New Rewritings and Optimizations for Regular-path queries

Gosta Grahne and Alex Thomo
Concordia University

Databases and the Web

- **Databases** and the **Web** are interconnected at many levels.
- Web sites are empowered by databases.
- A collection of Web pages are a tempting target for a database.

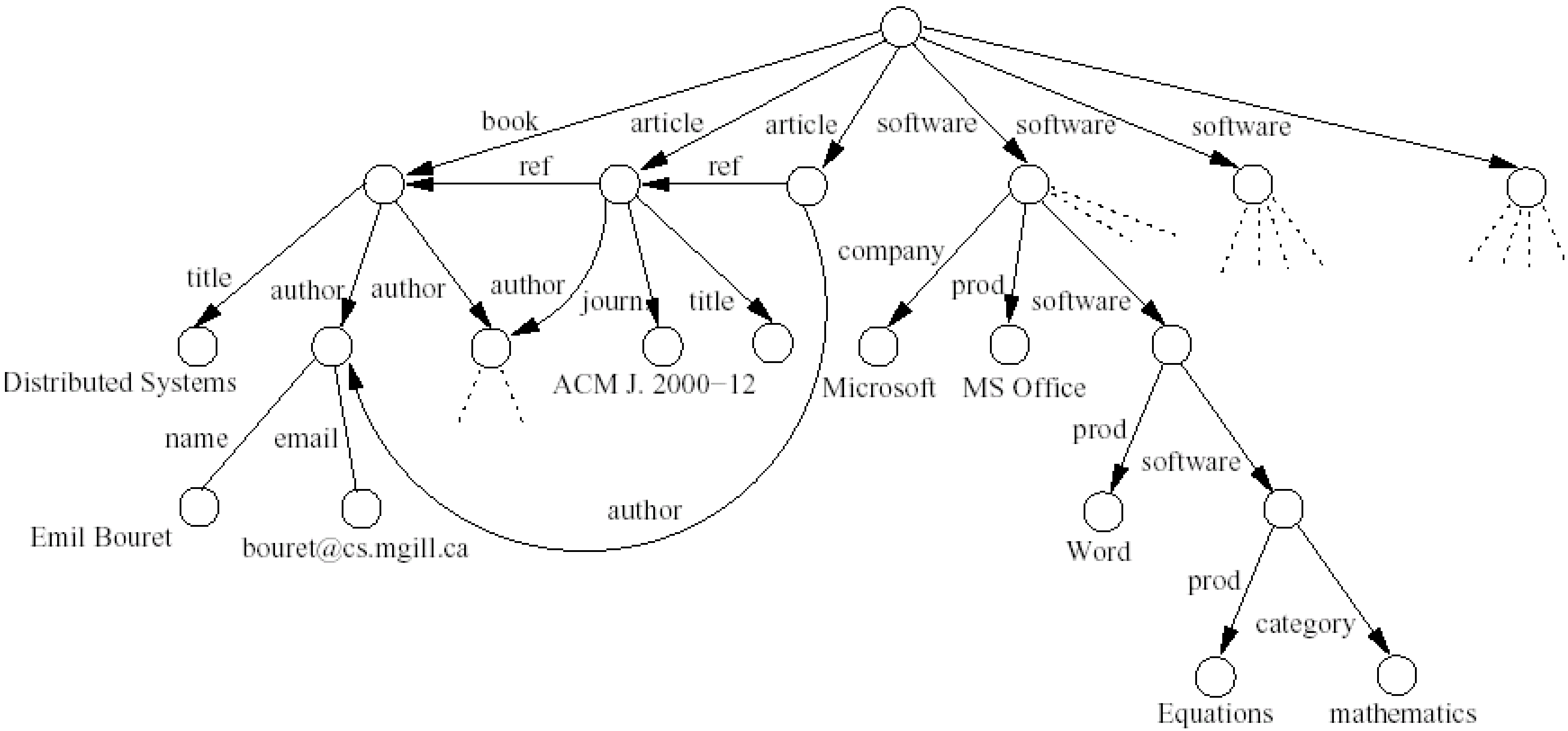
Graph Modeling

- As a result it is natural to model such a database as a **labeled directed graph**.

- The **nodes** of such graph are the Web pages, and the **edges** are the Web links

- Another example of data modeled by graphs is **XML**.
  - The nested structure of **XML** elements and **idref** links along with **xlink**'s are very naturally represented by graphs.

Graph Modeling –Example



Querying Graph Data

- **Desideratum:** To have a mechanism which navigates arbitrarily long paths.

