SPARQL Execution as Fast as SQL Execution on Relational Data

Juan F. Sequeda and Daniel P. Miranker

Department of Computer Science, University of Texas at Austin
{jsequeda, miranker}@cs.utexas.edu

Abstract. Relational Database to RDF (RDB2RDF) systems executes SPARQL queries on the relational data. Past studies have shown that RDB2RDF systems do not perform well, in other words, the execution time of a SPARQL query on a RDB2RDF system compared to its semantically equivalent SQL query is much slower. Therefore, we ask ourselves, *what optimizations are needed in order to support effective SPARQL execution on relationally stored data*? We experimented on Microsoft SQL Server, using the Barton and Berlin SPARQL Benchmark, and Ultrawrap, an automatic RDB2RDF wrapping system that has been architectured to leverage the SQL optimizer. Our initial results identify two important optimizations for effective SPARQL execution using Ultrawrap: *detection of unsatisfiable conditions* and *self-join elimination*.

Keywords: Relational Databases, RDB2RDF, Query optimizations

1 Introduction

Nearly 70% of the world's web sites are backed by SQL databases [13]. The proportion suggests that the success of the Semantic Web rests on methods for integrating those databases. Integrating relational databases with the Semantic Web can be accomplished by two types of Relational Database to RDF (RDB2RDF) wrapper systems: extract relational data, translate it to RDF and load the results into a triple-store [14] or provide SPARQL execution over the relational data without replicating the data [1-3, 7]. Standardization of the translation process is the subject of a W3C working group [6, 11]. In our research, we focus on the latter.

Two studies that evaluate SPARQL execution of RDB2RDF systems with native SQL execution on the relational database conclude: existing RDB2RDF systems do not compete with the traditional relational database [8,12]. The conclusion is surprising in the face of Angles and Gutierrez's result that SPARQL is equivalent in expressive power to relational algebra [5] and because one would expect that these RDB2RDF systems would translate a SPARQL query to SQL and that this query would be well optimized by the database. Hence, we ask ourselves: *what optimizations are needed in order to support effective SPARQL execution on relationally stored data?*

Our approach was to use Ultrawrap, a RDB2RDF system that makes maximal use of existing SQL infrastructure on top of Microsoft SQL Server using the Barton and

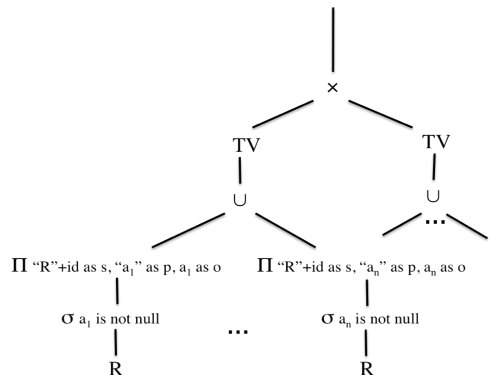
Berlin SPARQL Benchmark, and determine if and how it might produce physical plans that performed comparably to semantically equivalent native SQL queries. The results of our evaluation lead us to identify two important optimizations in order to effectively execute SPARQL queries using Ultrawrap: *detection of unsatisfiable conditions* and *self-join elimination*. We observe that Microsoft SQL Server only has the first optimization; nevertheless, most benchmark queries on Ultrawrap approach the performance of the semantically equivalent native SQL queries.

2 Ultrawrap

Ultrawrap is an RDB2RDF system that does not implement any optimizations of its own and passes all the "*hard work*" to the SQL optimizer. It consists of compile time which 1) translates the relational schema to OWL and 2) the relational data to RDF through SQL Views. The run time first 3) translates SPARQL to SQL on the views and 4) the SQL optimizer is responsible for all the physical mappings and their implementation.

