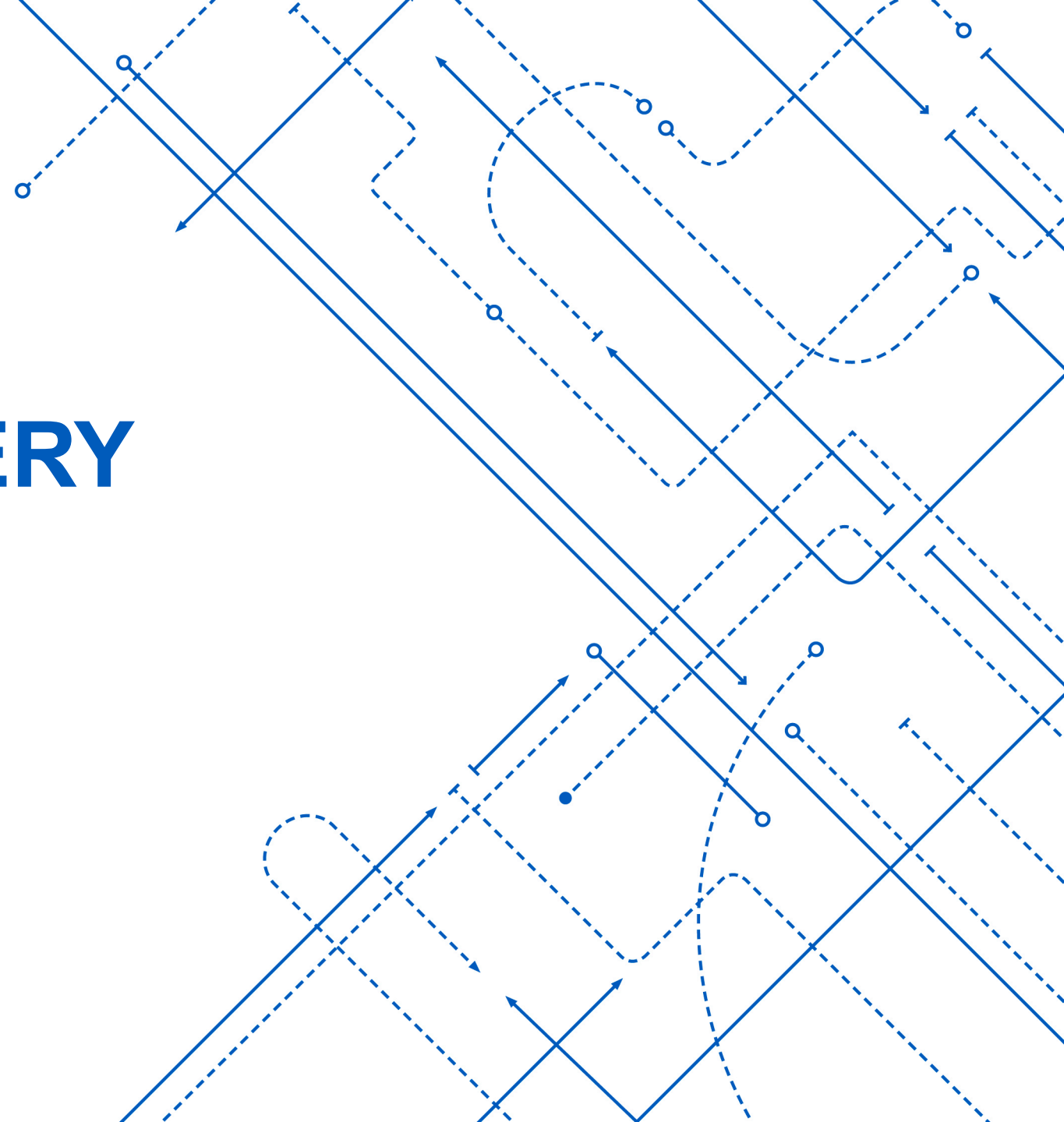


CSE460/560 DATA MODELS AND QUERY LANGUAGES

XPath

Cheng-En Chuang
(Slides Adopted from Jan Chomicki and Ning Deng)

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Outline

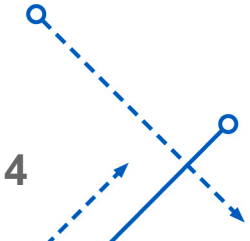
1. Programming Languages for Semistructured Data
2. XPath

Outline

1. **Programming Languages for Semistructured Data**
2. XPath

Introduction

- XPath
  - a language for describing sets based on paths in a graph of semi-structured data
- XQuery
  - a language that queries and transforms semi-structured data such as XML

