Tutorial – XPath, XQuery

CSCC43 - Introduction to Databases

XPath Terminology

- **Node**
  - document root, element, attribute, text, comment, ...
- **Relationship**
  - parent, child, sibling, ancestor, descendent, …
- *Exercise*: *Identify nodes and relationships in following xml document*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
     <!-- a bookstore database -->
    <book isbn=“111111” cat=“fiction”>
      <!-- a particular book -->
      <title lang=“chn”>Harry Potter</title>
      <price unit=“us”>79.99</price>
    </book>
    <book  isbn=“222222” cat=“textbook”>
      <title lang=“eng”>Learning XML</title>
      <price unit=“us”>69.95</price>
    </book>
    <book isbn="333333" cat="textbook">
      <title lang="eng">Intro. to Databases</title>
      <price unit="usd">39.00</price>
    </book>
</bookstore>
```

document root does not correspond to anything in the document

Node selector

Expression	**Description**
/	Selects the *document root* node (absolute path)
node	Selects the node (relative path)
//	Selects all descendent nodes of the current node that match the selection
.	Selects the current node
..	Selects the parent of the current node
@	Selects attribute nodes

Node selector: exercise

Result	**Path Expression**
Selects the *document root* node	? ?
Selects the *bookstore element* node	? ?
Selects all *book element* nodes	? ?
Selects all *price element* nodes	? ?
Selects all *lang attribute* nodes	?
?	./././.
?	/bookstore//@lang/../..
?	./book/tilte/@lang

Node selector : exercise sol

Result	**Path Expression**
Selects the *document root* node	**/**
	/.
Selects the *bookstore element* node	**/bookstore**
	./bookstore
Selects all *book element* nodes	**/bookstore/book**
	//book
Selects all *price element* nodes	**bookstore/book/price**
	//price
Selects all *lang attribute* nodes	**//@lang**
Selects the *document root* node	./././.
Selects all the *book element* nodes	/bookstore//@lang/../..
Selects the empty set	./book/tilte/@lang

Node selector: more exercise

Result	**Path Expression**
Selects *text* nodes of all *price element* nodes	?
Select all child nodes of *book element* nodes	?
Select all *comment* nodes	?
Select all nodes except attribute nodes	?
Select all attribute nodes	?
?	/bookstore/book/text()
?	/bookstore/book/title/..//@*

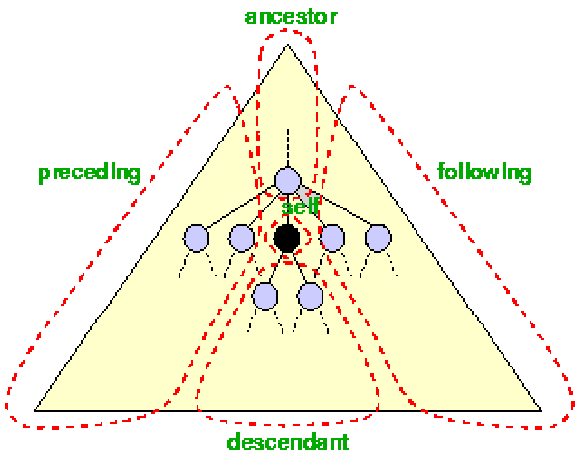
Node selector: more exercise sol

Result	**Path Expression**
Selects *text* nodes of all *price element* nodes	**//price/text()**
Select all child nodes of *book element* nodes	**/bookstore/book/***
Select all *comment* nodes	**//comment()**
Select all nodes except attribute nodes	**//node()**
Select all attribute nodes	**//@***
Selects empty set	/bookstore/book/text()
Select all attribute nodes which are descendant of *book element* nodes	/bookstore/book/title/..//@*

XPath Syntax and Semantics

- Syntax
  - locationStep1/locationStep2/…
    - locationStep = axis::nodeSelector[predicate]
- Semantics
  - Find all nodes specified by locationStep1
    - Find all nodes specified by axis::nodeSelector
    - Select only those that satisfy predicate
  - For each such node N:
    - Find all nodes specified by locationStep2 using N as the current node
    - Take union
  - For each node returned by locationStep2 do the same using locationStep3, …

Complete set of Axes



- ***self*** -- the context node itself
- ***child*** -- the children of the context node
- ***descendant*** -- all descendants (children+)
- ***parent*** -- the parent (empty if at the root)
- ***ancestor*** -- all ancestors from the parent to the r
- ***descendant-or-self*** -- the union of descendant and self
- ***ancestor-or-self*** -- the union of ancestor and self

