Translating Relational Schemas to XML Schemas

Dongwon Lee, Murali Mani, Frank Chiu, Wesley W. Chu
UCLA

Our goals.

• Map from relational to XML model automatically.
• Maintain semantic constraints during the mapping.
• Existing work such as XML Extender (from IBM), db2XML, XML-DBMS, SilkRoute, EXPERANTO rely on user-specified mapping, and do not maintain semantic constraints.
**CoT: Constraint-based Translation: Step 1**

- Consider IND $s[\alpha] \subseteq t[\beta]$, where $\alpha \subseteq X$, $\beta \subseteq Y$, $\beta$ is primary key, and $\alpha$ is non-nullable.

  - If $\alpha$ is unique, $M(t) = (Y, s^?)$, else $M(t) = (Y, s^*)$
  - $M(s) = (X - \alpha)$
  - Key for $s$ is $(K_s - \alpha)$

<table>
<thead>
<tr>
<th>Sname</th>
<th>Advisor</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Muntz</td>
<td>DB</td>
</tr>
<tr>
<td>John</td>
<td>Muntz</td>
<td>N/W</td>
</tr>
</tbody>
</table>

**CoT: Step 2**

- Consider tables, $s$, $t_1$, $t_2$ with column set $X$, $Y_1$, $Y_2$, and INDs $s[\alpha] \subseteq t_1[\beta_1]$, and $s[\gamma] \subseteq t_2[\beta_2]$, where $\beta_1$, $\beta_2$ are primary keys and $\alpha$, $\gamma$ are non-nullable

- Translate one IND as in Step 1, and translate the other to IDREF as:

  $M(t_1) = (Y_1, s^*)$, $M(t_2) = (Y_2)$, $M(s) = (X - \alpha - \gamma)$, $A(t_2) = \{ID_{t_2::ID}\}$, $A(s) = \{Ref_{t_2::IDREF}\}$

<table>
<thead>
<tr>
<th>Cname</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>4760</td>
</tr>
<tr>
<td>N/W</td>
<td>4549</td>
</tr>
</tbody>
</table>
CoT: Step 3

- Consider a relational schema with tables \{t_1, t_2, ..., t_n\} and INDs \{\alpha \subseteq t_i | \beta \subseteq t_j\}
- Construct an IND-graph
- Identify top-nodes as:
  - Nodes that do not have any IND are top-nodes
  - In a strongly connected component formed from table-set, S, if there is no IND from a node in S to a node outside S, then one of the nodes in S must be a top-node.
- Perform BFS and translate the IND as in Step 1 or Step 2.

**IND-Graph**

- Student (Sid, Name, Advisor)
- Emp (Eid, Name, ProjName)
- Prof (Eid, Name, Teach)
- Course (Cid, Title, Room)
- Dept (Dno)
- Proj (Pname)

M (course) = (Cid, Title, Room, prof')
M(prof) = (student')
M(student) = (Sid, Name)
M(emp) = (Eid, Name, dept', proj')
M(dept) = (Dno)
M(proj) = (Pname)

A(emp) = \{ID_emp::ID, Ref_proj::IDREF\}
A(prof) = \{Ref_emp::IDREF\}
A(proj) = \{ID_proj::ID\}
Implementation Grammar syntax

\[
\begin{align*}
\text{root} & \rightarrow \text{\{course \*, emp \*\}} \\
\text{course} & \rightarrow \text{\{Cid, Title, Room, prof \*\}} \\
\text{emp} & \rightarrow \text{\{Eid, Name, dept \*, proj \*\}} \\
\text{@emp} & \rightarrow \text{\{@ID_emp, @Ref_proj\}} \\
\text{prof} & \rightarrow \text{\{student \*\}} \\
\text{@prof} & \rightarrow \text{\{@Ref_emp\}} \\
\text{student} & \rightarrow \text{\{Sid, Name\}} \\
\text{dept} & \rightarrow \text{\{Dno\}} \\
\text{proj} & \rightarrow \text{\{Pname\}} \\
\text{@proj} & \rightarrow \text{\{@ID_proj\}} \\
\end{align*}
\]

\[
\begin{align*}
\text{primKeys (course)} & = \text{Cid} \\
\text{primKeys (emp)} & = \text{Eid} \\
\text{primKeys (prof)} & = \text{@Ref_emp} \\
\text{primKeys (student)} & = \text{Sid} \\
\text{primKeys (dept)} & = \text{Dno} \\
\text{primKeys (proj)} & = \text{Pname} \\
\end{align*}
\]

CoT Implementation

- Converts a set of relations into a single XML document, and a schema for it.
- Document presently constructed using DOM.
- Resulting schema has following features
  - Local tree grammar
  - No recursion
  - An element type corresponding to a relation occurs only once in the grammar
Conclusions

- Automatically map from relational to a good XML model.
- Maintain semantic constraints.
- Remove some of the redundancies that could have been present in the relational model.