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

School of Computer Science and Engineering, Southeast University, China
RDF Pattern Rewriting using View

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Marry</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>Chris</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Bob</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Frank</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>David</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
RDF Pattern Rewriting using Views

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Marry</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>Chris</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Bob</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Frank</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>David</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Fathers and sons who are coauthored with each other?

School of Computer Science and Engineering, Southeast University, China
RDF Pattern Rewriting using View

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Marry</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>Chris</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Bob</td>
</tr>
<tr>
<td>Chris</td>
<td>coAuthor</td>
<td>Frank</td>
</tr>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>David</td>
</tr>
</tbody>
</table>

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)
RDF Pattern Rewriting using View $V_1$

$(?x, \text{Marry}, ?y), (\_x, \text{isMother}, \_z), (\_y, \text{coAuthor}, \_z)$
RDF Pattern Rewriting using View

View $V_1$

Husbands and their wives

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)
RDF Pattern Rewriting using View

View $V_1$

- Husbands and their wives: $(?x, \text{Marry}, ?y)$
- Mothers and their sons: $(?x, \text{isMother}, ?z)$

$(?x, \text{Marry}, ?y), (?x, \text{isMother}, ?z), (?y, \text{coAuthor}, ?z)$
RDF Pattern Rewriting using View $V_1$

$(?x, \text{Marry}, ?y), (?x, \text{isMother}, ?z), (?y, \text{coAuthor}, ?z)$
RDF Pattern Rewriting using View

View $V_1$

- Husbands and their wives: $(?x, \text{Marry}, ?y)$
- Mothers and their sons: $(?x, \text{isMother}, ?z)$

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

Coauthors: $(?z, \text{coAuthor}, ?y)$

$(?x, \text{Marry}, ?y), (?x, \text{isMother}, ?z), (?y, \text{coAuthor}, ?z)$
RDF Pattern Rewriting using View 

(?x, Marry, ?y), (?x, isMother, ?z), (?y, coAuthor, ?z)
RDF Pattern Rewriting using View 1

View $V_1$

$(?x, \text{Marry}, ?y), (\text{}?x, \text{isMother}, ?z), (\text{}?y, \text{coAuthor}, ?z)$
RDF Pattern Rewriting using View
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 $V_1$

David’s brothers who are coauthored with their fathers?
RDF Pattern Rewriting using View $V_1$

David's brothers who are coauthored with their fathers?
RDF Pattern Rewriting using View $V_1$

David’s brothers who are coauthored with their fathers?
RDF Pattern Rewriting using View

View $V_1$

<table>
<thead>
<tr>
<th>V1</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
RDF Pattern Rewriting using View [V1]

View [V1]

<table>
<thead>
<tr>
<th>[V1]</th>
<th>?x</th>
<th>?y</th>
<th>?z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[Q’1]</th>
<th>?a</th>
<th>?c</th>
<th>?b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
</tbody>
</table>

School of Computer Science and Engineering, Southeast University, China
RDF Pattern Rewriting using View 

View $V_1$

Pattern $Q_1$

<table>
<thead>
<tr>
<th>$[V1]$</th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$[Q’1]$</th>
<th>$a$</th>
<th>$c$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
RDF Pattern Rewriting using Views

<table>
<thead>
<tr>
<th>[V1]</th>
<th>?x</th>
<th>?y</th>
<th>?z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q’1</th>
<th>?a</th>
<th>?c</th>
<th>?b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
RDF Pattern Rewriting using View

\[
[V1] \begin{array}{ccc}
?a & isMother & David \\
Alice & isMother & David \\
... & ... & ...
\end{array}
\]

\[
Q'1 \begin{array}{ccc}
?a & ?c & ?b \\
Alice & Bob & Chris \\
... & ... & ...
\end{array}
\]
RDF Pattern Rewriting using Views

<table>
<thead>
<tr>
<th>?a</th>
<th>isMother</th>
<th>David</th>
<th>?c</th>
<th>?b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>isMother</td>
<td>David</td>
<td>Bob</td>
<td>Chris</td>
</tr>
</tbody>
</table>

Join on ?a

[V1] | ?x | ?y | ?z |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Q’1 | ?a | ?c | ?b |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>Chris</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
RDF Pattern Rewriting using View $V_1$

$\Phi = \{ ?x \rightarrow ?a, ?y \rightarrow ?c, ?z \rightarrow ?b \}$
RDF Pattern Rewriting using Views

\[ \Phi = \{ ?x \rightarrow ?a, \ ?y \rightarrow ?c, \ ?z \rightarrow ?b \} \]

