A Demonstration of TripleProv: Tracking and Querying Provenance over Web Data

Marcin Wylot
eXascale Infolab
University of Fribourg
Switzerland
marcin.wylot@unifr.ch

Philippe Cudré-Mauroux
eXascale Infolab
University of Fribourg
Switzerland
pcm@unifr.ch

Paul Groth
Elsevier Labs
The Netherlands
p.groth@elsevier.com

ABSTRACT

The proliferation of heterogeneous Linked Data on the Web poses new challenges to database systems. In particular, the capacity to store, track, and query provenance data is becoming a pivotal feature of modern triple stores. In this demonstration, we present TripleProv: a new system extending a native RDF store to efficiently handle the storage, tracking and querying of provenance in RDF data. In the following, we give an overview of our approach providing a reliable and understandable specification of the way results were derived from the data and how particular pieces of data were combined to answer the query. Subsequently, we present techniques enabling to tailor queries with provenance data. Finally, we describe our demonstration and how the attendees will be able to interact with our system during the conference.

1. INTRODUCTION

With the rapid expansion of the Linked Open Data (LOD) cloud, developers are able to query and integrate large collections of disparate online data. As the LOD cloud is rapidly growing, so is its heterogeneity.

The heterogeneity of RDF data combined with the ability to easily integrate it makes tracing back and querying the provenance of query results become essential in data processing. For example, one may want to analyze which sources were instrumental in providing results, how data sources were combined, to validate or invalidate results, or to delve deeper into data related to the results retrieved. One also may want to filter results based on provenance information, e.g., *find me all the titles of articles about "Obama", but derive the answer only from sources attributed to "US News"*.

Within the database and the Web communities, there have been several efforts in developing models and syntaxes to interchange provenance, which resulted in the recent W3C PROV recommendation [3]. However, less attention has been given to the efficient handling of provenance data within RDF database systems. While some systems store quadruples or named graphs, to the best of our knowledge, no current triple store is able to automatically derive provenance data for the results it produces or to tailor queries with provenance data.

To support these use-cases, the most common mechanism used within RDF data management is named graphs. This mechanism was recently standardized in RDF 1.1. [4]. Named graphs associate a set of triples with a URI. Using this URI, metadata including provenance can be associated with the graph. While named graphs are often used for provenance, they are also used for other purposes, for example, to track access control information. Thus, while RDF databases (i.e., triple stores) support named graphs, there has only been a relatively small amount of work specifically focused on provenance within the triple store itself and much of it has been focused on theoretical aspects of the problem [1].

In the following, we present a demonstration of TripleProv, a new RDF database system supporting the transparent and automatic derivation of detailed provenance information for arbitrary queries and the execution of queries incorporating provenance predicates. TripleProv is based on a native RDF store [7], which we have extended with different physical models to store provenance data on disk in a compact fashion. The system allows to trace provenance at two different granularity levels as well as to execute provenance-enabled queries. This demonstration builds on our previous work [5, 6].

The rest of this paper is organized as follows: we define provenance polynomials and how to track provenance at different levels of granularity in Section 2; we introduce provenance-enabled queries in Section 3; in Section 4, we present our new provenance-aware query execution strategy allowing to store, track, and query provenance data; finally, we discuss our demonstration scenario, its user experience, as well as takeaway points in Section 6.

2. PROVENANCE POLYNOMIALS

The first question we tackle is how to represent provenance information that we want to return to the user in addition to the results themselves. Beyond listing the various sources involved in the query, we want to be able to characterize the specific ways in which each source contributed to the query

Proceedings of the VLDB Endowment, Vol. 8, No. 12

results. Here, we leverage the notion of *provenance polynomials* [5, 2], which is an algebraic structure representing how the data is combined to derive the query answer using different relational algebra operators (e.g., UNION, JOINS). We focus on the practical realization of this model within a high performance triple store to answer queries seen as useful in practice. Specifically, we focus on *the capability to pinpoint and trace back, for each query result, the exact pieces of data from which the result was selected and how they were combined to deliver the result.*

At the logical level, we use in the provenance polynomials two basic operators to express the way the pieces of data were combined to answer the query. The first one ($\oplus$) to represent unions of sources, and the second ($\otimes$) to represent joins between sources.

Unions are used in two cases when generating the polynomials. First, they are used when a constraint or a projection can be satisfied with triples coming from multiple sources (meaning that there are more than one instance of a particular triple which is used for a particular operation). Second, unions are also used when multiple entities satisfy a set of constraints or projections.

