Considering Vocabulary Mappings in Query Plans for Federations of RDF Data Sources

Sijin Cheng[1], Sebastián Ferrada[1], and Olaf Hartig[1]

Dept. of Computer and Information Science (IDA), Linköping University
`{sijin.cheng, sebastian.ferrada, olaf.hartig}@liu.se`

Abstract Federations of RDF data sources offer great potential for queries that cannot be answered by a single data source. However, querying such federations poses several challenges, one of which is that different but semantically-overlapping vocabularies may be used for the respective RDF data. Since the federation members usually retain their autonomy, this heterogeneity cannot simply be homogenized by modifying the data in the data sources. Therefore, handling this heterogeneity becomes a critical aspect of query planning and execution. We introduce an approach to address this challenge by leveraging vocabulary mappings for the processing of queries over federations with heterogeneous vocabularies. This approach not only translates SPARQL queries but also preserves the correctness of results during query execution. We demonstrate the effectiveness of the approach and measure how the application of vocabulary mappings affects on the performance of federated query processing.

1 Introduction

RDF federations play a crucial role in facilitating integrated access to distributed data, allowing users to query and retrieve information seamlessly from multiple sources. By leveraging federations, users can harness the collective knowledge stored in the various independent federation members, enabling applications such as semantic search, data integration, and knowledge discovery. Despite the use of International Resource Identifiers (IRIs) in RDF to universally identify individuals, concepts, and predicates (e.g., `ex:Bob`, `schema:Person`, `foaf:knows`),[1] the independent datasets may employ their own local vocabulary, using different IRIs to refer to the same things (e.g., `foaf:Person`, `schema:knows`). Hence, queries expressed in a global vocabulary, which encompasses terms from multiple sources, will fail to retrieve meaningful results when executed against federation members with differing local vocabularies. For instance, a query requesting entities of type `schema:Person` cannot retrieve entities of the equivalent type `foaf:Person`.

Vocabulary heterogeneity is further a problem when considering more complex queries that need to compute joins among the data in the federation members, when each federation member has its own unique vocabulary, or when the relationship among the IRIs used in the different datasets is more intricate than a one-to-one equivalence (e.g., subclasses, unions).

Despite the availability of many well-understood and well-performing ontology alignment approaches and corresponding tools, there is very little research on

[1] When writing concrete IRIs using the usual shorthand notation with prefix names such as `ex:`, we use the prefix names declared at `http://prefix.cc/popular/all.sparql`.

using the resulting mappings for integrated querying of multiple RDF datasets. The few related works [10,12,11] describe rather ad hoc methods for query translation, either not providing a thorough formal treatment, not showing clear query planning methods, or not introducing result reconciliation. In contrast, our work in this paper is the first to provide a systematic and formal approach to consider mappings among the vocabularies when processing queries over federations.

First, we define what the expected result of a query in a vocabulary-aware setting is; then, we introduce a new query plan operator to translate solutions from a local to the global vocabulary; and finally, we introduce an algorithm that produces correct, vocabulary-aware query plans. We evaluate our approach in federations with different vocabulary mapping scenarios. Our experiments show that there is no overhead in planning time when considering vocabulary mappings; however, it takes slightly longer to execute the queries than in a baseline scenario with materialized mapped data.

2 Related Work

The problem of achieving semantic interoperability across RDF data sources is a topic of research since many years. Various approaches have been proposed to address this challenge from different angles, including a focus on methods for representing vocabulary mappings [3,14], discovery of correspondences and mappings between different RDF graphs [9,6] and between their ontologies [7].

Our work relies on such ontology/vocabulary mappings to enable users to issue queries using a unified vocabulary over federations of RDF data sources with heterogeneous vocabularies, thereby facilitating seamless retrieval of results from multiple sources. While most work on SPARQL query federation engines has introduced various approaches to process queries over multiple RDF data sources (e.g., [2,4,15,16,17]), these engines are only capable of processing queries that directly use the vocabularies as used by the data sources, assuming that the user knows for each federation member what classes and properties it uses.

A few exceptions exist: Wang et al. introduce an approach to consider instance mappings when executing SPARQL queries over federations of SPARQL endpoints [18]. Joshi et al. describe a system called ALOQUS that considers both instance and vocabulary mappings when executing SPARQL queries over federations of SPARQL endpoints [10]. However, in contrast to our work in this paper, there is no clear definition of what exactly the query result is that the authors would consider correct and complete. Moreover, Joshi et al. do not provide any information about the types of vocabulary mappings considered and how exactly the mappings are used to rewrite (sub)queries.

Makris et al. also explore the use of vocabulary mappings and instance mappings in SPARQL query rewriting for accessing federations of RDF data sources [12,11], Their approach supports a wide range of mapping types, covering classes, object properties, datatype properties, and individuals. Compared to our method, however, their approach is concerned only with defining how to rewrite a given SPARQL query into a set of SPARQL queries which may then be used to access the federation members, without considering the treatment of vocabulary mappings during result reconciliation, without formally defining the expected query results, and also without providing any evaluation to assess the effectiveness and performance of the approach. In contrast, our approach is explicitly embedded into the query planning and query execution processes.

3 Preliminaries

As is usual for papers about RDF and SPARQL [13], we assume four pairwise disjoint, countably infinite sets: $\mathcal{I}$ (all IRIs), $\mathcal{B}$ (all blank nodes), $\mathcal{L}$ (all literals), and $\mathcal{V}$ (all query variables). An *RDF triple* is a tuple $(s, p, o) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$. A set of such triples is called an *RDF graph.* As for SPARQL, the major component of every SPARQL query is its *graph pattern* [13], where the most basic type of such a graph pattern is a *triple pattern*, which is a tuple $(s, p, o) \in (\mathcal{V} \cup \mathcal{I}) \times (\mathcal{V} \cup \mathcal{I}) \times (\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$. Other forms of graph patterns considered in this paper can be constructed recursively by combining two such patterns, P_1 and P_2, using the operator AND or UNION [13]; i.e., $(P_1 \text{ AND } P_2)$ and $(P_1 \text{ UNION } P_2)$.

To formally abstract the concept of a federation, we use the following definition, which slightly adapts the notion of a federation as defined in our earlier work [5]. The main difference of this adaptation is that, for the sake of simplifying the discussion in this paper, we focus only on federations of SPARQL endpoints.

Definition 1. A **federation member** *fm* is a SPARQL endpoint. A **federation** *fed* is a tuple (M, g) where M is a finite and nonempty set of federation members and g is a function that maps every federation member $fm \in M$ to an RDF graph (which is considered to be the graph that *fm* provides access to), such that the graph of every member $fm \in fed$ uses a disjoint set of blank nodes; i.e., $\text{bnodes}\big(g(fm)\big) \cap \text{bnodes}\big(g(fm')\big) = \emptyset$ for every other member $fm' \in M$.

4 Vocabulary-Aware Formalization of Queries

This section defines an evaluation semantics for using SPARQL graph patterns as queries over federations in which the data of the federation members is captured based on RDF vocabularies that may be different from the vocabulary used in the queries. This semantics considers mappings between the global query vocabulary and the vocabularies used locally at the federation members. We begin by introducing formal abstractions of vocabularies and vocabulary mappings.

4.1 Vocabularies and Vocabulary Mappings

For the purposes of the work in this paper, the relevant aspect of the notion of an RDF vocabulary is that it introduces two disjoint sets of IRIs; namely, IRIs that denote properties and that can be used in the predicate position of triples, as well as IRIs of classes, as can be used in the object position of triples that have `rdf:type` as predicate. Hence, we abstract the notion of a vocabulary formally as a pair of such sets of IRIs. That is, a *vocabulary* v is a pair $(\mathbb{C}, \mathbb{R})$, where $\mathbb{C} \subset \mathcal{I}$ is a finite set of IRIs of classes and $\mathbb{R} \subset \mathcal{I}$ is a finite set of IRIs of properties, and we assume that $\mathbb{C}$ and $\mathbb{R}$ are disjoint; i.e., $\mathbb{C} \cap \mathbb{R} = \emptyset$.

