Overview

- Approaches to data integration
  - Centralized, virtual data integration
    - Providing a unified and transparent view over a collection of heterogeneous sources
    - Use mappings between the sources to do querying
    - No explicit copying of data out of sources
    - We will only cover this here...
  - Data warehousing
    - Data copied into a centralized system, and made to conform to a schema
  - P2P data integration
    - Significantly more decentralized
- Very nice tutorials at: [DEIS’10]
Key Challenges

- Source: Tutorial by M. Lenzerini
- Data extraction, cleaning, and reconciliation
- How to model and specify the global schema
  - Relational, XML, Graph, or RDF, etc.
- How to discover and specifying the mappings between sources and global schema
- How to answer queries against global schema
- Limitations in mechanisms for accessing sources
- Query optimization
### Example

| Source 1: Used cars for sale. | Accepts as input a category or model of car, and optionally a price range and a year range. For each car that satisfies the conditions, gives model, year, price, and seller contact information. |
| Source 2: Luxury cars for sale. All cars in this database are priced above $20,000 | Accepts as input a category of car and an optional price range. For each car that satisfies the conditions, gives model, year, price, and seller contact information. |
| Source 3: Vintage cars for sale (cars manufactured before 1950). | Accepts as input a model and an optional year range. Gives model, year, price, and seller contact information for qualifying cars. |
| Source 4: Motorcycles for sale. | Accepts as input a model and an optional price range. Gives model, year, price, and seller contact information. |
| Source 5: Car reviews database. Contains reviews for cars manufactured after 1990. | Accepts as input a model and a year. Output is a car review for that model and year. |
Example

**Example 2.2** The following query asks for models, prices, and reviews of sportscars for sale that were manufactured no earlier than 1992 (query \( Q \) of Example 1.1):

\[
q(m, p, r) \leftarrow \text{CarForSale}(c), \text{Category}(c, \text{sportcar}), \text{Year}(c, y), y \geq 1992, \text{Price}(c, p), \text{Model}(c, m), \text{ProductReview}(m, y, r)
\]

- **Option 1:**
  - Ask Source 1 for models and prices of all sportscars manufactured after 1992.
  - For each model, get a review from Source 5

- **Option 2:**
  - Ask Source 2 for models, year, prices.
  - Select where year \( \geq 1992 \)
  - For each model, get a review from Source 5
Overview

- Challenges:
  - How to maintain the information about sources
  - How to incorporate constraints on queries that can be asked

- Information Manifold System
  - Declaratively specify contest and query capabilities of the sources
  - Efficient algorithm that uses source descriptions to create query plans
  - No attempt to solve *entity resolution* problem
  - Relational model
Contents:

- Would prefer not to change the “world view” (global schema) when new source added
- Descriptions should be “tight” (describe the source at a fine-grained level)
- Solution: LAV

Capabilities:

- $S_{in}$: minimal set of parameters that must be specified
- $S_{out}$: parameters that can be returned by the source
- $S_{sel}$: selections that can be applied
- Only one capability record assigned to a source
  - In general, there could be more (later work)
### Describing Information Sources

<table>
<thead>
<tr>
<th>Source 1: Used cars for sale.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents: ( V_1(c) \subseteq \text{CarForSale}(c), \text{UsedCar}(c) )</td>
</tr>
<tr>
<td>Capabilities: ( { { \text{Model}(c), \text{Category}(c) } }, { { \text{Model}(c), \text{Category}(c), \text{Year}(c), \text{Price}(c), \text{SellerContact}(c) } }, { \text{Year}(c), \text{Price}(c) } } ), 1, 4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source 2: Luxury cars for sale. All cars in this database are priced above $20,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents: ( V_2(c) \subseteq \text{CarForSale}(c), \text{Price}(c,p), p \geq 20000 )</td>
</tr>
<tr>
<td>Capabilities: ( { { \text{Category}(c) } }, { { \text{Model}(c), \text{Category}(c), \text{Year}(c), \text{Price}(c), \text{SellerContact}(c) } }, { \text{Price}(c) } } ), 1, 3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents: ( V_3(c) \subseteq \text{CarForSale}(c), \text{Year}(c,y), y \leq 1950 )</td>
</tr>
<tr>
<td>Capabilities: ( { { \text{Model}(c) } }, { { \text{Model}(c), \text{Category}(c), \text{Year}(c), \text{Price}(c), \text{SellerContact}(c) } }, { \text{Year}(c) } } ), 1, 2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source 4: Motorcycles for sale.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents: ( V_4(c) \subseteq \text{Motorcycle}(c) )</td>
</tr>
<tr>
<td>Capabilities: ( { { \text{Model}(c) } }, { { \text{Model}(c), \text{Year}(c), \text{Price}(c), \text{SellerContact}(c) } }, { \text{Price}(c) } } ) , 1, 2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source 5: Car reviews database. Contains reviews for cars manufactured after 1990.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents: ( V_5(m,y,r) \subseteq \text{Car}(c), \text{Model}(c,m), \text{Year}(c,y), \text{ProductReview}(m,y,r) )</td>
</tr>
<tr>
<td>Capabilities: ( { { m, y } }, { m, y, r }, { } } ), 2, 2)</td>
</tr>
</tbody>
</table>

