RDF Pattern Matching using Sortable Views

Zhihong Chong, He Chen, Zhenjie Zhang
Hu Shu, Guilin Qi, Aoying Zhou

chong.seu@gmail.com
cse.seu.edu.cn/people/zhchong

School of Computer Science and Engineering, Southeast University, China

RDF Pattern Matching using Sortable Views

Zhihong Chong, He Chen, Zhenjie Zhang
Hu Shu, Guilin Qi, Aoying Zhou

chong.seu@gmail.com
cse.seu.edu.cn/people/zhchong

School of Computer Science and Engineering, Southeast University, China

RDF Pattern Matching using Sortable Views

Zhihong Chong, He Chen, Zhenjie Zhang
Hu Shu, Guilin Qi, Aoying Zhou

chong.seu@gmail.com
cse.seu.edu.cn/people/zhchong

School of Computer Science and Engineering, Southeast University, China

Outline

- RDF Pattern Rewriting using Views
- Sortable View
- Rewriting using Sortable Views
- Evaluation

RDF Pattern Rewriting using View

subject	predicate	object
Alice	Marry	Bob
Alice	isMother	Chris
Chris	coAuthor	Bob
Chris	coAuthor	Frank
Alice	isMother	David
…	….	…

RDF Pattern Rewriting using View

subject	predicate	object
Alice	Marry	Bob
Alice	isMother	Chris
Chris	coAuthor	Bob
Chris	coAuthor	Frank
Alice	isMother	David
…	….	…

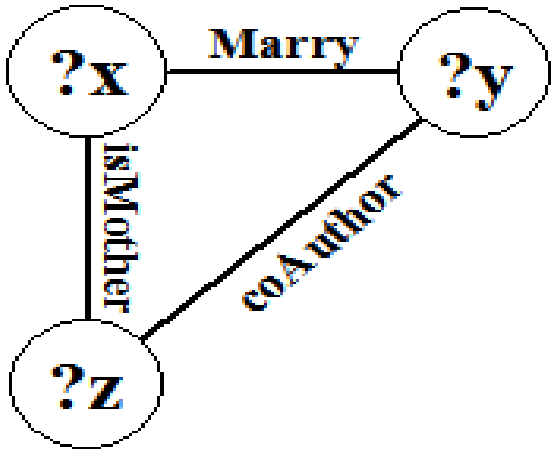
Fathers and sons who are coauthored with each other?

RDF Pattern Rewriting using View

subject	predicate	object
Alice	Marry	Bob
Alice	isMother	Chris
Chris	coAuthor	Bob
Chris	coAuthor	Frank
Alice	isMother	David
…	….	…

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

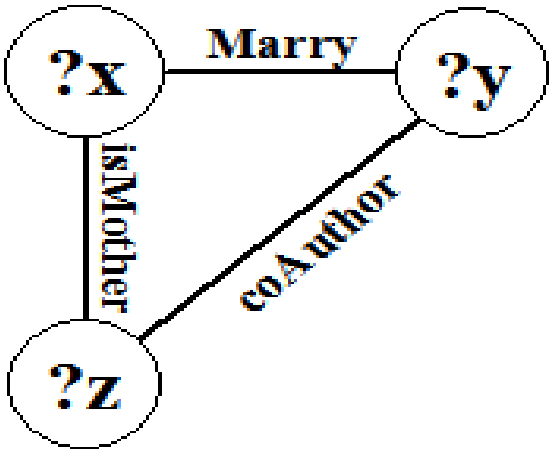
RDF Pattern Rewriting using View



View V_1

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View

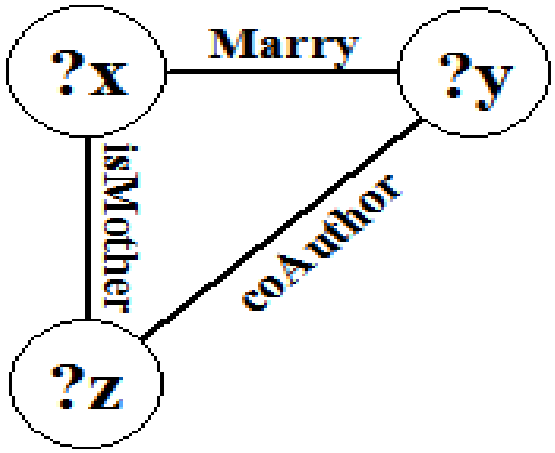


View V_1

Husbands and their wives
(?x, Marry, ?y)

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View



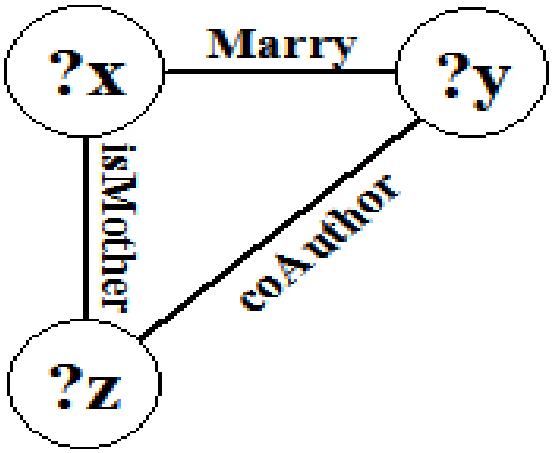
View V_1

Husbands and their wives
(?x, Marry, ?y)

Mothers and their sons
(?x, isMother, ?z)

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View



View V_1

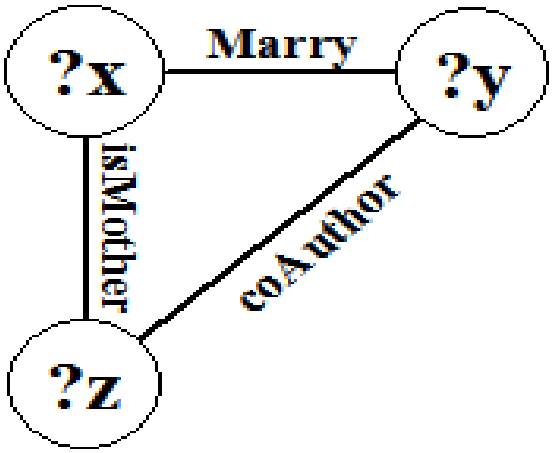
Join on ?x:
Fathers (?y) and their sons (?z)

Husbands and their wives
(?x, Marry, ?y)

Mothers and their sons
(?x, isMother, ?z)

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View



View V_1

Join on ?x:
Fathers (?y) and their sons (?z)

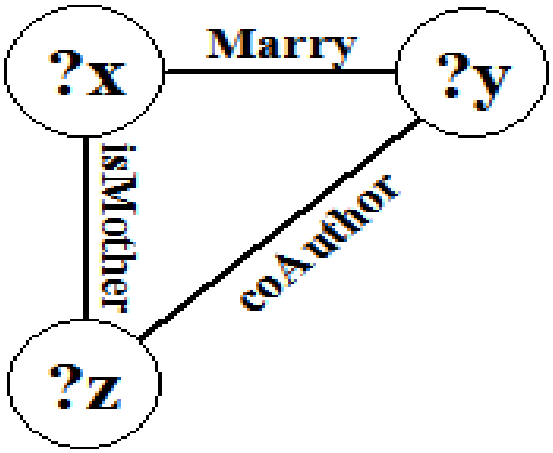
Husbands and their wives
(?x, Marry, ?y)

Mothers and their sons
(?x, isMother, ?z)

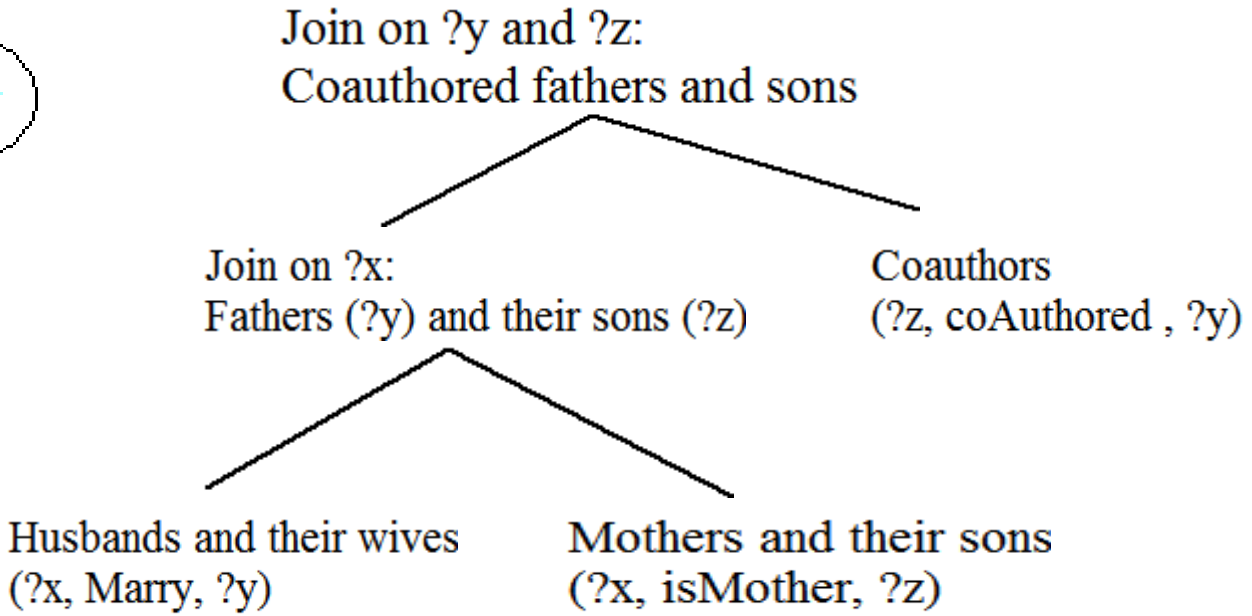
Coauthors
(?z, coAuthored , ?y)

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View

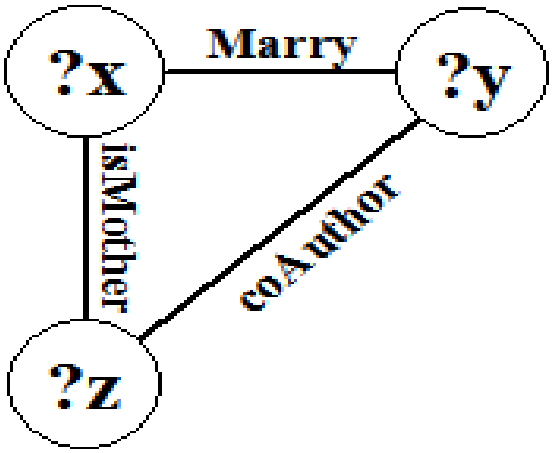


View V_1



(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View

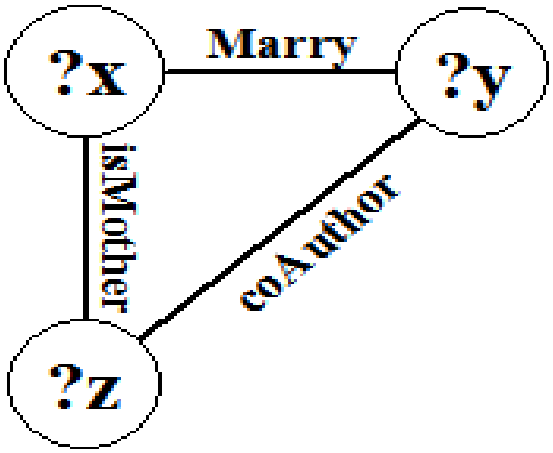


View V_1

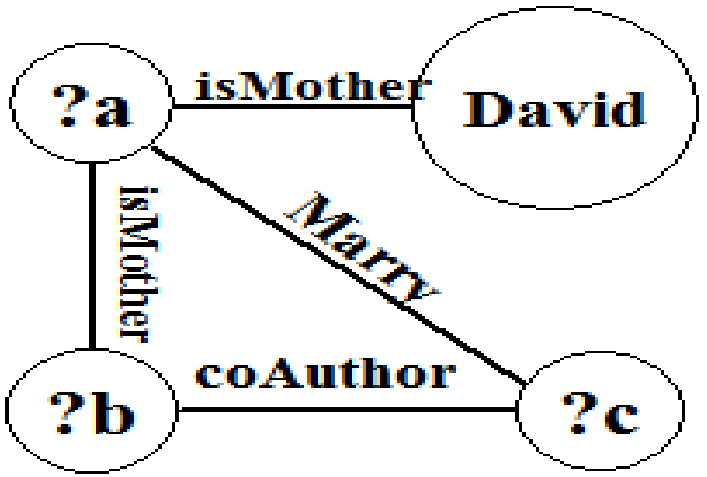
$[V_1]$	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)

RDF Pattern Rewriting using View

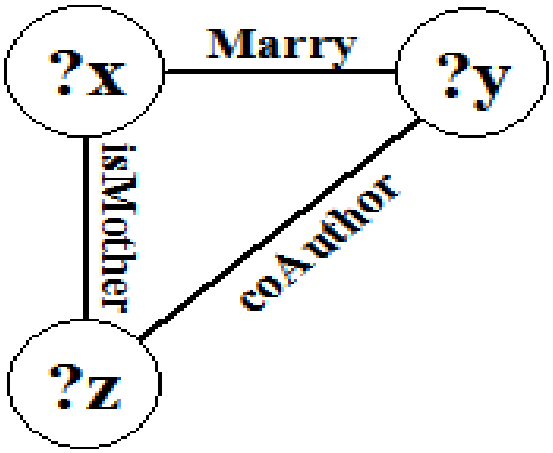


View V_1

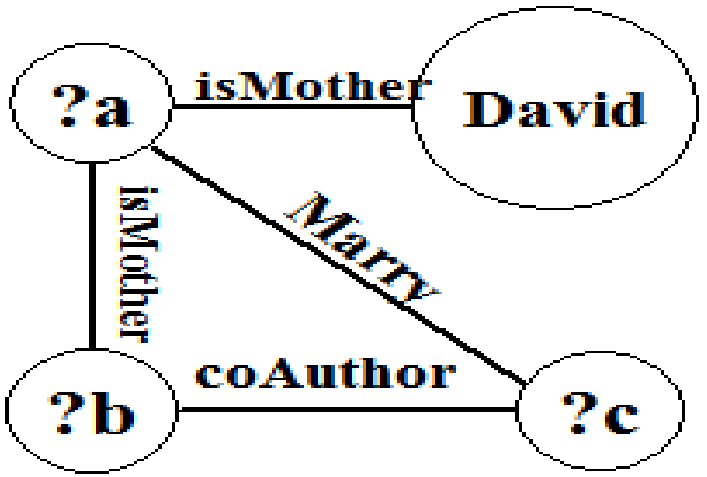


Pattern Q_1

RDF Pattern Rewriting using View



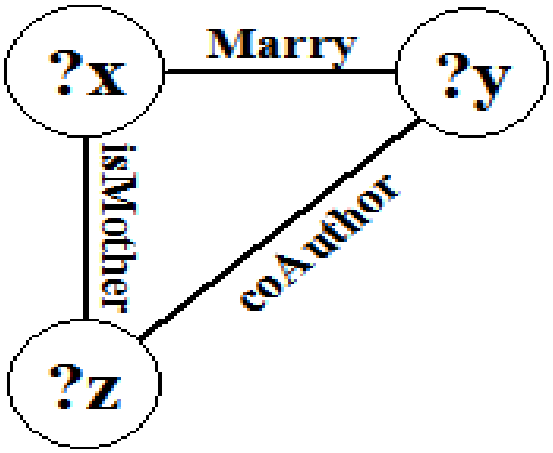
View V_1



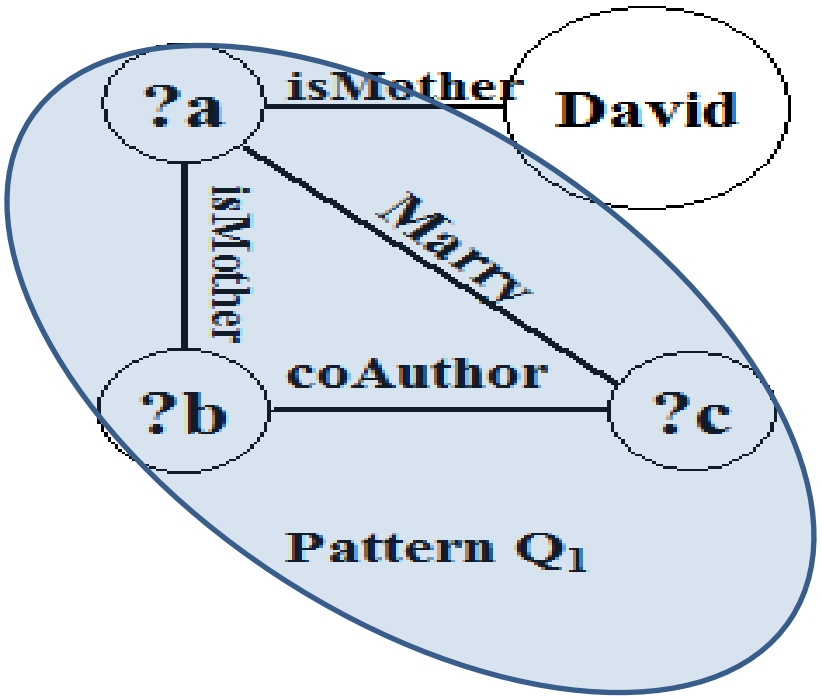
Pattern Q_1

David's brothers who are coauthored with their fathers?

