Answering Queries Using Views: A Survey

Paper by Alon Halevy
Presentation by Rachel Pottinger
Discussion by Meeta Mistry

Reminders

- A view is a stored query
- A datalog query example:
  q(code):- Airport(code, city),
  Feature(city, “Beach”)
  Find all airport codes of cities that have beaches

Answering Queries Using Views – basic definition

- Answer a query using a view rather than using the underlying base table
- Query: q(code):- Airport(code, city),
  Feature(city, POI)
- View:
  feature-code(code,POI):- Airport(code, city),
  Feature(city,POI)
- Rewriting using view:
  q(code):-feature-code(code,POI)

Two distinct problems:

- Query optimization
- Data integration
- Physical Data Independence

AQUV in Query Optimization Goals

- Speed Query Processing
- Still need exact answers

AQUV in Query Optimization: Closed World Assumption

- Closed World Assumption
  - Views are complete
  - Think of as “If and only if”
  - feature-code(code, POI):- Airport(code, city),
    Feature(city, POI)
    retrieves *all* airport codes for cities with beaches
- How do we know this holds? Given from problem – can’t tell from view definition

AQUV in Query Optimization: Looking for Equivalent Rewritings

- Rewritings must be *equivalent*
  - Think of as "rewritten query must retrieve *exactly* the same answers as the original query"
- Equivalent ex:
  Query: q(code):- Airport(code, city),
  Feature(city,POI)
  View: feature-code(code,POI):- Airport(code, city),
  Feature(city, POI)
  Equivalent Rewriting: q(code):-feature-code(code, POI)
- Non-equivalent ex:
  Same Query
  View: Beach-code(code):- Airport(code,city),
  Feature(city, "Beach")
- Non-equivalent rewriting: q(code):-beach-code(code)

AQUV in Query Optimization: Can still access base relations

- Can access views *and* base relations
- Ex:
  - Query:
    q(code, URL):- Airport(code, city),
    Feature(city,POI), Webinfo(POI, URL)
  - View:
    feature-code(code,POI):- Airport(code, city),
    Feature(city, POI)
  - Rewriting:
    q(code,URL):-feature-code(code,POI),
    Webinfo(POI, URL)

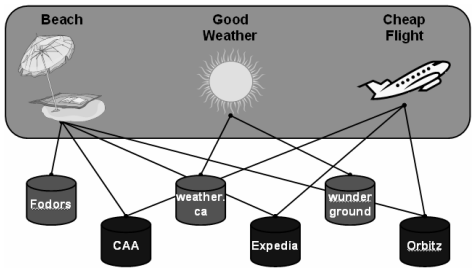
AQUV in Query Optimization: General Algorithm

- Fold into System-R style optimizer
- It's just another access path

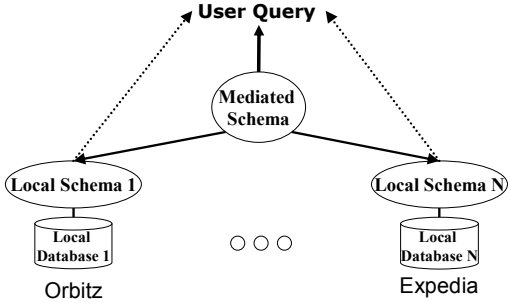
AQUV in Query Optimization: Discussion

- Imagine that you're building a query optimizer. Would you consider it worthwhile to use views when answering queries? Why or why not? Would you try it only for certain kinds of queries? Which ones?

AQUV in Data Integration: Example: Planning a Beach Vacation



Potential Data Integration Architecture: Local-As-View (LAV)

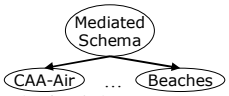


Local sources are views on mediated schema

Local As View (LAV)

LAV: local source is *materialized view* over mediated schema

Mediated Schema:
Airport(code, city)
Feature(city, attraction)

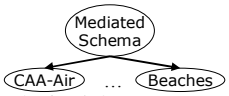


Local Sources/Views:
CAA-Air(code, city) :- Airport(code, city)
Beaches(code) :- Airport(code, city), Feature(city, "Beach")

Local As View (LAV)

LAV: local source is *materialized view* over mediated schema

Mediated Schema:
Airport(code, city)
Feature(city, attraction)