Given this notion of a vocabulary, we can now define formally what it means for an RDF graph or a graph pattern to be expressed in terms of a vocabulary.

Definition 2. Let $v = (\mathbb{C}, \mathbb{R})$ be a vocabulary. An RDF graph G **is expressed in terms of** v if, for every triple $t = (s, p, o)$ in G, it holds that i) $p \in \mathbb{R}$ and

ii) if p is the IRI `rdf:type`, then $o \in \mathbb{C}$. Similarly, a SPARQL graph pattern P **is expressed in terms of** v if, for every triple pattern $tp = (s, p, o)$ in P, it holds that i) $p \in \mathbb{R}$ or $p \in \mathcal{V}$ and ii) if p is the IRI `rdf:type`, then $o \in \mathbb{C}$ or $o \in \mathcal{V}$.

Mappings between vocabularies are then defined as follows.

Definition 3. Let $v = (\mathbb{C}, \mathbb{R})$ and $v' = (\mathbb{C}', \mathbb{R}')$ be vocabularies. A **vocabulary mapping** VM from v to v' is a finite set of *mapping rules* where each such mapping rule r is an expression of one of the following five forms.

RuleType1: $c \equiv c'$ is a mapping rule if $c \in \mathbb{C}$ and $c' \in \mathbb{C}'$.
RuleType2: $c \sqsubseteq c'$ is a mapping rule if $c \in \mathbb{C}$ and $c' \in \mathbb{C}'$.
RuleType3: $c_1 \sqcup \ldots \sqcup c_n \equiv c'$ is a mapping rule if $\{c_1, \ldots, c_n\} \subseteq \mathbb{C}$ and $c' \in \mathbb{C}'$.
RuleType4: $p \equiv p'$ is a mapping rule if $p \in \mathbb{R}$ and $p' \in \mathbb{R}'$.
RuleType5: $p \sqsubseteq p'$ is a mapping rule if $p \in \mathbb{R}$ and $p' \in \mathbb{R}'$.

Notice that, by design, the IRIs of the source vocabulary are on the left-hand sides of the mapping rules, whereas the IRIs of the target vocabulary are on the ride-hand sides. Notice also that the syntax of these rules resembles the syntax of terminological axioms of Description Logics, and so does their semantics. To define this semantics we introduce the following function which specifies the result of applying such rules to individual RDF triples.

Definition 4. Let $v = (\mathbb{C}, \mathbb{R})$ and $v' = (\mathbb{C}', \mathbb{R}')$ be vocabularies, let VM be a vocabulary mapping from v to v', and let $r \in VM$ be a mapping rule in VM. The **application** of r to an RDF triple $t = (s, p, o)$, denoted by $\mathsf{apply}(r, t)$, is an RDF triple that is defined as follows, depending on the form of r.

1. If r is either of the form $c \equiv c'$ or of the form $c \sqsubseteq c'$, with $c \in \mathbb{C}$ and $c' \in \mathbb{C}'$, and p is the IRI `rdf:type` and $o = c$, then $\mathsf{apply}(r, t) = (s, \texttt{rdf:type}, c')$.
2. If r is of the form $c_1 \sqcup \ldots \sqcup c_n \equiv c'$, with $\{c_1, \ldots, c_n\} \subseteq \mathbb{C}$ and $c' \in \mathbb{C}'$, and p is the IRI `rdf:type` and $o \in \{c_1, \ldots, c_n\}$, then $\mathsf{apply}(r, t) = (s, \texttt{rdf:type}, c')$.
3. If r is either of the form $p' \equiv p''$ or of the form $p' \sqsubseteq p''$, with $p' \in \mathbb{R}$ and $p'' \in \mathbb{R}'$, and $p = p'$, then $\mathsf{apply}(r, t) = (s, p'', o)$.
4. In all other cases, $\mathsf{apply}(r, t) = t$.

Example 1. If r is the mapping rule `foaf:Person` $\equiv$ `schema:Person` and t is the triple (`ex:Bob`, `rdf:type`, `foaf:Person`), then $\mathsf{apply}\big(r, t\big)$ = (`ex:Bob`, `rdf:type`, `schema:Person`).

For the purpose of defining the aforementioned evaluation semantics, we broaden the application of mapping rules both from individual triples to whole RDF graphs and from individual mapping rules to whole vocabulary mappings:

Definition 5. Let v and v' be vocabularies, and let VM be a vocabulary mapping from v to v'. The **application** of VM to an RDF graph G, denoted by $\mathsf{apply}(VM, G)$, is the RDF graph returned by Algorithm 1 for VM and G.

Example 2. Consider the vocabulary mapping $VM = \{$`foaf:knows` $\equiv$ `schema:knows`, `foaf:Person` $\equiv$ `schema:Person`$\}$. Then, $\mathsf{apply}(VM, G) = G'$ for G and G' as follows.

$G = \{$(`ex:Bob`, `rdf:type`, `foaf:Person`),
(`ex:Bob`, `foaf:name`, `"Bob"`),
(`ex:Bob`, `foaf:knows`, `ex:Eve`)$\}$

$G' = \{$(`ex:Bob`, `rdf:type`, `schema:Person`),
(`ex:Bob`, `foaf:name`, `"Bob"`),
(`ex:Bob`, `schema:knows`, `ex:Eve`)$\}$

Algorithm 1: Applies a vocabulary mapping VM to an RDF graph G.

```
1 $G' \leftarrow \{t \in G \mid \mathsf{apply}(r,t) = t$ for all $r \in VM\}$;
2 while there exists $r \in VM$ and $t \in G$ with $\mathsf{apply}(r,t) \neq t$ and $\mathsf{apply}(r,t) \notin G'$ do
3     add $\mathsf{apply}(r,t)$ to $G'$;
4 end
5 return $G'$
```

Note 1. By Definition 5, every triple of G that is translated to a different triple is not present anymore in the resulting RDF graph. In contrast, triples for which none of the mapping rules in VM has an effect are kept as they are (cf. line 1 of Algorithm 1). For instance, (`ex:Bob`, `foaf:name`, `"Bob"`) in Example 2 is such a triple.

Note 2. In some cases, applying a vocabulary mapping to an RDF graph has no effect at all. For instance, for the empty vocabulary mapping $VM_\emptyset$, it holds that $\mathsf{apply}(VM_\emptyset, G) = G$ for every RDF graph G. Similarly, given a (non-empty) vocabulary mapping VM from a vocabulary $v = (\mathbb{C}, \mathbb{R})$ to $v' = (\mathbb{C}', \mathbb{R}')$, if an RDF graph G does not use vocabulary v (i.e., none of the triples in G has any of the IRIs of $\mathbb{R}$ as its predicate and none of the triples with predicate `rdf:type` has any of the IRIs of $\mathbb{C}$ as its object), then it holds that $\mathsf{apply}(VM, G) = G$.

Given our definitions of vocabularies and vocabulary mappings, we can now introduce a vocabulary-aware view of a federation. To this end, we augment the notion of a federation (cf. Definition 1) with a so-called *vocabulary context* that establishes a global vocabulary in terms of which the federation can be queried.

Definition 6. A **vocabulary context** cxt for a federation $fed = (M, g)$ is a pair (v_{G}, vm) where v_{G} is a vocabulary (considered as the global vocabulary) and vm is a function that maps every federation member $fm \in M$ to a vocabulary mapping from some vocabulary v_{fm} (considered to be used by fm) to v_{G}.

