Ontology-based Data Federation – A Framework Proposal

Zhenzhen Gu[1], Diego Calvanese[1,3,†], Marco Di Panfilo[1,†], Davide Lanti[1,*,†], Alessandro Mosca[1,†] and Guohui Xiao[2,†]

[1]*Free University of Bozen-Bolzano, Bolzano, Italy*
[2]*University of Bergen, Bergen, Norway*
[3]*Umeå University, Umeå, Sweden*

Abstract

Ontology-based data access (OBDA) is a well established approach to information management that facilitates the access to relational data sources through the mediation of a conceptual domain view, given in terms of an ontology, and the use of a declarative mapping between the data layer and the ontology. We formally introduce here the notion of *ontology-based data federation* (OBDF) to denote a framework that combines OBDA with a data federation layer where multiple heterogeneous sources are virtually exposed as a single relational database. We discuss opportunities and challenges of OBDF, and propose novel techniques to make query answering in the OBDF setting more efficient. Our techniques are validated through an extensive experimental evaluation based on the Berlin SPARQL Benchmark. This work is an abridged version of [1].

Keywords

OBDA, Data Federation, Query Optimization

1. Introduction

Ontology-based data access (OBDA) [2, 3, 4] is a well-established paradigm for querying data sources via a mediating ontology that has been successfully applied in many different domains [5]. In OBDA, the ontology is expressed in a lightweight conceptual modeling language, such as OWL 2 QL [6], which has its formal foundations in the Description Logics of the DL-Lite family [7]. Typically, it is assumed that the underlying data are stored in a single relational data source, to which the ontology elements are mapped in a declarative way. Specifically, in each *mapping*, a SQL query over the source is mapped to a class / property of the ontology, specifying how the data retrieved from the database (DB) should be used to create instances and values that populate the class / property.

Notably, for query answering, OBDA follows a *virtual* approach, i.e., the data are not actually extracted from the source to populate the classes and properties, but instead a SPARQL query [8] posed over the ontology is transformed on-the-fly into a SQL query over the data source. Such

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy
*Corresponding author.
†These authors contributed equally.
0000-0002-7346-6093 (Z. Gu); 0000-0001-5174-9693 (D. Calvanese); 0000-0002-9284-2488 (M. Di Panfilo); 0000-0003-1097-2965 (D. Lanti); 0000-0003-2323-3344 (A. Mosca); 0000-0002-5115-4769 (G. Xiao)

transformation takes into account both the ontology axioms (in what is generally called a *rewriting* step [7]) and the mappings (in an *unfolding* step [2, 9]), and typically may lead to a substantial blow-up in the size of the resulting SQL query w.r.t. the size of the original SPARQL query. Due to this, sophisticated optimization techniques have been proposed and implemented in commercial and open source OBDA systems [10, 11, 3, 12]. Such techniques exploit the available information about constraints in the data source (e.g., primary and foreign keys), the form of the mappings, and the structure of the query in order to optimize the SPARQL-to-SQL query translation process and generate a final query that is not only as compact as possible but also efficient to execute [9, 13].

So far, OBDA optimization techniques have been tailored for queries that are executed over a *single* data source to which the OBDA system is mapped. In many settings, however, there is the need to virtually access multiple, possibly heterogeneous, data sources in an integrated way. In this case, one can resort to *data federation* [14, 15], where multiple autonomous data sources are exposed transparently as a unified federated relational schema, usually called *virtual database*. Data federation is an active research area that has been extensively studied over the years, and many mature and highly-optimized data federation tools are currently available, both in the database community and in the Semantic Web community [16].

Data federation tools can be naturally used in combination with OBDA systems, by accessing them as if they were a single relational data source.[1] However, to the best of our knowledge, in current OBDA systems no provision is taken for the optimization of the generated SQL query to account for the fact that the evaluation of a SQL query in a data federation system is fundamentally different from query evaluation by a standard relational DBMS engine.

In our work, we address these issues by formalizing the novel setting of *Ontology-based Data Federation* (OBDF, for short) and studying dedicated optimization strategies tailored to the federated setting.

The present paper is an abridged version of [1], and we refer to that article for further details that we are not able to provide here due to space limitations.

