Ensuring License Compliance in Federated Query Processing

Benjamin Moreau, Patricia Serrano-Alvarado

Ensuring License Compliance in Federated Query Processing

Benjamin Moreau
Nantes University, LS2N, CNRS
OpenDataSoft
Benjamin.Moreau@ls2n.fr
Benjamin.Moreau@opendatasoft.com

Patricia Serrano-Alvarado
Nantes University, LS2N, CNRS
Patricia.Serrano-Alvarado@ls2n.fr

ABSTRACT

When two or more licensed data sources participate in the evaluation of a federated query, the query result must be protected by a license that is compliant with each license of the involved datasets. However, such a license does not always exist, and this leads to a query result that cannot be reused. We propose to deal with this issue during the federated query processing by discarding datasets of conflicting licenses. But, a query with an empty result set can be obtained. To face this problem, we use query relaxation techniques. Our problem statement is, *given a SPARQL query and a federation of licensed datasets, how to guarantee a relevant and non-empty query result whose license is compliant with each license of involved datasets?* In a distributed environment, the challenge is to limit communication costs when the query relaxation process is necessary. In this paper, to detect and prevent license conflicts, we propose FLiQue, a license-aware query processing strategy for federated query engines. Experiments show that FLiQue guarantees license compliance, and if necessary, can find relevant relaxed federated queries with a limited overhead in terms of execution time.

KEYWORDS

Licenses, Federated Query, Privacy, Linked Data, RDF, Query Relaxation

1 INTRODUCTION AND MOTIVATION

A *federated SPARQL query* can retrieve information from several RDF data sources distributed across the Linked Data. When two or more licensed data sources participate in the evaluation of a federated query, the query result must be protected by a license that is compliant with each license of involved datasets.

To facilitate reuse, data owners should systematically associate licenses with resources before sharing or publishing them[5, 19]. Licenses specify precisely the conditions of reuse of data, i.e., what actions are permitted, obliged, and prohibited. The W3C Open Digital Rights Language (ODRL) [8] allows to define machine readable licenses. The Data Licenses Clearance Center (DALICC) [14], proposes a library of well-known standard licenses. We consider that a license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i. If l_j is compliant with l_i, then l_i is compatible with l_j. Unfortunately, it is not always possible to find a license compliant with each license of datasets involved in a federated query[12]. If such a license does not exist, the query result cannot be licensed and, thus, should not be reused nor published. We consider that a query whose result set cannot be licensed should not be executed. Notice that having the rights to query several datasets individually does not mean having the rights to execute a federated query involving these datasets. Consider datasets of LargeRDFBench[16], a benchmark for federated query processing. Figure 1 shows the compatibility graph of Creative Commons licenses[1] that protect LargeRDFBench datasets. License CC BY is compatible with itself, with CC BY-SA, CC BY-NC, and CC BY-NC-SA. Thus, datasets protected by CC BY can be queried along with other datasets protected by these licenses. But the whole set of datasets of Figure 1 cannot be queried together because there is no license compliant with the fourth licenses. For instance, there is no license with which CC BY-SA and CC BY-NC-SA are both compatible. Linked Data catalogs like Datahub[2], show that the problem of license compliance when combining datasets can be frequent. For instance, Datahub shows 382 datasets with the license CC BY-SA and 292 with license CC BY-NC. One solution to the

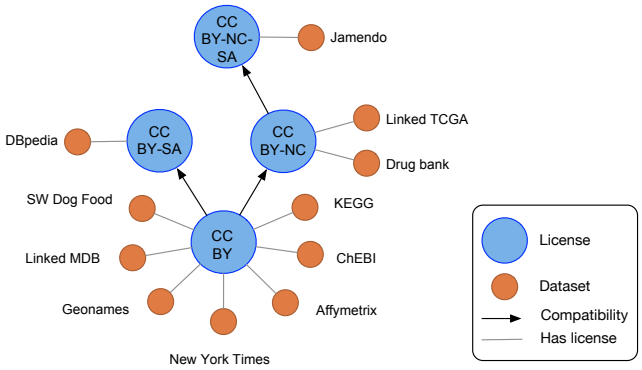

Figure 1: The compatibility graph of licenses for datasets of LargeRDFBench.

incompatibility of licenses is to negotiate with data providers to change a conflicting license, e.g., to ask DBpedia to change their license to CC BY or CC BY-NC. But negotiation takes time and is not always possible. A second solution is to discard datasets that

[1]This compatibility graph conforms to the license compatibility chart shown in https://wiki.creativecommons.org/wiki/Wiki/cc_license_compatibility.
[2]https://old.datahub.io

are protected by conflicting licenses. However, this solution can lead to a query with an empty result set. To face this problem, we use query relaxation techniques. That is, we use *relaxation rules* to relax triple patterns to match triples of other datasets. Consider the query Q of Listing 1 which asks for students enrolled in a course held at the University of Nantes taught by *ex:Jamy*. $D2$ and $D3$ should not be queried together because their licenses. Thus, the result set of the query cannot be licensed. Creating a sub-federation without $D2$ makes the result set licensable with CC BY-NC or with another license compliant with CC BY-NC (e.g., CC BY-NC-SA). The problem is that the query gives no result because there is no more dataset to evaluate $tp4$. The case is similar if $D3$ is discarded because there is no more dataset to evaluate $tp1$ and $tp2$. As there is no sub-federation able to produce a licensable and non-empty result set for Q, we use query relaxation techniques. In such relaxation, for instance, instead of asking for students in $tp1$, the query could ask for persons, or instead of asking for courses taught by *ex:Jamy* in $tp4$, the query could ask for courses taught by anybody.

```
SELECT ?student WHERE {
  ?student rdf:type ex:Student .                  #tp1@{D3} CC BY-NC
  ?student ex:enrolledIn ?course .                #tp2@{D3} CC BY-NC
  ?course  ex:heldAt ex:UniversityOfNantes .      #tp3@{D1} CC BY
  ex:Jamy  ex:teaches ?course .                   #tp4@{D2} CC BY-SA
}
```

Listing 1: A SPARQL query Q annotated with the capable datasets for each triple pattern and dataset licenses.

