Evaluating SPARQL-to-SQL translation in ontop

Mariano Rodriguez-Muro, Martin Rezk, Josef Hardi, Mindaugas Slusnys
Timea Bagosi and Diego Calvanese

KRDB Research Centre, Free University of Bozen-Bolzano
{rodriguez,mrezk, josef.hardi, mindaugas.slusnys,
timea.bagosi, calvanese}@inf.unibz.it

Abstract. In this paper we evaluate the performance of the SQL queries generated by ontop, a system that uses a formal approach to translate and optimize SPARQL queries and R2RML mappings. We show that the performance of ontop's SQL queries is superior to that of the performance of well known systems that rely on SQL to execute SPARQL, and superior to to that of well-known triple stores. We highlight some of the techniques and factors that allow for this.

1 Introduction

The integration of SPARQL and RDF with RDBMs is crucial for the adoption Semantic Technologies in industry. This importance is reflected in the creation of the R2RML [3] standard for mapping RDBMs into RDF, and in all the research focused towards translating SPARQL queries into efficient SQL over RDBs by means of mappings. This last topic is specially relevant since such techniques allow to use SPARQL and the RDF data model without costly ETL (i.e., of Extract Transform and Load from RDBs to RDF) processes, and may also allow to profit from the features of industrial strength RDBMS that are not available in triples stores, e.g., redundancy, robust transaction support, security, etc.

Execution of SPARQL with SQL is common. However, previous techniques often suffered from problems that limited their use in practice. Some techniques assume a fixed relational schema and do not support general mapping languages like R2RML, others generate complex SQL queries that do not perform well, last, some generate efficient SQL but use techniques that are not grounded formally and have limited scope. In this paper we evaluate ontop[1] [5,4], a system that tackles these issues.

ontop allows to query virtual RDF graphs defined by a relational DB and an R2RML mapping. The core of the query answering technique implemented ontop is depicted in Figure 1. In a first step (R2RML) mappings and SPARQL query are translated into a set of Datalog rules that capture the semantics of the execution of the SPARQL query over the original database. Second, the program is optimized using query containment based techniques and Semantic Query Optimization, in particular we:

[1] available at `ontop.inf.unibz.it` for Protege 4, OWLAPI, Sesame and stand-alone endpoint

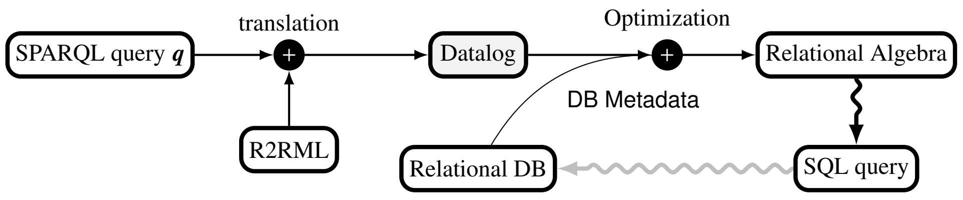

Fig. 1. Query answering with mappings in ontop

– use *SLD-resolution* to compute a *partial evaluation* of the program in which FILTER expressions and JOIN conditions are expressed in terms of the original database columns and not over the RDF terms constructed from these (e.g., consider that URI templates in R2RML mappings generate URI's on the fly). This will allows the DBMS to exploit any indexes defined on the original tables;
– optimize the query(ies) using Semantic Query Optimization (SQO) with respect to w.r.t. Primary Keys to avoid redundant self-joins.

Last, the optimized program is translated into an equivalent relational algebra expression, the SQL query is generated and executed by the DBMS. This rule based technique for SPARQL, R2RML and SQL provides a formal framework that gives clear guarantees w.r.t. to the semantics of the technique and which is extendible with more advanced optimizations (e.g., to other forms of constraints like Foreign Keys or Check constraints, etc.) and functionality, including OWL 2 entailment regimes (OWL 2 QL is already available in ontop and not discussed in here).

In this paper we show that the performance of the SQL queries generated by ontop using these techniques is superior to that of other systems that also perform SPARQL through SQL, and superior to that of well known triple stores.

2 Evaluation

This evaluation provides an overview of the performance of DB2 and MySQL while executing SQL queries generate by ontop. We use two benchmarks scenarios with a total of 36 queries and over 350 million triples. We considered two systems that offer similar functionality to ontop (i.e., SPARQL trough SQL and mappings): Virtuoso RDF Views 6.1 (open source edition) and D2RQ 0.8.1 Server over MySQL. We also compare performance with 3 well known triple stores, i.e., OWLIM 5.3, Stardog 1.2 and Virtuoso RDF 6.1 (Open Source). For ontop we used two different DB engines as backend, i.e., MySQL and DB2. The benchmarks used are:

BSBM The Berlin SPARQL Benchmark (BSBM) [2] evaluates the performance of query engines using use cases from e-commerce domain. The benchmark comes with a suite of tools for data generation and query execution. The benchmark also includes a relational version of the data, for which mappings can be created (D2RQ mappings are included).

