Towards Optimizing Ontology-Based Data Federation: Performance Insights from Experimental Studies

Marco Di Panfilo[1]

[1]*Supervised by Diego Calvanese and Davide Lanti,*
Free University of Bozen-Bolzano

Abstract

Ontology-Based Data Federation (OBDF) is a recently introduced framework that utilizes virtual knowledge graphs for real-time, unified data querying across diverse data sources. This approach significantly enhances data integration efficiency. As part of a PhD research, we aim to further investigate and optimize the performance of OBDF. To this end, we have set up and conducted extensive experiments with OBDF using three representative data federation systems, which allow for a comprehensive assessment of the framework and the extensions developed in this thesis. This paper provides a recap of OBDF and details the experimental work conducted and its results, paving the way for further OBDF refinements to be developed during the PhD thesis.

Keywords

OBDA, OBDF, VKG, Data Federation, Query Optimization

1. Introduction

Data is recognized as one of the most valuable resources, whether the context is industrial, academic, or private [1]. In today's world, timely and accurate access to data, and the insights derived from it, are crucial for driving meaningful impact and informed decision-making. Granting such an access to data, though, becomes even more challenging when the data does not originate from a single consistent source but rather from multiple heterogeneous sources that may be inconsistent with one another. This challenge leads to the problem of *Data Integration*, which involves ensuring timely and accurate accessibility of multiple data sources to the end-user.

Today's rapid expansion in the volume, variety, and velocity of data [1] presents significant challenges for traditional data integration solutions, particularly those based on data warehousing. These approaches are becoming increasingly costly and face difficulties to maintain data freshness. In contrast, data integration based on a *virtual approach* addresses these challenges by enabling the uniform and virtual management of heterogeneous data. This is achieved by unifying diverse sources, often controlled by different authorities, under a common global schema without the need to materialize a replica of the integrated data [2]. Within the context of virtual data integration, a popular approach is provided by *Data Federation* systems. Traditionally defined as a type of metadatabase management system, these systems follow the virtual paradigm by transparently mapping multiple autonomous databases – often relational – into a single, unified federated relational database [3].

Ontology-Based Data Access (OBDA) [4] also implements a virtual approach. OBDA allows exposing and querying of a (typically relational) data source as a *Virtual Knowledge Graph* (VKG) built around an *ontology* that formally represents knowledge as a set of concepts and roles within a domain. Ontologies in OBDA are expressed in lightweight conceptual languages from the *DL-Lite* family of description logics [5], making it possible to interpret data stored in a single relational database in a semantically rich way and with a terminology closer to the one of domain experts. This facilitates users' access to complex data, without sacrificing performance, as OBDA systems nowadays are capable of supporting efficient querying of large amounts of data.

RuleML+RR'24: Companion Proceedings of the 8th International Joint Conference on Rules and Reasoning, September 16–22, 2024, Bucharest, Romania

mdipanfilo@unibz.it (M. Di Panfilo)
0000-0002-9284-2488 (M. Di Panfilo)

Building on the same virtual roots, the combination of OBDA — to facilitate access to complex data — and Data Federation — to integrate multiple data sources — appears particularly appealing. While existing OBDA systems can be straightforwardly layered on top of data federation systems, this solution leads to inefficient native queries. Several research challenges remain open in order to fully realize the considered combination in real-world scenarios. The goal of the research that we intend to carry out within the PhD is to study such setting in depth and improve the currently available techniques and technologies along different directions. On the one hand, the aim is to optimize the query processing task to make data access as simple, fast, and accurate as possible, while minimizing computational resource requirements. On the other hand, we are also interested in making more convenient for users to access federated heterogeneous data through a VKG, e.g., by providing (semi-)automated means to bootstrap the VKG mappings and/or ontology.

Along these lines, we previously introduced *Ontology-Based Data Federation* (OBDF) [6, 7], a principled framework that combines Data Federation and OBDA, which we evaluated on a data federation system (Teiid), demonstrating substantial query performance improvements. Building on this contribution and to further advance this PhD research, we are now conducting an extensive experimental evaluation of the developed OBDF algorithms, extending our analysis to include two additional data federation systems (Dremio, Denodo). These experiments enable us to assess the generalization capabilities of our approaches, laying the foundation for future research. Specifically, we address the following research question: *do the performance improvements observed in [6, 7] generalize across different data federation systems?* In this paper, we revisit OBDF using an illustrative example and report on the new experiments, which represent the novel contributions of this work.

The paper is structured as follows: Section 2 provides a review of the state of the art, including key concepts and previous work in Ontology-Based Data Access (OBDA), Data Federation, and attempts at their integration. Section 3 introduces the OBDF framework, detailing its formalization, the use of data hints, and the optimization techniques applied, supported by illustrative examples. Section 4 reports on our extended experiments and discusses the results obtained from evaluating the OBDF framework across different data federation systems. Finally, Section 5 concludes the paper with a summary of findings and outlines the next steps in this PhD research, as well as potential future research directions.

2. State of the Art

The core of this PhD research lies in the combination of two techniques: Data Federation and OBDA, with our initial focus on optimizing query answering in the resulting setting. Therefore, we report here the relevant state of the art for each of these two techniques along with previous attempts towards their combination.

2.1. Ontology-Based Data Access (OBDA)

OBDA [8, 9] is a widely used paradigm developed since the mid-2000s [4, 10], aimed at facilitating access to large-scale databases through domain ontologies. The classical OBDA framework comprises three core components: the user-facing ontology $\mathcal{T}$ consisting of a *DL-Lite* TBox, the underlying database $\mathcal{D}$, and the mappings $\mathcal{M}$ that connect the database data to the ontology. We call the database schema Σ and the database instance D. In traditional OBDA query answering, a SPARQL user query Q over the ontology $\mathcal{T}$ is first translated into a SQL query q over the database, which is then evaluated over D. Given a so-called *OBDA specification* $(\mathcal{T}, \mathcal{M}, \Sigma)$, the process of producing q out of Q is carried out in two steps: *(i) query rewriting*, which is the process of rewriting Q into a query $Q_{\mathcal{T}}$ that takes into account the intensional knowledge in $\mathcal{T}$; *(ii) query unfolding*, which is the process of translating the rewritten query $Q_{\mathcal{T}}$ based on the mappings $\mathcal{M}$ into a SQL query that can be evaluated over the database instance D.

Query optimization [11] occurs during the translation process, to produce a query that can be efficiently executed over the database. To simplify the query answering procedure, the concept of $\mathcal{T}$-mappings was introduced [12, 13]. This approach involves encoding the axioms from $\mathcal{T}$ into $\mathcal{M}$ to obtain an extended mapping set $\mathcal{M}_{\mathcal{T}}$, allowing $\mathcal{T}$ to be disregarded during query answering, with only $\mathcal{M}_{\mathcal{T}}$ being considered. Therefore, we can focus our attention on the unfolding. In this work we consider the variant of the unfolding where a *join of union of conjunctive queries* (JUCQ) [14, 15] is produced at the intermediate step. This strategy is also the one implemented in the state-of-the-art OBDA system Ontop [16]. This JUCQ query will be further optimized through algebraic transformations, often by pushing the joins at the bottom of the algebraic tree, where they can often be simplified, to obtain a query that has the shape of a *union of conjunctive queries* (UCQ) [17].

2.2. Data Federation

Data Federation is a well-studied problem in several related fields such as Database and the Semantic Web, as reported in [3]. The primary goal of data federation is *federated query answering*, which entails accessing multiple, potentially heterogeneous data sources through a unified schema, often referred to as a *virtual database* (VDB). To compute the answers for a federated query over the VDB, a data federation system can delegate operations (e.g., joins) or entire sub-queries to individual data sources, or perform them itself. We thus distinguish between: *local operations*, which are handled within a single source, and *federated operations*, which affect more than one source and whose execution, which is computationally more expensive, involves the data federation system.