2. Preliminaries

We introduce now the technical preliminaries necessary for the remainder of the paper.

Relational Algebra (RA). We assume the reader to be familiar with fundamental notions of RA. As conventions, we use Σ to denote a (relational) *DB schema*, D to denote an *instance of a DB schema*, and $\mathsf{sig}(\mathsf{A})$ to denote the *signature* of a RA expression A, which consists of the tuple $(a_1, \ldots, a_n)$ of *attributes* of the relation generated by A. When we want to make the signature of a RA expression A explicit, we use the notation $\mathsf{A}(a_1, \ldots, a_n)$. We introduce the abbreviation $\pi_{r_1/a_1,\ldots,r_k/a_k}$ for the combination $\rho_{r_1/a_1,\ldots,r_k/a_k}\pi_{a_1,\ldots,a_k}$ of *projection* and *renaming*.

Ontology-based Data Access (OBDA). We rely here on the classic framework presented in [4]. Due to space limitations, we assume the reader to be familiar with *ontologies* and *Description Logics* notation, and refer to the extensive literature on the subject [17].

[1] See, e.g., https://ontop-vkg.org/tutorial/federation/.

An *OBDA specification* $\mathcal{O}$ is a triple $(\mathcal{T}, \mathcal{M}, \Sigma)$, where:

- $\mathcal{T}$ is an *ontology* including *class inclusion axioms* $B \sqsubseteq C$ and *role inclusion axioms* $S \sqsubseteq T$,
- Σ is a relational DB schema, and
- $\mathcal{M}$ is a set of *OBDA-mappings* (or simply, *mappings*) between $\mathcal{T}$ and $\mathcal{M}$, of the form $\mathsf{A} \rightsquigarrow C(\mathsf{f}(\mathbf{a}))$ or $\mathsf{A} \rightsquigarrow P(\mathsf{f}(\mathbf{a}), \mathsf{g}(\mathbf{b}))$, where A is a RA expression over Σ, $\mathbf{a}$ and $\mathbf{b}$ are sets of attributes in $\mathsf{sig}(\mathsf{A})$, C is a class name of $\mathcal{T}$, P is a property name of $\mathcal{T}$, and $\mathsf{f}(\mathbf{a})$ and $\mathsf{g}(\mathbf{b})$ are *(R2RML) IRI templates* [18]. Such IRI templates specify how DB values are transformed into IRIs and RDF literals, making use of the attributes in $\mathsf{sig}(\mathsf{A})$. We call A the *source part* and $C(\mathsf{f}(\mathbf{a}))$ (resp., $P(\mathsf{f}(\mathbf{a}), \mathsf{g}(\mathbf{b}))$) the *target part* of the mapping.

An *OBDA instance* is a pair $(\mathcal{O}, D)$, where D is a DB instance of Σ.

For the semantics of an OBDA instance, we refer to [2]. Intuitively, an OBDA instance exposes a (virtual) *RDF graph* that can be queried through SPARQL [8]. The graph is *virtual* in the sense that RDF triples are not materialized. Instead, to answer a SPARQL query, the query is translated on-the-fly into an equivalent SQL query over the database, called its *translation*, through a process known as *unfolding* [2]. Different unfolding procedures have been proposed in the literature. For this work, we focus on two variants: the *classical one* aiming at producing a *union of conjunctive queries* (UCQ) [2], and the one aiming at producing a *join of unions of conjunctive queries* (JUCQ) [19, 20]. In state-of-the-art systems, the latter form usually provides an intermediate translation, which later is transformed into an UCQ translation, through standard *structural optimizations* [21, 3, 22].

