SAFE: Policy Aware SPARQL Query Federation Over RDF Data Cubes

Yasar Khan[1], Muhammad Saleem[2], Aftab Iqbal[1], Muntazir Mehdi[1], Aidan Hogan[3], Panagiotis Hasapis[4], Axel-Cyrille Ngonga Ngomo[2], Stefan Decker[1], and Ratnesh Sahay[1]

[1] Insight Centre for Data Analytics, NUI Galway, Ireland
[2] AKSW, University of Leipzig, Germany
[3] Department of Computer Science, Universidad de Chile
[4] INTRASOFT International, Luxembourg

Abstract. Several query federation engines have been proposed for accessing public Linked Open Data sources. However, in many domains, resources are sensitive and access to these resources is tightly controlled by stakeholders; consequently, privacy is a major concern when federating queries over such datasets. In this paper, we present SAFE, a SPARQL query federation engine that enables decentralised, policy-aware access to clinical information represented as RDF data cubes. We validate the performance of the system with experiments over real-world datasets provided by three clinical organisations as well as legacy Linked Datasets. In our evaluation, we show that SAFE enables granular graph-level access control over distributed clinical RDF data cubes and efficiently reduces the query execution time when compared with general-purpose SPARQL query federation engine.

1 Introduction

Inspired by the publication of hundreds of Linked Datasets on the Web, researchers have been investigating federated querying techniques to enable access to this decentralised content. Query federation aims to offer clients a single-point-of-access through which distributed data sources can be queried in unison. In the context of Linked Data, various optimised query federation engines have been proposed that can federate multiple SPARQL interfaces [2, 10, 14, 19, 20].

However, in the context of the Healthcare and Life Sciences (HCLS) domain – where data-integration is often vital – real-world datasets contain sensitive information: strict ownership is granted to individuals working in hospitals, research labs, clinical trial organisers, *etc.* Therefore, the legal and ethical concerns on (i) preserving the anonymity of patients (or clinical subjects); and (ii) respecting data ownership through access control; are key challenges faced by the data analytics community working within the HCLS domain. Our focus in this paper is on point (ii), *i.e.*, proposing a policy-based access control mechanism for user-restricted resources residing at different locations.

The key challenges for federated querying are efficient source selection (*i.e.*, determining which sources are (ir)relevant) and query planning (*i.e.*, determining an efficient query execution strategy). Query-federation engines often apply source selection at the level of endpoints, whereas in a controlled environment, a user may only have access

to certain information *within* an endpoint. Adding an access control layer to existing SPARQL query federation engines thus adds unique challenges: (i) source selection should be granular enough to enable effective access control, and (ii) it should be policy-aware to avoid wasteful requests to unauthorised resources. Therefore, in this paper we present SAFE, a SPARQL query federation engine that supports policy-based access to sensitive statistical data. SAFE is motivated by the needs of three clinical organisations in the context of an EU project who wish to enable *controlled* federation over statistical clinical data – such as data from clinical trials – owned and hosted by multiple clinical sites, represented in the form of *data cubes*: multi-dimensional arrays of numeric data.

SAFE extends upon the FedX engine [19] with two novel contributions: (i) GRAPH-LEVEL SOURCE SELECTION in order to enable graph-based access-control and (ii) OPTIMISATIONS FOR FEDERATING QUERIES OVER STATISTICAL DATA that are represented using the RDF Data Cube Vocabulary. With these modifications, we show that when compared with FedX, SAFE can (i) support more granular graph-level access control on top, and can (ii) efficiently reduce the query execution time when federating over RDF data cubes. It is important to note that no existing SPARQL query federation engine supports policy-aware access control over statistical datasets. Therefore, we argue that a specialised extension is required in general-purpose query federation engines – like FedX – to address the specific challenges in combining statistical and distributed datasets with access restrictions.

The rest of the paper is structured as follows: Section 2 discusses our motivational scenario where data from different clinical locations need to be queried and aggregated. Section 3 discusses related work. Section 4 presents the three main components of SAFE query processing. Section 5 presents evaluation of SAFE against internal and external data sets and Section 6 concludes our work.

2 Motivating Scenario

Our work is motivated by the needs of three clinical organisations: University Hospital Lausanne (CHUV)[1], Cyprus Institute of Neurology and Genetics (CING)[2], and ZEINCRO[3]. These organisations wish to develop a platform for analysing clinical data across multiple clinical sites, which would allow for increasing the total number of patients that are included in each analysis, thus increasing the statistical power of conclusions related to biomarkers, effectiveness and/or side-effects of drugs or combinations of drugs, correlations between patient groups, *etc.* The ultimate goal is to enable the collaborative identification of new drugs and treatments while reducing the high costs associated with clinical trials.

Use of Linked Data: With these goals in mind, the three clinical organisations mentioned are partners in the Linked2Safety EU project[4]. The two main goals of the Linked2Safety project are (i) the discovery of eligible patient data—also known as subject selection

[1] `http://www.chuv.ch/`
[2] `http://www.cing.ac.cy/`
[3] `http:/www.zeincro.com/`
[4] `http://www.linked2safety-project.eu/`

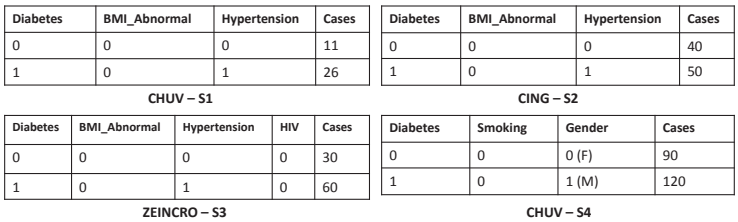

Fig. 1: Example (2D) data cubes published by CHUV, CING and ZEINCRO

criteria—that can be recruited for clinical trials from multiple clinical sites; and (ii) enabling multi-centre epidemiological studies enabling better understanding of relationships between pathological processes, risk factors, adverse events, and between genotype and phenotype. Although Linked Data technologies can help enable multi-site interoperability and integration, the community largely focuses on datasets that can be made open to the public. In contrast, clinical data is often of an extremely sensitive nature and there is often strict legislation in place protecting the privacy of patients.