On these bases, federated query answering generally involves several key steps: identifying the data sources relevant to the query components and partitioning the query into sub-queries; devising a plan for evaluating these sub-queries, including determining the order of execution and the types of joins to be used for merging sub-query results; and finally, executing the sub-queries on the identified sources and combining the results according to the established query plan [3]. To carry out these steps, most data federation systems rely on the *relational model* [18]. Query answering in the relational model typically relies on SQL and *relational algebra* as the underlying query formalization framework.

In recent years, there have been significant advancements in data federation systems in both academia and industry. These advancements have led to mature implementations, including open-source and/or freely available systems such as Teiid[1], Denodo[2], and Dremio[3], which combine federation capabilities, data security, and, not least, usable interfaces.

2.3. Combining Data Federation and OBDA

Although a combination of OBDA and Data Federation appears both reasonable – given that both approaches are inherently virtual – and practically interesting, as it would extend OBDA capabilities beyond a single source and make federated queries more accessible to domain experts, this research direction has been sparsely explored in the literature. To the best of our knowledge, few works have attempted to fully integrate these approaches.

An interesting work in this area is Obi-Wan [19], an OBDA system capable of integrating heterogeneous data sources, including relational, graph-based, and NoSQL databases. Obi-Wan follows the classical OBDA framework by rewriting queries based on the ontology and mappings and then evaluating these queries using a mediator across multiple heterogeneous data sources. However, Obi-Wan primarily serves as a proof-of-concept and does not include specific optimization techniques tailored to the federated setting, limiting its applicability in real-world, complex scenarios. Other systems like Squerall [20] and PolyWeb [21, 22], also attempt to address the integration of heterogeneous data sources by leveraging OBDA mappings. However, they lack reasoning support and therefore do not qualify as fully-fledged OBDA systems according to the definition in the literature [8]. Overall, the mentioned works highlight the challenges faced and the substantial effort required when building OBDA systems over federated data sources from scratch.

[1] https://teiid.io/

[2] https://www.denodo.com/en

[3] https://www.dremio.com

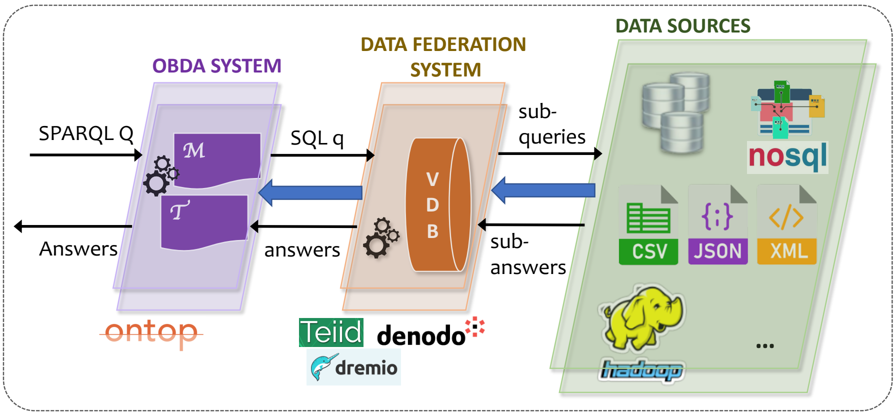


Figure 1: OBDF framework and query answering procedure. [6]

Our OBDF framework [6, 7] takes a different approach, aiming to combine the strengths of state-of-the-art OBDA and Data Federation systems. Rather than reinventing the process of federated query answering, OBDF leverages existing, highly-optimized data federation systems and enhances them through a refined query rewriting process. This process is designed to minimize the computational overhead associated with federated joins, thereby ensuring that queries can be executed more efficiently by the data federation systems. To achieve this, OBDF incorporates data hints and hint-based optimization techniques, which are discussed in detail in the following section. Furthermore, the OBDF framework has been tested on an adapted version of the Berlin SPARQL Benchmark (BSBM) [23] using a single data federation system (Teiid). This evaluation will be extended to three data federation systems and elaborated upon in this paper.

3. OBDF Framework

This section presents the OBDF framework [6, 7], whose extension and validation are part of this PhD research. We focus here on the core concepts and intuitions behind OBDF also through the discussion of an illustrative example, referring the reader to [6] for further specific details about the algorithms implementing OBDF.

3.1. Overview and Formalization

The main objective of OBDF is to perform query answering using existing optimized OBDA and data federation systems. To capture the semantics of such combination, we formally introduce the notion of *OBDF specification*, as a triple $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \Sigma_{\mathbb{S}})$ where $\mathcal{T}$ is an ontology, $\Sigma_{\mathbb{S}}$ is a VDB schema over (heterogeneous) data sources $\mathbb{S}$, and $\mathcal{M}$ is a set of mappings from $\Sigma_{\mathbb{S}}$ to $\mathcal{T}$. This definition is derived from the established one of OBDA specification, where we replace the database schema Σ with the VDB schema $\Sigma_{\mathbb{S}}$.

Figure 1 depicts the full process of query answering in OBDF and the involved systems. A SPARQL query Q is posed over the VKG exposed by the OBDA system. The OBDA system is responsible to *rewrite* the query with respect to the ontology $\mathcal{T}$ and in a second step to *unfold* it according to the mappings $\mathcal{M}$ of the OBDA system. The resulting SQL query q is forwarded to the data federation system, like Teiid, Denodo, or Dremio, to be evaluated over the VDB data instance $\mathbb{D}$ through the issuing of sub-queries to the involved data sources. This way, the federation system facilitates the integration of potentially heterogeneous data sources, enabling the OBDA system to interact with it as it would with a single relational database in a standard OBDA setting.

The OBDF framework inherits challenges from both OBDA and data federation systems, along with new challenges that emerge from their interaction. The OBDA system alone is not aware of the different types of sources and naive query rewriting and unfolding of a SPARQL query can generate inefficient queries for the data federation system. For instance, an unfolding typically contains several joins, which poses a challenge in the federated setting where many of these joins could be federated, thus expensive to compute. In a data federation setting it is also likely that the data is spread over different sources and thus results are overlapping and redundant, which can further complicate query processing. If this system is not optimized, it can lead to inefficient operations and does not scale for big data analysis.

To address these challenges, we introduced a novel unfolding procedure for OBDF [6], which relies on specific statistics computed over the sources. We call these *data hints*.

3.2. Data Hints

The execution of federated joins in federated query answering is inherently time-intensive. A pivotal strategy in OBDF query optimization involves the preliminary identification of federated joins potentially present in SQL query translations. This step facilitates their query-time restructuring into forms more amenable to efficient evaluation by data federation systems. As articulated in the standard OBDA formalization [8] and expounded in [15], all possible joins in SPARQL to SQL translations can be identified by analyzing the ontology and mappings within the OBDA specification. A central aspect is to provide an algorithm for doing so, as proposed in [6] and may be the subject of future work.

Moreover, scenarios involving multiple data sources inherently involve data redundancy. Within OBDF, the top-level ontology provides a unified semantic overlay for diverse data sources. Our approach is to analyze the ontology and the mappings in order to identify redundancies across different data sources and thus optimize SQL query translations by pruning redundant subqueries.

To encapsulate the query optimization methodology in OBDF, three types of data hints were thus identified:

- *Empty Federated Join* $\mathcal{E}$ refers to joins across distinct data sources that are expected to yield no results within the current data federation context. Formally, $\mathsf{A} \bowtie_{\mathbb{D}} \mathsf{B} = \emptyset$, where $\mathsf{A} \bowtie \mathsf{B}$ denotes a federated join, and the subscript $_{\mathbb{D}}$ indicates that the equivalence must hold in the current federated data instance $\mathbb{D}$.