- **Solution:** Recursively querying, through regular path queries,

- **Example:** *ref* . (author+title+journal)* specifies all the **paths** connecting pairs ***(x,y)*** of **related objects**,
  - where ***x*** can be an article and ***y*** can be author, title or journal.

Recursive Querying: The Problem

- The navigation is very expensive.

- It can involve many
  - **physical accesses**,
  - network connections
  - transfers of Web pages.

Optimizing Using Views

- Relevant views can greatly optimize the query evaluation.
- **View = Query + Answer**
- **Cached view:** The answer exists in temporary memory.
- **Materialized view:** The answer is stored in persistent memory.

Views (Cont.)

- Example: We have cached the view:

  V = software

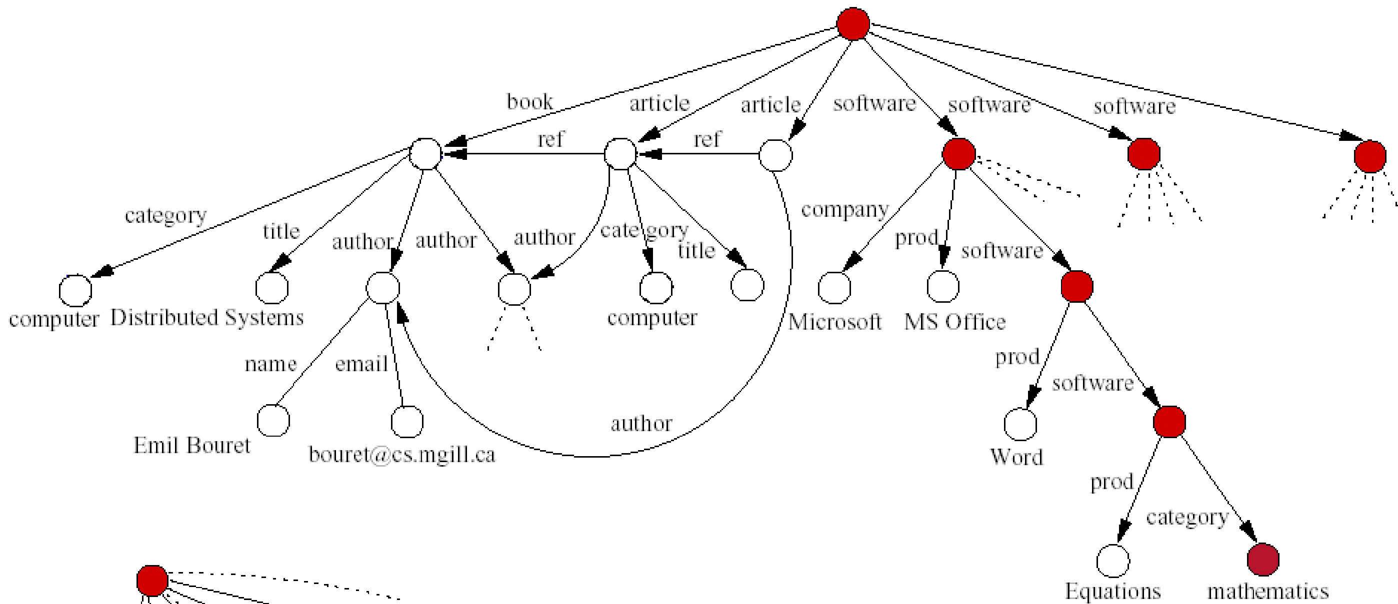
- And, we want to answer the query:
  - ***What categories are addressed by the software packages?***
- It's an exhaustive query expressed by

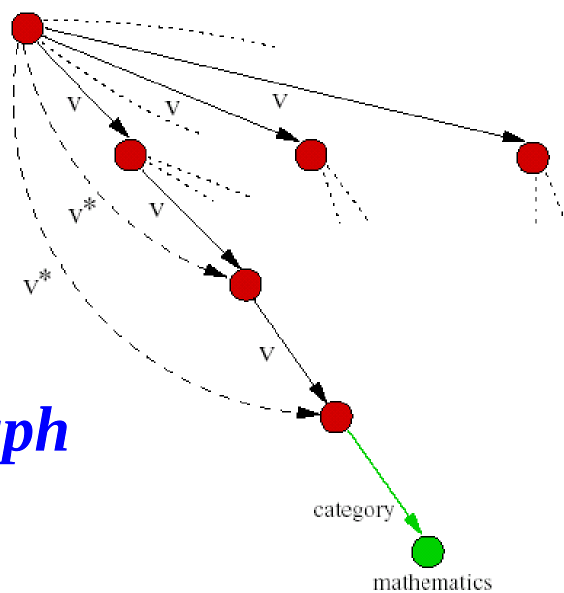
  Q = software*.category

- The cached view can greatly optimize the evaluation of this query if rewritten as

  Q′ = V*.software

Views (Cont.)





