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

- “Answering queries using views: A survey” by Alon Halevy. VLDB Journal 10:270-294 (2001).

Problem statement

Suppose we are given a query Q over a database schema and a set of views V1, ..., Vn over the same schema.

- Can we answer Q using only the answers to V1, ..., Vn ?
- 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

Prof(name, area)
Course(cnum, title)
Teaches(prof, cnum, quarter, eval)
Registered(student, cnum, quarter)

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

Q(s,t):- Teaches(p,c,q,e), Prof(p,a), Registered(s,c,q), Course(c,t), c>500, a='DB'

V_G: Registration records of graduate-level courses (>400) and above.

V_G (s,t,c,q):- Registered(s,c,q), Course(c,t), c>400

Can we use V_G to answer Q?
And if so, why would we want to?

Example 1: Optimization

Prof(name, area)
Course(cnum, title)
Teaches(prof, cnum, quarter, eval)
Registered(student, cnum, quarter)

V_G(s,t,c,q):- Registered(s,c,q), Course(c,t), c>400

Q(s,t):- Teaches(p,c,q,e), Prof(p,a), Registered(s,c,q), Course(c,t), c>500, a='DB'

Q(s,t):- Teaches(p,c,q,e), 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

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"

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):- V_{DB} (t,p,c,u), u="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):- 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: 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

How can we use the Sources to answer this?
Can we get a "complete" answer?

Example 3: Integration

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

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

$\Downarrow$

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 Q1 is *contained* in a query Q2 if for all database instances D, Q1(D)$\subseteq$Q2(D). Q1 and Q2 are *equivalent* if Q1 is contained in Q2 and Q2 is contained in Q1.

- **Equivalent rewritings**: Let Q be a query and $\mathcal{V}$={V1,…,Vn} 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 Q1 such that Q' is contained in Q1, Q1 is contained in Q, and Q1 is not equivalent to Q'.
- **Certain answers**: Let the sets of tuples v1, … , vn be the extensions of views V1, … , Vn.
  - 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 Vi(D)=vi for every i, 1<=i<=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 Vi(D) *contains* vi for every i, 1<=i<=n.

Example: certain answer

R(A, B)
V1(a):- R(a,b)
V2(b):- R(a,b)

Q(a,b):- R(a,b)

v1={(c1)}, v2={(c2)}

- **Close-world assumption**: (c1, c2) must be in R and is therefore a certain answer to Q.
- **Open-world assumption**: since V1 and V2 are not necessarily complete, (c1, c2) is not a certain answer. E.g. consider R={(c1,d), (e,c2)}

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

Teaches(prof, cnum, quarter, eval)
Registered(student, cnum, quarter)
Area(prof, name)
Advises(prof, student)

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

View V1:

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

Q(p,s,q):- V1(p,s,q), Advises(p,s), q> 'winter 1998'

Example: unusable views

Teaches(prof, cnum, quarter, eval)
Registered(student, cnum, quarter)
Area(prof, name)
Advises(prof, student)

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 ψ 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 ψ, 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 ψ 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

What about queries with grouping and aggregation?

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

coarser grouping than

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

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

Data integration: the Bucket algorithm

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

V1(s,c,q,t):- Registered(s,c,q), Course(c,t), c>500, q> 'Aut98'
V2(s,p,c,q):- Registered(s,c,q) Teaches(p,c,q)
V3(s,c):- Registered (s,c,q), q< 'Aut94'
V4(p,c,t,q):- Registered(s,c,q), Course(c,t), Teaches(p,c,q), q< 'Aut97'

Teaches(p,c,q)	Registered(s,c,q)	Course(c,t)
V2(s',p,c,q)	V1(s,c,q,t')	V1(s',c,q',t)
V4(p,c,t',q)	V2(s,p',c,q)	V4(p',c,t,q')

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’

Teaches(p,c,q)	Registered(s,c,q)	Course(c,t)
V2(s′,p,c,q)	V1(s,c,q,t′)	V1(s′,c,q′,t)
V4(p,c,t′,q)	V2(s,p′,c,q)	V4(p′,c,t,q′)

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

Teaches(p,c,q)	Registered(s,c,q)	Course(c,t)
V2(s',p,c,q)	V1(s,c,q,t')	V1(s',c,q',t)
V4(p,c,t',q)	V2(s,p',c,q)	V4(p',c,t,q')

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

Q'(s,c,p):- V4(p,c,t',q),V1(s,c,q,t'),V4(p',c,t,q'), c>300, q> 'Aut95'

Would be eliminated because of conflicting conditions (in V4, q< 'Aut97'; in 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’

Teaches(p,c,q)	Registered(s,c,q)	Course(c,t)
V2(s’,p,c,q)	V1(s,c,q,t’)	V1(s’,c,q’,t)
V4(p,c,t’,q)	V2(s,p’,c,q)	V4(p’,c,t,q’)

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

Q’(s,c,p):- V4(p,c,t’,q),V2(s,p,c,q), c>300, q> ‘Aut95’

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

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

Q’(s,c,p):- V4(p,c,t’,q),V2(s,p,c,q), c>300, q> ‘Aut95’

Summary

- 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)