We emphasize that different vocabulary contexts may be used for a federation, which makes it possible to query a federation from the perspective of different global vocabularies (as long as the relevant vocabulary mappings are available in a corresponding vocabulary context). Moreover, notice that the notion of a vocabulary context also captures cases in which some federation members directly use the global vocabulary. In such cases, the function vm of the corresponding vocabulary context simply assigns the empty vocabulary mapping to these federation members. Also, if multiple federation members share a common vocabulary that is not the global one, they can be assigned the same vocabulary mapping for that shared vocabulary.

In the remainder of this paper we assume that the federation members contain only instance data rather than statements about their vocabulary terms. Formally, we capture this assumption as follows: Given a federation $fed = (M, g)$ and a vocabulary context $cxt = (v_{\mathsf{G}}, vm)$ for fed, we assume that, for every triple $t = (s, p, o)$ in the RDF graph $g(fm)$ of every federation member $fm \in M$, it holds that i) $s \notin (\mathbb{C}_{fm} \cup \mathbb{R}_{fm})$, ii) if p is not the IRI `rdf:type`, then $o \notin \mathbb{C}_{fm}$, and iii) $o \notin \mathbb{R}_{fm}$, where $v_{fm} = (\mathbb{C}_{fm}, \mathbb{R}_{fm})$ is the vocabulary used by fm (i.e., the vocabulary that the vocabulary mapping $vm(fm)$ maps to).

4.2 Vocabulary-Aware Evaluation Semantics

At this point, we have all the necessary elements to define the vocabulary-aware evaluation semantics of SPARQL patterns over federations. Informally, the idea is to define the result of a SPARQL pattern to be the same as the result of evaluating the pattern over the (virtual) union of the RDF graphs of the federation members in a global-vocabulary view of the queried federation. Formally, we define this view and the resulting evaluation semantics as follows.

Definition 7. Let $\mathit{fed} = (M, g)$ be a federation and $\mathit{cxt} = (v_{\mathsf{G}}, \mathit{vm})$ be a vocabulary context for fed. The cxt**-based global-vocabulary view of** fed is the federation $\mathit{fed}' = (M', g')$ such that $M' = M$ and, for every federation member $\mathit{fm} \in M$, it holds that $g'(\mathit{fm}') = \mathsf{apply}\big(\mathit{vm}(\mathit{fm}), g(\mathit{fm})\big)$.

Definition 8. Let fed be a federation, $\mathit{cxt} = (v_{\mathsf{G}}, \mathit{vm})$ be a vocabulary context for fed, and $\mathit{fed}' = (M', g')$ be the cxt-based global-vocabulary view of fed. The cxt**-based evaluation** of a SPARQL graph pattern P over fed, denoted by $[\![P]\!]^{\mathit{cxt}}_{\mathit{fed}}$, is a set of solution mappings defined as $[\![P]\!]^{\mathit{cxt}}_{\mathit{fed}} := [\![P]\!]_{G_{\mathsf{union}}}$ where G_{union} is the RDF graph $G_{\mathsf{union}} = \bigcup_{\mathit{fm} \in M'} g'(\mathit{fm})$, and $[\![P]\!]_{G_{\mathsf{union}}}$ is the evaluation of P over G_{union} as defined by Pérez et al. [13].

Example 3. Consider a federation $\mathit{fed} = (M, g)$ with a single federation member fm such that $g(\mathit{fm})$ is the RDF graph G given in Example 2, and let tp be the triple pattern $(?s, \texttt{schema:knows}, ?o)$. Notice that the IRI in this triple pattern does not occur in the graph of fm. Hence, using this triple pattern directly as a query over the federation would result in no solutions mappings. In contrast, consider $\mathit{cxt} = (v_{\mathsf{G}}, \mathit{vm})$ as a vocabulary context for fed such that $\mathit{vm}(\mathit{fm})$ is the vocabulary mapping VM of Example 2. Then, the RDF graph of fm in the cxt-based global-vocabulary view of fed is RDF graph G' in Example 2 and, thus, the query result is $[\![\mathit{tp}]\!]^{\mathit{cxt}}_{\mathit{fed}} = \{\mu\}$ with $\mu = \{?s \rightarrow \texttt{ex:Bob}, ?o \rightarrow \texttt{ex:Eve}\}$.

Example 4. Still considering the federation and the vocabulary context of Example 3, assume now that the triple pattern $\mathit{tp}' = (?x, \texttt{rdf:type}, ?t)$ is given as a query over the federation. The query result in this case is $[\![\mathit{tp}']\!]^{\mathit{cxt}}_{\mathit{fed}} = \{\mu'\}$ with $\mu' = \{?x \rightarrow \texttt{ex:Bob}, ?t \rightarrow \texttt{schema:Person}\}$.

5 Vocabulary-Aware Query Plans

While the query evaluation semantics in the previous section defines the expected query results to be produced by a vocabulary-aware federation engine, we now establish a formal foundation to consider vocabulary mappings when creating query execution plans in such an engine. To this end, we build on FedQPL, which is a formal language to represent logical plans for queries over heterogeneous federations [5]. The basic idea of our approach is as follows: Given a FedQPL expression that represents a logical plan for a query expressed in terms of the global vocabulary, rewrite this expression into a FedQPL expression that represents a vocabulary-aware plan. In such a plan, requests to federation members are expressed in terms of the vocabularies used locally at these federation members, and the results retrieved via such requests are translated back to the global vocabulary. The latter is necessary to correctly join intermediate results from federation members with different local vocabularies and also to eventually present the overall query result in terms of the global vocabulary. To capture such a translation of intermediate results explicitly in logical plans represented by FedQPL we extend FedQPL with a new operator called $\mathsf{l2g}$.

5.1 Vocabulary-Aware Extension of FedQPL

The following definition specifies our extended syntax of FedQPL.

Definition 9. Let $\mathit{fed} = (M, g)$ be a federation and $\mathit{cxt} = (v_{\mathsf{G}}, \mathit{vm})$ be a vocabulary context for *fed*. A **FedQPL expression** for *fed* and *cxt* is an expression φ constructed from the following grammar, in which req, gpAdd, join, union, mj, mu, $\mathsf{l2g}$, (, and) are terminal symbols, *fm* is a federation member in M, P is a graph pattern, VM is a vocabulary mapping in the image of *vm*, and Φ is a nonempty set of FedQPL expressions (constructed recursively from the same grammar).

$$\begin{aligned}\varphi ::= \quad & \mathsf{req}^{P}_{\mathit{fm}} \quad | \quad \mathsf{gpAdd}^{P}_{\mathit{fm}}(\varphi) \quad | \quad \mathsf{join}(\varphi,\varphi) \quad | \quad \mathsf{union}(\varphi,\varphi) \quad | \\ & \mathsf{mj}\ \Phi \quad | \quad \mathsf{mu}\ \Phi \quad | \quad \mathsf{l2g}^{\mathit{VM}}(\varphi)\end{aligned}$$

Before focusing on the new operator ($\mathsf{l2g}$), we briefly describe the other operators of FedQPL: req captures the intention to request the result for a graph pattern from a federation member. gpAdd captures the intention to interact with a federation member to obtain solution mappings for a graph pattern that are compatible with the result produced by the given subplan and, then, join these solution mappings into this result. join and union capture the intention to join, respectively union, the results of two subplans within the federation engine (i.e., without interacting with any federation member); mj and mu are multiway variants of join and union, respectively. For more details, several examples, and the formal semantics of these operators, refer to our earlier work on FedQPL [5].

The new operator is then meant to capture the application of a given vocabulary mapping VM to every solution mapping produced by the subplan. To define the semantics of this operator formally, we first introduce the corresponding notion of applying vocabulary mappings to solution mappings.

