Provenance in Databases: Past, Current, Future

Peter Buneman  University of Edinburgh
Wang-Chiew Tan  UC Santa Cruz
Goal of this tutorial

- Overview of research developments in provenance in databases.
- Audience: General database audience and people who work with scientific data.
Definition of Provenance (aka Lineage)

From Merriam-Webster online dictionary.

1: Origin, Source
2: the history of ownership of a valued object or work of art or literature
Importance of Provenance

- No different for electronic or digital artifacts.
- A complete record of provenance in scientific computations can help
  - determine the quality and the trust one places on the scientific result
  - typically regarded to be as important as the result itself!
- Scientific workflows:
  - ensure repeatability/verifiability, avoid duplication of efforts, etc.
- Databases:
  - Know the reliability, quality of data (applies to scientific workflows as well)
  - understanding the transport of annotations between data sources and views, probabilistic/uncertain databases, view update and maintenance
Two granularities of provenance

- Workflow (coarse-grained) provenance:
  - Records the complete history (or workflow) of the derivation of some dataset.
  - Typically the case for workflow systems.
    - record the sequence of steps taken in a workflow system to derive the dataset
    - may also involve a record of external devices, such as sensors, cameras, or other data collecting equipments
    - some steps may be treated as “black-boxes”, whose details are not important

- Data (fine-grained) provenance:
  - An account of the derivation of a piece of data in a dataset
  - The focus of this tutorial
Example of Workflow Provenance [Cohen, Cohen-Boulakia, Davidson 06]
A typical question:
- For a given database query Q, a database D, and a tuple t in the output of Q(D), which parts of D “contribute” to t?
- The provenance of tuple (John, D01, Mary) in the output consists of the source facts R(John, D01) and S(D01, Mary) according to the query Q.
- The question could also be applied to an attribute value, a table, or any subtree in hierarchical/tree-like data.
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
Research and Applications of Provenance: A Timeline


Research and Applications of Provenance: A Timeline

- 1990
- 1997

- MONDRIAN: Annotating and Querying Databases through Colors and Blocks. ICDE 2006.
Research and Applications of Provenance: A Timeline


This list of papers is by no means exhaustive!
Data Provenance:
Two general approaches

- **Annotation-based approach**: 
  - changes the transformation from Q to Q’ to carry extra information to the target database
  - Usually, the source database, the definitions of Q and Q’ are no longer needed afterwards. No recomputation required.

- **Non annotation-based approach**: 
  - Q is unchanged
  - Subsequent access to source and access to the definition of Q may be needed. Some recomputation required.
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
Supporting Fine-grained Data Lineage in Database Visualization Environment [Woodruff & Stonebraker 97]

- Build the capability of retrieving fine-grained (data) provenance into a DBMS when results of queries are not annotated with provenance.

- Idea:
  - User registers data processing functions and corresponding inverse functions in the DBMS.
  - Given a specific datum to invert, an inversion planner
    - infers which inverse function to use
    - constructs a plan
    - executes the plan by calling the corresponding sequence of functions within the DBMS.

- Problem: not all functions are invertible.
Weak Inverse and Verification

- A weak inverse $f^w$ of a function $f$ approximates provenance.
- Applies to a larger class of functions.
- A separate verification phase is used to refine the set identified by the weak inverse.

Verification phase: verification function $f^v$ takes as input $f^w(o)$ and $I$ and returns a subset $I'$ of $f^w(o)$.
- $I'$ is complete if the provenance of $o$ w.r.t. $f$ is $\subseteq I'$.
- $I'$ is pure if $I'$ $\subseteq$ the provenance of $o$ w.r.t. $f$.

Problem: Weak inverse and verification function need to be provided by the system administrator.
Tracing the Lineage of View Data in a Warehousing Environment. [Cui, Widom and Wiener 2000]

- Recognized the problem with WS97’s approach.
- Proposed to automatically compute the “inverse” function of f when f is a relational algebra query (with aggregates)
- Defined “an output tuple’s derivation in the source database for an operator” (select, project, join, union, negation, aggregates)

R
\begin{tabular}{l|l}
Emp & Dept \\
\hline
John & D01 \\
Susan & D02 \\
Anna & D04 \\
\end{tabular}

S
\begin{tabular}{l|l|l}
Did & Mgr \\
\hline
D01 & Mary \\
D02 & Ken \\
D03 & Ed \\
\end{tabular}

R \bowtie S
\begin{tabular}{l|l|l}
\hline
Emp & Dept & Mgr \\
\hline
John & D01 & Mary \\
Susan & D02 & Ken \\
\end{tabular}

(John, D01, Mary)’s tuple derivation in R and S according R \bowtie S is R(John, D01) and S(D01, Mary).
Main results

- Defined the view data provenance problem.
- Provenance tracing is invariant for SPJ views
  - The provenance of a tuple is always the same across two equivalent SPJ queries.
- Provenance tracing is not invariant for views with aggregates
  - The provenance of a tuple may not be the same across two equivalent SPJA queries.
- Algorithms for tracing provenance of data in relational views with aggregates in both set and bag semantics.
- Investigated trade-offs in materializing auxiliary views for efficient provenance tracing.
Why and Where: A Characterization of Data Provenance
[Buneman, Khanna and T. 2001]

A further distinction is made between why and where-provenance.
- Why-provenance: “why is a piece of data in the output?”
- Where-provenance: “where is this piece of data copied from?”