RDF Pattern Rewriting using View

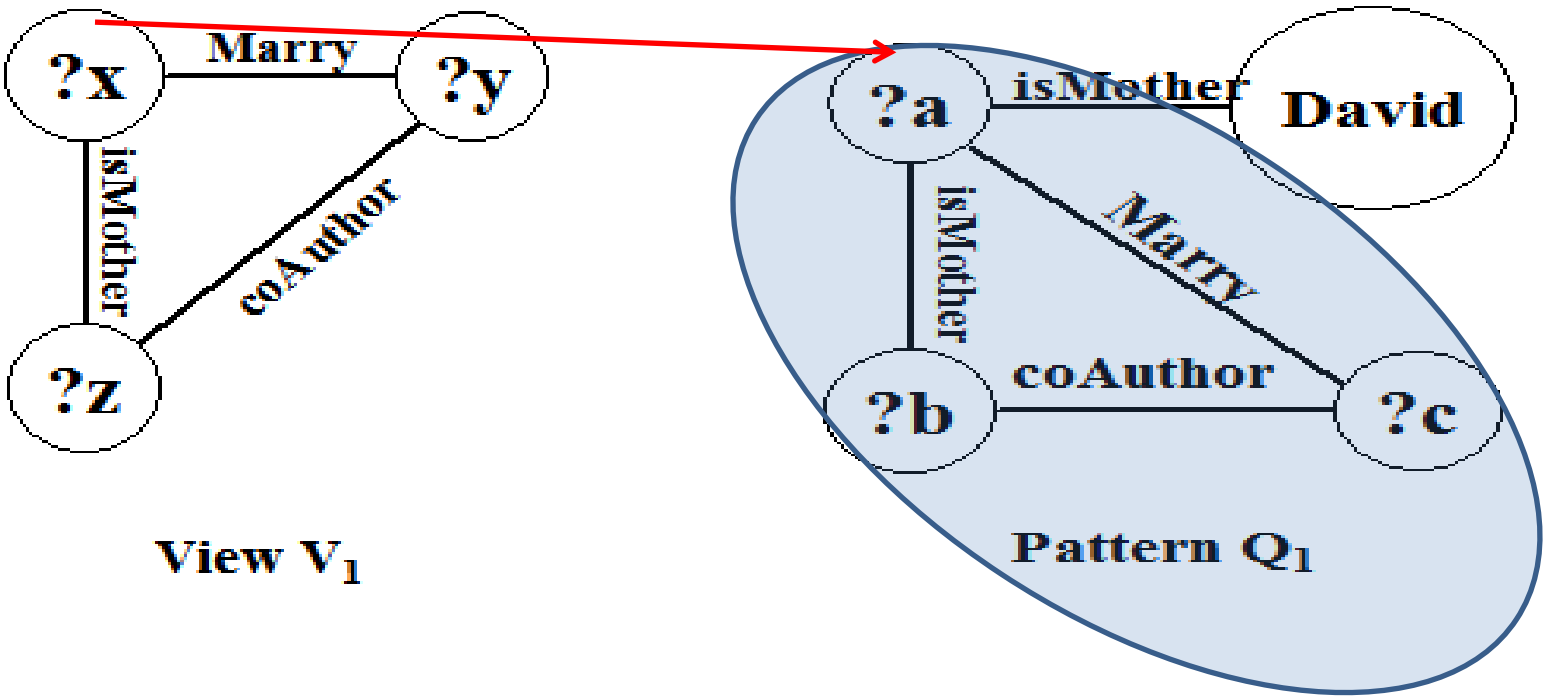


View V_1



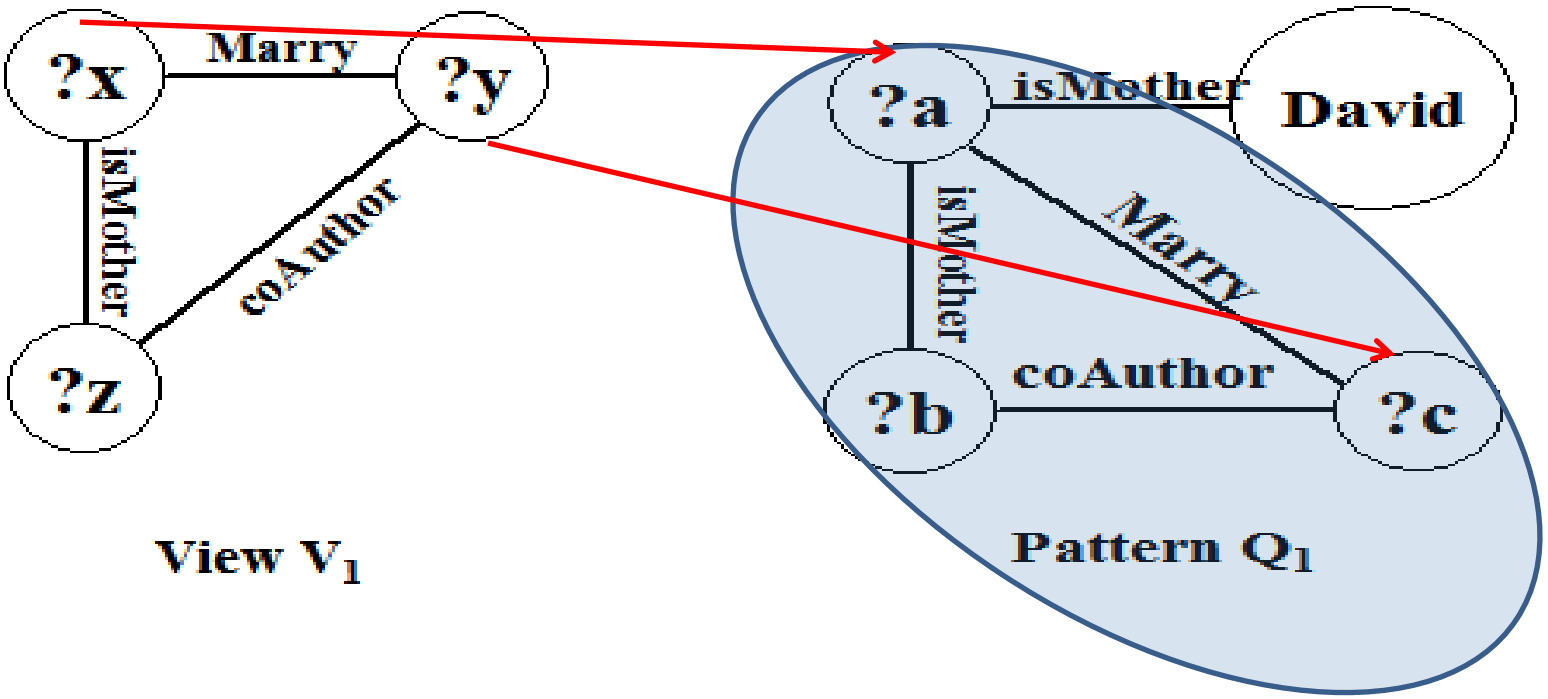
David's brothers who are coauthored with their fathers?

RDF Pattern Rewriting using View



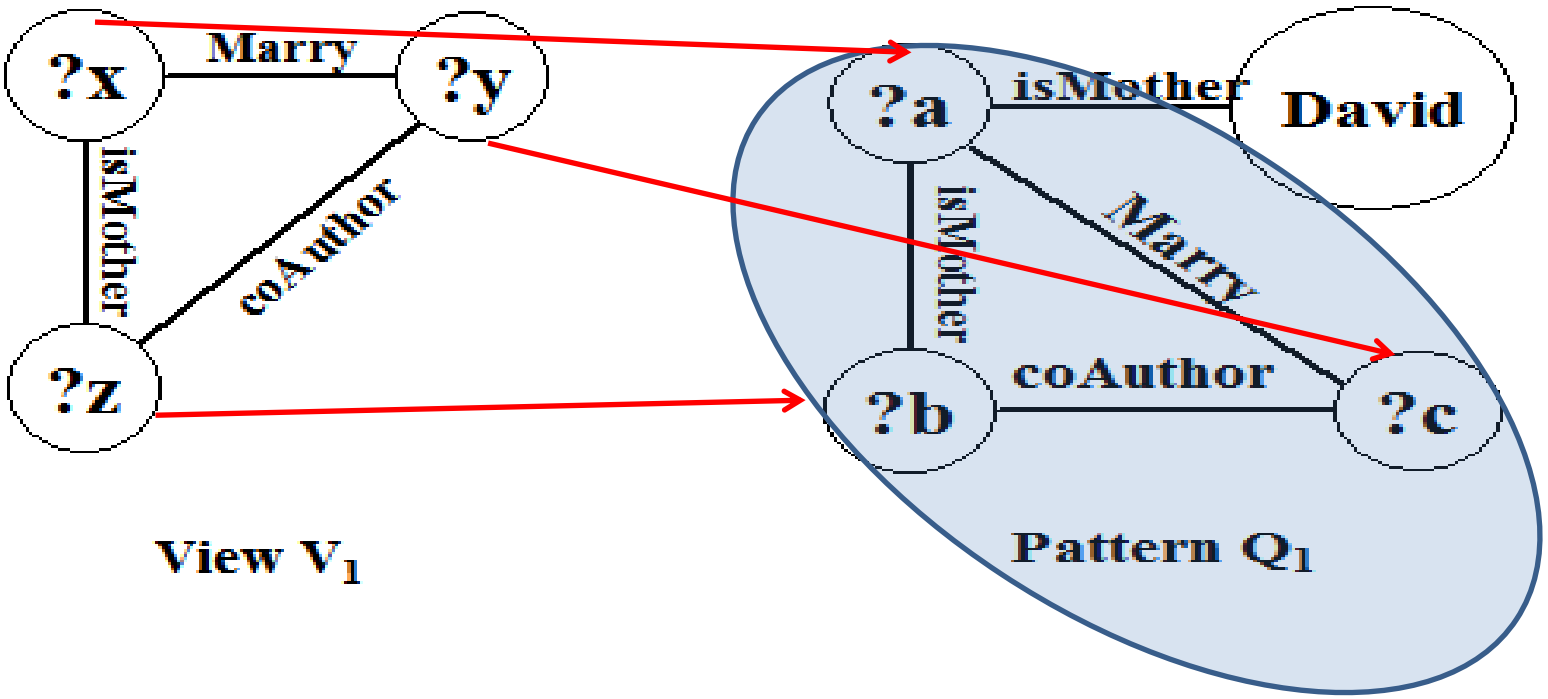
David's brothers who are coauthored with their fathers?

RDF Pattern Rewriting using View



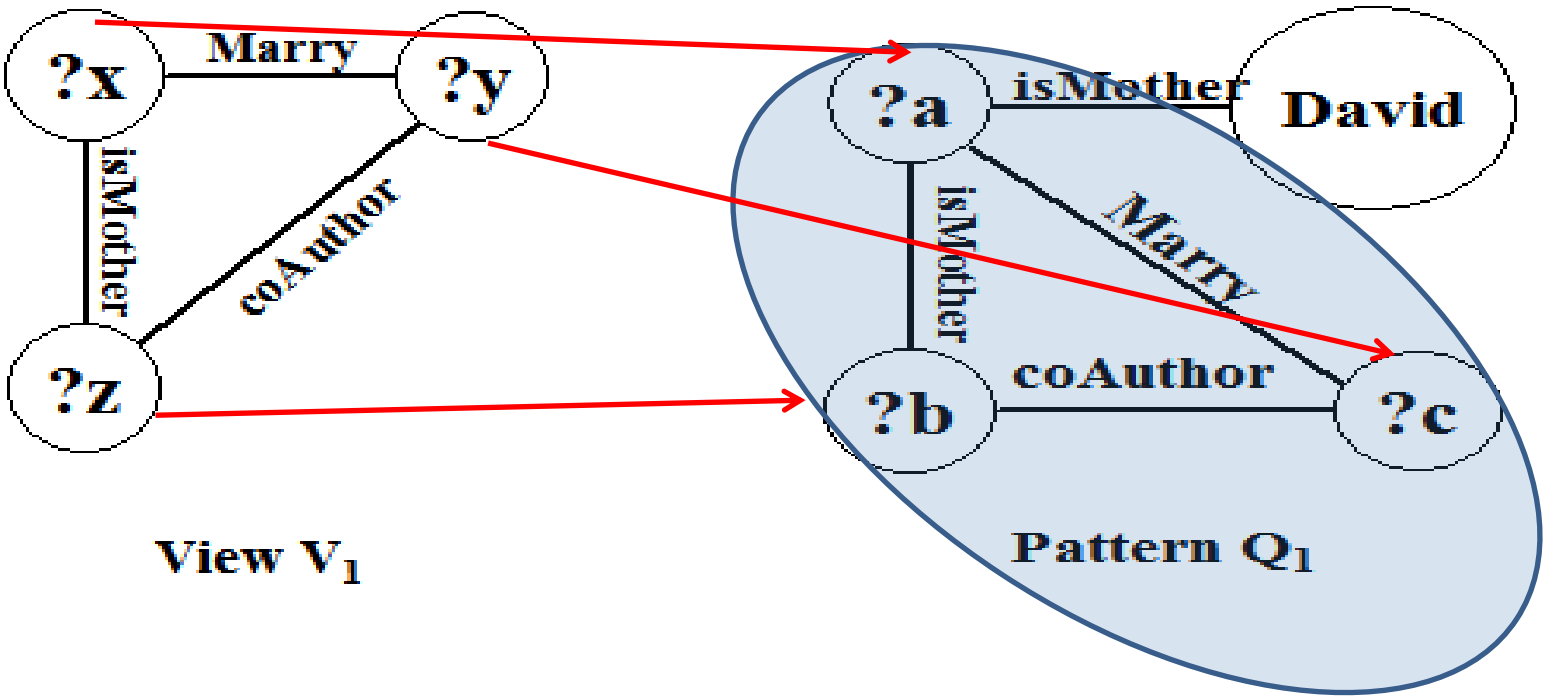
David's brothers who are coauthored with their fathers?

RDF Pattern Rewriting using View



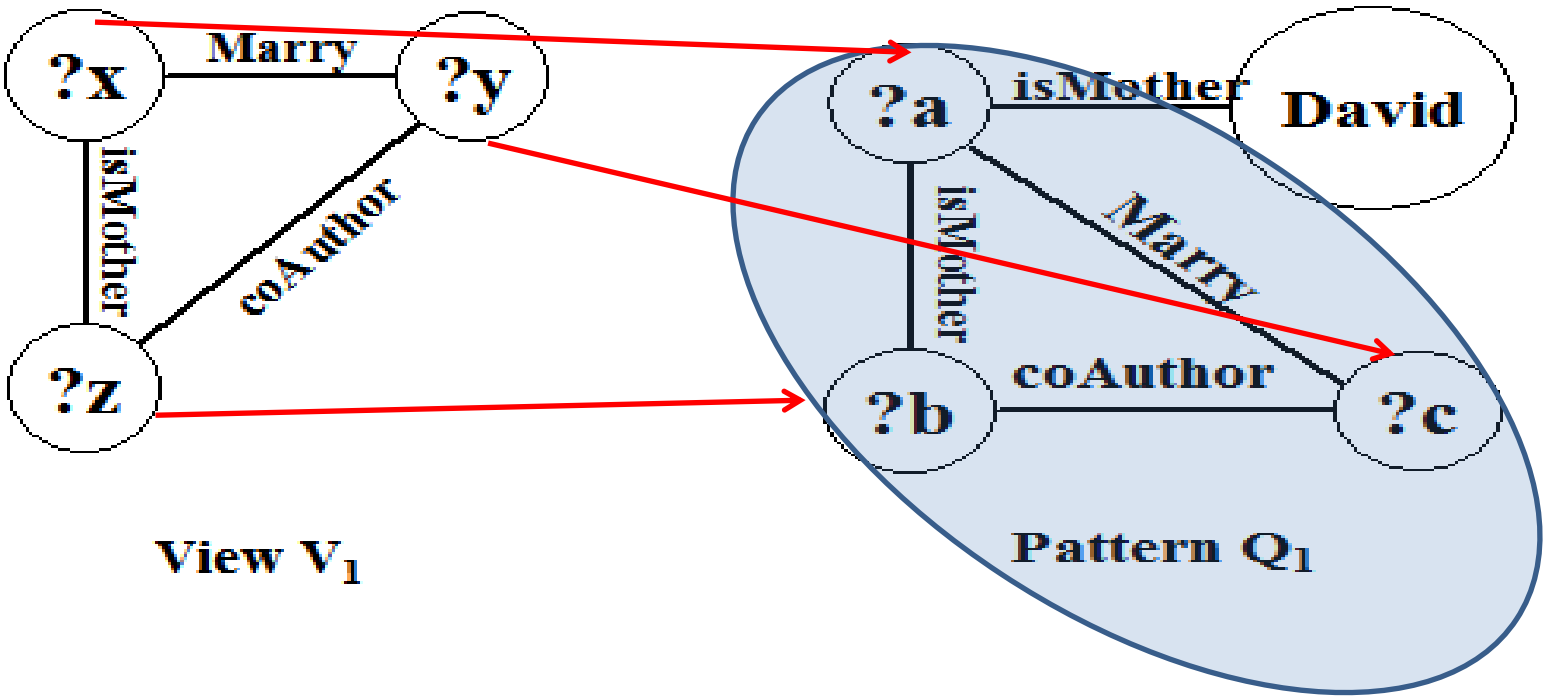
David's brothers who are coauthored with their fathers?