View Graph

Red nodes would have been accessed.

Green nodes are actually accessed.

Rewritings – An Example

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \quad V_2 = S+R \quad V_3 = T^5$
- **What could be a rewriting?**

- $Q^{(1)} = v_1 v_2 v_1 + v_3$
- $Q^{(2)} = v_1 T + v_1 v_2 v_1 + v_3$
- $Q^{(3)} = v_3$
- $Q^{(4)} = v_1 T + v_1 S v_1 + v_3 + T^5$
- $Q^{(5)} = v_1 T + v_3$
- $Q^{(6)} = v_1 T + v_1 S v_1 + v_3$

Rewritings ($Q^{(1)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \quad V_2 = S+R \quad V_3 = T^5$
- $\boldsymbol{Q^{(1)} = v_1 v_2 v_1 + v_3}$ *(Grahne & Thomo 2000)*

  - This is a **"rough"** rewriting based on view relevance only
    - whenever there is a query word $\boldsymbol{w}$ such that $\boldsymbol{w} \in \boldsymbol{V_i \ldots V_j}$, replace it with $\boldsymbol{v_i \ldots v_j}$ in the rewriting.

  - It is not **"contained,"** i.e.
    - if we substitute lower-case $\boldsymbol{v}$'s with the corresp. $\boldsymbol{V}$'s, the language we get is not always cont. in $\boldsymbol{Q}$.

Rewritings ($Q^{(2)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \qquad V_2 = S+R \qquad V_3 = T^5$

- $Q^{(2)} = v_1 T + v_1 v_2 v_1 + v_3$ *(Grahne & Thomo 2001)*

  - This is also a **"rough"** rewriting based on view relevance only. *Exhaustively* we do:
    - whenever there is a query **sub**-word w such that $w \in V_i \ldots V_j$, replace it with $v_i \ldots v_j$ in the rewriting.

  - It is not **"contained,"** i.e.
    - if we substitute lower-case v's with the corresp. V's the language we get is not always cont. in Q.

Rewritings ($Q^{(3)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \quad V_2 = S+R \quad V_3 = T^5$

- $\boldsymbol{Q^{(3)} = v_3}$ *(Calvanese, De Giacomo, Lenzerini, Vardi 99)*

  - This is a **“contained”** rewriting
    - whenever there is a query word $\boldsymbol{w}$ such that $\boldsymbol{w} \in \boldsymbol{V_i \ldots V_j}$, and $\boldsymbol{V_i \ldots V_j} \subseteq \boldsymbol{Q}$ replace it with $\boldsymbol{v_i \ldots v_j}$ in the rewriting.

    - Unfortunately, **it is not always exact** (see e.g.)

    - We can **optimize** with it, but we have also to answer on the ***DB*** the difference ***(RS)T + (RS)S(RS)***

Rewritings ($Q^{(4)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \qquad V_2 = S+R \qquad V_3 = T^5$

- $\boldsymbol{Q^{(4)} = v_1 T + v_1 S v_1 + v_3 + T^5}$

  (Calvanese, De Giacomo, Lenzerini, Vardi 99)

  - This is also a **"contained"** rewriting
    - Enrich as needed (for exactness) the view set with elementary one-symbol views and then compute $\boldsymbol{Q^{(3)}}$.
    - Unfortunately, we could get unnecessary words, e.g. $\boldsymbol{T^5}$.

Rewritings ($Q^{(5)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \quad V_2 = S+R \quad V_3 = T^5$
- $Q^{(5)} = v_1 T + v_3$ *(Grahne & Thomo 2001)*

  - This is also a **"contained"** rewriting
    - Compute the max-contained subset of $Q^{(2)}$
    - Recall $Q^{(2)} = v_1 T + v_1 v_2 v_1 + v_3$. The max-contained subset is $Q^{(5)} = v_1 T + v_3$.
    - Unfortunately, we could get **non-exact** rewritings as the example witnesses.
    - So, we need to compute also the diff. ***(RS)S(RS)***.
    - It is bigger than $Q^{(3)}$ and optimal compared to $Q^{(4)}$

Rewritings ($Q^{(6)}$)

- $Q = (RS)T + (RS)S(RS) + T^5$

  $V_1 = RS \qquad V_2 = S+R \qquad V_3 = T^5$
- $\mathbf{Q^{(6)} = v_1 T + v_1 S v_1 + v_3}$

  (Current paper: Grahne & Thomo 2003)

  - **Exact and Optimal** rewriting. *Exhaustively* we do:
    - whenever there is a query **(sub)**-word $\boldsymbol{w}$ such that $\boldsymbol{w} \in \boldsymbol{V_i \ldots V_j}$, and $\boldsymbol{V_i \ldots V_j} \subseteq \boldsymbol{Q}$ replace it with $\boldsymbol{v_i \ldots v_j}$ in the rewriting.