FishMark The FishMark benchmark [1] is a benchmark for RDB-to-RDF systems that is based on an extract of the FishBase DB, a publicly available database about fish species. The benchmark comes with an extract of the database (approx. 16 M triples in RDF and SQL version), and 22 SPARQL queries obtained from the logs of FishBase. The queries are substantially larger (max 25 atoms, mean 10) than those in BSBM. Also, they make extensive use of OPTIONAL graph patterns.

The basic setup for the experiment is as follows: In the case of BSBM, out of the 12 query templates of BSBM (i.e., queries with place holders for constant values) a predefined sequence of 25 of these templates constitutes a *Query Mix*; then a BSBM run is the instantiation of a query mix with random constants and execution of the resulting queries. Performance is then measured in Query Mixes per Hour (QMpH), to compute QMpH we ran 150 query mixes, out of which 50 are considered warm up runs and their statistics are discarded. The collected statistics for QMpH over BSBM instances with 25, 100 and 200 million triples (or the equivalent in relational form). In the case of FishMark, the 22 queries are already instantiated and they constitute the query mix. We ran 150 query mixes, discarding the initial 50. In both cases, we tested with 1, 4, 8, 16 and 64 simultaneous clients.

In order to only get the performance of *SQL* queries generated by ontop we exploited ontop's simple SQL caching mechanism which stores SQL queries generated for any SPARQL query that has been rewritten previously. This allows to avoid the rewritten process completely and hence, the cost of query execution of a cached query is only the cost of evaluating the SQL query over the DBMS. To force the use of this cache, we re-ran the BSBM benchmark 5 more times (and averaged the results). For FishMark, this was not necessary since the queries are always the same. All experiments were conducted on a HP Proliant server with 24 Intel Xeon CPUs (144 cores @3.47GHz), 106GB of RAM and a 1TB 15K RPM HD. The OS is Ubuntu 12.04 64-bit edition. All the systems run as SPARQL end-points. All configuration files are available online[2]. The results are summarized in Figure 2.

Discussion. First we note that the D2RQ server always ran out of memory, timed out in some queries or crashed. This is why it doesn't appear in our summary table. D2RQ's SPARQL-to-SQL technique is not well documented, however, by monitoring the queries being sent by D2RQ to MySQL, it appears that D2RQ doesn't translate the SPARQL query into a single SQL query, instead it computes multiple queries and retrieves part of the data from the database. We conjecture that D2RQ then uses this data to compute the results. Such approach is, in general, limited in scalability and prone to large memory consumption, being the last point the reason for the observed behavior. Also, Virtuoso Views is not included in the FishMark benchmark because it provided wrong results, we reported this to the developers which confirmed the issue. Also, we did not run ontop with DB2 for FishMark due to errors during data loading.

Next, we can see is that for BSBM in almost every case, the performance obtained with ontop's SQL queries executed by MySQL or DB2 outperforms all other systems by a large margin. The only cases in which this doesn't hold are when the number of

