

Nesting-based Relational-to-XML Schema Translation



Dongwon Lee Murali Mani
Frank Chiu Wesley W. Chu

UCLA / CSD
<http://www.cs.ucla.edu/db/>



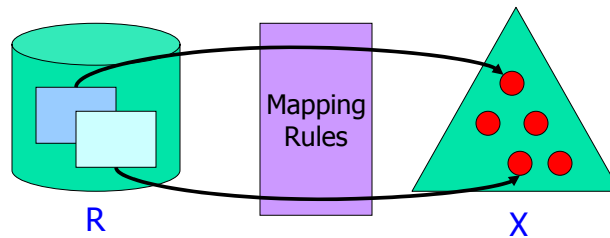
Overview

- Motivation
- Input and Output Models
- Proposed Techniques
 - Flat Translation (FT)
 - Nesting-based Translation (NeT)
 - Translation using Inclusion Dependency (NeT^C)
- Conclusion



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**



WebDB'01

3



Motivation (contd.)

- Users have to provide a target XML schema **X** first
- Then, provide mapping rules from **R** to **X**
- Problematic when dealing with
 - Large relational schema
 - Old relational schema
- Goal: translate a relational schema **R** to a **"GOOD"** XML schema **X** so that users can use **X** for other purposes

WebDB'01

4



Input and Output Model

- Concise and precise notations for input and output
- Dozen XML schema language proposals
 - Proposals from W3C: DTD, XML-Schema
 - Proposals related to ISO/OASIS: RELAX, TREX, RELAX+TREX=?
 - Industry: XDR (Microsoft), DSD (AT&T), SOX (CommerceOne), ...
 - Misc.: Schematron, DDML, ...
- Different focus, different expressive power



Input and Output Model (contd.)

- Use a formalism instead of one schema language proposal
 - Borrow structure from DTD and RELAX
 - Borrow type from XML-Schema
 - Borrow constraints from XML-Schema and Schematron
 - Borrow notations from [Fan & Simeon; PODS 00]
- In the talk, however, I will use DTD as output notation for simplicity



Flat Translation (FT)

- R -> X
 - Table of R to element of X
 - Column of R to
 - Attribute of X (in attribute-oriented mode)
 - Element of X (in element-oriented mode)
- Simple and straightforward translation
- But relational model is "flat" while XML model is "hierarchical"
- Need to take advantage of such hierarchical feature to capture the original schema more accurately
 - XML model provides RE (?, *, +, |)