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>( R \bowtie S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp</td>
<td>Dept</td>
<td>Did</td>
</tr>
<tr>
<td>John</td>
<td>D01</td>
<td>D01</td>
</tr>
<tr>
<td>Susan</td>
<td>D02</td>
<td>D02</td>
</tr>
<tr>
<td>Anna</td>
<td>D04</td>
<td>D03</td>
</tr>
</tbody>
</table>

(John, D01, Mary)’s tuple derivation in R and S according \( R \bowtie S \) is \( R(John, D01) \) and \( S(D01, Mary) \).

A further distinction is made between why and where-provenance.

- Why-provenance: “why is a piece of data in the output?”
- Where-provenance: “where is this piece of data copied from?”

The Emp value of (John, D01, Mary) in the output is copied from the Emp value of R(John, D01).
Main results

- A framework for describing and understanding provenance in a special tree-like model
  - The location of any piece of data can be uniquely described by a path from the root.
- Data provenance is examined from two perspectives:
  - Why-provenance
  - Where-provenance
- A language specific to the tree model was proposed.
  - Identified a class of queries whose why-provenance is preserved under rewriting.
  - Identified a class of queries whose where-provenance is preserved under rewriting.
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
Debugging Schema Mappings with Routes [Chiticariu and T. 06]

- A schema mapping is a set of logical assertions that describes the correspondence between two schemas.
- It is the key element in data exchange and data integration systems.

Source schema $S$  \hspace{1cm}  Target schema $T$

$\mathcal{M}$

Source instance $I$  \hspace{1cm}  Target instance $J$

- Data Exchange [Haas et al 05, FKMP05]
  - Translate data conforming to a source schema $S$ into data conforming to a target schema $T$ so that the schema mapping $\mathcal{M}$ is satisfied.
Debugging Schema Mappings

- Debugging schema mappings: the process exploring, understanding and refining a schema mapping, at the level of schema mappings, through the use of (test) data

Source schema $S$ \( M \) Target schema $T$

Source instance $I$ \( \xleftarrow{M} \xrightarrow{XSLT/XQuery/Java} \) Target instance $J$

No debugging tools at this level!!

XSLT/XQuery/Java debugging tools.
One of the debugging features: Routes

- Intuitively, a *route* describes the relationship between source and target data through schema mappings.

- Debugging with routes:
  - Examine the routes between individual source and target data elements
  - See how these are constrained by schema mappings
  - Adjust mappings
  - Routes are a form of provenance

- Example next.
Example of a Schema Mapping

**S:** MANHATTAN CREDIT
CardHolders:
  - cardNo
  - limit
  - ssn
  - name
Dependents:
  - accNo
  - ssn
  - name

**T:** FARGO FINANCE
Accounts:
  - accNo
  - creditLine
  - accHolder
Clients:
  - ssn
  - name

**Source-to-target dependencies, \( \Sigma_{st} \):**

\[ m_1 : \text{CardHolders}(cn,l,s,n) \rightarrow \exists L (\text{Accounts}(cn,L,s) \land \text{Clients}(s,n)) \]

\[ m_2 : \text{Dependents}(an,s,n) \rightarrow \text{Clients}(s,n) \]

**Target dependencies, \( \Sigma_t \):**

\[ m_3 : \text{Clients}(s,n) \rightarrow \exists A \exists L (\text{Accounts}(A,L,s)) \]

**Source instance I**

<table>
<thead>
<tr>
<th>CardHolders</th>
<th>123</th>
<th>$15K</th>
<th>ID1</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependents</td>
<td>123</td>
<td>ID2</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

**Target instance J_1**

<table>
<thead>
<tr>
<th>Accounts</th>
<th>123</th>
<th>L_1</th>
<th>ID1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A_2</td>
<td>L_2</td>
<td>ID2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clients</th>
<th>ID1</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID2</td>
<td>Bob</td>
</tr>
</tbody>
</table>
Example Debugging Scenario I

Source instance I

<table>
<thead>
<tr>
<th>Cardholders</th>
<th>123</th>
<th>$15K</th>
<th>ID1</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependents</td>
<td>123</td>
<td>ID2</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

Target instance J

<table>
<thead>
<tr>
<th>Accounts</th>
<th>123</th>
<th>L_1</th>
<th>ID1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients</td>
<td>ID1</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ID2</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

A route for the Accounts tuple

Cardholders

| 123 | $15K | ID1 | Alice |

15K is not copied over to the target

\[ m_1 : \text{Cardholders}(cn,l,s,n) \rightarrow \exists L \ (\text{Accounts}(cn,L,s) \land \text{Clients}(s,n)) \]

\[ m'_1 : \text{Cardholders}(cn,l,s,n) \rightarrow \text{Accounts}(cn,l,s) \land \text{Clients}(s,n) \]

Unknown credit limit?

