CS 595 - Hot topics in database systems: **Data Provenance**

1. Introduction to Data Provenance

Boris Glavic

August 29, 2012
Outline

1. Origin of Term
2. Relational Algebra Primer
3. What is Provenance?
4. Types of Provenance Information
5. Use Cases and Application Domains
6. Provenance Generation, Storage, and Querying
7. Recap
Data Provenance

Information about the creation process and origin of data
Why do we call it Provenance?

Origin of the Term
- From art dealing

Alternative Terms
- Lineage
- Data Pedigree
Why do we call it Provenance?

**Origin of the Term**
- From art dealing

**Alternative Terms**
- Lineage for kings
- Data Pedigree
Why do we call it Provenance?

Origin of the Term
- From art dealing

Alternative Terms
- Lineage for kings
- Data Pedigree for dogs
Why do we call it Provenance?

**Origin of the Term**
- From art dealing for pieces of art

**Alternative Terms**
- Lineage for kings
- Data Pedigree for dogs
Provenance in Art

Given a piece of art

- How do we know . . .
  - if it is authentic?
  - who created it?
  - if it has been altered?

Example

Jan Van Eyck - Arnolfini Portrait
Provenance in Art

Provenance

- French *provenir*, "to come from"
- Chronology of the ownership or location of an historical object

Example

Jan Van Eyck - Arnolfini Portrait

1434 - Painting dated by van Eyck; presumably owned by the sitters.

before 1516 - In possession of Don Diego de Guevara (d. Brussels 1520), a Spanish career courtier of the Habsburgs (himself the subject of a fine portrait by Michael Sittow in the National Gallery of Art). He lived most of his life in the Netherlands, and may have known the Arnolfinis in their later years. By 1516 he had given the portrait to Margaret of Austria, Habsburg Regent of the Netherlands.

1516 - Painting is the first item in an inventory of Margaret's paintings, made in her presence at Mechelen. The item says (in French): "a large picture which is called Hernoul le Fin with his wife in a chamber, which was given to Madame by Don Diego, whose arms are on the cover of the said picture; done by the painter Johannes." A note in the margin says "It is necessary to put on a lock to close it: which Madame has ordered to be done."

1523-4 - In another Mechelen inventory, a similar description, this time the name of the subject is given as "Arnoult Fin".

1558 - In 1530 the painting was inherited by Margaret's niece Mary of Hungary, who in 1556 went to live in Spain. It is clearly described in an inventory taken after her death in 1558, when it was inherited by Philip II of Spain. A painting of two of his young daughters commissioned by Philip clearly copies the pose of the figures (Prado).[1]

1599 - A German visitor saw it in the Alcazar Palace in Madrid. Now it had verses from Ovid painted on the frame: "See that you promise: what harm is there in promises? In promises anyone can be rich." It is very likely that Velázquez knew the painting, which may have influenced his Las Meninas, which shows a room in the same palace.

1700 - In an inventory after the death of Carlos II it was still in the palace, with shutters and the verses from Ovid.

1794 - Now in the Palacio Nuevo in Madrid.

1816 - The painting is now in London, in the possession of Colonel James Hay, a Scottish soldier. He claimed that after being seriously wounded at the Battle of Waterloo the previous year, the painting hung in the room where he convalesced. He fell in love with it, and persuaded the owner to sell. More relevant to the real facts is no doubt Hay's presence at the Battle of Vitoria, where a large coach loaded by King Joseph Bonaparte with easily portable artworks from the Spanish royal collections was first plundered by British troops, before what was left was recovered by their commanders and returned to Madrid. Hay offered the painting to the Prince Regent, later George IV of England, via Sir Thomas Lawrence. The Prince had it on approval for two years at Carlton House before returning it in 1818.
Outline

1. Origin of Term
2. Relational Algebra Primer
3. What is Provenance?
4. Types of Provenance Information
5. Use Cases and Application Domains
6. Provenance Generation, Storage, and Querying
7. Recap
Relational Algebra

- Formalizes queries over relational data
- Is an algebra over relations
  - Types of operators
  - An operator produces a single output relation from one or more input relations.
- Relations
  - A relation is a set of tuples with the same schema
  - Tuple is a list of values
- Composable
  - Output of an operator can be used as input to another operator!
  - Can build complex queries by combining simple operators
Types of Operators

- Selection $\sigma_C$
- Projection $\pi_A$
- Joins
  - Theta-join $\bowtie_C$
  - Cross-product $\times$
  - Outer joins $\bowtie, \bowtie\cap, \bowtie\cap$
- Aggregation with group-by $\alpha_{agg,G}$
- Set Operations
  - Union $\cup$
  - Intersection $\cap$
  - Set difference $-$
- Relation access $R$
Selection

**Signature**

- $\sigma_C(R)$
- $R$ is the input relation
- $C$ is a logical condition (selection condition)
  - Logical operators: AND ($\land$), OR ($\lor$), and NOT ($\neg$)
  - Comparison operators:
    - E.g., equality ($=$) or smaller equals ($\leq$)
    - Refer to constants and attributes
  - Calls to functions?
  - E.g., $name = 'Peter' \land salary \leq 1000$
Selection

**Definition**

\[
[[\sigma_C(R)]] = \{ t \mid t \in R \land t \models C \}\]
Selection

Example

\[ Employee = \{(Peter, 100), (Heinz, 4000)\} \]

\[ [[\sigma_{name='Peter'}(Employee)] = \{(Peter, 100)\} \]
**Projection**

---

**Signature**

- $\pi_A(R)$
- $R$ is the input relation
- $A$ is a list of *projection expressions*
  - Attributes form $R$
  - Functions calls and operators
    - E.g., $a + b$
  - Renaming, $a \rightarrow b$
Projection

**Definition**

\[ [[\pi_A(R)]] = \{ t \mid \exists u \in R \land u.A = t \} \]
Projection

Example

\[ \text{Employee} = \{ (Peter, 100), (Heinz, 4000) \} \]
\[ \pi_{\text{salary}}(\text{Employee}) = \{ (100), (4000) \} \]
### Join

**Signature**

- \( R \bowtie_C S \)
- \( R, S \) are the input relations
- \( C \) is a logical condition (join condition)
  - Same as selection condition
  - Only equality conditions and \( \land \Rightarrow \text{Equi-join} \)
Join

Definition

$$[[R \bowtie_C S]] = \{ t \bowrightarrow t' \mid t \in R \land t' \in S \land t \bowrightarrow t' \models C \}$$
Join