- *Containment Redundancy* $\mathcal{C}$ refers to detectable redundancies that are across data sources. Formally, given a data instance $\mathbb{D}$ and two algebraic expressions A and B, we say that A is data-contained in B, denoted as $\mathsf{A} \subseteq_{\mathbb{D}} \mathsf{B}$, if the set of answers of A over the data instance $\mathbb{D}$ is contained in the set of answers B over the data instance $\mathbb{D}$. We use $\mathsf{A} \equiv_{\mathbb{D}} \mathsf{B}$ to indicate that $\mathsf{A} \subseteq_{\mathbb{D}} \mathsf{B}$ and $\mathsf{B} \subseteq_{\mathbb{D}} \mathsf{A}$, meaning that A and B produce equivalent results within the data instance $\mathbb{D}$.

- *Materialized Views* M, as discussed in [17], are known to significantly improve query processing performance and are supported by some data federation systems such as Teiid and Dremio.

In [6, §5.1], we presented an algorithm $\mathsf{hintify}$ on how to extract the data hints from a specific data instance $\mathbb{D}$. This algorithm enumerates possible federated joins and redundancies as mentioned above, restricted to VDB relations labeled as *static* (i.e., whose content is not expected to change), and analyzes them on the current data instance $\mathbb{D}$ (e.g., checking whether a federated join is empty) to derive hints that can be expected to hold across data instances. Materialized views are introduced for selected non-empty federated joins, according to customizable heuristics and pending possible further research work to optimize such selection.

3.3. Hints-Based Query Optimization

In our exploration of optimizing SQL translations of SPARQL queries within the OBDF framework, we focus on three integral components: *query optimization rules*, *cost model*, and *unfolding algorithm*.

Query Optimization Rules Our methodology extends standard optimization rules in OBDA, like the self join elimination rule [24], and introduces a comprehensive set of query optimization rules, shown

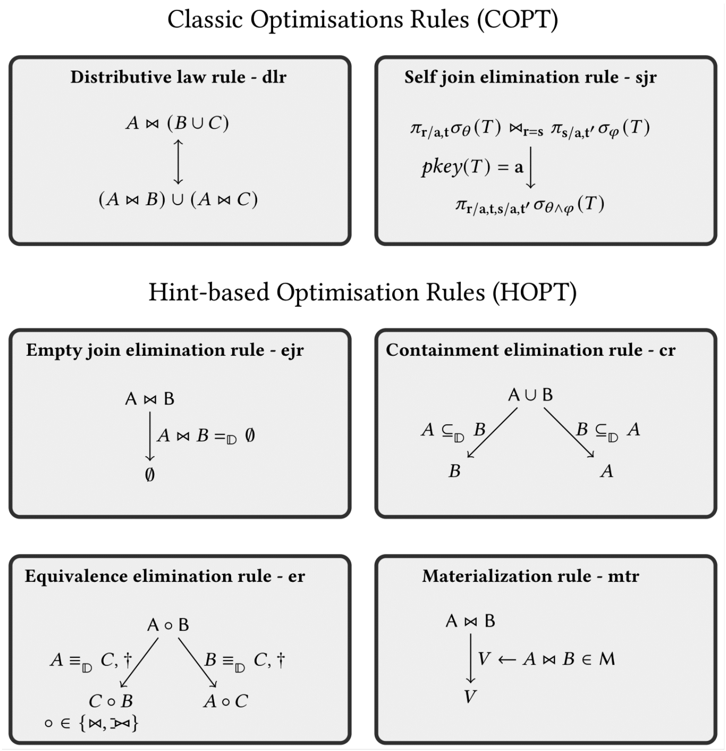

Classic Optimisations Rules (COPT)
Distributive law rule - dlr
$A \bowtie (B \cup C)$
$(A \bowtie B) \cup (A \bowtie C)$
Self join elimination rule - sjr
$\pi_{r/a,t}\sigma_\theta(T) \bowtie_{r=s} \pi_{s/a,t'}\sigma_\varphi(T)$
$pkey(T) = a$
$\pi_{r/a,t,s/a,t'}\sigma_{\theta\wedge\varphi}(T)$
Hint-based Optimisation Rules (HOPT)
Empty join elimination rule - ejr
$A \bowtie B$
$A \bowtie B =_{\mathbb{D}} \emptyset$
$\emptyset$
Containment elimination rule - cr
$A \cup B$
$A \subseteq_{\mathbb{D}} B$
$B \subseteq_{\mathbb{D}} A$
B
A
Equivalence elimination rule - er
$A \circ B$
$A \equiv_{\mathbb{D}} C, \dagger$
$B \equiv_{\mathbb{D}} C, \dagger$
$C \circ B$
$A \circ C$
$\circ \in \{\bowtie, \rtimes\}$
Materialization rule - mtr
$A \bowtie B$
$V \leftarrow A \bowtie B \in M$
V

Figure 2: List of standard optimization rules applied by OBDA systems (first group), and the new rules specific to OBDF (second group). The former is not complete (e.g., we omit trivial transformations like commutativity rule). The letters A, B, C denote relational algebra expressions, whereas T denotes base relations. We use standard relational algebra notation, where for conciseness we introduce the abbreviation $\pi_{r_1/a_1,\dots,r_n/a_n}$ for the combination $\rho_{r_1/a_1,\dots,r_n/a_n}\pi_{a_1,\dots,a_n}$ of *projection* and *renaming*.

in Figure 2. These rules are determined on the principles of semantic equivalence and the utilization of pre-computed data hints as in Section 3.2.

The top part of Figure 2 shows the following classic optimization rules:

- The *distributive rule* (dlr) is typically used to "push" joins into unions, in order to transform JUCQ into UCQ unfoldings. This is the core optimization applied by state-of-the-art OBDA systems.
- The *self-join elimination rule* (sjr) handles the explosion of self-join operations during the unfolding, arising from the mismatch between the n-ary model of relational databases against the ternary model of RDF graphs.

The bottom part of Figure 2 introduces the novel *hint-based rules*:

- The *Empty Join Elimination Rule* (ejr) removes unnecessary empty joins in SQL queries. The rule is applicable if the empty federated join hint $\mathsf{A} \bowtie_{\mathbb{D}} \mathsf{B} = \emptyset$ belongs to the set of hints.
- The *Containment Elimination Rule* (cr) aims to eliminate redundant unions in SQL queries using the identified set of containment redundancy hints.
- The *Equivalence Elimination Rule* (er) modifies or replaces join operations in SQL queries. It checks for data equivalence, i.e., two-sided containment, similar to rule *cr*, and also assesses a condition † to ensure that the cost of the new expression is less than the original cost. The method for cost computation is detailed below.
- Finally, the *Materialization Rule* (mtr) enhances the efficiency of federated joins in SQL queries by utilizing pre-computed views. This rule replaces complex join operations with these more optimized views.

Cost Model Our approach incorporates a straightforward cost model [6, §5.3] that assigns an evaluation cost to each relational algebra expression. This model follows a set of heuristic principles:

- Preference for local joins over federated joins,
- Prioritization of efficient data sources,
- Elimination of redundant or empty sub-expressions,
- Utilization of materialized views, when available.

In addition to these federation-specific heuristics, the model integrates standard OBDA heuristics, providing a comprehensive framework for evaluating query efficiency.

Hints-Based Unfolding Algorithm The hints-based unfolding algorithm defines how the query optimization rules and the cost model are integrated in the query processing workflow of an OBDF framework. The algorithm $\mathsf{unfold}_{\mathsf{OBDF}}$ was originally presented in [6, §5.4] and since then it has been further streamlined based on implementation and evaluation feedback, and is currently being improved for additional optimizations.