- Mappings may contain recursive assertions.
- Applies to chains of mappings.
Main results

- How can one compute all routes and concisely display the computed routes?
  - An algorithm that computes \textit{ALL} routes for a set of selected data (source or target, relational or XML).
  - The routes are represented in a route forest, a polynomial size representation.
- An algorithm that computes one route and alternative routes as needed.
- \textbf{Note:} No extra information is carried to the target.

Annotation-based approach for understanding schema mappings.
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
A Polygen Model for Heterogenous Database Systems: A Source Tagging Perspective
[Wang & Madnick 90]

Main results:
- multiple (poly) source (gen) perspective.
- Keep track of which data sources and intermediate data sources were used to generate the data of interest.

- An attribute of a tuple is a triple: <d, o, i>
  - d: data, o: originating sources, i: intermediate sources

- Operational definition on how one can compute the originating sources and intermediate sources of basic relational algebra operators.
  - project, cartesian product, restrict, union, difference
Annotation Propagation Rules of
[Buneman, Khanna and T. 02]

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>A&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>Project</td>
<td>A&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>Join</td>
<td>R&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; R&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>Union</td>
<td>R&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>Rename</td>
<td>R A&lt;sub&gt;1&lt;/sub&gt; A&lt;sub&gt;2&lt;/sub&gt; A&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
</tbody>
</table>
Join

\[
\begin{align*}
\Pi_{\text{Name}, \text{Dept1}, \text{Manager}}(\sigma_{\text{Dept1}=\text{Dept2}}(P \times R))
\end{align*}
\]
On Propagation of Deletions and Annotations through Views
[Buneman, Khanna, T. 02]

- Annotations (not just originating sources) can be propagated from source to output.
- Annotations are propagated based on where data is copied from.
- similar to the way “originating sources” were computed.
- Join instead of cartesian product

- The relationship between locations in the result of a query and input relations is made explicit through the propagation rules.

- A location is a triple: (R, t, A)
  - R is a relation name, t is a tuple in the relation R, A is an attribute of R
  - Points to a column of a tuple in a relation
The Annotation Placement Problem:

- Given a view $V = Q(S)$ and an annotation “A” placed in the view $V$, decide if there is an annotation in the source database $S$ that when propagated to the view, produces no other annotation except “A”.
  - $Q = \text{query}$
  - $S = \text{source database}$
  - “side-effect-free annotation” : an annotation on the source that produces no other annotation except “A” in the view
### Understanding the transport of annotations between source and views

#### NYRestaurants

<table>
<thead>
<tr>
<th>Restaurant</th>
<th>Cost</th>
<th>Type</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peacock Alley</td>
<td>$$$</td>
<td>French</td>
<td>10022</td>
</tr>
<tr>
<td>Bull &amp; Bear</td>
<td>$$$</td>
<td>Seafood</td>
<td>10022</td>
</tr>
<tr>
<td>Pacifica</td>
<td>$</td>
<td>Chinese</td>
<td>10013</td>
</tr>
<tr>
<td>Soho Kitchen &amp; Bar</td>
<td>$</td>
<td>American</td>
<td>10022</td>
</tr>
</tbody>
</table>

#### All Restaurants

<table>
<thead>
<tr>
<th>Restaurant</th>
<th>Cost</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peacock Alley</td>
<td>$$$</td>
<td>French</td>
</tr>
<tr>
<td>Bull &amp; Bear</td>
<td>$$$</td>
<td>Seafood</td>
</tr>
<tr>
<td>Pacifica</td>
<td>$</td>
<td>Chinese</td>
</tr>
<tr>
<td>Soho Kitchen &amp; Bar</td>
<td>$</td>
<td>American</td>
</tr>
</tbody>
</table>

#### Cheap Restaurants

<table>
<thead>
<tr>
<th>Restaurant</th>
<th>Cost</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacifica</td>
<td>$</td>
<td>Chinese</td>
</tr>
<tr>
<td>Soho Kitchen &amp; Bar</td>
<td>$</td>
<td>American</td>
</tr>
</tbody>
</table>

Serves fine French Cuisine in elegant setting. Formal attire.
Extensive wine list!
Yummy chicken curry!!
Main results

- **Theorem:**
  - It is NP-hard to decide if there is a side-effect-free annotation for a PJ query in normal form.
  - There is a polynomial time algorithm for deciding if there is a side-effect-free annotation for SPJU queries which do not simultaneously contain Project and Join operations.