Containment mapping \( \Phi \)
RDF Pattern Rewriting using View

\[ \Phi = \{ ?x \rightarrow ?a, \ ?y \rightarrow ?c, \ ?z \rightarrow ?b \} \]

\[ Q_1 = \Phi(V_1) \text{ union } \{(a, \text{isMother}, \text{David})\} \]
RDF Pattern Rewriting using Views

\[ \Phi = \{ \text{?.x} \rightarrow \text{?.a}, \text{?.y} \rightarrow \text{?.c}, \text{?.z} \rightarrow \text{?.b} \} \]

\[ Q_1 = \Phi(V_1) \cup \{ (\text{?.a, isMother, David}) \} \]

\[ [Q_1] = \Phi([V_1]) \times \{ (\text{?.a, isMother, David}) \} \]
RDF Pattern Rewriting using View
RDF Pattern Rewriting using View
RDF Pattern Rewriting using View
RDF Pattern Rewriting using View
RDF Pattern Rewriting using Views

\[ Q - \Gamma(Q) \]

\[ \Gamma(Q) = \phi(V_1) \cup \phi(V_2) \]
RDF Pattern Rewriting using Views

\[ Q - \Gamma(Q) \]

\[ \Gamma(Q) = \phi(V_1) \cup \phi(V_2) \cup \phi(V_3) \]
RDF Pattern Rewriting using Views

\[ Q - \Gamma(Q) \]

\[ \Gamma(Q) = \phi(V_1) \cup \phi(V_2) \cup \phi(V_3) \]

\[ [Q] = [\phi_1(V_1)] \times [\phi_2(V_2)] \times [\phi_3(V_3)] \times [Q - \Gamma(Q)] \]
RDF Pattern Rewriting using Views

\[ Q - \Gamma(Q) \]

\[ \Gamma(Q) = \phi(V_1) \cup \phi(V_2) \cup \phi(V_3) \]

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

NP-hard to find these views \( V_i \) with their \( \phi_i \)!!
Outline

• RDF Pattern Rewriting using Views

• Sortable View

• Rewriting using Sortable Views

• Evaluation

School of Computer Science and Engineering, Southeast University, China
Sortable View

- The problem of rewriting pattern using views is \textbf{NP-hard} in general cases

- \textbf{Sortable View}
Sortable View

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

\[ V_2[1] \quad (\text{?}p, \text{coAuthor, ?o}) \]

\[ V_2[2] \quad (\text{?n, isMother, ?o}) \]

\[ V_2[3] \quad (\text{?m, Marry, ?n}) \]

School of Computer Science and Engineering, Southeast University, China
Sortable View

\[ V_2[1] \quad (?p, coAuthor, ?o) \quad \Rightarrow \quad V_2[2] \quad (?n, isMother, ?o) \quad \Rightarrow \quad V_2[3] \quad (?m, Marry, ?n) \]
Sortable View

1. All variables are equivalent

Sortable View

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

\[
\begin{align*}
V_2[1] &= (?p, coAuthor, ?o) \\
V_2[2] &= (?n, isMother, ?o) \\
V_2[3] &= (?m, Marry, ?n)
\end{align*}
\]
Sortable View

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

\[
V_2[1] \overset{?p, coAuthor, ?o}{\sim} V_2[2] \overset{?n, isMother, ?o}{\sim} V_2[3] \overset{?m, Marry, ?n}{\sim}
\]
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

\[
\]

\[
(?p, coAuthor, ?o) \rightarrow (?n, isMother, ?o) \rightarrow (?m, Marry, ?n)
\]
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

\[ V_2[1]: (?p, coAuthor, ?o) \lessdot V_2[2]: (?n, isMother, ?o) \lessdot V_2[3]: (?m, Marry, ?n) \]
Sortable View

\[(Frank, \text{coAuthor}, ?o') \prec (?n', \text{isMother}, ?o') \prec (Bob, \text{Marry}, ?n')\]

\[V_2[1] \quad ?p, \text{coAuthor}, ?o\]  \[V_2[2] \quad ?n, \text{isMother}, ?o\]  \[V_2[3] \quad ?m, \text{Marry}, ?n\]
Sortable View

(Frank, coAuthor, ?o') ≺ (?n', isMother, ?o') ≺ (Bob, Marry, ?n')

or

(?n', isMother, ?o') ≺ (Bob, Marry, ?n') ≺ (Frank, coAuthor, ?o')

\[ V_2[1] \quad \quad \quad V_2[2] \quad \quad \quad V_2[3] \]

(?p, coAuthor, ?o) \quad \quad (\?n, isMother, ?o) \quad \quad (?m, Marry, ?n)
Sortable View

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

$q_2[1]$
$(Frank, coAuthor, ?o')$

$q_2[2]$
$(?n', isMother, ?o')$

$q_2[3]$
$(Bob, Marry, ?n')$

$v_2[1]$
$(?p, coAuthor, ?o)$

$v_2[2]$
$(?n, isMother, ?o)$

$v_2[3]$
$(?m, Marry, ?n)$
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 $Q_2[i]$. 

$Q_2[1]$
(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)
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 $Q_2[i]$.

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

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

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

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

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

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

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

\[
\begin{align*}
Q_2[1] & \quad (\text{Frank, coAuthor, } ?o') \\
Q_2[2] & \quad (?n', \text{isMother, } ?o') \\
Q_2[3] & \quad (\text{Bob, Marry, } ?n') \\
V_2[1] & \quad (?p, \text{coAuthor, } ?o) \\
V_2[2] & \quad (?n, \text{isMother, } ?o) \\
V_2[3] & \quad (?m, \text{Marry, } ?n)
\end{align*}
\]
Sortable View

**Theorem 2.** Given pattern $Q$ against sortable view $V$, $V \sqsupseteq Q$ if and only if there is a containment mapping $\phi$ 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 $\text{order}(V)$.

$(?x, \text{isMother}, ?z)$  $(?y, \text{coAuthor}, ?z)$

$(?x, \text{Marry}, ?y)$
Sortable View

**THEOREM 2.** Given pattern $Q$ against sortable view $V$, $V \supseteq Q$ if and only if there is a containment mapping $\phi$ 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 $\text{order}(V)$.

$(?x, \text{isMother}, ?z) \quad (?)y, \text{coAuthor}, ?z)$

$(?x, \text{Marry}, ?y)$
Sortable View

**Theorem 2.** Given pattern \( Q \) against sortable view \( V \), \( V \models Q \) if and only if there is a containment mapping \( \phi \) 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 \( \text{order}(V) \).

\[
(?x, \text{isMother}, ?z) \quad (\quad) \quad (\quad)
\]

\[
(?x, \text{Marry}, ?y) \quad (\quad) \quad (\quad)
\]

\[
(?y, \text{coAuthor}, ?z) < (?x, \text{isMother}, ?z) < (?x, \text{Marry}, ?y)
\]
Sortable View

**Theorem 2.** Given pattern $Q$ against sortable view $V$, $V \sqsubseteq Q$ if and only if there is a containment mapping $\phi$ 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) \rightarrow (?y, coAuthor, ?z) \rightarrow (?x, Marry, ?y)

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

**Theorem 2.** Given pattern $Q$ against sortable view $V$, $V \supseteq Q$ if and only if there is a containment mapping $\phi$ 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 $\text{order}(V)$.
Sortable View

THEOREM 2. Given pattern $Q$ against sortable view $V$, $V \supseteq Q$ if and only if there is a containment mapping $\phi$ 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 $\text{order}(V)$.
Outline

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

(\texttt{v}, \texttt{coAuthor}, \texttt{v})

id(\texttt{V}_2)

id(\texttt{V}_4)

(\texttt{v}, \texttt{isBrother}, \texttt{v})

id(\texttt{V}_2)

id(\texttt{V}_6)

(\texttt{v}, \texttt{isMother}, \texttt{v})

id(\texttt{V}_1)

id(\texttt{V}_5)

id(\texttt{V}_3)

(\texttt{v}, \texttt{Marry}, \texttt{v})

...
Rewriting using Sortable Views

(?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

id(V₂) → id(V₂) → id(V₁) → id(V₁)

id(V₄) → id(V₆) → id(V₅) → id(V₃)

(?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

(?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

id(V_2)
id(V_4)

id(V_2)
id(V_6)

id(V_1)
id(V_5)
...

id(V_1)
id(V_3)
...

(?c, isMother, ?d) → (?v, isMother, ?v)

(?c, Marry, Bob)

(?d, isBrother, ?e)
Rewriting using Sortable Views

(?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

id(V₂)

id(V₄)

id(V₂)

id(V₆)

id(V₁)

id(V₅)

...

id(V₁)

id(V₃)

...

