Optimizing FOL reducible query answering: understanding performance challenges
Damian Bursztyn, François Goasdoué, Ioana Manolescu

To cite this version:

HAL Id: hal-01400568
https://inria.hal.science/hal-01400568
Submitted on 22 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Optimizing FOL reducible query answering: understanding performance challenges

Damian Bursztyn¹, François Goasdoué² and Ioana Manolescu¹

¹INRIA & LIX, Ecole Polytechnique, France  ²U. Rennes 1 & INRIA, France

Abstract. Semantic Web data management raises the challenge of answering queries under constraints (i.e., in the presence of implicit data). To bridge the gap between this extended setting and that of query evaluation provided by database engines, a reasoning step (w.r.t. the constraints) is necessary before query evaluation. A large and useful set of ontology languages enjoys FOL reducibility of query answering: queries can be answered by evaluating a SQLized first-order logic (FOL) formula (obtained from the query and the ontology) directly against the explicitly stored data (i.e., without considering the ontological constraints).

Our demonstration showcases to the attendees, and analyzes, the performance of several reformulation-based query answering techniques, including one we recently described in [5], applied to the lightweight description logic DL-LiteR underpinning the W3C's OWL2 QL profile.

1 Introduction

Ontology-based data access (OBDA, in short) [10] aims at enabling query answering over a database, i.e., facts, subject to ontological (deductive) constraints modeling the considered application domain. An ontology, for instance, may specify that any professor is a human, has a name, and must teach some class. Constraints greatly increase the usefulness of a database: for instance, a query asking for all the humans must return all the professors, as they are known to be human, even though they are not explicitly stored as humans in the database.

Research on OBDA query answering has bloomed recently through the proposal of many languages for expressing ontological constraints, e.g., Datalog [6], Description Logics [1], Existential Rules [2], or RDF Schema for RDF graphs [4]. In particular, techniques based on first-order logic (FOL) reducibility (a.k.a. rewritability) of query answering, e.g., [7][8][13][14][16], reduce the task of computing the answer to the given query against a knowledge base (KB) to FOL query evaluation against the KB's facts alone (a.k.a. ABox), by compiling the KB's domain knowledge (a.k.a. TBox) into a so-called reformulated query. This technique is known as reformulation-based query answering, which we denote Ref in short. Evaluating a reformulated query (translated into SQL) in a highly optimized Relational Database Management System (RDBMS) storing the KB’s facts suffices to compute the complete query answer. However,

¹ http://www.w3.org/2001/sw/Specs.html
reformulated queries still raise significant performance challenges to RDBMSs (specifically, to query optimizers), due to their complex structure [4,5].

Our demonstration compares and analyzes the performance of state-of-the-art alternative Ref techniques for the lightweight DL-LiteR description logic [7], which underpins the W3C’s OWL2 QL standard for Semantic Web applications. They mainly differ w.r.t. the FOL languages in which query reformulations are produced. In particular, our recent Ref technique described in [5] departs from the literature by considering expressive enough FOL reformulation languages to admit several query reformulation alternatives, from which one with best estimated evaluation cost is picked and sent to an RDBMS. This way we avoid the reformulation performance pitfalls of the previous Ref techniques from the literature: each of them is capable of exploring a unique query reformulation, which may be turn out too complex to evaluate efficiently by an RDBMS.

We prepared a set of scenarios (data, constraints, and queries) allowing the audience to experiment with a variety of Ref techniques, and to evaluate them through well established RDBMSs: DB2 (with or without the DB2 efficient RDF store [3]) and Postgres. In particular, we show that (i) Ref techniques yielding a unique query reformulation may lead to poor performance or simply fail - on moderate-sized databases and simple constraints - on all the systems, because reformulated queries may be syntactically huge and (ii) a cost-based query reformulation approach allows avoiding such performance issues and makes Ref feasible - and efficient - in the same setting(s).

2 Cover-based query answering optimization

RDBMS query optimizers enable efficient evaluation by exploring a set of logical and physical plans, and choosing the one minimizing a cost estimation function. Since the number of possible plans is very high [12], modern optimizers rely on heuristics to explore only a few alternatives; this works (very) well for small-to-moderate size conjunctive queries (CQs in short). However, FOL reformulations go beyond CQs in general, and may be extremely large, leading the RDBMS to perform poorly (an example appears at [http://bit.ly/1TqeVMA](http://bit.ly/1TqeVMA)).

