Reexamining some Holy Grails of Provenance

Boris Glavic       Renée J. Miller

University of Toronto

TaPP 2011, June 21, 2011
Insensitivity to Query Rewrite

- Equivalent queries have the same provenance
- \( Q \equiv Q' \Rightarrow \mathcal{P}(Q, l, t) = \mathcal{P}(Q', l, t) \)
Insensitivity to Query Rewrite

- Equivalent queries have the same provenance
- $Q \equiv Q' \Rightarrow \mathcal{P}(Q, I, t) = \mathcal{P}(Q', I, t)$
- **Caveat**: Which queries are equivalent?
  - Set vs. Bag semantics
  - Query language / Operators
## Stability

### Stability with Respect to Query Language Extension

- Extend query language with new operators \(\Rightarrow\) no change to provenance of queries that do not use new operators
Where [Buneman et al., 2003]
Captures which attribute values in the result of a query have been copied from which attribute values in the instance.

**Representation:** \( P(Attr(I)) \)

- **Where**: Operator-level syntax-based annotation propagation
- **IWhere**: Insensitive variant: Union of Where for all \( Q' \) with \( Q' \equiv Q \)
Where

Sensitive, traditionally attributed to being based on query syntax
Depends on the internal data-flow inside the query
  How values are routed through the query

IWhere

Insensitive by combining Where for all equivalent queries
Counterintuitive effect that if \((R, t, A)\) is in the provenance then all \((R, t', A)\) with \(t.A = t'.A\) are in the provenance too.
  Reason: Can construct equivalent query adding self-join on \(A\)
Where

Example

\[ Q_a = R \]
\[ Q_b = \pi_{A,B}(R \bowtie_{A=C} \pi_{A\rightarrow C,B\rightarrow D}(R)) \]

Provenance

\[ \text{Where}(Q_a, a_1, A) = \{(r_1, A)\} \]
\[ \text{Where}(Q_b, a_1, A) = \{(r_1, A), (r_2, A)\} \]
\[ \text{IWhere}(Q_a, a_1, A) = \text{IWhere}(Q_b, a_1, A) = \{(r_1, A), (r_2, A)\} \]
Arguments for Insensitivity

1. Traditionally observed as advantageous in database research
   ⇒ Tradition not a solid argument

2. External implementation of sensitive semantics. Computing provenance for a query different from the one that will be executed by DBMS
   ⇒ No way to solve this, but provenance based on user query seems to be reasonable

3. Implementation of sensitive semantics in DB-engine limits optimizer search space
   ⇒ Inensitive semantics may be harder to compute
   ⇒ Lack of practical experience
   ⇒ “Realistic” sensitivity example?
Instability of IWhere

- $IWhere$ is union of $Where$ for all equivalent queries
Instability of IWhere

- \( IWhere \) is union of \( Where \) for all equivalent queries
- e.g., SPJ and USPJ equivalences are different
Instability of \( I\text{Where} \)

- \( I\text{Where} \) is union of \( \text{Where} \) for all equivalent queries
- e.g., SPJ and USPJ equivalences are different
- e.g., union \( Q \) with a join of \( Q \) with some other relation
Instability of IWhere

- $IWhere$ is union of $Where$ for all equivalent queries
- e.g., SPJ and USPJ equivalences are different
- e.g., union $Q$ with a join of $Q$ with some other relation
- Let $UWhere$ be $IWhere$ for USPJ queries
Instability of IWhere

- \( IWhere \) is union of \( Where \) for all equivalent queries
- e.g., SPJ and USPJ equivalences are different
- e.g., union \( Q \) with a join of \( Q \) with some other relation
- Let \( UWhere \) be \( IWhere \) for USPJ queries
- \( \Rightarrow UWhere \) an attribute value is annotated with all annotations from attribute positions that have the same value

Holy Grails
Instability of IWhere

Example

\[ Q_a = R \]

\[
\begin{array}{c|c|c}
\hline
 & A & B \\
\hline
r_1 & 1 & 2 \\
\hline
r_2 & 1 & 3 \\
\hline
r_3 & 2 & 3 \\
\hline
r_4 & 2 & 5 \\
\hline
\end{array}
\]

\[ S \]

\[
\begin{array}{c|c|c}
\hline
 & C & \\
\hline
s_1 & 2 & \\
\hline
s_2 & 3 & \\
\hline
\end{array}
\]

\[ Q_a & Q_b \]

\[
\begin{array}{c|c|c}
\hline
 & A & B \\
\hline
a_1 & 1 & 2 \\
\hline
a_2 & 1 & 3 \\
\hline
a_3 & 2 & 3 \\
\hline
a_4 & 2 & 5 \\
\hline
\end{array}
\]

