

# Information Integration

Mediators

Answering Queries Using Views



Jeff Ullman, Stanford Univ.

(presentation shortened, some minor modifications by R. Möller)

# Importance of Information Integration

- ◆ Very many modern DB applications involve combining databases.
- ◆ Sometimes a “database” is not stored in a DBMS --- it could be a spreadsheet, flat file, XML document, etc.




# Challenges

-  *Legacy databases* : DB's get used for many applications.
  - ◆ You can't change its structure for the sake of one application, because it will cause others to break.
-  *Incompatibilities* : Two, supposedly similar databases, will mismatch in many ways.

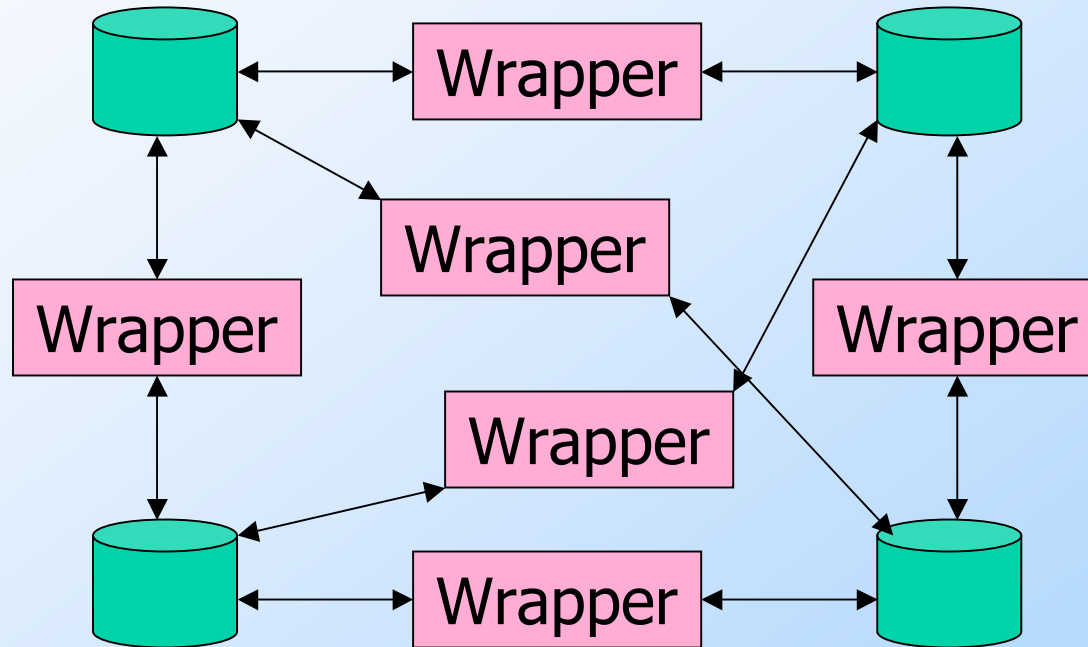
# Examples: Incompatibilities

- ◆ *Lexical* : `addr` in one DB is `address` in another.
- ◆ *Value mismatches* : is a “red” car the same color in each DB? Is 20 degrees Fahrenheit or Centigrade?
- ◆ *Semantic* : are “employees” in each database the same? What about consultants? Retirees? Contractors?

