Answering queries using views

Susan B. Davidson
CIS 700: Advanced Topics in Databases
MW 1:30-3
Towne 309

http://www.cis.upenn.edu/~susan/cis700/homepage.html
Assigned reading

Problem statement

Suppose we are given a query Q over a database schema and a set of views V_1, ..., V_n over the same schema.

- Can we answer Q using only the answers to V_1, ..., V_n?
- What is the maximal set of tuples in the answer of Q that we can obtain from the views?
- If we can access both the views and the database relations, what is the cheapest query execution plan for answering Q?

Query optimization, database design, data integration
Example 1: Optimization

Q: Students and course titles for students registered in PhD level courses (>500) taught by professors in databases.

\[ Q(s,t):- \text{Teaches}(p,c,q,e), \text{Prof}(p,a), \text{Registered}(s,c,q), \text{Course}(c,t), c>500, a='DB' \]

\[ V_G: \text{Registration records of graduate-level courses (>400) and above.} \]

\[ V_G(s,t,c,q):- \text{Registered}(s,c,q), \text{Course}(c,t), c>400 \]

Can we use \( V_G \) to answer \( Q \)?
And if so, why would we want to?
Example 1: Optimization

\[ V_G(s,t,c,q) :\text{Registered}(s,c,q), \text{Course}(c,t), c>400 \]

\[ Q(s,t) :\text{Teaches}(p,c,q,e), \text{Prof}(p,a), \text{Registered}(s,c,q), \text{Course}(c,t), c>500, a='DB' \]

\[ Q(s,t) :\text{Teaches}(p,c,q,e), \text{Prof}(p,a), V_G(s,t,c,q), c>500, a='DB' \]
Example 2: Integration

Mediated Schema:
Course(cnum, title, univ)
Teaches(prof, cnum, quarter, eval, univ)

Source 1: Listing of all courses titled DB systems

\[ V_{DB}(t,p,c,u) \] :- Teaches(p,c,q,e,u), Course(c,t,u), t="DB Systems"

Source 2: PhD-level courses being taught at UW

\[ V_{UWPhD}(t,p,c,u) \] :- Teaches(p,c,q,e,u), Course(c,t,u), u="UW", c>500

Q: Who teaches DB Systems courses at UW?

\[ Q(p) \] :- Teaches(p,c,q,e,u), Course(c,t,u), t="DB Systems", u="UW"

How can we use the Sources to answer this?
Example 2: Integration

Source 1: Listing of all courses titled DB systems

\[ V_{DB}(t,p,c,u) : \text{Teaches}(p,c,q,e,u), \text{Course}(c,t,u), \text{t} = \text{“DB Systems”} \]

Q: Who teaches DB Systems courses at UW?

Q(p):- Teaches(p,c,q,e,u), Course(c,t,u), t = “DB Systems”, u = “UW”

\[ Q(p) : \text{V}_{DB}(t,p,c,u), u = \text{“UW”} \]
Example 3: Integration

Mediated Schema:
Course(cnum, title, univ)
Teaches(prof, cnum, quarter, eval, univ)

Source 1: Listing of all courses titled DB systems

\[ V_{DB}(t,p,c,u) :\text{- Teaches}(p,c,q,e,u), \text{Course}(c,t,u), t=\text{“DB Systems”} \]

Source 2: PhD-level courses being taught at UW

\[ V_{UWPhD}(t,p,c,u) :\text{- Teaches}(p,c,q,e,u), \text{Course}(c,t,u), u=\text{“UW”}, c>500 \]

Q: Give the title and number of all graduate level courses at UW.

\[ Q(t,c) :\text{- Teaches}(p,c,q,e,u), \text{Course}(c,t,u), t=\text{“DB Systems”}, u=\text{“UW”}, c>400 \]

How can we use the Sources to answer this? Can we get a “complete” answer?
Example 3: Integration

Q: Give the title and number of all graduate level courses at UW.

Q(t,c):- Teaches(p,c,q,e,u), Course(c,t,u), t="DB Systems", u="UW", c>400

Q(t,c):- V_{DB}(t,p,c,u), u="UW", c>400
Q(t,c):- V_{UWPhD}(t,p,c,u)

Note that courses that are not Ph.D. level or database courses will not be returned in the answer.
Some definitions

- **Query containment and equivalence**: A query $Q_1$ is *contained* in a query $Q_2$ if for all database instances $D$, $Q_1(D) \subseteq Q_2(D)$. $Q_1$ and $Q_2$ are *equivalent* if $Q_1$ is contained in $Q_2$ and $Q_2$ is contained in $Q_1$.

- **Equivalent rewritings**: Let $Q$ be a query and $\mathcal{V} = \{V_1, ..., V_n\}$ be a set of view definitions. The query $Q'$ is an *equivalent* rewriting of $Q$ using $\mathcal{V}$ if:
  - $Q'$ refers only to the views in $\mathcal{V}$, and
  - $Q'$ is equivalent to $Q$

The number of possible rewritings of a query using views is exponential in the size of the query.
Some definitions, cont.

