CS 595 - Hot topics in database systems: 

Data Provenance

I. Database Provenance
I.1 Provenance Models and Systems

Boris Glavic

September 10, 2012
1. **Lineage**
   - Provenance Model
   - Compositional Tracing Rules
   - WHIPS Datawarehouse Implementation
   - Applications
   - Recap
Lineage

Provenance Model

Lineage Provenance Model

Rationale

- Similar to why: Models which input tuples are sufficient to derive an output tuple $t$ of query $Q$
  - Attempts to guarantee relevance (no false positives)
  - Attempts to not miss provenance (no false negatives)

Provenance Representation

- List of Sets of Tuples (relations)
- Each relation in provenance is subset of input relation to $Q$

Transformation Language

- Relational Algebra (ASPJ-Set)
  - Aggregation, Selection, Projection, Join, Set-operations
  - Here mostly Set-semantics
Lineage Provenance Model

Implementation

- Stored procedures that …

1. Break query into traceable segments
2. Recursively track back one segment at a time
   - Execute one or more tracing queries to trace provenance of segment
Declarative Definition

Overview
- Defines Provenance of single algebra operators
- Provenance for queries: transitivity

Model
- Operator \( op \) with one or two input relations \( R_1, R_2 \)
  - Keep it simple: limit most parts to \( op(R_1, R_2) \)
- Provenance of tuple \( t \in [[op(R_1, R_2)]] \)
- Subsets \( R_1^*, R_2^* \) of the input relations of the operator
- \( \Rightarrow \) Provenance is List of Sets of Tuples
Provenance Representation - Example

Example

\[ q = \big\langle \text{itemId=}id \big\rangle (\text{sales, items}) \]

\[ \text{sales}^* = \{ s_1 \} \quad \text{items}^* = \{ i_1 \} \]

Example

\[
\begin{array}{|c|c|c|c|}
\hline
\text{shop} & \text{itemId} & \text{id} & \text{price} \\
\hline
\text{t}_1 & \text{Migros} & 1 & 100 \\
\text{t}_2 & \text{Migros} & 3 & 25 \\
\text{t}_3 & \text{Coop} & 3 & 25 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{shop} & \text{itemId} \\
\hline
\text{s}_1 & 1 \\
\text{s}_2 & 3 \\
\text{s}_3 & 3 \\
\hline
\end{array}
\quad
\begin{array}{|c|c|c|}
\hline
\text{id} & \text{price} \\
\hline
\text{i}_1 & 100 \\
\text{i}_2 & 10 \\
\text{i}_3 & 25 \\
\hline
\end{array}
\]

op

R1

R2
Provenance Model

Provenance Representation - Example

Example

\[
q = \triangleleft_{\text{itemId}=\text{id}} (\text{sales}, \text{items})
\]

\[
\text{sales}^* = \{s_1\} \quad \text{items}^* = \{i_1\}
\]
Provenance Model

Declarative Definition - Conditions

Definition (Lineage - Single operator)

- Operator \( op(R_1, R_2) \) + tuple \( t \in [[op(R_1, R_2)]] \)
- A list \( op^{-1}_{(R_1,R_2)}(t) = (R_1^*, R_2^*) \) with \( R_i^* \subseteq R_i \) is provenance of \( t \)
- iff it fulfills following three conditions
Condition (1)

1. Sufficiency

\[ \left[ op(R_1^*, R_2^*) \right] = \{ t \} \]
Condition (1)

1. Sufficiency

- $[[op(R_1^*, R_2^*)]] = \{t\}$
- $\Rightarrow$ Note difference to Why-Provenance (equality vs. set inclusion)
1. Sufficiency

- $[[\text{op}(R^*_1, R^*_2)]] = \{t\}$
- $\Rightarrow$ Note difference to Why-Provenance (equality vs. set inclusion)
- $\Rightarrow$ Also some avoidance of false positives
  - Tuples that derive output tuples $\neq t$
**Provenance Model**

**Condition (1) - Example**

**Example**

```
SELECT *
FROM sales, items
WHERE itemId = id
```

**Example**

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

[[itmld=itmld \(sales, items\)]]
Condition (1) - Example

Example

\[ T_1^* = (\{s_1\}, \{i_1\}) : \text{YES: } [\text{op}(T_1^*)] = \{t_1\} \]

