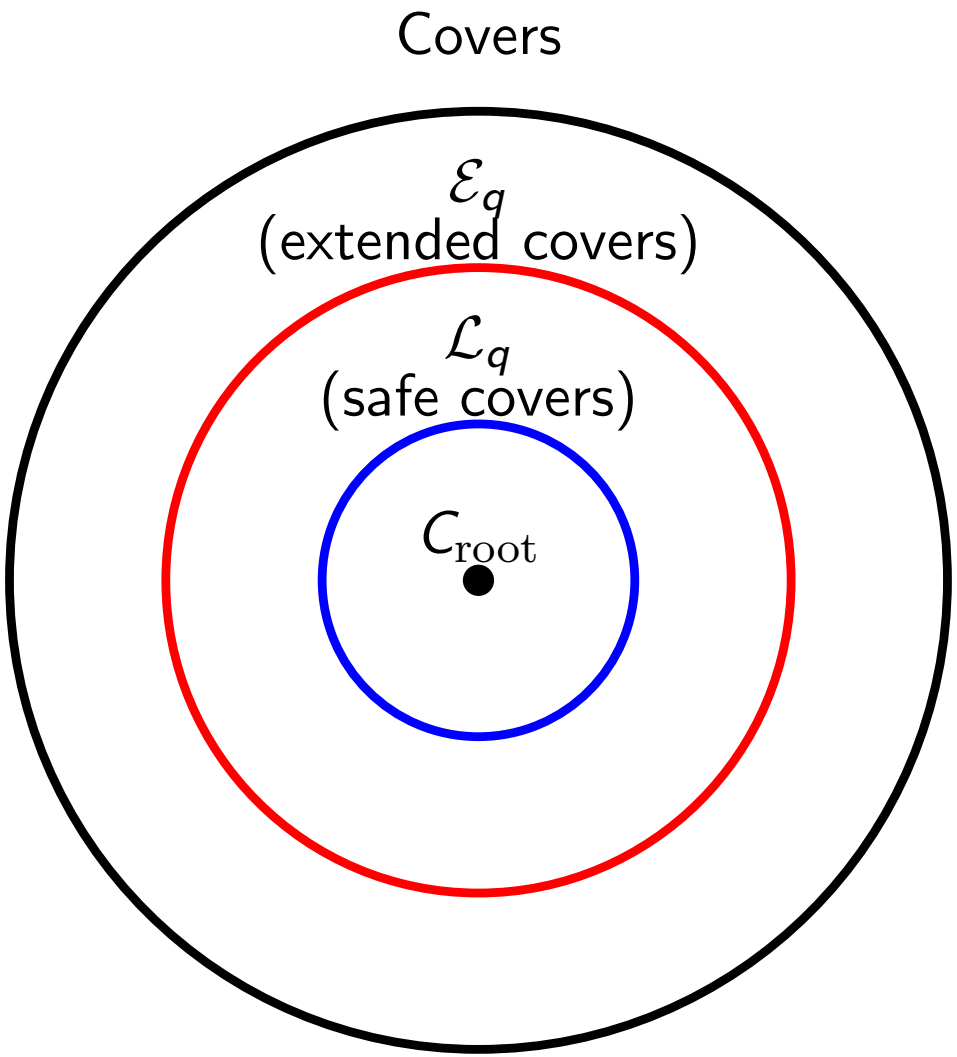
- $f_0 = \{A(x)\}$
- $f_1 = \{R(x, y), R'(z, y)\}$
- $f_2 = \{A(x), R(x, y)\}$

The extended cover-based FOL reformulation of $q$ w.r.t. $C_3$ is:

$$q^e(\mathbf{x}) \text{:-} \quad q^{\mathtt{UCQ}}_{|f_1 \| f_1}(\mathbf{x}) \text{:-} \quad ((R(\mathbf{x}, y) \wedge R'(z, y)) \vee R'(\mathbf{x}, y) \vee B(\mathbf{x}))$$

$$q^{\mathtt{UCQ}}_{f_2 \| f_0}(\mathbf{x}) \text{:-} \quad ((A(\mathbf{x}) \wedge R(\mathbf{x}, y)) \vee (A(\mathbf{x}) \wedge R'(\mathbf{x}, y)) \vee (A(\mathbf{x}) \wedge B(\mathbf{x})))$$