Legal and ethical implications of patient privacy: According to EU Data Protection Directive 95/46/EC[5], clinical studies that involve patient-specific information must adhere to data-access restrictions that preserve patient anonymity. More specifically, a data access mechanism must ensure that patient identity cannot be discovered by any direct or indirect means using the dataset. Similar legislation exists in other jurisdictions. To avoid sharing of individual patient records, the Linked2Safety consortium has developed a data mining approach for transforming original clinical data into statistical summaries that may aggregate (or indeed redact) multiple dimensions of raw data.

The result is a set of anonymised data cubes whose dimensions correspond to insensitive clinical parameters without personal information [3]. The resulting multidimensional output contains sufficient granularity to quickly decide if the dataset is relevant for a given analysis – *e.g.*, to understand the scale and dimensions of the data – and to perform high-level meta-analysis of aggregated data. These data cubes are represented in a standard format – namely RDF Data Cube vocabulary per the recent W3C recommendation [15] – to enable interoperability (*e.g.*, use of controlled vocabularies for dimensions) and to allow the later use of Linked Data publishing/access methods.

Although the data considered are aggregated and do not contain personal information about patients, deanonymisation may still be possible [8]: one cannot open a dataset and *fully* guarantee that it will not (indirectly) compromise patient anonymity. Likewise, if a (bio)medical dataset necessarily involves genetic data, there exist identifying markers by which patients can be directly deanonymised; thus genetic data can only be pseudoanonymised. Given such issues, in practice, sharing clinical datasets – even aggregated statistics – is often conducted under a strict legal framework between parties.

In order to employ stricter data access restrictions on the anonymised multi-dimensional RDF data cubes, we then require an access-control–based query-federation approach that enforces and optimises for restricted user access over these RDF data cubes. To further illustrate and motivate, we now walk though an example.

[5] `http://www.dataprotection.ie/docs/EU-Directive-95-46-EC/89.htm`

```
1  PREFIX qb: <http://purl.org/linked-data/cube#>
2  PREFIX sehr: <http://hcls.deri.ie/l2s/sehr/1.0/>
3  SELECT ?diabetes ?bmi ?hypertension ?cases
4  WHERE { ?dataset a qb:DataSet.
5          ?observation qb:dataSet  ?dataset;
6          a qb:Observation;  sehr:Diabetes ?diabetes ;
7          sehr:BMI_Abnormal ?bmi ;
8          sehr:Hypertension ?hypertension ; sehr:Cases ?cases . }
```

Fig. 2: Example subject selection criteria for clinical trials

Figure 1 shows four sample data cubes published by three different clinical sites. Each observation represents the total number of patients exhibiting a particular adverse event. For example, the `CHUV-S1` observations describe the total number of patients (in the **Cases** column) that exhibit a particular combination of three adverse events: **Diabetes**, (Abnormal) **BMI_Abnormal** (Body Mass Index) and/or **Hypertension**. The value 0 or 1 indicates if the condition is present or not. For example, the second row in `CHUV-S1` shows that there are 26 cases presenting with both **Diabetes** and **Hypertension** but without **BMI_Abnormal**.

Once the data are published by clinical sites, they should be accessible to clinical researchers. Figure 2 shows a sample SPARQL query specifying subject-selection criteria, asking for the counts of cases that involve some combination of diabetes, abnormal BMI, and hypertension. An answer returned by the query, *i.e.*, number of cases, will play a major role in deciding the resources (*i.e.*, number of subjects, location, *etc.*) required for conducting a clinical trial. However, answering such a query requires integrating RDF data cubes with three dimensions – **Diabetes**, **Hypertension**, **BMI_Abnormal** – and the respective counts originating from multiple clinical sites.

Referring back to Figure 1, only three of the datasets (`CHUV-S1`, `CING-S2` and `ZEINCRO-S3`) contain all required dimensions. An answer returned by the query (Figure 2) should list counts (*i.e.*, *cases*) from these three RDF data cubes. However, assuming that the policy restrictions are applied to the user (say *James*), who wants to execute the query and has access to `CHUV-S1` and `CING-S2` RDF data cubes only. Therefore, the query federation engine should retrieve results only from `CHUV-S1` and `CING-S2` and should not consider `ZEINCRO-S3` for querying.

Hence, one of the key requirements in the context of the Linked2Safety project is to support federation of queries over clinical data distributed at multiple clinical sites by taking into account the data access policies (Figure 3 (c): shows a data access policy) assigned to the users (Figure 3 (a): shows a user profile for James) executing those queries. Since RDF data cubes are self-contained entities associated with additional provenance information (*e.g.*, creator, location, *etc.*; see Figure 3 (b)), they are modelled using named graphs [6] as supported in SPARQL: each named graph contains only one data cube and its provenance information.