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

Element-oriented

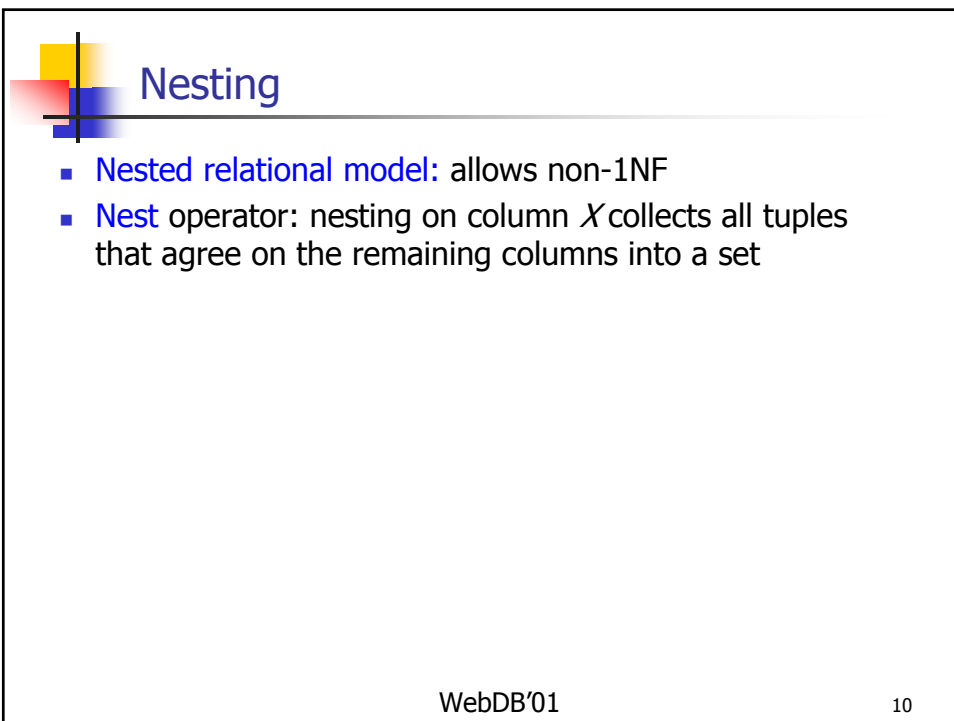
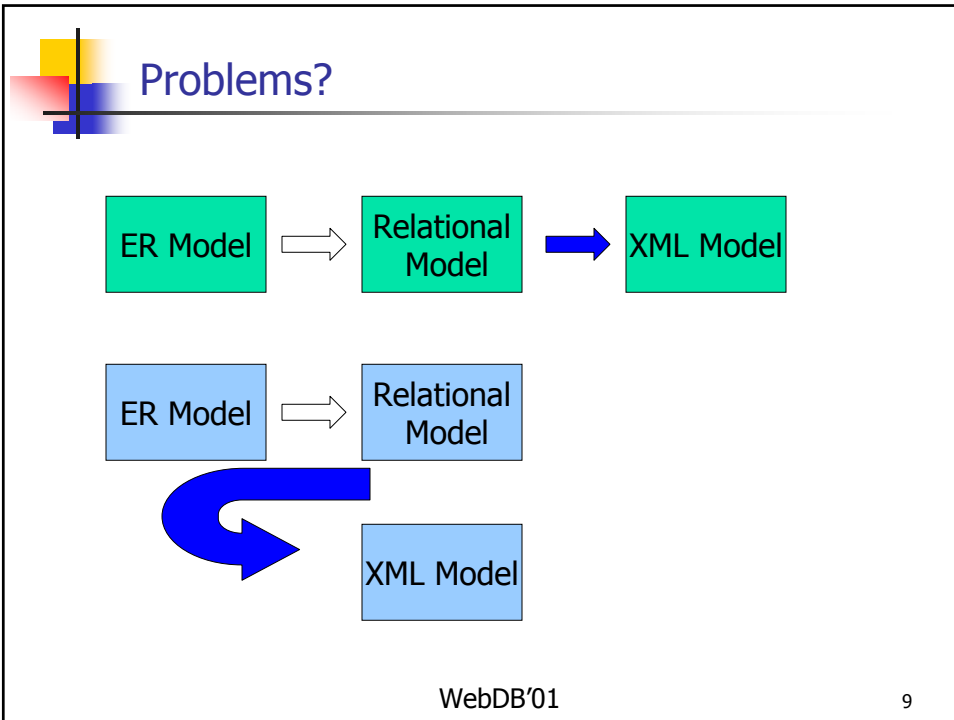
```
<!ELEMENT R (A, B, C)>
<!ATTLIST R K ID #REQUIRED>
<!ELEMENT A (#PCDATA)>
<!ELEMENT B (#PCDATA)>
<!ELEMENT C (#PCDATA)>

<R K="1">
  <A/> <B/> <C/>
</R>
<R K="2">
  <A/> <B/> <C/>
</R>
```

Attribute-oriented

```
<!ELEMENT R (EMPTY)>
<!ATTLIST R K ID #REQUIRED
           A CDATA
           B CDATA
           C CDATA>

<R K="1" A="..." B="..." C="..."/>
<R K="2" A="..." B="..." C="..."/>
<R K="3" A="..." B="..." C="..."/>
```



Nesting (contd.)

A	B	C
1	a	10
1	a	20
2	a	10
3	a	10
4	b	10
4	b	20
5	b	20

→

A+	B	C
{1,2,3}	a	10
1	a	20
4	b	10
{4,5}	b	20

A	B	C+
1	a	{10,20}
2	a	10
3	a	10
4	b	{10,20}
5	b	20

→

A	B	C+
1	a	{10,20}
2	a	10
3	a	10
4	b	{10,20}
5	b	20

WebDB'01 11

Nesting (contd.)

