Database Optimization Techniques for Semantic Queries

Ioana Manolescu

INRIA and U. Paris-Sud, France ioana.manolescu@inria.fr

Techniques for efficiently managing Semantic Web data have attracted significant interest from the data management and knowledge representation communities. In particular, as RDF is the most widely used model for Semantic Web data, a great deal of effort has been invested, especially in the database community, into algorithms and tools for efficient RDF query evaluation.

Semantic Web data can be seen as a colection of *facts* enriched with ontological schemas, or semantic *constraints*, based on which *reasoning* can be applied to infer new information. Taking into account this implicit information is required in order to produce complete answers to queries. The difficulty in doing so depends on the expressive power of the constraints being used to describe the semantics of the data. One of the simplest constraint languages currently used in conjunction with RDF databases is RDF Schema (RDFS, in short), whose core consists of the rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain and rdfs:range predefined predicates, which allow characterizing the relationships between classes (unary relations) and properties (binary relations). More expressive formal constraints languages can be found in the DL-Lite family [6], the Datalog$^{\pm}$ dialect [5] etc.

The literature provides two classes of techniques for implementing reasoning, namely *query reformulation* and *database saturation*. The former consists of compiling the constraints into the query, making it syntactically more complex, while the latter compiles the constraints into the data, i.e., it adds all the consequences of the facts and the constraints to the database. The performance of these techniques depends on the expressive power of the ontological schema language, as well as on the characteristics of the data and queries. While saturation appears simple and robust, it is not always feasible and it may also perform poorly, especially in a distributed setting.

Efficient query answering through FOL reformulation This talk describes some of our recent work (joint with François Goasdoué from Université de Rennes 1, France, and our PhD students Damian Bursztyn and Alexandra Roatiş), on the topic of efficiently evaluating reformulated queries by adequately translating them to first-order logic (FOL). This translation enables their evaluation by relational database management systems (RDBMSs), leveraging thus their efficient storage and processing engines.

Our first goal was to provide a FOL query reformulation algorithm for an expressive fragment of RDF in the presence of RDFS constraints. A difficulty in this context is due to the peculiar features of RDF, in particular the presence of the so-called *blank nodes* which can be seen as encoding a form of incomplete information [1]; further, going beyond the expressive power of typical DL dialects, RDF queries may span over both the data (facts) and the schema (constraints).

In [7], we have identified *the database fragment of RDF*, extending the expressive power of previously formalized RDF fragments, in particular by the usage of blank nodes in the schema, and have provided a reformulation-based query answering algorithm for this fragment. Intuitively, the algorithm is based on intelligently exploiting the query and the schema constraints in order to obtain an SQL query whose evaluation through a standard RDBMS yields the query answer.

Next, we have considered the issue of efficiently evaluating queries reformulated under RDFS constraints [4]. For relatively simple queries and constraints, such reformulated queries can be large – unions of hundreds, if not thousands of conjunctive queries (we call such unions UCQs, in short). It turns out that modern efficient RDBMSs are extremely poor at efficiently evaluating such huge UCQs, when they do not simply fail to evaluate them. To make it possible (or more efficient) to evaluate such reformulations, we have devised a cost-based optimization approach whereas we explore a set of syntactically equivalent, alternative *Joins of Unions of Conjunctive Queries* (JUCQ, in short) FOL reformulations of the original query. We estimate the performance of evaluating such alternative JUCQs through the RDBMS engine, and pick the one promising to lead to the best evaluation performance. We have shown that this technique improves reformulated query performance by up to four orders of magnitude.

In recent work [2, 3], we enlarged our focus to consider the efficient evaluation of reformulated queries in the presence of DLLite-R constraints, strictly more expressive than the RDFS schema language. It turns out that our JUCQ optimization approach developed for RDFS does not translate as such to the DL-LiteR context, since some of the alternatives which lead to equivalent reformulations in [4] yield incomplete answers under DL-LiteR constraints. We devised a general *cover-based optimization technique*, applicable to any query and constraint setting enjoying FOL reducibility, to improve the performance of evaluating a reformulated query, regardless of the algorithm actually used for reformulation. This technique relies on a set of so-called *safe query covers*, each of which is guaranteed to leed to an equivalent FOL query reformulation. Further, we introduce the notion of *extended covers*, based on which more alternative FOL reformulations can be explored, leading to better performance gains.

Looking ahead, we believe that the efficient evaluation of queries under constraints needs to combine performance benefits brought by many different techniques. Clearly, efficient storage and indexing techniques can clearly make a big impact on the performance of query evaluation per se; intelligent reformulation algorithms are needed to efficiently inject the constraints in the query. We believe there is a significant potential of performance savings in the area of *query optimization*, since today's efficient optimizers have been tuned to perfection for workloads of a very different nature than those resulting from constraint-based reformulation. In the abovementioned works, we have attempted to "optimize outside/before the RDBMS optimizer", exactly exploiting the knowledge (understanding) of constraints to which today's server are oblivious. We plan to continue work in this area, working toward a "full optimization approach" where

the best benefit is made out of the RDBMS server's strengths (storage, indexing, join processing etc.) while avoiding its "blind spots", notably found today in its optimizer's heuristics.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] D. Bursztyn, F. Goasdoué, and I. Manolescu. Efficient query answering in dl-lite through fol reformulation (extended abstract). In *DL Workshop*, 2015.

[3] D. Bursztyn, F. Goasdoué, and I. Manolescu. Efficient query answering in the presence of DL-LiteR constraints. Research Report RR-8714, INRIA Saclay, Apr. 2015.

[4] D. Bursztyn, F. Goasdoué, and I. Manolescu. Optimizing Reformulation-based Query Answering in RDF. In *EDBT*, Brussels, Belgium, Mar. 2015.

[5] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242, 2010.

[6] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. of Automated Reasoning (JAR)*, 2007.

[7] F. Goasdoué, I. Manolescu, and A. Roatiş. Efficient Query Answering against Dynamic RDF Databases. In *EDBT*, 2013.