<table>
<thead>
<tr>
<th>sales</th>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>s_2</td>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>s_3</td>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>items</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>i_2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>i_3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[[\text{itemId} = id (sales, items)]]</th>
</tr>
</thead>
<tbody>
<tr>
<td>shop</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Migros</td>
</tr>
<tr>
<td>Migros</td>
</tr>
<tr>
<td>Coop</td>
</tr>
</tbody>
</table>
Condition (1) - Example

Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: NO: $[[\text{op}(T_2^*)]] = \{t_1, t_2\}$
- $i_3$ irrelevant

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>Migros</td>
</tr>
<tr>
<td>$s_2$</td>
<td>Migros</td>
</tr>
<tr>
<td>$s_3$</td>
<td>Coop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

$[[\bowtie_{\text{itemId}=\text{id}} (\text{sales, items})]]$

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Condition (1) - Example

Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: NO: $[[\text{op}(T_2^*)]] = \{t_1, t_2\}$
  - $i_3$ irrelevant
- $T_3^* = (\{s_2\}, \{i_1\})$: NO: $[[\text{op}(T_3^*)]] = \emptyset$

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>Migros</td>
</tr>
<tr>
<td>$s_2$</td>
<td>Migros</td>
</tr>
<tr>
<td>$s_3$</td>
<td>Coop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

$[[\text{itemId}=id \ (sales, items)]]$

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Condition (1) - Example

Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: NO: $[[\text{op}(T_2^*)]] = \{t_1, t_2\}$
  - $i_3$ irrelevant
- $T_3^* = (\{s_2\}, \{i_1\})$: NO: $[[\text{op}(T_3^*)]] = \emptyset$
- $T_4^* = (\{s_1\}, \{i_1, i_2\})$: YES: $[[\text{op}(T_4^*)]] = \{t_1\}$
  - $i_2$ irrelevant

Example

- **sales**
  - shop: Migros, Migros, Coop
  - itemId: 1, 3, 3

- **items**
  - id: 1, 2, 3
  - price: 100, 10, 25

- $[[\text{.itemId}\text{=id } (sales, items)]]$
  - shop: Migros, Migros, Coop
  - itemId: 1, 3, 3
  - id: 1, 3
  - price: 100, 25
2. Contribution (false positives)

- \( \forall i, t^* \in R_i^* : \left[ \left[ \text{op}(\ldots, R_{i-1}^*, \{t^*\}, R_i^*, \ldots) \right] \right] \neq \emptyset \)
- Every tuple contributes something to the result
Condition (2)

2. Contribution (false positives)

- $\forall i, t^* \in R^*_i : [[op(\ldots, R^*_{i-1}, \{t^*\}, R^*_i, \ldots)]] \neq \emptyset$
- Every tuple contributes something to the result
- $\Rightarrow$ Avoid false positives
Condition (2) - Example

Example

```
SELECT * 
FROM sales, items 
WHERE itemId = id
```

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

[[\itemld\=(sales, items)]]
Condition (2) - Example

Example

- \(T_1^* = (\{s_1\}, \{i_1\})\): YES: \([op(T_1^*)]) = \{t_1\}\n
- \(T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})\): YES: e.g., \([op(\{s_1, s_2\}, \{i_3\})] = \{t_2\}\n
- \(T_3^* = (\{s_2\}, \{i_1\})\): NO: \([\{s_2\}, \{i_1\}] = \emptyset\n
- \(T_4^* = (\{s_1\}, \{i_1, i_2\})\): NO: \([op(\{s_1\}, \{i_2\})] = \emptyset\n
\[\triangleleft itemld = id (sales, items)\]

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Lineage

Provenance Model

**Condition (2) - Example**

Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: YES: e.g., $[[\text{op}(\{s_1, s_2\}, \{i_3\})]] = \{t_2\}$
  - $i_3$ irrelevant

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

[[\text{itemId} = \text{id} (sales, items)]]

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Condition (2) - Example

Example

- \( T_1^* = (\{s_1\}, \{i_1\}) \): YES: \( [[op(T_1^*)]] = \{t_1\} \)
- \( T_2^* = (\{s_1, s_2\}, \{i_1, i_3\}) \): YES: e.g., \( [[op(\{s_1, s_2\}, \{i_3\})]] = \{t_2\} \)
  - \( i_3 \) irrelevant
- \( T_3^* = (\{s_2\}, \{i_1\}) \): NO: \( [[\{s_2\}, \{i_1\}]] = \emptyset \)

