XML Data Management

Peter Wood

ADM 2013/4
Outline

1. Introduction
2. XML Fundamentals
3. XPath
4. XQuery
Chapter 1

Introduction
What is XML?

- The eXtensible Markup Language (XML) defines a generic syntax used to mark up data with simple, human-readable tags.
- Has been standardized by the World Wide Web Consortium (W3C) as a format for computer documents.
- Is flexible enough to be customized for domains as diverse as:
  - Web sites
  - Electronic data interchange
  - News feeds (RSS, e.g., BBC World News)
  - Vector graphics
  - Mathematical expressions
  - Microsoft Word documents
  - Music libraries (e.g., iTunes)
  - . . .
What is XML? (2)

- Data in XML documents is represented as strings of text
- This data is surrounded by text markup, in the form of *tags*, that describes the data
- A particular unit of data and markup is called an *element*
- XML specifies the exact syntax of how elements are delimited by tags, what a tag looks like, what names are acceptable, and so on
Which is Easier to Understand?

TCP/IP
Stevens
Foundations of Databases
Abiteboul
Hull
Vianu
The C Programming Language
Kernighan
Ritchie
Prentice Hall

...
XML vs. HTML

- Markup in an XML document looks similar to that in an HTML document.
- However, there are some crucial differences:
  - XML is a meta-markup language: it doesn’t have a *fixed* set of tags and elements.
  - To enhance interoperability, people may agree to use only certain tags (as defined in a DTD or an XML Schema).
  - Although XML is flexible in regard to elements that are allowed, it is strict in many other respects (e.g., closing tags are required).
  - Markup in XML only describes a document’s structure; it doesn’t say anything about how to display it.
XML versus Relational Data

- Why not use data from relational databases for exchange?
- XML is more flexible:
  - XML data is *semi-structured* rather than structured
  - Conformance of the data to a schema is not mandatory
  - XML schemas, if used, allow for more varied structures
- Relational data can always be (and often is) wrapped as XML
Future of XML

- XML offers the possibility of truly cross-platform, long-term data formats:
  - Much of the data from the original moon landings is now effectively lost
  - Even reading an older Word file might already be problematic

- XML is a very simple, well-documented data format

- Any tool that can read text files can read an XML document

- XML may be the most portable and flexible document format since the ASCII text file
XML and Querying

- XML is emerging as a universal format for data interchange among cross-platform applications.
- If an application is viewed as a source of information in XML format, it is logical to pose queries against this format.
- Therefore, (standardized) query languages for XML data are extremely important.
Yet Another Language?

Why do we need yet another query language specifically for XML?

- Ease-of-Use:
  - Domain-specific languages let you work with domain concepts directly (single line in an XML query language can accomplish the equivalent of hundreds of lines of code in C, C++, Java, or other general-purpose languages)
  - Declarative rather than descriptive (saying what you want, not how you want it done)
Yet Another Language? (2)

Why do we need yet another query language specifically for XML?

- Performance:
  - Optimized for special tasks, this may limit applicability, but often yields superior performance
  - Declarative nature allows sophisticated optimization in implementation without affecting language itself
XML Querying with SQL

Why not use an existing query language like SQL?

- SQL is a well-established standard for retrieving information from relational databases
- SQL has recently been enhanced with “structured types”
- Maybe SQL could be extended further to meet XML query requirements?
Advantages of Using SQL

- Developers could leverage considerable investments made in SQL implementations
- Users would not need to learn a completely new language
Disadvantages of Using SQL

- There is an impedance mismatch between relational data and XML data (the “world” is viewed completely different by them)
- Relational data is very regular, flat, and unordered (organized in tables with rows and columns)
- XML data is irregular, nested, and follows an intrinsic document order (organized in a tree-like structure)
What To Do?

- Significant data model differences led to the conclusion that new languages are needed
- Working group of W3C defined XML querying requirements and had a close look at SQL before reaching this conclusion
- Same working group started work on designing these new languages
In these lectures we are going to look at

- some basic notions of XML
- how to query XML documents (XPath, XQuery)
Literature

- H. Katz (editor). *XQuery from the Experts*. Addison Wesley, 2004
- These slides . . .
Chapter 2

XML Fundamentals
Elements, Tags, and Data

- A very simple, yet complete, XML document:

  ```xml
  <person>
    Alan Turing
  </person>
  ```

- Composed of a single *element* whose name is `person`

- Element is delimited by the *start tag* `<person>` and the *end tag* `</person>`

- Everything between the start tag and end tag (exclusive) is the element’s *content*
Elements, Tags, and Data (2)