In order to publish clinical data cubes as RDF (Figure 1) and describe user profiles along with their access rights (Figure 3) used within query federation process, the Linked2Safety consortium has developed two vocabularies: (i) *Semantic EHR Model (prefix "`sehr`")* describes the clinical terminologies used by the three clinical partners; and (ii) *Access Policy Model (prefix "`lmds`")* describes the user profiles (their activity,

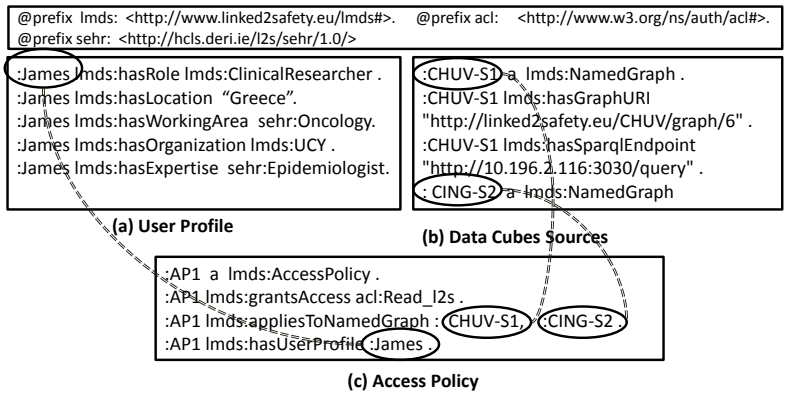

Fig. 3: Snippets of user profile, access policy and data cube source information

location, organisation, position and role) and their respective access rights (*e.g.*, read, write). Considering space limitations, further details of these two vocabularies are out of scope for this paper; we instead refer the readers to dedicated papers on the Semantic EHR Model [16] and the Access Policy Model [11].

3 Related Work

Various approaches have been proposed to query RDF data in a decentralised manner. Here, we focus on approaches that look at federating multiple SPARQL endpoints. Likewise, a number of works have looked at access control over Linked Data. We now briefly discuss both categories of work in the following.

SPARQL Query Federation: Many query federation engines have been proposed for SPARQL (*e.g.*, [1,2,4,10,13,14,18–20]). Such engines accept an input query, decompose it into sub-queries, decide relevance of individual data sources (typically considering sources at the level of endpoints) for sub-queries, forward the sub-queries to the individual endpoints accordingly and merge the final results for the query. Such engines aim to find and execute optimised query plans that minimise initial latency and total runtimes. This can be achieved by (i) using accurate source selection to minimise irrelevant messages, (ii) implementing efficient join algorithms, (iii) and using caching techniques to avoid repeated sub-queries. Source selection is typically enabled using a local index/catalogue and/or probing sources with queries at runtime. The former approach assumes some knowledge of the content of the underlying endpoints and requires update/synchronisation strategies. However, the latter approach incurs a higher runtime cost. Thus, many engines support a hybrid of index and query-based source selection.

Table 1 gives an overview of existing SPARQL query federation engines with respect to source selection type, physical join operators, use of caching and explicit support for updates. We also remark on whether code is available for the system.

In this setting, our work builds upon an existing federated engine – FedX – with support for an access-control layer over statistical data represented as RDF data cubes.

Table 1: Overview of existing SPARQL query federation engines

Systems	Source Selection	Join Type	Code	Policy	Cache	Update
ADERIS [13]	index	nested loop	✓	✗	✗	✗
ANAPSID [1]	query & index	adaptive	✓	✗	✗	✓
Avalanche [4]	query & index	distributed, merge	✗	✗	✓	✗
DARQ [14]	index	nested loop, bind	✓	✗	✗	✗
DAW [18]	query & index	based on underlying system	✗	✗	✓	✗
FedSearch [2]	query & index	bind, pull-based rank	✗	✗	✓	✗
FedX [19]	query	nested loop, bind	✓	✗	✓	✗
LHD [20]	query & index	hash, bind	✓	✗	✗	✗
SPLENDID [10]	query & index	hash, bind	✓	✗	✗	✗
SAFE	query & index	nested loop, bind	✓	✓	✓	✓

Access Control for SPARQL: Various authors have explored access control models for SPARQL query engines. Gabillon and Letouzey [9] propose applying access control over named graphs and views, which are defined as graphs dynamically generated using SPARQL `CONSTRUCT` or `DESCRIBE` queries. Costabello et al. [7] propose SHI3LD: an access control framework for SPARQL 1.1 query engines that operates on the level of named graphs where permissions are based on the context of the user in the setting of a mobile device; permissions are checked using SPARQL `ASK` queries. Kirrane et al. [12] propose using stratified Datalog rules to enforce an access control model that operates over quad patterns, thus offering higher granularity of control.

SAFE is designed specifically to query statistical RDF data cubes in a distributed setting, where access control is coupled with source selection and both operate on the same level of granularity: named graphs. Access control – deny or allow access – is based on user profiles and their access rights.

4 Methodology and Architecture

SAFE's architecture is summarised in Figure 4, which shows its three main components: (i) Source Selection: performs multilevel source selection based on the capabilities of data sources; (ii) Policy Aware Query Planning: filters the selected data sources based on access rights defined for each user; and (iii) Query Execution: performs the execution of sub-queries against the selected sources and merges the results returned. In the following, we describe these components in detail.

4.1 Source Selection:

SAFE performs a tree-based two-level source selection as shown in Figure 5. At Level 1, like other query federation engines [1, 10, 17, 19, 20], we do *triple-pattern-wise endpoint selection*, *i.e.*, we identify the set of relevant endpoints that will return non-empty results for the individual triple pattern in a query. At Level 2 (unlike other query federation engines), SAFE performs *triple-pattern-wise named graph selection*, *i.e.*, we identify