Data Federation. Federating multiple, possibly heterogeneous data sources consists in exposing a unified view of such sources, usually called *virtual database* (VDB). In this paper, a (data) source, denoted by S, can be an RDB, a NoSQL DB, or of some other type. Consider a set $\mathbb{S} = \{S_1, \ldots, S_n\}$ of sources to be federated, and a function (given implicitly with $\mathbb{S}$) transforming the (possibly, non-relational) schema of each source S_i into a corresponding relational schema Σ_i, with the property that all such schemas are pairwise-disjoint. Then, the *federated VDB schema* (for $\mathbb{S}$) is the union $\mathbb{\Sigma}_{\mathbb{S}} = \bigcup_{i=1}^{n} \Sigma_i$. In the following, we use letters T, U to denote database tables, and a subscript i (e.g., in T_i) to indicate that S_i is the source of table T. Additionally, given an arbitrary RA expression A, $\mathsf{src}(\mathsf{A})$ denotes the set of sources of the relations in A, and $\mathsf{occ}(S_i, \mathsf{A})$ denotes the total number of occurrences in A of relations from S_i. A data federation instance $\mathbb{D}$ for $\mathbb{\Sigma}_{\mathbb{S}}$ is the relational instance $\bigcup_i D_i$ consisting of the union of an instance of each (relational) source schema in $\mathbb{\Sigma}_{\mathbb{S}}$. Hence, given a query q, $\mathsf{ans}(q, \mathbb{D})$ denotes the set of answers of q evaluated over the federation instance $\mathbb{D}$.

Local Operations vs. Federated Operations. To compute the answers to a federated query, a data federation system can delegate operations (e.g., joins and unions) to the data sources, or perform the operations itself. In this paper, we distinguish between *local operations* (e.g., joins), which are performed within a data source, and *federated operations* (e.g., joins across multiple sources), which have to be handled at the level of the federation system.

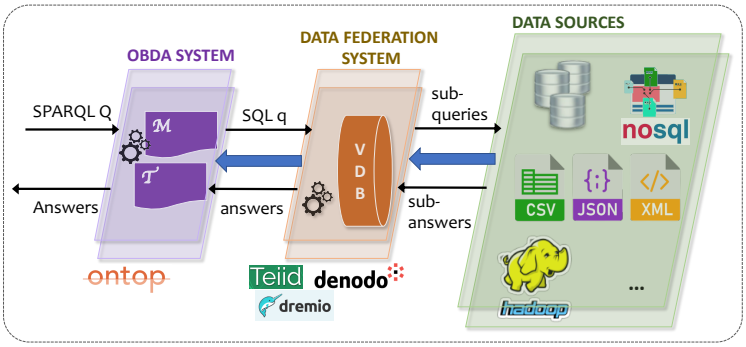

Figure 1: OBDF framework and query answering procedure.

3. Ontology-based Data Federation

We present now our general framework for enriching OBDA with data federation capabilities.

Definition 1 (OBDF [1]) Given an ontology $\mathcal{T}$, a federated VDB schema $\mathbb{\Sigma}_\mathbb{S}$, and a set $\mathcal{M}$ of mappings from $\mathbb{\Sigma}_\mathbb{S}$ to $\mathcal{T}$, an *ontology-based data federation (OBDF) specification* is the OBDA specification $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \mathbb{\Sigma}_\mathbb{S})$. ◁

Hence, the notions of *OBDF instance* and *answers to a query over an OBDF instance* coincide with their OBDA counterpart. Figure 1 depicts the full process of query answering in an OBDF scenario. A federation engine (e.g., Teiid[2] or Denodo[3]) is responsible for the federation of the data sources, and an OBDA system, in this case Ontop [3, 12], interacts with the federation engine *as it would normally do with a single relational database.*

Opportunities and Challenges. In line with the FAIR principles[4], OBDA allows users to publish data according to shared, agreed-upon vocabularies, enabling interoperability between applications. Furthermore, the ontology constitutes both a documentation about the data and a basis for enabling reasoning-based services, such as query answering w.r.t. the ontology. The added value of data federation is to extend the OBDA paradigm to multiple, possibly non-relational sources. While benefiting from both OBDA and data federation, OBDF combines their challenges. The next example shows possible issues with a naive implementation of OBDF.