As for the join operator, it can also be used in two ways: to express the fact that sources were joined to handle a constraint or a projection, or to handle object-subject or object-object joins between a few sets of constraints.

RDF data provenance can be modeled at different granularity levels. Current approaches, typically, return a list of named graphs from which the answer was computed [5]. Our system, besides generating polynomials summarizing the complete provenance of results, also supports two levels of granularity. First, a lineage l_i (i.e., an element appearing in a polynomial) can represent the source of a triple, (e.g., the fourth element in a quadruple). We call this granularity level *source-level*. Second, a lineage can represent a quadruple (i.e., a triple plus its corresponding source). This second type of lineage produces polynomials consisting of all the pieces of data (i.e., quadruples) that were used to answer the query, including all intermediate results. We call this level of granularity *triple-level*.

For more detail on provenance polynomials in TripleProv, we refer the reader to our previous work [5].

3. PROVENANCE-ENABLED QUERIES

Besides providing to a user a provenance polynomial, i.e., a reliable and understandable description of the way the query answer was derived, we also give the possibility to tailor RDF queries with provenance data. In this work, we adopt the view proposed in the W3C PROV Family of Documents[1] defining a model, corresponding serializations and other supporting definitions to enable the interoperable interchange of provenance information in heterogeneous environments such as the Web. We also adopt the terminology of Cyganiak's original NQuads specification[2], where the *context value* refers to the provenance or source of the triple. We note that context values often are used to refer the named graph to which a triple belongs. Based on this background, we introduce the following terminology used within this paper:

[1] `http://www.w3.org/TR/prov-overview/`
[2] `http://sw.deri.org/2008/07/n-quads/`

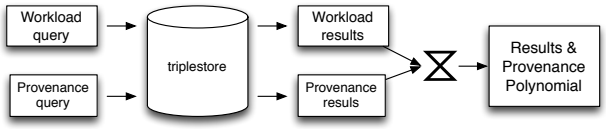

Figure 1: Executing provenance-enabled queries; both a workload and a provenance query are given as input to a triplestore, which produces results for both queries and then combine them to obtain the final results and the provenance polynomial.

Definition 1. A **Workload Query** is a query producing results a user is interested in. These results are referred to as *workload query results*.

Definition 2. A **Provenance Query** is a query that selects a set of data from which some workload query results should originate. Specifically, a *Provenance Query* returns a set of *context values* whose triples will be considered during the execution of a *Workload Query*.

Definition 3. A **Provenance-Enabled Query** is a pair consisting of a *Workload Query* and a *Provenance Query*, producing results a user is interested in (as specified by the *Workload Query*) and originating only from data preselected by the *Provenance Query*.

We assume a strict separation of the workload query on one hand and the provenance query on the other (as illustrated in Figure 1). Provenance and workload results are joined to produce a final result. A consequence of our design is that workload queries can remain unchanged, while the whole process of applying provenance filtering takes place during query execution. The body of the provenance query specifies the set of context values that identify which triples will be used when executing the workload queries.

For the full details on executing provenance-enabled queries in TripleProv, we refer the reader to our previous work [6].

4. PROVENANCE IN QUERY PROCESSING

Given a SPARQL query, our system first analyzes the query to produce a physical query plan, i.e., a tree of operators that are then called iteratively to retrieve molecules (basic storage units in TripleProv) susceptible of containing data relevant to the query. RDF molecules [7] are similar in their simplest form to property tables and store, for each subject, the list or properties and objects related to that subject.

The molecules are retrieved by taking advantage of a molecule index to locate all elements containing values specified in the query.

In parallel to the classical query execution process, TripleProv keeps track of the entities that have been instrumental in producing results for the query. For each molecule inspected, our system keeps track of the provenance of any triple matching the current pattern being handled. In case multiple molecules are used to construct the final results, the system keeps track of the local provenance of the molecules by performing a union of the local provenance data using a global provenance structure (see [5] for details).

To tailor RDF quires with provenance data, we propose several query execution strategies. The simplest strategy is

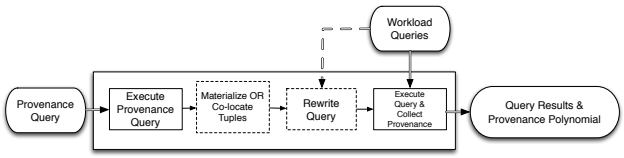

Figure 2: Generic provenance-aware query execution pipeline, where both the workload queries and the provenance query get executed in order to produce the final results and the provenance polynomial.

to execute both the RDF query and the provenance query independently, and to join both result sets based on the provenance polynomial. We also propose a strategy to pre-materialize some of the data based on the provenance specification. Another way to execute a provenance-enabled query we present is through dynamic query rewriting; in that case, the workload query is rewritten using the provenance query (or some of its results) and only then is the query executed.

