Querying over Federated SPARQL Endpoints—A State of the Art Survey

Nur Aini Rakhmawati Jürgen Umbrich
Marcel Karnstedt Ali Hasnain
Michael Hausenblas

DERI Technical Report 2013-06-07
June 2013

DERI Galway
IDA Business Park
Lower Dangan
Galway, Ireland
http://www.deri.ie/

DERI Technical Report
DERI Technical Report 2013-06-07, June 2013

Querying over Federated SPARQL Endpoints—A State of the Art Survey

Nur Aini Rakhmawati[1] Jürgen Umbrich[1] Marcel Karnstedt[1]
Ali Hasnain[1] Michael Hausenblas[1]

Abstract. The increasing amount of Linked Data and its inherent distributed nature have attracted significant attention throughout the research community and amongst practitioners to search data, in the past years. Inspired by research results from traditional distributed databases, different approaches for managing federation over SPARQL Endpoints have been introduced. SPARQL is the standardised query language for RDF, the default data model used in Linked Data deployments and SPARQL Endpoints are a popular access mechanism provided by many Linked Open Data (LOD) repositories. In this paper, we initially give an overview of the federation framework infrastructure and then proceed with a comparison of existing SPARQL federation frameworks. Finally, we highlight shortcomings in existing frameworks, which we hope helps spawning new research directions.

Keywords: Federation, SPARQL, RDF, Linked Data.

[1]DERI, National University of Ireland, Galway, Ireland. firstname.lastname@deri.org

Acknowledgements: This work has been carried out in the Linked Data Research Centre (LiDRC) and has been supported by Science Foundation Ireland under Lion 2 and the European Commission's FP7 Support Action LOD-Around-The-Clock (LATC), project no. 256975, Intelligent Information Management (ICT-2009.4.3).

Contents

1 Introduction **1**

2 Related Works **3**

3 Motivating Example **3**

4 Infrastructure for Querying Linked Data **4**

5 Federation over SPARQL Endpoints Basic Approach and Technique **7**
5.1 Architecture of Federation over SPARQL Endpoints 7
5.2 Basic Steps of Federation over SPARQL Endpoints 9
5.2.1 Query parser 9
5.2.2 Source Selection 10
5.2.3 Query Planning and Execution 12

6 The Existing Federation over SPARQL Endpoints Frameworks **13**
6.1 The Existing Federation over SPARQL Endpoints Frameworks Based on Their Architecture . 13
6.1.1 Frameworks Support SPARQL 1.1 Federation Extension 13
6.1.2 Frameworks Supports Federation over SPARQL Endpoint, build on top of SPARQL 1.0 14
6.1.3 Frameworks Supports Federation over SPARQL Endpoint, build on top of SPARQL 1.1 16

7 Desired Features **17**

8 Challenges **18**

9 Conclusion **20**

1 Introduction

The Resource Description Framework (RDF)[1] was introduced in the last decade and now has become a standard for exchanging data in the Web. At present, huge amount of data has been converted to RDF. The SPARQL Protocol and RDF Query Language (SPARQL)[2] was officially introduced in 2008 to retrieve RDF data as easily as SQL[3] does for relational databases. As Web of data grows with more applications rely on it, the number of SPARQL Endpoints constructing SPARQL queries over Web of Data using HTTP also grows fast. SPARQL Endpoint becomes main preferences to access data because it is a flexible way to interact with Web of Data by formulating query like SQL in traditional database. Additionally, it returns query answer in several formats, such as XML and JSON which are widely used as data exchange standard in various applications. This situation has attracted people to aggregate data from multiple SPARQL Endpoints akin to conventional distributed databases. For instance, NeuroWiki[4] collects data from multiple life science RDF store by utilizing LDIF framework [SMI$^+$] and RKBexplorer[5] gathers research publication information from more than 20 datasets under `rkbexplorer.com` domain [MGSS10].

Querying data in the Web of Data context is more challenging than collecting information in traditional distributed databases, as Web of Data has no global schema, typically offering heterogeneity in terms of vocabularies [FCOO12]. The publishers use their own vocabulary to produce their data, therefore the consumer should understand the layout of the dataset before querying it. In contrast, traditional databases provide global schema, allowing consumers to request data in a straightforward manner. To deal with the lack of a global schema, querying data from multiple sources could be solved by link establishing among datasets [HB11]. The publisher only needs to generate a link from his dataset to other dataset by doing entity matching among multiple datasources. Entity matching is the process of connecting two entities located in different datasets, related to each other. SILK [JIB10] and LIMES [NA11] are two tools allowing to generate links semi-automatically. The set of interlinked Web of Data datasets creates Linked Data[6]. According to Linked Open Data (LOD) cloud statistics[7], as of September 2011, 89.83 % of datasets has more than 1000 links to other datasets. Those links are beneficial to aggregate data from multiple datasets. Consider, for example, all drug information in DBpedia (`http://dbpedia.org`) can be connected with drugs in Drugbank (`http://www4.wiwiss.fu-berlin.de/drugbank/)` by the *owl:sameAs* relation; which is an identity link that joins two entities having the same identity. This may result in easy querying data amongst multiple SPARQL Endpoints. For example, if we search data about a drug, we can collect data from *Drugbank*[8], *DBPedia*[9] and *Kegg*[10] SPARQL Endpoints as shown in Figure 1. The oval in the Figure 1 represents entity whereas the box denotes property value. Querying data from three datasets produces drug information such as its indication and compounds. We obtain drug information and its indication from Drugbank and DBpedia by

[1] `http://www.w3.org/RDF/`
[2] `http://www.w3.org/TR/rdf-sparql-query/`
[3] `http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498`
[4] `http://neurowiki.alleninstitute.org/`
[5] `http://www.rkbexplorer.com/explorer/`
[6] `http://www.w3.org/standards/semanticweb/data`
[7] `http://lod-cloud.net/state/`
[8] `http://www4.wiwiss.fu-berlin.de/drugbank/sparql`
[9] `http://dbpedia.org/sparql`
[10] `http://s4.semanticscience.org:16036/sparql`

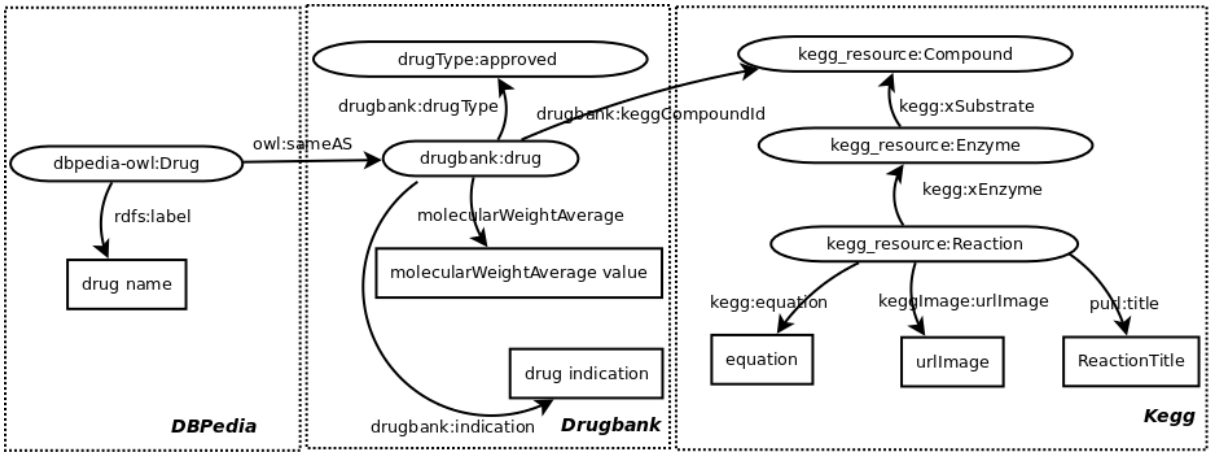

Figure 1: Example Data relation located at DBpedia, Drugbank and Kegg Dataset

utilizing *owl:sameAs*. The *owl:sameAs* link is a shortcut to query data from two dataset without knowing Drugbank and DBpedia schemas.

In this study, we will focus primarily on federation over SPARQL Endpoint infrastructure, as the LOD cloud statistics reports that 68.14% of the RDF repositories are equipped with SPARQL Endpoints. Other infrastructures that use query languages such as RQL[11], RDQL[12], SeRQL[13] are beyond the scope of this study. Aside from giving an overview of querying over SPARQL Endpoints, we will compare existing federation frameworks based on their platform, infrastructure properties, query processing strategies, etc.—to chose any framework for small and large-scale systems. Further, we highlight shortcomings in the current federation frameworks that could open an avenue in the research of federated queries. In addition, we propose several features that should be added for further federation framework development.