Comments

- $Q^{(1)}$ and $Q^{(2)}$ are "relevance" rewritings.
  - To evaluate the query we need to cache **path histories** for pairs ***(x,y)*** in view extensions.
    - E.g. if a path $v_1v_2v_1$ exists between ***x*** and ***y*** in the view graph we need to know if its annotation is **"_*R*_"** or **"_*S*_"**.

- $Q^{(3)}$ ***if exact*** can be used to answer the query on the view graph only.

  Otherwise, the difference with the query has to be evaluated on the ***DB***.

Comments (Cont.)

- $Q^{(4)}$ and $Q^{(5)}$ try to minimize the query portion that has to be answered on ***DB***, *had we used* $Q^{(3)}$.

- $Q^{(4)}$ can introduce **"non-optimal"** words (recall T^5).

- $Q^{(5)}$ goes **"too far"** by not allowing any sub-word belonging to some view language (recall $v_1 S v_1$), and so can be **non-exact**.

Comments (Cont.)

- As we can see the "best in class" is $Q^{(6)}$.
- There is **no "non-optimal"** word, with respect to database symbols.
- v_1Sv_1 is not really non-optimal, because if we remove it, there is no way to make up the information we loose, by any better combination of views with fewer database symbols.

Formally Comparing Rewritings

- We introduce a partial order for $\Omega\cup\Delta$ languages, where
  - Ω is the view representative symbol alphabet
  - Δ is the database alphabet

- Let $\boldsymbol{V}=\{V_1,\ldots,V_n\}$ be a set of views. Then, $\boldsymbol{Q_1} \leq_{\mathbf{V}} \boldsymbol{Q_2}$ if it is possible to **replace** some occurrences of **view words** in the words of $\boldsymbol{Q_1}$ and obtain $\boldsymbol{Q_2}$ as a result.

Formally Comparing (Cont.)

- Let $\leq_{\mathbf{V},Q}$ be the **restriction** of $\leq_{\mathbf{V}}$ in the set of the **"contained"** rewritings.

- The **"bigger"** a rewriting the **"better"** it is

- Obviously, we are interested in $\leq_{\mathbf{V},Q}$-maximal and exact rewritings.

Formally Comparing - Names

- $Q^{(1)}$–possibility rewriting, **PR**.
  - It is $\leq_V$-maximal which implies that it is also $\leq_{V,Q}$-maximal.
- $Q^{(2)}$–possibility partial rewriting, **PPR**.
  - It is $\leq_V$-maximal which implies that it is also $\leq_{V,Q}$-maximal.
- $Q^{(3)}$–maximally contained rewriting, **MCR**.
  - It is the biggest Ω language, **"contained"** in Q
  - Clearly it is $\leq_{V,Q}$-maximal, since there is no sub-word on Δ (so we cannot replace any)

Names (Cont.)

- $Q^{(4)}$–max. cont. partial rewriting, **MCPR,**
  - It is the biggest $\Omega \cup \Delta_x$ language, **“contained”** in ***Q*.** ($\Delta_x \subseteq \Delta$)
  - It is **not** $\leq_{v,Q}$-maximal as shown by the example: (T^5 can be replaced by v_3)

- $Q^{(5)}$–exhaustive partial rewriting, **EPR**.
  - It is $\leq_{v,Q}$-maximal (as subset of PPR).

- $Q^{(6)}$–maximal partial rewriting **MPR**,
  - It is the union of all $\leq_{v,Q}$-maximal languages, hence it is $\leq_{v,Q}$-maximal
  - We prove: It is also **exact**.