  Many complexity issues go away for key-preserving operations. See CIKM 06 paper by Cong, Fan and Geerts.

- **Corollary:**
  - It is NP-complete to decide if a source tuple is part of a witness for a view tuple.
  - It is NP-complete to decide if a source annotation appears as a particular view annotation.

Later shown to be DP-hard. See DBPL03 paper by T.
An Annotation Management System for Relational Databases (DBNotes) [Bhagwat, Chiticariu, T., Vijayvargiya 04]

- The propagation scheme of [WM90] and [BKT02] is essentially based on where data is copied from.

- The default propagation scheme in DBNotes

- Problem: Propagation of annotations is dependent on the syntax of the query
Default annotation propagation behavior is syntax-dependent

Given two relation schemas: \( R(A,B), S(B,C) \)

- **SELECT** \( r.A, r.B, s.C \)
  FROM \( R r, S s \)
  WHERE \( r.B \equiv_a s.B \)

- **SELECT** \( r.A, s.B, s.C \)
  FROM \( R r, S s \)
  WHERE \( r.B \equiv_a s.B \)

- **SELECT** *
  FROM \( R \) NATURAL JOIN \( S \)

**Answer 1**
\[
\begin{array}{ccc}
1 & 2 & 3 \\
\end{array}
\]

**Answer 2**
\[
\begin{array}{ccc}
1 & 2 & 3 \\
\end{array}
\]

**Answer 3**
\[
\begin{array}{ccc}
1 & 2 & 3 \\
\end{array}
\]

---

BCTV04 Provenance in Databases 37
Two other propagation schemes

- The default-all propagation scheme
  - The “most general” way of propagating annotations
  - Propagates annotations according to where data is copied from in all equivalent formulations of the given query.
  - Theorem: Two equivalent queries (union of SPJ queries) will always propagate annotations in the same way.

- Observation: one may want to control propagation of annotations

- The custom propagation scheme
  - Allows the user to specify where to obtain annotations from.
Syntax of pSQL queries

```
SELECT DISTINCT selectlist
FROM fromlist
WHERE wherelist
PROPAGATE DEFAULT | DEFAULT-ALL |
  r_1.A_1 TO B_1, ..., r_m.A_m TO B_m
```
Example – the CUSTOM and DEFAULT scheme

Given two relation schemas: R(A,B), S(B,C)

```
SELECT r.A AS A_1, r.B AS A_2, s.C AS A_3
FROM   R r, S s
WHERE  r.B = s.B
PROPAGATE r.B TO A_2

SELECT r.A AS A_1, r.B AS A_2, s.C AS A_3
FROM   R r, S s
WHERE  r.B = s.B
PROPAGATE s.B TO A_2
```

R____  S____
1  2   2  3

Ans1
1  2  3

Ans2
1  2  3
Example – the DEFAULT-ALL scheme

Given two relation schemas: $R(A,B)$, $S(B,C)$

```sql
SELECT r.A AS $A_1$, r.B AS $A_2$, s.C AS $A_3$
FROM   $R$ r, $S$ s
WHERE  r.B = s.B
PROPAGATE DEFAULT-ALL
```

- Given a pSQL query $Q$ with DEFAULT-ALL propagation scheme and a database $D$, how can we compute the result of $Q(D)$?
  - There may be infinitely many queries that are equivalent to $Q$.
  - It is therefore impossible execute every equivalent query of $Q$.
- Solution: Compute a finite query basis of the set of all equivalent queries of $Q$. 
Main results

- pSQL query language that supports different propagation schemes.

- An algorithm to compute a finite query basis of a pSQL query with default-all propagation scheme.

- An algorithm that translates pSQL queries (default, default-all, or custom) into one or more SQL queries according to the underlying storage schema that also takes into account storage scheme of annotations.
Mondrian: Annotating and querying databases through colors and blocks
[Geerts, Kementsietsidis, Milano 06]

- Annotations on sets of values
  - Any subset of attributes of a tuple in a relation can be annotated
- A color algebra that can query both values and annotations
  - Complete
    - as expressive as color relational algebra queries (positive relational algebra queries)
- Color Relational Algebra queries:
  - A query that returns a color database when applied on any color database
  - A color database is essentially a relational database with extra columns in each relation for storing annotations
- Minimal
  - every operator in the algebra is necessary
On the Expressiveness of Implicit Provenance in Query and Update Languages [Buneman, Cheney and Vansummeren 07]
(Provenance of these slides: Vansummeren → Buneman → Tan)