Figure 2 gives a high-level perspective on the query execution process. The provenance and workload queries are provided as input; the query execution process can vary depending on the exact strategy chosen, but typically starts by executing the provenance query and optionally pre-materializing or co-locating data; the workload queries are then optionally rewritten—by taking into account some results of the provenance query. Finally we execute the workload query and simultaneously we collect information of entities contributing to the answer and the way they are combined. The process returns as output the workload query results, restricted to those which are following the specification expressed in the provenance query and the provenance polynomial describing the way the results were derived.

We give more detail on the execution process, storage models, various query execution and indexing strategies, and the system architecture in our previous work [7, 5, 6].

5. PERFORMANCE

We implemented two different storage models to store provenance information, two granularity levels for the provenance polynomials, and five different query execution strategies for the provenance-enabled queries. Fort detailed discussion, experiment scenarios, and the performance evaluation we refer the reader to our previous works [5, 6].

Overall, the performance penalty created by tracking provenance in TripleProv ranges from a few percents to almost 350%. Clearly, we observe a significant difference between the two main provenance storage models implemented. We also notice considerable difference between the two granularity levels. Clearly, the more detailed provenance granularity requires more time for query execution than the simpler one, because of the more complete physical structures that need to be created and updated while collecting the intermediate results sets.

Our implementations supporting provenance-enabled queries overall outperform the vanilla TripleProv. This is unsurprising, since the selectivity of provenance data in the datasets allows us to avoid unnecessary operations on tuples which do not add to the result [6].

The *Full Materialization* strategy, where we pre-materialize all relevant subsets of the molecules, makes the query execution on average 44 times faster than the vanilla version. The price for the performance improvement is the time we have to spend to materialize molecules, in our experiments it was 95 seconds, which can however be amortized by executing enough workload queries. This strategy consumed about 2% more memory for handling the materialized data. The *Partial Materialization* strategy performs on average 35 times faster. The advantage over the *Full Materialization* strategy is that for *Partial Materialization*, the time to execute a provenance query and materialize data is 475 times lower and takes only 0.2 second.

6. DEMONSTRATION

The demonstration will put the conference attendees in the position of a database user. In that sense, they will be able to pose provenance-enabled queries against TripleProv and to retrieve results along with provenance polynomials showing how the different pieces of data were combined to derive the results. The system will be deployed on a Linux-based laptop with a graphical user interface. We will provide two different Web datasets: the Billion Triples Challenge (BTC)[3] and the Web Data Commons (WDC)[4]. Both datasets are collections of RDF data gathered from the Web. They represent two very different kinds of RDF data. The Billion Triple Challenge dataset was created based on datasets provided by Falcon-S, Sindice, Swoogle, SWSE, and Watson using the MultiCrawler/SWSE framework. The Web Data Commons project extracts all Microformat, Microdata and RDFa data from the Common Crawl Web corpus and provides the extracted data for download in the form of RDF-quads or CSV-tables for common entity types (e.g., products, organizations, locations, etc.).

The aforementioned datasets ideally represent the variety of Linked Data coming from multiple sources and produced in various ways; they gather data with different levels of trustability and usability, different degrees of authoritativeness, and gathered automatically from independent and potentially unknown sources. Those heterogeneous datasets incorporate provenance information, which we leverage to provide to the user a reliable and understandable description of the way the executed query was answered, i.e., the way the answer was derived: a provenance polynomial. We also enable the users to tailor queries with provenance data, including or excluding some specific lineage.

For the demonstration, we will provide a set of exemplary queries showing the differences in query execution when answering provenance-enabled queries. The queries will be designed to highlight the way the data can be derived. We will also provide various provenance queries, hence allowing to restrict the data used to produce the answers in different ways. The users will be able to arbitrarily combine and modify both the workload and the provenance queries, and to observe the resulting effects on the output (i.e., on the provenance polynomials and query results).

To further illustrate our demonstration, we present a few examples of provenance-enabled queries that are simplified versions of what a user will be able to execute on our system. In the examples below, *context values* are denoted as *?ctx*.

A common case is to ensure that the data used to produce the answer comes from a set of trusted sources.

[3] `http://km.aifb.kit.edu/projects/btc-2009/`
[4] `http://webdatacommons.org/`

Given a workload query that retrieves titles of articles about "Obama":

```
SELECT ?t WHERE  {
  ?a <type> <article> .
  ?a <tag> <Obama> .
  ?a <title> ?t . }
```

One may want to ensure that the articles retrieved come from sources attributed to the government:

```
SELECT ?ctx WHERE {
 ?ctx prov:wasAttributedTo <government> .}
```

As per the W3C definition, provenance is not only about the source of data but is also about the manner in which the data was produced. Thus, one may want to ensure that the articles in question were edited by somebody who is a "SeniorEditor" and that articles where checked by a "Manager". Thus, we could apply the following provenance query while keeping the same "Obama" workload query:

```
SELECT ?ctx WHERE {
 ?ctx prov:wasGeneratedBy <articleProd>.
 <articleProd> prov:wasAssociatedWith ?ed .
 ?ed rdf:type <SeniorEdior>  .
 <articleProd> prov:wasAssociatedWith ?m .
 ?m rdf:type <Manager> . }
```

Another way to apply provenance-enabled queries is for scenarios in which data is integrated from multiple sources. For example, we may want to aggregate the chemical properties of a drug (e.g., its potency) provided by one database with information on whether it has regulatory approval provided by another:

```
SELECT ?potency ?approval WHERE {
  ?drug <name> ''Sorafenib'' .
  ?drug ?link ?chem.
  ?chem <potency> ?potency .
  ?drug <approvalStatus> ?approval }
```

6.1 User Experience

The user will be able to specify in real-time restrictions on the data to be used to derive answers for the workload queries using provenance queries. In addition, he will be able to modify the provenance specification and observe how results evolve with different provenance queries. Besides being able to execute his own queries, the user will also be able to experiment with sets of pre-written queries; those queries will be prepared to highlight the influence of provenance-enabled queries on query execution and results. After executing the query, the system will display the results of the workload query obtained from data satisfying both the workload and the provenance specification.

The system will also display a provenance polynomial in addition to the results (i.e., a high-level specification of how the results were derived). The provenance polynomial will be returned at two levels of granularity (see Section 2). The source-level polynomial and the corresponding triple-level polynomials will be visually connected, in order for the user to be able to examine and compare both polynomials simultaneously. The demonstration will also allow the user to analyze which part of the workload query was derived using which pieces of data: for each query result, the user will be able to identify the exact source(s) from which the result was derived. Leveraging the provenance polynomial, the user will be able to understand how particular pieces of data were combined to deliver the final result, thus he will be able to trace back the query execution process.

6.2 Takeaways

This demonstration highlights the importance of tracking the provenance of Web data. TripleProv offers an interactive and pragmatic experience for users interested in exploring trade-offs related to supporting provenance. Through our demonstration, users will be able to experience differences when executing queries with various provenance specifications. The system will also provide the user a detailed specification on the way results were derived and on how particular pieces of data were combined to answer the query. A visual interface will show how results evolve as the provenance specification gets updated as well as how different pieces of data can be combined when answering provenance-enabled queries.

7. ACKNOWLEDGMENTS

This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459 and by the Dutch national program COMMIT.

8. REFERENCES

[1] F. Geerts, G. Karvounarakis, V. Christophides, and I. Fundulaki. Algebraic structures for capturing the provenance of sparql queries. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 153–164, New York, NY, USA, 2013. ACM.

[2] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40. ACM, 2007.

[3] P. Groth and L. Moreau (eds.). PROV-Overview. An Overview of the PROV Family of Documents. W3C Working Group Note NOTE-prov-overview-20130430, World Wide Web Consortium, Apr. 2013.

[4] D. W. R. Cyganiak and M. L. (Ed.). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, February 2014. http://www.w3.org/TR/rdf11-concepts/.

[5] M. Wylot, P. Cudre-Mauroux, and P. Groth. TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 455–466, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[6] M. Wylot, P. Cudré-Mauroux, and P. Groth. Executing Provenance-Enabled Queries over Web Data. In *Proceedings of the 24rd International Conference on World Wide Web*, WWW '15, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[7] M. Wylot, J. Pont, M. Wisniewski, and P. Cudré-Mauroux. dipLODocus[RDF]: short and long-tail RDF analytics for massive webs of data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 778–793, Berlin, Heidelberg, 2011. Springer-Verlag.