Comparing Query Answering in OBDA Tools over W3C-Compliant Specifications

Manuel Namici and Giuseppe De Giacomo

DIAG, Sapienza, University of Rome
lastname@diag.uniroma1.it

Abstract. The Ontology-Based Data Access (OBDA) paradigm aims at providing to the users a unified and shared conceptual view of the domain of interest (ontology), while still enabling the data to be stored in different data sources. Such data are mapped to the ontology through declarative specifications. In this work we consider the ontology expressed in OWL 2 QL, relational sources, the mapping expressed in R2RML, and the user queries expressed in SPARQL. In this W3C-compliant setting, we compare query answering in the two main tools for OBDA, namely, Mastro and Ontop, by resorting to the NPD Benchmark, and the full-fledged OBDA application developed for the Italian Automobile Club (ACI). We also discuss how R2RML support is added to Mastro.

1 Introduction

The Ontology-Based Data Access (OBDA) paradigm [10] aims at providing to the users of the system a unified and shared conceptual view of the domain of interest (ontology), while still enabling the data to be stored in different data sources, which are managed by a relational database. In an OBDA system the link between the data stored at the sources and the ontology is provided through a declarative specification given in terms of a set of mappings.

The interest in the adoption of the OBDA paradigm has lead to the creation of prototype tools, that have evolved into full-fledged systems. While these tools effectively enable to answer queries over the ontology, their use in industrial applications still represents a challenge due to the performance requirements that have to be met.

In this work we focus on comparing two systems for OBDA, namely, Mastro[1] and Ontop[2]. To do so, we first add support for R2RML mappings in Mastro, a tool for Ontology-Based Data Access developed at Sapienza, University of Rome. R2RML is the W3C recommendation for expressing mappings in an OBDA specification, and was a missing feature in order to enable Mastro to use a completely-standard specification. We then proceed in performing a comparison between these systems over two OBDA specifications: (i) The NPD Benchmark [6,3,5] based on the Norwegian Petroleum Directorate (NPD) use-case

[1] https://www.obdasystems.com/mastro
[2] https://ontop.inf.unibz.it/

adopted in the Optique Project[3], and adapted for benchmarking purposes in the OBDA setting, that is available online[4]. (ii) The ACI application that is currently in development between Sapienza University of Rome and the Italian Automobile Club (ACI).

The rest of this work is organized as follows: In Section 2, we give an overview of the systems that we have considered, Mastro and Ontop. In Section 3 we briefly describe how support for the standard R2RML mapping language has been integrated in Mastro. In Section 4, we discuss the comparison over the NPD Benchmark, a specification developed by the University of Oslo, and adapted for its use as a benchmark in the OBDA setting [6,3,5]. In Section 5, we discuss the comparison over an application of the OBDA paradigm, developed in collaboration between Sapienza University of Rome and the Italian Automobile Club (ACI), that is used to evaluate the benefits of the OBDA approach in a real industrial setting.

2 Overview of the systems

Mastro is a Java tool for Ontology-Based Data Access, developed by OBDA Systems[5] and Sapienza University of Rome. The theoretical foundations underpinning the system are those described in [1,2,10].

Ontologies in Mastro are specified in a logic of the *DL-Lite* family of lightweight DLs, that is the logic underpinning the OWL 2 QL profile of the standard OWL 2 language. The system is equipped with a module that enables ontologies expressed in the OWL 2 language to be approximated in the fragment of *DL-Lite* supported by Mastro, using the semantic approach presented in [4]. The query language supported by Mastro is a subset of the SPARQL 1.0, corresponding to the class of (union of) conjunctive queries.

The mappings in Mastro are expressed in an internal format, written in XML syntax. In addition, through the course of this work, we adapted the system to support most of the R2RML standard as an alternative mapping language. The set of mappings in Mastro can be represented as a triple $\langle \mathcal{M}_o, \mathcal{M}_v, \Sigma \rangle$, where:

- $\mathcal{M}_v$ constitutes the set of so-called *view predicate mappings*. Each assertion in $\mathcal{M}_v$ has the following form:

$$q_{DB}(\boldsymbol{x}) \rightsquigarrow v(\boldsymbol{x})$$

  where $q_{DB}(\boldsymbol{x})$ is a query over the alphabet of the data sources (i.e. an SQL query over the source database), whose free variables are in $\boldsymbol{x}$, and $v(\boldsymbol{x})$ is a view predicate (not in the ontology alphabet), whose free variables are from $\boldsymbol{x}$.

