Adaptive RDF Query Processing Based on Provenance

Marcin Wylot[1], Philippe Cudré-Mauroux[1], and Paul Groth[2(✉)]

[1] University of Fribourg, Fribourg, Switzerland
{marcin,phil}@exascale.info
[2] VU University Amsterdam, Amsterdam, The Netherlands
p.t.groth@vu.nl

Given the increasing amounts of RDF data available from multiple heterogenous sources, as evidenced by the Linked Open Data Cloud, there is a need to track provenance within RDF data management systems [1]. In [8], we presented TripleProv, a database system supporting the transparent and automatic capture of detailed provenance information for arbitrary queries. A key focus of TripleProv is the efficient implementation of provenance-enabled queries over large scale RDF datasets. TripleProv is based on a native RDF store, which we have extended with two different physical models to store provenance data on disk in a compact fashion. In addition, TripleProv supports several new query execution strategies to derive provenance information at two different levels of aggregation. At one level, the exact sources for a query results can be identified. The second, more detailed level, provides the full lineage of the query results including the various constraints, projections and joins involved in answering the query. In addition to these levels of aggregation at the data source level, we support tracking the provenance at the quadruple level. That is, every quad (i.e. tuple) is annotated and those annotations are tracked through the query processing pipeline. This tracking is done by leveraging the concept provenance polynomials [3]. That is capturing the provenance representation as a formula over tuples. Our work follows on from previous work on annotating or coloring RDF triples [2,9] by focusing on both scale and query adaptivity.

At the logical level, we use two basic operators to express the provenance polynomials. 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). The following polynomial:

$$l1 \oplus l2 \oplus l3$$

for instance, encodes the fact that a given result can originate from three different sources ($l1$, $l2$, or $l3$). 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

B. Ludäscher and B. Plale (Eds.): IPAW 2014, LNCS 8628, pp. 264–266, 2015.
DOI: 10.1007/978-3-319-16462-5_29

object-subject or object-object joins between a few sets of constraints. The following polynomial:

$$(l1 \oplus l2) \otimes (l3 \oplus l4)$$

for example, encodes the fact that sources $l1$ or $l2$ were joined with sources $l3$ or $l4$ to produce results.

Provenance polynomials can be used to compute a trust or information quality score based on the sources used in the result.

TripleProv works on large scale real world data. We have tested the system on two datasets consisting of over 110 million triples each. Each dataset is roughly 25 GB in size. The datasets are drawn, respectively, from two crawls of the Web: the Billion Triple Challenge[1] and the Web Data Commons[2] [7].

Based on this foundation, this work presents preliminary results on *adaptively modifying query execution* based on provenance. Specifically, we have extended TripleProv to allow a specific list of sources (e.g. trusted sources) to be provided which are to be used when answering a query. Additionally, one can also specify a list of sources to avoid during query execution (e.g. a list of untrusted sources). The specified lists are checked at every stage of query execution process. This means that even at the level of intermediate results, which are not necessarily presented as an output, we ensure that these data sources are not touched. We note that this trigger based approach allows for potentially dynamic changes in the source list at query execution.

Such adaptive query processing is useful for a number of use cases. For instance, one could restrict the results of a query to certain subsets of sources or use provenance for access control such that only certain sources will appear in a query result. Identifying results (i.e., particular triples) with overlapping provenance is also another prospective use case. Additionally, one could detect whether a particular result would still be valid when removing a source dataset. We could also extend our approach to with Hartig's tSPARQL [4] to be able to query trust annotations in combination with provenance sources.

In [8], we found that provenance tracking within the database caused between a 60–70% overhead. While this is acceptable for many use cases, it would be beneficial if the performance would be faster. We believe that by taking advantage of knowing data provenance one could potentially optimize the performance of the database. We note that our approach focused on adjusting the pipeline of query processing verses querying provenance after the fact as in other systems [5,6]. An interesting area of work would be to study the trade off between runtime query adaptation based on provenance and post hoc provenance queries.

This work is a first step towards showing how provenance can be used to make it easier to work with heterogenous RDF data.

Acknowledgements. This work was funded in part by the Swiss National Science Foundation under grant number PP00P2_128459 and by the Data2Semantics project in the Dutch national program COMMIT.

[1] http://km.aifb.kit.edu/projects/btc-2009/.
[2] http://webdatacommons.org/.

References

1. Ding, L., Peng, Y., da Silva, P.P., McGuinness, D.L.: Tracking RDF graph provenance using RDF molecules. In: International Semantic Web Conference (2005)
2. Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., Christophides, V.: Coloring RDF triples to capture provenance. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 196–212. Springer, Heidelberg (2009)
3. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 31–40. ACM (2007)
4. Hartig, O.: Querying trust in RDF data with tSPARQL. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 5–20. Springer, Heidelberg (2009)
5. Karvounarakis, G., Ives, Z.G., Tannen, V.: Querying data provenance. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 951–962. ACM (2010)
6. Miles, S.: Electronically querying for the provenance of entities. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 184–192. Springer, Heidelberg (2006)
7. Mühleisen, H., Bizer, C.: Web data commons - extracting structured data from two large web corpora. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M. (eds.), LDOW. CEUR Workshop Proceedings, vol. 937. CEUR-WS.org (2012)
8. Wylot, M., Cudré-Mauroux, P., Groth, P.: Tripleprov: efficient processing of lineage queries over a native rdf store. In: Proceedings of the 23rd Intenational World Wide Web Conference (WWW'2014) (2014)
9. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. Web Semant. **11**, 72–95 (2012)