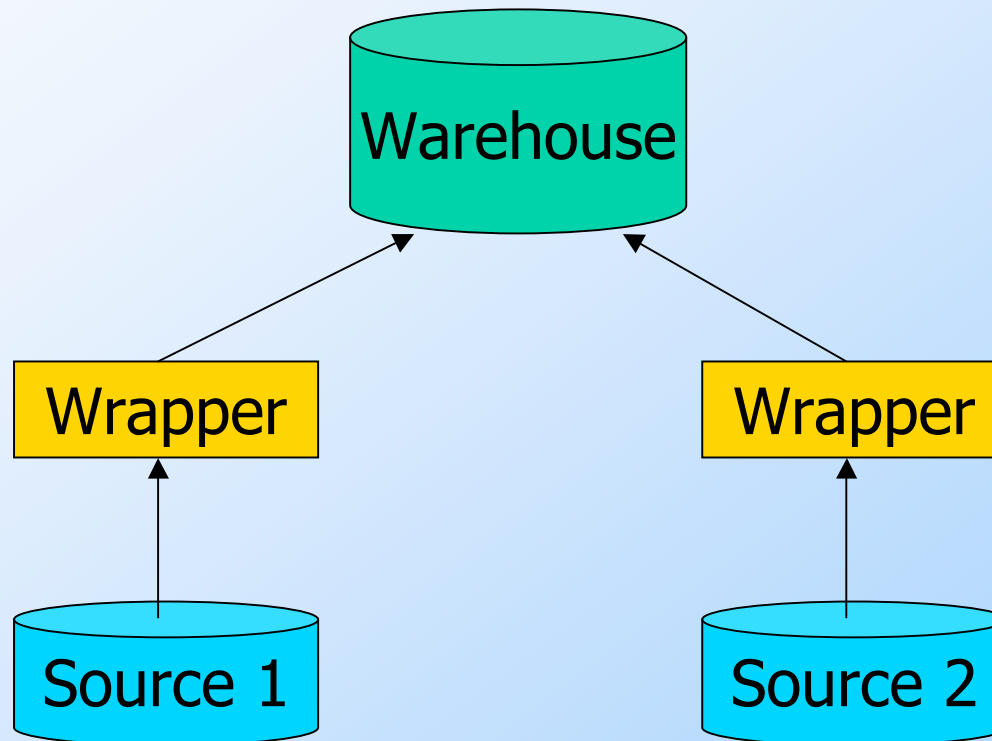
# Integration Architectures

-  *Federation* : everybody talks directly to everyone else.
-  *Warehouse* : Sources are translated from their local schema to a global schema and copied to a central DB.
-  *Mediator* : Virtual warehouse --- turns a user query into a sequence of source queries.

