Full-Text and Structural XML Indexing on B+-Tree

Toshiyuki Shimizu
Graduate School of Information Science, Nagoya University

Masatoshi Yoshikawa
Information Technology Center, Nagoya University

DEXA 2005 - 24th August

Outline

- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - Node Labeling Scheme
  - Text Fragments in the Index
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

Background

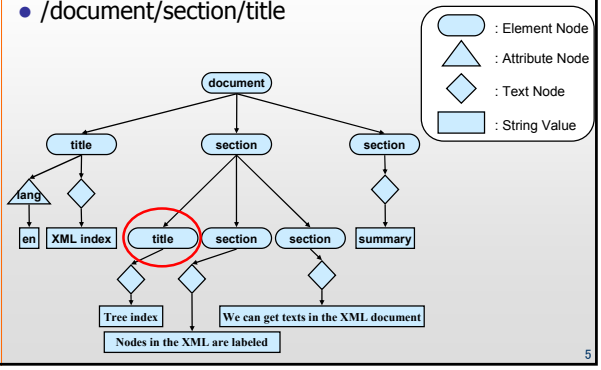
- Increase of XML Documents
- Efficient processing of XML queries is an important research topic
  - XPath, XQuery,…
- Accelerating structural and full-text searches for XML documents is very useful
  - Most past studies handle only structural searches

Related Works

- For structural searches
  - Node indexing
    - XISS, XR-tree
  - Path indexing
    - DataGuide, Index Fabric, F+B Index, APEX, …
  - Sequence-based indexing
    - ViST, PRIX
- For content (full-text) searches
  - On the Integration of Structure Indexes and Inverted Lists [Kaushik et al. 04]
    - Use inverted lists and structure index

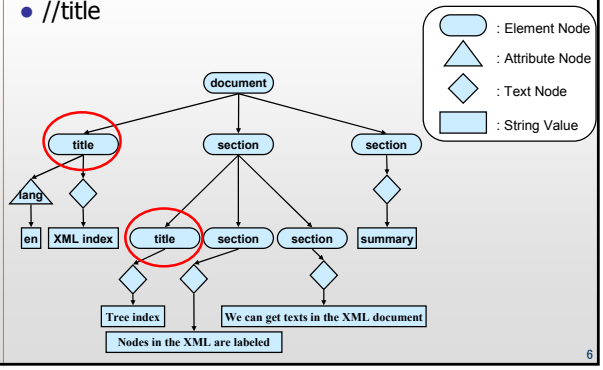
XPath Query for a Sample XML Document : Structural Search

- /document/section/title



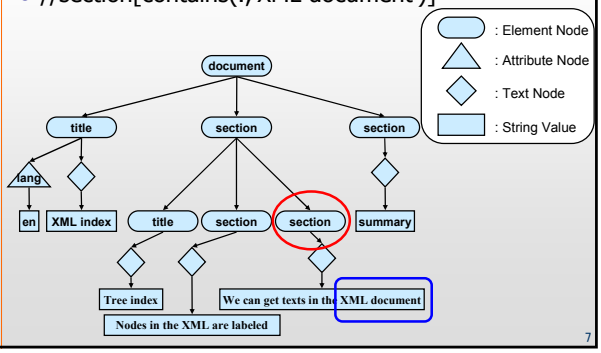
XPath Query for a Sample XML Document : Structural Search

- //title



XPath Query for a Sample XML Document : Content (Full-Text) Search

- //section[contains(.,'XML document')]



Outline

- Background, Related Works
- **Summary of our Contribution**
- Property of our Indices
  - Node Labeling Scheme
  - Text Fragments in the Index
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

Summary of our Contribution (1/2)

XICS : XML Indices
for Content and Structural search

- Use of B$^+$-Tree
  - B$^+$-tree is widely used in many database systems
  - Important design choice from a practical point of view
- Node identifier including path information
  - Path information is important for XML query processing

