Learning for Semantic Query Optimization in Information Mediators

Chun-Nan Hsu
Dept of Computer Science & Engineering
Arizona State University
USA
Architecture of information mediators

Human & Computer Users

User Services:
• Query
• Monitor
• Update

Information Integration Service

Mediation

Heterogeneous Data Sources

Unprocessed, Unintegrated Details

Abstracted Information

Text, Images/Video, Spreadsheets
Hierarchical & Network Databases
Relational Databases
Object & Knowledge Bases

Mediator

Wrapper

SQL

ORB

Semantic Integration
Agent/Module Coordination
Translation and Wrapping

Mediation

Human & Computer Users

Mediator

Mediator

Mediator

Mediator

Mediator

Agent/Module Coordination
Translation and Wrapping

Wrapper

Wrapper

Wrapper

Wrapper

Wrapper

Semantic Integration
Agent/Module Coordination
Translation and Wrapping

Abstracted Information

Unprocessed, Unintegrated Details

Mediation

Heterogeneous Data Sources

Text, Images/Video, Spreadsheets
Hierarchical & Network Databases
Relational Databases
Object & Knowledge Bases
Information mediators

- Flexible integration of heterogeneous information sources (databases, texts, web pages etc.)
- Key ideas:
  » users access data through a **domain model**
  » information sources represented by a **source model**
  » the mediator **reformulates** domain model query into source model sub-queries
  » the mediator constructs a **query plan** that determines the orders of data flow and execution to retrieve data
- Enable new applications of information systems
  » E-commerce, global health-care IS, etc.
Query planning in information mediators

- Query: Retrieve seaports deep enough for ship “2701”.

```
retrieve assets@unisys
assets(?ship, ?draft):-
assets(?ship, ?id, ?draft),
id-code = "2701".
```

```
geo@isi
geo(?port, ?name, ?depth):-
seaport(?port, ?name, ?depth)
```

```
join (?draft < ?depth)
```

output
Latest work in information mediators

- IM
  - Levy, Srivastava, Kirk, et al. At AT&T Lab
  - query reformulation, relevant source selections

- TSIMMS
  - Hammer, Garcia-Molina, Papakonstantinou, Ullman at Stanford
  - object-based data modeling

- SIMS
  - Arens, Knoblock, Chunnan Hsu, et al. at ISI of USC
  - flexible query planner, *adaptive semantic query optimizer*
Basic idea of adaptive semantic query optimization

Input Query
Give me all the papers written by “Chunnan”

Optimized Query
Give me all the “AI” papers written by “Chunnan”

R1: If AUTHOR is an “AIer” ⇒ PAPER is “AI” paper
R2: “Chunnan” is an “AIer”
R3: ...

PESTO
Query Optimizer

BASIL
learner/KDDer

Semantic Rules

Databases
Novel features and contributions of PESTO

- Use more expressive relational rules
- Optimize a larger class of queries
  - queries with arbitrary join topology
  - joins with multiple comparand attributes
  - unions, intersections, other set operators
- Therefore...
  - detect more optimization opportunities
  - execute queries faster
- See
  - Hsu & Knoblock 93 (CIKM93)
  - Hsu & Knoblock 97 (Submitted to IEEE TKDE)
Using relational rules in semantic query optimization

- Range rules are propositional
  - IF `seaport(?port-name,?city,?storage,_,_)` \(\land\) `city(?city,“Malta”,_,_)`
  - \(\Rightarrow\) ?storage > 2,000,000

- Relational rules are first-ordered, predicate logic
  - IF `city(?city,?population,_,_)` \(\land\) ?population > 3,000,000
  - \(\Rightarrow\) airport(?airport-name,?city,_,_)

- Relational rules are useful in detecting unnecessary relational joins
  - the dominant cost factor of query execution
Desiderata of learning

Input Query

Semantic Query Optimization

Reformulated Query

Semantic Rules

Learning!

Databases

applicable?

operational?

yield high saving?
Induce alternative query and operational rules

Query Q → Database → Operationalization rule pruning → Semantic rules

Inductive query formation +

Equivalence of Q and q

Alternative Query q
Inductive formation of efficient equivalent query

Database DB:

<table>
<thead>
<tr>
<th>A1 *</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1.8</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0.7</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1.4</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0.6</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>1.6</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>2.8</td>
<td>2</td>
</tr>
</tbody>
</table>

Candidate sub-goals:

<table>
<thead>
<tr>
<th>Candidates</th>
<th>gain</th>
<th>cost</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>?A2=0.7 or 0.6</td>
<td>6</td>
<td>16</td>
<td>0.38</td>
</tr>
<tr>
<td>0.5 &lt; ?A2 &lt; 1</td>
<td>5</td>
<td>16</td>
<td>0.31</td>
</tr>
<tr>
<td>?A2 &lt; 1</td>
<td>5</td>
<td>8</td>
<td>0.62</td>
</tr>
<tr>
<td>?A3 = 2</td>
<td>1</td>
<td>8</td>
<td>0.12</td>
</tr>
<tr>
<td>?A1 = “C”</td>
<td>6</td>
<td>1</td>
<td>6.00*</td>
</tr>
</tbody>
</table>


Induce operational rules

- Induce an equivalent query $Q'$ for $Q$ from data
  $Q(\text{?A1,?A2,?A3}) \leftarrow \text{DB(\text{?A1,?A2,?A3), ?A2 < 1, ?A3 = 2.}$
  $Q'(\text{?A1,?A2,?A3}) \leftarrow \text{DB(\text{?A1,?A2,?A3), ?A1 = “C”.}$

- Equivalence of $Q'$ and $Q$:
  $\text{DB(\text{?A1,?A2,?A3}) \land (\text{?A1 = “C”)}$
  $\iff \text{DB(\text{?A1,?A2,?A3}) \land (\text{?A2 < 1) \land (\text{?A3 = 2)}}$

- Derive Rules:
  $\text{DB(\text{?A1,?A2,?A3}) \land (\text{?A1 = “C”) \implies (?A2 < 1) }$
  $\text{DB(\text{?A1,?A2,?A3}) \land (\text{?A1 = “C”) \implies (?A3 = 2)}$
  $\text{DB(\text{?A1,?A2,?A3}) \land (\text{?A2 < 1) \land (\text{?A3 = 2)}} \implies (\text{?A1 = “C”)}$
Learning relational rules

- Apply *Inductive logic programming* techniques (e.g., FOIL by Quinlan, 1990) in alternative query formation and operationalization

- Key ideas:
  - construct *database sub-goals* (e.g., `db(?x,?y)`) as well as *built-in sub-goals* (e.g., `?x > 100`) as candidates
  - use uniform evaluation heuristics for both types of sub-goals
  - use a join-path graph to assure that resulting rules are valid in operationalization

- See
  - Hsu & Knoblock, 1994, Machine Learning Conference
Novel features and contributions of BASIL

- Learn relational rules
- Adapt to changes of query patterns
- Yield effective rules for optimization
- Yield *ROBUST* rules, so that they will remain valid after database changes
- About robustness of knowledge, See
  - Hsu & Knoblock 1995, KDD Conference
  - Hsu & Knoblock 1996, AAAI Conference
  - Hsu & Knoblock 1997, (invited to submit to new Data Mining / KDD journal)
Dealing with database changes

transactions: insert/ delete/ update

Consistent?
Robustness of knowledge

- Intuitively, robustness can be estimated as
  \[
  \frac{\text{# of database states consistent with the rule}}{\text{# of possible database states}}
  \]

- Alternatively, a rule is \textit{robust} given a current database state if transactions that invalidate the rule are unlikely to be performed.