Provenance

\[
\text{Where}(Q_a, a_3, A) = \{(r_3, A)\}
\]

\[
\text{IWhere}(Q_a, a_3, A) = \{(r_3, A), (r_4, A)\}
\]

\[
\text{UWhere}(Q_a, a_3, A) = \{(r_3, A), (r_4, A), (r_1, B), (s_1, C)\}
\]
Conclusions

Take Away Messages

- Be careful how to achieve a property
- Insensitivity less applicable to semantics that address internal data-flow
  - Queries with the same external but possibly different internal behaviour have the same provenance

Some Things I’d Like to See

- “Declarative” Semantics \( \Rightarrow \) derive operator-level construction
- Semantics model processing, but have a insensitive “core”
- The never-ending quest: Deal with Negation
- Other data-models (order)
## Questions

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Sound</th>
<th>Complete</th>
<th>Responsible</th>
<th>Insensitive (set)</th>
<th>Insensitive (bag)</th>
<th>Stable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Why</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wit</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Why</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td>IWhy</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Where</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Where</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>?</td>
<td>X</td>
</tr>
<tr>
<td>IWhere</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td><strong>How</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Lineage-based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lineage</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>PI-CS</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>C-CS</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td><strong>Causality</strong></td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Reexamining some Holy Grails of Provenance
## Semantics Summary

<table>
<thead>
<tr>
<th>Representation</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{P}(\text{Attr}(I))$</td>
<td>Where, IWhere</td>
</tr>
<tr>
<td>$\mathbb{P}(\mathbb{P}(\text{Tuple}(I)))$</td>
<td>Wit, Why, IWhy</td>
</tr>
<tr>
<td>$\mathbb{N}[\text{Tuple}(I)]$</td>
<td>How</td>
</tr>
<tr>
<td>${&lt; R_1^<em>, \ldots, R_n^</em> &gt;</td>
<td>R_i^* \subseteq R_i(Q) }$</td>
</tr>
<tr>
<td>$\mathbb{P}({&lt; t_1, \ldots, t_n &gt;</td>
<td>t_i \in R_i(Q) \lor t_i = \bot })$</td>
</tr>
<tr>
<td>$\mathbb{P}(\text{Tuple}(I))$</td>
<td>Causality</td>
</tr>
</tbody>
</table>

Reexamining some Holy Grails of Provenance
Sound, Complete, Responsible

- **Sound**: Provenance of $t$ produces nothing different from $t$.
  - $t' \neq t \Rightarrow t' \not\in Q(P(Q, I, t))$
Sound, Complete, Responsible

- **Sound**: Provenance of $t$ produces nothing different from $t$.
  - $t' \neq t \Rightarrow t' \not\in Q(\mathcal{P}(Q, l, t))$
- **Caveat**: Semantics of evaluating query over provenance

Reexamining some Holy Grails of Provenance
Sound, Complete, Responsible

- **Sound**: Provenance of $t$ produces nothing different from $t$.
  - $t' \neq t \Rightarrow t' \notin Q(P(Q,I,t))$

- **Complete**: Provenance of $t$ produces at least $t$
  - $t \in Q(P(Q,I,t))$

Reexamining some Holy Grails of Provenance
### Sound, Complete, Responsible

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sound:</strong> Provenance of $t$ produces nothing different from $t$.</td>
</tr>
<tr>
<td>$t' \neq t \Rightarrow t' \notin Q(P(Q, l, t))$</td>
</tr>
<tr>
<td><strong>Complete:</strong> Provenance of $t$ produces at least $t$</td>
</tr>
<tr>
<td>$t \in Q(P(Q, l, t))$</td>
</tr>
<tr>
<td><strong>Caveat:</strong> Semantics of evaluating query over provenance</td>
</tr>
</tbody>
</table>

Reexamining some Holy Grails of Provenance
Sound, Complete, Responsible

- **Sound:** Provenance of $t$ produces nothing different from $t$.
  - $t' \neq t \Rightarrow t' \notin Q(\mathcal{P}(Q, I, t))$
- **Complete:** Provenance of $t$ produces at least $t$
  - $t \in Q(\mathcal{P}(Q, I, t))$
- **Responsible:** Every tuple in the provenance of $t$ is necessary to derive $t$

Reexamining some Holy Grails of Provenance
Sound, Complete, Responsible

- **Sound**: Provenance of $t$ produces nothing different from $t$.
  - $t' \neq t \Rightarrow t' \notin Q(P(Q, I, t))$