Figure 2: Source descriptions for the sources in Figure 1.
Query Execution

- How to specify a query plan?
- Example:

**Example 3.1** Consider our query asking for sports cars manufactured in 1992 or later:

\[
q(m, p, r) \leftarrow CarForSale(c), Category(c, sportscar), \\
\text{Year}(c, y), y \geq 1992, Price(c, p), \\
\text{Model}(c, m), ProductReview(m, y, r)
\]

The following is a semantically correct plan:

\[
P_1 : Q(m, p, r) \leftarrow \\
V_1(c) (\{\text{Category}(c) : \text{sportscar}\}, \{\text{Price}(c), \text{Model}(c)\}, \\
\{\text{Year}(c) \geq 1992\}), \\
V_5(m, y, r) (\{m : \text{Model}(c), y : \text{Year}(c)\}, \{r\}, \{\}).
\]

To see why, we can verify that the expansion query \(P'_1\) of \(P_1\) obtained by unfolding the augmented descriptions of \(V_1\) and \(V_5\) is contained in the original query:

\[
P'_1 : Q(m, p, r) \leftarrow \text{CarForSale}(c), \text{UsedCar}(c), \\
\text{Model}(c, m), \text{Category}(c, t), t = \text{sportscar}, \text{Year}(c, y), \\
\text{Price}(c, p), \text{ProductReview}(m, y, r), y \geq 1992. \square
\]
Algorithm for Answering Queries

- Problem similar to that of “answering queries using views”
  - Known to be NP-Complete
- Proposed algorithm:
  - For each “subgoal” in the query, find the source relations that can provide that information
  - For every possible combinations of relations across subgoals, check if the plan is semantically correct
  - For semantically correct plans, try to see if it is “executable” (given the capabilities)
Mappings

- **Source:** Tutorial by M. Lenzerini
- **GAV (Global-as-view) vs LAV (Local-as-view) vs GLAV**
- **Imagine:**
  - Sources:
    - \( r_1(\text{Title, Year, Director}) \) since 1960, european directors
    - \( r_2(\text{Title, Critique}) \) since 1990
  - Global schema:
    - \( \text{movie(Title, Year, Director), european(Director), review(Title, Critique)} \)
- **We can do:**
  - (1) Specify how to go from sources to global schema (GAV)
    - \( r_1(\text{Title, Year, Director}) \rightarrow \text{movie(Title, Year, Director)} \)
    - \( r_1(\text{Title, Year, Director}) \rightarrow \text{european(Director)} \)
    - \( r_2(\text{Title, Critique}) \rightarrow \text{review(Title, Critique)} \)
  - (2) Or: specify the sources as subsets of the global schema (LAV)
    - \( \text{movie(t, y, d), european(d), y \geq 1960 \rightarrow r_1(t, y, d)} \)
    - \( \text{movie(t, y, d), review(t,r), y \geq 1990 \rightarrow r_2(t, r)} \)
- Some fundamental differences w.r.t. how to answer queries
- Information Manifold (paper reading) is an example of LAV)
Data Cleaning

- Many issues with correlating data across sources
- Entity resolution (also called de-duplication):
  - Same entity referred to differently in different sources
    - Misspellings, Acronyms, Transformations, Abbreviations, etc.
    - Different formats: email address vs person name
- Very active research area
Schema Matching

- From "Corpus-based Schema Matching; Madhavan et al.; ICDE 2005"
- Goal: identifying corresponding elements in different schemas
  - As a pre-processing step to generating "schema mappings"
- Challenging task
  - Exact semantics of the data often only understood by the designers of the schema
  - Base techniques:
    - Linguistic matching of names for elements
    - Detecting overlap in the choice of data types and representation of data values
    - Considering pattern in relationships between elements
    - Using domain knowledge
Schema Matching

E. Rahm, P.A. Bernstein: A survey of approaches to automatic schema matching

Fig. 2. Classification of schema matching approaches

Further criteria:
- Match cardinality
- Auxiliary information used

Sample approaches