3 Two Important Relational Optimizations

Upon succeeding in wrapping a database and reviewing query plans, we observed that two relational database optimizations are important for effective execution of SPARQL queries under our organization: (1) *detection of unsatisfiable conditions* and (2) *self-join elimination*. In Fig. 1 we show an initial query plan on Ultrawrap and Fig 2 and Fig 3 show how the query plan evolves through these optimizations.


×
TV
TV
∪
∪
…
Π "R"+id as s, "a1" as p, a1 as o
Π "R"+id as s, "an" as p, an as o
σ a1 is not null
…
σ an is not null
R
R
Fig. 1. Initial logical query plan

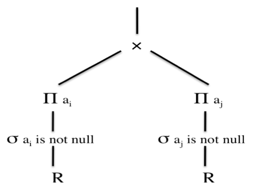


Fig. 2. Logical query plan after detection of unsatisfiable conditions optimizations

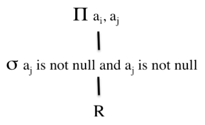


Fig. 3. Logical query plan after self-join elimination optimization

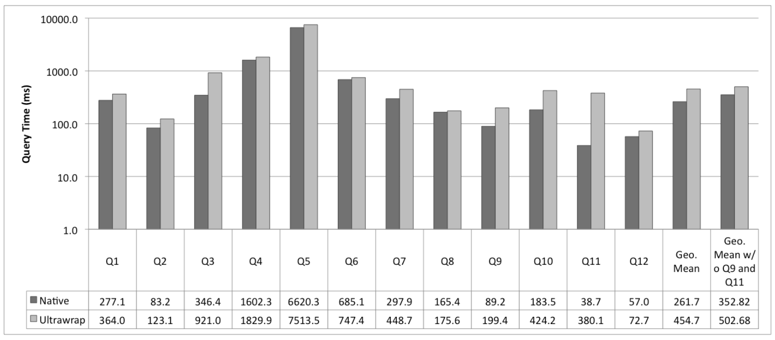
Perhaps, not by coincidence, these two optimizations are among semantic query optimization (SQO) methods introduced in the 1980's [8]. In SQO, the objective is to use integrity constraints to rewrite a query into a semantically equivalent one and eliminate contradictions. These techniques were initially designed for deductive databases and then integrated in commercial relational databases [9].

Detection of Unsatisfiable Conditions. The idea is to determine that the query result will be empty if the existence of another answer would violate some integrity constraint in the database. This would imply that the answer to the query is empty and therefore the database does not need to be accessed [8].
Self-Join Elimination. The idea is to determine integrity constraints that are used to eliminate a literal clause in the query. This implies that a join could also be eliminated if the table that is being dropped does not contribute any attributes in the results [8].

3 Experimental Results

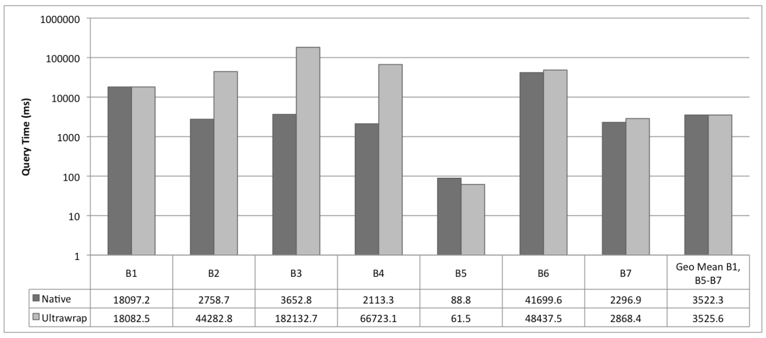
We used the query set of Barton [4] on a dump of DBLP and Berlin SPARQL Benchmark [8] and compared the query execution time of SPARQL queries on Ultrawrap with the semantically equivalent native SQL query. All experiments were conducted using Microsoft SQL Server 2008 R2 Developer Edition running on Microsoft Windows Server 2008 R2 on top of VMWare having access to Intel Xeon X7460 2.66 GHz with four cores and 8 GB of RAM. For all benchmarks we ran the experiments under cold (Fig 1-2) and warm cache. To calculate query time, we used the SET STATISTICS ON option and used the elapsed time that this option returns. We ran each query twenty times. For cold cache, between each query, we would restart the database and provide the average times.