Example

\[ \text{Employee} = \{(Peter, 1), (Heinz, 2)\} \]
\[ \text{Department} = \{(1, CS), (2, HR)\} \]
\[ [[\text{Employee} \bowtie_{\text{depId}=\text{Id}} \text{Department}]] = \{(Peter, 1, 1, CS), (Heinz, 2, 2, HR)\} \]
Outer Joins: Left-outer Join

**Signature**

- $R \join_{C} S$
- $R, S$ are the input relations
- $C$ is a logical condition (*join condition*)
Outer Joins: Left-outer Join

**Definition**

\[
[[R \bowtie C \ S]] = \{(t \uparrow t') \mid t \in R \land t' \in S\}
\cup \{((t_1 \uparrow \text{null}(S)) \mid t \in R \land (\forall t' \in S : (t \uparrow t') \models C)\}
\]
Outer Joins: Left-outer Join

Example

\[
\text{Employee} = \{(Peter, 1), (Heinz, null)\}
\]

\[
\text{Department} = \{(1, CS), (2, HR)\}
\]

\[
[[\text{Employee} \leftnatural_{\text{depId}=\text{Id}} \text{Department}]] = \{(Peter, 1, 1, CS), (Heinz, null, null, null)\}
\]
## Aggregation

### Signature

- $\alpha_{agg,G}(R)$
  - $R$ is the input relation
  - $agg$ list of aggregation functions
    - E.g., $\text{sum}(a)$ if $a$ attribute of $R$
  - $G$ is a list of group-by expressions
    - Attributes
    - Operators and function expressions
Aggregation

Definition

\[
[[\alpha G, \text{agg}(R)]] = \{(t.G, res_1, \ldots, res_m) \mid t \in R \\
\land \forall i \in \{1, m\} : res_i = agg_i(\pi_{b_i}(\sigma_{G=t.G}(R)))\}
\]

- \(b_i\) expression used as aggregation function input
  - E.g., \(a\) for \(\text{sum}(a)\)
- \(res_i\) is result of computing aggregation function for a tuple
Aggregation

Example

\[
Employee = \{(Peter, 1, 3000), (Heinz, 2, 4000), (Jule, 1, 2000)\}
\]

\[
\left[\alpha_{\text{sum(salary), depId}}\right] = \{(5000, 1), (4000, 2)\}
\]
Union

**Signature**

- $R \cup S$
- $R$ and $S$ are the input relations
- $R$ and $S$ have to have the same schema
Union

**Definition**

\[
[[R \cup S]] = \{ t \mid t \in R \lor t \in S \}
\]
### Union

**Example**

\[
\text{Employee} = \{ (Peter), (Heinz), (Jule) \} \\
\text{Manager} = \{ (Peter), (Gertrud) \} \\
[[\text{Employee} \cup \text{Manager}]] = \{ (Peter), (Heinz), (Jule), (Getrud) \}
\]
Intersection

**Signature**

- $R \cap S$
- $R$ and $S$ are the input relations
- $R$ and $S$ have to have the same schema
Intersection

Definition:

\[
[[R \cap S]] = \{ t \mid t \in R \land t \in S \}
\]
Intersection

Example

\[
\text{Employee} = \{(Peter), (Heinz), (Jule)\}
\]
\[
\text{Manager} = \{(Peter), (Gertrud)\}
\]
\[
[\text{Employee} \cap \text{Manager}] = \{(Peter)\}
\]
Set Difference

Signature

- \( R - S \)
- \( R \) and \( S \) are the input relations
- \( R \) and \( S \) have to have the same schema
Set Difference

Definition

\[[R - S]] = \{ t \mid t \in R \land t \notin S \}\]
Set Difference

Example

\[
\text{Employee} = \{(Peter), (Heinz), (Jule)\}
\]

\[
\text{Manager} = \{(Peter), (Gertrud)\}
\]

\[
[\text{Employee} - \text{Manager}] = \{(Heinz), (Jule)\}
\]
Bag vs. Set semantics

- So far: Relations are sets (Set semantics)
  - ⇒ A tuple appears at most one time
Bag vs. Set semantics

- So far: Relations are sets *(Set semantics)*
  - A tuple appears at most one time
- This is different from SQL and database implementations
  - Tuple can appear more than once
Bag vs. Set semantics

- So far: Relations are sets (*Set semantics*)
  - A tuple appears at most one time
- This is different from SQL and database implementations
  - Tuple can appear more than once
  - In relation in DB only if no Primary key
Bag vs. Set semantics

- So far: Relations are sets (Set semantics)
  - A tuple appears at most one time
- This is different from SQL and database implementations
  - Tuple can appear more than once
  - In relation in DB only if no Primary key
  - This is called Bag semantics
Bag vs. Set semantics

- So far: Relations are sets (Set semantics)
  - A tuple appears at most one time
- This is different from SQL and database implementations
  - Tuple can appear more than once
  - In relation in DB only if no Primary key
  - This is called Bag semantics
- Bag semantics
  - Formally: assign a multiplicity $\geq 1$ to each tuple in a relation
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism
- Why bag semantics?
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism
- Why bag semantics?
  - Correctness
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism

- Why bag semantics?
  - Correctness
    - (e.g., projecting on non-unique attribute, then aggregate)
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism

- Why bag semantics?
  - Correctness
    - (e.g., projecting on non-unique attribute, then aggregate)
  - Performance
Bag vs. Set semantics cont.

- Why set semantics?
  - Cleaner formalism

- Why bag semantics?
  - Correctness
    - (e.g., projecting on non-unique attribute, then aggregate)
  - Performance
    - Some operators require costly duplicate removal under set semantics
Bag semantics: Notation

How to write multiplicities

- Use power notation to express the multiplicity of a tuple
- $t^n \in R$ denotes tuple $t$ exists with multiplicity $n$ in relation $R$
Bag semantics: Operators

**Duplicate Removal**

- $\delta(R)$
- Returns a copy of $R$ with all multiplicities set to one
Bag semantics: Other operators