- ***following-sibling*** -- siblings to the right
- ***preceding-sibling*** -- siblings to the left
- ***following*** -- all following nodes in the document, excluding descendants

- ***preceding*** -- all preceding nodes in the document, excluding ancestors
- ***attribute*** -- the attributes of the context node

Axes: exercise

Result	**Path Expression**
Selects *book element* nodes	?
Select all *isbn attribute* nodes	?
Select *title* **and** *price element* nodes	?
?	/child::book
?	/bookstore/book/following-sibling::book
?	/bookstore/node()/descendant-or-self::node()
?	/descendant::title/@*/parent::title/following::node()

Axes: exercise (sol)

Result	**Path Expression**
Selects *book element* nodes	**/descendant::book**
Select all *isbn attribute* nodes	**//book/attribute::isbn**
Select *title* and *price element* nodes	**//book/title | //book/price**
Selects empty set	/child::book
Selects the second *book element* node	/bookstore/book/following-sibling::book
Select all nodes (except attributes) that are descendants of the *bookstore element* node	/bookstore/node()/descendant-or-self::node()
Select all nodes (except attributes) after the first title node	/descendant::title/@*/parent::title/following::node()

Predicate: summary

- ***[position() op #], [last()]***
  - op: =, !=, <, >, <=, >=
  - test position among siblings
- ***[attribute::name op “value"]***
  - op: =, !=, <, >, <=, >=
  - test equality of an attribute
- ***[axis:nodeSelector]***
  - test pattern

Predicate: exercise

Result	**Path Expression**
Selects the first *book element* that is the child of the bookstore element.	?
	?
Select *book element* nodes which has a child *title element* with *lang* attribute value no equal to “eng”.	?
Selects the second to last *book element*	?
Selects all nodes which have an attr	?
Selects nodes with an attribute named *lang* or has a child element named *price.*	?
Selects all the *title element* of all *book elements* with a price greater than 35.00	/bookstore/book[price>35.00]/title
?	/bookstore/book[position()>1 and attribute::isbn="111111"]
?	/bookstore/book/title[last()]

Predicate: exercise sol

Result	**Path Expression**
Selects the first *book element* that is the child of the bookstore element.	**/bookstore/book[1]**
	/bookstore/book[position()=1]
Select *book element* nodes which has a child *title element* with *lang* attribute value no equal to “eng”.	**/bookstore/book[child::title/attribute::lang!="eng"]**
Selects the second to last *book element*	**/bookstore/book[last()-1]**
Selects all nodes which have an attr	**//node()[@*]**
Selects nodes with an attribute named *lang* or has a child element named *price.*	**//node()[@lang or child::price]**
Selects all the *title element* of all *book elements* with a price greater than 35.00	/bookstore/book[price>35.00]/title
Select the empty set	/bookstore/book[position()>1 and attribute::isbn="111111"]
Select the last *title element* node of all *book element* nodes	/bookstore/book/title[last()]

XPath: exercise

- *Question:* ***find the title and price of non fiction books with a price more than 50 USD.***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
     <!-- a bookstore database -->
    <book isbn=“111111” cat=“fiction”>
      <!-- a particular book -->
      <title lang=“chn”>Harry Potter</title>
      <price unit=“us”>79.99</price>
    </book>
    <book  isbn=“222222” cat=“textbook”>
      <title lang=“eng”>Learning XML</title>
      <price unit=“us”>69.95</price>
    </book>
    <book isbn="333333" cat="textbook">
      <title lang="eng">Intro. to Databases</title>
      <price unit="usd">39.00</price>
    </book>
</bookstore>
```