We initially present related survey and evaluation of Federation query at Section 2. A real-world use case of to motivate query federation in the Health Care and Life Sciences (HCLS) domain is presented in Section 3. Section 4 introduces the concept of data integration in Linked Data that gives the necessary foundation of infrastructure for querying over Linked Data. We provide an overview of federation architectures and detail the phase of querying over SPARQL Endpoint in Section 5 and in Section 6 introduce existing federation frameworks, supporting either SPARQL 1.0 or 1.1[14]. We also categorize them based on their architecture and querying process and investigate features that should be added in the existing frameworks. Finally, we discover challenges that should be considered in the future development of federation query in Section 8. We conclude our finding in Section 9.

[11] http://139.91.183.30:9090/RDF/RQL/
[12] http://www.w3.org/Submission/RDQL/
[13] http://www.w3.org/2001/sw/wiki/SeRQL
[14] http://www.w3.org/TR/sparql11-query/

2 Related Works

More general investigations w.r.t. querying Linked Data have been performed elsewhere [BGHS12, GS11a, LT10]. [BGHS12] mentioned nine myths and five challenges arising in the Federation over Linked Data. Based on their observation, they suggested to consider Linked Data as a service not as distributed data. [GS11a] explained the Federation query infrastructure, whereas [LT10] focused on the basics of federation query processing strategy.

A number of studies [MVC+12, SHS+12] compare federation frameworks by evaluating their performance. [MVC+12] tests federation frameworks by using FedBech [SGH+11b] in various networking environment and data distribution. Similar to [MVC+12], [SHS+12] conducts an experiment in the FedBench to evaluate federation frameworks on large scale Life Science datasets. In this survey, we investigate and compare more existing Federation over SPARQL Endpoints frameworks based on their strategy such as source selection and execution plan.

3 Motivating Example

The HCLS domain advocated Linked Data from its early days, and at present a considerable portion of the Linked Data cloud is comprised of datasets from Linked Data for Life Sciences (LD4LS) domain [HFDD12]. LD4LS currently comprises multiple datasets from certain HCLS projects, namely *bio2rdf*[15], the Health Care and *Life Sciences Knowledge base*[16] (HCLS Kb), *linkedlifedata*[17], Linked Open Drug Data effort[18] and the *Neurocommons*[19]. These efforts have been derived and are still motivated in biomedical facilities in the recent years, partially caused by the decrease in price for acquiring large datasets such as genomics sequences and the trend towards personalized medicine, pharmacogenomics and integrative bioinformatics, to access and query life sciences data. [HK04] described the high demand of biological datasets integration to help life science researcher. Although the publication of datasets as RDF is a significant milestone to achieve the ability to query these healthcare and other biological datasets, to this date, it is a big deal to enable a query-able Web of HCLS data.

To achieve the ability for assembling queries encompassing multiple graphs hosted at various places, it is therefore critically necessary that vocabularies and ontologies are reused [Pol10]. This can be achieved either by ensuring that the multiple datasets make use of the same vocabularies and namespaces or assemble a federated query over multiple datasets to retrieve a meaningful information. In order to understand need of federation, consider two SPARQL Endpoints DrugBank and Kegg (Figure 1) that publish information regarding drugs and compounds. Both the datasets have different useful information about the same concepts. In order to find the answer of the question *Find the Chemical equations and Reaction titles of reactions related to only those drugs which are approved along with average Molecular Weight*, one should send the query to two SPARQL Endpoints Drugbank and Kegg. Kegg contains two concepts namely Chemical equations and Reaction title whereas the information like average Molecular Weight and "approved drugs" is present at Drugbank endpoint. This complex and related information can be retrieved using Query 1

[15] `http://bio2rdf.org/`
[16] `http://www.w3.org/TR/hcls-kb/`
[17] `http://linkedlifedata.com/`
[18] `http://www.w3.org/wiki/HCLSIG/LODD`
[19] `http://neurocommons.org/page/Main_Page`

Query 1: Data Integration Linked Data Example in Life Science Domain

```
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/>
PREFIX drugType: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugtype/>
PREFIX kegg:<http://bio2rdf.org/kegg_vocabulary:>
PREFIX keggImage:<http://bio2rdf.org/ns/bio2rdf#>
PREFIX purl:<http://purl.org/dc/elements/1.1/>

SELECT distinct ?drug  ?drugtype  ?compound ?molecularWeightAverage ?
    ReactionTitle ?ChemicalEquation

WHERE {
    ?drug drugbank:drugType drugType:approved .
    ?drug drugbank:keggCompoundId ?compound .
    ?drug drugbank:molecularWeightAverage ?molecularWeightAverage.
    ?enzyme kegg:xSubstrate ?compound .
    ?Chemicalreaction kegg:xEnzyme ?enzyme .
    ?Chemicalreaction kegg:equation ?ChemicalEquation .
    ?Chemicalreaction keggImage:urlImage "http://www.genome.jp/Fig/
       reaction_small/R05248.gif".
    ?Chemicalreaction purl:title ?ReactionTitle
}
```

that should be federated to Kegg and Drugbank endpoints.

4 Infrastructure for Querying Linked Data

Based on data source location, the infrastructure for querying Linked Data can be divided into two categories, namely 1) central repository and distributed repositories. Central repository has similar characteristic as of data warehousing in traditional databases, where the data is collected in advance in a single repository before query processing (Figure 2). Sindice[TDO07] is an example of a central repository, which crawls data, indexes it and provides APIs and a SPARQL Endpoint for accessing the data. The efficiency of query time is one advantage because data has already been placed at one location. The single data location in this scenario offers following benefits : no network communication and no source selection required. However, due to the frequently changing data source, the data synchronization could be a problem [UKHP12]. Furthermore, the data storage needs a lot of space in order to keep data in one place. Not only consuming more space, but this approach is resource intensive in regard to processing large scale data.

As opposed to central repository, querying Linked Data in distributed repositories environment does not need crawling data beforehand. We group distributed repositories in two systems: Link Traversal and Federation. Discovering data by following HTTP URIs is the basic idea in the link traversal system. As illustrated in Figure 3, without any data knowledge, relevant data sources are detected during runtime execution [Har11]. Link traversal provides high freshness of the data since the data is directly accessed from data source. The query execution is initially from one single triple pattern as starting point. Determining the starting point is a vital task in this system because it influences the flow process of whole of query execution. The wrong starting point decision

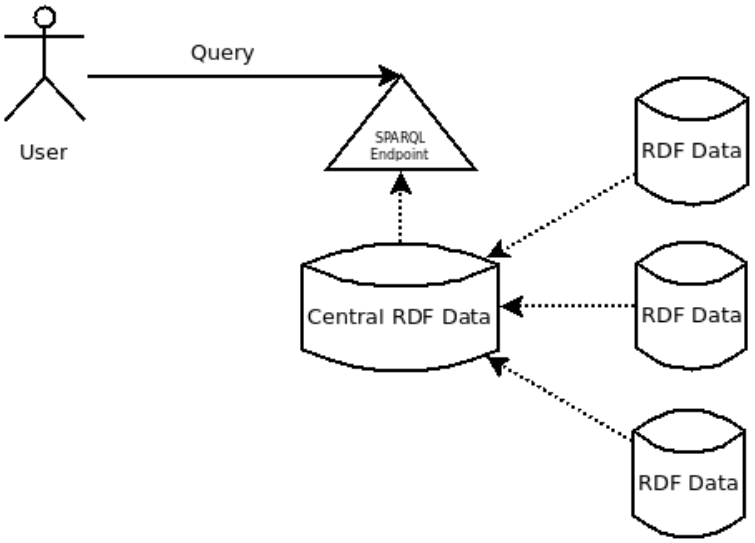

Figure 2: Central Repository

can increase intermediate results, as a result the bandwidth usage goes up. Another drawback of this system is the limitation of type executable query such as query pattern consisting unbound predicate.