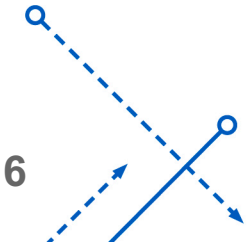


Outline

1. Programming Languages for Semistructured Data
2. **XPath**

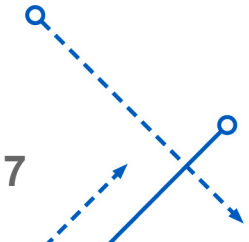
XPath

- Data Model
  - Tree-based: a sequence of items
    - Nodes: document root, element, attribute
    - Atomic values: integer, real, string …
- Every XPath query refers to a document
- Document node
  - Constructed from an XML document by `doc(doc_uri)`
  - The document node represents the XML document itself
    - Not the root element
    - Root element is a child of document node



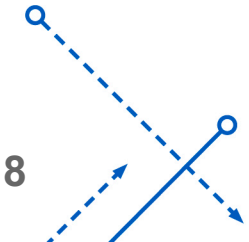
XPath Expressions

- Absolute Path
  - $/ T_1 / T_2 / \ldots / T_n$
  - Where each T_i is a tag
- Evaluation starts with a sequence of one node: the document node
- Each T_i is evaluated in turn, starting with T_1
- To evaluate T_i
  - The context sequence ***S*** of T_i is the results up until T_{i-1}
  - Examine the items of ***S*** in order, for each one
    - Find all sub-elements whose tag is T_i
    - Append them to the output sequence
- Relative Path Expressions
  - Relative to the current node or sequence of nodes



XPath Expression

- Describes a set of paths in a document
- Returns a sequence of nodes
- Evaluated in a context
- Absolute (starting at document root) or relative
- Consists of steps separated by /
- Wildcards (*)



Example XML - Book

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer&apos;s Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
   </book>
</catalog>
```

Example XML

doc("book.xml")/catalog/book/author

```
1 <author>Gambardella, Matthew</author>
2 <author>Ralls, Kim</author>
```

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <catalog>
3     <book id="bk101">
4        <author>Gambardella, Matthew</author>
5        <title>XML Developer&apos;s Guide</title>
6        <genre>Computer</genre>
7        <price>44.95</price>
8        <publish_date>2000-10-01</publish_date>
9        <description>An in-depth look at creating applications
10       with XML.</description>
11    </book>
12    <book id="bk102">
13       <author>Ralls, Kim</author>
14       <title>Midnight Rain</title>
15       <genre>Fantasy</genre>
16       <price>5.95</price>
17       <publish_date>2000-12-16</publish_date>
18    </book>
19 </catalog>
```

XPath Expressions (Attributes)

- Absolute Path
  - $/\ T_1/\ T_2/\ \ldots\ /\ T_n/\ @A$
  - Where each T_i is a tag
  - A is an attribute of T_n
- To evaluate the expression, first compute the expression $/\ T_1/\ T_2/\ \ldots\ /\ T_n/$
- For each element in the result sequence
  - If attribute A exists in its opening tag
  - Append its value to the output sequence

Example XML

doc("book.xml")/catalog/book/@id/string()

```
1 "bk101"

2 "bk102"
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer&apos;s Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
   </book>
</catalog>
```

XPath Expressions (Axes)

- XPath provides a rich set of axes (mode of navigation)
- Two axes so far: child (default) and attribute (@)
- Usage: Prefix a tag or attribute with an axis name
  - / *child*:: T_1 / *child*:: T_2 / … / *child* :: T_n/
    - Equivalent to / T_1/ T_2/ … / T_n /
  - / *child*:: T_1 / *child*:: T_2 / … / *child* :: T_n/ *attribute*::*A*
    - Equivalent to / T_1/ T_2/ … / T_n / @*A*
  - Other axes
    - Parent (“..”)
    - Ancestor
    - Descendant
    - Next-sibling (to the right)
    - Previous-sibling (to the left)
    - Self (“.”)
    - Descendant-or-self (“//”)

Axes

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Example XML

doc("book.xml")//author

```
1 <author>Gambardella, Matthew</author>
2 <author>Ralls, Kim</author>
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer&apos;s Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
   </book>
</catalog>
```

Other Expressions

- Wildcards: “*”
  - for any tag or attribute
  - E.g. doc(“/db/book.xml”)/catalog/*
- Conditions Boolean expressions within square brackets
  - Usage
    - TAG [*i*] : true only for the *i*-th child of the parent TAG
    - TAG [T] : true if TAG element has one or more sub-elements with tag T
    - TAG [@A] : true if TAG element has an attribute A

Example XML - Movie

```
1)  <? xml version="1.0" encoding="utf-8" standalone="yes" ?>
2)  <Movies>
3)      <Movie title = "King Kong">
4)          <Version year = "1933">
5)              <Star>Fay Wray</Star>
6)          </Version>
7)          <Version year = "1976">
8)              <Star>Jeff Bridges</Star>
9)              <Star>Jessica Lange</Star>
10)         </Version>
11)         <Version year = "2005" />
12)     </Movie>
13)     <Movie title = "Footloose">
14)         <Version year = "1984">
15)             <Star>Kevin Bacon</Star>
16)             <Star>John Lithgow</Star>
17)             <Star>Sarah Jessica Parker</Star>
18)         </Version>
19)     </Movie>
20) </Movies>
```