The number of possible relaxed queries can be huge. To find the most relevant relaxed queries efficiently, we use approaches that compute relaxed queries from the most to the least *similar* to the original query [2, 6, 7]. But the most similar queries may produce no results. In a distributed environment, verifying each relaxed query is not feasible. So the challenge is to find the most similar relaxed queries that return a non-empty result while limiting communication costs. Our research question is, *given a SPARQL query and a federation of licensed datasets, how to guarantee a relevant and non-empty query result whose license is compliant with each license of involved datasets?* The challenge is to limit the communication cost when the relaxation process is necessary. We propose FLiQue[3], a Federated License-aware Query processing strategy. FLiQue is designed to detect and prevent license conflicts and gives informed feedback with licenses able to protect a result set of a federated query. If necessary, it applies distributed query relaxation to propose a set of most similar relaxed queries whose result set can be licensed. Our contributions are:

- a license-aware query processing strategy,
- an implementation of a license-aware federated query engine, and
- an experimental evaluation of our approach.

In the next, Section 2 discuses related works, Section 3 introduces FLiQue, Section 4 shows experimental results, and Section 5 concludes.

2 RELATED WORK

To our knowledge, there is no federated query engine that ensures license compliance with all licenses involved in query execution. Many works focus on access control over linked data using policy-based [1, 9, 10, 15], view-based [3], or query-rewriting [13] approaches. In these works, datasets are protected by *access control rules* that prevent non-authorized users from querying data of each dataset. These approaches do not resolve our problem statement because having the right to query datasets individually does not mean that it should be possible to execute a federated query involving these datasets.

Compatibility graph of licenses. To know if a result set can be licensed, we need to know the license(s) with whom all licenses of datasets involved in a federated query are compatible. A compatibility graph of licenses contains a set of licenses partially ordered by compatibility. It can be defined by hand using, for instance, the license compatibility chart of Creative Commons. But licenses used in the Linked Data are not limited to Creative Commons licenses. Works like [20] address the problem of license compatibility and license combination. If licenses are compatible, a new license compliant with combined ones is generated. This approach allows defining the compatibility graph of licenses progressively. However, it does not allow us to know all the compliant licenses that can be used to protect a result set. CaLi [11, 12], is a lattice-based model for license orderings. It automatically positions a license over a set of licenses in terms of compatibility and compliance. It uses restrictiveness relations and constraints among licenses to define compatibility. CaLi defines all the licenses that can be expressed with a set of actions. For instance, the CaLi ordering for the set of 7 actions used by Creative Commons has 972 licenses. CaLi can provide all the licenses than can protect a result set ordered by restrictiveness. It can also identify which licenses are in conflict. In this work, we use CaLi to verify license compliance. When the result set of a federated query cannot be licensed, we propose to define sub-federations that avoid license conflicts. If there is no sub-federation able to produce a licensable and non-empty result set for the user query, we propose alternative queries through query relaxation.

Query relaxation. Query relaxation techniques are used to provide an alternative for queries producing no result. [7] proposes query relaxation using RDFS entailment and RDFS ontologies. The idea consists of relaxation rules that use information from the ontology; these include relaxing a class to its super-class, relaxing a property to its super-property, etc. Other relaxations include dropping triple patterns, replacing constants with variables, and suppressing join dependencies. All possible relaxed queries are organized in a lattice called *relaxation graph*. The size of the relaxation graph grows combinatorially with the number of relaxation rules, the richness of the ontology, and the relaxation possibilities of each triple pattern in the original query. [2, 6] focus on obtaining a certain number of alternative results (top-k) by relaxing a query that produces no results. Their challenge is to execute as less as possible relaxed queries to obtain the top k results. Relaxed queries are executed in a similarity-based ranking order to avoid executing all relaxed queries in the relaxation graph. The *information content* is used to measure the *similarity* between a relaxed query and the original query. That is, statistical information about the concerned dataset, like the number of entities per class and the number of triples per property. But, the number of failing relaxed queries

[3] In French, FLiQue is a homophone of *flic*, which means *cop*.

executed before obtaining the top-k results can be considerable. Thus, it is necessary to identify unnecessary relaxations that do not generate new answers. Relaxed queries containing unnecessary relaxation should not be executed. [6] proposes OBFS (Optimized Best First Search algorithm) to identify unnecessary relaxations in a similarity-based relaxation graph. It is based on the selectivity of relaxations using the number of entities per class or the number of triples per property. If the selectivity is the same before and after the relaxation, the relaxation is considered unnecessary. That is, if the number of entities of a class is equal to the number of entities of its super-class, then the class relaxation does not generate new answers. The same idea is used for property relaxation. [2] proposes OMBS (Optimized Minimal-failing-sub-queries-Based Search algorithm) as an improvement to OBFS. The contribution of OMBS is to identify the minimal sets of triple patterns in failing queries that fail to return answers. These failing sets of triple patterns are called Minimal Failing Sub-queries (MFS). MFS existing in a query must be relaxed, otherwise, the query fails in producing results. Relaxed queries where the MFS are not relaxed are considered unnecessary. OMBS defines optimal similarity-based relaxation graphs where relaxed queries producing no results (based on MFS), or not new results (based on selectivity) are not executed. We use OMBS to find relaxed queries with non-empty result sets. To limit the communication overhead, during the *distributed query relaxation* process, we use *data summaries*.

Data summaries. Some federated query engines, use statistics to reduce the number of requests sent to data sources during query processing, in particular in the source selection and query optimization steps. For instance, SPLENDID[4], uses VOID descriptions of datasets to speed-up query processing. VOID descriptions contain basic statistical information about datasets, such as the number of entities per class and the number of triples per property. HIBISCuS[17], a join-aware source selection algorithm, discards dataset that are relevant for a triple pattern, but that do not contribute to a query result. It proposes data summaries, called *dataset capabilities*, containing all the distinct properties with all the URI authorities of their subjects and objects. CostFed[18], an index-assisted federated engine for SPARQL endpoints, extends the join-aware source selection of HIBISCuS by considering URI prefixes instead of URI authorities. The dataset capabilities calculated by CostFed are more precise, its source selection chooses, in general, more pertinently the data sources for each query. We use the licenses of the data sources identified by a source selection process to know if the result set of a federated query (or a relaxed federated query) would be licensable. We use the join-aware source selection of CostFed.

3 A FEDERATED LICENSE-AWARE QUERY PROCESSING STRATEGY

We propose FLiQue, a federated license-aware query processing strategy to detect and prevent license conflicts. Our approach gives informed feedback with licenses that can protect a result set of a federated query. When the result set of a federated query cannot be licensed, we define sub-federations that avoid license conflicts. If there is no sub-federation able to produce a licensable and non-empty result set, we propose alternative relaxed federated queries.

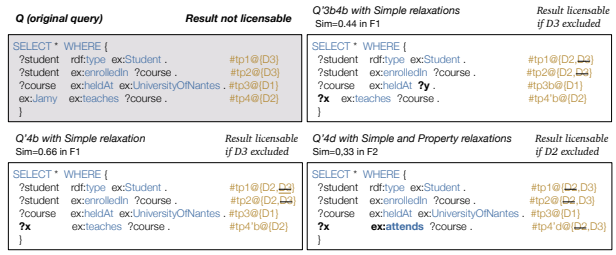

Figure 2: Example of SPARQL query Q and some relaxed queries Q'.

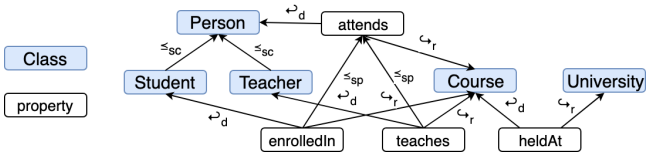

Figure 3: Ontology representing courses in a university. $\preceq_{sc}$ is rdfs:subClassOf, $\preceq_{sp}$ is rdfs:subPropertyOf, $\hookleftarrow_d$ is rdfs:domain, and $\hookrightarrow_r$ is rdfs:range.

FLiQue is located between the query parsing and the query optimization functions of a federated query engine. A join-aware source selection [18], selects the *capable datasets* for each triple pattern of a query. Using a compatibility graph of licenses [12], we search for licenses compliant with each license of the chosen capable datasets. Then, the query is executed and the result set returned with the licenses that can protect it. If no compliant license exists, we identify the license conflicts and define sub-federations that avoid these conflicts. If one sub-federation can produce a licensable and non-empty result, the query is executed. Otherwise, based on the OMBS approach [2], we propose to the query issuer a set of relaxed queries whose result sets are licensable and non-empty. Several sub-federations may produce a licensable and non-empty result. In that case, we choose the sub-federation that produces a result set licensable by the least restrictive license.[4] Consider the query Q of Listing 1, and the federation containing datasets $D1$, $D2$, and $D3$ shown in Tables 1-3. As there is no license compliant with the licenses of $D2$ and $D3$, the result set of Q cannot be licensed. Thus, our strategy defines the sub-federations $F1$={$D1$, $D2$} and $F2$={$D1$, $D3$} that avoid license conflicts. The source selection for Q over $F1$ and $F2$ fails to obtain a data source for each triple pattern. This launches a process of federated query relaxation for each sub-federation. To avoid verifying that the result set of an important number of relaxed queries is not empty, our strategy defines, by sub-federation, an optimal similarity-based relaxation graph. When we find one licensable and non-empty relaxed query, we stop the relaxation process. OMBS guarantees that a candidate query is the most similar to Q for a sub-federation. Figure 2 shows Q and three relaxed queries. Figure 3 shows the ontology used in our example. As we explain next, $Q'4b$ and $Q'4d$ are the most similar licensable,

[4]Other options could be defined, for example, based on the cardinality estimations of result sets or based on the number of involved data sources.

Subject	Predicate	Object
ex:UniversityOfNantes	rdf:type	ex:University
ex:SemanticWeb	rdf:type	ex:Course
ex:SemanticWeb	ex:heldAt	ex:UniversityOfNantes
ex:Databases	rdf:type	ex:Course
ex:Databases	ex:heldAt	ex:UniversityOfNantes

Table 1: Dataset D1 containing courses. D1 has licence CC BY.