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Condition (2) - Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: YES: e.g.,
  $[[\text{op}(\{s_1, s_2\}, \{i_3\})]] = \{t_2\}$
  - $i_3$ irrelevant
- $T_3^* = (\{s_2\}, \{i_1\})$: NO: $[[\text{op}(\{s_2\}, \{i_1\})]] = \emptyset$
- $T_4^* = (\{s_1\}, \{i_1, i_2\})$: NO: $[[\text{op}(\{s_1\}, \{i_2\})]] = \emptyset$
  - $i_2$ irrelevant

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
3. Maximality (false negatives)

- \[ \forall R_1', R_2' \subseteq R_1, R_2 : (R_1', R_2') \geq (R_1^*, R_2^*) \land (R_1', R_2') \models (1), (2) \]

- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)
Condition (3)

3. Maximality (false negatives)

- \( \forall R'_1 \subseteq R_1, R'_2 \subseteq R_2 : (R'_1, R'_2) \succeq (R^*_1, R^*_2) \land (R'_1, R'_2) \models (1), (2) \)
- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)
- Why needed or good? Danger of reintroducing false positives
Condition (3)

3. Maximality (false negatives)

- \( \forall R_1', R_2' \subseteq R_1, R_2 : (R_1', R_2') \geq (R_1^*, R_2^*) \land (R_1', R_2') \models (1), (2) \)
- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)
- \( \Rightarrow \) Why needed or good? Danger of reintroducing false positives
- \( \Rightarrow \) Deals with certain cases of double negation
Condition (3) - Example

Example

- $T^*_1 = (\{s_1\}, \{i_1\})$: YES: $[op(T^*_1)] = \{t_1\}$

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

$\triangledownitemId = id(sales, items)$
Condition (3) - Example

Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: NO: e.g., $[[\text{op}(\{s_1\}, \{i_3\})]] = \{t_2\}$
  - $i_3$ irrelevant

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

Example
Example

• \( T_1^* = (\{s_1\}, \{i_1\}) \): YES: \( [[\text{op}(T_1^*)]] = \{t_1\} \)

• \( T_2^* = (\{s_1, s_2\}, \{i_1, i_3\}) \): NO: e.g., \( [[\text{op}(\{s_1\}, \{i_3\})]] = \{t_2\} \)
  • \( i_3 \) irrelevant

• \( T_3^* = (\{s_2\}, \{i_1\}) \): NO: \( [[\{s_2\}, \{i_1\}]] = \emptyset \)

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>3</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Coop</td>
<td>3</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

\([\text{sales}, \text{items}]\)
Example

- $T_1^* = (\{s_1\}, \{i_1\})$: YES: $[[\text{op}(T_1^*)]] = \{t_1\}$
- $T_2^* = (\{s_1, s_2\}, \{i_1, i_3\})$: NO: e.g., $[[\text{op}(\{s_1\}, \{i_3\})]] = \{t_2\}$
  - $i_3$ irrelevant
- $T_3^* = (\{s_2\}, \{i_1\})$: NO: $[[\{s_2\}, \{i_1\}]] = \emptyset$
- $T_4^* = (\{s_1\}, \{i_1, i_2\})$: NO: $[[\text{op}(\{s_1\}, \{i_2\})]] = \emptyset$
  - $i_2$ irrelevant
Condition (3) - Set Difference Drawback

Example

- \( op(R, S) = (R - S) \)
- \( q^{-1}((1)) = (\{(1)\}, \{(2), (3)\}) \)
- Tuples from \( S \) irrelevant

Example

\[
\text{SELECT} \,* \text{ FROM } R \\
\text{EXCEPT} \\
\text{SELECT} \,* \text{ FROM } S
\]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( S )</th>
<th>( Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>r₁</td>
<td>s₁</td>
<td>t₁</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Single Operator Definition - Recap

Definition (Lineage for Single Operators)

- Operator \( op(R_1, R_2) + \) tuple \( t \in [[op(R_1, R_2)]] \)
- A list \( op^{-1}_{(R_1, R_2)}(t) = (R_1^*, R_2^*)\) with \( R_i^* \subseteq R_i \) is provenance of \( t \)
- iff it fulfills following three conditions:

