A Provenance Model for Manually Curated Data

James Cheney

Joint work with Peter Buneman, Adriane Chapman, and Stijn Vansummeren

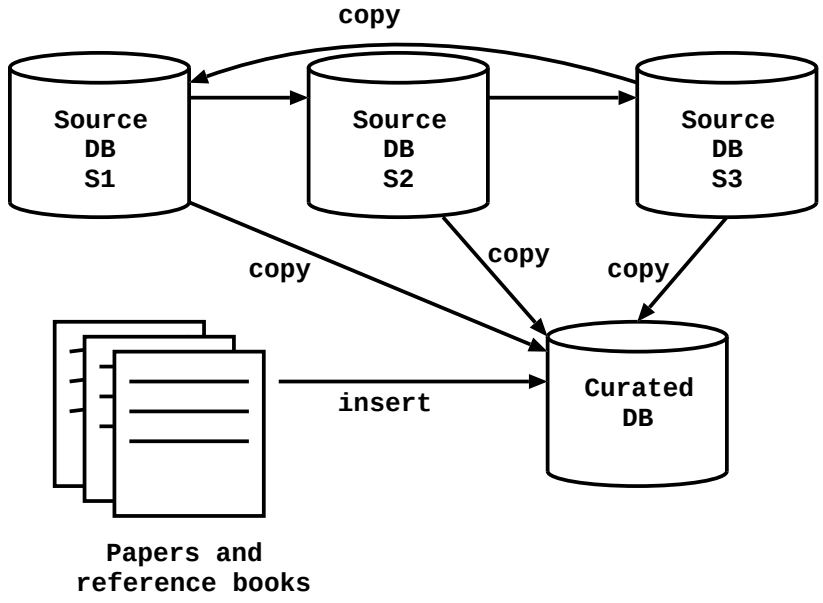
IPAW 2006

May 4, 2006

Chicago, Illinois

Curated databases

- Many scientific databases (especially bioinformatics) are constructed largely “by hand”
- as opposed to by fixed, automatic process such as a view or workflow



- We call such DBs *(manually) curated*

State of practice

- Currently, curators manually add links (e.g. URLs) from copied data to relevant source(s)
- Drawbacks:
  - Time consuming
  - Error prone
  - Danger of link rot (if remote database/Web site changes structure)
  - No support for provenance-based queries
- Can we provide automated support for this process?
- First step: develop a coherent data model for provenance information describing curation process

Constraints

- This is a highly constrained problem: a good solution should
  - be decentralized
  - be data model-independent
  - require minimal changes to curator practice
  - require minimal changes to DB systems
  - be robust in the face of changes to DB structure
  - scale gracefully to multiple cooperating DBs
  - be efficient/scale to large DBs

Constraints

- This is a highly constrained problem: a good solution should
  - be decentralized
  - be data model-independent
  - require minimal changes to curator practices
  - require minimal changes to DB systems
  - be robust in the face of changes to DB structure
  - scale gracefully to multiple cooperating DBs
  - be efficient/scale to large DBs

These are the most important factors for immediate applicability to manually curated data

Prior work

- Most approaches to provenance consider *static* data
  - In databases, provenance investigated for *queries/views* of fixed database
  - In scientific computation, provenance defined for *workflows* that construct new data from existing data
- Prior work does not consider *dynamic* data that can be updated, copied, or deleted