3.4. Example

In this section, we provide concrete examples demonstrating the application of the data hints discussed in Section 3.2, in conjunction with the $\mathsf{unfold}_{\mathsf{OBDF}}$ [6] algorithm. For simplicity in this example, we will focus primarily on the key objective of the cost model, which is to *prefer local joins over federated joins*. This should suffice to illustrate the algorithm's operation and main intuitions.

Figure 3 depicts the entire process of query optimization. As a first step, we describe all the input variables required by the $\mathsf{unfold}_{\mathsf{OBDF}}$ algorithm:

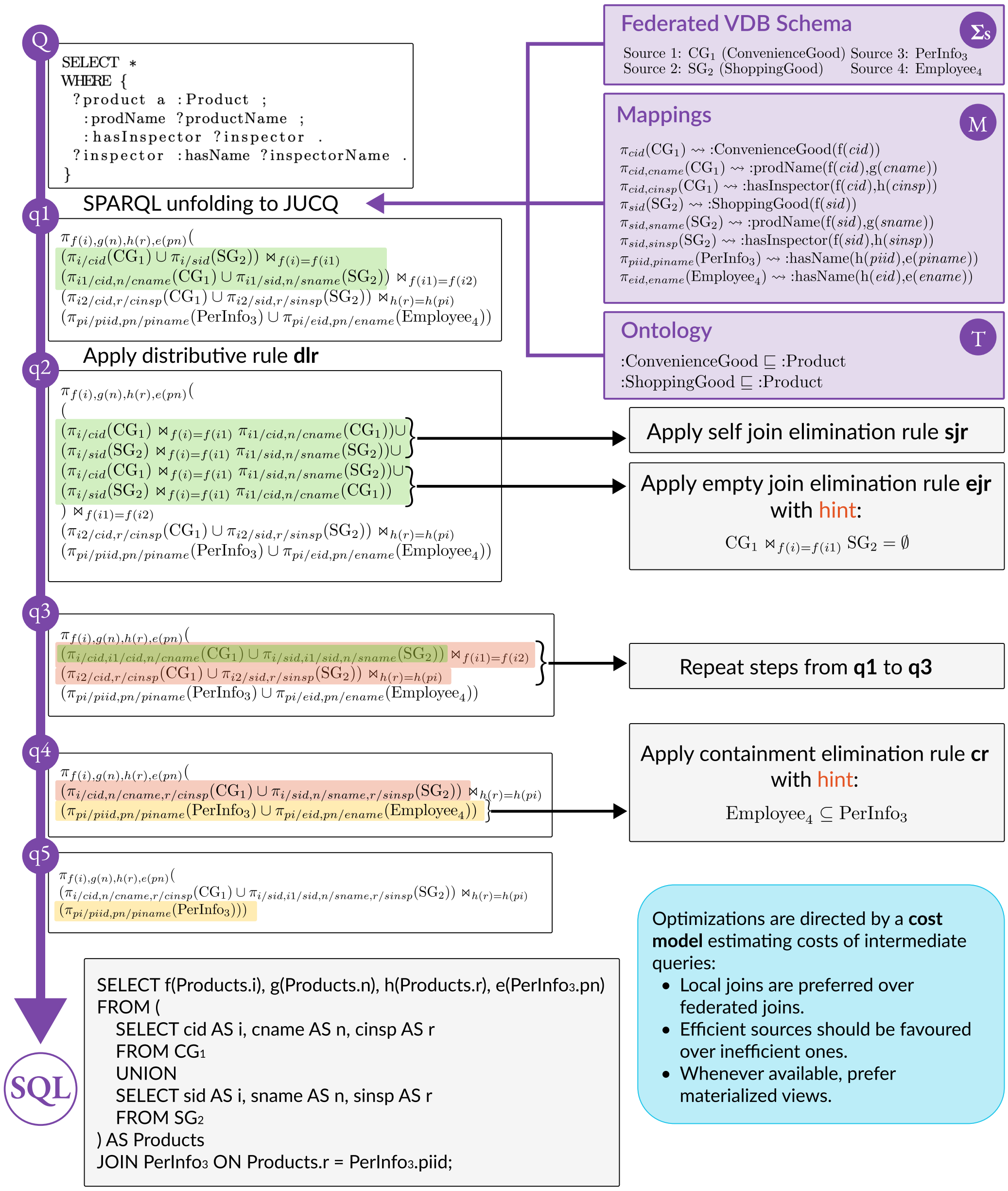
OBDF Specification $\mathcal{F} = (\mathcal{T}, \mathcal{M}, \Sigma_{\mathbb{S}})$:

The OBDF specification $\mathcal{F}$ consists of the ontology $\mathcal{T}$ the mappings $\mathcal{M}$ and the Federated VDB schema $\Sigma_{\mathbb{S}}$. They are depicted with a purple background on the top right of Figure 3.

- The **Ontology** $\mathcal{T}$ consists of two axioms stating that *ConvenienceGoods* and *ShoppingGoods* are both *Products*.
- The **Federated VDB Schema** $\Sigma_{\mathbb{S}}$ (Virtual Database Schema) comprises relation names used by the federation system, capable of integrating multiple federated sources. In our example, subscripts are used to specify the exact federated source for each relation, with the sources ranging from Source 1-4.
- The **Mappings** $\mathcal{M}$ define how the data in the data sources is mapped to the concepts and roles of the virtual knowledge graph. In our example, the source component, positioned on the left side of the mapping, is represented by a query in algebraic form that operates over the data sources. Conversely, the target component on the right is described by a first-order logic atom, which includes IRI templates f, g, h and details on the transformation of database values into RDF literals. The mappings show that ConvenienceGoods and their attributes are sourced from the table CG_1, while ShoppingGoods are derived from the table SG_2. Additionally, the role $hasName$, which specifies the names of inspectors, links to the tables $\mathsf{PerInfo}_3$ and $\mathsf{Employee}_4$.

SPARQL Query Q:

The topmost box Q of Figure 3 contains the SPARQL query of our example. We use a SPARQL query to retrieve information about all products. This includes each product's name, the inspector associated with the product, and the inspector's name.