RDF Pattern Rewriting using View



V1	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

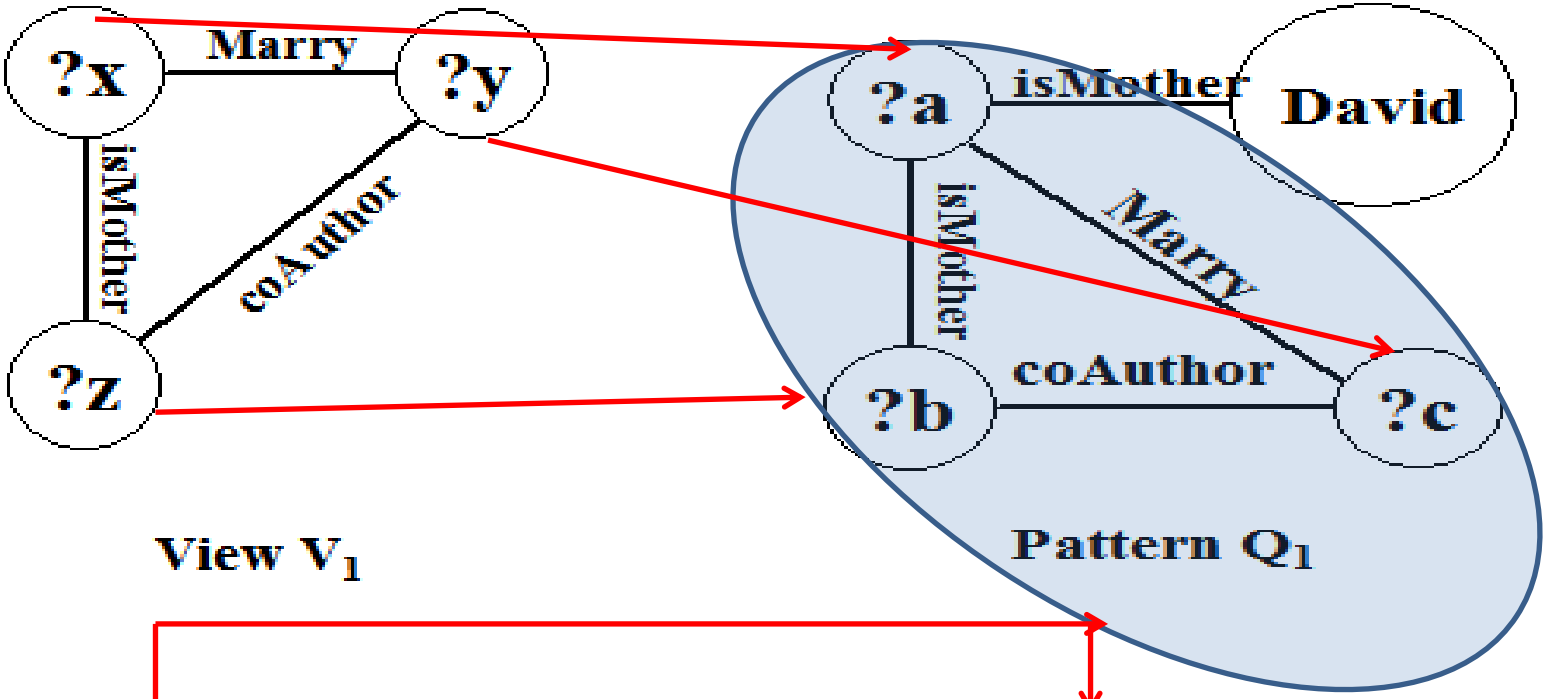
RDF Pattern Rewriting using View



[V1]	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

[Q'1]	?a	?c	?b
	Alice	Bob	Chris
	…	…	…

RDF Pattern Rewriting using View

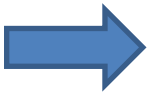


[V1]	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

[Q'1]	?a	?c	?b
	Alice	Bob	Chris
	…	…	…

RDF Pattern Rewriting using View

[V1]	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

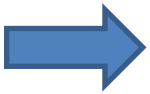


Q'1	?a	?c	?b
	Alice	Bob	Chris
	…	…	…

RDF Pattern Rewriting using View

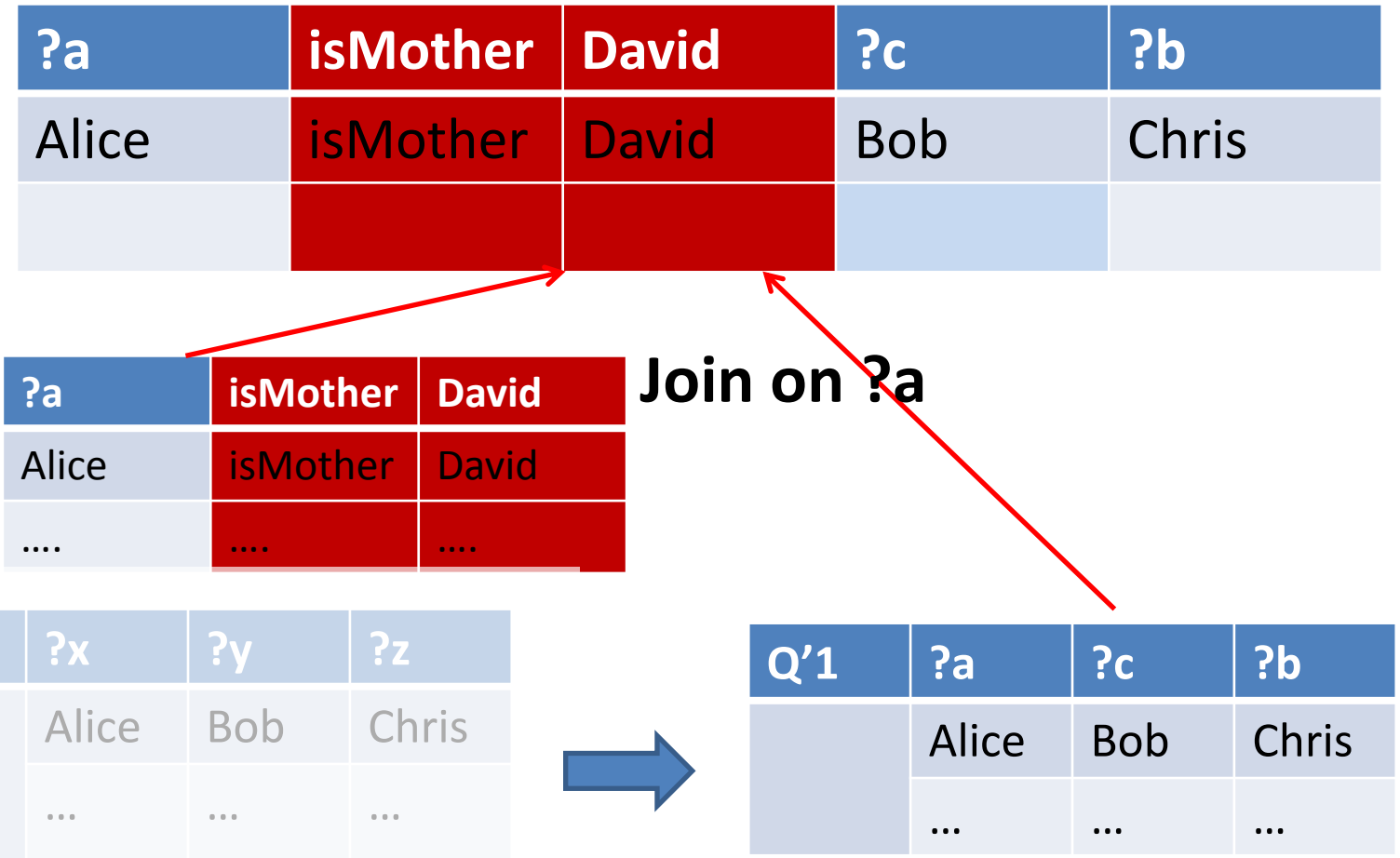
?a	isMother	David
Alice	isMother	David
….	….	….