Definitions

\[ [[\pi_A(R)]] = \{ t^n \mid n = \sum_{u^m \in R \land u.A = t} m \} \]

\[ [[\sigma_C(R)]] = \{ t^n \mid t^n \in R \land t \models C \} \]

\[ [[\alpha_{G,\text{agg}}(R)]] = \{ (t.G, res_1, \ldots, res_m)^1 \mid t^n \in R \]
\[ \land \forall i \in \{1, m\} : res_i = \text{agg}_i(\pi_{B_i}(\sigma_{G=t.G}(R))) \} \]

\[ [[R \bowtie_C S]] = \{ (t_1 \triangleright t_2)^{n \times m} \mid t_1^n \in R \land t_2^m \in S \]
\[ \land (t_1 \triangleright t_2) \models C \} \]

\[ [[R \bowtie C S]] = \{ (t_1 \triangleright t_2)^{n \times m} \mid t_1^n \in R \land t_2^m \in S \}
\[ \cup \{ (t_1 \triangleright \text{null}(S))^n \mid t_1^n \in R \]
\[ \land (\not\exists t_2 \in S : (t_1 \triangleright t_2) \models C) \} \]
Bag semantics: Other operators

Definitions

\[
\begin{align*}
[[R \cup S]] &= \{ t^{n+m} \mid t^n \in R \land t^m \in S \} \\
[[R \cap S]] &= \{ t^{\min(n,m)} \mid t^n \in R \land t^m \in S \} \\
[[R - S]] &= \{ t^{n-m} \mid t^n \in R \land t^m \in S \}
\end{align*}
\]
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Origin of Term</td>
</tr>
<tr>
<td>2. Relational Algebra Primer</td>
</tr>
<tr>
<td>3. What is Provenance?</td>
</tr>
<tr>
<td>- Provenance in Data Processing</td>
</tr>
<tr>
<td>- An Abstract View on Provenance</td>
</tr>
<tr>
<td>- Running Example</td>
</tr>
<tr>
<td>4. Types of Provenance Information</td>
</tr>
<tr>
<td>5. Use Cases and Application Domains</td>
</tr>
<tr>
<td>6. Provenance Generation, Storage, and Querying</td>
</tr>
</tbody>
</table>
Data Provenance

Information about the **creation process** and **origin** of data
Given a piece of data

- How do we know . . .
  - which data it is derived from?
  - which transformations (SQL) where used to create it?
  - who created it?
  - . . .

Example

<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>
Provenance in Data Processing

Given a piece of data

- How do we know ... 
  - which data it is derived from? 
  - which transformations (SQL) were used to create it? 
  - who created it? 
  - ... 

Example

Compute the revenue for each shop as sum of prices of items sold

```
SELECT shop, 
  sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

<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>

<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>
Provenance in Data Processing

Given a piece of data

- How do we know . . .
  - which data it is derived from?
  - which transformations (SQL) were used to create it?
  - who created it?
  - . . .

Definition (Data Provenance)

Information about the origin and creation process of data.

Example

<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>

```sql
SELECT shop, 
  sum(price) AS rev 
FROM sales, items 
WHERE itemId = id 
GROUP BY shop
```

<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>
An Abstract View on Provenance

Abstract View

Data

- Structured? Schemata?
- Atomic units? (Data items)
An Abstract View on Provenance

Abstract View

Data

- Structured? Schemata?
- Atomic units? (Data items)

Transformations

- Consume input data
- Produce output data
- Hierarchical composition?
- Fixed set of atomic operations?
An Abstract View on Provenance

Abstract View

Data

- Structured? Schemata?
- Atomic units? (Data items)

Transformations

- Consume input data
- Produce output data
- Hierarchical composition?
- Fixed set of atomic operations?

Provenance

Information about the creation process and origin of data
Abstract View
An Abstract View on Provenance

Abstract View
An Abstract View on Provenance

Abstract View

Provenance Information (possible)

Provenance for
Running Example

Scenario

- You are an analyst for a garden supply shop
- You have to compute the first quarter revenue for each shop location
- Datawarehouse with sales data
- Use SQL to compute the required information from the warehouse
### Running Example

#### Example (Input Data)

**Employee**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
</tr>
</tbody>
</table>

**Shop**

<table>
<thead>
<tr>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1.000.000</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>4.000</td>
</tr>
</tbody>
</table>

**Item**

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>Fertilizer</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>Rake</td>
<td>9</td>
</tr>
</tbody>
</table>

**Sales**

<table>
<thead>
<tr>
<th>Employee</th>
<th>Item</th>
<th>Amount</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>342</td>
<td>2</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>342</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>555</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Example (SalesTotal Query)

```
CREATE VIEW SalesTotal AS
SELECT Location AS Shop, Month, SSN AS Employee, 
     Price * Amount AS Totalprice 
FROM Employee E, Shop H, Item I, Sales S 
WHERE E.WorksFor = H.Location 
    AND E.SSN = S.Employee 
    AND I.Id = S.Item
```

Example (Results)

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>Totalprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>123</td>
<td>199</td>
</tr>
<tr>
<td>New York</td>
<td>1</td>
<td>342</td>
<td>2048</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>342</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>555</td>
<td>9</td>
</tr>
</tbody>
</table>
Running Example

**Example (MonthlyRevenue Query)**

```sql
CREATE VIEW MonthlyRevenue
SELECT Shop, Month, sum(Totalprice) AS Revenue
FROM SalesTotal
GROUP BY Shop, Month
```

**Example (Results)**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
Running Example

Example (RevenueFirstQ Query)

```
CREATE VIEW RevenueFirstQ
SELECT Shop, sum(Revenue) AS Revenue
FROM MonthlyRevenue
WHERE Month < 5
GROUP BY Shop
```
Running Example

Compute First Quarter Revenue

RevenueFirstQ

MonthlyRevenue

SalesTotal

Employee Shop Item Sales
Running Example

Tracing an Error

Problem

- One result tuple of your query looks suspicious
- You expect the input data to be the culprit
- How to know which input data affected which output data
Running Example

Tracing an Error

Problem

- One result tuple of your query looks suspicious
- You expect the input data to be the culprit
- How to know which input data affected which output data

This is Data Provenance
Running Example

Example Data

Compute First Quarter Revenue

RevenueFirstQ

MonthlyRevenue

SalesTotal

Example

Employee

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Name</th>
<th>Shop</th>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Peter Peterson</td>
<td>New York</td>
<td>1,000,000</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>Jane Jameson</td>
<td>New York</td>
<td>500,000</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>Bob Brown</td>
<td>Wuppertal</td>
<td>200,000</td>
<td></td>
</tr>
</tbody>
</table>

Shop

<table>
<thead>
<tr>
<th>Shop ID</th>
<th>Name</th>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Main Street</td>
<td>New York</td>
<td>2,500,000</td>
</tr>
</tbody>
</table>

Item

<table>
<thead>
<tr>
<th>Item ID</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Laptops</td>
<td>599</td>
</tr>
<tr>
<td>2</td>
<td>Peripherals</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>Bike</td>
<td>98</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Item ID</th>
<th>Amount</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Running Example

Example Data

Compute First Quarter Revenue

RevenueFirstQ

α

σ

MonthlyRevenue

MonthlyRevenue

α

SalesTotal

SalesTotal

Employee

Shop

Item

Sales
## Example Data

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
<td>Wuppertal</td>
<td>4,000</td>
</tr>
</tbody>
</table>

### EmployeeShop

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
<td>Wuppertal</td>
<td>4,000</td>
</tr>
</tbody>
</table>

### Sales

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>Totalprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>1</td>
<td>199</td>
</tr>
<tr>
<td>New York</td>
<td>2</td>
<td>7</td>
<td>200</td>
</tr>
<tr>
<td>New York</td>
<td>4</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

### Example

#### SalesTotal

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>Totalprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>1</td>
<td>199</td>
</tr>
<tr>
<td>New York</td>
<td>2</td>
<td>7</td>
<td>200</td>
</tr>
<tr>
<td>New York</td>
<td>4</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>2</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>
Running Example

Lessons learned

• Which inputs belong to provenance of outputs?
• Even if we know: How to get it?
• Manually?
• Not reasonable for big data or complex query!
• Need system that tracks it automatically!
Lessons learned

• Which inputs belong to provenance of outputs?
Lessons learned

- Which inputs belong to provenance of outputs?
  - hard
Lessons learned

- Which inputs belong to provenance of outputs?
  - hard

- Even if we know: How to get it?
Lessons learned

- Which inputs belong to provenance of outputs?
  - hard

- Even if we know: How to get it?

- Manually?
Lessons learned

- Which inputs belong to provenance of outputs?
  - hard
- Even if we know: How to get it?
- Manually?
  - Not reasonable for big data or complex query!
Lessons learned

- Which inputs belong to provenance of outputs?
  - hard

- Even if we know: How to get it?

- Manually?
  - Not reasonable for big data or complex query!

- Need system that tracks it automatically!
Outline

1. Origin of Term

2. Relational Algebra Primer

3. What is Provenance?

4. Types of Provenance Information
   - Data Provenance
   - Transformation Provenance
   - Other

5. Use Cases and Application Domains

6. Provenance Generation, Storage, and Querying

Recap
Types of Provenance Information

- Data Provenance
- Transformation Provenance
- Additional Information
## Types of Provenance Information

### Provenance Types

- **Data Provenance**
  - From which *input data* is which *output data* derived from

- **Transformation Provenance**

- **Additional Information**
Types of Provenance Information

Provenance Types

- Data Provenance
  - From which input data is which output data derived from

- Transformation Provenance
  - Which transformations contributed in which way to which output data

- Additional Information
Types of Provenance Information

Provenance Types

- **Data Provenance**
  - From which input data is which output data derived from

- **Transformation Provenance**
  - Which transformations contributed in which way to which output data

- **Additional Information**
  - Execution environment (state of the world)
  - Involved Users
Data Provenance

Which/How input data influences output data
Data Provenance

Which/How input data influences output data

- Data Granularity
  - Attribute value
  - Tuple
  - Relation
## Data Provenance

### Which/How input data influences output data

- **Data Granularity**
  - Attribute value
  - Tuple
  - Relation

- **Transformation Granularity**
  - Query with view unfolding
  - Query block
  - Algebra operator
Data Provenance

Which/How input data influences output data

- Data Granularity
  - Attribute value
  - Tuple
  - Relation

- Transformation Granularity
  - Query with view unfolding
  - Query block
  - Algebra operator

- “True” Data Dependencies?
  - Black-box: An output depends on all inputs
  - Fine-grained: Dependencies depending on how data is processed by transformation
Data Provenance

Data Granularity

Example (Relation)

RevenueFirstQ

<table>
<thead>
<tr>
<th>Shop</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>2265</td>
</tr>
</tbody>
</table>

```
CREATE VIEW RevenueFirstQ
SELECT Shop, sum(Revenue) AS Revenue
FROM MonthlyRevenue
WHERE Month < 5
GROUP BY Shop
```

MonthlyRevenue

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
**Data Provenance**

**Data Granularity**

**Example (Tuple)**

**RevenueFirstQ**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>2265</td>
</tr>
</tbody>
</table>

**RevenueFirstQ**

```
CREATE VIEW RevenueFirstQ
SELECT Shop, sum(Revenue) AS Revenue
FROM MonthlyRevenue
WHERE Month < 5
GROUP BY Shop
```

**MonthlyRevenue**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
Data Provenance

Data Granularity

Example (Attribute Value)

<table>
<thead>
<tr>
<th>Shop</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>2265</td>
</tr>
</tbody>
</table>

RevenueFirstQ

CREATE VIEW RevenueFirstQ
SELECT Shop, sum(Revenue) AS Revenue
FROM MonthlyRevenue
WHERE Month < 5
GROUP BY Shop

MonthlyRevenue

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
Data Provenance

Transformation Granularity

Example

Compute First Quarter Revenue
MonthlyRevenue
RevenueFirstQ
SalesTotal
Data Provenance

Transformation Granularity

Example

