

# Ontology-Based Data Federation

Extended Abstract

Zhenzhen Gu<sup>1</sup>, Davide Lanti<sup>1</sup>, Alessandro Mosca<sup>1</sup>, Guohui Xiao<sup>2,3</sup>, Jing Xiong<sup>1</sup> and Diego Calvanese<sup>1,3,4</sup>

<sup>1</sup>KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, 39100 Bolzano, Italy

<sup>2</sup>Faculty of Social Sciences, University of Bergen, Bergen, Norway

<sup>3</sup>Ontopic S.R.L., 39100 Bolzano, Italy

<sup>4</sup>Department of Computing Science, Umeå University, 901 87 Umeå, Sweden

## Abstract

We formally introduce *ontology-based data federation* (OBDF), to denote a framework combining ontology-based data access (OBDA) with a data federation layer, which virtually exposes multiple heterogeneous sources as a single relational database. In this setting, the SQL queries generated by the OBDA component by translating user SPARQL queries are further transformed by the data federation layer so as to be efficiently executed over the data sources. The structure of these SQL queries directly affects their execution time in the data federation layer and their optimization is crucial for performance. We propose here novel optimizations specific for OBDF, which are based on “hints” about existing data redundancies in the sources, empty join operations, and the need for materialized views. Such hints can be systematically inferred by analyzing the OBDA mappings and ontology and exploited to simplify the query structure. We also carry out an experimental evaluation in which we show the effectiveness of our optimizations.

## Keywords

Ontology-based data access, OBDA, data federation, query optimization

## 1. Research Problem and Contribution

Ontology-based data access (OBDA) [1, 2, 3], also known as *Virtual Knowledge Graphs*, is a well-established paradigm for querying data sources via a mediating ontology. Such paradigm has been successfully applied in many different domains [4]. In OBDA, the ontology is expressed in a lightweight ontology languages, such as OWL 2 QL [5], which has its formal foundations in the DL-Lite family [6]. 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. 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 posed over the ontology is transformed in a sequence of steps into a SQL query over the data source [6, 1, 7].

---


 DL 2022: 35th International Workshop on Description Logics, August 7–10, 2022, Haifa, Israel


 zhenzhen.gu@unibz.it (Z. Gu); davide.lanti@unibz.it (D. Lanti); alessandro.mosca@unibz.it (A. Mosca);

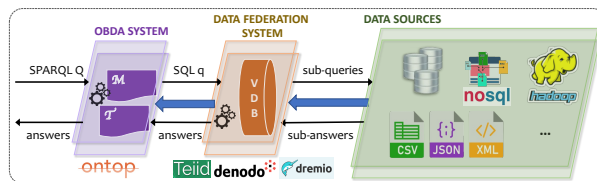
guohui.xiao@uib.no (G. Xiao); jing.xiong@unibz.it (J. Xiong); calvanese@inf.unibz.it (D. Calvanese)

 0000-0002-7346-6093 (Z. Gu); 0000-0003-1097-2965 (D. Lanti); 0000-0003-2323-3344 (A. Mosca);

0000-0002-5115-4769 (G. Xiao); 0000-0002-3604-9645 (J. Xiong); 0000-0001-5174-9693 (D. Calvanese)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)



**Figure 1:** The general framework of OBDF and the full query answering procedure.

Sophisticated optimization techniques have been proposed and actually implemented in most commercial and open source OBDA systems that are currently available [8, 9, 2, 10].

So far, such techniques have been tailored towards optimizing 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* [11, 12], where multiple autonomous data sources are exposed transparently as a unified federated relational schema, usually called *virtual database*. Data federation has been studied extensively over the years, and is still an active research area, and many mature and highly-optimized data federation systems are currently available, both in the database community<sup>1</sup> and in the Semantic Web community (such as [13]).

Data federation systems are already supported by OBDA systems, by accessing them as if they were a single relational data source<sup>2</sup>. 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. Essentially, this is due to *federated joins*, i.e., joins that involve different data sources of the federation.

In this work, we consider specifically this setting, which we call *Ontology-based Data Federation* (OBDF), and provide the following contributions. (i) We formalize OBDF systems, where a collection of multiple, possibly heterogeneous and federated data sources are accessed via mappings from an ontology. (ii) We study query optimization in OBDF, by devising a set of optimization techniques specifically tailored towards federated data sources. (iii) We carry out an experimental evaluation in which we assess the effectiveness of the proposed optimizations.

