Effective Schema Conversion between XML and Relational Models

Wesley W. Chu
With Dongwon Lee and Murali Mani

University of California, Los Angeles

Conversion from XML to Relational

<table>
<thead>
<tr>
<th>Query</th>
<th>Database</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Query</td>
<td>XML doc</td>
<td>&lt;book&gt; &lt;price&gt;10&lt;/price&gt; &lt;name&gt;XML&lt;/name&gt; &lt;book&gt;</td>
</tr>
<tr>
<td>SQL</td>
<td>Knowledge</td>
<td>RDB</td>
</tr>
</tbody>
</table>

query processing query relaxation
## Difficulties

- No 1-to-1 mapping
- Set \((a^*, (b+ | c?))\), Recursion

```xml
<list name="A"><item>1</item><item>2</item> </list>
<list name="B"><item>3</item><item>4</item> </list>
```

- Fragmentation & inlining

```xml
<!ELEMENT A (B|C)>
1. Fragmented: 3 tables A, B, C
2. Inlined: 1 table D
```

## Constraints in DTD

- Domain constraint

```xml
<!ATTLIST author gender (M|F) #REQUIRED
married (Y|N) #IMPLIED>
```

- Cardinality constraint

```xml
<!ELEMENT book (title,author+,ref*,price?)>
- title: must be 1
- author: at least 1
- ref: any occurrence
- price: at most 1
```
Constraints in DTD (cont)

- Inclusion Dependency (IND)

```xml
<!ELEMENT person (name,(email|phone)?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT contact EMPTY>
<!ATTLIST contact aid IDREF #REQUIRED>
<!ELEMENT editor (person*)>
<!ATTLIST editor eid IDREFS #IMPLIED>
```

\[ aid \subseteq id, eid \subseteq id \]

Constraints in DTD (cont)

- Equality-Generating Dependency (EGD)
  - Values in one columns require values in other columns be equal
  - In XML, EGD is disguised as “Singleton” property
  - When an element instance \( x \) of type \( X \) satisfies the singleton property towards its sub-element instances \( y_i \) and \( y_j \) of type \( Y \), \( y_i \) and \( y_j \) must be equal
  - 1-to-\{0,1\} and 1-to-\{1\} cardinality cases

\[ X \rightarrow X \cdot Y \]
Constraints in DTD (cont)

- **Tuple-Generating Dependency (TGD)**
  - Require some tuples of a certain form be **present**
  - In XML, TGD is disguised as “Not-Nullness” property
  - Child property \((P \rightarrow \star C)\): Every element of type \(P\) must have at least one child element of type \(C\)
    - 1-to-{1} and 1-to-{1,...} cardinality cases
  - Parent property \((C \rightarrow \star P)\): Every element of type \(C\) must have a parent element of type \(P\)
    - Only can be enforced by semantic knowledge since any proper element can be a root w/o parent

CPI Algorithm

- Uses structure-oriented translation algorithm (e.g., hybrid inlining algorithm [VLDB 99]) as basis
- Preserves **constraints** during the translation
- Convert DTD to a digraph with annotated edge types
- Identify **top nodes**:
  - source nodes,
  - child of \(*\) and \(+\) nodes, or
  - recursive nodes with indegree>1
- Map top nodes \(T\) to **table** \(t\) (to avoid non-1NF); map leaf nodes reachable from \(T\) to **column** \(c\) of \(t\) via inlining unless \(T\) is another top node
- New columns for bookkeeping
  - \(fk\_key\), \(parent\_elm\), \(root\_elm\), \(ordinal\), …
Conference.dtd

<!ELEMENT conf (title, date, editor?, paper*)>
<!ATTLIST conf    id      ID        #REQUIRED>
<!ELEMENT title   (#PCDATA)>  
<!ELEMENT date    EMPTY>  
<!ATTLIST date    year   CDATA     #REQUIRED  mon    CDATA    #REQUIRED  day    CDATA    #IMPLIED>
<!ELEMENT editor (person*)>  
<!ATTLIST editor  eids IDREFS    #IMPLIED>
<!ELEMENT paper (title, contact?, author, cite?)>  
<!ATTLIST paper id      ID        #REQUIRED>
<!ELEMENT contact EMPTY>  
<!ELEMENT author (person*)>  
<!ATTLIST author  id      ID        #REQUIRED>
<!ELEMENT person (name, (email|phone)?)>  
<!ATTLIST person  id      ID        #REQUIRED>
<!ELEMENT name EMPTY>  
<!ATTLIST name    fn     CDATA     #IMPLIED  ln    CDATA    #REQUIRED>
<!ELEMENT email   (#PCDATA)>  
<!ELEMENT cite (paper*)>  
<!ATTLIST cite    id     ID     #REQUIRED  format (ACM|IEEE) #IMPLIED>