Example 1 Consider an enterprise, whose data is spread across different sources $\mathbb{S} = \{S_1, \ldots, S_4\}$ that need to be integrated. Consider an OBDF specification $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \mathbb{\Sigma}_\mathbb{S})$, with $\mathcal{T}$ and $\mathcal{M}$ as in Figure 2, where each relation in $\mathcal{M}$ has a subscript i denoting the source S_i to which the relation belongs. For the SPARQL query Q in Figure 2, asking for products' and inspectors' information, the unfolding procedure would produce the SQL query q in the same figure. Intuitively, each line corresponds to the union of the SQL definitions in $\mathcal{M}$ for the corresponding atom in the SPARQL query (indicated between parentheses in the right

[2] teiid.io/
[3] www.denodo.com/
[4] https://www.go-fair.org/fair-principles/

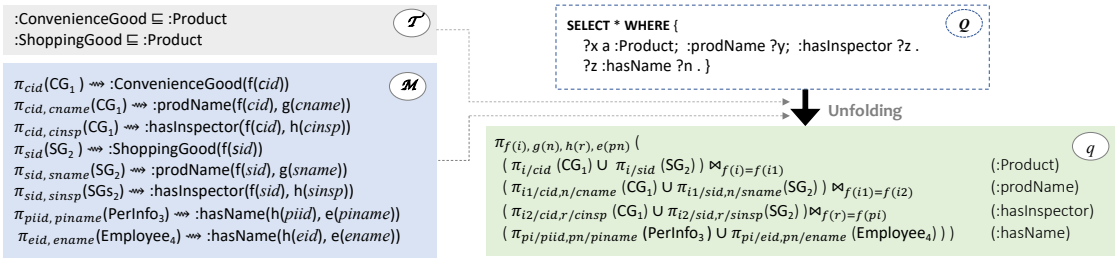

Figure 2: An example of unfolding a SPARQL query Q into a SQL query q w.r.t. the ontology $\mathcal{T}$ and mapping set $\mathcal{M}$.

margin), modulo the axioms in the ontology. Observe that this query is already verbose, with 3 federated joins and 4 federated unions across the different sources. At this point, state-of-the-art OBDA systems typically apply structural optimizations transforming the JUCQ q into a UCQ, by pushing the join operators to the bottom level of the algebra tree. After this transformation, one can easily verify that the obtained query would consists of $2^4 = 16$ unions of CQs, where each CQ has 3 join operators, thus amounting to 48 joins in total. Hence, transforming JUCQs into UCQs blindly can substantially increase the number of federated, thus inefficient, operations.

To complicate the picture, it is often the case that certain relations hold across the different sources: for instance, relation PerInfo might contain the names of all the employees in the enterprise, rendering the last union in q redundant. Similarly, ConvenienceGoods and ShoppingGoods might be disjoint, rendering all joins between CG_1 and SG_2 empty. ◁

We introduce now a novel query unfolding procedure specific to the OBDF setting, able to choose the best strategy between UCQ and JUCQ unfoldings and to exploit relations holding across different data sources. This procedure relies on so-called *data hints*, which are meta-information describing certain properties of the instances being federated.

4. Data Hints and Query Optimization in OBDF

In [20] it was shown that it is possible to determine *a-priori* all the joins between relations that can occur in the SQL translation of a user query. This can be done by an offline analysis of the OBDA specification, that is, by collecting pairs of atoms with *compatible IRI templates*. We exploit this idea to automatically gather different kinds of meta-information, called *data hints* (or, simply, *hints*), that we use to optimize query answering in OBDF.

4.1. Data Hints

Consider a fixed federated VDB schema $\mathbb{\Sigma}_\mathbb{S}$. We identify three kinds of hints: *empty federated joins*, *containment redundancies*, and *materialized views*.

The first kind of hint, *empty federated join*, annotates which joins are expected to be empty when evaluated over the current data federation instance. Formally, given an instance $\mathbb{D}$ of $\mathbb{\Sigma}_\mathbb{S}$ and a federated join expression FJ over $\mathbb{\Sigma}_\mathbb{S}$, we say that FJ is an *empty federated join w.r.t.* $\mathbb{D}$, denoted as $\mathsf{FJ} =_\mathbb{D} \emptyset$, if $\mathsf{ans}(\mathsf{FJ}, \mathbb{D}) = \emptyset$.

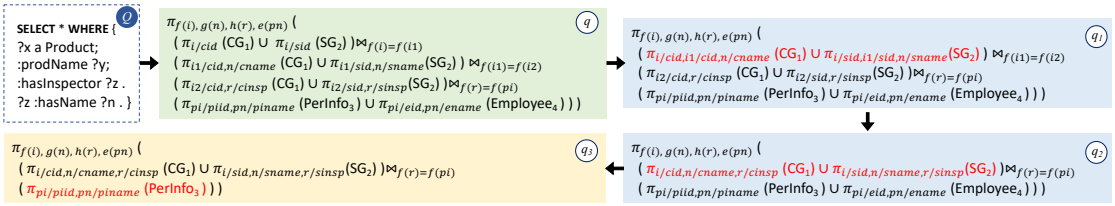

Figure 3: An example of translating a SPARQL query to SQL under hints. Expressions changed at each step (w.r.t. the previous step) are highlighted in red.

The second kind of hint, *containment redundancy*, annotates the presence of redundancy (typically across different data sources). Formally, given an instance $\mathbb{D}$ of $\mathbb{\Sigma}_\mathbb{S}$ and two expressions A and B over $\mathbb{\Sigma}_\mathbb{S}$, we say that A is *data-contained in* B *w.r.t.* $\mathbb{D}$, denoted as $\mathsf{A} \subseteq_\mathbb{D} \mathsf{B}$, if $\mathsf{ans}(\mathsf{A}, \mathbb{D}) \subseteq \mathsf{ans}(\mathsf{B}, \mathbb{D})$. We use $\mathsf{A} \equiv_\mathbb{D} \mathsf{B}$ to indicate that $\mathsf{A} \subseteq_\mathbb{D} \mathsf{B}$ and $\mathsf{B} \subseteq_\mathbb{D} \mathsf{A}$.

The third kind of hint, *materialized view*, exploits the ability to specify materialized views provided by data federation systems. In our formalization, we assume the presence of an extra source to store the materialization of the views, where such source could be the federation system itself. This is motivated by the fact that it is often impossible or impractical to store the views directly in the sources, due to access policies, source ownership, etc. Formally, let M be a set of (SQL) view definitions. We denote by $\mathbb{\Sigma}_\mathbb{S}^\mathsf{M}$ the VDB schema $\mathbb{\Sigma}_\mathbb{S} \cup \Sigma_\mathsf{M}$, where Σ_M is the relational schema of a special data source S^M materializing the views defined in M. Observe that, consequently, an instance $\mathbb{D}^\mathsf{M}$ of $\mathbb{\Sigma}_\mathbb{S}^\mathsf{M}$ is a VDB instance $\mathbb{D} \cup D_\mathsf{M}$ such that $\mathbb{D}$ is an instance of $\mathbb{\Sigma}_\mathbb{S}$ and D_M is an instance of Σ_M conforming to the view definitions in M.

In our framework, we also assume two labeling functions characterizing whether a source is *efficient* or *inefficient* when answering queries, and whether a source is *dynamic* (i.e., its content is expected to change frequently) or *static* (i.e., its content is not expected to change), respectively. The idea is that the information carried by data hints is reliable only when measured over static sources, and therefore optimizations based on it should care about this aspect.

4.2. Query Optimization in OBDF

We now discuss our solution to optimize SQL translations of SPARQL queries posed over an OBDF system. The main intuition is that, in OBDF, the ontology and mappings contain information to guide the discovery of data hints. The overall method consists of two parts: *(1)* an offline *hints pre-computation part*, and *(2)* an on-line *translation optimization part*. Details on both parts are provided in [1], where a *cost model* is introduced to guide the optimization. For each query q, $\mathsf{Cost}(q)$ is defined as a pair $(\sum_i c_i, \#ineff)$, with c_i the cost of each federated join FJ_i in q (we set $c_i = n + m$ for $\mathsf{FJ}_i = \cup_{j=1}^n \mathsf{A}_j \bowtie \cup_{k=1}^m \mathsf{B}_k$) and $\#ineff$ the number of occurrences in q of relations over *inefficient* sources; a partial order $\prec$ is then introduced to compare costs. We here present an example of the on-line translation optimization part.