Subject	Predicate	Object
ex:Jamy	rdf:type	ex:Teacher
ex:Jamy	rdf:type	ex:Person
ex:Jamy	ex:attends	ex:SemanticWeb
ex:Jamy	ex:teaches	ex:SemanticWeb
ex:LaVoix	rdf:type	ex:Teacher
ex:LaVoix	rdf:type	ex:Person
ex:LaVoix	ex:attends	ex:Databases
ex:LaVoix	ex:teaches	ex:Databases
my:Tarzan	rdf:type	ex:Student
my:Tarzan	rdf:type	ex:Person
my:Tarzan	ex:attends	ex:Databases
my:Tarzan	ex:enrolledIn	ex:Databases

Table 2: Dataset D2 containing teachers and students. D2 has licence CC BY-SA.

Subject	Predicate	Object
ex:Jeanne	rdf:type	ex:Student
ex:Jeanne	rdf:type	ex:Person
ex:Jeanne	ex:attends	ex:SemanticWeb
ex:Jeanne	ex:enrolledIn	ex:SemanticWeb

Table 3: Dataset D3 containing students. D3 has licence CC BY-NC.

and non-empty relaxed query for $F1$ and $F2$ respectively. In the next, Section 3.1 shows the relaxation techniques we use. Section 3.2 presents the information content measures that allow us to rank relaxed queries. Section 3.3 shows the data summaries that allow limiting communication costs. Finally, Section 3.4 explains how we define the similarity-based relaxation graph.

3.1 Query relaxation techniques

In this work, we use query relaxation using RDFS entailment and RDFS ontologies. We consider that ontologies of datasets are accessible and that SPARQL endpoints expose saturated RDF data (or support on-the-fly entailment) according to the RDFS entailment rules rdfs7 and rdfs9. We use the relaxations of triple patterns and queries as proposed in [6].

Triple Pattern Relaxation. Given two triple patterns tp and tp', tp' is a relaxed triple pattern obtained from tp, denoted $tp \prec tp'$, by applying one or more triple pattern relaxations. We use the three following triple pattern relaxations:

- *Simple relaxation* replaces a constant of a triple pattern by a variable.

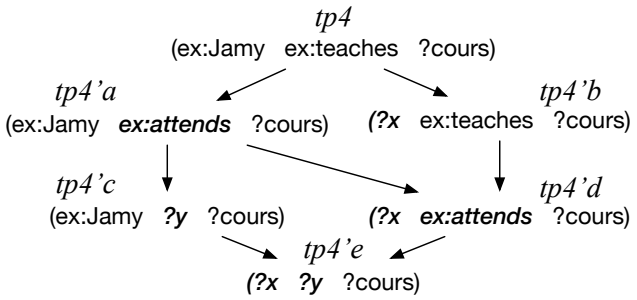

Figure 4: Relaxation lattice of triple pattern $tp4$ of query Q.

- *Type relaxation* replaces a class C of a triple pattern with its super-class C'. It is based on the rdfs9 rule (rdfs:subClassOf).
- *Property relaxation* replaces a property P of a triple pattern with its super-property P'. It is based on the rdfs7 rule (rdfs:subPropertyOf).

The set of all possible relaxed triple patterns of tp can be represented as a lattice called a *relaxation lattice of a triple pattern*. Figure 4 shows this lattice for triple pattern $tp4$ of Q. $tp4'b$, $tp4'c$ and $tp4'e$ show simple relaxations. $tp4'a$ shows a property relaxation. This lattice has three levels of relaxation.

Query Relaxation. Given two queries Q and Q', Q' is a relaxed query obtained from Q, denoted $Q \prec Q'$, by applying one or more triple pattern relaxations to triple patterns of Q. $\prec$ is a partial order over the set of all possible relaxed queries of Q. This order can be represented as a lattice, called a *relaxation lattice of a query* (or relaxation graph). Figure 2 shows the query Q and three relaxed queries of its relaxation graph where, $Q \prec Q'4b \prec Q'4d$ and $Q \prec Q'4b \prec Q'3b4b$.

3.2 Information content measures

Analyzing all relaxed queries is time-consuming and unnecessary. We use *information content measures* to compute the similarity of relaxed queries to the original query. To avoid the analysis of an important number relaxed queries, our approach generates and executes relaxed queries from the most to the least similar. This execution allows to verify that the result set is not empty. It is stopped when the first result is returned. We use the *similarity mesures* proposed in [6], and explained in the following.

Similarity between terms. FLiQue uses three similarity measures for terms in a triple pattern. They correspond to the three relaxations described in Section 3.1.

- *Similarity between classes.* is $Sim(C, C') = \frac{IC(C')}{IC(C)}$ where $IC(C)$ is the information content of C: $-logPr(C)$, where $Pr(C) = \frac{|Instances(C)|}{|Instances|}$ is the probability of finding an instance of class C in the RDF dataset.
- *Similarity between properties.* is $Sim(P, P') = \frac{IC(P')}{IC(P)}$ where $IC(P)$ is the information content of P: $-logPr(P)$, where $Pr(P) = \frac{|Triples(P)|}{|Triples|}$ is the probability of finding a property of P in triples of the RDF dataset.

- *Similarity between constants and variables* is $Sim(T_{const}, T_{var}) = 0$.

Similarity between triple patterns. Given two triple patterns tp and tp', such that $tp \prec tp'$, the similarity of the triple pattern tp' to the original triple pattern tp, denoted $Sim(tp, tp')$, is the sum of the similarities between the terms of the triple patterns:

$$Sim(tp, tp') = \frac{1}{3}.Sim(s, s') + \frac{1}{3}.Sim(p, p') + \frac{1}{3}.Sim(o, o')$$

where s, p, o, s', p' and o' are respectively the subject, predicate and object of the triple pattern tp and the relaxed triple pattern tp'. If tp' and tp'' are two relaxations obtained from tp and $tp' \prec tp''$ then $Sim(tp, tp') \geq Sim(tp, tp'')$.

