Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
Άνοιξη 2007

ΗΥ463 - Συστήματα Ανάκτησης Πληροφοριών
Information Retrieval (IR) Systems

Ανάκτηση Πληροφοριών από Δομημένα Έγγραφα
(Information Retrieval in Structured Documents)

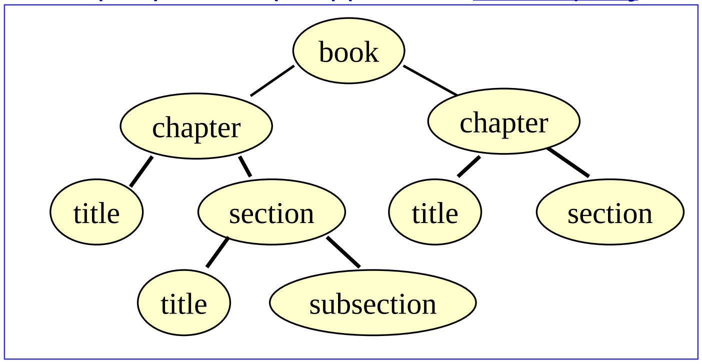
Γιάννης Τζίτζικας
Διάλεξη : 20 -21
Ημερομηνία : 6/8-6-2007

Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery
- **Επερωτήσεις Ανάκτησης XML**
  - XIRQL, XRANK, TeXQuery
- **Ευρετηρίαση** XML εγγράφων

Δομημένα Έγγραφα

- Εδώ τα έγγραφα έχουν **δομή** που μπορεί να αξιοποιηθεί κατά την ανάκτηση
- Η δομή μπορεί να είναι:
  - Ένα προκαθορισμένο σύνολο **πεδίων**
    - title, author, abstract, etc.
  - Δομή **υπερκειμένου** (hypertext)
  - Μια **ιεραρχική δομή**
    - Book, Chapter, Section, etc.
    - μπορεί να περιλαμβάνει και συνδέσμους



Πληροφοριακές ανάγκες και δομή εγγράφων

Παράδειγμα πληροφοριακής ανάγκης:

Retrieve all documents which contain a page in which the string "atomic holocaust" appears in italic in the text surrounding a Figure whose label contains the word *earth*

Η αντίστοιχη επερώτηση μπορεί να μοιάζει με:

same-page(near("atomic holocaust", Figure(label("earth"))))

μια απλή επερώτηση "atomic holocaust earth" θα επέστρεφε μια απάντηση με πολύ μικρή ακρίβεια

CAS queries (Content And Structure Queries)

Μοντέλα Ανάκτησης Δομημένων Εγγράφων
(Structured Text Retrieval Models)

- Κίνητρο
  - Η κλασσική (keyword-based) προσέγγιση ΑΠ, θεωρεί ότι τα έγγραφα είναι **επίπεδα**
- Αδυναμίες
  - σχετικά την <u>επιλογή</u> των εγγράφων της απάντησης
    - ο χρήστης δεν μπορεί να εκφράσει δομικά κριτήρια
  - σχετικά με την <u>βαθμολόγηση</u> των εγγράφων
    - η εμφάνιση μιας λέξης στον τίτλο είναι το ίδιο σημαντική με την εμφάνιση της σε μία υποσημείωση του κειμένου
      - Η δομή του κειμένου αποτελεί πρόσθετη πληροφορία που θα μπορούσε να ληφθεί υπόψη.
        - » Π.χ. οι λέξεις που εμφανίζονται στον τίτλο ή στις κεφαλίδες θα μπορούσαν να λάβουν υψηλότερο βάρος.

Τι είναι η XML;
(eXtensible Markup Language)

- XML was created in 1998 (it is a simplified version of SGML (1986))
- 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

XML: Η βασική δομή

An XML document is an **ordered, labeled tree**

character data leaf nodes contain the actual data (text strings)

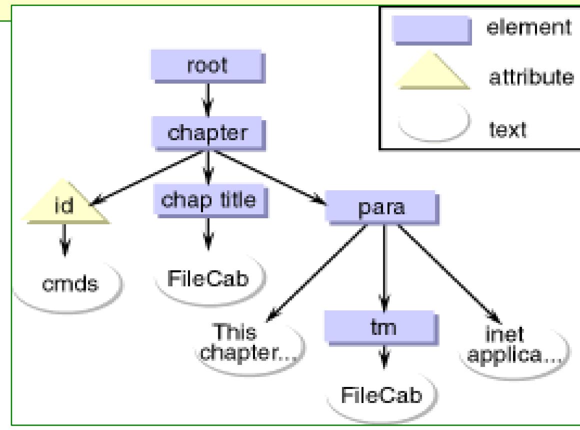
element nodes, are each labeled with

- a **name** (often called the element *type*), and
- a set of **attributes**, each consisting of a **name** and a **value**

and can have child nodes

Παράδειγμα:

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



XML vs HTML

- HTML was created in 1992, XML was created in 1998
- 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: Σχεδιαστικοί στόχοι

- **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 the classical relational databases
- XML is more structured than simple IR

XML Schemas

- Schema = syntax definition of XML language
- Schema language = formal language for expressing XML schemas
- Examples
  - DTD
  - XML Schema (W3C)

Example: DTD of DBLP

```
<!ELEMENT dblp (article|inproceedings|proceedings|book|incollection|phdthesis|mastersthesis|www)*>
<!ENTITY % field
    "author|editor|title|booktitle|pages|year|address|journal|volume|number|month|url|ee|cdrom|cite|publisher|note|crossref|isbn|series|school|chapter">
<!ELEMENT article     (%field;)*>
<!ATTLIST article
                 key CDATA #REQUIRED
                 reviewid CDATA #IMPLIED
                 rating CDATA #IMPLIED
                 mdate CDATA #IMPLIED
>
<!ELEMENT inproceedings (%field;)*>
<!ATTLIST inproceedings key CDATA #REQUIRED
                 mdate CDATA #IMPLIED
>
<!ELEMENT proceedings   (%field;)*>
<!ATTLIST proceedings   key CDATA #REQUIRED
                 mdate CDATA #IMPLIED
>
<!ELEMENT book       (%field;)*>
<!ATTLIST book       key CDATA #REQUIRED
                 mdate CDATA #IMPLIED
>
```