Definition 10. The **application** of a vocabulary mapping VM to a solution mapping μ, denoted by $\mathsf{apply}(\mathit{VM}, \mu)$, is a set of solution mappings obtained by performing Algorithm 2 with VM and μ, where $\mathsf{apply}\big(\mathit{VM}, u\big)$ in line 5 is:

$$\begin{aligned}\mathsf{apply}\big(\mathit{VM}, u\big) = \; & \{x' \mid u \equiv x' \text{ is a mapping rule in } \mathit{VM}\} \cup \\ & \{x' \mid u \sqsubseteq x' \text{ is a mapping rule in } \mathit{VM}\} \cup \\ & \{c' \mid c_1 \sqcup \ldots \sqcup c_n \equiv c' \text{ with } u \in \{c_1, \ldots, c_n\} \text{ is in } \mathit{VM}\}.\end{aligned}$$

Example 5. Consider the vocabulary mapping $\mathit{VM} = \big\{\texttt{foaf:knows} \equiv \texttt{schema:knows}, \texttt{foaf:knows} \equiv \texttt{ex:acquaintedWith}, \texttt{ex:Student} \sqsubseteq \texttt{schema:Person}\big\}$. For the solution mapping $\mu = \{?x \rightarrow \texttt{ex:Student}, ?y \rightarrow \texttt{foaf:knows}\}$, we have $\mathsf{apply}(\mathit{VM}, \mu) = \{\mu_1, \mu_2\}$ with

$$\mu_1 = \big\{?x \rightarrow \texttt{schema:Person}, ?y \rightarrow \texttt{schema:knows}\big\} \text{ and}$$
$$\mu_2 = \big\{?x \rightarrow \texttt{schema:Person}, ?y \rightarrow \texttt{ex:acquaintedWith}\big\}.$$

We are now ready to extend the definition of the semantics of FedQPL to cover expressions that contain the new operator. Since the original definition is

Algorithm 2: Applies vocabulary mapping VM to solution mapping μ.

```
1 $\Omega \leftarrow \{\mu_\emptyset\}$, where $\mu_\emptyset$ is the empty solution mapping, i.e., $\mathrm{dom}(\mu_\emptyset) = \emptyset$;
2 forall $?v \in \mathrm{dom}(\mu)$ do
3     $X \leftarrow \emptyset$; // initially empty set of RDF terms, to collect new bindings for $?v$
4     if $\mu(?v)$ is an IRI $u \in \mathcal{I}$ then
5         $X \leftarrow \mathsf{apply}(VM, u)$, where $\mathsf{apply}(VM, u)$ as in Definition 10;
6     if $X$ is empty then
7         $X \leftarrow \{\mu(?v)\}$;
8     $\Omega \leftarrow \{\mu' \cup \{?v \to x\} \mid \mu' \in \Omega \text{ and } x \in X\}$;
9 return $\Omega$
```

based on a recursively-defined evaluation function [5, Definition 6], our extension in this paper consists of adding a new case to this recursive definition.

Definition 11. Let *fed* be a federation, *cxt* be a vocabulary context for *fed*, and φ be a FedQPL expression for *fed* and *cxt*. The **result produced by** φ, denoted by $\mathsf{sols}(\varphi)$, is a set of solution mappings that is defined as follows.

1. If φ is of the form $\mathsf{l2g}^{VM}(\varphi')$, then $\mathsf{sols}(\varphi) := \bigcup_{\mu \in \mathsf{sols}(\varphi')} \mathsf{apply}(VM, \mu)$.
2. If φ is of any other form, then $\mathsf{sols}(\varphi)$ is defined as in [5, Definition 6].

Example 6. Consider federation *fed* of Examples 3–4, which consists of a single federation member *fm* with the following RDF graph G (as given in Example 2):

$G = \{(\texttt{ex:Bob}, \texttt{rdf:type}, \texttt{foaf:Person}), (\texttt{ex:Bob}, \texttt{foaf:name}, \texttt{"Bob"}), (\texttt{ex:Bob}, \texttt{foaf:knows}, \texttt{ex:Eve})\}$.

Moreover, consider the FedQPL expression $\varphi = \mathsf{l2g}^{VM}(\mathsf{req}^{tp'}_{fm})$ where the vocabulary mapping $VM = \{\texttt{foaf:knows} \equiv \texttt{schema:knows}, \texttt{foaf:Person} \equiv \texttt{schema:Person}\}$ is the same as in Example 2 and the triple pattern tp' is $(?x, \texttt{rdf:type}, ?t)$. Notice that the req operator in φ issues this triple pattern to be executed locally at the federation member *fm* and, thus, produces the following result: $\mathsf{sols}(\mathsf{req}^{tp'}_{fm}) = \{\mu\}$ with $\mu = \{?x \to \texttt{ex:Bob}, ?t \to \texttt{foaf:Person}\}$. The $\mathsf{l2g}$ operator then lifts this result to the global vocabulary: $\mathsf{sols}(\varphi) = \{\mu'\}$ with $\mu' = \{?x \to \texttt{ex:Bob}, ?t \to \texttt{schema:Person}\}$.

While the semantics of FedQPL defines the result that a plan represented by a FedQPL expression produces, it also needs to be shown that this result is indeed the expected result for the query for which the plan has been created. If that is the case, we say that the expression is *correct*. Formally, we define this correctness property for our extended version of FedQPL expressions as follows.

Definition 12. Let P be a SPARQL graph pattern, *fed* be a federation, and *cxt* be a vocabulary context for *fed*. A FedQPL expression φ for *fed* and *cxt* **is correct for** P if it holds that $\mathsf{sols}(\varphi) = [\![P]\!]^{cxt}_{fed}$.

Example 7. By comparing Examples 4 and 6, we observe that $\mathsf{sols}(\varphi) = [\![tp']\!]^{cxt}_{fed}$. That is, the result produced by the FedQPL expression φ in Example 6 is the same as the result expected for triple pattern $tp' = (?x, \texttt{rdf:type}, ?t)$ over federation $fed = (\{fm\}, g)$ in vocabulary context *cxt*, where *fed* and $cxt = (v_{\mathsf{G}}, vm)$ with $vm(fm) = VM$ as in Example 6. Therefore, φ is correct for tp' over *fed* in *cxt*.

Algorithm 3: Given a source assignment φ and a vocabulary context $cxt = (v_{\mathsf{G}}, vm)$, both for the same federation *fed*, this algorithm rewrites φ into a vocabulary-aware FedQPL expression for *fed* and *cxt*.

```
1  if $\varphi$ is of the form $\mathsf{req}^{P}_{fm}$ then
2      $VM \leftarrow vm(fm)$;
3      $P' \leftarrow \mathsf{apply}(VM, P)$, where $\mathsf{apply}(VM, P)$ as in Definition 13;
4      if $P' \neq P$ then $P' \leftarrow (P \text{ UNION } P')$;
5      return $\mathsf{l2g}^{VM}(\mathsf{req}^{P'}_{fm})$;

   // If $\varphi$ is not of the form $\mathsf{req}^{P}_{fm}$, then it is either of the form $\mathsf{mj}\,\Phi$
   // or of the form $\mathsf{mu}\,\Phi$ (because it is a source assignment).
6  $\Phi' \leftarrow \emptyset$;  // initially empty set of FedQPL expressions
7  forall $\varphi_i \in \Phi$ do
8      $\varphi'_i \leftarrow$ result of Algorithm 3 for $\varphi_i$ and $cxt$;
9      $\Phi' \leftarrow \Phi' \cup \{\varphi'_i\}$;
10 if $\varphi$ is of the form $\mathsf{mj}\,\Phi$ then return $\mathsf{mj}\,\Phi'$;  else return $\mathsf{mu}\,\Phi'$;
```

Example 8. Consider the triple pattern $tp = (?s, \texttt{schema:knows}, ?o)$. For the same federation *fed* and vocabulary context $cxt = (v_{\mathsf{G}}, vm)$ as in the previous examples (Examples 6 and 7), Example 3 shows that the expected result of *tp* is $[\![tp]\!]^{cxt}_{fed} = \{\mu\}$ with $\mu = \{?s \rightarrow \texttt{ex:Bob}, ?o \rightarrow \texttt{ex:Eve}\}$. It is not difficult to see that the FedQPL expression $\varphi' = \mathsf{l2g}^{VM}(\mathsf{req}^{tp''}_{fm})$ with $tp'' = (?s, \texttt{foaf:knows}, ?o)$ produces the exact same result and, thus, is correct for *tp* over *fed* in *cxt*.