Conference.xml

<conf id="er99">
<title>Int'l Conference on Conceptual Modeling (ER)</title>
<date> <year>1999</year> <mon>May</mon> <day>20</day> </date>
<editor eids="sheth bossy">
  <person id="klavans">
    <name fn="Judith" ln="Klavans"/><email>kla@columbia.edu</email>
  </person>
</editor>
<paper id="p1">
<title>Indexing Model for Structured...</title>
<contact aid="dao"/>
  <author><person id="dao"><name fn="Tuong" ln="Dao"/></person></author>
</paper>
<paper id="p2">
<title>Logical Information Modeling of...</title>
<contact aid="shah"/>
  <author>
    <person id="shah" name fn="Kshitij" ln="Shah"/></author>
  </author>
</paper>
</conf>
Conference.xml (cont)

```
<conf>
  <paper id="p3">
    <title>Making Sense of Scientific Info...</title>
    <author>
      <person id="bossy">
        <name fn="Marcia" ln="Bossy"/><phone>391.4337</phone>
      </person>
    </author>
  </paper>
  <paper id="p7">
    <title>Constraints-preserving Transformation...</title>
    <contact aid="lee"/>
    <author>
      <person id="lee">
        <name fn="Dongwon" ln="Lee"/><email>dongwon@cs.ucla.edu</email>
      </person>
    </author>
  </paper>
</conf>
```

Annotated DTD Graph
CPI Steps

CPI Steps (cont)
id, title cannot be NULL

- cite_id, contact_aid, cite_format can be NULL
- fk_conf is a FK to conf
- fk_cite is included in cite_id (I.e., fk_cite ⊆ cite_id)
- id is a PK
- cite_id is UNIQUE
**Paper Table after CPI**

<table>
<thead>
<tr>
<th>id</th>
<th>root_elm</th>
<th>parent_elm</th>
<th>fk_conf</th>
<th>fk_cite</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>conf</td>
<td>conf</td>
<td>er99</td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td>conf</td>
<td>conf</td>
<td>er99</td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td>conf</td>
<td>cite</td>
<td>c100</td>
<td></td>
</tr>
<tr>
<td>p7</td>
<td>paper</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>title</th>
<th>contact_aid</th>
<th>cite_id</th>
<th>cite_format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing...</td>
<td>dao</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical...</td>
<td>shah</td>
<td>c100</td>
<td>ACM</td>
</tr>
<tr>
<td>Making...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraints...</td>
<td>lee</td>
<td>c200</td>
<td>IEEE</td>
</tr>
</tbody>
</table>

**Schema after CPI**

```sql
CREATE TABLE paper (  
  id              NUMBER NOT NULL,  
  title           VARCHAR(50) NOT NULL,  
  contact_aid     NUMBER,  
  cite_id         NUMBER,  
  cite_format     VARCHAR(50) CHECK (VALUE IN ('ACM', 'IEEE')),  
  root_elm        VARCHAR(20) NOT NULL,  
  parent_elm      VARCHAR(20),  
  fk_cite         VARCHAR(20) CHECK (fk_cite IN (SELECT cite_id FROM paper)),  
  fk_conf         VARCHAR(20),  
  PRIMARY KEY (id),  
  UNIQUE (cite_id),  
  FOREIGN KEY (fk_conf) REFERENCES conf(id),  
  FOREIGN KEY (contact_aid) REFERENCES person(id)
);
```
CPI Algorithm Results

- Uses structure-oriented translation algorithm (e.g., hybrid inlining [VLDB 99]) as basis
- Preserves constraints during the translation

<table>
<thead>
<tr>
<th>DTD Name</th>
<th>Element/Attribute</th>
<th>Table/Column</th>
<th>INDs</th>
<th>EGDs</th>
<th>TGDs</th>
<th>Domain Depd.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Play</td>
<td>21/0</td>
<td>14/46</td>
<td>1</td>
<td>17</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Tstmt</td>
<td>28/0</td>
<td>17/52</td>
<td>0</td>
<td>17</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>vCard</td>
<td>23/1</td>
<td>8/19</td>
<td>0</td>
<td>18</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>ICE</td>
<td>47/157</td>
<td>27/283</td>
<td>0</td>
<td>43</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>MusicML</td>
<td>12/17</td>
<td>8/34</td>
<td>0</td>
<td>9</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>OSD</td>
<td>16/15</td>
<td>15/37</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>PML</td>
<td>46/293</td>
<td>41/355</td>
<td>0</td>
<td>29</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>XBel</td>
<td>9/13</td>
<td>9/36</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XMI</td>
<td>94/633</td>
<td>129/3013</td>
<td>143</td>
<td>10</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>BSML</td>
<td>112/2495</td>
<td>104/2685</td>
<td>181</td>
<td>99</td>
<td>33</td>
<td>33</td>
</tr>
</tbody>
</table>