[V1]	?x	?y	?z
	Alice	Bob	Chris
	…	…	…

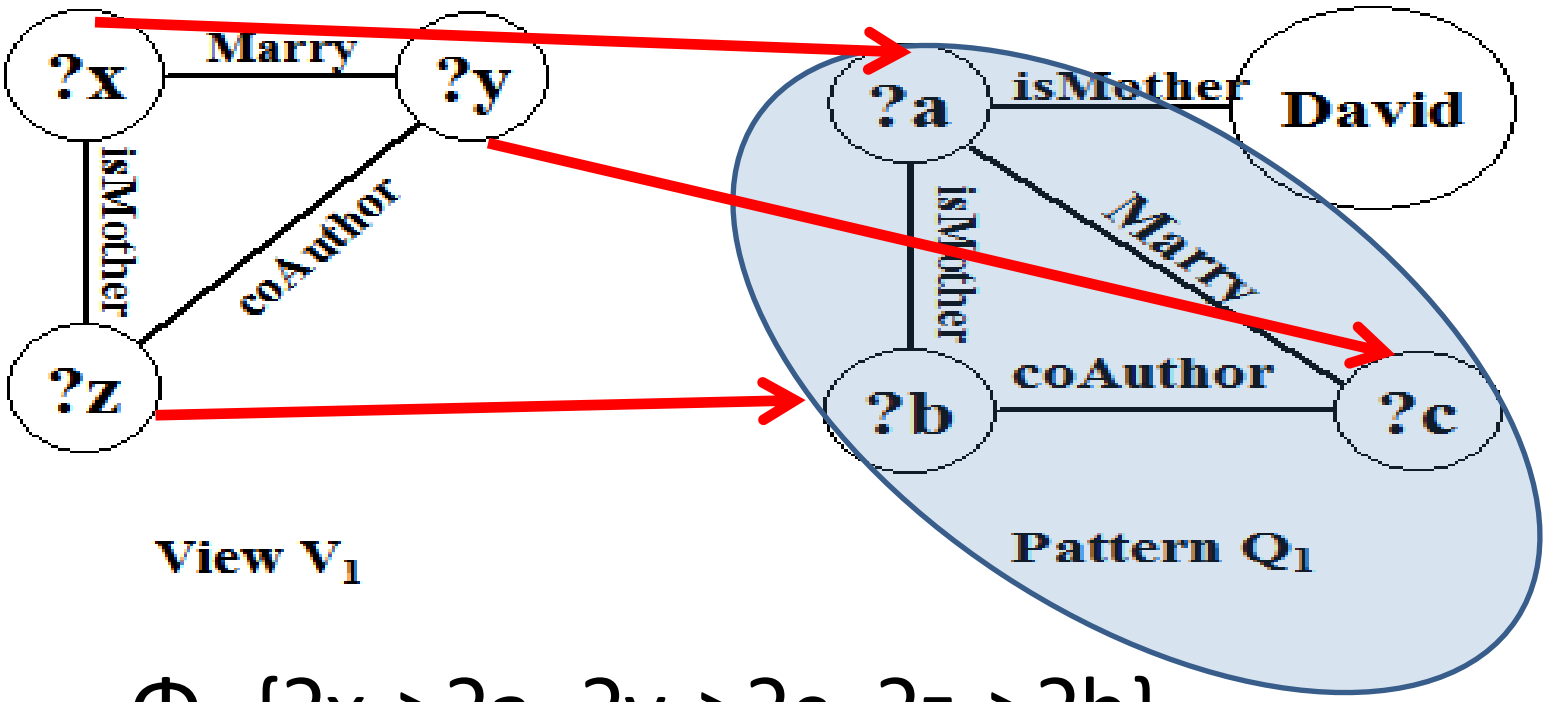


Q'1	?a	?c	?b
	Alice	Bob	Chris
	…	…	…

RDF Pattern Rewriting using View

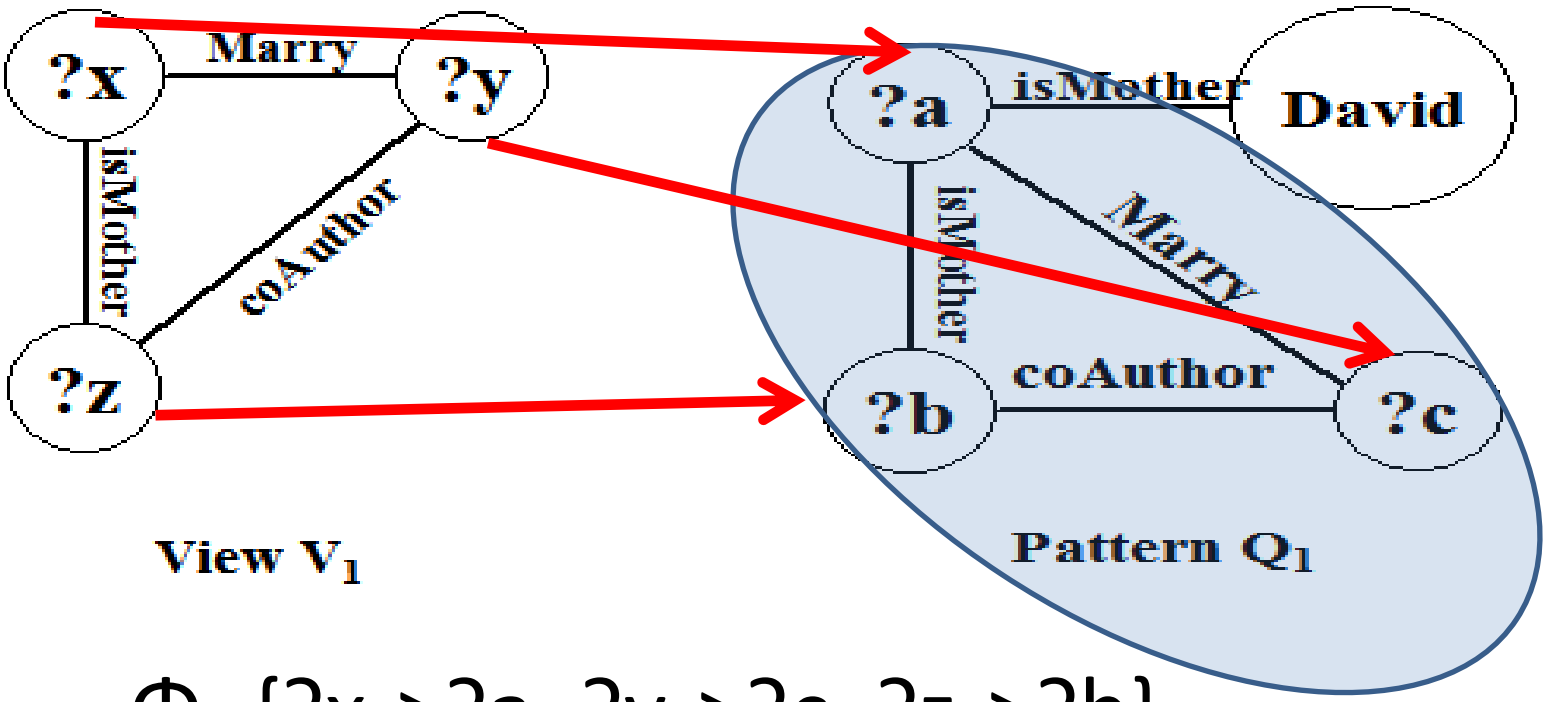


RDF Pattern Rewriting using View



Φ={?x->?a, ?y->?c, ?z->?b}

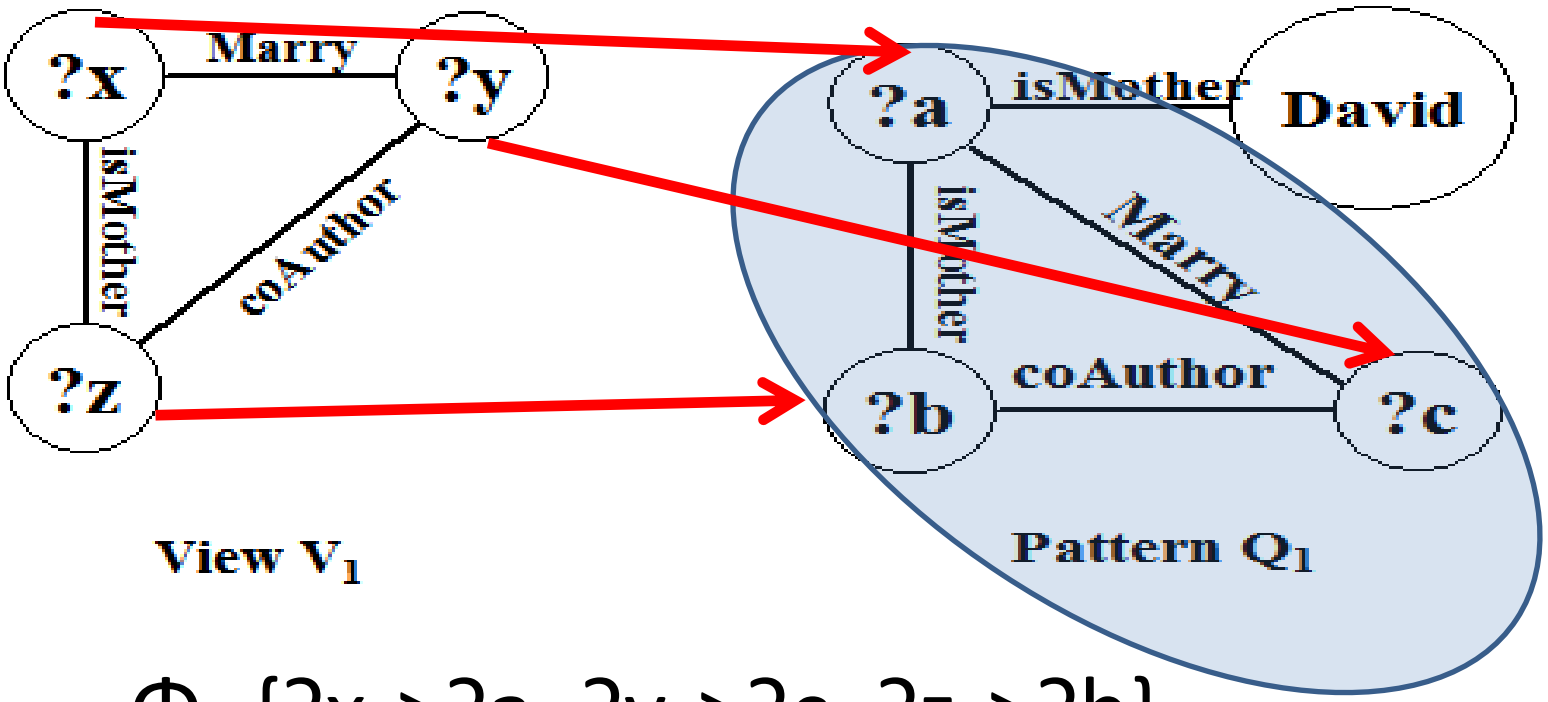
RDF Pattern Rewriting using View



Φ={?x->?a, ?y->?c, ?z->?b}

Containment mapping Φ

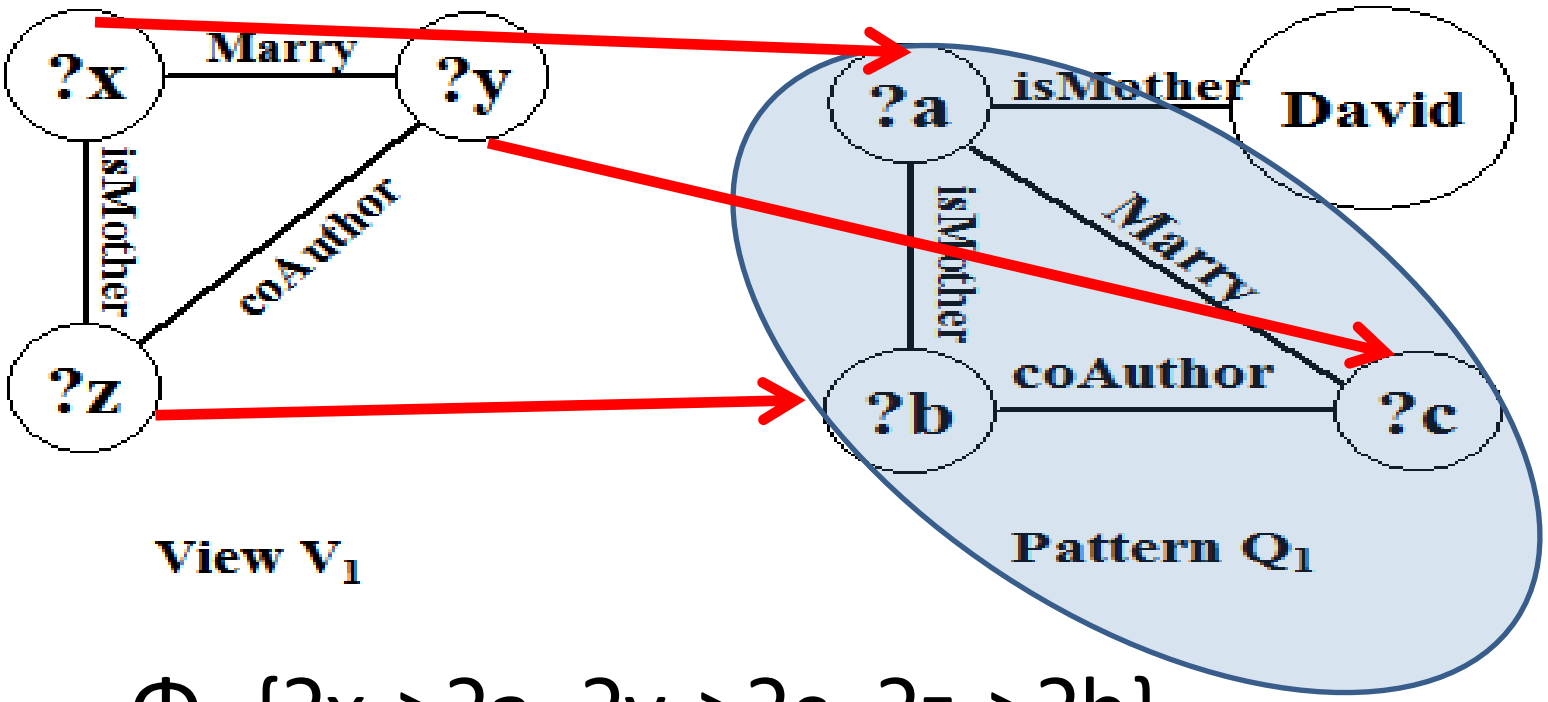
RDF Pattern Rewriting using View



Φ={?x->?a, ?y->?c, ?z->?b}

Q_1=Φ(V_1) union {(?a, isMother, David)}

RDF Pattern Rewriting using View

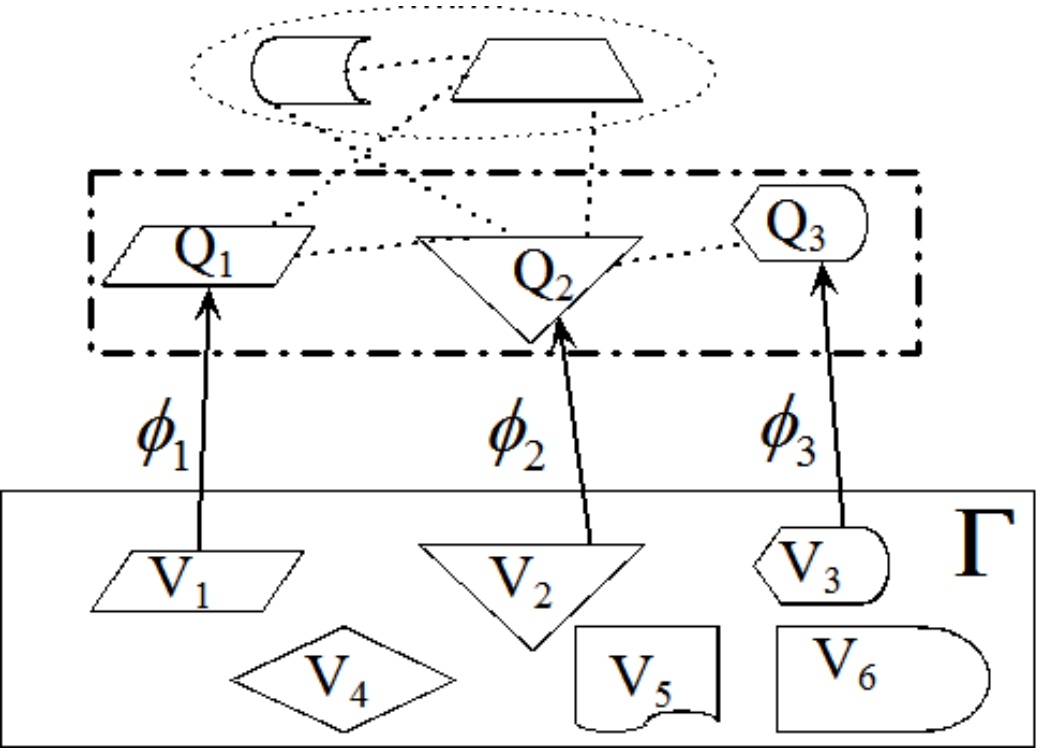


Φ={?x->?a, ?y->?c, ?z->?b}

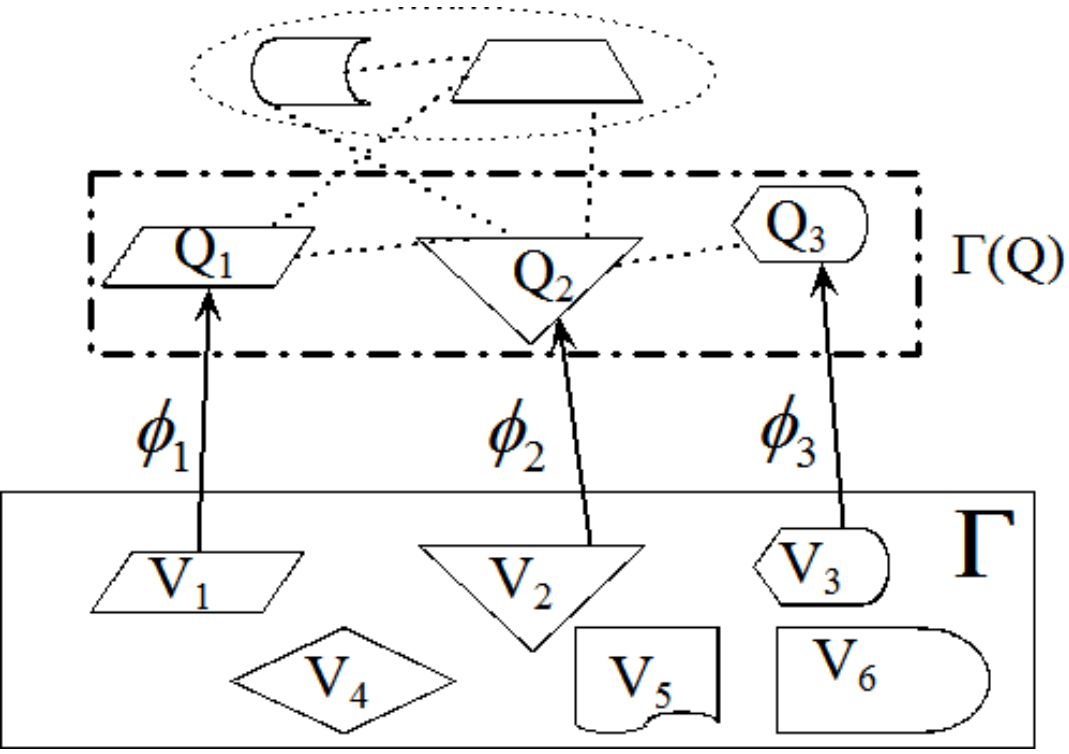
Q_1=Φ(V_1) union {(?a, isMother, David)}

[Q_1]=Φ([V_1])$\bowtie$[{(?a, isMother, David)}]