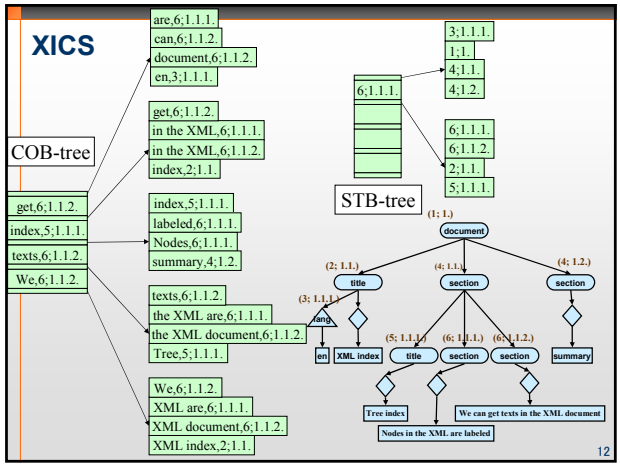
Summary of our Contribution (2/2)

- Retaining unique phrase of suffix texts
  - Keep phrase information
  - Novel approach for keeping text fragments
- Key ordering based on reverse path
  - For ancestor-descendant query
  - Search-key mapping
- Index compression using difference information
  - Difference text information among lined keys
  - Prefix B-tree approach

Outline

- Background, Related Works
- Summary of our Contribution
- **Property of our Indices**
  - Node Labeling Scheme
  - Text Fragments in the Index
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

XICS

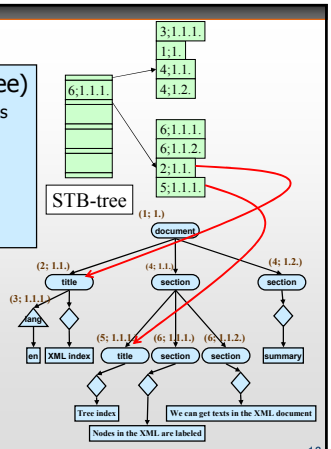


XICS

STB-tree (STructure B+-tree)

B+-Tree Index for Structural Searches

- Key : Node identifier
- Pointer : Disk address of a node corresponding to the node identifier
- Example of target query : //title

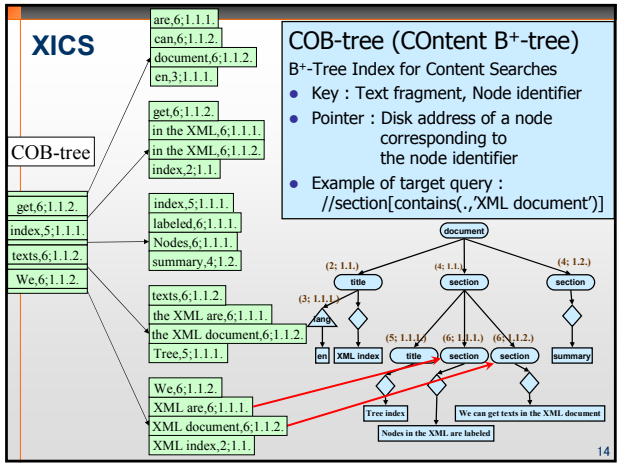


XICS

COB-tree (COntent B+-tree)

B+-Tree Index for Content Searches

- Key : Text fragment, Node identifier
- Pointer : Disk address of a node corresponding to the node identifier
- Example of target query : //section[contains(.,'XML document')]



Outline

- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - **Node Labeling Scheme**
  - Text Fragments in the Index
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

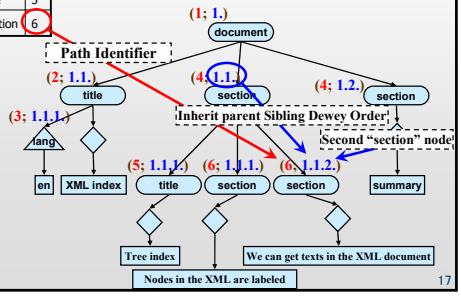
Node Labeling Scheme

- Node labeling schemes play an important role in XML query processing
  - (preorder, postorder), Dewey Order, …
- We have designed a node labeling scheme which contains path information and sibling information
  - Path Identifier
  - Sibling Dewey Order

