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

I. Database Provenance

I.1 Provenance Models and Systems

Boris Glavic

September 12, 2012

Outline

Perm and the Witness List Provenance Models

Perm - **P**rovenance **E**xtension of the **R**elational **M**odel

- Database system (*PostgreSQL*) with provenance support
- **Provenance semantics**: Supports several provenance semantics
  - **Witness list** model
  - **Perm Influence Contribution Semantics** (Why/Lineage-like)
- **Provenance generation**: *PostgreSQL* query dialect[a]
- **Provenance querying**: Full SQL

[a] minus recursion

Roadmap

- Choose one supported provenance model (PI-CS)
- Explain all aspects of the system
  - The model itself
  - Provenance Representation
  - Provenance Generation
  - Implementation
- Application to data exchange + transformation provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

Model Overview

Requirements

1. Consistent with intuitive expectations
2. Defined for complete SQL (if possible)

Model Overview

Requirements

1. Consistent with intuitive expectations
2. Defined for complete SQL (if possible)

Language feature	PI-CS	Why	Lineage
Aggregation	X	-	X
Set-Operations	X	*	X
Nested Subqueries	X	-	-

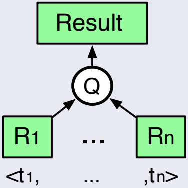
* = partially supported

ILLINOIS INSTITUTE OF TECHNOLOGY

Provenance Representation

Witness List

- List of input tuples
  - One from each input relation
  - ⇒Leafs of algebra tree
  - Or special value ⊥ (no tuple)



Provenance Representation

Witness List

- List of input tuples
  - One from each input relation
  - ⇒Leafs of algebra tree
  - Or special value ⊥ (no tuple)

Definition (Provenance)

Provenance of tuple t in the result of query q is

- Bag of witness lists
- ⇒Subset of $\mathcal{W}(q, I) = (R_1 \cup \{\bot\}) \times \ldots \times (R_n \cup \{\bot\})$
- Bags:

$$\mathcal{W}(q, I) = \{< t_1, \ldots, t_n >^{m_1 \times \ldots \times m_n} \mid t_i^{m_i} \in R_i \vee (t_i = \bot \wedge m_i = 1)\}$$

Witness Lists - Example

Example

$$q = \bowtie_{itemId=id} (sales, items)$$

$$< s_1, i_1 > \in \mathcal{W}(q, I)$$

Example

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

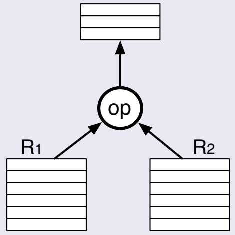
	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25



Witness Lists - Example

Example

$$q = \bowtie_{itemId=id} (sales, items)$$

$$< s_1, i_1 > \in \mathcal{W}(q, I)$$

Example

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

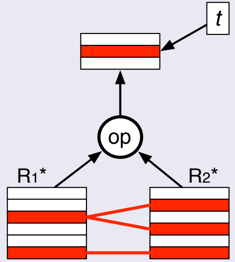
	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

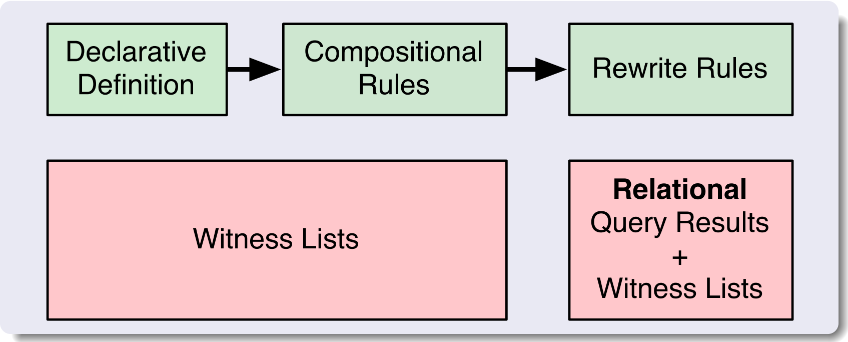
	id	price
i_1	1	100
i_2	2	10
i_3	3	25



Perm Influence Contribution Semantics (PI-CS)

- Similar goal as Why-provenance and Lineage
- Declarative definition that derived from Lineage
  - Overcomes some disadvantages
- Compositional rules
  - Bag comprehensions
- Relational Representation
- Algebraic Rewrites

Approach Roadmap



Declarative Definition

Perm-Why Provenance Semantics

- Declarative definition
- Based on *Lineage* (Stanford, Widom) [Cui et al., ICDE '00]
  - Provenance Representation
  - For complex queries (nested subqueries, outer joins)

Definition (PI-CS)

$\mathcal{PI}(op, t, I)$ is the unique subset of $\mathcal{W}(op, I)$ that fulfills the following conditions:

ILLINOIS INSTITUTE OF TECHNOLOGY

Definition Conditions

Condition (1)

- The provenance of t^x generates **exactly** t
- ⇒Executing the query over the provenance returns t
- (1): $[[op(calPI(op, t, I))]] = \{t^x\}$

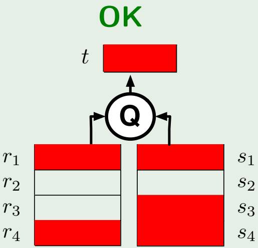
ILLINOIS INSTITUTE OF TECHNOLOGY

Definition Conditions

Condition (1)

- The provenance of t^x generates **exactly** t
- ⇒Executing the query over the provenance returns t
- (1): $[[op(calPI(op, t, I))]] = \{t^x\}$

Example

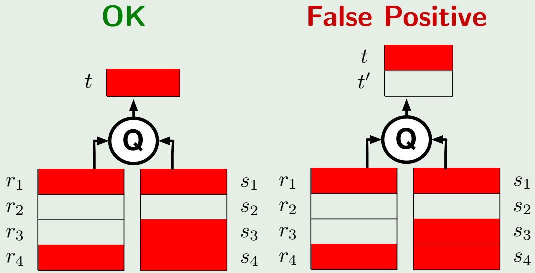


Definition Conditions

Condition (1)

- The provenance of t^x generates **exactly** t
- ⇒Executing the query over the provenance returns t
- (1): $[[op(calPI(op, t, I))]] = \{t^x\}$

Example

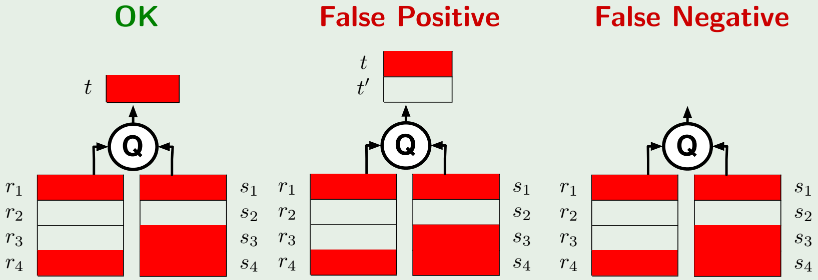


Definition Conditions

Condition (1)

- The provenance of t^x generates **exactly** t
- ⇒Executing the query over the provenance returns t
- (1): $[[op(calPI(op, t, I))]] = \{t^x\}$

Example



Condition (1) - Example

Example

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

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (1) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(\mathcal{PI}_1)]] = \{t_1\}$

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (1) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(\mathcal{PI}_1)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: NO: $[[op(\mathcal{PI}_2)]] = \{t_1, t_2\}$
  - $s_2 + i_3$ irrelevant

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (1) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(\mathcal{PI}_1)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: NO: $[[op(\mathcal{PI}_2)]] = \{t_1, t_2\}$
  - $s_2 + i_3$ irrelevant
- $\mathcal{PI}_3 = \{< s_2, i_1 >\}$: NO: $[[op(\mathcal{PI}_3)]] = \emptyset$

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (1) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(\mathcal{PI}_1)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: NO: $[[op(\mathcal{PI}_2)]] = \{t_1, t_2\}$
  - $s_2 + i_3$ irrelevant
- $\mathcal{PI}_3 = \{< s_2, i_1 >\}$: NO: $[[op(\mathcal{PI}_3)]] = \emptyset$
- $\mathcal{PI}_4 = \{< s_1, i_1 >, < s_1, i_2 >\}$: YES: $[[op(\mathcal{PI}_4)]] = \{t_1\}$
  - i_2 irrelevant

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Definition Conditions Cont.

Condition (2)

- Each witness list *contributes* something to t
- Evaluating the query over the witness list $\Rightarrow$result not empty
- (2) $\forall w \in \mathcal{PI}(op, t, I) : [[op(w)]] \neq \emptyset$

Definition Conditions Cont.

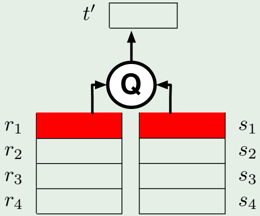
Condition (2)

- Each witness list *contributes* something to t
- Evaluating the query over the witness list $\Rightarrow$ result not empty
- (2) $\forall w \in \mathcal{PI}(op, t, I) : [[op(w)]] \neq \emptyset$

Example

$\{ < r_1, s_1 >,$
$< r_4, s_3 >,$
$< r_4, s_4 > \}$

OK

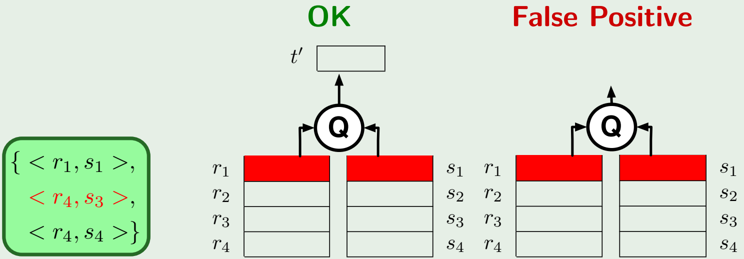


Definition Conditions Cont.

Condition (2)

- Each witness list *contributes* something to t
- Evaluating the query over the witness list ⇒result not empty
- (2) $\forall w \in \mathcal{PI}(op, t, I) : [[op(w)]] \neq \emptyset$

Example



Condition (2) - Example

Example

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

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (2) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(< s_1, i_1 >)]] = \{t_1\}$

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (2) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(< s_1, i_1 >)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: YES: $[[op(< s_2, i_3 >)]] = \{t_2\}$
    - $s_2 + i_3$ irrelevant

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (2) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(< s_1, i_1 >)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: YES: $[[op(< s_2, i_3 >)]] = \{t_2\}$
  - $s_2 + i_3$ irrelevant
- $\mathcal{PI}_3 = \{< s_2, i_1 >\}$: NO: $[[op(< s_2, i_1 >)]] = \emptyset$

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Condition (2) - Example

Example

- $\mathcal{PI}_1 = \{< s_1, i_1 >\}$: YES: $[[op(< s_1, i_1 >)]] = \{t_1\}$
- $\mathcal{PI}_2 = \{< s_1, i_1 >, < s_2, i_3 >\}$: YES: $[[op(< s_2, i_3 >)]] = \{t_2\}$
  - $s_2 + i_3$ irrelevant
- $\mathcal{PI}_3 = \{< s_2, i_1 >\}$: NO: $[[op(< s_2, i_1 >)]] = \emptyset$
- $\mathcal{PI}_4 = \{< s_1, i_1 >, < s_1, i_2 >\}$: NO: $[[op(< s_1, i_2 >)]] = \emptyset$
  - i_2 irrelevant

Example

sales

	shop	itemId
s_1	Migros	1
s_2	Migros	3
s_3	Coop	3

items

	id	price
i_1	1	100
i_2	2	10
i_3	3	25

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

	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

Definition Conditions Cont.

Condition (3)

- Avoid subsumed witness lists
- (3) $w, w' \in \mathcal{W}(q, I) : w \prec w' \wedge w \in \mathcal{PI}(q, t, I) \Rightarrow w' \notin \mathcal{PI}(q, t, I)$
- $w \prec w' : (\forall i : w[i] = w'[i] \vee w[i] = \perp) \wedge (\exists i : w'[i] \neq \perp \wedge w[i] = \perp)$
- Similarity to Minimal elements for Why-provenance!

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (3) and Subsumption example

Example

$w =< t_1, \perp >$ $\qquad$ $w' =< t_1, t_2 >$

$w'' =< t_1, t_3 >$ $\qquad$ $w''' =< \perp, t_3 >$

Condition (3) and Subsumption example

Example

$$w = < t_1, \perp > \qquad w' = < t_1, t_2 >$$
$$w'' = < t_1, t_3 > \qquad w''' = < \perp, t_3 >$$

- $w \prec w'$: YES

Condition (3) and Subsumption example

Example

$$w =< t_1, \perp > \qquad w' =< t_1, t_2 >$$

$$w'' =< t_1, t_3 > \qquad w''' =< \perp, t_3 >$$

- $w \prec w'$: YES
- $w' \prec w$: NO

Condition (3) and Subsumption example

Example

$$w = < t_1, \perp > \qquad w' = < t_1, t_2 >$$
$$w'' = < t_1, t_3 > \qquad w''' = < \perp, t_3 >$$

- $w \prec w'$: YES
- $w' \prec w$: NO
- $w \prec w''$: YES

Condition (3) and Subsumption example

Example

$$w = < t_1, \perp > \qquad w' = < t_1, t_2 >$$
$$w'' = < t_1, t_3 > \qquad w''' = < \perp, t_3 >$$

- $w \prec w'$: YES
- $w' \prec w$: NO
- $w \prec w''$: YES
- $w'' \prec w'$: NO

Condition (3) and Subsumption example

Example

$$w = < t_1, \perp > \qquad w' = < t_1, t_2 >$$

$$w'' = < t_1, t_3 > \qquad w''' = < \perp, t_3 >$$

- $w \prec w'$: YES
- $w' \prec w$: NO
- $w \prec w''$: YES
- $w'' \prec w'$: NO
- $w''' \prec w'$: NO

Condition (3) and Subsumption example

Example

- $q = op(R, S) = (R - S)$
- $\mathcal{PI}(q, t, I) = \{< r_1, \perp >\}$

Example

R

	a
r_1	1

S

	b
s_1	2
s_2	3

Q

	a
t_1	1

Condition (3) and Subsumption example

Example

- $q = op(R, S) = (R - S)$
- $\mathcal{PI}(q, t, I) = \{< r_1, \perp >\}$
- Tuples from S irrelevant

Example

R

	a
r_1	1

S

	b
s_1	2
s_2	3

Q

	a
t_1	1

Condition (3) and Subsumption example

Example

- $q = op(R, S) = (R - S)$
- $\mathcal{PI}(q, t, I) = \{< r_1, \perp >\}$
- Tuples from S irrelevant
- E.g., $< r_1, \perp > \prec < r_1, s_1 >$

Example

R

	a
r_1	1

S

	b
s_1	2
s_2	3

Q

	a
t_1	1

Definition Conditions Cont.

Condition (4)

- No contributing tuples omitted
- ⇒Provenance is maximal set fulfilling conditions (1),(2),(3)
- (4): $\nexists \mathcal{P} \supset \mathcal{P}' \subseteq \mathcal{W}(q, I) : \mathcal{P}' \models (1), (2), (3)$

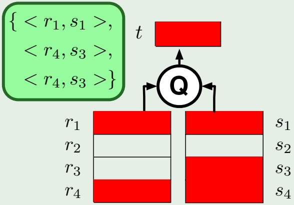
Definition Conditions Cont.

Condition (4)

- No contributing tuples omitted
- ⇒Provenance is maximal set fulfilling conditions (1),(2),(3)
- (4): $\nexists \mathcal{P} \supset \mathcal{P}' \subseteq \mathcal{W}(q, I) : \mathcal{P}' \models (1), (2), (3)$

Example

OK



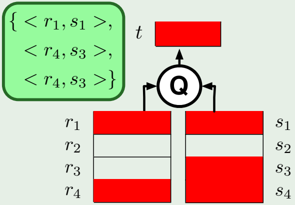
Definition Conditions Cont.

Condition (4)

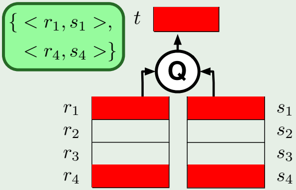
- No contributing tuples omitted
- ⇒Provenance is maximal set fulfilling conditions (1),(2),(3)
- (4): $\nexists \mathcal{P} \supset \mathcal{P}' \subseteq \mathcal{W}(q, I) : \mathcal{P}' \models (1), (2), (3)$

Example

OK



False Negative



Definition Application

Example

Result

	a
t_1	1

↑

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

GY

Definition Application

Provenance

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Example

Result

	a
t_1	1

```sql
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Definition Application

Provenance

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Conditions?

1. Returns t_1

Example

Result

	a
t_1	1

↑

```sql
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Definition Application

Provenance

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Conditions?

1. Returns t_1
2. Each witness list contributes

Example

Result

	a
t_1	1

↑

```sql
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Definition Application

Provenance

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Conditions?

1. Returns t_1
2. Each witness list contributes
3. No subsumptions

Example

Result

	a
t_1	1

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Definition Application

Provenance

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Conditions?

1. Returns t_1
2. Each witness list contributes
3. No subsumptions
4. Cannot add anything without breaking 1, 2, or 3

Example

Result

	a
t_1	1

↑

```sql
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

GY

Definition Recap

Definition

Perm Influence Contribution Semantics (PI-CS) $\mathcal{PI}(op, t, I)$, the provenance of t^x from the result of op over instance I, is the unique subset of $\mathcal{W}(op, I)$ that fulfills the following conditions:

- (1): $[[op(\mathcal{PI}(op, t, I))]] = \{t^x\}$
- (2) $\forall w \in \mathcal{PI}(op, t, I) : [[op(w)]] \neq \emptyset$
- (3): $w, w' \in \mathcal{W}(q, I) : w \prec w' \land w \in \mathcal{PI}(q, t, I) \Rightarrow w' \notin \mathcal{PI}(q, t, I)$
- (4): $\nexists \mathcal{P} \supset \mathcal{P}' \subseteq \mathcal{W}(q, I) : \mathcal{P}' \models (1), (2), (3)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Provenance for Queries

Transitivity

- Recursive definition similar to Lineage
- Query = operator(subquery): $q = op(q_1)$
- Compute provenance for operator over result of subquery
- Substitute tuples in witness lists with their provenance according to subquery

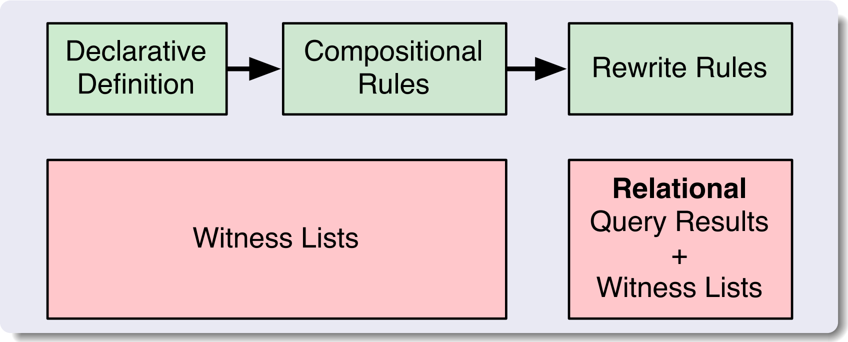
ILLINOIS INSTITUTE OF TECHNOLOGY

Compositional Rules

‘

Outline

Approach Roadmap - Compositional Rules



Compositional Rules

- A more computable form than declarative definition
- Similar idea as for Lineage
- Prove equivalence with declarative definition

Compositional Rules Definition

Definition (Compositional Rules)

$$\mathcal{PI}(R, t) = \{< t >^n \mid t^n \in R\}$$

$$\mathcal{PI}(\sigma_C(q_1), t) = \mathcal{PI}(q_1, t)$$

$$\mathcal{PI}(\pi_A(q_1), t) = \{w^n \mid w^n \in \mathcal{PI}(q_1, u) \wedge u.A = t\}$$

$$\mathcal{PI}(\alpha_{G,agg}(q_1), t) = \{w^n \mid w^n \in \mathcal{PI}(q_1, u) \wedge u.G = t.G\} \cup \{<\perp>\mid Q_1 = \emptyset \wedge \|G\| = 0\}$$

$$\mathcal{PI}(q_1 \bowtie_C q_2, t) = \{(w_1 \blacktriangleright w_2)^{n \times m} \mid w_1^n \in \mathcal{PI}(q_1, t.\mathbf{Q_1}) \wedge w_2^m \in \mathcal{PI}(q_2, t.\mathbf{Q_2})\}$$

$$\mathcal{PI}(q_1 ⟕_C q_2, t) = \begin{cases} \{(w \blacktriangleright \perp (q_2))^n \mid w^n \in \mathcal{PI}(q_1, t.\mathbf{Q_1})\} & \text{if } t \not\models C \\ \mathcal{PI}(q_1 \bowtie_C q_2, t) & \text{else} \end{cases}$$

Compositional Rules Definition

Definition (Compositional Rules)

$$
\begin{aligned}
\mathcal{PI}(q_1 \cup q_2, t) = \{(w \blacktriangleright \bot\ (q_2))^n \mid w^n \in \mathcal{PI}(q_1, t)\} \\
\cup \{(\bot\ (q_1) \blacktriangleright w)^n \mid w^n \in \mathcal{PI}(q_2, t)\} \\
\mathcal{PI}(q_1 \cap q_2, t) = \{(w_1 \blacktriangleright w_2)^{n \times m} \mid w^n \in \mathcal{PI}(q_1, t) \\
\wedge\ w_2^m \in \mathcal{PI}(q_2, t)\} \\
\mathcal{PI}(q_1 - q_2, t) = \{(w \blacktriangleright \bot\ (q_2))^n \mid w^n \in \mathcal{PI}(q_1, t)\}
\end{aligned}
$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Compositional Semantics Example

Example

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

RevenueFirstQ

Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

ILLINOIS INSTITUTE OF TECHNOLOGY

Compositional Semantics Example

Example

$q = \alpha_{shop,sum(revenue)}(q_1)$ $q_1 = \sigma_{month<5}(M)$

RevenueFirstQ

Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

ILLINOIS INSTITUTE OF TECHNOLOGY

Compositional Semantics Example

Example

$$q = \alpha_{shop,sum(revenue)}(q_1) \qquad q_1 = \sigma_{month<5}(M)$$

$$\mathcal{PI}(q, t_1) = \mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1)$$

$$\mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t) = \{w^n \mid w^n \in \mathcal{PI}(q_1, u) \wedge u.shop = t.shop\}$$

$$\mathcal{PI}(q_1, t) = \mathcal{PI}(M, t)$$

RevenueFirstQ

Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

Compositional Semantics Example

Example

$$q = \alpha_{shop,sum(revenue)}(q_1) \qquad q_1 = \sigma_{month<5}(M)$$

$$\mathcal{PI}(q, t_1) = \mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1)$$

$$\mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1) = \{< s_1 >, < s_2 >\}$$

$$\mathcal{PI}(q_1, s_1) =< m_1 >$$

$$\mathcal{PI}(q_1, s_2) =< m_2 >$$

RevenueFirstQ

Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

Compositional Semantics Example

Example

$$q = \alpha_{shop,sum(revenue)}(q_1) \qquad q_1 = \sigma_{month<5}(M)$$

$$\mathcal{PI}(q, t_1) = \mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1)$$

$$\mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1) = \{< m_1 >, < m_2 >\}$$

$$\mathcal{PI}(q_1, s_1) =< m_1 >$$

$$\mathcal{PI}(q_1, s_2) =< m_2 >$$

RevenueFirstQ

Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

Compositional Semantics Example

Example

$$q = \alpha_{shop,sum(revenue)}(q_1) \qquad q_1 = \sigma_{month<5}(M)$$

$$\mathcal{PI}(q, t_1) = \{< m_1 >, < m_2 >\}$$
$$\mathcal{PI}(\alpha_{shop,sum(revenue)}(q_1), t_1) = \{< m_1 >, < m_2 >\}$$
$$\mathcal{PI}(q_1, s_1) =< m_1 >$$
$$\mathcal{PI}(q_1, s_2) =< m_2 >$$

RevenueFirstQ

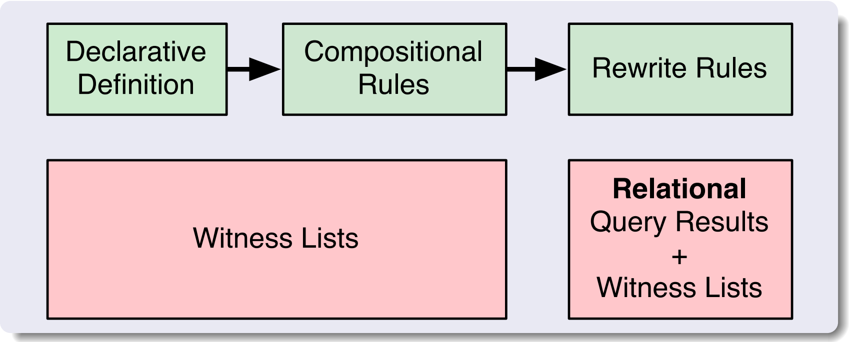
Shop	Revenue	
New York	2265	t_1

MonthlyRevenue/Q_1

Shop	Month	Revenue	
New York	1	2247	m_1/s_1
New York	3	18	m_2/s_2
Wuppertal	5	9	m_3

Outline

Approach Roadmap - Compositional Rules



Requirements for Provenance Representation

Problem Statement

- For each result tuple t
- Several *contributing* witness lists
- How to represent this information?

Requirements for Provenance Representation

Problem Statement

- For each result tuple t
- Several *contributing* witness lists
- How to represent this information?

Requirements

1. Comprehensible for human
   - Should help user to understand data
2. Query-able
   - Because provenance can be huge
3. Relationship data and provenance
   - Queries that combine both

Provenance Representation

Problem Statement

- For each result tuple t
- Several *contributing* witness lists
- How to represent this information?

Example (Provenance)

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Example

Result

	a
t_1	1

↑

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

GY

Provenance Representation

Problem Statement

- For each result tuple t
- Several *contributing* witness lists
- How to represent this information?

Example (Provenance)

$$\{< r_1, s_1 >, < r_1, s_2 >\}$$

Solution

- Provenance + normal tuples in *single* relation

Example

Result

	a
t_1	1

↑

```
SELECT DISTINCT R.a
FROM R, S
WHERE R.a = S.c;
```

↑ ↑

R

	a	b
r_1	1	2
r_2	3	6

S

	c	d
s_1	1	2
s_2	1	3

Solution

Relation with Provenance + Normal Data

- **Data**: result tuple + all tuples from a witness list
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)
  - Generate self-explanatory names: P(a)

Example (Normal Query Result)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result q

a
1

OF TECHNOLOGY

Solution

Relation with Provenance + Normal Data

- **Data**: result tuple + all tuples from a witness list
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)
  - Generate self-explanatory names: P(a)

Example (Provenance Representation)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result + Provenance

a	P(a)	P(b)	P(c)	P(d)
1	1	2	1	2
1	1	2	1	3

OF TECHNOLOGY

Solution

Relation with Provenance + Normal Data

- **Data**: result tuple + all tuples from a witness list
  - Result tuple might have to be duplicated!
- **Schema**: + input attributes (renamed)
  - Generate self-explanatory names: P(a)

Example (Provenance Representation)

R

a	b
1	2
3	4

S

c	d
1	2
1	3

$\xrightarrow{q}$

Result + Provenance

a	P(a)	P(b)	P(c)	P(d)
1	1	2	1	2
1	1	2	1	3

OF TECHNOLOGY

Representation Definition

Definition (Relational Provenance Representation)

The relational representation Q^{PI} for the PI-CS provenance of a query q is defined as:

$$Q^{PI} = \{(t \blacktriangleright w[1]' \blacktriangleright \ldots \blacktriangleright w[n]')^m \mid t^p \in Q \wedge w^m \in \mathcal{PI}(q,t)\}$$

$$w[i]' = \begin{cases} w[i] & \text{if } w[i] \neq \perp \\ null(q_i) & \text{else} \end{cases}$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Discussion

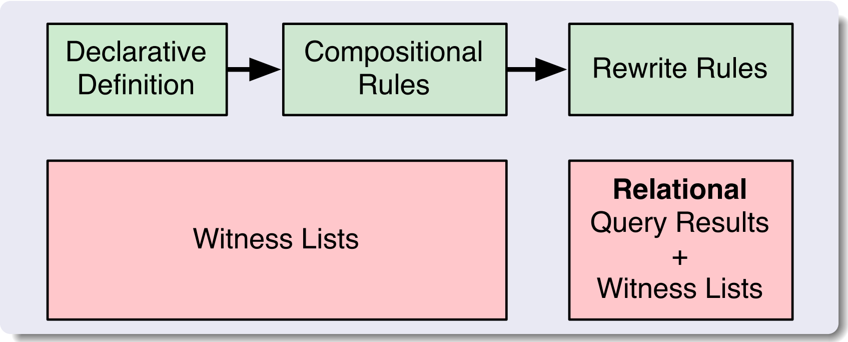
- Comprehensible by human?
  - Yes, complete tuples
  - Better than alternatives, e.g., sets of tuple ids
- Query-able?
  - Standard relation $\Rightarrow$SQL
  - Same language for to query data + provenance!
- Relationship data + provenance
  - Explicitly modelled
- Disadvantages
  - Does not exploit overlap in provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

1. Witness List based Provenance Models
   - Introduction
   - Provenance Model
   - Compositional Rules
   - Provenance Representation
   - Provenance Generation through Query Rewrite
   - Implementation
   - Recap

Approach Roadmap - Compositional Rules



Problem Statement

How to generate provenance?

- According to Provenance Semantics
- Using Perm representation

Challenges

- Storage Requirements: size of provenance
- Time-efficient: complex operations
- How access necessary information?

Perm Provenance Generation

On-demand

- Observation: provenance not needed for every query

ILLINOIS INSTITUTE OF TECHNOLOGY

Perm Provenance Generation

On-demand

- Observation: provenance not needed for every query
- ⇒Only generate when requested by the user

Perm Provenance Generation

On-demand

- Observation: provenance not needed for every query
- ⇒Only generate when requested by the user
- ⇒Solves storage problem

ILLINOIS INSTITUTE OF TECHNOLOGY

Perm Provenance Generation

On-demand

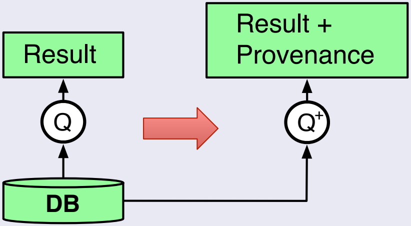
- Observation: provenance not needed for every query
- ⇒Only generate when requested by the user
- ⇒Solves storage problem
- Works without storing additional data?

ILLINOIS INSTITUTE OF TECHNOLOGY

Query Rewrite

Approach

- Rewrite query $Q \rightarrow$ query Q^+
  - Q^+ computes provenance + original results of Q
  - by adding provenance to the inputs (duplicate attributes)
  - propagates provenance through the operations of the query



Query Rewrite

Approach

- Rewrite query $Q \rightarrow$ query Q^+
  - Q^+ computes provenance + original results of Q
  - by adding provenance to the inputs (duplicate attributes)
  - propagates provenance through the operations of the query
- Rewrite defined as rules
  - One rule for each "type of operation"
  - Query rewritten by recursive application to each operation

ILLINOIS INSTITUTE OF TECHNOLOGY

Query Rewrite Cont.

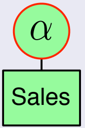
Advantages

- Q^+ is single standard SQL
  - $\Rightarrow$Reuse DB technology
- Q^+ accesses the same data as Q
  - $\Rightarrow$No need to store extra information
- Provenance generation as orthogonal SQL extension
  - $\Rightarrow$Queries over provenance benefit from optimizer
- Correctness formally proven

Example Query Rewrite

Example Query

```sql
SELECT sum(revenue) AS total, shop
FROM sales
GROUP BY shop;
```



Example (DB + Query Result)

Sales

shop	month	revenue
Migros	Jan	100
Migros	Feb	10
Migros	Mar	10
Coop	Jan	25
Coop	Feb	25

Result

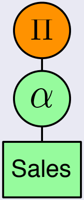
total	shop
120	Migros
50	Coop

Example Query Rewrite

Rewrite: 1

Example Query

```
SELECT total, shop, P(Q+)
FROM
    (SELECT sum(revenue) AS total, shop
    FROM sales GROUP BY shop) AS orig
```



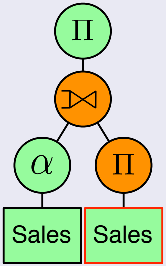
ILLINOIS INSTITUTE OF TECHNOLOGY

Example Query Rewrite

Rewrite: 2

Example Query

```
SELECT total, shop, P(Q+)
FROM
    (SELECT sum(revenue) AS total, shop
    FROM sales GROUP BY shop) AS orig
    LEFT OUTER JOIN
    (SELECT shop AS shop′, P(orig+)
    FROM sales+
    ) AS orig+
    ON (shop = shop’);
```

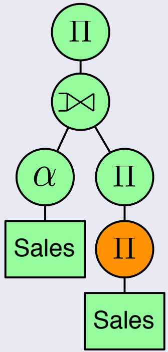


Example Query Rewrite

Rewrite: 3 Provenance Attrs: 1

Example Query

```
SELECT total, shop, P(Q+)
FROM
    (SELECT sum(revenue) AS total, shop
    FROM sales GROUP BY shop) AS orig
    LEFT OUTER JOIN
    (SELECT shop AS shop′, P(orig+)
    FROM
        (SELECT shop, month, revenue,
                shop AS P(shop),
                month AS P(month),
                revenue AS P(revenue)
        FROM sales) AS sales+
    ) AS orig+
    ON (shop = shop′);
```



Example Query Rewrite

Rewrite: 3 Provenance Attrs: 2

Example Query

```
SELECT total, shop, P(Q+)
FROM
    (SELECT sum(revenue) AS total, shop
    FROM sales GROUP BY shop) AS orig
    LEFT OUTER JOIN
    (SELECT shop AS shop′, P(shop), P(month), P(revenue)
    FROM
        (SELECT shop, month, revenue,
                shop AS P(shop), month AS P(month),
                revenue AS P(revenue)
        FROM sales) AS sales+
    ) AS orig+
    ON (shop = shop′);
```

Example Query Rewrite

Rewrite: 3 Provenance Attrs: 3

Example Query

```
SELECT total, shop, P(shop), P(month), P(revenue)
FROM
    (SELECT sum(revenue) AS total, shop
    FROM sales GROUP BY shop) AS orig
    LEFT OUTER JOIN
    (SELECT shop AS shop′, P(shop), P(month), P(revenue)
    FROM
        (SELECT shop, month, revenue,
                shop AS P(shop), month AS P(month),
                revenue AS P(revenue)
        FROM sales) AS sales+
    ) AS orig+
    ON (shop = shop′);
```

Example Rewritten Query Result

Example (Result + Provenance)

total	shop	P(shop)	P(month)	P(revenue)
120	Migros	Migros	Jan	100
120	Migros	Migros	Feb	10
120	Migros	Migros	Mar	10
50	Coop	Coop	Jan	25
50	Coop	Coop	Feb	25

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Rewritten Query Result

Example (Result + Provenance)

total	shop	P(shop)	P(month)	P(revenue)
120	Migros	Migros	Jan	100
120	Migros	Migros	Feb	10
120	Migros	Migros	Mar	10
50	Coop	Coop	Jan	25
50	Coop	Coop	Feb	25

Example Rewritten Query Result

Example (Result + Provenance)

total	shop	P(shop)	P(month)	P(revenue)
120	Migros	Migros	Jan	100
120	Migros	Migros	Feb	10
120	Migros	Migros	Mar	10
50	Coop	Coop	Jan	25
50	Coop	Coop	Feb	25

Rewrite Rules Notation

- $\mathcal{P}(q)$: Provenance attributes of Q^+
- $\mathcal{N}(R)$: Provenance attribute names for relation R
- $\blacktriangleright$: Concatenate two lists
- $=_n$: Equality that considers null values as equal

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$q = R:$	$q^{+} = \pi_{\mathbf{R},\mathbf{R}\rightarrow\mathcal{N}(\mathbf{R})}(R)$	**(R1)**
$q = \sigma_C(q_1):$	$q^{+} = \sigma_C({q_1}^{+})$	**(R2)**
$q = \pi_A(q_1):$	$q^{+} = \pi_{A,\mathcal{P}(q^{+})}({q_1}^{+})$	**(R3)**

Provenance Attribute List Rewrite

$$\mathcal{P}(q^{+}) = \begin{cases} \mathcal{P}({q_1}^{+}) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}({q_1}^{+}) \blacktriangleright \mathcal{P}({q_2}^{+}) & \text{else} \end{cases}$$

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$$q = \alpha_{G,agg}(q_1): \quad q^+ = \pi_{G,agg,\mathcal{P}(q^+)}(\alpha_{G,agg}(q_1) \;⟗_{G=_n X}\; \pi_{G \to X,\mathcal{P}(q_1{}^+)}(q_1{}^+)) \quad \textbf{(R4)}$$

$$q = \delta(q_1): \quad q^+ = q_1^+ \quad \textbf{(R5)}$$

Provenance Attribute List Rewrite

$$\mathcal{P}(q^+) = \begin{cases} \mathcal{P}(q_1{}^+) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}(q_1{}^+) \blacktriangleright \mathcal{P}(q_2{}^+) & \text{else} \end{cases}$$

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$$q = q_1 \bowtie_C q_2 : \qquad q^+ = \pi_{\mathbf{Q_1},\mathbf{Q_2},\mathcal{P}(q^+)}({q_1}^+ \bowtie_C {q_2}^+) \qquad \textbf{(R6)}$$

$$q = q_1 ⟕_C q_2 : \qquad q^+ = \pi_{\mathbf{Q_1},\mathbf{Q_2},\mathcal{P}(q^+)}({q_1}^+ ⟕_C {q_2}^+) \qquad \textbf{(R7)}$$

Provenance Attribute List Rewrite

$$\mathcal{P}(q^+) = \begin{cases} \mathcal{P}({q_1}^+) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}({q_1}^+) \blacktriangleright \mathcal{P}({q_2}^+) & \text{else} \end{cases}$$

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$$q = q_1 \cup q_2 : \quad q^+ = (q_1{}^+ \times null(\mathcal{P}(q_2{}^+))) \cup (\pi_{\mathbf{Q_1}, \mathcal{P}(q^+)}(q_2{}^+ \times null(\mathcal{P}(q_1{}^+)))) \quad \textbf{(R8)}$$

Provenance Attribute List Rewrite

$$\mathcal{P}(q^+) = \begin{cases} \mathcal{P}(q_1{}^+) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}(q_1{}^+) \blacktriangleright \mathcal{P}(q_2{}^+) & \text{else} \end{cases}$$

OF TECHNOLOGY

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$$q = q_1 \cap q_2 : \quad q^+ = \pi_{\mathbf{Q_1}, \mathcal{P}(q^+)}(\delta(q_1 \cap q_2) \bowtie_{\mathbf{Q_1} =_n X} \pi_{\mathbf{Q_1} \to X, \mathcal{P}(q_1^+)}(q_1^+) \bowtie_{\mathbf{Q_1} =_n Y} \pi_{\mathbf{Q_2} \to Y, \mathcal{P}(q_2^+)}(q_2^+)) \quad \textbf{(R9)}$$

Provenance Attribute List Rewrite

$$\mathcal{P}(q^+) = \begin{cases} \mathcal{P}(q_1^+) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}(q_1^+) \blacktriangleright \mathcal{P}(q_2^+) & \text{else} \end{cases}$$

Rewrite Rules Definitions

Definition (Rewrite Rules)

Structural Rewrite

$$q = q_1 - q_2: \quad q^+ = \pi_{\mathbf{Q_1}, \mathcal{P}(q^+)}(\delta(q_1 - q_2) \bowtie_{\mathbf{Q_1} =_n X} \pi_{\mathbf{Q_1} \rightarrow X, \mathcal{P}(q_1{}^+)}(q_1{}^+) \times null(\mathcal{P}(q_2{}^+))) \quad \textbf{(R10)}$$

Provenance Attribute List Rewrite

$$\mathcal{P}(q^+) = \begin{cases} \mathcal{P}(q_1{}^+) & \text{if } q = \sigma_C(q_1) \mid \pi_A(q_1) \mid \alpha_{G,agg}(q_1) \mid \delta(q_1) \\ \mathcal{N}(R) & \text{if } q = R \\ \mathcal{P}(q_1{}^+) \blacktriangleright \mathcal{P}(q_2{}^+) & \text{else} \end{cases}$$

Outline

Implementation

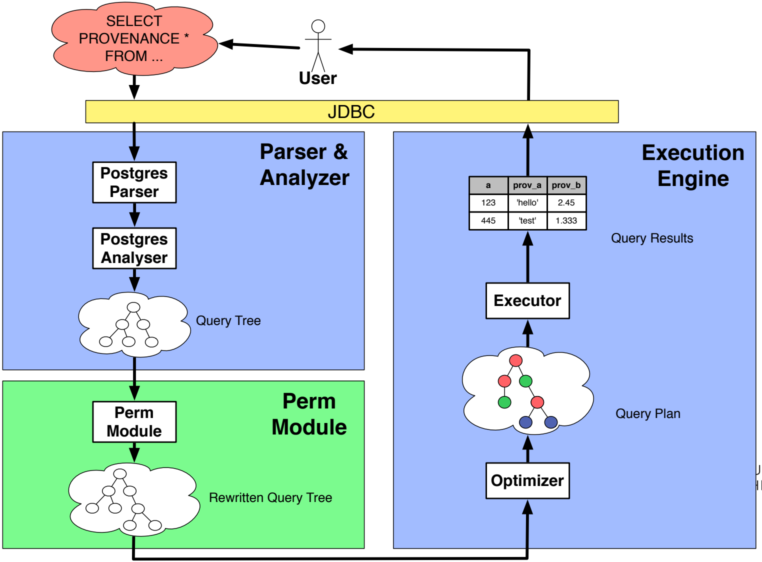
Perm

- modified *PostgreSQL* server
- *SQL-PLE*: language extension for provenance

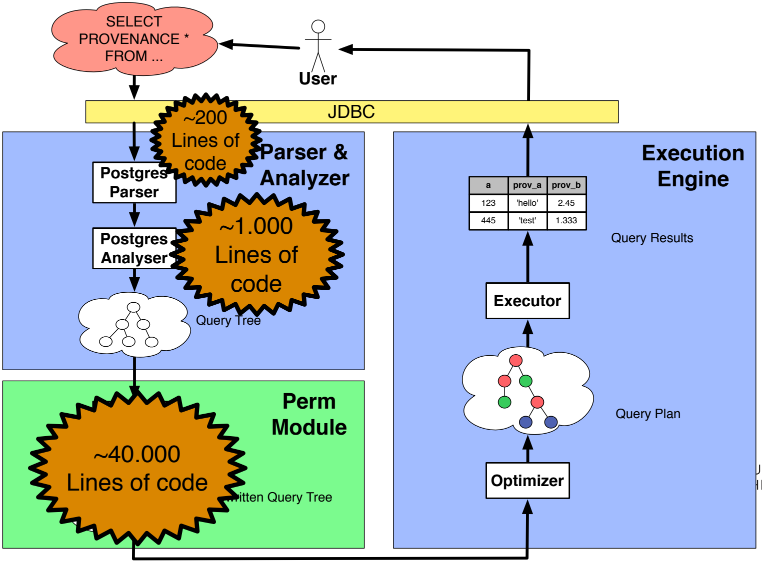
Facts

- Postgres Client Interfaces
  - JDBC
  - psql (command line)
  - ...
- Perm Module (Provenance)
  - implements query rewrites
- Open-source (http://permdbms.sourceforge.net/)

Under the Hood



Under the Hood



a	prov_a	prov_b
123	'hello'	2.45
445	'test'	1.333

Efficiency?

Provenance Generation

- On demand $\Rightarrow$ No storage required
- Single SQL query $\Rightarrow$ Reuse DBMS optimizer
- Optimizations (nested Subqueries)

Provenance Querying

- Selections applied by the query
- ... push into provenance generation

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

Recap

Witness List Model

- **Representation**: Bag of lists of tuples
  - Relational representation of query results + provenance
- **Declarative Definition**:
  - For single algebra operators
  - Sufficiency
  - Maximality (Avoid false negatives)
  - Minimality (Avoid false positives)
  - Relevance (Avoid irrelevant tuples)
- **Compositional Rules**:
  - Transitivity
- **Query Rewrite Rules**:
  - Single algebra operators
  - Propagate provenance alongside query results

Recap

Perm Implementation

- PostgreSQL extension
- `PROVENANCE` keyword activates rewrite
- Query $q \Rightarrow$ Query q^+
  - q^+ is standard SQL/algebra query
  - Use unmodified optimizer of system

ILLINOIS INSTITUTE OF TECHNOLOGY

Provenance Model Comparison

Property	Why	Lin	PI-CS
Representation	Set of Set of Tuples	List of Set of Tuples	Set/Bag of List of Tuples
Language Support	USPJ	ASPJ-Set	ASPJ-Set + Nested subqueries
Semantics	Set	Set + Bag*	Bag
Variants	Wit, Why, IWhy	Set/Bag	Influence + Copy
Definition	Decl. - Synt. - Decl./Synt.	Decl. + Synt.	Decl. + Synt.
Design Principles	Sufficiency - No false positives	Sufficiency + No false negatives + no false positives	Sufficiency + No false negatives + No false positives
Systems	-	WHIPS	Perm
Insensitivity	Yes - No - Yes	No	No

ILLINOIS INSTITUTE OF TECHNOLOGY

Literature

- Boris Glavic, Renée J. Miller, Gustavo Alonso,.
  Using Sql For Efficient Generation And Querying Of Provenance Information
  (to appear)
- Boris Glavic, Gustavo Alonso, Renée J. Miller, and Laura M. Haas.
  TRAMP: Understanding the Behavior of Schema Mappings through Provenance.
  In VLDB '10: Proceedings of the 36th International Conference on Very Large Data Bases, 2010.
- Boris Glavic.
  Perm: Efficient Provenance Support for Relational Databases.
  PhD thesis, University of Zurich, 2010.
- Boris Glavic and Gustavo Alonso.
  Provenance for Nested Subqueries.
  In EDBT '09: proceedings of the 12th international conference on extending database technology, 982–993, 2009.
- Boris Glavic and Gustavo Alonso.
  Perm: Processing Provenance and Data on the same Data Model through Query Rewriting.
  In Icde '09: proceedings of the 25th international conference on data engineering, 174–185, 2009.
- Boris Glavic and Gustavo Alonso.
  The Perm Provenance Management System in Action.
  In SIGMOD '09: Proceedings of the 35th SIGMOD International Conference on Management of Data (demonstration), 1055–1058, 2009.

ILLINOIS INSTITUTE OF TECHNOLOGY