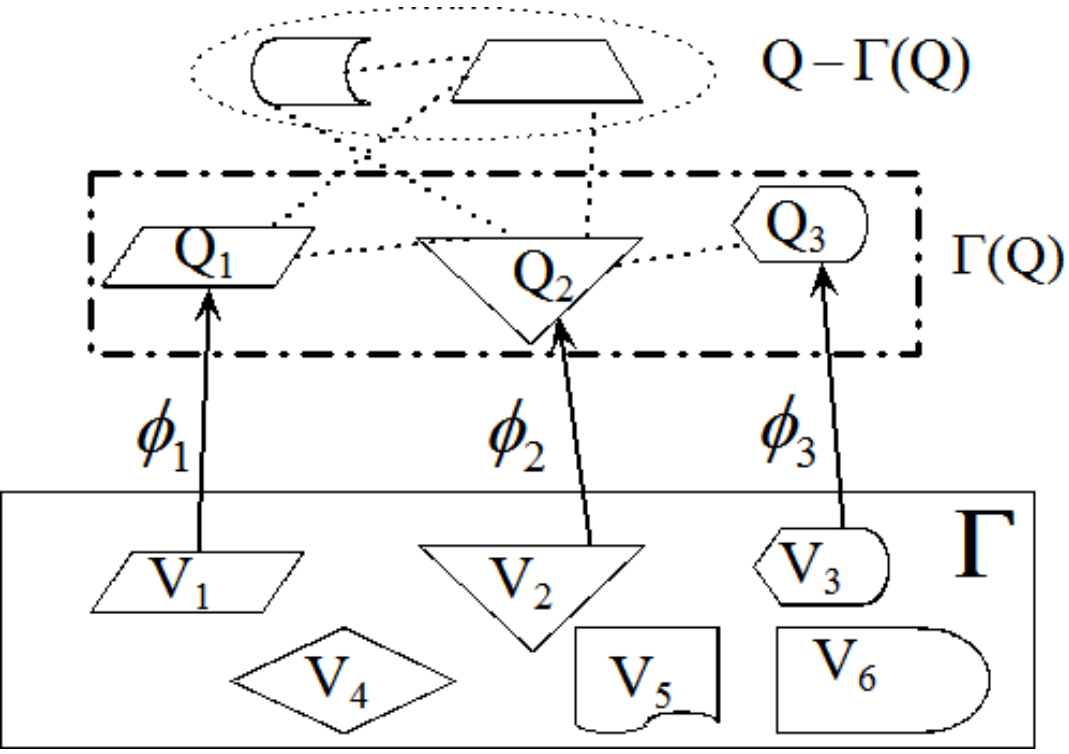
RDF Pattern Rewriting using View



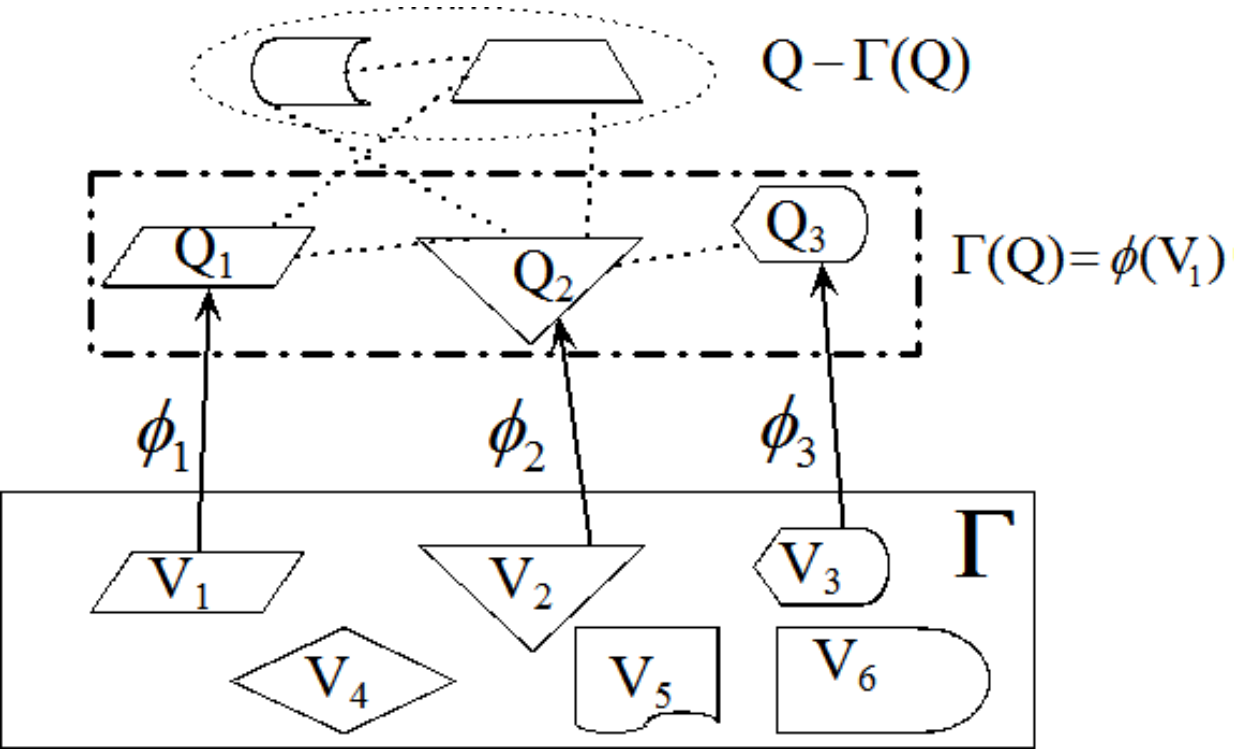
RDF Pattern Rewriting using View



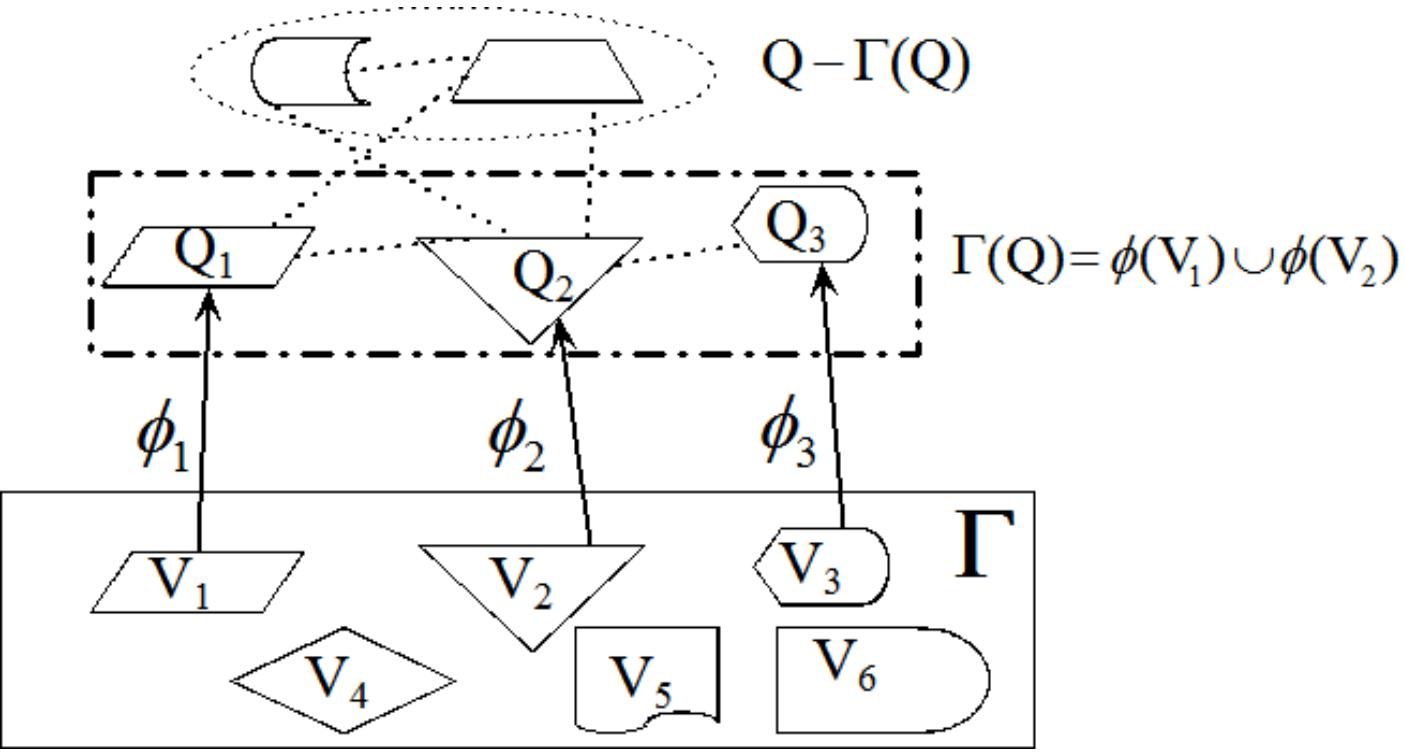
RDF Pattern Rewriting using View



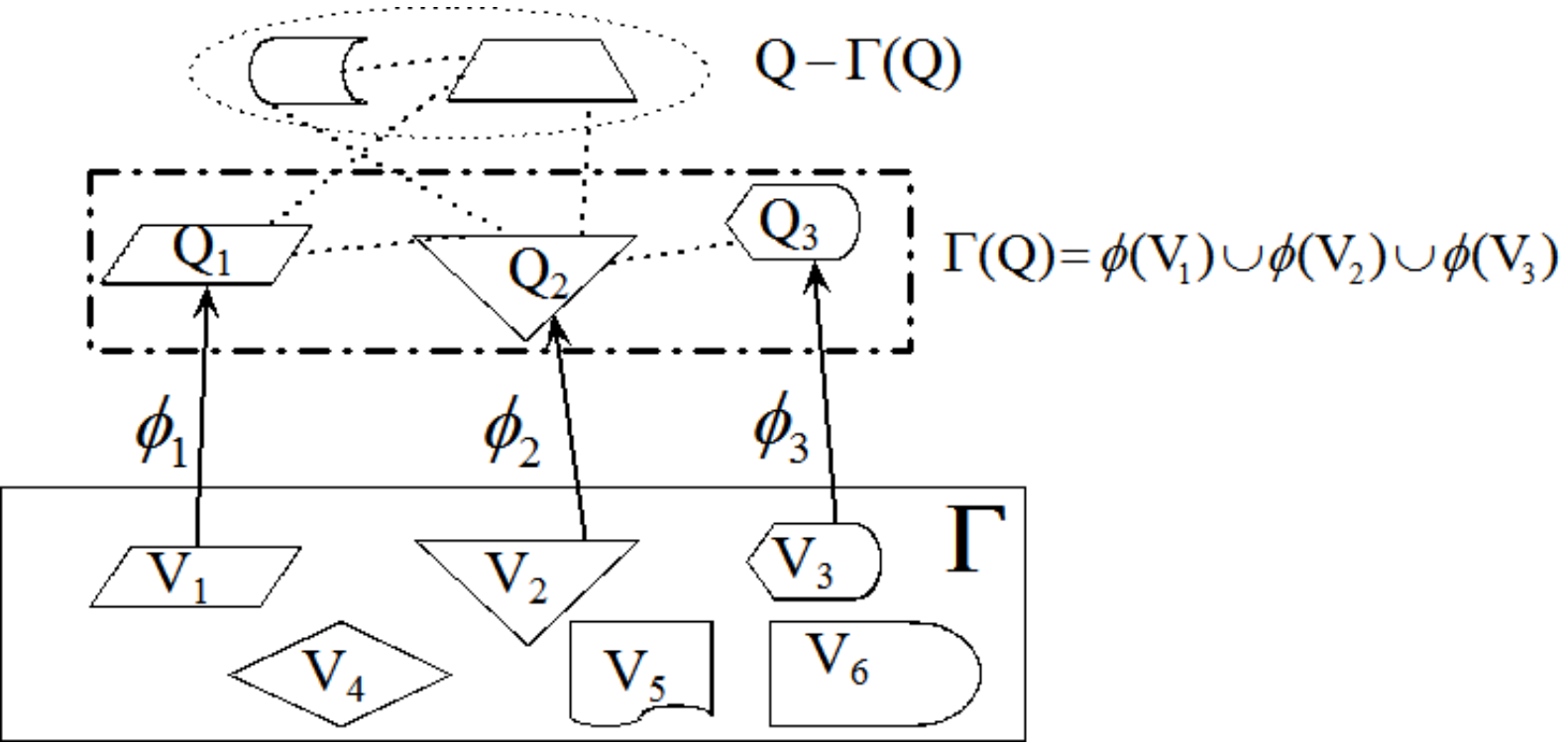
RDF Pattern Rewriting using View



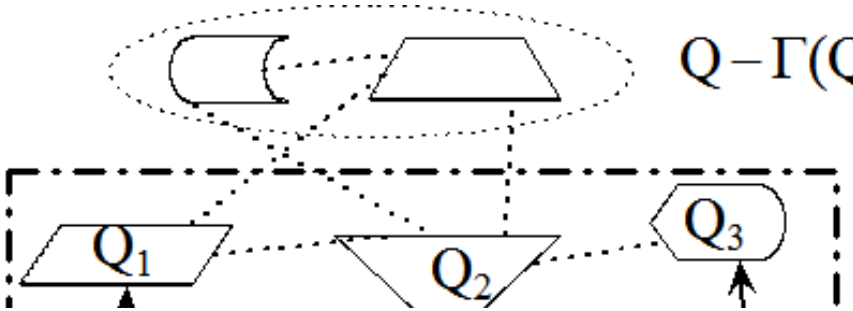
RDF Pattern Rewriting using View



RDF Pattern Rewriting using View



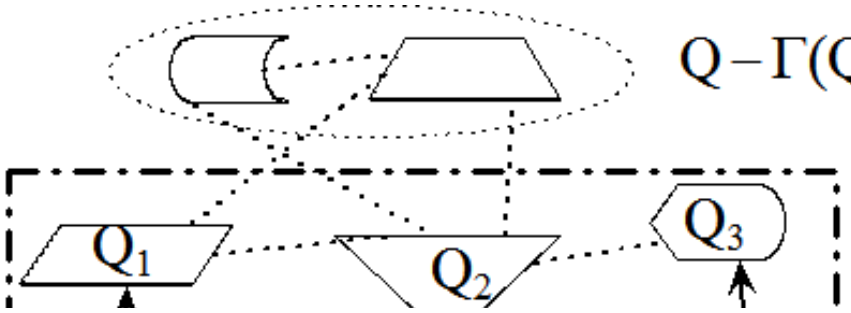
RDF Pattern Rewriting using View



$$\Gamma(Q) = \phi(V_1) \cup \phi(V_2) \cup \phi(V_3)$$

$$[Q] = [\phi_1(V_1)] \bowtie [\phi_2(V_2)] \bowtie [\phi_3(V_3)] \bowtie [Q - \Gamma(Q)]$$

RDF Pattern Rewriting using View



$$\Gamma(Q) = \phi(V_1) \cup \phi(V_2) \cup \phi(V_3)$$

$$[Q] = [\phi_1(V_1)] \bowtie [\phi_2(V_2)] \bowtie [\phi_3(V_3)] \bowtie [Q - \Gamma(Q)]$$

NP-hard to find these views V_i with their ϕ_i!!

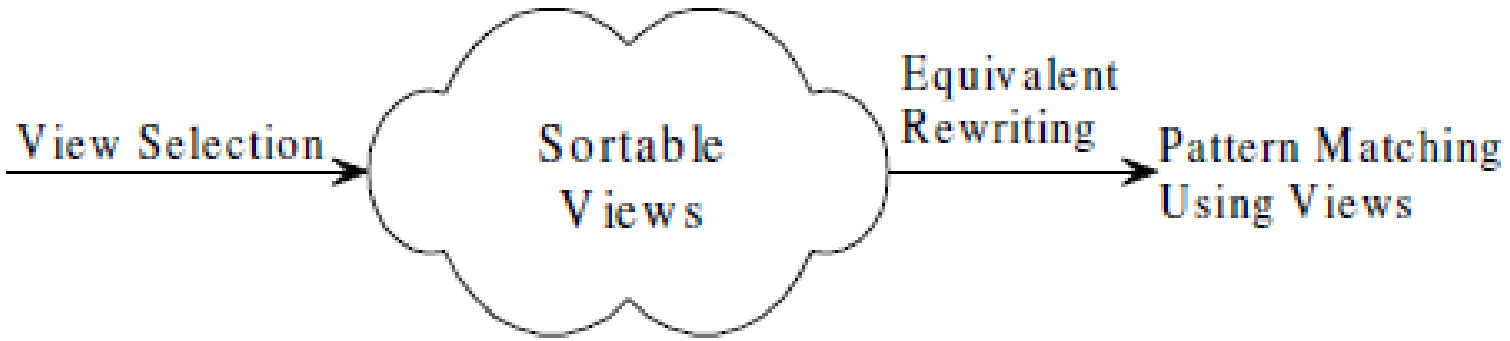
Outline

- RDF Pattern Rewriting using Views
- **Sortable View**
- Rewriting using Sortable Views
- Evaluation

Sortable View

- The problem of rewriting pattern using views is **NP-hard** in general cases

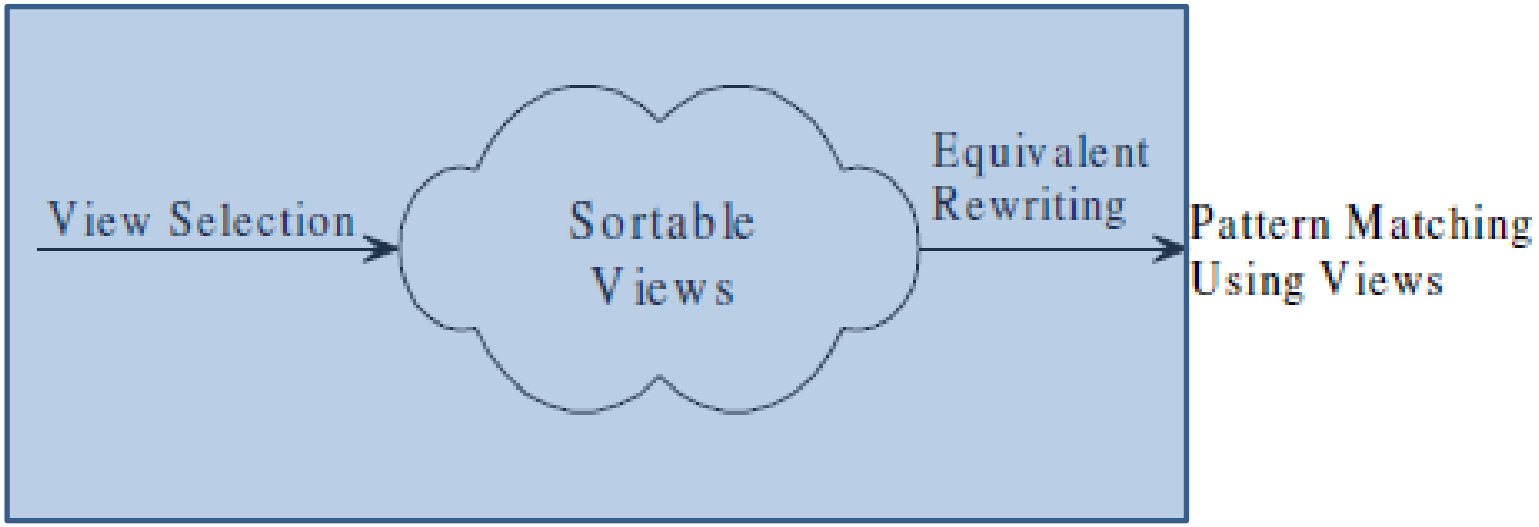
- **Sortable View**



Sortable View

- The problem of rewriting pattern using views is **NP-hard** in general cases

- **Sortable View**



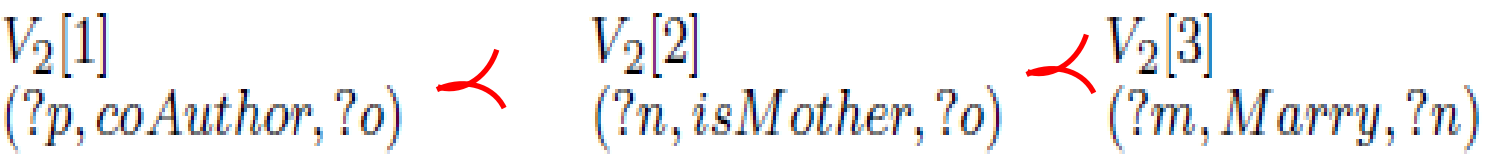
Sortable View

$V_2[1]$
$(?p, coAuthor, ?o)$

$V_2[2]$
$(?n, isMother, ?o)$

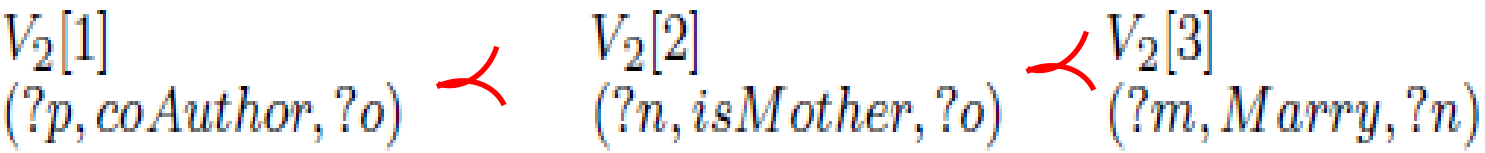
$V_2[3]$
$(?m, Marry, ?n)$

Sortable View



Sortable View

1. All variables are equivalent



Sortable View

1. All variables are equivalent
2. Two constants(strings) are ordered by their lexical order.

$V_2[1]$ $(?p, coAuthor, ?o)$ ≺ $V_2[2]$ $(?n, isMother, ?o)$ ≺ $V_2[3]$ $(?m, Marry, ?n)$

Sortable View

1. All variables are equivalent
2. Two constants(strings) are ordered by their lexical order.

3. Two triple patterns are ordered by

$V_2[1]$
$(?p, coAuthor, ?o)$ ≺ $V_2[2]$
$(?n, isMother, ?o)$ ≺ $V_2[3]$
$(?m, Marry, ?n)$

Sortable View

1. All variables are equivalent
2. Two constants(string) are ordered by their lexical order.

3. Two triple patterns are ordered by
   - Comparing their subject components first

$V_2[1]$ $(?p, coAuthor, ?o)$ ≺ $V_2[2]$ $(?n, isMother, ?o)$ ≺ $V_2[3]$ $(?m, Marry, ?n)$

Sortable View

1. All variables are equivalent
2. Two constants(string) are ordered by their lexical order.

3. Two triple patterns are ordered by
   - Comparing their subject components first
   - If equivalent, comparing their predict components

$V_2[1]$ $(?p, coAuthor, ?o)$ ≺ $V_2[2]$ $(?n, isMother, ?o)$ ≺ $V_2[3]$ $(?m, Marry, ?n)$

Sortable View

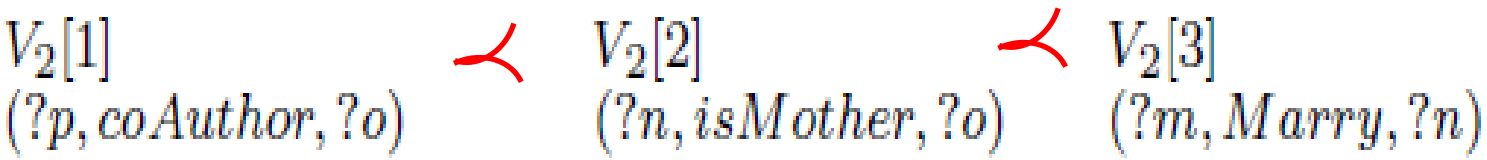
1. All variables are equivalent
2. Two constants(string) are ordered by their lexical order.