**Definition**

1. \( [[op(R_1^*, R_2^*)]] = \{t\} \)
2. \( \forall i, t^* \in R_i^* : [[op(\ldots, R_{i-1}^*, \{t^*\}, R_i^*, \ldots)]] \neq \emptyset \)
3. \( \forall R_1' \subseteq R_1, R_2' \subseteq R_2 : (R_1', R_2') \geq (R_1^*, R_2^*) \land (R_1', R_2') \models (1), (2) \)
Solution to Definition is unique?

- Exactly one subset of input relations...
- fulfills conditions (1), (2), and (3)?

Proven for ASPJ-Set queries

Proof strategy: For each operator

1. Assume there are two $D_1$ and $D_2$ that fulfill conditions
2. Show that they have to be the same
3. ... or that one does not fulfill the conditions
Uniqueness

Proven for ASPJ-Set queries

Proof strategy: For each operator

1. Assume there are two $D_1$ and $D_2$ that fulfill conditions
2. Show that they have to be the same
3. ... or that one does not fulfill the conditions

Example $\alpha$
Proven for ASPJ-Set queries

**Proof strategy:** For each operator

1. Assume there are two $D_1$ and $D_2$ that fulfill conditions
2. Show that they have to be the same
3. ... or that one does not fulfill the conditions

**Example $\alpha$**

- WLOG $t' \in D_1$ so that $t' \not\in D_2$
Uniqueness

Proven for ASPJ-Set queries

**Proof strategy:** For each operator

1. Assume there are two $D_1$ and $D_2$ that fulfill conditions
2. Show that they have to be the same
3. ... or that one does not fulfill the conditions

**Example $\alpha$**

- WLOG $t' \in D_1$ so that $t' \notin D_2$
- $t'$ has same values in group-by attributes as $t$ (aggregation semantics)
Uniqueness

Proven for ASPJ-Set queries

**Proof strategy**: For each operator

1. Assume there are two $D_1$ and $D_2$ that fulfill conditions
2. Show that they have to be the same
3. ... or that one does not fulfill the conditions

**Example $\alpha$**

- WLOG $t' \in D_1$ so that $t' \notin D_2$
- $t'$ has same values in group-by attributes as $t$ (aggregation semantics)
- $D_1$ is not maximal
Uniqueness Caveat

- New operators may break uniqueness
- Example: Nested subqueries
Extend Definition for Queries

Recursive Definition

Query $q$

1. $q = R$: $q^{-1}_R(t) = \{t\}$

2. $q = op(q_1, \ldots, q_k)$: Contribution
   - Assume $t' \in [[q_i]]$ contributes to $t$ according to single operator definition
   - $+ t^* \in R^*_j$ contributes to $t'$ transitively

3. $\Rightarrow q^{-1}_{(R_1, \ldots, R_m)}(t) = (R^*_1, \ldots, R^*_m)$ with each $t^* \in R^*_i$ fulfills two conditions above
Condition (3) - Double Negation Example

Example

- \( q = R - (S - T) \)

Example

```sql
SELECT * FROM R
EXCEPT
(SELECT * FROM S
EXCEPT
SELECT * FROM T)
```
Condition (3) - Double Negation Example

Example

- \( q = R - (S - T) \)
- \( q^{-1}(t) = (\{r_1\}, \{s_2\}, \{t_1, t_2\}) \)
- relevant tuple \( t_1 \) included!
- \( s_2, t_2 \) irrelevant
Condition (3) - Counter Example

Example

\( q = R - (S - T) \)

Example

```
SELECT * FROM R
EXCEPT
(SELECT * FROM S
EXCEPT
SELECT * FROM T)
```
Condition (3) - Counter Example

Example

- $q = R - (S - T)$
- $q^{-1}(t) = (\{r_1\}, \{\}, \{\})$
- relevant tuple $t_1$ not included!
Provenance Model

Extend Definition for Sets of Outputs

- Provenance of Sets of tuples $T$ instead of single tuple $t$
- Define as union of provenance for each tuple in $T$
- $\Rightarrow$ Union of the individual $R_i^*$!
Sets of Outputs - Example