Approach

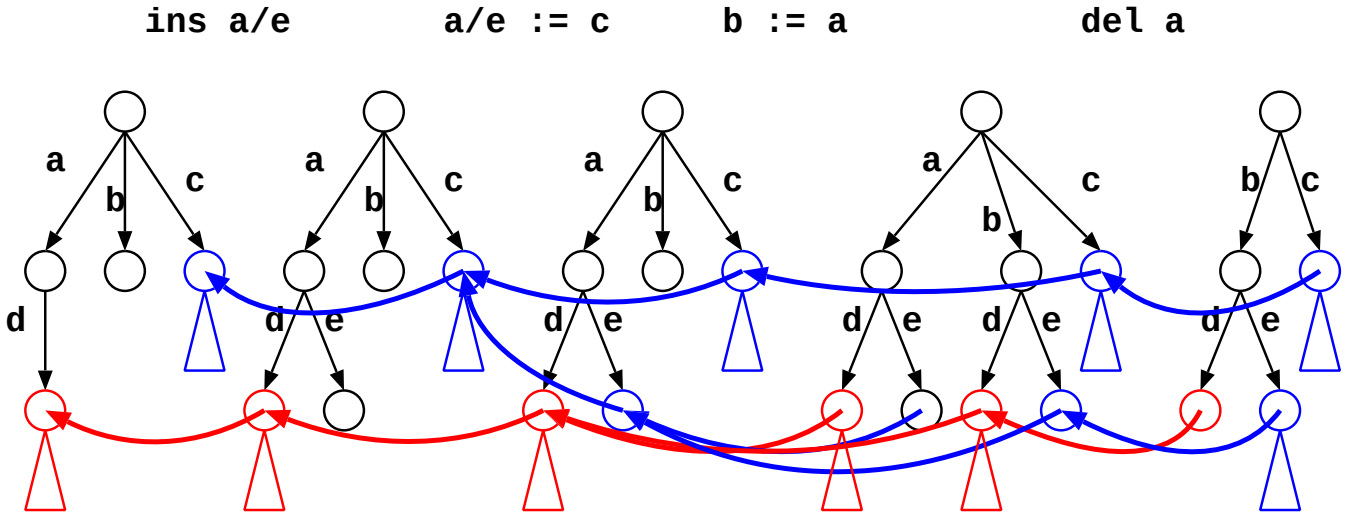
- To simplify matters, we consider only a single dynamic database with several static source databases
- We also view databases *abstractly* as mappings from locations (“keys”) to values
- There are many possible instantiations of this framework:
  - Table names/keys/field names addressing data in an RDBMS
  - XPointers addressing data in XML documents
  - Line/column numbers addressing data in text files
  - (x,y) coordinates addressing data in images
- For concreteness, we’ll deal with paths addressing data in trees.

Update language

- We model the curator's actions in modifying the database as a sequence of "simple" updates
  - **Insertion**: ins p v means "insert the new location p with value v"
  - **Deletion**: del p means "delete the location p"
  - **Copy-paste**: $p := q$ means "copy the data at q into location p"

