Information Integration

Problem: Related data exists in many places and in many forms. They talk about the same things, but differ in the model, schema, or terminology.

Goal: To provide a uniform interface to a multitude of data sources.

Three Approaches

1. **Federated Databases**: The sources are independent, but one source can call the other.
2. **Warehousing:** Make copies of information at each data source centrally.
   - Reconstruct data at regular intervals (daily/weekly/monthly), but it is never up-to-date.
3. **Mediation:** Create a view of all information, but do not make copies.
   - Answer queries by sending appropriate queries to the sources.

Mediator Approach

- Users pose queries in terms of a mediated schema.
- There must be some description of the relationship between the source relations and the mediated schema.
- The query processor must be able to reformulate a query posed in terms of the mediated schema into a query against the source schemas.
- Use a restricted form of first-order logic;

Conjunctive Queries

$$q(\bar{X})\ :-\ e_1(\bar{X}_1), \ldots, e_n(\bar{X}_n)$$

where $e_1, \ldots, e_n$ are database relations, and $\bar{X}_1, \ldots \bar{X}_n$ are database relations and $\bar{X}_1, \ldots, \bar{X}_n$ are tuples of variables and constants.

Queries with unions are expressed by multiple rules with the same head predicate.

A view refers to a named query, and it is said to be materialized if its results are stored in the database.

Query Containment and Equivalence

A query Q_1 is said to be contained in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$ if for any database $\mathbf{D}$, $Q_1(\mathbf{D} \subseteq Q_2(\mathbf{D}$.

How do we express the equivalence of two queries.

The Problem

- Need a description of the relation between the source relations and the global relations. Two main approaches.
- Need to rewrite the user query expressed in the mediated schema into a query expressed in the source schema.

  So, given such a query Q, find a query q' that uses only the source relations, such that:

  - $Q' \models Q$
  - Q' provides all possible answers to Q given the sources

Global as View

GAV: For each relation `R` in the mediated schema, we write a query over the source relations specifying how to obtain `R`'s tuples from the sources.

Example: We have two sources DB1 and DB2 containing titles actors and years of movies

```
MovieActor(title, actor) <--
             DB1(id, title, actor, year)

MovieActor(title, actor) <--
             DB2(id, title, actor, year)
```

If we then add a third source DB3 that provides movie reviews, we might add:

```
MovieReview(title, review) <--
         DB1(id, title, actor, year) AND
                 DB3(id, review)
```

Queries

Find reviews for movies starring Marlon Brando:

```
q(title, review) :-
        MovieActor(title, ‘Brando’) AND
        MovieActor(title, review).
```

Unfolding the descriptions of `MovieActor` and `MovieReview` will yield the following queries over the source relations:

```
q(title, review) :-
     DB1(id, title, ‘Brando’, year) AND
     DB3(id, review)


q(title, review) :-
         DB1(id, title, ‘Brando’, year) AND
         DB2(id, title, ‘Brando’, year) AND
              DB3(id, review)
```

The second clause is clearly redundant.

Local as View

LAV: The contents of each data source are described as a query over the mediated schema.

Example: Suppose we have two sources: (1) V1 containing the titles, years, and directors of Amercian comedies produced after 1960 and (2) V2 containing movie reviews produced after 1990.

```
V1(title, year, director) -->
         Movie(title, year,director,genre) AND
         American(director) AND
         year >= 1960 AND genre = ’Comedy’.

V1(title, review) -->
         Movie(title, year,director,genre) AND
         year >= 1990 AND
         MovieReview(title, review).
```

Queries

Find reviews of commedies produced after 1950:

```
q(title, review) :-
        Movie(title,year,director,'Comedy') AND
        year >= 1950 AND
        MovieReview(title, review).
```

Unfolding

```
q'(title, review) :-
      V1(title, year, director)  AND
      V2(title,review)
```

The reformulated query is not equivalent to original.

Comparison

- **GAV**
  - Query Reformulation is very simple.
  - Adding sources is more difficult.
- **LAV**
  - Adding sources is easy.
  - Query reformulation is difficult.

Systems

- **GAV**
  - TSIMMIS (Stanford)
  - HERMES (University of Maryland)
- **LAV**
  - Information Manifold (AT&T)
  - InfoMaster (Stanford)
  - Tukwila (University of Washington)