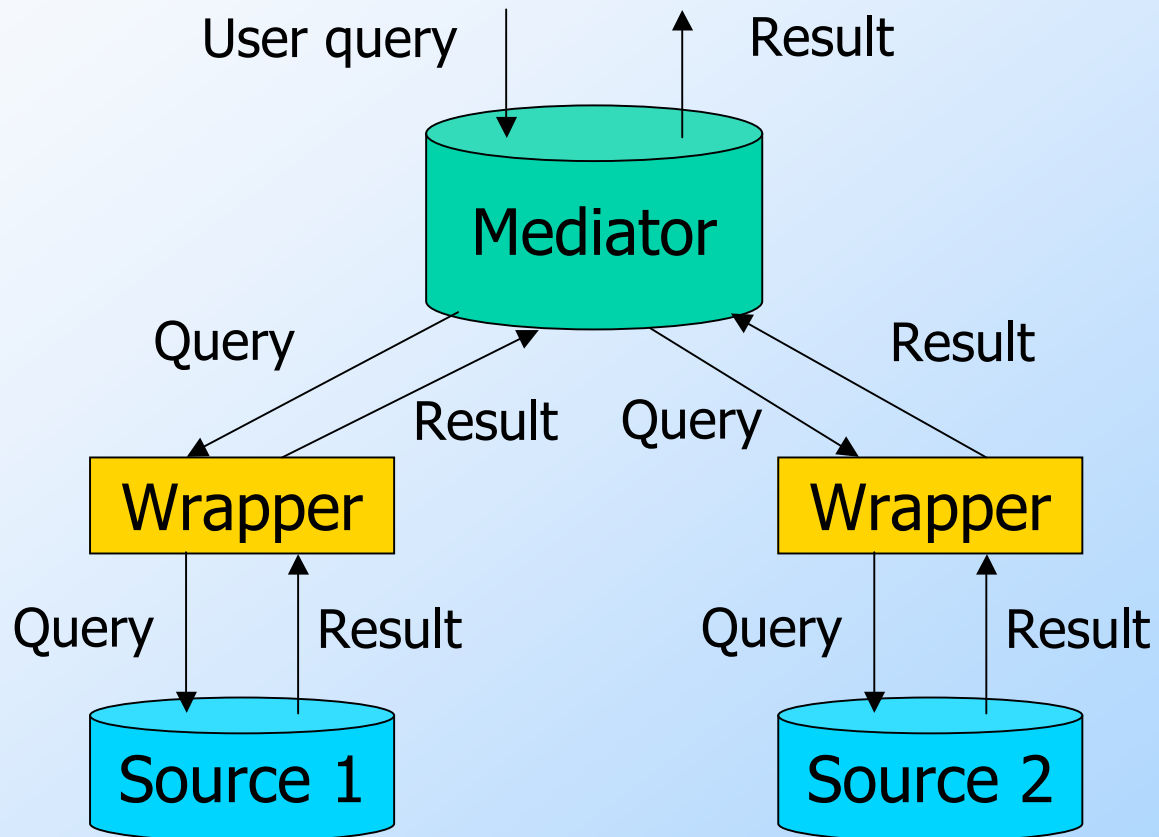
# Federations



# Warehouse Diagram



# A Mediator



# View-Centric Mediation

- ◆ Key assumptions:
  - ✍ There is a set of global predicates that define the schema.
    - ◆ These do not exist as stored relations.
  - ✍ Each data source has its capabilities defined by views, which are (typically) CQ's whose subgoals involve the global predicates.

# Assumptions --- Continued

3. A query is (typically) a CQ over the global predicates.
4. A *solution* is an expression (union of CQ's, typically) involving the views.
  - ▶ Ideally, the solution is equivalent to the query.
  - ▶ In practice, we have to be happy with a solution maximally contained in the query.

# Interpretation of Views

- ◆ A view describes (some of) the facts that are available at the source.
- ◆ A view does not define exactly what is at the source.
  - ◆ Example: a view  $v(X) :- p(X, 10)$  says that the source has some  $p$ -facts with second component 10 ---  $v$  could even be empty although  $p(X, 10)$  is not.

# Put Another Way ...

- ◆ The :- separator between head and body of a view definition should not be interpreted as "if."
- ◆ Rather, it is "only if."

# Example

- ◆ Global predicates:

$\text{emp}(E)$  = "E is an employee."

$\text{phone}(E,P)$  = "P is a phone of E."

$\text{office}(E,O)$  = "O is an office of E."

$\text{mgr}(E,M)$  = "M is E's manager."

$\text{dept}(E,D)$  = "D is E's department."

# Example --- Continued

- ◆ Three sources each provide one view:

At source S1: view  $v1(E,P,M)$  defined by:

$$v1(E, P, M) \text{ :- emp}(E) \ \& \ \text{phone}(E, P) \ \& \\ \text{mgr}(E, M)$$

- ◆ Interpretation: “every triple  $(e,p,m)$  at S1 is an employee, one of their phones, and their manager.”
- ◆ It does not say “S1 has all  $E-P-M$  facts.”

# Example: Sources S2 and S3

## ◆ At S2:

```
v2 (E, O, D) :- emp (E) & office (E, O) &
                dept (E, D)
```

- ▶ S2 has (some of the) employee-office-department facts.

## ◆ At S3:

```
v3 (E, P) :- emp (E) & phone (E, P) &
              dept (E, 'toy')
```

- ▶ S3 has (some) toy-department phones.

# Example: A Query

```
q1(P,O) :- phone('sally',P) &  
           office('sally',O)
```



► Find Sally's office and phone.

◆ There are two useful solutions:

```
s1(P,O) :- v1('sally',P,M) &  
           v2('sally',O,D)
```

```
s2(P,O) :- v3('sally',P) &  
           v2('sally',O,D)
```




# What Makes a Solution $S$ Useful?

-  There must be no other solution containing  $S$ .
-   $S$ , when expanded from views into global predicates, is contained in the query.

# Expanding Views

- ◆ Suppose we have a subgoal  $v(X,Y)$  in a solution, and  $v$  is defined by:

$$v(A,B) \text{ :- } p(A,X) \ \& \ q(X,B)$$

-  Find unique variables for the local variables of the view (those that appear only in the body).
-  Substitute variables of the subgoal for variables of the head.
-  Use the resulting body as the substitution.

# Example

$v(A, B) :- p(A, X) \ \& \ q(X, B)$

**becomes:**

$v(A, B) :- p(A, X1) \ \& \ q(X1, B)$

**Then substitute  $A \rightarrow X, B \rightarrow Y$ ; yields body:**

$p(X, X1) \ \& \ q(X1, Y)$

# Important Points

- ◆ To test containment of a solution in a query, we expand the solution first, then test CQ containment of the expansion in the query.
- ◆ The view definition describes what any tuples of the view look like, so CQ containment implies that the solution will provide only true answers.

# The Picture

Query:  $q(X,Y) :- p(X,Z) \& \dots$

Soln:  $q(A,B) :- v(A,C,D) \& w(B,E) \& \dots$

Exp:  $q(A,B) :- p(A,U) \& \dots \& r(B,V) \& \dots$

Is there a containment mapping?