An Example of XML Document with Node Identifiers

/document	1
/document/title	2
/document/title/@lang	3
/document/section	4
/document/section/title	5
/document/section/section	6

(Path Identifier; Sibling Dewey Order)



Outline

- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - Node Labeling Scheme
  - **Text Fragments in the Index**
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

Text Fragments in COB-tree (1/2)

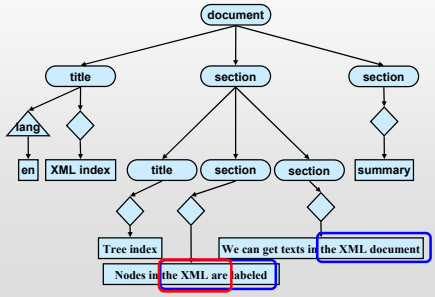
- Use Suffix Texts
  - ex) Suffix Texts of "**Nodes in the XML are labeled**"

Nodes in the XML are labeled
in the XML are labeled
the XML are labeled
XML are labeled
are labeled
labeled

- Keeping all phrase information in the index increases the index size

Text Fragments in COB-tree (2/2)

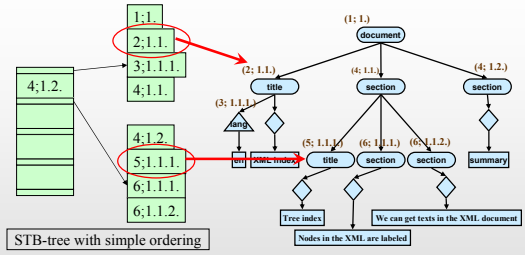
- Keep only the words necessary to distinguish the phrase from other phrases



Outline

- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - Node Labeling Scheme
  - Text Fragments in the Index
  - **Order of the Search Keys**
  - Text Compression in the Index
- Query Processing
- Experiments
- Conclusions, Future Works

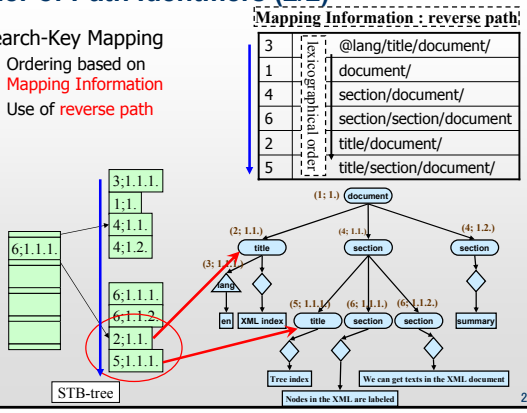
Order of Path Identifiers (1/2)



- Ordering path identifiers simply by their value is not efficient
  - A query that contains "//", such as "//title"

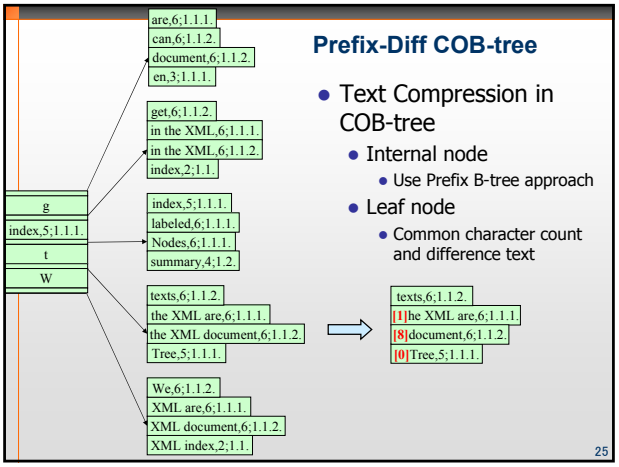
Order of Path Identifiers (2/2)