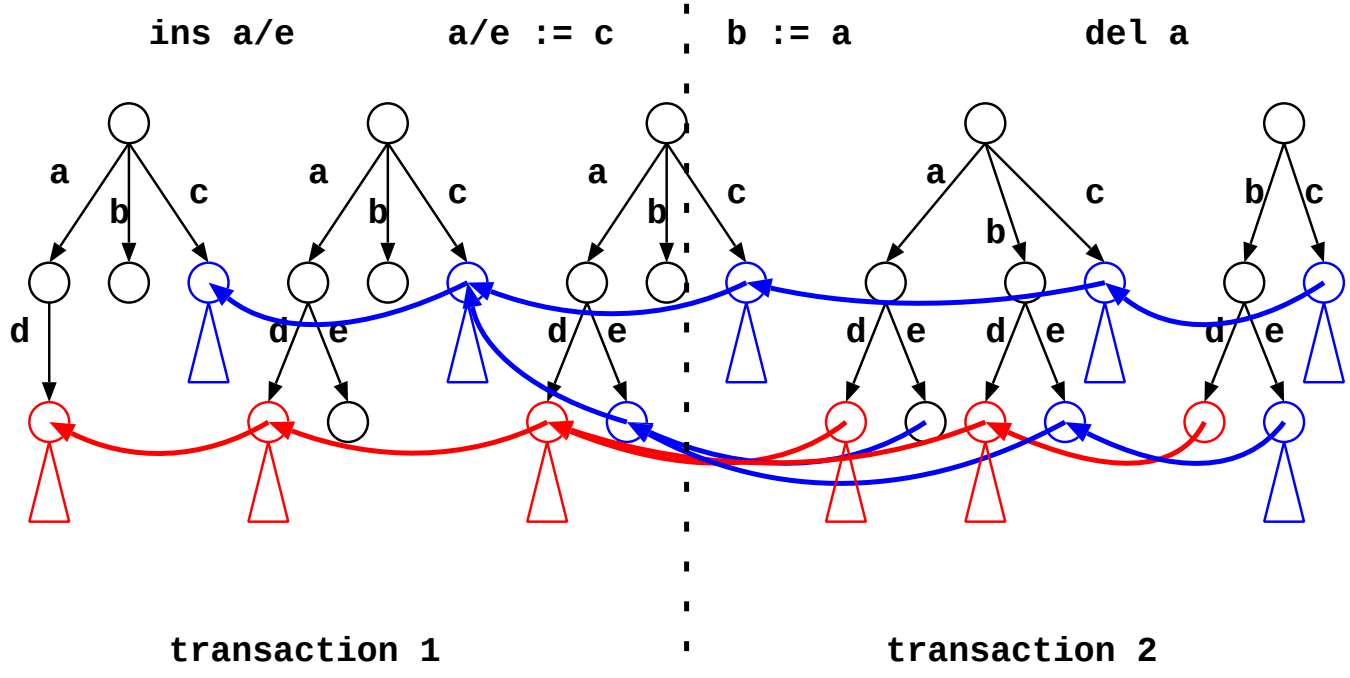
History

- A *history* is a sequence of DB versions, together with *provenance links* indicating where the data in each version “came from”



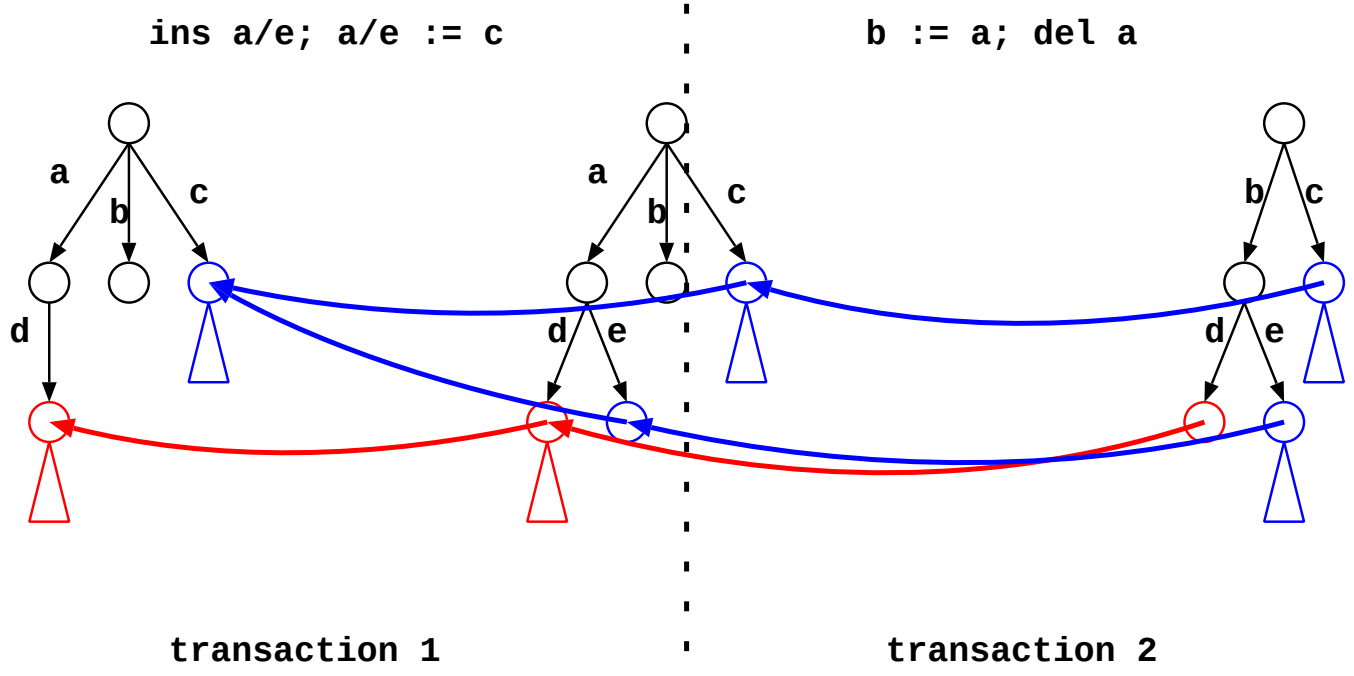
History

- A *history* is a sequence of DB versions, together with *provenance links* indicating where the data in each version “came from”
- We can *refine* a history by grouping update operations into *transactions*