Observe that the triple pattern tp'' in the FedQPL expression φ' of Example 8 is a version of the given triple pattern *tp* translated to the local vocabulary of the queried federation member. In contrast, for the FedQPL expression in Example 7 such a translation of the given triple pattern (tp', in this case) was not necessary. In the following, we introduce an algorithm to create *correct* vocabulary-aware plans represented as FedQPL expressions.

5.2 Creation of Vocabulary-Aware Query Plans

Our algorithm for creating a vocabulary-aware logical plan is designed based on the assumption that the query engine has already created an initial logical plan for the given global query, which is expressed in terms of the global vocabulary. In particular, we assume that this initial logical plan is the output of the source selection & query decomposition step, which is the first major query processing step in a query federation engine [1,17].[2] As shown in our earlier work [5], the output of existing source selection & query decomposition approaches can be captured by a fragment of FedQPL that consists of only three operators: req, mj, and mu. We call the FedQPL expressions in this fragment *source assignments* [5].

Consequently, the main input to our algorithm in this section—see Algorithm 3—is such a source assignment in which the graph pattern of every req operator is expressed in terms of the global vocabulary. Another input is a vocabulary context for the federation considered by the given source assignment.

[2] Taking into account vocabulary mappings also during source selection is an orthogonal problem that we consider out of scope of our work in this paper.

Algorithm 4: Applies vocab. map. VM to triple pattern $tp = (s, pr, o)$.

```
1  $NewPreds \leftarrow \{p \mid r \in VM$ s.t. $r$ is of the form $p \equiv p'$ or $p \sqsubseteq p'$, with $p' = pr\}$;
2  if $NewPreds$ is empty then $NewPreds \leftarrow \{pr\}$;
3  $NewObjs \leftarrow \{c \mid r \in VM$ s.t. $r$ is of the form $c \equiv c'$ or $c \sqsubseteq c'$, with $c' = o\} \cup$
                 $\{c_1, \ldots, c_n \mid r \in VM$ of the form $c_1 \sqcup \cdots \sqcup c_n \equiv c'$, with $c' = o\}$;
4  if $NewObjs$ is empty then $NewObjs \leftarrow \{o\}$;
5  $NewTPs \leftarrow \emptyset$;  // initially empty set of triple patterns
6  foreach $pr' \in NewPreds$ do
7      foreach $o' \in NewObjs$ do
8          $NewTPs \leftarrow NewTPs \cup \{tp'\}$ where $tp' = (s, pr', o')$;
9  $P' \leftarrow$ combine all triple patterns in $NewTPs$ into a UNION graph pattern;
10 return $P'$;
```

Then, the algorithm rewrites the given source assignment recursively while keeping the overall structure of mj and mu operators exactly as given within the source assignment (lines 6–10 in Algorithm 3). Hence, the only thing that the algorithm actually changes are the req operators. Each such operator is replaced by a subplan consisting of a new req operator with an l2g operator on top (lines 1–5). The vocabulary mapping of this l2g operator is the one associated with the federation member of the replaced req operator (line 2), and the graph pattern of the new req operator is obtained by translating the graph pattern of the replaced req operator (line 3) and, then, combining the translated pattern with the original one (line 4). While we shall discuss the reason for the latter step later (cf. Note 3), the translation of graph patterns (line 3) is defined as follows.

Definition 13. The **application** of a vocabulary mapping VM to a graph pattern P, denoted by $\mathsf{apply}(VM, P)$, is a graph pattern determined as follows.

1. If P is a triple pattern, then $\mathsf{apply}(VM, P)$ is the graph pattern that is obtained by performing Algorithm 4 with VM and P as input.
2. If P is a basic graph pattern $B = \{tp_1, ..., tp_n\}$, then $\mathsf{apply}(VM, P)$ is the graph pattern $(tp'_1 \text{ AND } \ldots \text{ AND } tp'_n)$ where $tp'_i = \mathsf{apply}(VM, tp_i)$ for $1 \leq i \leq n$.
3. If P is of the form $(P_1 \text{ AND } \ldots \text{ AND } P_n)$, then $\mathsf{apply}(VM, P)$ is the graph pattern $(P'_1 \text{ AND } \ldots \text{ AND } P'_n)$ where $P'_i = \mathsf{apply}(VM, P_i)$ for $1 \leq i \leq n$.
4. If P is of the form $(P_1 \text{ UNION } \ldots \text{ UNION } P_n)$, then $\mathsf{apply}(VM, P)$ is the graph pattern $(P'_1 \text{ UNION } \ldots \text{ UNION } P'_n)$ where $P'_i = \mathsf{apply}(VM, P_i)$ for $1 \leq i \leq n$.

Example 9. Consider the triple pattern $tp = (?s, \texttt{schema:knows}, ?o)$ and the vocabulary mapping $VM = \{\texttt{foaf:knows} \equiv \texttt{schema:knows}, \texttt{foaf:Person} \equiv \texttt{schema:Person}\}$, as in Example 8. Then, by Definition 13, it holds that $\mathsf{apply}(VM, tp)$ is the triple pattern $(?s, \texttt{foaf:knows}, ?o)$ which, unsurprisingly, is the same as tp'' in Example 8. In contrast, for $tp' = (?x, \texttt{rdf:type}, ?t)$ in Example 7, $\mathsf{apply}(VM, tp') = tp'$.

Example 10. For the vocabulary mapping $VM' = \{\texttt{ex:Student} \sqsubseteq \texttt{schema:Person}, \texttt{ex:Professor} \sqsubseteq \texttt{schema:Person}\}$ and the triple pattern $tp = (?x, \texttt{rdf:type}, \texttt{schema:Person})$, we have $\mathsf{apply}(VM, tp) = \big((?x, \texttt{rdf:type}, \texttt{ex:Student}) \text{ UNION } (?x, \texttt{rdf:type}, \texttt{ex:Professor})\big)$.

While Examples 9–10 focus on translating graph patterns, the following example illustrates our main translation algorithm for a whole source assignment.

Example 11. We continue with the previous example (Example 10) in which the triple pattern *tp* is assumed to be expressed in terms of a global vocabulary. Now, consider a federation *fed* with a federation member *fm* that uses the corresponding local vocabulary; i.e., a possible vocabulary context for *fed* associates *fm* with the vocabulary mapping VM' of Example 10. Assume furthermore that *fm* has been identified to be the only member of *fed* that may have data to produce a nonempty result for *tp*. Hence, we have the single-operator expression req^{tp}_{fm} as the source assignment for executing *tp* over the federation. Algorithm 3 can then be used to rewrite this source assignment into a vocabulary-aware plan, which creates the FedQPL expression $\mathsf{l2g}^{VM'}\big(\mathsf{req}^{P''}_{fm}\big)$ where P'' is the graph pattern $\big((?x, \texttt{rdf:type}, \texttt{schema:Person})\ \text{UNION}\ P'\big)$ that contains $P' = \big((?x, \texttt{rdf:type}, \texttt{ex:Student})\ \text{UNION}\ (?x, \texttt{rdf:type}, \texttt{ex:Professor})\big)$ from Example 10.

