MiniCount: Efficient Rewriting of COUNT-Queries Using Views

Vasilis Vassalos

Václav Lín and Prodromos Malakasiotis

Athens University of Economics and Business

University of Economics, Prague

Rewriting queries using views

Given query Q over a database schema and a set of views $V_1, V_2, ..., V_n$ over the same schema, is it possible to answer Q using the views $V_1, V_2, ..., V_n$ by rewriting it into a query Q' over the views?

- Total vs partial rewritings
- Equivalent vs maximally contained rewritings
- Languages
  - ➥ For Q
  - ➥ For $V_1, V_2, ..., V_n$
  - ➥ For Q'

Why we care

- Data Integration
  - ➥ Putting together information residing in *autonomous, heterogeneous* information sources expressed as views over a global schema
  - ➥ Capability-based rewriting
  - ➥ Aggregate views: Summary information, statistical information [AAH+01]
- Query optimization
  - ➥ Include materialized views in optimization process
  - ➥ Top-k processing
- Data warehousing and decision support
  - ➥ View selection for a data warehouse
  - ➥ CUBE views

Outline

- Rewriting Conjunctive Queries: Basic theory
- Rewriting Conjunctive Queries: The algorithms
- Conjunctive Count Queries
- Finding CCQ Rewritings
- The MiniCount algorithm
  - ➥ MiniCount Descriptors
- Experiments
- Related work and Conclusions

Rewriting Conjunctive Queries

$$\begin{array}{rcl} q(\overline{x}) & \leftarrow & p_1(\overline{x_1}), \ldots, p_n(\overline{x_n}) \\ v_j(\overline{y_j}) & \leftarrow & body_j, \text{for } j = 1 \ldots m \end{array}$$

Rewriting :

$$r(\overline{x}) \leftarrow \bigwedge_{j \in \{1,\ldots,m\}} v_j$$

Contained rewriting: $r \sqsubseteq_V q$ if, for any database D, $r(D_V) \subseteq q(D_V)$

Maximally contained rewriting: A set of rewritings R s.t. any rewriting is contained in R.

To compare rewritings and queries: **Unfolding**.

r^u: For all v_j appearing in rewriting r

- Unify view head v_j in r with the view definition
- Replace occurence of v_j in r with the body of v_j

Unfolding property: $r^u(D) \equiv r(D_V)$

How to find rewritings

Basic properties of rewritings [LMSS95]:

- Length of rewriting up to n
- No new constants or variables needed

Basic Strategy

- Generate rewritings
  - ➥ Instantiate views in all possible ways
    - ➤ Using query variables
  - ➥ Create rewritings of length up to n using the instantiations in the body and the same query head
- Test rewritings
  - ➥ (Conjunctive) Query containment

We can do better

- Each view must contribute something
  - ➥ Must *cover* at least one subgoal
- Each view must satisfy some conditions
  - ➥ (e.g.) Distinguished variables in the query must be distinguished in the view

Only generate contained rewritings $\Rightarrow$ Avoid containment checks

MiniCon algorithm [PL01] for conjunctive query rewriting

Conjunctive `COUNT` queries (CCQ)

$$q(prof, count) \leftarrow teach(prof, course), visitor(prof)$$

Select	$teach.prof$, `COUNT`($prof$)
From	$teach$, $visitor$
Where	$teach.prof = visitor.prof$
Group By	$prof$

"... give me the number of courses that each visiting professor teaches."

Basic terminology:

- $prof$ – free variable
- $course$ – bound variable
- $teach(prof, course)$ – bound atom
- $visitor(prof)$ – free atom

Maximally Contained Rewritings for CCQ

$$\begin{array}{rcl} q(prof, grade, count) & \leftarrow & study(student, course, grade), teach(prof, course) \\ v_1(p, c, count) & \leftarrow & teach(p, c), fulltime(p) \\ v_2(s, c, g, count) & \leftarrow & study(s, c, g) \\ v_3(p, g, count) & \leftarrow & study(s, c, g), teach(p, c), visitor(p) \end{array}$$

- Language for Q: CCQ
- Language for V_i?
  - ➥ Count queries are sensitive to multiplicities $\Rightarrow$ we use only count views and unaggregated views (evaluated as multisets) for rewritings.
- Language for Q'?
  - ➥ Given a set V of conjunctive count views, Cohen, Nutt & Sagiv (2003) define a class of queries $PUnf(V)$ over V that are usable as rewritings of conjunctive count queries