Q
q1
SPARQL unfolding to JUCQ
$\pi_{f(i),g(n),h(r),e(pn)}($
$(\pi_{i/cid}(\mathrm{CG}_1) \cup \pi_{i/sid}(\mathrm{SG}_2)) \bowtie_{f(i)=f(i1)}$
$(\pi_{i1/cid,n/cname}(\mathrm{CG}_1) \cup \pi_{i1/sid,n/sname}(\mathrm{SG}_2)) \bowtie_{f(i1)=f(i2)}$
$(\pi_{i2/cid,r/cinsp}(\mathrm{CG}_1) \cup \pi_{i2/sid,r/sinsp}(\mathrm{SG}_2)) \bowtie_{h(r)=h(pi)}$
$(\pi_{pi/piid,pn/piname}(\mathrm{PerInfo}_3) \cup \pi_{pi/eid,pn/ename}(\mathrm{Employee}_4))$
q2
Apply distributive rule **dlr**
$\pi_{f(i),g(n),h(r),e(pn)}($
$($
$(\pi_{i/cid}(\mathrm{CG}_1) \bowtie_{f(i)=f(i1)} \pi_{i1/cid,n/cname}(\mathrm{CG}_1))\cup$
$(\pi_{i/sid}(\mathrm{SG}_2) \bowtie_{f(i)=f(i1)} \pi_{i1/sid,n/sname}(\mathrm{SG}_2))\cup$
$(\pi_{i/cid}(\mathrm{CG}_1) \bowtie_{f(i)=f(i1)} \pi_{i1/sid,n/sname}(\mathrm{SG}_2))\cup$
$(\pi_{i/sid}(\mathrm{SG}_2) \bowtie_{f(i)=f(i1)} \pi_{i1/cid,n/cname}(\mathrm{CG}_1))$
$) \bowtie_{f(i1)=f(i2)}$
$(\pi_{i2/cid,r/cinsp}(\mathrm{CG}_1) \cup \pi_{i2/sid,r/sinsp}(\mathrm{SG}_2)) \bowtie_{h(r)=h(pi)}$
$(\pi_{pi/piid,pn/piname}(\mathrm{PerInfo}_3) \cup \pi_{pi/eid,pn/ename}(\mathrm{Employee}_4))$
q3
$\pi_{f(i),g(n),h(r),e(pn)}($
$(\pi_{i/cid,i1/cid,n/cname}(\mathrm{CG}_1) \cup \pi_{i/sid,i1/sid,n/sname}(\mathrm{SG}_2)) \bowtie_{f(i1)=f(i2)}$
$(\pi_{i2/cid,r/cinsp}(\mathrm{CG}_1) \cup \pi_{i2/sid,r/sinsp}(\mathrm{SG}_2)) \bowtie_{h(r)=h(pi)}$
$(\pi_{pi/piid,pn/piname}(\mathrm{PerInfo}_3) \cup \pi_{pi/eid,pn/ename}(\mathrm{Employee}_4))$
q4
$\pi_{f(i),g(n),h(r),e(pn)}($
$(\pi_{i/cid,n/cname,r/cinsp}(\mathrm{CG}_1) \cup \pi_{i/sid,n/sname,r/sinsp}(\mathrm{SG}_2)) \bowtie_{h(r)=h(pi)}$
$(\pi_{pi/piid,pn/piname}(\mathrm{PerInfo}_3) \cup \pi_{pi/eid,pn/ename}(\mathrm{Employee}_4))$
q5
$\pi_{f(i),g(n),h(r),e(pn)}($
$(\pi_{i/cid,n/cname,r/cinsp}(\mathrm{CG}_1) \cup \pi_{i/sid,i1/sid,n/sname,r/sinsp}(\mathrm{SG}_2)) \bowtie_{h(r)=h(pi)}$
$(\pi_{pi/piid,pn/piname}(\mathrm{PerInfo}_3)))$
SQL
Federated VDB Schema
Σ_S
Source 1: CG_1 (ConvenienceGood) Source 3: $\mathrm{PerInfo}_3$
Source 2: SG_2 (ShoppingGood) Source 4: $\mathrm{Employee}_4$
Mappings
M
$\pi_{cid}(\mathrm{CG}_1) \rightsquigarrow$:ConvenienceGood(f(cid))
$\pi_{cid,cname}(\mathrm{CG}_1) \rightsquigarrow$:prodName(f(cid),g($cname$))
$\pi_{cid,cinsp}(\mathrm{CG}_1) \rightsquigarrow$:hasInspector(f(cid),h($cinsp$))
$\pi_{sid}(\mathrm{SG}_2) \rightsquigarrow$:ShoppingGood(f(sid))
$\pi_{sid,sname}(\mathrm{SG}_2) \rightsquigarrow$:prodName(f(sid),g($sname$))
$\pi_{sid,sinsp}(\mathrm{SG}_2) \rightsquigarrow$:hasInspector(f(sid),h($sinsp$))
$\pi_{piid,piname}(\mathrm{PerInfo}_3) \rightsquigarrow$:hasName(h($piid$),e($piname$))
$\pi_{eid,ename}(\mathrm{Employee}_4) \rightsquigarrow$:hasName(h(eid),e($ename$))
Ontology
T
:ConvenienceGood $\sqsubseteq$:Product
:ShoppingGood $\sqsubseteq$:Product
Apply self join elimination rule **sjr**
Apply empty join elimination rule **ejr** with hint:
$\mathrm{CG}_1 \bowtie_{f(i)=f(i1)} \mathrm{SG}_2 = \emptyset$
Repeat steps from **q1** to **q3**
Apply containment elimination rule **cr** with hint:
$\mathrm{Employee}_4 \subseteq \mathrm{PerInfo}_3$

```
SELECT *
WHERE {
 ?product a :Product ;
  :prodName ?productName ;
  :hasInspector ?inspector .
 ?inspector :hasName ?inspectorName .
}
```

```
SELECT f(Products.i), g(Products.n), h(Products.r), e(PerInfo₃.pn)
FROM (
   SELECT cid AS i, cname AS n, cinsp AS r
   FROM CG₁
   UNION
   SELECT sid AS i, sname AS n, sinsp AS r
   FROM SG₂
) AS Products
JOIN PerInfo₃ ON Products.r = PerInfo₃.piid;
```

Optimizations are directed by a **cost model** estimating costs of intermediate queries:

- Local joins are preferred over federated joins.
- Efficient sources should be favoured over inefficient ones.
- Whenever available, prefer materialized views.

Figure 3: Example of query optimization using the Hint-Based Unfolding Algorithm. $q1$ illustrates the relational algebra representation of the SPARQL query Q, produced by standard unfolding techniques. Following the $\text{unfold}_{\text{OBDF}}$ algorithm through steps $q1$ to $q5$, a combination of standard and novel hint-based optimization techniques is applied, effectively reducing the number of federated joins and unions, thereby enhancing query efficiency.

Optimization Data Hints:

In this example we assume that the pre-computation of the hints following the hintify algorithm in [6] has detected the following data hints.

- **Empty Federated Join Hints** $\mathcal{E}$: $\mathrm{CG}_1 \bowtie \mathrm{SG}_2 \equiv_{\mathbb{D}} \emptyset$

  The presence of an empty federated join hint between the relations CG_1 and SG_2 signifies that the sets of ConvenienceGoods in CG_1 and ShoppingGoods in SG_2 are disjoint in this specific database instance, indicating no shared records between them.

- **Containment Redundancy Hints** $\mathcal{C}$: $\mathrm{Employee}_4 \subseteq_{\mathbb{D}} \mathrm{PerInfo}_3$

This containment redundancy hint indicates that, for this specific database instance, all records from the relation $\mathsf{Employee}_4$ are already included within the records of $\mathsf{PerInfo}_3$. Therefore, this implies that the records in $\mathsf{Employee}_4$ are a subset of those in $\mathsf{PerInfo}_3$. Consequently, performing a union of both tables would result in no additional records beyond what is already present in $\mathsf{PerInfo}_3$, underscoring the unnecessary nature of combining $\mathsf{Employee}_4$ with $\mathsf{PerInfo}_3$.

- **Materialized Views Hints** M: $\emptyset$

  In this particular example, no hints related to materialized views have been identified.

The SPARQL query Q is transformed into an SQL query as follows:

- **Q → q1** The initial phase of the $\mathsf{unfold}_{\mathsf{OBDF}}$ algorithm involves invoking the $\mathsf{unfold}_{\mathsf{wrap}}$ algorithm. This algorithm, as detailed in [15, 14], is an unfolding method that results in a query in JUCQ form, as illustrated in the relational algebra expression of $q1$ in Figure 3.

- **q1 → q2** Subsequently, in the transition from $q1$ to $q2$, highlighted in green in Figure 3, the distributive law rule (dlr) is applied to the initial join, effectively pushing the join operation inside the unions. This transformation produces a union of four joins involving the relations CG_1 and SG_2.

- **q2 → q3** In $q2$, several optimization opportunities become apparent. Firstly, the self-join elimination rule (sjr) can be employed to remove the join with the same relation on the renamed primary key cid. More notably, the empty join elimination rule (ejr) can be applied, given the hint that $\mathsf{CG}_1 \bowtie \mathsf{SG}_2 \equiv_{\mathbb{D}} \emptyset$. These optimizations lead to the removal of all joins in the green-highlighted section of $q2$, resulting in the more efficient $q3$. Our cost model supports this optimization from $q1$ to $q3$, as it reduces the number of federated joins from three to two.

