Querying Heterogeneous P2P XML Databases

Angela Bonifati, Icar CNR (Italy)

Joint work with Elaine Q.Chang, Laks V.S.Lakshmanan and Terence Ho, UBC (Canada)

Motivating example

Consider two schemas of medical centers containing similar and related XML data
Motivating example

- **In one schema** (MON), patients and admissions appear under separate groups and are connected by id/idref links.

- **In the second schema** (BOS), the admissions are grouped by each symptom and the patients appear underneath them.
Differences between the two schemas

**Value differences:**
- ID GDB:123 in schema MON is SP:456 in schema BOS
- Addressed by [Clio, Hyperion] with mapping tables
- Affect relational tables as well as XML data, and many other formats

<table>
<thead>
<tr>
<th>MONPatient</th>
<th>BOSPatient</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>Name</td>
<td>InsName</td>
</tr>
<tr>
<td>ID GDB:123</td>
<td>SP:456</td>
</tr>
</tbody>
</table>

**Structure differences:**
- Admission subtree in schema MON is different from the admission subtree in BOS
- Affect all hierarchical data models (e.g., XML)
- Addressed in Clio [Clio], Piazza [Piazza]
Differences between the two schemas

- **Data/metadata conflicts:**
  - Problem values in MON are represented as tags in BOS
  - *Not considered* in XML data integration thus far

Underlying P2P scenario

- **We consider heterogeneous XML data and schemas in P2P databases**
  - There is no common mediated schema
  - Each peer has a local schema *to be mapped* to the neighbors schemas (i.e. acquaintances)
  - The mapping process has to be kept simple:
    - Either aided by (semi-)automatic mapping tools (e.g. Cupid, COMA)
    - Or manually provided
    - In both cases, the user may want to check the correctness
Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- Semantics of peer queries
  - a novel forward translation
- Query translation algorithm
  - a significant subset of XQuery
  - works against and along the direction of rules
- Experimental evaluation of HePToX¹
- Conclusions and future work

¹ HePToX stands for “Heterogeneous Peers Talk!”

A P2P Network of Heterogeneous Hospitals

Montreal

Peer₁
DTD₁

Problem Date Event

Boston

Peer₂
DTD₂

Admission Coronary Pulmonary

New York

Peerₙ
DTDₙ

ID InsName
Motivations for Health care/Medical scenario

– Given that:
  • (1) patients transfer between hospitals
  • (2) the transfers do not always happen between the same set of hospitals, and
  • (3) there is no global mediated schema

– P2P is a natural choice for health care!

Options for a patient

– Consider a P2P network of hospitals and an unfortunate patient moving among them:

  – Option#1: the patient carries his/her own files and query translation is done manually
    • error-prone
    • unfeasible with several moves and with frequent joins/leaves of peers
Options for a patient

– Option#2: the hospital db admin has a very good insight of the two schemas and manually writes the mappings:
  
  ● not easy to find a person who knows the rule machinery that well!

– Option#3: the hospital db admin provides informal arrows/boxes correspondences w.r.t. a set of acquaintances:
  
  ● made simple for users/applications that do not know the underlying mappings machinery
  ● kept lightweight for a new peer entering the network
Crossing heterogeneous data

- Besides patients:
  - doctors: track patients
  - insurance companies: define the policy for a set of patients
  - etc.

- ...also benefit from crossing heterogeneous data

Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- Semantics of peer queries
  - a novel forward translation
- Query translation algorithm
  - a significant subset of XQuery
  - works against and along the direction of rules
- Experimental evaluation of HePToX
- Conclusions and future work
A set of informal correspondences

- We consider:
  - **Leaf-to-leaf arrows:**
    - *one-to-one arrows*
  - **Boxes:** semantically similar values are grouped together
  - **Leaf-to-boxes arrows:**
    - *special one-to-many arrows*
- We can handle arbitrary one-to-many or many-to-many correspondences in HePToX:
  - *e.g. union*

