Reformulating Aggregate Queries using Views

Abhijeet Mohapatra, Michael R. Genesereth

What is the average rating of 'Man of Steel'?

Common Schema

IMDB

Movie	Rating
Man of Steel	8.1
Twilight	5.4

Movie	Views
Man of Steel	100
Twilight	10

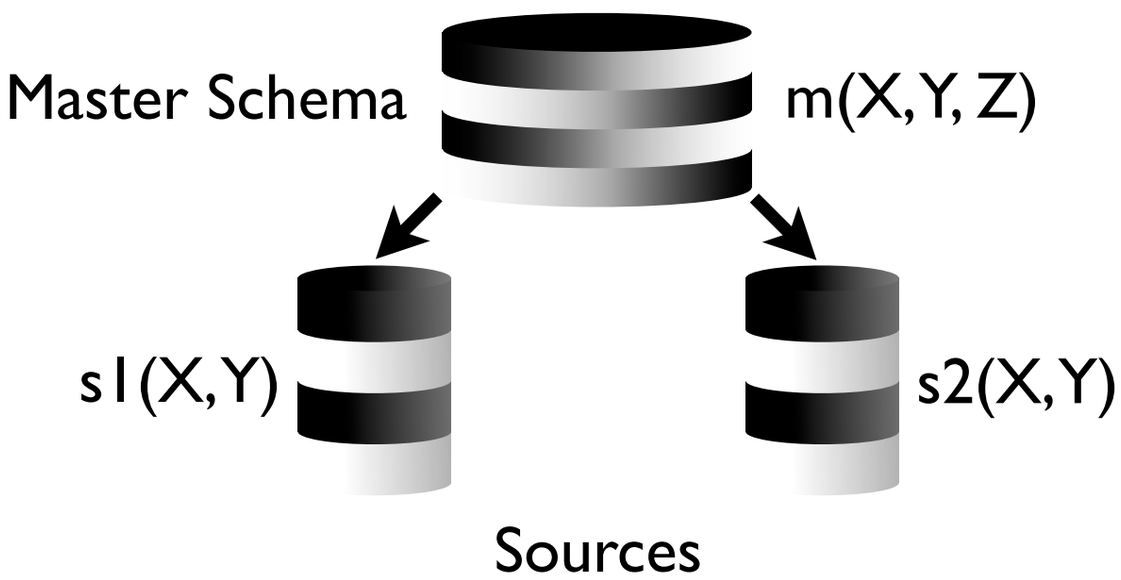
(ratings out of 10.0)

Netflix

Movie	Rating	Reviews
Man of Steel	4.3	200
Twilight	1.0	550

(ratings out of 5.0)

Global as View (GAV)



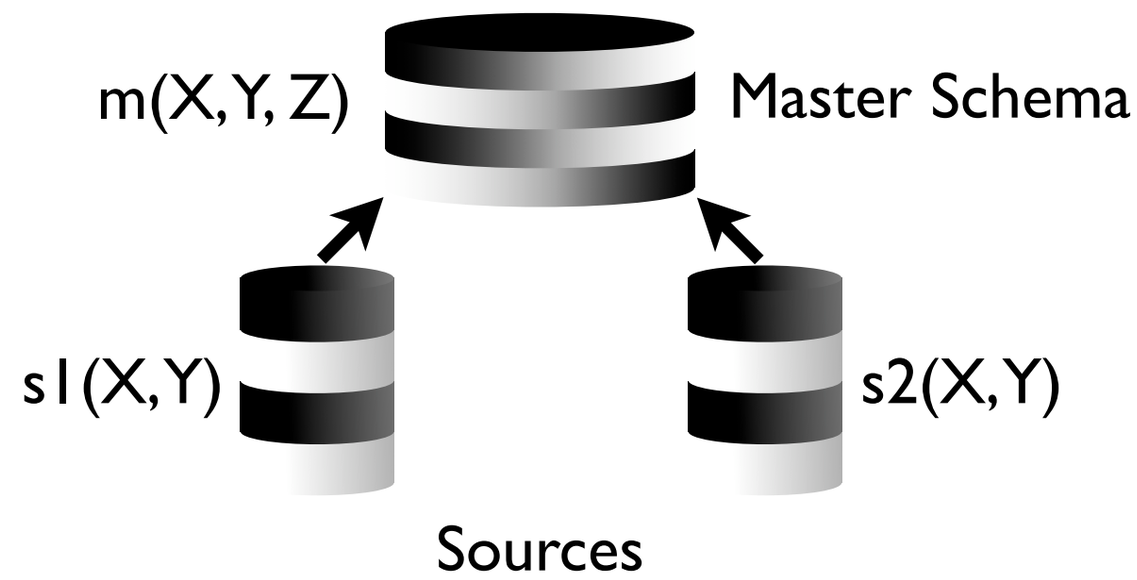
- Master Schema mapped to Sources

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

m(X, Y, s2) :- s2(X, Y)

