XML and XML Querying

COSC 304 – Introduction to Database Systems

XML

Extensible Markup Language (***XML***) is a markup language that allows for the description of data semantics.

- XML can describe any type of data because the markup terms are user-defined.
- XML is **case-sensitive** unlike HTML.
- XML is a standard by the World Wide Web Consortium (W3C).

Advantages of XML:

- Simplicity, open standard, extensibility, interoperability, separation of data and presentation

XML Components

An XML document is a text document that contains markup in the form of ***tags***. An XML document consists of:

- An XML *declaration line* indicating the XML version.
- *Elements* (or tags) called *markup*. Each element may contain free-text, attributes, or other nested elements.
  - Every XML document has a single root element.
  - Tags, as in HTML, are matched pairs, as <item> … </item>.
  - Closing tags are not needed if the element contains no data: <item/>
  - Tags may be nested.
- An *attribute* is a name-value pair declared inside an element.
- Comments

XML data is ordered by nature.

XML Example

```
<?xml version = "1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="dept.xsl"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="https://myloc/schema.xsd">
<!-- Emp/Dept in XML -->
 <Dept dno = "D1">
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept></root>
```

XML declaration

Stylesheet for presentation

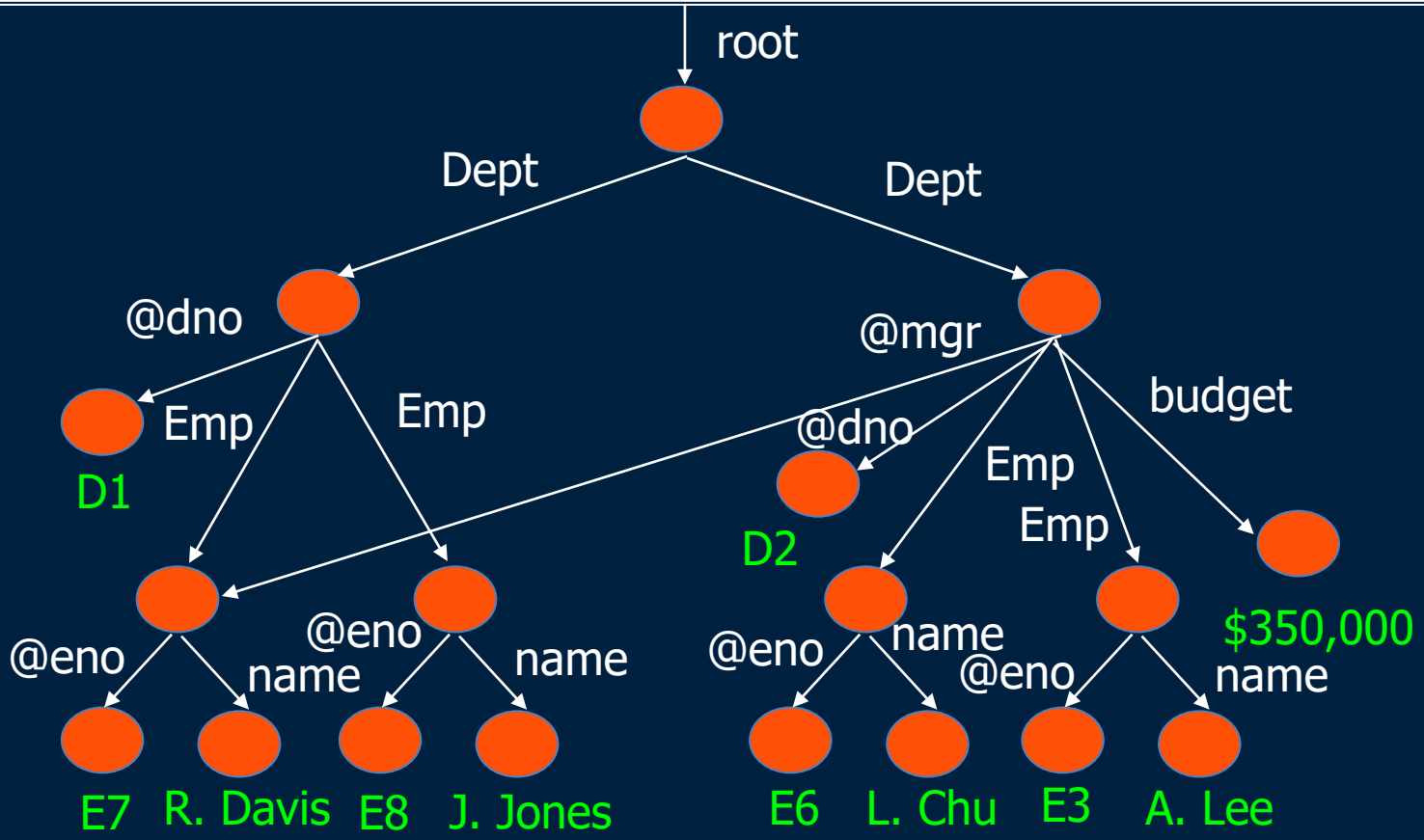
XML Schema for validation

Attribute

Comment

Element reference

XML (tree view)