[3] `http://www.optique-project.eu/`
[4] `https://github.com/ontop/npd-benchmark`
[5] `http://www.obdasystems.com`

– $\mathcal{M}_o$ constitutes the set of so-called *ontology predicate mappings*, that associates atomic predicates in the ontology alphabet to conjunctive queries over the alphabet of the views [9]. Assertions in M_o have the following form:

$$q_v(\boldsymbol{x}) \rightsquigarrow C(f(\boldsymbol{y})) \tag{1}$$
$$q_v(\boldsymbol{x}) \rightsquigarrow P(f_1(\boldsymbol{y_1}), f_2(\boldsymbol{y_2})) \tag{2}$$
$$q_v(\boldsymbol{x}) \rightsquigarrow A(f(\boldsymbol{y_1}), y_2) \tag{3}$$

Assertions of the form (1) are called *concept mapping assertions*, where $q_v(\boldsymbol{x})$ is a conjunctive query over the views in $\mathcal{M}_v$, with free variables $\boldsymbol{x}$, $\boldsymbol{y} \subseteq \boldsymbol{x}$, f is a function term and C is an atomic concept in the ontology.
Assertions of the form (2) are called *role mapping assertions*, where $q_v(\boldsymbol{x})$ is a conjunctive query over the views in $\mathcal{M}_v$, with free variables $\boldsymbol{x}$, $\boldsymbol{y_1}, \boldsymbol{y_2} \subseteq \boldsymbol{x}$, f_1 and f_2 are function symbols used to build objects out of the values stored at the data layer, and P is an atomic predicate in the ontology.
Assertions of the form (3) are called *attribute mapping assertions*, where $q_v(\boldsymbol{x})$ is a conjunctive query over the views in $\mathcal{M}_v$, with free variables $\boldsymbol{x}$, $\boldsymbol{y_1} \subseteq \boldsymbol{x}$, $y_2 \in \boldsymbol{x}$, f is a function term and A is an atomic attribute in the ontology.
Furthermore, the conjunctive queries in the left-hand side of the mappings can also express conditions over the variables in the mapping, in the form of (in)equalities, and relational operators.

– Σ is a set of *data constraints* over the view predicates. Data constraints are used during query reformulation in order to reduce the size of the final rewriting by applying techniques that take advantage of the semantics of the views expressed through the constraints. Such techniques are usually referred to as *semantic query optimization* (SQO). Assertions in Σ are divided into:
  - Key Constraints, which are assertions of the form:

$$Key(v[i_1, ..., i_k])$$

    where v is a view predicate, and $i_1, ..., i_k$ is a set of pairwise distinct integers ranging from 1 to the arity of v. Intuitively, these assertions state that the subset of the attributes of v corresponding to the integers $i_1, ..., i_k$ is an identifier of its tuples. These constraints correspond to the notion of primary keys in relational databases.
  - Inclusion Constraints, which are assertions of the form:

$$v1[i_1, ..., i_k] \subseteq v2[j_1, ..., j_k]$$

    where $v1, v2$ are two distinct view predicates, $i_1, ..., i_k$ is a set of pairwise distinct integers ranging from 1 to the arity of $v1$, and $j_1, ..., j_k$ is a set of pairwise distinct integers ranging from 1 to the arity of $v2$. These assertions state that the projection of the view $v1$ over the attributes corresponding to the indexes $i_1, ..., i_k$ is contained in the projection of the view $v2$ over the attributes corresponding to the indexes $j_1, ..., j_k$.

- Denial Constraints, which are assertions of the form:

$$CQ_{M_v} \rightarrow \bot$$

where CQ_{M_v} is a conjuntive query over the view predicates in M_v. These assertions are used to state the query CQ_{M_v} evaluates to an empty set and can be used to remove entire sub-queries from the final union of conjunctive queries.