## 2. Ontology-based Data Federation

**Data Federation.** We first formalize data federation, where multiple, possibly heterogeneous data sources are integrated in a unified view, usually called *virtual database* (VDB). A data source  $S$  can be an RDB, a NoSQL DB, a CSV source, or of some other type, with the only requirement that it supports a query language. Given a set  $\mathcal{S} = \{S_1, \dots, S_n\}$  of sources to be federated, and a function (given implicitly with  $\mathcal{S}$ ) transforming the (possibly, non-relational) schema of each source  $S_i$  into a corresponding relational schema  $\Sigma_i$ , a *VDB schema* (for  $\mathcal{S}$ ) is the disjoint union  $\bar{\Sigma} = \bigcup_{i=1}^n \Sigma_i$ . In the following, we use letters  $T, U$  to denote database tables, and the subscript  $i$  (e.g.,  $T_i$ ) to indicate that table  $T$  belongs to source  $S_i$ . A data federation

<sup>1</sup>E.g., Denodo ([www.denodo.com](http://www.denodo.com)), Dremio ([www.dremio.com](http://www.dremio.com)), and Teiid ([teiid.io](http://teiid.io)).

<sup>2</sup>See, e.g., [ontop-vkg.org/tutorial/federation/](http://ontop-vkg.org/tutorial/federation/).

instance  $\mathbb{D}$  for  $\Sigma$  is the relational instance  $\bigcup_i D_i$  made of the union of all instances of the source schemas in  $\Sigma$ . For a query  $q$ ,  $\text{ans}(q, \mathbb{D})$  denotes the set of answers of  $q$  evaluated over the data federation instance  $\mathbb{D}$ .

**OBDF.** Given an ontology  $\mathcal{T}$ , a VDB schema  $\Sigma$  for a set  $\mathcal{S}$  of data sources, and a set  $\mathcal{M}$  of mappings from  $\Sigma$  to  $\mathcal{T}$ , an *ontology-based data federation specification* is the OBDA specification  $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \Sigma)$ . The notions of *OBDF instance* and answers to a query over an OBDF instance coincide with their OBDA counterpart. Figure 1 depicts the OBDF approach, where query answering is carried out by translating an input SPARQL query into a SQL query over the VDB, and then having the federation system evaluate such query.

**Challenges.** In OBDF, multiple sources are mapped to the same ontology, therefore the translated SQL query typically contains *federated joins*, i.e., joins involving different data sources. Computing such joins is expensive [13], since they cannot be evaluated locally and may cause large amounts of data to be transferred from the sources to the federation system. Besides, as also observed in our preliminary experiments, evaluating queries over inefficient sources (e.g., CSV files) slows down the overall query answering significantly. This problem is not addressed by optimization techniques in current OBDA systems, since they treat all tables present in the VDB schema in the same way. Therefore, new techniques to optimize query answering in OBDF are needed.

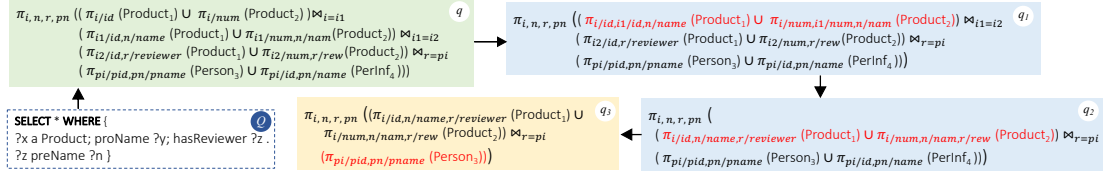
### 3. Query Optimization in Ontology-based Data Federation

We study query optimization in OBDF, by devising a set of optimization techniques specifically tailored to federated data sources. We propose a set of “hints” based on *empty federated joins*, *data redundancy*, and *materialized views*, which can be pre-computed by analyzing the sources, the mappings, and the ontology, and we provide a novel optimized query unfolding algorithm that minimizes the number of federated joins and reduces the access to inefficient data sources.

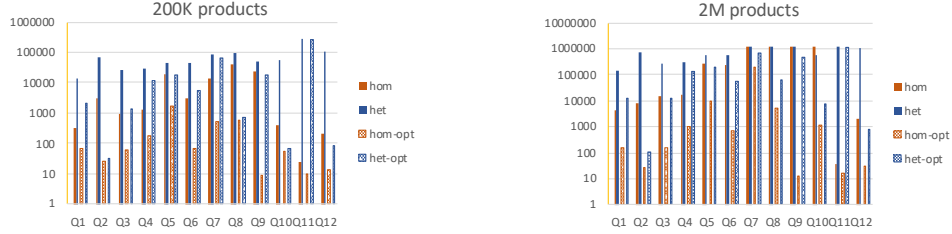
**Method.** The overall method consists of two parts: (1) an off-line *hint pre-computation part*, and (2) an on-line *translation optimization part*. We provide an algorithm to pre-compute the mentioned types of “hints”, based on a static analysis of the mappings and the ontology. The number of hints is bound to  $n^2$ , where  $n$  is the number of attributes in the DB schema. If the data-source is updated, then the hints are no-longer valid. Hence, our approach works best in mostly static scenarios, e.g., where new data undergoes a period of validation before being released.

Our concrete approach to query optimization consists of a set of query optimization rules that take into account the pre-computed hints, a simple cost model that distinguishes efficient/inefficient and static/dynamic data sources, and a novel optimized query unfolding algorithm that minimizes the number of federated joins and the access to inefficient data sources.