Example: DTD of DBLP

```
<!ELEMENT author    (#PCDATA)>
<!ELEMENT address   (#PCDATA)>
<!ELEMENT booktitle (#PCDATA)>
<!ELEMENT pages     (#PCDATA)>
<!ELEMENT year      (#PCDATA)>
<!ELEMENT journal   (#PCDATA)>
<!ELEMENT volume    (#PCDATA)>
<!ELEMENT number    (#PCDATA)>
<!ELEMENT month     (#PCDATA)>
<!ELEMENT url       (#PCDATA)>
<!ELEMENT ee        (#PCDATA)>
<!ELEMENT cdrom     (#PCDATA)>
<!ELEMENT cite      (#PCDATA)>
<!ELEMENT school    (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ATTLIST publisher
            href CDATA #IMPLIED
>
<!ELEMENT note      (#PCDATA)>
<!ATTLIST cite
            label CDATA #IMPLIED
>
```

Εφαρμογές της XML

- XHTML
- CML – chemical markup language
- WML – wireless markup language
- ThML – theological markup language
- IEEE INEX data collection (scientific articles)
- Shakespeare's plays in XML
  - http://www.oasis-open.org/cover/bosakShakespeare200.html
- DBLP in XML
  - http://dblp.uni-trier.de/xml/
- SIGMOD Record in XML
  - http://www.acm.org/sigmod/record/xml/
- United States Library of Congress in XML
  - http://lcweb.loc.gov/crsinfo/xml/
- ...

XML-IR vs. traditional IR

- Query format
  - The queries in traditional IR are formulated in **natural language**
  - The queries in XML-IR can **take advantage of the structure**
- Unit of retrieval
  - In traditional IR, a **document** is the natural unit of retrieval
  - In XLM-IR **any element** of any XML document can be considered as a retrievable unit
- Indexing
  - In traditional IR a document is a **bag of words**
  - In XLM-IR a document is a **tree of bag of words**

Ανάκτηση Πληροφοριών από Δομημένα Έγγραφα
Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery, [IBM Haifa]
- **Επερωτήσεις Ανάκτησης XML**
  - XIRQL, XRANK, TeXQuery
- **Ευρετηρίαση** XML εγγράφων

XML Retrieval: **Evaluation**

INEX (Initiative for the Evaluation of XML retrieval)
  - Similar to TREC for text retrieval, started in 2002 (http://qmir.dcs.qmw.ac.uk/inex/)
- Document collections
  - 12.000 scientific articles from IEEE Computer Society 1995 – 2002. About 500M, each article consists of 1500 XML nodes on average. Typical structure: title, author, abstract, sections, sub-sections, sub-subsections, paragraphs, tables, figures, lists, citations, bibliography, author information.
- **INEX Topics:**
  - **Content-Only** topics
    - ignore the structure of the documents (i.e. they are traditional IR topics), e.g.:
      - Find information on the use of formal logics to model or reason about UML diagrams
  - **Content-And-Structure (CAS)** topics
    - are aware of the documents' structure, e.g.:
      - /article[./fm//yr='2000' AND about(.,'intelligent transportation systems')]//sec[about(.,'automation vehicle')]
    - CAS topics = mixture of structural expressions and natural language queries

XML Retrieval Evaluation: **INEX Tasks**

INEX 2003 had 3 tasks

- **Content-Only task (CO)**
- **Strict Content-And-Structure task (SCAS)**
  - The elements returned should answer the content and satisfy the structural constraints
- **Vague Content-And-Structure task (VCAS)**
  - The elements returned should answer the content but not necessarily satisfy the structural constraints

Αξιολόγηση Κλασσικής ΑΠ vs. Αξιολόγηση Ανάκτησης XML

Classical IR Retrieval Evaluation

- A document = *a well-distinguishable unit* representing a retrievable entity.
- Documents are considered as units of (approximately) equal size.
- Given a ranked output list, users look at one document after another and then stop at an arbitrary point. Thus, non-linear forms of output are not considered.

(most of the above assumptions do not hold for XML retrieval)

XML Retrieval Evaluation

- XML documents consist of nested structures where document components of varying granularity may be retrieved, which cannot always be regarded as separate units.
- The size of the retrieved components vary from elements such as author names or paragraphs to complete documents or books.
- When multiple components from the same document are retrieved, a linear ordering of the result items may confuse the user as they may be interspersed with components from other documents. That's why some systems cluster components from the same document together, resulting in non-linear outputs.

XML Retrieval Effectiveness

- **CO queries**
  - ***effectiveness*** = ability to retrieve the **most specific** relevant document components, which are **exhaustive** to the topic of request.
- **CAS queries**
  - ***effectiveness*** = ability to retrieve the **most specific** relevant document components, which are **exhaustive** to the topic of request and **also match the structural constraints** specified in the query.

Why Relevance is not enough ?

- The basic threshold for relevance was defined as a piece of text that mentions the topic of request.
- => container components of relevant document components in a nested XML structure, albeit too large components, are also regarded as relevant.
- => Relevance as a single criterion is not sufficient for the evaluation of content-oriented XML retrieval.
- Hence, we need another measure based on the size of the components

XML Retrieval Evaluation Metrics: Exhaustivity & Specificity

- **[rel] Topical relevance (or *exhaustivity*)**
  - the extent to which the information contained in a document component satisfies the user's information need, e.g. measures the **exhaustivity of the topic within a component**.
- **[cov] Component coverage (or *specificity*)**
  - It reflects the extent to which a document component is focused on the information need, e.g. measures the **specificity of a component** with regards to the topic (i.e. whether it discusses no other irrelevant topics).

XML Retrieval Evaluation Metrics: Scale of **Exhaustivity**

- **[rel] Topical relevance (or exhaustivity)**
  - **Scale = {0,1,2,3}**
  - 0: Irrelevant
    - The document component does not contain any information about the topic of request.
  - 1: Marginally relevant
    - The document component mentions the topic of request, but only in passing.
  - 2: Fairly relevant
    - The document component contains more information than the topic description, but this information is not exhaustive. In the case of multi-faceted topics, only some of the sub-themes or viewpoints are discussed.
  - 3: Highly relevant
    - The document component discusses the topic of request exhaustively. In the case of multi-faceted topics, all or most sub-themes or viewpoints are discussed

XML Retrieval Evaluation Metrics: Scale of **Specificity**

- **Component coverage (or specificity)**
  - **Scale = {N,L,S,E}**
  - N: No coverage
    - The topic or an aspect of the topic is not a theme of the document component.
  - L: Too large
    - The topic or an aspect of the topic is only a minor theme of the document component.
  - S: Too small
    - The topic or an aspect of the topic is the main or only theme of the document component, but the component is too small to act as a meaningful unit of information when retrieved by itself.
  - E: Exact coverage
    - The topic or an aspect of the topic is the main or only theme of the document component, and the component acts as a meaningful unit of information when retrieved by itself.