3. Two triple patterns are ordered by
   - Comparing their subject components first
   - If equivalent, comparing their predict components
   - And so on

$V_2[1]$ $(?p, coAuthor, ?o)$ ≺ $V_2[2]$ $(?n, isMother, ?o)$ ≺ $V_2[3]$ $(?m, Marry, ?n)$

Sortable View

$$(Frank, coAuthor, ?o') \prec (?n', isMother, ?o') \prec (Bob, Marry, ?n')$$



Sortable View

$$(Frank, coAuthor, ?o') \prec (?n', isMother, ?o') \prec (Bob, Marry, ?n')$$

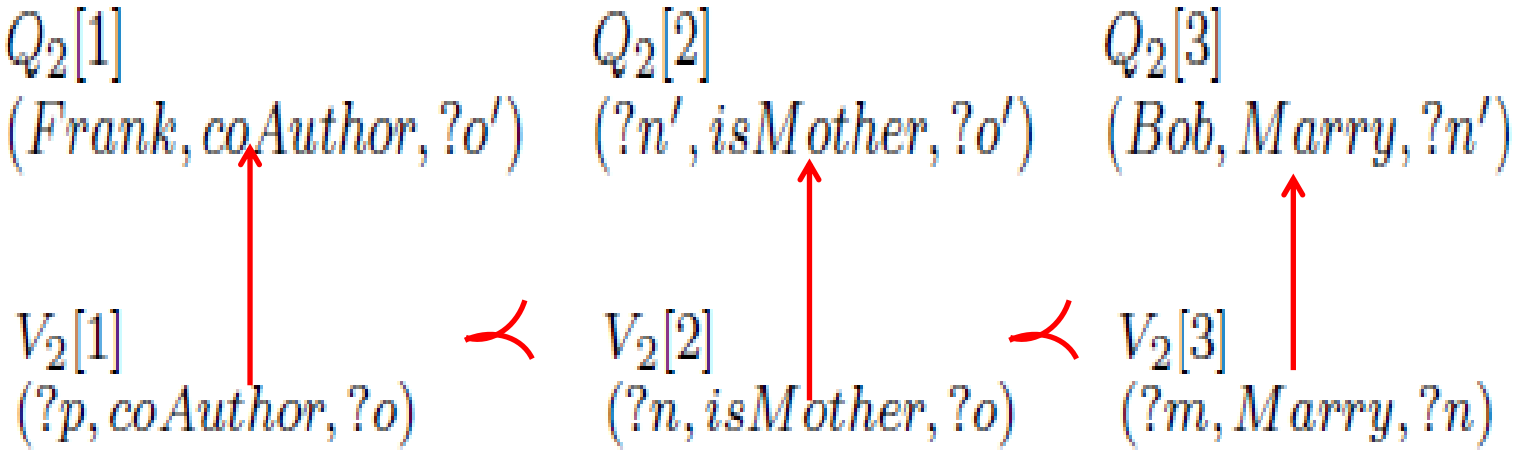
or

$$(?n', isMother, ?o') \prec (Bob, Marry, ?n') \prec (Frank, coAuthor, ?o')$$

$V_2[1]$ $(?p, coAuthor, ?o)$ $\prec$ $V_2[2]$ $(?n, isMother, ?o)$ $\prec$ $V_2[3]$ $(?m, Marry, ?n)$

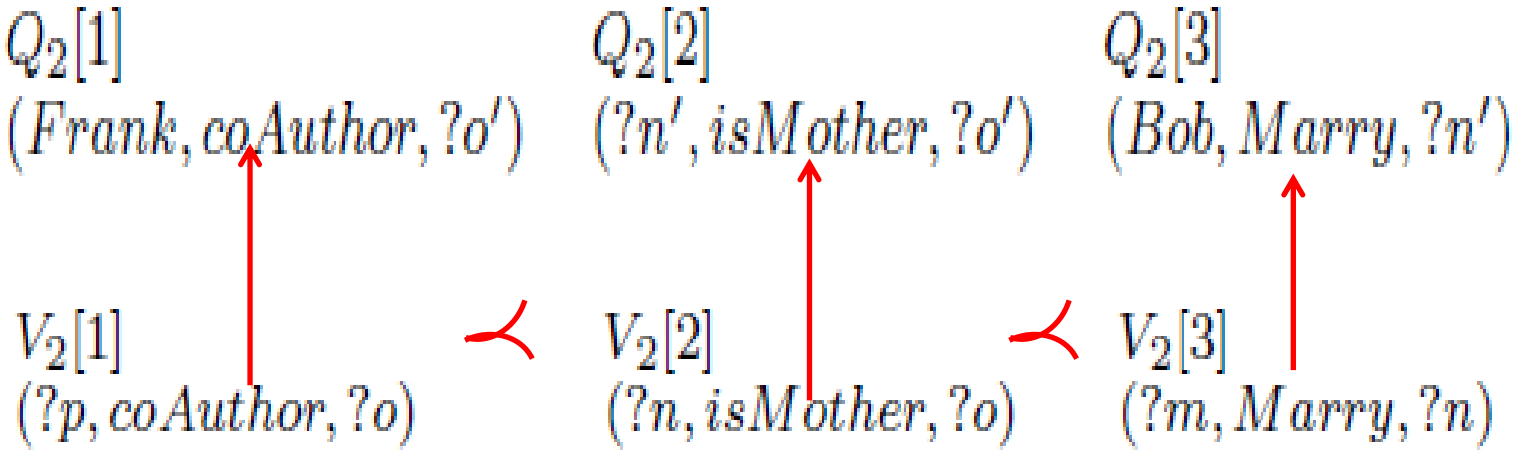
Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*



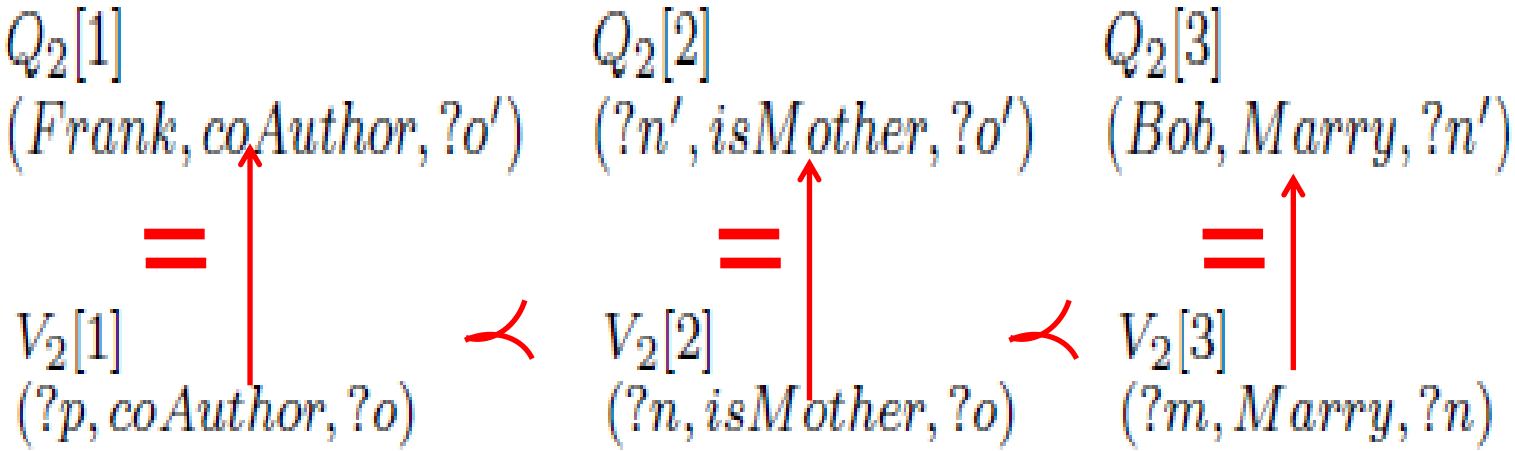
Sortable View

Serializing these triple patterns in Q_2 in terms of the sorted triple patterns in V_2 such that $V_2[i]$ is equivalent with $Q_2[i]$.



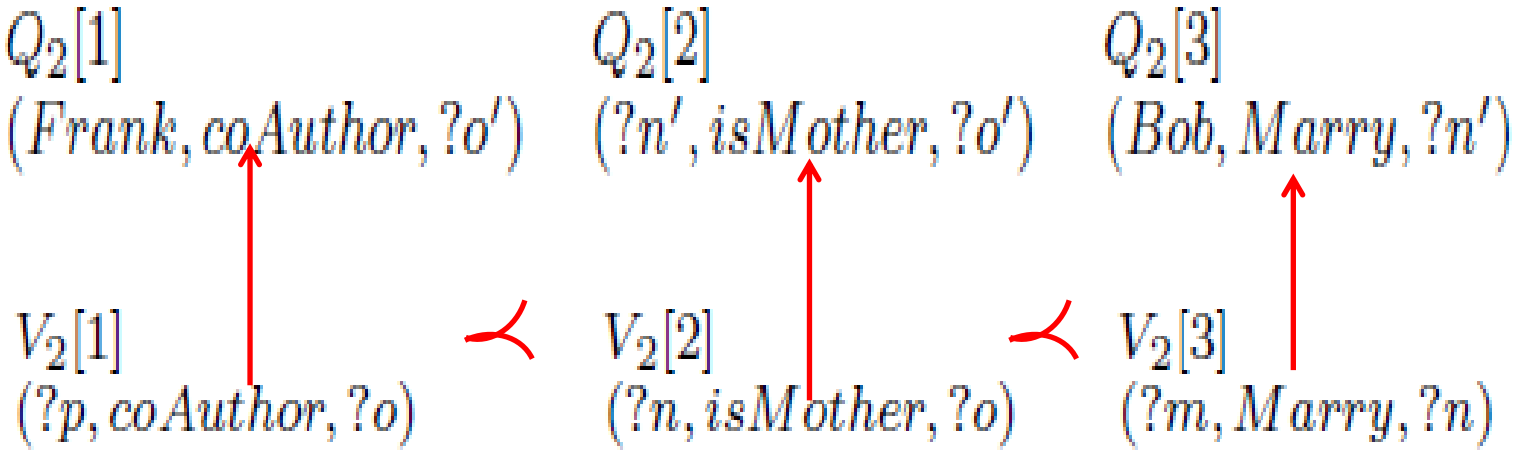
Sortable View

Serializing these triple patterns in Q_2 in terms of the sorted triple patterns in V_2 such that $V_2[i]$ is equivalent with $Q_2[i]$.



Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*



Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*

(?x, isMother, ?z)

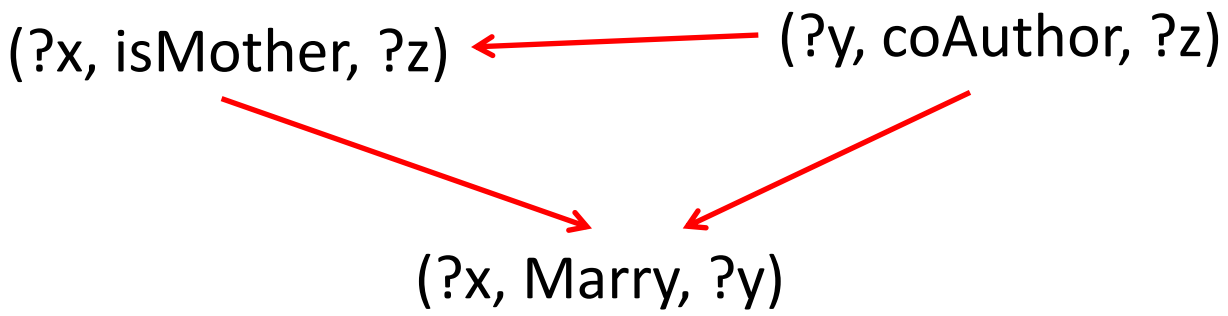
(?y, coAuthor, ?z)

(?x, Marry, ?y)

Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

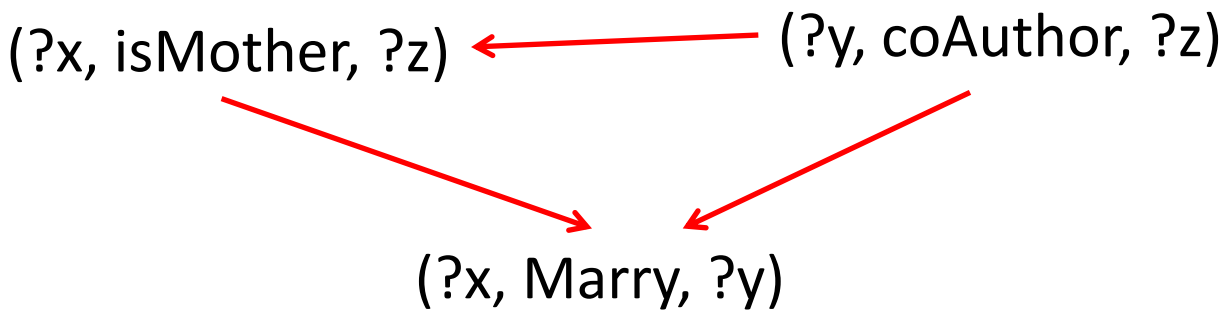
THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*



Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*

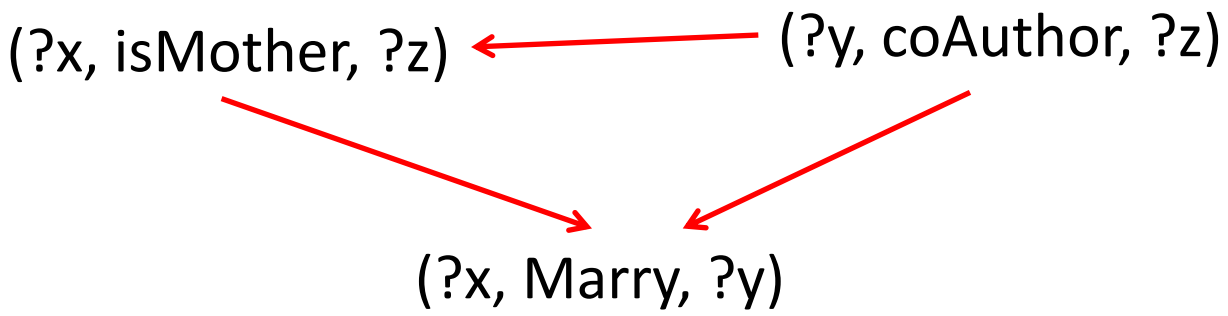


(?y, coAuthor, ?z)<(?x, isMother, ?z)<(?x, Marry, ?y)

Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*

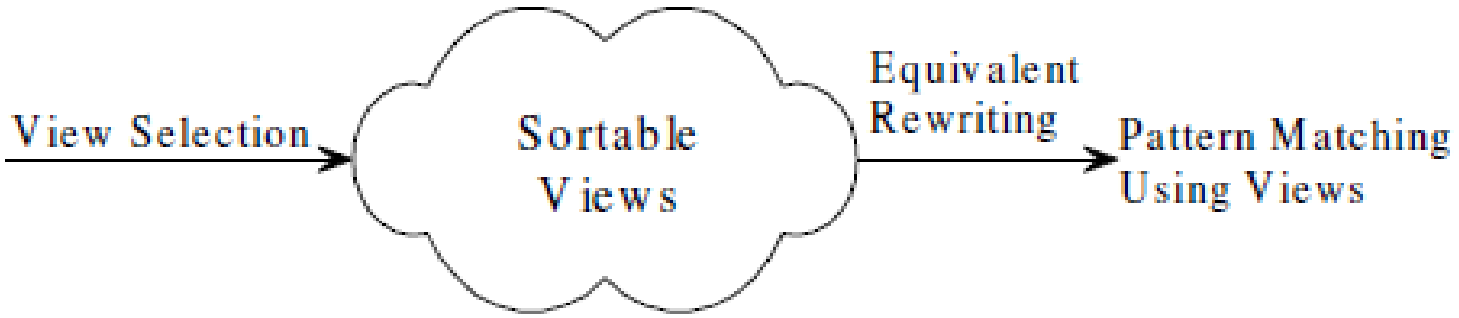


(?y, coAuthor, ?z)<(?x, isMother, ?z)<(?x, Marry, ?y)

Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

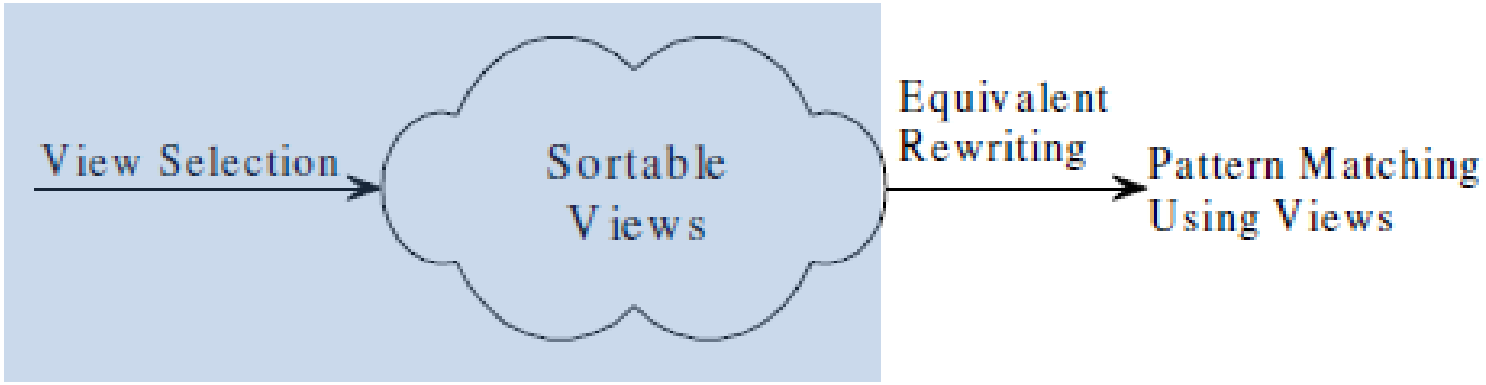
THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*



Sortable View

THEOREM 2. *Given pattern Q against sortable view V, $V \sqsupseteq Q$ if and only if there is a containment mapping ϕ such that $\phi(V[i]) = Q[i]$ for V-serialization of Q.*

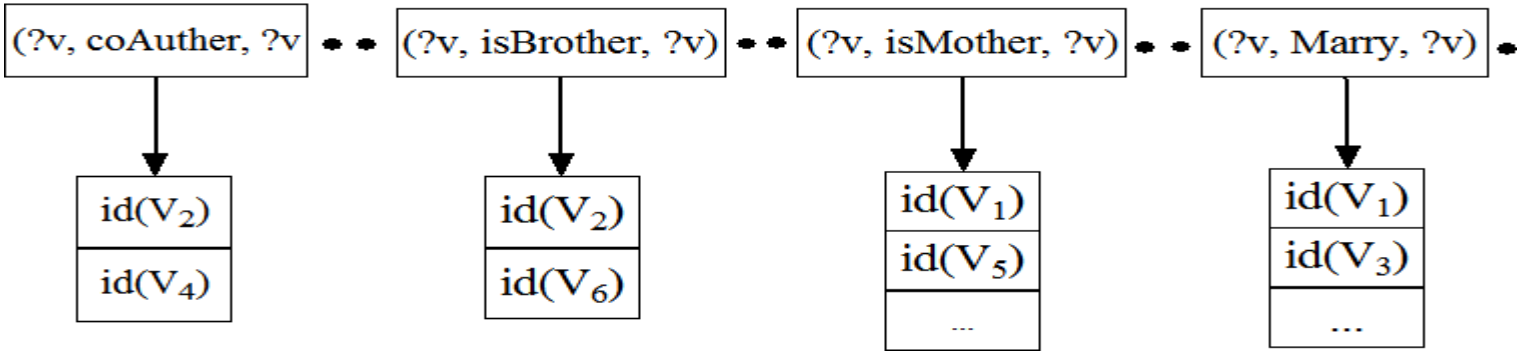
THEOREM 3. *Pattern V is sortable if and only if there is a unique topological sort of order(V).*