# Search space



# Part VI

# Performance

# CQ-to-JUCQ query reformulation performance

Given the query

$$
\begin{aligned}
q_1(x, y) \text{ :- } & x \text{ rdf:type } y, && (\mathbf{t_1})\\
& x \ ub{:}degreeFrom \ \text{“}http://www.University532.edu\text{”}, && (\mathbf{t_2})\\
& x \ ub{:}memberOf \ \text{“}http://www.Department1.University7.edu\text{”} && (\mathbf{t_3})
\end{aligned}
$$

and the LUBM 100M benchmark:

| JUCQ | #reformulations | exec. time (ms) |
|---|---|---|
| $(t_1, t_2, t_3)^{ref}$ | $2,256$ | **6,387** |
| $(t_1)^{ref} \bowtie (t_2)^{ref} \bowtie (t_3)^{ref}$ | 195 | **1,074,026** |
| $(t_1, t_2)^{ref} \bowtie (t_3)^{ref}$ | 755 | $1,968$ |
| $(t_1)^{ref} \bowtie (t_2, t_3)^{ref}$ | 200 | $846,710$ |
| $(t_1, t_3)^{ref} \bowtie (t_2)^{ref}$ | 568 | **554** |
| $(t_1, t_2)^{ref} \bowtie (t_1, t_3)^{ref}$ | $1,316$ | $2,734$ |
| $(t_1, t_2)^{ref} \bowtie (t_2, t_3)^{ref}$ | 764 | $2,289$ |
| $(t_1, t_3)^{ref} \bowtie (t_2, t_3)^{ref}$ | 576 | 588 |

# Datasets and RDBMS engines

**DBLP** (**8 M**) and **LUBM** (**1 M** and **100 M**) millions triples

- PostgreSQL 9.3.2
- System A
- System B

# Reformulation algorithms

## Basic CQ-to-UCQ algorithm

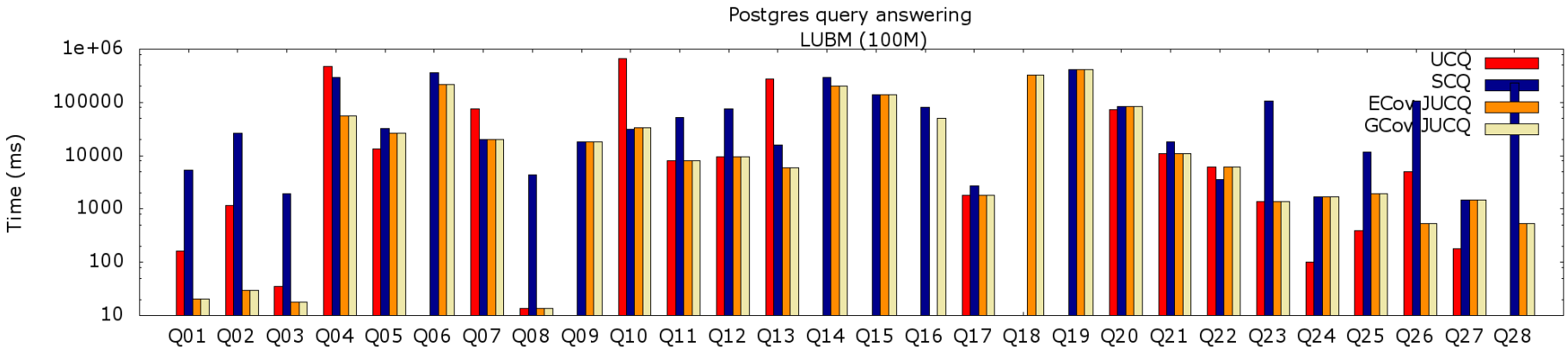
Picked that of [Goasdoué et al., EDBT'13] since it handles the largest known fragment of RDF.

Comparison:

1. **UCQ** reformulation
2. **SCQ** reformulation
3. **Greedy JUCQ** reformulation
4. **Exhaustive JUCQ** reformulation

