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

I. Database Provenance

I.1 Provenance Models and Systems

Boris Glavic

April 29, 2020

Outline

Provenance Model

- Data Provenance: Which input data contributed to which output data?

ILLINOIS INSTITUTE OF TECHNOLOGY

Provenance Model

- Data Provenance: Which input data contributed to which output data?
- but . . . what does “contributed to” really mean?
- Provenance model models “contribution”

ILLINOIS INSTITUTE OF TECHNOLOGY

Why more than one Provenance Model?

- **Evolution** - Improving on previous models?

ILLINOIS INSTITUTE OF TECHNOLOGY

Why more than one Provenance Model?

- **Evolution** - Improving on previous models?
- **Granularities** - Provenance for different granularities is different

ILLINOIS INSTITUTE OF TECHNOLOGY

Why more than one Provenance Model?

- **Evolution** - Improving on previous models?
- **Granularities** - Provenance for different granularities is different
- **Transformation Language** - E.g., subsets of SQL

ILLINOIS INSTITUTE OF TECHNOLOGY

Why more than one Provenance Model?

- **Evolution** - Improving on previous models?
- **Granularities** - Provenance for different granularities is different
- **Transformation Language** - E.g., subsets of SQL
- **Set/Bag semantics**

ILLINOIS INSTITUTE OF TECHNOLOGY

Why more than one Provenance Model?

- **Evolution** - Improving on previous models?
- **Granularities** - Provenance for different granularities is different
- **Transformation Language** - E.g., subsets of SQL
- **Set/Bag semantics**
- **Computation/Size**
  - How large is Provenance?
  - How hard to compute?

ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

Database Schema

- **Database schema** $\mathcal{S}$: Set of relation schemata
  - $\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$
- **Relation schema** $\mathcal{R}$: Name + list of attribute name - domain pairs
  - $\mathcal{R}(A_1 : D_1, \ldots, A_n : D_n)$
  - $\mathcal{R}$ = Relation name
  - A_i = Attribute name
  - D_i = Domain name

ILLINOIS INSTITUTE OF TECHNOLOGY

Database Schema

- **Database schema** $\mathcal{S}$: Set of relation schemata
  - $\mathcal{S} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$
- **Relation schema** $\mathcal{R}$: Name + list of attribute name - domain pairs
  - $\mathcal{R}(A_1 : D_1, \ldots, A_n : D_n)$
  - $\mathcal{R}$ = Relation name
  - A_i = Attribute name
  - D_i = Domain name

Example

$\texttt{PersonDB} = \{Person, Address\}$
$Person(Name : String, AddrId : Int)$
$Address(Id : Int, City : String, Street : String)$

Database Instance

- **Database Instance** I: Set of relations confirming to schema $\mathcal{S}$
  - One relation instance for each relation schema in $\mathcal{S}$
- **Relation Instance** R: Set of tuples confirming to relation schema $\mathcal{R}$
- **Tuple** t: List of values
  - $(d_1, \ldots, d_n)$ with $\forall i \in \{1, \ldots, n\} : d_i \in D_i$
  - $\Rightarrow d_i$ is an element from domain D_i

Example

Person

	Name	AddrId
p_1	Peter	1
p_2	Alice	1
p_3	Heinz	2

Address

	Id	City	Street
a_1	1	Chicago	51st
a_2	2	Evanston	10th

GY

Query Evaluation

Query

- Expression q in some query language (e.g., SQL)
  - Relation schemata $\rightarrow$ (result) relation schema
- Defined for one or more schemata:
  - Only accesses relations, attributes from schema

Query Evaluation

- $Q(I)$:
  - Evaluating query q for schema $\mathcal{S}$
  - Over some instance I for schema $\mathcal{S}$
  - use Q if I clear

GY

Query Evaluation

Example

Query q: $\pi_{name}(Person)$

Person

	Name	AddrId
p_1	Peter	1
p_2	Alice	1
p_3	Heinz	2

$Q(I)$

	Name
t_1	Peter
t_2	Alice
t_3	Heinz

ILLINOIS INSTITUTE OF TECHNOLOGY

Why-Provenance

Rationale

- Models which input tuples are sufficient to derive an output tuple t of query Q

Why-Provenance

Rationale

- Models which input tuples are sufficient to derive an output tuple t of query Q