- Hard to add sources
- Easy to query

Local as View (LAV)



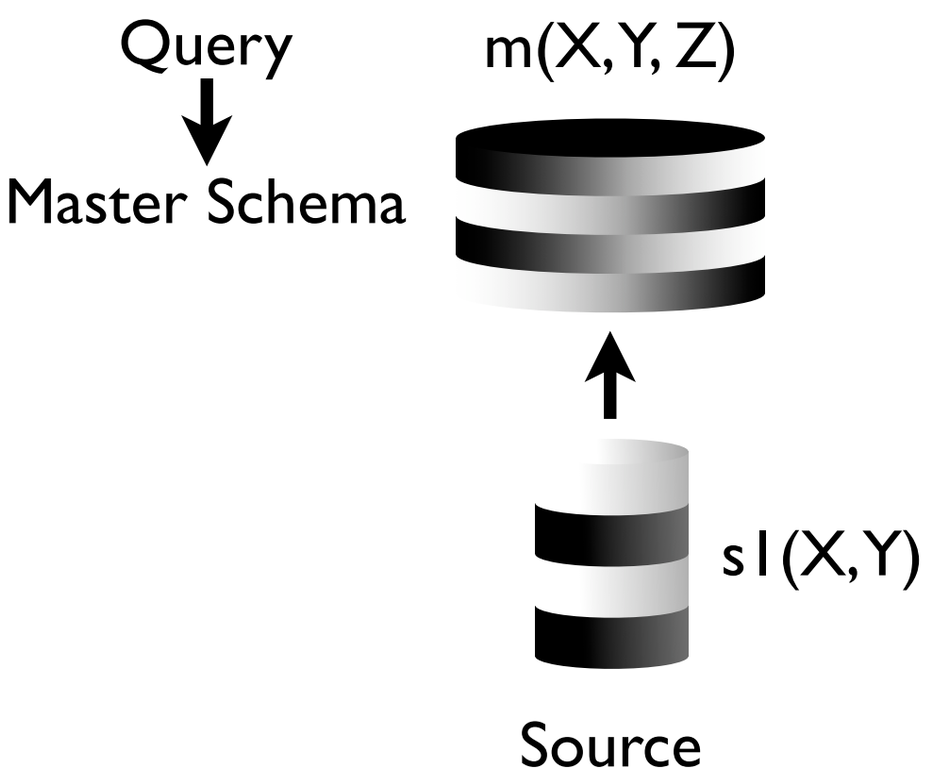
- Sources mapped to Master Schema

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

s2(X, Z) :- m(X, Y, s2)

- Easy to add sources
- Hard to query

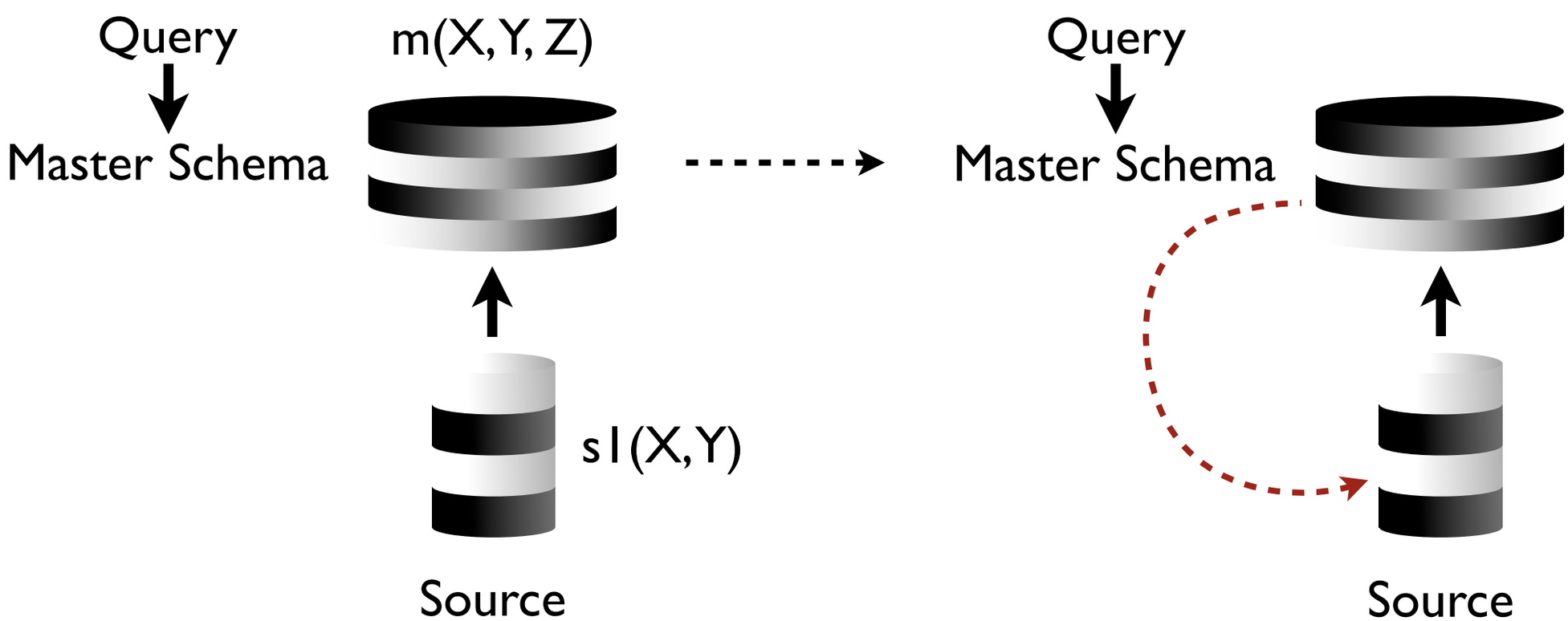
Query Answering in LAV ⟷ Answering a Query using Views