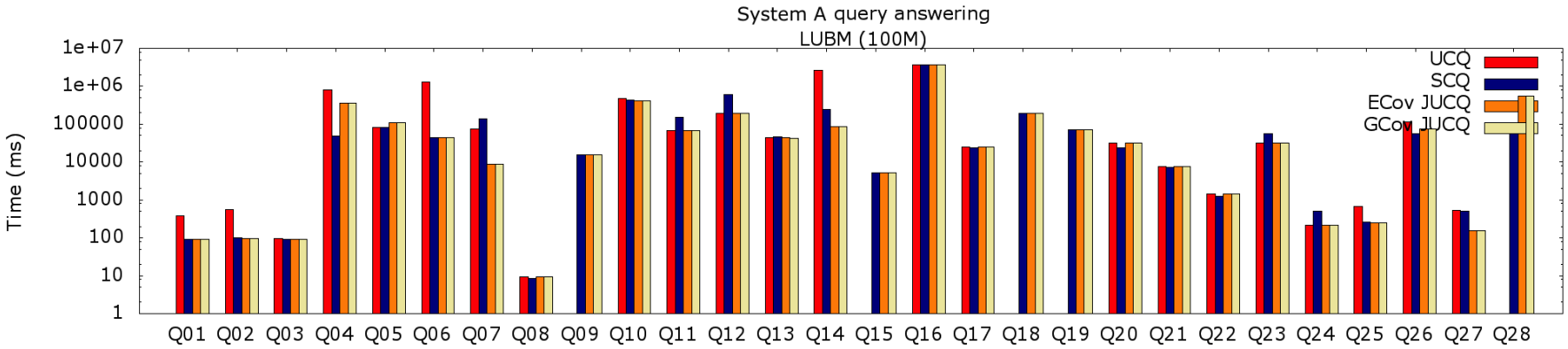
# Query answering on LUBM 100 M using PostgreSQL

28 queries; 2 to 6 atoms; 1 to 318,089 reformulations



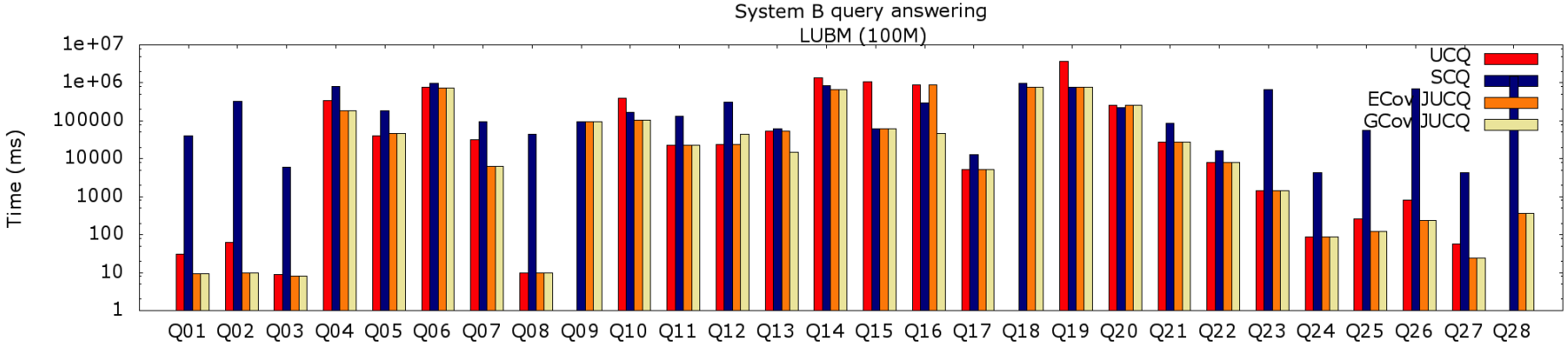
# Query answering on LUBM 100 M using System A

28 queries; 2 to 6 atoms; 1 to 318,089 reformulations



# Query answering on LUBM 100 M using System B

28 queries; 2 to 6 atoms; 1 to 318,089 reformulations



# Optimized reformulation: take-home message

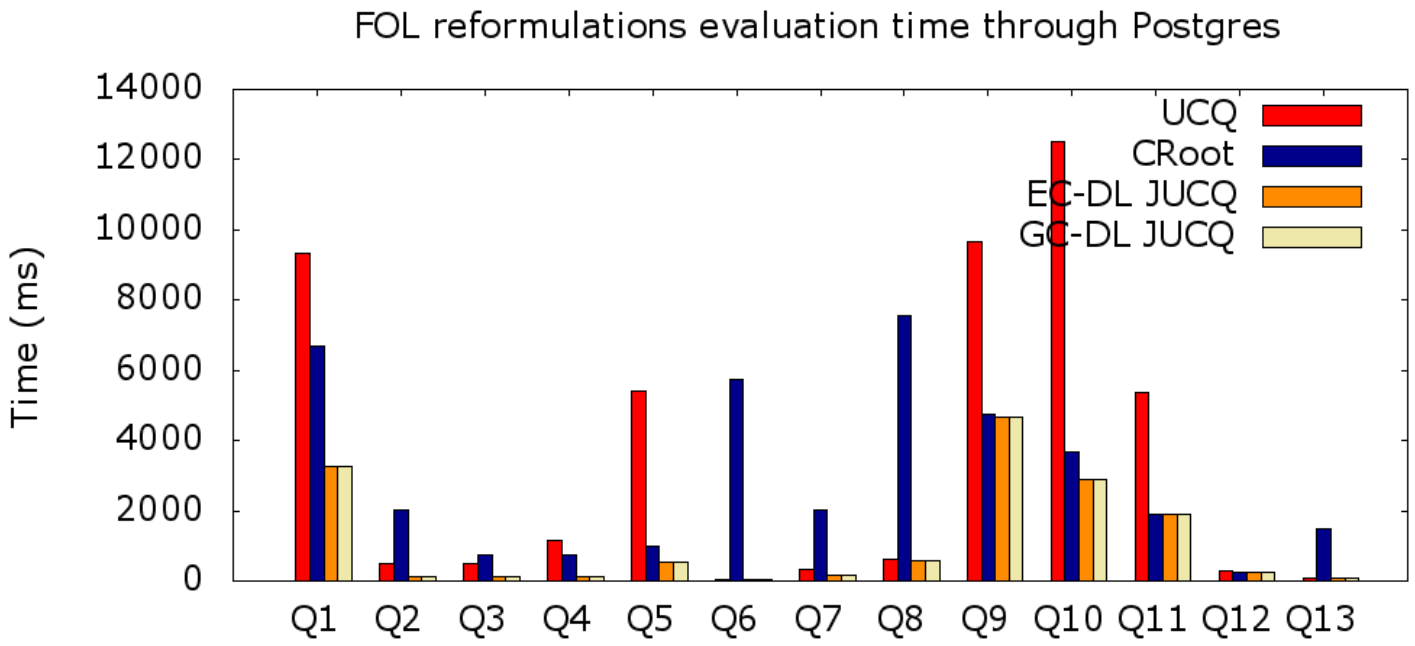
1. Equivalent SQL syntaxes are not equal from the RDBMS optimizer perspective (**inside** or **outside** well-supported dialect)
2. Chosing the reformulation with the help of textbook cost model formulas makes queries feasible or efficient when they were not
3. This amounts to enlarging the optimizer's "can-do" dialect, at a very modest performance overhead.

# Experiments on DL-Lite$_{\mathcal{R}}$ reformulation

1. CQ-to-UCQ reformulation
2. $C_{\mathrm{root}}$ reformulation
3. Our cover-based **greedy** reformulation
4. Our cover-based **exhaustive** reformulation (as a yardstick for the quality of the solution our greedy finds)

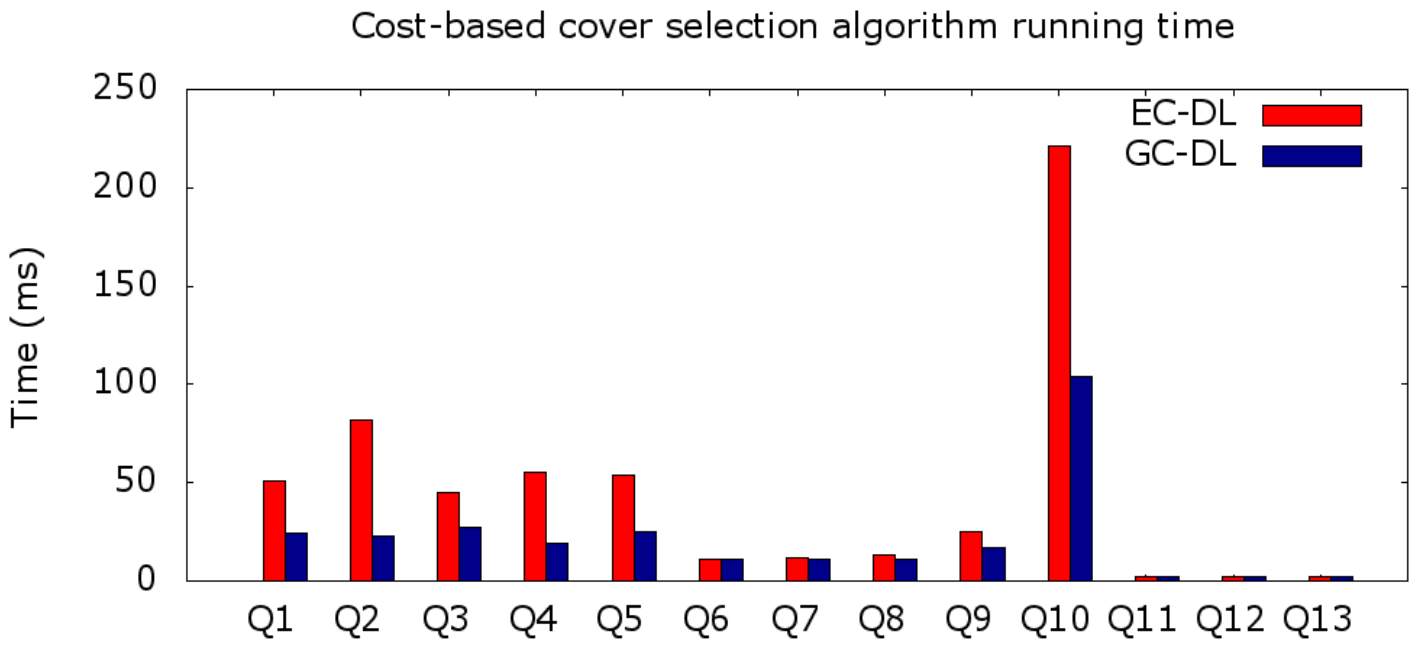
# Query answering on $\text{LUBM}_{20}^{\exists}$ 15 millions facts dataset using PostgreSQL

13 queries; 2 to 10 atoms, 5.77 on average; 35 to 667 reformulations, 290.2 on average.



# Algorithms running time on $\text{LUBM}^{\exists}_{20}$ 15 millions facts dataset

13 queries; 2 to 10 atoms, 5.77 on average; 35 to 667 reformulations, 290.2 on average.



# Part VII

# Conclusion and perspectives

# Where we stand

**Constraints** (a.k.a. semantics) are crucial for applications, so the push is continuous for chosing "the right constraint language"

# Where we stand

**Constraints** (a.k.a. semantics) are crucial for applications, so the push is continuous for chosing "the right constraint language"
We considered semantic query answering through **FOL reduction (i.e., SQL)**.

# Where we stand

**Constraints** (a.k.a. semantics) are crucial for applications, so the push is continuous for chosing "the right constraint language"
We considered semantic query answering through **FOL reduction (i.e., SQL)**.
Not any SQL query resulting from reformulation is handled well by current RDBMSs!

# Where we stand

**Constraints** (a.k.a. semantics) are crucial for applications, so the push is continuous for chosing "the right constraint language"
We considered semantic query answering through **FOL reduction (i.e., SQL)**.
Not any SQL query resulting from reformulation is handled well by current RDBMSs!
Vast performance differences between different reformulations of the same query; cost-based approach

# What is ahead

Continuous push on the **expressivity** - **efficiency** frontier; updates...

# What is ahead

Continuous push on the **expressivity** - **efficiency** frontier; updates...

RDBMSs are highly efficient for some forms of queries (typically conjunctive queries of medium size), not for all FOL reductions of queries under constraints

# What is ahead

Continuous push on the **expressivity - efficiency** frontier; updates...

RDBMSs are highly efficient for some forms of queries (typically conjunctive queries of medium size), not for all FOL reductions of queries under constraints

RDBMSs have convenient features (indexing, join order, transactions) that make them **attractive back-ends for query answering**.

# What is ahead

Continuous push on the **expressivity - efficiency** frontier; updates...

RDBMSs are highly efficient for some forms of queries (typically conjunctive queries of medium size), not for all FOL reductions of queries under constraints

RDBMSs have convenient features (indexing, join order, transactions) that make them **attractive back-ends for query answering**.

"Novel" platforms (MapReduce, NoSQL...) will raise **the same query evaluation performance problems** anyway.

# What is ahead

Continuous push on the **expressivity - efficiency** frontier; updates...

RDBMSs are highly efficient for some forms of queries (typically conjunctive queries of medium size), not for all FOL reductions of queries under constraints

RDBMSs have convenient features (indexing, join order, transactions) that make them **attractive back-ends for query answering**.

"Novel" platforms (MapReduce, NoSQL...) will raise **the same query evaluation performance problems** anyway.

Most attractive languages currently: **DL-Lite, Datalog**$^{\pm}$

# What is ahead

Continuous push on the **expressivity - efficiency** frontier; updates...

RDBMSs are highly efficient for some forms of queries (typically conjunctive queries of medium size), not for all FOL reductions of queries under constraints

RDBMSs have convenient features (indexing, join order, transactions) that make them **attractive back-ends for query answering**.

"Novel" platforms (MapReduce, NoSQL...) will raise **the same query evaluation performance problems** anyway.

Most attractive languages currently: **DL-Lite, Datalog**$^{\pm}$

**New relevance of query optimization literature and research!**