OptiRef: Query Optimization for Knowledge Bases

Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, Hélène Jaudoin

OptiRef: Query Optimization for Knowledge Bases

Wafaa El Husseini
Univ. Rennes, CNRS, IRISA
Lannion, France
wafaa.el-husseini@irisa.fr

Cheikh-Brahim El Vaigh
Univ. Bourgogne
Dijon, France
cheikh-brahim.el-vaigh@u-bourgogne.fr

François Goasdoué
Univ. Rennes, CNRS, IRISA
Lannion, France
fg@irisa.fr

Hélène Jaudoin
Univ. Rennes, CNRS, IRISA
Lannion, France
helene.jaudoin@irisa.fr

ABSTRACT

Ontology-mediated query answering (OMQA) consists in asking database queries on a knowledge base (KB); a KB is a set of facts, the KB's database, described by domain knowledge, the KB's ontology.

FOL-rewritability is the main OMQA technique: it reformulates a query w.r.t. the KB's ontology so that the evaluation of the reformulated query on the KB's database computes the correct answers. However, because this technique embeds the domain knowledge relevant to the query into the reformulated query, a reformulated query may be complex and its optimization is the crux of efficiency.

We showcase OptiRef that implements a novel, general optimization framework for efficient query answering on datalog±, description logic, existential rules, OWL and RDF/S KBs. OptiRef optimizes reformulated queries by rapidly computing, based on a KB's database summary, simpler (contained) queries with the same answers. We demonstrate OptiRef's effectiveness on well-established benchmarks: performance is significantly improved in general, up to several orders of magnitude in the best cases!

CCS CONCEPTS

• **Information systems** → **Query optimization**; *Semantic web description languages*; • **Computing methodologies** → *Knowledge representation and reasoning*.

KEYWORDS

Existential rules, query optimization, data summarization

ACM Reference Format:
Wafaa El Husseini, Cheikh-Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin. 2023. OptiRef: Query Optimization for Knowledge Bases. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3543873.3587342

1 MOTIVATIONS

Ontology-mediated query answering (OMQA), which consists in answering database queries on knowledge bases (KBs), has received considerable attention in the *database*, *knowledge representation* and *semantic web* fields [1]. OMQA is non-trivial because answers must be computed from the explicit data *stored* in the KB's database plus the implicit data *logically entailed* from the KB's database using the KB's ontology, i.e., the domain knowledge.

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	$q(\bar{x}) = \bigwedge_{i=1}^{n} UCQ_i$	$q(\bar{x}) = \Pi_{\bar{x}}(\bowtie_{i=1}^{n} UCQ_i)$
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	$q(\bar{x}) = \bigvee_{i=1}^{n} SCQ_i$	$q(\bar{x}) = \bigcup_{i=1}^{n} SCQ_i$

Table 1: Languages used for FOL-rewritability

	Query reformulation language			
KB language	UCQ	USCQ	JUCQ	datalog^{nr}
datalog±/existential rules	[10, 11, 14]	[22]		[12, 19]
description logics/OWL2	[6, 8, 20, 23]		[5]	[21]
RDF/S	[2, 9]		[4]	

Table 2: Main related works on FOL-rewritability for CQs

FOL-rewritability is the widely-investigated OMQA technique introduced in [6]. It reduces query answering on KBs to *query answering on databases* provided by DBMSs: a query q on a KB is reformulated w.r.t. the KB's ontology $\mathcal{O}$ into a *query reformulation* $q^{\mathcal{O}}$, in which the knowledge from $\mathcal{O}$ relevant to q is embedded, then the answers to q are obtained by *evaluating* $q^{\mathcal{O}}$ on the KB's database with a DBMS. FOL-rewritability has been studied for queries expressed as (union of) *conjunctive queries (CQs)*; KBs expressed using *datalog*± and *existential rules*, *description logics* and *OWL2*, and *RDF/S*; query reformulations expressed as *unions of CQs (UCQs)* and *non-recursive datalog programs (datalog^{nr})* that unfold to UCQs, *unions of semi-CQs (USCQs)*, and *joins of UCQs (JUCQs)*. Table 1 recalls these languages. Main related works are referenced in Table 2.

We argue that the crux of efficient FOL-rewritability is the *optimization of query reformulations*: a query reformulation may be complex and costly to evaluate [5, 22] because it logically models *all* the – worst-case exponentially many – specializations (i.e., particular cases) of the query w.r.t. the ontology of the queried KB [6]. We therefore showcase OptiRef that implements a *novel optimization framework* for FOL-rewritability. It applies to *all* the works on FOL-rewritability in the literature (e.g., Table 2) by considering *expressive* KBs expressed using existential or datalog± rules. It optimizes query reformulations using a KB's database summary into simpler (contained) queries with the same answers. We demonstrate on well-established benchmarks for DL-lite$_{\mathcal{R}}$ KBs, how OptiRef *significantly* improves query answering performance in general, up to *several* orders of magnitude. DL-lite$_{\mathcal{R}}$ is a practical fragment of datalog± and existential rules; it is the description logic that underpins the W3C's *OWL2 QL profile* for OMQA on large KBs.

This paper is an in-depth study and generalization of preliminary ideas sketched in [3, 13].

WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA

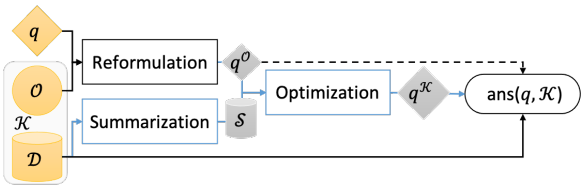

Figure 1: Standard and revisited FOL-rewritability

2 ANSWERING QUERIES ON A KB

KBs. We consider KBs expressed with datalog± or existential rules. A KB is of the form $\mathcal{K} = (\mathcal{O}, \mathcal{D})$, where $\mathcal{O}$ is the KB's ontology and $\mathcal{D}$ is the KB's database. An ontology is a set of *deductive rules* of the form $\forall \bar{x}_1 (q_1(\bar{x}_1) \rightarrow q_2(\bar{x}_2))$, where q_1, q_2 are *conjunctive queries* (CQs, recall Table 1) with sets of answer variables $\bar{x}_1, \bar{x}_2$ respectively, such that $\bar{x}_2 \subseteq \bar{x}_1$ and $\bar{x}_1$ is exactly the set of variables that occur in q_1. A database is a set of *incomplete facts*, i.e., within which *existential variables* model missing or unknown values.

Notation. We use small letters for constants, e.g., f, h, etc. and small italic letters for variables, e.g., x, y, etc. Also we omit existential quantifiers in rules: existential variables are those that solely appear in the right-hand side of $\rightarrow$.

Example 2.1 (Running example). Let us consider the following DL-lite$_{\mathcal{R}}$ KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$:

$\mathcal{O} = \{ r_1 : ww(x, y) \rightarrow ww(y, x), r_2 : sup(x, y) \rightarrow ww(x, y),$
$r_3 : PhD(x) \rightarrow sup(y, x)\},$
$\mathcal{D} = \{R(\mathsf{f}), R(\mathsf{h}), sup(\mathsf{f}, \mathsf{w}), sup(\mathsf{h}, \mathsf{w}), PhD(\mathsf{w}), ww(\mathsf{f}, \mathsf{h})\}.$

The ontology $\mathcal{O}$ states: working with (ww) someone is a symmetric relation (r_1), supervising someone (sup) is a particular case of working with someone (r_2), and PhD students (PhD) are necessarily supervised (r_3). The database $\mathcal{D}$ states: François (f) and Hélène (h), who are researchers (R), supervise Wafaa (w), who is a PhD student. Additionally, François works with Hélène.

Query answering. We consider queries from the languages shown in Table 1. The arity of a query q is the cardinality of its set $\bar{x}$ of free or answer variables. A *certain answer to a query* $q(\bar{x})$ of arity n on a KB $\mathcal{K}$ is a tuple of n constants $\bar{\mathsf{t}}$ such that $\mathcal{K} \models q(\bar{\mathsf{t}})$, where $q(\bar{\mathsf{t}})$ is the Boolean query obtained by instantiating $\bar{x}$ with $\bar{\mathsf{t}}$ in q. From now, we note $ans(q, \mathcal{K})$ the answer set of q on $\mathcal{K}$.

Example 2.2 (Cont.). Let us consider the CQ asking for the supervisees who work with the researcher h: $q(x) = \exists y\, R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge sup(y, x)$. Its answer set on $\mathcal{K}$ is $ans(q, \mathcal{K}) = \{\mathsf{w}\}$: w can be obtained from $R(\mathsf{h}) \in \mathcal{D}$, $sup(\mathsf{f}, \mathsf{w}) \in \mathcal{D}$ and the entailed fact $ww(\mathsf{h}, \mathsf{w})$ deduced from $sup(\mathsf{h}, \mathsf{w}) \in \mathcal{D}$ and r_2.

Notation. We omit the existential quantifiers to simplify the notation of queries, because non-answer variables are necessarily existentially quantified in the query languages we consider (recall Table 1). For instance, the CQ $q(x) = \exists y\, R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge sup(y, x)$ of Example 2.2 is now written $q(x) = R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge sup(y, x)$.

FOL-rewritability. The main technique to perform query answering on KBs is *FOL-rewritability*. This technique is illustrated in Figure 1 using the black solid and dashed edges. The query q is first reformulated based on the ontology $\mathcal{O}$ into the query $q^{\mathcal{O}}$ called a *reformulation of q w.r.t. $\mathcal{O}$*, so that the relational evaluation of $q^{\mathcal{O}}$ on the database $\mathcal{D}$, denoted by $eval(q^{\mathcal{O}}, \mathcal{D})$, provides the correct answer set of q on $\mathcal{K}$, i.e., $eval(q^{\mathcal{O}}, \mathcal{D}) = ans(q, \mathcal{K})$.

The formal properties of FOL-rewritability, which motivated the optimizations we devised and we implemented in OptiRef, are [6]: (i) *$q^{\mathcal{O}}$ is equivalent to q w.r.t. $\mathcal{O}$* and *encodes **all** the – worst-case exponentially many – non-redundant queries contained in q w.r.t. $\mathcal{O}$* and (ii) *for **any** database $\mathcal{D}'$, $eval(q^{\mathcal{O}}, \mathcal{D}') = ans(q, (\mathcal{O}, \mathcal{D}'))$ holds.*

FOL-rewritability has been studied in the literature for triples of query, ontology and query reformulation languages (recall Table 1), which we term *OMQA settings*.

Example 2.3 (Cont.). The next UCQ q^{UCQ}, USCQ q^{USCQ} and JUCQ q^{JUCQ} are the reformulations of q w.r.t. $\mathcal{O}$ respectively computed by the Rapid [8], Compact [22] and GDL [5] tools.

$$
\begin{aligned}
q^{\text{UCQ}}(x) = {} & (R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge sup(y, x)) && (1)\\
& \vee (R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge PhD(x)) && (2)\\
& \vee (\boldsymbol{R(\mathsf{h}) \wedge sup(\mathsf{h}, x)}) && \textbf{(3)}\\
& \vee (R(\mathsf{h}) \wedge ww(x, \mathsf{h}) \wedge sup(y, x)) && (4)\\
& \vee (R(\mathsf{h}) \wedge ww(x, \mathsf{h}) \wedge PhD(x)) && (5)\\
& \vee (R(\mathsf{h}) \wedge sup(x, \mathsf{h}) \wedge sup(y, x)) && (6)\\
& \vee (R(\mathsf{h}) \wedge sup(x, \mathsf{h}) \wedge PhD(x)) && (7)
\end{aligned}
$$

$$
\begin{aligned}
q^{\text{USCQ}}(x) = {} & (\boldsymbol{R(\mathsf{h})})\\
& \wedge (ww(\mathsf{h}, x) \vee \boldsymbol{sup(\mathsf{h}, x)} \vee ww(x, \mathsf{h}) \vee sup(x, \mathsf{h}))\\
& \wedge (\boldsymbol{sup(y, x)} \vee PhD(x))
\end{aligned}
$$

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

q^{UCQ} is the union of all the non-redundant CQs contained in q w.r.t. $\mathcal{O}$. q^{USCQ} and q^{JUCQ} encode the same CQs up to the distributive property of $\wedge$ and $\vee$. The only answer to q on $\mathcal{K}$ (i.e., w) results from **(3)** in q^{UCQ}, shown in **bold**, as well as from the logical combination of the subqueries shown in **bold** in q^{USCQ} and q^{JUCQ} (from which **(3)** can be recovered by distributing the $\wedge$'s over the $\vee$'s).

Problem statement. We focus on the problem of optimizing FOL-rewritability in ***all*** the OMQA settings, ***at once***.

3 REVISING FOL-REWRITABILITY

FOL-rewritability under scrutiny. FOL-rewritability is by definition *truly independent* of the queried data: a query q is reformulated into a query $q^{\mathcal{O}}$ such that for ***any*** database $\mathcal{D}$, $ans(q, (\mathcal{O}, \mathcal{D}))$ is obtained by the evaluation of $q^{\mathcal{O}}$ on $\mathcal{D}$. However, when answering a query q on a *particular KB* $\mathcal{K} = (\mathcal{O}, \mathcal{D})$, the database $\mathcal{D}$ at hand is *just **one of all*** the possible databases the reformulation $q^{\mathcal{O}}$ accommodates to. We argue that this *innate universality* of $q^{\mathcal{O}}$ limits the performance of its evaluation by a DBMS on a *fixed* database $\mathcal{D}$: some queries contained in q w.r.t. $\mathcal{O}$ that $q^{\mathcal{O}}$ encode may be ***irrelevant to*** $\mathcal{D}$, i.e., have no answer on $\mathcal{D}$, while *evaluating them may take significant time.* In Example 2.3, all the CQs in q^{UCQ} but **(3)** are irrelevant to the database $\mathcal{D}$, as well as in q^{USCQ} and q^{JUCQ}, where they are encoded up to the distributive property of $\wedge$ and $\vee$.

Towards more efficient FOL-rewritability. We therefore revise the definition of FOL-rewritability so that a query is not just reformulated w.r.t. the KB's ontology, but *also w.r.t. the KB's database*, i.e., w.r.t. the *whole KB*. FOL-rewritability applies to an OMQA setting iff for any query q and any KB $\mathcal{K} = (\mathcal{O}, \mathcal{D})$, there exists a query reformulation $q^{\mathcal{K}}$ of q w.r.t. $\mathcal{K}$ such that (i) $q^{\mathcal{K}}$ is *contained in q*

w.r.t. $\mathcal{O}$ and (ii) $eval(q^{\mathcal{K}}, \mathcal{D}) = ans(q, \mathcal{K})$ holds. We note that when $q^{\mathcal{K}}$ is *equivalent to* q w.r.t. $\mathcal{O}$ (i.e., to $q^{\mathcal{O}}$), a particular case of containment, the original and revised definitions of FOL-rewritability *coincide*. Our new definition essentially trades the universality of a query reformulation for its specificity to, we say its *optimization for*, a fixed database. It enables $q^{\mathcal{K}}$ to model ***not all*** the – worst-case exponentially many – queries contained in q w.r.t. $\mathcal{O}$ (condition (i)), but ***at least*** those that contribute to the answer set of q on $\mathcal{K}$ (condition (ii)). In Example 2.3, removing from $q^{\text{UCQ}}, q^{\text{USCQ}}, q^{\text{JUCQ}}$ none, some or all of the irrelevant subqueries make them reformulations of q w.r.t. $\mathcal{K}$; the more subqueries irrelevant to $\mathcal{D}$ are removed, the more $q^{\text{UCQ}}, q^{\text{USCQ}}, q^{\text{JUCQ}}$ are *optimized for* $\mathcal{D}$. To distinguish FOL-rewritability that only focuses on the *ontology* at hand, from its revision that focuses on the *whole KB* at hand, we term the former *O-FOL-rewritability* and the latter *KB-FOL-rewritability*.

4 OPTIMIZING QUERY REFORMULATIONS

OptiRef **system.** We devised the OptiRef system to turn ***all*** the query reformulation algorithms for O-FOL-rewritability from ***all*** the currently known OMQA settings into query reformulation algorithms for KB-FOL-rewritability. The technique it implements for this *O- to KB-FOL-rewritability* is illustrated in Figure 1 using the black and blue solid edges. Similarly to O-FOL-rewritability, the query q is first reformulated w.r.t. the ontology $\mathcal{O}$ into the reformulation $q^{\mathcal{O}}$ of q w.r.t. $\mathcal{O}$. This Reformulation step is achieved by using ***any*** off-the-shelf query reformulation algorithm for O-FOL-rewritability. Then, $q^{\mathcal{O}}$ is optimized for the database $\mathcal{D}$ to obtain the reformulation $q^{\mathcal{K}}$ of q w.r.t. $\mathcal{K} = (\mathcal{O}, \mathcal{D})$. This Optimisation step (discussed below) is achieved by *rewriting* $q^{\mathcal{O}}$, which is equivalent to q w.r.t. $\mathcal{O}$, into $q^{\mathcal{K}}$, which is *contained in* q w.r.t. $\mathcal{O}$, while *retaining correctness of OMQA*. $q^{\mathcal{K}}$ is simpler, hence faster to evaluate, than $q^{\mathcal{O}}$: $q^{\mathcal{K}}$ is obtained by removing subqueries *irrelevant* to $\mathcal{D}$ in $q^{\mathcal{O}}$. This optimization step uses a *summary* $\mathcal{S}$ of $\mathcal{D}$; $\mathcal{S}$ results from a Summarization step (discussed below). It is computed *offline upon database loading* and *maintained upon database updates*. Using $\mathcal{S}$ instead of $\mathcal{D}$ achieves a good trade-off between the number of identified subqueries irrelevant to $\mathcal{D}$ and the optimization time.

Summarization **step (Fig. 1).** We consider *quotient databases* as summaries, which we obtain by adapting the *quotient operation* from graph theory to our databases. This operation has been widely used for graph database summarization [7, 16]. It basically fuses *equivalent* nodes, database values in our case (i.e., constants and variables), according to an *equivalence relation* $\equiv$. Quotient databases are *homomorphic approximations* of databases, hence enjoy an essential property for identifying CQs with no answer on a database. This property directly follows from the existence of a database-to-summary homomorphism: *if q_σ has no answer on $\mathcal{S}$, then q has no answer on $\mathcal{D}$, where q_σ is obtained from q by replacing its constants by their images through the $\mathcal{D}$-to-$\mathcal{S}$ homomorphism σ*.

Because ontology languages are centered on *concepts* modeled by unary relations, which are interrelated using *relationships* modeled by n-ary relations, our summarization is *centered on the instances of concepts* stored in the database $\mathcal{D}$. Our equivalence relation $\equiv_\Omega$ is such that (i) all the values that are instances of a same concept in $\mathcal{D}$ are represented by a *single value* in $\mathcal{S}$ and, to preserve *joins* within the data, (ii) for concepts whose instances share some value(s) in $\mathcal{D}$, all their values are represented by a *same single value* in $\mathcal{S}$.

Example 4.1 (Cont.). $\equiv_\Omega$ defines two equivalence classes of values in $\mathcal{D}$: $\{\mathsf{f}, \mathsf{h}\}$ for the researchers and $\{\mathsf{w}\}$ for the PhD students. These two classes are respectively represented by r and p in the summary $\mathcal{S} = \{R(\mathsf{r}), sup(\mathsf{r}, \mathsf{p}), PhD(\mathsf{p}), ww(\mathsf{r}, \mathsf{r}), ww(\mathsf{r}, \mathsf{p})\}$ of $\mathcal{D}$, with the $\mathcal{D}$-to-$\mathcal{S}$ homomorphism σ such that $\sigma(\mathsf{w}) = \mathsf{p}$, $\sigma(\mathsf{f}) = \sigma(\mathsf{h}) = \mathsf{r}$.

Optimization **step (Fig. 1).** We devised a *recursive optimization function* Ω that rewrites $q^{\mathcal{O}}$ into $q^{\mathcal{K}}$. Ω identifies CQs with no answer on $\mathcal{D}$ using $\mathcal{S}$ and performs a bottom-up propagation of their removal within the query reformulation. Ω's base case is for CQs: $\Omega(q, \mathcal{S})$ is $\varnothing$ if q_σ has no answer on $\mathcal{S}$ and is q itself otherwise, where $\varnothing$ is the empty relation and σ is the $\mathcal{D}$-to-$\mathcal{S}$ homomorphism as seen above. Ω's first recursive case is for $\wedge$: $\Omega(\wedge_{i=1}^{n} q_i, \mathcal{S})$ is $\varnothing$ if some $\Omega(q_i, \mathcal{S}) = \varnothing$ and is $\wedge_{i=1}^{n} \Omega(q_i, \mathcal{S})$ otherwise. Ω's second recursive case is for $\vee$: $\Omega(\vee_{i=1}^{n} q_i, \mathcal{S})$ is $\varnothing$ if $\Omega(q_i, \mathcal{S}) = \varnothing$ for each i and is $\vee_{i=1, \Omega(q_i, \mathcal{S}) \neq \varnothing}^{n} \Omega(q_i, \mathcal{S})$ otherwise.

Crucially, $q^{\mathcal{K}}$ is a *reformulation of q w.r.t. $\mathcal{K}$* because $q^{\mathcal{K}} = \Omega(q^{\mathcal{O}}, \mathcal{S})$ and each subquery in $q^{\mathcal{O}}$ is recursively rewritten by Ω into *a contained one with same answer set on $\mathcal{D}$*, hence ***$q^{\mathcal{K}}$ is contained in q w.r.t. $\mathcal{O}$*** and ***$eval(q^{\mathcal{K}}, \mathcal{D}) = ans(q, \mathcal{K})$ holds***.

Example 4.2 (Cont.). The optimization of $q^{\text{UCQ}}, q^{\text{USCQ}}, q^{\text{JUCQ}}$ for $\mathcal{D}$ using $\mathcal{S}$, which preserves the answer set, produces:

$$\begin{aligned}
\Omega(q^{\text{UCQ}}, \mathcal{S}) = q^{\text{UCQ}}_{\mathcal{S}}(x) \quad &= (R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge sup(y, x)) \\
&\vee (R(\mathsf{h}) \wedge ww(\mathsf{h}, x) \wedge PhD(x)) \\
&\vee (\boldsymbol{R(\mathsf{h}) \wedge sup(\mathsf{h}, x)}) \\
\Omega(q^{\text{USCQ}}, \mathcal{S}) = q^{\text{USCQ}}_{\mathcal{S}}(x) &= (\boldsymbol{R(\mathsf{h})}) \\
&\wedge (ww(\mathsf{h}, x) \vee \boldsymbol{sup(\mathsf{h}, x)} \vee \boldsymbol{ww}(x, \mathsf{h})) \\
&\wedge (\boldsymbol{sup(y, x)} \vee PhD(x)) \\
\Omega(q^{\text{JUCQ}}, \mathcal{S}) = q^{\text{JUCQ}}_{\mathcal{S}}(x) &= (\boldsymbol{R(\mathsf{h})}) \wedge ((ww(\mathsf{h}, x) \wedge sup(y, x)) \\
&\qquad \vee (ww(\mathsf{h}, x) \wedge PhD(x)) \\
&\qquad \vee (\boldsymbol{sup(\mathsf{h}, x)})
\end{aligned}$$

where the subqueries in gray would have been additionally removed (with typically much longer optimization time) if the reformulations had been optimized by Ω using $\mathcal{D}$ instead of $\mathcal{S}$.

5 OUTLINE OF THE DEMONSTRATION

We showcase OptiRef to highlight the time performance benefits that it brings to answering queries on KBs. We selected the practically relevant and reasonably expressive OMQA setting of CQs asked on DL-lite$_\mathcal{R}$ KBs, as it enjoys: (i) FOL-rewritability based on UCQ, USCQ and JUCQ reformulations, (ii) well-established benchmarks, notably LUBM$^\exists$ [17] and NPD [15], and (iii) the W3C's OWL2 QL profile to be used in real applications.

OptiRef is a JAVA tool that relies on the IBM DB2, MySQL and PostgreSQL DBMSs to store and query databases and their summaries, and on the Rapid [8], Compact [22] and GDL [5] JAVA tools to compute, respectively, UCQ, USCQ and JUCQ query reformulations of CQs w.r.t. DL-lite$_\mathcal{R}$ ontologies. OptiRef also features a PHP/jQuery-based GUI to offer a nice visual and interactive attendee experience (see figures below).

The demonstration first introduces O-FOL-rewritability, i.e., the state-of-the art baseline, and motivates KB-FOL-rewritability based on Figure 1 and this paper's simple examples. Then, the attendees can *interact* with OptiRef to *experiment* with O- and KB-FOL-rewritability on LUBM$^\exists$ and NPD KBs, and to *assess and understand* the time performance benefits that KB-FOL-rewritability brings.

The attendees can *select and edit predefined queries* or *write their own queries* (Fig. 2). The GUI indicates the query syntax to use and relations are chosen by browsing the LUBM$^{\exists}$ or NPD ontology with the Protégé ontology editor [18].

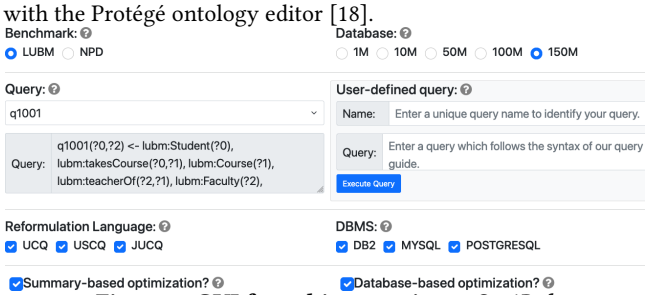

Figure 2: GUI for asking queries to OptiRef

To *assess* the performance improvements that OptiRef brings with its Ω optimization function and summaries computed as database quotients with the $\equiv_\Omega$ equivalence relation, the attendees can *compare* the total query answering times to answer a query q on a KB $\mathcal{K}$ through the reformulation q^O of q w.r.t. $\mathcal{K}$'s ontology $\mathcal{O}$, i.e., O-FOL-rewritability, and the optimization performed by Ω of q^O into a reformulation $q^{\mathcal{K}}$ of q w.r.t. $\mathcal{K}$ using either $\mathcal{K}$'s database $\mathcal{D}$ or a summary $\mathcal{S}$ of it, i.e., KB-FOL-rewritability. The attendees can also observe how these times break down into reformulation, optimization and evaluation times (Fig. 3). They will learn that, for all the DBMSs, though to different extents, using $\mathcal{D}$ may improve or worsen performance, as optimization time may or may not amortize at evaluation time, while using $\mathcal{S}$ improves performance, *significantly* in general and up to *several orders of magnitude*.

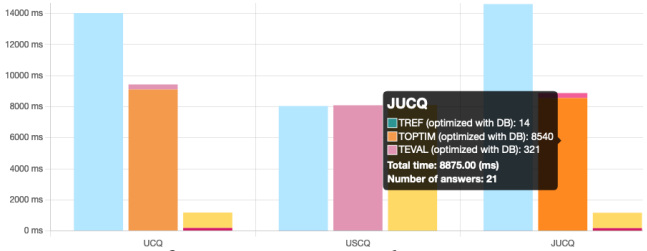

Figure 3: GUI for assessing OptiRef **query answering performance: query answering times are shown for O-FOL-rewritability (left) and KB-FOL-rewritability when using the database (center) or its summary (right)**

To *understand* the time performance improvements, the attendees can *visualize* the logical plans and SQL of the reformulations in order to observe to which extent q^O has been optimized into $q^{\mathcal{K}}$ when Ω uses $\mathcal{D}$ or $\mathcal{S}$. They can also *compare* the optimized reformulations w.r.t. the time to produce them (which depends on the DBMS at hand) and ratio of identified subqueries with no answer on $\mathcal{D}$ (Fig. 4): 100% when Ω uses $\mathcal{D}$ and 80% on average when Ω uses $\mathcal{S}$. The attendees can also *browse* OptiRef*'s statistics* to observe that the $\mathcal{D}$-to-$\mathcal{S}$ size reduction is *close to 90%* for all the databases and that summarization time is *linear in data size* though visibly different from a DBMS to another (same code, data layout, indexes, etc.). They will learn that performance improvements follows from the use of summaries that, at the same time, are *small enough* for *fast* optimization time and allow a *high* ratio of identified subqueries with no answer for *significant* query reformulation optimization.

Acknowledgements. This work was supported by ANR CQFD (ANR-18-CE23-0003) and ARED Gedeon (LTC & Région Bretagne).

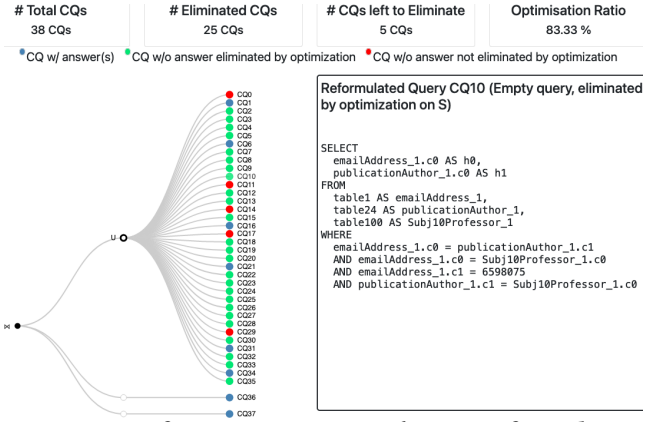

Figure 4: GUI for examining OptiRef **query reformulation optimization: the blue CQs are useful, i.e., they have answers, while the others are useless, i.e., they do not have answer; green ones are eliminated but red ones are not eliminated**

REFERENCES

[1] Meghyn Bienvenu. 2016. Ontology-Mediated Query Answering: Harnessing Knowledge to Get More from Data. In *IJCAI*.
[2] Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier. 2019. Reformulation-Based Query Answering for RDF Graphs with RDFS Ontologies. In *ESWC*.
[3] Maxime Buron, Cheikh Brahim El Vaigh, and François Goasdoué. 2021. Towards Faster Reformulation-based Query Answering on RDF graphs with RDFS ontologies. Short paper. In *ISWC*.
[4] Damian Bursztyn, François Goasdoué, and Ioana Manolescu. 2015. Optimizing Reformulation-based Query Answering in RDF. In *EDBT*.
[5] Damian Bursztyn, François Goasdoué, and Ioana Manolescu. 2016. Teaching an RDBMS about ontological constraints. *PVLDB* (2016).
[6] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *J. Autom. Reasoning* (2007).
[7] Sejla Cebiric, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. 2019. Summarizing semantic graphs: a survey. *VLDB J.* (2019).
[8] Alexandros Chortaras, Despoina Trivela, and Giorgos Stamou. 2011. Optimized Query Rewriting for OWL2QL. In *CADE*.
[9] François Goasdoué, Ioana Manolescu, and Alexandra Roatis. 2013. Efficient query answering against dynamic RDF databases. In *EDBT*.
[10] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2011. Ontological queries: Rewriting and optimization. In *ICDE*.
[11] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2014. Query Rewriting and Optimization for Ontological Databases. *ACM TODS* (2014).
[12] Georg Gottlob and Thomas Schwentick. 2012. Rewriting Ontological Queries into Small Nonrecursive Datalog Programs. In *KR*.
[13] Wafaa El Husseini, Cheikh Brahim El Vaigh, François Goasdoué, and Hélène Jaudoin. 2021. Ontology-mediated query answering: performance challenges and advances. Short paper. In *ISWC*.
[14] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. 2015. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web* (2015).
[15] Davide Lanti, Martín Rezk, Guohui Xiao, and Diego Calvanese. 2015. The NPD Benchmark: Reality Check for OBDA Systems. In *EDBT*.
[16] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* (2018).
[17] Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. 2013. The Combined Approach to OBDA: Taming Role Hierarchies Using Filters. In *ISWC*.
[18] Mark A. Musen. 2015. The protégé project: a look back and a look forward. *AI Matters* 1, 4 (2015).
[19] Giorgio Orsi and Andreas Pieris. 2011. Optimizing Query Answering under Ontological Constraints. *PVLDB* (2011).
[20] Héctor Pérez-Urbina, Ian Horrocks, and Boris Motik. 2009. Efficient Query Answering for OWL 2. In *ISWC*.
[21] Riccardo Rosati and Alessandro Almatelli. 2010. Improving Query Answering over DL-Lite Ontologies. In *KR*.
[22] Michaël Thomazo. 2013. Compact Rewritings for Existential Rules. In *IJCAI*.
[23] Tassos Venetis, Giorgos Stoilos, and Giorgos B. Stamou. 2012. Incremental Query Rewriting for OWL 2 QL. In *DL*.