Lecture 14: XML Publishing & Storage Midterm Review

Wednesday, October 30, 2002

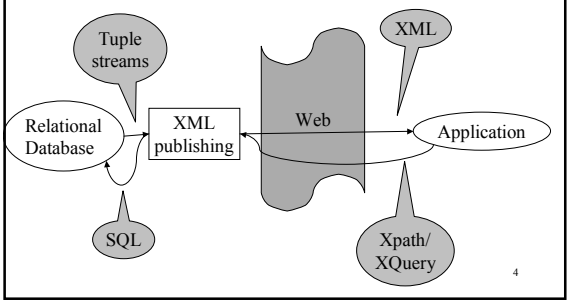
Outline

- XML publishing
- XML storage
- Final review

XML from/to Relational Data

- XML publishing:
  - relational data → XML
- XML storage:
  - XML → relational data

XML Publishing



XML Publishing

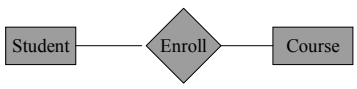
- Exporting the *data* is easy: we do this for HTML
- Translating XQuery → SQL is hard

XML publishing systems:

- Research: Xperanto (IBM/DB2), SilkRoute (AT&T Labs and UW)
  - XQuery → SQL
- Commercial: SQL Server, Oracle
  - only Xpath → SQL and with restrictions

XML Publishing

Will follow SilkRoute, more or less



- Relational schema:

  Student(<u>sid</u>, name, address)

  Course(<u>cid</u>, title, room)

  Enroll(<u>sid, cid,</u> grade)

XML Publishing

```
<xmlview>
   <course>  <title> Operating Systems </title>
             <room> MGH084 </room>
             <student> <name> John </name>
                       <address> Seattle </address >
                       <grade> 3.8 </grade>
             </student>
             <student> …</student>
      …
   </course>
   <course>  <title> Database </title>
             <room> EE045 </room>
             <student> <name> Mary </name>
                       <address> Shoreline </address >
                       <grade> 3.9 </grade>
             </student>
             <student> …</student>
      …
   </course>
   …
</xmlview>
```

Group by courses: redundant representation of students

Other representations possible too

XML Publishing

First thing to do: design the DTD:

```
<!ELEMENT xmlview (course*)>
<!ELEMENT course (title, room, student*)>
<!ELEMENT student (name,address,grade)>
<!ELEMENT name    (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT grade   (#PCDATA)>
<!ELEMENT title   (#PCDATA)>
```


Now we write an XQuery to export relational data → XML
Note: result is is the right DTD

```
<xmlview>
{ FOR $x IN /db/Course/row
  RETURN
    <course>
      <title>   { $x/title/text() } </title>
      <room> { $x/room/text() } </room>
      { FOR  $y IN /db/Enroll/row[cid/text() = $x/cid/text()]
             $z IN /db/Student/row[sid/text() = $y/sid/text()]
        RETURN <student> <name>    { $z/name/text() } </name>
                         <address> { $z/address/text() } </address>
                         <grade>   { $y/grade/text() } </grade>
               </student>
      }
    </course>
}
</xmlview>
```


XML Publishing

Query: find Mary's grade in Operating Systems

XQuery

```
FOR $x IN /xmlview/course[title/text()="Operating Systems"],
      $y IN $x/student/[name/text()="Mary"]
RETURN  <answer> $y/grade/text() </answer>
```

SilkRoute does this automatically

SQL

```
SELECT Enroll.grade
FROM Student, Enroll, Course
WHERE Student.name="Mary" and Course.title="OS"
    and Student.sid = Enroll.sid and Enroll.cid = Course.cid
```


XML Publishing

How do we choose the output structure ?

- Determined by agreement
- Or dictated by committees
  - XML dialects (called *applications*) = DTDs
- XML Data is often nested, irregular, etc
- No normal forms for XML

XML Storage

- Most often the XML data is small
  - E.g. a SOAP message
  - Parsed directly into the application (DOM API)
- Sometimes XML data is large
  - need to store/process it in a database
- The XML storage problem:
  - How do we choose the schema of the database ?

XML Storage

Two solutions:

- Schema derived from DTD
- Storing XML as a graph: "Edge relation"

13

Designing a Schema from DTD

Design a relational schema for:

```
<!DOCTYPE company [
  <!ELEMENT company   ((person|product)*)>
  <!ELEMENT person  (ssn, name, office?, phone*)>
  <!ELEMENT ssn       (#PCDATA)>
  <!ELEMENT name     (#PCDATA)>
  <!ELEMENT office    (#PCDATA)>
  <!ELEMENT phone    (#PCDATA)>
  <!ELEMENT product  (pid, name, ((price,availability)|description))>
  <!ELEMENT pid    (#PCDATA)>
  <!ELEMENT description    (#PCDATA)>
]>
```

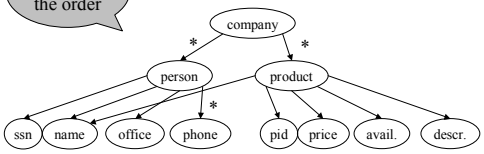
14

Designing a Schema from DTD

First, construct the DTD graph:

```
<!DOCTYPE company [
  <!ELEMENT company   ((person|product)*)>
  <!ELEMENT person  (ssn, name, office?, phone*)>
  <!ELEMENT ssn       (#PCDATA)>
  <!ELEMENT name     (#PCDATA)>
  <!ELEMENT office    (#PCDATA)>
  <!ELEMENT phone    (#PCDATA)>
  <!ELEMENT product  (pid, name, ((price,availability)|description))>
  <!ELEMENT pid    (#PCDATA)>
  <!ELEMENT description    (#PCDATA)>
]>
```

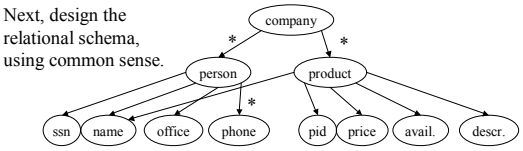
We ignore the order



15

Designing a Schema from DTD

Next, design the relational schema, using common sense.