Example

\[ T = \{ t_1, t_2 \} \]

Example

\[ \exists_{itemId= itemId} (sales, items) \]

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>Migros</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>t_2</td>
<td>Migros</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>t_3</td>
<td>Coop</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>Migros</td>
</tr>
<tr>
<td>s_2</td>
<td>Migros</td>
</tr>
<tr>
<td>s_3</td>
<td>Coop</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_1</td>
<td>100</td>
</tr>
<tr>
<td>i_2</td>
<td>10</td>
</tr>
<tr>
<td>i_3</td>
<td>25</td>
</tr>
</tbody>
</table>
Sets of Outputs - Example

Example

\[ T = \{ t_1, t_2 \} \]

\[ sales^* = \{ s_1, s_2 \} \quad items^* = \{ i_1, i_2 \} \]

Example

\[
\left[\left[ \times_{itemId=id} \right. \left( sales, items \right) \right]\]
\]

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>Migros</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>t_2</td>
<td>Migros</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>t_3</td>
<td>Coop</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
<th>id</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>Migros</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>s_2</td>
<td>Migros</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>s_3</td>
<td>Coop</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>
Provenance Model

Bag Semantics?

- Provenance not unique anymore
- $\Rightarrow$ Derivation Set: set of all derivations
- $\Rightarrow$ Derivation Pool: Union of all derivations
Insensitivity to Query Rewrite

Counter Example

- $q_1 = (R \cup S)$
- $q_2 = (S \cup R)$
- $q_1^{-1}(t) = (R^*, S^*)$
- $q_2^{-1}(t) = (S^*, R^*)$
### Advantages
- Better for a few negation cases
- Straight-forward extension for new operators
- Size limited
- Studied for aggregation + all set ops

### Disadvantages
- Strange behaviour for some negation cases
- Awkward modelling of bag semantics
- Assumption of transitivity
- Non-relational model: lists of relations
Use Notation introduced by authors

- Helps understanding their framework
- Direct translation to the standard notation we will be using throughout the course:
  - Assume instance $I = (R_1, \ldots, R_n)$
  - $q_{(R_1, \ldots, R_m)}^{-1}(t)$
  - $\Leftrightarrow Lin(q, t, I)$
Outline

1. Lineage
   - Provenance Model
   - Compositional Tracing Rules
   - WHIPS Datawarehouse Implementation
   - Applications
   - Recap
Compositional Tracing Rules

Tracing Rules

- Declarative definition nice, but . . .
- how to compute provenance?
Declarative definition nice, but . . .
how to compute provenance?

For single operators:
- Use query to compute provenance in one input relation
- I.e., literally $q_i^{-1}(t) = R_i^*$ run over the inputs
- $\Rightarrow \|number of inputs\|$ queries for each operator
- Prove correctness!
Compositional Tracing Rules

Tracing Rules

- Declarative definition nice, but ...
- how to compute provenance?

For queries:

- One operator at a time to restrictive (= expensive)
- Group operators into segments that can be computed in one query
- Reorder operators (normalization) to ...
  - increase segments size
  - simplify computation
Compositional Tracing Rules

Single operators

Rules

1. $\sigma_C^{-1}(R)(t) = (\{t\})$
2. $\pi_A^{-1}(R)(t) = (\sigma_A=t(R))$
3. $\bowtie_C^{-1}(R_1,\ldots,R_m)(t) = (\{t.R_1\}, \ldots, \{t.R_m\})$
4. $\alpha_{g,agg}^{-1}(R)(t) = (\sigma_G=t.G(R))$
5. $\cup^{-1}(R_1,\ldots,R_m)(t) = (\sigma_{R_1=t}(R_1), \ldots, \sigma_{R_m=t}(R_m))$
6. $-^{-1}(R_1,R_2)(t) = (\{t\}, R_2)$
Example Aggregation

**Query**

```
SELECT shop, sum(price) AS rev
FROM sales
GROUP BY shop
```

$q = \alpha_{\text{shop, sum(price)}}(\text{sales})$

**Example**

<table>
<thead>
<tr>
<th>shop</th>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>lawnmower</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>Shovel</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>Shovel</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>125</td>
</tr>
<tr>
<td>Coop</td>
<td>25</td>
</tr>
</tbody>
</table>
Example Aggregation

**Query**

```sql
SELECT shop, sum(price) AS rev
FROM sales
GROUP BY shop

q = α_{shop,sum(price)}(sales)
q^{-1}(t) = σ_{shop=t.shop}(sales)
```

**Example**

<table>
<thead>
<tr>
<th>shop</th>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>lawnmower</td>
<td>100</td>
</tr>
<tr>
<td>Migros</td>
<td>Shovel</td>
<td>25</td>
</tr>
<tr>
<td>Coop</td>
<td>Shovel</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>125</td>
</tr>
<tr>
<td>Coop</td>
<td>25</td>
</tr>
</tbody>
</table>
Example Aggregation

Query

```
SELECT shop, sum(price) AS rev
FROM sales
GROUP BY shop
```

\[
q = \alpha_{\text{shop}, \text{sum}(\text{price})}(\text{sales}) \\
q^{-1}(t) = \sigma_{\text{shop}=t.\text{shop}}(\text{sales}) \\
q^{-1}(t_1) = \sigma_{\text{shop}=\text{'Migros'}}(\text{sales}) = \{s_1, s_2\}
\]

Example

<table>
<thead>
<tr>
<th>shop</th>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>Migros</td>
<td>100</td>
</tr>
<tr>
<td>s_2</td>
<td>Migros</td>
<td>25</td>
</tr>
<tr>
<td>s_3</td>
<td>Coop</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>shop</th>
<th>rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>125</td>
</tr>
<tr>
<td>Coop</td>
<td>25</td>
</tr>
</tbody>
</table>
Compositional Tracing Rules

Queries

1. Which operator tracing queries can be combined into a single one?
2. How to reorder operators to combine operators into segments that can be traced in one step?
Compositional Tracing Rules

SPJ Query

Query

- \( q = \pi_A(\sigma_C(R_1 \bowtie C_1 \ldots \bowtie C_{m-1} R_m)) \)
- every SPJ query can be rewritten into this form!

Tracing query(ies)

- \( \text{Split}_{R_1, \ldots, R_m}(\sigma_{A=t \land C}(R_1 \bowtie C_1 \ldots \bowtie C_{m-1} R_m)) \)

Split operator

- \( \text{Split}_{A_1, \ldots, A_m}(R) = (\pi_{A_1}(R), \ldots, \pi_{A_m}(R)) \)
ASPJ Query

Query

- \( q = \alpha_{G,\text{agg}}(\pi_A(\sigma_{C(R_1 \bowtie C_1 \cdots \bowtie C_{m-1}} R_m))) \)
- Aggregation operators cannot be moved around freely
- Form segments of \( \alpha - \pi - \sigma - \bowtie \)
  - Probably operators left out
  - Replace with trivial operators

Tracing query(ies)

- \( \text{Split}_{R_1,\ldots,R_m}(\sigma_{G=t.G \land C(R_1 \bowtie C_1 \cdots \bowtie C_{m-1}} R_m)) \)
Compositional Tracing Rules

ASPJ Discussion

- Each ASPJ-segment has to be traced on its own
- Tracing query needs access to inputs of segment
- ⇒ Need to store intermediate results for each segment
- ⇒ Or recompute large parts of the query several times
Queries with Set Operations

**Union**
- Can be integrated into ASPJ-segments to form AUSPJ-segments
  - Union as top-most operation
  - Simply larger splits

**Set Difference**
- Needs single D-segment
Outline

1. Lineage
   - Provenance Model
   - Compositional Tracing Rules
   - WHIPS Datawarehouse Implementation
   - Applications
   - Recap
Approach

- Tracing rules nice, but . . .
- how to implement computation?
- For views in data warehouse
Query Normalization + Segmentation

- User defines view
- Parse Query
- Reorder operators to normalize
- Split into segments
- Create stored procedures that update intermediate results on base table update
Tracing Queries

- How to implement $SPLIT$ operator?
  - Store results of $SPLIT$ input
  - One query to extract everything from stored $SPLIT$ input

- Other tracing queries
  - $\Rightarrow$ Use normal query functionality of DBMS
Discussion

Advantages
- Computable in middleware
- Query-able representation
  - compare Why-provenance

Disadvantages
- Does not model which tuples used together
- Strange semantics for
  - Bag semantics
  - Set difference
Outline

1. Lineage
   - Provenance Model
   - Compositional Tracing Rules
   - WHIPS Datawarehouse Implementation
   - Applications
   - Recap
**Problem**

- Consider SPJ view $V(D)$ over instance $D$ (set semantics)
- How to delete $t$ from view?
- $\Delta D$: instance update that causes $t$ to disappear from view
- $\Delta D$ is **exact** if only $V(D - \Delta D) = \Delta V = V(D) - \{t\}$
- $E = \Delta V - \{t\}$: Side-effect

**Idea**

- Provenance assumed to be available
  - In contrast to view maintenance could be ok compute on the fly!
- Use provenance to help us determine which inputs to delete from input
CREATE VIEW ActiveCS AS
SELECT DISTINCT E.Name AS Emp
FROM Employee E, Project P, Assigned A
WHERE E.Id = A.Emp AND P.Name = A.Project
AND Dep = CS

<table>
<thead>
<tr>
<th>Employee</th>
<th>Project</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>e1</td>
<td>1</td>
<td>Peter</td>
</tr>
<tr>
<td>e2</td>
<td>2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>e2</td>
<td>3</td>
<td>Michael</td>
</tr>
<tr>
<td>p1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>Webpage</td>
<td>2</td>
</tr>
<tr>
<td>a4</td>
<td>Fire CS</td>
<td>3</td>
</tr>
</tbody>
</table>
Deletion Propagation Example

\[ \nu = \pi_{E.\text{Name}}(E \bowtie E.\text{Id}=A.\text{Emp} \bowtie A.\text{Project}=P.\text{Name} P) \]

### Example

#### Employee

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
</tr>
<tr>
<td>2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
</tr>
</tbody>
</table>

#### Project

<table>
<thead>
<tr>
<th>Name</th>
<th>Dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>CS</td>
</tr>
<tr>
<td>Webpage</td>
<td>CS</td>
</tr>
<tr>
<td>Fire CS</td>
<td>HR</td>
</tr>
</tbody>
</table>

#### Assigned

<table>
<thead>
<tr>
<th>Project</th>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>1</td>
</tr>
<tr>
<td>Server</td>
<td>2</td>
</tr>
<tr>
<td>Webpage</td>
<td>2</td>
</tr>
<tr>
<td>Fire CS</td>
<td>3</td>
</tr>
</tbody>
</table>
View Deletion - Approach

Preliminaries
- Candidates for deletion = provenance $t$
- $v = \pi_A(\sigma_C(R_1 \bowtie \ldots \bowtie R_n))$

Approach
- Deleting whole provenance?
  - $\Rightarrow$ probably unnecessary side-effects
- Which (if any) subset of provenance can be deleted without side-effects?
- Part of provenance may be exclusive to $t$?
  - Deleting exclusive tuple $\Rightarrow$ no side-effects
Applications

View Deletion - Exclusive Lineage

Definition (Exclusive Lineage)

- Tuples that are only in lineage of \( t \)
- \( t_i \in R_{i}^{**} : t_i \in R_i \land V(R_1, \ldots \{t_i\} \ldots , R_n) = \{t\} \)
- If deleting exclusive lineage is solution \( \Rightarrow \) it is exact
Deletion Propagation Example

Example

\[ v = \pi_{E.\text{Name}}(E \bowtie_{E.\text{Id}=A.\text{Emp}} A \bowtie_{A.\text{Project}=P.\text{Name}} P) \]

\[ E^* = \{e_2\} \quad A^* = \{a_2, a_3\} \quad P^* = \{p_1, p_2\} \]
Deletion Propagation Example

Example

\[ v = \pi_{E.\text{Name}} (E \bowtie E.\text{Id}=A.\text{Emp} \ A \bowtie A.\text{Project}=P.\text{Name} \ P) \]

\[ E^* = \{e_2\} \]
\[ A^* = \{a_2, a_3\} \]
\[ P^* = \{p_1, p_2\} \]

\[ E^{**} = \{e_2\} \]
\[ A^{**} = \{a_2, a_3\} \]
\[ P^{**} = \{p_2\} \]
Deletion Propagation Example

Example

\[ \nu = \pi_{E.\text{Name}}(E \bowtie E.\text{Id}=A.\text{Emp} \ A \bowtie A.\text{Project}=P.\text{Name} \ P) \]

\[ E^* = \{e_2\} \quad A^* = \{a_2, a_3\} \quad P^* = \{p_1, p_2\} \]

\[ E^{**} = \{e_2\} \quad A^{**} = \{a_2, a_3\} \quad P^{**} = \{p_2\} \]

Exact solution!

<table>
<thead>
<tr>
<th>Employee Id</th>
<th>Name</th>
<th>Project Name</th>
<th>Dep</th>
<th>Assigned Project Id</th>
<th>Assigned Employee Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1</td>
<td>Peter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2</td>
<td>2</td>
<td>Gertrud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2</td>
<td>3</td>
<td>Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>Server</td>
<td>CS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>Webpage</td>
<td>CS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>Fire CS</td>
<td>HR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a1</td>
<td>Server</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>Server</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>Webpage</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a4</td>
<td>Fire CS</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
View Deletion - Algorithm

1. Compute lineage
2. Compute exclusive lineage
   - For each $t_i \in R_i^*: \{t\} = v(R_1, \ldots \{t_i\}, \ldots, R_n) \Rightarrow t_i \in R_i^{**}$
3. Delete exclusive lineage: Delete $t$? $\Rightarrow$ done!
4. For each $i$: if deletion $R_i^*$. . .
   - causes exact deletion of $t$? $\Rightarrow$ done!
   - otherwise store side-effect size
5. For each $k$: enumerate subsets of Lineage of size $k$
   - For each subset: delete and compute side-effects
Lineage does not model which tuples were used together

- Harder to check whether we have solution
- Harder to check for side-effects
- Conditional exclusive lineage? Hard with lineage
- View maintenance similar issues
Outline

1. Lineage
   - Provenance Model
   - Compositional Tracing Rules
   - WHIPS Datawarehouse Implementation
   - Applications
   - Recap
Lineage Model

- **Representation**: List of relations
- **Declarative Definition**:
  - For single algebra operators
  - **Sufficiency**
  - **Maximality** (Avoid false negatives)
  - **Minimality** (Avoid false positives)
  - **Relevance** (Avoid irrelevant tuples)
- **Compositional Rules**:
  - Normalization + Segmentation
- Limitations
WHIPS Implementation

- Normalize query
- Break into segments that can be traced in one step
- Trace from result to source tables
- Tracing steps implemented as stored procedures
- Choice which intermediate views to create
## Provenance Model Comparison

<table>
<thead>
<tr>
<th>Property</th>
<th>Why</th>
<th>Lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>Set of Set of Tuples</td>
<td>List of Set of Tuples</td>
</tr>
<tr>
<td>Language Support</td>
<td>USPJ</td>
<td>ASPJ-Set</td>
</tr>
<tr>
<td>Semantics</td>
<td>Set</td>
<td>Set + Bag*</td>
</tr>
<tr>
<td>Variants</td>
<td>Wit, Why, IWhy</td>
<td>Set/Bag</td>
</tr>
<tr>
<td>Design Principles</td>
<td>Sufficiency - No false positives</td>
<td>Sufficiency + No false negatives + no false positives</td>
</tr>
<tr>
<td>Systems</td>
<td>-</td>
<td>WHIPS</td>
</tr>
<tr>
<td>Insensitivity</td>
<td>Yes - No - Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Literature I

Yingwei Cui.
Lineage Tracing in Data Warehouses.

Yingwei Cui and Jennifer Widom.
Lineage Tracing for General Data Warehouse Transformations.

Yingwei Cui and Jennifer Widom.
Run-time Translation of View Tuple Deletions using Data Lineage.

Yingwei Cui, Jennifer Widom, and Janet L. Wiener.
Tracing the Lineage of View Data in a Warehousing Environment.

Yingwei Cui and Jennifer Widom.
Practical Lineage Tracing in Data Warehouses.

Yingwei Cui and Jennifer Widom.
Storing Auxiliary Data for Efficient Maintenance and Lineage Tracing of Complex Views.
In DMDW ’00: Proceedings of the 2th International Workshop on Design and Management of Data Warehouses, 2000.
Yingwei Cui and Jennifer Widom.
Lineage Tracing in a Data Warehousing System.
In ICDE ’00: Proceedings of the 16th International Conference on Data Engineering (demonstration),