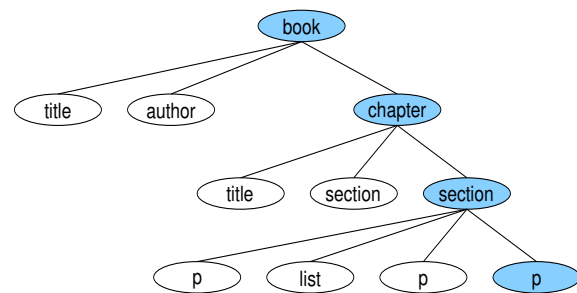


Inverted File Structures for XML Retrieval

XML retrieval: search with respect to content *and* structure of documents

Structural information to be accessed at retrieval time:

Element names	'London' in the <i>title</i> of an image vs. 'London' in the <i>location of creation</i>
Element index	<i>first</i> chapter of a book, <i>first</i> author of a publication
Ancestor / descendant	find a chapter which <i>includes</i> a theorem and the corresponding proof
Preceding / following	document that explains vector space model <i>based on</i> uncertain inference



Concept for encoding this information: **Path** as a sequence of steps

/ book[1,1] / chapter[1,3] / section[2,3] / p[3,4]

Approaches for accessing paths from inverted list postings:

PIL: Include paths in inverted lists

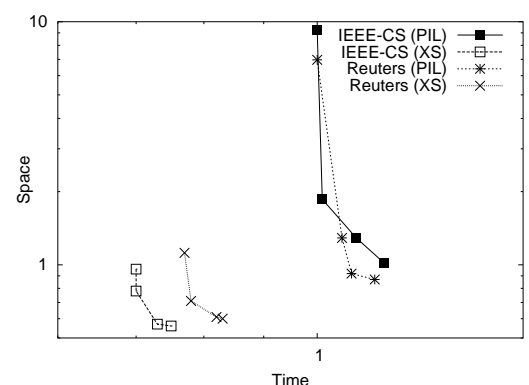
- Postings of inverted lists contain complete path information:
path -> path_length element+ element_index+
sequence_index+
- Example: < 4 <book chapter section p> <1 1 2 3>
<1 3 3 4> >
- indexing: requires vast amounts of disk space
- + retrieval: decode information needed only

XS Tree: encode structure of whole document

- Additional data structure describing structure of a document
- Structure represented as a list of document nodes in pre-order; parent-child relationship via level numbers
- Example: <1 book> <2 title> <2 author> <2 chapter>
<3 title> <3 section> <3 section> <4 p> <4 list>
<4 p> <4 p>
- Postings contain compact *handles* which can be resolved at retrieval time to paths via the respective XS tree
- + indexing: compact and fast indexing
- retrieval: decoding of XS tree expensive

Evaluation

- Efficiency: experimental evaluation
- For both approaches: four variants with different compression levels
- Indexing time and size, retrieval time
- Collections: Reuters (many small articles); IEEE journal articles (complex structure)
- Indexing



- Retrieval