- Compares the expressive power of queries that implicitly and explicitly manipulate provenance. Also deals with update languages.
- In *explicit* provenance, the query produces both an answer and a description of provenance.

```sql
SELECT * FROM R WHERE A <> 5
UNION
SELECT A, 7 AS B FROM R WHERE A = 5
UPDATE R SET B = 7 WHERE A = 5
```

These give different (implicit) provenance for the (5,7) tuple.
To model provenance in this case, we need to consider provenance at several levels: data value, tuple, table ...
A complex object model is used for uniformity but here, we show relational examples.
Modeling implicit and explicit provenance

- The relationship between components of the input and output can be described through colors.

\[
Q = \text{SELECT } * \text{ FROM } R \text{ WHERE } A <> 5 \cup \text{SELECT } A, 7 \text{ AS } B \text{ FROM } R \text{ WHERE } A = 5
\]

```
A  B
3  4
5  6
```

```
A  B
3  4
5  7
```
Modelling implicit and explicit provenance

- The relationship between components of the input and output can be described through colors.

\[
Q = \text{UPDATE R SET B=7 WHERE A=5}
\]
Explicit provenance

- In explicit provenance, colors are “first-class” values, and queries can manipulate colors.
- Each value (atom, tuple, table ...) is paired with its color.

- Note that we need a complex object/nested relational algebra to manipulate such structures.
### Types of explicit provenance operations

- **Copying** – if an input and output value have the same color then they are identical.
- **Kind preserving** – if an input and output value have the same color then
  - they are identical if they are atomic.
  - have the same kind (both sets or both tuples,...) otherwise.
- Copying implies kind-preserving.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Input | Copying | Kind-preserving | Neither
Main Results

Modulo some technical conditions:

- The default provenance semantics for query languages, similar to that of [Wang & Madnick 90] and [Bhagwat et al. 04], has the same expressive power as the explicitly definable provenance operations that are copying.

- The default provenance semantics for the update language of [Liefke & Davidson 99] has the same expressive power as the explicitly definable provenance operations that are kind-preserving.
  - [Liefke & Davidson 99] : a complex-object update language that is a natural extension of SQL’s update language
Provenance Semirings
[Green, Karvounarakis and Tannen 07]

- Provenance semirings: Provenance representation can be captured using a semiring of polynomials.

- Definition:
  - Let X be the set of tuple ids of a database instance I. The positive algebra provenance semiring for I is the semiring of polynomials with variables from X and coefficients from \( \mathbb{N} \), denoted by \( (\mathbb{N}[X], +, ., 0, 1) \).
    - \( (\mathbb{N}[X], +, 0) \) and \( (\mathbb{N}[X], ., 1) \) are commutative monoids
    - . is distributive over +
    - 0.a = a.0 = 0
Why-provenance and Provenance polynomials

\[ Q(R) = \Pi_{AC}(\Pi_{AB}(R) \Join \Pi_{BC}(R)) \cup \Pi_{AC}(R) \Join \Pi_{BC}(R) \]

Examples of \( \mathbb{N}[X] \)-relation.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>tuple ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>p</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>r</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>s</td>
</tr>
</tbody>
</table>

Why-provenance

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>{p}</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

Provenance polynomials

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>(2p^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>(pr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>(2r^2 + rs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>(2s^2 + rs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>
## Why-provenance and Provenance polynomials

\[ R \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>tuple ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>p = 2</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>r = 5</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>s = 1</td>
</tr>
</tbody>
</table>

\[ Q(R) = \Pi_{AC}(\Pi_{AB}(R) \Join \Pi_{BC} (R) \cup \Pi_{AC}(R) \Join \Pi_{BC}(R)) \]

### Provenance polynomials

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>expression</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>2p^2</td>
<td>8</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>pr</td>
<td>10</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td>pr</td>
<td>10</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>2r^2 + rs</td>
<td>55</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>2s^2 + rs</td>
<td>7</td>
</tr>
</tbody>
</table>

The semantics of positive relational algebra on \( \mathbb{N}[X] \)-relations factors through the semantics of \( \mathbb{N}[X] \)-relations in provenance semirings.
Main results

- Provenance semirings: Provenance representation can be captured using a semiring of polynomials (with integer coefficients).
- The semantics of positive relational algebra on K-relations for any semiring K factors through the semantics of the same in provenance semirings.
- Results are extended to datalog queries (with recursion) by considering semirings with fixed points.
- Many more ...
Intensional Associations Between Data and Metadata
[Srivastava & Velegrakis 07]

Main idea:
- Use queries to describe the intensional relationship between data and metadata
- Relational join is extended to support joins based on multiple values.
Intensional Associations Between Data and Metadata
[Srivastava & Velegrakis 07]

Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Loc</th>
<th>PhoneLine</th>
<th>CircuitID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFLAC</td>
<td>bus</td>
<td>NJ</td>
<td>332211</td>
<td>245-6983</td>
</tr>
<tr>
<td>J. Lu</td>
<td>res</td>
<td>NY</td>
<td>233439</td>
<td>245-7363</td>
</tr>
<tr>
<td>H. Ford</td>
<td>res</td>
<td>NJ</td>
<td>913308</td>
<td>245-7564</td>
</tr>
<tr>
<td>AMEX</td>
<td>bus</td>
<td>NY</td>
<td>498200</td>
<td>343-5002</td>
</tr>
<tr>
<td>NJC</td>
<td>bus</td>
<td>NJ</td>
<td>392413</td>
<td>981-5002</td>
</tr>
<tr>
<td>BCT</td>
<td>bus</td>
<td>NJ</td>
<td>544361</td>
<td>273-6019</td>
</tr>
</tbody>
</table>

Customers relation is the result of integrating data from several distributed sources. Need to keep provenance information: The source database, IP address, and communication protocol for data in the Customers table.

**Solution 1:** Add three columns for each column in Customers table.
Intensional Associations Between Data and Metadata
[Srivastava & Velegrakis 07]

Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Loc</th>
<th>PhoneLine</th>
<th>CircuitID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFLAC</td>
<td>bus</td>
<td>NJ</td>
<td>332211</td>
<td>245-6983</td>
</tr>
<tr>
<td>J. Lu</td>
<td>res</td>
<td>NY</td>
<td>233439</td>
<td>245-7363</td>
</tr>
<tr>
<td>H. Ford</td>
<td>res</td>
<td>NJ</td>
<td>913308</td>
<td>245-7564</td>
</tr>
<tr>
<td>AMEX</td>
<td>bus</td>
<td>NY</td>
<td>498200</td>
<td>343-5002</td>
</tr>
<tr>
<td>NJC</td>
<td>bus</td>
<td>NJ</td>
<td>392413</td>
<td>981-5002</td>
</tr>
<tr>
<td>BCT</td>
<td>bus</td>
<td>NJ</td>
<td>544361</td>
<td>273-6019</td>
</tr>
</tbody>
</table>

Provenance

<table>
<thead>
<tr>
<th>Rf1</th>
<th>Source</th>
<th>IP</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>NJDB</td>
<td>147.42.7.8</td>
<td>http</td>
</tr>
<tr>
<td>Q2</td>
<td>3State</td>
<td>148.62.1.11</td>
<td>ftp</td>
</tr>
</tbody>
</table>

Solution 2: Keep a separate Provenance table. Use queries to capture data of interest to associated metadata.

Q1: select Name, Type, PhoneLine from Customers where Loc = “NJ”
### Intensional Associations Between Data and Metadata

[Srivastava & Velegrakis 07]

#### Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Loc</th>
<th>PhoneLine</th>
<th>CircuitID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFLAC</td>
<td>bus</td>
<td>NJ</td>
<td>332211</td>
<td>245-6983</td>
</tr>
<tr>
<td>J. Lu</td>
<td>res</td>
<td>NY</td>
<td>233439</td>
<td>245-7363</td>
</tr>
<tr>
<td>H. Ford</td>
<td>res</td>
<td>NJ</td>
<td>913308</td>
<td>245-7564</td>
</tr>
<tr>
<td>AMEX</td>
<td>bus</td>
<td>NY</td>
<td>498200</td>
<td>343-5002</td>
</tr>
<tr>
<td>NJC</td>
<td>bus</td>
<td>NJ</td>
<td>392413</td>
<td>981-5002</td>
</tr>
<tr>
<td>BCT</td>
<td>bus</td>
<td>NJ</td>
<td>544361</td>
<td>273-6019</td>
</tr>
</tbody>
</table>

#### Provenance

<table>
<thead>
<tr>
<th>Rf1</th>
<th>Source</th>
<th>IP</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>NJDB</td>
<td>147.42.7.8</td>
<td>http</td>
</tr>
<tr>
<td>Q2</td>
<td>3State</td>
<td>148.62.1.11</td>
<td>ftp</td>
</tr>
</tbody>
</table>

**Solution 2:** Keep a separate Provenance table. Use queries to capture data of interest to associated metadata.

**Q2:** select Loc, PhoneLine, CircuitID from Customers where Type = “business”
Intensional Associations Between Data and Metadata
[Srivastava & Velegrakis 07]

Customers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Loc</th>
<th>PhoneLine</th>
<th>CircuitID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFLAC</td>
<td>bus</td>
<td>NJ</td>
<td>332211</td>
<td>245-6983</td>
</tr>
<tr>
<td>J. Lu</td>
<td>res</td>
<td>NY</td>
<td>233439</td>
<td>245-7363</td>
</tr>
<tr>
<td>H. Ford</td>
<td>res</td>
<td>NJ</td>
<td>913308</td>
<td>245-7564</td>
</tr>
<tr>
<td>AMEX</td>
<td>bus</td>
<td>NY</td>
<td>498200</td>
<td>343-5002</td>
</tr>
<tr>
<td>NJC</td>
<td>bus</td>
<td>NJ</td>
<td>392413</td>
<td>981-5002</td>
</tr>
<tr>
<td>NJC</td>
<td>bus</td>
<td>NJ</td>
<td>913308</td>
<td>245-7564</td>
</tr>
<tr>
<td>BCT</td>
<td>bus</td>
<td>NJ</td>
<td>544361</td>
<td>273-6019</td>
</tr>
</tbody>
</table>

One can also associate data with data through queries.