Federation uses a query mediator to transform a user query into several sub queries and generates results from the integrated data sources. As the data sources need not be collected in one repository, the data is more up-to-date than central repository's result, but query processing time takes longer than the case of central repository infrastructure. The system only invests little resource such space and time because of no earlier crawling data phase. In contrast to central repository, overhead communication between mediator and data source often occurs in the federation system since source selection is required during query execution. There are two kinds of federation frameworks: federation over single repositories (Figure 4) and federation over SPARQL Endpoints (Figure 5). In the federation over single repositories such as Sesame Sail Federation[20], the federation query interface delivers sub queries through native API of its repository. Nevertheless, not all repositories support this API. Furthermore, the loading data from RDF dump to repository must be done in advance before query transmitted, thus we do not take account this distribution in our observation. The other distributed query type requires SPARQL Endpoint as bridge between federation layer and datasource (we will describe this type of federation in section 6). Most of the federation frameworks are compatible with this type, since RDF store are generally equipped with SPARQL Endpoint. Several mediators sometimes [LWB08, AV11] offer wrapper supporting other data format like CSV and XML. Hence, in this study we only consider to survey federation over SPARQL Endpoints and federation supporting wrapper to access non RDF graph that could be accessed like SPARQL Endpoints.

[20] `http://www.openrdf.org/alibaba.jsp`

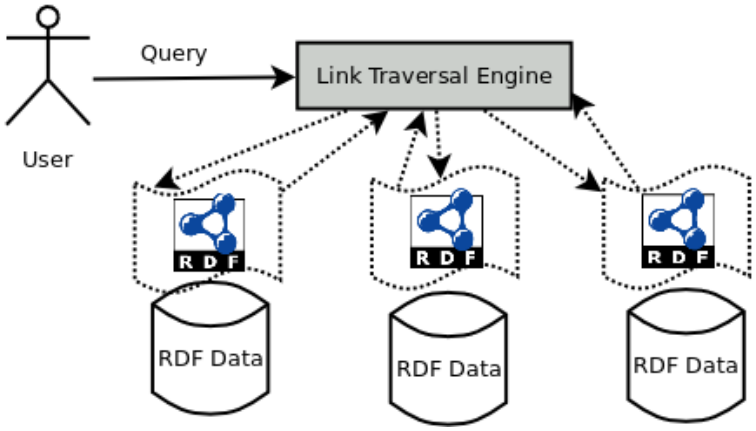

Figure 3: Link Traversal

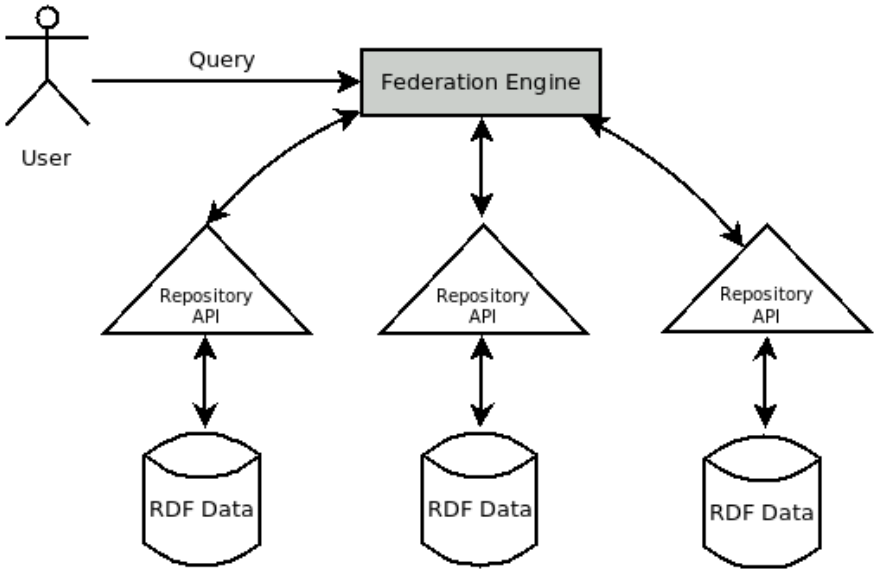

Figure 4: Federation over Single Repositories

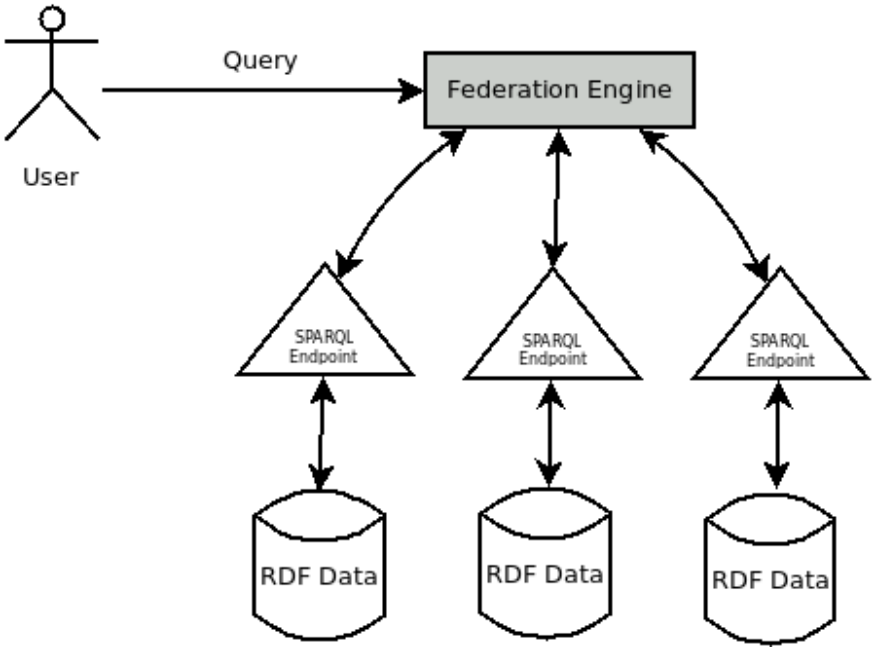

Figure 5: Federation over SPARQL Endpoint

5 Federation over SPARQL Endpoints Basic Approach and Technique

This section provides an overview of architecture and basic mechanisms that can be used in any kind of federation over SPARQL Endpoints framework.

5.1 Architecture of Federation over SPARQL Endpoints

SPARQL 1.1 is designed to tackle limitations of the SPARQL 1.0, including updates operations, aggregates, or federation query support. As of this writing, not all query engines support SPARQL 1.1. Therefore, we discuss the federation frameworks that support either SPARQL 1.0 or SPARQL 1.1. There are three kinds of architecture of federation over SPARQL Endpoints (Figure 6) namely a) framework has capability to execute SPARQL 1.1 query, b) framework accepts SPARQL 1.0, then it rewrites query to SPARQL 1.1 syntax before passing it to SPARQL 1.1 engine and c) framework handles SPARQL 1.0 and processes the query in several phases by interacting with SPARQL 1.0 engine of each SPARQL Endpoints. Those system that has already supported SPARQL 1.1 allow user to execute query federation over SPARQL Endpoints by using *SERVICE* operator (Figure 6.a). The query processor sends each sub query to defined SPARQL Endpoint and join the result from SPARQL Endpoint. Basically, SPARQL 1.0 allows us to query data from remote data sources, however it does not retrieve specified remote SPARQL Endpoints. As described in the Query 2., it only fetches remote graphs or graphs with the name in a local store.

At present, the SPARQL 1.1 is the simplest solution to yield data from multiple sources. The W3C recommendation of SPARQL 1.1 formalizes rule to query in multiple SPARQL Endpoints by

Query 2: Example of Federated SPARQL Query in the SPARQL 1.0

```
SELECT ?drugname ?indication
WHERE {
FROM <http://localhost/dbpedia.rdf>
{
        ?drug a dbpedia-owl:Drug .
        ?drug rdfs:label ?drugname .
        ?drug owl:sameAs ?drugbank .
}
FROM <http://localhost/drugbank.rdf>
{
        ?drugbank drugbank:indication ?indication .
}
}
```

Query 3: Example of Federated SPARQL Query in the SPARQL 1.1

```
SELECT ?drugname ?indication
WHERE {
SERVICE <http://dbpedia.org/sparql>
{
        ?drug a dbpedia-owl:Drug .
        ?drug rdfs:label ?drugname .
        ?drug owl:sameAs ?drugbank .
}
SERVICE <http://www4.wiwiss.fu-berlin.de/drugbank/sparql>
{
?drugbank drugbank:indication ?indication .
}
}
```

using *SERVICE* operator. However, users must have prior knowledge regarding the data location before writing a query because the data location must be mentioned explicitly. As seen in the Query 3, the Drugbank and DBPedia SPARQL Endpoints are mentioned after SERVICE operator to obtain the list of drugs and their associated diseases. In order to assist users in term of data source address, it allows us to define a list of SPARQL Endpoints as data beforehand and attach it as variable in the SPARQL query. Besides *SERVICE*, SPARQL 1.1 also introduces *VALUES* as one of SPARQL Federation extension. It can reduce the intermediate results during query execution by giving constrains from the previous query to the next query.