**Example.** Consider an OBDF specification  $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \Sigma)$  and a data federation instance  $\mathbb{D}$  of  $\Sigma$  containing the *static* sources  $S_1$ – $S_4$  to be federated. The VDB schema  $\Sigma$  contains, possibly among others, the following relation schemas, which are mapped to the ontology  $\mathcal{T}$ :  $\text{Product}_1(id, name, reviewer)$  for  $S_1$ ,  $\text{Product}_2(num, nam, rew)$  for  $S_2$  (where  $id$  and  $num$  are primary keys of  $\text{Product}_1$  and  $\text{Product}_2$  respectively)  $\text{Person}_3(pid, pname)$  for  $S_3$ , and  $\text{PerIn}_4(id, name, address)$  for  $S_4$ . We also assume to have the empty federated join hint  $\text{Product}_1 \bowtie_{id=num} \text{Product}_2 \equiv_{\mathbb{D}} \emptyset$  and the redundancy hint  $\pi_{pid, pname}(\text{Person}_3) \equiv_{\mathbb{D}}$



**Figure 2:** An example of optimizing a translated SQL query based on the pre-computed hints.



**Figure 3:** Results (time in ms) of SQL query evaluation for the BSBM experimentation.

$\pi_{pid/id,pn/pname/name}(\text{PerInf}_4)$ , and that  $S_1$ ,  $S_2$ , and  $S_3$  are labelled as *efficient*, while  $S_4$  as *inefficient*.

Figure 2 illustrates how the above hints and labelling are exploited to unfold a SPARQL query  $Q$ . The translation proceeds as follows, where we assume that query operators are processed from left to right. (1) The federated join  $\bowtie_{i=i1}$  is first unfolded into a union of 4 joins and then translated into  $\pi_{i/id,i1/id,n/name}(\text{Product}_1) \cup \pi_{i/num,i1/num,n/nam}(\text{Product}_2)$  on the basis of the empty federated join hint and the elimination of self-joins. (2) Similarly to the previous step, the intermediate query  $q_1$  is translated into  $q_2$ . (3) Finally, on the basis of the data redundancy hint and the given source labelling, the union between  $\text{Person}_3$  and  $\text{PerInf}_4$  is removed, and only the projection over the efficient source  $\text{Person}_3$  is kept in the resulting query  $q_3$ . Each step reduces the cost of  $q$ .

**Experimentation.** We have carried out an extensive experiment to verify the effectiveness of the proposed optimizations, where we have employed main-stream DB engines (like PostgreSQL, MySQL, and MongoDB), the OBDA system Ontop, and the data federation engine Teiid. Based on the BSBM benchmark [14], we have created two OBDF specifications: a homogeneous one, where we have used as sources five relational databases, and a heterogeneous one, where some of these five data sources are translated into NoSQL DBs and CSV files. For scalability testing, we have generated three groups of instances using the BSBM data generation tool, setting the number of products to be 20K, 200K, and 2M, respectively. Thus, each OBDF specification has three instances. For space reasons, we show in Figure 3 only the results of evaluating the SQL translations of 12 SPARQL queries in BSBM for the instances with 200K and 2M products, where *hom* and *het* denote the results of evaluating the SQL queries translated by Ontop over the homogeneous and heterogeneous situation respectively, and *hom<sub>opt</sub>* and *het<sub>opt</sub>* denote the results of evaluating the optimized SQL queries over the two situations. From Figure 3, we can observe that our optimization strategies are effective for OBDF (e.g, for 200K products and  $Q_2$ , we observe a reduction from 2942.2 ms to 24.9 ms in query answering time for the homogeneous situation)

## Acknowledgments

This research has been partially supported by the EU H2020 project INODE (grant agreement No. 863410), by the Italian PRIN project HOPE (2019-2022), by the Free University of Bozen-Bolzano through the project MP4OBDA, and by the “Fusion Grant” project HIVE. D. Calvanese is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We thank our colleagues, in particular Francesco Corcoglioniti, for their discussions and feedback.

## References

- [1] 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.
- [2] 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.
- [3] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyashev, 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.
- [4] 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.
- [5] 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/>.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] J. F. Sequeda, D. P. Miranker, Ultrawrap: SPARQL execution on relational data, *J. of Web Semantics* 22 (2013) 19–39.
- [10] 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.
- [11] A. P. Sheth, J. A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, *ACM Computing Surveys* 22 (1990) 183–236.

- [12] L. M. Haas, E. T. Lin, M. A. Roth, Data integration through database federation, *IBM Systems J.* 41 (2002) 578–596.
- [13] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, FedX: Optimization techniques for federated query processing on linked data, in: *Proc. of the 10th Int. Semantic Web Conf. (ISWC)*, volume 7031 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 601–616.
- [14] C. Bizer, A. Schultz, The Berlin SPARQL benchmark, *Int. J. on Semantic Web and Information Systems* 5 (2009) 1–24.