Example 2 Consider again the OBDF specification $\mathcal{F}$ and SPARQL query Q from Example 1, and an OBDF instance $(\mathcal{F}, \mathbb{D})$. Suppose all "*id*" columns to be primary keys for the respective tables. Suppose we have the empty federated join hint $\mathsf{CG}_1 \bowtie_{cid=sid} \mathsf{SG}_2 =_\mathbb{D} \emptyset$ and the containment redundancy hint $\pi_{pid/piid,pname/piname}(\mathsf{PerInfo_3}) \equiv_\mathbb{D} \pi_{pid/eid,pname/ename}(\mathsf{Employee_4})$,

and that S_1, S_2, and S_3 have been labelled as *efficient*, while S_4 as *inefficient*. Then, Figure 3 illustrates how the above hints and labels are exploited in order to further unfold the SQL translation q of Q. The translation goes as follows, where we assume that the operators in the query expression are processed in order from left to right:

1. The federated join $\bowtie_{\mathsf{f}(i)=\mathsf{f}(i1)}$ is first unfolded into a union of 4 joins and then traslated into $\pi_{i/cid,i1/cid,n/cname}(\mathsf{CG_1}) \cup \pi_{i/sid,i1/sid,n/sname}(\mathsf{SG_2})$ on the basis of the empty join hint and the application of optimization rule **sjr** [1, Figure 3].
2. Similarly, the intermediate query q_1 is translated into query q_2.
3. Based on the containment redundancy hint and the given source labelling, the union between $\mathsf{PerInfo_3}$ and $\mathsf{Employee_4}$ is then removed by the application of rule **cr** [1, Figure 3], and only the projection over the fastest source $\mathsf{PerInfo_3}$ is kept in the resulting query q_3.

Each unfolding step reduces the cost of the query, and $\mathsf{Cost}(q_3) \prec \mathsf{Cost}(q)$. ◁

5. Evaluation

We have carried out an extensive experiment to verify the effectiveness of the proposed optimizations. The material for reproducing the experiments, and an appendix of this work with additional details, are available at https://github.com/efghk321456/sc.

5.1. Experimental Setup

Our experiment is based on the well-known Berlin SPARQL Benchmark (BSBM) [23]. BSBM is built around an e-commerce use case in which a set of products is offered by different vendors and consumers have posted reviews about products.

We generate 5 data sets $\mathcal{D}_1$–$\mathcal{D}_5$ as non-disjoint subsets of the original BSBM tables, introducing data partitioning (e.g., "horizontal" table split) and redundancy (e.g., table replication across data sets) to simulate the scenarios where data from different sources are mapped to the same classes in the ontology. We store the data sets in different database systems and derive 8 data sources in total. $\mathcal{D}_1$–$\mathcal{D}_5$ are first stored in RDBs, obtaining data sources S_1–S_5. We also convert the tables in $\mathcal{D}_2$ and $\mathcal{D}_4$ to CSV files to obtain two more data sources S'_2 and S'_4. We additionally convert $\mathcal{D}_5$ into JSON files, store them in MongoDB, and obtain a further data source S'_5.

We use the ontology and mappings from [13] for OBDA, with minor mappings modifications for handling the different DBs. As baselines, we generate two OBDA specifications using two centralized RDBs: sc1, with the original (disjoint, non-partitioned) BSBM tables, and sc2, with the (replicated, partitioned) tables in S_1–S_5. We create two OBDF specifications over the *Teiid* data federation system: a homogeneous (relational) one, hom, defined over sources S_1–S_5, and a heterogeneous one het (in which some data are also in CSV files and MongoDB), defined over the sources $\{S_1, S'_2, S_3, S'_4, S'_5\}$. For each OBDF specification, the hints include 3 empty federated joins, 1 data redundancy, and 6 materialized views, the latter stored in a local PostgreSQL DB.