The Rewritings of Count Queries

Given query q and views v_1, v_2

$$\begin{array}{rcl} q(x, count) & \leftarrow & a(x,z), c(x,u) \\ v_1(x,y,count) & \leftarrow & a(x,z), b(x,y) \\ v_2(x,count) & \leftarrow & c(x,u) \end{array}$$

Consider p over v_1, v_2:

$$\begin{array}{rcl} p(x,y,sum(w_1 \times w_2)) & \leftarrow & v_1(x,y,w_1), v_2(x,w_2) \\ r(x,w) & \leftarrow & p(x,y,w) \ . \end{array}$$

p is equivalent *modulo the views* to its *unfolding*

$$p^u(x,y,count) \leftarrow a(x,z), b(x,y), c(x,u) \ .$$

p^u returns for each value of x the same count as $q \Rightarrow r$ is a rewriting of q. Note that:

- $b(x,y)$ is free in p^u,
- there is a bijection between bound atoms of q and p^u,
- p^u enforces all the joins that q does.

The theory of Count rewritings (Cohen, Nutt & Sagiv, 2003)

- Only consider rewritings whose unfolding is a count query
- Only* the following rewritings (defining set $\mathit{Unf}(V)$) have this property

$$p\Big(\overline{s}, sum\big(\prod_{i=1}^{l} w_i\big)\Big) \leftarrow \bigwedge_{i=1}^{l} v_i(\tau_i \overline{s}_i, w_i)$$

- *and their k-*projections* (defining set $P\mathit{Unf}(V)$)

$$r(\overline{s}', w) \leftarrow p(\overline{s}, w)$$

So a rewriting r is the projection of some query $p \in \mathit{Unf}(V)$.

Using Unaggregated Views

Instead of count views

$$\begin{aligned} v_1(x, y, count) &\leftarrow a(x, z), b(x, y) \\ v_2(x, count) &\leftarrow c(x, u) \end{aligned}$$

we could use unaggregated views

$$\begin{aligned} v'_1(x, y) &\leftarrow a(x, z), b(x, y) \\ v'_2(x) &\leftarrow c(x, u) \end{aligned}$$

evaluated as multisets.
$\Rightarrow$ These queries carry exactly the same information as v_1 and v_2.

Finding rewritings

A k-containment mapping from CCQ q to q'

- maps *the first* k terms in the head of q to the first k terms in the head of q'
- maps bound variables of q 1-1 to bound variables of q'
- maps bound atoms of q onto the bound atoms of q'
- maps all the atoms of q to atoms in q'

A query $r \in PUnf(V)$ (r is the projection of a query p) is a contained rewriting of query q *if and only if* there exists a k-containment mapping between q and unfolding p^u.

Size theorem: Rewritings of queries with n atoms have at most n views and no new constants.

Naive algorithm

- Candidate rewriting generation: For all subsets of the view set V, create all queries in $Unf(V)$ and take all their k-projections.
- Containment Check: Check for the existence of a k-containment mapping between the query and each of the candidate rewritings.
- Correct but intractable.

MiniCount

Our efficient algorithm for answering queries using views with COUNT using characterization by Cohen et al.

Efficiency is achieved by careful bookkeeping

- It never considers a combination of views that does not cover all the atoms of q *or*
- does not enforce all the joins *or*
- performs a k-projection on the wrong head variables *or*
- has "more" bound variables than appropriate

Views Usable in Rewritings

$$\mathbf{q(x, y, count) \leftarrow a(x, z), c(y, u), b(z, y)}$$

$$v_1(x, y, count) \leftarrow a(x, z), b(z, y)$$

View v_1 is *usable* – it covers atoms a and b of q and it enforces the join of a and b.

$$v_2(x, count) \leftarrow a(x, z)$$

View v_2 is *not usable* – variable z is bound and no combination of v_2 with other views will enforce the join of a and b.

$$v_3(y, u) \leftarrow c(y, u), f(y, u)$$

v_3 is *not usable*. Consider rewriting

$$p^u(x, y, \overset{?}{u}, count) \leftarrow \underbrace{c(y, u), f(y, u)}_{v_3}, \underbrace{a(x, z), b(z, y)}_{v_1}$$

Variable u may be bound or free in p^u – in either case, can't map bound atoms of q onto bound atoms of p^u.