# Important Points --- (2)

- ◆ There is no guarantee a solution supplies any answers to the query.
- ◆ Comparing different solutions by testing if one solution is contained in another must be done at the level of the unexpanded views.

# Example

- ◆ Two sources might have similar views, defined by:

$$\begin{aligned} v1(X, Y) & :- p(X, Y) & v2(X, Y) \\ & :- p(X, Y) \end{aligned}$$

- ◆ But the sources actually have different sets of  $p$ -facts.

# Example --- Continued

- ◆ Then, the two solutions:

$s1(X, Y) \text{ :- } v1(X, Y) \quad s2(X, Y)$   
 $\text{ :- } v2(X, Y)$

have the same expansions,  $p(X, Y)$ , but there is no reason to believe one solution is contained in the other.

- ▶ One view could provide lots of  $p$ -facts, the other, few or none.

# Important Points --- (3)

- ◆ On the other hand, when one solution, unexpanded, is contained in another, we can be sure the first provides no answers the second does not.

# Example

- ◆ Here are two solutions:

$s1(X, Y) :- v1(X, Z) \ \& \ v2(Z, Y)$

$s2(X, Y) :- v1(X, Z) \ \& \ v2(W, Y)$

- ◆ There is a containment mapping  $s2 \rightarrow s1$ .
  - ▶ Thus,  $s1 \subseteq s2$  at the level of views.
- ◆ No matter what tuples  $v1$  and  $v2$  represent,  $s2$  provides all answers  $s1$  provides.

# The Office Example

```
q1 (P, O) :- phone('sally', P) &
             office('sally', O)
```

```
v1 (E, P, M) :- emp(E) & phone(E, P) &
                mgr(E, M)
```

```
v2 (E, O, D) :- emp(E) & office(E, O) &
                dept(E, D)
```

```
v3 (E, P) :- emp(E) & phone(E, P) &
             dept(E, 'toy')
```

# Office Example --- Solutions

```
s1(P, O) :- v1('sally', P, M) &  
           v2('sally', O, D)
```

```
s2(P, O) :- v3('sally', P) &  
           v2('sally', O, D)
```

# Expansion of S1

```
e1(P, O) :- emp('sally') &  
  phone('sally', P) & mgr('sally', M)  
  & emp('sally') & office('sally', O)  
  & dept('sally', D)
```

```
q1(P, O) :- phone('sally', P) &  
  office('sally', O)
```

Containment  
mapping q1->e1

# Office Example --- Concluded

- ◆ Mapping from  $q_1$  to  $s_2$  is similar.
- ◆ Notice we have used the head predicate to name the solution, expansion, etc.
  - ▶ Technically, head predicates have to be the same, but that's not a problem here.
- ◆ Expansions are properly contained in query --- not equivalent.

# Finding All Solutions to a Query

- ◆ Key idea: LMSS (Levy-Mendelzon-Sagiv-Srivastava) test.
- ◆ If a query has  $n$  subgoals, then we only need to consider solutions with at most  $n$  subgoals.
  - ◆ Any other solution must be contained in one with  $\leq n$  subgoals.

# Proof of LMSS Theorem

- ◆ Suppose the query has  $n$  subgoals, and a solution  $S$  has  $>n$  subgoals.
- ◆ Look at the expansion diagram again – at least one subgoal (view) in the solution has an expansion to which no query subgoal maps.

# Expansion Diagram

←  $n$  of these →

Query:  $q(X,Y) :- p(X,Z) \& \dots$

Soln:  $q(A,B) :- v(A,C,D) \& w(B,E) \& \dots$

Exp:  $q(A,B) :- p(A,U) \& \dots \& r(B,V) \& \dots$

← More than  $n$  of these →

# Proof --- Continued

- ◆ Consider the new solution  $S'$ , which removes from  $S$  every subgoal whose expansion is not a target of the CM from the query.
- ◆ Clearly  $S \subseteq S'$ .
  - ◆ In general, throwing away subgoals grows the result of the CQ.
- ◆ But  $S'$  has at most  $n$  subgoals.

# Example

- ◆ In our running “office” example, we can immediately conclude that the solution

`s3(P, O) :- v1('sally', P, M) &  
v2('sally', O, D) & v3(E, P)`

is not minimal.

- ▶ It has more subgoals than the query.
- ▶ In fact, it is contained in s1.