5.2. Query Evaluation and Result Analysis

To test scalability, we generate three groups of instances using the BSBM data generation tool, setting the number of products to 20K, 200K, and 2M, respectively. For each OBDF instance,

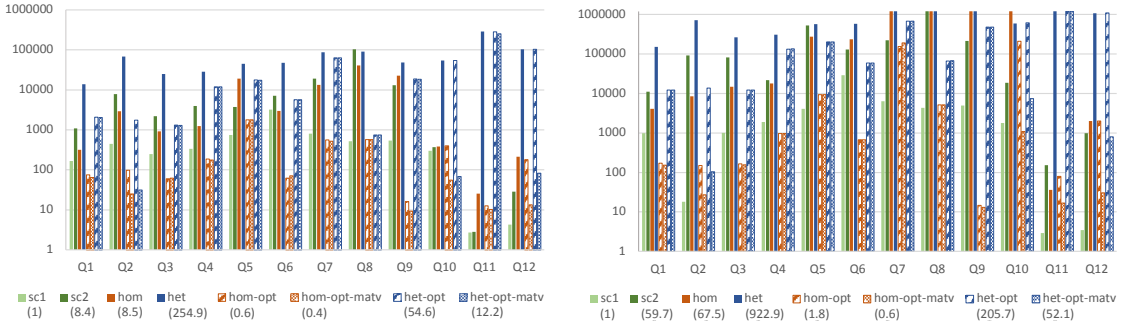

Figure 4: Evaluation times (ms) of BSBM queries Q_1–Q_{12} for scale factors 200K (left) and 2M (right). Numbers in parentheses are (geometric) mean evaluation times across queries normalized relative to sc1 baseline (e.g., for scale factor 200K, query evaluation in sc2 is 8.4 times slower than in sc1, on average).

the hints, including data redundancy, empty federated joins, and materialized views, are precomputed (see Section 4.2). For space reasons, we only report the results on 200K and 2M.

We consider the 12 SPARQL queries Q_1 to Q_{12} from the BSBM benchmark. The SQL queries without hint-based optimization are generated by Ontop, and the optimized ones are produced manually following the approach of Section 4.2. The SQL queries evaluation times are reported in Figure 4: $\mathsf{hom}_{\mathsf{opt}}$ and $\mathsf{het}_{\mathsf{opt}}$ denote the evaluation with the hints of empty federated joins and redundancies, while $\mathsf{hom}_{\mathsf{opt}}^{\mathsf{matv}}$ and $\mathsf{het}_{\mathsf{opt}}^{\mathsf{matv}}$ employ all the hints, including materialized views.

By analyzing the query evaluation times in Figure 4, we can conclude that data partitioning alone can make query answering less efficient (sc1 vs. sc2, the latter 8.4 and 59.7 times slower for 200K and 2M, respectively), that further adding a federation layer does not have a significant impact on query answering (hom vs. sc2), and that the federation of heterogeneous data sources leads to a significant decrease in performance due to the expensive access to non-relational sources (hom vs. het). The optimization with hints is found to be very effective, particularly in homogeneous cases ($\mathsf{hom}_{\mathsf{opt}}$ / $\mathsf{hom}_{\mathsf{opt}}^{\mathsf{matv}}$ vs. hom) where the performance is much better than without hints, often in the order of magnitudes. Even in heterogeneous cases ($\mathsf{het}_{\mathsf{opt}}$ / $\mathsf{het}_{\mathsf{opt}}^{\mathsf{matv}}$ vs. het), optimization helps, especially when materialized views are used, which per se improve the performance dramatically.

6. Conclusions

We have introduced the *ontology-based data federation* (OBDF) setting and have studied the problem of optimizing query translations in this setting. Specifically, we have provided techniques to optimize query translation in OBDF that are based on source data information that can be automatically computed in an offline stage by exploiting the information encoded in an OBDF specification. We have performed an extensive empirical evaluation, showing that our techniques have a significant impact on the overall performance of query answering.

In this work, we laid the foundations of OBDF. In future work we plan to further investigate hint-based optimizations, as well as implement our algorithms in an actual system. We will also investigate more sophisticated ways of handling static and dynamic sources (e.g., [24]), which might be necessary in order to apply OBDF in complex, real-world scenarios.

References

[1] Z. Gu, D. Lanti, A. Mosca, G. Xiao, J. Xiong, D. Calvanese, Ontology-based data federation, in: Proc. of the 11th Int. Joint Conf. on Knowledge Graphs (IJCKG), ACM, 2022, pp. 10–19. doi:10.1145/3579051.3579070.

[2] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. on Data Semantics 10 (2008) 133–173. doi:10.1007/978-3-540-77688-8_5.