MiniCount descriptors

An *MCD (MiniCount Descriptor)* for a query q and a view v_i is a seven-tuple of:

- h_i: a head homomorphism on v_i
- $\theta_i : var(q) \rightarrow terms(v_i)$, a partial mapping
- $\overline{Y}_i$ – the h_i-image of $var_f(v_i)$,
- G_i^f, subset of free atoms of q which are θ_i-covered by some atom in $h_i(v_i)$
- G_i^b, subset of bound atoms of q which are θ_i-covered by some atom in $h_i(v_i)$,
- $B_i = \{x \in var_b(G_i^b) : \theta_i(x) \in var_f(h_i(v_i))\}$, a set of "blocked variables" ,and
- E_i, a set of "exportable variables" (details in the paper)

First Phase of MiniCount

Determine conditions under which the given views are usable in rewritings

A view may only be usable when there is a partial mapping from the query to the view, such that

- it covers at least one query subgoal
- free variables of the query are mappable to free variables of the view, and
- the partial mapping is injective on the bound variables of the query, and
- $\theta_i(B_i) \cap E_i = \emptyset$ and
- ... details in the paper! (Property 1)

FormMCDs

```
For each subgoal of the query
    For each subgoal of each view
        Choose the least restrictive head homomorphism to match
        the subgoal of the query
        If we map the variables of the query subgoal to the view then
            Add MCD for each possible extension of the mapping
            that satisfies Property 1
```

MiniCount Algorithm Example

$q_1(x, y, count) \leftarrow a(x, z) \wedge c(y, u) \wedge b(z, y) \wedge e(x)$

$v_1(y, u, s, count) \leftarrow c(y, u) \wedge d(y, s)$

$v_2(x, y, t, count) \leftarrow a(x, z) \wedge b(z, y) \wedge e(t)$

$v_3(x, count) \leftarrow a(x, z) \wedge e(x)$

$v_4(y, u, count) \leftarrow c(y, u) \wedge f(y, u)$

$v_5(y, z, count) \leftarrow b(z, y)$

$v_6(x, y, z, count) \leftarrow a(x, z) \wedge b(z, y) \wedge c(y, u) \wedge e(x) \wedge g(x, y)$

Forming the $MCDs$

4 $MCDs$ created:

MCD_1		MCD_2	
h_1	$x \rightarrow x, y \rightarrow y, t \rightarrow x$	h_2	$x \rightarrow x, y \rightarrow y, z \rightarrow z$
θ_1	$x \rightarrow x, z \rightarrow z, y \rightarrow y$	θ_2	$x \rightarrow x, z \rightarrow z,$ $y \rightarrow y, u \rightarrow u$
$\overline{Y}_1$	$\langle x, y, x \rangle$	$\overline{Y}_2$	$\langle x, y, z \rangle$
G_1^f	$\{e\}$	G_2^f	$\{e\}$
G_1^b	$\{a, b\}$	G_2^b	$\{a, b, c\}$
B_1	$\emptyset$	B_2	$\{z\}$
E_1	$\emptyset$	E_2	$\emptyset$

MCD_3		MCD_4	
h_3	$y \rightarrow y, u \rightarrow u, s \rightarrow s$	h_4	$y \rightarrow y, z \rightarrow z$
θ_3	$y \rightarrow y, u \rightarrow u$	θ_4	$z \rightarrow z, y \rightarrow y$
$\overline{Y}_3$	$\langle y, u, s \rangle$	$\overline{Y}_4$	$\langle y, z \rangle$
G_3^f	$\emptyset$	G_4^f	$\emptyset$
G_3^b	$\{c\}$	G_4^b	$\{b\}$
B_3	$\{u\}$	B_4	$\{z\}$
E_3	$\{s\}$	E_4	$\emptyset$

Second Phase of MiniCount

The MCDs from the first phase are combined to form rewritings.

The only combinations of MCDs that are considered are those that

(a) cover all the atoms of the query,

(b) don't have overlap in the mappings of bound atoms

(c) ensure that there is a bound "source" atom for view each bound atom.

(Property 2)

MiniCount is *sound* and *complete*.

Combining the MCDs

15 MCD Combinations

Combination $\{MCD_1, MCD_3\}$ satisfies Property 2 providing the rewriting:

$p_1(x, y, s, sum(w_1 \times w_2)) \leftarrow v_1(y, u, s, count) \wedge v_2(x, y, x, count)$

$r_1(x, y, w) \leftarrow p_1(x, y, s, w)$

Combination $\{MCD_2\}$ satisfies Property 2 providing the rewriting:

$p_2(x, y, w_6) \leftarrow v_6(x, y, z, w_6)$

$r_2(x, y, w) \leftarrow p_2(x, y, w)$

Combinations:

$\{MCD_1\}, \{MCD_3\}, \{MCD_4\}$,
$\{MCD_1, MCD_4\}, \{MCD_3, MCD_4\}$

violate Property 2 a)

