CompSci 590.6

Understanding Data:
Theory and Applications

Lecture 10

Why-Not (Data-based)
+
Deletion Propagation

Instructor: Sudeepa Roy
Email: *sudeepa@cs.duke.edu*

Today's Paper(s)

Huang-Chen-Doan-Naughton

On the Provenance of Non-Answers to Queries over Extracted Data

PVLDB 2008

Buneman-Khanna-Tan

On Propagation of Deletions and Annotations through Views

PODS 2002

Part-I

Why-Not (Data-based)

Huang-Chen-Doan-Naughton, 2008

Why-not approaches

- Query based
  - Lecture-8
  - Chapman-Jagadish'09: find out frontier query operator
  - Tran-Chan'10: find out changes to query operator that returns missing answer
- Data based
  - Huang et al'08 (this paper)
  - find out changes in data that can return the missing answer
  - also see Herschel-Hernandez'10 (Artemis)

Huang et al.'08

- Provenance of non-answers
  - Some conference system returns that X was not on PC of conference Y
  - But actually X was on the PC
- Why does not it appear in the answer?
  - bugs in extractors?
  - inaccuracies in sources?
  - incomplete coverage of sources?

Why care about non-answers?

- Help developer debug the system

or

- Help developer understand why they got the result they did

Provenance of non-answers

- could this non-answer become an answer?
- if so, how?
- by tuple insertion or updates
- but there could be infinitely many tuples
- allow proxy tuples

Concepts

- Trusted table
  - correct and complete
  - no need to consider updates or insertions
- Trusted attribute
  - its values in the existing tuples are correct
  - updates can be ignored
- tuples are generated by running extractors over documents
  - for each tuple, store the document name along with it
- Data source table S_i
  - for each data table R_i
  - fk_i and pk_i relationship

Example

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

Answer
= (Stanford, 1)

```
SELECT o.SCHOOL, r.RANK FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL AND o.STATE = 'ca' AND
      o.OPENING = 'yes' AND r.RANK <= 4;
```

- Opening
  - school, state are trusted
  - opening is not, collected from the web (extracted)
- Ranking
  - both collected from the web (extracted)

Example

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

Why is (Berkeley, 3) not in the answer?

```
SELECT o.SCHOOL, r.RANK FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL AND o.STATE = 'ca' AND
      o.OPENING = 'yes' AND r.RANK <= 4;
```

- not in top-4?
- does not have job opening?
- not in CA?

Example

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

Why is (Berkeley, 3) not in the answer?

```
SELECT o.SCHOOL, r.RANK FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL AND o.STATE = 'ca' AND
      o.OPENING = 'yes' AND r.RANK <= 4;
```

- not in top-4?
  - no, rank =3
- does not have job opening?
  - if (Berkeley, ca, yes) is inserted, it will become an answer

Provenance: Answer

- Query Q
- Mentions relations R_1, .., R_n
- Database D
- t an answer to Q
- Provenance of t
  - $t_i \in R_i(D)$, i = 1..n – base tuples that yield a derivation of t
  - and corresponding s_i if the source table S_i of R_i exists

Provenance: Potential Answer

- How non-answers can be potential answers
- Updates
  - Type 1: insertion of a tuple
  - Type 2: modification of an attribute value
  - deletions don't help for SPJ queries
- if no type-1, type-2 updates
  - then the non-answer t is "never-answer"
- if there is such a sequence
  - then t is a potential answer

Provenance: Potential Answer

- D': a database by type-1/type-2 update from D
- **null**$_i$ = (null,…,null): proxy tuple for Ri with all null values
- t is a non-answer
- t is a potential answer if
  - there exists a D' that satisfies the constraints
  - t belongs to Q[D']
- Provenance of t =
  - say t'$_i$ gives a potential derivation of t
  - t_i is the corresponding original tuple
  - t_i can be **null**$_i$ when t'$_I$ is inserted
  - provenance = t_i and t'$_i$ where i = 1..n

Provenance: Potential answer

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

```
SELECT o.SCHOOL, r.RANK FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL AND o.STATE = 'ca' AND
      o.OPENING = 'yes' AND r.RANK <= 4;
```

- (berkeley, 3) is a non-answer
- openings'(berkeley, ca, yes) along with ranking(berkeley, 3) gives a derivation
- hence a potential answer
- provenance = openings(null, null, null), openings'(berkeley, ca, yes), ranking(berkeley, 3)
  - without trsuted table, any combination can return missing tuple

Issues so far

- We are giving useful info
- But, if we do not have trust on allowable updates, then any combination of base tuples can be modified to yield a derivation
- e.g.
  - openings(mit, ma, no) -> openings'(berkeley, ca, yes)
  - ranking(mit,2) -> ranking'(berkeley, 3)
- Also many potential answers would exist making little sense

Example

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

- e.g. (cmu, 4) is a non-answer
- change the following
  - openings(cmu, pa, yes) -> openings'(cmu, ca, yes)
  - there is ranking(cmu, 4)

- (cmu, 4) becomes a potential answer
- but cmu is not in CA

Solution: Assume Trust

- If a table is trusted to be complete
  - no type-1 update allowed
  - otherwise, it is appendable
- If a table is trusted to be correct
  - no type-2 update allowed
- If an attribute is trusted to be correct
  - no type-2 update allowed
- Only updates to untrusted data allowed

Revisit examples

- Suppose openings(school, state, -) attributes are trusted
- (cmu, 4) is a non-answer
  - change the following
  - openings(cmu, pa, yes) -> openings'(cmu, ca, yes):
    NOT ALLOWED!
  - assuming the table to be complete, cannot insert (cmu, ca, yes)
  - there is ranking(cmu, 4)

- openings(mit, ma, no) -> openings'(berkeley, ca, yes)
  NOT ALLOWED!
  - ranking(mit,2) -> ranking'(berkeley, 3)

Never answer

- Can never be an answer given the constraints and trust
- e.g. (edgewood, 1)
  - if we trust the ranking table
  - irrespective of any update to the openings table

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

Algorithm: Overview

- The base tuples in provenance of potential answers
  - must appear in the db
  - or, must be null tuple
  - the trusted attributes must satisfy the selection predicates unless it is null
  - the values of two trusted values of two tuples must satisfy any join predicate

Algorithm through example

SCHOOL	STATE	OPENING
stanford	ca	yes
mit	ma	no
cmu	pa	yes

Table 1: openings

SCHOOL	RANK
stanford	1
mit	2
berkeley	3
cmu	4

Table 2: ranking

Assume that

- ranking is trusted
- openings(SCHOOL, STATE) are trusted attributes
- openings(OPENING) is not trusted
- SCHOOL should be unique in openings

```
SELECT o.SCHOOL, r.RANK
FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL
    AND o.STATE = 'ca'
    AND o.OPENING = 'yes'
    AND r.RANK <= 4
```

WHY-NOT QUESTION:

(berkeley, 3) is the missing answer

Computing provenance of (berkeley, 3)

```
SELECT o.SCHOOL, r.RANK
FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL
    AND o.STATE = 'ca'
    AND o.OPENING = 'yes'
    AND r.RANK <= 4
```

- Trusted
- Specifying non-answer
- Hypothetical update

```
SELECT o.SCHOOL, r.RANK
FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL
    AND o.STATE = 'ca'
    AND o.OPENING = 'yes'
    AND r.RANK <= 4
```

- Build predicates for the "provenance query" by retaining all predicates on trusted tables or trusted attributes
- Augment untrusted tables with null proxy tuples
- Evaluate the provenance query by applying the trusted predicates to tables mentioned in the user query

Computing provenance of (berkeley, 3)

```
SELECT o.SCHOOL, r.RANK
FROM openings o, ranking r
WHERE o.SCHOOL = r.SCHOOL
      AND o.STATE = 'ca'
      AND o.OPENING = 'yes'
      AND r.RANK <= 4
```

- Trusted
- Specifying non-answer
- Hypothetical update

Warning: This is a high-level overview, more care is needed
See the next slide and algorithm in the paper

```
SELECT o.SCHOOL, r.RANK, o.OPENING
FROM openings o RIGHT OUTER JOIN
        ranking r ON o.SCHOOL = r.SCHOOL
WHERE o.STATE = 'ca'
      AND o.OPENING = 'yes'
      AND r.RANK <= 4
      AND r.SCHOOL = 'berkeley'
```

o.SCHOOL	o.STATE	o.OPENING	r.SCHOOL	r.RANK
null->berkeley	null->CA	null->YES	berkeley	3

Provenance Query and Result for (berkeley, 3)

```
SELECT o.SCHOOL ||'->'||'berkeley',
   o.STATE ||'->'||'ca', o.OPENING ||'->'||'yes',
   os.URL, os.EXTRACTOR, /*from trusted tables*/
   r.SCHOOL, r.RANK, /* from trusted tables*/
   rs.URL, rs.EXTRACTOR /*from trusted tables*/
FROM openings o RIGHT OUTER JOIN ranking r
     ON o.SCHOOL = r.SCHOOL,
     openings_sources os, ranking_source rs
WHERE r.RANK <= 4 AND r.RANK = 3 AND
      os.SCHOOL = r.SCHOOL AND
      os.SCHOOL = 'berkeley' AND os.STATE = 'ca';
```

o.SCHOOL	o.STATE	o.OPENING	os.URL	os.EXTRACTOR	r.SCHOOL	r.RANK	rs.URL	...
null→berkeley	null→ca	null→yes	http://cs.berkeley...	job-extractor	berkeley	3	http://cra.org...	...

Table 4: Provenance of an example non-answer: (berkeley, 3)

Assumes source tables
os : for openings
rs: for ranking

Part-II

Deletion Propagation

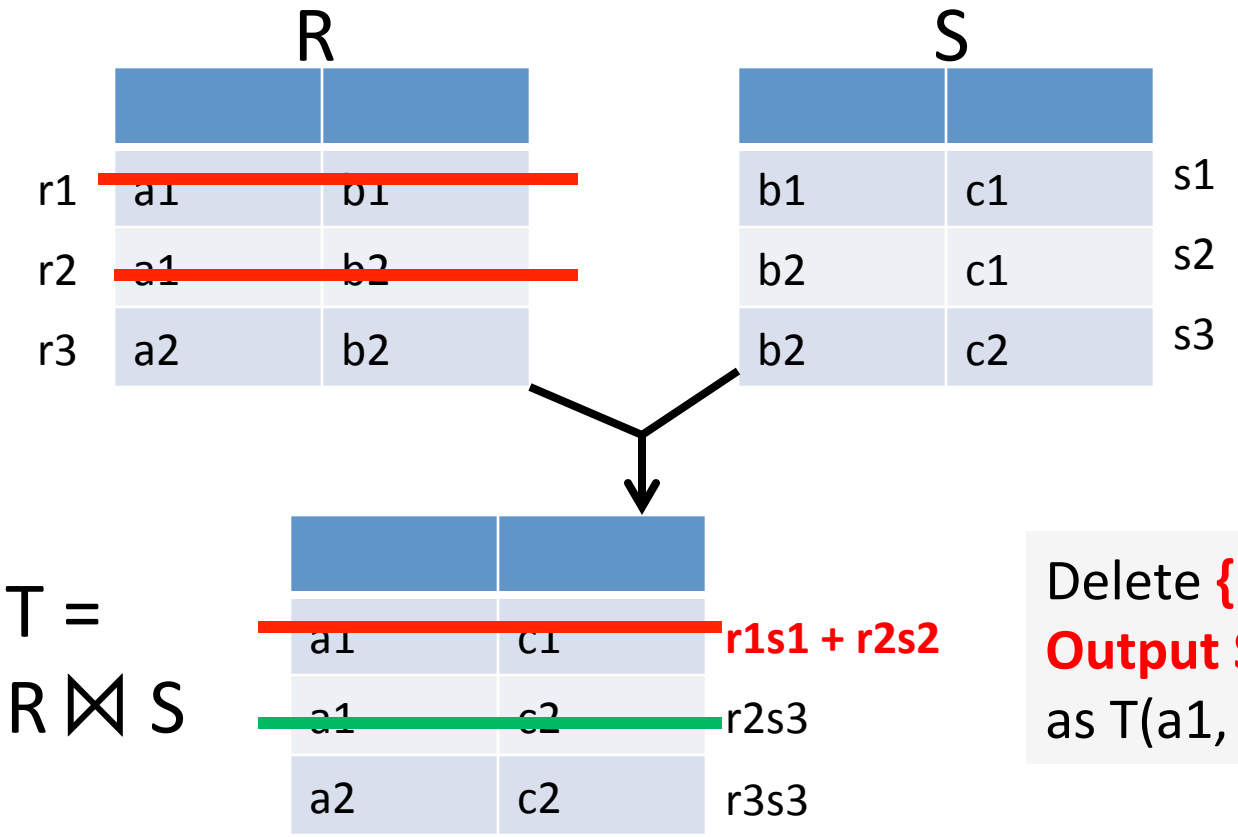
Buneman-Khanna-Tan, 2002

Deletion propagation problem

- An output tuple is to be deleted
- Delete a set of source tuples to achieve this
- Trivial answer: delete all source tuples
  - not enough
- Optimization problem Find a set of source tuples, having minimum side effect either in
  - output (view): delete as few other output tuples as possible
  - source: delete as few source tuples as possible
- Recall Boolean provenance annotations (Lecture 6)

View Side Effect

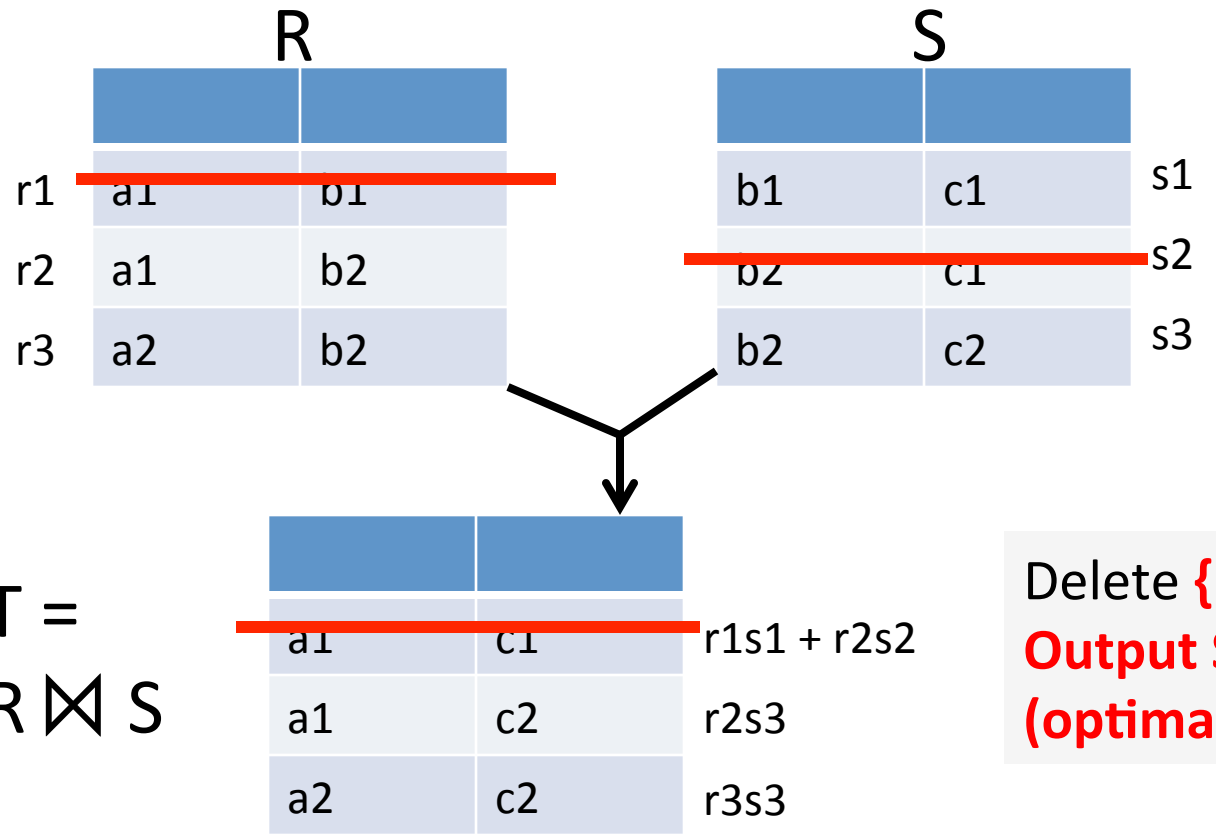
- To delete T(a1, c1)
- Need to delete one of 4 combinations: {r1, s1} x {r2, s2}



Delete **{r1, r2}**
Output Side Effect = 1
as T(a1, c2) is also deleted

View Side Effect

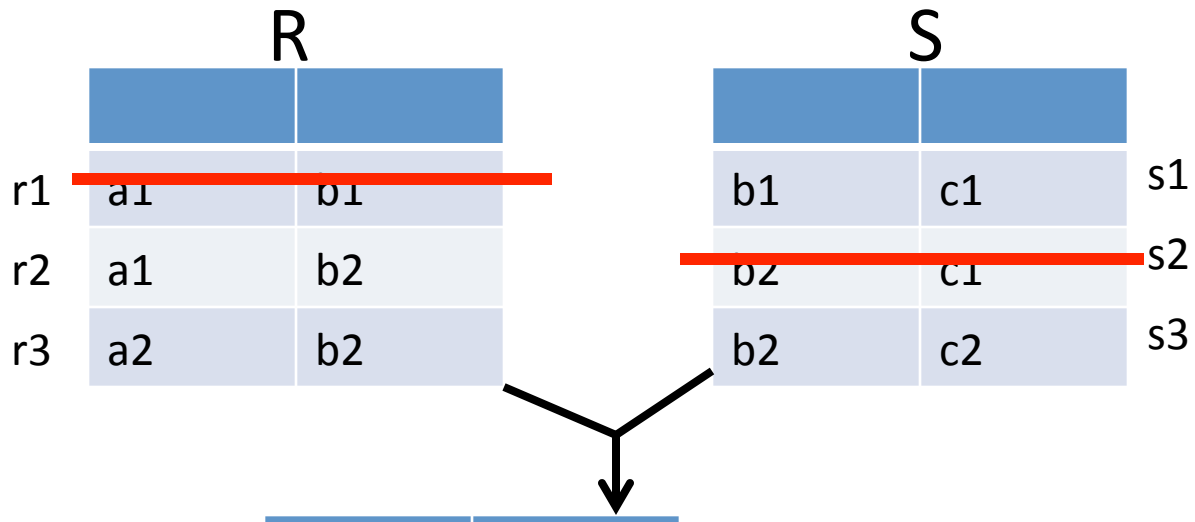
- To delete T(a1, c1)
- Need to delete one of 4 combinations: {r1, s1} x {r2, s2}



Delete **{r1, s2}**
Output Side Effect = 0 (optimal)

Source Side Effect

- To delete T(a1, c1)
- Need to delete one of 4 combinations: {r1, s1} x {r2, s2}



T = R ⋈ S

a1	c1	r1s1 + r2s2
a1	c2	r2s3
a2	c2	r3s3

Source side effect = #source tuples to be deleted **= 2** (**optimal** for any of four combinations)

Summary of Complexity Results

- S: SELECT σ P: PROJECT π J: JOIN $\bowtie$ U: UNION $\cup$
- RED: proof in class

Query class	Deciding whether there is a side-effect free deletion	Finding the minimum source deletion
PJ	NP-Hard	NP-Hard
JU	NP-Hard	NP-Hard
SPU	P	P
SJ	P	P

Poly-time algorithm: SPU

- Boolean provenance of the form: $r_1 + r_2 + \ldots + r_p$
- View-side effect:
  - unique solution
  - need to remove all of $r_1, r_2, \ldots r_p$
  - first pass: select tuples that satisfy "selection condn"
  - second pass: select the ones that projects to the specified output tuples t
  - extends to union
- Source-side effect:
  - the same algorithm

Poly-time algorithm: SJ

- Boolean provenance of the form: $r_1 . r_2 . \ldots . r_K$
  - k = #relations in the join query
- View-side effect:
  - for all i = 1.. k, check if r_i contributes to another output tuple
  - if yes, there will be a view side-effect
  - choose i with minimum side effect
- Source-side effect:
  - choose any of $r_1, r_2, \ldots r_p$
  - optimal source side effect = 1

NP-hardness

- On whiteboard
  - PJ for view side effect
  - Reduction from monotone 3-SAT
  - every clause has either all positive or all negative literals
    - (x1 + x2 + x3) or (¬x1 + ¬x2 + ¬x3)

- NP-hardness proofs for source-side effects:
  - Reduction from the hitting set problem

- Note:
  - different query classes have different complexity depending on the problem being considered