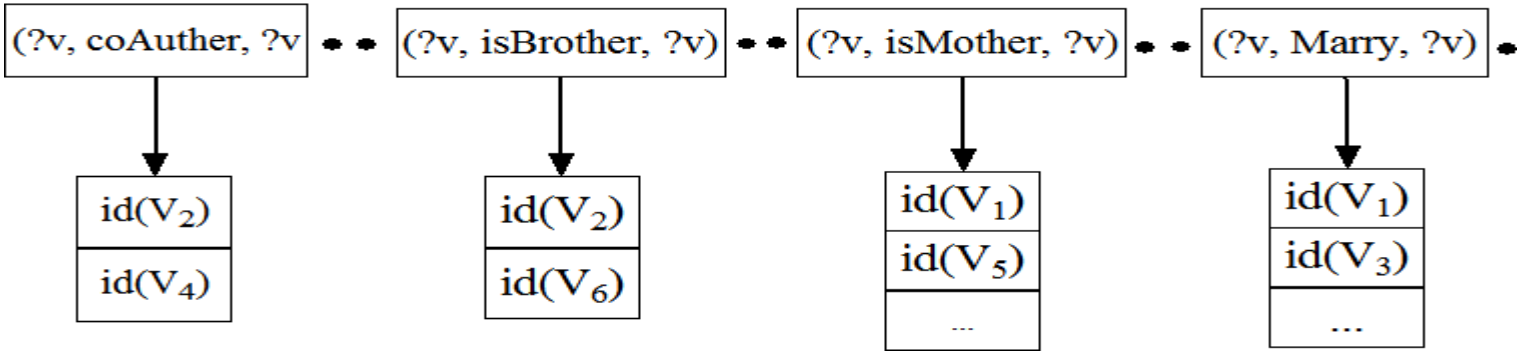
Outline

- RDF Pattern Rewriting using Views
- Sortable View
- **Rewriting using Sortable Views**
- Evaluation

Rewriting using Sortable Views



Rewriting using Sortable Views

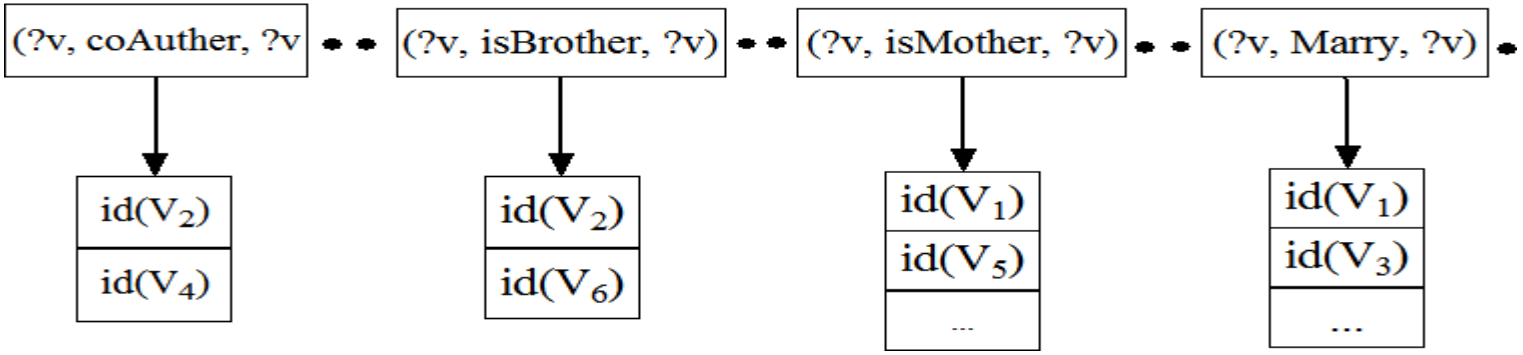


(?c, isMother, ?d)

(?c, Marry, Bob)

(?d, isBrother, ?e)

Rewriting using Sortable Views



(?c, isMother, ?d)

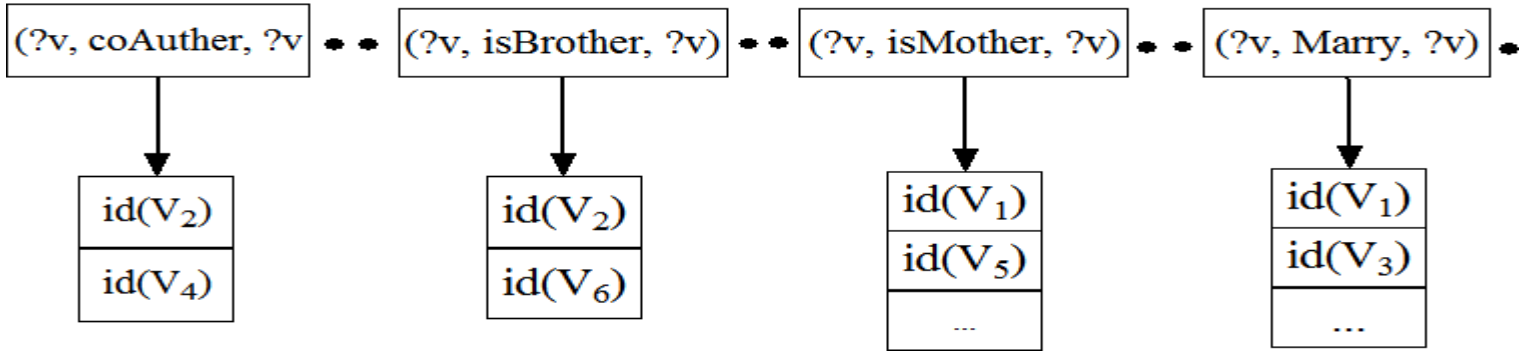
(?c, Marry, Bob)

(?d, isBrother, ?e)

Expand each triple pattern by replacing constants with variables ?v with the invariant:

Each triple pattern must contain at least one constant.

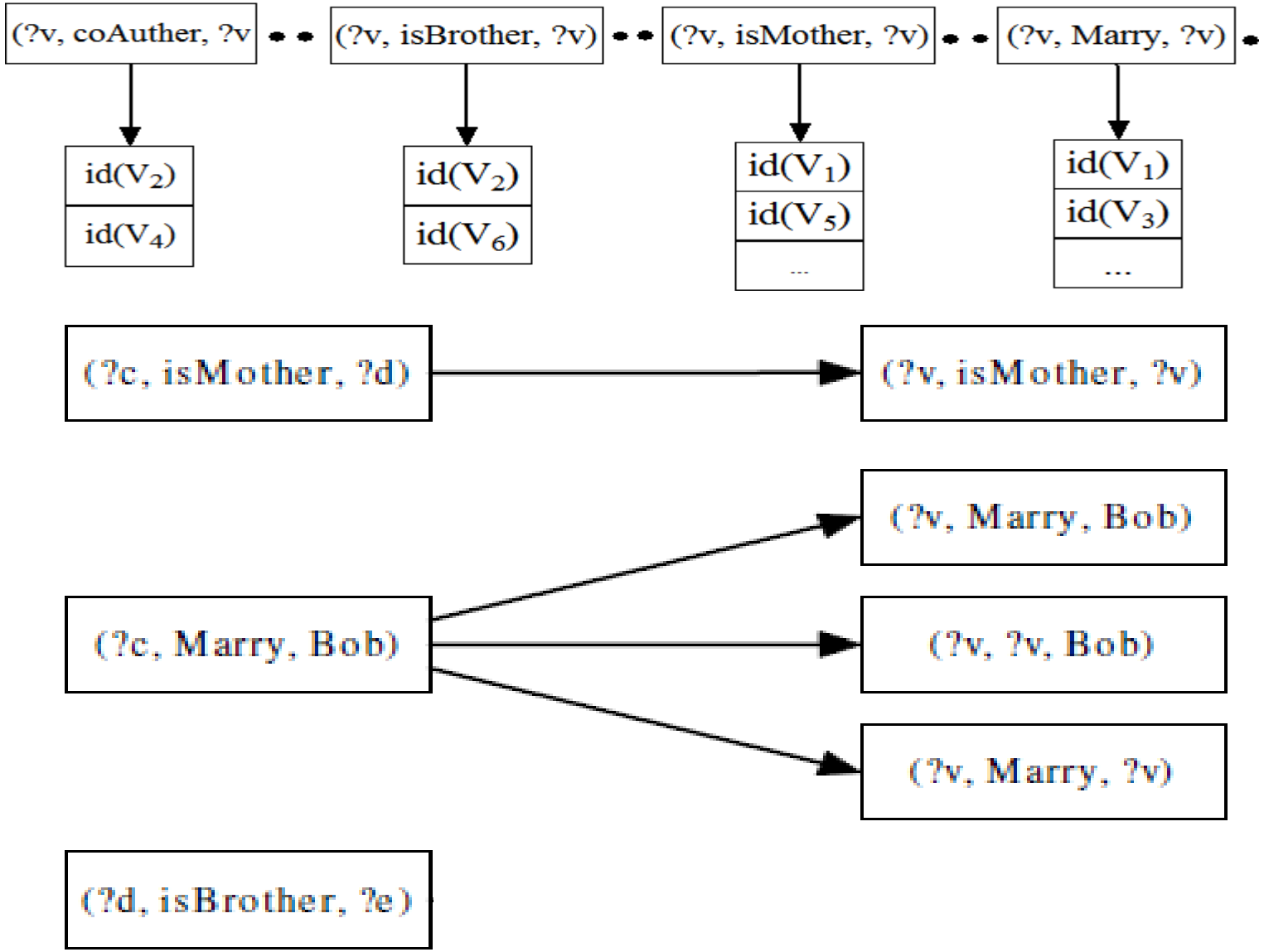
Rewriting using Sortable Views



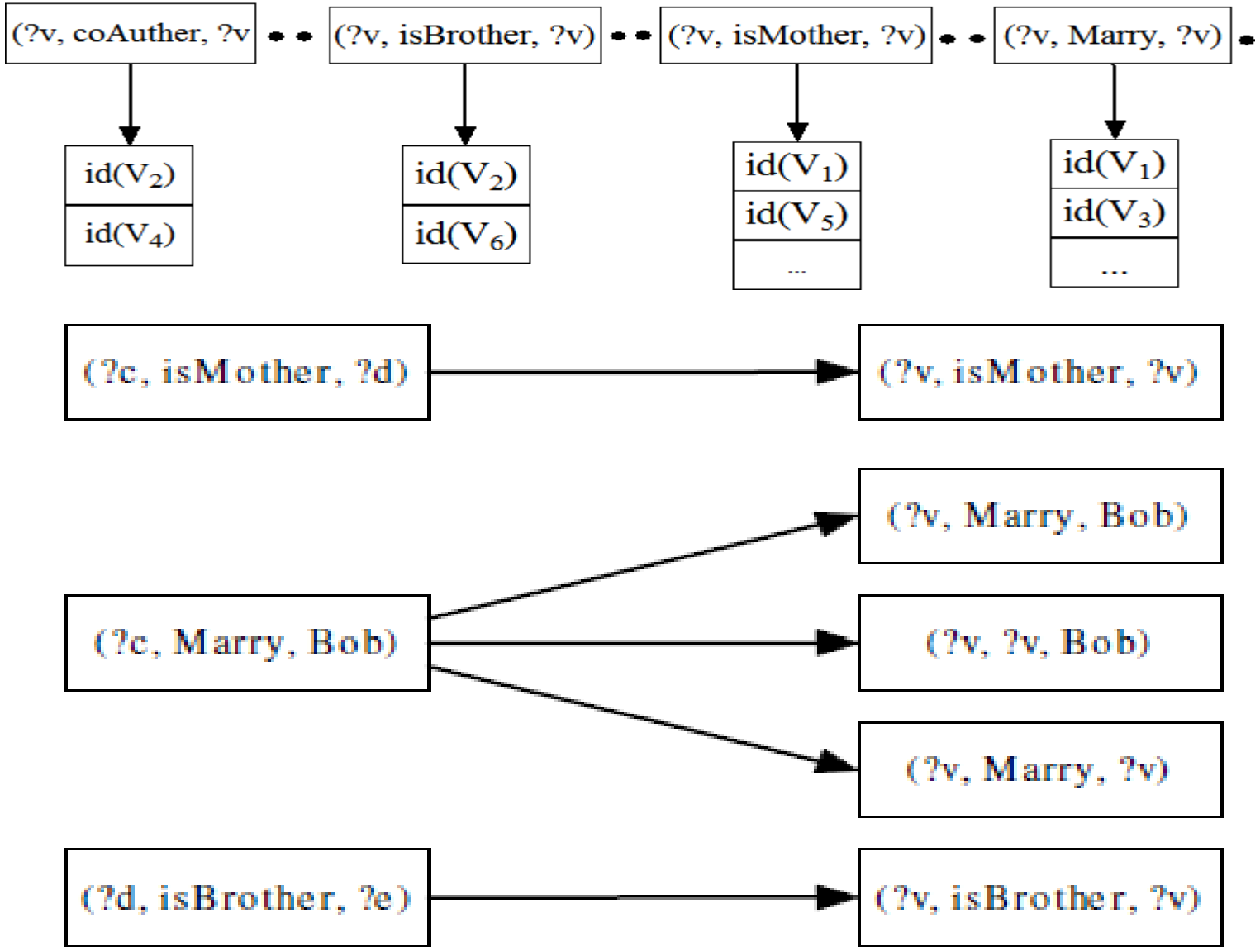
(?c, Marry, Bob)

(?d, isBrother, ?e)