- **Maximally-contained rewritings**: $Q'$ is a maximally-contained rewriting of $Q$ using $\mathcal{V}$ if:
  - $Q'$ refers only to the views in $\mathcal{V}$
  - $Q'$ is contained in $Q$, and
  - there is no rewriting $Q_1$ such that $Q'$ is contained in $Q_1$, $Q_1$ is contained in $Q$, and $Q_1$ is not equivalent to $Q'$.

- **Certain answers**: Let the sets of tuples $v_1, ... , v_n$ be the extensions of views $V_1, ... , V_n$.
  - Tuple $a$ is a certain answer to $Q$ under the **closed world assumption** if $a$ is in $Q(D)$ for all database instances $D$ such that $V_i(D)=v_i$ for every $i$, $1\leq i\leq n$.
  - Tuple $a$ is a certain answer to $Q$ under the **open-world assumption** if $a$ is in $Q(D)$ for all databases instances $D$ such that $V_i(D)$ contains $v_i$ for every $i$, $1\leq i\leq n$. 
Example: certain answer

• **Close-world assumption:** \((c_1, c_2)\) must be in \(R\) and is therefore a certain answer to \(Q\).

• **Open-world assumption:** since \(V_1\) and \(V_2\) are not necessarily complete, \((c_1, c_2)\) is not a certain answer. E.g. consider \(R=\{(c_1,d), (e,c_2)\}\)

\[
\begin{align*}
R(A, B) \\
V_1(a) & :- R(a,b) \\
V_2(b) & :- R(a,b) \\
Q(a,b) & :- R(a,b) \\
v_1=\{(c_1)\}, v_2=\{(c_2)\}
\end{align*}
\]
When is a view usable for a query? (Intuition)

- The set of relations in the body overlap with that of the query, and it selects attributes used in the query.
- Any predicates applied in the view must be equivalent or logically weaker than those in the query (for equivalence).
- If the view applies a logically stronger predicate it may be part of a contained rewriting.
Example: useable view

Q: Triplets (p,s,q) where the student is advised by the professor and has taken a class taught by them since winter 1998.

\[
Q(p,s,q) :\text{-} \ \text{Registered}(s,c,q), \ \text{Teaches}(p,c,q,e), \ \text{Advises}(p,s), \ q > \text{‘winter 1998’}
\]

View V1:

\[
V1(p,s,q) :\text{-} \ \text{Registered}(s,c,q), \ \text{Teaches}(p,c,q,e), \ q > \text{‘winter 1997’}
\]

\[
Q(p,s,q) :\text{-} \ V1(p,s,q), \ \text{Advises}(p,s), \ q > \text{‘winter 1998’}
\]
Example: unusable views

Q(p,s,q):- Registered(s,c,q), Teaches(p,c,q,e),
      Advises(p,s), q> ‘winter 1998’

V1(p,s,q):- Registered(s,c,q), Teaches(p,c,q,e), q> ‘winter 1997’

V2(s,q):- Registered(s,c,q), Teaches(p,c,q,e), q> ‘winter 1998’

V3(p,s,q1):- Registered(s,c,q1), Teaches(p,c,q2,e), q1> ‘winter 1998’

V4(p,s,q):- Registered(s,c,q), Teaches(p,c,q,e), Advises(p,s),
      Area(p,n), q> ‘winter 1998’

V5(p,s,q):- Registered(s,c,q), Teaches(p,c,q,e), q> ‘winter 1999’
Conditions for a view to be usable for Q

• There is a 1-1 mapping $\psi$ from relational predicates in the body of $V$ to those in the body of $Q$ which preserves relation names. Note that this mapping also induces a mapping between variables in $V$ and $Q$.

• $V$ either applies the join and selection predicates in $Q$ on the variables of the relations in the domain of $\psi$, or applies them to a logically weaker selection and the variables on which predicates still need to be applied appear as head variables.

• $V$ retains as head variables any variables that appear in $\psi$ with “unmatched” selections in $Q$. (Unless they can be recovered through other usable views.)
Using views in query optimization (conjunctive queries)

- The paper describes how to adapt a System-R style query optimizer to use materialized views.
- Rewriting queries using views is incorporated in commercial query optimizers
  - E.g. Microsoft SQL Server, IBM DB2, Oracle 8i
If the query requires the same or coarser grouping than performed in the view, and the aggregated column is either still available or can be reconstructed from other attributes, then the view is usable for the query.

What about queries with grouping and aggregation?

