Tracking and Visualizing Provenance using PostgreSQL

Alexander Rush (srush@mit.edu)
Nirmesh Malviya (nirmesh@mit.edu)

December 9, 2010

Abstract

Many scientific and business applications require keeping detailed track of the origin of data items. Such history is referred to as provenance or lineage of data. The most basic data provenance operations involve tracking which data items an existing data item was derived from (backward data provenance) and what all data items were derived from a given data item (forward data provenance).

In this work, we modify the internals of PostgreSQL, a popular open source database, to capture data provenance for a subset of SQL DDL. We have also developed an interactive web interface to query and visualize forward and backward provenance for data items in different databases.

1 Introduction

A large number of applications such as scientific data management, data integration, information extraction and business analytics involve processing base data to generate a large amount of new data derived from the base data. Capturing which data items contributed to creation of a new data item is important in the face of issues like questionable quality of base data, potential uncertainty associated with it, as well as the authority and trust assessment of the user performing the operations. We see that if data history is not traced in any form as the operations are performed, it would be impossible to track what future tables potential errors in early stage data collection may have propagated to. Tracking down errors and updating all tuples derived from an erroneous data item can be prohibitively costly or even impossible in the absence of this historical information.

This metadata capturing historical information about a data item is referred to as the *provenance* of the item. The type and amount of provenance information required for different applications vary widely, and depending on the information stored, provenance can be used to derive probabilities associated with a derived data item using uncertainty information about data items that produced it [20], for debugging schema mappings [10], learning authority of data sources [10], using user feedback for adjusting ordering in ranking systems [19] and many others [17]. Source provenance, a subset of the different types of historical information about a data item, only pertains to what data sources an item was derived from (we can also have transformation provenance, see Section 2 for a discussion).

Traditionally, attempts to track provenance of data were confined to data processing stages outside of the data management system. With increased availability of cheap storage in recent years and

data explosion online (and otherwise), the problem of capturing, storing and querying provenance has been extensively studied [9, 13, 12]. A number of provenance management solutions have been proposed or developed from the ground up in recent years [15, 18, 20, 2, 8, 11, 6]. However, to the best of the our knowledge, till date none of the widely used commercial database systems offer even basic support for tracking provenance of all queries executed against the database system.

In this project, we have added support for tracking source provenance inside a popular open source database system PostgreSQL [1] for a subset of SQL. Our goal is not to compete with existing ground up provenance solutions in terms of efficiency of storing or querying provenance but rather to implement a basic provenance system in a widely used existing large scale system.

In summary, our contributions in this project are:

- We implement a basic source provenance system in PostgreSQL, which eagerly captures provenance of newly created data items, allowing for the ability to track provenance using Postgres.
- We develop an interactive web interface to query forward and backward data provenance on existing tables.

The rest of this paper is organized as follows. We explain provenance in Section 2 and briefly discuss the state of the art in this area in Section 3. We describe our implementation in Section 5 and show our query interface in Section 6. We finally conclude in Section 7.

2 Background

The term *provenance* or *lineage* of data broadly refers to what source a data item came from, what transformations were applied on the original data to arrive at the current data item, how the current data item has been updated over time and what other data items might have been derived from the current data item. Broadly speaking, provenance information belongs to one of two categories: *source provenance* and *transformation provenance* [5]. Source provenance refers to source data from which a data element was derived, whereas transformation provenance is concerned with the different processes or operations that were used to arrive at the data element starting from the source data. A slightly different characterization [13] classifies provenance as mostly either data-based, in which well defined data models are used to track fine-grained provenance of data elements, or process-based, where the processes generating a data element along with coarse-grained provenance is captured.

The proposal for the SciDB project[18] lists three requirements for a useful provenance system:

- For any data element, we would like to recover the **derivation history**. This would be helpful in identifying if a data element originated from a suspect or unreliable data source.
- If a data element is found to have had value some value in error, we would like to trace forward to see what other data elements could have been affected from the element's previous erroneous value. If a portion of data is identified as erroneous, we can use this to determine what other data elements in the data store have values that cannot be relied on as a result of this.

