

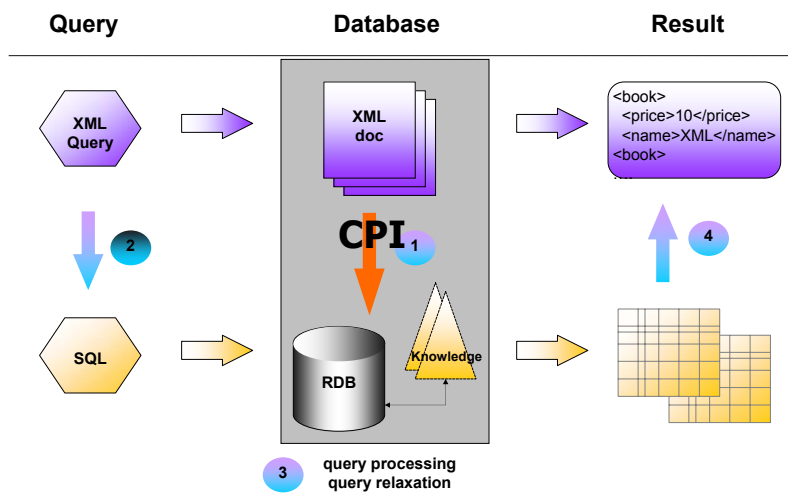
Effective Schema Conversion between XML and Relational Models

Wesley W. Chu

With Dongwon Lee and Murali Mani

University of California, Los Angeles

Conversion from XML to Relational



Difficulties

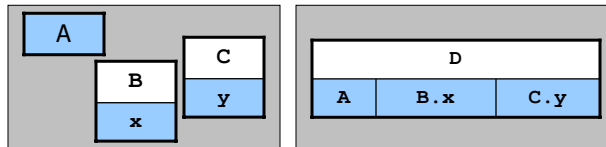
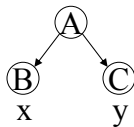
- No 1-to-1 mapping
- Set (a^* , $(b+c?)$), Recursion

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

- Fragmentation & inlining

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

list	
name	item
A	1,2
B	3,4



Constraints in DTD

- Domain constraint

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

- Cardinality constraint

```
<!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)

```
<!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_1 and y_2 of type Y , y_1 and y_2 must be equal
- 1-to- $\{0,1\}$ and 1-to- $\{1\}$ cardinality cases

$X \rightarrow X.Y$

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 ($P \rightarrow C$): Every element of type P must have at least one child element of type C
 - 1-to- $\{1\}$ and 1-to- $\{1, \dots\}$ cardinality cases
 - Parent property ($C \rightarrow P$): 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>
<!ATTLIST contact aid IDREF #REQUIRED>
<!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"/></person>
      <person id="sheth">
        <name fn="Amit" ln="Sheth"/><email>amit@cs.uga.edu</email>
      </person>
    </author>
  </paper>
</conf>
```

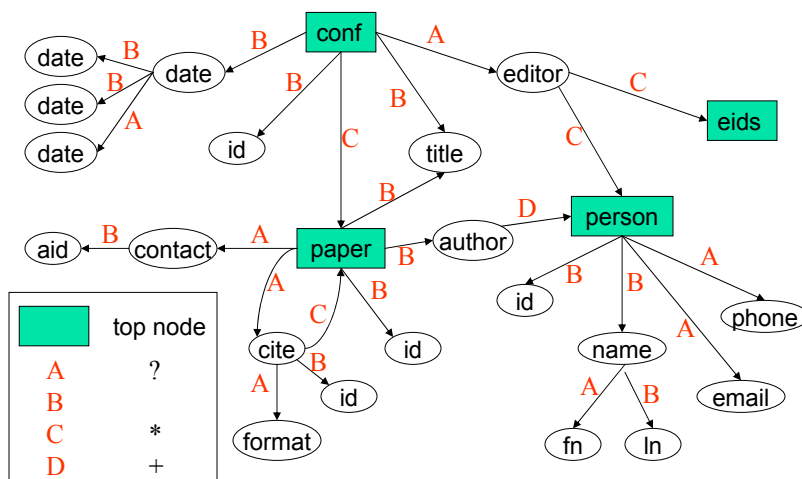
Conference.xml (cont)

```

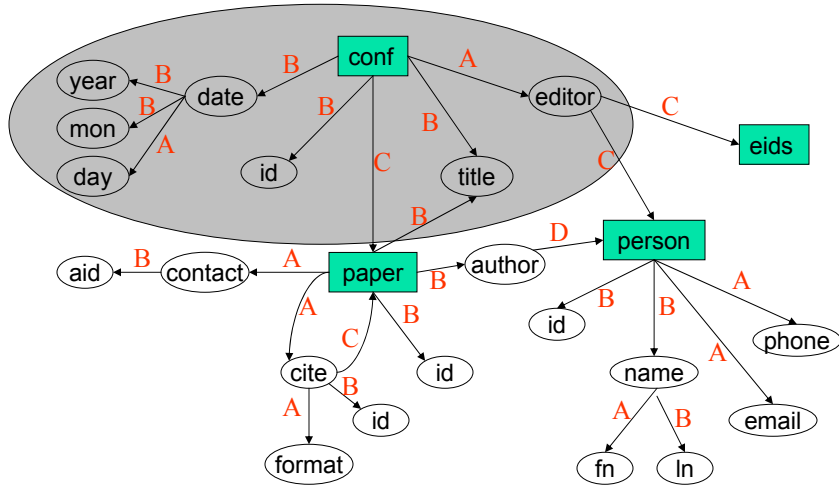
<cite id="c100" format="ACM">
  <paper id="p3">
    <title>Making Sense of Scientific Info...</title>
    <author>
      <person id="bossy">
        <name fn="Marcia" ln="Bossy"/><phone>391.4337</phone>
      </person>
    </author>
  </paper>
</cite>
</paper>
</conf>
<paper id="p7">
  <title>Constraints-preserving Transformation...</title>
  <contact aid="lee"/>
  <author>
    <person id="lee">
      <name fn="Dongwon" ln="Lee"/><email>dongwon@cs.ucla.edu</email>
    </person> </author>
  <cite id="c200" format="IEEE"/>
</paper>

```

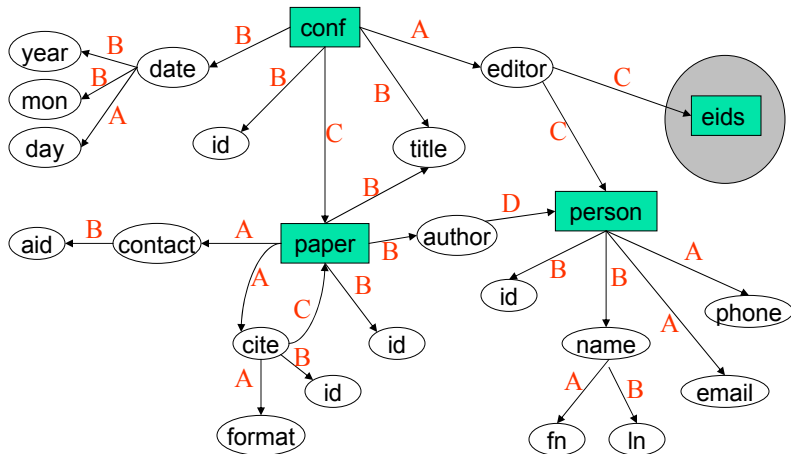
Annotated DTD Graph



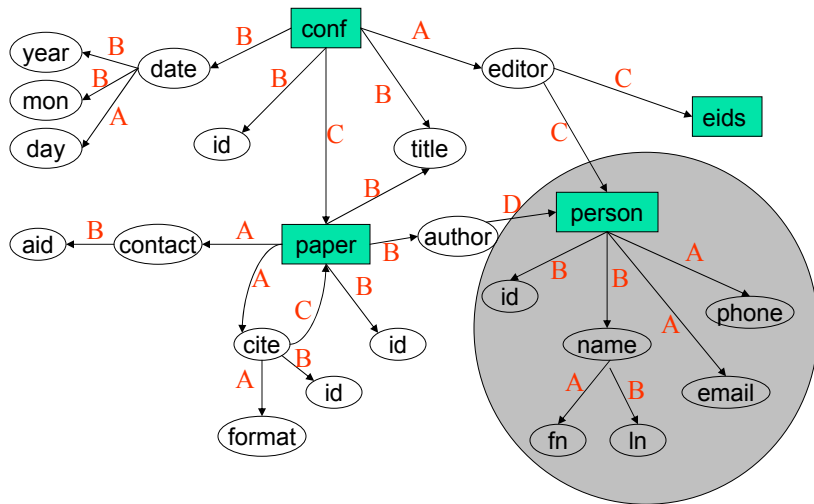
CPI Steps



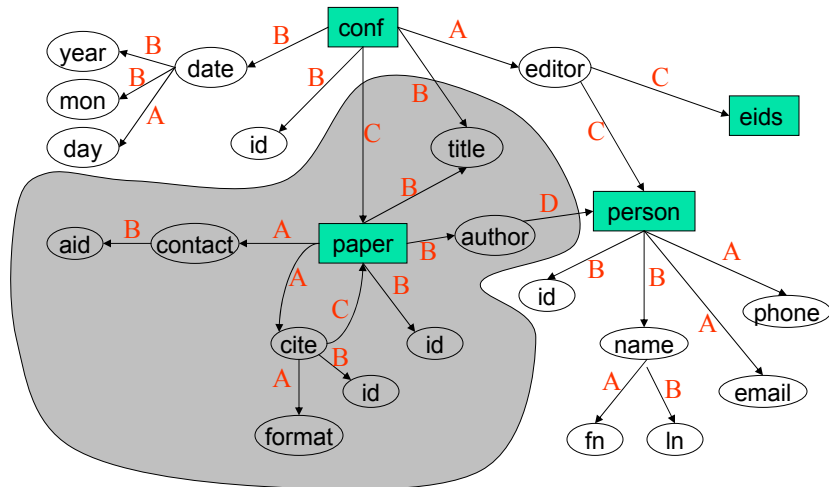
CPI Steps (cont)



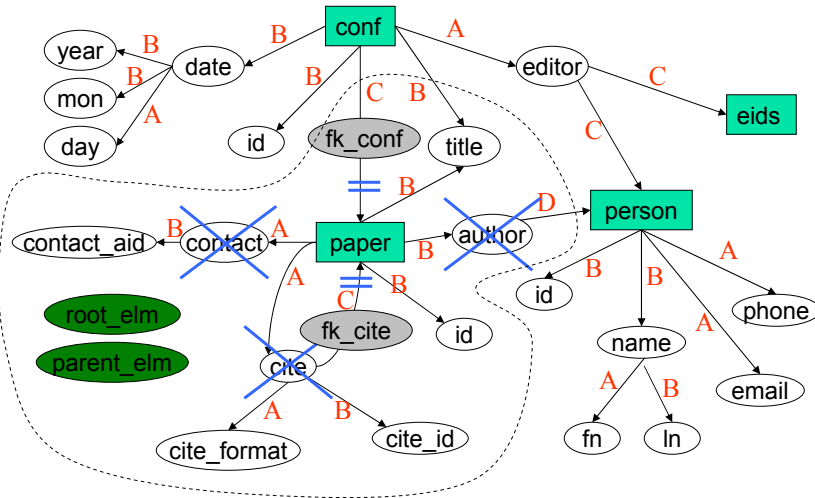
CPI Steps (cont)



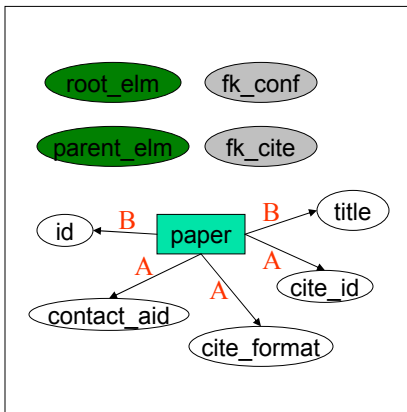
CPI Steps (cont)



CPI Steps (cont)



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., $fk_cite \subseteq cite_id$)
- **id** is a PK
- **cite_id** is UNIQUE

Paper Table after CPI

id	root_elm	parent_elm	fk_conf	fk_cite
p1	conf	conf	er99	
p2	conf	conf	er99	
p3	conf	cite		c100
p7	paper			

title	contact_aid	cite_id	cite_format
Indexing...	dao		
Logical...	shah	c100	ACM
Making...			
Constraints...	lee	c200	IEEE

Schema after CPI

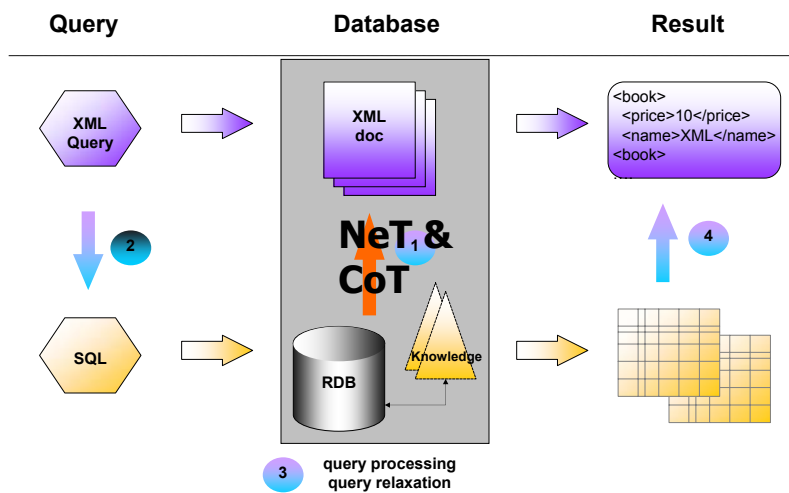
```
CREATE TABLE paper (  
  id          NUMBER          NOT NULL,  
  title       VARCHAR(50)     NOT NULL,  
  contact_aid NUMBER,  
  cite_id     NUMBER,  
  cite_format VARCHAR(50)  
             CHECK (VALUE IN ("ACM", "IEEE")),  
  root_elm   VARCHAR(20)     NOT NULL,  
  parent_elm VARCHAR(20),  
  
  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

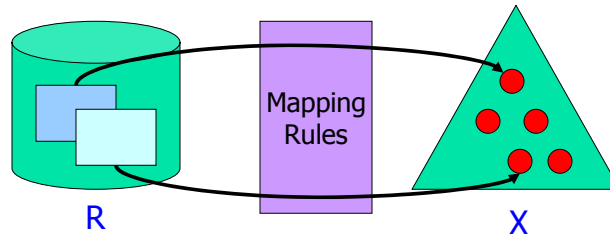
DTD Name	Element/Attribute	Table/Column	INDs	EGDs	TGDs	Domain Depd.
Play	21/0	14/46	1	17	30	30
Tstmt	28/0	17/52	0	17	22	22
vCard	23/1	8/19	0	18	13	13
ICE	47/157	27/283	0	43	60	60
MusicML	12/17	8/34	0	9	12	12
OSD	16/15	15/37	0	2	2	2
PML	46/293	41/355	0	29	36	36
XBel	9/13	9/36	4	9	1	1
XMI	94/633	129/3013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

Conversion from Relational to XML



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

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

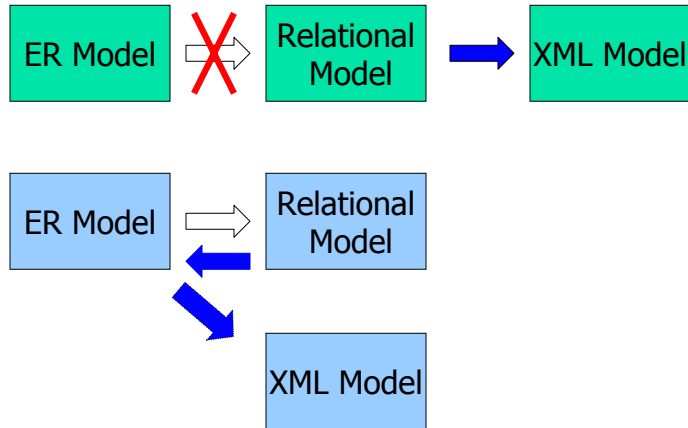
<R K="1">
  <A/> <B/> <C/>
</R>
<R K="2">
  <A/> <B/> <C/>
</R>
```

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="..."/>
```