The lack of knowledge data information is a main problem to execute federation query on single RDF store. Thus, several efforts have been introduced to address that issue (Figure 6.b). The user can write a query blindly without knowing the data location. These federation models can executes Query 3 or Query 2 without a SPARQL Endpoint declared. By removing SERVICE or FROM keywords, those two queries can be replaced by Query 4. These framework architectures provide an interface to translate query from SPARQL 1.0 to SPARQL 1.1 format. The core part of this interface is query rewriting component. After parsing and decomposing the query, this

Query 4: Example of Federation SPARQL Query in the SPARQL 1.0 without SPARQL Endpoint specified

```
SELECT ?drugname ?indication
WHERE {
?drug a dbpedia-owl:Drug .
?drug rdfs:label ?drugname .
?drug owl:sameAs ?drugbank .
?drugbank drugbank:indication ?indication .
}
```

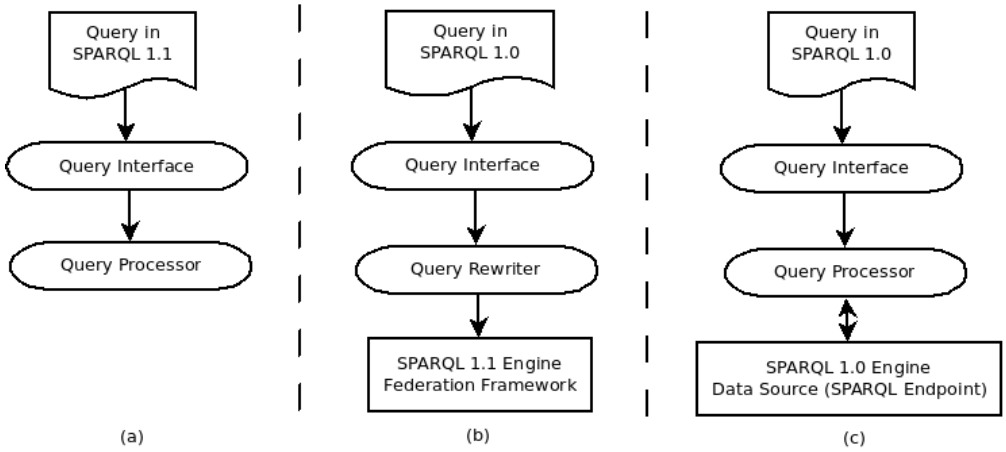

Figure 6: Architecture of Federation over SPARQL Endpoint

component adds destination address of this query by inserting *SERVICE* operators in each sub query. Further on, the result of query rewriter will be executed by internal SPARQL 1.1 processor system.

Since not all current SPARQL Endpoints can handle SPARQL 1.1 query, several systems (Figure 6.c) developed query execution processor to execute federation SPARQL query in SPARQL 1.0 format. The processor has responsibility to manage query processing such as maintain data catalogue, determine relevant sources, plan the query execution and join all results after retrieving data from SPARQL Endpoints.

5.2 Basic Steps of Federation over SPARQL Endpoints

A mediator holds important role to manage an incoming query from user, deliver query to each data source and gives back the result to the user. We explain some important concept around of federation frameworks for executing a query that is depicted at Figure 7. in the following section.

5.2.1 Query parser

In this initial phase, SPARQL query is transformed to internal pattern which can be in the list of

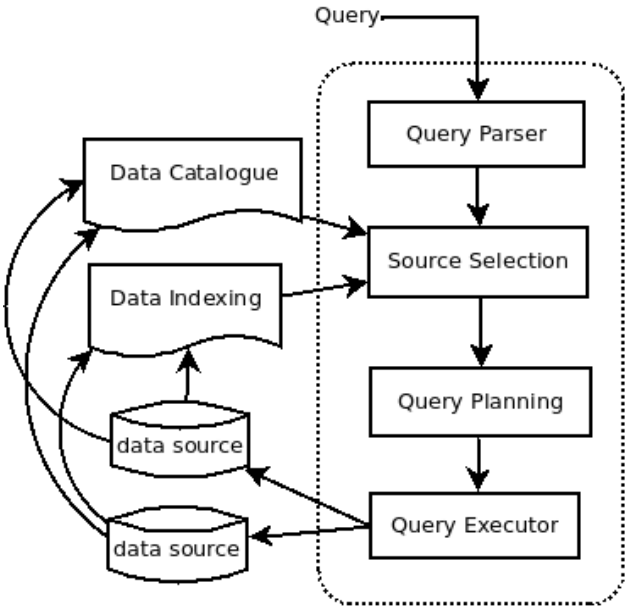

Figure 7: Federated SPARQL Query Process

Basic Graph Pattern (BGP), abstract syntax tree [GS11b], or other formats. BGP is a set of triple patten where one triple consists of subject, predicate and object. One or more variables could be in subject, predicate and object position. The outcome of the parsing step is useful for later steps, particularly in query optimization.

5.2.2 Source Selection

Instead of sending every piece of query to all data sources, the mediator should determine the relevant source of sub query carefully. The simple query might not become a big issue when delivering query to all destinations, however, regardless the capability of source for answering a query, the transmission of complex query with many intermediate results could lead expensive communication cost. Choosing a relevant source for a query could be done in several following methods:

- ASK SPARQL Query
  ASK SPARQL query returns boolean value that decides whether the query can be answered by the SPARQL Endpoint or not. For instance, to solve Query 1, the mediator will send Query 5 to seek sources that can answer sub query *?drug drugbank:molecularWeightAverage ?molecularWeightAverage*. By sending ASK query, the bandwidth usage of query execution can be reduced significantly because a query is only transferred to the Endpoint responding true value. Furthermore, it can detect the changing data easily in runtime. The limitation of ASK is that it is only a binary decision and it can not detect redundancy data among data sources, therefore [HS12] extended the ASK operation during data source selection by including a sketch: estimation number of result and summary of the result. In addition, this

Query 5: Example ASK SPARQL Query to Select Relevant Source

```
PREFIX drugbank: <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/>
ASK {
   ?drug drugbank:molecularWeightAverage ?molecularWeightAverage .
}
```

methods is only suitable for few number of source participated since it takes longer time for waiting each SPARQL Endpoint answering ASK query.

- Data catalogue
  By looking up data catalogue of a dataset, the mediator can predict the suited sources for a query. [LWB08, GS11b] utilize VoID [AH09] as data catalogue. LOD cloud statistic states that 32.2% of LOD datasets provide dataset descriptions expressed in VoID. VoID expresses the metadata of the dataset and its relation with other datasets. It is useful for registering new data source in the SemWIQ [LWB08], whereas it is used to estimate the SPARQL query pattern cardinalities [NM11] in SPLENDID [GS11b]. [LWB08, GS11b] also employs RDFStats [LW09] which is built on SCOVO [HHR+09]. RDFStats comprises instance statistics and histogram of class, property and value type. Both of VoID and RDFStats can be used independently in any case from those frameworks. Service Description [21] describes the data availability from SPARQL Endpoint, data statistic and the restriction of query pattern. [KSB+10] constructs a data catalogue containing list of predicates during setup and querying phase. In general, data catalogue only provides a list of predicate for each source since the number of predicates is less than number of subjects and objects. Based on the catalogue, mediator commonly does pre computing statistic that is needed for query optimization. The data catalogue can be updated during query execution especially for frequently changing data. The freshness of data catalogue has impact on the accuracy of source selection, but it consumes much bandwidth during updating process. Furthermore, the SPARQL query delivered for updating data catalogue is expensive operation which might be refused by SPARQL Endpoint.

- Data Indexing
  According to LOD cloud statistic, 63% of data sources do not expose their data catalogues. In order to overcome the lack of data catalogue, the data indexing process could be done before or during query execution. [HHK+10] introduced indexing method by applying QTrees [HKK+07]. This indexing consists of description of instances and schema that could assist mediator to determine relevant source for a query. In general, the drawback of data indexing is needs more storage than data catalogue as it usually indexes every triple. In order to deal with space problem, [BB10] detects the relevant query sources by obtaining information from either Search Engine such as Sindice[22] or Web Directory after query parsing step. As a result, a summary of ontological prefixes, predicates and classes are indexed for query execution process. However, network latency issue during execution arises because it relies on third party indexing.

[21] http://www.w3.org/TR/sparql11-service-description/
[22] http://sindice.com/

- Caching
  Most of the federation frameworks load statistical information during initialization phase because they want to reduce the bandwidth consumption during runtime. However, the source selection process could be not accurate, if the data is frequently changing. To tackle this shortcoming, the cache during query execution can be employed for later query execution. The information stored at a cache is likely similar to data catalogue. To decrease the communication cost, the mediator does not deliver new query, but the updating statistic data source information based on the result of sub query answers from SPARQL Endpoint.