[2] https://babbage.inf.unibz.it/trac/obdapublic/wiki/BSBMFISH13aBench

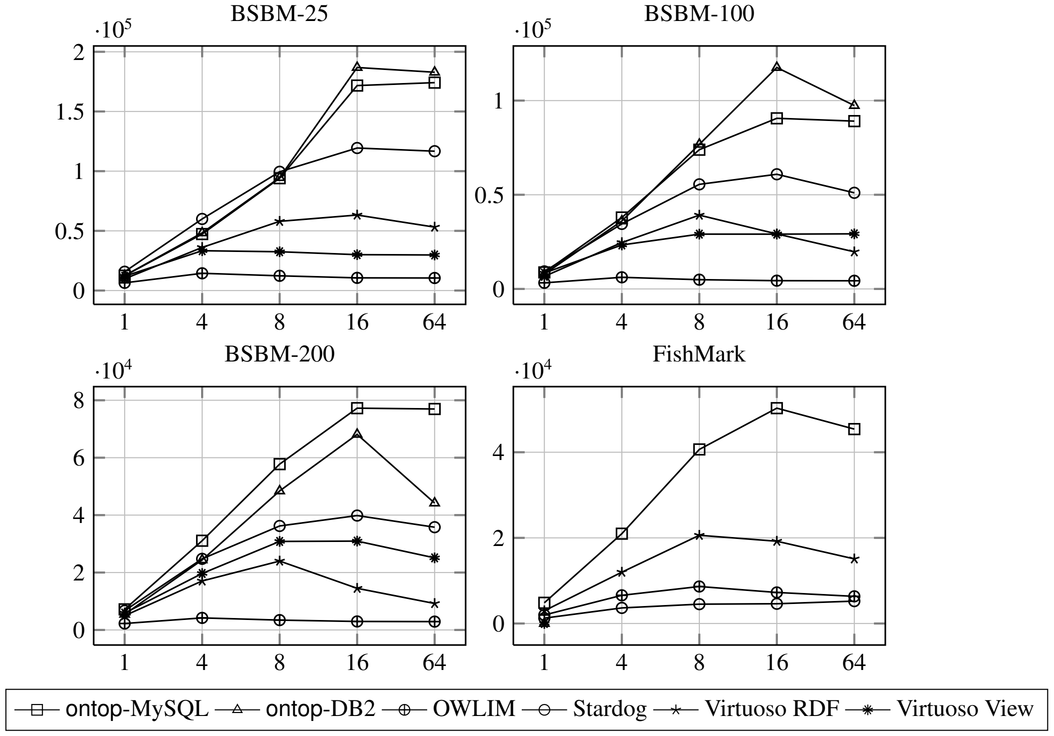

Fig. 2. Query performance comparison summary. X axis = parallel clients, Y axis = Query Mixes per Hour (QMpH, higher is better)

clients is less than 16 and the dataset is small (BSBM 25). This can be explained as follows.

Note that in ontop performance can be divided in three parts, *(i)* the cost of generating the SQL query , *(ii)* the cost of execution over the RDBMs and *(iii)* cost of fetching and transforming the SQL results into RDF terms. When the queries are cached, *(i)* is absent, and if the scenario includes little data (i.e., BSBM 25), the cost of *(ii)*, both for MySQL and DB2, is very low and hence *(iii)* dominates. We attribute the performance difference to a poor implementation of *(iii)* in ontop, and the fact triple stores do not need to perform this step.

From 16 parallel clients however, executing ontop's SQL queries with MySQL or DB2 outperforms other systems by a large margin. We attribute this to DB2's and MySQL's better handling of parallel execution (i.e., better transaction handling, table locking, I/O, caching, etc.).

When the datasets are larger, e.g., BSBM 100 and 200, for ontop *(i)* stays the same. From *(ii)* and *(iii)* we have that the former dominates since in both benchmarks queries return few results. Then, again, we have that MySQL's an DB2's advantage over triple stores can be attributed to similar reasons as before, better I/O, planning, caching, etc.

This is witnessed by the fact that DB2 and MySQL with ontop's SQL outperform the rest already at 1 single client for BSBM 100 and BSBM 200.

These experiments do not allow to fully see the benefit of the optimizations on SQL that is performed by ontop; this would require to be able to enable and disable them, and in ontop this is not possible at the moment. However, some observations are possible, in particular we can see the strong effect of SELF JOIN elimination by Primary Keys. Consider the FishMark benchmark that has little data, only 16M triples, but in which in almost all queries ontop's SQL executed over MySQL (we didn't run DB2 in this case) outperforms the rest almost in every case even from 1 single client. In this setting, 1 client and little data, the cost of *(ii)* falls in the cost of planning and executing JOINs and LEFT JOINs by the DBMS or triple store. At the same time, in FishMark, the original tables are structured in such a way that many of the SPARQL JOINs can be simplified dramatically when expressed as optimized SQL. For example, consider the FishMark query:

```
SELECT ?order ?family ?genus ?species ?occ ?name ?gameref ?game
WHERE {
 ?ID fd:cComName ?name; fd:coC_Code ?ccode; fd:cSpecCode ?x.
 ?x  fd:sGenus ?genus; fd:sSpecies ?species; fd:sGameFish ?game;
     fd:sGameRef ?gameref; fd:sFamCode ?f .
 ?f  fd:fFamily ?family; fd:fOrder ?order .
 ?c  fd:cSpecCode ?x; fd:cStatus ?occ; fd:cC_Code ?cf;
     fd:cGame 1 . ?cf fd:cPAESE "Indonesia" . }
```

This query expresses a total of 16 Join operations. When translated into SQL, ontop is able to generate the following query:

```
SELECT V3.FamilyOrder AS order, V3.Family AS family,
 V1.Genus AS genus, V1.Species AS species, V4.Status AS occ,
 V1.ComName AS name, V1.GameRef AS gameref, V1.GameFish AS game
FROM species V1, comnames V2, families V3, country V4, countref V5
WHERE V1.SpecCode = V2.SpecCode AND V4.Game = 1 AND V5.PAESE =
 'Indonesia' AND V4.C_Code = V5.C_Code AND V1.Genus = V8.Genus
```

A simple and flat SQL query (easy to execute) with a total of 3 Joins. Note that the use of a large number of JOIN operations is intrinsic to SPARQL since the RDF data model is ternary. However, if the data is stored in a n-ary schema (as usual in RDBMs), ontop can use semantic query optimization w.r.t. primary keys to construct the optimal query over the n-ary tables. Triple stores has no means to do this since data is de-normalized once it is transformed into RDF.