Technicians

<table>
<thead>
<tr>
<th>Name</th>
<th>Contact</th>
<th>Company</th>
<th>XJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Farkas</td>
<td>241113</td>
<td>AT&amp;T</td>
<td>Qc</td>
</tr>
<tr>
<td>Gilbert</td>
<td>561391</td>
<td>Verizon</td>
<td>Qt</td>
</tr>
<tr>
<td>Henry</td>
<td>762329</td>
<td>AT&amp;T</td>
<td>Qc1</td>
</tr>
<tr>
<td>James</td>
<td>133865</td>
<td>CISCO</td>
<td>Qt1</td>
</tr>
</tbody>
</table>

Qc1: select CircuitID from Customers where Type="res"

Qt1: select * from Technicians where Company="CISCO"
Selections and joins on Q-values

select  distinct p.Source
from    Provenance p
where   p.Rf1[Name] "AFLAC"

select  *
from    Customers c, XJ x
where   c.Loc="NJ" and c.Type = "bus" and
        x.Qc[CircuitID] [c.CircuitID]

select  p2.*
from     Provenance p1, Provenance p2
where    p1.Source = "NJDB" and
        p1.Rf1[PhoneLine] p2.Rf2[PhoneLine]
Main results

- Use queries to describe the intensional relation between data and metadata.
- Relational join is extended to support joins based on multiple values.
- An implementation on top of an RDBMS and comprehensive experimental evaluation.
Provenance in Curated Databases [Buneman, Chapman and Cheney 06]

- Curated databases – databases that are manually created by scientists by studying and analyzing information from many sources.
  - Not the result of executing a query on existing data sources.
- Current db technology provide little help for managing provenance for such databases.
- **Observation:** Much content of a curated database is derived or copied from other sources, often other curated databases.
Copy-and-Paste Model

- Two assumptions:
  - A curated database can be viewed as a tree.
  - The edges of the tree can be labeled in such a way that a given sequence of labels occurs on at most one path from the root.
- User actions are modeled as a sequence of insert, delete and copy operations.
delete T/c5 from T;
Provenance in Databases


copy S1/a/y into T/c1/y;

<table>
<thead>
<tr>
<th>Tid</th>
<th>Op</th>
<th>Loc</th>
<th>Src</th>
</tr>
</thead>
<tbody>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5</td>
<td>⊥</td>
</tr>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5/x</td>
<td>⊥</td>
</tr>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5/y</td>
<td>⊥</td>
</tr>
<tr>
<td>122</td>
<td>C</td>
<td>T/c1/y</td>
<td>S1/a1/y</td>
</tr>
</tbody>
</table>
Main results:
- notion of a provenance aware transaction
- recording provenance for transactions rather than individual operations.
- hierarchical compression of provenance description
- Experimental validation of compression technique

<table>
<thead>
<tr>
<th>Tid</th>
<th>Op</th>
<th>Loc</th>
<th>Src</th>
</tr>
</thead>
<tbody>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5</td>
<td>⊥</td>
</tr>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5/x</td>
<td>⊥</td>
</tr>
<tr>
<td>121</td>
<td>D</td>
<td>T/c5/y</td>
<td>⊥</td>
</tr>
<tr>
<td>122</td>
<td>C</td>
<td>T/c1/y</td>
<td>S1/a1/y</td>
</tr>
<tr>
<td>123</td>
<td>I</td>
<td>T/c2</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
ULDBs: Databases with Uncertainty and Lineage
[Benjelloun, Sarma, Halevy, Widom 06]

- ULDBs – Uncertainty Lineage Databases.
  - The basis of Trio system (Stanford University)
- Lineage can be used to resolve uncertainty
  - Web search, “crime-solver” example
- Lineage can be used to compute confidences correctly
## Crime-solver Example

### Tables:

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw(witness, car)</th>
<th>ID</th>
<th>Drives(person, car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>(Amy, Mazda)</td>
<td></td>
<td>(Amy, Toyota) ?</td>
</tr>
<tr>
<td>22</td>
<td>(Betty, Honda)</td>
<td>32</td>
<td>(Jimmy, Toyota)</td>
</tr>
<tr>
<td>31</td>
<td>(Jimmy, Mazda)</td>
<td>33</td>
<td>(Billy, Mazda)</td>
</tr>
<tr>
<td>32</td>
<td>(Jimmy, Toyota)</td>
<td>34</td>
<td>(Billy, Honda)</td>
</tr>
</tbody>
</table>

These two tuples cannot coexist.