All aforementioned source selection strategies can be applied to dynamic data such as data stream as long as it is always updated. As updating task consumes much bandwidth, we should have a mechanism that only requests for frequently changing data as proposed by [UKHP12] or only updates data information periodically.

5.2.3 Query Planning and Execution

In the second phase of query processing, the query mediator decomposes the query and builds multiple sub queries according to the result of source selection. One sub query can be delivered to multiple sources if many sources can contribute the answer. Based on selection source stage, mediator usually does matching pattern.

The construction of sub query is also considered before query transmission process. One single triple pattern can match either one source or multiple sources. To reduce repetition of one triple query sending to multiple source, a triple can be grouped with other triples in one sub query. The grouping query triples can also minimize the intermediate join process. [SHH+11] and [AHED12] proposed Exclusive Group scheme to cluster related triple patterns in one sub query. After building sub queries for each data source, the mediator arranges the order of sub queries in various combination of execution plans. Hence, federation framework employs statistical information to compute cost execution of each plan. Later on, the execution plan with the lowest cost will be chosen and executed in the following strategies:

- Nested Loop Join
  Nested loop join is not optimal solution for complex query since every previous scanning results will be joined to next result.

- Bind join
  To improve the nested loop join, bind join, firstly introduced by [HKWY97], passes the intermediate result to become filter for next query. As a result, the transfer cost can be minimized, but the query runtime takes longer because the query mediator have to wait for the complete answer of previous query.

- Hash join
  Hash join is implemented in [GS11b] which joins all intermediate result locally after submitting sub queries in parallel. This join could boost the runtime performance for small intermediate result. However, the transmission cost will be higher if the intermediate result is large.

Table I: The Existing Frameworks Support SPARQL 1.1 Federation Extension.

Framework	Platform	SERVICE	BINDINGS	VALUES
ARQ	Jena	✓	✗	✓
SPARQL-FED	Virtuoso	✓	✗	✓
Sesame	Sesame	✓	✓	✓
SPARQL-DQP	OGSA-DAI and OGSA-DQP	✓	✓	✗

6 The Existing Federation over SPARQL Endpoints Frameworks

This section presents the insight on existing federation over SPARQL Endpoint based on their architectures. To give better explanation, they will be classified based on their features. Ultimately, we propose several features that should be considered for next development.

6.1 The Existing Federation over SPARQL Endpoints Frameworks Based on Their Architecture

As described in the section 5.1, three federation architectures categories have been developed recently. We explain the existing federation based on those categories in this section.

6.1.1 Frameworks Support SPARQL 1.1 Federation Extension

As of this writing, several RDF store systems have been able to process federation query, but not all of them support *VALUES* keyword. Instead of handling *VALUES*, a number of frameworks has supported *BINDING* which is also addressed to reduce the size of intermediate results. The list of existing frameworks supporting SPARQL 1.1 presents at Table III.

ARQ

ARQ[23], a query engine processor for Jena, has supported federated query by providing *SERVICE* and *VALUES* operator. ARQ implements nested loop join to gather retrieved result from multiple SPARQL Endpoints. In term of security, the credential value to connect ARQ service must be initialized in the pre-configuration[24].

Sesame

Previously, Sesame already supported federation SPARQL query by using SAIL AliBaba extension[25] at 2009, but it can not execute SPARQL 1.1. Instead, it integrates multiple datasets into a virtual single repository to execute federated query in SPARQL 1.0. It can execute federation SPARQL query either RDF dump or SPARQL Endpoint by using its API. The data source must be registered in advance during setup phase. The simple configuration file only containing the list of SPARQL Endpoint address can cause poor performance since

[23] http://jena.apache.org/documentation/query/index.html
[24] http://jena.hpl.hp.com/Service#
[25] http://www.openrdf.org/doc/alibaba/2.0-alpha2/alibaba-sail-federation/index.html

it sends query to all data source without source selection. In order to optimize the query execution, it offers additional features in the configuration file namely predicate and subject prefixes owned by one dataset. According to the configuration, it can do prefix matching to predict the relevant source for a sub query. The join ordering is decided by calculating the size of basic graph pattern. The new version of sesame (2.7)[26] is able to handle SPARQL 1.1 which provides Federation extension features including *SERVICE* and *VALUES* operator.

SPARQL-FED

Virtuoso 6.1 allows to execute SPARQL queries to remote SPARQL Endpoint through SPARQL-FED[27]. The remote SPARQL Endpoint must be declared after SERVICE operator.

SPARQL-DQP

It is built on top of the OGSA-DAI [AHH+07] and OGSA-DQP [LMH+09] infrastructures. It transforms incoming SPARQL query to SQL, as it implements SQL optimization techniques to generate and optimize query plans. The optimization strategy is based on OGSA-DQP algorithm which does not need any statistic information from data sources. No SPARQL Endpoint registration is required because the SPARQL Endpoint must be written in query. The OGSA-DAI manages a parallel hash join algorithm to reorder query execution plan.

6.1.2 Frameworks Supports Federation over SPARQL Endpoint, build on top of SPARQL 1.0

In order to overcome data location knowledge, several federation frameworks have been developed recently without specifying SPARQL Endpoint address in the query. The federation framework acts as mediator [HSTW11] that transfers SPARQL query from user to multiple data source either RDF repository or SPARQL Endpoints. Before delivering query to destination source, it breaks down a query into sub queries and selects the destination of each sub query. In the end, the mediator must join the retrieved result from the SPARQL Endpoints. Following are overview of the current of federation frameworks and summarized in Table II.

DARQ

DARQ (Distributed ARQ) [QL08] is an extension of ARQ which provides transparent query to access multiple distributed endpoint by adopting query mediator. The service description which consists of data description and statistical information has to declare in advance before query processing to decide where a sub query should go. According to the list of predicates in the service description, it re-writes the query, creates sub query and design the query planning execution. The query planning is based on estimated cardinality cost. DARQ implements two join strategies : Nested Loop Join and Bind Join.

Splendid

Splendid [GS11b] extends Sesame which employs VoID as data catalogue. The VoID of dataset is loaded when the system started then ASK SPARQL query is submitted to each dataset for verification. Once the query is arrived, the system builds sub queries and join order for optimization. Based on the statistical information, the bushy tree execution plan

[26] `http://www.openrdf.org/index.jsp`
[27] `http://www.openlinksw.com/dataspace/dav/wiki/Main/VirtSparqlCxml`

is generated by using dynamic programming [SAC+79]. Similar to DARQ, it computes join cost based on cardinality estimation. It provides two join types: hash join and bind join to merge result locally.

FedX

FedX [SHH+11] is also developed on top of the Sesame framework. It is able to run queries over either Sesame repositories or SPARQL Endpoints. During initial phase, it loads the list of data sources without its statistical information. The source selection is done by sending SPARQL ASK queries. The result of a SPARQL ASK query is stored in a cache to reduce communication for successive query. Intermediate result size is minimized by a rule based join optimizer according to cost estimation. It implements Exclusive Groups to cluster related patterns for one relevant data source. Beside grouping patterns, it also groups related mapping by using single SPARQL UNION query. Those strategies can decrease the number of query transmission and eventually, it reduces the size of intermediate results. As complementary, it came with Information Workbench for demonstrating the federated approach to query processing with FedX.

ADERIS

ADERIS (Adaptive Distributed Endpoint RDF Integration System) [KSB+10] fetches the list of predicates provided by data source during setup stage. The predicate list can be used to decide destination source for each sub query pattern. During query execution, it constructs predicate tables to be added in query plan. One predicate table belongs to one sub query pattern. The predicate table consists of two columns : subject and object which is filled from intermediate results. Once two predicate tables have completed, the local joining will be started by using nested loop join algorithm. The predicate table will be deleted after query is processed. ADERIS is suitable for data source who does not expose data catalogue, but it only handles limited query patterns such as UNION and OPTIONAL. The simple GUI for configuration and query execution are provided by ADERIS.

Avalanche

Avalanche [BB10] does not maintenance the data source registration as its data source participant depends on third party such as search engine and web directory. Apart from that, it also stores set of prefixes and schemas to special endpoints. The statistic of data source is always up to date since it always requests the related data source statistic to search engine after query parsing. To detect the data source that contributes to answer a sub query, it calculates the cardinality of each unbound variables. The combinations of sub queries are constructed by utilizing best first search approach. All sub queries are executed in parallel process. To reduce the query response time, it only retrieves first K results from SPARQL Endpoint.