- At any point, we would like to reproduce the construction of the current data in the data store. This would be of use in propagating updates downstream when data element errors are identified and corrected.

Thus we can have two types of provenance queries, *forward* and *backward* provenance. In forward provenance, given some data element, we ask what data elements were produced from it (and optionally what processing operations were applied to them during the element's creation). In backward provenance, given some data element, we ask where it originally came from (and optionally what processing led to its creation).

Fig **??** shows examples of these two queries. We assume that our original data is $s1$, it leads to $s2$ and $s3$. In turn, $s4$ is derived from $s3$. Our forward query from $s1$ leads to each of the other data elements, while the backward query from $s4$ leads back to the root.

Provenance can either be computed and stored eagerly by capturing forward and backward data pointers (for example [20]) or lazily by just recording the query log along with a knowledge of inversion operations [18]. The two strategies have vastly different storage requirements and query efficiency. We discuss some work on this in Section 3.

A more detailed survey of data provenance approaches, including a discussion of conceptual properties of provenance models and different storage and recording strategies can be found in [9].

For our implementation, we exclusively focus on eager source data provenance.

3 Related Work

In particular, we hope to respond to the challenge given by [7].

> Recording the log is easy. The hard part is to create a provenance query language and an efficient implementation.

Recent work in this area includes the Trio/LIVE system and the Panda project. However, neither of these projects handle backward provenance efficiently or provide language support for querying provenance. This lack of historical record can make database systems impractical for scientific research.

One practical issue with available database systems is the difficulty of tracing data history.

[3]

[17]

[16]

[9]

[14]

[4, 3, 17]

[**?**]

- Implements a technique for backward provenance by connecting derived tuples to the previous tuples. Unfortunately, this technique has very high space complexity and may not scale in practice.
- Panda [13]

  Panda is an early stage proposal for a general provenance system for data. This system does not propose a practical algorithm or language features for provenance.
- SciDB [7]

  The SciDB system proposes a provenance system. Current proposals have low space complexity but with non-trivial query time.

In the LIVE provenance system built as part of the Trio project, the database stores a back pointer for each derived data element. In our example, it would store $(s1, s2)$, $(s1, s3)$, and $(s3, s4)$ explicitly. It can then perform very efficient backwards queries by walking along this tree. Unfortunately, this technique requires storing a derivation pair for each derived and its parents, which can be very expensive.

The SciDB proposal suggests a different way to implement provenance. For forward queries, the propose using the databases version control system and following forward deltas. They note that backward provenance is more difficult to implement. Two possible methods they mention are to also include backward deltas as part of version control or to implement an algorithm which can reverse the queries from the log.

The other open quesion surrounding provenance is how to include provenance queries within SQL. There have been relatively few proposed solutions for this problem. The LIVE system implements a keyword "valid at" which queries a data element at a specific revision. For instance the query:

```
SELECT * FROM <table-name> VALID AT <revision-number>
```

This query returns the rows from a table at a specific revision number. This syntax allows a query to access a revision, but does not allow us to specify a provenance query the directly.

4 System Architecture

Figure of system architecture – provenance storage and querying.

5 Tracking Provenance inside PostgreSQL

Catalog

5.1 Simple selects - queries on One Table

5.2 Handling joins

5.3 Aggregates

5.4 Changes to the storage manager for disk based workloads

6 Visualizing Forward and Backward Provenance

couple of screenshots, and one showing the graph.

don't support stored procedures , they are black boxes, and unless something useful is known about the underlying functions, we anyway do not get to know too much about them.

7 Conclusion and Future Work

Just reword the abstract.

References

[1] Postgresql. `www.postgresql.org`.

