CS276B

Text Retrieval and Mining
Winter 2005

Lecture 12

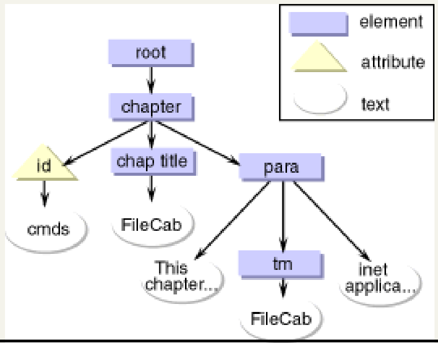
What is XML?

- eXtensible Markup Language
- A framework for defining markup languages
- No fixed collection of markup tags
- Each XML language targeted for application
- All XML languages share features
- Enables building of generic tools

Basic Structure

- An XML document is an **ordered, labeled tree**
- **character data** leaf nodes contain the actual data (text strings)
- **element nodes**, are each labeled with
  - a name (often called the element *type*), and
  - a set of **attributes**, each consisting of a name and a value,
  - can have child nodes

XML Example



XML Example

<chapter id="cmds">
<chaptitle>FileCab</chaptitle> <para>This chapter describes the commands that manage the <tm>FileCab</tm>inet application.</para> </chapter>

Elements

- Elements are denoted by markup tags
- <foo attr1="value" … > thetext </foo>
- Element start tag: foo
- Attribute: attr1
- The character data: thetext
- Matching element end tag: </foo>

XML vs HTML

- HTML is a markup language for a specific purpose (display in browsers)
- XML is a framework for defining markup languages
- HTML can be formalized as an XML language (XHTML)
- XML defines logical structure only
- HTML: same intention, but has evolved into a presentation language

XML: Design Goals

- **Separate syntax from semantics** to provide a common framework for structuring information
- Allow **tailor-made markup** for any imaginable application domain
- Support **internationalization** (Unicode) and **platform independence**
- Be the **future of (semi)structured information** (do some of the work now done by databases)

Why Use XML?

- Represent semi-structured data (data that are structured, but don’t fit relational model)
- XML is more flexible than DBs
- XML is more structured than simple IR
- You get a massive infrastructure for free

Applications of XML

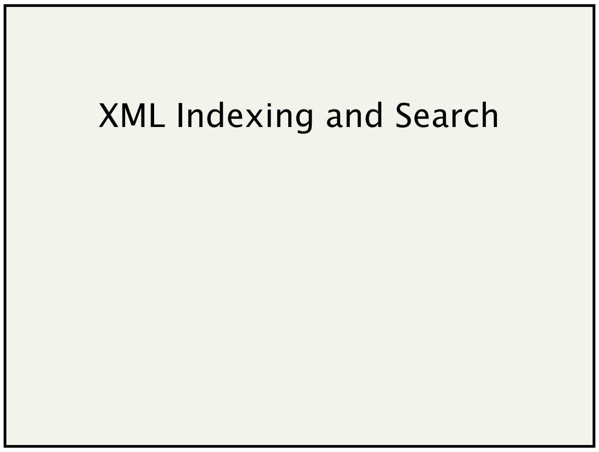
- XHTML
- CML – chemical markup language
- WML – wireless markup language
- ThML – theological markup language
  - <h3 class="s05" id="One.2.p0.2">Having a Humble Opinion of Self</h3> <p class="First" id="One.2.p0.3">EVERY man naturally desires knowledge <note place="foot" id="One.2.p0.4"> <p class="Footnote" id="One.2.p0.5"><added id="One.2.p0.6"> <name id="One.2.p0.7">Aristotle</name>, Metaphysics, i. 1. </added></p> </note>; but what good is knowledge without fear of God? Indeed a humble rustic who serves God is better than a proud intellectual who neglects his soul to study the course of the stars. <added id="One.2.p0.8"><note place="foot" id="One.2.p0.9"> <p class="Footnote" id="One.2.p0.10"> Augustine, Confessions V. 4. </p> </note></added> </p>

XML Schemas

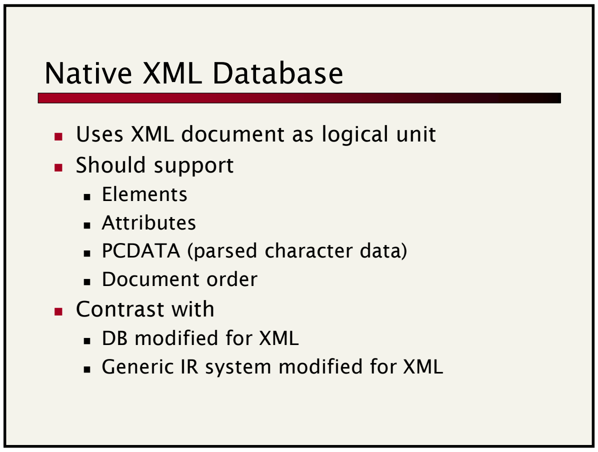
- Schema = syntax definition of XML language
- Schema language = formal language for expressing XML schemas
- Examples
  - Document Type Definition
  - XML Schema (W3C)
- Relevance for XML IR
  - Our job is much easier if we have a (one) schema

XML Tutorial

- http://www.brics.dk/~amoeller/XML/index.html
- (Anders Møller and Michael Schwartzbach)
- Previous (and some following) slides are based on their tutorial

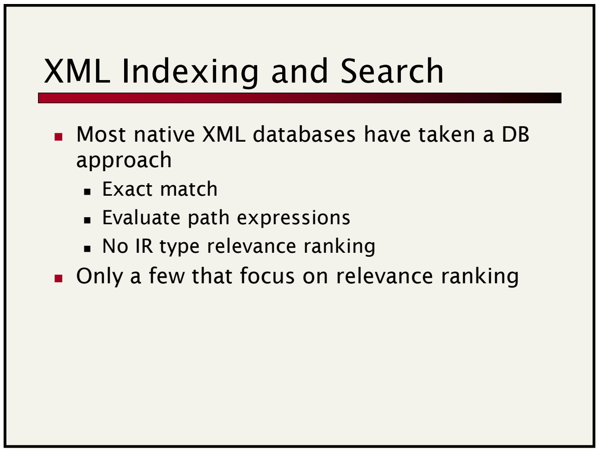


XML Indexing and Search



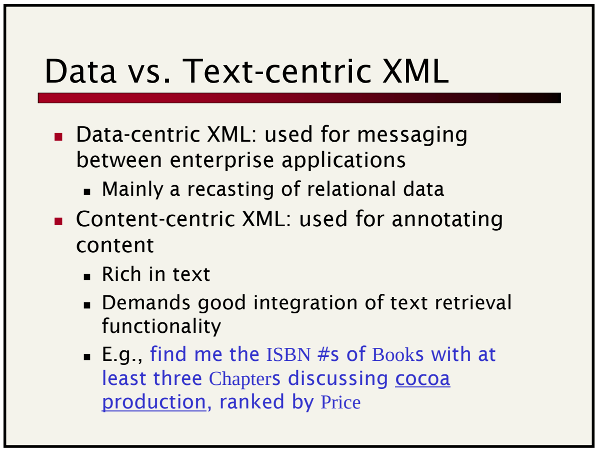
Native XML Database

- Uses XML document as logical unit
- Should support
  - Elements
  - Attributes
  - PCDATA (parsed character data)
  - Document order
- Contrast with
  - DB modified for XML
  - Generic IR system modified for XML