Two hospital schemas
Semantics of correspondences

- The arrows/boxes do not tell how the source schema is physically transformed into the target schema
- They tell how the data of the source is exchanged with the target [Clio, Hyperion]: Data Exchange Semantics
- As a consequence, there may be source data not represented in the target and viceversa

\[
\begin{align*}
\text{<Event>}
\quad \text{<Problem>}
\quad \text{Coronary}
\quad \text{</>}
\quad \text{</>}
\quad \text{</>}
\quad \text{<Event>}
\quad \text{<Problem>}
\quad \text{Orthopedics}
\quad \text{</>}
\quad \text{</>}
\end{align*}
\]

\[
\begin{align*}
\text{<Admission>}
\quad \text{<Coronary>}
\quad \text{…}
\quad \text{</>}
\quad \text{</>}
\quad \text{<Pulmonary>}
\quad \text{</>}
\end{align*}
\]

Disclaimer

- In HepToX, there are no special correspondences for keys or referential constraints across schemas; these could be encoded as value correspondences with mapping tables [Clio]

- Underlying messages:
  - even an unfamiliar user can provide our intuitive correspondences
  - we can draw complex mappings (e.g. using boxes)
Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- Semantics of peer queries
  - a novel forward translation
- Query translation algorithm
  - a significant subset of XQuery
  - works against and along the direction of rules
- Experimental evaluation of HePToX
- Conclusions and future work

From arrows/boxes to actual mapping rules

Rule #1,
Rule #2,
....
Rule #n

From graphical notation to logical mapping rules
A rule inference algorithm

- **Algorithm Input:** Source and Target Schemas, A Set of Arrows/Boxes
- **Algorithm Output:** A set of Datalog-like mapping rules
- The Algorithm consists of the following steps:
  1. Determine source and target groups of nodes such that each group captures “units” of information
  2. For each source group, build the corresponding tree expression
  3. For each target group, identify all minimal sets of source groups that populate information into the target tree expressions and fill in the rules

- Can be split into a group formation algorithm and a rule construction algorithm

Gathering Nodes into Groups

- Groups are built by identifying group nodes

\[
\begin{align*}
\text{sg1} &= \{\text{MonGenHosp, Patient, Hist, Event, EProblem, EDate}\} \\
\text{sg2} &= \{\text{MonGenHosp, Patient, Treat, Date, Desc, Doc}\} \\
\text{sg3} &= \{\text{MonGenHosp, Admission, Problem, AdminDate, DisDate, @PatRef}\} \\
\text{sg4} &= \{\text{MonGenHosp, Patient, @ID, MedCr#, Name}\}
\end{align*}
\]
Merging Groups (if applicable)

- Given two groups, let $u$ be the least common ancestor, they can be merged
  - if all descendants of $u$ in one group are mandatory single-valued relative to the root and all descendants of $u$ in the other group are mandatory to the root

\[ sg3 = \{\text{MonGenHosp, Admission, Admission}_\text{Problem} \} \]

\[ sg3' = \{\text{MonGenHosp, Admission} \} \]

\[ sg3'' = \{\text{MonGenHosp, Admission, AdminDate, DisDate, } @\text{PatRef} \} \]

\[ sg3 = \{\text{MonGenHosp, Admission, Problem, AdminDate, DisDate, } @\text{PatRef} \} \]

Merging Groups (if applicable)

- Or:
  - if all descendants of $u$ in one group are mandatory single-valued relative to the root

\[ sg1' = \{\text{MonGenHosp} \} \]

\[ sg1'' = \{\text{MonGenHosp, Patient, Hist, Event, EProblem, EDate} \} \]

\[ sg1 = \{\text{MonGenHosp, Patient, Hist, Event, EProblem, EDate} \} \]

LCA
Nr. Target Groups = Nr. Rules!

- Our group formation algorithm ensures that the subgraphs induced by groups are connected.
- How many target groups you have determines how many rules you need for the mapping.
- For the hospital example:
  - \( sg_1 \ldots, sg_2 \ldots, sg_3 \ldots, sg_4 \ldots \) (two are merged)
  - \( tg_1 = \{MassGeneral, Admission, Pulmonary, Coronary, @ID, InsName, Policy#, Enter, Leave, Patient\} \)
  - \( tg_2 = \{MassGeneral, Progress, Symptom, SDate, SDesc\} \)
  - \( tg_3 = \{MassGeneral, Progress, Treatment, TDate, TDesc\} \)
  - \( tg_4 = \{MassGeneral, Progress, @PatRef\} \)
- Pairs of Groups Connected by Arrows:
  - \((sg_3, tg_1), (sg_4, tg_1), (sg_1, tg_2), (sg_2, tg_3), (sg_3, tg_4)\)