XML Retrieval Evaluation Metrics: Single (Combined) metrics

- Recall that
  - range(Relevance) = {0, 1, 2, 3}
  - range(Coverage) = {N, S, L,E}.
- Examples of single metrics:

•Strict

$$f(rel,cov) = \begin{cases} 1 & \text{if } rel = 3 \text{ and } cov = E \\ 0 & otherwise \end{cases}$$

•Generalized

$$f(rel,cov) = \begin{cases} 1.00 & \text{if } (rel,cov) = 3E \\ 0.75 & \text{if } (rel,cov) \in \{2E, 3L\} \\ 0.50 & \text{if } (rel,cov) \in \{1E, 2L, 2S\} \\ 0.25 & \text{if } (rel,cov) \in \{1S, 1L\} \\ 0 & \text{if } (rel,cov) = 0N \end{cases}$$

Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery, [IBM Haifa]
- **Επερωτήσεις Ανάκτησης XML**
  - XIRQL, XRANK, TeXQuery
- **Ευρετηρίαση** XML εγγράφων

Επερωτήσεις Δομής XML
XPath

XPath is a declarative language for navigating trees and selecting nodes

- Declarative language for
  - **Addressing** (used in XLink/XPointer and in XSLT)
  - **Pattern matching** (used in XSLT and in XQuery)
- Example:
  - child::section[position()<6] / descendant::cite / attribute::href

XPath: Location Paths (and Steps)

- **Location path:** a sequence of **location steps** separated by /
  - locationPath : locationStep/locationStep/locationStep/locationStep
- A single **location step** has the form:

axis :: *node-test* [*predicate*]

XPath: Location Paths (and Steps)

A single **location step** has the form:

axis :: *node-test* [*predicate*]

The **axis** selects a set of *candidate nodes* (e.g. the child nodes of the context node). Axes in Xpath:

- ancestor
- ancestor-or-self
- attribute
- child
- descendent
- descendent-or-self
- following
- following-sibling, namespace
- parent
- preceding
- preceding-sibling
- self

The **node-test** performs an *initial filtration* of the candidates based on their

- ***types*** (chardata node, processing instruction,etc), or
- ***names*** (e.g. element name).

The **predicates** (zero or more) cause a further, *more complex, filtration*.

Example location steps:

- child::section[position()<6]

Xpath: Examples of Location Paths

- /bib/book/author/name
- /bib/book//name/*/zip
- /bib/book[author/name="**Manousos**"]
- /bib/book/[year="**2002**" and author[name="**Manousos**" and country="**ΓR**"]]

XQuery

- XQuery is a functional language, resembling SQL
- XQuery contains XPath as a sublanguage

Example

```
FOR $s in document(bla.xml)/article//sec
WHERE contains ($s, "vehicle")
RETURN $s
```

XQuery Expressions:

- path expressions
- element constructors
- list expressions
- conditional expressions
- quantified expressions
- datatype expressions

FLWR Expressions

```
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

XQuery Example

"For each ingredient, the recipes that it is used in":

```
FOR $i IN distinct(document("recipes.xml")//ingredient/@name)
RETURN <ingredient name={$i}>
            { FOR $r IN document("recipes.xml")//recipe
              WHERE $r//ingredient[@name=$i]
              RETURN $r/title
            }
       </ingredient>
```

