Information Integration

Mediators

Semistructured Data

Answering Queries Using Views

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.

Example Applications

1. Enterprise Information Integration: making separate DB's, all owned by one company, work together.
2. Scientific DB's, e.g., genome DB's.
3. Catalog integration: combining product information from all your suppliers.
4. Etc., etc.

Challenges

1. *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.
2. *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?

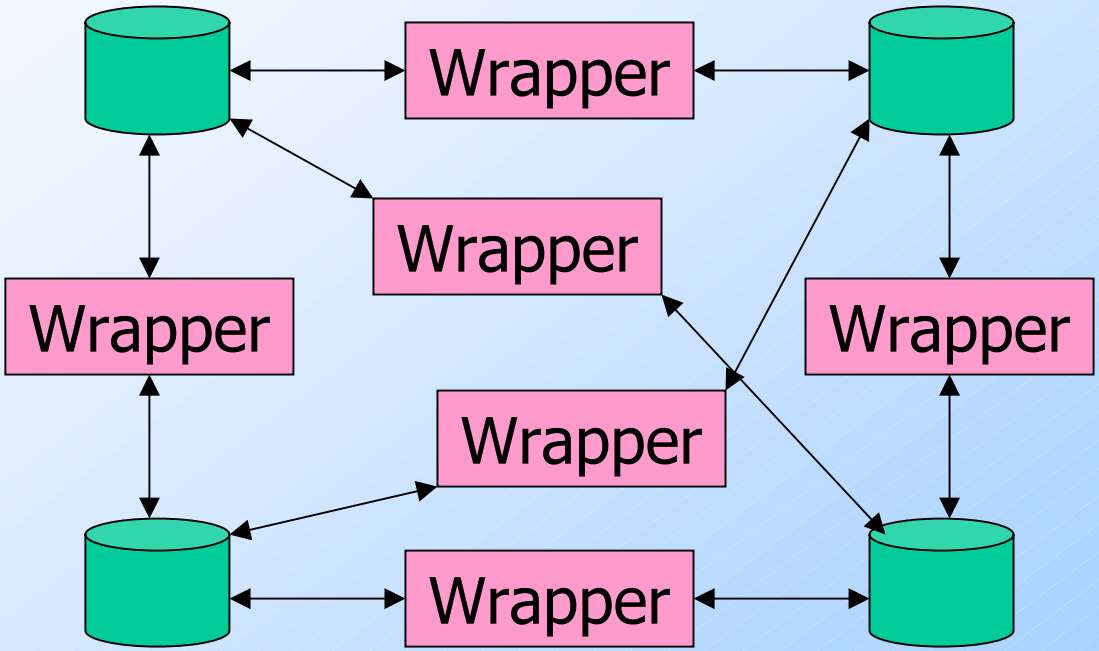
What Do You Do About It?

- Grubby, handwritten translation at each interface.
  - Some research on automatic inference of relationships.
- *Wrapper* (aka "adapter") translates incoming queries and outgoing answers.