```
Compute First Quarter Revenue

RevenueFirstQ

MonthlyRevenue

SalesTotal

Sales

Employee

Shop

Item

Employee

Item

Sales

Compute First Quarter Revenue

RevenueFirstQ

MonthlyRevenue

SalesTotal

```

**Example: Compute First Quarter Revenue**

- **RevenueFirstQ**
  - **MonthlyRevenue**
    - **SalesTotal**
      - **Employee**
        - **Shop**
          - **Item**
            - **Sales**

**Data Provenance Table**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2241</td>
</tr>
<tr>
<td>New York</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

**SalesTotal Table**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>TotalPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2</td>
<td>2241</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

**Employee Table**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter</td>
<td>Male</td>
<td>32</td>
<td>1990</td>
</tr>
<tr>
<td>456</td>
<td>John</td>
<td>Female</td>
<td>28</td>
<td>1994</td>
</tr>
<tr>
<td>789</td>
<td>Maria</td>
<td>Female</td>
<td>35</td>
<td>1987</td>
</tr>
</tbody>
</table>

**Shop Table**

<table>
<thead>
<tr>
<th>Location</th>
<th>Date</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1990-01-01</td>
<td>100000</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>1990-01-01</td>
<td>4000</td>
</tr>
</tbody>
</table>

**Item Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>Fertilizer</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Seeds</td>
<td>10</td>
</tr>
</tbody>
</table>

**Employee Table**

<table>
<thead>
<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>Male</td>
<td>30</td>
<td>1990</td>
</tr>
<tr>
<td>2</td>
<td>Maria</td>
<td>Female</td>
<td>35</td>
<td>1985</td>
</tr>
<tr>
<td>3</td>
<td>Peter</td>
<td>Male</td>
<td>28</td>
<td>1993</td>
</tr>
</tbody>
</table>

**Shop Table**

<table>
<thead>
<tr>
<th>Location</th>
<th>Date</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1990-01-01</td>
<td>100000</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>1990-01-01</td>
<td>4000</td>
</tr>
</tbody>
</table>

**Item Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>Fertilizer</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Seeds</td>
<td>10</td>
</tr>
</tbody>
</table>
Data Provenance

Transformation Granularity

Example

SalesTotal

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>TotalPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>3</td>
<td>2018</td>
</tr>
<tr>
<td>New York</td>
<td>2</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

EmployeeShop

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
<td>Wuppertal</td>
<td>4,000</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>179</td>
</tr>
<tr>
<td>2</td>
<td>Hedgecutter</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>Mower</td>
<td>9</td>
</tr>
</tbody>
</table>

Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
</tr>
</tbody>
</table>

Shop

<table>
<thead>
<tr>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>4,000</td>
</tr>
</tbody>
</table>

Item

<table>
<thead>
<tr>
<th>id</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>179</td>
</tr>
<tr>
<td>2</td>
<td>Hedgecutter</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>Mower</td>
<td>9</td>
</tr>
</tbody>
</table>

Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinzmann</td>
<td>Wuppertal</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>Employee</th>
<th>Item</th>
<th>Amount</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>62</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Data Provenance

Data Dependencies

Example (Black-box)

**RevenueFirstQ**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>2265</td>
</tr>
</tbody>
</table>

**MonthlyRevenue**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>
Data Provenance

Data Dependencies

Example (Fine-grained)

**RevenueFirstQ**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>2265</td>
</tr>
</tbody>
</table>

**MonthlyRevenue**

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2247</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

```sql
CREATE VIEW RevenueFirstQ
SELECT Shop, sum(Revenue) AS Revenue
FROM MonthlyRevenue
WHERE Month < 5
GROUP BY Shop
```
Transformation Provenance

Which/How transformations contributed to output data

- Transformations that generated output (transitive?)
- Only the ones that had actual effect
- Workflow template/program vs. workflow run/execution
### Additional Information

- **A small subset**
  - OS version

- Other
  - OS version
  - Version of library linked against
  - Environment variables
  - User that executed the process
  - Current main memory content
  - Room temperature?
  - Geographical location
  - Butterfly that flapped in china
A small subset

- OS version
- Version of library linked against
## Additional Information

### A small subset

- OS version
- Version of library linked against
- Environment variables
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
- Room temperature?
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
- Room temperature?
- Geographical location

Butterfly that flapped in China...
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
- Room temperature?
- Geographical location
- ...

Additional Information:

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
- Room temperature?
- Geographical location
- ...
Additional Information

A small subset

- OS version
- Version of library linked against
- Environment variables
- User that executed the process
- Current main memory content
- Room temperature?
- Geographical location
- ...
- Butterfly that flapped in china
Outline

1. Origin of Term
2. Relational Algebra Primer
3. What is Provenance?
4. Types of Provenance Information
5. Use Cases and Application Domains
   - Use Cases
   - Debugging
   - Annotation Propagation
   - Deletion Propagation
Use Cases

- Debugging (tracking the sources of errors)
- Propagating annotations
- Gain deeper understanding of data and transformations
  - Estimate quality, trust
- Improvement of other data processing technologies
  - Probabilistic databases
  - Deletion propagation
  - Testing
Application Domains

- Complex database queries, e.g., datawarehousing
- E-science and curated databases
- Data integration/exchange
- Workflow systems
### Application Domains

- Complex database queries, e.g., datawarehousing
- E-science and curated databases
- Data integration/exchange
- Workflow systems
- ⇒ Application domain with complex, multi-stage data processing
  - Map-Reduce style processing and its “frontends” like Pig
  - Simulations
  - …
Debugging

Origin of Result Tuples

- Tuple in result suspicious/wrong/interesting

Example

SalesTotal

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>Totalprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>1</td>
<td>199</td>
</tr>
<tr>
<td>New York</td>
<td>1</td>
<td>2</td>
<td>2048</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

CREATE VIEW SalesTotal AS
SELECT Location AS Shop, Month, SSN AS Employee, Price + Amount AS Totalprice
FROM Employee E, Shop H, Item I, Sales S
WHERE E.WorksFor = H.Location
AND E.SSN = S.Employee
AND I.Id = S.Item
Origin of Result Tuples

- Tuple in result suspicious/wrong/interesting
- Learn more by looking at relevant inputs (provenance)

Example