In BSBM this optimization is weaker since queries are smaller and have fewer joins, however a trend pointing to this same observation can be seen. Consider the results for Q2, Q3 and Q4 in Table 2 Q2, Q3 and Q4.

Conclusions. In this evaluation we confirmed that in BSBM and FishMark, the SQL queries generated by ontop (executed by 2 representative RDBMS) can provide much better performance than that of executing SPARQL queries executed over similar SQL-based systems (D2RQ, Virtuoso Views) and well known triple stores. This is a strong

	ontop-MySQL	ontop-DB2	OWLIM	Stardog	V. RDF	V. Views
Q1	3,22k	1,28 k	1,43k	1,42	23	45
Q2	1,40k	928	276	790	123	315
Q3	1,92k	1,12k	111	543	155	341
Q4	1,53k	955	295	669	140	265
Q5	27	72	2	29	14	48
Q6	-	-	-	-	-	-
Q7	9,78k	1,59k	541	400	101	316
Q8	13,99k	1,68k	3,22k	648	96	180
Q9	13,62k	1,01k	3,7k	1,99k	59	188
Q10	20,22k	2,04k	3,9k	610	228	305
Q11	20,75k	2,04k	3,52k	1,89k	1,66k	1,13k
Q12	11,34k	1,64k	5,99k	1,4k	1,25k	1,19k
QueryMix	44,19k	76,96k	2,91k	35,79k	9,21k	25,11k

Fig. 3. Summary of results (per query) for BSBM-200 with 64 clients. Individual queries are in *queries per second*, totals are in *query mixes per hour*

pointer that on-the-fly SPARQL-to-SQL translation might be a much better option than the ETL approach in some use cases, e.g., when the data is already stored in a relational schema and the SPARQL queries do not use advanced SPARQL features (i.e., regular paths).

Several points were neither discussed nor evaluated in this paper. For example, the cost of generating the SQL queries is at the moment very high in ontop; in our experiments we observed a dramatic performance drop when considering the SQL generation process, often putting ontop at the same level of performance than the worst of the other systems. This observation calls for several points of action. First, we need to understand how much of this cost can be attributed to poor implementation (ontop is an academic system after all). Second, the template based queries involved in BSBM and FishMark can be considered a common use case, e.g., applications often repeat the same queries over and over, only variating some constants. With this in mind, a *prepared-statement like approach* or a more intelligent caching mechanism for query rewritings could be devised so that the cost of SQL generation is removed.

Last, we note the choice of the optimizations and techniques implemented in ontop is not arbitrary, they are guided by empirical experiences on what constitute efficient SQL on a wide range of modern RDBMs engines. The evaluation presented here shows that those choices, as a whole, seem to be *on the right track*; however, it doesn't provide a deep understanding of the individual benefit of each of these choices and optimizations and further work in this direction is required.

References

1. Samantha Bail, Sandra Alkiviadous, Bijan Parsia, David Workman, Mark van Harmelen, Rafael S. GonÃSalves, and Cristina Garilao. FishMark: A linked data application benchmark. In *Proc. of the Joint Workshop on Scalable and High-Performance Semantic Web Systems*

(SSWS+HPCSW 2012), volume 943, pages 1–15. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/`, 2012.

2. Christian Bizer and Andreas Schultz. The Berlin SPARQL benchmark. *Int. Journal On Semantic Web and Information Systems*, 5(2):1–24, 2009.
3. Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF mapping language. http://www.w3.org/TR/r2rml/, September 2012.
4. Mariano Rodriguez-Muro and Diego Calvanese. Quest, an owl 2 ql reasoner for ontology-based data access. In *Proc. of the 9th Int. Workshop on OWL: Experiences and Directions (OWLED 2012)*, volume 849 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, 2012.
5. Mariano Rodriguez-Muro, Josef Hardi, and Diego Calvanese. Quest: Effcient sparql-to-sql for rdf and owl. In *Proc. of the ISWC 2012 Posters Demonstrations Track (ISWC-PD 2012)*, volume 914 of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, 2012.