All other Combinations violate Property 2 b)

Experimental results

- query size: 2-4 subgoals
- view set size: 2-6 views with 2-4 subgoals

Naive Algorithm		
Examples	Rewritings Checked	Execution Time(sec)
Example 1	17971	2.0565
Example 2	139	0.0145
Example 3	11	0.0008
Example 4	86	0.0080
Total	18207	2.0798

MiniCount Algorithm			
Examples	MCDs	Combinations	Exec. Time(sec)
Example 1	4	15	0.0277
Example 2	4	15	0.0012
Example 3	2	3	0.0003
Example 4	3	7	0.0007
Total	13	40	0.0299

More than an order of magnitude improvement

Random queries and views

1000 executions on random queries and views.

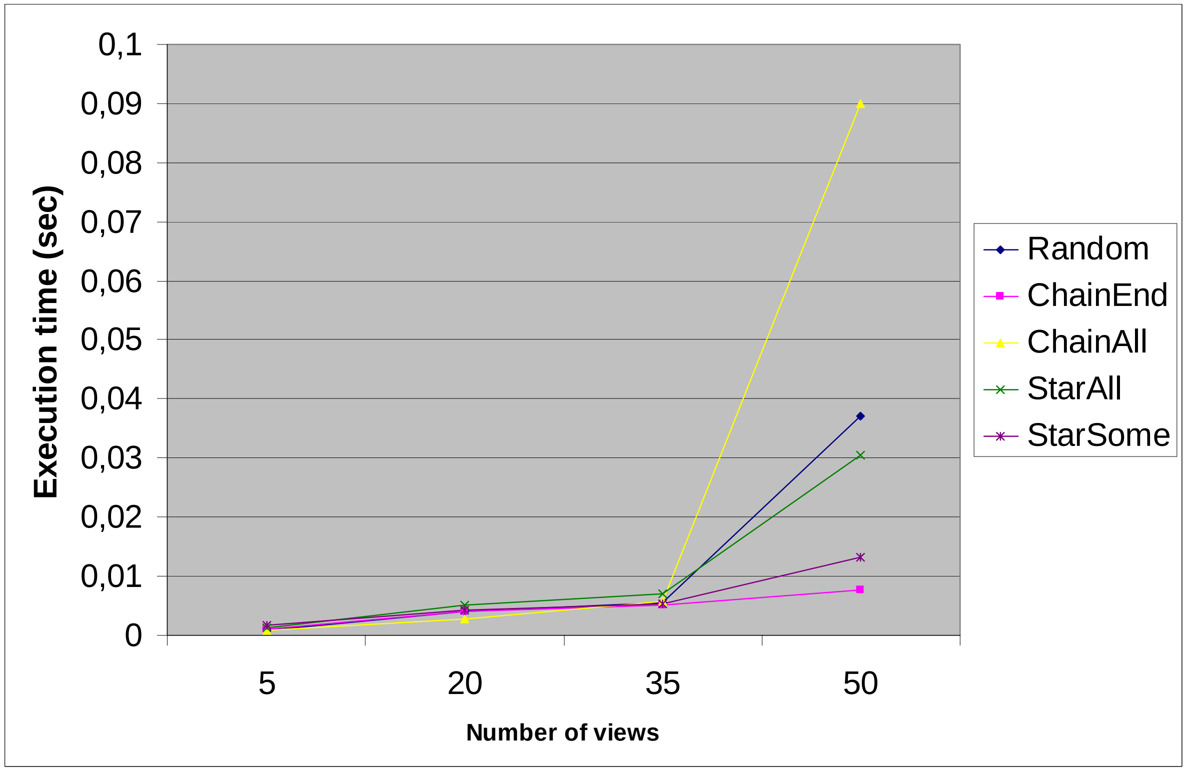
Six subgoals with one or two variables

One to six of them form the body of the query

4-6 views are generated in the same way.

Naive Algorithm			
Examples	Rewritings Checked		Execution Time(sec)
1000	34673085		56659,1912
MiniCount Algorithm			
Examples	MCDs	Combinations Checked	Execution Time(sec)
1000	1808	7160	23,2904

Scalability



Related work

- Very active research area
- For CQ: [LMSS95, LRO96, DG97, ALM02, ACGP06]
- Survey: [H01]
- Efficient algorithm for CQ: MiniCon [PL01] (cf. [M02])
- Aggregate queries and views:[SDJL96,CNS99, AC05, GRT99,GT00]
- Query containment and equivalence for aggregate queries: [CNS01, CNS03]

Thank You!

Contact: `vassalos@aueb.gr`