Reformulating Aggregate Queries using Views

Abhijeet Mohapatra, Michael R. Genesereth
What is the average rating of ‘Man of Steel’?

**Common Schema**

<table>
<thead>
<tr>
<th>Movie</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man of Steel</td>
<td>8.1</td>
</tr>
<tr>
<td>Twilight</td>
<td>5.4</td>
</tr>
</tbody>
</table>

(ratings out of 10.0)
**LAV Integration**

**Global as View (GAV)**

- Master Schema mapped to Sources
- m(X, Y, Z)
- s1(X, Y) :- s1(X, Y)
- s2(X, Y) :- s2(X, Y)
- Hard to add sources
- Easy to query

**Local as View (LAV)**

- Sources mapped to Master Schema
- m(X, Y, Z)
- s1(X, Y) :- m(X, Y, s1)
- s2(X, Z) :- m(X, Y, s2)
- Easy to add sources
- Hard to query
Query Answering in LAV

Answering a Query using Views

Query

m(X, Y, Z)

Sources are views over Master Schema predicates

s1(X, Y, s1) :- m(X, Y, Z)

s1(X, Y)

Master Schema

Source
Query Answering in LAV

Answering a Query using Views

Inverse Method
(Duschka and Genesereth ’97)
Inverse Method [Duschka and Genesereth ’97]

Sources are mapped to Master Schema

\[ s_l(X, Y) \implies m(X, Y, s_l) \]

Inverse Rules

\[ m(X, Y, s_l) \implies s_l(X, Y) \]

We can reformulate \( m(X, Y, Z) \) using the source \( s_l \)
Inverse Method [Duschka and Genesereth ’97]

Equivalent reformulations cannot always be generated!!

\[ s_1(X, Y) :- m(X, Y, Z) \]

**Inverse Rules**

\[ m(X, Y, f(X, Y)) :- s_1(X, Y) \]

Generates equivalent reformulations when they exist
Leverage Inverse Method to reformulate non-aggregate queries using views

What if the query and / or views contain aggregates?

Finding equivalent reformulations is undecidable!
Prior Work

Small fraction of prior work on query reformulation addresses aggregate queries (references in paper)

Reformulations are considered in restricted settings:

- **Same aggregates** in the query and the views
- **Built-in aggregates** only (min, max, sum and count)
- **Central Rewritings** [Afrati ’01]
Representing Datalog using sets and tuples

Extend Datalog using the `setof` operator

\[ t(X, W) :- \text{setof}(Y, m(X, Y), W) \]

For every \( X \), the set \( W = \{ Y \mid m(X, Y) \} \)

\[
\begin{array}{c|c}
  \text{m(X,Y)} & \text{t(X,W)} \\
  \hline
  a & \{1, 2\} \\
  a & \{1\} \\
  b & \{1\}
\end{array}
\]
Representations

Aggregates are predicates over sets

- `sum({ }, 0)`
- `sum({X | Y}, S) :- sum(Y, T), S = T + X`

- `avg(W, A) :- sum(W, S), count(W, C), A = S / C`

Modular definition

- `t(X, W) :- setof(Y, m(X, Y), W)`
- `s1(X, S) :- t(X, W), sum(W, S)`

<table>
<thead>
<tr>
<th>m(X, Y)</th>
<th>t(X, W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a 1</td>
<td>a {1, 2}</td>
</tr>
<tr>
<td>a 2</td>
<td>b {1}</td>
</tr>
<tr>
<td>b 1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t(X, W)</th>
<th>s1(X, S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a {1, 2}</td>
<td></td>
</tr>
<tr>
<td>b {1}</td>
<td></td>
</tr>
<tr>
<td>a 3</td>
<td></td>
</tr>
<tr>
<td>b 1</td>
<td></td>
</tr>
</tbody>
</table>
Inverting aggregate views

\[ s I (X, S) \quad : \quad \text{setof}(Y, m(X,Y), W), \quad \text{sum}(W, S) \]

Rewrite using an auxiliary view \( t(X, W) \)

aggregate view \( \quad : \quad \text{auxiliary view}, \quad \text{aggregate predicate} \)

\[ s I (X, S) \quad : \quad t(X, W), \quad \text{sum}(W, S) \]

\[ t(X, W) \quad : \quad \text{setof}(Y, m(X,Y), W) \]

Inverting aggregate views \( \Rightarrow \) Inverting views with sets
Inverting views with sets

\[ t(X, W) \quad \text{:-} \quad \text{setof}(Y, m(X, Y), W) \]

Inverse Rule \((t^{-1})\)

\[ m(X, Y) \quad \text{:-} \quad t(X, W), Y \in W \]
Inverting aggregate views

\[ s_1(X, S) \ :- \ \text{setof}(Y, m(X, Y), W), \ \text{sum}(W, S) \]

Rewrite using auxiliary views

\[ s_1(X, S) \ :- \ t(X, W), \ \text{sum}(W, S) \]

\[ t(X, W) \ :- \ \text{setof}(Y, m(X, Y), W) \]

Invert modified view definition

\[ m(X, Y) \ :- \ t(X, W), Y \in W \]

\[ t(X, f(X, S)) \ :- \ s_1(X, S) \]

\[ \text{sum}(f(X, S), S) \ :- \ s_1(X, S) \]
Algorithm $\text{Invert}_{\text{agg}}$

**Input**: Query $q$ and set of views $V$

**Expand** the non-recursive aggregates in $q$ as $q'$

**Rewrite** the views $\in V$ using auxiliary views

**Invert** the modified view definitions ($V^{-1}$)

**Generate** functional dependencies $\Gamma$

**Output**: Query plan $\{q'\} \cup V^{-1} \cup \Gamma$
Example

Query

\[ q(M, A) \quad :\quad \text{setof}(R, \text{movie}(M, R), W), \quad \text{avg}(W, A) \]

Views

\[ \text{score}(M, S) \quad :\quad \text{setof}(R, \text{movie}(M, R), W), \quad \text{sum}(W, S) \]
\[ \text{views}(M, C) \quad :\quad \text{setof}(R, \text{movie}(M, R), W), \quad \text{count}(W, C) \]
Modified Query

\[ q'(M,A) \] :- setof(Y, movie(M, R), W), sum(W, S) count(W, C), A = S / C

Inverse Rules \((V^{-1})\)

movie(M, R) :- t(M, W), R \in W
t(M, f(M, S)) :- score(M, S)
t(M, g(M, C)) :- views(M, C)
sum(f(M, S), S) :- score(M, S)
count(g(M, C), C) :- views(M, C)

\[ \Gamma: t(M, X) \& t(M, Y) \Rightarrow X = Y \]
What is the average rating of ‘Man of Steel’?

\[
q('\text{Man of Steel}', A) \\
 t('\text{Man of Steel}', W), \text{sum('Man of Steel', S)}, \text{count('Man of Steel', C)}, A = \frac{S}{C}
\]

<table>
<thead>
<tr>
<th>t(M, W)</th>
<th>q('Man of Steel', 8.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man of Steel</td>
<td>f('Man of Steel', 16600)</td>
</tr>
<tr>
<td>Man of Steel</td>
<td>g('Man of Steel', 2000)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sum(W, S)</th>
<th>count(W, C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f('Man of Steel', 16600)</td>
<td>16600</td>
</tr>
<tr>
<td>g('Man of Steel', 2000)</td>
<td>2000</td>
</tr>
</tbody>
</table>
Properties of $\text{Invert}_{\text{agg}}$

• If a query can be equivalently reformulated using a supplied set of views, $\text{Invert}_{\text{agg}}$ computes such a reformulation.

• Size of the query plan is linear in the size of the input.
Examples:

1. BOOM Project (UCB)
   pathCount(X, Y, count<C>) :- path(X, Z, C)
   stableRoute(X, Y) :- pathCount(X, Y, Z), router(Y), Z > 10

2. A deductive approach to AI planning [Brogi ’03]
   add\(i\)(Cond) :- fired\(i\)(\(\alpha\)), postcond(\(\alpha\), Cond, pos)
Thank you!