Q: 
```
select year, count(*), max(evaluation)
from Teaches
where cnum > 500
``` 
group by year

V: 
```
select cnum, year, max(evaluation), count(*) as offerings
from Teaches
where cnum > 400
``` 
group by cnum, year

Q': 
```
select year, sum(offerings), max(evaluation)
from V
where cnum > 500
``` 
group by year
Data integration: the Bucket algorithm

\[ Q(s,c,p) :\text{-} \text{Teaches}(p,c,q), \text{Registered}(s,c,q), \text{Course}(c,t), \]
\[ c>300, q>\text{‘Aut95’} \]

\[ V_1(s,c,q,t) :\text{-} \text{Registered}(s,c,q), \text{Course}(c,t), c>500, q>\text{‘Aut98’} \]

\[ V_2(s,p,c,q) :\text{-} \text{Registered}(s,c,q) \text{Teaches}(p,c,q) \]

\[ V_3(s,c) :\text{-} \text{Registered}(s,c,q), q<\text{‘Aut94’} \]

\[ V_4(p,c,t,q) :\text{-} \text{Registered}(s,c,q), \text{Course}(c,t), \text{Teaches}(p,c,q), q<\text{‘Aut97’} \]

<table>
<thead>
<tr>
<th>Teaches(p,c,q)</th>
<th>Registered(s,c,q)</th>
<th>Course(c,t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_2(s',p,c,q))</td>
<td>(V_1(s,c,q,t'))</td>
<td>(V_1(s',c,q',t))</td>
</tr>
<tr>
<td>(V_4(p,c,t',q))</td>
<td>(V_2(s,p',c,q))</td>
<td>(V_4(p',c,t,q'))</td>
</tr>
</tbody>
</table>

**Step 1:** Create buckets for relational subgoals of \(Q\) containing relevant views.
Data integration: the Bucket algorithm

Q(s,c,p):- Teaches(p,c,q), Registered(s,c,q), Course(c,t), c>300, q> ‘Aut95’

<table>
<thead>
<tr>
<th>Teaches(p,c,q)</th>
<th>Registered(s,c,q)</th>
<th>Course(c,t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2(s’,p,c,q)</td>
<td>V1(s,c,q,t’)</td>
<td>V1(s’,c,q’,t)</td>
</tr>
<tr>
<td>V4(p,c,t’,q)</td>
<td>V2(s,p’,c,q)</td>
<td>V4(p’,c,t,q’)</td>
</tr>
</tbody>
</table>

Step 2: Combine elements from buckets, checking joint conditions.

Q'(s,c,p):- V2(s’,p,c,q),V1(s,c,q,t’),V1(s’,c,q’,t), c>300, q> ‘Aut95’

Can be simplified to:
Q'(s,c,p):- V2(s,p,c,q),V1(s,c,q,t’), c>300, q> ‘Aut95’
Data integration: the Bucket algorithm

\[ Q(s,c,p) :\text{-} \text{Teaches}(p,c,q), \text{Registered}(s,c,q), \text{Course}(c,t), c>300, q> 'Aut95' \]

<table>
<thead>
<tr>
<th>\text{Teaches}(p,c,q)</th>
<th>\text{Registered}(s,c,q)</th>
<th>\text{Course}(c,t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{V2}(s',p,c,q)</td>
<td>\text{V1}(s,c,q,t')</td>
<td>\text{V1}(s',c,q',t)</td>
</tr>
<tr>
<td>\text{V4}(p,c,t',q)</td>
<td>\text{V2}(s,p',c,q)</td>
<td>\text{V4}(p',c,t,q')</td>
</tr>
</tbody>
</table>

\textbf{Step 2:} Combine elements from buckets, checking joint conditions.

\[ Q'(s,c,p) :\text{-} \text{V4}(p,c,t',q), \text{V1}(s,c,q,t'), \text{V4}(p',c,t,q'), c>300, q> 'Aut95' \]

Would be eliminated because of conflicting conditions (in \text{V4}, q< 'Aut97'; in \text{V1}, q> 'Aut98')
Data integration: the Bucket algorithm

Q(s,c,p):- Teaches(p,c,q), Registered(s,c,q), Course(c,t), c>300, q> ‘Aut95’

\[
\begin{array}{|c|c|c|}
\hline
\text{Teaches}(p,c,q) & \text{Registered}(s,c,q) & \text{Course}(c,t) \\
\hline
V_2(s',p,c,q) & V_1(s,c,q,t') & V_1(s',c,q',t) \\
V_4(p,c,t',q) & V_2(s,p',c,q) & V_4(p',c,t,q') \\
\hline
\end{array}
\]

Step 2: Combine elements from buckets, checking joint conditions.

Q'(s,c,p):- V_4(p,c,t',q), V_2(s,p,c,q), c>300, q> ‘Aut95’

Would also be produced (after simplification), yielding the answer:

Q'(s,c,p):- V_2(s,p,c,q), V_1(s,c,q,t'), c>300, q> ‘Aut95’
Q'(s,c,p):- V_4(p,c,t',q), V_2(s,p,c,q), c>300, q> ‘Aut95’
Query rewriting using views has many practical applications (e.g. physical data independence, query optimization, data integration)

- Data integration: maximally-contained rewritings, and assume that there is a large number of views
- Query optimization: equivalent rewritings, and assume a smaller number of views

The problem raises many challenges, from theoretical foundations to practical implementation

- Completeness of query rewriting algorithm: Given a set of views and a query, will the algorithm always find a rewriting of the query using the views if one exists?
- Integrity constraints, e.g. keys and foreign keys?
- Applications to object-query languages and semi-structured data (e.g. XML)