Sources are views over Master Schema predicates

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

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

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

s1(X,Y)	
a	b
b	c

Inverse Rules

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

m(X,Y,Z)		
a	b	s1
b	c	s1

We can reformulate m(X,Y,Z) using the source s1

Inverse Method [Duschka and Genesereth '97]

Equivalent reformulations cannot always be generated!!

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

s1(X,Y)	
a	b
b	c

Inverse Rules

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

m(X,Y,Z)		
a	b	f(a, b)
b	c	f(b, c)

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]

Representation

Extend Datalog using sets and tuples

Generate sets using setof operator

t(X, W) :- setof(Y, m(X, Y), W)

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

m(X, Y)	
a	1
a	2
b	1

→

t(X, W)	
a	{1, 2}
b	{1}

Representation

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)

m(X,Y)	
a	1
a	2
b	1

→

t(X,W)	
a	{1, 2}
b	{1}

s1(X, S) :- t(X, W), sum(W, S)

t(X,W)	
a	{1, 2}
b	{1}

→

s1(X, S)	
a	3
b	1

Inverting aggregate views

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

Rewrite using an auxiliary view t(X,W)

aggregate view :- auxiliary view, aggregate predicate

s1(X, S) :- t(X,W), sum(W, S)

t(X,W) :- setof(Y, m(X,Y), W)

Inverting aggregate views ⇒ Inverting views with sets

Inverting views with sets

t(X, W) :- setof(Y, m(X, Y), W)

Inverse Rule (t^{-1})

m(X, Y) :- t(X, W), Y ∈ W

t(X, S)	
a	{1, 2}
b	{1}

→

m(X, Y)	
a	1
a	2
b	1

Inverting aggregate views

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

Rewrite using auxiliary views

s1(X, S) :- t(X, W), sum(W, S)

t(X, W) :- setof(Y, m(X, Y), W)

Invert modified view definition

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

t(X, f(X, S)) :- s1(X, S)

sum(f(X, S), S) :- s1(X, S)

Algorithm Invert_{agg}

Input : Query q and set of views V

Expand the non-recursive aggregates in q as q′

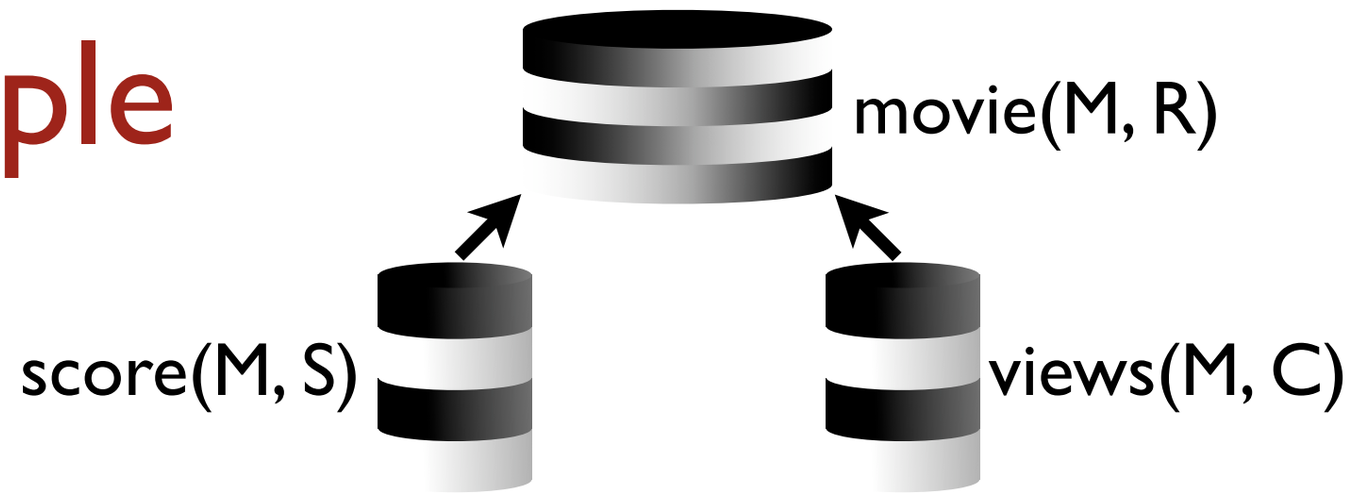
Rewrite the views $\in$ V using auxiliary views

