Chapter 13

Nested Optional Join for Efficient Evaluation of SPARQL Nested Optional Graph Patterns

Artem Chebotko
University of Texas - Pan American, USA

Shiyong Lu
Wayne State University, USA

ABSTRACT

Relational technology has shown to be very useful for scalable Semantic Web data management. Numerous researchers have proposed to use RDBMSs to store and query voluminous RDF data using SQL and RDF query languages. This chapter studies how RDF queries with the so called well-designed graph patterns and nested optional patterns can be efficiently evaluated in an RDBMS. The authors propose to extend relational algebra with a novel relational operator, nested optional join (NOJ), that is more efficient than left outer join in processing nested optional patterns of well-designed graph patterns. They design three efficient algorithms to implement the new operator in relational databases: (1) nested-loops NOJ algorithm, NL-NOJ, (2) sort-merge NOJ algorithm, SM-NOJ, and (3) simple hash NOJ algorithm, SH-NOJ. Using a real life RDF dataset, the authors demonstrate the efficiency of their algorithms by comparing them with the corresponding left outer join implementations and explore the effect of join selectivity on the performance of these algorithms.

INTRODUCTION

The Semantic Web (Berners-Lee, Hendler, & Lassila, 2001; Shadbolt, Berners-Lee, & Hall, 2006) 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 data is represented in Resource Description Framework (RDF) (W3C, 2004a, 2004b), the standard language for annotating resources on the Web, and queried using the SPARQL (W3C, 2008) query language for RDF that has been recently proposed by the World Wide Web Consortium. RDF data is a collection of statements, called triples, of the form <s,p,o>, where s is a subject, p is a predicate and o is an object, and each triple states the relation between the subject and the object. Such collection
Nested Optional Join for Efficient Evaluation

of triples can be represented as a directed graph, in which nodes represent subjects and objects, and edges represent predicates connecting from subject nodes to object nodes. SPARQL allows the specification of triple and graph patterns to be matched over RDF graphs.

Increasing amount of RDF data on the Web drives the need for its efficient and effective management. In this light, numerous researchers (Abadi, Marcus, Madden, & Hollenbach, 2007; Agrawal, Somani, & Xu, 2001; Alexaki, Christophides, Karvounarakis, & Plexousakis, 2001; Beckett & Grant, 2003; Broekstra, Kampman, & Harmelen, 2002; Erling, 2001; Harris & Gibbins, 2003; Ma, Su, Pan, Zhang, & Liu, 2004; Narayanan, Kurc, & Saltz, 2006; Pan & Hefflin, 2003; Sintek & Kiesel, 2006; Stoffel, Taylor, & Hendler, 1997; Theoharis, Christophides, & Karvounarakis, 2005; Volz, Oberle, Motik, & Staab, 2003; Wilkinson, Sayers, Kuno, & Reynolds, 2003) have proposed to use RDBMSs to store and query RDF data using the SQL and SPARQL query languages.

An important class of graph patterns that are mostly common in RDF queries in practice is the so called well-designed graph patterns (Perez, Arenas, & Gutierrez, 2006a). A well-designed graph pattern gp can contain arbitrary many optional graph patterns that can be nested in each other as in the following equation:

\[
gp_1 \text{OPT}(gp_2 \text{OPT}(gp_3 \text{OPT}(...(gp_{n-1} \text{OPT}(gp_n))...))
\] (1)

where each gp_1, gp_2, gp_3, ..., gp_{n-1}, gp_n can be a basic graph pattern (set of triple patterns) or another graph pattern with optional sub-patterns such as (1), OPT indicates an optional graph pattern that follows it, and parenthesis define the order of evaluation. In (1), gp_2, gp_3, ..., gp_{n-1}, gp_n are optional graph patterns, and each gp_i \geq 3, is a nested optional graph pattern with respect to gp_i. By the definition of a well-designed graph pattern, the following property for gp holds: for any sub-pattern (gp_x \text{OPT}(gp_y)) in gp, if a variable ?v occurs both outside this sub-pattern and inside gp_y, then ?v also occurs in gp_x. The formal semantics of the well-designed graph patterns with nested optional patterns is defined by Perez et al. (2006a) and W3C (2008). Informally, it can be summarized as follows:

- **Basic semantics of optional graph patterns.** The evaluation of an optional graph pattern is not obligated to succeed, and in the case of failure, its variables are unbound. For example, in (1), gp_n does not have to succeed for gp_{n-1} to succeed.

- **Semantics of shared variables in optional graph patterns.** In general, shared variables must be bound to the same values. Variables can be shared among subjects, predicates, objects, and across each other. For example, in (1), if a variable ?v occurs both inside gp_1 and gp_2, it must be bound to the same value in both graph patterns.

- **Semantics of nested optional graph patterns.** Before a nested optional graph pattern can succeed, all containing optional graph patterns must have succeeded. For example, in (1), gp_3 corresponds to an optional graph pattern nested inside another optional graph pattern gp_2, and gp_3 can only succeed if gp_2 succeeds.

Therefore, a well-designed graph pattern gp as in (1) can have n-2 nested optional graph sub-patterns gp_3, ..., gp_{n-1}, gp_n, and an efficient evaluation of these nested patterns is very important. While the literature on the SPARQL-to-SQL translation is abundant (see the related work section), a few researches (Chebotko, Lu, Jamil, & Fotouhi, 2006; Cyganiak, 2005) study the translation of RDF queries with nested optional