Web Semantics: Science, Services and Agents on the World Wide Web 22 (2013) 19–39

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Ultrawrap: SPARQL execution on relational data

Juan F. Sequeda *, Daniel P. Miranker

Department of Computer Science, University of Texas at Austin, United States

ARTICLE INFO

Article history:
Received 3 April 2012
Received in revised form
22 August 2013
Accepted 25 August 2013
Available online 2 September 2013

Keywords:
Semantic web
Relational databases
SPARQL
SQL
RDF
RDB2RDF

ABSTRACT

The Semantic Web's promise of web-wide data integration requires the inclusion of legacy relational databases,[1] i.e. the execution of SPARQL queries on RDF representation of the legacy relational data. We explore a hypothesis: existing commercial relational databases already subsume the algorithms and optimizations needed to support effective SPARQL execution on existing relationally stored data. The experiment is embodied in a system, *Ultrawrap*, that encodes a logical representation of the database as an RDF graph using SQL views and a simple syntactic translation of SPARQL queries to SQL queries on those views. Thus, in the course of executing a SPARQL query, the SQL optimizer uses the SQL views that represent a mapping of relational data to RDF, and optimizes its execution. In contrast, related research is predicated on incorporating optimizing transforms as part of the SPARQL to SQL translation, and/or executing some of the queries outside the underlying SQL environment.

Ultrawrap is evaluated using two existing benchmark suites that derive their RDF data from relational data through a Relational Database to RDF (RDB2RDF) Direct Mapping and repeated for each of the three major relational database management systems. Empirical analysis reveals two existing relational query optimizations that, if applied to the SQL produced from a simple syntactic translations of SPARQL queries (with bound predicate arguments) to SQL, consistently yield query execution time that is comparable to that of SQL queries written directly for the relational representation of the data. The analysis further reveals the two optimizations are not uniquely required to achieve a successful wrapper system. The evidence suggests effective wrappers will be those that are designed to complement the optimizer of the target database.

1. Introduction

We postulate that by carefully constructing unmaterialized SQL views[2] to create a logical representation of a legacy relational database as an RDF graph [2], the existing algorithmic machinery in SQL optimizers is already sufficient to effectively execute SPARQL queries [3] on native relational data [4,5]. Thereby, legacy relational database systems may be made upwardly compatible with the Semantic Web [6], while simultaneously minimizing the complexity of the wrapping system. This is in contrast to related efforts, detailed below, that are predicated on preprocessing and/or optimizing the SQL query before sending it to the SQL optimizer [7–9].

To clarify the focus of this research, consider the taxonomy in Fig. 1. In RDF data management there are efforts that concern Triplestores and those that concern legacy Relational Databases. Triplestores are database management systems whose data model is RDF, and support at least SPARQL execution against the stored contents. *Native triplestores* are those that are implemented from scratch [10–12]. *RDBMS-backed Triplestores* are built by adding an application layer to an existing relational database management system. Within that literature is a discourse concerning the best database schema, SPARQL to SQL query translations, indexing methods and even storage managers, (i.e. column stores vs. row stores) [13–16]. *NoSQL Triplestores* are also being investigated as possible RDF storage managers [17–19]. In all three cases, RDF is the primary data model.

The research herein is concerned with the mapping of legacy relational data with the Semantic Web, a.k.a Relational Database to RDF (RDB2RDF). Within that, the research concerns *Wrapper Systems* that present a logical RDF representation of relational data that is physically stored in an RDBMS such that no copy of the relational data is made. It follows that some or all of a SPARQL query evaluation is executed by the SQL engine. An alternative is the relational data is *extracted* from the relational database, *translated* to RDF, and *loaded* (*ETL*) into a triplestore [20].

Since both RDBMS-backed Triplestores and RDB2RDF Wrapper systems involve relational databases and translation from SPARQL

* Corresponding author.
E-mail addresses: jsequeda@cs.utexas.edu (J.F. Sequeda), miranker@cs.utexas.edu (D.P. Miranker).

[1] By legacy, we mean software/data already in wide use such that an organization is not willing to relinquish the investment.

[2] Unmaterialized views are virtual tables that are defined by a query over other tables in the database. They are not stored in the database but can be queried as if they existed [1].

1570-8268/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.websem.2013.08.002

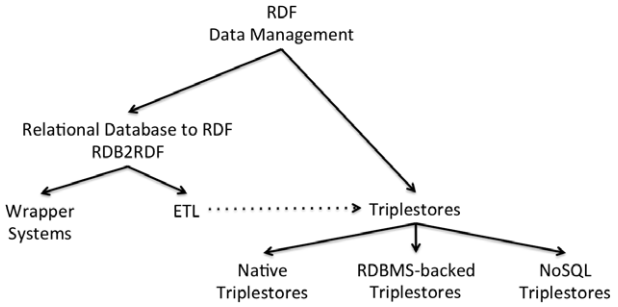

Fig. 1. Taxonomy of RDF data management.

