Tracking and Visualizing Provenance using PostgreSQL

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

December 9, 2010

Abstract

Implemented the ability to track provenance inside postgresql

Implemented a visualizer that makes querying forward and backward provenance easier.

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.

1 Introduction

One practical issue with available database systems is the difficulty of tracing data history. This lack of historical record can make database systems impractical for scientific research. An error in early stage data collection may propagate to future derived tables. Tracking down errors and updating all tuples derived from it can be prohibitively costly or even impossible with current database systems.

The problem of storing and quering previous data history is known as *provenance*. 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.

Our goal is to improve upon existing provenance systems, focusing on the SciDB project. The SciDB proposal [4] lists three requirements for a useful provenance system -

- For any data element, we would like to recover the derivation history.
- If a data element is updated, we would like to trace forward to see other effected elements.
- At any point, we would like to reproduce the construction of the current data.

In particular, we hope to respond to the challenge given by [1].

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

2 Background

There are 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 what processing operations were applied to it. In backward provenance, given some data element, we ask where it originally came from and 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.

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.

3 Related Work

- Trio / LIVE [5, 3]

  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 [2]

  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 [1]

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

4 System Architecture

5 Tracking Provenance inside PostgreSQL

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

7 Experiments

three graphs – CPU, memory and storage overhead of tracking provenance averaged over a certain number of queries over the usual

8 Conclusion and Future Work

Implemented the ability to track provenance inside postgresql

Implemented a visualizer that makes querying forward and backward provenance easier.

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.

References

[1] 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.

[2] 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).

[3] 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.

[4] 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)*.

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