Local Sources/Views:
CAA-Air(code, city) :- Airport(code, city)
Beaches(code) :- Airport(code, city), Feature(city, "Beach")

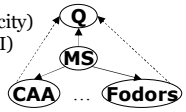
- Adding new sources is easy
- Rewriting queries is NP-complete

AQUV in Data Integration: Assumptions

- Open World Assumption
  - Each source only has *some* of the tuples
  - Read as "if → then"
  - Fodors(city, POI) :- Feature(city, POI)
    Fodors has *some* Features
  - This is an assumption – you can't tell from view definition
- Can't access base relations
  - May not be able to find an equivalent rewriting

AQUV in Data Integration: Maximally Contained Rewritings

Query:
Dest(code) :- Airport(code, city), Feature(city, "Beach")
Sources/Views:
CAA-Air(code, city) :- Airport(code, city)
Fodors(city, POI) :- Feature(city, POI)

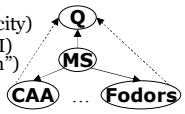


Rewriting:
Dest(code):-CAA-Air(code, city), Fodors(city, "Beach")

Maximally Contained Rewriting: all answers to Query are a subset of those of Rewriting, and Rewriting contains all possible answers given local sources

Answering Queries Using Views

Query:
Dest(code) :- Airport(code, city), Feature(city, "Beach")
Sources/Views:
CAA-Air(code, city) :- Airport(code, city)
Fodors(city, POI) :- Feature(city, POI)
Sun-Surf(city) :- Feature(city, "Beach")



Rewriting:
Dest(code):-CAA-Air(code, city), Fodors(city, "Beach") ∪
Dest(code):-CAA-Air(code, city), Sun-Surf(city)

Maximally Contained Rewriting: all answers to Query are a subset of those of Rewriting, and Rewriting contains all possible answers given local sources

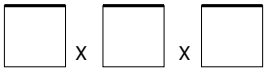
How do we find the Maximally Contained Rewriting?

AQUV in Data Integration: Discussion

- Consider the use of equivalent rewritings in database integration. What problems might this pose? What are the possible benefits?
- Can you think of other contexts than data integration where a maximally-contained rewriting would make sense?

Naïve Solution: Bucket Algorithm

- Created as part of Information Manifold, Levy et al.
- Algorithm:
  1. Create a bucket for each query subgoal, place all relevant views into the bucket:



Q(X):- $g_1(x_1)$, …, $g_n(x_n)$

  2. For each element in cross product of the buckets, check containment

Subgoal Interaction

The Bucket Algorithm does not recognize interactions:

Query:
Dest(code) :- Airport(code, city), Feature(city, "Beach")

Sources/Views:
Orbitz(code):- Airport(code, city)
Beaches(code) :- Airport(code, city), Feature(city, "Beach")
Frommers(city, POI):-Feature(city,POI)

Bucket would check:
Dest'(code):-Orbitz(code),Frommers(city,"Beach")

Expanding this gets:
Dest'(code):-Airport(code,_),Feature(city, "Beach")

All answers to Dest' are not answers Dest (containment)

The MiniCon Algorithm: Phase One [Pottinger & (Ha)Levy: VLDB]

Query:
Dest(code) :- Airport(code, city), Feature(city, "Beach")

Sources/Views:
Orbitz(code) :- Airport(code, city)
Beaches(code) :- Airport(code, city),Feature(city, "Beach")

Rewriting:
Dest(code) :- Beaches(code)

Create MiniConDescriptions (MCDs): View subgoals linked by existential variables *must* be mapped together

MiniCon Algorithm Phase Two: Combine MCDs with non-overlapping subgoals

Combine MCDs with non-overlapping subgoals

Query:
Dest(code) :- Airport(code, city), Feature(city, "Beach"), Flight("YVR", code, airline, number)

Sources/Views:
Orbitz(code) :- Airport(code, city)
Beaches(code) :- Airport(code, city),Feature(city, "Beach")
Expedia(orig, dest) :- Flight(orig, dest, airline, number)

Rewriting:
Dest(code) :- Beaches(code), Expedia("YVR", code)

Fewer Combinations
No Explicit Containment Check

AQUV Algorithms: Discussion

- Does the computational complexity of these problems surprise you? Do they seem harder or easier than expected?
- How would you scale the complexity of each of the algorithms presented in terms of the completeness of the algorithms?