Introduction to Data Management
CSE 344

Lecture 15: XQuery and E/R Diagrams
Announcements

• Homework 5 due Thursday, 11 pm
• Webquiz 6 due next Tuesday, 11pm

• Today:
  XQuery (12.2) and E/R diagrams (4.1-4.6)
Midterm Statistics

- Mean: 76.93
- Median: 78
- Std dev: 11.12

- Will pass back in sections tomorrow
XQuery

• Standard for high-level querying of databases containing data in XML form
• Based on Quilt, which is based on XML-QL
• Uses XPath to express more complex queries
  – Every XPath expression is itself a (simple) XQuery or can be part of a more complex query

• Reference: sec. 12.2
FLWR ("Flower") Expressions

FOR ... 
LET... 
WHERE... 
RETURN...

Zero or more
Zero or more
Zero or one
Exactly one
FOR-WHERE-RETURN

Find all book titles published after 1995:

FOR $x$ IN doc("bib.xml")/bib/book
WHERE $x/year/text() > 1995
RETURN $x/title

Result:
<title>
Principles of Database and Knowledge Base Systems
</title>
**FOR-WHERE-RETURN**

Equivalently

```
RETURN $x$
```

And even shorter:

```
```
FOR-WHERE-RETURN

• Find all book titles and the year when they were published:

   FOR $x$ IN doc("bib.xml")/ bib/book
   RETURN <answer>
      <title>{ $x/title/text() } </title>
      <year>{ $x/year/text() } </year>
   </answer>

Result:
   <answer> <title> Foundations … </title> <year> 1995 </year> </answer>
   <answer> <title> Principles … </title> <year> 1998 </year> </answer>
FOR-WHERE-RETURN

Lesson: Always put { } around Xpath expressions

• Notice the use of “{“ and “}”
• What is the result without them?

FOR $x$ IN doc("bib.xml")/bib/book
RETURN <answer>
  <title> $x/title/text() </title>
  <year> $x/year/text() </year>
</answer>

<answer>  <title> $x/title/text() </title>  <year> $x/year/text() </year> </answer>
<answer>  <title> $x/title/text() </title>  <year> $x/year/text() </year> </answer>
<answer>  <title> $x/title/text() </title>  <year> $x/year/text() </year> </answer>

Nesting

• For each author of a book by Freeman, list all books he/she published:

```
FOR $b IN doc("bib.xml")/bib,
  $a IN $b/book[publisher/text()="Freeman"]/author
RETURN <result>
  { $a,
    FOR $t IN $b/book[author/text()=$a/text()]/title
    RETURN $t
  }
</result>
```

In the `RETURN` clause comma concatenates XML fragments
Result

<pre> &lt;result&gt;
   &lt;author&gt;Jeffrey D. Ullman&lt;/author&gt;
   &lt;title&gt;Principles of Database ...&lt;/title&gt;
&lt;/result&gt;</pre>
Aggregates

Find all books with more than 3 authors:

```
FOR $x$ IN doc("bib.xml")/bib/book
WHERE count($x/author)>3
RETURN $x$
```

count = a function that counts
avg = computes the average
sum = computes the sum
distinct-values = eliminates duplicates
Aggregates

Same thing:

```
FOR $x IN doc("bib.xml")/bib/book[count(author)>3]
RETURN $x
```
Eliminating Duplicates

Print all authors:

```
FOR $a IN distinct-values($book/author/text())
RETURN <author> { $a } </author>
```

Note: distinct-values applies ONLY to values, NOT elements
The LET Clause

Find books whose price is larger than average:

\[
\text{FOR } \$b \text{ in } \text{doc("bib.xml")}/\text{bib} \\
\text{LET } \$a:=\text{avg}(\$b/\text{book}/\text{price}/\text{text()}) \\
\text{FOR } \$x \text{ in } \$b/\text{book} \\
\text{WHERE } \$x/\text{price}/\text{text()} > \$a \\
\text{RETURN } \$x
\]

LET enables us to declare variables
SQL and XQuery Side-by-side

Product(pid, name, maker, price)  Find all product names, prices, sort by price

SELECT x.name, x.price
FROM Product x
ORDER BY x.price

FOR $x in doc("db.xml")/db/Product/row
ORDER BY $x/price/text()
RETURN <answer>
    { $x/name, $x/price }
</answer>
XQuery’s Answer