- **q3 → q4** The section of $q3$ highlighted in red mirrors the characteristics of the green-highlighted part of $q1$. Thus, the same optimization techniques - dlr, sjr, and ejr - can be applied, leading to the more simplified query $q4$, which contains only one federated join.

- **q4 → q5** In the final transition from $q4$ to $q5$, the $\mathsf{unfold}_{\mathsf{OBDF}}$ algorithm checks for unions where the containment elimination rule (cr) can be applied. Given the containment redundancy hint $\mathsf{Employee}_4 \subseteq_{\mathbb{D}} \mathsf{PerInfo}_3$, the query can be further optimized by removing the $\mathsf{Employee}_4$ relation from the union, as all records in $\mathsf{Employee}_4$ are already present in $\mathsf{PerInfo}_3$.

- **q5 → SQL** The final query, $q5$, represents the optimized execution plan, which is then translated into an SQL query.

The illustrated transformation demonstrates the optimization of the original SPARQL query depicted in Q. Initially, the query involved three federated joins and four federated unions across relations from four different sources. The optimized query in $q5$ significantly reduces complexity, featuring only one federated join and one federated union.

4. Experiments and Results

In this section, we report on the extended experiments we conducted using our hint-based optimization approach, and discuss and analyze the results obtained.

4.1. Experimental Setup and Methodology

We extend the experiments in [6] by expanding our tests from a single data federation system to three different systems: Teiid, Denodo, and Dremio. This broader evaluation allows us to assess the general effectiveness of our optimizations across diverse data federation systems.

As in [6], our evaluation is based on the *Berlin SPARQL Benchmark (BSBM)* from [23],[4] which we adapt here to the OBDF scenario. This well known synthetic benchmark comes in two aligned RDF and SQL versions, which together allow evaluating and comparing query answering performance of RDF triplestores, RDBMSs, and OBDA systems as well as demonstrated in [25, 26]. BSBM is designed around an e-commerce scenario. It simulates a diverse marketplace where multiple vendors offer products coming from producers and subsequently reviewed by consumers. Synthetic data for such scenario can be created through the *BSBM data generator*. Given a *scale factor* n corresponding to the desired number of products, the generator produces a corresponding dataset.

We start from the OBDA instance $((\mathcal{T}, \mathcal{M}, \Sigma), D)$ of BSBM (for a given scale factor n) and we aim at transforming it into an OBDF instance $((\mathcal{T}, \mathcal{M}', \Sigma'), \mathbb{D})$. There, $\mathbb{D}$ is a VDB instance obtained using partitioning and replication to reorganize D into multiple federated database instances D_i, which we allocate to either: (i) a set of relational sources, to test performance in a *homogeneous setting* (hom); or (ii) a mix of relational and NoSQL sources, to test a *heterogeneous setting* (het)[5]. Σ' is the VDB schema for $\mathbb{D}$, which matches Σ except for partitioned and replicated tables and is obtained by setting up multiple data federation systems. Lastly, $\mathcal{M}'$ are the new mappings derived from $\mathcal{M}$, which produce the same virtual ABox, which we use with other specification components to configure the tested system.

Besides the aforementioned hom and het settings for the obtained OBDF instances, we also consider two reference *centralized settings* sc1 and sc2, where all data is stored in a single PostgreSQL database directly accessed by the OBDA system without a data federation system: sc1 employs OBDA instance $((\mathcal{T}, \mathcal{M}, \Sigma), D)$ for the original BSBM, whereas sc2 employs OBDA instance $((\mathcal{T}, \mathcal{M}', \Sigma'), \mathbb{D})$ where data partitioning and replication are introduced, although in a centralized setting.

To conduct our tests, we slightly adapt and reuse the 12 queries of BSBM Explore. We evaluate query answering with and without our approach focusing on execution time, measured in milliseconds (ms), which refers to the total time required to evaluate either an individual query Q or a query mix M, the latter comprising one execution of each of the 12 BSBM Explore queries. For a query mix, the execution time is determined by geometrically averaging the execution times of its individual queries, calculated as $T_M = \sqrt[|M|]{\prod_{Q \in M} T_Q}$, where T_M represents the mix execution time and T_Q represents the execution time for each query. Note that using a geometric mean to evaluate a query mix, rather than a simple arithmetic mean or sum, helps to prevent slower queries with large execution times from disproportionately influencing the results, particularly when the execution times of faster queries are several orders of magnitude smaller, as observed in some BSBM queries.

We evaluate the aforementioned metrics for different configurations of our evaluation environment, and specifically:

- for different scale factors n of the BSBM benchmark, 2k, 20k 200k.
- for each setting sc1, sc2, hom, het;
- for each data federation system Teiid, Dremio, Denodo (only for federated settings hom, het);
- with or without our hint-based optimization approach, considering these three conditions:
  - base: no hint-based optimization (our baseline, all four settings);
  - opt: hint-based optimization without materialized views (only for settings hom, het);
  - optm: hint-based optimization also using materialized views (only for settings hom, het).

For each configuration we run a number n_w of warm up mixes, which serve to initialize caches and whose execution time we do not measure, followed by a number n_t of test mixes, over which we compute and average our metrics. We set $n_t = 20$ to have sufficient mixes to conduct statistical analyses. We set $n_w = 50$ after having verified that measured average execution times (on a moving average on $n_t = 20$ mixes) do not change considerably by running further warm up mixes.

Note that we report optimizations with and without materialized views for two primary reasons. First, the application of materialized views in our experiments was limited to only 2 out of 12 queries

[4] http://wbsg.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/index.html

[5] Specifically PostgresSQL, MariaDB, MySQL, MS SQL Server for setting hom and PostgresSQL, MS SQL Server, MongoDB and CSV files for setting het. See [6] for further details, such as the allocations of tables to data sources.

Table 1

Mean execution times (ms) for each Query mix M, averaged over 20 test runs. The table organizes execution times by baselines (sc1, sc2) and data federation systems in the hom and het settings. It also categorizes the data based on optimization levels: base (not optimized), opt (optimized), and optm (optimized with materialized views), alongside the BSBM scale (2k-200k). Each Query mix M corresponds to the geometric mean of the execution times of queries Q_1 - Q_{12}.

		2k			**20k**			**200k**		
setting	**system**	**base**	**opt**	**optm**	**base**	**opt**	**optm**	**base**	**opt**	**optm**
sc1		41			44			69		
sc2		150			304			873		
hom	**Teiid**	107	60	**59**	282	72	**71**	1408	126	**108**
	Denodo	143	87	**85**	415	113	**105**	2098	183	**154**
	Dremio	352	175	**155**	537	198	**178**	1660	294	**243**
	avg	200	107	**100**	411	128	**118**	1722	201	**168**
het	**Teiid**	533	299	**217**	5120	2179	**1139**	51690	17896	**6877**
	Denodo	294	187	**157**	1818	926	**629**	17938	7963	**3837**
	Dremio	780	489	**418**	2394	1321	**973**	13141	8440	**4652**
	avg	536	325	**264**	3111	1475	**914**	27590	11433	**5122**