To address this, we introduce the cover-based query answering technique [5] to define a space of equivalent FOL reformulations of a CQ. A cover defines how the query is split into subqueries, that may overlap, called fragment queries, such that substituting each subquery with its FOL reformulation (obtained from any state-of-the-art technique) and joining the corresponding (reformulated) subqueries, may yield a FOL reformulation of the given query; this can be a Join of Unions of CQ (JUCQ) [4], or a slightly more complex form (two layers of joins-of-unions), denoted JUSCQ [15]. Not every cover of a query leads to a FOL reformulation; but every cover which does, yields an alternative cover-based FOL reformulation of the original query. Crucially for our problem, a smart cover choice may lead to a cover-based reformulation whose evaluation is more efficient. Thus, the cover-based technique amounts to circumventing the difficulty of modern RDBMSs to efficiently evaluate FOL reformulations in general.
We address the performance challenges raised by the evaluation of state-of-the-art FOL reformulations through RDBMSs for DL-Lite$_R$ in two steps.

First, we have identified a \textit{space of alternative equivalent FOL queries} for a given input query, belonging to richer languages than those considered so far in the literature. Allowing several FOL queries (a.k.a. FOL reformulations) is crucial for efficiency, as such equivalent alternatives may have very different evaluation performance when evaluated through an RDBMS. Therefore, instead of having a single (fixed) FOL reformulation that may be inefficient, we select the one with lowest estimated evaluation cost among possible (equivalent) alternatives.

Second, we devised a cover search algorithm, namely \textbf{GDL (Greedy Covers for DL)} \cite{5}, that relies on a \textit{cost estimation function} $\epsilon$ which returns the cost of evaluating a given query $q$ through an RDBMS storing the facts, to select the cover with the best (estimated) cost. The algorithm starts with the root cover (the cover with maximal number of fragments among all the covers for $q$ and $T$ that lead to FOL reformulations), $C_{\text{root}}$, and explores the search space adding one atom to a fragment (leading to a new cover) when the cost model suggests the resultant cover may lead to a more efficient query answering strategy. The exploration stops when no possible moves improve the cost of the currently selected best cover.

We demonstrate \textit{experimentally} the effectiveness and the efficiency of our query answering technique for DL-Lite$_R$, by deploying our query answering technique on top of Postgres and DB2, using several alternative data layouts \cite{5}.

\section{Demonstration outline}

Our demo analyzes OBDA with a particular focus on performance and completeness. We demonstrate different query reformulation techniques, which for a given CQ produce respectively: UCQ \cite{7}, USCQ \cite{15}, and JUCQs or JUSCQs \cite{5} (the latter include UCQ, USCQ and $C_{\text{root}}$ reformulations). The GUI allows the attendee to compare and understand the performance of evaluating each of them.

(i) We use well-established RDBMSs, namely PostgreSQL 9.3.2 and DB2 Express-C 10.5, on top of which queries can be answered using any cover: a fixed one (i.e., a UCQ, USCQ or $C_{\text{root}}$), a user-chosen one (a JUCQ or JUSCQ), or the best one w.r.t. cost (selected by GDL). (ii) We showcase the same alternatives on top of the DB2 efficient RDF store \cite{3}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{Understanding cover-based Query Answering.}
\end{figure}
We rely on **real and synthetic RDF data sets**, such as the *Norwegian Petroleum Directorate (NPD)* [9] and DL-LiteR version of LUBM, *LUBM*² [11].

**The demo attendee experience** is as follows. 1. Select an RDF graph (data and constraints). 2. Pick a query and answer it using different systems and techniques, comparing their performance. 3. Inspect the chosen FOL query as well as its SQL translation to understand the performance challenges its (evaluation) raises to the RDBMSs. 4. Observe the query answering runtime and examine: the chosen query plan; cardinalities and costs of (sub)queries; and (if the cover was selected by GDL) the space of explored alternatives, and their estimated costs (Figure 1).

**References**