Conversion from Relational to XML

Query Database Result

Query Processing & CoT

NeT,
&
CoT

<book>
<price>10</price>

<name>XML</name>

<book>
**Motivation**

- Many database products support conversion from RDB to XML documents
  - IBM DB2 XML Extender, XML-DBMS, Oracle9i, ...
  - Given a relational schema \( R \) and XML schema \( X \), generate XML documents conforming to \( X \)

- Problem Def: Given a relational schema \( R \), find an XML schema \( X \) that best describes \( R \)

---

**FT Example: \( R (K, A, B, C) \)**

**Element-oriented**

```xml
<!ELEMENT R (A, B, C)>
<!ATTLIST R K ID #REQUIRED>
<!ELEMENT A (#PCDATA)>
<!ELEMENT B (#PCDATA)>
<!ELEMENT C (#PCDATA)>

<R K="1">
  <A/> <B/> <C/>
</R>

<R K="2">
  <A/> <B/> <C/>
</R>

<R K="3">
  <A/> <B/> <C/>
</R>
```

**Attribute-oriented**

```xml
<!ELEMENT R (EMPTY)>
<!ATTLIST R K ID #REQUIRED>

<R K="1" A="..." B="..." C="...">
  <A/> <B/> <C/>
</R>

<R K="2" A="..." B="..." C="...">
  <A/> <B/> <C/>
</R>

<R K="3" A="..." B="..." C="...">
  <A/> <B/> <C/>
</R>
```
Problems?

Nesting

- **Nested relational model**: allows non-1NF
- **Nest** operator:
  - Select a column $X$ for nesting
  - Group all tuples that have the same values for the remaining columns into a set
Nesting (cont)

**nest\(_A(t)\)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>

**nest\(_C(t)\)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>{10,20}</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>{10,20}</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>

Nesting (cont)

**nest\(_C(t)\)**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>{10,20}</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>{10,20}</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>

**nest\(_A(nest\(_C(t)\))\)**

<table>
<thead>
<tr>
<th>A+</th>
<th>B</th>
<th>C+</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,2,3}</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>10</td>
</tr>
<tr>
<td>{4,5}</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A+</th>
<th>B</th>
<th>C+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>{10,20}</td>
</tr>
<tr>
<td>{2,3}</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>{10,20}</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>
Nesting Properties

- Some properties [Jaeschke & Schek; PODS 82]
  - $nest_A(nest_B(t)) \neq nest_B(nest_A(t))$
  - $nest_A(nest_B(t)) = nest_A(t)$

- **Lemma 1:** Functional Dependencies are preserved in nesting [Fischer et al; JCSS 85]
- **Lemma 2:** Applying nest operator on a non-key column $X$ fails
- **Lemma 3:** For a table $t$ with $n$ columns and $m$ prime columns ($m \leq n$), max # of necessary nesting is $N$:
  \[ N = m + m(m-1) + \ldots + m(m-1)\ldots(2)(1) \]

Nesting-based Translation (NeT)

- $R \rightarrow X$:
  - For each table in $R$, apply nesting repeatedly until no nesting succeeds.
  - If no nesting succeeds, do FT
  - Otherwise, for each column $c$ where nesting succeeded, convert $c$ to
    - $c^*$ if $c$ was nullable
    - $c^+$ if $c$ was not nullable
NeT Example

- R (A,B,C)
- Suppose
  - Nesting succeeded on columns A & C
  - Columns A and C are not nullable

- FT: <!ELEMENT R (A,B,C)>
- NeT: <!ELEMENT R (A+,B,C+)>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>20</td>
</tr>
</tbody>
</table>

NeT Example (cont)

FT: R (A,B,C)
<t> <A>1</A><B>a</B><C>10</C></t>
<t> <A>1</A><B>a</B><C>20</C></t>
<t> <A>2</A><B>a</B><C>10</C></t>
<t> <A>3</A><B>a</B><C>10</C></t>
<t> <A>4</A><B>b</B><C>10</C></t>
<t> <A>4</A><B>b</B><C>20</C></t>

NeT: R (A+,B,C+)
<t> <A>1</A><B>a</B><C>10</C></t>
<t> <A>2</A><B>a</B><C>10</C></t>
<t> <A>2</A><B>a</B><C>20</C></t>
<t> <A>3</A><B>a</B><C>10</C></t>
<t> <A>4</A><B>b</B><C>10</C></t>
<t> <A>4</A><B>b</B><C>20</C></t>

...
CoT: Translation using INDs

- NeT
  - Is only applicable for a single table at a time
  - Cannot draw a big picture when multiple tables exist in a schema
- CoT
  - Uses INDs to derive a more intuitive schema
  - For tables \( s \) and \( t \) with columns \( X \) and \( Y \), and an IND \( s[A] \subseteq t[B] \),

**Pull-up(\( s,t \))**
- Table \( t \):
  - If \( A \) is a super key, there is an 1:1 relationship btw. \( s \) and \( t \).
    - `<!ELEMENT t (Y, s)>`
  - If \( A \) is not a super key, there is an n:1 relationship from \( s \) to \( t \).
    - `<!ELEMENT t (Y, s*)>`
- Table \( s \):
  - `<!ELEMENT s (X-A)>`

**CoT Example**

- `student (Sname, Course, Advisor, Gender)`
- `professor (Pname, Age)`

**FT**
- `<!ELEMENT student (Sname, Course, Advisor, Gender)>`
- `<!ELEMENT professor (Pname, Age)>`

**CoT**
- `<!ELEMENT student (Sname, Course, Gender)>`
- `<!ELEMENT professor (Pname, Age, student*)>`
General CoT Algorithm

- Given a set of Inclusion Dependencies (INDs), creates a digraph s.t. there is an edge $i \rightarrow j$ for every IND $j \subseteq i$

- Identify Top Nodes
  - Source nodes
  - Node with highest outdegree among mutually recursive nodes

- For each top node $t$, do BFS until all nodes are visited.
  For each visit $v \rightarrow w$,
  - Do pull-up$(w,v)$

CoT Example

**Tables**
- student $(Sid, Name, Advisor)$
- emp $(Eid, Name, ProjName)$
- prof $(Eid, Name, Teach)$
- course $(Cid, Title, Room)$
- dept $(Dno, Mgr)$
- proj $(Pname, Pmgr)$

**INDs**
- student[Advisor] $\subseteq$ prof[Name]
- emp[ProjName] $\subseteq$ proj[Pname]
- prof[Teach] $\subseteq$ course[Cid]
- prof[Eid,Name] $\subseteq$ emp[Eid,Name]
- dept[Mgr] $\subseteq$ emp[Eid]
- proj[Pmgr] $\subseteq$ emp[Eid]
INDs Graph

Complex CoT Example (cont)

```xml
<!ELEMENT course (Cid, Title, Room, prof*)>
<!ELEMENT prof (Name, student*)>
<!ATTLIST prof Eid ID>
<!ELEMENT student (Sid, Name)>
<!ELEMENT emp (Eid, Name, ProjName, dept*, proj*)>
<!ATTLIST emp Ref_prof IDREF>
<!ELEMENT dept (Dno)>
<!ELEMENT proj (Pname)>
```
NeT Experimental Results

- Compared output with DB2XML v1.3
- Used MS Access NorthWind database

**DB2XML**

```xml
<ELEMENT Orders
(CustinerID,EmployeeID,ShipVia,ShipAddress,ShipCity,ShipCountry,ShipPostalCode)>  
<ELEMENT CustomerID (#PCDATA)>  
<ATTLIST CustomerID ISNULL (true|false) #IMPLIED>
...
<ELEMENT ShipPostalCode (#PCDATA)>  
<ATTLIST ShipPostalCode ISNULL (true|false) #IMPLIED>
```

**Net**

```xml
<ELEMENT Orders
(EmployeeID+,ShipVia* )
<ATTLIST Orders
CustomerID CDATA #REQUIRED
ShipAddress CDATA #IMPLIED
ShipCity CDATA #IMPLIED
ShipCountry CDATA #IMPLIED
ShipPostalCode CDATA #IMPLIED>
<ELEMENT EmployeeID (#PCDATA)>
<ELEMENT ShipVia (#PCDATA)>
```

1. Order is handled by multiple employees
2. Orders go through multiple shipping locations

CoT Experimental Results

- CoT identified many constraints
- CoT reduced redundancy

12% TPC-H schema
Conclusion

- CPI can capture various “constraints” from XML schema and preserve them in the final relational schema.

- Converting “flat” relational schema to “hierarchical” XML schema in 1-to-1 manner is not good.

- NeT can find the hidden semantics from data and use them to generate non-redundant XML schema.

- CoT can find “inter-relationships” of complex relational schema where multiple tables are interconnected.