Quest: Efficient SPARQL-to-SQL for RDF and OWL

Mariano Rodríguez-Muro, Josef Hardi and Diego Calvanese

KRDB Research Centre, Free University of Bozen-Bolzano
{rodriguez,josef,calvanese}@inf.unibz.it

Motivation. One of the most important uses of semantic technology is that of Ontology Based Data Access (OBDA), where the objective is to use shared vocabularies and ontologies as means to access data living in possibly disperse and heterogenous data sources (e.g., relational DBMS, XML databases, spreadsheets, etc.) Today this task often involves an ETL process in which the data is (E)xtracted from the source, (T)ransformed into RDF or OWL datasets in the target vocabulary and (L)oaded into a SPARQL end-point or an OWL reasoner. This process carries all the issues that come with e.g., the need for synchronisation mechanisms to keep data up-to-date, the extra cost in time and space due to the duplication process, the additional software complexity at the client side, etc. Often it would be better to have live access to the original sources to avoid these issues and to be able to exploit any kind of optimisations that the original source can offer. In the context of relational DBMS and SPARQL queries, there exist several systems that allow for this on-the-fly approach, e.g., the D2RQ engine, Triplify, Spyder, Virtuoso RDF views, etc. However, often these systems fall short either in support for semantics (entailment regimes) and/or in query answering performance, e.g., the systems may send multiple queries to the sources and perform operations in-memory or they may generate complex SQL queries that cannot be planned and executed efficiently by the DBMS. This reality forces the use of the ETL approach, sometimes even in use cases in which an on-the-fly approach would be evidently possible.

Quest. In this demo we introduce Quest [7], a new system that provides SPARQL query answering with support for OWL 2 QL and RDFS. Quest allows to link the vocabulary of an ontology to the content of a relational database through *mapping axioms*. These are used together with the ontology to answer a SPARQL query by means of a single SQL query that is then executed over the database. Quest uses highly-optimised query rewriting techniques to generate the SQL query which not only takes into account the entailments of the ontology and data, but is also 'lean' and simple so that it can be executed efficiently by any SQL engine. Quest supports commercial and open source databases, including database federation tools like Teiid to allow for Ontology Based Data Integration of relational and other sources (e.g., CSV, Excel, XML). Now we will briefly describe Quest's mapping language, the query answering process and the most relevant optimisation techniques used by the system. We will conclude with a brief description of the content of this demo.

Quest Mappings. Quest offers a powerful mapping language that often compensates for the lack of expressivity of RDFS and OWL 2 QL and that enables use cases that are often tackled with OWL 2 EL, OWL 2 RL or SWRL. Mappings axioms are the means to relate the content of a database to the vocabulary of an ontology. Quest's mappings have their formal foundation on GAV-sound mappings as defined in [4], however, extended to support all the features required from mapping language oriented towards the Semantic Web, e.g., patterns for URI construction, OWL/RDF datatypes, class and property mappings, URI and data constants in the mappings, etc. The result is a language that is equivalent in expressivity to the new W3C recommendation R2RML, but with a softer learning curve due to its simple and flexible Turtle-based syntax.

During the demo we will show the most relevant features of this language by means of a scenario based on the SQL version of IMDB and the Movie Ontology (MO) [3]. In the scenario, we want to query IMDB's data using SPARQL and MO's vocabulary and semantics. The IMDB database is very large and an ETL approach would be costly, i.e., 21 tables containing 43 million rows that generate approx. 42 million triples. Instead, we show how using Quest such scenario is realizable efficiently in an on-the-fly manner using query rewriting and mappings as the following:

```
tgt imdb:movie/{$id} a Movie; title $title;
       dbpedia:productionStartYear $production_year^^xsd:integer .
src SELECT id, title, production_year FROM title
       WHERE kind_id = 1

tgt imdb:movie/{$t.id} belongsToGenre Romance.
src SELECT t.id FROM title t, movie_info m WHERE t.id = m.movie_id
      AND m.info_type_id = 3 and m.info='Romance'
```

In these two mappings we can see that a mapping is composed by two parts, the *source* (**src**), a SQL query that brings some data from the database and a *target* (**tgt**), a template that, intuitively, indicates how to create ABox assertions/data-triples by replacing the column reference in the target with the actual values returned by the source query (in a similar fashion to CONSTRUCT queries in SPARQL). Note that the language offers complete freedom on the SQL queries, the templates, as well as in the way in which the object URIs are constructed. Instances declared in the ontology (e.g., `Romance`) can also be used in mappings, giving the possibility of hybrid data graphs, in which the dynamic and larger part of the data lives in the database and a static and small part is given in the ontology (e.g., in our scenario the ontology states 'Romance rdf:type Love'). Last, the familiar Turtle-based syntax for mappings exposes the semantics of the mappings to the user in a simple way, and doesn't contain any implementation specific features that are often exposed in languages like D2R and R2RML.