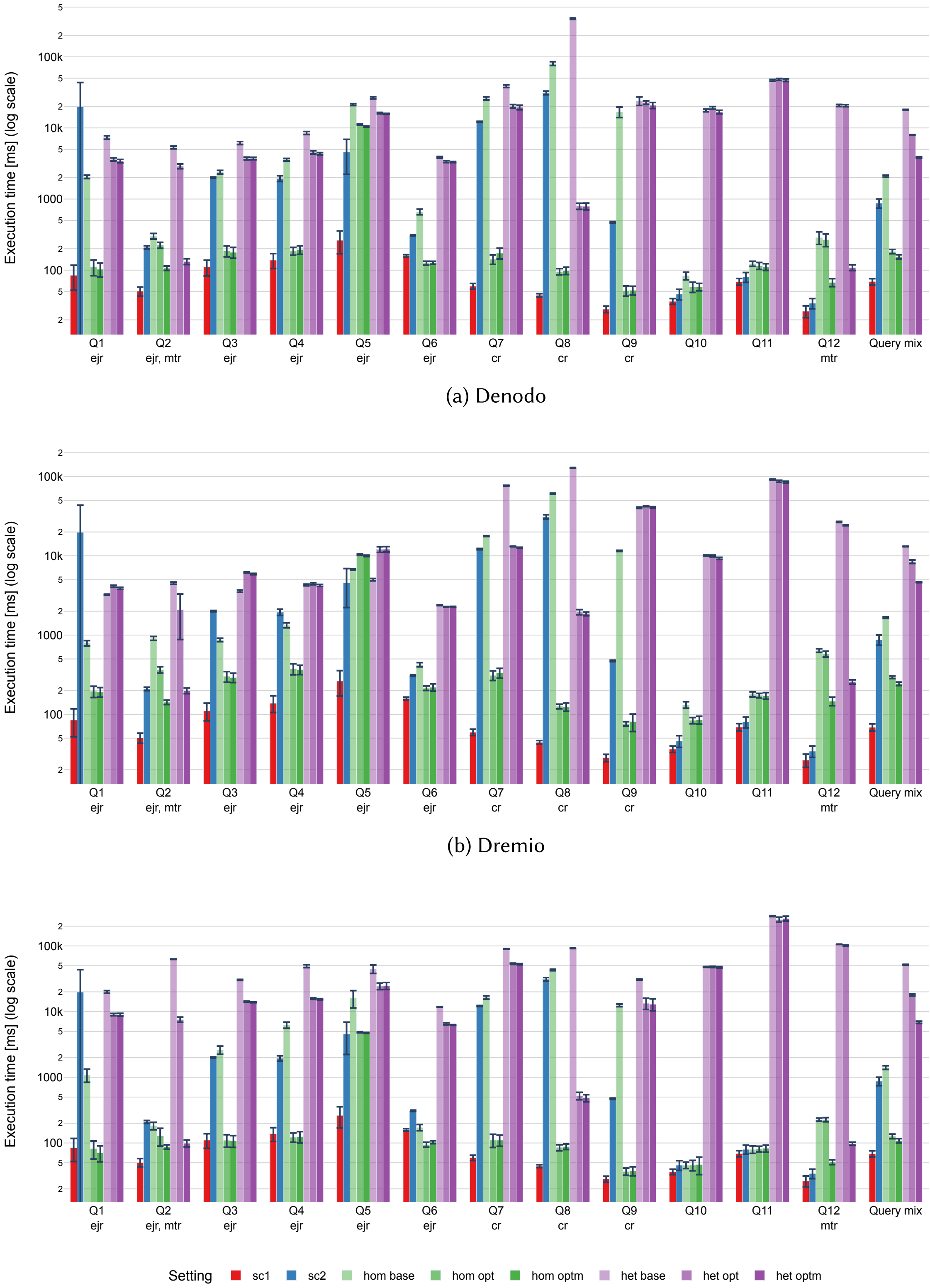
(Q_2 and Q_{12}). Second, materialized views come with a significant drawback: they require substantial storage space. In contrast, the only negative impact of leveraging empty federated join and containment redundancy hints, which we consider in opt, is a negligible increase of query reformulation time.

4.2. Results

Figure 4 shows the average execution time of the aforementioned settings for Denodo (a), Dremio (b) and Teiid (c) at a BSBM scale of 200k, for each query Q_1-Q_{12} and at query mix level. Furthermore, the x-axis also displays the hint-based optimization rules that have been adopted in the optimizations of the single queries. We observe that the application of hints has been very selective among the queries. The empty join elimination rule (ejr) together with the empty federated join hint has been used to optimize queries Q_1-Q_6. Meanwhile the containment redundancy hint and containment elimination rule (cr) has been used to optimize Q_7-Q_9. No data hints and rules have been applied to queries Q_{10}-Q_{11}, whereas materialization rule (mtr) has been used in Q_2 and Q_{12}.

An analysis of Figures 4 aligns with the expectations and validates the initial findings presented in [6]. The red bar, representing centralized setting sc1 without data partitioning, consistently shows the shortest execution times, particularly in contrast to centralized setting sc2 (blue bar), where data partitioning is implemented. In general, the execution times relative to setting sc2 are slightly lower than those for the baseline homogeneous setting hom, indicated by the first green bar, demonstrating that the introduction of a data federation layer has no major impact on query execution time. This is especially true at lower BSBM scales like 2k, where we observe sc2 underperforms compared to the homogeneous setting in queries Q_4, Q_7, Q_8, and Q_9. At higher BSBM scales, the blue bar relative to sc2 remains below the first green bar for the hom baseline, with Q_1 being an exception due to high data variance, as indicated by the error bars, and attributable to different PostgreSQL query execution plans for different instantiations of Q_1 depending on the specific choice of query template parameters.

Figure 4 reveals a key observation, appreciable by noting that for each federation system and query, the execution times are displayed in three stages (via different shades of color) – first without optimizations (base), then with optimizations but without materialization, and finally with materializations. In nearly every instance, the execution time with optimizations is consistently lower than when no optimizations are applied. The only exceptions are queries Q_{10} and Q_{11}, where no optimization rules were triggered, and queries Q_1, Q_3, and Q_5 in Dremio, which present an opportunity for further analysis to gain insights into the unexpected behavior.



(a) Denodo

(b) Dremio

(c) Teiid

Figure 4: Average execution time (± standard deviation) over 20 test runs for the federation systems Denodo (a), Dremio (b), and Teiid (c) over the BSBM scale of 200k. The x-axes represent queries Q_1 to Q_{12} and the query mix geometric averages, with the additional indication of applied hint-based optimization rules.

Further supporting these observations, the right-most set of bars labeled as query mix represents the geometric average of all the queries, where the reduction in execution time is clearly evident, further confirming the effectiveness of the hint-based optimizations. Table 1 provides the specific values corresponding to the right-most set of bars for the query mix of Figure 4 and also includes values for lower BSBM scales of 2k and 20k. As previously stated, the aggregated results of the query mix indicate consistent improvements. Notably, when using the materialized view and at a BSBM scale of 200k, we observe an average reduction in execution time of the query mix from 1.722 ms to 168 ms in the homogeneous setting and from 27.590 ms to 5.122 ms in the heterogeneous setting. This corresponds to an approximate 90% reduction in execution time for the homogeneous setting and a 77% reduction in the heterogeneous setting. Additionally, the table shows that these values decrease as the BSBM dataset scale increases, underscoring the enhanced efficiency of the optimizations in large data contexts.

Across all federation systems and BSBM scales, the inclusion of hint optimizations consistently yielded the shortest execution times, as revealed by Table 1. This outcome validates the effectiveness of hint-based optimization techniques proposed by [6] and examined in this study.

5. Conclusions and Future Work

In this paper, we presented an ongoing PhD research tackling the combination of OBDA and Data Federation, both in terms of efficient query evaluation and convenience for users in accessing federated data. We focused here on the former querying aspect, for which we previously introduced the Ontology-Based Data Federation (OBDF) framework [6], and for which we presented here a novel, extended evaluation aimed at assessing generalization of performance improvements across multiple federation systems. Notably, this was achieved by considering two additional data federation systems (Dremio and Denodo) in addition to the one (Teiid) previously considered in [6].

