ILLINOIS INSTITUTE OF TECHNOLOGY

CS520 Data Integration, Warehousing, and Provenance

8. Provenance

IIT DBGroup

Boris Glavic

http://www.cs.iit.edu/~glavic/

http://www.cs.iit.edu/~cs520/

http://www.cs.iit.edu/~dbgroup/

0

Outline

1

8.What is Data Provenance?

- **Metadata** describing the **origin** and **creation process** of **data**
  - **Data items**
    - Data item **granularity**
      - **A File**
      - **A Database**
      - **An Attribute value**
      - **A Row**
  - **Transformations**
    - Transformation **granularity**
      - **A program**
      - **A query**
      - **An operator in a query**
      - **A line in a program**

2

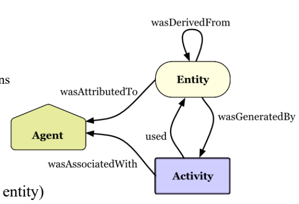
8.What is Data Provenance?

- **Provenance** records **dependencies**
  - **Data dependencies**
    - Data item x was used to generate data item y
  - **Dependencies between transformations and data**
    - Transformations generated a data item
    - Transformations used a data item

3

8. Provenance as graphs

- **Provenance graphs** (W3C PROV standard)
  - https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/
  - **Nodes**
    - **Entities**
      - what we call data items
    - **Activities**
      - what we call transformations
    - **Agents**
      - Trigger / control activities
      - E.g., users and machines

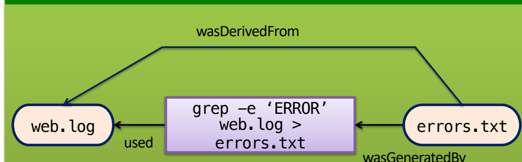


  - **Edges**
    - **wasDerivedFrom** (entity – entity)
      - Data dependencies
    - **wasGeneratedBy** (activity – entity)
      - Transformation generated an output data item
    - **used** (entity – activity)
      - Transformation read and input data item

4

8. PROV example

Example: find errors in a weblog with grep



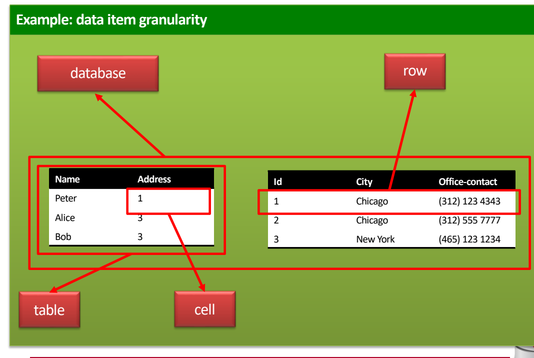
5

8. Provenance for Databases

- **Transformations**
  - **SQL queries**
  - **Updates and transactions**
  - **Procedural code**
- **Data items**
  - **Databases**
  - **Tables**
  - **Rows**
  - **Cells** (attribute value of a row)

6

8. Databases Prov. – Data items



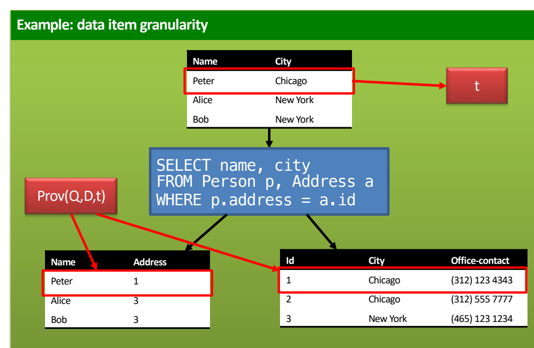
7

8. Provenance for Queries

- **Data dependencies**
  - For each **output tuple** (**cell**) of the query determine which **input tuples** (**cells**) of the query it depends on
- **Formally (kind of)**
  - Given database **D** and query **Q** and tuple **t** in **Q(D)**
    - **Prov(Q,D,t)** = the subset of **D** that was used to derive **t** through **Q**

