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 $R_i$
  – $f_k_i$ and $p_k_i$ relationship
Example

Table 1: openings

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>STATE</th>
<th>OPENING</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>ca</td>
<td>yes</td>
</tr>
<tr>
<td>mit</td>
<td>ma</td>
<td>no</td>
</tr>
<tr>
<td>cmu</td>
<td>pa</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 2: ranking

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>1</td>
</tr>
<tr>
<td>mit</td>
<td>2</td>
</tr>
<tr>
<td>berkeley</td>
<td>3</td>
</tr>
<tr>
<td>cmu</td>
<td>4</td>
</tr>
</tbody>
</table>

Answer = (Stanford, 1)

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

- **Ranking**
  - both collected from the web (extracted)
Example

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>STATE</th>
<th>OPENING</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>ca</td>
<td>yes</td>
</tr>
<tr>
<td>mit</td>
<td>ma</td>
<td>no</td>
</tr>
<tr>
<td>cmu</td>
<td>pa</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 1: openings

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>1</td>
</tr>
<tr>
<td>mit</td>
<td>2</td>
</tr>
<tr>
<td>berkeley</td>
<td>3</td>
</tr>
<tr>
<td>cmu</td>
<td>4</td>
</tr>
</tbody>
</table>

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;

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

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

Table 1: openings

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>STATE</th>
<th>OPENING</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>ca</td>
<td>yes</td>
</tr>
<tr>
<td>mit</td>
<td>ma</td>
<td>no</td>
</tr>
<tr>
<td>cmu</td>
<td>pa</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 2: ranking

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>1</td>
</tr>
<tr>
<td>mit</td>
<td>2</td>
</tr>
<tr>
<td>berkeley</td>
<td>3</td>
</tr>
<tr>
<td>cmu</td>
<td>4</td>
</tr>
</tbody>
</table>

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

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

• Query Q
• Mentions relations $R_1, \ldots, R_n$
• 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
- $null_i = (null,\ldots,null)$: proxy tuple for $R_i$ 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

- (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 trusted 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
• 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

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>STATE</th>
<th>OPENING</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>ca</td>
<td>yes</td>
</tr>
<tr>
<td>mit</td>
<td>ma</td>
<td>no</td>
</tr>
<tr>
<td>cmu</td>
<td>pa</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford</td>
<td>1</td>
</tr>
<tr>
<td>mit</td>
<td>2</td>
</tr>
<tr>
<td>berkeley</td>
<td>3</td>
</tr>
<tr>
<td>cmu</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: openings

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

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

WHY-NOT QUESTION: (berkeley, 3) is the missing 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
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

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

<table>
<thead>
<tr>
<th>o.SCHOOL</th>
<th>o.STATE</th>
<th>o.OPENING</th>
<th>r.SCHOOL</th>
<th>r.RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>null-&gt;berkeley</td>
<td>null-&gt;CA</td>
<td>null-&gt;YES</td>
<td>berkeley</td>
<td>3</td>
</tr>
</tbody>
</table>

• Trusted
• Specifying non-answer
• Hypothetical update
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*/
r.s.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’;
```

<table>
<thead>
<tr>
<th>o.SCHOOL</th>
<th>o.STATE</th>
<th>o.OPENING</th>
<th>os.URL</th>
<th>os.EXTRACTOR</th>
<th>r.SCHOOL</th>
<th>r.RANK</th>
<th>rs.URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>null→berkeley</td>
<td>null→ca</td>
<td>null→yes</td>
<td><a href="http://cs.berkeley">http://cs.berkeley</a>...</td>
<td>job-extractor</td>
<td>berkeley</td>
<td>3</td>
<td><a href="http://cra.org">http://cra.org</a>...</td>
</tr>
</tbody>
</table>

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 $T(a_1, c_1)$
- Need to delete one of 4 combinations: $\{r_1, s_1\} \times \{r_2, s_2\}$

Delete $\{r_1, r_2\}$
Output Side Effect = 1
as $T(a_1, c_2)$ is also deleted
**View Side Effect**

- To delete \( T(a_1, c_1) \)
- Need to delete one of 4 combinations: \( \{r_1, s_1\} \times \{r_2, s_2\} \)

\[
\begin{array}{ccc}
\text{R} & \text{S} \\
\hline
r_1 & a_1 & b_1 \\
r_2 & a_1 & b_2 \\
r_3 & a_2 & b_2 \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{S} & \\
\hline
s_1 & b_1 & c_1 \\
s_2 & b_2 & c_1 \\
s_3 & b_2 & c_2 \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{T} = & \\
\hline
r_1s_1 & r_2s_2 \\
\end{array}
\]

Delete \( \{r_1, s_2\} \)

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\}

\[
\begin{align*}
R & \quad S \\
r1 & a1 \quad b1 & b1 \quad c1 \\
r2 & a1 \quad b2 & b2 \quad c1 \\
r3 & a2 \quad b2 & b2 \quad c2
\end{align*}
\]

\[
T = R \Join S
\]

\[
\begin{align*}
& a1 \quad c1 \quad r1s1 + r2s2 \\
& a1 \quad c2 \quad r2s3 \\
& a2 \quad c2 \quad r3s3
\end{align*}
\]

Source side effect = \#source tuples to be deleted = 2 (optimal for any of four combinations)
Summary of Complexity Results

- **S**: SELECT $\sigma$  
- **P**: PROJECT $\pi$  
- **J**: JOIN $\bowtie$  
- **U**: UNION $\cup$

- **RED**: proof in class

<table>
<thead>
<tr>
<th>Query class</th>
<th>Deciding whether there is a side-effect free deletion</th>
<th>Finding the minimum source deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJ</td>
<td>NP-Hard</td>
<td>NP-Hard</td>
</tr>
<tr>
<td>JU</td>
<td>NP-Hard</td>
<td>NP-Hard</td>
</tr>
<tr>
<td>SPU</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>SJ</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>
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:** $r_1 \cdot r_2 \cdot \ldots \cdot r_K$
  – $k = \#\text{relations in the join query}$

• **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
    - \((x_1 + x_2 + x_3)\) or \((-x_1 + -x_2 + -x_3)\)

- **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