Example XML - Movie

- /Movies/Movie[//Star="Jessica Lange"]/@title
- /Movies/Movie/Version[1]/@year
- /Movies/Movie/Version[Star]
- /Movies/Movie/Version[@year]

```
1)  <? xml version="1.0" encoding="utf-8" standalone="yes" ?>
2)  <Movies>
3)      <Movie title = "King Kong">
4)          <Version year = "1933">
5)              <Star>Fay Wray</Star>
6)          </Version>
7)          <Version year = "1976">
8)              <Star>Jeff Bridges</Star>
9)              <Star>Jessica Lange</Star>
10)         </Version>
11)         <Version year = "2005" />
12)     </Movie>
13)     <Movie title = "Footloose">
14)         <Version year = "1984">
15)             <Star>Kevin Bacon</Star>
16)             <Star>John Lithgow</Star>
17)             <Star>Sarah Jessica Parker</Star>
18)         </Version>
19)     </Movie>
20) </Movies>
```

XPath Queries - 1

- Select the root element:
  - doc("abc.xml")/A

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 2

- Select the text content of C elements:
  - doc("abc.xml")//C/text()

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 3

- Select the B elements that are children of some D elements
  - doc("abc.xml")//D/B

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 4

- Select all elements that enclosed by the path A/C/D
  - doc("abc.xml")/A/C/D/*

Example

```
<A>
   <B></B>
   <C>John</C>
   <B></B>
   <D>
     <B></B>
   </D>
   <C>
     <D>
       <E>Mary</E>
     </D>
   </C>
</A>
```

XPath Queries - 5

- Select all elements
  - doc("abc.xml")//*

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 6

- Select the first B child of every D elements
  - doc("abc.xml")//D/B[1]

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
      <B>123</B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 7

- Select the last B child of every D elements
  - doc("abc.xml")//D/B[last()]

Example

```
<A>
    <B></B>
    <C>John</C>
    <B></B>
    <D>
      <B></B>
      <B>123</B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 8

- Select all id attributes
  - doc("abc.xml")//@id

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <B id="B2"></B>
    <D id="D1">
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 9

- Select B elements having an id attribute
  - doc("abc.xml")//B[@id]

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <B id="B2"></B>
    <D id="D1">
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 10

- Select B elements having some attribute
  - doc("abc.xml")//B[@*]

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <D id="D1">
      <B id="B2"></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 11

- Select B elements without attributes
  - doc("abc.xml")//B[not(@*)]

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <B id="B2"></B>
    <D id="D1">
      <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 12

- Select B elements with an id attribute with value “B1”
  - doc(“abc.xml”)//B[@id=“B1”]

Example

```
<A>
   <B id="B1"></B>
   <C>John</C>
   <B id="B2"></B>
   <D id="D1">
     <B></B>
   </D>
   <C>Mary</C>
</A>
```

XPath Queries - 13

- Select elements that have two B elements
  - doc("abc.xml")//*[count(B)=2]

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <D id="D1">
     <B></B>
     <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 14

- Select B and C elements
  - doc("abc.xml")//B | doc("abc.xml")//C

Example

```
<A>
    <B id="B1"></B>
    <C>John</C>
    <D id="D1">
     <B></B>
     <B></B>
    </D>
    <C>Mary</C>
</A>
```

XPath Queries - 15

- Select the parents of some B element
  - doc("abc.xml")//B/parent::*

Example

```
<A>
    <B id="B1"></B>
    <C>
        <D id="D1">
            <B></B>
            <B></B>
        </D>
    </C>
</A>
```

XPath Queries - 16

- Select the ancestor of some B element
  - doc("abc.xml")//B/ancestor::*

Example

```
<A>
    <B id="B1"></B>
    <C>
        <D id="D1">
            <B></B>
            <B></B>
        </D>
    </C>
</A>
```

XPath Queries - 17

- Select the elements following B elements
  - doc("abc.xml")//B/following::*

Example

```
<A>
   <B id="B1"></B>
   <C>
      <D id="D1">
        <B></B>
        <B></B>
      </D>
   </C>
</A>
```

XPath Queries - 18

- Select the intersection of B elements on certain paths
  - doc("abc.xml")/A//C//B intersect doc("abc.xml")/A/C/B

Example

```
<A>
   <B id="B1"></B>
   <C>
      <B></B>
      <B></B>
      <D id="D1">
        <B></B>
        <B></B>
      </D>
   </C>
</A>
```

XPath Queries - 19

- Select the set difference of B elements on certain paths
  - doc("abc.xml")/A//B except doc("abc.xml")/A/C/B

Example

```
<A>
    <B id="B1"></B>
    <C>
      <B></B>
      <B></B>
    </C>
</A>
```