Framework and Schema for Semantic Web Knowledge Bases

James P. McGlothlin

The University of Texas at Dallas
Richardson, TX, USA
jpmcglothlin@utdallas.edu

Problem and Motivation

The goal of our research is to provide scalable and efficient solutions for RDF knowledge management. There is significant research concerning schemata and tools for RDF storage and efficient querying. There is also significant research into ontology design, representation, extraction and alignment, and into representing uncertainty in ontologies. However, there are not efficient and scalable solutions that handle knowledge inference and uncertainty. Existing solutions for efficient RDF storage and querying do not provide for efficient inference queries. There are solutions for querying inference but they are not efficient or scalable. There are solutions for uncertainty reasoning but they are even less efficient and scalable.

Our goal is to provide a solution that supports knowledge inference, uncertainty reasoning, Bayesian networks and ontology alignment without sacrificing efficiency and scalability.

Contribution

Our contribution is that we materialize the information, store it and make it available for efficient retrieval. It is not our intention to create our own persistence mechanism, for this we use column store databases. However, we do so using a novel bit vector schema. It is not our intention to define the rules that govern inference. Instead, we define inference rules that implement the constructs defined by OWL (Web Ontology Language), and we provide a flexible framework for registering unique inference rules. It is not our intention to define the ontology representation for probabilistic reasoning. We instead define a framework that will allow technology and representation formats such as BayesOwl (Zhang et al., 2009), Pronto, and PR-OWL (Costa et al., 2008) to be used. Our contribution is that we provide a unique bit vector schema to materialize and persist this information, a flexible framework to integrate these technologies, a process for manipulating the data efficiently, and an efficient query engine to retrieve the information.

Proposed Solution

Our strategy is to materialize and store all possible knowledge. Inference is performed when triples are added rather than during query processing. This includes inference involving probability. For uncertain information, we store the inferred triples and the associated probability.

Schema. The first stage of our proposal is our schema. We must provide a schema that is efficient and scalable for concrete datasets before we can even address inference and uncertainty. Furthermore, materializing inferred triples will greatly increase the size of the dataset. Most schemata incur a query performance penalty when the size of the dataset grows. For our inference solution to be viable, we require a solution which does not incur this penalty.

We have designed a bit vector schema, RDFVector, specifically for RDF datasets. We store the RDF triples in three bit vector tables: the POTable, the SOTable and the PSTable. Each URI or literal is dictionary-encoded to a unique ID number which is the index into the bit vectors. As an example, the POTable includes four columns: the PropertyID, the SubjectID, the SubjectBitVector and a BitCount. The SubjectBitVector has a 1 for every subject that appears with that property and object in a RDF triple. For example, all subjects matching <*?s type text*> can be retrieved from a single tuple and returned as a single bit vector, indexed by the ID numbers. The BitCount is used to support aggregation queries and query optimization.

The bit vectors are the reason we are able to store inferred triples at little or no cost. Since there are bits in each vector for every known URI, our solution incurs a performance penalty only when the dataset's vocabulary (number of unique URIs) is increased, and OWL inference usually adds only a few terms to the vocabulary. For our experiments, we utilize the LUBM (Lehigh University Benchmark, http://swat.cse.lehigh.edu/projects/lubm/) dataset with >44 million RDF triples. For this dataset, 20,407,385 additional triples are inferred, yet only 22 unique URIs are added by this inference. Our schema pays no performance penalty for the addition of 20 million inferred triples, only for the 22 new URI terms.

In addition to the bit vectors, we provide a triples table. The triples table is the only table exposed to the inference rules and all user additions, deletions and updates are performed against this table. The triples table allows us to encapsulate our schema and to optimize processes for pushing updates across our other tables. The triples table also contains the inference count which is used to support deletions of inferred triples. The triples table is not used during queries; it is used only for manipulating the dataset.

Inference. The second stage of our proposal is to materialize inferred triples. Consider the triple *<Professor0 type FullProfessor>*. The LUBM ontology will allow us to infer 4 additional triples: *<Professor0 type Professor>, <Professor0 type Faculty>, <Professor0 type Employee>, <Professor0 type Person>*. Our strategy is to materialize and store all 4 inferred triples. As *Professor, Faculty, Employee* and *Person* exist elsewhere in the dataset, no new vocabulary is introduced. All that is required to add these triples to the bit vector tables is to change 0s to 1s in the vectors; the size of the bit vectors is not increased. Now, we can execute a query such as *List all persons* by reading a single bit vector. To query all persons with vertical partitioning (Abadi et al., 2007) or RDF-3X (Neumann&Weikum, 2008) would require 21 subqueries and 20 unions.

