Querying Web Polystores

Yasar Khan*, Antoine Zimmermann†, AlokKumar Jha*, Dietrich Rebholz-Schuhmann* and Ratnesh Sahay*
* Insight Centre for Data Analytics, National University of Ireland, Galway
Email: {yasar.khan, alok.kumar, rebholz, ratnesh.sahay}@insight-centre.org
† CNRS, Laboratoire Hubert Curien, Univ Lyon, MINES Saint-Étienne, France
Email: antoine.zimmermann@emse.fr

***Abstract*—The database, semantic web, and linked data communities have proposed solutions that federate queries over multiple data sources using a single data model. Nowadays, the data retrieval requirements originating from versatile and broad domains like healthcare and life sciences (HCLS) are changing this conventional trend – of federating query over a single data model – primarily due to the simultaneous use of different data models (CSV, JSON, RDB, RDF, XML, etc.) in a real-life scenario. It's now impractical to assume that the variety (graph, key-value, stream, text, table, tree, etc.) of high volume data residing in specialised storage engines will first be converted to a common data model, stored in a general-purpose data storage engine, and finally be queried over the Web. Nevertheless, in this era where genomics datasets are growing from petascale to exascale, it is now important to exploit such vast domain resources in their native data models. The key approach is to query the vast data resources from their native data models and specialised storage engines. In this paper, we propose a Web-based query federation mechanism – called PolyWeb – that unifies query answering over multiple native data models (CSV, RDB, and RDF). We demonstrate PolyWeb on a cancer genomics use-case where it is often the case that a description of biological and chemical entities (e.g., gene, disease, drug, pathways) span across multiple data models. In order to assess the benefits and limitations of evaluating queries over native data models, we evaluate PolyWeb with state-of-the-art query federation engine in terms of result completeness, source selection, and overall query execution time.**

***Index Terms*—Databases, Query Federation, Query Planning, Cancer Genomics, Semantic Web, Linked Data**

I. Introduction

The database experts have predicted the demise of "One Size Fits All" approach used in the data retrieval and management solutions [1]. This prediction is now evident, as in the last couple of years several data models (*e.g.*, text, CSV, Graph, JSON, RDB, RDF, XML) and storage engines are proliferating with overlapping requirements, use-cases, and user-bases. This is particularly true in complex domains like healthcare and life sciences (HCLS) where the organisations are facing a major shift in the data retrieval requirements; a simultaneous use of different data storage engines, data models, and supporting querying mechanism is needed to retrieve data from different interacting facilities (clinical measurement, medical history, laboratory test, demographics, etc.) [2].

On the same challenge of combining and querying data from several repositories, the central idea of Linked Data is to publish and link wide variety of independent Web data silos in a manner that is queryable as one connected set of datasets supporting advanced Web uses, organisations, and scientific discoveries. The Linked Open Data (LOD) Cloud as well as enterprise applications of Linked Data have shown success in connecting and querying resources across disparate data platforms [3]. Additionally, there is a wider availability of open-source and commercial tools that allow curating, publishing, aggregating, storing, and querying the Linked Data. The standard (typical) Linked Data approach to query independent data silos is to convert all the underlying native data models into the RDF data model and devise querying mechanism through which these independent data silos can be queried in unison. While this approach can be practical for simpler verticals, in case of the HCLS domain it is already predicted that 2–40 exabytes of storage capacity will be needed by 2025 just for the human genomes which will continue to grow approximately 40 petabytes of additional genomic information each year [4]. Nevertheless, raw storage is not the main concern, but the amount of variant data (graph, key-value, stream, text, table, tree, etc.) being queried and analysed is already seen as a major hurdle in the meaningful use of these vast amount of data.