```
CREATE VIEW SalesTotal AS
SELECT Location AS Shop, Month, SSN AS Employee, Price + Amount AS TotalPrice
FROM Employee E, Shop H, Item I, Sales S
WHERE E.WorksFor = H.Location
AND E.SSN = S.Employee
AND I.Id = S.Item
```

<table>
<thead>
<tr>
<th>Shop</th>
<th>Month</th>
<th>Employee</th>
<th>TotalPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>1</td>
<td>199</td>
</tr>
<tr>
<td>New York</td>
<td>1</td>
<td>2</td>
<td>2048</td>
</tr>
<tr>
<td>New York</td>
<td>3</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>5</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>

Employee

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>WorksFor</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Peter Peterson</td>
<td>New York</td>
</tr>
<tr>
<td>342</td>
<td>Jane Janeson</td>
<td>New York</td>
</tr>
<tr>
<td>555</td>
<td>Heinz Heinsmann</td>
<td>Wuppertal</td>
</tr>
</tbody>
</table>

Shop

<table>
<thead>
<tr>
<th>Location</th>
<th>Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1000000</td>
</tr>
<tr>
<td>Wuppertal</td>
<td>4000</td>
</tr>
</tbody>
</table>

Item

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lawnmower</td>
<td>199</td>
</tr>
<tr>
<td>2</td>
<td>Fertilizer</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>Rake</td>
<td>9</td>
</tr>
</tbody>
</table>

Sales

<table>
<thead>
<tr>
<th>Employee</th>
<th>Item</th>
<th>Amount</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>342</td>
<td>2</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>342</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>555</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Approach

1. Identify tuples of interest (I)
Approach

1. Identify tuples of interest ($I$)
2. Retrieve provenance
   - Need system that returns provenance for set $I$
Approach

1. Identify tuples of interest ($I$)
2. Retrieve provenance
   - Need system that returns provenance for set $I$
   - How to represent this info?
Debugging

Approach

1. Identify tuples of interest ($I$)
2. Retrieve provenance
   - Need system that returns provenance for set $I$
   - How to represent this info?
3. What if provenance large?
   - $\Rightarrow$ Query support? Visualization?
Annotation Propagation

Example

CREATE VIEW EnzymeProduce AS
SELECT Enzyme, Name AS Gene
FROM Gene G, Enzyme E
WHERE G.Id = E.ProducedBy

EnzymeProduce

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC 1.1.1.1</td>
<td>ALB</td>
</tr>
<tr>
<td>EC 1.97.1.6</td>
<td>ALB</td>
</tr>
</tbody>
</table>

Gene

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4q11-q13</td>
<td>ALB</td>
<td>{a_4}</td>
</tr>
<tr>
<td>18q21.3</td>
<td>BCL2</td>
<td>{}</td>
</tr>
</tbody>
</table>

Enzyme

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Weight</th>
<th>ProducedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC 1.1.1.1</td>
<td>45</td>
<td>4q11-q13</td>
</tr>
<tr>
<td>EC 1.97.1.6</td>
<td>12</td>
<td>4q11-q13</td>
</tr>
</tbody>
</table>

\(a_1\) Necessary for red blood cells
\(a_2\) Produced in liver
\(a_3\) Unhealthy
\(a_4\) Discovered by Edmond Hillary
Annotation Propagation

Which annotations in query result?

- Find provenance for tuple
- Attach union of annotations in provenance

Example

CREATE VIEW EnzymeProduce AS
SELECT Enzyme, Name AS Gene
FROM Gene G, Enzyme E
WHERE G.Id = E.ProducedBy
Annotation Propagation

**For Example**

- First result tuple
- **Provenance:** first tuples from Gene and Enzyme
- **Annotation:** $a_1, a_2, a_4$

**Example**

```
CREATE VIEW EnzymeProduce AS
SELECT Enzyme, Name AS Gene
FROM Gene G, Enzyme E
WHERE G.Id = E.ProducedBy
```

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC 1.1.1.1</td>
<td>ALB</td>
</tr>
<tr>
<td>EC 1.97.1.6</td>
<td>ALB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gene</th>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a_4}</td>
<td>4q11-q13</td>
<td>ALB</td>
</tr>
<tr>
<td>{}</td>
<td>18q21.3</td>
<td>BCL2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Weight</th>
<th>ProducedBy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC 1.1.1.1</td>
<td>45</td>
<td>4q11-q13</td>
</tr>
<tr>
<td>EC 1.97.1.6</td>
<td>12</td>
<td>4q11-q13</td>
</tr>
</tbody>
</table>
```
Annotation Propagation - Caveats

Potential Problems?

- What about negative influence?
- User should have control on propagation?
- What about annotations on
  - Attribute values
  - Spanning several tuples/relations/attributes
Deletion Propagation

Problem

- Given a *materialized view*
Deletion Propagation

Problem

- Given a *materialized view*
  - Query result stored as a table
Deletion Propagation

**Problem**

- Given a *materialized view*
  - Query result stored as a table
- How to update the view when input data changes
Deletion Propagation

**Problem**

- Given a *materialized view*
  - Query result stored as a table
- How to update the view when input data changes
  - Without recomputing the whole query
Deletion Propagation

Problem

- Given a *materialized view*
  - Query result stored as a table
- How to update the view when input data changes
  - Without recomputing the whole query
- **Deletion Propagation**: Update the view when input tuples are deleted?
Deletion Propagation Example

