Querying Heterogeneous Information Sources Using Source Descriptions

VLDB 1996

Alon Y. Levy – AT&T Laboratories
Anand Rajaraman – Stanford University
Joann J. Ordille – Bell Labs

Presentation By: Mirza Beg

Outline

- Problem Description
- Proposed System
- System Architecture
- Description of System Modules
- Algorithms
- Experiments & Results
- Discussion

Problem Statement

- Increasing number of structured data sources
- Interrelated data
- The user interacts with each information source separately and combine data !

Alternatively :

- How do we extract the relevant data for a given query ?

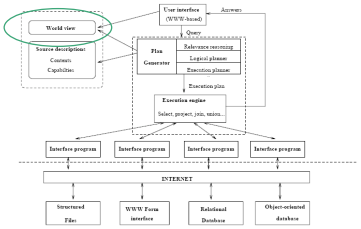
Solution

A System that:

- Provides a uniform query interface to distributed structured sources
- Uses source descriptions to describe data sources
- Generates executable query plans
- Returns the merged result set to the user

INFORMATION MANIFOLD

Information Manifold Architecture

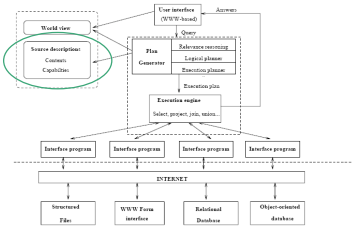


Information Manifold World View

- A virtual global schema on which the user can pose queries

Product {Model}
- **Automobile** {Model, Year, Category}
  - **Car** {Model, Year, Category}
    - **NewCar** {Model, Year, Category}
    - **UsedCar** {Model, Year, Category}
    - **CarForSale** {Model, Year, Category, SellerContact}
  - **Motorcycle** {Model, Year}

Information Manifold Source Descriptions



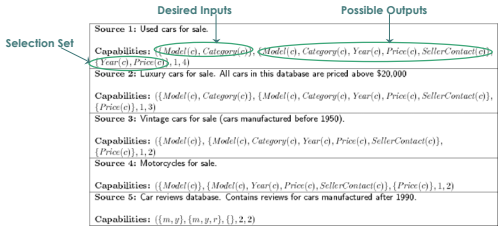
Source Descriptions for Auto Sources

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.

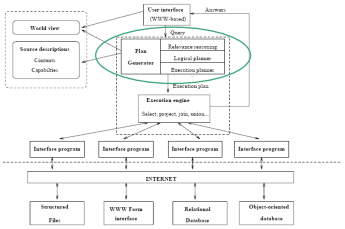
Content Records of Auto Sources

Source 1: Used cars for sale. **Contents:** $V_1(c) \subseteq CarForSale(c), UsedCar(c)$
Source 2: Luxury cars for sale. All cars in this database are priced above $20,000 **Contents:** $V_2(c) \subseteq CarForSale(c), Price(c,p), p \geq 20000$
Source 3: Vintage cars for sale (cars manufactured before 1950). **Contents:** $V_3(c) \subseteq CarForSale(c), Year(c,y), y \leq 1950$
Source 4: Motorcycles for sale. **Contents:** $V_4(c) \subseteq Motorcycle(c)$
Source 5: Car reviews database. Contains reviews for cars manufactured after 1990. **Contents:** $V_5(m,y,r) \subseteq Car(c), Model(c,m), Year(c,y), ProductReview(m,y,r)$

Capability Records of Auto Sources



Information Manifold Plan Generator



Query Reformulation Steps

$$Q(\overline{X}) \leftarrow R_1(\overline{Z}_1),\ \ldots,\ R_n(\overline{Z}_n), C_Q$$

- Prune irrelevant sources
- Split query into sub goals
- Generate conjunctive query plans
- Find an executable ordering of sub goals

Step 1. Bucket Algorithm

Algorithm CreateBuckets$(\mathcal{V},Q)$
Inputs: $\mathcal{V}$ is a set of content descriptions, and Q is a conjunctive query of the form $Q : Q(\bar{X}) \leftarrow R_1(\bar{X}_1), \ldots, R_m(\bar{X}_m), C_Q$.

Set $Bucket_i$ to $\emptyset$ for $1 \leq i \leq m$.
For $i = 1, \ldots, n$ do:
  For each $V \in \mathcal{V}$
    Let V be of the form:
    $V(\bar{Y}) \subseteq S_1(\bar{Y}_1), \ldots, S_n(\bar{Y}_n), C_V$
    For $j = 1, \ldots, n$ do
      If $R_i = S_j$ or R_i and S_j are nondisjoint classes
        Let ψ be the mapping defined on the variables of V as follows:
        If y is the j'th variable in $\bar{Y}_j$ and $y \in Y$
          then $\psi(y) = x_j$, where x_j is the j'th variable in $\bar{X}_i$.
          else $\psi(y)$ is a new variable that does not appear in Q or V.
        Let Q' be the 0-ary query:
        $Q' \leftarrow R_1(\bar{X}_1), \ldots, R_m(\bar{X}_m), C_Q, S_1(\psi(\bar{Y}_1)), \ldots, S_n(\psi(\bar{Y}_n)), \psi(C_V)$
        If $Satisfiable(Q)$ then add $\psi(V)$ to $Bucket_i$.
End.