Recently, an atypical approach of federating queries over disparate data models has been initiated – called PolyStore[1] [5] – that exploits different data models and storage engines in their native formats, i.e., without converting them to a common data model. The early demonstrators of PolyStore have shown promising outcomes in federating queries across disparate data models used in the Multiparameter Intelligent Monitoring in Intensive Care II Databases (MIMIC II) [6]. In this paper, we present a semantic approach – called PolyWeb – to federate query over the Web polystores containing cancer and biomedical datasets. We devise a query federation approach for Web polystores that focuses on source selection and join over disparate native data models. It is an open research problem to perform join across data sources[2] using different data models. Further, it is important to understand the gain and loss of querying data sources in their native format compared to the conventional approach of querying curated data sources – from several heterogeneous sources – using a common data model; we compare PolyWeb against the state-of-the-art SPARQL query federation engines. We start the paper by presenting a motivation scenario that requires to query over different cancer genomics datasets. We then discuss query

[1] http://wp.sigmod.org/?p=1629
[2] a data source stores and encapsulates a dataset

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

federation approaches developed using relational database and semantic Web methods. We present the PolyWeb architecture and evaluation of PolyWeb compared to state-of-art query federation engines, finally, we present our conclusion and various routes to optimise the federation process over native data models.

II. Motivating Scenario - Cancer Genomics

The Next Generation Sequencing (NGS) technologies are producing a massive amount of sequencing datasets. As said, there will be a top-up of approximately 40 petabytes of genomics information every year from a wide variety of data sources (hosting different databases, data formats, etc.) published by human genome research centres worldwide. Often, these datasets are published from isolated and different sequencing facilities. Consequently, the process of sharing and aggregating multi-site sequencing datasets are thwarted by issues such as the need to discover relevant data from different sources, built scalable repositories, the automation of data linkage, the volume of data and efficient querying mechanism. PolyWeb is motivated by the needs of the BIOOPENER project[3] which is aiming to link cancer and biomedical datasets providing interlinking and querying mechanisms to understand cancer progression from normal to diseased tissue with pathway components, which in turn helped to map mutations, associated phenotypes, and diseased pathways [7].

```
1  PREFIX : <http://sels.insight.org/cancer−genomics/schema/>
2  PREFIX gene: <http://sels.insight.org/cancer−genomics/gene/>
3  SELECT ∗ WHERE {
4  t1 ?cnv a :CNV .
5  t2 ?cnv :chr ?chr .
6  t3 ?cnv :start ?start .
7  t4 ?cnv :end ?end .
8  t5 ?cnv :sample ?sample .
9  t6 ?cnv :gene gene:MYH7 .
10 t7 ?cnv :disease ?disease .
11 t8 ?cnv :primary_site ?p_site .
12 t9 ?cnv :segment_mean ?seg_mean .
13 }
```

Fig. 1. Federated SPARQL Query

In cancer genomics, discoveries of biological and chemical entities (gene, pathway, drug, diseases, etc.) are available in several overlapping data sources containing complex genomic features, studies, and associations of such features. In order to understand the tumorigenesis, it is often the case that several genetic features, diseases, medical history, etc. are studied together. Considering the exponential growth and variety of genomics and biomedical datasets, it is impractical to assume that all these isolated and disparate datasets will be available in a single data model. In order to tap the vast knowledge locked in these disparate datasets, it's now important to exploit them in native formats. The example SPARQL query shown in Figure 1 retrieves association of a gene (MYH7) with a primary site (where tumour progression starts), disease, copy number variation (CNV), CNV location (start, end), CNV segment mean, and reported samples of patients. The CNV information gives insight into the structural variation of a gene which helps analyse the progression of cancer tumour, ultimately impacting on the prognosis and treatment of disease. The Figure 2 shows an unoptimised query plan which includes type of databases/models (*e.g.*, CSV, RDB, RDF) that can satisfy individual triples from three data sources. In our prior work we have developed a SPARQL federation engine – called SAFE [8] – that federates queries over genomics and clinical trial repositories represented in RDF model. Similarly, in our previous work, we have evaluated a wide variety of proposals (FedX, LHD, SPLENDID, FedSearch, GRANATUM, Avalanche, DAW, ANAPSID, ADERIS, DARQ, LDQPS, SIHJoin, WoDQA and Atlas) on how to execute SPARQL queries in federated settings [9].