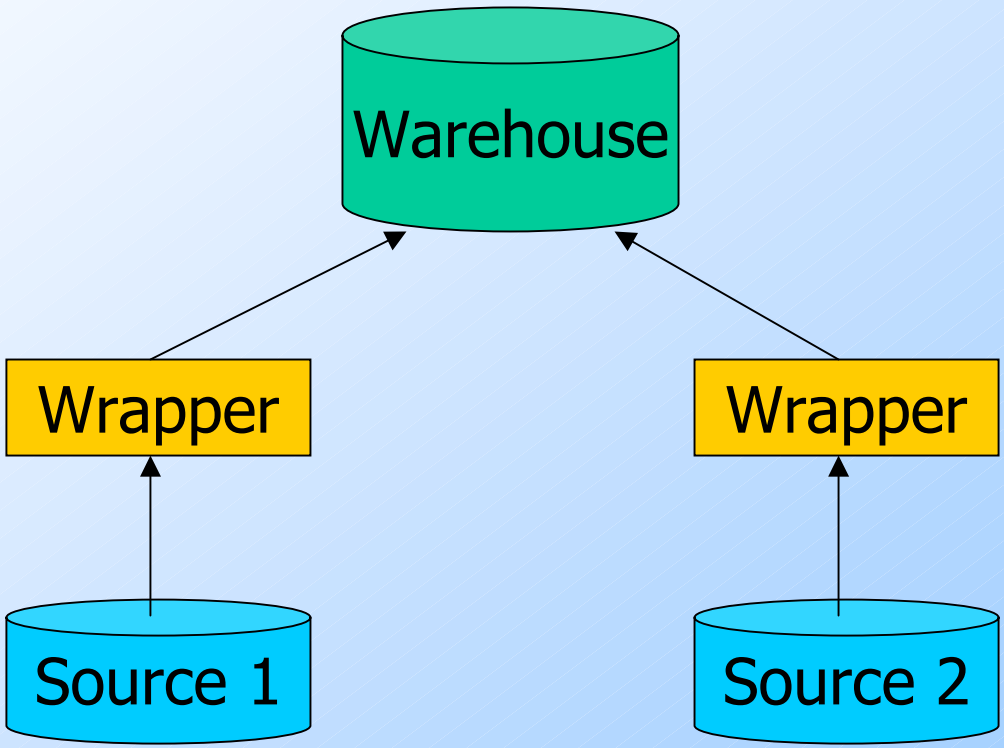
Integration Architectures

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

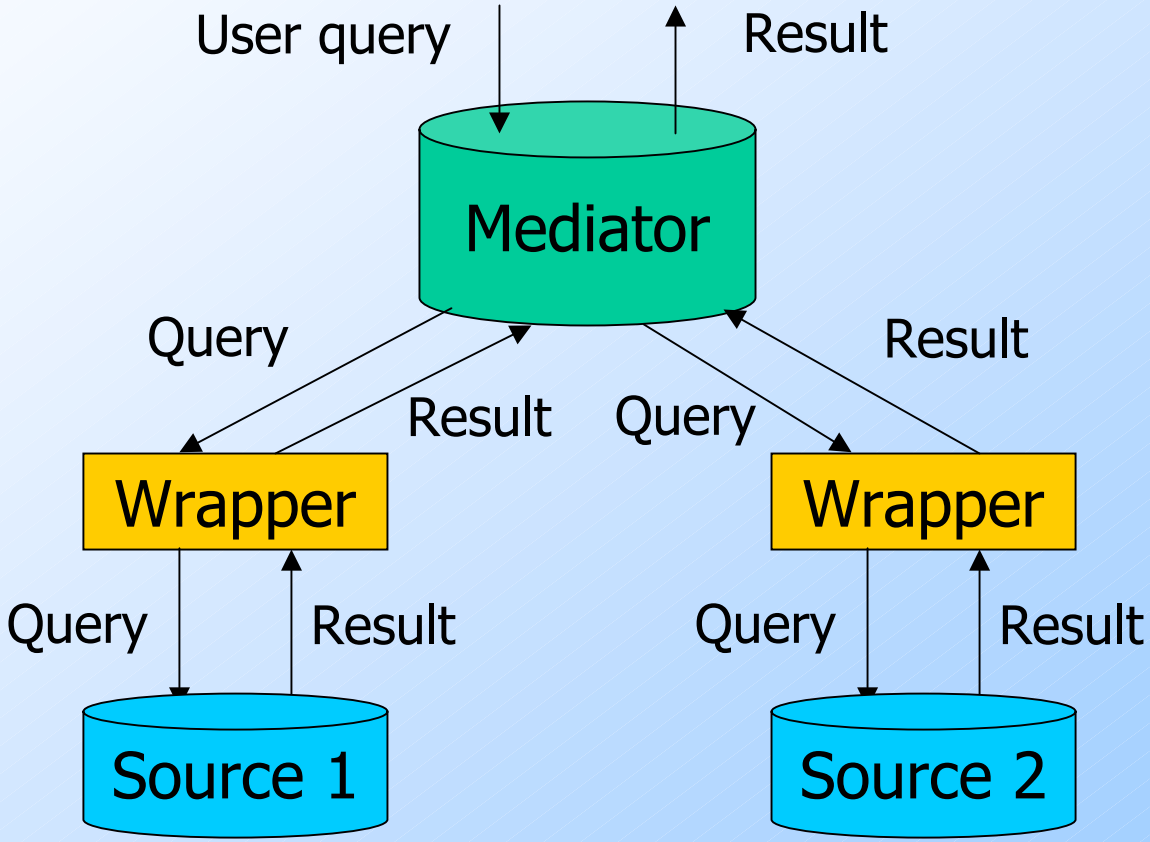
Federations



Warehouse Diagram



A Mediator


User query
Result
Mediator
Query
Result
Result
Query
Wrapper
Wrapper
Query
Result
Query
Result
Source 1
Source 2

Two Mediation Approaches

1. *Query-centric* : Mediator processes queries into steps executed at sources.
2. *View-centric* : Sources are defined in terms of global relations; mediator finds all ways to build query from views.

Example

- Suppose Dell wants to buy a bus and a disk that share the same protocol.
- Global schema:

```
Buses(manf,model,protocol)
Disks(manf,model,protocol)
```

- Local schemas: each bus or disk manufacturer has a (model,protocol) relation --- manf is implied.

Example: Query-Centric

- Mediator might start by querying each bus manufacturer for model-protocol pairs.
  - The wrapper would turn them into triples by adding the manf component.
- Then, for each protocol returned, mediator queries disk manufacturers for disks with that protocol.
  - Again, wrapper adds manf component.

Example: View-Centric

- Sources' capabilities are defined in terms of the global predicates.
  - E.g.,Quantum's disk database could be defined by QuantumView(M,P) = Disks('Quantum',M,P).
- Mediator discovers all combinations of a bus and disk "view," equijoined on the protocol components.

Comparison

- Query-centric is simpler to implement.
  - Lets you have control of what the mediator does.
- View centric is more extensible.
  - Same query engine works for any number of sources.
  - Add a new source simply by defining what it contributes as a view of the global schema.