Well-Formed and Valid XML Documents

An XML document is ***well-formed*** if it obeys the syntax of the XML standard. This includes:

- Having a single root element
- All elements must be properly closed and nested.

An XML document is ***valid*** if it is well-formed and it conforms to a Document Type Definition (DTD) or an XML Schema Definition (XSD).

- A document can be well-formed without being valid if it contains tags or nesting structures that are not allowed in its DTD/XSD.
- The DTD/XSD are schema definitions for an XML document.

XML Well-Formed Question

Question: How many of these two documents are well-formed?

1: `<x><a>Test</a><bT></bt></x>`

2: `<x>abc</x><y>def</y>`

A) 0

B) 1

C) 2

Namespaces

Namespaces allow tag names to be qualified to avoid naming conflicts. A naming conflict would occur when the same name is used by two different domains or vocabularies.

A namespace consists of two components:

- 1) A declaration of the namespace and its abbreviation.
- 2) Prefixing tag names with the namespace name to exactly define the tag's origin.

Namespaces Example

```
<?xml version = "1.0" encoding="UTF-8" standalone="no"?>
<root xmlns = "http://www.foo.com"
      xmlns:n1 = "http://www.abc.com">
 <Dept dno = "D1">
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <n1:budget>350000</n1:budget>
 </Dept>
</root>
```

Default namespace

n1 namespace

budget is a XML tag in the n1 namespace.

Schemas for XML

Although an unrestricted XML format is useful to some applications, database data normally has some structure, even though that structure may not be as rigid as relational schemas.

It is valuable to define schemas for XML documents that restrict the format of those documents.

Two ways of specifying a schema for XML:

- XML Schema
- Document Type Definition (DTD) (original, older)

Document Type Definitions (DTDs)

A ***Document Type Definition*** (***DTD***) defines the grammatical rules for the document. It is not required for an XML document but provides a mechanism for checking a document's validity. General DTD form:

```
<!DOCTYPE myroot [ <elements> ]>
```

A DTD is a set of document rules expressed using EBNF (Extended Backus-Naur Form) grammar. The rules limit:

- the set of allowable element names, how elements can be nested, and the attributes of an element among other things

DTD Example

```
<!DOCTYPE root [
 <!ELEMENT root(Dept+)>
 <!ELEMENT Dept(Emp*, budget?)>
       <!ATTLIST Dept dno ID #REQUIRED>
       <!ATTLIST Dept mgr IDREF #IMPLIED>
 <!ELEMENT budget (#PCDATA)>
 <!ELEMENT Emp (name)>
       <!ATTLIST Emp eno ID #REQUIRED>
 <!ELEMENT name (#PCDATA)>
]>
```

+ means 1 or more times

* means 0 or more times

? means 0 or 1 time

Element reference
(like a foreign key)

ID is a unique value that identifies the element

Parsed Character Data
(atomic value)

XML DTD Question

Question: How many of the following statements are **TRUE**?

1) Every XML document requires a DTD.
2) A document can be valid even if it does not have a DTD or XML Schema.
3) A + means 1 or more times.
4) A ? means 0 or more times.
5) A * means 0 or 1 times.
6) The order of elements listed in a DTD matters.

A) 1 **B)** 2 **C)** 3 **D)** 4 **E)** 5

XML Schema

XML Schema was defined by W3C to provide a standard XML schema language written in XML with better support for data modeling.