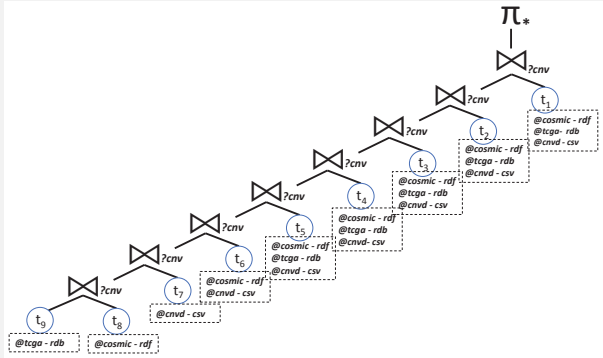

Fig. 2. Unoptimised query plan involving three (3) data sources and three datatypes (CSV, RDB, RDF)

However, we have no work to compare that federates over native data models. Thus the motivation for our research is to investigate, how to enable query federation over native data models. As argued in the introduction, we cannot use existing federation engines off-the-shelf since they are designed to federate over a single data model. Hence the core research question we tackle in this paper is: *How can we devise and implement a query federation approach that retrieves complete results from different native data models in a manner that allows us to compete with off-the-shelf querying engines?*

The benefit of PolyWeb is twofold, first, it reduces the data conversion cost and second, it delegates data querying load to specialised data storage engines, instead of loading curated data from multiple data models to a general-purpose data storage engine. In this paper, we demonstrate that PolyWeb is helpful in reducing the data conversion cost and still be able to retrieve complete result sets from different native data models.

[3]http://bioopenerproject.insight-centre.org

III. Related Work

The relational database and semantic Web research initiatives have offered several federation systems that can unify query answering across disparate databases. We now discuss related literature in these two broad areas.

Relational Databases: The relational database approaches to query federation are mainly focused around the closed-world enterprise settings which require predefined number of data sources, schema, and datasets to process queries across different business units within or between organizations. Some early [10] and more recent query federation approaches [11] target the challenge of distributed query processing and transactions, performance, and replica management. Several early data mediators [12] and middle ware systems [13] – primarily based on the Global as View (GAV) and Local as View (LAV) approaches [14] – developed in the last three decades had the similar motivation. However, the GAV and/or LAV based approaches function on costly assumptions, i.e., availability of global schema, availability of complete data sources, and their mappings; which are reasonable in a closed-world enterprise setting, but impractical to assume in a Web like open scenario. Recently the database community is exploring a new perspective – called Polystore [5] – to unify queries over multiple data models. Similarly, other domains such as the streaming and sensors data processing [15], enterprise analytics [16], social media [17]), etc. have taken initiatives to query over multiple data stores which natively support different data models.

Semantic Technologies: Many query federation engines have been proposed for a Web like scenario using SPARQL [9]. Such engines accept an input query, decompose it into sub-queries, decide relevance of individual data sources for sub-queries, forward sub-queries to individual endpoints accordingly and merge the final results for the query. Initiatives such as the Ontology-Based Data Access (OBDA) exploits ontologies specifically for accessing relational databases (RDB) [18], [19]; such initiatives are complementary to our PolyWeb proposal of querying different data models. The challenges of federating a given query over multiple data models are different in a Web like scenario. Often the datasets are selected from a large pool of prospective data sources and there is no control – unlike enterprises where data resources are controlled – on the availability of complete schemas and datasets. It's important to note that, the database research is investigating the normal federation approach (using single RDB data model) *vs.* Polystore based federation implemented using database technologies [5], [20]. Similarly, in this paper we are investigating normal federation (using single RDF data model) *vs.* Polystore based federation implemented using semantic technologies. To the best of our knowledge, no work has tackled the scenario of processing queries over a federation of different data models and sources in a Web like scenario.