From Groups to Tree Expressions

- Any source group induces a tree:
  - Example:
    - MonGenHosp \( \rightarrow \) $Montreal \[Patient \rightarrow $P \[Hist \rightarrow $H \[Event \rightarrow $E, EProblem \rightarrow $P, EDate \rightarrow $D]]\)

- If it is a DAG, we replicate the shared node as many times as necessary to create a tree
  - does not apply to nodes in a box
- Any target group induces a tree, but we do not know the node IDs until the rule body is actually written:
  - Example:
    - MassGeneral \( \rightarrow \) ?? [Admission \( \rightarrow \) ?? [$Tag \rightarrow ?? [$ID \rightarrow ?? , ..., Patient \( \rightarrow \) ??]]]
How a mapping rule would look like

- The mapping language is an adaptation of *SchemaLog*, a syntactic high-order language for querying and restructuring relational data.

- A mapping rule consists of *atoms* of the form:

  \(<\text{HeadOfRule}>\)

  \(\text{Tag}_1 \rightarrow \text{id}_1, \text{Tag}_2 \rightarrow \text{id}_2, \ldots, \text{Tag}_n \rightarrow \text{id}_n\)

  where id may be a variable $v$ or a Skolem function $f(v_1,\ldots,v_n)$

Rule Construction algorithm

- *Algorithm Input*: Source Groups and Target Groups
- *Algorithm Output*: A set of Datalog-like rules

- The Algorithm consists of the following steps:
  
  For each target group $tg$:
  
  (1) Start with rule skeleton $\text{TE}(tg) \leftarrow \text{TE}(sg_1),\ldots,\text{TE}(sg_j)$; fill in the variables that corresponds to leaf positions in $tg$
  
  (2) For root and single-value descendants of it, assign their ids as distinct skolem functions of the root variable
  
  (3) For each internal node, assign its id as skolem function of the single-value descendants. If any of these does not belong to $tg$, track back the arrow to find the $sg_i$ that contains its corresponding node and add $\text{TE}(sg_i)$ to the rule body
Building the rule step by step

- Applying step (1) of Rule Construction Algorithm:

\[
\text{MassGeneral} \rightarrow ??? \quad [\text{Admission} \rightarrow ??? \\
[$\text{AP}/\text{text()}$] \rightarrow ??? \quad \\
[@\text{ID} \rightarrow \$\text{ID}, \text{Policy} \rightarrow \$\text{M}, \\
\text{Enter} \rightarrow \$\text{AD}, \text{Leave} \rightarrow \$\text{DD}, \text{Patient} \rightarrow \$\text{N}]]
\]

\[\leftarrow \text{MonGenHosp} \rightarrow \$\text{Montreal} \quad [\text{Patient} \rightarrow \$\text{P} \\
[@\text{ID} \rightarrow \$\text{ID}, \text{MedCr#} \rightarrow \$\text{M}, \text{Name} \rightarrow \$\text{N}], \\
\text{Admission} \rightarrow \$\text{A} \quad [\text{Problem} \rightarrow \$\text{AP}, \text{AdmDate} \rightarrow \$\text{AD}, \\
\text{DisDate} \rightarrow \$\text{DD}, \text{@PatRef} \rightarrow \$\text{PR}], \text{SPR} = \$\text{ID}]
\]

Building the rule step by step

- Applying step (2) of Rule Construction Algorithm:

\[
\text{MassGeneral} \rightarrow \text{f1}($\text{Montreal}) \quad [\text{Admission} \rightarrow \text{f2}($\text{Montreal}) \\
[$\text{AP}/\text{text()}$] \rightarrow ??? \quad \\
[@\text{ID} \rightarrow \$\text{ID}, \text{Policy} \rightarrow \$\text{M}, \\
\text{Enter} \rightarrow \$\text{AD}, \text{Leave} \rightarrow \$\text{DD}, \text{Patient} \rightarrow \$\text{N}]]
\]

\[\leftarrow \text{MonGenHosp} \rightarrow \$\text{Montreal} \quad [\text{Patient} \rightarrow \$\text{P} \\
[@\text{ID} \rightarrow \$\text{ID}, \text{MedCr#} \rightarrow \$\text{M}, \text{Name} \rightarrow \$\text{N}], \\
\text{Admission} \rightarrow \$\text{A} \quad [\text{Problem} \rightarrow \$\text{AP}, \text{AdmDate} \rightarrow \$\text{AD}, \\
\text{DisDate} \rightarrow \$\text{DD}, \text{@PatRef} \rightarrow \$\text{PR}], \text{SPR} = \$\text{ID}
\]
Building the rule step by step

- Applying step (3) of Rule Construction Algorithm:

\[
\text{MassGeneral} \rightarrow f_1(\text{fモンtréal}) [\text{Admission} \rightarrow f_2(\text{fモンtréal})
\text{\$AP/text()} \rightarrow f_3(\text{\$AP/text()}, \text{\$ID, \$M, \$AD, \$DD, \$N})
[\text{\$ID} \rightarrow \text{\$ID, \text{Policy} \rightarrow \text{\$M,}
\text{Enter} \rightarrow \text{\$AD, \text{Leave} \rightarrow \text{\$DD, \text{Patient} \rightarrow \$N}])]
\]

\[
\text{MonGenHosp} \rightarrow \text{fモンtréal} [\text{Patient} \rightarrow \text{\$P}
[\text{\$ID} \rightarrow \text{\$ID, \text{MedCr#} \rightarrow \text{\$M, \text{Name} \rightarrow \$N}],
\text{Admission} \rightarrow \text{\$A [Problem} \rightarrow \text{\$AP, \text{AdmDate} \rightarrow \text{\$AD,}
\text{DisDate} \rightarrow \text{\$DD, \text{\$PatRef} \rightarrow \text{\$PR}], \text{\$PR} = \text{\$ID}]
\]

How a mapping rule would look like

- Here is one rule for the first target group \((\text{sg3, tg1}), (\text{sg4, tg1})\):

\[
\text{MassGeneral} \rightarrow f_1(\text{fモンtréal}) [\text{Admission} \rightarrow f_2(\text{fモンrtle})
\text{\$AP/text()} \rightarrow f_3(\text{\$AP/text()}, \text{\$ID, \$M, \$AD, \$DD, \$N})
[\text{\$ID} \rightarrow \text{\$ID, \text{Policy} \rightarrow \text{\$M,}
\text{Enter} \rightarrow \text{\$AD, \text{Leave} \rightarrow \text{\$DD, \text{Patient} \rightarrow \$N}])]
\]

\[
\text{MonGenHosp} \rightarrow \text{fモンrtle} [\text{Patient} \rightarrow \text{\$P}
[\text{\$ID} \rightarrow \text{\$ID, \text{MedCr#} \rightarrow \text{\$M, \text{Name} \rightarrow \$N}],
\text{Admission} \rightarrow \text{\$A [Problem} \rightarrow \text{\$AP, \text{AdmDate} \rightarrow \text{\$AD,}
\text{DisDate} \rightarrow \text{\$DD, \text{\$PatRef} \rightarrow \text{\$PR}], \text{\$PR} = \text{\$ID}]
\]
Components of mapping rules

- A Skolem function is bound to the mandatory single-value subelements of a node
- E.g. for the nodes in the box Pulmonary and Coronary:
  \[ f_3(\text{$AP/text()$, $ID$, $M$, $AD$, $DD$, $N}) \]
- If keys are known, we take advantage of them in the arguments of Skolem functions
- The equality $PR=ID$ ensures that the rule is safe
- Binding variables across rules are ‘glued together’

Components of mapping rules

MR#3:

Boston -> f1($Montreal) [[Progress -> f2($ID)]]
[Treatment -> f3($TDate, $TDesc)]
[Date -> $TDate, Desc -> $TDesc]]

\[ \text{Montreal} \rightarrow \text{$Montreal [Patient -> $P,}
[@ID -> $ID, MedCr# -> $M, Name -> $N, Treat -> $T,}
[Date -> $TDate, Desc -> $TDesc], Admission -> $A,}
[Problem -> $AP, AdmDate -> $AD, DisDate -> $DD, @PatRef -> $PR]}
\]

MR#4:

Boston -> f1($Montreal) [[Progress -> f2($ID)]][@PatRef -> $PR]

\[ \text{Montreal} \rightarrow \text{$Montreal [Admission -> $Adm,}
[Problem -> $Prob, AdmDate -> $AD, DisDate -> $DD, @PatRef -> $PR],}$PR = $ID.\]
Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- **Semantics of peer queries**
  - a novel *forward* translation
- Query translation algorithm
  - a significant subset of XQuery
  - works *against and along* the direction of rules
- Experimental evaluation of HePToX
- Conclusions and future work

Query Answering Semantics

Given a mapping m: $\Delta_1 \rightarrow \Delta_2$:

- **Backward Query Translation**:
  - means querying *against* the direction of the mapping m, e.g.
    - for each $D_1$, $Q_2^t(D_1) = Q_2(m(D_1))$

- **Forward Query Translation**:
  - means querying *along* the direction of the mapping m, e.g.
    - for each $D_2$, $Q_1^t(D_2) = \prod_{k} Q_1(m(D_1^k))$ all pre-images $D_1^k$ such that $D_2 = m(D_1^k)$
Query Answering Semantics

- Backward Query Translation:
  - similar to view expansion
  - the easiest of the two

- Forward Query Translation:
  - a key contribution of HePToX
  - not easy since the mapping $m$ may not be invertible

Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- Semantics of peer queries
  - a novel forward translation
- Query translation algorithm
  - a significant subset of XQuery
  - works against and along the direction of rules
- Experimental evaluation of HePToX
- Conclusions and future work
XQuery Translation Algorithm

- On a subset of XQuery corresponding to Joins of Tree Patterns (TP), with return arguments equal to leaf nodes:

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

Steps of the translation algorithm (Forward)

- Each XQuery query is reduced to a join of TP
- Each TP is 1) Expanded, 2) Translated w.r.t. each mapping rule
- The obtained translated TPs then undergo the phases of 3) Stitching and possibly 4) Contraction

TP1: $AP/text()='Coronary'

TP2: $PR=$PID

Join condition

$SPR=$PID

$SAP/text()='Coronary'

$SPID

$SPName

$SPDate/text()='12/25/2003'

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

$SPID

$SPDate

$SPName

$SPID

$SPDate/text()='12/25/2003'

$SPName

**TP1 versus Mapping Rule 1**

MassGeneral -> f1($Montreal) [Admission -> f2($Montreal)]

[$AP/text() -> f3($AP/text()), $ID, $M, $AD, $DD, $N]

[@ID -> $ID, Policy -> $M,
Enter -> $AD, Leave -> $DD, Patient -> $N]]

MonGenHosp -> $Montreal [Patient -> $P]

[@ID -> $ID, MedCr# -> $M, Name -> $N],
Admission -> $Adm [Problem -> $Prob, AdmDate -> $AD,
DisDate -> $DD, @PatRef -> $RefVal]], $RefVal = $ID

$AP/text()='Coronary'

**Expansion**

- Expansion: the TP is enriched with the nodes present in
the rule body (*red nodes are dummy nodes*)

$AP/text()='Coronary'
Translation

- Translation: the TP is translated against the rule head (while paying attention to dummy nodes!)

\[
\begin{align*}
\text{MassGeneral} & \quad f1(\text{Montreal}) \\
\text{Admission} & \quad f2(\text{Montreal}) \\
\text{Problem} & \quad f3(\text{AP/text()}, \text{ID}, \text{M}, \text{SAD}, \text{DD}, \text{N})
\end{align*}
\]

Substitutions

\[
\begin{align*}
\text{A} & \rightarrow \text{Adm} \\
\text{AP} & \rightarrow \text{Prob} \\
\text{PR} & \rightarrow \text{RefVal} \\
\text{ID} & \rightarrow \text{PID}
\end{align*}
\]

TP1 versus Mapping Rule 4

MassGeneral -> f1(\text{Montreal}) [Progress -> f2(\text{ID}) [@PatRef -> \text{PR}]]

\[
\begin{align*}
\text{MonGenHosp} & \rightarrow \text{Montreal} [\text{Admission} -> \text{A} \\
\text{Problem} & \rightarrow \text{Prob}, \text{AdmDate} -> \text{SAD}, \\
\text{DisDate} & \rightarrow \text{DD}, \text{@PatRef} -> \text{PR}], \text{PR} = \text{ID}
\end{align*}
\]
Rule Body (for Mapping Rule 4)

- The rule body builds a tree:

![Tree Diagram]

Expansion

- Expansion: the TP is enriched with the nodes present in the rule body (red nodes are dummy nodes)