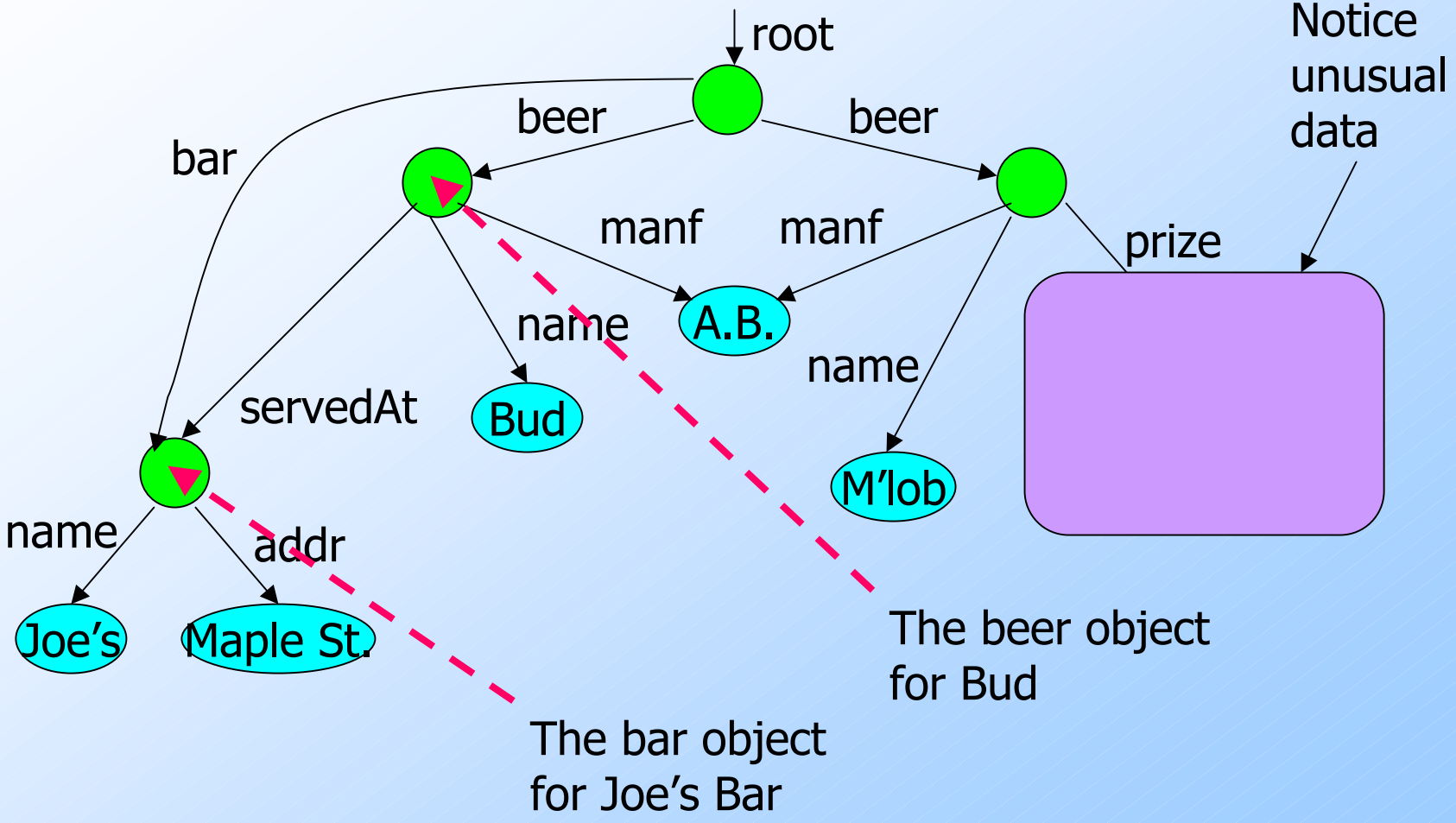
Semistructured Data

- A data model that is suited for integrating (slightly) incompatible sources.
- Based on labeled graphs.
- Key attribute: flexibility --- there is no schema; sources do not all need to have the same attributes.

Semistructured Data --- (2)

- Use semistructured data in place of the global schema.
  - Easier to translate sources with varying local schemas into one flexible schema.
- XML and its attendant standards (XSL, XQUERY, etc.) are really an implementation of semistructured data.

Example: Semistructured Data



XML and Semistructured Data

- XML (Extensible Markup Language) uses a semistructured data model to represent documents. Example:

```
<BARDOC><BAR><NAME>Joe's</NAME>
    <ADDR>Maple St.</ADDR></BAR>
<BAR> … </BAR> …
</BARDOC>
```

Semistructured Data and Logic

- You can represent a semistructured data graph (or XML document) as relations or predicates:
  - arcs(From, To, Label)
  - data(Node, Value)
- But queries about paths in the graph become complex joins.

More Likely Alternative

- Store XML documents as strings, either independent or as components of tuples.
- But the problem of integrating into a sensible whole remains.
- So does the problem of deciding the best way to answer a query.

View-Centric Mediation

- Key assumptions:
  1. There is a set of global predicates that define the schema.
     - These do not exist as stored relations.
  2. 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:

emp(E) = "E is an employee."

phone(E,P) = "P is a phone of E."

office(E,O) = "O is an office of E."

mgr(E,M) = "M is E's manager."

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

  - Interpretation: "every triple (*e,p,m*) at S1 is an employee, one of their phones, and their manager."
  - It does <u>not</u> 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?

1. There must be no other solution containing *S*.
2. *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) :- p(A,X) & q(X,B)
```

1. Find unique variables for the local variables of the view (those that appear only in the body).
2. Substitute variables of the subgoal for variables of the head.
3. 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->X, B->Y; yields body:

```
p(X,X1) & q(X1,Y)
```

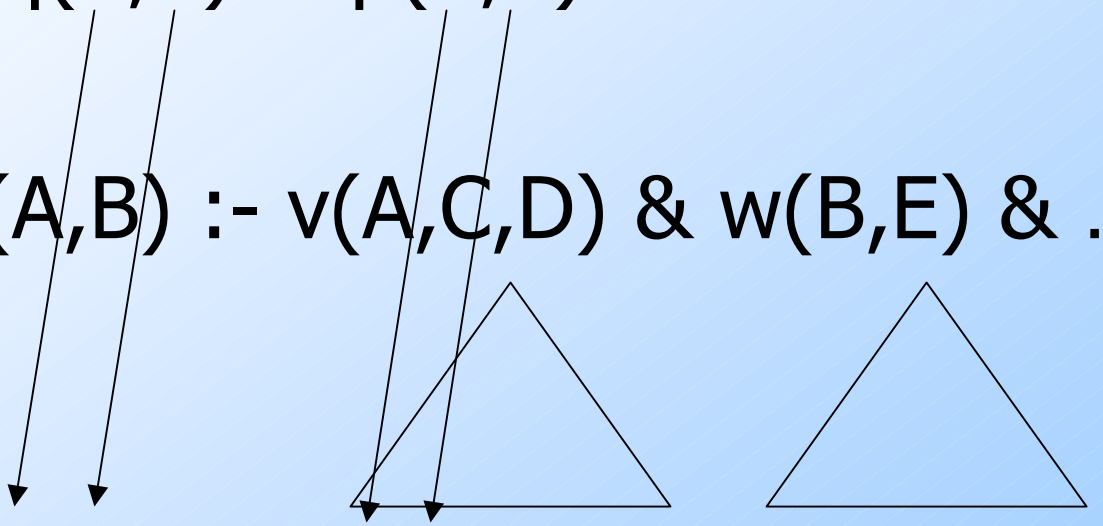
Important Points

- 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) & …

Soln: q(A,B) :- v(A,C,D) & w(B,E) & …



Exp: q(A,B) :- p(A,U) & … & r(B,V) & …

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:

```
v1(X,Y) :- p(X,Y)
v2(X,Y) :- p(X,Y)
```

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

Example --- Continued

- Then, the two solutions:

```
s1(X,Y) :- v1(X,Y)
s2(X,Y) :- 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 -> 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')
```

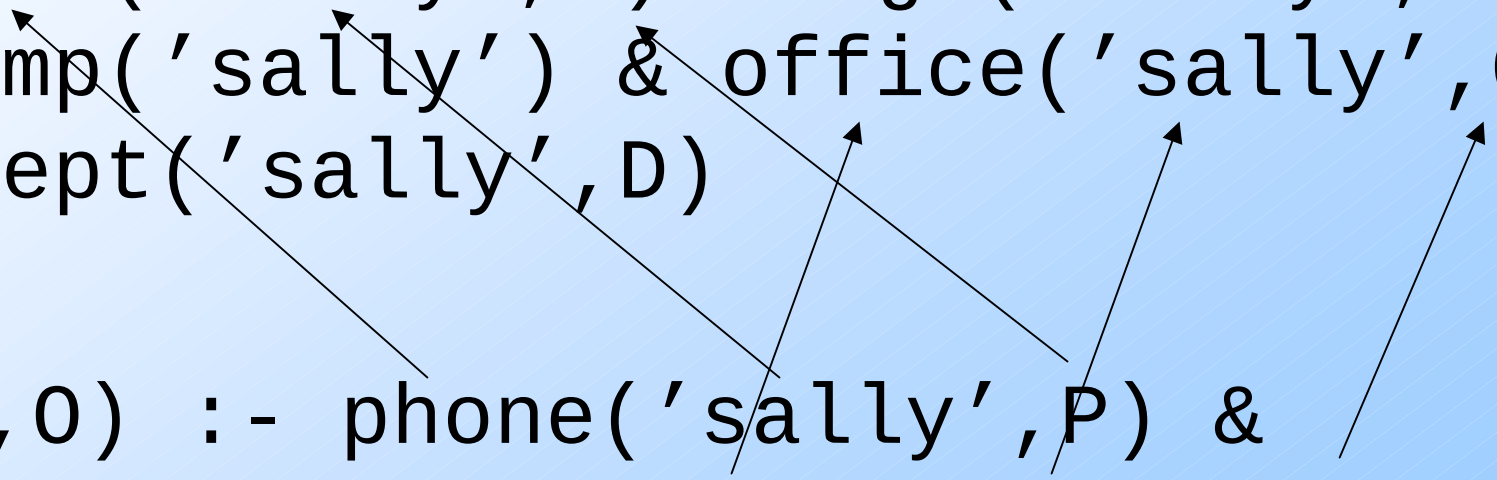
Office Example --- Solutions

```
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 q1 to s2 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.