IV. PolyWeb

The PolyWeb architecture is summarised in Figure 3, which shows its four main components: (i) Source Selection: performs source selection based on the capabilities of native data sources; (ii) Query Optimisation: performs cost-effective arrangement of a query (triples patterns) in a manner that reduces query joins and remote requests on the network; (iii) Query Planning: builds joins between query arguments; and (iv) Query Execution: performs the execution of sub-queries against the selected native data sources and merges the results returned. We now briefly describe the four components:

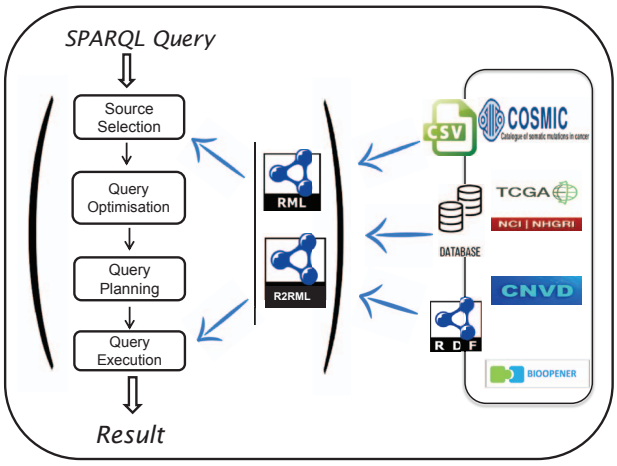

Fig. 3. PolyWeb Architecture

a) Source Selection: PolyWeb performs a predicate-based source selection which identifies the set of relevant data sources, returning non-empty results for the individual triple patterns in a query [8]. PolyWeb performs the sources selection on top of different data models and sources, *e.g.*, SPARQL end-points, RDB and CSV data-access points. PolyWeb relies on mapping definitions such as R2RML [21] and RML [22] mappings to identify relevant data sources. In our work, a mapping definition provides an RDF view of non-RDF data such that non-RDF data can be queried with SPARQL. PolyWeb has a very simple data-free[4] indexing mechanism, the index stores minimal information about given datasets. In case of RDF datasets, it stores only predicate IRIs (Internationalized Resource Identifier) obtained from individual dataset; otherwise, it exploits the mapping definitions to associate predicate IRIs to non-RDF datasets. PolyWeb uses an index of the predicates and an index of the mapping definitions for all datasets. Thanks to the mapping definitions, it is possible to identify non-RDF data sources that can return results from a SPARQL query. The input SPARQL query is broken down into BGPs (Basic Graph Pattern[5]). Each BGP is decomposed into triple patterns from which we get either a bound predicate (*i.e.*, an IRI), or an unbound predicate (*i.e.*, a variable). The predicates extracted from each triple pattern are matched against the set of predicates (obtained from index) present in each data source. If a predicate is found in the set of predicates belonging to a data source, then that particular data source is

[4]index stores only predicate IRIs

[5]http://www.w3.org/TR/2013/REC-sparql11-query-20130321/#BasicGraphPatterns

identified as a relevant data source for that particular triple pattern. For example, consider triple pattern t_2 in the example query (Figure 1). The predicate : chr in triple pattern t_2 is present in all the three data sources, according to the index. Hence, all the three data sources are identified as relevant data sources for triple pattern t_2. In the same way, relevant data sources are identified for all triple patterns of the example query. The current version of PolyWeb supports only single BGPs (excluding constructs such as OPTIONAL, UNION, etc.).

b) Query Optimisation, Planning and Execution: PolyWeb creates a federated query plan against the relevant data sources and executes that plan to get results from multiple data models and merges them into a single result set. In PolyWeb, we devised a query optimisation method – called predicate-based join group (PJG) – that reduces the cost of federated query processing by minimising the number of joins – a key factor that influences the evaluation of a given query – between intermediate results and hence the network data flow. PJG creates a set of unique triple patterns against a group of datasets and data sources. PJG helps in reducing the number of remote requests and the amount of data transferred on the network, which ultimately impact on query performances [9]. In case of our motivational scenario with total 9 unoptimised joins (Figure 2), EEG creates a set of unique triple patterns against a group of datasets which generates six (6) extended exclusive groups (see Figure 4). This is a significant reduction in the number of joins that need to be executed for the example query. In our example query, the joins between triple patterns t_1, t_2, t_3 and t_4 were eliminated at the federation level because of grouping these triple patterns using extended exclusive group approach. Later a cost model, based on counting the number of variable in a triple pattern [23], is applied to join-ordering to further optimise query execution. Low cost joins are pushed downwards, to be executed early, while high cost upwards.