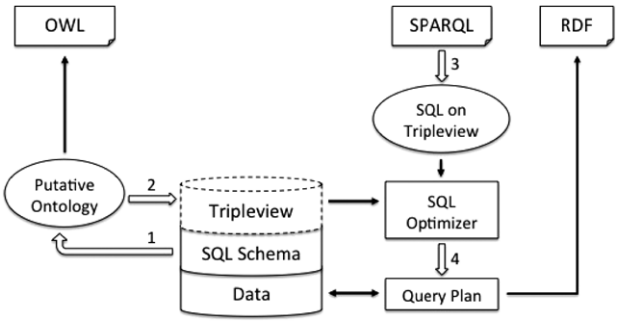

Fig. 2. Architecture of Ultrawrap.

to SQL, there is a potential for confusion. The difference is that RDBMS-backed Triplestores translate SPARQL queries to SQL queries that are executed on database schemas that model and store RDF. RDB2RDF Wrapper systems translate SPARQL queries to SQL queries that are executed on legacy database schemas that model and store relational data.

Approximately 70% of websites have relational database backends [21]. The sheer number of websites suggests the success of the Semantic Web is tied to maintaining compatibility and consistency with legacy RDBMSs. Wrapper systems enable Semantic Web applications to coexist with the legacy applications and avoid consistency problems simply by not creating a replicated copy of the data.

In 2008, Angles and Gutierrez showed that SPARQL is equivalent in expressive power to relational algebra [22]. Thus, one might expect the validity of this research's postulate to be a foregone conclusion. However, in 2009, two studies that evaluated three RDB2RDF wrapper systems, D2R, Virtuoso RDF Views and Squirrel RDF, came to the opposite conclusion; existing SPARQL to SQL translation systems do not compete with traditional relational databases [23,24].

A motivation for this paper is to resolve the apparent contradiction among the aforementioned papers. Toward that end we have built a system, Ultrawrap[3] [25]. Ultrawrap is organized as a set of four compilers with the understanding that the SQL optimizer forms one of the four compilers (Section 3 and Fig. 2).

In a two-step, off-line process, Ultrawrap defines a SQL view whose query component is a specification of a mapping from the relational data to an RDF triple representation, the *Tripleview*. In our experiments the Tripleview is not materialized, (i.e. the defining queries are not executed). Thus the view forms a logical specification of the mapping. Note that this view is extremely large, comprising a union of select-from-where queries, at least one query for each column in the relational database. At the onset of the research we first conducted experiments to confirm that such large view definitions would be parsed by RDBMSs without throwing an exception.

At runtime, a third compiler translates an incoming SPARQL query to a SQL query on the Tripleview. The translation is limited to macro-substitution of each logical SPARQL operator with its equivalent SQL operator. This is straightforward as each SPARQL query operator corresponds to an equivalent relational operator [22].

It follows from the SQL standard that an RDBMS must correctly execute the translated SPARQL query. Consequently, the target RDBMS' SQL system must both use the logical mapping represented in the Tripleview and optimize the resulting query, forming the fourth compiler.

Ultrawrap is evaluated using the three leading RDBMS systems and two benchmark suites, Microsoft SQL Server, IBM DB2 and Oracle RDBMS, and the Berlin and Barton SPARQL benchmarks (Section 5). The SPARQL benchmarks were chosen as a consequence of the fact that they derived their RDF content from a relational source. The Berlin Benchmark provides both SPARQL queries and SQL queries, where each query was derived independently from an English language specification. Since wrappers produce SQL from SPARQL we refer to the benchmark's SQL queries as *benchmark-provided SQL queries*. For Barton, the original relational data is not available and the creator of the benchmark did not create separate SPARQL and SQL queries. We located replacement relational data, namely a relational data dump of DBLP and created separate SPARQL and SQL queries derived independently from an English language specification. The benchmark-provided SQL queries have been tuned for use specifically against each benchmark database. We have packaged the new version of Barton for distribution [26].

By using benchmarks containing independently created SPARQL and SQL queries, and considering the effort and maturity embodied in the leading RDBMS's SQL optimizers, we suppose that the respective benchmark-provided SQL query execution time forms a worthy baseline, and the specific query plans to yield insight into methods for creating wrappers.

Our findings include:

- A mapping of relational data to a Tripleview comprising three columns does not instigate the SQL optimizers to use indexes. The view was refined to reflect physical schema properties (Section 3).
- Two known query optimizations, *detection of unsatisfiable conditions* and *self-join elimination* [27], when applied, not only result in comparable execution times between SPARQL and the benchmark-provided SQL queries with bound predicates, the optimizers will often produce identical query plans (Section 4).
- In some cases, a third optimizing transform, join predicate push down, can be as effective as the detection of unsatisfiable conditions (Section 5).
- SPARQL queries containing variables that bind to the predicate position remain troublesome. We relate this problem to an already described problem concerning the use of views in the implementation of data integration systems (Section 5).
- The impact of the self-join elimination optimization is a function of the selectivity and the number of properties in the SPARQL query that are co-located in a single table (Section 6).
- No system, including those that eliminated self equi-joins, eliminated the self left outer joins. The SPARQL optional operator is, by definition, a left outer join (Section 6).

By starting with a simple wrapper system and evaluating it with sophisticated SQL query optimizers we are able to identify existing, well understood optimization methods that enable wrappers. The results provide a foundation for identifying minimal requirements for effective wrapper systems.

[3] See acknowledgments.