![Expanded Tree Diagram]

Substitutions:
- $A$ -> $Adm$
- $AP$ -> $Prob$
- $PR$ -> $RefVal$

$AP/text()='Coronary'
Translation

- Translation: the TP is translated against the rule head (there are no dummy nodes this time)

Stitching

- Stitching: the obtained translated pieces are unified; unification is obtained via identity substitution
**Contraction**

- Contraction: leaf dummy nodes are dropped; internal dummy nodes are dropped if their children are all dummy.

```
MassGeneral
  f1($Montreal)
  Admission
    f2($Montreal)
      @ID
  $AP/text()
    f3($AP/text(), $ID, $M, $AD, $DD, $N)
    MassGeneral
      f2($ID)
      $PR
    $TDate
      f3($TDate/text(), $TDesc/text())
```

**TP1 after stitching and contraction**

```
$AP/text() = 'Coronary'
```

**TP2 after Stitching and Contraction**

- TP2 after translation against MR1, MR3 and MR4, stitching and contraction.

```
$TDate/text() = '12/25/2003'
```

Angela Bonifati, ICAR CNR, Italy

University of Leipzig, Nov.3 2005
The translated XQuery query

Trivial translation:

FOR $L1 IN /MassGeneral,
    $L2 IN $L1/Progress,
    $PatRef IN $L2/@PatRef,
    $PDate IN $L2/Treatment/Date/text(),
    $L3 IN $L1/Admission/Coronary,
    $PName IN $L3/Patient/text(),
    $PID IN $L3/@ID
WHERE $PDate = "12/25/2003"
AND $PatRef = $PID
RETURN $PName

A binding variable for each node in the TP

The translated XQuery query

Smart translation:

FOR $C IN /MassGeneral/Admission/Coronary,
    $P IN /MassGeneral/Progress
WHERE $C/@ID=$P/PatRef AND
    $P/Treatment/Date="12/25/2003"
RETURN {$C/Patient}

A binding variable for each node in the predicate
Correctness of Query Translation

- We have proven that for the considered fragment of XQuery, the query translation algorithm is correct w.r.t. the given semantics.
**Experimental guidelines**

- Each peer is connected to a set of acquaintances
  - Transitive mappings produce semantic paths

- The queries are propagated from a peer to its acquaintances and recursively on

- If multiple paths connect a pair of peers, we preferably choose the shortest path

- Cycles are detected by marking queries with a unique global ID

**Setup of experiments**

- As a P2P routing algorithm we used DHT-based FreePastry

- We emulate a real network with Emulab:
  - 50 real machines with full resources preemption (as opposed to Planetlab)
  - delay and bandwidth to simulate geographical network behavior

- As a query engine for XQuery, we use QIZX [QIZX]
Outline of experiments

- Rule Inference Algorithm
- Query Translation Across the Mappings
- Query Performance
- System Scalability

- All these experiments have been run for both synthetic and realistic datasets
  - XMark documents (10 schema variations)
  - DBResearch documents (19 variants, same used in Piazza)

Queries for XMark scheme

QMC = % of rules traversed by Qi / avg number of rules for each scheme pair

<table>
<thead>
<tr>
<th>Q</th>
<th>Query Description</th>
<th>QMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Selection with 1 filter, Mich. QR3, etc.</td>
<td>11,4%</td>
</tr>
<tr>
<td>Q2</td>
<td>Selection with 2 filters, Mich. QS5, etc.</td>
<td>13,7%</td>
</tr>
<tr>
<td>Q3</td>
<td>Selection with 3 filters, Mich. QS16, etc.</td>
<td>20,7%</td>
</tr>
<tr>
<td>Q4</td>
<td>Selection with 2 filters (1 nested), Mich. QS18, etc.</td>
<td>11,5%</td>
</tr>
<tr>
<td>Q5</td>
<td>Selection with 3 filters (2 nested), Mich. QS34 etc.</td>
<td>11,8%</td>
</tr>
<tr>
<td>Q6</td>
<td>1 Join, Selection with 2 filters, Mich. QJ1, etc.</td>
<td>28,6%</td>
</tr>
<tr>
<td>Q7</td>
<td>1 Join, Selection with 1 filter, Mich. QJ3, etc.</td>
<td>28%</td>
</tr>
<tr>
<td>Q8</td>
<td>3 Joins, Selection with 1 filter, No corresp. Mich.</td>
<td>62,5%</td>
</tr>
<tr>
<td>Q9</td>
<td>6 Joins, Selection with 1 filter, No corresp. Mich.</td>
<td>100%</td>
</tr>
<tr>
<td>Q10</td>
<td>9 Joins, Selection with 1 filter, No corresp. Mich.</td>
<td>100%</td>
</tr>
</tbody>
</table>
Rule Inference Algorithm Performance