A query plan (Figure 4) is executed in a bottom-up fashion, *i.e.*, it starts from the leaf nodes and traverse up the tree until a root node is reached to generate a combined result set for the input query. The joins are physically implemented in a nested loop bind fashion. Since a two given extended exclusive groups does not contain any common triple pattern, a simple implementation of nested loop bind is achieved by first materializing the left argument and binding the results of left argument with the right argument based on a set of identified join variables. As PolyWeb federates over different data models, the query execution component uses different transformations between queries and result sets. In our implementation, we have the following translation (i) SPARQL to SQL : If an argument of the query plan needs to execute over RDB dataset, then the argument (sub-query) is translated in the SQL format. The transformation between SQL and SPARQL queries is a non-trivial task, due to the difference in semantics between them. The SQL result set obtained from RDB sources are transformed back to the standard SPARQL result format; (ii) SPARQL to SQL (CSV sources): In case of a CSV data source the SPARQL query is translated to the SQL format used by Apache Drill[6]. The result set obtained is transformed from its native format to RDF format.

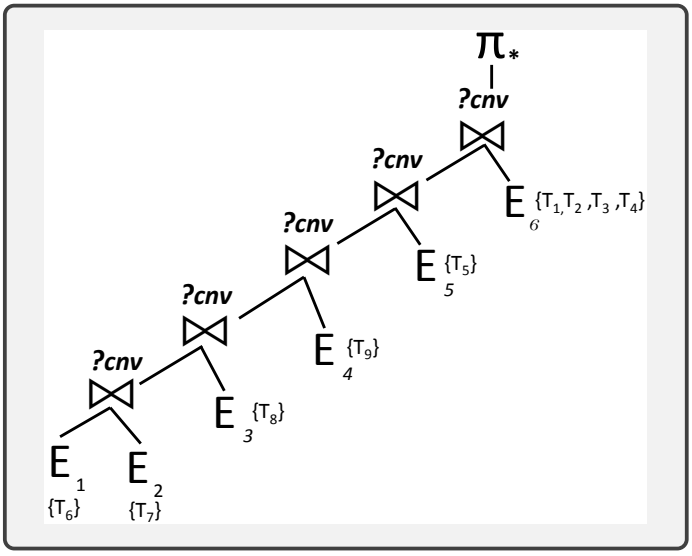

Fig. 4. PolyWeb: Optimised Federated Query Plan of Example Query (Figure 2)

V. Evaluation

This section highlights the results of our evaluation comparing PolyWeb with state-of-the-art SPARQL query federation engine for a variety of queries and datasets based on a series of metrics. Note that the experimental material discussed in the following section and an implementation of PolyWeb are available online[7]. We have selected three datasets for experimental evaluation (i) COSMIC-CNV; (ii) TCGA-OV-CNV; and (iii) CNVD-CNV, provided by COSMIC[8], TCGA[9] and CNVD[10] data providers. All these datasets are part of the studies conducted within the BIOOPENER project. These datasets are available in different data models, *e.g.*, COSMIC-CNV is available in the RDF format, TCGA-OV-CNV is available in RDB format (relational model) and CNVD-CNV is represented using the CSV format. In order to compare PolyWeb with the single data model query federation approaches (such as Fedx), these three datasets are converted into the RDF format. Table I gives an overview of the experimental datasets, for instance, COSMIC-CNV consists of 37 million triples (6.54 GB size in RDF format); with equivalent 27 million database records (3.2 GB size in raw format). The Type column represents the raw data format with number of records (rows) specified in column 9. The last column (RDFisation Time) represents the cost associated with the conversion of

[6]https://drill.apache.org/docs/sql-reference/
[7]https://github.com/yasarkhangithub/PolyWeb
[8]http://cancer.sanger.ac.uk/cosmic
[9]https://cancergenome.nih.gov/
[10]http://202.97.205.78/CNVD/

raw data to RDF. One of the main aims of PolyWeb is to avoid and/or reduce this data conversion cost.