### Projection:

\[
\Pi_{\text{witness, person}}(\text{Saw} \bowtie \text{Drives})
\]

<table>
<thead>
<tr>
<th>ID</th>
<th>Accuses(witness, person)</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>(Amy, Jimmy) ?</td>
</tr>
<tr>
<td>42</td>
<td>(Amy, Jimmy) ?</td>
</tr>
<tr>
<td>43</td>
<td>(Amy, Billy) ?</td>
</tr>
<tr>
<td>44</td>
<td>(Betty, Billy) ?</td>
</tr>
</tbody>
</table>

\[
\lambda(41,1) = \{(21,1), (31,1)\}
\]

\[
\lambda(42,1) = \{(21,2), (32,1)\}
\]

\[
\lambda(43,1) = \{(21,1), (33,1)\}
\]

\[
\lambda(44,1) = \{(22,1), (34,1)\}
\]
Confidence computation

<table>
<thead>
<tr>
<th>ID</th>
<th>Saw(witness,car)</th>
<th>ID</th>
<th>Drives(person,car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>(Amy, Acura): 0.8</td>
<td>31</td>
<td>(Hank, Acura): 0.6</td>
</tr>
<tr>
<td>22</td>
<td>(Betty, Acura): 0.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- [Dalvi & Suciu 04] showed that a naïve propagation of confidences during query processing may lead to incorrect confidences in the result.
- Query Plan 1: $\Pi_{\text{person}} (\Pi_{\text{car}} (\text{Saw} \Join \text{Drives}))$
  - $\Pr(21 \text{ or } 22) \times \Pr(31) = 0.528$ (correct)
- Query Plan 2: $\Pi_{\text{person}} (\text{Saw} \Join \text{Drives})$
  - $\Pr(\text{Amy, Acura, Hank}) = 0.8 \times 0.6 = 0.48$
  - $\Pr(\text{Betty, Acura, Hank}) = 0.4 \times 0.6 = 0.24$
  - $\Pr(\text{Hank}) = \Pr(61 \text{ or } 62) = 0.6048$ (incorrect !)
- Query plan 3: Use any query plan. Compute confidences later, as needed, based on provenance.
Outline of this tutorial

- Overview of Provenance
  - Workflow (coarse-grained) provenance vs. Data (fine-grained) provenance
- Data provenance
  - A timeline
  - Two approaches:
    - Non annotation-based vs. annotation-based approach
- Non annotation-based approach
  - An application: Debugging
- Annotation-based approach
  - An application: TRIO
- Future Research Directions
Future Research Directions [1/4]

- Combining workflow and data provenance.
  - Formalism for a workflow provenance so that nodes are not just “black boxes”.
  - Extend ideas from data provenance to more general languages (e.g., handle aggregates and loops).
- Reasoning about provenance over “black-boxes”.
  - Given a function $f$ that transforms an input $I$ to an output $O$,
    - how can one reason about the provenance of some output data in $O$ without exactly knowing what $f$ is?
  - Black-box treatment is useful for dealing with complex scripts or programs
Future Research Directions [2/4]

- Explore the relationship between provenance research in databases and techniques in program analysis/software engineering such as dependency analysis and slicing.

- Techniques from program analysis may help generalize provenance techniques in databases.

--- James Cheney, U. of Edinburgh

- Program slicing:
  - A slice of a program is a subset of the statements of a (imperative) program that are "relevant" to the value of a variable at the end. There are various precise characterizations...

\[
\begin{align*}
x &:= y*y; \\
w &:= z*z; \\
z &:= x*x;
\end{align*}
\]

Line 2 does not affect the value of z

\[
\begin{align*}
x &:= y*y; \\
z &:= x*x;
\end{align*}
\]
Future Research Directions [3/4]

- More practical applications of provenance
  - Use provenance to understand and debug a specification
- Uncertain/Probabilistic databases
- In trust policies. [Taylor & Ives 06]
- Workshop on Information Integration
  - Most likely, the next generation of II tools must be able to reason about provenance implicitly or explicitly.
Future Research Directions [4/4]

- Archiving
  - If a database may change, a complete record of provenance requires archiving past states of the database.
  - Especially important for scientific data
    - For verification of scientific results
  - Past work on archiving and archiving scientific datasets
  - More needs to be done
    - Archive databases sitting at possibly different sites, with possibly slightly different versions.
    - Archive databases whose schema may change and still be able to reason about relationships between different versions.
Thank you!