We thoroughly experimented with Microsoft SQL Server and identified that it only has the first optimization. Nevertheless, we show that most SPARQL queries on Ultrawrap execute at speeds that approach the execution speed of semantically equivalent native SQL queries. Queries with unbounded predicates are slower. Our initial experiments showed that IBM DB2 has both optimizations while MySQL does not have either of the optimizations. Future work consists of completely porting Ultrawrap to other commercial relational databases and evaluating the results.



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Geo. Mean	Geo. Mean w/o Q9 and Q11
■ Native	277.1	83.2	346.4	1602.3	6620.3	685.1	297.9	165.4	89.2	183.5	38.7	57.0	261.7	352.82
□ Ultrawrap	364.0	123.1	921.0	1829.9	7513.5	747.4	448.7	175.6	199.4	424.2	380.1	72.7	454.7	502.68

Fig. 1. Query time in miliseconds for Ultrawrap and Native relational database on BSBM (100 million triples)



	B1	B2	B3	B4	B5	B6	B7	Geo Mean B1, B5-B7
▨ Native	18097.2	2758.7	3652.8	2113.3	88.8	41699.6	2296.9	3522.3
□ Ultrawrap	18082.5	44282.8	182132.7	66723.1	61.5	48437.5	2868.4	3525.6

Fig. 2. Query time in miliseconds for Ultrawrap and Native relational database of Barton queries on DBLP dataset (45 million triples)

Acknowledgments. This work was supported by the National Science Foundation under grant 1018554. Juan Sequeda was supported by a NSF Graduate Research Fellowship.

References

1. Ultrawrap. http://ribs.csres.utexas.edu/ultrawrap/ .
2. D2R Server. http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/ .
3. SquirrelRDF. http://jena.sourceforge.net/SquirrelRDF .
4. D. Abadi, A. Marcus, S. Madden, and K. Hollenbach. Using the Barton libraries dataset as an RDF benchmark. Technical Report MIT-CSAIL-TR-2007-036, MIT.
5. R. Angles and C. Gutierrez. The Expressive Power of SPARQL. In ISWC, pages 114-129, 2008.
6. M. Arenas, E. Prud'hommeaux, and J. Sequeda. Direct mapping of relational data to RDF. W3C Working Draft 24 March 2011, http://www.w3.org/TR/rdb-direct-mapping/.
7. J. Barrasa, O. Corcho and A. Gomez-Perez. R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language. In SWDB, 2004.
8. C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. Int. J. Semantic Web Inf. Syst. 5(2), pages 1-24, 2009.
9. U. Chakravarthy, J. Grant and J. Minker. Logic-Based Approach to Semantic Query Optimization. ACM Trans. Database Syst. 15(2) pages 162-207, 1990.
10. Q. Cheng, J. Gryz, F. Koo, T.Y. Cliff Leung, L. Liu, X. Qian and K. Schiefer. Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. In VLDB, pages 687-698, 1999.
11. S. Das, S. Sundara and R. Cyganiak. R2RML: RDB to RDF Mapping Language. W3C Working Draft 24 March 2011, http://www.w3.org/TR/r2rml/.
12. A. Gray, N. Gray and I. Ounis. Can RDB2RDF Tools Feasibily Expose Large Science Archives for Data Integration? In ESWC, pages 491-505, 2009.
13. B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. Commun. ACM, 50:94–101, May 2007.
14. S. Sahoo, W. Halb, S, Hellmann, K. Idehen, T. Thibodeau, S. Auer, J. Sequeda, A. Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. W3C RDB2RDF XG Report, 2009.
15. S. Tirmizi, J. Sequeda and D. Miranker. Translating SQL Applications to the Semantic Web. In DEXA, pages 450-464, 2008.