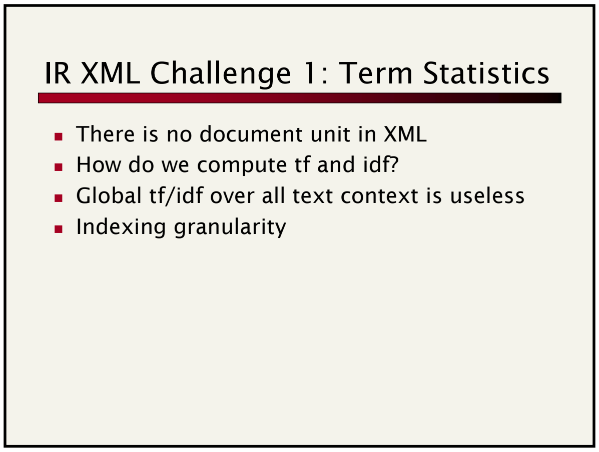
XML Indexing and Search

- Most native XML databases have taken a DB approach
  - Exact match
  - Evaluate path expressions
  - No IR type relevance ranking
- Only a few that focus on relevance ranking



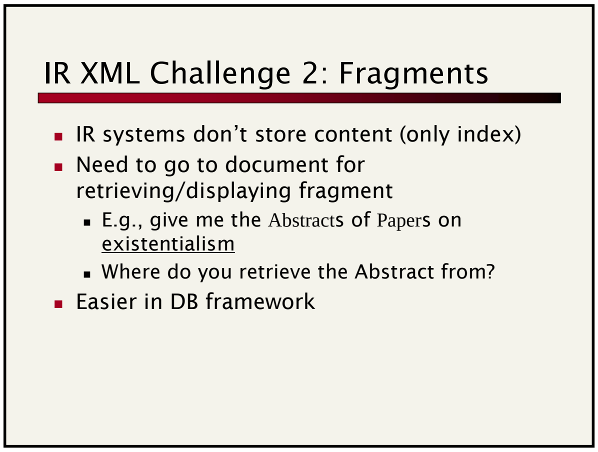
Data vs. Text-centric XML

- Data-centric XML: used for messaging between enterprise applications
  - Mainly a recasting of relational data
- Content-centric XML: used for annotating content
  - Rich in text
  - Demands good integration of text retrieval functionality
  - E.g., find me the ISBN #s of Books with at least three Chapters discussing cocoa production, ranked by Price



IR XML Challenge 1: Term Statistics

- There is no document unit in XML
- How do we compute tf and idf?
- Global tf/idf over all text context is useless
- Indexing granularity



IR XML Challenge 2: Fragments

- IR systems don't store content (only index)
- Need to go to document for retrieving/displaying fragment
  - E.g., give me the Abstracts of Papers on existentialism
  - Where do you retrieve the Abstract from?
- Easier in DB framework

IR XML Challenges 3: Schemas

- Ideally:
  - There is one schema
  - User understands schema
- In practice: rare
  - Many schemas
  - Schemas not known in advance
  - Schemas change
  - Users don't understand schemas
- Need to identify similar elements in different schemas
  - Example: employee

IR XML Challenges 4: UI

- Help user find relevant nodes in schema
  - Author, editor, contributor, "from:"/sender
- What is the query language you expose to the user?
  - Specific XML query language? No.
  - Forms? Parametric search?
  - A textbox?
- In general: design layer between XML and user

IR XML Challenges 5: using a DB

- Why you don't want to use a DB
  - Spelling correction
  - Mid-word wildcards
  - Contains vs "is about"
  - DB has no notion of ordering
  - Relevance ranking

Querying XML

- Today:
  - XQuery
  - XIRQL
- Lecture 15
  - Vector space approaches

XQuery

- SQL for XML
- Usage scenarios
  - Human-readable documents
  - Data-oriented documents
  - Mixed documents (e.g., patient records)
- Relies on
  - XPath
  - XML Schema datatypes
- Turing complete
- XQuery is still a working draft.