Similarity between queries. Given two queries Q and Q', such that $Q \prec Q'$, the similarity of the original query Q' to the original query Q, denoted $Sim(Q, Q')$, is the product of the similarity between triple patterns of the query:

$$Sim(Q, Q') = \prod_{i=1}^{n} w_i.Sim(tp_i, tp'_i)$$

Where tp_i is a triple pattern of Q, tp'_i a triple pattern of Q' and $w_i \in [0, 1]$ is the weight of triple patterns tp_i. Weight can be specified by the user to take into account the importance of a triple pattern tp_i in query Q. Thus $Sim(Q, Q') \in [0, 1]$ is a function that defines a total order among relaxed queries. This similarity function is monotone, i.e., given two relaxed queries $Q'(tp'_1, ..., tp'_n)$ and $Q''(tp''_1, ..., tp''_n)$ of the user query Q, if $Q' \prec Q''$ then $Sim(Q, Q')$ $\geq Sim(Q, Q'')$. Considering the query Q and datasets D1 and D2, $Sim(Q, Q'4b) = 0.66$ is greater than $Sim(Q, Q'3b4b) = 0.44$. This verifies the ordering of these relaxed queries, $Q'4b \prec Q'3b4b$, where $Q'4b$ is analyzed first.

3.3 Data summaries

Using dataset statistics and dataset capabilities as in [18], allow us to limit communication cost in the similarity calculation and the source selection process.

Dataset statistics. contain VOID descriptions, such as the number of entities per class and the number of triples per property. Having dataset statistics is twofold; they allow computing similarities, and they help in the source selection process. Tables 4 and 5 show respectively statistics about properties and classes for F1 and F2. In Table 4, the property *ex:teaches* has no triples in F2. So there is no data source for $tp4$. That allows us to identify $Q'4b$ as a failing query in F2.

Dataset capabilities. Capabilities contain the properties of a dataset with the common prefixes of their subjects and objects. The *rdf:type* property, is treated differently. The prefixes of its objects are replaced by all the classes used in the dataset. Dataset capabilities are used in the source selection process. The goal is to discard datasets that individually return results for a triple pattern, but that fail to perform joins with other triple patterns of the query. For multiple triple patterns of a query sharing a variable, the dataset capabilities allow identifying data sources that do not share the same URIs prefixes and thus whose joins yield empty results. This information

	Number of triples	
Property	**F1 = {D1, D2}**	**F2 = {D1, D3}**
ex:enrolledIn	1	1
ex:teaches	2	0
ex:heldAt	2	2
ex:attends	3	1
rdf:type	9	5
Total	17	9

Table 4: Statistics of properties in federations F1 and F2.

	Number of entities	
Class	**F1 = {D1, D2}**	**F2 = {D1, D3}**
ex:University	1	1
ex:Student	1	1
ex:Teacher	2	0
ex:Course	2	2
ex:Person	3	1
Total	6	4

Table 5: Statistics of classes in federations F1 and F2.

	F1 = {D1, D2}		F2 = {D1, D3}	
Property	**subjPrefixes**	**objPrefixes**	**subjPrefixes**	**objPrefixes**
rdf:type	ex: my:Tarzan	ex:Person ex:Student ex:Teacher	ex:	ex:University ex:Course ex:Student ex:Person
ex:heldAt	ex:	ex:UniversityOfNantes	ex:	ex:UniversityOfNantes
ex:attends	ex: my:Tarzan	ex:	ex:Jeanne	ex:SemanticWeb
ex:teaches	ex:	ex:		
ex:enrolledIn	my:Tarzan	ex:Database	ex:Jeanne	ex:SemanticWeb

Table 6: Capabilities of federations F1 and F2.

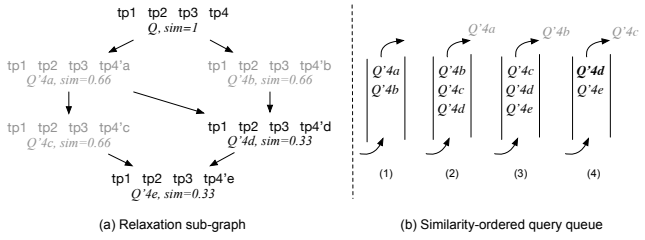

Figure 5: Relaxation sub-graph of Q over $F2$ with relaxations of $tp4$.

allows performing an optimal source selection by limiting the communication with the data sources. Table 6 shows the capabilities of F1 and F2. Consider $tp4'a$ of $Q'4a$ that asks for *ex:Jamy ex:attends ?cours*. Table 4 shows one triple for *ex:attends* but capabilities of this property in F2 show only one subject prefix that is *ex:Jeanne*, not *ex:Jamy*. Consider the join $tp3$. $tp4'c$ of $Q'4c$: *{?course ex:heldAt ex:UniversityOfNantes . ex:Jamy ?y ?cours}*, Table 4 shows two triples for *ex:heldAt*. Capabilities of *ex:heldAt* do not discard this join. But, analyzing the subject and object capabilities of whatever property (the property of $tp4'c$ is a variable), we notice that when there exists *ex:SemanticWeb* in the object, the subject contains *ex:Jeanne*, not *ex:Jamy*, so the join dependency on *?cours* cannot be satisfied. Thus, thanks to the dataset capabilities of F2, we identify $Q'4a$ and $Q'4c$ as failing queries.