SPARQL query answering in Quest. The main service in Quest is SPARQL query answering by means of SQL query rewriting. During query answering Quest will take into account the semantics of the ontology vocabulary (defined in OWL 2 QL or RDFS axioms) as well as the mappings and the metadata of the database. The result of this process is one single SQL query that is executed over the original source. For example, consider the following SPARQL query over the MO ontology asking for

"the name of all directors that are also actors in the cast of a movie directed by themselfs such that the movie was produced between the years 2000 and 2001 in Eastern Asia, the movie is a romantic movie and has a user rating higher than 7"

In our use case, we would be able to get this information using the following SPARQL query:

```
SELECT $name $title ?rating WHERE
{ $m a Movie; title ?title; hasActor ?x; hasDirector ?x;
  isProducedBy $y; belongsToGenre $z; dbpedia:rating ?rating;
  dbpedia:productionStartYear ?year.
  $x dbpedia:birthName $name .
  $y hasCompanyLocation [ a Eastern_Asia ] .
  $z a mo:Love .
FILTER (?rating > 7.0 && ?year >= 2000 && ?year <= 2001) }
```

given the axioms in MO and the mappings to the IMDB SQL database Quest would compute the following SQL query:

```
SELECT V8.name AS name, V0.title AS title, V7.info AS rating
FROM title V0, cast_info V1, cast_info V3, company_name V4,
 company_name V4, movie_companies V5, movie_info V6,
 movie_info_idx V7, name V8, movie_companies V9, movie_info V10
WHERE V0.id = V1.movie_id        AND V0.id = V3.movie_id
 AND V0.id = V5.movie_id         AND V0.id = V6.movie_id
 AND V0.id = V7.movie_id         AND V0.id = V10.movie_id
 AND V4.id = V5.company_id       AND V4.id = V9.company_id
 AND V1.person_id = V3.person_id AND V1.person_id = V8.id
 AND V0.kind_id = 1              AND V3.role_id = 8
 AND (V1.role_id = 1 OR V1.role_id = 2)
 AND (V4.country_code = '[cn]' OR V4.country_code = '[jp]')
 AND V5.company_type_id = 2    AND V6.info_type_id = 3
 AND V7.info_type_id = 101     AND V9.company_type_id = 2
 AND V10.info_type_id = 3      AND V10.info = 'Romance'
 AND V7.info > '7.0'           AND V0.production_year >= 2000
 AND V0.production_year <= 2001
```

note that in MO there is a class (resp. property) hierarchy bellow the class Easter_Asia (res.p the property hasActor) and hence, Quest includes appropriate OR statements for the values of some of the columns to account for these semantics and in accordance to the mappings of the system.

Inspecting the resulting SQL query some of the benefits of using Quest can be noted. First, as with other OBDA systems, posing the query in terms of the ontology vocabulary is more natural, and follows almost the same logic as the query in natural language. Learning to formulate this query takes a matter of minutes after a fast inspection of the vocabulary, however, formulating such an SQL query for the IMDB database requires a considerable effort since the schema is very complex. With Quest the complexity of the underalying database is hidden and handled by the system. Second, the SQL query generated by Quest is very close to, and often coincides with the query that would be created by an SQL expert. In general, this is not the case in other OBDA/RDB2RDF systems which often issue not one query to the DBMS, but multiple queries that bring data back and forth to compute the answer localy., or in few more advanced system where a single SQL query is achieved, the query is often too complex to be executed efficiently or does not take into account the semantics of the ontology. These are not issues in Quest which is often able to offer dramatic performance improvements w.r.t. to other systems (e.g., on hardware found in an average laptop the previous query takes 1.5s in Quest, while D2RQ cannot return an answer after 5 minutes).

Optimisation. In order to generate these queries Quest resources to many optimisation techniques at different levels of the query answering process. Some of these techniques are aimed at efficiently handling the semantics of the ontology while others at generating optimal SQL. Here we briefly mention the major ones. During system initialisation, Quest will optimise both TBox and mappings. With respect to the mappings, Quest will first parse the SQL queries in the mappings and analyse the database metadata, extracting constraints (e.g., Primary Keys, etc.) and using this information, together with *query containment* checks to optimise the mappings (see T-mappings [5]). The TBox will also be optimised, eliminating redundant vocabulary and detecting any redundancy w.r.t. the mappings (see [6]). At query time, the system will go through 4 key steps: (*i*) SPARQL-to-Datalog translation, (*ii*) query rewriting w.r.t. the TBox, (*iii*) computation of a *partial evaluation* based on the mappings and finally (*iv*) SQL generation and execution. At each step, Quest will resource to query containment-based optimisations to eliminated redundant queries or optimise individual queries (e.g. eliminating redundant joins). As can be intuited, our optimisation theory relies heavily on our ability to analyse SQL to understand its semantics and on query containment for redundancy detection; because of this, as our implementation of the related algorithms improve so will the quality of the SQL generated by the system.

Content of the Demo. We will present the major features of Quest using the full version of the IMDB scenario introduced here. We will show how mappings can be created using our plugin for Protege, **-ontopPro-**, and we will query Quest through the same interface. We will discuss the features of the mapping language as well as the optimisations that Quest performs at the different stages of query answering. Queries will be run on top of a local copy of the IMDB database running on a PostgreSQL server, allowing us to show the performance of the system on inexpensive hardware and to compare it to other similar plattforms. Last, we will show how using a data federation tool like Teiid, we can integrate multiple and heterogeneous sources using the same techniques. Quest is part of the **-ontop-** framework for Ontology Based Data Access (OBDA) and can be downloaded from its website [2]. A video with a preview of the demo is available online [1].

References

1. **-ontop-** demo. http://obda.inf.unibz.it/ontop/iswc12/.
2. **-ontop-** website. http://obda.inf.unibz.it/protege-plugin/.
3. A. Bouza. MO the movie ontology, 2010. [Online; 26. Jan. 2010].
4. M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
5. M. Rodriguez-Muro and D. Calvanese. Dependencies: Making ontology based data access work in practice. In *Proc. of AMW 2011*, 2011.
6. M. Rodriguez-Muro and D. Calvanese. High performance query answering over dl-lite ontologies. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 308–318, 2012.
7. M. Rodriguez-Muro and D. 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.