XML Schema also allows you to:

- Define simple and complex data types
- Define groups of elements or attributes
- Define cardinality (# of occurrences of elements)
- Define key and uniqueness constraints and reference constraints
- Define schema references to simplify schema maintenance
- Use groups to allow for schema variations and choices
- Construct elements that are lists or unions of values

XML Schema Example

```
<?xml version = "1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name = "root">
 <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="Dept" minOccurs="1" maxOccurs="unbounded">
       <xsd:complexType>
         <xsd:sequence>
           <xsd:element name="Emp" minOccurs="0"
                                   maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
```

Root element is called `root`

Complex type contains other elements

Min and max number occurrences

XML Schema Example (2)

```
            <xsd:element name = "name" type = "xsd:string" />
          </xsd:sequence>
          <xsd:attribute name = "eno" type = "xsd:string" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="budget" minOccurs="0"
                                      type ="xsd:decimal" />
    </xsd:sequence>
    <xsd:attribute name = "dno" type = "xsd:string" />
    <xsd:attribute name = "mgr" type = "xsd:string" />
  </xsd:complexType>
 </xsd:element>
</xsd:sequence>
</xsd:complexType>
```

Simple type (has data type)

XML Schema Example (3)

```
<xsd:key name = "DeptKey">
        <xsd:selector xpath = "Dept" />
        <xsd:field xpath = "@dno" />
</xsd:key>
<xsd:key name = "EmpKey">
        <xsd:selector xpath = "Dept/Emp" />
        <xsd:field xpath = "@eno" />
</xsd:key>
<xsd:keyref name = "DeptMgrFK" refer = "EmpKey">
        <xsd:selector xpath = "Dept" />
        <xsd:field xpath = "@mgr" />
</xsd:keyref>
</xsd:element></xsd:schema>
```

Key constraints

Reference to another key (like a FK)

Other XML Technologies

An ***XML parser*** processes the XML document and determines if it is well-formed and valid (if a schema is provided).

Once a document is parsed, programs manipulate the document using one of two interfaces: DOM (tree-based) and SAX (event-based).

- Note: May process XML documents without a parser as document is a text file.

XSL (eXtenstible Stylesheet Language) defines how XML data is displayed.

- Similar to Cascading Stylesheet Specification (CSS) used with HTML.

XSLT (eXtenstible Stylesheet Language for Transformations) is a subset of XSL that provides a method for transforming XML (or other text documents) into other documents (XML, HTML).

Querying XML

XPath allows you to specify path expressions to navigate the tree-structured XML document.

XQuery is a full query language that uses XPath for path expressions (not studied).

Example XML Document

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Path Descriptions in XPath

XPath provides the ability to navigate through a document using path descriptors.

Path descriptors are sequences of tags separated by slashes /.

- If the descriptor begins with `/`, then the path starts at the root.
- If the descriptor begins with `//`, the path can start anywhere.
- You may also start the path by giving the document name such as `doc(depts.xml)/.`

A path descriptor denotes a sequence of nodes. Examples:

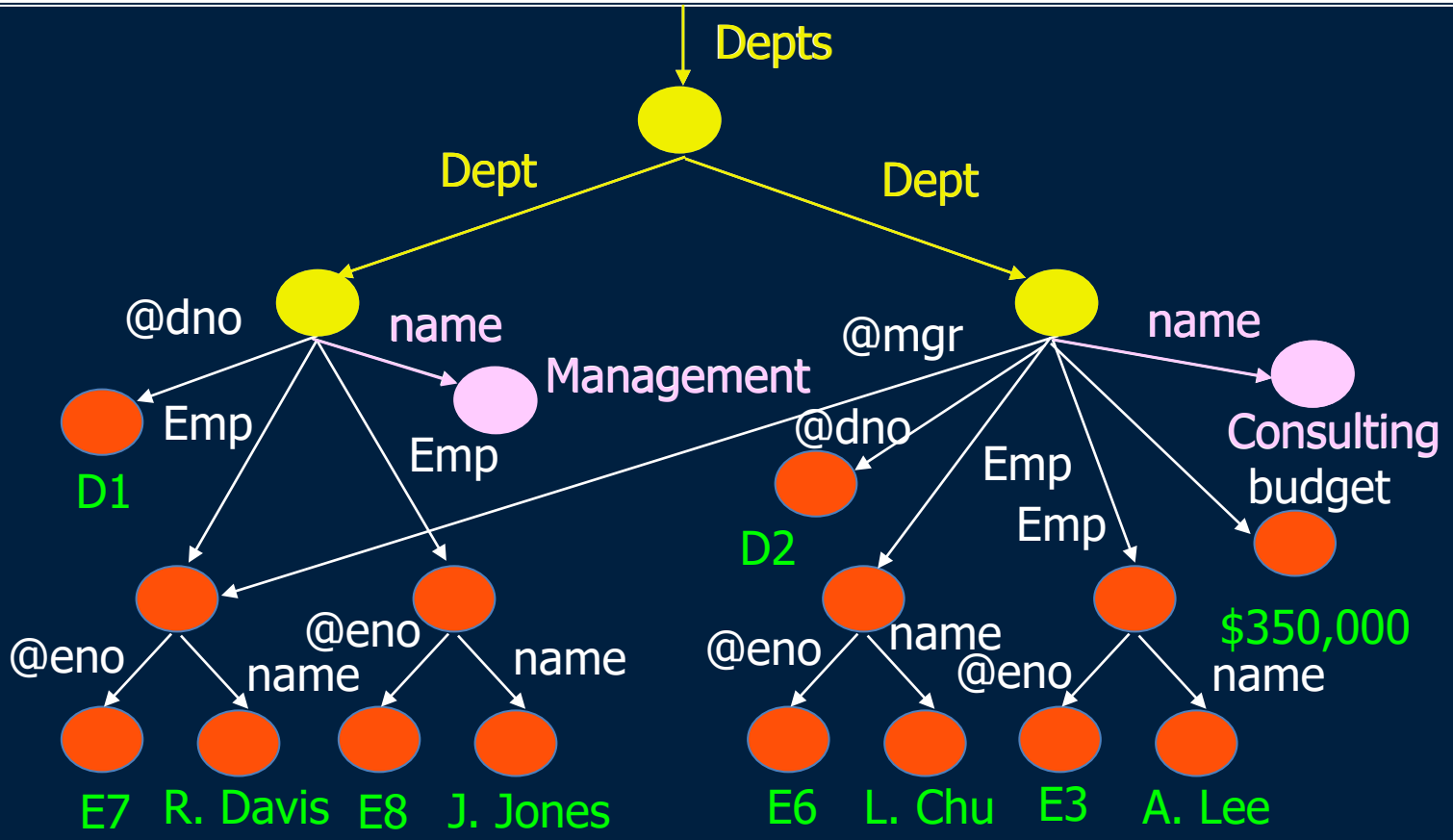
- `/Depts/Dept/name`
- `//Dept/name`
- `doc("depts.xml")/Depts/Dept/Emp/name`

Path: /Depts/Dept/name

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Path: /Depts/Dept/name (tree view)



Path: //Dept/name

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Path query returns same answer as previous one.

Path: //name

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Matches any name tag starting from anywhere in the document.

Path: /Depts/Dept

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
<Dept dno = "D1">
      <name>Management</name>
      <Emp eno="E7"><name>R. Davis</name></Emp>
      <Emp eno="E8"><name>J. Jones</name></Emp>
</Dept>
<Dept dno = "D2" mgr = "E7">
      <name>Consulting</name>
      <Emp eno="E6"><name>L. Chu</name></Emp>
      <Emp eno="E3"><name>A. Lee</name></Emp>
      <budget>350000</budget>
</Dept>
</Depts>
```

Wild Card Operator

The "*" wild card operator can be used to denote any ***single*** tag.

Examples:

- `/*/*/name` - Match any name that is nested 3 levels deep
- `//*` - Match anything

Path: /*/*/name

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Same as /Depts/Dept/name

Question: What is /*/*/* ?

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

How many results in answer?

A) 0 **B)** 2 **C)** 7 **D)** 9

Attributes

Attributes are referenced by putting a "@" in front of their name.

Attributes of a tag may appear in paths as if they were nested within that tag.

Examples:

- `/Depts/Dept/@dno` - `dno` attribute of `Dept` element
- `//Emp/@eno` - `eno` attribute of `Emp` element

Path: /Depts/Dept/@dno

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Question: What is /*/*/@eno ?

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

How many results in answer?

A) 0 **B)** 2 **C)** 4 **D)** 5

Predicate Expressions

The set of objects returned can be filtered by putting selection conditions on the path.

A ***predicate expression*** may be specified inside square brackets **[..]** following a tag. Only paths that have that tag and also satisfy the condition are included in the result of a path expression.

Examples:

- `/Depts/Dept/name[.="Management"]`
- `//Depts/Dept[budget>250000]`
- `//Emp[@eno="E5"]`

//Depts/Dept/budget[.>250000]