XQuery

- The **principal forms** of XQuery expressions are:
  - path expressions
  - element constructors
  - FLWR ("flower") expressions
  - list expressions
  - conditional expressions
  - quantified expressions
  - datatype expressions
- Evaluated with respect to a context

FLWR

- FOR $p IN document("bib.xml")//publisher LET $b := document("bib.xml")//book[publisher = $p] WHERE count($b) > 100 RETURN $p
- FOR generates an ordered list of bindings of publisher names to $p
- LET associates to each binding a further binding of the list of book elements with that publisher to $b
- at this stage, we have an ordered list of tuples of bindings: ($p,$b)
- WHERE filters that list to retain only the desired tuples
- RETURN constructs for each tuple a resulting value

Queries Supported by XQuery

- Location/position (“chapter no.3”)
- Simple attribute/value
  - /play/title contains “hamlet”
- Path queries
  - title contains “hamlet”
  - /play//title contains “hamlet”
- Complex graphs
  - Employees with two managers
- Subsumes: hyperlinks
- What about relevance ranking?

How XQuery makes ranking difficult

- All documents in set A must be ranked above all documents in set B.
- Fragments must be ordered in depth-first, left-to-right order.

XQuery: Order By Clause

```
for $d in document("depts.xml")//deptno
let $e := document("emps.xml")//emp[deptno
   = $d]
where count($e) >= 10
order by avg($e/salary) descending
return <big-dept> { $d,
   <headcount>{count($e)}</headcount>,
   <avgsal>{avg($e/salary)}</avgsal> } </big-
   dept>
```

XQuery Order By Clause

- Order by clause only allows ordering by “overt” criterion
  - Say by an attribute value
- Relevance ranking
  - Is often proprietary
  - Can’t be expressed easily as function of set to be ranked
  - Is better abstracted out of query formulation (cf. www)

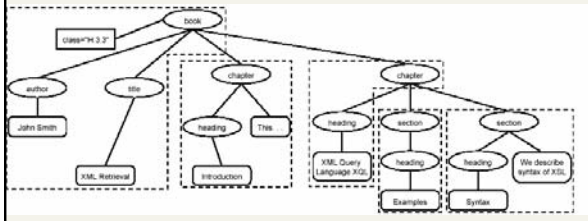
XIRQL

- University of Dortmund
  - Goal: open source XML search engine
- Motivation
  - “Returnable” fragments are special
    - E.g., don’t return a <bold> some text </bold> fragment
  - Structured Document Retrieval Principle
  - Empower users who don’t know the schema
    - Enable search for any person no matter how schema encodes the data
    - Don’t worry about attribute/element

Atomic Units

- Specified in schema
- Only atomic units can be returned as result of search (unless unit specified)
- Tf.idf weighting is applied to atomic units
- Probabilistic combination of “evidence” from atomic units

XIRQL Indexing



Structured Document Retrieval Principle

- A system should always retrieve the most specific part of a document answering a query.
- Example query: xql
- Document:

  <chapter> 0.3 XQL

  <section> 0.5 example </section>

  <section> 0.8 XQL 0.7 syntax </section>

  </chapter>
- Return section, not chapter

Augmentation weights

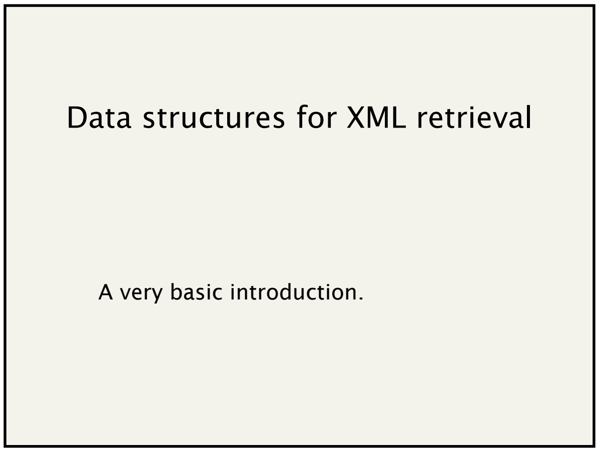
- Ensure that Structured Document Retrieval Principle is respected.
- Assume different query conditions are disjoint events -> independence.
- P(chapter,XQL)=P(XQL|chapter)+P(section|chapter)*P(XQL|section) – P(XQL|chapter)*P(section|chapter)*P(XQL|section) = 0.3+0.6*0.8-0.3*0.6*0.8 = 0.636
- Section ranked ahead of chapter