- Content of the above element is the text string *Alan Turing*
- Whitespace is part of the content (although many applications choose to ignore it)
- `<person>` and `</person>` are *markup*,
- The string *Alan Turing* and surrounding whitespace are *character data*
Elements, Tags, and Data (3)

- Special syntax for *empty elements*, elements without content
  - Each can be represented by a *single* tag that begins with `<` but ends with `/>`
- XML is case sensitive, i.e. `<Person>` is not the same as `<PERSON>` (or `<person>`)
XML Documents and Trees

XML documents can be represented as trees

```
<person>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>
    computer scientist
  </profession>
  <profession>
    mathematician
  </profession>
</person>
```
XML Documents and Trees (2)

- The `person` element contains three `child` elements: one `name` and two `profession` elements.
- The `person` element is called the `parent` element of these three elements.
- An element can have an arbitrary number of child elements and the elements may be nested arbitrarily deeply.
- Children of the same parent are called `siblings`.
- Overlapping tags are prohibited, so the following is not possible:
  ```xml
  <strong><em>
  example from HTML
  </strong></em>
  ```
Every XML document has one element without a parent.

This element is called the document’s root element (sometimes called document element).

The root element contains all other elements of a document.
Attributes

- XML elements can have *attributes*
- An attribute is name-value pair attached to an element’s start tag
- Names are separated from values by an equals sign
- Values are enclosed in single or double quotation marks
- Example:
  ```xml
  <person born='1912/06/23' died='1954/06/07'>
    Alan Turing
  </person>
  ``
- The order in which attributes appear is not significant
Attributes (2)

- We could model the contents of the original document as attributes:

  ```xml
  <person>
    <name first='Alan' last='Turing'/>
    <profession value='computer scientist'/>
    <profession value='mathematician'/>
  </person>
  ```

- This raises the question of when to use child elements and when to use attributes

- There is no simple answer
Attributes vs. Child Elements

- Some people argue that attributes should be used for metadata (about the element) and elements for the information itself
  - It’s not always easy to distinguish between the two
- Attributes are limited in structure (their value is simply a string)
- There can also be no more than one attribute with a given name
- Consequently, an element-based structure is more flexible and extensible
Entities and Entity References

- Character data inside an element may not contain, e.g., a raw unescaped opening angle bracket `<
- If this character is needed in the text, it has to be escaped by using the `&lt;` *entity reference*
- `lt` is the *name* of the entity; `&` and `;` delimit the reference
- XML predefines five entities:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>lt</td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td>amp</td>
<td><code>&amp;</code></td>
</tr>
<tr>
<td>gt</td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td>quot</td>
<td><code>&quot;</code></td>
</tr>
<tr>
<td>apos</td>
<td><code>'</code></td>
</tr>
</tbody>
</table>
CDATA Sections

- When an XML document includes samples of XML or HTML source code, all `<`, `>`, and `&` characters must be encoded using entity references
- This replacement can become quite tedious
- To facilitate the process, literal code can be enclosed in a CDATA section
- Everything between `<! [CDATA [ and ] ] >` is treated as raw character data
- The only thing that cannot appear in a CDATA section is the end delimiter `]]>`
Comments

- XML documents can also be commented
- Similar to HTML comments, they begin with <!-- and end with -->
- Comments may appear
  - anywhere in character data
  - before or after the root element
  - However, NOT inside a tag or another comment
- XML parsers may or may not pass along information found in comments
Processing Instructions

- In HTML, comments are sometimes abused to support nonstandard extensions (e.g., server-side includes)
- Unfortunately,
  - comments may not survive being passed through several different HTML editors and/or processors
  - innocent comments may end up as input to an application
- XML uses a special construct to pass information on to applications: a *processing instruction*
- It begins with `<?` and ends with `?>`
- Immediately following the `<?` is the target (possibly the name of the application)
Processing Instructions (2)

Examples:

- Associating a stylesheet with an XML document:
  
  ```xml
  <?xml-stylesheet type="text/xsl" href="style.xsl"?>
  ```

- Embedded PHP in (X)HTML:
  
  ```php
  mysql_connect("database...",
  "user",
  "password");
  ...
  mysql_close();
  ?>
  ```
XML Declaration

- The XML declaration looks like a processing instruction, but only gives some information about the document:

```xml
<?xml version='1.0' encoding='US-ASCII' standalone='yes'?>
```

- **version**: at the moment 1.0 and 1.1 are available (we focus on 1.0)
- **encoding**: defines the character set used (e.g. ASCII, Latin-1, Unicode UTF-8)
- **standalone**: determines if some other file (e.g. DTD) has to be read to determine proper values for parts of the document
Summary

- This chapter gave a brief summary of XML
- Only the most important aspects (which are needed later on) were covered