GDS

Graph Distributed SPARQL (GDS) [WTD11a] overcomes the limitation of their previous work [WTD11b] which can not handle multiple graphs. It is developed on top of Jena platform by implementing Minimum Spanning Tree (MST) algorithm and enhancing BGP representation. Based on Service description, MST graph is generated by exploiting Kruskal algorithm which aims to estimates the minimum set of triple patterns evaluation and execution order.

Table II: The Existing Frameworks Supports Federation over SPARQL Endpoints without reformulating query to SPARQL 1.1.

Framework	Catalogue	Platform	Source Selection	Cache	Query Execution	Source Tracking	GUI
DARQ	Service Description	Jena	Statistic of Predicate	✓	Bind Join or Nested Loop Join	Static	✗
ADERIS	Predicate List during setup phase	✗	Predicate List	✗	Nested Loop Join	Static	✓
FedX	✗	Sesame	ASK	✓	Bind Join parallelization	Dynamic	✓
Splendid	VoID	Sesame	Statistic + ASK	✗	Bind Join or Hash Join	Static	✗
GDS	Service Description	Jena	Statistic of Predicate	✓	Bind Join or Semi Join	Dynamic	✗
Avalanche	Search Engine	Avalanche	Statistic of predicates and ontologies	✓	Bind join	Dynamic	✗
Distributed SPARQL	✗	Sesame	✗	✗	Bind join	✗	✗

The query planning execution can be done by either semi join or bind join which is assisted by cache to reduce traffic cost.

Distributed SPARQL

In contrast to the above frameworks, users must declare the SPARQL Endpoint explicitly in the SPARQL query at Distributed SPARQL [ZS08]. Since it is developed for SPARQL 1.0 user, the SPARQL Endpoint address is mentioned after *FROM NAMED* clause. Consequently, this framework does not require any data catalogue to execute a query. As part of Networked Graphs [SS08], it is also built on the top of Sesame. To minimize number of transmission query during execution, it applies distributed semi join in the query planning.

6.1.3 Frameworks Supports Federation over SPARQL Endpoint, build on top of SPARQL 1.1

A number of frameworks were developed to accepts SPARQL query federation in SPARQL 1.0 format, but they are built on top of SPARQL query engine that support SPARQL 1.1 (Table III.)

ANAPSID

ANAPSID [AV11] is a framework to manage query execution with respect to data availability and runtime condition for SPARQL 1.1 federation. It enhances XJoin [UF00] operator and combines it with Symmetric Hash Join [DIR07]. Both of them are non blocking operator that save the retrieved result to hash table. Similar to others frameworks, it also has data catalogue that contains list of predicates. Additionally, execution time-out information of SPARQL Endpoint is added in the data catalogue. Therefore, the data catalogue is updated on the fly. Apart from updating data catalogue, it also updates the execution plan at runtime. The Defender [MVA12a, MVA12b] in ANAPSID has the purpose to split up the query from SPARQL 1.0 format to SPARQL 1.1 format. Not only splitting up the query, Defender also composes related sub query in the same group by exploiting bushy tree.

SemWIQ

SemWIQ is another system building on top of ARQ and part of the Grid-enabled Semantic Data Access Middleware (G-SDAM). It provides a specific wrapper to allows data source without equipped SPARQL Endpoint connected. The query federation relies on data summaries in RDFStats and SDV[28]. RDFStats is always up-to-date statistic information since the monitoring component periodically collects information at runtime and stores it into a cache. As the RDFstats also covers histogram String, Blank Node etc, it is more beneficial for SemWIQ to be able to execute any kind of query pattern. SDV is based on VoID which is useful for data source registration. The query is parsed by Jena SPARQL processor ARQ before optimization process. SemWIQ applies several query optimisation methods based on statistic cost estimation such as push-down of filter expressions, push down of optional group patterns, push-down of joins and join and union reordering. During optimization, the federator component inserts SERVICE keyword and SPARQL Endpoint for each sub query.

WoDQA

WoDQA (Web of Data Query Analyzer) [AHED12] also uses ARQ as a query processor. The source selection is done by analysing metadata in the VoID stores such as CKAN[29] and VoIDStore[30]. The source observation is based on Internationalized Resource Identifier (IRI), linking predicate and shared variables. It does not exploit any statistic information in the VoID of each dataset, but it only compares IRI or linking predicate to subject, predicate and object. The same variables in the same position are grouped in one sub query. After detecting relevant sources for each subquery, the SERVICE keyword is appended following with SPARQL Endpoint address.

7 Desired Features

We have seen existing federation SPARQL frameworks along with their behaviours and properties. Based on our summary and experience, we suggest several features that could be added into their framework.

[28] `http://purl.org/semwiq/mediator/sdv#`
[29] http://ckan.net/
[30] http://void.rkbexplorer.com/

Table III: The Existing Frameworks Supports Federation over SPARQL Endpoints, Reformulate query to SPARQL 1.1.

Framework	Catalogue	Platform	Source Selection	Cache	Query Execution	Source Tracking	GUI
SemWIQ	RdfStats+VoID	Jena	Statistic + Service	✓	Bind Join	Dynamic	✓
Anapsid	Predicate List and Endpoint status	Anapsid	Predicate List	✗	Symmetric Hash Join and XJoin	Dynamic	✓
WoDQA	VoID Stores	Jena	List of predicates and ontologies	✗	✗	Dynamic	✓

Query 6: Example of Link Predicate Problem

```
SELECT ?drug ?compoundname
WHERE {
?drug drugbank:keggCompoundID ?compound .
?compound rdfs:label ?compoundname .
}
```

- Hybrid data catalogue
  As described in the section 6, the data source registration could be done by the mediator as well as third party such as search engine. In term of querying in the Linked Open Data, the data source registered should be not limited. The framework could combine static and dynamic data source registration where the data source in the static registration is given higher priority than data source in the dynamic registration before delivering a query.

- Link Predicate Awareness

  Link predicate has ability to connect one entity to another entity in a different datasource. For instance, Figure 1 shows the `drugbank:keggCompoundID` link joining the entity `drugbank:drugs` in the Drugbank dataset with the class `kegg_resource:Compound` in the Kegg dataset and `owl:sameAS` link joining the entity `dbpedia-owl:Drug` in the DBpedia dataset with the entity `drugbank:drugs` in Drugbank. Assuming the link predicate connects two datasets, we should avoid to deliver the same destination of query patten containing the link predicate. In the case of Query 6, pattern *?compound rdfs:label ?compoundname* should not be send to Drugbank dataset as `drugbank:keggCompoundID` is a link predicate, even predicate *rdfs:label* occurs in all datasets.

8 Challenges

According to our investigation in Section 6 we note several challenges to be addressed in the future of federation framework development. Federation over SPARQL Endpoints has become actively

developed over the last four years, in particular source selection part. This field area is still infancy which faces several challenges that need to be tackled :

- Data Access and Security
  The data source and mediator are usually located in different locations, therefore the secure communication process among mediator and data source should be concerned. In addition, updating data should be supported in the SPARQL 1.1. Several SPARQL Endpoints provide authentication feature in order to restrict query access for limited user. However, the unauthorized interception between mediator and data source have not been undertaken by any federation frameworks yet. The public key cryptography could be implemented in the federation frameworks where mediator and data source share public and private key for data encryption during interaction.

- Data Allocation
  Since several RDF stores crawl data from other data source, the data redundancy could not be avoided in the Linked Open Data Cloud. Consequently, the federation over SPARQL Endpoint framework detects data source is located in multiple location. This such condition could increase communication cost during selection source and query execution stage particularly for federation system employing data statistic from third party. Furthermore, the redundancy data could increase intermediate results as more data duplication from multiple source. On the one hand, using popular vocabulary allows user to query easily, but on the other hand, the source prediction for a query will be a hard task. As pointed out by [RH12] when popular entities and vocabularies are distributed over multiple data sources, the performance of federation query is getting worse.

- Data Freshness
  The freshness is one most important measurement in the data integration because each data source might has different freshness value. Having up to date data catalogue is a must in the federation framework to achieve high freshness value. Inaccurate results could arise from inaccurate data catalogue. Nevertheless, updating data catalogue is a costly operation in term of query execution and traffic between data source and frameworks. Apart from data catalogue being static, the freshness could not be obtained when the high network latency occurs during communication process.