History

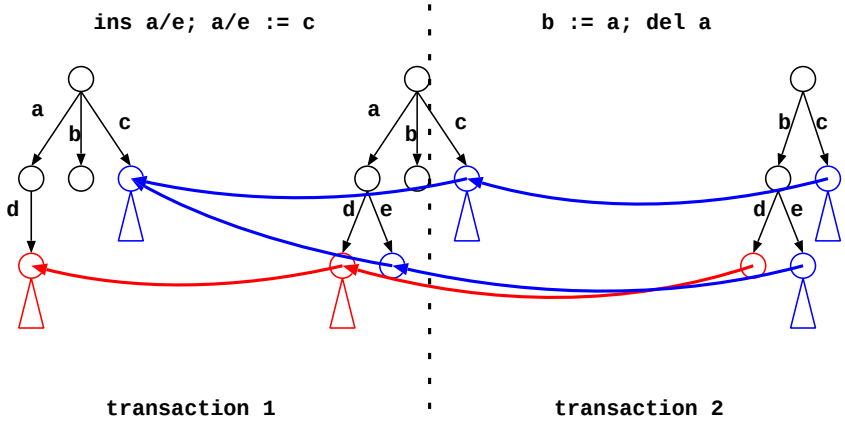
- A *history* is a sequence of DB versions, together with *provenance links* indicating where the data in each version “came from”
- We can *refine* a history by grouping update operations into *transactions*



Provenance data model

- The provenance data can be stored as a table $Prov(Tid, From, To)$

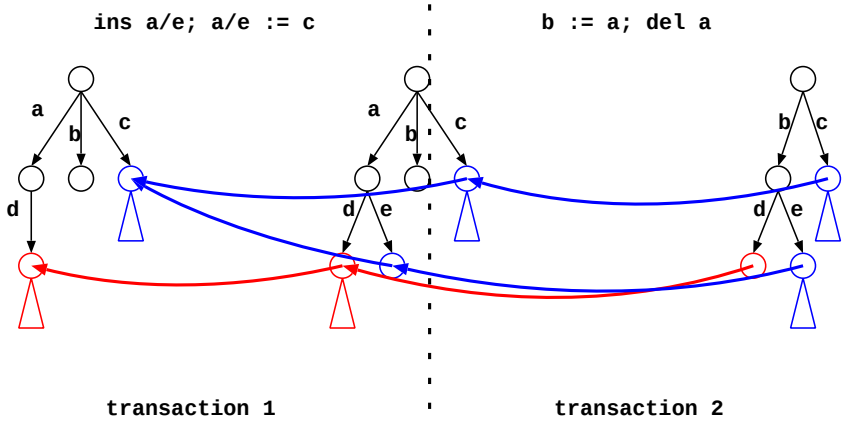
Prov		
Tid	From	To
1	c	c
1	c	a/e
1	a/d	a/d
2	c	c
2	b/d	a/d
2	b/e	a/e
2	a	NULL
2	a/d	NULL
2	a/e	NULL



Provenance data model

- Additional data can be stored in a side table $Trans(Tid, Uid, Time, ...)$

Prov		
Tid	From	To
1	c	c
1	c	a/e
1	a/d	a/d
2	c	c
2	b/d	a/d
2	b/e	a/e
2	a	NULL
2	a/d	NULL
2	a/e	NULL



Trans		
1	jcheney	Tue Apr 18 10:47 AM
2	jcheney	Tue Apr 18 12:37 PM

What can we do with this information?