Note 3. Notice that the UNION pattern P'' in Example 11 contains the given triple pattern *tp* in addition to the pattern P' that resulted from translating *tp* based on the vocabulary mapping VM'. Combining the translated pattern with the original pattern in this way is the effect of line 4 in Algorithm 3. Adding the original pattern is necessary to guarantee complete query results in cases in which a federation member uses a term of the global vocabulary even if, according to the vocabulary mapping for this federation member, there is a corresponding term in the local vocabulary of the federation member. As a simple example that illustrates such a case, assume that the RDF graph of federation member *fm* of Example 11 is $G = \big\{(\texttt{ex:Bob}, \texttt{rdf:type}, \texttt{schema:Person})\big\}$. Recall that the vocabulary mapping for *fm* is $VM' = \{\texttt{ex:Student} \sqsubseteq \texttt{schema:Person}, \texttt{ex:Professor} \sqsubseteq \texttt{schema:Person}\}$. Hence, *fm* uses the global IRI `schema:Person` even if, according to VM', there are two corresponding IRIs in the local vocabulary of *fm*. Nonetheless, by Definition 8, the expected result for the triple pattern $tp = (?x, \texttt{rdf:type}, \texttt{schema:Person})$ of Example 11 consists of the solution mapping $\mu = \{?x \rightarrow \texttt{ex:Bob}\}$ because $\mathsf{apply}\big(VM', G\big) = G$ (cf. Definition 5). However, a version of Algorithm 3 without line 4 would create the FedQPL expression $\varphi = \mathsf{l2g}^{VM'}\big(\mathsf{req}^{P'}_{fm}\big)$ with P' as in Example 11, and this plan would produce the empty result, $\mathsf{sols}(\varphi) = \emptyset$, because there are no matches for P' in the data of *fm*. In contrast, Algorithm 3 with line 4 creates $\mathsf{l2g}^{VM'}\big(\mathsf{req}^{P''}_{fm}\big)$ with $P'' = \big((?x, \texttt{rdf:type}, \texttt{schema:Person})\ \text{UNION}\ P'\big)$, which produces the expected result consisting of solution mapping μ.

While having line 4 in Algorithm 3 fixes the illustrated incompleteness issue in cases in which federation members unexpectedly use the global vocabulary, a similar issue exists if the global query uses terms of any of the local vocabularies even if there is a corresponding global term.

Example 12. Consider a federation $fed = \big(\{fm\}, g\big)$ with $g(fm) = \big\{(s, p_{\mathsf{L}}, o)\big\}$, and a vocabulary context $cxt = (v_{\mathsf{G}}, vm)$ such that $vm(fm) = \{p_{\mathsf{L}} \equiv p_{\mathsf{G}}\}$. Then, the expected result of evaluating the triple pattern $tp = (?s, p_{\mathsf{L}}, ?o)$ over *fed* is the empty result (i.e., $[\![tp]\!]^{cxt}_{fed} = \emptyset$) because there is no matching triple for

tp in $\mathsf{apply}\big(vm(fm), g(fm)\big) = \big\{(s, p_{\mathsf{G}}, o)\big\}$. However, given the source assignment req^{tp}_{fm}, Algorithm 3 (with or without line 4) translates this source assignment into the FedQPL expression $\mathsf{l2g}^{vm(fm)}\big(\mathsf{req}^{tp}_{fm}\big)$, which incorrectly produces a nonempty result consisting of solution mapping $\mu = \{?s \to s, ?o \to o\}$.

To fix the issue illustrated in the previous example, the only graph patterns that we support as global queries are the ones that do not contain any of the vocabulary terms used in the left-hand side of some mapping rule in the given vocabulary context. We call such patterns *purely global*, defined as follows.

Definition 14. Let $cxt = (v_{\mathsf{G}}, vm)$ be a vocabulary context. A SPARQL graph pattern P is **purely global in** cxt if the following properties hold.

1. If P is a triple pattern (s, p, o), then $p \notin \mathbb{A}$ and $o \notin \mathbb{A}$, where

$$\begin{aligned}\mathbb{A} = &\{x \mid x \equiv x' \text{ is a rule in some vocab. mapping in the image of } vm\} \cup \\ &\{c_1, \dots, c_n \mid c_1 \sqcup \dots \sqcup c_n \equiv c' \text{ is in some mapping in the image of } vm\}.\end{aligned}$$

2. If P is a BGP, then every triple pattern $tp \in P$ is purely global in cxt.
3. If P is of the form either $(P_1 \text{ AND } \dots \text{ AND } P_n)$ or $(P_1 \text{ UNION } \dots \text{ UNION } P_n)$, then every P_i is purely global in cxt, for $1 \le i \le n$.

Example 13. Considering cxt and tp of Example 12, tp is *not* purely global in cxt, and neither is any graph pattern that contains tp as a sub-pattern. In contrast, the triple pattern $(?s, p_{\mathsf{G}}, ?o)$ is purely global in cxt.

Finally, even if we focus only on purely-global graph patterns, the correctness of the vocabulary-aware FedQPL expressions produced by Algorithm 3 depends on the correctness of the source assignments from which they are produced. Since every source assignment that is given to Algorithm 3 is assumed to be expressed in terms of the global vocabulary, the notion of correctness of such source assignments is not the same as the correctness of vocabulary-aware FedQPL expressions as given in Definition 12. Instead, such source assignments are considered as FedQPL expressions for the global-vocabulary view of the queried federation (cf. Definition 7), which means that their req operators are considered to access the global-vocabulary view of the RDF graphs of the federation members, and the correctness of such source assignments is then defined as follows.

Definition 15. Let fed be a federation, $cxt = (v_{\mathsf{G}}, vm)$ be a vocabulary context for fed, and $fed' = (M', g')$ be the cxt-based global-vocabulary view of fed. Moreover, let P be a SPARQL graph pattern. A source assignment for fed' **is correct for** P if it holds that $\mathsf{sols}(\varphi) = [\![P]\!]_{G_{\mathsf{union}}}$, where $G_{\mathsf{union}} = \bigcup_{fm \in M'} g'(fm)$.

Finally, the correctness of Algorithm 3 can be stated as follows. Given a federation fed, a vocabulary context cxt for fed, a SPARQL graph pattern P that is purely global in cxt, and a source assignment φ for the cxt-based global-vocabulary view of fed, if φ is correct for P (as per Definition 15), then the FedQPL expression obtained by performing Algorithm 3 with φ and cxt as input is a FedQPL expression for fed and cxt that is correct for P (as per Definition 12). This correctness follows from Definitions 8, 11, 12, 14, and 15, and the definition of Algorithm 3 (including the corresponding Definition 13 with its Algorithm 4).

6 Evaluation

In this section we evaluate how vocabulary-awareness, as supported by our approach, affects the performance of federated query processing. To this end, we first describe the implementation employed for our study, along with the experiment setup. Thereafter, we present the measurements and discuss our observations. All artifacts required to reproduce our experiments, as well as the measurements obtained from these experiments, are available online.[3]

6.1 Implementation

We implemented the approach in our query federation engine HeFQUIN.[4] While the approach is independent of any particular plan-enumeration algorithm based on which the engine may select a specific query plan to execute, for the evaluation we use the approach in combination with a simple greedy plan-enumeration algorithm. This algorithm takes a FedQPL expression as produced by Algorithm 3 as input and constructs a left-deep execution plan for the multiway join in this expression. To this end, the algorithm starts by estimating the cardinality of the result of each subplan under the join, which is done by sending ASK requests to the corresponding federation members [16]. Thereafter, the algorithm picks a first subplan based on the estimated cardinality and uses it as the starting point for building up the left-deep plan. Subsequently, the algorithm iterates over the remaining subplans that can be joined with the partial left-deep plan that has been built so far, disregarding subplans that would introduce cross products unless no other subplans are available. During each step of this iteration, the algorithm considers the initially-determined cardinality for all available subplans and employs a greedy strategy to pick the subplan with the lowest cardinality among the available options. As the actual join algorithm in the resulting plans, we simply use the symmetric hash join algorithm.