3.4 Algorithm for the similarity-based relaxation graph

When the distributed query relaxation is necessary, we define an optimal similarity-based relaxation graph by sub-federation. The goal is to avoid verifying that the result set of an important number of relaxed queries is not empty. Relaxed queries are generated and executed from the most to the least similar. When we find one licensable and non-empty relaxed query that we call *candidate query*, we stop the relaxation process. In the following, we explain how FLiQue finds the candidate query for the sub-federation F2. First, the algorithm computes the MFS of the original query, MFS(Q)={ex:Jamy ex:teaches ?course}. It contains only $tp4$ because $F2$ does not contain a data source to evaluate $tp4$. Using the MFS, the algorithm considers only relaxed queries that contain a relaxation of $tp4$. The relaxation algorithm uses a query queue ordered by similarity. This query queue gives the analysis order of relaxed queries. Figure 5 shows (a) a relaxation sub-graph where $tp4$ is relaxed, and (b) the analysis process of relaxed queries with the query queue (failing relaxed queries are in gray). Relaxed queries of the first level, $Q'4a$ and $Q'4b$, are generated and inserted in the queue (1). The most similar relaxed query $Q'4a$ is analyzed. It is identified as a failing query. It is not executed, but it is relaxed, so $Q'4c$, and $Q'4d$ are generated and inserted in the query queue (2). Then, the first relaxed query in the queue, now $Q'4b$, is analyzed. It is also identified as a failing query so it is relaxed in $Q'4d$, which is already in the queue (3). Then, the first relaxed query in the queue, now $Q'4c$, is analyzed and identified as a failing query, so it is relaxed into $Q'4e$, which is inserted in the queue (4). Then, the first relaxed query in the queue, now $Q'4d$, is analyzed. It is executed returning a non-empty result set. Thus, $Q'4d$ (in bold) is the candidate query of query Q for the federation $F2$, and the relaxation process stops. The MFS and the failing relaxed queries of this example are identified thanks to data summaries without making requests to data sources (cf. Section 3.3). In this example, we found a candidate query only with the relaxation of $tp4$. But the relaxation may continue until all triple patterns are composed of variables. A threshold of similarity can be used to avoid such a case. Figure 2 shows the candidate query $Q'4d$, whose similarity with Q is 0.33. This query asks for students attending a course held at the University of Nantes. The candidate query for federation $F1$ is $Q'4b$, whose similarity with Q is 0.66. Figure 2 shows $Q'4b$, this query asks for students enrolled in a course held at the University of Nantes and taught by someone. CC BY-SA can protect relaxed queries for $F1$. Relaxed queries for $F2$ can be protected by CC BY-NC but also by CC BY-NC-SA because both licenses are compliant with licenses of $D1$ and $D3$. Table 7 shows the feedback returned to the query issuer so that she can choose which query to execute.

Sub-federation	Query	Similarity	Compliant licenses
F1={D1, D2}	Q'4b	0.66	CC BY-SA
F2={D1, D3}	Q'4d	0.33	CC BY-NC, CC BY-NC-SA

Table 7: Feedback with candidate queries for the user query Q.

Conflicting sources	Conflicting licenses	Queries
DBP, DB	CC BY-SA, CC BY-NC	S1, **S10, C9**
DBP, TCGA	CC BY-SA, CC BY-NC	**L7**
DBP, JA	CC BY-SA, CC BY-NC-SA	**L6**
DBP, DB, TCGA	CC BY-SA, CC BY-NC	**C10**
DBP, DB, TCGA, JA	CC BY-SA, CC BY-NC, CC BY-NC-SA	S6, **S8**, S9, C3, **C5, C8**, L1, L3, **L5, L8**

Table 8: The 16 queries of LargeRDFBench whose result set cannot be licensed. DBP (DBpedia), DB (Drug bank), TCGA (Linked TCGA), JA (Jamendo).

4 EXPERIMENTAL EVALUATION

The goal of our experimental evaluation is to measure the overhead produced by the implementation of our proposal. In particular, (a) when the result set of the original query is licensable, and (b) when the original query is relaxed.

4.1 Setup and implementation

FLiQue is implemented over CostFed, which relies on a join-aware triple-wise source selection. Recent studies show that the source selection of CostFed least overestimates the set of capable data sources, with a small number of ASK requests [16, 18]. These performances make CostFed a good choice for our license-aware query processing strategy. The join ordering of CostFed is based on left-deep join trees. It implements bind and symmetric hash joins. Our test environment uses LargeRDFBench[16]. This benchmark contains 32 queries that are executed over a federation of 11 data sources. We identified the license of each dataset (cf. Figure 1). We use a Creative Commons CaLi ordering [12] to verify compatibility and compliance among licenses. Our experiment runs on a single machine with a 160xIntel(R) Xeon(R) CPU E7-8870 v4 2.10GHz 1,5 Tb RAM. Each dataset of LargeRDFBench is saturated and made available using a single-threaded Virtuoso endpoint in a docker container with 4 Gb RAM. Between each query execution, caches are reset.