- Since $Prov$ and $Trans$ are standard relational tables, we can formulate many provenance queries as relational queries.
- Example: "Data was copied from p to q during transaction t"

$$Copied(t, p, q) \leftarrow Prov(t, p, q), p \neq q$$

- Example: "Data at p was inserted during transaction t"

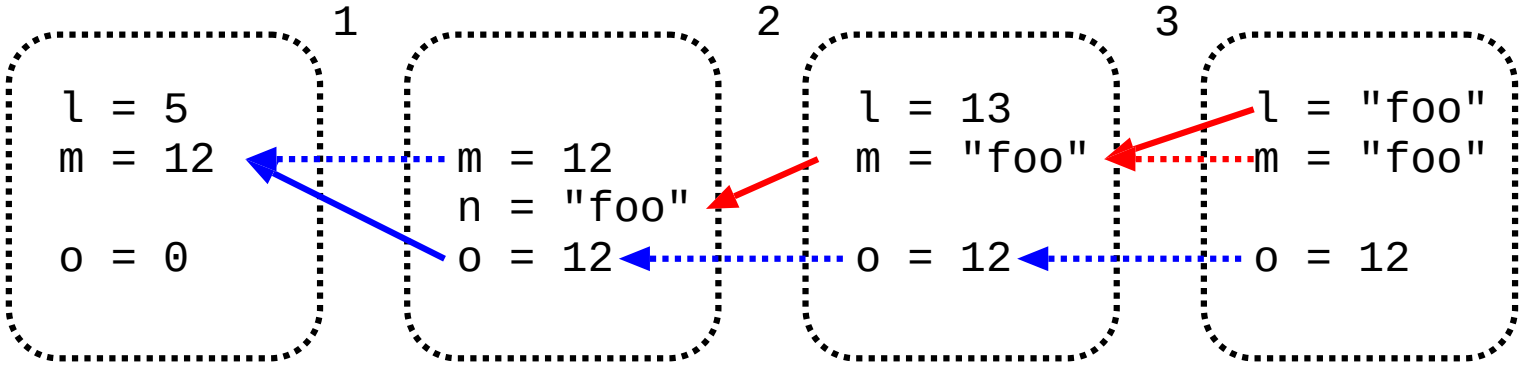
$$Inserted(t, p) \leftarrow Prov(t, NULL, p)$$

A query example

- Example: "Data at l at end of tid was originally inserted by during transaction u"

$$Q(l, tid, tid) \leftarrow Ins(tid, l).$$
$$Q(l, tid, u) \leftarrow Prov(tid, l, m), Q(m, tid - 1, uid).$$