Provenance Representation

- A set of witnesses
- A witness w is set of tuples
- $\Rightarrow$Set-semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

Witness

- Witness w for a tuple t in query result $Q(I)$
  - Intuitively: Sufficient set of tuples to derive t using q

Definition (Witness)

w is a witness for a tuple t in a query result $Q(I)$ iff:

1. $w \subseteq I$: Subset of all tuples in I
2. $t \in Q(w)$: Tuple in result of evaluating query over w

ILLINOIS INSTITUTE OF TECHNOLOGY

Witness Example

Example

```sql
SELECT shop, price
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

$Q(I)$

	shop	price
t_1	Migros	100
t_2	Migros	25
t_3	Coop	25

Witness Example

Example

- Witnesses for t_1
- $w_1 = \{s_1, i_1\}$: Is witness: $Q(w_1) = \{t_1\}$
- $w_2 = I$: Is witness: $Q(w_2) = Q(I)$
- $w_3 = \{s_1, i_1, i_3\}$: Is witness: $Q(w_3) = \{t_1\}$
- $w_4 = \{s_2, i_1, i_3\}$: No witness: $Q(w_3) = \{t_2\}$

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

$Q(I)$

	shop	price
t_1	Migros	100
t_2	Migros	25
t_3	Coop	25

GY

Set of Witnesses

- Given a tuple t in result of a query q over instance I
- Set of witnesses for t: $Wit(q, t, I)$
  - Use $Wit(q, t)$ or $Wit(t)$ if other parameters fixed

Definition (Witness set)

$Wit(q, t, I) = \{w \mid w \text{ is witness of } t \text{ for } Q(I)\}$

ILLINOIS INSTITUTE OF TECHNOLOGY

Set of Witnesses

Example

$$Why(q, t_1, I) = \{w_1, \ldots, w_n\}$$

$$w_1 = \{s_1, i_1\} \qquad w_2 = \{s_1, s_2, i_1\}$$

$$w_3 = \{s_1, i_1, i_2\} \qquad w_4 = \{s_1, i_1, i_3\}$$

$$\ldots\ldots \qquad w_n = I$$

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

$Q(I)$

	shop	price
t_1	Migros	100
t_2	Migros	25
t_3	Coop	25

GY

Properties

1. Instance is trivial witness
   - $I \in Wit(q, t, I)$ for any q and t
2. Superset of witness is also witness
   - $w \in Wit(q, t, I) \Rightarrow \forall I' \subset I : (w \cup I') \in Wit(q, t, I)$
   - Only for positive operators!
   - $\Rightarrow$Irrelevant tuples
3. Independent of query language (query = black box)

ILLINOIS INSTITUTE OF TECHNOLOGY

Space Complexity

- For positive operators:
  - Given a witness w
  - Can create new witnesses by deciding for each tuple in $I - w$ whether it should be in the witness
  - $\Rightarrow 2^{\|I-w\|} - 1$ additional witnesses of size up to $\|I\|$
  - $\Rightarrow O(2^{\|I\|})$ witnesses of size up

ILLINOIS INSTITUTE OF TECHNOLOGY

Space Complexity

Example

Instance $I = \{t_1, t_2, t_3, t_4\}$
Witness $w = \{t_1, t_2\}$
$I - w = \{t_3, t_4\}$ with $\|I - w\| = 2$
Either include t_3 or t_4 or both
$\Rightarrow 3 = 2^{\|I-w\|} - 1$ choices

$$w' = \{t_1, t_2, t_3\} \qquad w'' = \{t_1, t_2, t_4\} \qquad w'' = \{t_1, t_2, t_3, t_4\}$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Time Complexity

How to compute?

- Formalism gives no hint
- Straight forward approach:
    - For each subset I' of I
    - Compute $Q(I')$
    - Includes $t \Rightarrow I'$ is witness

ILLINOIS INSTITUTE OF TECHNOLOGY

Time Complexity

How to compute?

- Formalism gives no hint
- Straight forward approach:
  - For each subset I' of I
  - Compute $Q(I')$
  - Includes $t \Rightarrow I'$ is witness

Complexity

- Number of subsets of I times compute q
  - $2^{\|I\|}$ subsets
  - Complexity of $Q(I')$: polynomial in I

Time Complexity

How to compute?

- Formalism gives no hint
- Straight forward approach:
  - For each subset I' of I
  - Compute $Q(I')$
  - Includes $t \Rightarrow I'$ is witness

Complexity

- Number of subsets of I times compute q
  - $2^{\|I\|}$ subsets
  - Complexity of $Q(I')$: polynomial in I
- $\Rightarrow O(2^{\|I\|})$

Time Complexity

How to compute?

- Formalism gives no hint
- Straight forward approach:
  - For each subset I' of I
  - Compute $Q(I')$
  - Includes $t \Rightarrow I'$ is witness
- If positive ops $\Rightarrow$avoid computing $Q(I')$

Complexity

- Number of subsets of I times compute q
  - $2^{\|I\|}$ subsets
  - Complexity of $Q(I')$: polynomial in I
- $\Rightarrow O(2^{\|I\|})$

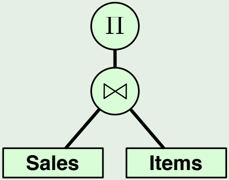
GY

Proof-Witness

Rationale

- $Wit(q, t, I)$ contains witnesses with irrelevant tuples
- Define witnesses based on query expression
- $\Rightarrow$Hope to avoid irrelevant tuples
- Include one tuple from each leaf of the alegra tree of q

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

Proof-Witness

- A proof-witness
  - contains one tuple from each algebra tree leaf of q
  - definition dependent on query expression (syntax)

ILLINOIS INSTITUTE OF TECHNOLOGY

Why-Provenance

- $Why(q, t, I)$ is set of all proof-witnesses for t
- Recursive compositional definition for algebra operators:
  - Accessing an relation $R \Rightarrow$ Base case
  - Selection σ_C
  - Projection π_A
  - Join $\bowtie_C$
  - Union $\cup$

ILLINOIS INSTITUTE OF TECHNOLOGY

Why-Provenance

- $Why(q, t, I)$ is set of all proof-witnesses for t
- Recursive compositional definition for algebra operators:

Definition

$$
\begin{aligned}
Why(R, t, I) &= \{\{t\}\} \\
Why(\sigma_C(q), t, I) &= Why(q, t, I) \\
Why(\pi_A(q), t, I) &= \bigcup_{u \in Q(I): u.A = t} Why(q, u, I) \\
Why(q_1 \bowtie_C q_2, t, I) &= \{(w_1 \cup w_2) \mid w_1 \in Why(q_1, t_1, I) \\
&\quad \wedge t_1 = t.Q_1 \wedge w_2 \in Why(q_2, t_2, I) \\
&\quad \wedge t_2 = t.Q_2\} \\
Why(q_1 \cup q_2, t, I) &= Why(q_1, t, I) \cup Why(q_2, t, I)
\end{aligned}
$$

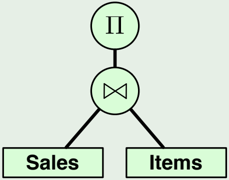
GY

Example Why-provenance computation

Example

```sql
SELECT shop, price
FROM sales, items
WHERE itemId = id
```

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



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

Example Why-provenance computation

Example

$$q = \pi_{shop,price}(q_1)$$
$$q_1 = sales \bowtie_{itemId=id} items$$

$Q(I)$

	shop	price
t_1	Migros	100
t_2	Migros	25
t_3	Coop	25

$Q_1(I)$

	shop	itemId	id	price
j_1	Migros	1	1	100
j_2	Migros	3	3	25
j_3	Coop	3	3	25

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 1) Substitute definitions

$$Why(q,t) = \bigcup_{u \in Q_1(I):u.(shop,price)=t} Why(q_1,u)$$

$$\begin{aligned} Why(q_1,t) =& \{(w_1 \cup w_2) \mid w_1 \in Why(sales,t_1) \\ & \wedge t_1 = t.sales \wedge w_2 \in Why(items,t_2) \\ & \wedge t_2 = t.items\} \end{aligned}$$

$$Why(sales,t) = \{\{t\}\}$$

$$Why(sales,t) = \{\{t\}\}$$

GY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 2) Evaluate for tuple top-down - 1

$$\Rightarrow Why(q, t_1) = Why(q_1, j_1)$$

$$\begin{aligned} Why(q_1, t) = \{(w_1 \cup w_2) \mid\ & w_1 \in Why(sales, t_1) \\ & \wedge t_1 = t.sales \wedge w_2 \in Why(items, t_2) \\ & \wedge t_2 = t.items\} \end{aligned}$$

$$Why(sales, t) = \{\{t\}\}$$

$$Why(sales, t) = \{\{t\}\}$$

GY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 3) Evaluate for tuple top-down - 2

$$
\begin{aligned}
Why(q, t_1) &= Why(q_1, j_1) \\
\Rightarrow Why(q_1, j_1) &= \{(w_1 \cup w_2) \mid w_1 \in Why(sales, s_1) \\
&\qquad \wedge w_2 \in Why(items, i_1)\} \\
Why(sales, t) &= \{\{t\}\} \\
Why(sales, t) &= \{\{t\}\}
\end{aligned}
$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 4) Evaluate for tuple top-down - 3

$$Why(q, t_1) = Why(q_1, j_1)$$

$$Why(q_1, j_1) = \{(w_1 \cup w_2) \mid w_1 \in Why(sales, s_1) \wedge w_2 \in Why(items, i_1)\}$$

$$\Rightarrow Why(items, i_1) = \{\{i_1\}\}$$

$$\Rightarrow Why(sales, s_1) = \{\{s_1\}\}$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 5) Substitute results bottom-up - 1

$$Why(q, t_1) = Why(q_1, j_1)$$
$$\Rightarrow Why(q_1, j_1) = \{\{s_1, i_1\}\}$$
$$Why(items, i_1) = \{\{i_1\}\}$$
$$Why(sales, s_1) = \{\{s_1\}\}$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Example Why-provenance computation

Example

Compute $Why(q, t_1, I)$: 6) Substitute results bottom-up - 2

$$\Rightarrow Why(q, t_1) = \{\{s_1, i_1\}\}$$
$$Why(q_1, j_1) = \{\{s_1, i_1\}\}$$
$$Why(items, i_1) = \{\{i_1\}\}$$
$$Why(sales, s_1) = \{\{s_1\}\}$$

ILLINOIS INSTITUTE OF TECHNOLOGY

Excursion: Query Equivalence

- Queries are equivalent if
    - Produce same result for all possible instances

Definition

Query equivalence Two queries q_1 and q_2 are equivalent ($q_1 \equiv q_2$) iff

- $\forall I : Q_1(I) = Q_2(I)$

Example

$$q_1 = R \cup S$$
$$q_2 = S \cup R$$

Insensitivity to Query Rewrite

What is that?

- Equivalent queries have same black-box behaviour
- Should have same provenance?

Definition (Insensitivity to Query Rewrite)

A provenance model $\mathcal{P}$ is called insensitive to query rewrite iff

- $q_1 \equiv q_2 \Rightarrow \mathcal{P}(q_1, t) = \mathcal{P}(q_2, t)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Insensitivity for *Wit* and *Why*

$Wit(q, t)$

- Is insensitive
- Follows from the definition

$Why(q, t)$

Not insensitive:

- Counterexample

Insensitivity for *Wit* and *Why*

Example

$$q_1 = \pi_a(R)$$
$$Why(q_1, t_1) = \{\{r_1\}\}$$
$$Why(q_1, t_2) = \{\{r_2\}\}$$

$$q_2 = \pi_a(R \bowtie_{b=c} \pi_{b \to c}(R))$$
$$Why(q_2, t_1) = \{\{r_1\}, \{r_1, r_2\}\}$$
$$Why(q_2, t_2) = \{\{r_2\}, \{r_1, r_2\}\}$$

Q_1

	a
t_1	1
t_2	2

Q_2

	a
t_1	1
t_2	2

R

	a	b
r_1	1	1
r_2	2	1

Properties

- A proof-witness is also a witness
  - $\Rightarrow Why(q,t,I) \subseteq Wit(q,t,I)$
  - **Proof**: Induction over the structure of a query
- Only for operators we defined a rule!
- Less redundancy
- Sensitive to query rewrite
- Computation efficient: Brute force approach
  - Store all intermediate results
  - Recursive implementation of rules
  - Trace back one step at a time

ILLINOIS INSTITUTE OF TECHNOLOGY

Space Complexity

- Limited by the number of algebra tree leafs (relations)
- $R_1, \ldots, R_n$ relations
- $\Rightarrow R_1 \times \ldots \times R_n$
- In practise usually much smaller!
  - E.g., join on foreign key $\Rightarrow$one witness list in provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Time Complexity

Straight forward implementation

- Compute all intermediate results
  - x operators in query
  - Each operator polynomial in instance size
- Each tracing step can be expressed as query over intermediate result
  - $\Rightarrow$polynomial in instance size
- $\Rightarrow$polynomial in instance size

ILLINOIS INSTITUTE OF TECHNOLOGY

Minimal Why-Provenance

Rationale

- Make Why-provenance insensitive to query rewrite
- Without loosing positive properties
  - Computable (within reasonable time bounds)
  - Size

Minimal Elements of a Set

Definition

Minimal Elements

- Given set of sets S
- Minimal element: does not contain in other elements
- Element $e \in S$ is minimal iff $\nexists e' \in S : e' \subset e$

Example

- $S = \{\{a, b, c\}, \{a, b\}, \{a, c\}\}$
- $\{a, b, c\}$ is not minimal, e.g., $\{a, b\}$ contained
- $\{a, b\}$ is minimal, no other elements contained in $\{a, b\}$

Minimal Why-Provenance

- Minimal elements of Why-Provenance
- $\Rightarrow$removes redundancy

Definition (Minimal Why-Provenance)

$$
\begin{aligned}
MWhy(q,t,I) = &\{w \mid w \in Why(q,t,I) \\
&\wedge \nexists w' \in Why(q,t,I) : w' \subset w\}
\end{aligned}
$$

Minimal Why-provenance example

Example

$$q_1 = \pi_a(R)$$

$$Why(q_1, t_1) = \{\{r_1\}\}$$
$$MWhy(q_1, t_1) = \{\{r_1\}\}$$

$$q_2 = \pi_a(R \bowtie_{b=c} \pi_{b \rightarrow c}(R))$$

$$Why(q_2, t_1) = \{\{r_1\}, \{r_1, r_2\}\}$$
$$MWhy(q_2, t_1) = \{\{r_1\}\}$$

Q_1

	a
t_1	1
t_2	2

Q_2

	a
t_1	1
t_2	2

R

	a	b
r_1	1	1
r_2	2	1

ILLINOIS INSTITUTE OF TECHNOLOGY

Alternative Definition

- Use $Wit(q, t, I)$ instead of $Why(q, t, I)$
- $\Rightarrow$Same results?

Definition ($MWit(q, t, I)$)

$$\begin{aligned} MWit(q, t, I) = &\{w \mid w \in Wit(q, t, I) \\ &\wedge \nexists w' \in Wit(q, t, I) : w' \subset w\} \end{aligned}$$

Equivalence of $MWhy(q, t, I) = MWit(q, t, I)$

$MWit(q, t, I) \subseteq Why(q, t, I)$

1. $w \in Wit(q, t, I) \Rightarrow \exists w' \subseteq w : w' \in Why(q, t, I)$
   - **Proof**: Induction over operators in q
2. $Why(q, t, I) \subseteq Wit(q, t, I)$

- $w \in MWit(q, t, I)$
- $\Rightarrow \exists w' \subseteq w : w' \in Why(q, t, I)$ (using 1)
- $\Rightarrow w' \in Wit(q, t, I)$ (using 2)
- $\Rightarrow w' = w$ (because w is minimal)

ILLINOIS INSTITUTE OF TECHNOLOGY

Equivalence of $MWhy(q, t, I) = MWit(q, t, I)$

$MWit(q, t, I) \subseteq MWhy(q, t, I)$

- Every witness in $MWit(q, t, I)$ is also in $MWhy(q, t, I)$
- Take one witness $w \in MWit(q, t, I)$
- $\Rightarrow w \in Why(q, t, I)$ $(MWit(q, t, I) \subseteq Why(q, t, I))$
- $\Rightarrow w \in MWhy(q, t, I)$
  - Assume: $w \notin MWhy(q, t, I)$
  - $\Rightarrow \exists w' \subset w : w' \in MWhy(q, t, I)$
  - $\Rightarrow w' \in Wit(q, t, I)$ $(Why(q, t, I) \subseteq Wit(q, t, I))$
  - $\Rightarrow w$ not minimal Contradiction!

ILLINOIS INSTITUTE OF TECHNOLOGY

Equivalence of $MWhy(q,t,I) = MWit(q,t,I)$

$MWhy(q,t,I) \subseteq MWit(q,t,I)$

- $w \in MWhy(q,t,I)$
- $\Rightarrow w \in Why(q,t,I) \Rightarrow w \in Wit(q,t,I)$
  $(MWhy(q,t,I) \subseteq Why(q,t,I) \subseteq Wit(q,t,I))$
- Suppose $\exists w' \subseteq w : w' \in Wit(q,t,I)$
- $\Rightarrow \exists w'' \subseteq w' : w'' \in Why(q,t,I)$
  $(w \in Wit(q,t,I) \Rightarrow \exists w' \subseteq w : w' \in Why(q,t,I))$
- $\Rightarrow w = w' = w''$ (w is minimal in $Why(q,t,I)$)

ILLINOIS INSTITUTE OF TECHNOLOGY

Equivalence of $MWhy(q, t, I) = MWit(q, t, I)$

$MWhy(q, t, I) = MWit(q, t, I)$

Holds because we have shown that

- $MWhy(q, t, I) \subseteq MWit(q, t, I)$
- $MWit(q, t, I) \subseteq MWhy(q, t, I)$

ILLINOIS INSTITUTE OF TECHNOLOGY

Insensitivity to Query Rewrite

- $MWit(q, t, I)$ is insensitive
  - Same argument as for $Wit(q, t, I)$
  - $\Rightarrow$Condition on black-box behaviour
- $MWhy(q, t, I) = MWit(q, t, I)$
- $\Rightarrow MWhy(q, t, I)$ is insensitive

ILLINOIS INSTITUTE OF TECHNOLOGY

Properties

- Insensitive to query rewrite
  - Side-effect: Remove redundancy
- Small size
- *MWhy*
  - Computation efficient: Compute *Why* + containment checks
  - Limited to defined rules
- *MWit*
  - Independent of query language
  - Computation not straight-forwards

ILLINOIS INSTITUTE OF TECHNOLOGY

Space Complexity

- At most $Why(q, t, I)$
- $\Rightarrow$polynomial in $\|I\|$

ILLINOIS INSTITUTE OF TECHNOLOGY

Time Complexity

- Approach
  - Compute $Why(q, t, I)$
  - Pairwise containment checks to find minimal elements
- Polynomial in size of instance I

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion Propagation

- Given a view
- How to update the view when input tuples are deleted

Deletion Propagation

- Given a view
- How to update the view when input tuples are deleted
- $\Rightarrow$Use Why-provenance

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion Propagation using Why

Approach

- Store Why-provenance for each tuple in view v
- Upon deletion:
  - Adapt provenance
  - Delete tuples from view with empty provenance

- Set of tuples D that got deleted
- For each tuple t in view
  - Remove witnesses from $Why(t, v, I)$ that contain tuples from D

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

GY

Deletion Propagation Example

Example

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

OF TECHNOLOGY

Deletion Propagation Example

Example

- Delete tuple from `Projects`

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

Deletion Propagation Example

Example

- What would be the effect on the view?

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

GY

Deletion Propagation - Approach

Assumption

- Assume we have Why-provenance for each tuple
  - $Why(t_1) = \{\{e_1, p_1, a_1\}\}$
  - $Why(t_2) = \{\{e_2, p_1, a_2\}, \{e_2, p_2, a_3\}\}$
- Set of deleted tuples ($D = \{p_1\}$)

ILLINOIS INSTITUTE OF TECHNOLOGY

Deletion Propagation Example

Example

- $Why(t_1) = \{e_1, p_1, a_1\} \rightarrow \{\}$
- $Why(t_2) = \{\{e_2, p_1, a_2\}, \{e_2, p_2, a_3\}\} \rightarrow \{\{e_2, p_2, a_3\}\}$

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

Recap

Flavours of Why-Provenance

- Set of Witnesses
- Why-Provenance
- Minimal Why-Provenance

Concepts

- Insensitivity to Query Rewrite
- Query Equivalence
- Query language Independence
- Sufficiency

GY

Literature

James Cheney, Laura Chiticariu, and Wang-Chiew Tan.
Provenance in Databases: Why, How, and Where.
Foundations and Trends in Databases, 1(4):379–474, 2009.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan.
Why and Where: A Characterization of Data Provenance.
In ICDT '01: Proceedings of the 8th International Conference on Database Theory, 316–330, 2001.