**CREATE VIEW ActiveCS AS**

```sql
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><strong>Id</strong></td>
<td><strong>Name</strong></td>
<td><strong>Project</strong></td>
</tr>
<tr>
<td>1</td>
<td>Peter</td>
<td>Server</td>
</tr>
<tr>
<td>2</td>
<td>Gertrud</td>
<td>Webpage</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
<td>Fire CS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Dep</strong></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>

**Deletion Propagation**

**Example**

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
### Deletion Propagation 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>

**ActiveCS**

- Emp
  - $t_1$: Peter
  - $t_2$: Gertrud
Deletion Propagation Example

- Delete tuple from Projects

### Employee

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>Peter</td>
</tr>
<tr>
<td>e2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>e3</td>
<td>Michael</td>
</tr>
</tbody>
</table>

### Project

<table>
<thead>
<tr>
<th>Name</th>
<th>Dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Server CS</td>
</tr>
<tr>
<td>p2</td>
<td>Webpage CS</td>
</tr>
<tr>
<td>p3</td>
<td>Fire CS HR</td>
</tr>
</tbody>
</table>

### Assigned

<table>
<thead>
<tr>
<th>Project</th>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
</tr>
<tr>
<td>a2</td>
<td>2</td>
</tr>
<tr>
<td>a3</td>
<td>2</td>
</tr>
<tr>
<td>a4</td>
<td>3</td>
</tr>
</tbody>
</table>

Example of Deletion Propagation:
- Delete tuple from Projects
- Update related tuples in other relations.
Deletion Propagation Example

What would be the effect on the view?

**ActiveCS**

<table>
<thead>
<tr>
<th>Emp</th>
<th>t_1</th>
<th>Peter</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_2</td>
<td>Gertrud</td>
<td></td>
</tr>
</tbody>
</table>

**Employee**

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_1</td>
<td>Peter</td>
</tr>
<tr>
<td>e_2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>e_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>
Deletion Propagation - Approach

Assumption

- Assume we have provenance for each tuple
  - For now a set of input tuples
    - \( P(t_1) = \{ e_1, p_1, a_1 \} \)
    - \( P(t_2) = \{ e_1, p_1, p_2, a_1, a_2 \} \)
  - Set of deleted tuples (\( D = \{ p_1 \} \))
Deletion Propagation - Approach

Assumption

- Assume we have provenance for each tuple
  - For now a set of input tuples
    - $P(t_1) = \{e_1, p_1, a_1\}$
    - $P(t_2) = \{e_1, p_1, p_2, a_1, a_2\}$

- Set of deleted tuples ($D = \{p_1\}$)

Approach

1. Remove $D$ from provenance
2. Remove tuples without justification from view
   - Set provenance model to simple
   - Will learn later how this actually works
Deletion Propagation Example

Example

- \( P(t_1) = \{ e_1, p_1, a_1 \} \rightarrow \{ e_1, a_1 \} \)
- \( P(t_2) = \{ e_1, p_1, p_2, a_1, a_2 \} \rightarrow \{ e_1, p_2, a_1, a_2 \} \)
Outline

1. Origin of Term
2. Relational Algebra Primer
3. What is Provenance?
4. Types of Provenance Information
5. Use Cases and Application Domains
6. Provenance Generation, Storage, and Querying
   - Provenance Generation
   - Provenance Storage
   - Provenance Querying

Recap
Provenance Generation

Manual vs. Automatic

- **Manual**: User has to provide provenance information
- **Automatic**: System generates provenance information automatically
- Design space: How much information has to be provided by the user or transformation developer?

Lazy vs. Eager

- **Eager**: Generate provenance while the transformation is running
- **Lazy**: Generate provenance later once it is requested
- Tradeoff: Retrieval time vs. execution overhead
Provenance Generation

**Generation**

**Approaches**

- Run transformation in supervised environment that tracks provenance
- Instrument the transformations to produce provenance
- Record some information during execution and reconstruct provenance from this information
Supervised Environment

**Idea**

- Modify execution environment of transformations to capture provenance

**Considerations**

- What provenance to capture?
- Which parts of system...
  - Are accessible?
  - Are modifiable?
- Supervision for all or only some transformations
Example

- Hadoop - Map/Reduce
- Modify the Hadoop system to
  - store relationships between input/output keys
  - for mappers and reducers
  - in HDFS?
Supervised Environment - Discussion

Advantages
- Can capture whatever provenance we want
- No modification to transformations

Disadvantages
- **Intrusive** May have to re-implement whole system
- Overhead for transformation execution
- Parts of the system may not be accessible (e.g., web-service composition)
Provenance Generation

Instrument Transformations

Idea

- Modify the transformation to track its own provenance

Considerations

- Transformation language expressive enough to compute its own provenance?
- How to represent provenance in the data model?
Provenance Generation

Instrument Transformations - Example

Example

- SQL queries
- Rewrite queries to produce their output + provenance information
- Possible?
Provenance Generation

Instrument Transformations - Example

Example

- SQL queries
- Rewrite queries to produce their output + provenance information
- Possible? yes, later in course
Provenance Generation

Instrument Transformations - Example

Example

- SQL queries
- Rewrite queries to produce their output + provenance information
- Possible? yes, later in course
- Build a middleware that does that over standard DBMS
Instrument Transformations - Discussion

Advantages

- **Non-intrusive**: Possible without changes to system
  - If we can gather enough information about transformation from outside
  - E.g., DBMS client
- No overhead if no provenance computed
- Same data model $\Rightarrow$ Querying
- No manual changes to transformations

Disadvantages

- Performance optimizations may be limited (overhead provenance computation)
- Data model may limit the provenance representation
Reconstruction

Idea

- Recover provenance from input + output data and knowledge about transformation

Considerations

- Possible to know what’s going on in the black box?
- Need to store extra information
Reconstruction - Example

- Simple SQL query
- Write program to
  - Analyse query
  - Retrieve input and output data
  - Compute provenance
Reconstruction - Discussion

Advantages

- **Non-intrusive**: No changes to system
- No overhead for transformation
- No storage costs or almost no storage

Disadvantages

- Not possible for complex operations
- Provenance generation may be more expensive
Eager Generation

Approach

- Generate provenance during transformation execution

Considerations

- Overhead for transformation?
- How to trigger?
Provenance Generation

Lazy Generation

Approach
- Generate provenance on request

Considerations
- Input/Output data still available?
- Transformation info available?
Provenance data can be orders of magnitude larger than input/output data.

⇒ Be clever when to store what at which level of abstraction.

⇒ Specialized compression for provenance.

⇒ Index structured for provenance specific retrieval patterns.
Why is provenance large?

Simplified explanation:
Why is provenance large?

Simplified explanation:

- Input data: size $N$
Provenance Storage

Why is provenance large?

Simplified explanation:

- Input data: size $N$
- Output data: size $M$
Why is provenance large?

Simplified explanation:

- Input data: size $N$
- Output data: size $M$
- Provenance is relationship between inputs and outputs
Why is provenance large?

Simplified explanation:

- Input data: size $N$
- Output data: size $M$
- Provenance is relationship between inputs and outputs
- $\Rightarrow$ Worst case: $N \times M$
Why is provenance large?

Simplified explanation:

- Input data: size $N$
- Output data: size $M$
- Provenance is relationship between inputs and outputs
- $\Rightarrow$ Worst case: $N \times M$
- Intermediate results?
Why is provenance large?

Simplified explanation:

- Input data: size $N$
- Output data: size $M$
- Provenance is relationship between inputs and outputs
- $\Rightarrow$ Worst case: $N \times M$
- Intermediate results?
  - Transformation is tree with $X$ nodes
Why is provenance large?

Simplified explanation:
- Input data: size $N$
- Output data: size $M$
- Provenance is relationship between inputs and outputs
- $\Rightarrow$ Worst case: $N \times M$
- Intermediate results?
  - Transformation is tree with $X$ nodes
  - $\Rightarrow \sim N \times M \times X$
Provenance Storage

What to store and when?

What?
- Only necessary level of detail
  - E.g., need attribute level provenance?
  - E.g., need provenance for intermediate results?

When?
- Provenance for all transformations?
- Only for specific type?
- Only when requested by user?
- Only when triggering event happened?
**Compression**

**Rationale**
- Provenance large, but has overlap
  - Exploit overlap to compress
  - Information loss?
  - Access/querying without decompression
  - Tradeoff: speed vs. size

**Approaches**
- Generic compression algorithms
  - Small size, slow?, probably no query
- Methods exploiting overlap being aware of provenance structure
  - Size less predictable, fast?, query may be possible
Provenance Storage

Index structures

Rationale

- Provenance querying needs efficient access to provenance data
- Traditional index structure useful?
- Can identify new access patterns?
  - Tree-path traversal?
- Static index or updates possible?

Approaches

- E.g., adapt IR retrieval index structures
Provenance Storage

Example Storage - Provenance tables

- Provenance Table
- input TID’s → output TID’s

Example