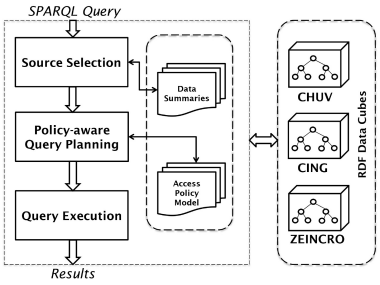

Fig. 4: SAFE architecture

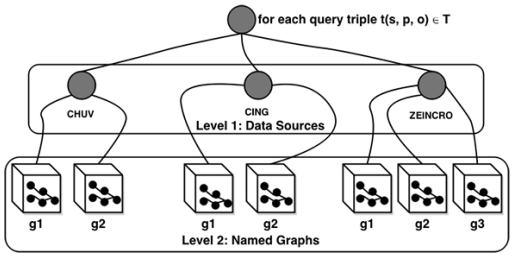

Fig. 5: Tree-based two level source selection

:source1	**lmds:endpointUrl**	<http://10.196.2.116:3030/query> .	CHUV
:source1	**lmds:cube**	:CHUV-s1 , :CHUV-s4 .	
:CHUV-s1	**lmds:graph**	<http:...CHUV-s1/graph/0007> .	
:CHUV-s1	**lmds:cubeProperties**	sehr:Diabetes, sehr:BMI_Abnormal, sehr:Hypertension, sehr:Cases .	
:CHUV-s4	**lmds:graph**	<http:...CHUV-s4/graph/0007> .	
:CHUV-s4	**lmds:cubeProperties**	sehr:Diabetes, sehr:Smoking, sehr:Gender, sehr:Cases .	
:source2	**lmds:endpointUrl**	<http://10.196.2.117:3030/query> .	CING
:source2	**lmds:cube**	:CING-s2 .	
:CING-s2	**lmds:graph**	<http:...CING-s2/graph/0007> .	
:CING-s2	**lmds:cubeProperties**	sehr:Diabetes, sehr:BMI_Abnormal, sehr:Hypertension, sehr:Cases .	
:source3	**lmds:endpointUrl**	<http://10.196.2.118:3030/query> .	ZEINCRO
:source3	**lmds:cube**	:ZEINCRO-s3 .	
:ZEINCRO-s3	**lmds:graph**	<http:...ZEINCRO-s3 /graph/0007> .	
:ZEINCRO-s3	**lmds:cubeProperties**	sehr:Diabetes, sehr:BMI_Abnormal, sehr:Hypertension, sehr:HIV, sehr:Cases .	

Fig. 6: SAFE data summaries

a set of relevant named graphs containing RDF data cubes for all relevant endpoints already identified at Level 1. SAFE relies on data summaries to identify relevant named graphs.

Data Summaries: We assume a set of datasets $\mathcal{D}$ where each dataset $D \in \mathcal{D}$ is a RDF dataset: $D := \{(u_1, G_1), \ldots (u_n, G_n)\}$, where each (u_i, G_i) is a named graph with (unique) URI u_i. In our case, named graphs refer to individual RDF data cubes as we do not consider a default graph. We denote all graph names by $\mathrm{names}(D)$ and a graph in the dataset by $D(u) := G$. We denote by $\mathrm{preds}(G) := \{p \mid \exists s, o : (s, p, o) \in G\}$ the set of all distinct predicates in G and by $\mathrm{preds}(D) := \bigcup_{(u,G)\in D} \mathrm{preds}(G)$ the set of all distinct predicates in D. For each dataset $D \in \mathcal{D}$, SAFE stores the following as a data summary: (i) the endpoint URL (`lmds:endpointUrl`), where each endpoint indexes a dataset D; (ii) the set of all graph names in a dataset D: $\mathrm{names}(D)$ where each graph contains a RDF data cube (`lmds:cube`/`lmds:graph`); and (iii) for each graph $G \in D$, the set of all predicates in G: $\mathrm{preds}(G)$ (`lmds:cubeProperties`).

We (informally) denote the set of all data summaries for $\mathcal{D}$ as $\mathcal{S}$ and the data summary for a particular source as $\mathcal{S}(D)$. A snippet of a data summary generated for the sample RDF data cubes published by three clinical sites (CHUV, CING, ZEINCRO) of Figure 1 is shown in Figure 6, where `CHUV` contains two RDF data cubes (`CHUV-S1`, `CHUV-S4`), CING contains one RDF data cube (`CING-s2`), and ZEINCRO also contains only one RDF data cube (`ZEINCRO-s3`). Before explaining the algorithm in the next section,

```
PREFIX acl: <http://www.w3.org/ns/auth/acl#>
PREFIX lmds: <http://www.linked2safety.eu/lmds#>
ASK WHERE {
?accessPolicy a lmds:AccessPolicy .
?accessPolicy lmds:appliesToNamedGraph ?namedGraph .
?namedGraph lmds:hasGraph :CHUV_S1 .
?accessPolicy lmds:grantsAccess acl:Read_l2s .
?accessPolicy lmds:hasRequesterProfile :James . }
```

Fig. 7: SPARQL query authenticating a user against a data cube/named graph

we wish to make a formal description of the sets that are calculated on-the-fly by the algorithm as part of the data summary:

1. The set of all predicates in a dataset D : $\text{preds}(D)$. This is the set-union of all $\text{preds}(G_i)$ for each $G_i \in D$; *e.g.*, $\text{preds}(\mathsf{CHUV}) = \{\texttt{sehr:Diabetes}, \texttt{sehr:BMI\_Abnormal}, \texttt{sehr:Hypertension}, \texttt{sehr:Smoking}, \texttt{sehr:Gender}, \texttt{sehr:Cases}\}$ (ref. Figure 1, Figure 6).
2. The set of unique predicates in a dataset D: $\text{upreds}(D) \coloneqq \{p \in \text{preds}(D) \mid \nexists D' \in \mathcal{D} : D \neq D' \wedge p \in \text{preds}(D')\}$; e.g., $\text{upreds}(\mathsf{CHUV}) = \{\texttt{sehr:Smoking}, \texttt{sehr:Gender}\}$ (ref. Figure 1, Figure 6).
3. The set of unique properties in a graph with name u: $\text{upreds}(u, D) \coloneqq \{p \in \text{preds}(D(u)) \mid \nexists u' : u' \neq u \wedge p \in \text{preds}(D(u)')\}$ (overloading $\text{upreds}(.,.)$ for use with graphs also); *e.g.*, $\text{upreds}(\texttt{:CHUV-s1}, \mathsf{CHUV}) = \{\texttt{sehr:BMI\_Abnormal}, \texttt{sehr:Hypertension}\}$ and $\text{upreds}(\texttt{:CHUV-s4}, \mathsf{CHUV}) = \{\texttt{sehr:Smoking}, \texttt{sehr:Gender}\}$ (ref. Figure 1, Figure 6).
4. The set of graph names in D with unique properties: $\text{unames}(D) \coloneqq \{u \in \text{names}(D) \mid \text{upreds}(u, D) \neq \emptyset\}$. For example, $\text{unames}(\mathsf{CHUV}) = \{\texttt{:CHUV-s1}, \texttt{:CHUV-s4}\}$.