We provide an inference engine to register and manage inference rules. When a triple is added to the triples table, the inference engine iterates through the inference rules and allows each rule to add inferred triples. We have implemented inference rules for all OWL constructs.

Deletions offer a special challenge for inferred triples. If the base triple is deleted, then any inferred triples should be deleted as well. However, a triple might be inferred from more than one base triple. We solve this problem by maintaining an inference count in the triples table, and only deleting the inferred triple when the count becomes 0.

Uncertainty. The third stage of our proposal is to materialize uncertain information. Our goal is to associate probability numbers with facts (RDF triples) in the database. There is no standard for representing probability in an ontology. Our goal is not to restrict the ontology representation. Therefore, as with provable inference, we allow inference rules to register.

To support probability, we modify the triples table and the bit vector tables. For the triples table, we add extra columns for the probability and a lineage explanation of the inference (the base triples(s) and inference rule(s)). For the bit vector tables, we add a threshold column. The vectors become bit vectors with 1s for every triple known with probability>=threshold. For vectors without probability, the threshold is 1. Below is an example POTable with probabilities and thresholds:

Property	Object	Threshold	SubjectBitVector	Count
hasDisease	LungCancer	1.0	101001010100100000	6
hasDisease	LungCancer	0.75	101101010101100001	9
hasDisease	LungCancer	0.5	111101010101101011	12
hasDisease	LungCancer	0.25	111101110101101011	13

We can now retrieve all subjects with >0.5 probability of having lung cancer by reading a single tuple, and we can still use bitwise operations to join vectors. If a query requests a probability not provided by the threshold, we use the bit vectors for efficient selection and then we access the triples table for precise comparison. However, our experiments show that we can materialize 100 thresholds (.01 precision) without a performance degradation, as only the threshold vector being queried is read into memory.

We propagate the probabilities throughout the system based on the simple Bayesian logic rule that *P(A)= P(A|B)*P(B)*. We recognize that there can be ambiguities when 2 inference rules attempt to add the same triple with different probabilities. When this occurs, that means multiple conditional probabilities are met. In such a case, we can determine *P(A|B^C)*. We utilize existing Bayesian Network reasoners such as BayesOWL to manage the graph and determine the probabilities. We support and register exception handlers to enable flexibility.

Progress to Date and Future Work

We have implemented our bit vector solution, RDFVector, and our inference materialization solution. We have implemented and registered inference rules to support all OWL constructs (see our RDFKB paper, McGlothlin &Khan, 2009). We have performed experiments with this solution for 23 queries using datasets of over 60 million triples. Our solution achieved better query performance than vertical partitioning, RDF-3X or triple store solutions. We have implemented our uncertainty reasoning solution and tested it with small synthetic ontologies and datasets.

In order to increase scalability further, we plan to utilize the Hadoop framework to provide a cloud computing solution. For uncertainty reasoning, we plan to identify or create large benchmarks so we can test for efficiency and scalability. We also plan to further integrate with existing tools and technology, and to create a more complete Bayesian Network reasoner. For ontology alignment, we have recently submitted our grant proposal and we our finishing the design phase. We plan to utilize BayesOWL in combination with our uncertainty reasoning framework to identify matching concepts and to calculate and persist similarity measurements as probabilities.

References

Abadi, D.J.; Marcus, A.; Madden, S.; Hollenbach, K.J. 2007. Scalable Semantic Web Data Management Using Vertical Partitioning. *In Proc. of VLDB*, 411-422.

Neumann, T.; Weikum, G. 2008. RDF-3X: a RISC-style engine for RDF. *In Proc. of VLDB*, 647-659.

Costa, P.C.G.D.; Laskey, K.B.; Laskey, K.J. 2008. PR-OWL: A Bayesian Ontology Language for the Semantic Web. *In Proc. of URSW*, 88-107.

Zhang, S.; Sun, Y.; Peng, Y.; Wang, X. 2009. BayesOWL: A Prototype System for Uncertainty in Semantic Web. *In Proc. of IC-AI*, 678-684

McGlothlin, J; Khan, L. 2009. RDFKB: Efficient Support For RDF Inference Queries and Knowledge Management. *In Proc. of IDEAS.*