```
SELECT shop, 
    sum(price) AS rev 
FROM sales, items 
WHERE itemId = id 
GROUP BY shop 
```

<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>

<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>
Provenance Storage

Example Storage - Provenance tables

- Provenance Table
- input TID's → output TID's

Provenance

<table>
<thead>
<tr>
<th>result</th>
<th>in</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>$i_1$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>$i_3$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$i_3$</td>
</tr>
</tbody>
</table>

Example

```
SELECT shop, 
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

<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>

<table>
<thead>
<tr>
<th>shop</th>
<th>itemId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migros</td>
<td>i1</td>
</tr>
<tr>
<td>Migros</td>
<td>i2</td>
</tr>
<tr>
<td>Coop</td>
<td>i3</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>
Querying

- Large amount of provenance information
Querying

- Large amount of provenance information
- Query support to extract information
  - Focus on parts of interest
    - Backward: Which data contributed to output?
    - Forward: Which data is derived from input?
    - Transitive closure
  - Correlated with input/output data
  - Summarize, abstract
Provenance Querying

Querying

- Large amount of provenance information
- Query support to extract information
  - Focus on parts of interest
    - Backward: Which data contributed to output?
    - Forward: Which data is derived from input?
    - Transitive closure
  - Correlated with input/output data
- Summarize, abstract

Example

For a subset of erroneous sales totals, which ones have been derived from input sales data from a shop in New York with a amount sold bigger than 100.
Provenance Querying

Querying

Approaches

- Extend query language for “normal” data
- New query language
Provenance Querying

Querying

Approaches

- Extend query language for “normal” data
  - Querying provenance in combination with “normal” data
  - Limitation on provenance representation
- New query language
Querying

Approaches

- Extend query language for “normal” data
  - Querying provenance in combination with “normal” data
  - Limitation on provenance representation
- New query language
  - Operations tailored for typical operations on provenance
  - “Re-inventing the wheel”
Provenance Querying

Querying Example

Example

<table>
<thead>
<tr>
<th>Provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>result</td>
</tr>
<tr>
<td>t₁</td>
</tr>
<tr>
<td>t₁</td>
</tr>
<tr>
<td>t₁</td>
</tr>
<tr>
<td>t₁</td>
</tr>
<tr>
<td>t₂</td>
</tr>
<tr>
<td>t₂</td>
</tr>
</tbody>
</table>

Example

```
result

<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>
```

```
SELECT shop,
      sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

Provenance:

- `t₁`:
  - `s₁`: Migros
  - `i₁`: 1
  - `i₃`: 3

- `t₂`:
  - `s₃`: Coop
  - `i₁`: 1
  - `i₂`: 2
  - `i₃`: 3
**Provenance Querying**

**Querying Example**

```sql
SELECT DISTINCT shop
FROM result r,
    Provenance p,
    items i
WHERE r.tid = p.result
AND p.in = i.tid
AND i.price > 90
```

**Example**

```sql
SELECT shop,
    sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

**Provenance:**

<table>
<thead>
<tr>
<th>result</th>
<th>in</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t_1)</td>
<td>(s_1)</td>
</tr>
<tr>
<td>(t_1)</td>
<td>(s_2)</td>
</tr>
<tr>
<td>(t_1)</td>
<td>(i_1)</td>
</tr>
<tr>
<td>(t_1)</td>
<td>(i_3)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>(s_3)</td>
</tr>
<tr>
<td>(t_2)</td>
<td>(i_3)</td>
</tr>
</tbody>
</table>

**Sales:**

<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>

**Items:**

<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>
Provenance Querying

Example

Example result

```
<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 SQL query:
```
SELECT shop, sum(price) AS rev
FROM sales, items
WHERE itemId = id
GROUP BY shop
```

Example provenance:
```
<table>
<thead>
<tr>
<th>result</th>
<th>in</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>s1</td>
</tr>
<tr>
<td>t1</td>
<td>s2</td>
</tr>
<tr>
<td>t1</td>
<td>i1</td>
</tr>
<tr>
<td>t1</td>
<td>i2</td>
</tr>
<tr>
<td>t2</td>
<td>s3</td>
</tr>
<tr>
<td>t2</td>
<td>i3</td>
</tr>
</tbody>
</table>
```
Outline

1. Origin of Term
2. Relational Algebra Primer
3. What is Provenance?
4. Types of Provenance Information
5. Use Cases and Application Domains
6. Provenance Generation, Storage, and Querying
7. Recap
Types of Provenance

- Data
- Transformation
- Other
- Granularities

Generation, Storage, and Querying

- **Generation**
  - Manual vs. automatic
  - Eager vs. lazy
  - Supervised environment, instrumentation, reconstruction

- **Storage**
  - Compression, Indices, What to keep?

- **Querying**
  - Extending transformation language
  - Develop new query language
Surveys I

- **Boris Glavic and Renée J. Miller.**
  Reexamining Some Holy Grails of Data Provenance.
  In Tapp '11: 3rd usenix workshop on the theory and practice of provenance, 2011.

- **James Cheney, Laura Chiticariu, and Wang-Chiew Tan.**

- **Robert Ikeda and Jennifer Widom.**
  Data Lineage: A Survey.

- **Juliana Freire, David Koop, Emanuele Santos, and Claudio T. Silva.**
  Provenance for Computational Tasks: A Survey.

- **Boris Glavic and Klaus R. Dittrich.**
  Data Provenance: A Categorization of Existing Approaches.

- **Susan B. Davidson, Sarah Cohen-Boulakia, Anat Eyal, Bertram Ludscher, Timothy McPhillips, Shawn Bowers, and Juliana Freire.**
  Provenance in Scientific Workflow Systems.
Surveys II

Wang-Chiew Tan.
Provenance in Databases: Past, Current, and Future.

Yogesh L. Simmhan, Beth Plale, and Dennis Gannon.
A Survey of Data Provenance in e-science.

Yogesh L. Simmhan, Beth Plale, and Dennis Gannon.
A Survey of Data Provenance Techniques.
Technical report, Indiana University, Bloomington IN 47405, 2005.

Wang-Chiew Tan.
Research Problems in Data Provenance.