Query answering for UCQ in Mastro is divided into four phases: (i) The first phase comprises the rewriting of the input UCQ according to the knowledge expressed by the TBox of the ontology. The rewriting process, performed by the Presto [13] algorithm, encodes the knowledge expressed by the TBox and the user query q into a UCQ q' so that the evaluation of q' produces the same results of q without taking into account axioms in the TBox. (ii) During the second phase the UCQ over the ontology is rewritten, taking into account mapping from $\mathcal{M}_o$, through a mapping *unfolding* step, based on the *partial evaluation* techniques described in [10], in order to deal with the presence of function symbols. The output of this phase is a UCQ expressed over the alphabet of the views in $\mathcal{M}_v$. This phase is referred to in Mastro as *high-level unfolding*. (iii) In the third stage of the query reformulation, *optimizations* are performed, taking into account inclusions between views, denial constraints and key constraints, so as to minimize the reformulated query by removing redundant joins or entire sub-queries contained into others. (iv) Finally, during the last stage of the query reformulation the optimized UCQ is rewritten taking into account mappings from $\mathcal{M}_v$ through a further *unfolding* step, so as to obtain an SQL query that can be directly evaluated over the input database. This last phase is referred to in Mastro as *low-level unfolding*.

The entire reformulation process in Mastro is implemented in a parallel approach, that is, each of the rewritings of the input UCQ is treated indipendently in the reformulation pipeline, and the union of all the answers is performed as the they are available to the system.

The other system the we considered is Ontop[6]. Ontop is an open source OBDA framework, written in Java, developed at the Free University of Bolzen-Bolzano and released under the terms of the Apache version 2.0 license.

Ontop is based on theoretical foundations from [1,10,11] and supports all the relevant W3C standards and major relational databases. Ontop supports OWL 2 QL and RDFS as the ontology languages, SPARQL 1.0 and the OWL 2 QL entailment regime of SPARQL 1.1 for the user queries, and R2RML as the mapping language. Additionally Ontop has its internal mapping language, in which mappings consist of (i) a *source*, which is an SQL query; (ii) a *target*, which is an RDF triple pattern, with placeholders from the attributes of the source query.

In Ontop, the query answering engine Quest transforms SPARQL queries over the virtual RDF graph into SQL queries, which are executed by the relational

[6] `http://ontop.inf.unibz.it`

database engine storing the data. The workflow of the translation is divided into two stages: The first stage happens during the system initialization, in which the information contained at the intentional level of the ontology is integrated into the mappings, generating the so-called $\mathcal{T}$-*mappings* [11]. The second stage happens during query execution, where the input SPARQL query is decomposed into a tree structure (referred to as the SPARQL algebra tree) and each node is translated into an SQL expression. This second stage makes makes use of the $\mathcal{T}$-mappings generated at system start-up and of the database integrity constraints. During the translation of this intermediate representation of the query into SQL, optimizations are applied in order to avoid redundant self-joins, sub-queries, and joins over complex expressions [12].

3 Adding R2RML Support to Mastro

Support for R2RML mappings in Mastro has been added through a module that translates from/to its internal format to/from a set of mappings expressed in the R2RML syntax. The process of translation of an R2RML mapping into the Mastro internal format is pretty straightforward: We associate to each different logical table in the original R2RML mapping a new view predicate mapping, where the right-hand side corresponds to the effective SQL query of the logical table, and for each mapping that associates this logical table to a predicate over the ontology, we create a new ontology predicate mapping, associating to it the corresponding view predicate. Since R2RML does not allow to express constraints over the logical tables of the form shown in 2, no data constraint is generated for the translated mapping. This means that effectively, when using R2RML as the mapping language, the system is not able to perform any optimization on the final rewriting, as this would require the specification of such constraints by the mapping designer.

On the other hand, when translating from its internal format to the R2RML format, we face the problem of how to represent mappings expressed as conjunctive queries over the alphabet of the view predicates. The solution we adopted is to take the unfolding of the right-hand side of such mappings, and associate this unfolding to a new logical table that can be used to construct the corresponding R2RML representation of the original mapping. Also, similarly to what happens in the opposite direction, the data constraints expressed over the view predicates are discarded during the translation, as they have no equivalent in R2RML.

The approach here presented has a pretty notable consequence: When using R2RML as the mapping language, Mastro is not able to perform any (other than very basic) optimizations over the produced rewriting, as it cannot take advantage of the SQO module, which requires data constraints to be explicitly stated in the mappings.

4 Comparison on the NPD Benchmark

The NPD Benchmark is a recent proposal [6,3,5], based on the *Norvegian Petroleum Directorate (NPD) FactPages* [7], for the purpose of the evaluation of the performance of an OBDA system in a real world scenario.

The NPD FactPages contains information regarding the petroleum activities on the Norwegian continental shelf. Such information is actively used by oil companies like Statoil. The Factpages are periodically synchronized with the NPD's databases.

The NPD FactPages have been mapped to the ontology and stored in a relational database. Together with the ontology, the benchmark is provided with a dump of the original database created from the NPD FactPages, the set of mappings expressed in the R2RML mapping language, and a set of queries that have been formulated by domain experts starting from an informal set of questions provided to the users of the NPD FactPages.

The NPD Ontology [14] describes activities on the Norwegian continental shelf (NCS), e.g., about companies that own or operate petroleum fields [14]. The ontology has been created by the University of Oslo, and presents rich hierarchies of classes and properties, axioms that infer new objects, and disjointness assertions.

The ontology is specified in OWL and for the purpose of the benchmark it has been restricted to the fragment corresponding to the OWL 2 QL profile. Overall is composed by about 350 concepts, 142 roles and 238 attributes, with a maximum hierarchy depth of 10. This restriction is essential for its use in the context of OBDA as guarantees first-order rewritability for the class of union of conjunctive queries.

The NPD specification provides a set of 1173 mapping assertions, characterized by an average of 1.7 joins per query. The mappings have been partially bootstrapped automatically from the database and the ontology, and partially created by hand, and are specified in the R2RML mapping language.

The mappings have purposely not optimized, to measure the efficiency of the optimization strategies employed by an OBDA system. This means that the number of mappings that refer to the same ontology predicate is in general very large, up to about 30 in some cases.

The 1.9 revision of the NPD benchmark devises 30 SPARQL queries of different complexity, defined by domain experts starting from an informal set of questions to the users of the NPD FactPages. Among the set of queries, some have been specifically generated to stress the efficiency of a system when reasoning with respect to existential variables.

Some of the characteristics of the queries are the presence of concepts with a rich hierarchy and the presence of aggregations. For the purpose of this experimentation, we are only interested in the subset of these SPARQL queries corresponding to the class of union of conjunctive queries, as this is the semantics for SPARQL queries that is supported by Mastro. The only exception is that

[7] `http://factpages.npd.no/factpages/`

we make is for the use of duplicate elimination from the results. This requires to change part of the query set (for a detailed description of each of the queries adopted, we refer the reader to [8]).

The original NPD databases is derived from the data published on the *Norvegian Petroleum Directorate* FactPages[8].

The data from FactPages has been translated from CSV files into a structured database. The generated schema consists of 70 tables with 276 distinct columns (about 1000 columns in total), and 94 foreign keys.

The schemas of the tables overlap in the sense that several attributes are replicated in several tables. In fact, there are tables with more than 100 columns. The total size of the initial database is about 60 MB.

Since OBDA are expected to work in the context of Big Data, the authors of the benchmark have provided a tool that enables the initial database instance to be scaled in order to obtain larger instances. The scaling process, implemented by the *Virtual Instances Generator* (VIG) [7][9], is performed by taking into account the axioms in the ontology, the structure of the mappings, and the database constraints in order to preserve a set of similarity measures in the original database.

Starting from this initial database, instances of different size have been created with the use of the VIG generator, and have been loaded into separate databases. Table 1 shows the scaling factor and the size for each of the generated databases that have been used in our experiment. the number of the database represents its scale with respect to the original instance.

Name	**Scale Factor**	**Size**
NPD1	1	60 MB
NPD10	10	710 MB
NPD50	50	2570 MB
NPD100	100	5300 MB

Table 1: generated databases

We ran the NPD Benchmark on both the version of Mastro 1.0.2 and Ontop 3.0-beta2[10], on the same physical system using the same set of generated databases. The underlying DBMS is MySQL version 5.7.21, running locally on the testing machine.

The specifications of the platform used for the experiments are the following:

CPU Intel Xeon E5-2670 running at 2.60ghz
RAM 16 GB DDR3 1600 mhz

[8] `http://factpages.npd.no/factpages/`
[9] `https://github.com/ontop/vig`
[10] The particular version of Ontop used is a snapshot of the development version 3.0-beta2, compiled on the 17th of October 2017

OS Ubuntu 17.10 running in a virtualized environment (4 cores)

The experimentation is performed in the following mode:

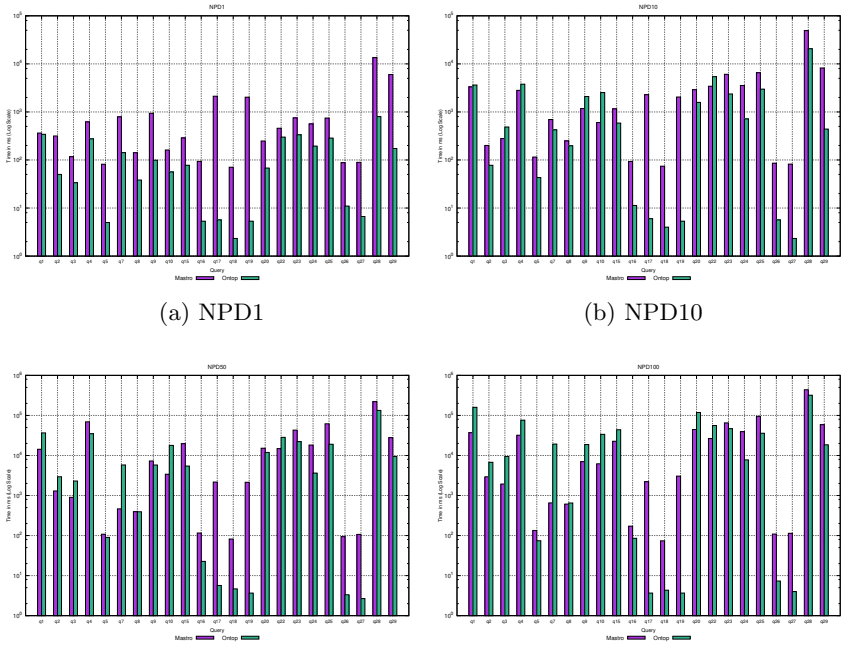
- We iterate over the set of queries. at each iteration we pick a random query from the set and evaluate it over the system. This is done in order to reduce the effect of the caching in the dbms.
  for each execution we store the results and the time it took to complete. we consider the combined time needed for evaluating the query and processing the set of results. queries are executed sequentially through the use of a testing platform designed specifically for this task, which accesses directly the internal apis of the systems.
- We repeat the process until all queries have been executed a fixed amount of time, in this case 5 executions were performed.
- Finally, we take the average of the execution times for each of the queries in the set.

In order to compare the systems under the same setting we enabled for both reasoning with respect to existential variables. Our metric of comparison is the total time taken to complete the execution of the query and to process the results. A comparison of the execution times for both systems for each of the database instance considered is shown in Figure 1.

We start our analysis by looking at some of the most interesting results. In particular, the queries where Ontop performs worse are those requiring reasoning with respect to the existential variables. Examples of these queries are $q9$ and $q10$, which cause the system to produce rewritings consisting of unions of tens of sub-queries.

Queries $q1$,$q2$,$q3$,$q4$,$q7$,$q15$,$q16$,$q25$ instead produce a simple SPJ rewriting, although the difference here is given by the fact that Ontop applies strictly the rules specified by the OWL 2 standard, performing the datatype casts and the generation of the IRIs directly at the SQL level, which causes a slowdown of the execution. Instead, Mastro adopts a less strict approach for dealing with datatypes, avoiding to perform casts and IRI-construction at the SQL level.

As for queries $q24$-$q25$, and $q28$-$q29$, it can be noted that in this case Mastro performs considerably slower than Ontop. This is due to the high number of mappings for the ontology predicates involved in these queries, that have to be unfolded by the system, and by the high number of sub-classes and sub-properties for the concepts and properties in the query, which cause the rewriting to grow exponentially in size. In Mastro, this phenomenon is mitigated with the addition of data constraints during the design of the views, in such a way that this redundancy is avoided. In this case, since view are generated automatically from the R2RML mapping, these constraints are not available to the system, effectively disabling all the optimizations that the system is capable of performing. Instead, in Ontop this problem is mitigated during the offline stage, when the mappings and the ontology are compiled to form the so-called $\mathcal{T}$-mappings, using the database constraints to optimize the mappings.



(a) NPD1 (b) NPD10

(c) NPD50 (d) NPD100

Fig. 1: Query Answering over NPD (Execution Time)

Finally, queries $q17$-$q19$, and $q26$-$q27$ are less interesting, as they produce empty unfoldings (due to mismatching function names in the mappings), and their execution time does not depend on the database size. For these queries, it can be noted that moving part of the computation to an offline stage gives Ontop a big advantage, as the system has to check a small amount of mappings at query execution time.