[3] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web J. 8 (2017) 471–487. doi:10.3233/SW-160217.

[4] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI), IJCAI Org., 2018, pp. 5511–5519. doi:10.24963/ijcai.2018/777.

[5] G. Xiao, L. Ding, B. Cogrel, D. Calvanese, Virtual Knowledge Graphs: An overview of systems and use cases, Data Intelligence 1 (2019) 201–223. doi:10.1162/dint_a_00011.

[6] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation, World Wide Web Consortium, 2012. Available at http://www.w3.org/TR/owl2-profiles/.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family, J. of Automated Reasoning 39 (2007) 385–429. doi:10.1007/s10817-007-9078-x.

[8] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, W3C Recommendation, World Wide Web Consortium, 2013. Available at http://www.w3.org/TR/sparql11-query.

[9] F. Priyatna, O. Corcho, J. F. Sequeda, Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph, in: Proc. of the 23rd Int. World Wide Web Conf. (WWW), 2014, pp. 479–490. doi:10.1145/2566486.2567981.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, D. F. Savo, The Mastro system for ontology-based data access, Semantic Web J. 2 (2011) 43–53.

[11] J. F. Sequeda, D. P. Miranker, Ultrawrap: SPARQL execution on relational data, J. of Web Semantics 22 (2013) 19–39.

[12] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalayci, L. Ding, J. Corman, B. Cogrel, D. Calvanese, E. Botoeva, The virtual knowledge graph system Ontop, in: Proc. of the 19th Int. Semantic Web Conf. (ISWC), volume 12507 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 259–277. doi:10.1007/978-3-030-62466-8_17.

[13] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, E. Botoeva, Efficient handling of SPARQL optional for OBDA, in: Proc. of the 17th Int. Semantic Web Conf. (ISWC), volume 11136 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 354–373. doi:10.1007/978-3-030-00671-6_21.

[14] A. P. Sheth, J. A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Computing Surveys 22 (1990) 183–236.

[15] L. M. Haas, E. T. Lin, M. A. Roth, Data integration through database federation, IBM Systems J. 41 (2002) 578–596.

[16] Z. Gu, F. Corcoglioniti, D. Lanti, A. Mosca, G. Xiao, J. Xiong, D. Calvanese, A systematic overview of data federation systems, Semantic Web J. (2022). doi:10.3233/SW-223201, to appear in print. Available at tinyurl.com/48tpyy88.

[17] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, 2nd ed., Cambridge University Press, 2007.

[18] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, World Wide Web Consortium, 2012. Available at http://www.w3.org/TR/r2rml/.

[19] D. Bursztyn, F. Goasdoué, I. Manolescu, Reformulation-based query answering in RDF: Alternatives and performance, Proc. of the VLDB Endowment 8 (2015) 1888–1891. URL: http://www.vldb.org/pvldb/vol8/p1888-bursztyn.pdf.

[20] D. Lanti, G. Xiao, D. Calvanese, Cost-driven ontology-based data access, in: Proc. of the 16th Int. Semantic Web Conf. (ISWC), volume 10587 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 452–470. doi:10.1007/978-3-319-68288-4_27.

[21] M. Rodriguez-Muro, M. Rezk, Efficient SPARQL-to-SQL with R2RML mappings, J. of Web Semantics 33 (2015) 141–169. doi:10.1016/j.websem.2015.03.001.

[22] D. Hovland, D. Lanti, M. Rezk, G. Xiao, OBDA constraints for effective query answering, in: Proc. of the 10th Int. Symp. on Rule Technologies: Research, Tools, and Applications (RuleML), volume 9718 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 269–286.

[23] C. Bizer, A. Schultz, The Berlin SPARQL benchmark, Int. J. on Semantic Web and Information Systems 5 (2009) 1–24.

[24] C. Bobed, F. Bobillo, S. Ilarri, E. Mena, Answering continuous description logic queries: Managing static and volatile knowledge in ontologies, Int. J. Semant. Web Inf. Syst. 10 (2014) 1–44. URL: https://doi.org/10.4018/IJSWIS.2014070101. doi:10.4018/IJSWIS.2014070101.