Query: $Q(l, 3, u)$

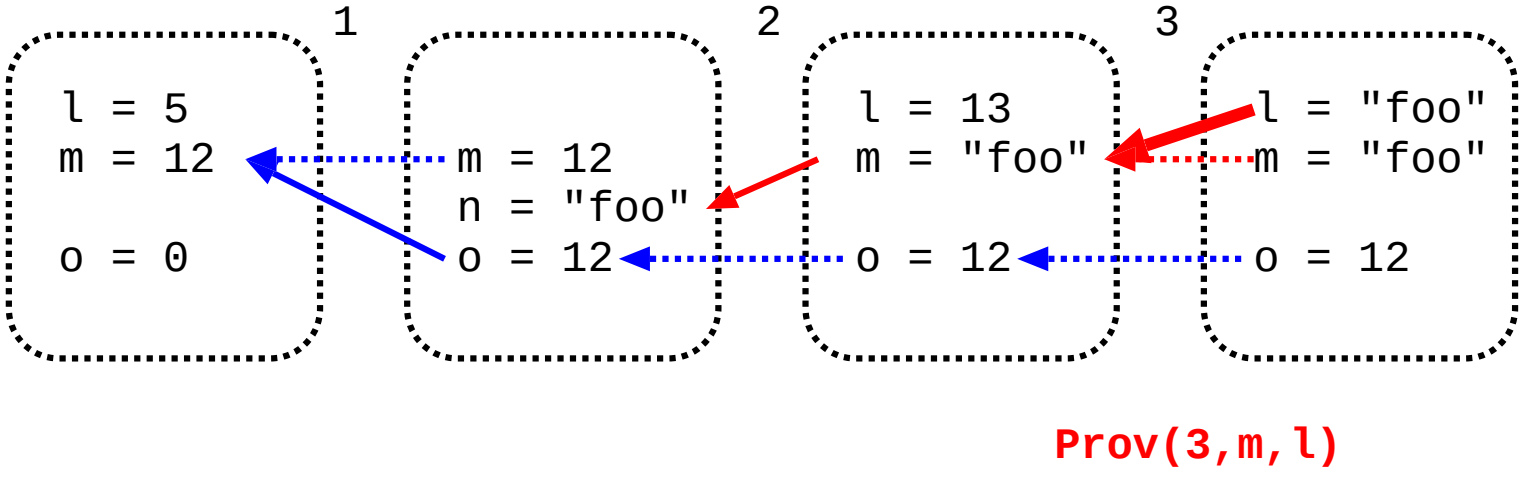


A query example

- Example: "Data at l at end of tid was originally inserted by during transaction u"

$$Q(l, tid, tid) \leftarrow Ins(tid, l).$$
$$Q(l, tid, u) \leftarrow Prov(tid, l, m), Q(m, tid - 1, uid).$$

Query: $Q(l, 3, u)$

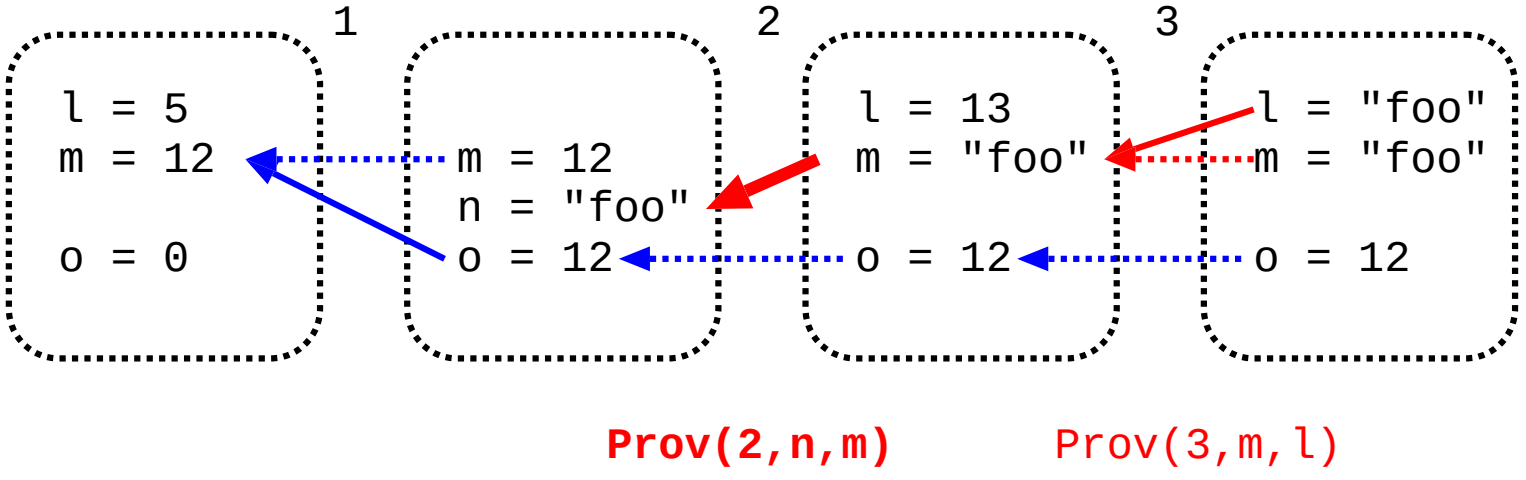


A query example

- Example: "Data at l at end of tid was originally inserted by during transaction u"

$$Q(l, tid, tid) \leftarrow Ins(tid, l).$$
$$Q(l, tid, u) \leftarrow Prov(tid, l, m), Q(m, tid - 1, uid).$$

