Project Proposal: Provenance in SciDB

Nirmesh Malviya, Vinnie Ramesh, and Alexander Rush
{nirmesh,vramesh, srush}@mit.edu

1 Overview

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 [**?**] 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 [**?**].

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

2 Related Work

- Trio / LIVE [**?**, **?**]

  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 [**?**]

  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 [**?**]

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

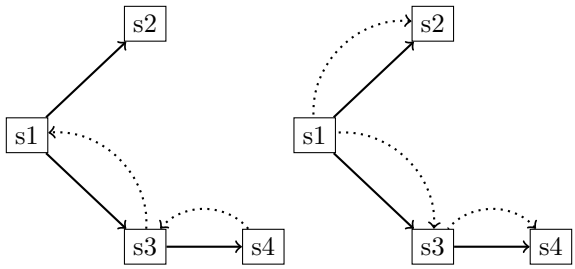

Figure 1: (a) Forward provenance (b) Backward provenance

3 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 3 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.

4 Deliverables

We hope to produce three tangible results from this project.

1. An language extension to SQL for specifying forward and backward provenance queries.
2. A space-efficient algorithm for performing these queries. Our hope is to utilize query logs as opposed to expensive data structures.
3. An possible prototype implementation of this technique.
4. As possible future work, an implementation of the system in SciDB.