The Piazza Project

Presented by:
Mengmeng Liu and Shirley Cohen
CIS 650
Agenda

- Project overview
- Basic model
- Query answering
- Mapping compositions
- Semantic Web connection
- Implementation aspects
Piazza Project Members

AnHai Doan
Oren Etzioni
Steven Gribble
Zack Ives
Alon Halevy
Jayant Madhavan
Peter Mork
Maya Rodrig
Dan Suciu
Igor Tatarinov
Piazza: Peer Data-Management

**Goal:** To enable users to share data across local or wide area networks in an ad-hoc, highly dynamic distributed architecture.

- Peers can:
  - Export base data
  - Provide views on base data
  - Serve as logical mediators for other peers
- Every peer can be both a server and a client.
- Peers join and leave the PDMS at will.
Basic Model of PDMS

• *peer schema and peer relations*
• *stored schema and stored relations*
• Queries are posed over relations from a specific peer schema and be reformulated in terms of stored relations
• Assumptions:
  – Relational data model
  – CQ without comparison predicates
  – Views refer to named queries
A PDMS example

Data integration: 1 mediated schema, $m$ mappings to sources
Peer data management system (PDMS):
- $n$ mediated peer schemas as few as $(n - 1)$ mappings between them – evaluated transitively
- $m$ mappings to stored relations
Schema mappings of PDMS

• Mappings between peer and stored relations
  – Storage descriptions: $A : R \subseteq Q$
  – Inclusion or equality
  – Q is a CQ over the schema of peer A and R is a stored relation at peer A
  – LAV mappings

• Mappings between different peer schemas
  – Peer mappings
  – GAV, LAV, GLAV mappings
Peer mappings

- Inclusion and equality mappings
  - $Q_1(A_1) \subseteq Q_2(A_2)$
  - a semantic mapping by stating that evaluating $Q_1$ over the peers $A_1$ will always produce the subset answers as evaluating $Q_2$ over $A_2$.
  - GLAV mapping

- Definitional mappings
  - Each mapping is a set of datalog rules whose head and body are both peer relations where the head contains one atom and the body contains several atoms.
  - If a peer relation appears only once in the head in a set of rules, it can be written as equalities.
  - Express disjunctions easily
  - Exploit the full power of GAV mappings
Summary of basic model

• A PDMS is specified by
  – A set of peers P1…Pn
  – A set of peer schemas S1…Sm
  – A mapping function from peers to peer schemas
  – A set of stored relations $R_i$ at each $P_i$
  – A set of peer mappings $LN$
  – A set of storage descriptions $DN$

• Benefits of PDMS
  – Scalability, extensibility and decentralization
  – Placement of query is arbitrary
  – Exploit transitive evaluation of semantic mappings
Complexity of query answering

• Query answering: Given a PDMS N, an instance of stored relations D and a query Q, find all certain answers of Q.

• Certain answers: \( \bigcap Q(I) \), \( I \) ranges over all data instances for PDMS N’s peer relations who are consistent with N and an instance D for N’s stored relations.

• Finding all certain answers in PDMS: undecidable!

• Observation: If a PDMS only includes storage descriptions and inclusion peer mappings, and peer mappings are acyclic, then a CQ can be answered in polynomial time. (non-recursive datalog)
Query answering in PDMS

• Query answering $\supseteq$ query rewriting $=$ query reformulation
• Problem: Given a set of peer mappings and storage descriptions and a query $Q$, output $Q'$ in terms of stored relations.
• Evaluating $Q'$ will always only produce certain answers to $Q$. If all certain answers can be found in PTIME, then $Q'$ will produce all certain answers.
• $Q'$ is the maximally-contained rewriting of $Q$. 
Query reformulation algorithm

- Use a rule-goal “tree” to expand the mappings
- Goal nodes are labeled with atoms of the peer relations, rule nodes are labeled with mapping rules.
- Algorithm:
  - Start with schemas being queried
    - Look up mappings, expand
    - Continue iteratively until queries are only over stored relations.
- Mappings in a PDMS may be a combination of LAV, GAV style mappings
  - Applies unfolding for GAV
  - Applies Minicon for LAV
  - What about GLAV?
  - A challenge to interleave them together.
An Example

Query: $Q(a_1, a_2) :- \text{SameProject}(a_1,a_2,p), \text{Author}(a_1,w), \text{Author}(a_2,w)$

Peer Mappings:

- $r_0$: $\text{SameProject}(a_1,a_2,p) \iff \text{ProjMember}(a_1,p), \text{ProjMember}(a_2,p)$
- $r_1$: $\text{CoAuthor}(a_1,a_2) \subseteq \text{Author}(a_1,w), \text{Author}(a_2,w)$

Storage Descriptions:

- $r_2$: $\text{S1}(a,p,s) \subseteq \text{ProjMember}(a,p), \text{Sched}(f,s,e)$
- $r_3$: $\text{CoAuthor}(f_1,f_2) = \text{S2}(f_1,f_2)$
Rule-Goal Tree Expansion

$q: Q(a1, a2) :- \text{SameProject}(a1, a2, p), \text{Author}(a1, w), \text{Author}(a2, w)$
Rule-Goal Tree Expansion

q: Q(a₁, a₂) :- SameProject(a₁,a₂,p), Author(a₁,w), Author(a₂,w)
Rule-Goal Tree Expansion

$q: Q(a1, a2) :- \text{SameProject}(a1, a2, p), \text{Author}(a1, w), \text{Author}(a2, w)$

Peer mappings:

$r0: \text{SameProject}(a1, a2, p) :- \text{ProjMember}(a1, p), \text{ProjMember}(a2, p)$

$r1: \text{CoAuthor}(a1, a2) \subseteq \text{Author}(a1, w), \text{Author}(a2, w)$
Rule-Goal Tree Expansion

q: Q(a₁, a₂) :- SameProject(a₁,a₂,p), Author(a₁,w), Author(a₂,w)

Peer mappings:

r₀: SameProject(a₁,a₂,p) :- ProjMember(a₁,p), ProjMember(a₂,p)

r₁: CoAuthor(a₁,a₂) ⊆ Author(a₁,w), Author(a₂,w)
Rule-Goal Tree Expansion

q: $Q(a_1, a_2) : \text{SameProject}(a_1, a_2, p), \text{Author}(a_1, w), \text{Author}(a_2, w)$

Storage descriptions:

r2: $S1(a, p, s) \subseteq \text{ProjMember}(a, p)$, $\text{Sched}(a, s, \text{end})$

r3: $\text{CoAuthor}(f_1, f_2) = S2(f_1, f_2)$
Rule-Goal Tree Expansion

q: \( Q(a_1, a_2) \) :- \( \text{SameProject}(a_1, a_2, p), \text{Author}(a_1, w), \text{Author}(a_2, w) \)

Storage descriptions:

r2: \( S_1(a, p, s) \subseteq \text{ProjMember}(a, p) \), \( \text{Sched}(a, s, \text{end}) \)

r3: \( \text{CoAuthor}(f_1, f_2) = S_2(f_1, f_2) \)
Rule-Goal Tree Expansion

$q: Q(a_1, a_2) :- \text{SameProject}(a_1, a_2, p), \text{Author}(a_1, w), \text{Author}(a_2, w)$
Rule-Goal Tree Expansion

$q: Q(a1, a2) :- \text{SameProject}(a1,a2,p), \text{Author}(a1,w), \text{Author}(a2,w)$

$Q'(a1,a2) :- \text{S1}(a1,p,\_), \text{S1}(a2,p,\_), \text{S2}(a1,a2)$
$\cup \text{S1}(a1,p,\_), \text{S1}(a2,p,\_), \text{S2}(a2,a1)$
Mapping Compositions

• Problem: combining two schema mappings into a single one
• Composition of schema mappings generalizes composition of queries
• Query composition corresponds to functional mappings
• Composition of queries implemented in most database commercial systems
• Evolution of schema mappings (GAV, LAV, GLAV, constraints)
Compositions in PDMS Setting
Problem Definition

$m_{13}$ is a composition of $m_{12}$ and $m_{23}$ if the certain answers obtained by way of $m_{13}$ for any query in a class of queries $L$ against schema $\sigma 3$ are precisely those that can be obtained by using $m_{12}$ and $m_{23}$ in sequence.
Composition Example

JJ Pickle
- Directory
  - pid
  - name
  - wphone
  - hphone

UT Austin
- Person
  - pid
  - name
- Phone
  - number
  - kind
  - pid

UT System
- Addrbook
  - name
  - address
- Phonebook
  - name
  - phone
Schema Mappings

1. `Directory(pid, _, wphone, _) --> Phone(wphone, “work”, pid)`

2. `Directory(pid, _, hphone) --> Phone(hphone, “home”, pid)`

3. `Directory(pid, name, _) --> Person(pid, name)`

4. `Person(pid, name) --> Addrbook(name, _)`

5. `Person(pid, name), Phone(number, kind, pid) --> Phonebook(name, number)"
Composed Mappings

1. Directory(pid, name, _) --> Person(pid, name)
2. Person(pid, name) --> Addrbook(name, _)
3. Directory(_, name, _) --> Addrbook(name, _)
4. Directory(pid, name, wphone, _) --> Phone(wphone, “work”, pid), Person(pid, name)
4. Phone(number, kind, pid), Person(pid, name) --> Phonebook(name, number) -->
   Directory(_, name, wphone, _) --> Phonebook(name, number)
Example with Infinite Mappings

\[ M_{ab} = \{ a_{rg}(x, y) \subseteq b_r(x, x_1), b_g(x_1, y) \} \]

\[ a_{gg}(x, y) \subseteq b_g(x, x_1), b_g(x_1, y) \} \]

\[ M_{bc} = \{ b_r(x, x_1), b_g(x_1, x_2), b_g(x_2, y) \subseteq c_{rgg}(x, y) \]}

\[ b_g(x, x_1), b_g(x_1, y) \subseteq c_{gg}(x, y) \} \]

\[ a_{gg}(x, y) \subseteq c_{gg}(x, y) \]

(By 2 and 4)
Example with Infinite Mappings

\[ M_{ab} = \{ a_{rg}(x, y) \subseteq b_r(x, x_1), b_g(x_1, y) \} \]  
\[ a_{gg}(x, y) \subseteq b_g(x, x_1), b_g(x_1, y) \} \]  
(1)
(2)

\[ M_{bc} = \{ b_r(x, x_1), b_g(x_1, x_2), b_g(x_2, y) \subseteq c_{rgg}(x, y) \} \]
\[ b_g(x, x_1), b_g(x_1, y) \subseteq c_{gg}(x, y) \} \]  
(3)
(4)

\[ a_{rg}(x, y), a_{gg}(x, y) \subseteq c_{rgg}(x, y) \] (By 1, 2, and 3)

So far so good.
Example with Infinite Mappings

\[ M_{ab} = \{ a_{rg}(x, y) \subseteq b_r(x, x_1), b_g(x_1, y) \} \]  \hspace{1cm} (1)

\[ a_{gg}(x, y) \subseteq b_g(x, x_1), b_g(x_1, y) \} \]  \hspace{1cm} (2)

\[ M_{bc} = \{ b_r(x, x_1), b_g(x_1, x_2), b_g(x_2, y) \subseteq c_{rgg}(x, y) \} \]  \hspace{1cm} (3)

\[ b_g(x, x_1), b_g(x_1, y) \subseteq c_{gg}(x, y) \} \]  \hspace{1cm} (4)

\[ a_{rg}(x, x_1), a_{gg}(x_1, x_2), \subseteq c_{rgg}(x, y_1), c_{gg}(y_1, y_2), \]
\[ \ldots, a_{gg}(x_n, x_{n+1}) \ldots, c_{gg}(y_{n-1}, y_n ) \]