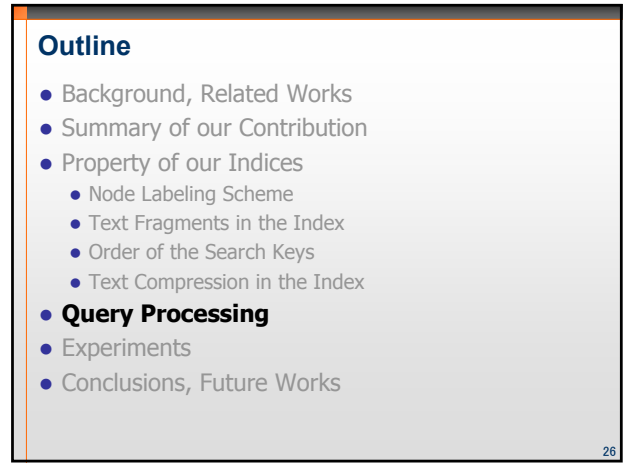
- Search-Key Mapping
  - Ordering based on Mapping Information
  - Use of reverse path

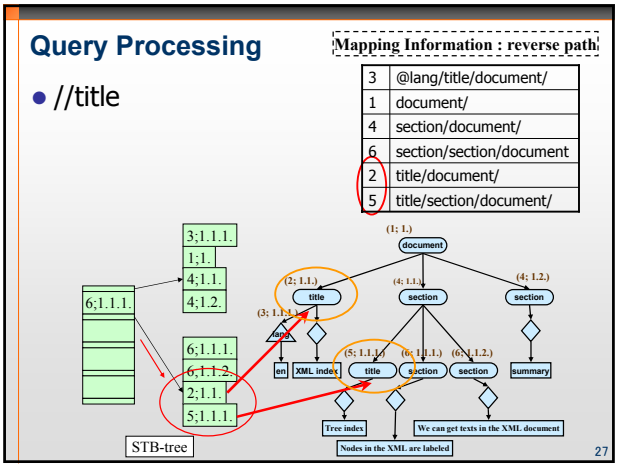


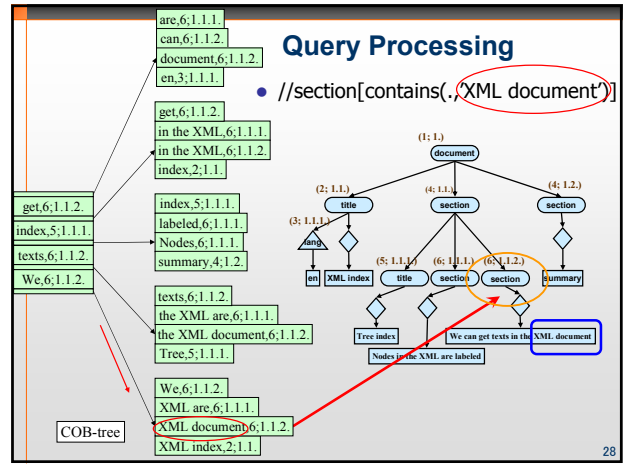
Outline

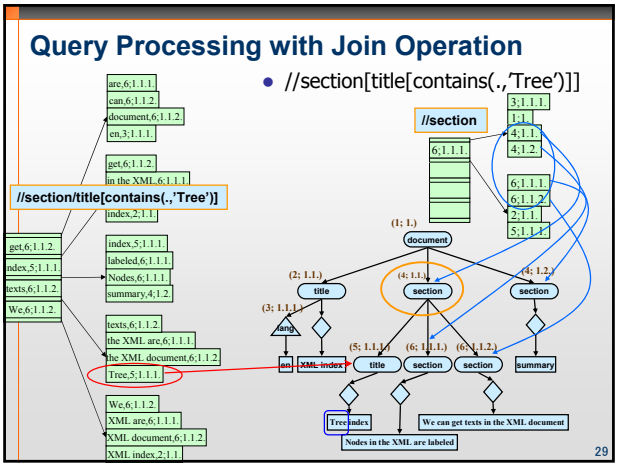
- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - Node Labeling Scheme
  - Text Fragments in the Index
  - Order of the Search Keys
  - **Text Compression in the Index**
- Query Processing
- Experiments
- Conclusions, Future Works