5.1. Summary of Results

The results of the presented extended experiments confirm the performance improvements and the findings from our previous work [6] across all the considered data federation systems. The improvements in query execution times due to OBDF optimizations are substantial, ranging from a 90% reduction (see Table 1) in the considered homogeneous setting anto a 77% reduction in the heterogeneous setting. Overall, the reported extended experiments indicate that the optimization techniques integrated within OBDF are effective regardless of the specific federation system employed, providing evidence for the framework's generalization and adaptation capabilities.

Based on these results, which may possibly further improve due to ongoing enhancements to the OBDF algorithms, we can conclude that efficient OBDA query evaluation over data federation systems is achievable and has been obtained within the scope of the OBDF framework, and we can turn our main focus on the other goal (convenience for users) of this PhD research.

5.2. Next Activities in this PhD Research

The immediate next steps in this PhD research will deal with completing the current revision of the OBDF algorithms. This will involve refining the algorithms, particularly those discussed in the context of hint-based query optimization, and preparing a comprehensive journal publication that encapsulates all contributions and results obtained thus far.

Following that, we will shift our focus from query execution and optimization techniques to supporting OBDF/OBDA users in designing a Virtual Knowledge Base system, with a particular emphasis on defining mappings. Building on the mapping patterns proposed in recent literature, an automated data-driven approach for bootstrapping mappings directly from data sources will be explored, to avoid the need for users to fully perform this task manually. This approach aims to enhance the flexibility and efficiency of the system, thereby facilitating more seamless integration and utilization of diverse data sources.

5.3. Related Future Work Directions

Further extensions of OBDF, based on the results obtained so far, present several promising avenues for future research, to be carried out either within (if opportunities will arise) or beyond this PhD research. These extensions regard:

- *Precomputation of hints and identification of materialized views.* Further research may delve deeper into optimizing the precomputation of hints, particularly focusing on the identification of materialized views. Ideally, this should involve balancing additional storage usage against improvements in query evaluation times due to the use of materialized views, potentially leading to more sophisticated heuristics that are finely tuned for different data environments.

- Algorithm $\mathsf{unfold}_{\mathsf{OBDF}}$. Additional research can be directed toward extending the $\mathsf{unfold}_{\mathsf{OBDF}}$ algorithm, enhancing its capability to handle more complex queries and data scenarios.

- *Support for graph databases.* A related new line of research may focus on integrating graph databases, like Neo4J, as a source within the Data Federation environment. Such source inherently provides additional query language features, such as navigational queries. To fully leverage these new capabilities, one should enrich the query language of the OBDF framework as well, allowing it to support and exploit these navigational features. This research should thus investigate how established query languages can be extended to incorporate the navigational elements typical of graph query languages. The primary challenge will be to reformulate navigational queries within the ontology context, taking full advantage of the graph database's inherent navigational capabilities. This enhancement has the potential to enable the OBDF framework to handle more complex queries, including those that return not only individual nodes or values but also entire paths or sub-graphs, thus significantly expanding the expressiveness and utility of the framework. This future research direction not only aims to enhance the theoretical underpinnings of the OBDF framework but also seeks to develop practical tools and methodologies that can be applied in real-world scenarios, making the system more robust and versatile for various applications leveraging graph data.

Acknowledgments

This work has been carried out while Marco Di Panfilo was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with Free University of Bozen-Bolzano.

References

[1] A. Labrinidis, H. V. Jagadish, Challenges and opportunities with big data, Proc. VLDB Endow. 5 (2012) 2032–2033. doi:10.14778/2367502.2367572.

[2] A. Doan, A. Y. Halevy, Z. G. Ives, Principles of Data Integration, Morgan Kaufmann, 2012.

[3] Z. Gu, F. Corcoglioniti, D. Lanti, A. Mosca, G. Xiao, J. Xiong, D. Calvanese, A systematic overview of data federation systems, Semantic Web 15 (2024) 107–165. doi:10.3233/SW-223201.

[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] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Dl-lite: Tractable description logics for ontologies, in: "Proc. of the 20th National Conf. on Artificial Intelligence (AAAI)", AAAI Press / The MIT Press, 2005, pp. 602–607. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-094.php.

[6] 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 2022, ACM, 2022, pp. 10–19. doi:10.1145/3579051.3579070.

[7] Z. Gu, D. Calvanese, M. D. Panfilo, D. Lanti, A. Mosca, G. Xiao, Ontology-based data federation - A framework proposal, in: Proceedings of the 31st Symposium of Advanced Database Systems, Galzingano Terme, Italy, July 2nd to 5th, 2023, volume 3478, CEUR-WS.org, 2023, pp. 210–219.

[8] 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.

[9] 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.

[10] 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.

[11] H. Pérez-Urbina, B. Motik, I. Horrocks, Tractable query answering and rewriting under description logic constraints, J. of Applied Logic 8 (2010) 186–209. doi:10.1016/j.jal.2009.09.004.

[12] M. Rodriguez-Muro, R. Kontchakov, M. Zakharyaschev, Ontology-based data access: Ontop of databases, in: Proc. of the 12th Int. Semantic Web Conf. (ISWC), volume 8218 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 558–573. doi:10.1007/978-3-642-41335-3_35.

[13] J. F. Sequeda, M. Arenas, D. P. Miranker, OBDA: query rewriting or materialization? in practice, both!, in: Proc. of the 13th Int. Semantic Web Conf. (ISWC), volume 8796 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 535–551. doi:10.1007/978-3-319-11964-9_34.

[14] D. Bursztyn, F. Goasdoué, I. Manolescu, Reformulation-based query answering in RDF: alternatives and performance, Proc. VLDB Endow. 8 (2015) 1888–1891. doi:10.14778/2824032.2824093.

[15] 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.

[16] 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.

[17] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison Wesley Publ. Co., 1995.

[18] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387. doi:10.1145/362384.362685.

[19] M. Buron, F. Goasdoué, I. Manolescu, M.-L. Mugnier, Obi-Wan: Ontology-based RDF integration of heterogeneous data, Proc. VLDB Endow. 13 (2020) 2933–2936. doi:10.14778/3415478.3415512.

[20] M. N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer, J. Lehmann, Squerall: Virtual ontology-based access to heterogeneous and large data sources, in: Proc. of the 18th Int. Semantic Web Conf. (ISWC), volume 11779, Springer, 2019, pp. 229–245. doi:10.1007/978-3-030-30796-7_15.

[21] Y. Khan, A. Zimmermann, A. Jha, V. Gadepally, M. d'Aquin, R. Sahay, One size does not fit all: Querying web polystores, IEEE Access 7 (2019) 9598–9617. doi:10.1109/ACCESS.2018.2888601.

[22] Y. Khan, A. Zimmermann, A. Jha, D. Rebholz-Schuhmann, R. Sahay, Querying web polystores, in: Proc. of IEEE Int. Conf. on Big Data (IEEE BigData), IEEE Computer Society, 2017, pp. 3190–3195. doi:10.1109/BIGDATA.2017.8258299.

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

[24] 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.

[25] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, E. Botoeva, Efficient handling of SPARQL OPTIONAL for OBDA (extended version), CoRR abs/1806.05918 (2018). URL: http://arxiv.org/abs/1806.05918. arXiv:1806.05918.

[26] J. Ploennigs, K. Semertzidis, F. Lorenzi, N. Mihindukulasooriya, Scaling knowledge graphs for automating AI of digital twins, CoRR abs/2210.14596 (2022). URL: https://doi.org/10.48550/arXiv.2210.14596. doi:10.48550/ARXIV.2210.14596. arXiv:2210.14596.