```
<?xml version="1.0"?>
<xql:result xmlns:xql="http://metalab.unc.edu/xql/">
  <ingredient name="beef cube steak">
    <title>Beef Parmesan with Garlic Angel Hair Pasta</title>
  </ingredient>
  <ingredient name="onion, sliced into thin rings">
    <title>Beef Parmesan with Garlic Angel Hair Pasta</title>
  </ingredient>
  ...
</xql:result>
```

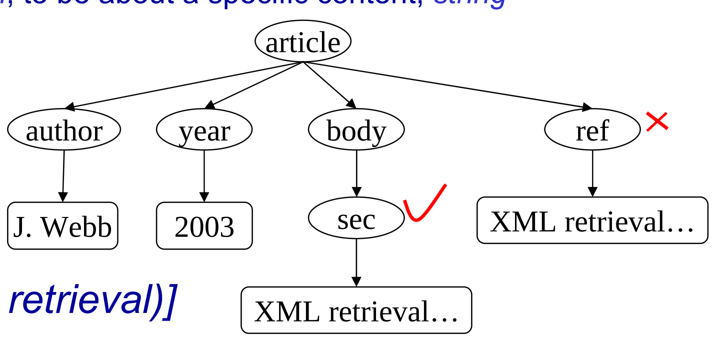
Møller and Schwartzbach

Ανάκτηση Πληροφοριών από Δομημένα Έγγραφα
Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery, [IBM Haifa]
- **Επερωτήσεις Ανάκτησης XML**
  - CO: XIRQL, XRANK, CAS: TeXQuery
- **Ευρετηρίαση** XML εγγράφων

Επερωτήσεις **Ανάκτησης** XML
(CAS queries)

- Μια επερώτηση
  - Προσδιορίζει το αναζητούμενο **περιεχόμενο** (content)
  - Περιέχει συγκεκριμένες **αναφορές στη δομή XML**
  - Καθορίζει τον τύπο του στοιχείου (element) που πρέπει να **επιστραφεί**
- Συγκεκριμένα
  - **Δομή**: **XPath expressions**
  - **Περιεχόμενο**:***about (path, string)*** functions
    - Specify a certain context, *path*, to be about a specific content, *string*
    - Basis for result ranking



*/article/body/sec[**about**(., XML retrieval)]*

Αποτίμηση **CO** (Content-Only) Επερωτήσεων

- Μελέτη Περιπτώσεων
  - **XIRQL**
  - **XRANK**

Case Study: **XIRQL**

Goal: open source XML search engine (University of Dormund)

Keypoints:

- Atomic Units: "Returnable" fragments
- *Structured Document Retrieval Principle*
- Empower users who don't know the schema
- Extends XQL (Xpath) by
  - probabilistic retrieval with weighted document indexing
  - relevance-oriented search (irrespective of structure)
  - (Extensible) data types with vague predicates
  - Structural relavitism (an item could be modeled as an element or an attribute)

XIRQL

- Ερωτήματα
  - Πως πρέπει να καθορίζουμε τα βάρη των όρων σε δομημένα έγγραφα;
  - Πως να αποτιμούμε επερωτήσεις που δεν κάνουν αναφορά στη δομή;
- Προσέγγιση
  - Γενίκευση του TF-IDF για δομημένα έγγραφα
- Τρόπος: Διάκριση των **ατομικών μονάδων** (atomic units) κάθε εγγράφου
- Ατομικές μονάδες (atomic units)
  - Atomic unit ≈ επίπεδο έγγραφο (flat doc) όπως στο κλασσικό IR
  - Αν μια επερώτηση δεν περιορίζει τον τύπο της απάντησης τότε η απάντηση της συγκροτείται από ατομικές μονάδες (τα υπόλοιπα στοιχεία των XML εγγράφων δεν θεωρούνται κατάλληλα για τις απαντήσεις).
    - E.g., don’t return a <bold> some text </bold> fragment

XIRQL: Ατομικές Μονάδες

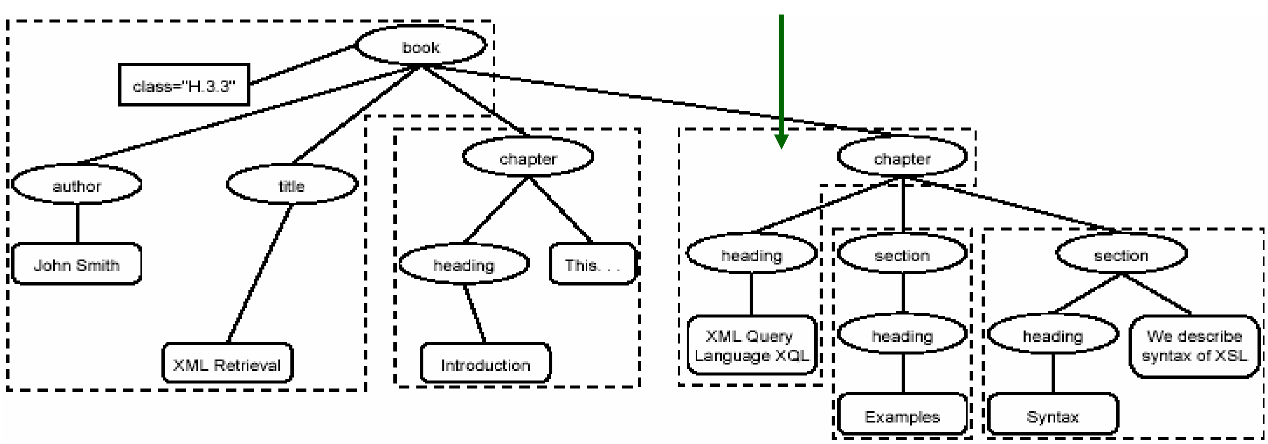
- Ποιες είναι οι **ατομικές μονάδες** ενός XML εγγράφου;
  - Παρατήρηση: Κείμενο υπάρχει μόνο στα *φύλλα* του XML δένδρου
  - Αν κάθε φύλλο = ατομική μονάδα, έχομε υπερβολική λεπτομέρεια:
    - Every item of an enumeration list would be an atomic unit
    - Every text would be an atomic unit
- Οι ατομικές μονάδες ορίζονται από τα **Index Objects**
  - Μόνο κόμβοι **συγκεκριμένων τύπων** μπορεί να είναι ρίζες των index objects
  - Ορίζουμε τους τύπους αυτούς στο DTD

XIRQL: Βάρυνση Όρων στις Ατομικές Μονάδες

- Απαιτήσεις
  - Για τη βάρυνση όρων θα προτιμούσαμε **disjoint index objects** (e.g. IDF)
  - Από την άλλη, θέλουμε να υποστηρίζουμε **φωλιασμένα index objects** και να έχουμε τη δυνατότητα **απαντήσεων μεταβλητής λεπτομέρειας**
- Τρόπος
  - Ευρετηριάζουμε πρώτα το κείμενο των χαμηλότερων index objects
  - Στα υψηλότερα index objects (που περιλαμβάνουν άλλα), ευρετηριάζουμε μόνο το κείμενο που δεν περιέχεται στα φωλιασμένα index objects
  - Έτσι επιτυγχάνουμε disjoint index objects

XIRQL: Βάρυνση Όρων στις Ατομικές Μονάδες

- Παράδειγμα:
  - Έστω ότι ρίζες των index objects είναι οι τύποι **book, chapter, section**



XIRQL: Structured Document Retrieval Principle

A system should always retrieve the <u>most specific</u> 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 <u>section</u>, not chapter

XIRQL: Augmentation weights

- Προκειμένου να ικανοποιηθεί η «Structured Document Retrieval Principle».
- Τα βάρη των όρων μειώνονται καθώς διαδίδονται στους πάνω όρους
  - Augmentation weights

XIRQL: Keypoints

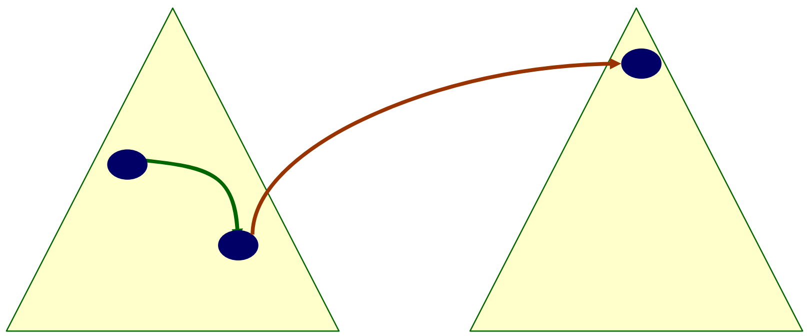
- Atomic Units
  - Specified in schema
  - Only atomic units can be returned as result of search (unless unit specified)
  - tf-idf weighting is applied to atomic units
  - Probabilistic combination of "evidence" from atomic units
- Data types
  - Example: person_name
  - Assign all elements and attributes with person semantics to this datatype
  - Allow user to search for "person" without specifying path

Case Study: **XRANK**

- Objective:
  - Support **keyword queries** (not structured queries) over a set of **hyperlinked XML documents.**
- Key Question:
  - How to rank the elements of XML ?
- Key characteristics of XRANK:
  - Generalizes **PageRank** for XML documents
    - If an XML has depth 2 then the system behaves like Google
  - Adopts 2 metrics for proximity among keywords
    - **keyword distance** (distance e.g. in words between these)
    - **ancestor distance** (distance of the most proximal common ancestor)

XRANK: Interlinked XML documents

- Links in XML documents
  - **Internal** (IDREFs)
    - e.g. <cite ref=«2»> Paper 2 </cite>
  - **External** (XLinks)
    - e.g. <cite xlink=«../paper/Paper3»> Paper 3 </cite>



XRANK: Queries and their answers

Query Q = any conjunction of keywords.

ans(Q) = { XML elements that contain all keywords of Q
if we first exclude the keywords of the subelements
that also contain all keywords of Q}

- Example: Let Q=«Crete»
  - If «Crete» appears once in one <*subsection*> of a chapter, then this <*chapter*> **is not** returned
  - If «Crete» appears in the introduction of the chapter and once in one <*subsection*> of the chapter, then <*chapter*> **is also** returned
    (they are independent occurences)

- Motivation:
  - (recall the «structured document retrieval principle» of XIRQL)

XRANK: How to rank the elements of ans(Q) ?

- Requirements
  - **Result specificity**
    - (the *more specific* an element is, the higher its rank should be)
  - **Keyword Proximity**
    - (the *proximity* of the query keywords should be taken into account)
  - **Link Awareness**
    - (ala Google)
- We want to define **R(e,Q)** for each **e** in **ans(Q).**

XRANK: Ranking Query Results
R(e,Q) =?

Q=k1 … kn

$$R(e,Q) = \left(\sum_{k_i \in Q} R(e,k_i) \right) * p(e,k_1,k_2,...,k_n)$$

Sum of the ranks wrt each keyword ki

Keyword proximity

$$R(e,k_i) = ElemRank(\text{subelem of e that contains } k_i) * a^{depth(\text{subelem of e that contains } k_i)-1}$$

$$0 < a < 1$$

PageRankStyle

So that broad elements receive low score

XRANK: Ranking Query Results
ElemRank

- ElemRank corresponds to the stationary probability of a random surfer who
  - with probability p1 follows a **hyperlink** from e
  - with probability p2 goes to one **subelement** of e
  - with probability p3 goes to the **parent** of e
  - with probability 1-p1-p2-p3 jumps to **random** element of a random document

Concerning evaluation, we have not experimental results

Case Study: **TeXQuery**

A full-text search extension to Xquery

- Precursor of the full-text language extensions to XPath 2.0 and XQuery 1.0 (currently under development by W3C)

Supports:

- Full-text search primitives
  - Boolean connectives
  - Phrase matching
  - Proximity distance
  - Stemming
  - Thesauri
- Scoring constructs

TeXQuery: Motivation

- With XQuery we can express only rudimentary full-text search
  - Example
    - **FOR $s in document(bla.xml)/article//sec**
    - **WHERE contains ($sec, "vehicle")**
    - **RETURN $s**
- We would like to be able to satisfy information needs like:
  - *Find the titles and contents of books whose content contains the phrases "usability", "Web site" and is in that order, in the same paragraph, using stemming if necessary to match the tokens.*

Stemming **Phrase matching** **Order specification** **Paragraph scope**

- Moreover, "contains" of XQuery does not rank or score the results

TeXQuery Primitives

TeXQuery supports two new kinds of expressions:

- **FTContainsExpr::= ContextExpr "ftcontains" FTSelection**
  - returns true if at least one node in ContextExpr satisfies FTSelection.
- **FTScoreExpr::= ContextExpr "ftscore" FTWeightedSelection**
  - allows supporting different scoring mechanisms
  - provides the necessary language for specifying the weights in query
  - returns a sequence of scores. Provides access to fine grained ranking (e.g., threshold and top-k.)

TeXQuery: Full-Text Search Primitives
FTSelections

FTSelections: A set of powerful FT search primitives which are fully composable (hence, they allow expressing complex searches)

FTSelection ::=	FT**String**Selection |	"software"
	FT**And**Connective |	"software" && "user"
	FT**Or**Connective |	"software" || "user"
	FT**Negation** |	!"hardware"
	FT**MildNegation** |	
	FT**Order**Selection |	
	FT**Scope**Selection |	scope: same sentence
	FT**Distance**Selection |	
	FT**Window**Selection |	window 5
	FT**Times**Selection |	
	FTSelection (FTContextModifier)*	

TeXQuery: **FTContainsExpressions**

- Example
  - //book ftcontains «usability» && «testing» same sentence window 5
- Keypoint: FTContainsExpression can be arbitrarily nested within other XQuery expressions
  - //book[.//section ftcontains «usability» && «testing»]/title
  - returns the titles of those books in which some section contains the tokens «usability» and «testing»

FTSelection ::= FTSelection (FTContextModifier)*

- **FTContextModifier** can modify the operational semantics of FTSelection such as stemming, stopwords, diacritics and case.

- **FTContextModifier**::= FT**Case**CtxMod |
  FT**Diacritics**CtxMod |
  FT**SpecialChar**CtxMod |
  FT**Stem**CtxMod |
  FT**Thesaurus**CtxMod |
  FT**StopWord**CtxSpec |
  FT**Language**CtxMod |
  FT**RegEx**CtxMod |
  FT**Ignore**CtxMod

  - with stems
  - without stopwords
  - /* ignore specified XML subtrees */

Scoring and Ranking: Remarks

FTScoreExpr::= ContextExpr "ftscore" FTWeightedSelection

- The ranking expression can be *different* from the search expression (recall that this is not possible in XIRQL)
- The syntax allows expressing user weights.
- Scoring function should satisfy:
  - If context node does not satisfy FTSelection used in *ftscore*, score is 0
  - If context node satisfies FTSelection used in *ftscore*, score should be in [0,1]
  - For context nodes that satisfy the FTSelection, a higher score implies a higher relevance to the FTSelection.
- Example
  - //book ftscore 'usability' weight 0.8 && 'testing' weight 0.2
  - The above actually specifies the weights of the query terms
  - The weights of these terms in the XML doc is independent (e.g. could be set as in XIRQL)

Integration with XQuery

- **Simple Example:**

```
for $book in
books/book ftcontains "usability" with stems && "software" && !"Rose“
return   <hit>
           $book
         </hit>
```

- **Top-K Example:**

```
for $hit at $i in
        for $book in books//section ftcontains "usability"
        let $score := $book ftscore "software" weight 0.7
        order by $score descending
        return <hit>$book<score>$score</score></hit>
where $i < 20
return $hit
```

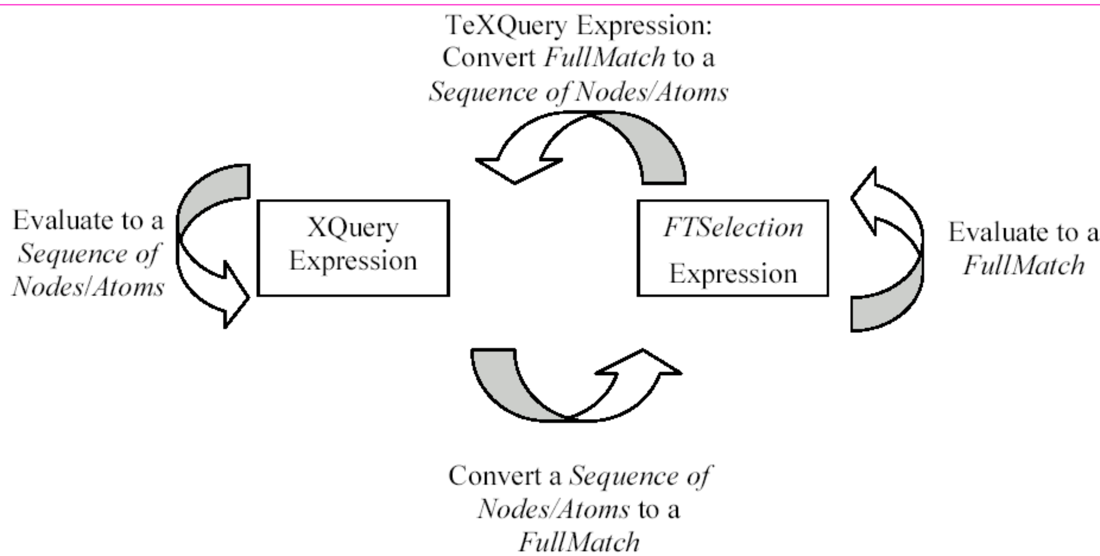
Filter condition is different from the scoring condition

TeXQuery Data Model

- Relies on a formal data model called **FullMatch** that based on the positions of words *within* XML nodes
- FullMatch has a hierarchical structure so it can be resprented in XML.
- This XML-2-XML transformation can be specified in Xquery itself.

- **XQuery** expressions take sequence(s) of nodes as input and evaluate to a sequence of nodes.
- **FTSelection** takes FullMatch(es) as input, and evaluates to a FullMatch in the FTS data model.
  - FullMatch captures linguistic token positions, and other information required for full composability of FTSelections.

Composing XQuery and TeXQuery Expressions



- TeXQuery expressions define a well-formed mapping between the fullmatch data model and XQuery data model.
- Fullmatch is only internal to TeXQuery. TeXQuery expressions still return a sequence of items that are fully composable with XQuery data model.

Ανάκτηση Πληροφοριών από Δομημένα Έγγραφα
Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery
- **Επερωτήσεις Ανάκτησης XML**
  - CO: XIRQL, XRANK, CAS: TeXQuery
- **Ευρετηρίαση** XML εγγράφων

Indexing and Searching XML Documents

Database approach:

(a) Map XML documents to relational databases
- XPath / XQuery queries are mapped to SQL queries

(b) Native XML Databases (Γηγενείς ΒΔ XML)
- **Examples: XIRQL, IBM Haifa system, Toxin**
- Uses XML document as logical unit
- Support of: Elements, Attributes, PCDATA, Document order
- Most native XML databases have taken a DB approach
  - Exact match, Evaluate path expressions, No IR type relevance ranking

IR approach:
- [A] Ignore tags (and do what in classical IR)
- [B] Ignore some tags and device new index structures
  - index terms and structure apart
  - index terms and structure together
    - naïve approach, Dewey Inverted List (DIL) from XRANK

Naïve approach for constructing an inverted index

- Approach: treat each element as a document
- An inverted list contains for each keyword, the list of documents that contain the keyword. For XML documents, it would contain the list of elements.
- This would result in large space overhead; because each inverted list would contain
  - XML elements that directly contain the keyword
  - XML elements that indirectly contain the keyword
- In addition the query results would be spurious
  - All elements are treated as independent documents. Results will not correspond to the desired semantics for XML keyword search (specificity will not be taken into account).

Dewey Decimal Classification (DDC)

- *The Dewey Decimal Classification* (DDC) system, devised by Melvil Dewey in the 1870s and first published in 1876, was published in its 22nd edition (DDC 22) in four volumes in 2003 (also available in a Web version). The DDC is the world's most widely used library classification system. The system is published in a full version as well as an abridged version. Dewey abridged 14th edition was released in 2004 and is designed for small public libraries and individual school libraries.

Dewey has a rich notational structure

The Lancashire cotton industry : a study in economic development

Assigned DDC Code: 338.476772109 4276

Schedules	3	Economics, Education, Society
	33	Economics and Management
	338	Industries, Products
	338.1 – 338.4	Specific kinds of industries
	338.4	Secondary Industries and Services
	338.47	Goods and Services
Built from Schedules	338.471 – 338.479	Subdivisions for Goods and Services
	338.476	Technology
	338.4767	Manufacturing
	338.47677	Textiles
	338.476772	Textiles of Seed hair fibres
	338.4767721	Cotton
Built from Table 1	338.47677210	Facet Indicator for Standard Subdivision
	338.476772109	Historical, geographic, persons treatment
Built from Table 2	338.4767721094	Europe Western Europe
	338.47677210942	England and Wales
	338.476772109427	Northwestern England and Isle of Man
	338.4767721094276	Lancashire

Indexing and Searching XML Documents
Dewey Encoding of XML

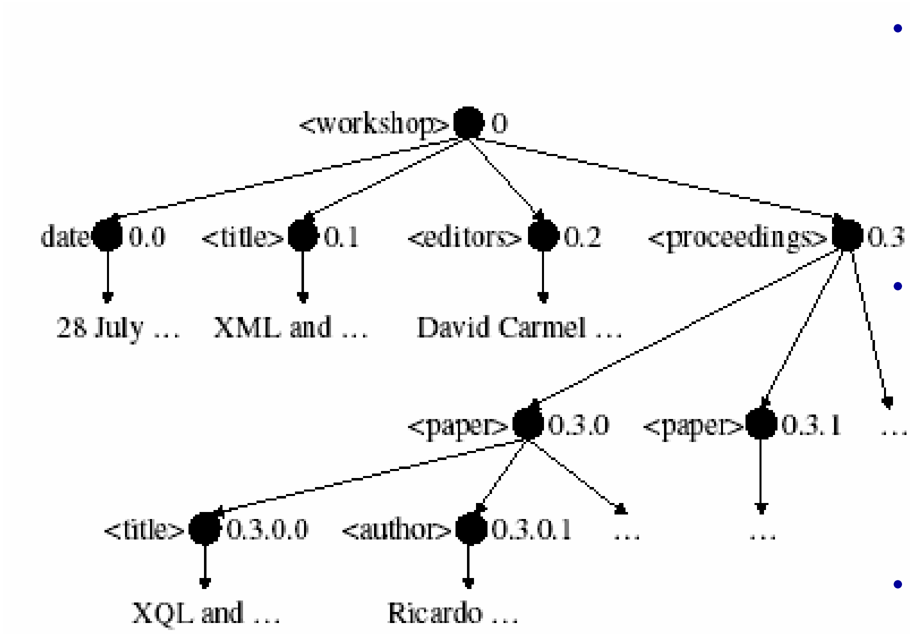
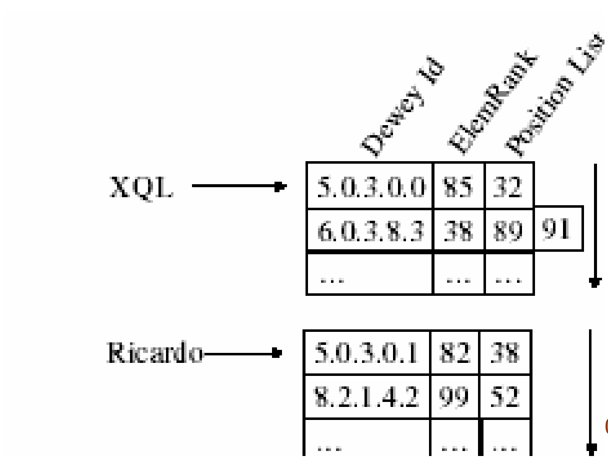


Figure 3: Dewey IDs

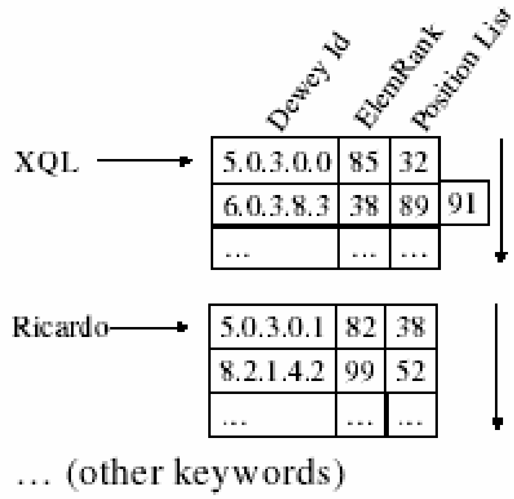
- Each element is assigned a **number** that represents its relative position among its siblings
- The **path** vector from the root to an element uniquely identifies the element and can be used as the element ID.
- The ID of an ancestor is a **prefix** of the ID of a descendant

Indexing and Searching XML Documents
Dewey Inverted List

- The inverted list for a keyword k contains the Dewey IDs of all the XML elements that <u>directly</u> contain the keyword k.
  - If we have multiple XML documents then the first component of each Dewey ID is the document ID.
- Then we put the positions of the occurences



Indexing and Searching XML Documents
Dewey Inverted List



we can infer the subsuming elements (their ID is a prefix).

In this example:

We can infer that the term XML also appears in the elements with code:

5.0.3.0

5.0.3

5.0

5

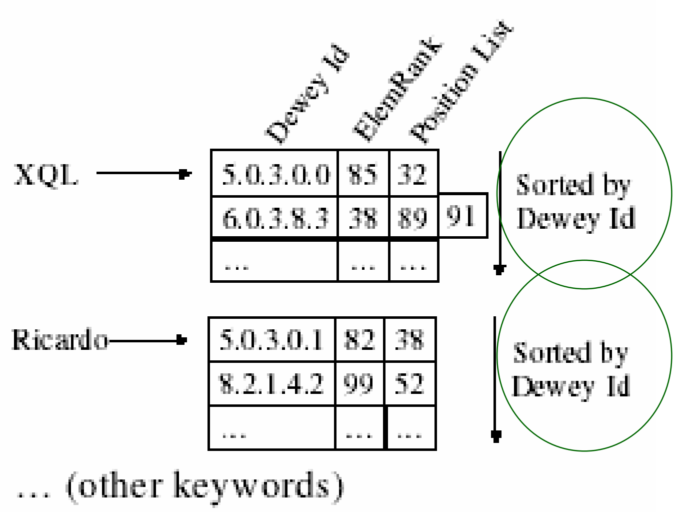
6.0.3.8

6.0.3

…

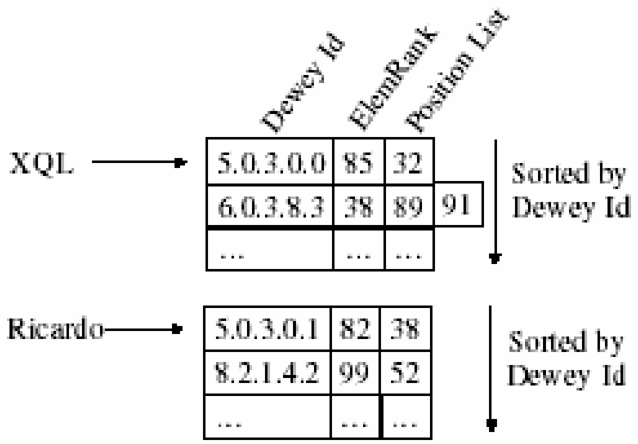
Indexing and Searching XML Documents
Dewey Inverted List

- Entries are sorted by Dewey IDs



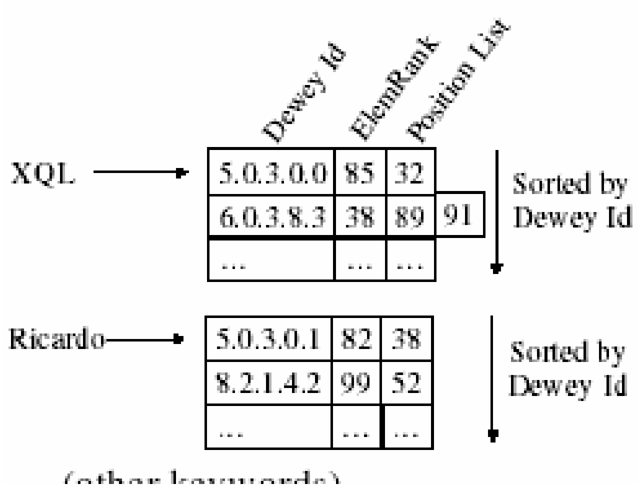
Indexing and Searching XML Documents
Query Processing with Dewey Inverted List

- for each word of the query we find the corresponding entries in the index
- then we need to find their common ancestor
  - this is easy (due to Dewey): longest common prefix of their IDs
  - due to sorting wrt Dewey Codes: they are all clustered together



Indexing and Searching XML Documents
Query Processing with Dewey Inverted List

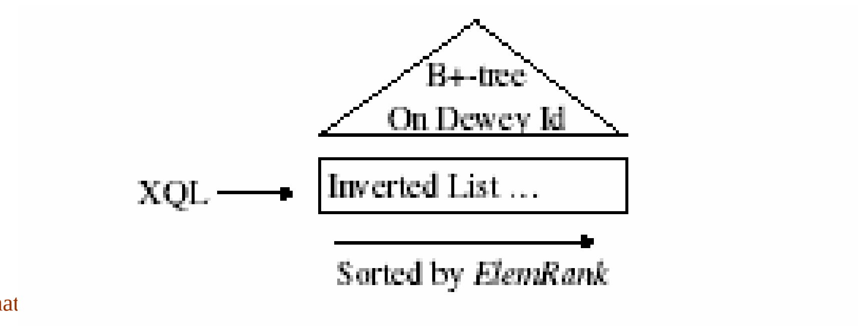
- Single pass algorithm over the query keyword inverted lists.The key idea:
  - Merge the query keyword inverted lists
  - Simultaneously compute the longest common prefix of the Dewey IDs in different lists.



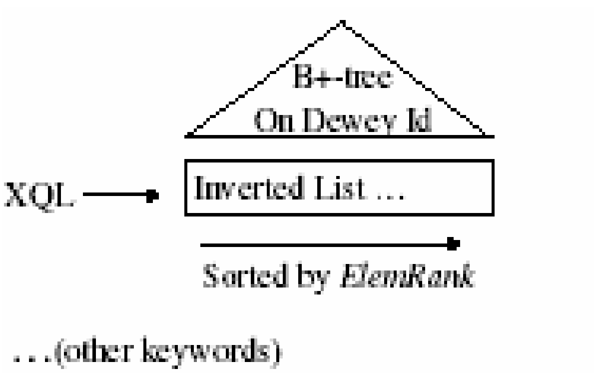
Example:
Q="XQL Ricardo"
List(XQL)=(5.0.3.0.0, ….)
List(Ricardo)=(5.0.3.0.1, …
➔ The element 5.0.3.0 contains both keywords

An alternative approach

- If inverted lists are long (due to common keywords or large document collections) even the cost of a single scan of the inverted list can be expensive, especially if the users want only the top few results.”
- An alternative approach:
  - Order the inverted lists by the ElemRank instead of by the Dewey ID.
  - This means that higher ranked results will appear first in the inverted list.
  - Each inverted list has a B+-tree index of the Dewey ID field.



An alternative approach > Query Processing

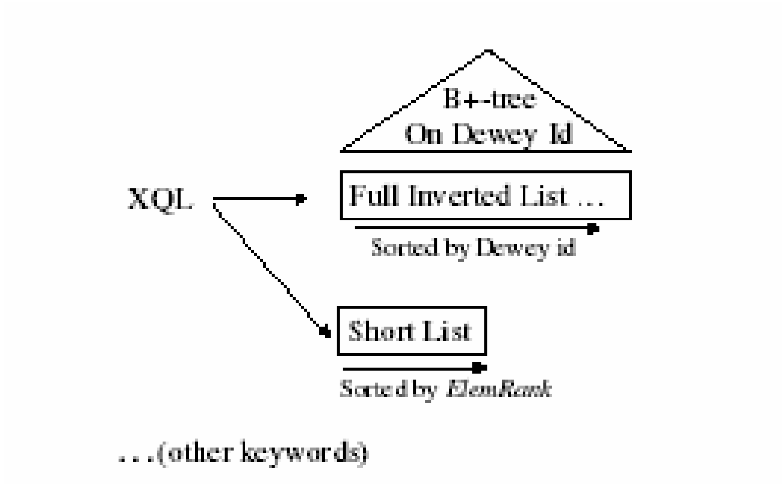


igure 8: Ranked Dewey Inverted List

- Consider an entry retrieved from the inverted list of keyword k_i.
  - The entry contains the Dewey ID d of a top-ranked element that directly contains the query keyword k_i.
  - To determine a query result the longest prefix of d that also contains the other query keywords needs to be determined (this can be computed based on the Btrees of the inverted lists of the other keywords)
- However It may perform worse than the first approach (where elements were ordered by their Dewey code) when there is a query where keywords are not correlated.

An alternative (hybrid) approach

- We combine the benefits of the previous approaches without having to replicate the entire inverted list index.



Ανάκτηση Πληροφοριών από Δομημένα Έγγραφα
Διάρθρωση Διάλεξης

- **Εισαγωγή**
  - Δομημένα Έγγραφα
  - Πληροφοριακές Ανάγκες και Δομή Εγγράφων (**CAS queries**)
  - Εισαγωγή στην XML
    - Ιστορικό, XML vs. HTML, σχεδιαστικοί στόχοι και εφαρμογές
- **Διαφορές μεταξύ Παραδοσιακής Ανάκτησης και Ανάκτησης XML**
- **Αξιολόγηση Ανάκτησης XML**
  - Συλλογές αξιολόγησης: **INEX**
  - Θέματα αξιολόγησης: **CO/CAS**
  - Μέτρα αξιολόγησης: **Exhaustivity, Specificity**
- **Επερωτήσεις Δομής XML**
  - XPath, XQuery
- **Επερωτήσεις Ανάκτησης XML**
  - CO: XIRQL, XRANK, CAS: TeXQuery
- **Ευρετηρίαση** XML εγγράφων