- Benchmark
  To date, benchmarks are generally proposed for single RDF stores such as LUBM [GPH05], BSBM [BS09], and SP2Bench [SHLP08]. Hence, they are not suitable for distributed infrastructure. FedBench [SGH+11a] is the only benchmark proposed for federated query which evaluated the federated query infrastructure performance including loading time and querying time. Those performance metrics are lack to evaluate federation framework. The federation framework benchmark should take into account several performance measurements from traditional distributed database such as query throughput. In addition, several metrics particularly occurring in the federation framework should be considered. For instance, the size of intermediate result, number of request, amount of data sent, etc. Apart from performance metric, due to heterogeneous data in the federation query, the evaluation of data quality become important measurement namely freshness, consistency, completeness and accuracy.

[MVA12a] added two more FedBench measurements, namely Endpoint Selection time and completeness. Furthermore, it evaluated performance federation framework in various environment. Since the FedBench has static dataset and query set, it is difficult task to evaluate framework for other dataset. To address this problem, SPARQL Linked Open Data Query Generator (SPLODGE) [GTS12] generates random query set for specified dataset. The query set generation is based on dataset characteristic that is obtained from its predicate statistic. Beside dataset characteristic, it also considers the query structure and complexity such as number of join, the query shape, etc to produce the query set.

- Overlapping Terminologies
  The data is generated, presented and published using numerous expressions, namespaces, vocabularies and terminologies, that significantly contain duplicate or complementary data [Qua06, BBDR+11]. As an example, there are multiple datasets in the LSLOD describing the concept *Molecule*- in Bio2RDFs kegg dataset, it is represented using *kegg:Compound* whereas in chebi, these are identified as *chebi:Compound* and in BioPax they are denoted as *biopax-level3.owl#SmallMolecule*, i.e, using different vocabularies and terminologies to explain the similar or related concepts [HFDD12] . Moreover different dataset contains different fractions of data or predicates about the same entities e.g: Chebi dataset contains data regarding the mass or the charge of a Molecule whereas Kegg dataset explains Molecules interaction with biological entities such as Proteins. Conceptual overlap and different datasets share data about the same entities in LD4LS can be seen in the Figure 8 . Therefore, the mapping rules among heterogeneous schemas is highly required in federation query. This task could be done by having global schema catalogue that maps related concepts or properties and generating more links among related entities.

- Provenance
  As more than one datasets involves in Federated SPARQL Query, the origin of data result will be prominence. Apart from number of sources, data redundancy often occurs in the Federation SPARQL query, particularly in Federation over Linked Open Data. It is because several publishers expose the same dataset. For example, Sindice contains DBpedia data: while a user is requesting DBpedia data, the DBpedia and Sindice SPARQL Endpoints are able to answer that query. The redundant result can not be avoided by Federation Framework using third party catalogue. Hence, the data provenance is important factor in the Federation over SPARQL Endpoint. [Har12] explains a notable provenance implementation in the Federation System called OPENPHACTS[31]. In order to tackle provenance issue, OPENPHACTS utilizes a Nanopublication [GGV10] which supports provenance, annotation, attribution and citation.

9 Conclusion

Federation query over SPARQL Endpoints made a significant progress in the recent years. Although a number of federation frameworks have already been developed, the field is still relatively far from maturity. Based on our experience with the existing federation frameworks, the frameworks mostly focus on source selection and join optimization during query execution.

[31] `http://www.openphacts.org/`

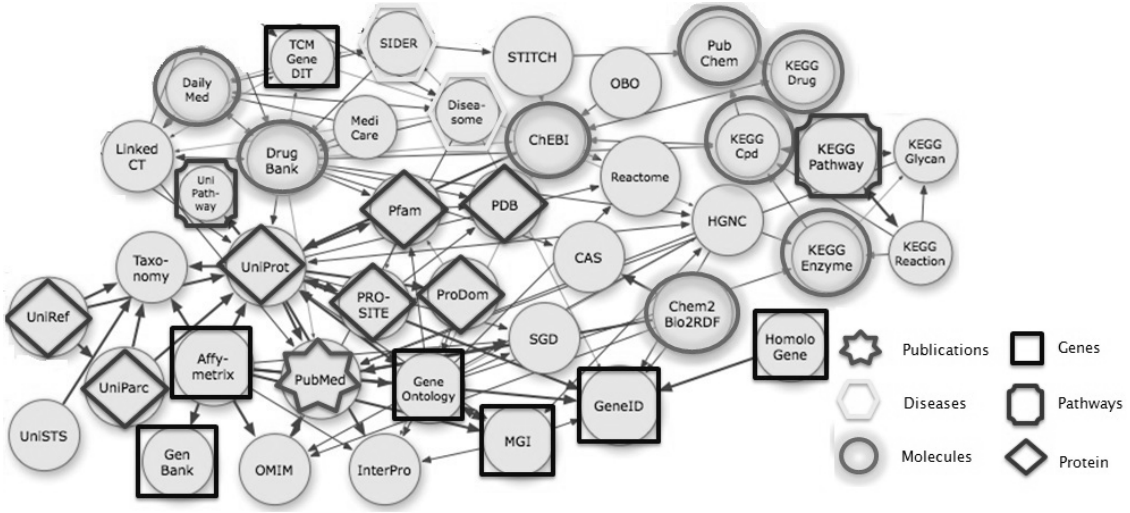

Figure 8: Different Life Science Datasets talks about same concepts

In this work, we have presented a list of federation frameworks over SPARQL Endpoints along with their features. According to this list, the user can have considerations to choose the suitable federation framework for their case. We have classified those framework into three categories: i) framework interprets SPARQL 1.1 query to execute federation SPARQL query covering *VALUES* and *SERVICE* operator; ii) framework handles SPARQL 1.0 query and has responsibility to find relevant source for a query and join incoming result from SPARQL Endpoints; and iii) framework accepts SPARQL 1.0 and translate the incoming query to SPARQL 1.1 format.

Based on the current generation of federation frameworks surveyed in this paper, it still requires further improvements to make frameworks more effective in a broader range of applications. We suggested several features that could be included in the future developments. Finally, we point out challenges for future research directions.

References

[AH09] Keith Alexander and Michael Hausenblas. Describing linked datasets - on the design and usage of void, the vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09*, 2009.

[AHED12] Ziya Akar, Tayfun Gkmen Hala, Erdem Eser Ekinci, and Ouz Dikenelli. Querying the web of interlinked datasets using void descriptions. In *Linked Data on the Web (LDOW2012)*, 2012.

[AHH+07] M. Antonioletti, N. P. Chue Hong, A. C. Hume, M. Jackson, K. Karasavvas, A. Krause, J. M. Schopf, M. P. Atkinson, B. Dobrzelecki, M. Illingworth, N. McDonnell, M. Parsons, and E. Theocharopoulous. Ogsa-dai 3.0 - the what's and whys. In *UK e-Science All Hands Meeting*, 2007.

[AV11] Maribel Acosta and Maria-Esther Vidal. Evaluating adaptive query processing techniques for federations of sparql endpoints. 10th International Semantic Web Conference (ISWC) Demo Session, November 2011.

[BB10] Cosmin Basca and Abraham Bernstein. *Avalanche: Putting the Spirit of the Web back into Semantic Web Querying*. 2010.

[BBDR+11] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, J. Ainsworth, J. Bhagat, P. Couch, D. Cruickshank, M. Delderfield, I. Dunlop, et al. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 2011.

[BGHS12] Heiko Betz, Francis Gropengießer, Katja Hose, and Kai-Uwe Sattler. Learning from the history of distributed query processing - a heretic view on linked data management. In *Proceedings of the 3rd Consuming Linked Data Workshop (COLD 2012)*, 2012.

[BS09] Christian Bizer and Andreas Schultz. The berlin sparql benchmark. *International Journal On Semantic Web and Information Systems*, 2009.

[DIR07] Amol Deshpande, Zachary Ives, and Vijayshankar Raman. Adaptive query processing. *Found. Trends databases*, 1(1):1–140, January 2007.

[FCOO12] A. Freitas, E. Curry, J.G. Oliveira, and S. O'Riain. Querying heterogeneous datasets on the linked data web: Challenges, approaches, and trends. *Internet Computing, IEEE*, 16(1):24–33, 2012.

[GGV10] Paul Groth, Andrew Gibson, and Jan Velterop. The anatomy of a nanopublication. *Inf. Serv. Use*, 30(1-2):51–56, January 2010.

[GPH05] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158 – 182, 2005. Selcted Papers from the International Semantic Web Conference, 2004 - ISWC, 2004.

[GS11a] Olaf Görlitz and Steffen Staab. *Federated Data Management and Query Optimization for Linked Open Data*. Web Data Management Trails, Spinger, 2011.

[GS11b] Olaf Görlitz and Steffen Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*, Bonn, Germany, 2011.