<answer>
  <name> abc </name>
  <price> 7 </price>
</answer>

<answer>
  <name> def </name>
  <price> 23 </price>
</answer>

Notice: this is NOT a well-formed document! (WHY ???)
Producing a Well-Formed Answer

```xml
<myQuery>
  { 
    FOR $x in doc("db.xml")/db/Product/row
    ORDER BY $x/price/text()
    RETURN <answer>
      { $x/name, $x/price }
    </answer>
  }
</myQuery>
```
XQuery’s Answer

<myQuery>
  <answer>
    <name> abc </name>
    <price> 7 </price>
  </answer>
  <answer>
    <name> def </name>
    <price> 23 </price>
  </answer>
  ....
</myQuery>

Now it is well-formed!
SQL and XQuery Side-by-side

For each company with revenues < 1M count the products over $100

```
SELECT y.name, count(*)
FROM Product x, Company y
WHERE x.price > 100 and x.maker=y.cid and y.revenue < 1000000
GROUP BY y.cid, y.name
```

```
FOR $r in doc("db.xml")/db,
   $y in $r/Company/row[revenue/text()<1000000]
RETURN
   <proudCompany>
      <companyName>{ $y/name/text() } </companyName>
      <numberOfExpensiveProducts>
         { count($r/Product/row[maker/text()=$y/cid/text()][price/text()>100])}
      </numberOfExpensiveProducts>
   </proudCompany>
```
SQL and XQuery Side-by-side

Find companies with at least 30 products, and their average price

**SQL**

```sql
SELECT y.name, avg(x.price)
FROM Product x, Company y
WHERE x.maker=y.cid
GROUP BY y.cid, y.name
HAVING count(*) > 30
```

**XQuery**

```xquery
FOR $r in doc("db.xml")/db,
    $y in $r/Company/row
LET $p := $r/Product/row[maker/text()=$y/cid/text()]
WHERE count($p) > 30
RETURN
    <theCompany>
        <companyName> { $y/name/text() } </companyName>
        <avgPrice> { avg($p/price/text()) } </avgPrice>
    </theCompany>
```

An element

A collection
XML Summary

• Stands for eXtensible Markup Language
  1. Advanced, self-describing file format
  2. Based on a flexible, semi-structured data model

• Query languages for XML
  – XPath
  – XQuery
E/R Diagrams and Database Design
Database Design

• Why do we need it?
  – Need a way to model real world entities in terms of relations
  – Not easy to go from real-world entities to a database schema

• Consider issues such as:
  – What entities to model
  – How entities are related
  – What constraints exist in the domain
  – How to achieve good designs

• Several formalisms exists
  – We discuss E/R diagrams

• Reading: Sec. 4.1-4.6
Database Design Process

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema
Entity / Relationship Diagrams

- Entity set = a class
  - An entity = an object

- Attribute

- Relationship
Keys in E/R Diagrams

• Every entity set must have a key
What is a Relation?

- A mathematical definition:
  - if $A, B$ are sets, then a relation $R$ is a subset of $A \times B$

- $A=\{1,2,3\}, \quad B=\{a,b,c,d\}$,
  - $A \times B = \{(1,a),(1,b), \ldots, (3,d)\}$
  - $R = \{(1,a), (1,c), (3,b)\}$

- **makes** is a subset of $\text{Product} \times \text{Company}$:
Multiplicity of E/R Relations

- one-one:
  - 1
  - 2
  - 3
  - a
  - b
  - c
  - d

- many-one
  - 1
  - 2
  - 3
  - a
  - b
  - c
  - d

- many-many
What does this say?
Multi-way Relationships

How do we model a purchase relationship between buyers, products and stores?

Can still model as a mathematical set (Q. how ?)

A. As a set of triples $\subseteq$ Person $\times$ Product $\times$ Store
Q: What does the arrow mean?

A: A given person buys a given product from at most one store

[Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]
Arrows in Multiway Relationships

Q: What does the arrow mean?

A: A given person buys a given product from at most one store AND every store sells to every person at most one product.
Converting Multi-way Relationships to Binary

Arrows go in which direction?
Converting Multi-way Relationships to Binary

date

Purchase

StoreOf

BuyerOf

Make sure you understand why!