TABLE I
OVERVIEW OF EXPERIMENTAL DATASETS

Datasets	№ trip	RDF Data	Type	Type Data	№ record	RDFisation Time
COSMIC-CNV	37 M	6.54 GB	RDF	3.2 GB	29 M	3 Hours
TCGA-OV-CNV	10 M	494 MB	RDB	212 MB	2.6 M	2 Minutes
CNVD-CNV	1.7 M	128 MB	TSV	34 MB	0.2 M	3 Seconds
Total	49 M	7 GB	-	3.5 GB	32 M	3 Hours

Each RDF dataset was loaded into a different Virtuoso (Open Source v.7.2.4.2) SPARQL endpoint on separate physical machines. CSV dataset is loaded into Apache Drill and relational dataset into MySQL database. All experiments are carried out on a local network, so that network cost remains negligible. The machines used for experiments have a 2.60 GHz Core i7 processor, 8 GB of RAM and 500 GB hard disk running a 64-bit Windows 7 OS. A total of 5 queries are designed to evaluate and compare the query federation performance of PolyWeb against the FedX engine. The queries are of varying complexity and have varying type of characteristics as noted in Table II where we summarise characteristics of these queries following similar dimensions to that used in the Berlin SPARQL benchmark [24].

TABLE II
FEDX AND POLYWEB RETURNS THE SAME NUMBER OF RESULTS FOR EACH QUERY

Characteristics	QE-1	QE-2	QE-3	QE-4	QE-5
№ of Patterns	5	2	5	5	9
№ of Results	3	603	15	1	64
Filters			✓	✓	
`LIMIT` modifier			✓	✓	
`DISTINCT` modifier		✓		✓	✓

For each query type we measured (i) the average source selection time; and (ii) the average query execution time to compare the performance of PolyWeb and FedX. Figure 5 compares the source selection time for PolyWeb and FedX for each query, where the y-axis is presented in log-scale. The rightmost set of bars compares the average source selection time over all queries. Given that the index of PolyWeb remains quite small relative to total data sizes, it can easily be loaded into memory, where lookups can be performed in milliseconds. In case of PolyWeb, source selection is performed in less than a millisecond for all queries. On the other hand, executing remote `ASK` queries are orders of magnitude more costly. Hence we see that the source selection time for PolyWeb is lower than FedX since PolyWeb uses `ASK` queries more sparingly, as previously discussed.

For each query, a mean query execution time was calculated for PolyWeb and FedX by running each query ten times. Figure 6 then compares the mean query execution times of PolyWeb and FedX for all queries. The y-axis is log-scale. We set a time-out of 30 minutes on query execution; with these settings, FedX times-out in the case of one query. Looking at query response times, FedX outperforms PolyWeb in most of the queries and PolyWeb outperforms FedX in complex queries (*i.e.*, **QE-5**). However, the results are comparable considering the additional factors (different data models, query conversion cost, query result conversion cost, etc.) which PolyWeb has to cope with.

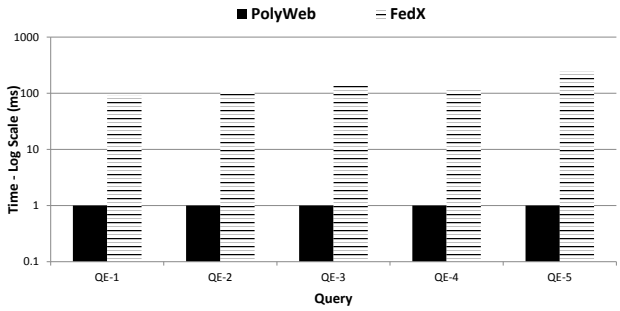

Fig. 5. Comparison of source selection time

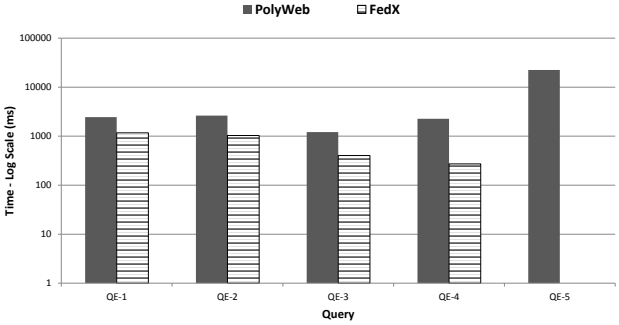

Fig. 6. Comparison of query execution time

Fedx and PolyWeb returned the same number of results corresponding to the five queries (Table II). FedX is designed to retrieve complete result sets and therefore, it implies that PolyWeb is capable of retrieving complete set of query results from different native data models.

VI. CONCLUSION & FUTURE WORKS

In this paper, we have presented PolyWeb: a Web query engine that federates querying answering over native data models. The work is motivated in particular by the needs BIOOPENER project that is aiming to link data across the large-scale cancer and biomedical repositories. PolyWeb aims to reduce the expensive data conversion cost, i.e., loading curated data from multiple data models and sources to a centralized data warehouse for querying; and still be able to retrieve complete results from different native data models. PolyWeb is able to avoid the data conversion cost (total 3 hours for 3.5 GB raw data). PolyWeb has significantly improved and/or comparatively equals to the normal (i.e., single data model) federation engine in terms of source and overall query execution time. In terms of future work, there are a number of possible routes to explore with respect to improving the performance of PolyWeb: (i) we plan to devise a mechanism to probe non-RDF datasets at run-time in the presence of unbound predicates; and (ii) the transformation between query

results (non-RDF and RDF datasets) is a non-trivial task and we plan to optimize such transformations in a more declarative fashion. We plan to include a dynamic cost model that will further prune the predicate-based join groups.

ACKNOWLEDGMENT

The work presented in this research paper has been partly funded by Science Foundation Ireland under Grant No. SFI/12/RC/2289, and by Agence Nationale de la Recherche under Grant No. ANR-14-CE24-0029.

REFERENCES

[1] M. Stonebraker and U. Çetintemel, ""one size fits all": An idea whose time has come and gone (abstract)," in *ICDE 2005, 5-8 April 2005, Tokyo, Japan*. IEEE Computer Society, 2005, pp. 2–11.

[2] O. Badawi, T. Brennan, A. L. Celi, M. Feng, and et. al, "Making big data useful for health care: A summary of the inaugural mit critical data conference," *JMIR Med Inform*, vol. 2, no. 2, p. e22, Aug 2014.

[3] M. Schmachtenberg, C. Bizer, and H. Paulheim, "Adoption of the linked data best practices in different topical domains," in *ISWC 2014, Riva del Garda, Italy, October 19-23, 2014*, 2014, pp. 245–260.

[4] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: Astronomical or genomical?" *PLOS Biology*, vol. 13, no. 7, pp. 1–11, 07 2015. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pbio.1002195

[5] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. B. Zdonik, "The bigdawg polystore system," *SIGMOD Record*, vol. 44, no. 2, pp. 11–16, 2015. [Online]. Available: http://doi.acm.org/10.1145/2814710.2814713

[6] A. Elmore, J. Duggan, M. Stonebraker, M. Balazinska, and et. al., "A demonstration of the bigdawg polystore system," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1908–1911, Aug. 2015.

[7] A. Jha, Y. Khan, M. Mehdi, M. R. Karim, Q. Mehmood, A. Zappa, D. Rebholz-Schuhmann, and R. Sahay, "Towards precision medicine: discovering novel gynecological cancer biomarkers and pathways using linked data," *Journal of Biomedical Semantics*, vol. 8, no. 1, p. 40, Sep 2017. [Online]. Available: https://doi.org/10.1186/s13326-017-0146-9

[8] Y. Khan, M. Saleem, M. Mehdi, A. Hogan, Q. Mehmood, D. Rebholz-Schuhmann, and R. Sahay, "SAFE: SPARQL federation over RDF data cubes with access control," *Journal of Biomedical Semantics*, vol. 8, no. 1, pp. 5:1–5:22, 2017.

[9] M. Saleem, Y. Khan, A. Hasnain, I. Ermilov, and A. N. Ngomo, "A fine-grained evaluation of SPARQL endpoint federation systems," *Semantic Web*, vol. 7, no. 5, pp. 493–518, 2015. [Online]. Available: http://dx.doi.org/10.3233/SW-150186

[10] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, "Mariposa: A wide-area distributed database system," *VLDB J.*, vol. 5, no. 1, pp. 48–63, 1996. [Online]. Available: http://dx.doi.org/10.1007/s007780050015

[11] D. J. DeWitt, A. Halverson, R. V. Nehme, S. Shankar, J. Aguilar-Saborit, A. Avanes, M. Flasza, and J. Gramling, "Split query processing in polybase," in *ACM SIGMOD ICMD, New York, NY, USA, June 22-27, 2013*.

[12] G. Wiederhold, "Mediators in the architecture of future information systems," *IEEE Computer*, vol. 25, no. 3, pp. 38–49, 1992. [Online]. Available: https://doi.org/10.1109/2.121508

[13] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang, "Optimizing queries across diverse data sources," in *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, 1997, pp. 276–285. [Online]. Available: http://www.vldb.org/conf/1997/P276.PDF

[14] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, 2002, pp. 233–246. [Online]. Available: http://doi.acm.org/10.1145/543613.543644

[15] E. Kharlamov, T. P. Mailis, K. Bereta, D. Bilidas, S. Brandt, E. Jiménez-Ruiz, S. Lamparter, C. Neuenstadt, and et. al., "A semantic approach to polystores," in *2016 IEEE ICBD, BigData 2016, Washington DC, USA, December 5-8, 2016*, 2016, pp. 2565–2573.

[16] E. Begoli, D. Kistler, and J. Bates, "Towards a heterogeneous, polystore-like data architecture for the US department of veteran affairs (VA) enterprise analytics," in *2016 IEEE ICBD, BigData 2016, Washington DC, USA, December 5-8, 2016*. IEEE, 2016, pp. 2550–2554.

[17] S. Dasgupta, K. Coakley, and A. Gupta, "Analytics-driven data ingestion and derivation in the AWESOME polystore," in *2016 IEEE ICBD, BigData 2016, Washington DC, USA, December 5-8, 2016*. IEEE, 2016, pp. 2555–2564.

[18] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev, "Ontology-based data access: Ontop of databases," in *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Alani, L. Kagal, A. Fokoue, P. T. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, Eds., vol. 8218. Springer, 2013, pp. 558–573.

[19] M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter, "Ontology-based data access: a study through disjunctive datalog, csp, and MMSNP," in *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, R. Hull and W. Fan, Eds. ACM, 2013, pp. 213–224.

[20] V. Gadepally, P. Chen, J. Duggan, A. J. Elmore, B. Haynes, J. Kepner, S. Madden, T. Mattson, and M. Stonebraker, "The bigdawg polystore system and architecture," in *2016 IEEE High Performance Extreme Computing Conference, HPEC 2016, Waltham, MA, USA, September 13-15, 2016*. IEEE, 2016, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/HPEC.2016.7761636

[21] S. Das, S. Sundara, and R. Cyganiak, "R2RML: RDB to RDF Mapping Language," World Wide Web Consortium (W3C), W3C Recommendation, Sep. 27 2012. [Online]. Available: http://www.w3.org/TR/2012/REC-r2rml-20120927/

[22] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. V. de Walle, "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data," in *Proc. of the Linked Data on the Web 2014*, 2014. [Online]. Available: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf

[23] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds, "Sparql basic graph pattern optimization using selectivity estimation," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 595–604. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367578

[24] C. Bizer and A. Schultz, "The Berlin SPARQL benchmark," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 2, pp. 1–24, 2009.