- **Complete**: Provenance of $t$ produces at least $t$
  - $t \in Q(P(Q, I, t))$

- **Responsible**: Every tuple in the provenance of $t$ is necessary to derive $t$

**Caveat**: ... from every alternative derivation in the provenance ...
  - $\Rightarrow$ factor provenance into alternative derivations

**Caveat**: Different ways to model that.
  - E.g., $\forall t' \in P(Q, I, t): t \notin Q(P(Q, I, t) - \{t'\})$
Sound, Complete, Responsible

Example

\[ Q_b = \pi_{A,B}(R \otimes_{A=C} \pi_{A \rightarrow C,B \rightarrow D}(R)) \]

Provenance

\[ \mathcal{P}(Q_b, l, a_1) = \{r_1, r_2\} \]

Reexamining some Holy Grails of Provenance
Lineage-based [Cui et al., 2000]

Operator-level declarative semantics similar to Why. Provenance is modeled as a list of subsets of the relations accessed by the query (leafs of the algebra tree of $Q$)

**Representation:** $\{< R_1^*, \ldots, R_n^* > | R_i^* \subseteq R_i(Q) \}$

- **Lineage:** List of subsets of the algebra-tree nodes
Lineage-based [Glavic et al., 2009]

Provenance is modeled as a set of **witness lists**. A witness list is a list of tuples - one from each relation accessed by the query.

**Representation:** \( \mathbb{P}(\{< t_1, \ldots, t_n > | t_i \in R_i(Q) \lor t_i = \bot\}) \)

- **PI-CS:** *Lineage* with different representation and broader query language coverage
- **C-CS:** Similar to *Where* but with tuple granularity
### Lineage-based

#### Example

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>r₂</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>r₃</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>r₄</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>2</td>
</tr>
<tr>
<td>s₂</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qc</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>1</td>
</tr>
<tr>
<td>c₂</td>
<td>2</td>
</tr>
<tr>
<td>c₃</td>
<td>3</td>
</tr>
<tr>
<td>c₄</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qd &amp; Qe</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₁</td>
<td>1</td>
</tr>
<tr>
<td>d₂</td>
<td>2</td>
</tr>
</tbody>
</table>

#### Provenance

- \( Pl - CS(Q_c, c_2) = \{ <\bot, r_1>, <r_3, \bot>, <r_4, \bot> \} \)
- \( Pl - CS(Q_d, d_1) = \{ <r_1, s_1>, <r_2, s_2> \} \)
- \( C - CS(Q_d, d_1) = \{ <r_1, \bot>, <r_2, \bot> \} \)
- \( Lineage(Q_d, d_1) = \langle \{r_1, r_2\}, \{s_1, s_2\} \rangle \)

---

Reexamining some Holy Grails of Provenance
Why [Buneman et al., 2003]

Why-provenance models provenance as a set of witnesses. A witness $w$ for a tuple $t$ is a subset of the instance $I$ where $t \in Q(w)$.

**Representation:** $\mathbb{P}(\mathbb{P}(\text{Tuple}(I)))$

- **Wit:** Set of all witnesses
- **Why:** Query-syntax based “proof-witnesses”
- **IWhy:** Minimal elements from Wit resp. Why
Example

\[ Q_a = R \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>r_2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>r_3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>r_4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>a_2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>a_3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a_4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Provenance

\[ Wit(Q_a, a_1) = \{ J \mid J \subseteq R \land r_1 \in J \} \]
\[ Why(Q_a, a_1) = \{ \{ r_1 \} \} \]
\[ lWhy(Q_a, a_1) = \{ \{ r_1 \} \} \]

Reexamining some Holy Grails of Provenance
Provenance Semirings [Green et al., 2007]

Tuples of relations annotated with elements from a semiring. Annotation propagation defined for positive relational algebra as operations of the semiring (set difference and aggregation later).

Representation: $\mathbb{N}(Tuple(I))$

- **How**: Most general form of annotations: polynomials over variable representing the instance tuples. Addition indicates alternative use of tuples; multiplication conjunctive use.
How

Example

\[ Q_b = \pi_{A,B}(R \Join_{A=C} \pi_{A\rightarrow C,B\rightarrow D}(R)) \]

Provenance

\[ \text{How}(Q_b, a_1) = r_1^2 + r_1 \times r_2 \]
Causality [Meliou et al., 2010]

Provenance is modeled as a set of causes. A cause $c \in I$ for a tuple $t$ is defined as follows:

1. $t \not\in Q(I - \{c\})$
2. there exists a set $C \subset I$ called contingency so that $t \in Q(I - C)$ and $t \not\in Q(I - C - \{c\})$

Representation: $\mathbb{P}(\text{Tuple}(I))$

- **Causality**: Set of all causes
Causality

\[ Q_a = R \]

**Provenance**

\[ \text{Causality}(Q_a, a_1) = \{ r_1 \} \]

### Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>r_2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>r_3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>r_4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>a_2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>a_3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>a_4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Insensitivity to Query Rewrite

- Equivalent queries have the same provenance
- \( Q \equiv Q' \Rightarrow \mathcal{P}(Q, I, t) = \mathcal{P}(Q', I, t) \)
- Set resp. Bag semantics

Overview

- **Insensitive (Set, Bag)** Wit, IWhy, IWhere, Causality
- **Insensitive (Bag)** How
- **Sensitive**: Lineage, PI-CS, C-CS, Where

Reexamining some Holy Grails of Provenance
Why

Wit

- Defined over black-box behaviour of query $\Rightarrow$ trivially insensitive

Why

- Sensitive, traditionally attributed to being based on query syntax
- $Why$ may contain tuples that do not contribute to $t$
- $\Rightarrow$ Equivalent queries that apply redundant computations may contain larger provenance
- Caveat: But why does this argument not apply to $Wit$?
- Positive queries: super-set of a witness is also a witness $\Rightarrow$ tuples used by redundant computations are in $Wit$
Why

Example

\[ Q_a = R \]
\[ Q_b = \pi_{A,B}(R \bowtie_{A=C} \pi_{A\rightarrow C,B\rightarrow D}(R)) \]

<table>
<thead>
<tr>
<th>R</th>
<th>\text{Q}_a &amp; \text{Q}_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>r_1</td>
<td>1</td>
</tr>
<tr>
<td>r_2</td>
<td>1</td>
</tr>
<tr>
<td>r_3</td>
<td>2</td>
</tr>
<tr>
<td>r_4</td>
<td>2</td>
</tr>
<tr>
<td>a_1</td>
<td>1</td>
</tr>
<tr>
<td>a_2</td>
<td>1</td>
</tr>
<tr>
<td>a_3</td>
<td>2</td>
</tr>
<tr>
<td>a_4</td>
<td>2</td>
</tr>
</tbody>
</table>

Provenance

\[ \text{Wit}(Q_a, a_1) = \text{Wit}(Q_b, a_1) = \{ J \mid J \subset R \land r_1 \in J \} \]
\[ \text{Why}(Q_a, a_1) = \{ \{ r_1 \} \} \]
\[ \text{Why}(Q_b, a_1) = \{ \{ r_1 \}, \{ r_1, r_2 \} \} \]
\[ \text{IWhy}(Q_a, a_1) = \text{IWhy}(Q_b, a_1) = \{ \{ r_1 \} \} \]
Lineage-based

- Provenance representation based on query syntax
- Trivial examples for sensitivity based on reordering of the arguments of commutative operators
Lineage-based

Example

\[ Q_d = \pi_A(R \bowtie_B C S) \]
\[ Q_e = \pi_A(S \bowtie_B C R) \]

Provenance

\[ \text{Lineage}(Q_d, d_1) = \langle \{r_1, r_2\}, \{s_1, s_2\} > \]
\[ \text{Lineage}(Q_e, d_1) = \langle \{s_1, s_2\}, \{r_1, r_2\} > \]
\[ PI - CS(Q_d, d_1) = \{< r_1, s_1 >, < r_2, s_2 >\} \]
\[ PI - CS(Q_e, d_1) = \{< s_1, r_1 >, < s_2, r_2 >\} \]

Reexamining some Holy Grails of Provenance
How

- Sensitive (Set):
- Insensitive (Bag): Operator semantics defined to take bag semantics into account
How

Example

\( Q_a = R \)

\( Q_b = \pi_{A,B}(R \bowtie_{A=C} \pi_{A\rightarrow C,B\rightarrow D}(R)) \)

Provenance

\( \text{How}(Q_a, a_1) = r_1 \)

\( \text{How}(Q_b, a_1) = r_1^2 + r_1 \times r_2 \)
Causality

- Trivially insensitive: Defined over the black-box behaviour of a query
Causality

Reexamining some Holy Grails of Provenance

Example

\[ Q_a = R \]
\[ Q_b = \pi_{A,B}(R \bowtie_{A=C} \pi_{A\rightarrow C,B\rightarrow D}(R)) \]

Provenance

\[ \text{Causality}(Q_a, a_1) = \text{Causality}(Q_b, a_1) = \{ r_1 \} \]