Step 1. Bucket Algorithm

Given a query Q:

- Find a relevant source
- Create a bucket for this sub-goal
- Check source for Satisfiability
- Add information source to bucket for this sub-goal

Example: Contents and Capabilities

Source 1: Used cars for sale. **Contents:** $V_1(c) \subseteq CarForSale(c), UsedCar(c)$ **Capabilities:** $(\{Model(c), Category(c)\}, \{Model(c), Category(c), Year(c), Price(c), SellerContact(c)\}, \{Year(c), Price(c)\}, 1, 4)$
Source 2: Luxury cars for sale. All cars in this database are priced above $20,000 **Contents:** $V_2(c) \subseteq CarForSale(c), Price(c, p), p \geq 20000$ **Capabilities:** $(\{Model(c), Category(c)\}, \{Model(c), Category(c), Year(c), Price(c), SellerContact(c)\}, \{Price(c)\}, 1, 3)$
Source 3: Vintage cars for sale (cars manufactured before 1950). **Contents:** $V_3(c) \subseteq CarForSale(c), Year(c, y), y \leq 1950$ **Capabilities:** $(\{Model(c)\}, \{Model(c), Category(c), Year(c), Price(c), SellerContact(c)\}, \{Price(c)\}, 1, 2)$
Source 4: Motorcycles for sale. **Contents:** $V_4(c) \subseteq Motorcycle(c)$ **Capabilities:** $(\{Model(c)\}, \{Model(c), Year(c), Price(c), SellerContact(c)\}, \{Price(c)\}, 1, 2)$
Source 5: Car reviews database. Contains reviews for cars manufactured after 1990. **Contents:** $V_5(m, y, r) \subseteq Car(c), Model(c, m), Year(c, y), ProductReview(m, y, r)$ **Capabilities:** $(\{m, y\}, \{m, y, r\}, \{\}, 2, 2)$

Bucket Algorithm: Example

$$q(m_1, p_1, r_1) \leftarrow CarForSale(c_1), Category(c_1, \text{sportscar}), Year(c_1, y_1), y \geq 1992, Price(c_1, p_1), Model(c_1, m_1), ProductReview(m_1, y_1, r_1)$$

$$V_1'(m, t, y, p, s) \subseteq CarForSale(c), UsedCar(c), Model(c, m), Category(c, t), Year(c, y), Price(c, p), SellerContact(c, s)$$

find the mapping $c \rightarrow c_1$

$$CarForSale(c_1), Category(c_1, \text{sportscar}), Year(c_1, y_1), y_1 \geq 1992, Price(c_1, p_1), Model(c_1, m_1), ProductReview(m_1, y_1, r_1), UsedCar(c_1), SellerContact(c_1, s)$$

Source 1 is added to $bucket_1$
Source 2 is added to $bucket_2$
Source 3 does not get added because $(y \leq 1950, y \geq 1992)$ is not satisfiable

Step 2. Finding an Executable Ordering

- Considering all possible combinations of information sources, enumerate semantically correct plans

Step 2. Algorithm for finding an Executable Ordering

- Maintain a list of available parameters
- At every point add to the ordering any sub-goal whose input requirements are satisfied
- Push as many selections as possible to the sources

Step 3. Checking Containment

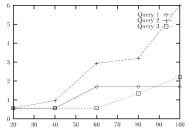
- Minimize each plan by removing redundant sub-goals

Experimental Results

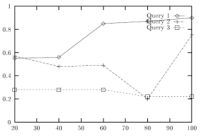
Query	Number of sources	Max. bucket size	Plans enumerated	Plans generated	Time per plan (sec.)	Total time (sec.)
1	20	1	7	1	0.55	0.55
	40	1	7	1	0.56	0.56
	60	2	26	2	0.85	1.70
	80	2	26	2	0.85	1.70
	100	2	26	2	0.85	1.70
2	20	2	7	1	0.57	0.57
	40	3	11	2	0.48	0.96
	60	5	35	6	0.49	2.95
	80	6	44	8	0.40	3.20
	100	7	72	8	0.75	6.00
3	20	2	8	2	0.28	0.56
	40	2	8	2	0.28	0.56
	60	2	8	2	0.28	0.56
	80	6	49	6	0.22	1.32
	100	10	120	10	0.22	2.20

Query 1: Find titles and years of movies featuring Tom Hanks
Query 2: Find titles and reviews of movies featuring Tom Hanks
Query 3: Find telephone number(s) for Alaska Airlines

Experimental Results (cont.)



Total query planning time in seconds versus number of information sources.



Average time per plan in seconds versus number of information sources.

Conclusions

- A novel system that provides a DB-like query interface to distributed structured information sources
- Frees the user from interacting with each information source individually
- Integrates data from multiple sources and filters information
- Information Manifold applicable to WWW and company-wide d-DB's

Open Questions

- How to automatically extract contents and capabilities from sources ?
- Are there better algorithms to determine the relevant sources ?
- Scalability ?
- Overall Performance issues ?

Discussion Points

- A foundational paper in web-data mining.
- Substantial impact on current integration systems.
- Contents & capabilities at the core of the system yet no proposed generation algorithm.
- Experiments carried out on a very small set of queries.

Questions ?