(?c, isMother, ?d) → (?v, isMother, ?v)

(?c, Marry, Bob) → (?v, Marry, Bob)

(?v, ?v, Bob)

(?v, Marry, ?v)

(?d, isBrother, ?e)
Rewriting using Sortable Views

Diagram showing relationships and identifiers involving predicates such as `coAuthor`, `isBrother`, `isMother`, and `Marry`. Node labels include `id(V_2)`, `id(V_4)`, `id(V_1)`, `id(V_6)`, and `id(V_5)`. Edges illustrate the flow of data and relationships between variables and identifiers.
Rewriting using Sortable Views
Rewriting using Sortable Views

(?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

id(V_2) id(V_2) id(V_1) id(V_4) id(V_2) id(V_6) id(V_1) id(V_5)...

(?c, isMother, ?d) → (?v, isMother, ?v) → (?v, Marry, Bob) → (?v, Marry, ?v)

(?c, Marry, Bob) → (?v, ?v, Bob) → (?v, Marry, ?v)

(?d, isBrother, ?e) → (?v, isBrother, ?v)
Rewriting using Sortable Views

- (?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

- (?c, isMother, ?d)
- (?c, Marry, Bob)
- (?d, isBrother, ?e)
Rewriting using Sortable Views

(?v, coAuthor, ?v) → (?v, isBrother, ?v) → (?v, isMother, ?v) → (?v, Marry, ?v)

id(V_2)  id(V_2)  id(V_1)  id(V_4)

id(V_4)  id(V_6)  id(V_5)  ...

(?c, isMother, ?d) → (?v, isMother, ?v)

(?c, Marry, Bob) → (?v, Marry, Bob)

(?d, isBrother, ?e) → (?v, isBrother, ?v)

#id V_1 = 2
#id V_2 = 1
#id V_6 = 1
Rewriting using Sortable Views

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

$id V_1 = 2$
$id V_2 = 1$
$id V_6 = 1$

.....

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views

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

---

**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 $\#id(V_i)$ among the marked lists;
6. for each $\#id(V_i) = |V_i|$ do
7. invoke contain$^+(V_i, Q)$ for further verification;
8. insert $\phi_i(V_i)$ into $\Gamma(Q)$ if $V_i \supseteq Q$;
9. end for
THEOREM 4. Given sortable view in base $\Gamma$, for any $V_i$ in $\Gamma$, $P(V_i, Q) = V_i$ if and only if $\#id(V_i) = |V_i|$. 

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 $\#id(V_i)$ among the marked lists;
6: for each $\#id(V_i) = |V_i|$ 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)

$id V_1 = 2$
$id V_2 = 1$
$id V_6 = 1$
Rewriting using Sortable Views

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views

$V_1$  $V_2$  $V_3$  $V_4$

Q
Rewriting using Sortable Views

\[ V_1, V_2, V_3, V_4 \]

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views

\[ V_1 \quad V_2 \quad V_3 \quad V_4 \]

Q

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views

V₁ V₂ V₃ V₄

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views

School of Computer Science and Engineering, Southeast University, China
Rewriting using Sortable Views
Rewriting using Sortable Views

\[ V_1 \quad V_2 \quad V_3 \quad V_4 \]

Q

School of Computer Science and Engineering, Southeast University, China
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

School of Computer Science and Engineering, Southeast University, China
## Evaluation

<table>
<thead>
<tr>
<th>para.</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{L} )</td>
<td>average length of patterns</td>
<td>10</td>
</tr>
<tr>
<td>( \mathcal{V} )</td>
<td>average number of variables</td>
<td>5</td>
</tr>
<tr>
<td>( M )</td>
<td>mode of triple generation</td>
<td>D</td>
</tr>
<tr>
<td>( V )</td>
<td>mode of variable generation</td>
<td>D</td>
</tr>
</tbody>
</table>
Evaluation

<table>
<thead>
<tr>
<th>para.</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{L}$</td>
<td>average length of patterns</td>
<td>10</td>
</tr>
<tr>
<td>$\mathcal{V}$</td>
<td>average number of variables</td>
<td>5</td>
</tr>
<tr>
<td>$\mathcal{M}$</td>
<td>mode of triple generation</td>
<td>D</td>
</tr>
<tr>
<td>$\mathcal{V}$</td>
<td>mode of variable generation</td>
<td>D</td>
</tr>
</tbody>
</table>

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

School of Computer Science and Engineering, Southeast University, China
Evaluation

Figure 8: Query length vs. time
Evaluation

Table 4: Average length $L$ of patterns in $\Gamma_g$ 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