Datatypes

- Example: person_name
- Assign all elements and attributes with person semantics to this datatype
- Allow user to search for “person” without specifying path

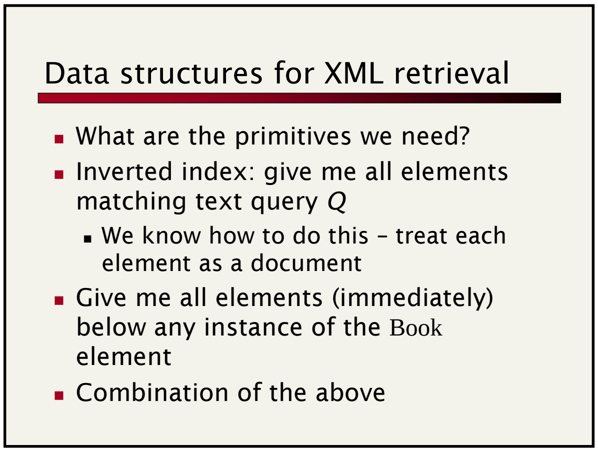
XIRQL: Summary

- Relevance ranking
- Fragment/context selection
- Datatypes (person_name)
- Semantic relativism
  - Attribute/element



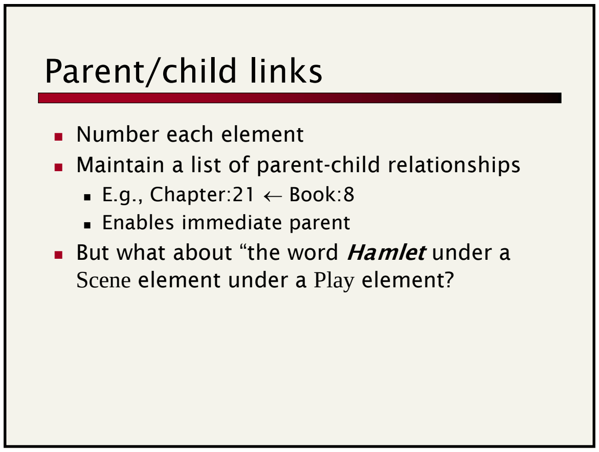
Data structures for XML retrieval

A very basic introduction.



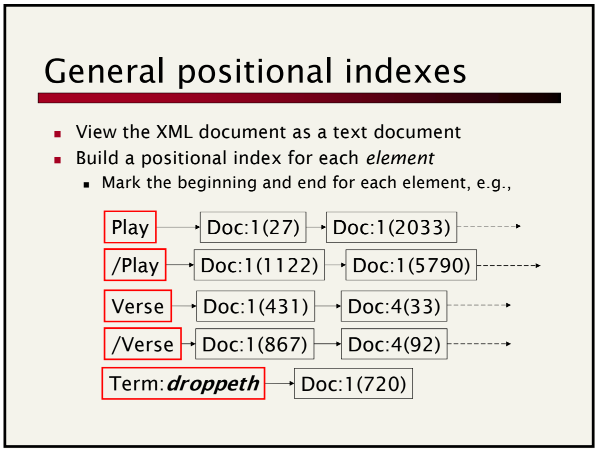
Data structures for XML retrieval

- What are the primitives we need?
- Inverted index: give me all elements matching text query *Q*
  - We know how to do this – treat each element as a document
- Give me all elements (immediately) below any instance of the Book element
- Combination of the above



Parent/child links

- Number each element
- Maintain a list of parent-child relationships
  - E.g., Chapter:21 ← Book:8
  - Enables immediate parent
- But what about "the word ***Hamlet*** under a Scene element under a Play element?



General positional indexes

- View the XML document as a text document
- Build a positional index for each *element*
  - Mark the beginning and end for each element, e.g.,

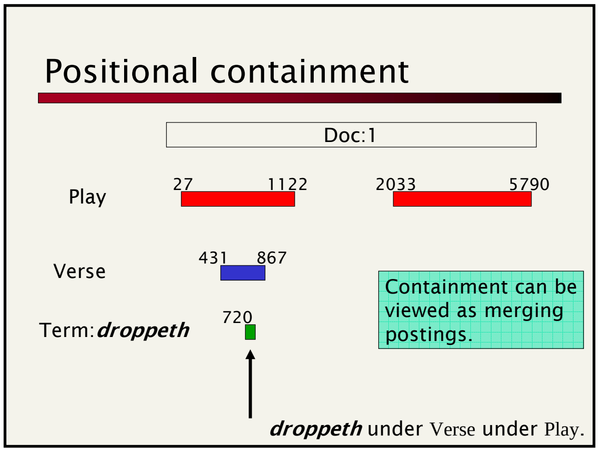
Play → Doc:1(27) → Doc:1(2033) →

/Play → Doc:1(1122) → Doc:1(5790) →

Verse → Doc:1(431) → Doc:4(33) →

/Verse → Doc:1(867) → Doc:4(92) →

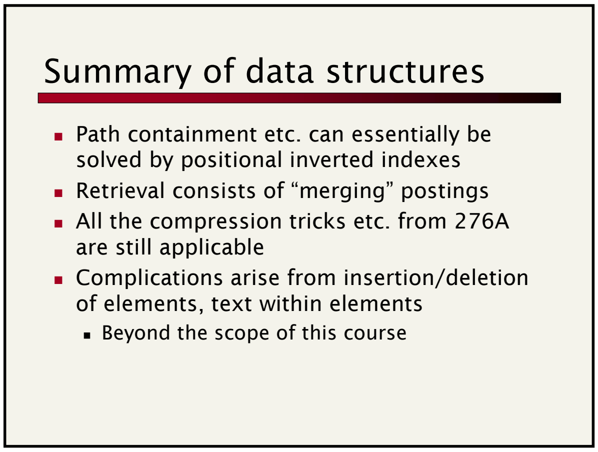
Term:***droppeth*** → Doc:1(720)



Positional containment

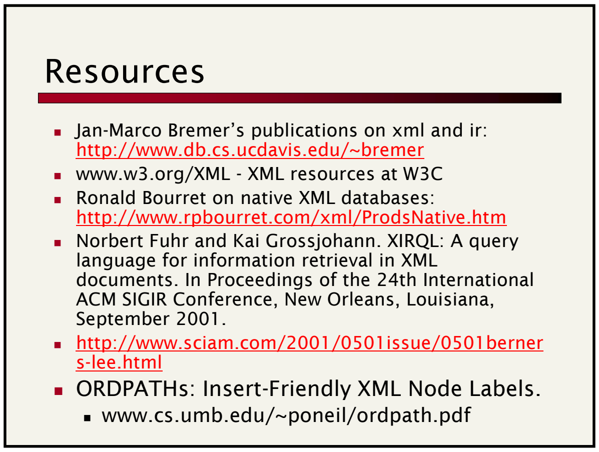
Containment can be viewed as merging postings.

droppeth under Verse under Play.



Summary of data structures

- Path containment etc. can essentially be solved by positional inverted indexes
- Retrieval consists of "merging" postings
- All the compression tricks etc. from 276A are still applicable
- Complications arise from insertion/deletion of elements, text within elements
  - Beyond the scope of this course



Resources

- Jan-Marco Bremer’s publications on xml and ir: http://www.db.cs.ucdavis.edu/~bremer
- www.w3.org/XML - XML resources at W3C
- Ronald Bourret on native XML databases: http://www.rpbourret.com/xml/ProdsNative.htm
- Norbert Fuhr and Kai Grossjohann. XIRQL: A query language for information retrieval in XML documents. In Proceedings of the 24th International ACM SIGIR Conference, New Orleans, Louisiana, September 2001.
- http://www.sciam.com/2001/0501issue/0501berners-lee.html
- ORDPATHs: Insert-Friendly XML Node Labels.
  - www.cs.umb.edu/~poneil/ordpath.pdf