Sequence is infinite.
Inverse Rules

Inverted LAV rules commonly used in data integration systems (like Piazza)

Claim: We can use composition to optimize a set of inverse rules

Example:
LAV mapping: \( \forall xy \; Rxy \rightarrow \exists z \; Sxz, Tzy \)
Skolemized: \( \forall xy \; Rxy \rightarrow Sx \; f(xy), T \; f(x,y) \; y \)
Inverse Rules: \( Sx \; f(x,y) \) : - \( Rxy \)
\( T \; f(x,y) \; z \) : - \( Rxy \)
Universal solution (without chasing): \( Uxy \) : - \( Sxz, Tzy \)
Certain answers: \( Qx \) : - \( Uxy \)
Technical Results

Madhavan, Halevy (VLDB 2003)
• Setting: Query language L is a union of conjunctive queries and mappings are source-to-target and target-to-target dependencies (GLAV mappings)
• Showed that the result of composition may be an infinite set of formulas and proposed algorithms for the cases when composition can be done.

Fagin, Kolaitis, Popa, Tan (PODS 2004)
• Setting: Definition of mapping composition independent of query language
• Showed that full source-to-target constraints are closed under composition, but that embedded source-to-target constraints are not.

Bernstein, Green, Melnick, Nash (VLDB 2006)
• Setting: SQL Server implementation
• Showed that composition can be done efficiently.
What is the Semantic Web?

• Built on top of the regular web
• Several languages for representing information: RDF, RDF Schema, and OWL
• A lot of representation logics
• Some tools exist for implementing pieces of it
The Semantic Web (Alon’s view)

• Sharing structured data at web scale
  – You can pose meaningful queries on web sites.
  – Ontologies provide the *semantic glue*.
  – Internal implementation of web sites left open.

• Agents perform tasks:
  – Query one or more web sites
  – Perform updates (e.g., set schedules)
  – Coordinate actions
  – Trust each other (or not).

• *i.e.*, agents operating on a gigantic heterogeneous distributed database.
Getting there

• Robust infrastructure for querying
  – Peer data management systems.

• Facilitate mapping between different structures. Need tools for:
  – Locating relevant structures
  – Easily joining the semantic web.

• Disconnect between RDF and today’s data providers
  – Piazza maps XML to RDF.
Piazza Mapping Language Example

XML Example

**Source:**
*source.xml*

- authors
  - author*
    - full-name
  - publication*
    - title
  - pub-type

**Target:**
*target.xml*

- pubs
  - book*
    - title
    - author*
      - name
    - publisher*
      - name

```xml
<pubs>
  <book>
    {: $a IN document("source.xml")/authors/author}
    $t IN $a/publication/title,
    $typ IN $a/publication/pub-type
    WHERE $typ = "book" : }
    <title> { $t }</title>
    <author>
      <name> {: $a/full-name :} </name>
    </author>
  </book>
</pubs>
```
# Piazza Mapping Language Example

<table>
<thead>
<tr>
<th>Source: source.xml</th>
<th>Target: target.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>authors</td>
<td>pubs</td>
</tr>
<tr>
<td>author*</td>
<td>book*</td>
</tr>
<tr>
<td>full-name</td>
<td>title</td>
</tr>
<tr>
<td>publication*</td>
<td>author*</td>
</tr>
<tr>
<td>title</td>
<td>name</td>
</tr>
<tr>
<td>pub-type</td>
<td>publisher*</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
</tbody>
</table>

```
<pubs>
  <book piazza:id={$t}>
    {: $a IN document("source.xml")/authors/author
      $t IN $a/publication/title,
      $typ IN $a/publication/pub-type
      WHERE $typ = "book" : }
    <title> {$t }</title>
    <author piazza:id={$t}>
      <name> {: $a/full-name :} </name>
    </author>
  </book>
</pubs>
```
System Architecture

Acknowledgements

• Alon Halevy for slides on Semantic Web
• Zack for slides on rule-goal tree example
• TJ and Greg for insights on Piazza
• Val for suggestions