A	B	C+
1	a	{10,20}
2	a	10
3	a	10
4	b	{10,20}
5	b	20

→

A+	B	C+
1	a	{10,20}
{2,3}	a	10
4	b	{10,20}
5	b	20

WebDB'01 12



Nesting (contd.)

- Some properties [Jaeschke & Schek; PODS 82]
 - $\text{nest}_A(\text{nest}_B(t)) \neq \text{nest}_B(\text{nest}_A(t))$
 - $\text{nest}_A(\text{nest}_A(t)) = \text{nest}_A(t)$
- Functional Dependencies are preserved against nesting [Fischer et al; JCSS 85]
- Applying nest operator on a non-prime column X yields no changes

- For a table t with n columns and m prime columns ($m \leq n$), max # T of necessary nesting is:
$$T = m + m(m-1) + \dots + m(m-1)\dots(2)(1)$$



Nesting-based Translation (NeT)

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



NeT Example

- R (A,B,C)
- Nesting was succeeded on columns B & C

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



NeT Example (contd.)

A	B	C
1	a	10
1	a	20
2	a	10
3	a	10
4	b	10
4	b	20
5	b	20

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> <C>20</C> </t>
<t> <A>2</A><A>3</A>
    <B>a</B><C>10</C></t>
<t> <A>4</A><B>b</B>
    <C>10</C><C>20</C> </t>...

```




Translation using INDs (NeT^C)

- NeT
 - Is only applicable for a single table at a time
 - Cannot draw a big picture when multiple tables exist in a schema
- NeT^C
 - Uses inclusion dependencies 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*)

- 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**)>
- <!ELEMENT *s* (*X*-*A*)>

WebDB'01 17



NeT^C Example

student (Sname, Course, Advisor, Gender)

professor (Pname, Age)

↑

FT

<!ELEMENT student (Sname, Course, Advisor, Gender)

<!ELEMENT professor (Pname, Age)

NeT^C

<!ELEMENT student (Sname, Course, Gender)

<!ELEMENT professor (Pname, Age, student*)

WebDB'01 18



General NeT^C Algorithm

- Given a set of INDs, creates a digraph s.t. there is an edge $i \rightarrow j$ for every IND $j[\dots] \subseteq i[\dots]$
- 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 node $v \rightarrow w$, pull-up(w, v)



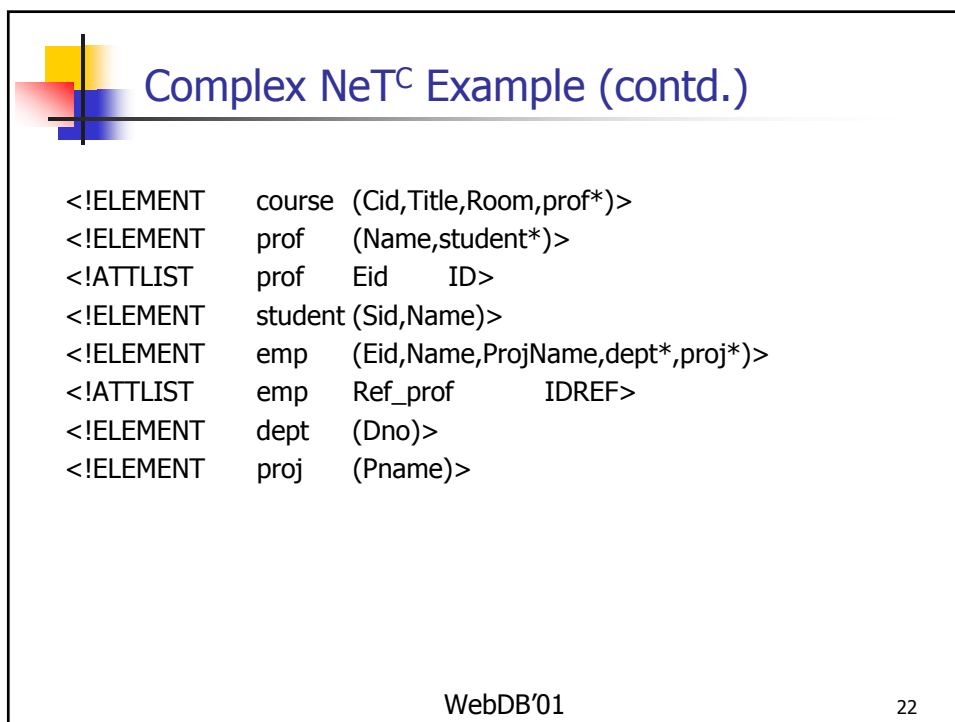
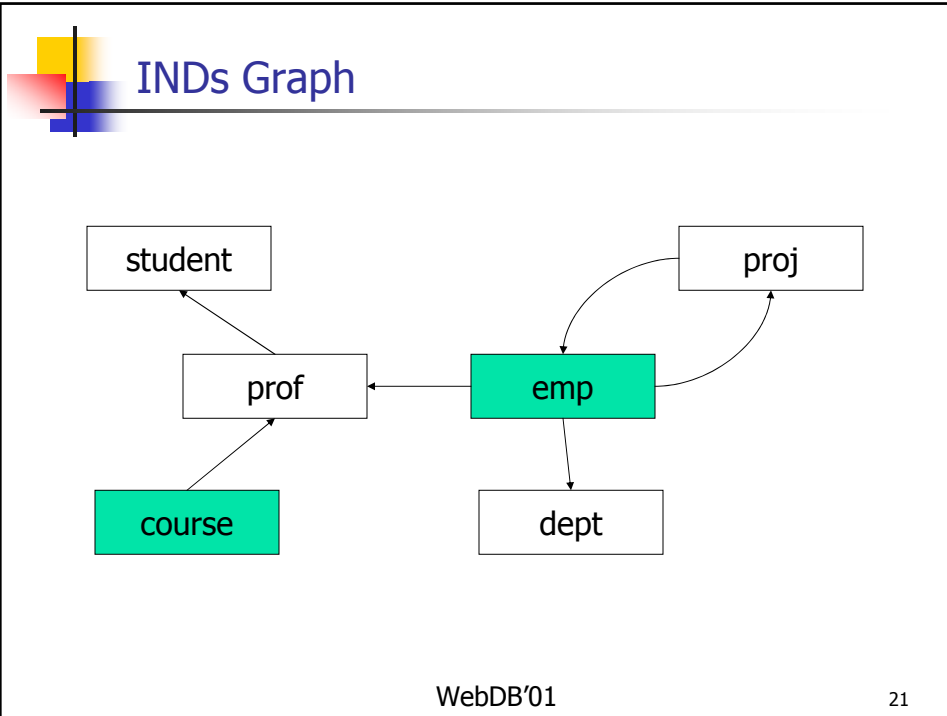
Complex NeT^C Example

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]





Experimental Results

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

DB2XML

```
<!ELEMENT Orders
  (CustomerID,EmployeeID,ShipVia,Ship
  Address,ShipCity,ShipCountry,ShipPo
  stalCode)>
<!ELEMENT CustomerID (#PCDATA)>
<!ATTLIST CustomerID ISNULL
  (true|false) #IMPLIED>
...
<!ELEMENT ShipPostalCode
  (#PCDATA)>
<!ATTLIST ShipPostalCode ISNULL
  (true|false) #IMPLIED>
```

NeT

```
<!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)>
```

WebDB'01

23



Related Work

- Conversion * -> DTD
 - XML Documents -> DTD: STORED [SIGMOD98], XTRACT [SIGMOD 00]
 - View Description -> DTD: MIX [ICDE 99]
 - Relational Schema -> DTD: XML-DBMS, DB2 Extender, Oracle 9i iFS, Informix ObjectTranslator, ...
- Conversion DTD -> R
 - XRel [TOIT 01], Inlining [VLDB 99], ...
 - XML-DBMS, ...
- SilkRoute, XPERANTO, ...

WebDB'01

24



Conclusion

- 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 more intuitive XML schema

- Improvements are ongoing
 - Nesting on multiple attributes
 - Information capacity aspects