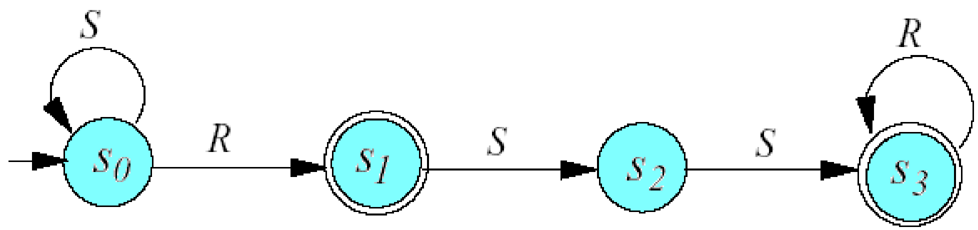
Formally Comparing (Final)

	$\leq_{V,Q}$-maximal	Exactness
PR	Yes	No
PPR	Yes	No
MCR	Yes	No
MCPR	No	Yes
EPR	Yes	No
MPR	Yes	Yes

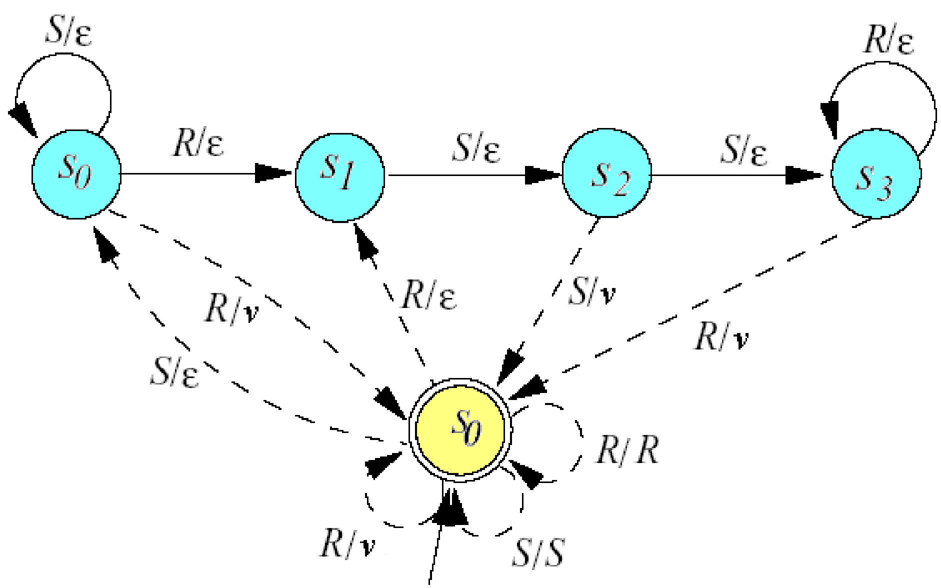
Constructing MPR

- Suppose we have **only one view** V (Ω $=\{v\}$)
- How we can replace **arbitrarily** view words appearing as **sub**-words in some ***L***?

V automaton



*Transducer **T** that replaces arbitrarily occurrences of view words*



*We take **T(L)***

Constructing MPR (Cont.)

- However, in general for a query ***Q***, ***T(Q)*** is **not "contained"** in ***Q***.
  - E.g. ***Q=RSR*** and ***V=S+T***, ***T(Q) = RvR*** "$\not\subset$" ***Q.***

- Instead we compute $(T(Q^c))^c$. ((.)^c complement)
  - As we show it is the **biggest "contained"** arbitrary **replacement** one could achieve.

- However, what we like is an **exhaustive "contained" replacement**.

Constructing MPR (Cont.)

- To achieve our goal we should filter out the words having at least a **sub**-word **eligible** for replacement.

- Formally speaking we need to solve the language equation:
  - Find the biggest languages $\boldsymbol{X}, \boldsymbol{Y} \subseteq \Omega \cup \Delta$

$$\boldsymbol{XVY} \subseteq (\boldsymbol{T}(\boldsymbol{Q}^c))^c$$

- Finally, $\boldsymbol{MPRV(Q)} = (\boldsymbol{T}(\boldsymbol{Q}^c))^c \cap (\boldsymbol{XVY})^c$

Comparing + Complexity

	$\leq_{V,Q}$-maximal	Exactness	Complexity
PR	Yes	No	PTIME
PPR	Yes	No	EXPTIME
MCR	Yes	No	2EXPTIME
MCPR	No	Yes	2EXPTIME
EPR	Yes	No	2EXPTIME
MPR	Yes	Yes	3EXPTIME

All bounds are tight.

Other Contributions

- Presenting an improved algorithm for evaluating **regular path queries** using any **exact** partial rewriting.

- Maximally optimizing **conjunctive path queries** using all the available cached information.

References

- Gösta Grahne, Alex Thomo. New Rewritings and Optimizations for Regular Path Queries. ICDT 2003: 242-258
- Gösta Grahne, Alex Thomo. Algebraic Rewritings for Optimizing Regular Path Queries. ICDT 2001: 301-315
- Gösta Grahne, Alex Thomo. An Optimization Technique for Answering Regular Path Queries. WebDB (Selected Papers) 2000: 215-225
- Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. PODS 1999: 194-204