4.2 Performance of FLiQue vs CostFed

To measure the overhead produced by FLiQue, we compare two different federated query engines: CostFed and CostFed+FLiQue (that we call FLiQue to simplify). They correspond to the original implementation of CostFed[5] and our extension of CostFed that includes FLiQue[6]. CostFed executes a query without considering licenses while FLiQue ensures license compliance of the result set. We executed all queries 5 times with each federated query engine. We measured the time in milliseconds to return the first result of each query. Using the capable data sources by query, and the compatibility graph of licenses, we identified 16 queries whose result set cannot be licensed. Table 8 shows these queries, their conflicting capable data sources and conflicting licenses. 10 need to be relaxed, they are shown in bold. We recall that the DBpedia license (CC BY-SA) is not compliant with the licenses of Jamendo (CC BY-NC-SA), Linked TCGA and Drug bank (CC BY-NC). The average time to check license conflicts is 296 milliseconds what is negligible.

Evaluation of queries that do not need relaxation. Figure 6 presents the execution of 22 queries of LargeRDFBench. For 16 queries,

[5]https://github.com/dice-group/CostFed
[6]Our code repository is hidden for anonymity reasons.

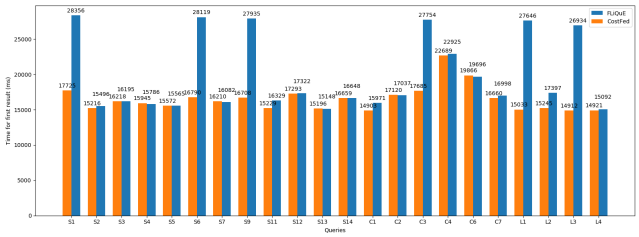

Figure 6: Average time to get the first result of the 22 queries of LargeRDFBench that can produce a licensable result set without relaxation.

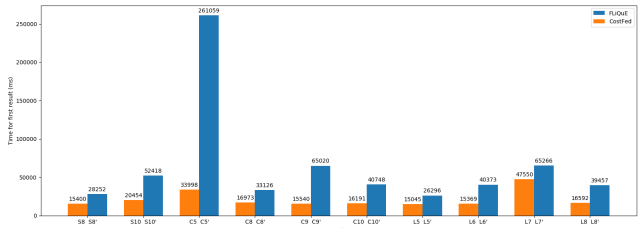

Figure 7: Average time to get the first result of the 10 queries of LargeRDFBench that need relaxation to produce a licensable result set.

{S2, S3, S4, S5, S7, S11, S12, S13, S14, C1, C2, C4, C6, C7, L2, L4}, FLiQue finds a license that can protect the result set when the query is executed on the complete federation. For these queries, the overhead of FLiQue is negligible and corresponds to the time to check license conflicts among the capable datasets. This overhead depends on the number of distinct licenses that protect the capable datasets. For 6 queries, {S1, S6, S9, C3, L1, L3}, FLiQue does not find a license that can protect the result set when the query is executed over the complete federation. However, it finds a sub-federation such that the original query returns a non-empty result set that is licensable. In this case, the overhead of FLiQue corresponds to the time to check license conflicts, to compute sub-federations, and to execute the original query on these sub-federations until the first result is returned. This overhead depends on the number of tested sub-federations. The number of sub-federations depends on the number of distinct conflicting licenses by query. In our test environment, this number is always 2. For instance, conflicting licenses CC BY-SA, CC BY-NC, and CC BY-NC-SA can be separated into two non-conflicting sets {CC BY-SA} and {CC BY-NC, and CC BY-NC-SA}. These sub-federations are ordered by the number of datasets in the federation. In the benchmark, the average time to generate the sub-federations and find a non-empty result set is 11020 milliseconds. For these 6 queries, we remark that this overhead is almost constant. That is because, a non-empty result set is found when FLiQue executes the original query on the second sub-federation.

Evaluation of queries that are relaxed. Figure 7 represents the execution of 10 queries of LargeRDFBench that need relaxation to return a non-empty result set that can be protected by a license. For each query, we compare the time to get the first result of the original query for CostFed, and the time to get the first result of the first candidate query found by FLiQue. The FLiQue overhead corresponds to the time to check license conflicts, to compute sub-federations, and to find the first candidate query. We remark that the execution time of an original query and a candidate query is not comparable. They are not the same query, and they are not executed on the same number of data sources. To have an idea (non-representative) of the similarities, the maximum is 0, 811 (L5'), the minimum is 0, 077 (C8'), the average is 0, 487, and the median is 0.603.[7] Overhead varies a lot depending on the queries. It depends on the number of generated and executed failing relaxed queries,

[7]Candidate queries and their similarities are included as supplemental material.

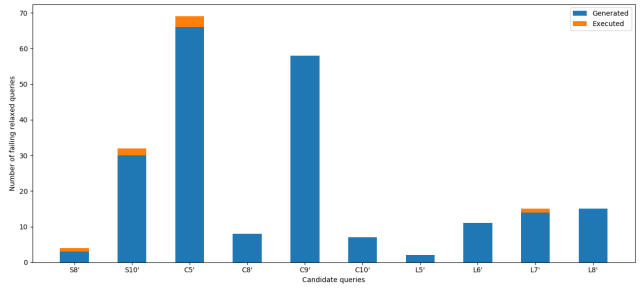

Figure 8: Number of generated and executed failing relaxed queries until finding each candidate query.

before finding the first candidate query. Figure 8 shows the number of failing relaxed queries, (1) generated, and (2) executed before finding each candidate query. We recall that an important number of generated relaxed queries are identified as failing thanks to data summaries. The candidate query $C5'$, is found after generating 69 failing relaxed queries, but only 3 were executed. In contrast, candidate query $L5'$ is found after generating 3 failing relaxed queries but executing only one. For 6 out of 10 relaxed queries, FLiQue does not need to execute any generated relaxed query to identify them as failing. With this benchmark, on average FLiQue generates 21.4 failing relaxed queries, and executes 1.75 failing relaxed queries. Thus, we consider that FLiQue success in limiting communication costs during the relaxation of queries whose result set cannot be licensed. We use SPARQL 1.0. We think that with SPARQL 1.1 and the SERVICE clause, the number of license conflicts detected by FLiQue would be less.

5 CONCLUSION

In this work, we propose FLiQue, a federated license-aware query processing strategy. It ensures that a license protects the result set of any SPARQL query. To our knowledge, this is the first work that uses query relaxation in a distributed environment. Our implementation extends an existing federated query engine with our license-aware query processing strategy. Our prototype demonstrates the usability of our approach. Experimental evaluation shows that FLiQue ensures license compliance with a limited overhead in terms of execution time. FLiQue is a step towards facilitating and encouraging the publication and reuse of licensed resources in the Web of

Data. FLiQue is not a data access control strategy. It empowers well-intentioned data users in respecting the licenses of datasets involved in a federated query.

REFERENCES

[1] Luca Costabello, Serena Villata, and Fabien Gandon. 2012. Context-Aware Access Control for RDF Graph Stores. In *European Conference on Artificial Intelligence (ECAI)*.

[2] Géraud Fokou, Stéphane Jean, Allel Hadjali, and Mickaël Baron. 2016. RDF Query Relaxation Strategies Based on Failure Causes. In *Extended Semantic Web Conference (ESWC)*.

[3] Alban Gabillon and Léo Letouzey. 2010. A View Based Access Control Model for SPARQL. In *International Conference on Network and System Security (NSS)*.

[4] Olaf Görlitz and Steffen Staab. 2011. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *Workshop Consuming Linked Data (COLD) collocated with ISWC*.

[5] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2020. Knowledge Graphs. *CoRR* abs/2003.02320 (2020).

[6] Hai Huang, Chengfei Liu, and Xiaofang Zhou. 2012. Approximating Query Answering on RDF Databases. *Journal of World Wide Web* 15 (2012).

[7] Carlos A Hurtado, Alexandra Poulovassilis, and Peter T Wood. 2008. Query Relaxation in RDF. *Journal on Data Semantics X* (2008).

[8] Renato Iannella and Serena Villata. 2018. ODRL Information Model 2.2. *W3C Recommendation* (2018).

[9] Yasar Khan, Muhammad Saleem, Aftab Iqbal, Muntazir Mehdi, Aidan Hogan, Axel-Cyrille Ngonga Ngomo, Stefan Decker, and Ratnesh Sahay. 2014. SAFE: Policy Aware SPARQL Query Federation Over RDF Data Cubes. In *Semantic Web Applications and Tools for Life Sciences (SWAT4LS)*.

[10] Sabrina Kirrane, Ahmed Abdelrahman, Alessandra Mileo, and Stefan Decker. 2013. Secure Manipulation of Linked Data. In *International Semantic Web Conference (ISWC)*.

[11] Benjamin Moreau, Patricia Serrano-Alvarado, Matthieu Perrin, and Emmanuel Desmontils. 2019. A License-Based Search Engine. In *Extended Semantic Web Conference (ESWC), Demo*.

[12] Benjamin Moreau, Patricia Serrano-Alvarado, Matthieu Perrin, and Emmanuel Desmontils. 2019. Modelling the Compatibility of Licenses. In *Extended Semantic Web Conference (ESWC)*.

[13] Said Oulmakhzoune, Nora Cuppens-Boulahia, Frédéric Cuppens, Stephane Morucci, Mahmoud Barhamgi, and Djamal Benslimane. 2014. Privacy Query Rewriting Algorithm Instrumented by a Privacy-Aware Access Control Model. *Annals of Telecommunications* 69 (2014).

[14] Tassilo Pellegrini, Giray Havur, Simon Steyskal, Oleksandra Panasiuk, Anna Fensel, Victor Mireles, Thomas Thurner, Axel Polleres, Sabrina Kirrane, and Andrea Schönhofer. 2019. DALICC: A License Management Framework for Digital Assets. *Proceedings of the Internationales Rechtsinformatik Symposion (IRIS)* 10 (2019).

[15] Pavan Reddivari, Tim Finin, Anupam Joshi, et al. 2007. Policy-Based Access Control for an RDF Store. In *Workshop Semantic Web for Collaborative Knowledge Acquisition (SWeCKa) collocated with IJCAI*.

[16] Muhammad Saleem, Ali Hasnain, and Axel-Cyrille Ngonga Ngomo. 2018. LargeRDFBench: a Billion Triples Benchmark for Sparql Endpoint Federation. *Journal of Semantic Web* 48 (2018).

[17] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. 2014. HIBISCuS: Hypergraph-Based Source Selection For SPARQL Endpoint Federation. In *Extended Semantic Web Conference (ESWC)*.

[18] Muhammad Saleem, Alexander Potocki, Tommaso Soru, Olaf Hartig, and Axel-Cyrille Ngonga Ngomo. 2018. CostFed: Cost-Based Query Optimization for SPARQL Endpoint Federation. In *International Conference on Semantic Systems (SEMANTICS)*.

[19] Oshani Seneviratne, Lalana Kagal, and Tim Berners-Lee. 2009. Policy-Aware Content Reuse on the Web. In *International Semantic Web Conference (ISWC)*.

[20] Serena Villata and Fabien Gandon. 2012. Licenses Compatibility and Composition in the Web of Data. In *Workshop Consuming Linked Data (COLD) collocated with ISWC*.