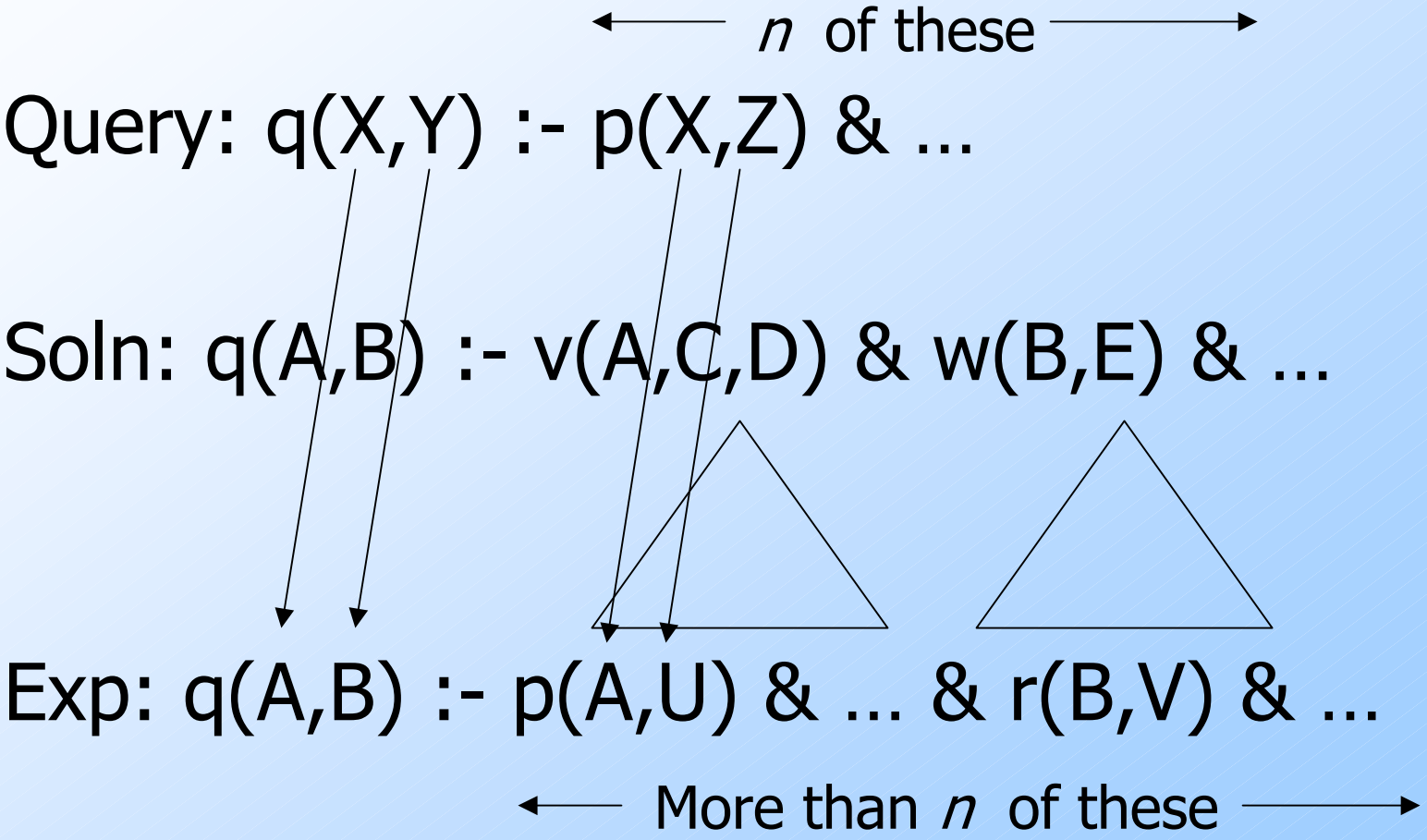
Finding All Solutions to a Query

- 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



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.