8

8. Databases Prov. – Data items



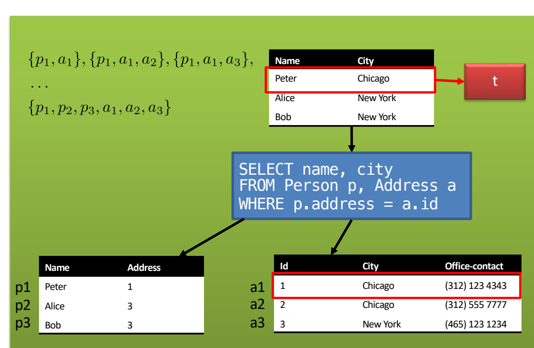
9

8. Formalizing data dependencies

- **How to formalize data dependencies?**
  - **Access**: query did read the data
    - No! Everything depends on everything!
  - **Sufficiency**: the provenance is enough to produce the result tuple **t**
    - **t** is in **Q(Prov(Q,D,t))**
    - Guarantees that everything that was needed to produce **t** is in the provenance

10

8. Sufficiency - Example



11

8. Sufficiency cont.

- **Is sufficiency enough?**
  - No, sufficiency does not prevent irrelevant inputs to be included in the provenance!
  - Sufficiency does not uniquely define provenance
- **Monotone Queries**
  - A query **Q** is monotone if

$$\forall D, D' : D \subseteq D' \Rightarrow Q(D) \subseteq Q(D')$$

- **For all monotone queries Q:**
  - If D is sufficient then so is any superset of D
  - in particular the input database D is sufficient

8. Why provenance

- **Rationale:** define provenance as the set of all sufficient subsets of the input
  - this uniquely defines provenance
  - this does not solve the redundancy issue!
- **Why provenance**:

$$Why(Q, D, t) = \{D' \mid D' \subseteq D \wedge t \in Q(D')\}$$

- Each sufficient subset of D in the why provenance is called a witness

8. Minimality

- **Rationale:**
  - Remove tuples that do not contribute to the result
  - If a subset of a witness is already sufficient then everything not in the subset is unnecessary and should be removed
- **Definition**

D' is a minimal witness for t if $\forall D' \subset D'' : t \notin Q(D'')$

8. Minimal Why provenance

- **Minimal Why provenance**:
- Only include minimal witnesses

$$MWhy(Q, D, t) = \{D' \mid D' \in Why(Q, D, t) \wedge \nexists D'' \subset D' : D'' \in Why(Q, D, t)\}$$

8. Sufficiency - Example

$$MWhy(Q, D, T) = \{p_1, a_1\}$$

Name	City
Peter	Chicago
Alice	New York
Bob	New York

t

```
SELECT name, city
FROM Person p, Address a
WHERE p.address = a.id
```

	Name	Address
p1	Peter	1
p2	Alice	3
p3	Bob	3

	Id	City	Office-contact
a1	1	Chicago	(312) 123 4343
a2	2	Chicago	(312) 555 7777
a3	3	New York	(465) 123 1234

8. Why provenance - discussion

- **Independent of query syntax**
  - Queries are treated as blackbox functions
  - Equivalent queries have the same provenance!
- How to compute this efficiently?
  - The discussion so far only gives a brute force exponential time algorithm
    - For each subset D' of D test whether it is a witness
    - Then for every witness test whether it is minimal by testing for a subset relationship with all other witnesses
  - Top-down rules that calculate MWhy in a syntax driven manner

8. MWhy – top-down recursion

- Define top-down syntax-driven rules
  - calculate a set of witnesses
  - Minimizing the result of these rules returns MWhy

$$W(R,t,I) = \{\{t\}\}$$
$$W(\sigma_\theta(Q),t,I) = W(Q,t,I)$$
$$W(\pi_A(Q),t,I) = \bigcup_{u \in Q(I): u.A = t} W(Q,u,I)$$
$$W(Q_1 \bowtie_\theta Q_2,t,I) = \{(w_1 \cup w_2) \mid w_1 \in W(Q_1,t_1,I) \wedge w_2 \in W(Q_2,t_2,I) \wedge t = (t_1,t_2)\}$$
$$W(Q_1 \cup Q_2,t,I) = W(Q_1,t,I) \cup W(Q_2,t,I)$$

18 CS520 - 8) Provenance

18

8. Why provenance – discussion 2

- **This works well for set semantics, but not bag semantics**
  - Minimization can lead to incorrect results with bag semantics
  - Treating the provenance as sets of tuples does not align well with bags
- **This only encodes data dependencies**
  - We know from which tuples we have derived a result, but not how the tuples were combined to produce the result

19 CS520 - 8) Provenance

19

8. Semiring annotations - Agenda

- **We will now discuss a model that …**
  - Provides provenance for both sets and bags
  - Allows us to track how tuples where combined
  - Can express many other provenance models including MWhy
  - Can also express bag and set semantics and other extensions of the relational model such as the incomplete databases we discussed earlier

20 CS520 - 8) Provenance

20

8. Annotations on Data

- **Annotations**
  - Allow data to be associated with additional metadata
    - Comments from users
    - Trust annotations
    - Provenance
    - …
  - Here we are interested in annotations on the tuples of a table

21 CS520 - 8) Provenance

21

8. K-relations

- **Annotation domain**
  - We fix a set K of possible annotations
  - Examples
    - Powerset(Powerset(D)) = all possible sets of witnesses
      - We can annotate each tuple with its Why or MWhy provenance
    - Natural numbers
      - We can simulate bag semantics by annotating each tuple with its multiplicity
    - A set of possible world identifiers D1 to Dn
      - Incomplete databases

22 CS520 - 8) Provenance

22

8. K-relations

- **K-relations**
  - We fix a set **K** of possible annotations
  - **K** has to have a distinguished element $\mathbf{0_K}$
  - Assume some data domain **U**
  - An n-ary K-relation is a function

$$\mathcal{U}^n \rightarrow K$$

    - We associate an annotation with every possible n-ary tuple
    - $\mathbf{0_k}$ is used to annotate tuples that are not in the relation
    - Only finitely many tuples are allowed to be mapped to a non-zero annotation

23 CS520 - 8) Provenance

23

8. Example – bag semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

Bag Semantics

Name	Address
Peter	1
Peter	1
Peter	1
Alice	3
Alice	3
Bob	3

N-relation

Name	Address	Annotation
Peter	1	3
Alice	3	2
Bob	3	1

24

8. Example – set semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

Bag Semantics

Name	Address
Peter	1
Peter	1
Peter	1
Alice	3
Alice	3
Bob	3

B-relation

Name	Address	Annotation
Peter	1	true
Alice	3	true
Bob	3	true

$\mathbb{B} = \{false, true\}$

25

8. Example – incomplete DBs

ILLINOIS INSTITUTE OF TECHNOLOGY

Incomplet Database

D_1

Name	Address
Peter	1
Peter	2
Bob	3

D_2

Name	Address
Peter	1
Alice	2
Bob	3

Ω -relation

Name	Address	Annotation
Peter	1	{D1,D2}
Peter	2	{D1}
Alice	2	{D2}
Bob	3	{D1,D2}

$$\Omega = \mathcal{P}(\{D_1, D_2\})$$
$$= \{\emptyset, \{D_1\}, \{D_2\}, \{D_1, D_2\}\}$$

26

8. Example – MWhy

ILLINOIS INSTITUTE OF TECHNOLOGY

MWhy

Name	Address
Peter	1
Peter	2
Bob	3

MWhy(p1) = {{x1}}
MWhy(p2) = {{x2,a1},{x3}}
Mwhy(p3) = {{x4,a1},{x4,a2}}

PosBool[X]-relation

Name	Address	Annotation
Peter	1	{{x1}}
Peter	2	{{x2,a1},{x3}}
Bob	3	{{x4,a1},{x4,a2}}

$$X = D$$
$$PosBool[X] = \mathcal{P}(\mathcal{P}(X))$$

27

8. K-relations – Query semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Annotated Databases are powerful**
  - We can many different types of information
  - However, what is the right query semantics?
    - e.g., bag and set semantics queries do not have the same semantics, let along queries over incomplete databases or calculating provenance
- **Query Semantics**
  - Split the query semantics into two parts
    - One part is generic and independent of the choice of K
    - One part is specific to the choice of K
  - => every K has to be paired with operations that define how annotations propagate through queries
    - The generic semantics uses these operations to calculate query result annotations

28

8. Semirings

ILLINOIS INSTITUTE OF TECHNOLOGY

- **A semiring** $\mathcal{K} = (K, \oplus_{\mathcal{K}}, \otimes_{\mathcal{K}}, 0_{\mathcal{K}}, 1_{\mathcal{K}})$
  - K is the set of elements of semiring
    - We use them as annotations
  - There are two binary operations
    $\oplus_{\mathcal{K}}, \otimes_{\mathcal{K}} : K \times K \rightarrow K$
    - We will use them to combine annotations of input tuples
      - Addition will be used to model operations that are disjunctive in nature (union, projection)
      - Multiplication will be used to model operations that are conjunctive (join)
  - Two distinguished elements $0_{\mathcal{K}}, 1_{\mathcal{K}}$

29

8. Semiring Laws

ILLINOIS INSTITUTE OF TECHNOLOGY

- **A semiring** $\mathcal{K} = (K, \oplus_{\mathcal{K}}, \otimes_{\mathcal{K}}, 0_{\mathcal{K}}, 1_{\mathcal{K}})$

$$
\begin{array}{rcll}
k_1 \oplus_{\mathcal{K}} k_2 &=& k_2 \oplus_{\mathcal{K}} k_1 & \text{(commutativity)} \\
k_1 \oplus_{\mathcal{K}} (k_2 \oplus_{\mathcal{K}} k_3) &=& (k_1 \oplus_{\mathcal{K}} k_2) \oplus_{\mathcal{K}} k_3 & \text{(associativity)} \\
k_1 \otimes_{\mathcal{K}} k_2 &=& k_2 \otimes_{\mathcal{K}} k_1 & \text{(commutativity)} \\
k_1 \otimes_{\mathcal{K}} (k_2 \otimes_{\mathcal{K}} k_3) &=& (k_1 \otimes_{\mathcal{K}} k_2) \otimes_{\mathcal{K}} k_3 & \text{(associativity)} \\
k \oplus_{\mathcal{K}} 0_{\mathcal{K}} &=& k & \text{(neutral element)} \\
k \otimes_{\mathcal{K}} 1_{\mathcal{K}} &=& k & \text{(neutral element)} \\
k \otimes_{\mathcal{K}} 0_{\mathcal{K}} &=& 0_{\mathcal{K}} & \text{(annihilation by zero)} \\
k_1 \otimes_{\mathcal{K}} (k_2 \oplus_{\mathcal{K}} k_3) &=& (k_1 \otimes_{\mathcal{K}} k_2) \oplus (k_1 \otimes_{\mathcal{K}} k_3) & \text{(distributivity)}
\end{array}
$$

30 CS520 - 8) Provenance

30

8. Semirings - Examples

ILLINOIS INSTITUTE OF TECHNOLOGY

$$
\begin{aligned}
\mathbb{N} &= (\mathbb{N}, +, \cdot, 0, 1) \\
\mathbb{B} &= (\mathbb{B}, \vee, \wedge, false, true) \\
\mathcal{K}_{MWhy}[X] &= (\mathcal{P}(\mathcal{P}(X)), \cup, \Cup, \emptyset, \{\emptyset\}) \\
\mathcal{K}_{\Omega}[X] &= (\mathcal{P}(\Omega), \cup, \cap, \emptyset, \Omega) \\
\mathbb{N}[X] &= (\mathbb{N}[X], +, \cdot, 0, 1)
\end{aligned}
$$

31 CS520 - 8) Provenance

31

8. Provenance Polynomials

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Semiring** $\mathbb{N}[X] = (\mathbb{N}[X], +, \cdot, 0, 1)$
  - N[X] is the set of all polynomials over variables X
    - Intuitively X are tuple identifiers
  - **Provenance polynomials** are used to track provenance for **bag semantics**!
  - Provenance polynomials record how a result has been derived by combining input tuples
    - Multiplication means conjunctive use (as in join)
    - Addition means disjunctive use

32 CS520 - 8) Provenance

32

8. K-relations – Query semantics

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Positive relational algebra (RA⁺)**
  - Selection, projection, cross-product, renaming, union

Union: $(R_1 \cup R_2)(t) = R_1(t) \oplus_{\mathcal{K}} R_2(t)$

Join: $(R_1 \bowtie R_2)(t) = R_1(t[R_1]) \otimes_{\mathcal{K}} R_2(t[R_2])$

Projection: $(\pi_A(R))(t) = \bigoplus_{t=t'[A]} R(t')$

Selection: $(\sigma_\theta(R))(t) = R(t) \otimes_{\mathcal{K}} \theta(t)$

$$\theta(t) = \begin{cases} 0_{\mathcal{K}} & \text{if } t \models \theta \\ 1_{\mathcal{K}} & \text{otherwise} \end{cases}$$

33 CS520 - 8) Provenance

33

8. Query Semantics - Bags

ILLINOIS INSTITUTE OF TECHNOLOGY

City	N
Chicago	1
New York	1*1+1*1 = 2

$$\pi_{City}(\sigma_{address=id}(person \times address))$$

Name	Address	N
Peter	1	1
Alice	3	1
Bob	3	1

Id	City	Office-contact	N
1	Chicago	(312) 123 4343	1
2	Chicago	(312) 555 7777	1
3	New York	(465) 123 1234	1

34 CS520 - 8) Provenance

34

8. Query Semantics - MWhy

ILLINOIS INSTITUTE OF TECHNOLOGY

City	MWhy
Chicago	$\{\{x_1, x_4\}\}$
New York	$\{\{x_2, x_6\}, \{x_3, x_6\}\}$

$$\pi_{City}(\sigma_{address=id}(person \times address))$$

Name	Address	MWhy
Peter	1	$\{\{x_1\}\}$
Alice	3	$\{\{x_2\}\}$
Bob	3	$\{\{x_3\}\}$

Id	City	Office-contact	MWhy
1	Chicago	(312) 123 4343	$\{\{x_4\}\}$
2	Chicago	(312) 555 7777	$\{\{x_5\}\}$
3	New York	(465) 123 1234	$\{\{x_6\}\}$

35 CS520 - 8) Provenance

35

8. Query Semantics - PP

ILLINOIS INSTITUTE OF TECHNOLOGY

36 CS520 - 8) Provenance

36

8. Provenance Polynomials - Computability

ILLINOIS INSTITUTE OF TECHNOLOGY

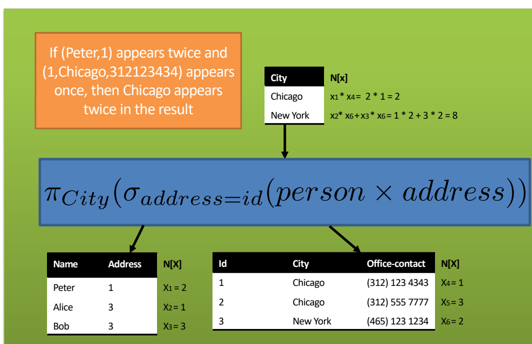
- Recall our requirements of sufficiency and minimality
- Provenance polynomials fulfill a stronger requirement: **computability**
  - Given the result of a query in N[X], we can compute the query result in any other semiring K under a given assignment of input tuples (variables of the polynomials) to annotations from K

37 CS520 - 8) Provenance

37

8. Query Semantics - PP

ILLINOIS INSTITUTE OF TECHNOLOGY



38 CS520 - 8) Provenance

38

8. Homomorphisms

ILLINOIS INSTITUTE OF TECHNOLOGY

- A function h from semiring K1 to K2 is a homomorphism if

$$h(k_1 \oplus_{\mathcal{K}_1} k_2) = h(k_1) \oplus_{\mathcal{K}_2} h(k_2)$$
$$h(k_1 \otimes_{\mathcal{K}_1} k_2) = h(k_1) \otimes_{\mathcal{K}_2} h(k_2)$$
$$h(0_{\mathcal{K}_1}) = 0_{\mathcal{K}_2}$$
$$h(1_{\mathcal{K}_1}) = 1_{\mathcal{K}_2}$$

- **Theorem**: Homomorphism commute with queries $Q(h(D)) = h(Q(D))$
- **Proof Sketch**: queries are defined using semiring operations which commute with homomorphisms

39 CS520 - 8) Provenance

39

8. Fundamental theorem

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Theorem**: Homomorphism commute with queries

$$Q(h(D)) = h(Q(D))$$

- **Proof Sketch**: queries are defined using semiring operations which commute with homomorphisms
- **Theorem**: Any assignment X -> K induces a semiring homomorphism N[X] -> K

40 CS520 - 8) Provenance

40

8. Summary

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Provenance is information about the origin and creation process of data**
  - Data dependencies
  - Dependencies between data and the transformations that generated it
- **Provenance for Queries**
  - **Correctness criteria**:
    - sufficiency, minimality, computability
  - **Provenance models:**
    - Why, MWhy, Provenance polynomials

41 CS520 - 8) Provenance

41