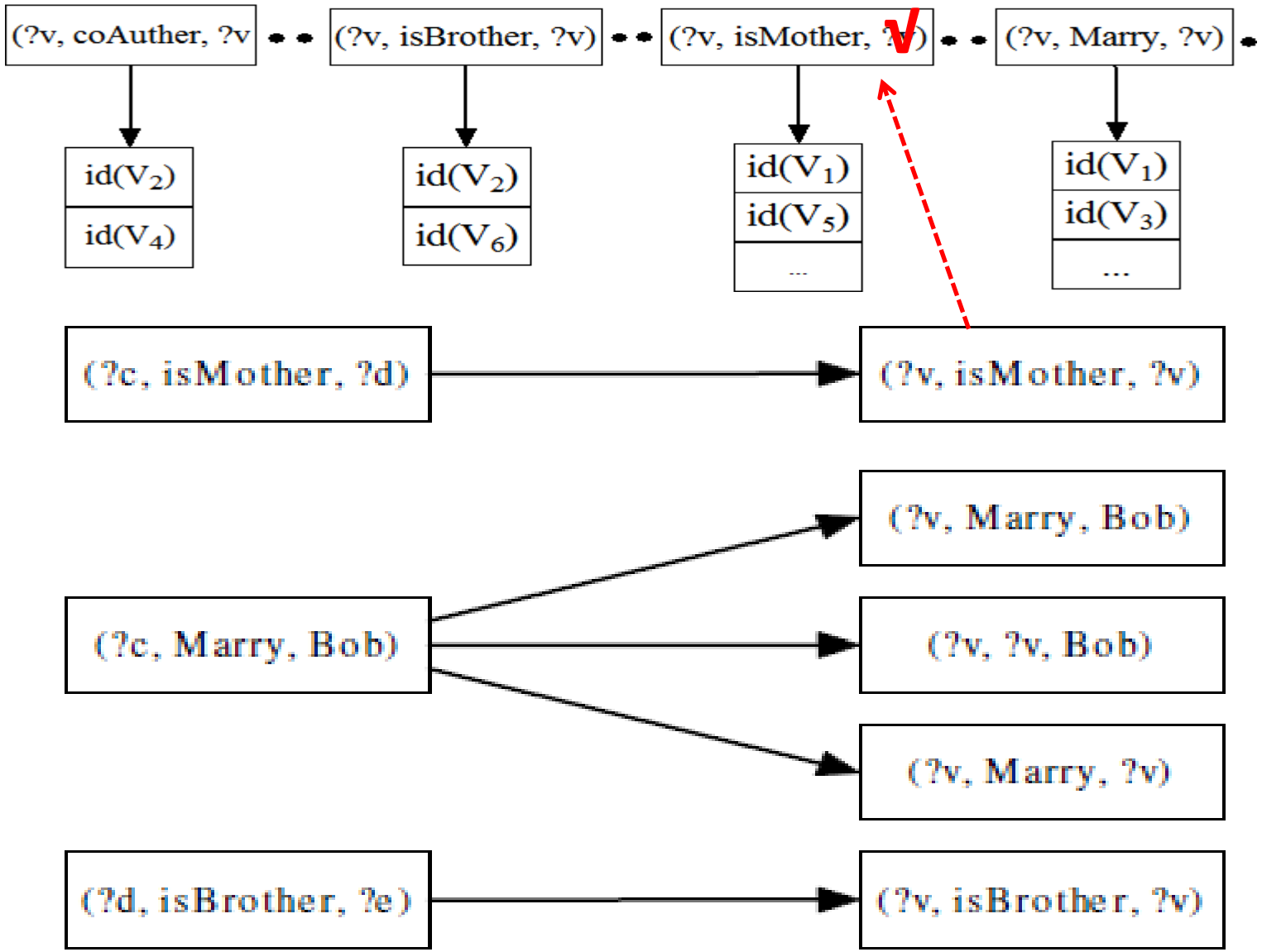
Rewriting using Sortable Views



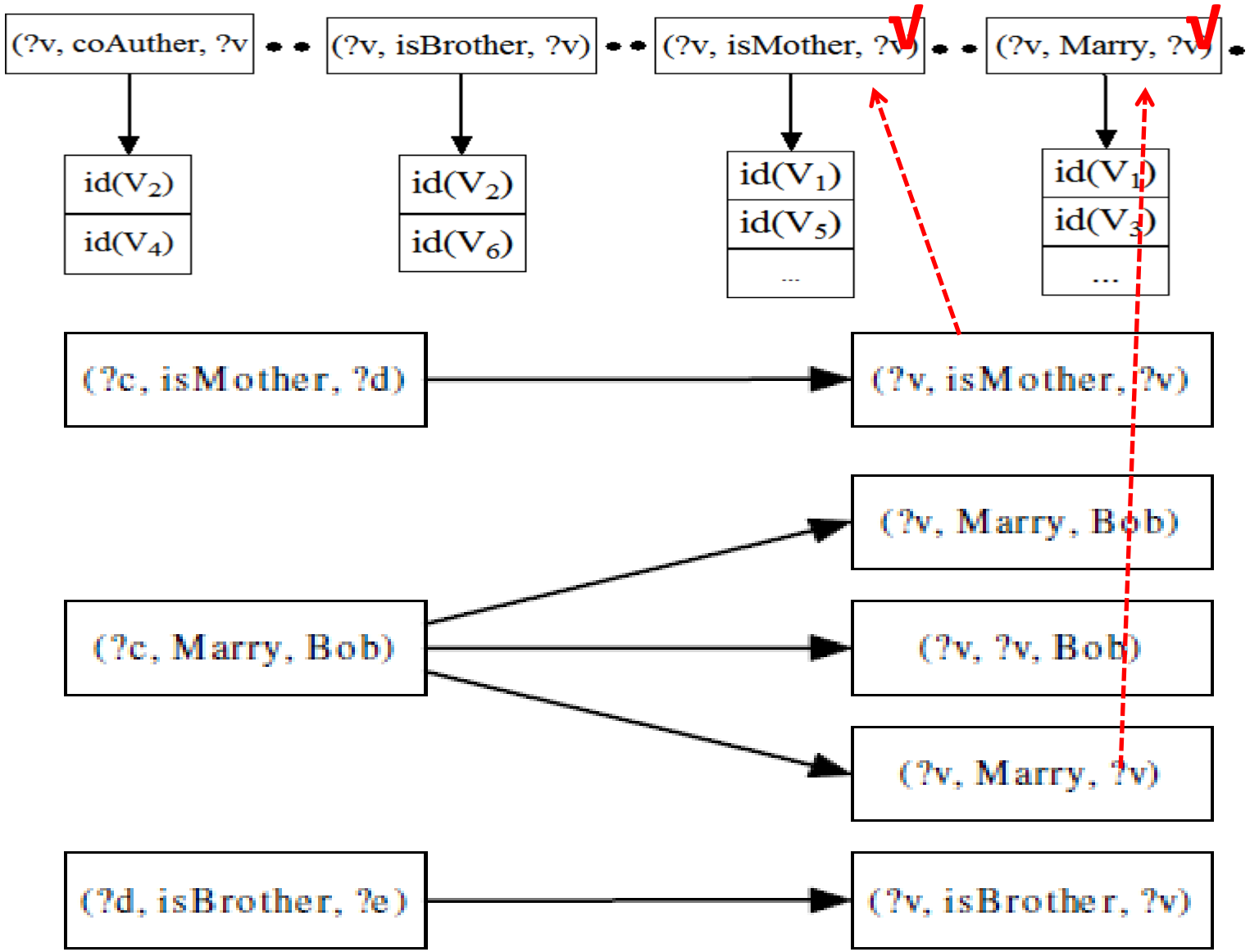
Rewriting using Sortable Views



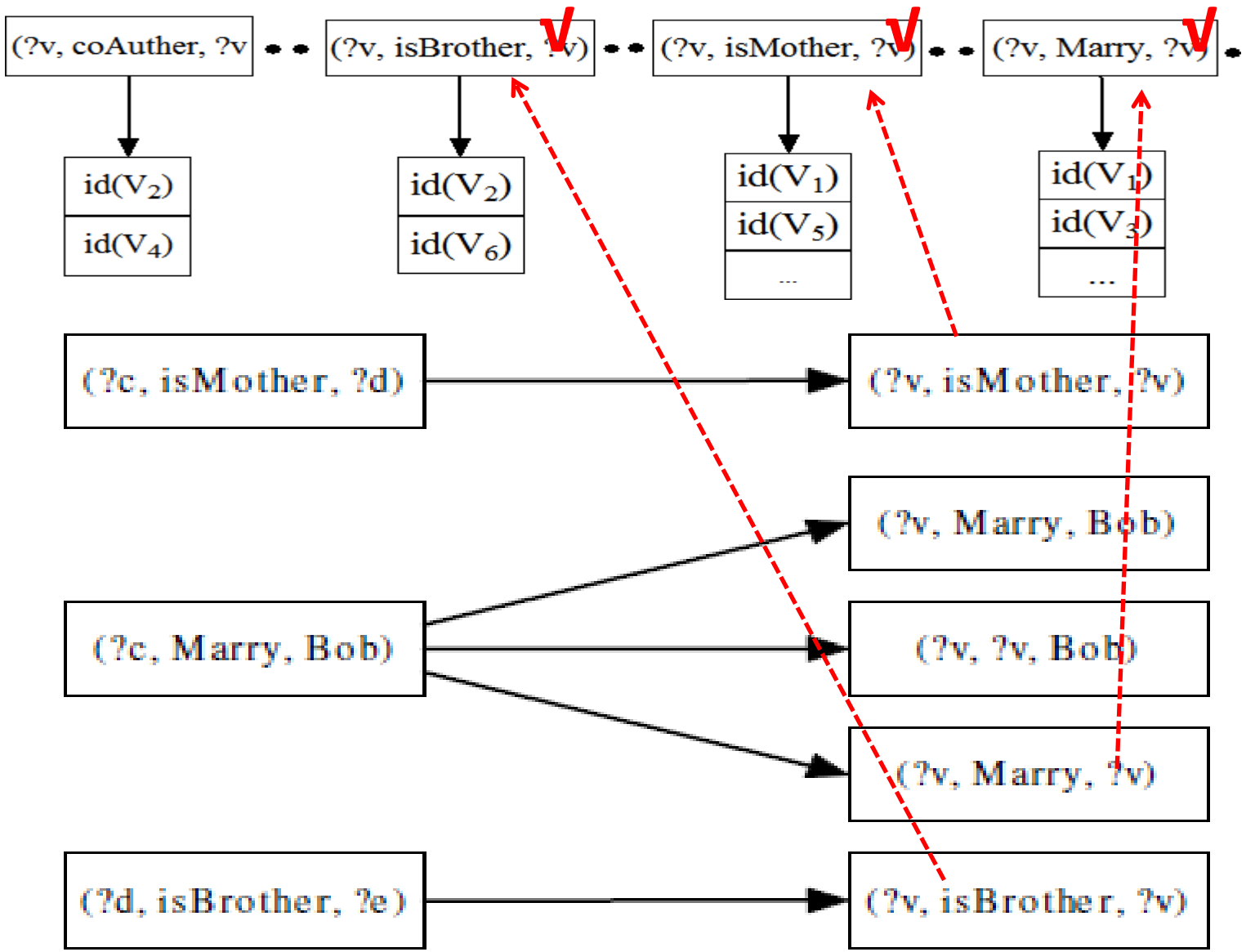
Rewriting using Sortable Views



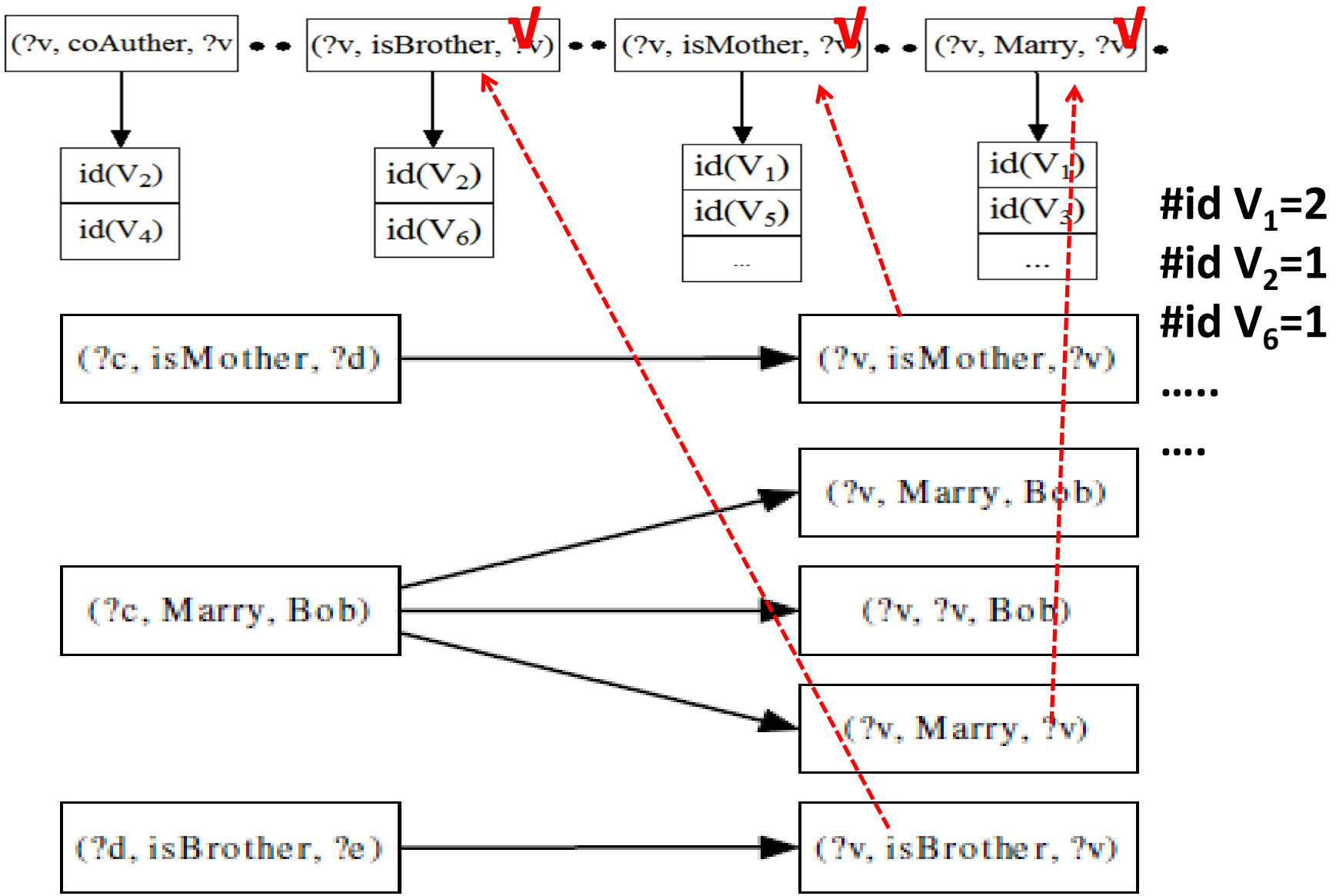
Rewriting using Sortable Views



Rewriting using Sortable Views



Rewriting using Sortable Views



Rewriting using Sortable Views

THEOREM 4. *Given sortable view in base* Γ, *for any* V_i *in* Γ, $P(V_i, Q) = V_i$ *if and only if* $\#id(V_i) = |V_i|$.

#id V_1=2
#id V_2=1
#id V_6=1
.....
....

Rewriting using Sortable Views

THEOREM 4. *Given sortable view in base* Γ, *for any* V_i *in* Γ, $P(V_i, Q) = V_i$ *if and only if* $\#\mathsf{id}(V_i) = |V_i|$.

Algorithm 4 $\mathbf{rewrite}^+(Q, \Gamma)$

1: **for** each $q \in Q$ **do**
2: expand q;
3: mark the entry equal to one of the expansions;
4: **end for**
5: count $\#\mathsf{id}(V_i)$ among the marked lists;
6: **for** each $\#\mathsf{id}(V_i) = |V_i|$ **do**
7: invoke $\mathsf{contain}^+(V_i, Q)$ for further verification;
8: insert $\phi_i(V_i)$ into $\Gamma(Q)$ if $V_i \sqsupseteq Q$;
9: **end for**

#id V_1=2
#id V_2=1
#id V_6=1
.....
....

Rewriting using Sortable Views

THEOREM 4. *Given sortable view in base* Γ, *for any* V_i *in* Γ, $P(V_i, Q) = V_i$ *if and only if* $\#\mathsf{id}(V_i) = |V_i|$.

#id V_1=2
#id V_2=1
#id V_6=1
.....
....

Algorithm 4 rewrite$^+(Q, \Gamma)$

1: **for** each $q \in Q$ **do**
2: expand q;
3: mark the entry equal to one of the expansions;
4: **end for**
5: count $\#\mathsf{id}(V_i)$ among the marked lists;
6: **for** each $\#\mathsf{id}(V_i) = |V_i|$ **do**

$Q_2[1]$
$(Frank, coAuthor, ?o')$

$Q_2[2]$
$(?n', isMother, ?o')$

$Q_2[3]$
$(Bob, Marry, ?n')$

$V_2[1]$
$(?p, coAuthor, ?o)$

$V_2[2]$
$(?n, isMother, ?o)$

$V_2[3]$
$(?m, Marry, ?n)$

Rewriting using Sortable Views

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Q

Rewriting using Sortable Views

Reduced to set cover problem with minimum cost

Q

Outline

- RDF Pattern Rewriting using Views
- Sortable View
- Rewriting using Sortable Views
- **Evaluation**

Evaluation

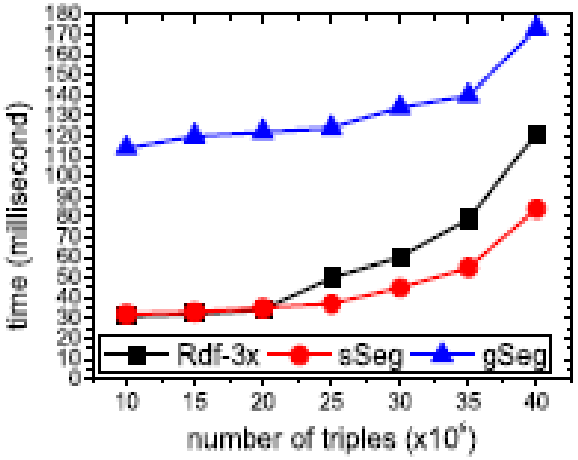
para.	description	default
$\mathcal{L}$	average length of patterns	10
$\mathcal{V}$	average number of variables	5
M	mode of triple generation	D
V	mode of variable generation	D

Evaluation

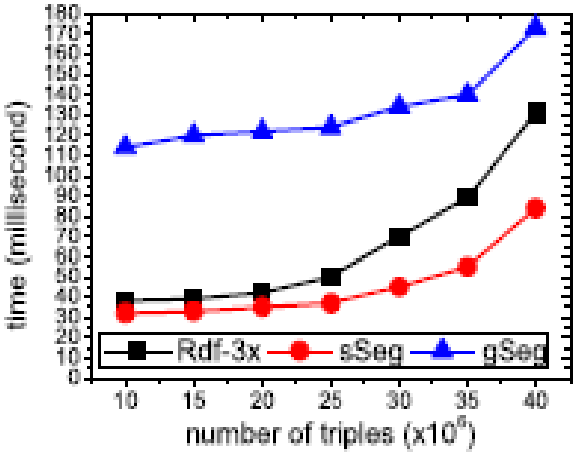
para.	description	default
$\mathcal{L}$	average length of patterns	10
$\mathcal{V}$	average number of variables	5
M	mode of triple generation	D
V	mode of variable generation	D

sSeg
gSeg
Rdf-3x

Evaluation



(a) $L = 11$, M=D and V=D



(b) $L = 11$, M=F and V=F

Figure 5: Scalability tests on two query workloads

Evaluation

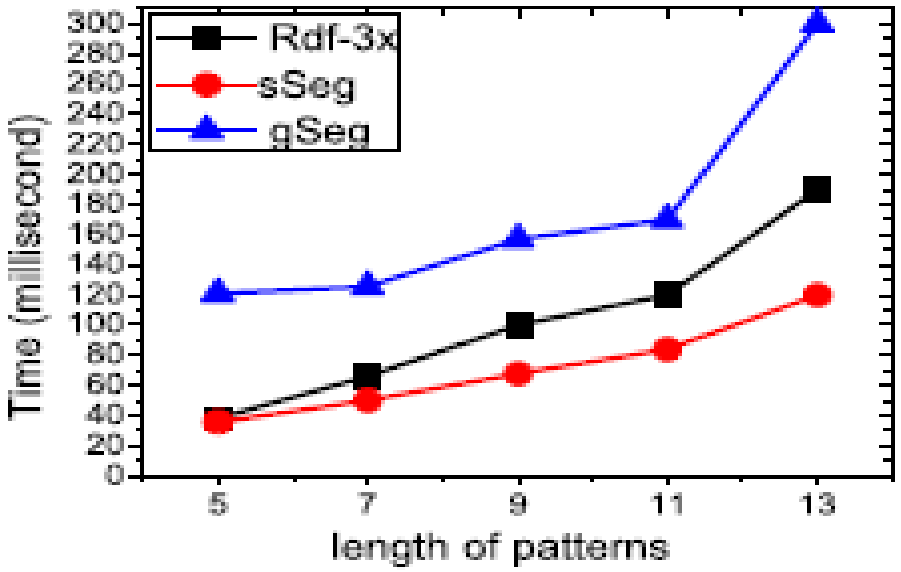


Figure 8: Query length vs. time

Evaluation

$\mathcal{L}$ on Γ_g	$\mathcal{L}$ on Γ_s	C_{Γ_s}	sSeg	gSeg
3	3	30%	1.1	110.6
5	4	33%	2.1	121
7	6	37%	3.4	204
9	7.5	41%	3.6	417

Table 4: Average length L of patterns in Γ_g vs. Time of pattern segmentation (milliseconds, $C_{\Gamma_g} = 30\%$)

Outline

- RDF Pattern Rewriting using Views
- Sortable View
- Rewriting using Sortable Views
- Evaluation

Conclusion

- **RDF Pattern Rewriting using Sortable Views**
  - Simplify containment mapping
  - Using inverted index structure to speed up rewriting
- **Optimum rewriting is also considered**
- **Evaluation**

Thanks!

Q&A