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

I. Database Provenance

I.1 Provenance Models and Systems

Boris Glavic

September 10, 2012

Outline

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

Lineage Provenance Model

Rationale

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

Provenance Representation

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

Transformation Language

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

Lineage Provenance Model

Implementation

- Stored procedures that . . .

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

Declarative Definition

Overview

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

Model

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

OF TECHNOLOGY

Provenance Representation - Example

Example

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

$$sales^* = \{s_1\} \qquad items^* = \{i_1\}$$

Example

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

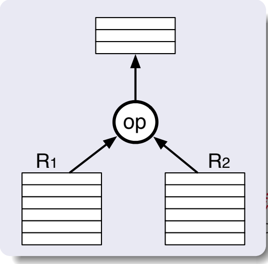
	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

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



Provenance Representation - Example

Example

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

$$sales^* = \{s_1\} \qquad items^* = \{i_1\}$$

Example

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

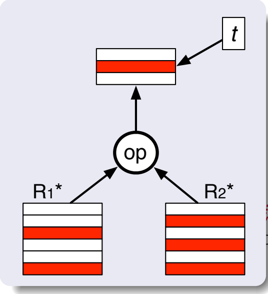
	shop	itemId	id	price
t_1	Migros	1	1	100
t_2	Migros	3	3	25
t_3	Coop	3	3	25

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



Declarative Definition - Conditions

Definition (Lineage - Single operator)

- Operator $op(R_1, R_2)$ + tuple $t \in [[op(R_1, R_2)]]$
- A list $op^{-1}_{(R_1,R_2)}(t) = (R^*_1, R^*_2)$ with $R^*_i \subseteq R_i$ is provenance of t
- iff it fulfills following three conditions

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (1)

1. Sufficiency

- $[[op(R_1^*, R_2^*)]] = \{t\}$

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (1)

1. Sufficiency

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (1)

1. Sufficiency

- $[[op(R_1^*, R_2^*)]] = \{t\}$
- $\Rightarrow$Note difference to Why-Provenance (equality vs. set inclusion)
- $\Rightarrow$Also some avoidance of false positives
  - Tuples that derive output tuples $\neq t$

Condition (1) - Example

Example

```sql
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

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

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

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

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

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

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

Condition (2)

2. Contribution (false positives)

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (2)

2. Contribution (false positives)

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (2) - Example

Example

```sql
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

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

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

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

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

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

Example

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

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

Condition (3)

3. Maximality (false negatives)

- $\nexists R'_1 \subseteq R_1, R'_2 \subseteq R_2 : (R'_1, R'_2) \geq (R^*_1, R^*_2) \wedge (R'_1, R'_2) \models (1), (2)$
- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (3)

3. Maximality (false negatives)

- $\nexists R'_1 \subseteq R_1, R'_2 \subseteq R_2 : (R'_1, R'_2) \geq (R^*_1, R^*_2) \wedge (R'_1, R'_2) \models (1), (2)$
- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)
- ⇒Why needed or good? Danger of reintroducing false positives

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (3)

3. Maximality (false negatives)

- $\nexists R'_1 \subseteq R_1, R'_2 \subseteq R_2 : (R'_1, R'_2) \geq (R^*_1, R^*_2) \land (R'_1, R'_2) \models (1), (2)$
- Provenance is maximal among the subsets of the inputs that fulfill conditions (1) and (2)
- $\Rightarrow$Why needed or good? Danger of reintroducing false positives
- $\Rightarrow$Deals with certain cases of double negation

Condition (3) - Example

Example

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

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 (3) - Example

Example

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

Example

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

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 (3) - Example

Example

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

Condition (3) - Set Difference Drawback

Example

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

Example

```
SELECT * FROM R
EXCEPT
SELECT * FROM S
```

R

	a
r_1	1

S

	b
s_1	2
s_2	3

Q

	a
t_1	1

Single Operator Definition - Recap

Definition (Lineage for Single Operators)

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

Definition

1. $[[op(R_1^*, R_2^*)]] = \{t\}$
2. $\forall i, t^* \in R_i^* : \left[\left[op(\ldots, R_{i-1}^*, \{t^*\}, R_i^*, \ldots)\right]\right] \neq \emptyset$
3. $\nexists R_1' \subseteq R_1, R_2' \subseteq R_2 : (R_1', R_2') \geq (R_1^*, R_2^*) \wedge (R_1', R_2') \models (1), (2)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Uniqueness

Solution to Definition is unique?

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

Proven for ASPJ-Set queries

Proof strategy: For each operator

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Uniqueness

Proven for ASPJ-Set queries

Proof strategy: For each operator

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

Example α

GY

Uniqueness

Proven for ASPJ-Set queries

Proof strategy: For each operator

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

Example α

- WLOG $t' \in D_1$ so that $t' \notin D_2$

Uniqueness

Proven for ASPJ-Set queries

Proof strategy: For each operator

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

Example α

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

Uniqueness

Proven for ASPJ-Set queries

Proof strategy: For each operator

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

Example α

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

Uniqueness Caveat

- New operators may break uniqueness
- Example: Nested subqueries

Extend Definition for Queries

Recursive Definition

Query q

1. $q = R$: $q^{-1}_{(R)}(t) = (\{t\})$
2. $q = op(q_1, \ldots, q_k)$: Contribution
   1. Assume $t' \in [[q_i]]$ contributes to t according to single operator definition
   2. $+\ t^* \in R_j^*$ contributes to t' transitively
3. $\Rightarrow q^{-1}_{(R_1,\ldots,R_m)}(t) = (R_1^*, \ldots, R_m^*)$ with each $t^* \in R_i^*$ fulfills two conditions above

ILLINOIS INSTITUTE OF TECHNOLOGY

Condition (3) - Double Negation Example

Example

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

Example

```
SELECT * FROM R
EXCEPT
(SELECT * FROM S
EXCEPT
SELECT * FROM T)
```

R

	a
r_1	1

S

	b
s_1	1
s_2	2

T

	b
t_1	1
t_2	3

Q

	a
t	1

Condition (3) - Double Negation Example

Example

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

Example

R

	a
r_1	1

S

	b
s_1	1
s_2	2

T

	b
t_1	1
t_2	3

Q

	a
t	1

Condition (3) - Counter Example

Example

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

Example

```
SELECT * FROM R
EXCEPT
(SELECT * FROM S
EXCEPT
SELECT * FROM T)
```

R

	a
r_1	1

S

	b
s_1	1

T

	b
t_1	1
t_2	3

Q

	a
t	1

Condition (3) - Counter Example

Example

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

Example

R

	a
r_1	1

S

	b
s_1	1

T

	b
t_1	1
t_2	3

Q

	a
t	1

Extend Definition for Sets of Outputs

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

Sets of Outputs - Example

Example

$$T = \{t_1, t_2\}$$

Example

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

	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

Sets of Outputs - Example

Example

$T = \{t_1, t_2\}$

$sales^* = \{s_1, s_2\}$ $items^* = \{i_1, i_2\}$

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

Bag Semantics?

- Provenance not unique anymore
- $\Rightarrow$**Derivation Set**: set of all derivations
- $\Rightarrow$**Derivation Pool**: Union of all derivations

ILLINOIS INSTITUTE OF TECHNOLOGY

Insensitivity to Query Rewrite

Counter Example

- $q_1 = (R \cup S)$
- $q_2 = (S \cup R)$
- $q_1^{-1}(t) = (R^*, S^*)$
- $q_2^{-1}(t) = (S^*, R^*)$

Declarative Lineage Definition - Discussion

Advantages

- Better for a few negation cases
- Straight-forward extension for new operators
- Size limited
- Studied for aggregation + all set ops

Disadvantages

- Strange behaviour for some negation cases
- Awkward modelling of bag semantics
- Assumption of transitivity
- Non-relational model: lists of relations

Notational Remarks

Use Notation introduced by authors

- Helps understanding their framework
- Direct translation to the standard notation we will be using through-out the course:
  - Assume instance $I = (R_1, \ldots, R_n)$
  - $q^{-1}_{(R_{i_1}, \ldots, R_{i_m})}(t)$
  - $\Leftrightarrow Lin(q, t, I)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

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

Tracing Rules

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Tracing Rules

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

For single operators:

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Tracing Rules

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

For queries:

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

Single operators

Rules

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Aggregation

Query

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

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

Example

sales

	shop	item	price
s_1	Migros	lawnmower	100
s_2	Migros	Shovel	25
s_3	Coop	Shovel	25

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

OF TECHNOLOGY

Example Aggregation

Query

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

$$q = \alpha_{shop,sum(price)}(sales)$$
$$q^{-1}(t) = \sigma_{shop=t.shop}(sales)$$

Example

sales

	shop	item	price
s_1	Migros	lawnmower	100
s_2	Migros	Shovel	25
s_3	Coop	Shovel	25

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

Example Aggregation

Query

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

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

Example

sales

	shop	item	price
s_1	Migros	lawnmower	100
s_2	Migros	Shovel	25
s_3	Coop	Shovel	25

result

	shop	rev
t_1	Migros	125
t_2	Coop	25

Queries

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

ILLINOIS INSTITUTE OF TECHNOLOGY

SPJ Query

Query

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

Tracing query(ies)

- $Split_{R_1,\dots,R_m}(\sigma_{A=t \wedge C}(R_1 \bowtie_{C_1} \dots \bowtie_{C_{m-1}} R_m))$

Split operator

- $Split_{A_1,\dots,A_m}(R) = (\pi_{A_1}(R), \dots, \pi_{A_m}(R))$

ILLINOIS INSTITUTE OF TECHNOLOGY

ASPJ Query

Query

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

Tracing query(ies)

- $Split_{R_1,\ldots,R_m}(\sigma_{G=t.G \wedge C}(R_1 \bowtie_{C_1} \ldots \bowtie_{C_{m-1}} R_m))$

ASPJ Discussion

- Each ASPJ-segment has to be traced on its own
- Tracing query needs access to inputs of segment
- $\Rightarrow$Need to store intermediate results for each segment
- $\Rightarrow$Or recompute large parts of the query several times

ILLINOIS INSTITUTE OF TECHNOLOGY

Queries with Set Operations

Union

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

Set Difference

- Needs single D-segment

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

Approach

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Query Normalization + Segmentation

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Tracing Queries

- How to implement *SPLIT* operator?
  - Store results of *SPLIT* input
  - One query to extract everything from stored *SPLIT* input
- Other tracing queries
  - $\Rightarrow$Use normal query functionality of DBMS

ILLINOIS INSTITUTE OF TECHNOLOGY

Discussion

Advantages

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

Disadvantages

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

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

View update - Propagate Deletions to Base Relations

Problem

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

Idea

- Provenance assumed to be available
  - In contrast to view maintenance could be ok compute on the fly!
- Use provenance to help us determine which inputs to delete from input

Deletion Propagation Example

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
```

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Project

	Name	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Assigned

	Project	Emp
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Deletion Propagation Example

Example

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

ActiveCS

	Emp
t_1	Peter
t_2	Gertrud

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Project

	Name	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Assigned

	Project	Emp
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

View Deletion - Approach

Preliminaries

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

Approach

- Deleting whole provenance?
  - $\Rightarrow$probably unnecessary side-effects
- Which (if any) subset of provenance can be deleted with-out side-effects?
- Part of provenance may be exclusive to t?
  - Deleting exclusive tuple $\Rightarrow$no side-effects

View Deletion - Exclusive Lineage

Definition (Exclusive Lineage)

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

Deletion Propagation Example

Example

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

$$E^* = \{e_2\} \qquad A^* = \{a_2, a_3\} \qquad P^* = \{p_1, p_2\}$$

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Project

	Name	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Assigned

	Project	Emp
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Deletion Propagation Example

Example

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

$$E^* = \{e_2\} \qquad A^* = \{a_2, a_3\} \qquad P^* = \{p_1, p_2\}$$

$$E^{**} = \{e_2\} \qquad A^{**} = \{a_2, a_3\} \qquad P^{**} = \{p_2\}$$

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Project

	Name	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Assigned

	Project	Emp
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

Deletion Propagation Example

Example

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

$$E^* = \{e_2\} \qquad A^* = \{a_2, a_3\} \qquad P^* = \{p_1, p_2\}$$

$$E^{**} = \{e_2\} \qquad A^{**} = \{a_2, a_3\} \qquad P^{**} = \{p_2\}$$

Exact solution!

Employee

	Id	Name
e_1	1	Peter
e_2	2	Gertrud
e_2	3	Michael

Project

	Name	Dep
p_1	Server	CS
p_2	Webpage	CS
p_3	Fire CS	HR

Assigned

	Project	Emp
a_1	Server	1
a_2	Server	2
a_3	Webpage	2
a_4	Fire CS	3

View Deletion - Algorithm

Algorithm

1. Compute lineage
2. Compute exclusive lineage
   - `For each` $t_i \in R_i^* : \{t\} = v(R_1, \ldots \{t_i\} \ldots, R_n) \Rightarrow t_i \in R_i^{**}$
3. Delete exclusive lineage: Delete t? $\Rightarrow$done!
4. `For each` i: if deletion R_i^* ...
   - causes exact deletion of t? $\Rightarrow$done!
   - otherwise store side-effect size
5. `For each` k: enumerate subsets of Lineage of size k
   - `For each` subset: delete and compute side-effects

View Deletion - Discussion

Lineage does not model which tuples were used together

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

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

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

Recap

Lineage Model

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

OF TECHNOLOGY

Recap

WHIPS Implementation

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

Provenance Model Comparison

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

Literature I

Yingwei Cui.
Lineage Tracing in Data Warehouses.
PhD thesis, Stanford University, 2002.

Yingwei Cui and Jennifer Widom.
Lineage Tracing for General Data Warehouse Transformations.
In VLDB '01: Proceedings of the 26th International Conference on Very Large Data Bases, 471–480, 2001.

Yingwei Cui and Jennifer Widom.
Run-time Translation of View Tuple Deletions using Data Lineage.
Technical report, Stanford University, 2001.

Yingwei Cui, Jennifer Widom, and Janet L. Wiener.
Tracing the Lineage of View Data in a Warehousing Environment.
ACM Transactions on Database Systems (TODS), 25(2):179–227, 2000.

Yingwei Cui and Jennifer Widom.
Practical Lineage Tracing in Data Warehouses.
In ICDE '00: Proceedings of the 16th International Conference on Data Engineering, 367–378, 2000.

Yingwei Cui and Jennifer Widom.
Storing Auxiliary Data for Efficient Maintenance and Lineage Tracing of Complex Views.
In DMDW '00: Proceedings of the 2th International Workshop on Design and Management of Data Warehouses, 2000.

ILLINOIS INSTITUTE OF TECHNOLOGY

Literature II

Yingwei Cui and Jennifer Widom.
Lineage Tracing in a Data Warehousing System.
In ICDE '00: Proceedings of the 16th International Conference on Data Engineering (demonstration), 683–684, 2000.