```
<?xml version = "1.0" encoding="UTF-8" ?>

<Depts>
 <Dept dno = "D1">
       <name>Management</name>
       <Emp eno="E7"><name>R. Davis</name></Emp>
       <Emp eno="E8"><name>J. Jones</name></Emp>
 </Dept>
 <Dept dno = "D2" mgr = "E7">
       <name>Consulting</name>
       <Emp eno="E6"><name>L. Chu</name></Emp>
       <Emp eno="E3"><name>A. Lee</name></Emp>
       <budget>350000</budget>
 </Dept>
</Depts>
```

Note no budget element in first Dept so does not match path.

Axes and Abbreviations

XPath defines ***axes*** that allow us to go from the current node to other nodes. An axis to traverse is specified by putting the axis name before the tag name to be matched such as `child::Dept`.

Common axes have abbreviations:

- The default axis is `child::` which contains all children. Since it is the default, the child axis does not have to be explicitly specified.
  - `/Depts/Dept` is shorthand for `/Depts/child::Dept`
- **@** is a shorthand for the `attribute::` axis.
  - `/Depts/Dept/@dno` is short for `/Depts/Dept/attribute::dno`
- `..` is short for the `parent::` axis.
- `.` is short for the `self::` axis (current node).
- **//** is short for `descendant-or-self::` axis
  - // matches any node or any of its descendants

Summary of XPath Constructs

Symbol	Usage
/	Root element or separator between path steps
*	Match any single element name
@X	Match attribute X of current element
//	Match any descendant (or self) of current element
[C]	Evaluate condition on current element
[N]	Picks the N^{th} matching element (indexed from 1)
..	Matches parent element
.	Matches current element

DTD for Questions

```
<!DOCTYPE Bookstore [
 <!ELEMENT Bookstore (Book | Magazine)*>
 <!ELEMENT Book (Title, Authors, Remark?)>
 <!ATTLIST Book ISBN CDATA #REQUIRED>
 <!ATTLIST Book Price CDATA #REQUIRED>
 <!ATTLIST Book Edition CDATA #IMPLIED>
 <!ELEMENT Magazine (Title)>
 <!ATTLIST Magazine Month CDATA #REQUIRED>
 <!ATTLIST Year CDATA #REQUIRED>
 <!ELEMENT Title (#PCDATA)>
 <!ELEMENT Authors (Author+)>
 <!ELEMENT Remark (#PCDATA)>
 <!ELEMENT Author (First_Name, Last_Name)>
 <!ELEMENT First_Name (#PCDATA)>
 <!ELEMENT Last_Name (#PCDATA)>]
```

Example XML Document for Questions

```
<?xml version="1.0" encoding="UTF-8" ?>

<Bookstore>
 <Book ISBN="ISBN-0-201-70857-4" Price="65" Edition="3rd">
 <Title>Database Systems</Title>
 <Authors>
   <Author><First_Name>Thomas</First_Name><Last_Name>Connolly</Last_Name> </Author>
   <Author><First_Name>Carolyn</First_Name><Last_Name>Begg</Last_Name></Author>
  </Authors>
 </Book>
 <Book ISBN="ISBN-0-13-031995-3" Price="75">
 <Title>Database Systems: The Complete Book</Title>
 <Authors>
 <Author><First_Name>H.</First_Name><Last_Name>Garcia-Molina</Last_Name></Author>
  <Author><First_Name>Jeffrey</First_Name><Last_Name>Ullman</Last_Name> </Author>
  <Author> <First_Name>Jennifer</First_Name> <Last_Name>Widom</Last_Name> </Author>
 </Authors>
 <Remark> Amazon.com says: Buy these books together for a great deal!</Remark>
 </Book> </Bookstore>
```

XPath Questions

What are the elements selected by these XPath queries:

- `/Bookstore/*/Title`
- `//First_Name[.="Thomas"]`
- `//Last_Name[.="Ullman"]/../../..[@Price < 60]`

Write XPath queries to retrieve:

- all book titles
- all books < $70
- all last names anywhere
- all books containing a remark
- all book titles where the book < $80 and Ullman is an author
- retrieve the second book

Conclusion

Extensible Markup Language (***XML***) is a markup language that allows for the description of data semantics.

An XML document does not need a schema to be *well-formed*. An XML document is *valid* if it conforms to its schema (DTD or XML Schema).

XPath is a language for specifying paths through XML documents.

Objectives

- List some advantages of XML.
- Given an XML document, determine if it is well-formed.
- Given an XML document and a DTD, determine if it is valid.
- Know the symbols (?,*,+) for cardinality constraints in DTDs.
- Compare and contrast ID/IDREFs in DTDs with keys and foreign keys in the relational model.
- List some advantages that XML Schema has over DTDs.
- Explain why and when namespaces are used.
- Given an XML document and query description, write an XPath query to retrieve the appropriate node sequence to answer the query.
- Given an XML document and an XPath expression, list the result of evaluating the expression.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA