Data & Knowledge Engineering 68 (2009) 973–1000

Contents lists available at ScienceDirect

Data & Knowledge Engineering

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

Semantics preserving SPARQL-to-SQL translation

Artem Chebotko [a,*], Shiyong Lu [b], Farshad Fotouhi [b]

[a] Department of Computer Science, University of Texas-Pan American, 1201 West University Drive, Edinburg, TX 78539, USA
[b] Department of Computer Science, Wayne State University, 431 State Hall, 5143 Cass Avenue, Detroit, MI 48202, USA

ARTICLE INFO

Article history:
Received 6 July 2008
Received in revised form 2 April 2009
Accepted 3 April 2009
Available online 16 April 2009

Keywords:
SPARQL-to-SQL translation
SPARQL semantics
SPARQL
SQL
RDF
query
RDF store
RDBMS

ABSTRACT

Most existing RDF stores, which serve as metadata repositories on the Semantic Web, use an RDBMS as a backend to manage RDF data. This motivates us to study the problem of translating SPARQL queries into equivalent SQL queries, which further can be optimized and evaluated by the relational query engine and their results can be returned as SPARQL query solutions. The main contributions of our research are: (i) We formalize a relational algebra based semantics of SPARQL, which bridges the gap between SPARQL and SQL query languages, and prove that our semantics is equivalent to the mapping-based semantics of SPARQL; (ii) Based on this semantics, we propose the first provably semantics preserving SPARQL-to-SQL translation for SPARQL triple patterns, basic graph patterns, optional graph patterns, alternative graph patterns, and value constraints; (iii) Our translation algorithm is generic and can be directly applied to existing RDBMS-based RDF stores; and (iv) We outline a number of simplifications for the SPARQL-to-SQL translation to generate simpler and more efficient SQL queries and extend our defined semantics and translation to support the bag semantics of a SPARQL query solution. The experimental study showed that our proposed generic translation can serve as a good alternative to existing schema dependent translations in terms of efficient query evaluation and/or ensured query result correctness.

1. Introduction

The Semantic Web [7,47] has recently gained tremendous momentum due to its great potential for providing a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. Semantic annotations for various heterogeneous resources on the Web are represented in Resource Description Framework (RDF) [56,58], the standard language for annotating resources on the Web, and searched using the query language for RDF, called SPARQL [59], that has been proposed by the World Wide Web Consortium (W3C) and has recently achieved the recommendation status. Essentially, RDF data is a collection of statements, called *triples*, of the form (s, p, o), where s is called *subject*, p is called *predicate*, and o is called *object*, and each triple states the relation between a subject and an object. Such a collection of triples can be viewed as a directed graph, in which nodes represent subjects and objects, and edges represent predicates connecting from subject nodes to object nodes. To query RDF data, SPARQL allows the specification of triple and graph patterns to be matched over RDF graphs.

Explosive growth of RDF data on the Web drives the need for novel database systems, called *RDF stores*, that can efficiently store and query large RDF datasets. Most existing RDF stores, including Jena [63,62], Sesame [9], 3store [27,28], KAON [54], RStar [35], OpenLink Virtuoso [22], DLDB [38], RDFSuite [3,52], DBOWL [37], PARKA [50], RDFProv [12], and RDFBroker [48] use a relational database management system (RDBMS) as a backend to manage RDF data. The main advantage of the

* Corresponding author. Tel.: +1 956 381 2577; fax: +1 956 384 5099.
E-mail addresses: artem@cs.panam.edu (A. Chebotko), shiyong@wayne.edu (S. Lu), fotouhi@wayne.edu (F. Fotouhi).

0169-023X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2009.04.001

RDBMS-based approach is that a mature and vigorous relational query engine with transactional processing support can be reused to provide major functionalities for RDF stores. The main challenge of this approach is that one needs to resolve the conflict between the graph RDF data model and the target relational data model. This usually requires various mappings, such as schema mapping, data mapping, and query mapping, to be performed between the two data models. One of the most difficult problems in this approach is the translation of SPARQL queries into equivalent relational algebra expressions and SQL queries, which can be further optimized and evaluated by the relational query engine and their results can be returned as SPARQL query solutions.

