Ontology-Mediated Query Answering: Performance Challenges and Advances

Wafaa El Husseini[0000−0003−2344−5769], Cheikh-Brahim El Vaigh[0000−0002−9843−3001], François Goasdoué[0000−0003−4532−7974], and Hélène Jaudoin[0000−0002−9898−5789]

Univ Rennes, Lannion, France
{wafaa.el-husseini,cheikh-brahim.el-vaigh,fg,jaudoin}@irisa.fr

Abstract. Ontology-mediated query answering (OMQA) is a recent data management trend in the Artificial Intelligence, Database and Semantic Web areas, which aims at answering database queries on knowledge bases. Because it is an intricate combination of automated reasoning and database query evaluation, it raises major performance challenges. In this demonstration, we showcase a decade of OMQA optimization to understand "Where do we stand now and how did we get there?" and we highlight a promising new OMQA optimization that brings further significant performance improvement to discuss "What's next?".

Keywords: query answering · knowledge base · summary · optimization

1 Motivations

Ontology-mediated query answering [1] (OMQA) amounts to answering database queries on a knowledge base (KB), i.e., data described by an ontology modelling the background knowledge of the application domain. It goes beyond database query answering that produces answers just from the data, by identifying additional answers through reasoning about the data with the help of the ontology. OMQA has become recently a hot topic in the Artificial Intelligence, Database and Semantic Web communities, notably for *conjunctive queries* (a.k.a. the core select-project-join database queries) on KBs expressed with *Description Logics*, on which builds the W3C's *Web Ontology Language (OWL2)*, as well as with the closely related *Datalog±* and *Existential Rules*.

Three OMQA techniques have been devised in the literature. They amount to *reducing* OMQA to standard relational database query evaluation by compiling the ontology knowledge in the query via *reformulation or rewriting* (e.g., [5,13,3]), in the data via *saturation or materialization* (e.g., [11]), or in both via *combined or hybrid approaches* (e.g., [9]). The OMQA technique based on query reformulation, on which we focus in this demonstration, is the by far the most studied

and utilized one. It was introduced in [4] and consists in reformulating every incoming conjunctive query (CQ) q into a union of conjunctive queries (UCQ) q' using the KB's ontology, so that the standard database evaluation of (the SQLized) q' on the KB's data stored in a relational database yields the correct answers to q. Crucially, such a reformulated query q' *enumerates* within its union *all* the (worst case exponentially many) specializations of q w.r.t. the ontology knowledge. In practice, reformulations may be large, in which case relational database management systems (RDBMSs), even modern ones, cannot answer them efficiently (all the CQs in the UCQ are evaluated). To mitigate this performance issue, investigations have focused on alternative reformulation languages that may be easier to evaluate than large UCQs, by limiting redundancies across CQs within UCQs: *minimal* UCQs [5] and *unions of semi-CQs* [13] (USCQs). However, such OMQA approaches produce a single reformulation for every query, thus are doomed to fail or to poor performance if the produced reformulation is too costly to evaluate by the RDBMS at hand. Based on this observation, the language of *joins of UCQs* (JUCQs) was considered to explore a space of semantically equivalent but syntactically different reformulations, among which one with lowest estimated evaluation cost on the RDBMS is chosen [3,2].

With this demonstration, our ambition is twofold. First, we want to showcase a decade of OMQA optimization to understand *"Where do we stand now and how did we get there?"* For that, we *compare and analyze* the performance improvement that the literature has brought over the years with the state-of-the-art UCQ, then USCQ and finally JUCQ reformulations. Second, we want to discuss *"What's next?"* by highlighting a novel advance on OMQA optimization that we propose, and which allows improving significantly the performance of UCQ, USCQ *and* JUCQ reformulations using a summary of the queried data [7], up to several orders of magnitude in some cases! To substantiate the discussion, we focus on OMQA for CQs on KBs expressed with the *DL-lite*$_\mathcal{R}$ description logic [4]; this setting is particularly relevant from both the technical and practical viewpoints: (*i*) the UCQ, USCQ and JUCQ reformulations were defined for it and (*ii*) DL-lite$_\mathcal{R}$ underpins the QL profile of OWL2, i.e., the well-established W3C standard for OMQA on large data volume.

In Sec. 2, we recall the reformulation-based OMQA technique, point out why the evaluation performance of state-of-the-art query reformulations is intrinsically limited, and discuss our new optimization based on data summaries to circumvent this performance issue. Then, in Sec. 3, we present the demo scenarios we prepared, and sketch the demo experience we planned for the attendees.

2 Reformulation-based OMQA and optimization thereof

A DL-lite$_\mathcal{R}$ KB $\mathcal{K}$ consists of a database called an ABox, noted $\mathcal{A}$, described by an ontology called a TBox, noted $\mathcal{T}$; we note such a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

Reformulation-based OMQA amounts to reformulating ***a*** CQ q using ***a*** TBox $\mathcal{T}$ into a reformulation $q^\mathcal{T}$ so that ***for all*** ABox $\mathcal{A}'$, the relational evaluation of $q^\mathcal{T}$ on $\mathcal{A}'$ stored in an RDBMS as a relational database computes the exact

answer set of q on the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}')$. We argue that the *universality* of a $q^{\mathcal{T}}$ reformulation *needlessly limits* its evaluation performance: when reformulation-based OMQA is used to answer q on a particular KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the ABox $\mathcal{A}$ at hand is ***just one of all*** the possible ABoxes $q^{\mathcal{T}}$ accommodates to. Because of that, some sub-queries in $q^{\mathcal{T}}$ may be useless when evaluating $q^{\mathcal{T}}$ on $\mathcal{A}$: they have *no answer* on it, hence *do not participate in producing q's answers* from it. Their evaluation, which may take significant time, should thus be avoided to the extent possible; this is the goal of the new optimization we devise below.

As the names of the state-of-the-art query reformulation languages suggest, reformulations encode semantically equivalent but syntactically different algebraic combinations of sub-CQ queries: UCQ, USCQ and JUCQ. Our optimization aims at *pruning rapidly away from query reformulations* such sub-CQs with no answer on the queried KB's ABox. It relies on the notion of *ABox summary* [8], which was proposed for fast consistency checking and instance retrieval with the $\mathcal{SHIN}$ DL [6] , and which we straightforwardly adapt to DL-lite$_{\mathcal{R}}$: ABox facts are restricted to instances of *atomic* concepts and roles, i.e., just concept and role *predicates* instead of $\mathcal{SHIN}$ *formulae*. A summary $\mathcal{S}$ of an ABox $\mathcal{A}$ is an ABox homomorphic to $\mathcal{A}$ defined by a so-called *canonical function* f from the constants in $\mathcal{A}$ to the constants in $\mathcal{S}$: $\mathcal{S} = \mathcal{A}_f$, where $\mathcal{A}_f$ is obtained from $\mathcal{A}$ by replacing every constant c by its image $f(c)$ through f. A central property of such a summary, which directly follows from the existence of the $\mathcal{A}$-to-$\mathcal{S}$ homomorphism defined by the function f, is that if a CQ q has some answer on $\mathcal{A}$, then its translation q_f has some answer on $\mathcal{S}$, where q_f is obtained from q by replacing every constant c by its image $f(c)$ through f [7]. The crux of our optimization is to exploit the *contraposition* of this property, i.e., ***if*** q_f has no answer on $\mathcal{S}$, ***then*** q has no answer on $\mathcal{A}$. We therefore *optimize* a UCQ, USCQ or JUCQ reformulation by pruning away from it each of its sub-CQs with no answer on the KB's ABox summary. Finally, and crucially for our summary usage, ABox summaries can be rapidly computed and maintained upon ABox updates [8].

3 Demonstration outline

For the demo [1], we selected a set of well-established DL-lite$_{\mathcal{R}}$ benchmarks, notably **LUBM$^{\exists}$** [12] and **NPD** [10] that are respectively based on synthetic and real data. For each, we stored ABoxes of increasing sizes, from small-to-large, in the popular **DB2** (v11.5), **MySQL** (v8.0.25) and **PostgreSQL** (v12.7) RDBMSs, on top of which we deployed the off-the-shelf **Rapid** (v0.93) [5], **Compact** (v1.0b6) [13] and **GDL** (v1.0) [3] query reformulation tools[2] for (minimal) UCQ, USCQ and JUCQ reformulations respectively. We also developed a **PHP/JSP/jQuery-based GUI** (see Fig. 1 and Fig. 2) that offers a nice

[1] A one-minute teaser video of our demo is available at: http://people.irisa.fr/Francois.Goasdoue/ISWC21/demo-iswc21-teaser.mp4

[2] Query reformulation time is typically negligible w.r.t. reformation evaluation time, thus the particular choice of tools we made does not affect our conclusions.

visual attendee experience in order to *examine the performance challenges and advances in OMQA optimization.*

Scenario 1 *"Where do we stand now and how did we get there?"***:** We answer this question by carrying out performance analyses through our GUI. For each, we select: (*i*) ABoxes and queries from a benchmark based on *a collection of statistics* (e.g., ABox and query sizes, number of answers per ABox/query pair, etc) but also on the *graphical inspection of the queries*, (*ii*) query reformulation languages, and (*iii*) RDBMS(s). For a given analysis, query answering times are reported by RDBMSs, then by ABoxes, then by queries; they are displayed using stacked bar charts to show, in each case, *the time spent in reformulating the query w.r.t. the time spent in evaluating the query reformulation* (Fig. 1). Further, query reformulations can be *graphically inspected* to understand the challenges raised by their evaluation through RDBMSs. Based on such analysis, we first show that the UCQ and USCQ reformulation approaches *are not robust* OMQA solutions because the single reformulation they explore for a given query may lead to poor performance or just fail in being evaluated on moderate-to-large ABoxes, on all RDBMSs. We also point out that the performance of the JUCQ reformulation approach strongly depends on the *quality of the cost model* used to select the reformulation to send to the RDBMS. Because of that, though in principle the JUCQ reformulation approach should be competitive w.r.t. UCQ and USCQ reformulation ones, this is not always the case.

Fig. 1. Performance analysis

Scenario 2 *"What's next?"***:** We answer this question by advertising the potential of our summary-based optimization for further significant improvement of OMQA performance (up to several order of magnitude) for the UCQ, USCQ *and* JUCQ reformulation approaches. When enabled, (*i*) *statistics* are also reported for ABox summaries (size and compression ratio, time to be computed, and average update time upon update) when setting up a performance analysis and (*ii*) stacked bar chars also display for every optimized re-

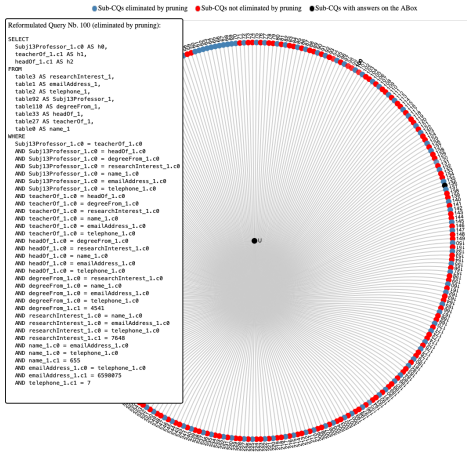

Fig. 2. UCQ reformulation inspection

formulation *the time spent in pruning its empty sub-CQs based on the ABox summary* (Fig. 1). Further, to understand the performance gain, the graphical inspection of an optimized UCQ, USCQ or JUCQ query reformulation also highlights its *empty* sub-CQs on the ABox, and which of them are *pruned out* based on the summary.

Acknowledgements

This work was partially supported by the ANR project CQFD (ANR-18-CE23-0003) and the ARED project Gedeon (Région Bretagne & Lannion Trégor Communauté).

References

1. Bienvenu, M.: Ontology-mediated query answering: Harnessing knowledge to get more from data. In: IJCAI (2016)
2. Bursztyn, D., Goasdoué, F., Manolescu, I.: Optimizing FOL reducible query answering: Understanding performance challenges. In: ISWC (2016)
3. Bursztyn, D., Goasdoué, F., Manolescu, I.: Teaching an RDBMS about ontological constraints. PVLDB (2016)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning (2007)
5. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL2QL. In: CADE (2011)
6. Dolby, J., Fokoue, A., Kalyanpur, A., Schonberg, E., Srinivas, K.: Scalable highly expressive reasoner (SHER). J. Web Semant. **7**(4) (2009)
7. El Vaigh, C.B., Goasdoué, F.: A well-founded graph-based summarization framework for description logics. In: International workshop on Description Logics (2021)
8. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The summary abox: Cutting ontologies down to size. In: ISWC (2006)
9. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in dl-lite. In: KR (2010)
10. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: EDBT (2015)
11. Leone, N., Manna, M., Terracina, G., Veltri, P.: Fast query answering over existential rules. ACM Trans. Comput. Log. **20**(2) (2019)
12. Lutz, C., Seylan, I., Toman, D., Wolter, F.: The combined approach to OBDA: taming role hierarchies using filters. In: ISWC (2013)
13. Thomazo, M.: Compact rewritings for existential rules. In: IJCAI (2013)