- HepToX GUI allows the user to draw a few arrows/boxes and lets us generate the corresponding rules accordingly.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>XMark</th>
<th>DBResearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Nr. Of Arrows</td>
<td>65</td>
<td>7</td>
</tr>
<tr>
<td>Avg Nr. Of Rules</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Rule Generation Time</td>
<td>45ms</td>
<td>25ms</td>
</tr>
</tbody>
</table>

- Tested on a P4machine, with 3.0GHz and 2GB memory.

Query Translation in HePToX

- The query translation time is affected by 3 parameters:
  - Avg number of acquaintances across all sets
  - ‘Degree of heterogeneity’ within the acquaintances
  - Total number of distinct schemas in the network
Avg Query Time VS. Nr. Of Acquaintances

In order to keep the number of translations reasonably low, we have to choose a number of acquaintances of (at least) 4.

Query Translation Time

When number of acquaintances = 4, the query translation time reaches a stable point at almost the same No. of distinct schemas.
Query Composition for Querying XMark

Minimal overhead due to query translation; missing queries (Q8, Q9, Q10) did not actually terminate in the QIZX engine.

Query Completion VS. Timeout

The difference between one common schema and 10 different schemas is negligible.
System Scalability

Query completion follows a regular logarithmic curve; this is with only Pastry (no Emulab)

About HePToX demo at last VLDB

- HePToX has been demonstrated at VLDB05
- The prototype has the following features:
  - Draw mappings and show the generation of rules
  - Show the query translation algorithm at work
  - Show a real network emulation with Emulab

- HePToX is implemented in Java:
  - Uses QizX [QizX], FreePastry and Emulab
  - It consists of ~10,000 lines of code

- Here are some snapshots....
Demo Screenshots 1/2

Schema Mappings By Boxes/Arrows and Corresponding Datalog-like Rules

Demo Screenshots 2/2

Details of Query Translation Algorithm (for each pair <TP, MR>)
Outline

- Application to an Health Care P2P scenario
- Informal specification of mappings
- Novel rule inference algorithm
- Semantics of peer queries
  - a novel forward translation
- Query translation algorithm
  - a significant subset of XQuery
  - works against and along the direction of rules
- Experimental evaluation of HePToX
- Conclusions and future work

Wrapping up

- Previous work has not considered data/metadata interplay for XML data translation
- We have proposed a novel algorithm for inferring precise rules from arrows/boxes on DTDs
- We have devised an efficient query translation algorithm
- We have a new semantics for forward direction of translation
- We have conducted an experimental study to assess HePToX utility and scalability
Future Work

- Incorporating **keys**, or possibly broader functional dependencies
- Dealing with **GTPs**, i.e. extension of TP for XQuery queries
- Studying the **information-preservation** problem within our class of transformations

Grazie
References 1/2

- The HepTox Web Site: http://www.cs.ubc.ca/~laks/heptox.html
- [QIZX] http://www.xfra.net/qizxopen/

References 2/2

If there is any time left...

HePToX versus [Clio]

- Clio is one of the pioneers in schema mapping discovery, translation and maintenance
- Operates on both XML and relational
- Moreover, under Hyperion, they introduce mapping tables

- Groups are similar to primary paths in Clio, which are not able to account for nodes in boxes
- Clio/Hyperion do not handle data/metadata conflicts as in HePToX
- Internal nodes with complex structures across schemas are nicely addressed in HePToX
HePToX versus [Piazza]

- Piazza defines GAV/LAV-style mappings across peers schemas
- The mapping language is XQuery, whereas in our case the mappings are fairly intuitive
- The forward query translation in HePToX resembles query answering using views, but leverages Skolem functions and the special nature of mappings in HePToX

- Piazza does not handle data/metadata conflicts as in HePToX

HePToX versus [Bohannon], [Benedikt]

- [Bohannon] and [Benedikt] can handle all kinds of mappings including recursive ones, but not data/metadata conflicts as in HePToX

- [Bohannon] has node-to-path mappings
HePToX versus [iFuice]

- [iFuice] includes a mediator offering an access to the sources and their mappings
- Mappings are instance-driven
- HePToX does not look at instances when building mappings

HePToX versus [COMA++]

- COMA++ is a comprehensive tool for schema and ontology mappings
  - features for *composing, merging and comparing* different mappings
- It is interesting to see whether
  - COMA++ and HePToX could be cascaded, i.e.
  - mappings in COMA++ could be used in HePToX and if COMA++ can generate HePToX mappings
How to get to the rules: the underlying algebra

- The rules are written by looking at the source and target groups
- What is important is that the rules capture a class of algebraic transformations:
  - UnNest/Nest
  - Flip/Flop
  - Merge/Split

Nesting/UnNesting XML trees

**DTDs:**

- A child/parent relationship is flattened by means of ID/IDREFs.
- Instances are modified accordingly.
Flipping/Flopping XML Trees

- Flip/Flop: Nodes values in a schema become tags into another schema:
  - notice that nodes can be pulled up the hierarchy at an arbitrary level

Instances:

```
Records
  +-----------------+-----------------+-----------------+
  | Patient         | Patient         | Patient         |
  +-----------------+-----------------+-----------------+
  | ID              | ID              | ID              |
  | P1 John          | P2 Jane          | P3 Mary          |
  +-----------------+-----------------+-----------------+
  | Ailment          | Ailment          | Ailment          |
  | Pulmonary        | Coronary         | Pulmonary        |
  +-----------------+-----------------+-----------------+
```

Splitting/Merging XML Trees

- Nodes having a common parent are grouped/ungrouped together:
  - notice that this corresponds to ‘grouping by tag’

DTD:

```
Records
  +-----------------+-----------------+-----------------+
  | Patient         | Patient         | Patient         |
  +-----------------+-----------------+-----------------+
  | ID              | ID              | ID              |
  | P1 John          | P2 Jane          | P3 Mary          |
  +-----------------+-----------------+-----------------+
  | Ailment          | Ailment          | Ailment          |
  | Pulmonary        | Coronary         | Pulmonary        |
  +-----------------+-----------------+-----------------+
```

Instance:

```
Records
  +-----------------+-----------------+-----------------+
  | Patient         | Patient         | Patient         |
  +-----------------+-----------------+-----------------+
  | ID              | ID              | ID              |
  | P1 John          | P2 Jane          | P3 Mary          |
  +-----------------+-----------------+-----------------+
  | Ailment          | Ailment          | Ailment          |
  | Pulmonary        | Coronary         | Pulmonary        |
  +-----------------+-----------------+-----------------+
```
HepTox vs. previous literature

- [Clio, Hyperion, Piazza, Lenzerini] all considered first-order language mappings
  - Clio extended with XML and Piazza consider XML grouping but not data/metadata transformations as HepToX does
- [Fagin] considered the problem of data exchange for FO rules and FO+aggregates/grouping (SQL)
  - they do not say how to generate those higher-order mappings

Capturing Union Rules

- HePToX supports union mappings:
  - two or more source collections can be mapped to the same target collection

\[ r^* \rightarrow s^* \rightarrow \text{tgt} \]

\[ A, B, C, D, E, F \]
Capturing Union Rules

- 1st scenario:
  - \( r \) is auto, \( s \) is car, \( t \) is automobile

- 2nd scenario:
  - \( r \) is euroCar, \( s \) is amCar, \( t \) is automobile
**Capturing Union Rules**

- We introduce the *label mapping tables*:

<table>
<thead>
<tr>
<th>src schema label</th>
<th>op</th>
<th>tgt schema label</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto</td>
<td>=</td>
<td>automobile</td>
</tr>
<tr>
<td>car</td>
<td>=</td>
<td>automobile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>src schema label</th>
<th>op</th>
<th>tgt schema label</th>
</tr>
</thead>
<tbody>
<tr>
<td>euroCar</td>
<td>∈</td>
<td>automobile</td>
</tr>
<tr>
<td>amCar</td>
<td>∈</td>
<td>automobile</td>
</tr>
</tbody>
</table>