5 Comparison on the ACI application

The second scenario that we considered is the one currently in development within a joint project between Sapienza University of Rome and the Italian Automobile Club (ACI). The main objective of the project is assessing the benefits of the OBDA approach as a way of extracting data of interest for the ACI users.

The ACI ontology comprises about 500 concepts, 200 roles and 200 attributes, Among the most important notions in this ontology are the concept of *vehicle*, characterizing also its evolution over time, through the notion of *state*, and the concept of *subject*, modeling the possible roles played by the subjects (physical people or organizations) with respect to the taxation concerning vehicles.

The most important characteristic of this specification is that while the ontology has been designed to accurately reflect the reality of the domain of interest, the data sources are structured as to accomodate the requirements of the applications that make use of them. This large *semantic gap* between the data sources and the ontology has a large impact on the complexity of the mappings used to map the data at the sources to the ontology.

In this particular application the database is managed by an instance of the Oracle DBMS, for which we were granted accessed remotely through the use of a VPN. The relevant portion of the data for our experiment is distributed across 6 schemas. These schemas are composed of hundreds of tables, but for what regards the portion of the domain that is of interest in this experiment we concentrate on about 90 relational tables, ranging from information regarding the domain of PRA (Pubblico Registro Automobilistico), to those regarding the taxation concerning the vehicles. Some of these tables count from 200 million tuples up to above 1 billion tuples, with a number of attributes ranging from 30 to 100. The overall size of the portion of interest of the data source is several gigabytes of data (an accurate estimate was not possible).

The specification comprises 976 ontology mappings, composed from a set of about 110 views over the data source. About 300 of these ontology mappings are built from single view atoms, while the remaining are specified as conjunctive queries over the view predicates. For the purpose of our experimentation the mappings have been previously translated in R2RML through the use of the approach described in Section 3, and then imported back into both systems. During the translation process the constraints over the view predicates are discarded, since they cannot be expressed in R2RML.

We devised a set of 10 queries for our experiment (we refer the reader to [8] for a detailed description of the query set). Query $q1$ to $q5$ are basic navigational queries, that span some of the relevant part of the ontology. The remaining queries $q6$ to $q10$ are variations of the real queries used in the original experimentation of the project. These queries are built from a set of *competency questions*, defined by interviewing the domain experts over the practical questions that have to be posed to the system, and are used to validate the quality and the coherency of the specification.

We ran the queries on Mastro, version 1.0.2, and Ontop 3.0-beta2[11], using the same specification composed by the ontology and the mappings exported in R2RML.

The specifications of the platform used for the experiments are the same adopted in the case of the NPD Benchmark (cf. 4). In order to compare the systems under the same setting we enabled for both reasoning with respect to existential variables. Our metric of comparison is the total time taken to complete the execution of the query and to the process the results. Due to restrictions in the amount of time we were allowed for experiments, the queries were executed with a timeout of 3 hours. As a reference, we report also the time needed by the

[11] The particular version of Ontop used is a snapshot of the development version 3.0-beta2, compiled on the 17th of October 2017

Mastro system using the original mapping specification to evaluate the same set of queries. Figure 2 shows the overall execution times for both systems.

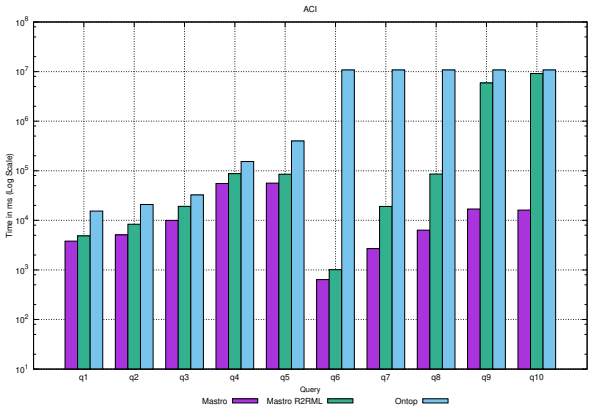


Fig. 2: Query Answering over ACI (Execution Time)

The first thing we notice from our experimentation is that in the case of the more complex queries ($q6 - 10$) Ontop was not able to complete in the time allowed, even though both systems are using the same specification. By further inspecting the generated SQL rewritings, We identified a few reasons for this phenomenon: (i) One reason is that, in the case of Ontop, the entire rewriting is executed as a single, complex SQL query, and the database management system was not able to compute an efficient query plan for such large queries. (ii) Another reason is that when the optimizations performed to reduce the size of the $\mathcal{T}$-mappings fail, because there may be missing database constraints, or possibly due to the complexity of the queries in the mappings, the system produces rewriting containing complex sub-queries, composed of unions of several SPJ queries and these types of queries are not evaluated efficiently. (iii) Finally, having the objects to be constructed directly at the level of the data sources, through the use inherently poorly performing operations such as string concatenation and type casts, is not feasible in real industrial applications. We noticed that in this particular case little to no optimizations were applied by Ontop, which caused it to produce rewritings where the intermediate views are composed by complex nested unions of *select-project-join* queries, that are not dealt with efficiently by the DBMS.

On the other hand, the approach adopted in Mastro, of splitting the queries in several, simpler conjunctive queries, still enabled the system to complete the task in the time allowed, even if in this case there are no optimizations performed, as the constraints over the views are not expressible in R2RML, which cause the time taken by the system to increase up to almost three orders of magnitude for the most difficult query.

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. J. Autom. Reasoning **39**(3), 385–429 (2007). https://doi.org/10.1007/s10817-007-9078-x, https://doi.org/10.1007/s10817-007-9078-x
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artif. Intell. **195**, 335–360 (2013). https://doi.org/10.1016/j.artint.2012.10.003, https://doi.org/10.1016/j.artint.2012.10.003
3. Calvanese, D., Lanti, D., Rezk, M., Slusnys, M., Xiao, G.: A scalable benchmark for OBDA systems: Preliminary report. In: Informal Proceedings of the 3rd International Workshop on OWL Reasoner Evaluation (ORE 2014) co-located with the Vienna Summer of Logic (VSL 2014), Vienna, Austria, July 13, 2014. pp. 36–43 (2014), http://ceur-ws.org/Vol-1207/paper_5.pdf
4. Console, M., Mora, J., Rosati, R., Santarelli, V., Savo, D.F.: Effective computation of maximal sound approximations of description logic ontologies. In: The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II. pp. 164–179 (2014). https://doi.org/10.1007/978-3-319-11915-1_11, https://doi.org/10.1007/978-3-319-11915-1_11
5. Lanti, D., Rezk, M., Slusnys, M., Xiao, G., Calvanese, D.: The NPD benchmark for OBDA systems. In: Proceedings of the 10th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014. pp. 3–18 (2014), http://ceur-ws.org/Vol-1261/SSWS2014_paper1.pdf
6. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD Benchmark: Reality Check for OBDA Systems. In: Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015. pp. 617–628 (2015). https://doi.org/10.5441/002/edbt.2015.62, https://doi.org/10.5441/002/edbt.2015.62
7. Lanti, D., Xiao, G., Calvanese, D.: Data scaling in OBDA benchmarks: The VIG approach. CoRR **abs/1607.06343** (2016), http://arxiv.org/abs/1607.06343
8. Namici, M.: R2RML mappings in OBDA systems: Enabling comparison among OBDA tools. CoRR **abs/1804.01405** (2018), http://arxiv.org/abs/1804.01405
9. Pinto, F.D., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013. pp. 561–572 (2013). https://doi.org/10.1145/2452376.2452441, http://doi.acm.org/10.1145/2452376.2452441
10. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking Data to Ontologies. J. Data Semantics **10**, 133–173 (2008). https://doi.org/10.1007/978-3-540-77688-8_5, https://doi.org/10.1007/978-3-540-77688-8_5
11. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-Based Data Access: Ontop of Databases. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I. pp. 558–573 (2013). https://doi.org/10.1007/978-3-642-41335-3_35, https://doi.org/10.1007/978-3-642-41335-3_35

12. Rodriguez-Muro, M., Rezk, M.: Efficient SPARQL-to-SQL with R2RML mappings. J. Web Sem. **33**, 141–169 (2015). https://doi.org/10.1016/j.websem.2015.03.001, https://doi.org/10.1016/j.websem.2015.03.001
13. Rosati, R., Almatelli, A.: Improving Query Answering over DL-Lite Ontologies. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010 (2010), http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1400
14. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the norwegian petroleum directorate's factpages as semantic web data. In: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II. pp. 162–177 (2013). https://doi.org/10.1007/978-3-642-41338-4_11, https://doi.org/10.1007/978-3-642-41338-4_11