[GTS12] Olaf Görlitz, Matthias Thimm, and Steffen Staab. Splodge: Systematic generation of sparql benchmark queries for linked open data. In *International Semantic Web Conference (1)*, pages 116–132, 2012.

[Har11] Olaf Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 154–169, Berlin, Heidelberg, 2011. Springer-Verlag.

[Har12] Lee Harland. Open phacts: A semantic knowledge infrastructure for public and commercial drug discovery research. In Annette ten Teije, Johanna Vlker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez, editors, *EKAW*, volume 7603 of *Lecture Notes in Computer Science*, pages 1–7. Springer, 2012.

[HB11] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[HFDD12] Ali Hasnain, Ronan Fox, Stefan Decker, and Helena. F Deus. Cataloguing and Linking Life Sciences LOD Cloud. In *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012), OEDW 2012*, 2012.

[HHK+10] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. Data summaries for on-demand queries over linked data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 411–420, New York, NY, USA, 2010. ACM.

[HHR+09] Michael Hausenblas, Wolfgang Halb, Yves Raimond, Lee Feigenbaum, and Danny Ayers. Scovo: Using statistics on the web of data. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 708–722, Berlin, Heidelberg, 2009. Springer-Verlag.

[HK04] Thomas Hernandez and Subbarao Kambhampati. Integration of biological sources: current systems and challenges ahead. *SIGMOD Rec.*, 33(3):51–60, September 2004.

[HKK+07] Katja Hose, Marcel Karnstedt, Anke Koch, Kai-Uwe Sattler, and Daniel Zinn. Processing rank-aware queries in p2p systems. In *Proceedings of the 2005/2006 international conference on Databases, information systems, and peer-to-peer computing*, DBISP2P'05/06, pages 171–178, Berlin, Heidelberg, 2007. Springer-Verlag.

[HKWY97] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 276–285, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[HS12] Katja Hose and Ralf Schenkel. Towards benefit-based rdf source selection for sparql queries. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, SWIM '12, pages 2:1–2:8, New York, NY, USA, 2012. ACM.

[HSTW11] Katja Hose, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. Database foundations for scalable rdf processing. In Axel Polleres, Claudia dAmato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data*, volume 6848 of *Lecture Notes in Computer Science*, pages 202–249. Springer Berlin / Heidelberg, 2011.

[JIB10] Anja Jentzsch, Robert Isele, and Christian Bizer. Silk - generating rdf links while publishing or consuming linked data. In *Poster at the International Semantic Web Conference (ISWC2010), Shanghai*, 2010.

[KSB+10] Shinji Kikuchi, Shelly Sachdeva, Subhash Bhalla, Steven Lynden, Isao Kojima, Akiyoshi Matono, and Yusuke Tanimura. Adaptive integration of distributed semantic web data. *Databases in Networked Information Systems*, 5999:174–193, 2010.

[LMH+09] Steven Lynden, Arijit Mukherjee, Alastair C. Hume, Alvaro A. A. Fernandes, Norman W. Paton, Rizos Sakellariou, and Paul Watson. The design and implementation of ogsa-dqp: A service-based distributed query processor. *Future Gener. Comput. Syst.*, 25(3):224–236, March 2009.

[LT10] Günter Ladwig and Thanh Tran. Linked data query processing strategies. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I*, ISWC'10, pages 453–469, Berlin, Heidelberg, 2010. Springer-Verlag.

[LW09] Andreas Langegger and Wolfram Woss. Rdfstats - an extensible rdf statistics generator and library. In *Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application*, DEXA '09, pages 79–83, Washington, DC, USA, 2009. IEEE Computer Society.

[LWB08] Andreas Langegger, Wolfram Wöß, and Martin Blöchl. A semantic web middleware for virtual data integration on the web. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 493–507, Berlin, Heidelberg, 2008. Springer-Verlag.

[MGSS10] Ian Millard, Hugh Glaser, Manuel Salvadores, and Nigel Shadbolt. Consuming multiple linked data sources: Challenges and experiences. In *COLD*, 2010.

[MVA12a] Gabriela Montoya, Maria-Esther Vidal, and Maribel Acosta. Defender: a decomposer for queries against federations of endpoints. 9th Extended Semantic Web Conference (ESWC) Demo Session, Mai 2012.

[MVA12b] Gabriela Montoya, Maria-Esther Vidal, and Maribel Acosta. A heuristic-based approach for planning federated sparql queries. In *Proceedings of the 3rd Consuming Linked Data Workshop (COLD 2012)*, 2012.

[MVC+12] Gabriela Montoya, Maria-Esther Vidal, Óscar Corcho, Edna Ruckhaus, and Carlos Buil Aranda. Benchmarking federated sparql query engines: Are existing testbeds enough? In *International Semantic Web Conference (2)*, pages 313–324, 2012.

[NA11] Axel-cyrille Ngonga Ngomo and Sren Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, volume 15, pages 2312–2317, 2011.

[NM11] T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 984 –994, april 2011.

[Pol10] A. Polleres. Semantic web technologies: From theory to standards. In *21st National Conference on Artificial Intelligence and Cognitive Science, NUI Galway*, 2010.

[QL08] Bastian Quilitz and Ulf Leser. Querying distributed rdf data sources with sparql. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 524–538, Berlin, Heidelberg, 2008. Springer-Verlag.

[Qua06] J. Quackenbush. Standardizing the standards. *Molecular systems biology*, 2(1), 2006.

[RH12] Nur Aini Rakhmawati and Michael Hausenblas. On the impact of data distribution in federated sparql queries. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 255 –260, sept. 2012.

[SAC+79] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, SIGMOD '79, pages 23–34, New York, NY, USA, 1979. ACM.

[SGH+11a] Michael Schmidt, Olaf Görlitz, Peter Haase, Günter Ladwig, Andreas Schwarte, and Thanh Tran. Fedbench: A benchmark suite for federated semantic data query processing. In *International Semantic Web Conference (1)*, pages 585–600, 2011.

[SGH+11b] Michael Schmidt, Olaf Grlitz, Peter Haase, Gnter Ladwig, Andreas Schwarte, and Thanh Tran. Fedbench: A benchmark suite for federated semantic data query processing. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 585–600. Springer, 2011.

[SHH+11] Andreas Schwarte, Peter Haase, Katja Hoose, Ralf Schenkel, and Michael Schmidt. Fedx: A federation layer for distributed query processing on linked open data. In *ESWC*, 2011.

[SHLP08] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. Sp2bench: A sparql performance benchmark. *CoRR*, abs/0806.4627, 2008.

[SHS+12] Andreas Schwarte, Peter Haase, Michael Schmidt, Katja Hose, and Ralf Schenkel. An experience report of large scale federations. *CoRR*, abs/1210.5403, 2012.

[SMI+] Andreas Schultz, Andrea Matteini, Robert Isele, Pablo N. Mendes, Christian Bizer, and Christian Becker. LDIF - A Framework for Large-Scale Linked Data Integration. In *21st International World Wide Web Conference (WWW2012), Developers Track*, April.

[SS08] Simon Schenk and Steffen Staab. Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 585–594, New York, NY, USA, 2008. ACM.

[TDO07] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: weaving the open linked data. In *Proceedings of the 6th international The semantic web and 2nd Asian*

conference on Asian semantic web conference, ISWC'07/ASWC'07, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag.

[UF00] Tolga Urhan and Michael J. Franklin. XJoin: A Reactively-Scheduled Pipelined Join Operator. *IEEE Data Engineering Bulletin*, 23(2):27–33, 2000.

[UKHP12] Jrgen Umbrich, Marcel Karnstedt, Aidan Hogan, and JosianeXavier Parreira. Hybrid sparql queries: Fresh vs. fast results. In Philippe Cudr-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jrme Euzenat, Manfred Hauswirth, JosianeXavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web ISWC 2012*, Lecture Notes in Computer Science, pages 608–624. Springer Berlin Heidelberg, 2012.

[WTD11a] X. Wang, T. Tiropanis, and H. Davis. Querying the web of data with graph theory-based techniques. 2011.

[WTD11b] Xin Wang, Thanassis Tiropanis, and Hugh C. Davis. Evaluating graph traversal algorithms for distributed sparql query optimization. In Jeff Z. Pan, Huajun Chen, Hong-Gee Kim, Juanzi Li, Zhe Wu, Ian Horrocks, Riichiro Mizoguchi, and Zhaohui Wu, editors, *JIST*, volume 7185 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2011.

[ZS08] Jan Zemánek and Simon Schenk. Optimizing sparql queries over disparate rdf data sources through distributed semi-joins. In *International Semantic Web Conference (Posters & Demos)*, 2008.