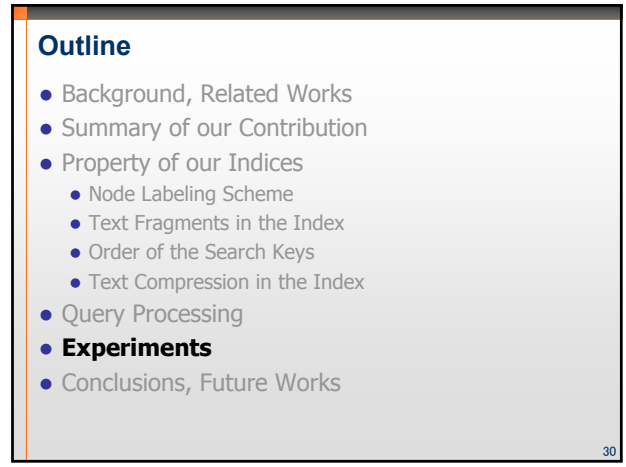

are,6;1.1.1.
can,6;1.1.2.
document,6;1.1.2.
en,3;1.1.1.
get,6;1.1.2.
in the XML,6;1.1.1.
in the XML,6;1.1.2.
index,2;1.1.
g
index,5;1.1.1.
t
W
index,5;1.1.1.
labeled,6;1.1.1.
Nodes,6;1.1.1.
summary,4;1.2.
texts,6;1.1.2.
the XML are,6;1.1.1.
the XML document,6;1.1.2.
Tree,5;1.1.1.
We,6;1.1.2.
XML are,6;1.1.1.
XML document,6;1.1.2.
XML index,2;1.1.
Prefix-Diff COB-tree
Text Compression in COB-tree
Internal node
Use Prefix B-tree approach
Leaf node
Common character count and difference text
texts,6;1.1.2.
[1]he XML are,6;1.1.1.
[8]document,6;1.1.2.
[0]Tree,5;1.1.1.
25


Outline
Background, Related Works
Summary of our Contribution
Property of our Indices
Node Labeling Scheme
Text Fragments in the Index
Order of the Search Keys
Text Compression in the Index
Query Processing
Experiments
Conclusions, Future Works
26


Query Processing
//title
Mapping Information : reverse path
3 @lang/title/document/
1 document/
4 section/document/
6 section/section/document
2 title/document/
5 title/section/document/
3;1.1.1.
1;1.
4;1.1.
4;1.2.
6;1.1.1.
6;1.1.1.
6,1.1.2.
2;1.1.
5;1.1.1.
STB-tree
(1; 1.) document
(2; 1.1.) title
(4; 1.1.) section
(4; 1.2.) section
(3; 1.1.1.) lang
en
XML index
(5; 1.1.1.) title
(6; 1.1.1.) section
(6; 1.1.2.) section
summary
Tree index
We can get texts in the XML document
Nodes in the XML are labeled
27


Query Processing
//section[contains(.,'XML document')]
are,6;1.1.1.
can,6;1.1.2.
document,6;1.1.2.
en,3;1.1.1.
get,6;1.1.2.
in the XML,6;1.1.1.
in the XML,6;1.1.2.
index,2;1.1.
get,6;1.1.2.
index,5;1.1.1.
texts,6;1.1.2.
We,6;1.1.2.
index,5;1.1.1.
labeled,6;1.1.1.
Nodes,6;1.1.1.
summary,4;1.2.
texts,6;1.1.2.
the XML are,6;1.1.1.
the XML document,6;1.1.2.
Tree,5;1.1.1.
We,6;1.1.2.
XML are,6;1.1.1.
XML document,6;1.1.2.
XML index,2;1.1.
COB-tree
(1; 1.) document
(2; 1.1.) title
(4; 1.1.) section
(4; 1.2.) section
(3; 1.1.1.) lang
en
XML index
(5; 1.1.1.) title
(6; 1.1.1.) section
(6; 1.1.2.) section
summary
Tree index
We can get texts in the XML document
Nodes in the XML are labeled
28