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

Generate functional dependencies Γ

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

Example



Query

q(M, A) :- setof(R, movie(M, R), W), avg(W, A)

Views

score(M, S) :- setof(R, movie(M, R), W), sum(W, S)

views(M, C) :- setof(R, movie(M, R), W), 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 ∈ 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)

Γ: t(M, X) & t(M, Y) ⇒ X = Y

score(M, S)	
Man of Steel	16600
Twilight	2550

views(M, S)	
Man of Steel	2000
Twilight	100

What is the average rating of 'Man of Steel'?

q('Man of Steel', A)

t('Man of Steel', W), sum('Man of Steel', S),
count('Man of Steel', C), A = S / C

t(M, W)	
Man of Steel	f('Man of Steel', 16600)
Man of Steel	g('Man of Steel', 2000)

q('Man of Steel', 8.3)

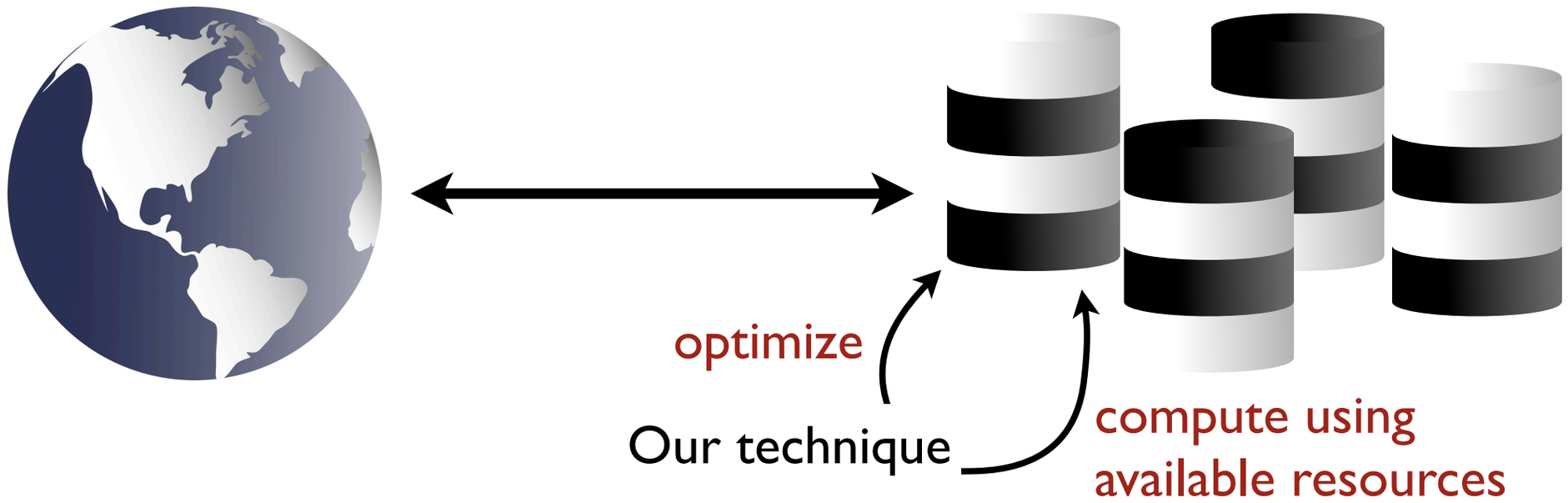
sum(W, S)	
f('Man of Steel', 16600)	16600

count(W, C)	
g('Man of Steel', 2000)	2000

Properties of Invert_{agg}

- If a query can be equivalently reformulated using a supplied set of views, Invert_{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^j(Cond) :- $fired^j$(α), postcond(α, Cond, pos)

Thank you!