- *Answer:*
  - */bookstore/book[attribute::cat!="fiction" and price>50.00]/title | /bookstore/book[attribute::cat!="fiction" and price>50.00]/@isbn*

XPath: exercise

- *Question:* ***find average price of textbooks.***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <!-- a bookstore database -->
   <book isbn=“111111” cat=“fiction”>
     <!-- a particular book -->
     <title lang=“chn”>Harry Potter</title>
     <price unit=“us”>79.99</price>
   </book>
   <book  isbn=“222222” cat=“textbook”>
     <title lang=“eng”>Learning XML</title>
     <price unit=“us”>69.95</price>
   </book>
   <book isbn="333333" cat="textbook">
     <title lang="eng">Intro. to Databases</title>
     <price unit="usd">39.00</price>
   </book>
</bookstore>
```

- *Answer:*
  - ***sum****(/bookstore/book[attribute::cat="textbook"]/price/number(text())) div* ***count****(/bookstore/book[attribute::cat="textbook"]/price)*

XPath: exercise

- *Question:* ***find the titles of textbooks on XML.***

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <!-- a bookstore database -->
   <book isbn=“111111” cat=“fiction”>
     <!-- a particular book -->
     <title lang=“chn”>Harry Potter</title>
     <price unit=“us”>79.99</price>
   </book>
   <book  isbn=“222222” cat=“textbook”>
     <title lang=“eng”>Learning XML</title>
     <price unit=“us”>69.95</price>
   </book>
   <book isbn="333333" cat="textbook">
     <title lang="eng">Intro. to Databases</title>
     <price unit="usd">39.00</price>
   </book>
</bookstore>
```

- *Answer:*
  - */bookstore/book[attribute::cat="textbook" and contains(title, "XML")]/title/text()*

XQuery Example

Q1: *Create a new document which contain only the isbn and title of textbooks.*

```
<textbooks>
    { for $book in doc("bookstore.xml")//book
     where $book/@cat="textbook"
     return <textbook isbn="$book/@isbn">{$book/title}</textbook>
    }
</textbooks>
```

Result:

```
<textbooks>
    <textbook isbn="222222">
      <title lang="eng">Learning XML</title>
    </textbook>
    <textbook isbn="333333">
      <title lang="eng">Intro. to Databases</title>
    </textbook>
</textbooks>
```

XQuery Syntax and Semantics

- Syntax (FLWR)

for	*variable bindings*	(like from in SQL)
let	*variable bindings*	(like from in SQL)
where	*condition*	(like where in SQL*)*
return	*document*	(like select in SQL*)*

- Semantics
  - The for and let clause binds variables to elements specified by an XQuery expression.
    - for: bind a variable to each element in the returned set
    - let: bind a variable to the whole set of elements
  - Filter out nodes that do not satisfy the condition of the where clause .
  - For each retained tuple of bindings, instantiate the return clause.

XQuery Example Again

```
<textbooks>
    { for $book in doc("bookstore.xml")//book
      where $book/@cat="textbook"
      return <textbook isbn="$book/@isbn">{$book/title}</textbook>
    }
</textbooks>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <!-- a bookstore database -->
    <book isbn="111111" cat="fiction">
      <!-- a particular book -->
      <title lang="chn">Harry Potter</title>
      <price unit="us">79.99</price>
    </book>
    <book isbn="222222" cat="textbook">
      <title lang="eng">Learning XML</title>
      <price unit="us">69.95</price>
    </book>
    <book isbn="333333" cat="textbook">
      <title lang="eng">Intro. to Databases</title>
      <price unit="usd">39.00</price>
    </book>
</bookstore>
```

```
<textbooks>
  <textbook isbn="222222">
    <title lang="eng">Learning XML</title>
  </textbook>
  <textbook isbn="333333">
    <title lang="eng">Intro. to
Databases</title>
  </textbook>
</textbooks>
```

XQuery Example Modified

Q2:

```
<textbooks>
    { let $book := doc("bookstore.xml")//book
      where $book/@cat="textbook"
      return <textbook isbn="$book/@isbn">{$book/title}</textbook>
    }
</textbooks>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <!-- a bookstore database -->
    <book isbn="111111" cat="fiction">
       <!-- a particular book -->
       <title lang="chn">Harry Potter</title>
       <price unit="us">79.99</price>
    </book>
    <book  isbn="222222" cat="textbook">
       <title lang="eng">Learning XML</title>
       <price unit="us">69.95</price>
    </book>
    <book isbn="333333" cat="textbook">
       <title lang="eng">Intro. to Databases</title>
       <price unit="usd">39.00</price>
    </book>
</bookstore>
```

```
<textbooks>
  <textbook isbn="111111 222222 333333">
    <title lang="chn">Harry Potter</title>
    <title lang="eng">Learning XML</title>
    <title lang="eng">Intro. to Databases</title>
  </textbook>
</textbooks>
```

XQuery Exercise - Basic

Q3: *Find the title and price of the book with isbn “222222”*

```
for $book in doc("bookstore.xml")//book
where $book[@isbn="222222"]
return <book>{ $book/title, $book/price}</book>
```

Result:

```
<book>
    <title lang="eng">Learning XML</title>
    <price unit="usd">69.95</price>
</book>
```

XQuery Exercise - Ordering

Q4: *Produce a list of non-fictions with their title and price, sorted by price.*

```
<nonfiction-list>
    { for $book in doc("bookstore.xml")//book, $title in $book/title, $price in $book/price
      where $book/@cat!="fiction"
      order by $price/text()
      return <nonfiction>{ $title, $price}</nonfiction>
    }
</nonfiction-list>
```

Result:

```
<nonfiction-list>
    <nonfiction>
      <title lang="eng">Intro. to Databases</title>
      <price unit="usd">39.00</price>
    </nonfiction>
    <nonfiction>
      <title lang="eng">Learning XML</title>
      <price unit="usd">69.95</price>
    </nonfiction>
 </nonfiction-list>
```

XQuery Exercise - Aggregation

Q5: Find title of the the textbook with highest price.

```
<textbooks>
{ let $prices := doc("bookstore.xml")//book[@cat="textbook"]/price
  let $max := max($prices)
  return
        <max-price-textbook price="{$max}">
                          {for $book in doc("bookstore.xml")//book
                           where $book/price = $max
                           return $book/title
                           }
        </max-price-textbook>
 }
</textbooks>
```

Result:

```
<textbooks>
    <max-price-textbook price="69.95">
        <title lang="eng">Learning XML</title>
    </max-price-textbook>
</textbooks>
```

XQuery Exercise - Restructuring

Q6: *Restructure the document to organize books by categories.*

```
<summary-by-category>
      { let $categories :=
          for $category in doc("bookstore.xml")//book/@cat
          return $category
      for $cat in distinct-values($categories)
      return
          <category id="{$cat}">
                            { for $book in doc("bookstore.xml")//book
                                          where $book[@cat = $cat]
                                          return $book }
          </category>
}
</summary-by-category>
```

Result:

```
<bookstore>
      <book isbn="111111" cat="fiction">
            <title lang="chn">Harry Potter</title>
            <price unit="us">79.99</price>
      </book>
      <book  isbn="222222" cat="textbook">
            <title lang="eng">Learning
XML</title>
            <price unit="us">69.95</price>
      </book>
      <book isbn="333333" cat="textbook">
            <title lang="eng">Intro. to
Databases</title>
            <price unit="usd">39.00</price>
      </book>
</bookstore>
```

```
<summary-by-category>
      <category id="fiction">
            <book isbn="111111" cat="fiction">
                  <title lang="chn">Harry Potter</title>
                  <price unit="usd">79.99</price>
            </book>
      </category>
      <category id="textbook">
            <book isbn="222222" cat="textbook">
                  <title lang="eng">Learning XML</title>
                  <price unit="usd">69.95</price>
            </book>
            …
      </category>
</summary-by-category>
```

XQuery Exercise - Restructuring

Q7: *Restructure the document to produce the total price and count of books in each category.*

```
<price-by-category>
    { let $categories :=
        for $category in doc("bookstore.xml")//book/@cat
        return $category
      for $cat in distinct-values($categories)
      return
        <category id="{$cat}">
                        { let $prices-in-cat := doc("bookstore.xml")//book[@cat=$cat]/price
                                        return <price total="{sum($prices-in-cat)}" count="{count($prices-in-cat)}"/>
                        }
        </category>
    }
</price-by-category>
```

Result:

```
<bookstore>
        <book isbn="111111" cat="fiction">
                <title lang="chn">Harry Potter</title>
                <price unit="us">79.99</price>
        </book>
        <book  isbn="222222" cat="textbook">
                <title lang="eng">Learning
XML</title>
                <price unit="us">69.95</price>
        </book>
        <book isbn="333333" cat="textbook">
                <title lang="eng">Intro. to
Databases</title>
                <price unit="usd">39.00</price>
        </book>
</bookstore>
```

$\longrightarrow$

```
<price-by-category>
    <category id="fiction">
        <price total="79.99" count="1" />
    </category>
    <category id="textbook">
        <price total="108.95" count="2" />
    </category>
</price-by-category>
```