6.2 General Experiments Setup

All experiments described in this paper have been performed on a server machine with two 8-core Intel Xeon E5-2667 v3@3.20GHz CPUs and 256 GB of RAM. The machine runs a 64-bit Debian GNU/Linux 10 server operation system. Federation members used in the experiments are SPARQL endpoints set up using docker images of Virtuoso v7.2.5.

Datasets: The datasets utilized in our evaluation are generated using the dataset generator of the Lehigh University Benchmark (LUBM) [8], which is a popular benchmark in the Semantic Web community for evaluating the performance of storage and reasoning systems for RDF data. These benchmark datasets capture a fictional scenario of universities that consist of departments with both students

[3] `https://github.com/LiUSemWeb/HeFQUIN-VocabMappingsExperiments`

[4] `https://github.com/LiUSemWeb/HeFQUIN`

Table 1. The mapping rules that constitute the mapping from the vocabulary used in the generated datasets (and also in federation Fed1) to the global vocabulary.

Type	Mapping Rule
RuleType1	lubm:Course $\equiv$ global:Course
RuleType1	lubm:GraduateStudent $\equiv$ global:GraduateStudent
RuleType2	lubm:GraduateStudent $\sqsubseteq$ global:Student
RuleType2	lubm:Lecturer $\sqsubseteq$ global:Faculty
RuleType2	lubm:GraduateCourse $\sqsubseteq$ global:Course
RuleType2	lubm:UndergraduateStudent $\sqsubseteq$ global:Student
RuleType3	lubm:UndergraduateStudent $\sqcup$ lubm:GraduateStudent $\equiv$ global:Student
RuleType3	lubm:AssistantProfessor $\sqcup$ lubm:AssociateProfessor $\sqcup$ lubm:FullProfessor $\equiv$ global:Professor
RuleType4	lubm:advisor $\equiv$ global:supervisor
RuleType4	lubm:worksFor $\equiv$ global:worksAt
RuleType4	lubm:teacherOf $\equiv$ global:teaches
RuleType4	lubm:telephone $\equiv$ global:phoneNumber
RuleType4	lubm:emailAddress $\equiv$ global:email
RuleType4	lubm:researchInterest $\equiv$ global:researchTopic
RuleType4	lubm:memberOf $\equiv$ global:memberOf
RuleType4	lubm:subOrganizationOf $\equiv$ global:isPartOf
RuleType5	lubm:publicationAuthor $\sqsubseteq$ global:writtenBy
RuleType5	lubm:doctoralDegreeFrom $\sqsubseteq$ global:degreeFrom
RuleType5	lubm:mastersDegreeFrom $\sqsubseteq$ global:degreeFrom
RuleType5	lubm:undergraduateDegreeFrom $\sqsubseteq$ global:degreeFrom
RuleType5	lubm:headOf $\sqsubseteq$ global:worksAt
RuleType5	lubm:takesCourse $\sqsubseteq$ global:registersCourse

and faculty of different types (e.g., lecturers, assistant professors). These people engage in activities such as teaching or taking courses, may be co-authors of publications, and have degrees from universities. For our evaluation, we generated such data for ten universities and split into ten separate datasets (one per university). One notable aspect of these datasets is that they are interlinked through different types of "degree from" relationships; i.e., students and faculty of a university described in one dataset may have a degree from a university described in another dataset. We leverage this feature in the test queries used for our evaluation (see below). Another relevant aspect of these datasets is that not all classes and properties of the LUBM schema are used explicitly in the generated datasets. For instance, while the LUBM schema contains classes such as `lubm:FullProfessor`, `lubm:AssociateProfessor` and `lubm:AssistantProfessor` as subclasses of `lubm:Professor`, the generated datasets contain only these subclasses. We leverage this aspect to establish a separation between local vocabularies and global vocabulary being used in our evaluation.

Base Vocabulary Mapping: While we consider three different federations for our evaluation (see below), for all of them we aim to use the same global vocabulary. As the basis of this global vocabulary we use the classes and properties that are not used explicitly in the generated datasets, encompassing IRIs such as `global:degreeFrom` and `global:Professor`. Additionally, we create a few supplementary IRIs (e.g., `global:worksAt`, `global:registersCourse`) to allow us to consider a wider range of different types of mapping rules. After establishing our global vocabulary, we manually constructed a mapping from the local vocabulary consisting of

Table 2. Characteristics of the queries used in the evaluation. C, P, and I refer to the types of terms, representing class IRIs, predicate IRIs, and instance IRIs, respectively.

	query plan translation			interm.result translation			joins are on	#triple patterns	types of joins	#solution mappings
Query	$\equiv$	$\sqsubseteq$	$\sqcup$	$\equiv$	$\sqsubseteq$	$\sqcup$				
Q1	P	P	C				I	6	s-s, o-o	356
Q2	P	P	C				I	5	s-s, o-o	382,803
Q3	P	P				C	C,I	6	s-s, o-o	158
Q4	P	C	C		P		P	6	s-s, s-o, o-o, p-p	270
Q5	C,P	C	C				I	5	s-s, s-o, o-o	229,170
Q6	P	C	C	C	C	C	I	5	s-s, s-o, o-o	274,699
Q7	C	C	C	P	C,P	C	C,I	5	s-s, s-o, o-o	233,560

the class and property that are used explicitly in the generated datasets to this global vocabulary. Table 1 lists the mapping rules of this vocabulary mapping.

Next, we describe the three federations used for the evaluation. Each of them consists of ten federation members, created based on the aforementioned datasets.

Fed0: As a baseline, we set up a federation in which all federation members employ the global vocabulary. Thus, no vocabulary translation is needed for querying this federation. To create the datasets of the ten members of Fed0, we simply use an implementation of Algorithm 1 to apply the vocabulary mapping of Table 1 to each of the ten datasets produced by the LUBM dataset generator.

Fed1: For setting up this federation, we use the generated datasets as they are. Therefore, the vocabulary used in these datasets—which is the same in all ten of them—becomes the local vocabulary of the members in this federation. Consequently, when querying Fed1, the vocabulary mapping of Table 1 can be used commonly for all federation members.

Fed2: This federation is structurally the same as Fed1, but with the following simple variation of the local vocabularies. For each federation member, we change the IRIs of the vocabulary terms used in the dataset of that member by appending a member-specific suffix (ranging from 0 to 9) to each such IRI. For instance, the class IRI `lubm:Course` becomes `lubm:Course0` for the first federation member, `lubm:Course1` for the second, etc. As a result, the local vocabularies used by the ten members of Fed2 are different from one another (not structurally but in terms of their IRIs). As a consequence, the vocabulary mapping of Table 1 has to be adapted accordingly for each federation member and, hence, every federation member in Fed2 is associated with a different vocabulary mapping. For instance, the versions of the first mapping rule in Table 1 for the first two members of Fed2 are `lubm:Course0` $\equiv$ `global:Course` and `lubm:Course1` $\equiv$ `global:Course`, respectively.