Person(<u>ssn</u>, name, office)
Phone(<u>ssn, phone</u>)
Product(<u>pid</u>, name, price, avail., descr.)

Which attributes may be NULL ?

16

Designing a Schema from DTD

What happens to queries:

```
FOR $x IN /company/product[description]
RETURN  <answer> { $x/name, $x/description } </answer>
```

```
SELECT Product.name, Product.description
FROM    Product
WHERE Product.description IS NOT NULL
```

17

Storing XML as a Graph

Sometimes we don't have a DTD:

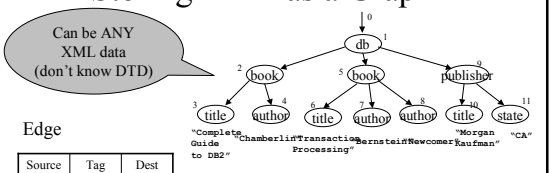
- How can we store the XML data ?

Every XML instance is a *tree*

- Store the edges in an Edge table
- Store the #PCDATA in a Value table

18

Storing XML as a Graph



Edge

Source	Tag	Dest
0	db	1
1	book	2
2	title	3
2	author	4
1	book	5
5	title	6
5	author	7
. . .	. . .	. . .

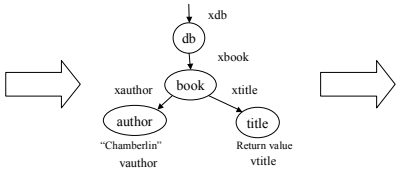
Value

Source	Val
3	Complete guide . . .
4	Chamberlin
6	. . .
. . .	. . .

Storing XML as a Graph

What happens to queries:

```
FOR $x IN /db/book[author/text()="Chamberlin"]
RETURN  $x/title
```



Storing XML as a Graph

What happens to queries:

A 6-way join !!!

```
SELECT vtitle.value
FROM    Edge xdb, Edge xbook, Edge xauthor, Edge xtitle,
        Value vauthor, Value vtitle
WHERE xdb.source = 0                    and xdb.tag = 'db'
      and xdb.dest = xbook.source       and xbook.tag = 'book'
      and xbook.dest = xauthor.source   and xauthor.tag = 'author'
      and xbook.dest = xtitle.source    and xtitle.tag = 'title'
      and xauthor.dest = vauthor.source and vauthor.value = 'Chamberlin'
      and xtitle.dest = vtitle.source
```


Storing XML as a Graph

Edge relation summary:

- Same relational schema for every XML document:
  Edge(Source, Tag, Dest)
  Value(Source, Val)
- Generic: works for *every* XML instance
- But inefficient:
  - Repeat tags multiple times
  - Need many joins to reconstruct data

Other XML Topics

- Name spaces
- XML API:
  - DOM = "Document Object Model"
- XML languages:
  - XSLT
- XML Schema
- Xlink, XPointer
- SOAP

Available from www.w3.org
(but don't spend rest of your life reading those standards !)

Research on XML Data Management at UW

- Processing:
  - Query languages (XML-QL, a precursor of XQuery)
  - Tukwila
  - XML updates
- XML publishing/storage
  - SilkRoute
  - STORED
- XML tools
  - XML Compressor: Xmill
  - XML Toolkit (xsort, xagg, xgrep, xtransf, etc)
- Theory:
  - Typechecking
  - Xpath containment

The Midterm

- Open book exam
  - And open notebooks, lecture notes, ...
  - But no computers
- Four questions:
  1. SQL
  2. E/R
  3. Relational model (algebra, FDs, normal forms)
  4. XML

25

1. SQL

- Selection/project/join
- Understand well duplicates
- Aggregate queries
  - avoid nested queries when a GROUP BY suffices
- Nested queries
  - More difficult ones are with ANY, and NOT IN
- Updates, table creations, views

26

2. E/R Diagrams

- One/many v.s. many/many relationships
- Inheritance
- Translation to relations
  - Remember: no table for one/many !

27

3. Relational Model

- Relational model
  - What is a semijoin ?
  - FD's: make sure you can compute S^+, find keys
  - Normal forms: BCNF, 3NF

28

4. XML

- XML:
  - Basic syntax: elements + attributes
  - DTDs: elements only
  - The tree model
  - Canonical XML view of a relation (<row>...)
- XPath
- XQuery
  - Use it to publish XML from relational data
- How to store XML data

29

Final Thoughts

- Open book
  - But read the book *before* the exam
- Some question(s) may be hard(er)
  - Answer first the questions that are easier
- The answers should not be very complex

30