BIT-BY-BIT: INDEXING AND QUERYING RDF DATA USING COMPRESSED BIT-VECTORS

By

Medha Atre

An Abstract of a Thesis Submitted to the Graduate Faculty

of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: COMPUTER SCIENCE

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Prof. James A. Hendler, Thesis Adviser

Prof. Mohammed J. Zaki, Member

Prof. Sibel Adali, Member

Dr. Ora Lassila, Member

Rensselaer Polytechnic Institute
Troy, New York

August 2011
(For Graduation December 2011)

ABSTRACT

The Resource Description Framework (RDF) – a standard for representing semantically annotated data – is widely being adapted for information representation in various domains, e.g., biotechnology (UniProt), government data (the data.gov project), scholarly resources (DBLP), web resources (DBPedia), connections between people (FOAF) and many more. Due to the increasing use of RDF as a standard for data representation in the past few years, the amount of RDF data available on the web has increased at a break-neck speed. This has necessitated addressing two important issues – efficient ways of (a) storing, and (b) querying the RDF data.

SPARQL – a standard query language for querying the RDF data – closely follows the SQL query language. As is the case for SQL, SPARQL's join queries (also known as *basic graph pattern (BGP)* queries) are the main building blocks of other SPARQL queries. BGP queries are performance intensive. In this thesis, we propose a novel way of storing RDF data and an efficient way of processing SPARQL join queries – a system called *BitMat* – which is specifically aimed at *low-selectivity* queries. BitMat uses the well-known technique of *compressed bit-vectors* to store RDF data by representing it as a 3-dimensional bit-cube (subject, predicate, object as each dimension of the bitcube). The key aspect of our join query processing is a novel 2-phase algorithm. In the first phase – based on the idea of *semi-joins* – we do aggressive pruning of the candidate RDF triples, and in the second phase, we directly *stitch* the final results from the pruned RDF triples without building any intermediate join tables – similar to the concept of *multi-way joins*.

Further, we extend our technique of pruning the candidate RDF triples to process DISTINCT and OPTIONAL clauses in SPARQL. Specifically for the DISTINCT clause, BitMat's pruning phase can be used to choose BitMats that must be processed for the completion of the query and omit the ones which are not required. This further reduces the memory requirements of the query processor, because the unwanted BitMats can simply be removed from the memory. An OP-

TIONAL SPARQL query works is same as a "left outer join" query. The results for an OPTIONAL query can be generated by simple modification of the BitMat's second phase of final result generation algorithm. Other SPARQL clauses like OFFSET, LIMIT, ASK can be handled by the BitMat algorithm as well. We have included a detailed description of all the SPARQL clauses that can exploit BitMat algorithm in this thesis.

Please note that for the scope of this thesis work, we have considered SPARQL recommendation version 1.0. Most constructs discussed in this document can be used as is in SPARQL version 1.1, except the "property paths" introduced in SPARQL 1.1. RDF data is primarily graph data. Another common graph problem is to find the relationship between two distant resources in terms of a specific type of path between them. Such queries are commonly known as *constrained reachability queries*. SPARQL 1.1 recommendations have addressed a subset of this problem by introducing "property paths". The problem of exploring the associativity between two nodes in the graph (constrained reachability) is of practical interest to the users. Consider a query formulated as an order of edge labels "* :friendOf .* :studiedAt .* :locatedIn .*" between the start node *Tom* and the destination node *Greece* and is essentially asking – "does there exist any path between nodes *Tom* and *Greece* in the network, such that an edge labeled *:friendOf* appears on it followed somewhere by the an edge labeled *:studiedAt* followed by an edge labeled *:locatedIn*?" We focus on this specific subset of problem which we call – *label-order constrained reachability (LOCR)*. We propose a system – *BitPath* – which uses compressed bit-vectors to build specific indices for each node in the RDF graph and a query processing algorithm based on *greedy pruning* and *divide-and-conquer* technique to solve this problem.

In summary, this thesis addresses two important aspects of querying RDF data – (a) performance intensive SPARQL queries, and (b) label-order constrained reachability queries. We show that using characteristics of compressed bit-vectors in both – BitMat and BitPath algorithms – we can come up with memory efficient solutions for these problems.