Query Processing with Join Operation
//section[title[contains(.,'Tree')]]
//section/title[contains(.,'Tree')]
are,6;1.1.1.
can,6;1.1.2.
document,6;1.1.2.
en,3;1.1.1.
get,6;1.1.2.
in the XML,6;1.1.1.
index,2;1.1.
get,6;1.1.2.
index,5;1.1.1.
texts,6;1.1.2.
We,6;1.1.2.
index,5;1.1.1.
labeled,6;1.1.1.
Nodes,6;1.1.1.
summary,4;1.2.
texts,6;1.1.2.
the XML are,6;1.1.1.
the XML document,6;1.1.2.
Tree,5;1.1.1.
We,6;1.1.2.
XML are,6;1.1.1.
XML document,6;1.1.2.
XML index,2;1.1.
//section
3;1.1.1.
1;1.
4;1.1.
4;1.2.
6;1.1.1.
6;1.1.1.
6;1.1.2.
2;1.1.
5;1.1.1.
(1; 1.) document
(2; 1.1.) title
(4; 1.1.) section
(4; 1.2.) section
(3; 1.1.1.) lang
en
XML index
(5; 1.1.1.) title
(6; 1.1.1.) section
(6; 1.1.2.) section
summary
Tree index
We can get texts in the XML document
Nodes in the XML are labeled
29

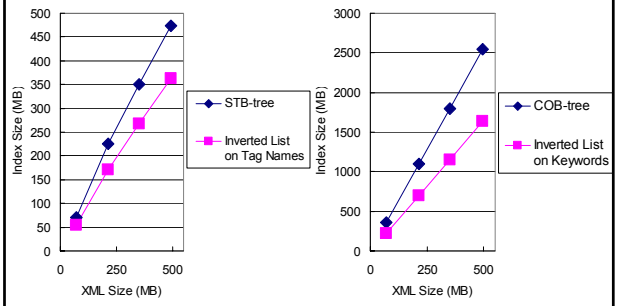

Outline
Background, Related Works
Summary of our Contribution
Property of our Indices
Node Labeling Scheme
Text Fragments in the Index
Order of the Search Keys
Text Compression in the Index
Query Processing
Experiments
Conclusions, Future Works
30

Experiments

- Implementation using GiST (Generalized Search Tree)
- INEX XML document set
  - The articles of the IEEE Computer Society's publications are marked up in XML
- Index Size and Query Processing Time
  - Compare with the method of "On the Integration of Structure Indexes and Inverted Lists" [Kaushik et al. 04]
    - Inverted List on Tag Names and Keywords
    - We call this method *Integration* here
  - For comparison, we also retain a single word in our approach
- Experimental Setup
  - CPU : Pentium M 1.40 GHz
  - RAM : 512 MB
  - HDD : 5,400 rpm
  - OS : Windows XP SP1

Index Size

- XICS is about 1.4 times larger than *Integration*



Query Processing Time

Q1	/books/journal/title
Q2	//sec/st
Q3	//article/fm/abs/p[contains(., 'software')]
Q4	//article[contains(./fm/abs/p, 'software')]
Q5	//sec[contains(./st, 'animation')]
Q6	//bdy/sec[1]/st[contains(., 'XML')]

* Use whole INEX document set (about 495 MB)

Search keys include path information

Restrict nodes in the join operation

Processing Time (ms)

	XICS	Integration
	76	65
	410	1,037
Q3	79	19,040
Q4	278	21,450
Q5	332	3,649
Q6	68	46,611

Index Traversal Time (ms)

	XICS	Integration
Q1	76	65
Q2	410	1,037
Q3	79	5,137
Q4	188	446
Q5	321	510
Q6	68	1,658

Outline

- Background, