Information Sciences 198 (2012) 1–23

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Top-k retrieval for ontology mediated access to relational databases

Umberto Straccia

Istituto di Scienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche (CNR), Pisa, Italy

ARTICLE INFO

Article history:
Received 5 February 2010
Received in revised form 11 October 2010
Accepted 18 February 2012
Available online 1 March 2012

Keywords:
Description logic
Top-k retrieval
Ontology

ABSTRACT

We address the problem of evaluating ranked top-k queries in the context of ontology mediated access over relational databases. An ontology layer is used to define the relevant abstract concepts and relations of the application domain, while facts with associated score are stored into a relational database. Queries are conjunctive queries with ranking aggregates and scoring functions. The results of a query may be ranked according to the score and the problem is to find efficiently the top-k ranked query answers.

1. Introduction

Description Logics (DLs) [4] provide popular features for the representation of structured knowledge. Nowadays, DLs have gained even more popularity due to their application in the context of the *Semantic Web*. DLs play a particular role as they are essentially the theoretical counterpart of state of the art languages to specify ontologies, such as *OWL 2*.[1]

It becomes also apparent that in these contexts, data are typically very large and dominate the intentional level of the ontologies. Hence, while in the above mentioned contexts one could still accept reasoning, specifically query answering, that is exponential on the intentional part, it is mandatory that reasoning and query answering is polynomial in the data size, *i.e.* in *data complexity* [35].

Recently, efficient management of large amounts of data and its computational complexity analysis has become a primary concern of research in DLs and in ontology reasoning systems [3,7,8,12,13,17,18,29]. This is especially important in case of ontology mediated access to relational databases for data intensive applications, where data is stored into a database and a data complexity tractable DL is used to define the relevant abstract concepts and relations of the application domain. Specifically, the adoption of such a DL allowed the development of an efficient query answering method that can roughly described as follows: given a query and an ontology, rewrite the query using the ontology as several queries that can then be submitted as SQL queries over the database. The effectiveness of this method is based on the fact that the rewriting of the query can be computed efficiently and SQL query answering may take advantage of the performance of state of the art database engines.

In this paper, we address a relatively novel issue for DLs with a huge data repository, namely the problem of *evaluating ranked top-k queries*. Usually, an answer to a query is a set of tuples that satisfy a query. Each tuple does or does not satisfy the predicates in the query. However, very often the information need of a user involves so-called *scoring predicates* [14]. For instance, a user may need: "Find *cheap* hotels *near* to the conference location". Here, *cheap* and *near* are scoring predicates. Unlike the classical case, tuples satisfy now these predicates to a score. In the former case the score may depend, *e.g.*, on the price, while in the latter case it may depend *e.g.*, on the distance between the hotel location and the conference location.

E-mail address: straccia@isti.cnr.it

[1] http://www.w3.org/TR/2009/REC-owl2-overview-20091027.

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.02.026

Therefore, a major problem we have to face with in such cases is that now an answer is a set of tuples *ranked* according to their *score*. This poses a non-negligible challenge in case we have to deal with a huge amount of instances. Indeed, virtually every tuple may satisfy a query with a non-zero score and, thus, has to be ranked. Computing all these scores, ranking them and then selecting the top-*k* ones is not feasible in practice for large size databases [14].

So far, there are very view works addressing specifically the top-k retrieval problem for logic-based languages in general and, specifically, in the DL context [15,24,26,28,32]. With respect to these works, the contribution of this paper can be summarised as follows.

- We consider a more general DL language than the one considered in top-k retrieval, such as described in [15,24,26,28]. Specifically,
  - at the ontology level, while the language is closely related to *OWL QL* (a so-called profile of the web ontology language *OWL 2* [2]), we further allow scoring functions to occur in the left hand side of an axiom to combine scores and assigning the result to the axiom's right hand side;
  - at the query level, we consider Datalog like conjunctive queries with ranking aggregates and scoring functions with scoring predicates in the same way as [14]. Our query language will subsume [15,16,22,24,26–28].
- We provide a reasoning algorithm for the specified language. Specifically,
  - we provide a generalised reformulation algorithm in the spirit of [8], that given a query and an ontology, rewrites the query as several queries;
  - we then provide a novel algorithm for efficient submission of these queries to the database. To this end we extend the so-called *Disjunctive Threshold Algorithm* provided in [23–26] to cope with ranking aggregates and main memory problems encountered in preliminary experiments.
- We conduct an experiment to asses both the effective SoftFacts system [1,33], and to illustrate some issues related to a naive approach to the top-k retrieval problem.

In the following, we proceed as follows. In the next section, we provide some basics on top-k retrieval for relational databases. Then, Section 3 describes our query and representation language, Section 4 describes the query answering algorithms, while Section 5 reports the experiments. Section 6 addresses related work and Section 7 concludes and provides an outlook to future work.

2. Top-k retrieval in relational databases

We recap here some salient notions related to the top-k retrieval problem for relational databases (see, *e.g.* [14]).

There are essentially three query models to specify the data objects to be scored, namely (i) top-k selection query; (ii) top-k join query; and (iii) top-k aggregate query, which we describe next. For illustrative purposes, we consider the following running example taken from [30].

Example 2.1. Consider a relational database excerpt (Fig. 1) about student's Curricula Vitæ, together with some basic information about the degrees they got.

2.1. Top-k selection query

In this model, the scores are assumed to be attached to base tuples. A top-k selection query is required to report the *k* tuples with the highest scores according to some user-defined scoring function.

Profile								
profID	firstName	lastName	height	age	cityOfBirth	address	city	...
2	Wayne	Hernandez	180	31	Berlin	Via Volta	Terni	...
34	Hillary 156	Gadducci	175	28	Bangalore	Church ST	New York	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮ ....

HasDegree		
profID	degID	mark
2	29	107
34	25	104
⋮	⋮	⋮

Degree	
degID	name
29	Civil_Structural_Engineering
25	Chemical_Engineering
⋮	⋮

Fig. 1. A relational database.