- New definition of robustness is \(1 - \Pr(t|d)\)
  - \(t\): transactions that invalidate the rule are performed
  - \(d\): database is in the current database state
Robustness estimation

- Step 1: Identify the class of invalidating transactions
- Step 2: Decompose each transaction into local variables based on a **Bayesian network model** of database transactions
- Step 3: Estimate local probabilities using
  - Laplace Law of Succession (Laplace 1820) or
  - m-Probability (Cestnik & Bratko 1991)
- Use information available in a database:
  - transaction log
  - expected size of tables, attribute range, distribution
Step 1: Find Transactions that Invalidate the Input Rule

- **R1**: The latitude of a Maltese Geographic location is greater than or equal to 35.89.

\[
\text{geoloc}(_, _, \text{?country}, \text{?latitude}, _) \land (\text{?country} = "\text{Malta}") \Rightarrow \text{?latitude} \geq 35.89
\]

- **Transactions that invalidate R1:**
  - T1: One of the existing tuples of geoloc with its country = “Malta” is updated such that its latitude < 35.89
  - T2: Insert an inconsistent tuple...
  - T3: Update a tuple whose latitude < 35.89 into “Malta”

- \[\text{Robust}(R1) = 1 - \Pr(t|d)\]

\[= 1 - (\Pr(T1|d) + \Pr(T2|d) + \Pr(T3|d))\]
Step 2: Decompose the Probabilities of Invalidating Transactions

Bayesian network model of rule invalidating transactions

\[
Pr(t|d) = Pr(x_1, x_2, x_3, x_4, x_5|d) = Pr(x_1|d) Pr(x_2| x_3, d) Pr(x_3|x_2, d) Pr(x_4| x_2, d) Pr(x_5| x_4, d)
\]
Step 3: Estimate Local Probabilities

- Estimate local probabilities using *Laplace Law of Succession* (Laplace 1820)
  \[
  \frac{r + 1}{n + k}
  \]

- Useful information for robustness estimation:
  - transaction log
  - expected size of tables
  - information about attribute ranges, value distributions

- When no information is available, use database schema information
Example of Robustness Estimation

- **R1**: `geoloc(_,_,?country,?latitude,_) & (?country = “Malta”) ⇒ ?latitude > or = 35.89

- **T1**: One of the existing tuples of `geoloc` with its country = “Malta” is updated such that its latitude < 35.89
  - p1: update? 1/3 = 0.33
  - p2: `geoloc`? 1/2 = 0.50
  - p3: `geoloc`, country = “Malta”? 4/80 = 0.05
  - p4: `geoloc`, latitude to be updated? 1/5 = 0.20
  - p5: latitude updated to < 35.89? 1/2 = 0.5

- \[ Pr(T1|d) = p1 \times p2 \times p3 \times p4 \times p5 = 0.008 \]

- Pr(T2|d) and Pr(T3|d) can be estimated similarly
Example (cont.): When additional information is available

- Naive
  - p1: update? 1/3 = 0.33

- Laplace
  - p1: update? \( \frac{\text{# of previous updates} + 1}{\text{# of previous transactions} + 3} \)

- m-Probability (Cestnik & Bratko 1991)
  - p1: update? \( \frac{\text{# of previous updates} + m \times \text{Pr(U)}}{\text{# of previous transactions} + m} \)
  - m is an expected number of future transactions
  - Pr(U) is a prior probability of updates
Applying robustness estimation in rule induction

- Learning effective and robust rules

Input query Q

Inductive Learning
(Hsu & Knoblock ML94)

r1 (saving = 10)
r2 (saving = 15)
r3 (saving = 18)

Robustness Estimation/Pruning
(Hsu & Knoblock AAAI96)

r1’ (saving = 10)
(robustness = 0.93)
r2’ (saving = 15)
(robustness = 0.98)
r3’ (saving = 18)
(robustness = 0.94)
Rule maintenance

- Rule Maintenance: Identify and repair inconsistent rules

Semantic rules
R1: 0.98
R2: 0.96
R3: 0.83
R4: 0.62

Semantic rules
consistent  R1: 0.985
R2: 0.92
inconsistent R3: 0.83
R4: 0.62
Finale

- PESTO saves up to 97%, and 41+% on average for simple multi-database query plans
- Higher saving expected for complex, expensive query plans to web sources
- All rules learned automatically by BASIL
- Totally invisible from users
- Will be essential of information mediators like SIMS
- For more information:
  » Chunnan Hsu, PhD Thesis, 1996, U of Southern California
  » mailto: chunnan@asu.edu
  » http://www.isi.edu/sims/chunnan/