Query: $Q(l, 3, u)$

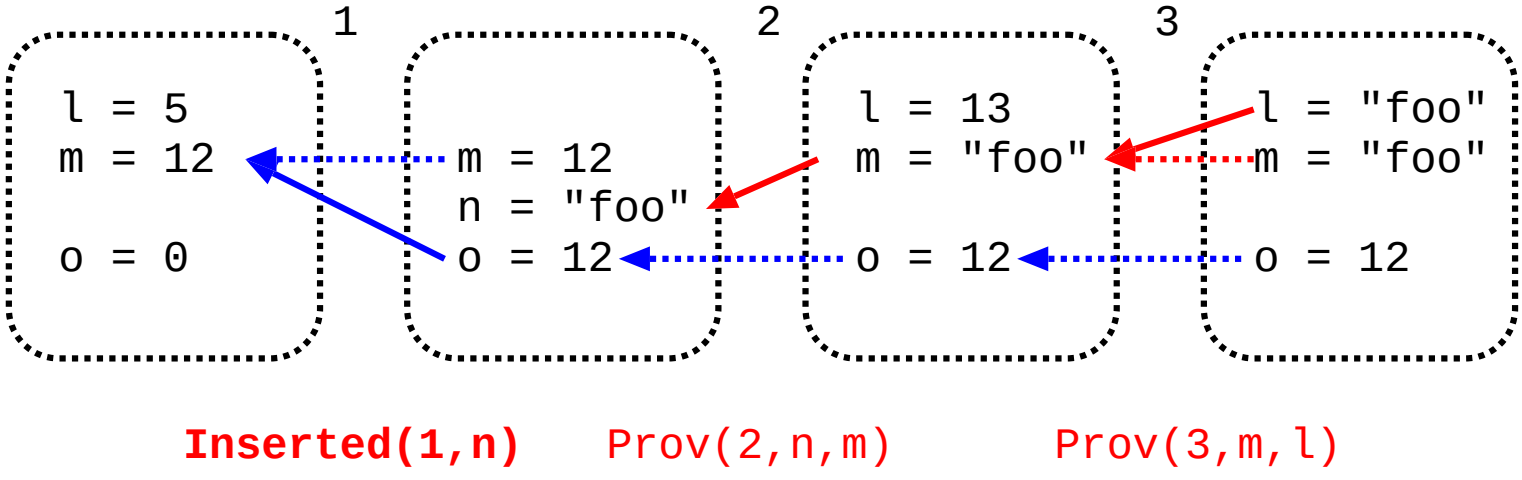


A query example

- Example: "Data at l at end of tid was originally inserted by during transaction u"

$$Q(l, tid, tid) \leftarrow Ins(tid, l).$$
$$Q(l, tid, u) \leftarrow Prov(tid, l, m), Q(m, tid - 1, uid).$$

Query: $Q(l, 3, u) \Rightarrow u = 1$



Challenging issues

- We believe the following issues are the most important for evaluating a solution (in order of importance):
  1. Minimizing the impact of *provenance tracking* on curation performance
  2. Minimizing the space required for storing provenance data
  3. Providing efficient & expressive provenance querying facilities
- since provenance tracking must be performed at every step, but provenance queries are relatively rare.

Example: efficient storage

- The provenance relation defined above contains edges for unchanged data (e.g. $Prov(1, c, c)$, $Prov(2, c, c)$)
- Updates usually modify only *a small part* of the data, so this is wasteful.
- If we explicitly store only provenance edges that involve changes, such unchanged provenance links can always be *inferred*.
- For tree-structured data, further optimizations are possible since the provenance of a child can often be inferred from its parent

Current & future work

- Have implemented a prototype system along with experimental evaluation
  - Proof-of-concept for efficient provenance tracking and storage
- Next steps:
  - Non-intrusive techniques for *collecting provenance* via user browsing/form submission actions
  - Larger scale experiments with more realistic data
  - Techniques for handling “bulk” queries and updates
  - Integrating with “workflow” provenance techniques
  - Combining/querying provenance records involving multiple databases

Conclusions

- Provenance management for “dynamic” data, such as curated databases, is a challenging and under-studied area
- We have developed a general-purpose model for provenance that can be instantiated in many ways
- We have also developed a proof-of-concept implementation for tracking provenance involving tree-structured and relational data
- Feedback welcome!