We identify three goals of SPARQL-to-SQL translation that are very important to achieve:

(1) *Correctness*. A *semantics preserving* translation is required to ensure that the semantics of a SPARQL query is equivalent to the semantics of this query translated into SQL, such that the SPARQL and SQL queries produce equivalent results.
(2) *Schema-independence*. A *generic* translation which does not depend on a particular relational database schema can be used for various database representations employed in existing RDF stores.
(3) *Efficiency*. An *efficient* translation should not only generate equivalent SQL queries quickly, but also ensure that generated queries are efficient in terms of their evaluation over a relational database.

Existing relational RDF stores implement different SPARQL-to-SQL translation algorithms based on subjective interpretations of the mapping-based semantics of SPARQL [59,39,40]. Although the mapping-based semantics of SPARQL defines a precise and concise SPARQL query evaluation mechanism, it does not support SPARQL-to-SQL translation directly. As a result, existing solutions succeed in approaching the goal of efficiency, but fail to show to be semantics preserving and/or generic. The major obstacle to the definition of a mathematically rigorous SPARQL-to-SQL translation is the gap between RDF and relational models, in general, and between SPARQL and SQL, in particular.

In this work, we define our relational algebra based semantics of SPARQL and propose the first provably semantics preserving and generic SPARQL-to-SQL translation. Furthermore, we extend the semantics and translation to support the bag semantics of a SPARQL query solution and outline our simplifications to the translation to generate simpler and more efficient SQL queries. Our main contributions are summarized in the following:

- We formalize a relational algebra based semantics of SPARQL as a function *eval*, which bridges the gap between SPARQL and SQL. We prove that *eval* is equivalent to the mapping-based semantics of SPARQL under the interpretation function[1] λ, which is used to establish the equivalence relationship[2] between two SPARQL solution representations: a relational representation and a mapping-based representation.
- We define a SPARQL-to-SQL translation as a function *trans* for core SPARQL constructs and prove that *trans* is semantics preserving with respect to the relational algebra based semantics of SPARQL under the interpretation function ϕ, which is used to establish the equivalence relationship between a relation produced by the relational algebra based SPARQL semantics *eval* and a relation produced by the evaluation of a *trans*-generated SQL query; *eval* and *trans* may produce relations with different relational attribute names due to the SQL naming constraints. *trans* supports the translation of SPARQL queries with triple patterns, basic graph patterns, optional graph patterns, alternative graph patterns, and value constraints. *trans* is the first provably semantics preserving translation in the literature.
- We achieve the generic property for our SPARQL-to-SQL translation *trans*, such that it supports both schema-oblivious and schema-aware database representations of existing RDBMS-based RDF stores. We do this by full separation of the translation from the relational database schema design (represented by RDF-to-Relational mappings α and β). We verify that *trans* can be implemented in at least 12 existing RDF stores, including Jena, Sesame, 3store, KAON, RStar, OpenLink Virtuoso, DLDB, RDFSuite, DBOWL, PARKA, RDFProv, and RDFBroker.
- We outline a number of simplifications for the SPARQL-to-SQL translation to generate simpler and more efficient SQL queries, and extend *eval* and *trans* to support the bag semantics of a SPARQL query solution.
- Finally, we conduct an experimental study to explore how our generic SPARQL-to-SQL translation compares to existing schema dependent translations and how our proposed simplifications affect query performance.

The big picture of our research flow is illustrated in Fig. 1. At the data level, we define RDF-to-Relational mappings α and β, which capture how an RDF graph is stored into a relational database. At the query level, the figure illustrates the first two contributions discussed above, where the dashed arrow represents the mapping-based semantics of SPARQL defined in [39], the dotted arrows represent our contributions to the definition of relational algebra based semantics of SPARQL, and the solid arrows represent our contributions to the definition of the SPARQL-to-SQL translation. The leftmost $\Longleftrightarrow$ arrow represents the equivalence between the two semantics definitions, and the rightmost $\Longleftrightarrow$ arrow represents that the translation is semantics preserving with respect to the relational algebra based semantics of SPARQL. The third contribution, the generic goal, is achieved by full separation of the translation from the relational database schema design via the use of mappings α and β, that are first defined at the data level and later passed as parameters to the translation.

[1] Here and after, by "under the interpretation function", we mean that the function is applied to a query solution.
[2] Here and after, by "equivalence" or "equivalence relationship", we mean the mathematical equivalence between sets of elements (represent query solutions) or functions (represent query language semantics), which should be clear from the context.