Reasoning with Large Data Sets

Darko Anicic

Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria
darko.anicic@deri.org

Abstract. Efficient reasoning is a critical factor for successful Semantic Web applications. In this context, applications may require vast volumes of data to be processed in a short time. We develop novel reasoning techniques which will extend current reasoning methods as well as existing database technologies in order to enable large scale reasoning. We propose advances and key design principles primarily in: making an efficient query execution plan as well as in memory, storage and recovery management. Our study is being implemented in Integrated Rule Inference System (IRIS) - a reasoner for Web Service Modeling Language.

1 Problem Statement

The Web Service Modeling Language WSML[1] is a language framework for describing various aspects related to Semantic Web (SW) services. We are developing IRIS[2] to serve as a WSML reasoner which handles large workload efficiently.

Current inference systems exploit reasoner methods developed rather for small knowledge bases [2]. These systems[3], although utilize mature and efficient relational database management systems (RDBMSs) and exploit a number of their evaluation strategies (e.g., query planning, caching, buffering etc.), cannot meet requirements for reasoning in complex SW applications. Reason for this is found in the fact that database techniques are rather developed for explicitly represented data, and need to be extended for dealing with implicit knowledge.

In this work we investigate a framework which generalizes relational databases by adding deductive capabilities to them. RDBMSs suffer some limitations w.r.t the expressivity of their language. Full support for recursive views is one of them [3]. Further on, negation as failure is recognized as a very important nonmonotonic property for the Semantic Web. RDBMSs, although deal with negation as failure, can not select a minimal fixpoint that reflects the intended meaning in situations where the minimal fixpoint may not be unique. Our framework, although exceeding capabilities of RDBMSs, does not compromise their performance.

Current reasoners cannot cope with large data sets (i.e., relations larger than system main memory). Hence a reasoner needs to deal effectively with portions of

[1] WSML: http://www.wsmo.org/TR/d16/d16.1/v0.2/.
[2] IRIS: http://sourceforge.net/projects/iris-reasoner/.
[3] Reasoners which utilize persistant storage: KAON2, Aditi, InstanceStore, DLDB.

relations (possible distributed over many machines), and sophisticated strategies for partition-level relation management are required. Consequently, a relevant topic for our present and future work is: *The development of effective optimization algorithms as well as distribution and memory management strategies for reasoning with large data sets.*

2 Efficient Large Scale Reasoning: an Approach

We will now give a short overview of our approach to achieving effective reasoning with large data sets.

Unlike other inference systems[4], which utilize SQL to access existential relations, we tightly integrate IRIS with its storage layer (i.e., rules are translated into relational algebra expressions and SQL is avoided as an unnecessary overhead). We extend embedded RDBMS query optimizer (which is rather designed to be used for extensional data) for derived relations. The estimation of the size and evaluation cost of the intensional predicates will be based on the adaptive sampling method [4, 1], while the extensional data will be estimated using a graph-based synopses of data sets similarly as in [5]. Further on, for reasoning with large relations, run time memory overflow may occur. Therefore in IRIS we are developing novel techniques for a selective pushing of currently processed tuples to disk. This technique will be further extended for data distributed over many disks (e.g., a cluster of machines). Such techniques aim to enable IRIS to effectively handle large workload which cannot fit in main memory of the system.

Our framework comprises a recovery manager and thus features fault-tolerant architecture. Using logging and replications we ensure that, when a crash occurs, the system may continue with an ongoing operation without loss of previously computed results.

3 Acknowledgment

I am grateful to Michael Kifer and my supervisors: Stijn Heymans and Dieter Fensel for their help in the work conceptualization and insightful discussions.

References

1. M. E. Vidal E. Ruckhaus and E. Ruiz. Query evaluation and optimization in the semantic web. In *ALPSWS2006 Workshop, Washington, USA.*
2. Dieter Fensel and Frank van Harmelen. Unifying reasoning and search to web scale. *IEEE INTERNET COMPUTING*, page 3, 2 2007.
3. Michael Kifer, Arthur Bernstein, and Philip M. Lewis. *Database Systems: An Application Oriented Approach.* Addison-Wesley, Boston, MA, USA, 2005.
4. R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. In *PODS '90*, NY, USA.
5. J. Spiegel and N. Polyzotis. Graph-based synopses for relational selectivity estimation. In *SIGMOD '06*, NY, USA.

[4] KAON2, QUONTO, InstanceStore and DLDB exploit SQL for querying.