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)

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



$nest_A(t)$

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

$nest_C(t)$

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

Nesting (cont)

$nest_C(t)$

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



$nest_A(nest_C(t))$

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

Nesting Properties

- Some properties [Jaeschke & Schek; PODS 82]

- $\text{nest}_A(\text{nest}_B(t)) \neq \text{nest}_B(\text{nest}_A(t))$
- $\text{nest}_A(\text{nest}_A(t)) = \text{nest}_A(t)$

- **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 :
$$N = m + m(m-1) + \dots + m(m-1)\dots(2)(1)$$

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+)>

NeT Example (cont)

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>...
  
```

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 .
<IELEMENT $t(Y, s)$ >
 - If A is not a super key, there is an $n:1$ relationship from s to t .
<IELEMENT $t(Y, s^*)$ >
- Table s :
 - <IELEMENT $s(X-A)$ >

CoT Example

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



FT

<IELEMENT student (Sname, Course, Advisor, Gender)
<IELEMENT professor (Pname, Age)

CoT

<IELEMENT student (Sname, Course, Gender)
<IELEMENT 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 $j[\dots] \subseteq i[\dots]$
- 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 visit $v \rightarrow w$,
 - Do pull-up(w, v)

CoT Example

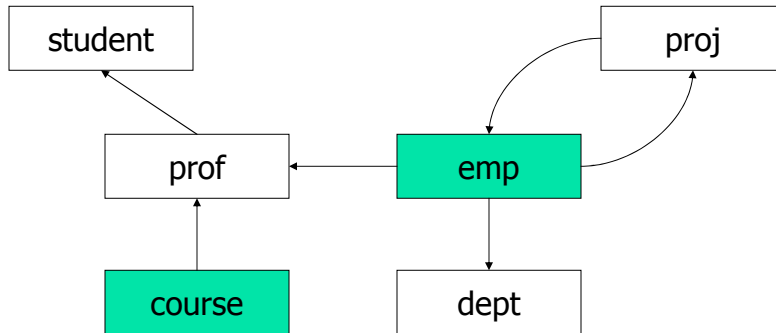
Tables

- 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)

```
<!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

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

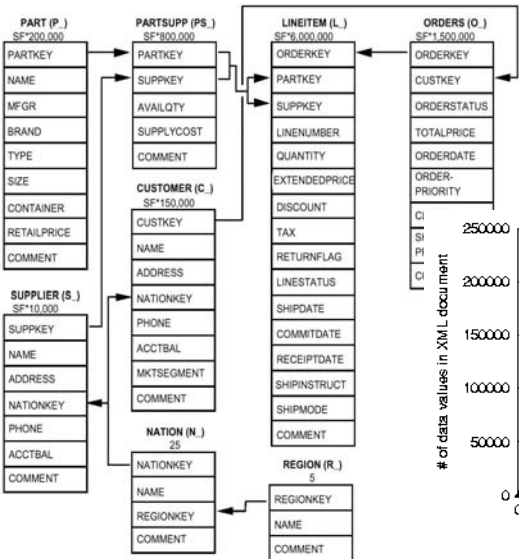
```

DB2XML
<!ELEMENT Orders
  (CustinerID,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>
    
```

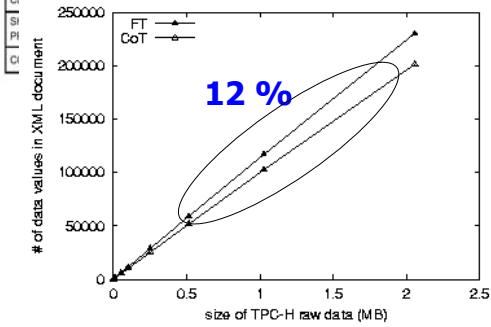
```

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)>
    
```

CoT Experimental Results



- CoT identified many constraints
- CoT reduced redundancy



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