4.2 Policy-Aware Query Planning and Query Execution:

Once the relevant sources are identified for a SPARQL query, the next step is to further filter these sources by authenticating the user that is making the request. Policy-Aware Query Planning is the process of identifying *capable sources*: relevant sources that the user has access rights for. Access policies on each source are defined in the access policy model [11]; for this, we consider graph-level control. The user authorisation is done by running SPARQL ASK queries encoding information about the user and the relevant named graphs against the access-policy store. Considering the example discussed in Section 2, the SPARQL query generated for authenticating the user `:James` for accessing the named graph `:CHUV-S1` is shown in Figure 7. This query asks if there is any access policy that grants read access to the user `:James` for the named graph `:CHUV-S1`, returning `true` or `false` as a result. As per the Figure 3 (c), this example will return `true`.

Relevant named graphs that return `false` will be filtered. Endpoints with capable named graphs are then queried using standard federation techniques. For this, we use the FedX query engine [19], amending the query rewriter to append the capable graph information for each endpoint.

Algorithm 1: Access policy-based triple pattern-wise source and named graph selection

```
Data: 𝒟 = {D_1, ..., D_n}, BGP = {bgp_1, ..., bgp_m}, 𝒮, P, user
/* sources, BGPs of a SPARQL query, SAFE summaries of sources, Access
   Policies, User                                                    */
1  ℛ ← {} ;                                          /* initialise relevance set */
2  for each bgp ∈ BGP do                                          /* for each BGP */
3      for each t ∈ bgp do                          /* for each triple pattern in BGP */
4          R ← {} ;          /* initialise set of capable source graphs for t */
5          s ← subj(t), p ← pred(t), o ← obj(t) ;
6          if bound(p) then                                /* if predicate p is bound */
7              for each D ∈ 𝒟 do                          /* for each data source in 𝒟 */
8                  U ← upreds(D) ;   /* source unique properties read from 𝒮(D) */
9                  A ← ⋃_{D'∈𝒟} upreds(D') ; /* all unique properties read from 𝒮(D')
                   */
10                 E ← preds(𝒟) ;                          /* all properties of 𝒟 */
11                 if p ∈ E ∧ (|bgp| = 1 ∨ (StarJoin(t, bgp, U) ∨ PathJoin(t, bgp, U)) ∨ (!StarJoin(t,
                   bgp, A \ U) ∧ !PathJoin(t,bgp,A \ U)) then
                       /* triple pattern-wise named graph selection          */
12                     UG ← {} ;
13                     for each ug ∈ unames(D) do          /* for each unique cube */
14                         if preds(bgp) ∩ upreds(ug, D) ≠ ∅ then   /* unique p in bgp */
15                             UG ← UG ∪ {ug} ;          /* ug might be relevant */
16                     if UG = ∅ then                  /* nothing unique so add all */
17                         R ← R ∪ ({t} × names(D) × {D}) ;
18                     else if |UG| = 1 then                       /* one unique cube */
19                         R ← R ∪ ({t} × UG × {D}) ;
                       /* if |UG| > 1, no source can match the join          */
20         else if !bound(p) then
               /* triple-pattern-wise source selection using ASK queries     */
21             for each D ∈ 𝒟 do
22                 if ASK(D, t) = true then
23                     R ← R ∪ ({t} × names(D) × {D}) ;     /* select all graphs */
           /* do policy-based graph filtering                                */
24         for each cg_i ∈ R do                          /* for each capable graph */
25             if ASK(D, cg_i, P, user) = false then   /* Authorise user against graph of a
               source using access policies */
26                 RemoveGraph(cg_i, R) ;              /* remove unauthorised graph */
27         ℛ ← ℛ ∪ {R}
28 return ℛ ;          /* return relevant sources and graphs for triple patterns */
```

4.3 Algorithm:

SAFE's access policy-based triple-pattern-wise source selection is shown in Algorithm 1. The algorithm is designed to exploit specific properties of RDF data cubes, particularly the locality of joins: assuming RDF data cubes are not split over sources, certain types of joins can only be answered locally. In particular, we define a subject–subject join (s–s join), where two triple patterns share (only) a subject variable, and a subject–object join (s–o join), where the join variable appears in the subject position of one triple pattern and the object of the other. For example, in Figure 2, triple patterns 1–2 (*lines 4–5*) form an s–o join and triple patterns 2–7 (*lines 5–8*) form an s–s join. As per the example, such joins would have to be answerable by one source/data cube; thus (reasonably) assuming that RDF data cubes are not split across sources, we can exploit this locality with a

join-aware strategy that reduces sources considered relevant while ensuring complete results. For example, in Figure 2, though many sources will match the first triple pattern, they will not be considered relevant unless they are relevant for later triple patterns also.

The algorithm takes the set of all available datasets $\mathcal{D}$, their data summaries $\mathcal{S}$, the access policy P, user info, and a SPARQL query containing a set of basic graph patterns[6] BGP as input (source selection only refers to BGPs, which may be extracted from features such as `UNION` or `OPTIONAL`, etc.). The algorithm returns the set of relevant sources and corresponding named graphs for individual triple patterns as output. We process the individual triple patterns of each BGP separately (*lines 1–2* of Algorithm 1). Given a triple pattern $t \in bgp \in BGP$ with bound predicate p, for each dataset $D \in \mathcal{D}$, we collect the dataset-unique properties U, all unique properties A, and all dataset properties E from data summaries (*lines 3–10* of Algorithm 1). A dataset D is relevant for triple pattern t if its predicate is a set member of E and either t forms a star (s–s) or path join (p–o) with any other triple pattern (in the same query) having a predicate in U or t does not form both star and path join with set difference A/U (*line 11* of Algorithm 1). Once a relevant source is selected, the next step is to identify the set of relevant graphs within that relevant dataset (*lines 12–19* of Algorithm 1). If triple pattern t belongs to a unique graph ug in selected data source D then only uq is selected as relevant graph (*lines 13–15* of Algorithm 1). If there is no unique graph in D then all graphs in D are selected as relevant (*lines 16–19* of Algorithm 1). If a predicate is not bound in t, we fall back to a standard strategy and make use of SPARQL `ASK` queries for source selection, *i.e.*, we send a SPARQL `ASK` request to each of the endpoints (*lines 20–23* of Algorithm 1). Once relevant graphs within relevant data sources are selected, the final step is to further prune the select capable graphs using policy-based filtering (*lines 24–26* of Algorithm 1). A capable graph cg is removed if the user does not have access (according to policies P) for cg.

As per our running example, consider the triple pattern $tp :=$ `"?observation sehr:Diabetes ?diabetes"` of the query given in Figure 2. Since the predicate is bound in tp, the condition given at *line 6* of Algorithm 1 holds. For the CHUV dataset, $U = \{$`sehr:Smoking`, `sehr:Gender`$\}$, $A = \{$`sehr:Smoking`, `sehr:Gender`, `sehr:HIV`$\}$, and $E = \{$`sehr:Diabetes`, `sehr:BMI_Abnormal`, `sehr:Cases`, `sehr:Smoking`, `sehr:Hypertension`, `sehr:Gender`$\}$ (*line 8–10*), and $A \setminus U$ = $\{$`sehr:HIV`$\}$. The predicate `sehr:Diabetes` $\in E$ and tp does not not form a star join (s–s) or path join (s–o) with $A \setminus U$ in the query. Therefore, the condition given at *line 11* of Algorithm 1 holds and the data source CHUV will be selected as relevant for TP. The next step is to select named graphs within the CHUV data source. For both named graphs (`CHUV-s1`, `CHUV-s4`) the condition given at *line 14* is true, therefore both named graph are selected as relevant. For both CING and ZEINCRO the condition given at *line 11* also hold; therefore they are also selected. Assuming James as the user who has requested the data, the response of ASK queries (*i.e.*, `true` or `false`) will authorise the user against each capable named graph (*line 25*). In this example, James' access will be denied for the named graph `ZEINCRO-s3` and granted for the two other named graphs as specified in Figure 7(b). Hence `ZEINCRO-s3` will be removed from the set of capable graphs and only CHUV will be queried to get the desired results.

[6] `http://www.w3.org/TR/sparql11-query/#BasicGraphPatterns`

Table 2: Overview of experimental datasets

Dataset	Type	№ trip	№ obsv	№ sub	№ pred	№ obj	data	i.size	i.time
CHUV	INT	0.8 M	96 K	96 K	36	88	31 MB	-	-
CING	INT	0.1 M	17 K	17 K	21	51	5 MB	-	-
ZEINCRO	INT	0.4 M	49 K	49 K	24	59	15 MB	-	-
Total	INT	1.3 M	162 K	162 K	81	198	51 MB	8 KB	10 sec
World Bank	EXT	77 M	10 M	10 M	58	40 K	19 GB	-	-
IMF	EXT	18 M	1.8 M	1.8 M	30	3151	3.5 GB	-	-
Eurostat	EXT	0.3 M	38 K	44 K	31	5717	205 MB	-	-
Trans. Int.	EXT	43 K	3939	4286	64	5290	9.2 MB	-	-
Total	EXT	95 M	12 M	2 M	183	54 K	23 GB	12 KB	571 sec

5 Evaluation

This section presents evaluation comparing SAFE against FedX to validate the extensions we have proposed. The experimental setup (*e.g.*, datasets, queries and metrics) for evaluation are as follows:

5.1 Experimental Setup

Datasets: We use two groups of datasets exploring two different use cases.

The first group of datasets (INTERNAL) are collected from the three clinical partners involved in our primary use case as described in Section 2. These datasets contain aggregated clinical data represented as RDF data cubes and are privately owned/restricted.

The second group of datasets (EXTERNAL) are collected from legacy Linked Data containing sociopolitical and economical statistics (in the form of RDF data cubes) from the World Bank, IMF (International Monitoring Fund), Eurostat and Transparency International. The World Bank data contains a comprehensive set of information about countries around the globe, such as observations on development indicators, financial statements, climate change, research projects, *etc.* The IMF data provides a range of time series data on lending, exchange rates and other economic and financial indicators. The Eurostat data provides statistical indicators that enable comparison between countries and regions across Europe. The Transparency International data includes a Corruption Perceptions Index (CPI), which ranks countries and territories based on how corrupt their public sector is perceived to be.

Table 2 gives an overview of the experimental datasets (**i.size** refers to index size and **i.time** refers to time taken for index generation). Each dataset was loaded into a different SPARQL endpoint (using Jena Fuseki) on separate physical machines.

Queries: A total of 12 queries are designed to evaluate and compare the query federation performance of SAFE against FedX. These queries are of varying complexity and have varying type of characteristics. For space reasons, the full list of queries is made available at `http://linked2safety.hcls.deri.org:8080/SAFE-Demo/`

Table 3: Summary of query characteristics

Characteristics	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
№ of Triple Patterns	9	7	9	16	7	8	11	10	7	7	3	7
№ of Sources	3	4	4	3	4	3	3	4	3	3	3	3
№ of Results	41	50	348	41	62	1983	5	10	1701	19656	570	41
Filters							✓					
More than 9 pattens	✓		✓	✓	✓	✓	✓	✓				
Negation							✓					
`LIMIT` modifier		✓	✓				✓	✓				
`ORDER BY` modifier		✓	✓					✓				
`DISTINCT` modifier		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
`REGEX` operator					✓							
`UNION` operator	✓											

Table 4: Sum of triple-pattern-wise sources selected for each query

System	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Avg
SAFE	8	10	13	16	15	13	15	16	7	7	9	7	11
FedX	9	13	16	24	20	14	16	19	15	17	9	16	16

`queries.html`. In Table 3, we summarise the characteristics of these queries (following similar dimensions to the Berlin SPARQL benchmark [5]), showing their varying complexity. The row for the number of sources indicates those matched by at least one triple pattern.

Metrics: For each query type we measured (i) the number of sources selected; (ii) the average source selection time; (iii) the average query execution time; and (iv) the number of `ASK` requests issued to sources. The performance of SAFE and FedX was compared based on these metrics. All experiments are carried out on a local network, so that network cost remains negligible. Machines used for experiments have a 2.60 GHz Core i5 processor, 8 GB of RAM and 500 GB hard disk running a 64-bit Windows 7 OS. The results produced by FedX and SAFE are the same for all queries.

5.2 Experimental Results

Triple pattern-wise sources selected: Table 4 shows the total number of triple pattern-wise (TP) sources selected by SAFE and FedX for all the queries. The last column in Table 4 shows the average number of TP sources selected by each approach. FedX performs optimal source selection at the triple-pattern-level using `ASK` queries for each triple pattern to find out precisely which sources can answer an individual triple pattern. By using join-aware source selection designed for RDF data cubes, SAFE manages to filter further potential sources that do not contribute to the end results, thus (as we will see) reducing response times.

Number of SPARQL ASK requests: Table 5 shows the total number of SPARQL `ASK` requests used to perform source selection for each query. FedX is an index-free approach and performs runtime SPARQL `ASK` requests during source selection for each

Table 5: Number of SPARQL `ASK` requests used for source selection

System	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Avg
SAFE	0	0	0	0	0	0	0	0	0	0	0	0	0
FedX	36	28	40	64	48	40	44	40	21	21	9	21	35

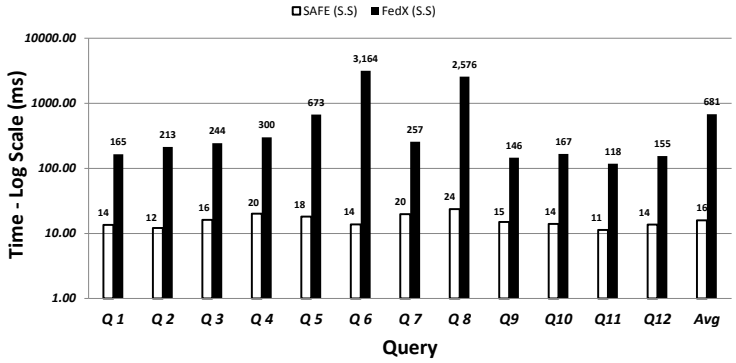

Fig. 8: Comparison of source selection time

triple pattern in query. Conversely, SAFE uses data summaries for source selection, reverting to SPARQL `ASK` requests only when there is an unbound predicate in a triple pattern. None of our evaluation queries have an unbound predicate; hence there are no SPARQL `ASK` requests for SAFE. Though flexible in the generic case, index-free approaches can incur a large cost in terms of SPARQL `ASK` requests used for source selection, which can in turn increase overall query execution time.

Source selection time: Figure 8 compares the source selection time of SAFE and FedX for all queries, where the y-axis is presented in log-scale. The rightmost pair of bars compares the average source selection time over all queries. As expected, the source selection time for SAFE is much lower than that of FedX. This is primarily attributable to SAFE's use of a domain-specific index for source-selection, which avoids incurring heavy traffic for `ASK` queries. The index can typically be pre-loaded into memory before query execution, which means that the source selection time for the presented use case(s) will be minimal.

Query execution time: For each query, the average query execution time was calculated for both approaches by running each query ten times. Figure 9 compares the overall query execution time of SAFE and FedX for all queries. Again, the y-axis is logscale and the rightmost pair of bars compares the average query execution times. The results shows that SAFE has significantly outperformed FedX in all queries in the context of the presented use cases. In fact, we see that FedX times-out in the case of three queries (in our experiments, we set queries to timeout after 25 minutes).

There are a number of factors that can influence the overall query execution time of a query federation engine, such as join type, join order selection, block and buffer size, etc. However, given that SAFE is based on the FedX architecture, we can attribute

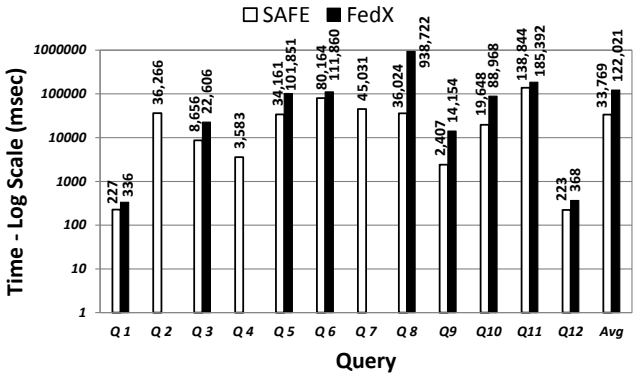

Fig. 9: Comparison of query execution time

the observed runtime improvements to three main factors: (i) source selection time is reduced (as we have seen in the previous sets of results); (ii) fewer sources are queried meaning less time spent waiting for responses; and (iii) triple patterns are more selective in SAFE, where, for example, our join-awareness makes it unlikely that all `rdf:type` triple patterns will need to be retrieved/queried for all sources but rather only from sources where such a triple pattern joins with a more selective one. Taken together, these three main observations explain the time saving observed for our presented use cases.

6 Conclusions and Future Work

In this paper, we have presented SAFE: a query federation engine that enables policy-based access to sensitive statistical datasets represented as RDF data cubes. The work is motivated in particular by the needs of three clinical organisations who wish to develop a platform for collaboratively analysing clinical data that spans multiple clinical sites, thus improving the statistical power of conclusions that can be drawn (versus one source alone). Clinical data – even in aggregated form – is of a highly sensitive nature, and thus query federation engines must take access policies into account.

SAFE is developed as an extension on top of the FedX federation engine to support two main features: (i) optimisations tailored for federating queries over RDF data cubes; and (ii) source selection on the level of named graphs that allows for integration with an existing access control layer. We evaluated these extensions based on our internal data sets (private data owned by clinical organisations) as well as external data sets (public data available from the LOD cloud) in order to measure the efficiency of SAFE against FedX. Our evaluation results show that, for our use-case(s), SAFE outperforms FedX in terms of fast source selection and query execution time. The source-code (AGPL License) and a demo for SAFE can be found at `http://linked2safety.hcls.deri.org:8080/SAFE-Demo/`.

ACKNOWLEDGMENTS*: This publication has emanated from research supported in part by the research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289,*

EU FP7 project Linked2Safety (contract number 288328), EU FP7 project GeoKnow (contract number 318159) and by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

References

1. M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: An adaptive query processing engine for sparql endpoints. In *ISWC*, 2011.
2. C. H. Andriy Nikolov, Andreas Schwarte. Fedsearch: Efficiently combining structured queries and full-text search in a sparql federation. In *ISWC*. 2013.
3. A. Antoniades, J. A. Keane, A. Aristodimou, C. Philipou, A. Constantinou, C. Georgousopoulos, F. Tozzi, K. C. Kyriacou, A. Hadjisavvas, M. Loizidou, C. Demetriou, and C. S. Pattichis. The effects of applying cell-suppression and perturbation to aggregated genetic data. In *BIBE*, pages 644–649. IEEE Computer Society, 2012.
4. C. Basca and A. Bernstein. Avalanche: Putting the spirit of the web back into semantic web querying. In *SSWS*, pages 64–79, November 2010.
5. C. Bizer and A. Schultz. The Berlin SPARQL benchmark. *IJSWIS*, 5(2):1–24, 2009.
6. J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW*, pages 613–622, 2007.
7. L. Costabello, S. Villata, and F. Gandon. Context-Aware Access Control for RDF Graph Stores. In *ECAI*, pages 282–287, 2012.
8. C. Dwork. Differential Privacy. In *ICALP (2)*, pages 1–12, 2006.
9. A. Gabillon and L. Letouzey. A View Based Access Control Model for SPARQL. In *NSS*, pages 105–112, 2010.
10. O. Görlitz and S. Staab. Splendid: Sparql endpoint federation exploiting void descriptions. In *COLD at ISWC*, 2011.
11. E. Kamateri, E. Kalampokis, E. Tambouris, and K. Tarabanis. The linked medical data access control framework. *Journal of Biomedical Informatics*, 2014. (in press).
12. S. Kirrane, A. Abdelrahman, A. Mileo, and S. Decker. Secure manipulation of linked data. In *ISWC*, pages 248–263, 2013.
13. S. Lynden, I. Kojima, A. Matono, and Y. Tanimura. Aderis: An adaptive query processor for joining federated sparql endpoints. In *OTM*, pages 808–817. 2011.
14. B. Quilitz and U. Leser. Querying distributed rdf data sources with SPARQL. In *ESWC*, pages 524–538, 2008.
15. Richard Cyganiak, Dave Reynolds, Jeni Tennison. The RDF Data Cube Vocabulary, January 2014.
16. R. Sahay, D. Ntalaperas, E. Kamateri, P. Hasapis, O. D. Beyan, M. F. Strippoli, C. Demetriou, T. Gklarou-Stavropoulou, M. Brochhausen, K. A. Tarabanis, T. Bouras, D. Tian, A. Aristodimou, A. Antoniades, C. Georgousopoulos, M. Hauswirth, and S. Decker. An ontology for clinical trial data integration. In *SMC*, pages 3244–3250. IEEE, 2013.
17. M. Saleem and A.-C. N. Ngomo. Hibiscus: Hypergraph-based source selection for sparql endpoint federation. In *ESWC*, 2014.
18. M. Saleem, A.-C. N. Ngomo, J. X. Parreira, H. F. Deus, and M. Hauswirth. Daw: Duplicate-aware federated query processing over the web of data. In *ISWC*, pages 561–576, 2013.
19. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *ISWC*, pages 601–616. 2011.
20. X. Wang, T. Tiropanis, and H. C. Davis. Lhd: Optimising linked data query processing using parallelisation. In *LDOW at WWW*, 2013.