[2] Practical lineage tracing in data warehouses. In *Proceedings of the 16th International Conference on Data Engineering* (Washington, DC, USA, 2000), IEEE Computer Society, pp. 367–.

[3] BUNEMAN, P., KHANNA, S., AND CHIEW TAN, W. Data provenance: some basic issues. In *In Foundations of Software Technology and Theoretical Computer Science* (2000), Springer, pp. 87–93.

[4] CHAPMAN, A. P., JAGADISH, H. V., AND RAMANAN, P. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2008), SIGMOD '08, ACM, pp. 993–1006.

[5] CHIEW TAN, W. Research problems in data provenance. *IEEE Data Engineering Bulletin 27* (2004), 45–52.

[6] CHITICARIU, L., TAN, W.-C., AND VIJAYVARGIYA, G. Dbnotes: a post-it system for relational databases based on provenance. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2005), SIGMOD '05, ACM, pp. 942–944.

[7] CUDRÉ-MAUROUX, P., KIMURA, H., LIM, K., ROGERS, J., SIMAKOV, R., SOROUSH, E., VELIKHOV, P., WANG, D., BALAZINSKA, M., BECLA, J., ET AL. A demonstration of SciDB: a science-oriented DBMS. *Proceedings of the VLDB Endowment 2*, 2 (2009), 1534–1537.

[8] FOSTER, I. T., VÖCKLER, J.-S., WILDE, M., AND ZHAO, Y. Chimera: Avirtual data system for representing, querying, and automating data derivation. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management* (Washington, DC, USA, 2002), SSDBM '02, IEEE Computer Society, pp. 37–46.

[9] GLAVIC, B., AND DITTRICH, K. Data provenance: A categorization of existing approaches.

[10] GREEN, T. J., KARVOUNARAKIS, G., IVES, Z. G., AND TANNEN, V. Update exchange with mappings and provenance. In *Proceedings of the 33rd international conference on Very large data bases* (2007), VLDB '07, VLDB Endowment, pp. 675–686.

[11] GROTH, P., MILES, S., AND MOREAU, L. Preserv: Provenance recording for services. In *In Proceedings of the UK OST e-Science second All Hands Meeting 2005 (AHM05), Nottingham,UK* (2005).

[12] HALEVY, A., FRANKLIN, M., AND MAIER, D. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (New York, NY, USA, 2006), PODS '06, ACM, pp. 1–9.

[13] IKEDA, R., AND WIDOM, J. Panda: A system for provenance and data. In *Proceedings of the 2nd USENIX Workshop on the Theory and Practice of Provenance (TaPP10)* (2010).

[14] KARVOUNARAKIS, G., IVES, Z. G., AND TANNEN, V. Querying data provenance. In *Proceedings of the 2010 international conference on Management of data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 951–962.

[15] SARMA, A., THEOBALD, M., AND WIDOM, J. LIVE: A lineage-supported versioned DBMS. In *Scientific and Statistical Database Management: 22nd International Conference, Ssdbm 2010, Heidelberg, Germany, June 30-July 2, 2010, Proceedings* (2010), p. 416.

[16] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. A survey of data provenance in e-science. *SIGMOD Record 34* (2005), 31–36.

[17] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. A survey of data provenance techniques. Tech. rep., 2005.

[18] STONEBRAKER, M., BECLA, J., DEWITT, D., LIM, K., MAIER, D., RATZESBERGER, O., AND ZDONIK, S. Requirements for science data bases and SciDB. In *4th Biennial Conference on Innovative Data Systems Research (CIDR09)*.

[19] TALUKDAR, P. P., JACOB, M., MEHMOOD, M. S., CRAMMER, K., IVES, Z. G., PEREIRA, F., AND GUHA, S. Learning to create data-integrating queries. *Proc. VLDB Endow. 1* (August 2008), 785–796.

[20] WIDOM, J. Trio: A system for integrated management of data, accuracy, and lineage. Citeseer.