Queries: After creating the federations, we designed seven benchmark queries that are expressed in terms of the global vocabulary. Thus, they can be used for all three federations. As shown in Table 2, these queries differ regarding the types of vocabulary mapping rules that are relevant to them, both in the context

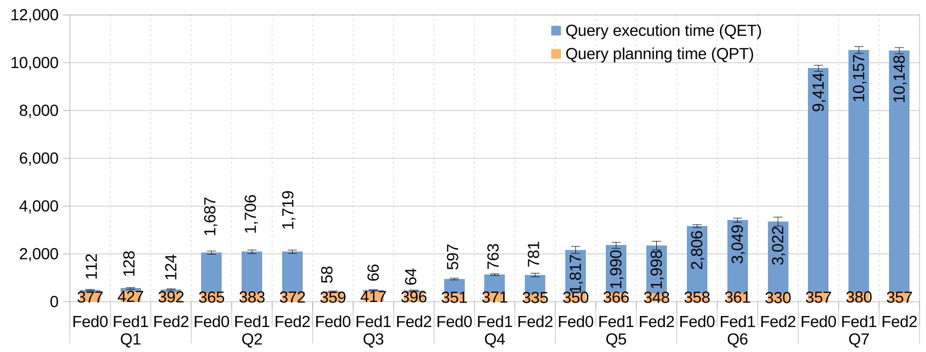

Figure 1. Query planning time (ms) and query execution time (ms) for the test queries over different federations. Fed0: All federation members use the global vocabulary, no mappings needed. Fed1: All federation members use the same vocabulary, different from the global vocabulary. Fed2: Each federation member uses a different vocabulary.

of the mapping-based rewriting of the initial query plans (cf. Section 5.2) and in the context of the translation of intermediate results (as per Definition 10).

Evaluation Metrics We report performance metrics based on the following definitions: i) *Query planning time* (**QPT**) is the amount of time elapsed since the input of a given source assignment until the plan for executing the query has been determined, which includes rewriting the given source assignment (if necessary, see Section 5.2) and selecting a join order. ii) *Query execution time* (**QET**) is the amount of time needed for executing the selected plan until completion.

6.3 Overhead of Considering Vocabulary Mappings

To identify the overhead of considering vocabulary mappings during query processing we compare the performance when executing the test queries over federations Fed1 and Fed2 using our approach versus executing them without vocabulary mappings over Fed0. For each federation, we execute the seven test queries sequentially. We run this process 11 times, with the first run as warm up. Figure 1 illustrates the average QPT and QET of the other 10 runs, with error bars representing the standard deviation of the average sum of QPT and QET.

As a first observation, we notice that the query planning times across all queries for the different federations differ only marginally. This observation suggests that the vocabulary-related query rewriting consumes no significant time.

In contrast, the query execution time increases noticeably when comparing the baseline (Fed0) to the cases in which the query plans are rewritten based on our approach (Fed1 and Fed2). This increase can be attributed to two factors.

The first, and major, factor is that the plans have been extended with l2g operators which perform extra work that is not done by the plans in the baseline case. The amount of this extra work differs for the different queries as the sizes of the intermediate results differ (some of the queries are more selective than others). For instance, there are 1,212 solution mappings to be processed by the l2g operators for Q1, whereas there are 312,248 solution mappings for the l2g operators for Q5; as a consequence, the QET of these two queries increases accordingly (+15 ms for Q1 versus +172 ms for Q5).

Another factor is that the rewritten req operators may retrieve a greater number of solution mappings compared to their baseline counterparts. In particular, such additional solution mappings may be retrieved because of the UNION patterns added by the translation process (see line 4 in Algorithm 3 and line 9 in Algorithm 4). For queries for which this is the case, these additional solution mappings cause even more extra work to be done by the l2g operators in the rewritten plans and, thereby, amplify the first factor. Among our test queries, this is the case for Q2–Q4. Yet, for Q2 and Q3, the number of additional solution mappings retrieved by the rewritten req operators is negligible (less than 5). For Q4, however, the rewritten req operators retrieve a total of 200 additional solution mappings, which contributes to an increased QET of +166 ms (+27.8%).

Overall, the increase in query execution time remains within an acceptable range, with six of the seven queries (all but Q4) experiencing an increase of less than 15% after rewriting the request operators and introducing l2g operators.

Finally, we compare the measurements for Fed1 and Fed2, which are only minimally different. This can be attributed to the fact that the datasets used in both federations are structurally the same. Although each federation member in Fed2 requires a different vocabulary mapping, these mappings are isomorphic to the vocabulary mapping used in Fed1. Consequently, the size of the intermediate results remains unchanged for each query when executed over Fed1 or over Fed2.

7 Concluding Remarks

This paper presents a formal pipeline to translate query plans for queries expressed in a global vocabulary into plans that consider the vocabulary heterogeneity of RDF federations. The translation includes the use of an operator that translates back the obtained local solutions into global ones, to be able to perform joins across federations members. The outcomes of our experimental study indicate that the integration of vocabulary mappings into query processing unsurprisingly introduces overhead, which, however, is within an acceptable range.

As part of our ongoing research, we are currently investigating various strategies to optimize query performance by applying rewriting rules to the logical query plans, with the aim of mitigating and minimizing these extra overheads.

As future work, we consider incorporating tools that can automatically generate mapping rules within our approach. In addition, we envision applying more complex mapping rules, such as those involving union and intersection. With respect to our broader long-term goals, we are interested in handling queries over federations with different types of data sources (i.e., not just RDF).

Acknowledgements. This work was funded by the National Graduate School in Computer Science, Sweden (CUGS), and by Vetenskapsrådet (the Swedish Research Council, project reg. no. 2019-05655).

References

1. Acosta, M., Hartig, O., Sequeda, J.F.: Federated RDF Query Processing. In: Encyclopedia of Big Data Technologies. (2019)
2. Acosta, M., Vidal, M.E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In: Proc. of the 11th International Semantic Web Conference (ISWC) (2011)
3. Bouquet, P., Giunchiglia, F., Van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing Ontologies. In: Proc. of the 2nd International Semantic Web Conference (ISWC) (2003)
4. Charalambidis, A., Troumpoukis, A., Konstantopoulos, S.: SemaGrow: Optimizing Federated SPARQL Queries. In: Proc. of the 11th SEMANTICS Conference (2015)
5. Cheng, S., Hartig, O.: FedQPL: A Language for Logical Query Plans over Heterogeneous Federations of RDF Data Sources. In: Proc. of the 22nd Int. Conf. on Information Integration and Web-based Applications & Services (iiWAS) (2020)
6. Collarana, D., Galkin, M., Traverso-Ribón, I., Vidal, M.E., Lange, C., Auer, S.: MINTE: Semantically Integrating RDF Graphs. In: Proc. of the 7th Int. Conference on Web Intelligence, Mining and Semantics (2017)
7. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. J. Web Semant. **3**(2-3), 158–182 (2005)
9. Isele, R., Bizer, C.: Active Learning of Expressive Linkage Rules Using Genetic Programming. Journal of Web Semantics **23**, 2–15 (2013)
10. Joshi, A.K., Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P., Damova, M.: Alignment-Based Querying of Linked Open Data. In: Proc. of the 11th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE) (2012)
11. Makris, K., Bikakis, N., Gioldasis, N., Christodoulakis, S.: SPARQL-RW: Transparent Query Access over Mapped RDF Data Sources. In: Proc. of the 15th Int. Conf. on Extending Database Technology (EDBT) (2012)
12. Makris, K., Gioldasis, N., Bikakis, N., Christodoulakis, S.: Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources. In: Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE) (2010)
13. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. ACM Trans. Database Syst. **34**(3) (2009)
14. Polleres, A., Scharffe, F., Schindlauer, R.: SPARQL++ for Mapping Between RDF Vocabularies. In: Proc. of the Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE) (2007)
15. Saleem, M., Potocki, A., Soru, T., Hartig, O., Ngomo, A.N.: CostFed: Cost-Based Query Optimization for SPARQL Endpoint Federation. In: Proc. of the 14th Int. Conf. on Semantic Systems (SEMANTICS) (2018)
16. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: Proceedings of the 10th International Semantic Web Conference (ISWC) (2011)
17. Vidal, M., Castillo, S., Acosta, M., Montoya, G., Palma, G.: On the Selection of SPARQL Endpoints to Efficiently Execute Federated SPARQL Queries. Trans. Large-Scale Data- and Knowledge-Centered Systems **25** (2016)
18. Wang, X., Tiropanis, T., Davis, H.C.: Optimising Linked Data Queries in the Presence of Co-reference. In: 11th Ext. Semantic Web Conference (ESWC) (2014)