# 20 years of semantic web data management

## from an optimization point of view

François Goasdoué
IRISA, Bard team
University of Rennes

15/6/2026

# Introduction

# The rise of semantic web data management (1/2)

Till the end of the $20^{th}$ century, there have been very few interaction between

- the **Database (DB)** community
  - data management w.r.t. simple mathematical structures for the sake of efficiency
- the **Knowledge Representation & Reasoning (KRR)** community
  - data management w.r.t. human cognitive schemes for the sake of intelligibility.

# The rise of semantic web data management (1/2)

Till the end of the $20^{th}$ century, there have been very few interaction between

- the **Database (DB)** community
  - data management w.r.t. simple mathematical structures for the sake of efficiency
- the **Knowledge Representation & Reasoning (KRR)** community
  - data management w.r.t. human cognitive schemes for the sake of intelligibility.

At the beginning of the $21^{th}$ century, a joint effort between DB and KRR led to the new era of **knowledge-based data management**.
It makes data management **one step closer to end-users** by advocating the use of:

- conceptual models from KRR as **human intelligible front-ends called ontologies**
- data models from DB as **efficient back-end storage**.

# The rise of semantic web data management (2/2)

The **World Wide Web Consortium (W3C)** has strongly contributed to the success of knowledge-based data management with its **semantic web recommendations**, i.e., **standards for knowledge-based data management**:

- the Resource Description Framework (RDF) - a simple graph data model
- the Web Ontology Language (OWL2) - lightweight description logic data models
- the SPARQL query language - a SQL-like query language for RDF/OWL2.

# The rise of semantic web data management (2/2)

The **World Wide Web Consortium (W3C)** has strongly contributed to the success of knowledge-based data management with its **semantic web recommendations**, i.e., **standards for knowledge-based data management**:

- the Resource Description Framework (RDF) - a simple graph data model
- the Web Ontology Language (OWL2) - lightweight description logic data models
- the SPARQL query language - a SQL-like query language for RDF/OWL2.

The advent of the RDF/OWL2/SPARQL standards has focused the attention on **practical semantic web data management**, which led to:

- key research contributions in Artificial Intelligence, Databases and Web venues
- key industrial or open source software (Jena, Oracle, RDFox,...)

# The rise of semantic web data management (2/2)

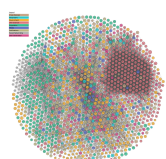
The **World Wide Web Consortium (W3C)** has strongly contributed to the success of knowledge-based data management with its **semantic web recommendations**, i.e., **standards for knowledge-based data management**:

- the Resource Description Framework (RDF) - a simple graph data model
- the Web Ontology Language (OWL2) - lightweight description logic data models
- the SPARQL query language - a SQL-like query language for RDF/OWL2.

The advent of the RDF/OWL2/SPARQL standards has focused the attention on **practical semantic web data management**, which led to:

- key research contributions in Artificial Intelligence, Databases and Web venues
- key industrial or open source software (Jena, Oracle, RDFox,...)

**Large societies** have promptly adhered to semantic web data management!



# The semantic web data management challenge

**Knowledge-based/Semantic web data management** consists in

**performing DB management tasks**

- e.g., consistency checking, query answering, etc.

**on knowledge bases (KBs) expressed in (or inspired from) KRR languages**

- e.g., RDF, description logics/OWL2, datalog$\pm$/existential rules, etc.

# The semantic web data management challenge

**Knowledge-based/Semantic web data management** consists in

**performing DB management tasks**

- e.g., consistency checking, query answering, etc.

**on knowledge bases (KBs) expressed in (or inspired from) KRR languages**

- e.g., RDF, description logics/OWL2, datalog$\pm$/existential rules, etc.

Performing these tasks correctly requires automated reasoning

# The semantic web data management challenge

**Knowledge-based/Semantic web data management** consists in

**performing DB management tasks**

- e.g., consistency checking, query answering, etc.

**on knowledge bases (KBs) expressed in (or inspired from) KRR languages**

- e.g., RDF, description logics/OWL2, datalog$\pm$/existential rules, etc.

Performing these tasks correctly requires automated reasoning, **which comes at a price!**

# The semantic web data management challenge

**Knowledge-based/Semantic web data management** consists in

**performing DB management tasks**

- e.g., consistency checking, query answering, etc.

**on knowledge bases (KBs) expressed in (or inspired from) KRR languages**

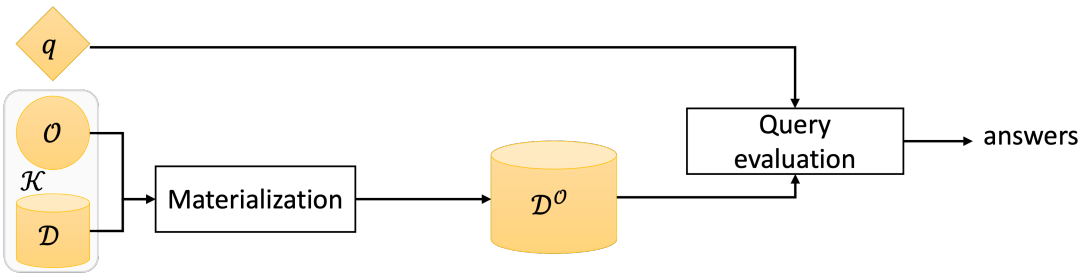
- e.g., RDF, description logics/OWL2, datalog$\pm$/existential rules, etc.

Performing these tasks correctly requires automated reasoning, **which comes at a price!**

We focus on ontology-mediated query answering (OMQA), i.e., query answering on KBs

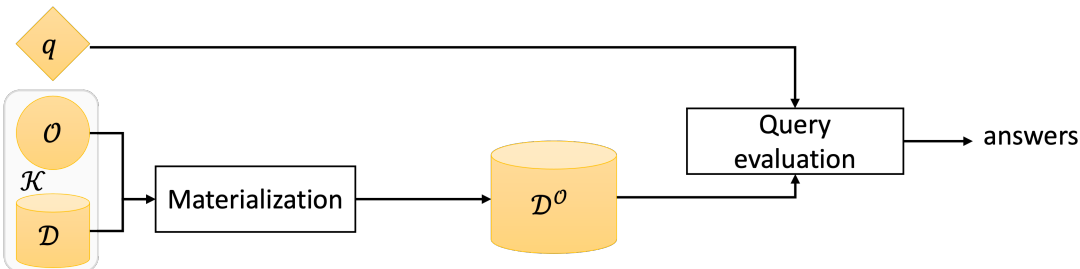
# OMQA via materialization (not the topic of this talk)

**Materialization** compiles domain knowledge into the data.



# OMQA via materialization (not the topic of this talk)

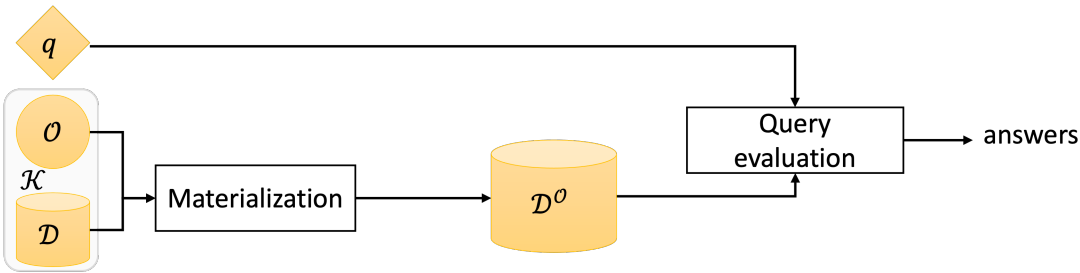
**Materialization** compiles domain knowledge into the data.




**Materialization is not always possible!**

# OMQA via materialization (not the topic of this talk)

**Materialization** compiles domain knowledge into the data.



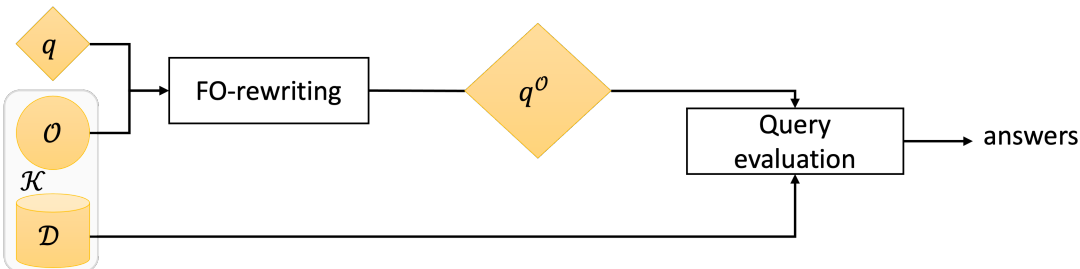

**Materialization is not always possible!**

When it is possible, it

- \+ is **fast at query time**
- – needs **efficient maintenance** when the database is updated ⇒ **optimization!**

# OMQA via FO-rewriting (topic of this talk)

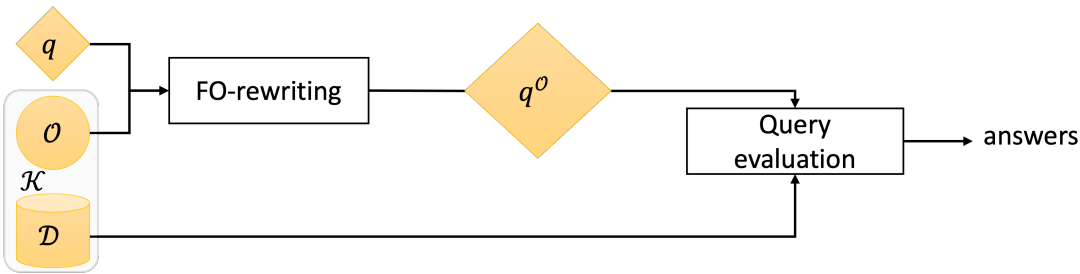
**FO-rewriting** compiles domain knowledge into the queries.




**FO-rewriting is not always possible!**

# OMQA via FO-rewriting (topic of this talk)

**FO-rewriting** compiles domain knowledge into the queries.




**FO-rewriting is not always possible!**

When it is possible, it

+ does not need **maintenance** when the database is updated.
− may produce complex rewritings that are **challenging to evaluate** ⇒ **optimization!**

# Overview

1. Introduction
2. Preliminaries
3. Act I: OMQA via FO-rewriting
4. Act II: query rewriting languages
5. Act III: data-dependent optimization of rewritings
6. Conclusion and perspectives

# Preliminaries

# Queries

We consider FO queries of the form $q(\bar{x}) = \phi$ with $\bar{x}$ the free variables of the formula $\phi$.

We will start with the following query languages:

| Language | First-Order Logic syntax | Relational algebra syntax |
|---|---|---|
| CQ | $q(\bar{x}) = (\exists \bar{y})\ \bigwedge_{i=1}^{n} atom_i$ | $q(\bar{x}) = \Pi_{\bar{x}}(\bowtie_{i=1}^{n} atom_i)$ |
| UCQ | $q(\bar{x}) = \bigvee_{i=1}^{n} CQ_i$ | $q(\bar{x}) = \bigcup_{i=1}^{n} CQ_i$ |

## Running example

We consider the CQ asking for the supervisees who work with **h** that is a researcher:

$$q(x) = (\exists y)\, R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x).$$

# Knowledge bases (KBs)

We consider FO KBs expressed using **existential rules**.

A **KB** $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ such that

- $\mathcal{O}$ is an **ontology** made of rules of the form $\forall \bar{x}(body(\bar{x}) \rightarrow head(\bar{x}))$, where $body(\bar{x})$ and $head(\bar{x})$ are CQs, denoted w.l.o.g. by $body(\bar{x}) \rightarrow head(\bar{x})$
- $\mathcal{D}$ is a **database** is a set of incomplete facts.

The semantics of $\mathcal{K}$ is that of the formula: $\bigwedge_{r \in \mathcal{O}} r \wedge \exists \bar{v} \bigwedge_{f \in \mathcal{D}} f$ with $\bar{v}$ the variables in $\mathcal{D}$.

## Running example (cont.)

We consider the KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ such that:

$$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$$

and

$$\mathcal{D} = \{R(\mathbf{f}), R(\mathbf{h}), sup(\mathbf{f}, \mathbf{w}), sup(\mathbf{h}, \mathbf{w}), PhD(\mathbf{w}), ww(\mathbf{f}, \mathbf{h}), R(u), ww(u, \mathbf{c}), PhD(\mathbf{c})\}$$

# Ontology-mediated query answering (OMQA)

We consider the standard **certain answer semantics** for OMQA.

A **(certain) answer to a query** $q(\bar{x})$ of arity $n$ on a KB $\mathcal{K}$ is a tuple $\bar{\mathbf{t}}$ of $n$ constants from $\mathcal{K}$ such that $\mathcal{K} \models q(\bar{\mathbf{t}})$, where

- $\models$ is the FO entailment relation
- $q(\bar{\mathbf{t}})$ is the Boolean query obtained by instantiating $\bar{x}$ with $\bar{\mathbf{t}}$ in $q$.

We denote by $ans(q, \mathcal{K})$ the *answer set* of $q$ on $\mathcal{K}$.

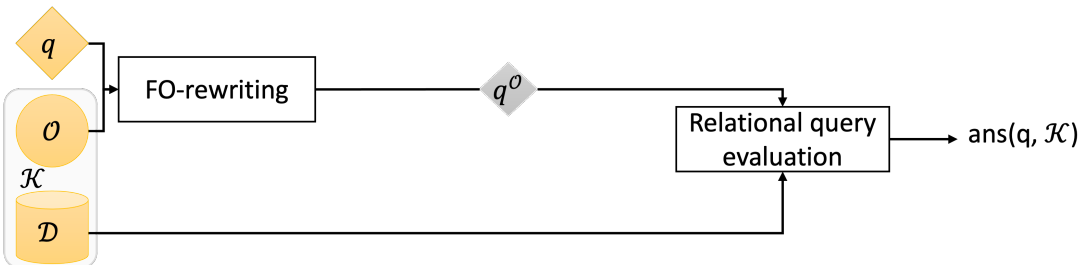
## Running example (cont.)

The answer set of $q(x) = R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)$ on $\mathcal{K}$ is $ans(q, \mathcal{K}) = \{\langle \mathbf{w} \rangle\}$:

- $R(\mathbf{h}) \in \mathcal{D}$
- the fact $ww(\mathbf{h}, \mathbf{w})$ entailed from $sup(\mathbf{h}, \mathbf{w}) \in \mathcal{D}$ and $r_2 = sup(x, y) \rightarrow ww(x, y)$
- either $sup(\mathbf{f}, \mathbf{w}) \in \mathcal{D}$ or $sup(\mathbf{h}, \mathbf{w}) \in \mathcal{D}$.

# Act I: OMQA via FO-rewriting

# OMQA via FO-rewriting



# OMQA via FO-rewriting by example

## Running example (cont.)

The query rewriting of

$$q(x) = R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)$$

w.r.t.

$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$

is:

$$\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)) && (1) \\
& \vee (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)) && (2) && = (1) + r_3 \\
& \vee (R(\mathbf{h}) \wedge sup(\mathbf{h}, x) \wedge \cancel{sup(y, x)}) && (3) && = (1) + r_2 \\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)) && (4) && = (1) + r_1 \\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)) && (5) && = (2) + r_1 \text{ or } (4) + r_3 \\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)) && (6) && = (4) + r_2 \\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)) && (7) && = (5) + r_2 \text{ or } (6) + r_3
\end{aligned}$$

# OMQA via FO-rewriting by example

## Running example (cont.)

The answer set of $q$ w.r.t. $\mathcal{K} = (\mathcal{O}, \mathcal{D})$ is then obtained by the relational evaluation of

$$
\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)) && (1)\\
& \vee (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)) && (2)\\
& \vee (R(\mathbf{h}) \wedge sup(\mathbf{h}, x)) && (3)\\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)) && (4)\\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)) && (5)\\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)) && (6)\\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)) && (7)
\end{aligned}
$$

on

$$\mathcal{D} = \{R(\mathbf{f}), R(\mathbf{h}), sup(\mathbf{f}, \mathbf{w}), sup(\mathbf{h}, \mathbf{w}), PhD(\mathbf{w}), ww(\mathbf{f}, \mathbf{h}), R(u), ww(u, \mathbf{c}), PhD(\mathbf{c})\}$$

i.e., $ans(q, \mathcal{K}) = eval(q^{\mathcal{O}}_{\mathrm{UCQ}}, \mathcal{D}) = \{\langle \mathbf{w} \rangle\}$.

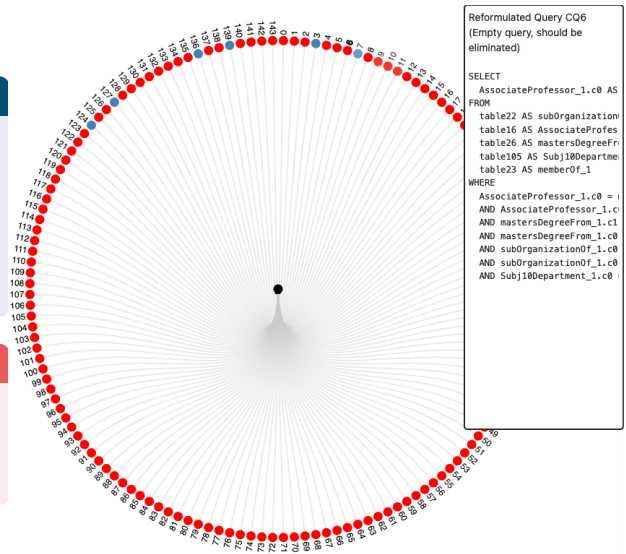
# OMQA via FO-rewriting is computationally expensive

## Cost of FO-rewriting

- The worst-case size of a rewriting $q^{\mathcal{O}}$ is **exponential** in the size of $q$ even for lightweight ontology languages
- The time to compute $q^{\mathcal{O}}$ is **negligible** w.r.t. the time of its evaluation on $\mathcal{D}$.

## Cost of evaluating a query rewriting

Query rewritings may be **complex** in practice and **challenging to evaluate**, even for highly-optimized RDBMSs.



# Take home message

1. FO-rewriting **reduces** query answering on KBs to query answering on databases
2. FO-rewriting is **computationally expensive**
3. FO-rewriting needs to be **optimized for RDBMSs**

# Related work and contributions

Some references from the literature on devising OMQA via FO-rewriting for CQs:

| | **Query rewriting language** |
|---|---|
| **Ontology language** | UCQ |
| datalog$\pm$/existential rules | [GOP11, GOP14, KLMT15] |
| description logics/OWL | [CDL$^+$07, PHM09, CTS11, VSS12] |
| RDFS | **[GMR13, BGMM19]** |

# Act II: query rewriting languages

# FO-rewriting optimization: two main options

1. Produce query rewritings faster
   - conceptually interesting but of limited interest from the practical point of view
     - time to produce the rewriting $<<$ time to evaluate the rewriting on the database
2. **Make the query rewritings faster to evaluate**

# Key observation: UCQ rewritings are highly redundant!

Logical redundancy in rewritings = redundant computation when rewritings are evaluated!

## Running example (cont.)

The query rewriting of $q(x) = R(\mathbf{h}) \land ww(\mathbf{h}, x) \land sup(y, x)$ w.r.t.

$$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$$

is:

$$\begin{aligned} q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = & (R(\mathbf{h}) \land ww(\mathbf{h}, x) \land sup(y, x)) \quad (1) \\ \lor & (R(\mathbf{h}) \land ww(\mathbf{h}, x) \land PhD(x)) \quad (2) \\ \lor & (R(\mathbf{h}) \land sup(\mathbf{h}, x) \land sup(y, x)) \quad (3) \\ \lor & (R(\mathbf{h}) \land ww(x, \mathbf{h}) \land sup(y, x)) \quad (4) \\ \lor & (R(\mathbf{h}) \land ww(x, \mathbf{h}) \land PhD(x)) \quad (5) \\ \lor & (R(\mathbf{h}) \land sup(x, \mathbf{h}) \land sup(y, x)) \quad (6) \\ \lor & (R(\mathbf{h}) \land sup(x, \mathbf{h}) \land PhD(x)) \quad (7) \end{aligned}$$

# Key observation: UCQ rewritings are highly redundant!

## Running example (cont.)

The query rewriting of $q(x) = R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)$ w.r.t.

$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$

is:

$$
\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = (\ & R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)\ ) && (1)\\
\vee (\ & R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)\ ) && (2)\\
\vee (\ & R(\mathbf{h}) \wedge sup(\mathbf{h}, x) \wedge sup(y, x)\ ) && (3)\\
\vee (\ & R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)\ ) && (4)\\
\vee (\ & R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)\ ) && (5)\\
\vee (\ & R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)\ ) && (6)\\
\vee (\ & R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)\ ) && (7)
\end{aligned}
$$

# Taming redundancy in rewritings: minimal UCQ rewritings

FO-rewriting techniques have been proposed to compute UCQ rewritings within which:

- each CQ is minimal (no redundant subCQs)
- no CQ is redundant with another

Pros & cons of such techniques

+ Such techniques optimize UCQ rewritings to the possible extent
- Limited performance improvement
  - The size of a minimal UCQ rewriting is **exponential** in the size of the query even for lightweight ontology languages
  - Minimal UCQ rewritings are **highly redundant**

# Minimal UCQ rewritings are highly redundant!

## Running example (cont.)

The query rewriting of $q(x) = R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)$ w.r.t.

$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$

is:

$$
\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = (\ & R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)\ ) && (1)\\
\vee (\ & R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)\ ) && (2)\\
\vee (\ & R(\mathbf{h}) \wedge sup(\mathbf{h}, x)) && (3)\\
\vee (\ & R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)\ ) && (4)\\
\vee (\ & R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)\ ) && (5)\\
\vee (\ & R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)\ ) && (6)\\
\vee (\ & R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)\ ) && (7)
\end{aligned}
$$

# Taming redundancy in rewritings: compact USCQ rewritings

**Breakthrough in 2013:** adopting a (relational) rewriting language other than UCQ!

[Tho13] introduces an FO-rewriting technique with USCQ as a rewriting language for **compact rewritings that drastically limit redundancy**.

| Language | First-Order Logic syntax | Relational algebra syntax |
|---|---|---|
| SCQ | $q(\bar{x}) = (\exists \bar{y}) \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} atom_i^j$ | $q(\bar{x}) = \Pi_{\bar{x}}(\bowtie_{i=1}^{n} \bigcup_{j=1}^{m_i} atom_i^j)$ |
| USCQ | $q(\bar{x}) = \bigvee_{i=1}^{n} SCQ_i$ | $q(\bar{x}) = \bigcup_{i=1}^{n} SCQ_i$ |

# Taming redundancy in rewritings: compact USCQ rewritings

This allows a shift from highly redundant rewritings...

## Running example (cont.)

The query rewriting of $q(x) = R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)$ w.r.t.

$$\mathcal{O} = \{r_1 = ww(x, y) \rightarrow ww(y, x), r_2 = sup(x, y) \rightarrow ww(x, y), r_3 = PhD(x) \rightarrow sup(y, x)\}$$

is:

$$
\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)) && (1)\\
\vee & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)) && (2)\\
\vee & (R(\mathbf{h}) \wedge sup(\mathbf{h}, x)) && (3)\\
\vee & (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)) && (4)\\
\vee & (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)) && (5)\\
\vee & (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)) && (6)\\
\vee & (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)) && (7)
\end{aligned}
$$

# Taming redundancy in rewritings: compact USCQ rewritings

. . . to rewriting with limited redundancy.

## Running example (cont.)

The query rewriting of

$$q(x) = R(\mathbf{h}) \land ww(\mathbf{h}, x) \land sup(y, x)$$

w.r.t.

$\mathcal{O} = \{r_1 = ww(x, y) \to ww(y, x), r_2 = sup(x, y) \to ww(x, y), r_3 = PhD(x) \to sup(y, x)\}$

is:

$$q^{\mathcal{O}}_{\mathrm{USCQ}}(x) = ( R(\mathbf{h}) ) \land ( ww(\mathbf{h}, x) \lor sup(\mathbf{h}, x) \lor ww(x, \mathbf{h}) \lor sup(x, \mathbf{h}) ) \land ( sup(y, x) \lor PhD(x) )$$

But. . . is this the killer rewriting language w.r.t. performance?

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $\mathit{rewrite}(a_1 \wedge a_2 \wedge a_3)$ | | |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $\mathit{rewrite}(a_1 \wedge a_2 \wedge a_3)$ | UCQ | |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $\mathit{rewrite}(a_1 \wedge a_2 \wedge a_3)$ | UCQ | **6,387** |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $rewrite(a_1 \wedge a_2 \wedge a_3)$ | UCQ | **6,387** |
| $rewrite(a_1) \wedge rewrite(a_2) \wedge rewrite(a_3)$ | | |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $\mathit{rewrite}(a_1 \wedge a_2 \wedge a_3)$ | UCQ | **6,387** |
| $\mathit{rewrite}(a_1) \wedge \mathit{rewrite}(a_2) \wedge \mathit{rewrite}(a_3)$ | SCQ | |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $\mathit{rewrite}(a_1 \wedge a_2 \wedge a_3)$ | UCQ | **6,387** |
| $\mathit{rewrite}(a_1) \wedge \mathit{rewrite}(a_2) \wedge \mathit{rewrite}(a_3)$ | SCQ | **1,074,026** |

# Use case: asking CQs on RDF KBs
# Performance with different but equivalent query rewritings

Given the query

$$
\begin{aligned}
q(x, y) \;= \; & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

the LUBM100M dataset and a CQ-to-UCQ rewriting algorithm *rewrite*

| Rewriting | a.k.a. | PostgreSQL exec. time (ms) |
|---|---|---|
| $rewrite(a_1 \wedge a_2 \wedge a_3)$ | UCQ | **6,387** |
| $rewrite(a_1) \wedge rewrite(a_2) \wedge rewrite(a_3)$ | SCQ | **1,074,026** |
| $rewrite(a_1 \wedge a_2) \wedge rewrite(a_3)$ | | 1,968 |
| $rewrite(a_1) \wedge rewrite(a_2 \wedge a_3)$ | | 846,710 |
| $rewrite(a_1 \wedge a_3) \wedge rewrite(a_2)$ | | **554** |
| $rewrite(a_1 \wedge a_2) \wedge rewrite(a_1 \wedge a_3)$ | | 2,734 |
| $rewrite(a_1 \wedge a_2) \wedge rewrite(a_2 \wedge a_3)$ | | 2,289 |
| $rewrite(a_1 \wedge a_3) \wedge rewrite(a_2 \wedge a_3)$ | | 588 |

# Cost-based FO-rewriting

**Breakthrough in 2015-16:** 1 CQ $\rightarrow$ N rewritings + cost model of rewriting evaluation

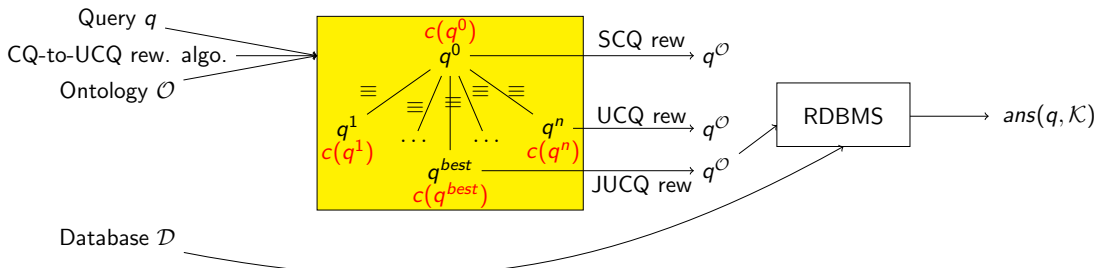
[BGM15, BGM16] introduces an FO-rewriting technique with JUCQ as a rewriting language so that:

- a CQ has a set of syntactically different but semantically equivalent rewritings
- out of which a rewriting with best estimated evaluation cost is selected.

| Language | First-Order Logic syntax | Relational algebra syntax |
|---|---|---|
| CQ | $q(\bar{x}) = (\exists \bar{y})\ \bigwedge_{i=1}^{n} atom_i$ | $q(\bar{x}) = \Pi_{\bar{x}}(\bowtie_{i=1}^{n} atom_i)$ |
| UCQ | $q(\bar{x}) = \bigvee_{i=1}^{n} CQ_i$ | $q(\bar{x}) = \bigcup_{i=1}^{n} CQ_i$ |
| JUCQ | $q(\bar{x}) = \bigwedge_{i=1}^{n} UCQ_i$ | $q(\bar{x}) = \Pi_{\bar{x}}(\bowtie_{i=1}^{n} UCQ_i)$ |

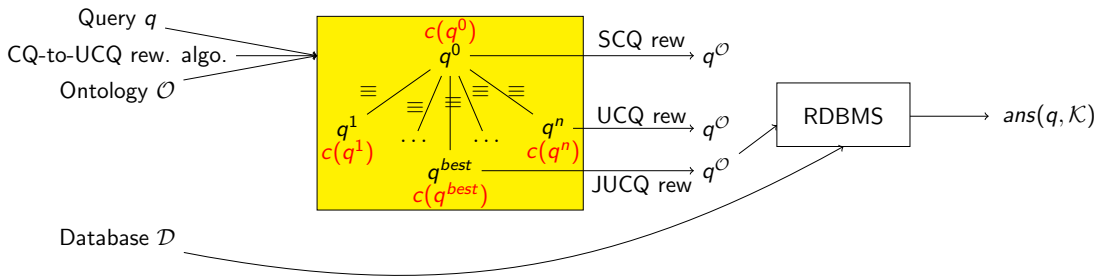
# Cost-based FO-rewriting: overview

To answer the CQ $q$ on the KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$:



# Cost-based FO-rewriting: overview

To answer the CQ $q$ on the KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$:




In general, the search space is too large for fast optimization time!

# Cost-based FO-rewriting: greedy CQ-to-JUCQ rewriting

Given the query

$$
\begin{aligned}
q(x, y) \;= {} & t(x, \text{rdf:type}, y), && (\mathbf{a_1}) \\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2}) \\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

a cost-based greedy exploration is:

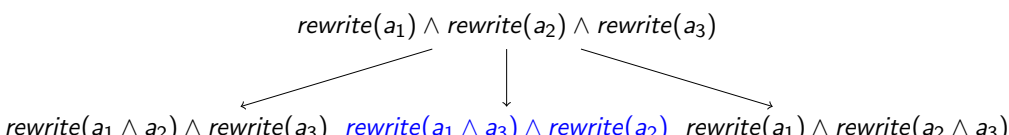
$$rewrite(a_1) \wedge rewrite(a_2) \wedge rewrite(a_3)$$

# Cost-based FO-rewriting: greedy CQ-to-JUCQ rewriting

Given the query

$$
\begin{aligned}
q(x, y) \;= {} & t(x, \text{rdf:type}, y), && (\mathbf{a_1})\\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http://www.University532.edu”}), && (\mathbf{a_2})\\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http://www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

a cost-based greedy exploration is:

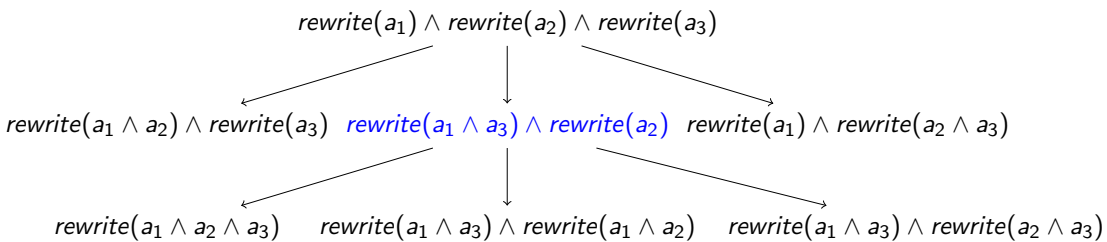


# Cost-based FO-rewriting: greedy CQ-to-JUCQ rewriting

Given the query

$$
\begin{aligned}
q(x,y) \;= {} & t(x, \text{rdf:type}, y), && (\mathbf{a_1})\\
& \wedge\, t(x, \text{lubm:degreeFrom}, \text{“http} : //\text{www.University532.edu”}), && (\mathbf{a_2})\\
& \wedge\, t(x, \text{lubm:memberOf}, \text{“http} : //\text{www.Department1.University7.edu”}) && (\mathbf{a_3})
\end{aligned}
$$

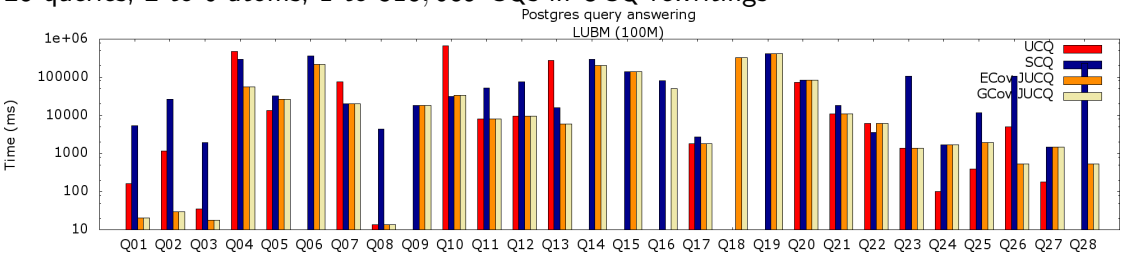
a cost-based greedy exploration is:



# Cost-based FO-rewriting: sample experimental results

Query answering on LUBM 100 M using PostgreSQL

28 queries; 2 to 6 atoms; 1 to 318,089 CQs in UCQ rewritings



# Take home message

1. Alternative query rewritings may have **very different evaluation times**
2. A cost model for rewriting evaluation gives rapidly **estimations of evaluation costs**
3. Selecting a rewriting from alternative ones using a cost-model is **the best option**

# Related work and contributions

Some references from the literature on devising OMQA via FO-rewriting for CQs:

| | **Query rewriting language** | | | |
|---|---|---|---|---|
| **Ontology language** | UCQ | USCQ | JUCQ | datalog$^{nr}$ |
| datalog$\pm$/existential rules | [GOP11, GOP14, KLMT15] | [Tho13] | | [OP11, GS12] |
| description logics/OWL | [CDL$^{+}$07, PHM09, CTS11, VSS12] | | **[BGM16]** | [RA10] |
| RDFS | [GMR13, BGMM19] | | **[BGM15]** | |

# Act III: data-dependent optimization of rewritings

# Key observation: rewritings are too complex w.r.t. OMQA!

Complex rewritings = rewritings that are costly to evaluate by RDBMSs!

# Key observation: rewritings are too complex w.r.t. OMQA!

Complex rewritings = rewritings that are costly to evaluate by RDBMSs!

## First-Order rewritability

Given a query language $\mathcal{L}_\mathcal{Q}$ and an ontology language $\mathcal{L}_\mathcal{O}$, the OMQA setting $(\mathcal{L}_\mathcal{Q}, \mathcal{L}_\mathcal{O})$ is FO-rewritable iff for each $q \in \mathcal{L}_\mathcal{Q}$ and $\mathcal{O} \in \mathcal{L}_\mathcal{O}$ there exists an FO-query $q^\mathcal{O}$ such that:

$$\text{for each database } \mathcal{D},\ \mathit{ans}(q, \langle \mathcal{O}, \mathcal{D} \rangle) = \mathit{eval}(q^\mathcal{O}, \mathcal{D})$$

where $\mathit{eval}(\cdot, \cdot)$ is the standard relational query evaluation.

# Key observation: rewritings are too complex w.r.t. OMQA!

Complex rewritings = rewritings that are costly to evaluate by RDBMSs!

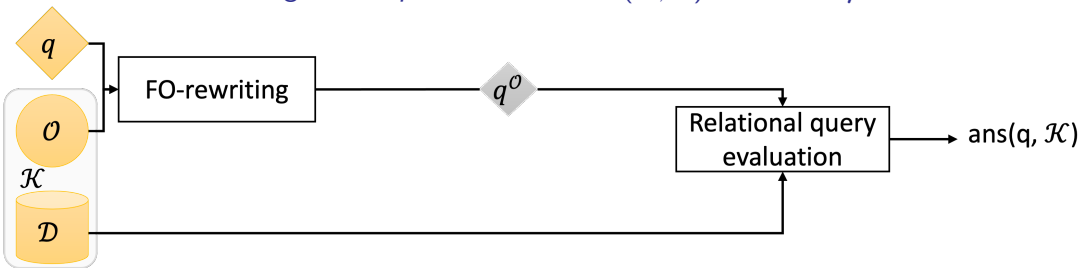
## First-Order rewritability

Given a query language $\mathcal{L}_\mathcal{Q}$ and an ontology language $\mathcal{L}_\mathcal{O}$, the OMQA setting $(\mathcal{L}_\mathcal{Q}, \mathcal{L}_\mathcal{O})$ is FO-rewritable iff for each $q \in \mathcal{L}_\mathcal{Q}$ and $\mathcal{O} \in \mathcal{L}_\mathcal{O}$ there exists an FO-query $q^\mathcal{O}$ such that:

for each database $\mathcal{D}$, $ans(q, \langle \mathcal{O}, \mathcal{D} \rangle) = eval(q^\mathcal{O}, \mathcal{D})$

where $eval(\cdot, \cdot)$ is the standard relational query evaluation.

OMQA via FO-rewriting: when $q$ is asked on $\mathcal{K} = (\mathcal{O}, \mathcal{D})$, $\mathcal{D}$ is **fixed/known!**



# Key observation: rewritings are too complex w.r.t. OMQA!

A **minimal UCQ rewriting** models

- all the (worst-case exponentially many) non-redundant specializations of the query

# Key observation: rewritings are too complex w.r.t. OMQA!

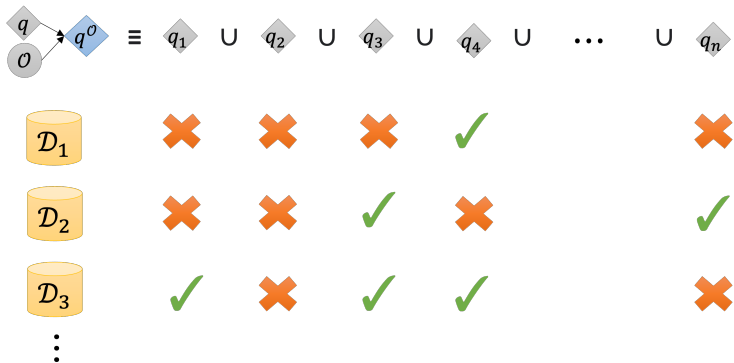
A **minimal UCQ rewriting** models

- all the (worst-case exponentially many) non-redundant specializations of the query
- **all the ways all the possible databases may store answers to the query!**

# Key observation: rewritings are too complex w.r.t. OMQA!

A **minimal UCQ rewriting** models

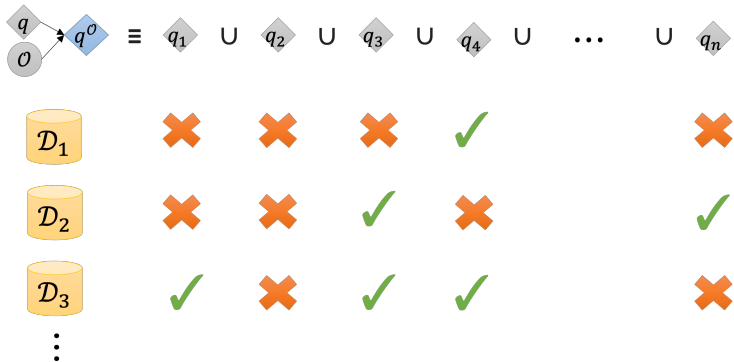
- all the (worst-case exponentially many) non-redundant specializations of the query
- **all the ways all the possible databases may store answers to the query!**



# Key observation: rewritings are too complex w.r.t. OMQA!

A **minimal UCQ rewriting** models

- all the (worst-case exponentially many) non-redundant specializations of the query
- **all the ways all the possible databases may store answers to the query!**




Any (USCQ, JUCQ,...) rewriting is equivalent to a minimal UCQ rewriting, hence somehow models all the ways all the possible databases may store answers to the query!

# Key observation: rewritings are too complex w.r.t. OMQA!

**Breakthrough in 2021:** data-dependent optimization of rewritings!

# Key observation: rewritings are too complex w.r.t. OMQA!

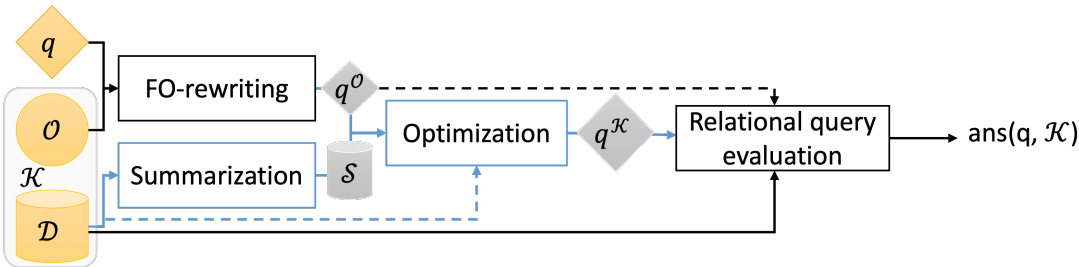
**Breakthrough in 2021:** data-dependent optimization of rewritings!

[HVGJ21, HVGJ23, HVGJ24] compute **simpler rewritings with the same answers (just) on the database at hand**.

# Key observation: rewritings are too complex w.r.t. OMQA!

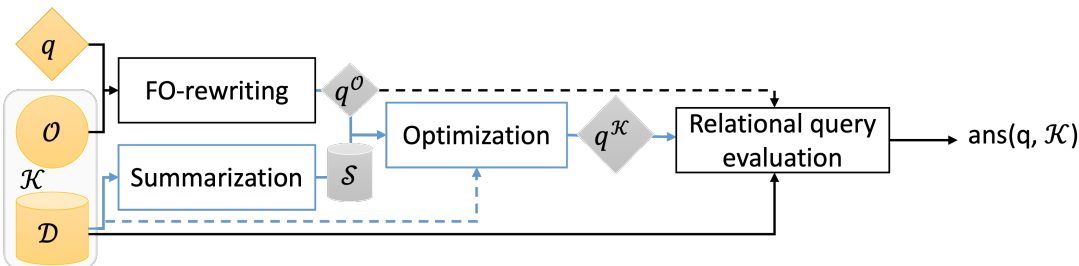
**Breakthrough in 2021:** data-dependent optimization of rewritings!

[HVGJ21, HVGJ23, HVGJ24] compute **simpler rewritings with the same answers (just) on the database at hand**.



# Data-dependent FO-rewriting: requirements

1. For **generality**, we optimize a $(\wedge, \vee)$-CQ rewriting $q^{\mathcal{O}}$
   - Rewriting languages uses CQ as basic building block: U**CQ**, US**CQ** and JU**CQ**
2. For **correctness**, we use an optimization function $\Omega$ such that $q^{\mathcal{K}} = \Omega(q^{\mathcal{O}}, \mathcal{S})$ where
   - $q^{\mathcal{K}} \models q^{\mathcal{O}}$ for **soundness**
   - $eval(q^{\mathcal{O}}, \mathcal{D}) = eval(q^{\mathcal{K}}, \mathcal{D})$ for **completeness**
3. For **efficiency**, we require for a "relevant" cost function $c$:
   - $c(\Omega(q^{\mathcal{O}}, \mathcal{S})) + c(eval(q^{\mathcal{K}}, \mathcal{D})) \leq c(eval(q^{\mathcal{O}}, \mathcal{D}))$



# Data-dependent FO-rewriting: overview

**Optimization technique:** we rewrite a $(\wedge, \vee)$-CQ rewriting from the bottom up, **while propagating the effect of removing subCQs with no answers on the database**.

## (General) Database summary

A summary $\mathcal{S}$ of a database $\mathcal{D}$ is a **homomorphic approximation** of it: $\mathcal{S} = \mathcal{D}_\sigma$ with $\sigma$ a database-to-summary homomorphism that maps each variable in $\mathcal{D}$ to a variable or constant in $\mathcal{S}$, and each constant in $\mathcal{D}$ to constant in $\mathcal{S}$.

A database is a summary of itself, with $\sigma$ the identity function.

# Data-dependent FO-rewriting: overview

**Optimization technique:** we rewrite a $(\wedge, \vee)$-CQ rewriting from the bottom up, **while propagating the effect of removing subCQs with no answers on the database**.

## (General) Database summary

A summary $\mathcal{S}$ of a database $\mathcal{D}$ is a **homomorphic approximation** of it: $\mathcal{S} = \mathcal{D}_\sigma$ with $\sigma$ a database-to-summary homomorphism that maps each variable in $\mathcal{D}$ to a variable or constant in $\mathcal{S}$, and each constant in $\mathcal{D}$ to constant in $\mathcal{S}$.

A database is a summary of itself, with $\sigma$ the identity function.

## Identifying empty CQs with a database summary

If a CQ $q$ has an answer on the database $\mathcal{D}$, then $q_\sigma$ has an answer on its summary $\mathcal{S}$

$\Leftrightarrow$

**If $q_\sigma$ has no answer on $\mathcal{S}$, then $q$ has no answer on $\mathcal{D}$**

with $\sigma$ the $\mathcal{D}$-to-$\mathcal{S}$ homomorphism.

# Data-dependent FO-rewriting: overview

In an optimization setting, summaries must be **small and fast to compute/maintain**.

## (Concrete) Database summary: quotient database w.r.t. the $\equiv_\Omega$ equivalence relation

- The database-to-summary homomorphism $\sigma$ is defined by an equivalence relation on the database terms.
- We use the $\equiv_\Omega$ equivalence relation such that $t_1 \equiv_\Omega t_2$ iff:
  - $t_1, t_2$ are instances of a same unary relation/class/concept
  - $t_1 \equiv_\Omega t_3$ and $t_2 \equiv_\Omega t_3$.

Quotient databases can be rapidly computed/maintained via union-find-delete structures.

## Running example (cont.)

$$\mathcal{D} = \{R(\mathbf{f}), R(\mathbf{h}), sup(\mathbf{f}, \mathbf{w}), sup(\mathbf{h}, \mathbf{w}), PhD(\mathbf{w}), ww(\mathbf{f}, \mathbf{h}), R(u), ww(u, \mathbf{c}), PhD(\mathbf{c})\}$$

is summarized into

$$\mathcal{S} = \{R(\mathbf{r}), sup(\mathbf{r}, \mathbf{p}), PhD(\mathbf{p}), ww(\mathbf{r}, \mathbf{r}), ww(\mathbf{r}, \mathbf{p})\}$$

where $\mathbf{r}$ represents $\{\mathbf{f}, \mathbf{h}, u\}$ and $\mathbf{p}$ represents $\{\mathbf{w}, \mathbf{c}\}$.

# Data-dependent FO-rewriting: overview

**Optimization technique:** we rewrite a $(\wedge, \vee)$-CQ rewriting from the bottom up, **while propagating the effect of removing subCQs with no answers on the database**.

## The $\Omega$ optimization function

- The optimization of a CQ $q$ is:

$$\Omega(q, \mathcal{S}) = \begin{cases} \varnothing \text{ if } eval(q_\sigma, \mathcal{S}) = \emptyset \\ q \text{ otherwise} \end{cases} \quad (1)$$

  where $q_\sigma$ is obtained from $q$ by replacing its constants by their images through $\sigma$.

- The optimization of a conjunction of subqueries $\bigwedge_{i=1}^{n} q_i$ is:

$$\Omega(\bigwedge_{i=1}^{n} q_i, \mathcal{S}) = \begin{cases} \varnothing \text{ if } \exists i \in [1, n]\ \Omega(q_i, \mathcal{S}) = \varnothing \\ \bigwedge_{i=1}^{n} \Omega(q_i, \mathcal{S}) \text{ otherwise} \end{cases} \quad (2)$$

- The optimization of a disjunction of subqueries $\bigvee_{i=1}^{n} q_i$ is:

$$\Omega(\bigvee_{i=1}^{n} q_i, \mathcal{S}) = \begin{cases} \varnothing \text{ if } \forall i \in [1, n]\ \Omega(q_i, \mathcal{S}) = \varnothing \\ \bigvee_{1 \leq i \leq n,\ \Omega(q_i, \mathcal{S}) \neq \varnothing} \Omega(q_i, \mathcal{S}) \text{ otherwise} \end{cases} \quad (3)$$

# Data-dependent FO-rewriting by example

## Running example (cont.)

The optimization of $q^{\mathcal{O}}_{\mathrm{UCQ}}$ with

$$\mathcal{S} = \{R(\mathbf{r}), sup(\mathbf{r}, \mathbf{p}), PhD(\mathbf{p}), ww(\mathbf{r}, \mathbf{r}), ww(\mathbf{r}, \mathbf{p})\}$$

and the database-to-summary homomorphism

$$\sigma(\mathbf{f}) = \sigma(\mathbf{h}) = \sigma(u) = \mathbf{r} \text{ and } \sigma(\mathbf{w}) = \sigma(\mathbf{c}) = \mathbf{p}$$

is $q^{\mathcal{K}}_{\mathrm{UCQ}}$ such that:

$$\begin{aligned}
q^{\mathcal{O}}_{\mathrm{UCQ}}(x) = & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)) && (1)\\
& \vee (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)) && (2)\\
& \vee (R(\mathbf{h}) \wedge sup(\mathbf{h}, x)) && (3)\\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge sup(y, x)) && (4)\\
& \vee (R(\mathbf{h}) \wedge ww(x, \mathbf{h}) \wedge PhD(x)) && (5)\\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge sup(y, x)) && (6)\\
& \vee (R(\mathbf{h}) \wedge sup(x, \mathbf{h}) \wedge PhD(x)) && (7)
\end{aligned}$$

$$\begin{aligned}
q^{\mathcal{K}}_{\mathrm{UCQ}} = \Omega(q^{\mathcal{O}}_{\mathrm{UCQ}}, \mathcal{S}) = & (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge sup(y, x)) && (1)\\
& \vee (R(\mathbf{h}) \wedge ww(\mathbf{h}, x) \wedge PhD(x)) && (2)\\
& \vee (R(\mathbf{h}) \wedge sup(\mathbf{h}, x)) && (3)
\end{aligned}$$

# Data-dependent FO-rewriting by example

## Running example (cont.)

The optimization of $q^{\mathcal{O}}_{\mathrm{USCQ}}$ with

$$\mathcal{S} = \{R(\mathbf{r}), sup(\mathbf{r}, \mathbf{p}), PhD(\mathbf{p}), ww(\mathbf{r}, \mathbf{r}), ww(\mathbf{r}, \mathbf{p})\}$$

and the database-to-summary homomorphism

$$\sigma(\mathbf{f}) = \sigma(\mathbf{h}) = \sigma(u) = \mathbf{r} \text{ and } \sigma(\mathbf{w}) = \sigma(\mathbf{c}) = \mathbf{p}$$

is $q^{\mathcal{K}}_{\mathrm{USCQ}}$ such that:

$$q^{\mathcal{O}}_{\mathrm{USCQ}}(x) = (R(\mathbf{h})) \wedge (ww(\mathbf{h}, x) \vee sup(\mathbf{h}, x) \vee\ ww(x, \mathbf{h}) \vee sup(x, \mathbf{h})) \wedge (sup(y, x) \vee PhD(x))$$

$$q^{\mathcal{K}}_{\mathrm{USCQ}} = \Omega(q^{\mathcal{O}}_{\mathrm{USCQ}}, \mathcal{S}) = (R(\mathbf{h})) \wedge (ww(\mathbf{h}, x) \vee sup(\mathbf{h}, x) \vee ww(x, \mathbf{h})) \wedge (sup(y, x) \vee PhD(x))$$

# Data-dependent FO-rewriting by example

## Running example (cont.)

The optimization of $q^{\mathcal{O}}_{\text{JUCQ}}$ with

$$\mathcal{S} = \{R(\mathbf{r}), sup(\mathbf{r}, \mathbf{p}), PhD(\mathbf{p}), ww(\mathbf{r}, \mathbf{r}), ww(\mathbf{r}, \mathbf{p})\}$$

and the database-to-summary homomorphism

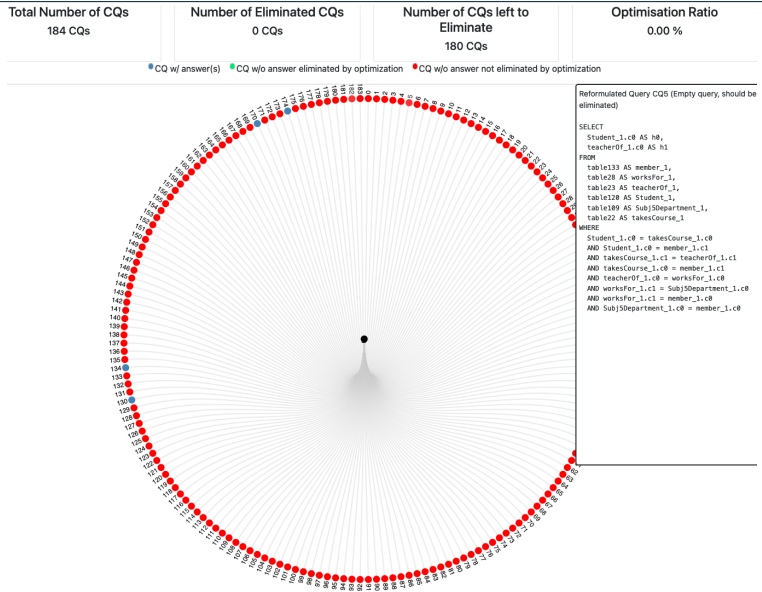
$$\sigma(\mathbf{f}) = \sigma(\mathbf{h}) = \sigma(u) = \mathbf{r} \text{ and } \sigma(\mathbf{w}) = \sigma(\mathbf{c}) = \mathbf{p}$$

is $q^{\mathcal{K}}_{\text{JUCQ}}$ such that:

$$\begin{aligned} q^{\mathcal{O}}_{\text{JUCQ}}(x) = (R(\mathbf{h})) \wedge \big( & (ww(\mathbf{h}, x) \wedge sup(y, x)) \\ & \vee (ww(\mathbf{h}, x) \wedge PhD(x)) \\ & \vee (sup(\mathbf{h}, x)) \\ & \vee (ww(x, \mathbf{h}) \wedge sup(y, x)) \\ & \vee (ww(x, \mathbf{h}) \wedge PhD(x)) \\ & \vee (sup(x, \mathbf{h}) \wedge sup(y, x)) \\ & \vee (sup(x, \mathbf{h}) \wedge PhD(x))\big) \end{aligned}$$

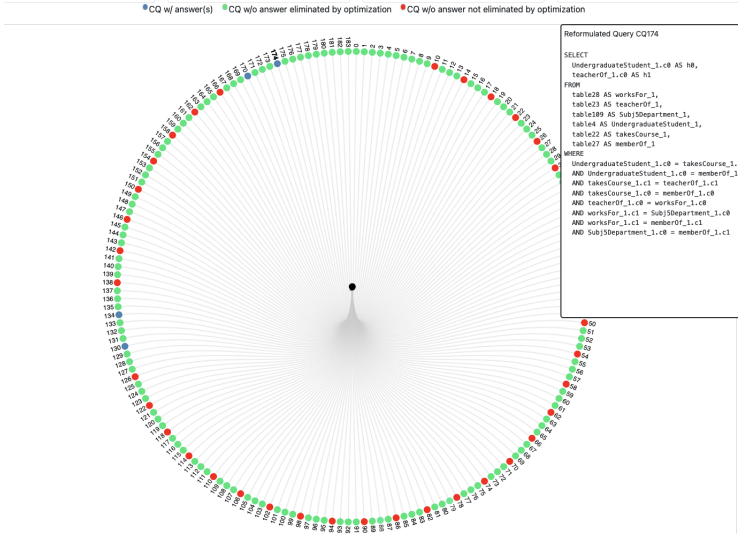
$$\begin{aligned} q^{\mathcal{K}}_{\text{JUCQ}}(x) = \Omega(q^{\mathcal{O}}_{\text{JUCQ}}, \mathcal{S}) = (R(\mathbf{h})) \wedge \big( & (ww(\mathbf{h}, x) \wedge sup(y, x)) \\ & \vee (ww(\mathbf{h}, x) \wedge PhD(x)) \\ & \vee (sup(\mathbf{h}, x))\big) \end{aligned}$$

# Sample experimental results

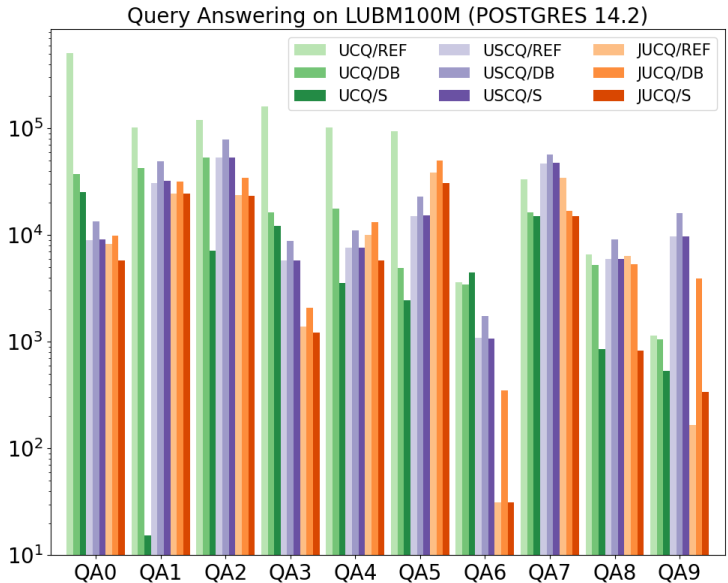


# Sample experimental results

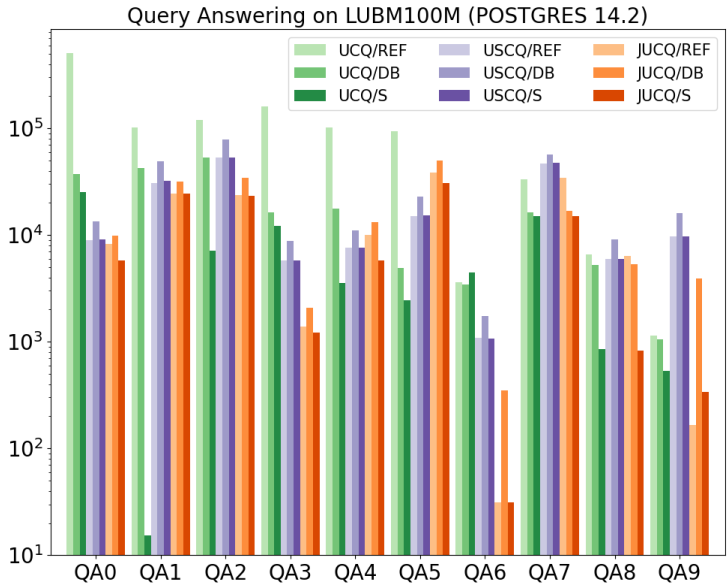
| Total Number of CQs | Number of Eliminated CQs | Number of CQs left to Eliminate | Optimisation Ratio |
|---|---|---|---|
| 184 CQs | 142 CQs | 38 CQs | 78.89 % |



# Sample experimental results

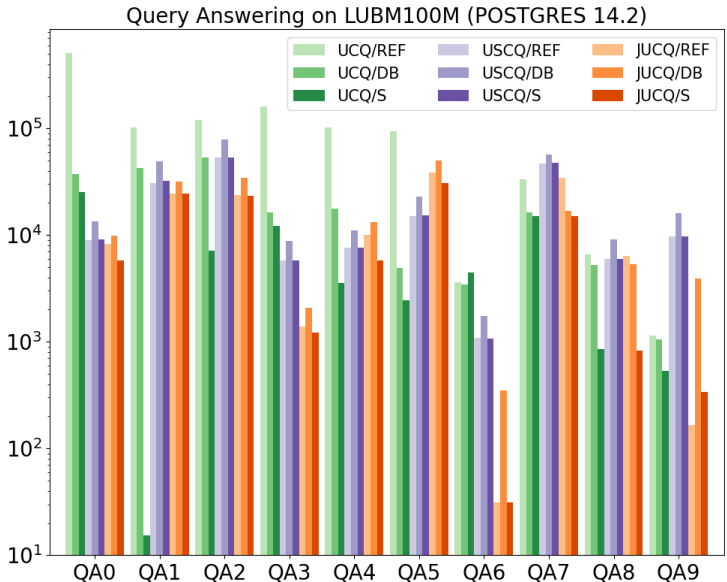


# Sample experimental results



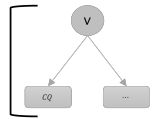

Optimization scope for UCQ/JUCQ/USCQ:

# Sample experimental results

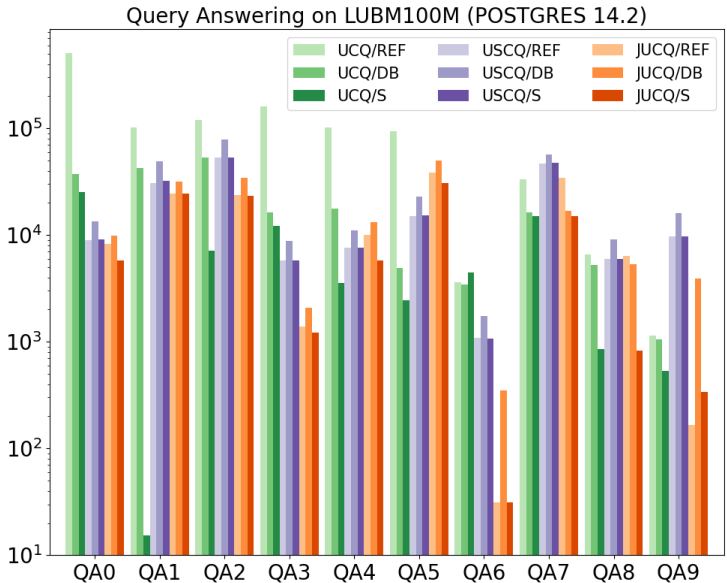



Optimization scope for UCQ/JUCQ/USCQ:

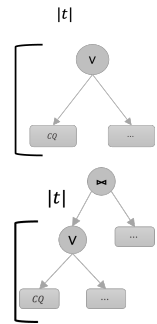
$|t|$



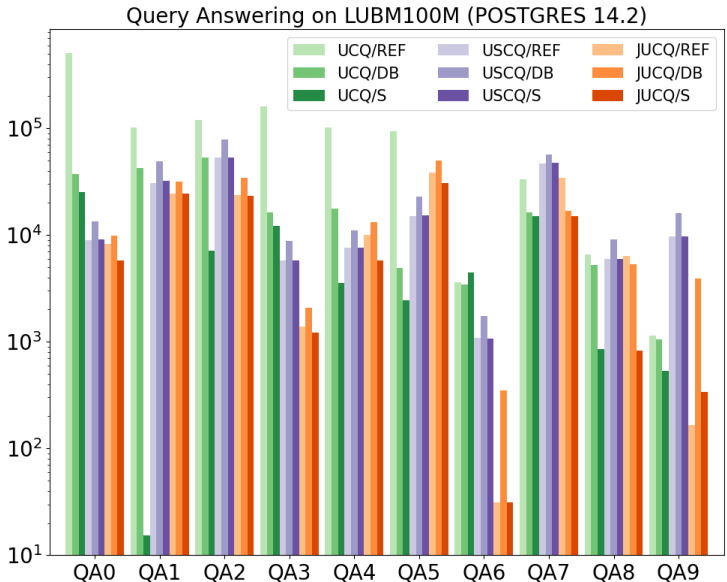
# Sample experimental results



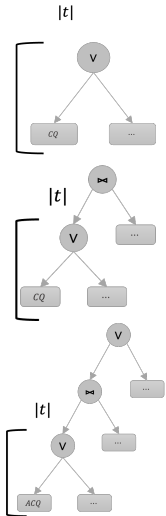

Optimization scope for UCQ/JUCQ/USCQ:



# Sample experimental results




Optimization scope for UCQ/JUCQ/USCQ:



# Take home message

1. FO-rewriting produces **complex rewritings w.r.t. OMQA on a fixed database**
2. OMQA via FO-rewriting can be **optimized for a database with a summary of it**
3. The summary-based optimization OMQA via FO-rewriting **significantly improves performance for UCQ and JUCQ rewritings**

# Related work and contributions

Some references from the literature on devising OMQA via FO-rewriting for CQs:

| | **Query rewriting language** | | | |
|---|---|---|---|---|
| **Ontology language** | UCQ | USCQ | JUCQ | datalog$^{nr}$ |
| datalog$\pm$/existential rules | [GOP11, GOP14, KLMT15]**[HVGJ24]** | [Tho13]**[HVGJ24]** | **[HVGJ24]** | [OP11, GS12] |
| description logics/OWL | [CDL$^{+}$07, PHM09, CTS11, VSS12] | | [BGM16] | [RA10] |
| RDFS | [GMR13, BGMM19] | | [BGM15] | |

# Conclusion and perspectives

# Main perspectives

1. FO-rewriting techniques that produce **directly** rewritings optimized for a database
   - with Maxime Buron (Clermont) and Cheikh-Brahim El Vaigh (Dijon)
2. Data-dependent FO-rewriting in OMQA settings that **are not FO-rewritable**
   - with Maxime Buron & Farouk Toumani (Clermont), Hélène Jaudoin (Rennes), and Quentin Manière & Federico Ulliana (Montpellier)

# References I



Damian Bursztyn, François Goasdoué, and Ioana Manolescu.
Optimizing reformulation-based query answering in RDF.
In *EDBT*, pages 265–276, 2015.


Damian Bursztyn, François Goasdoué, and Ioana Manolescu.
Teaching an RDBMS about ontological constraints.
*PVLDB*, pages 1161–1172, 2016.


Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier.
Reformulation-based query answering for RDF graphs with RDFS ontologies.
In *ESWC*, volume 11503, pages 19–35. Springer, 2019.


Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati.
Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family.
*J. Autom. Reasoning*, pages 385–429, 2007.

# References II



Alexandros Chortaras, Despoina Trivela, and Giorgos Stamou.
Optimized query rewriting for OWL2QL.
In *CADE*, pages 192–206, 2011.


François Goasdoué, Ioana Manolescu, and Alexandra Roatis.
Efficient query answering against dynamic RDF databases.
In *EDBT*, pages 299–310, 2013.


Georg Gottlob, Giorgio Orsi, and Andreas Pieris.
Ontological queries: Rewriting and optimization.
In *ICDE*, pages 2–13, 2011.


Georg Gottlob, Giorgio Orsi, and Andreas Pieris.
Query rewriting and optimization for ontological databases.
*ACM Trans. Database Syst.*, 39(3):25:1–25:46, 2014.

# References III



Georg Gottlob and Thomas Schwentick.
Rewriting ontological queries into small nonrecursive datalog programs.
In *KR*, 2012.


Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin.
Ontology-mediated query answering: performance challenges and advances.
In *ISWC*, 2021.


Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin.
Optiref: Query optimization for knowledge bases.
In *WWW*, pages 180–183, 2023.


Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin.
Query optimization for ontology-mediated query answering.
In Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee, editors, *WWW*, pages 2138–2148. ACM, 2024.

# References IV



Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo.
Sound, complete and minimal ucq-rewriting for existential rules.
*Semantic Web*, pages 451–475, 2015.


Giorgio Orsi and Andreas Pieris.
Optimizing query answering under ontological constraints.
*PVLDB*, 4(11):1004–1015, 2011.


Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik.
Efficient query answering for OWL 2.
In *ISWC*, pages 489–504, 2009.


Riccardo Rosati and Alessandro Almatelli.
Improving query answering over dl-lite ontologies.
In *KR*, 2010.

# References V


Michaël Thomazo.
Compact rewritings for existential rules.
In *IJCAI*, pages 1125–1131, 2013.

Tassos Venetis, Giorgos Stoilos, and Giorgos B. Stamou.
Incremental query rewriting for OWL 2 QL.
In *DL*, 2012.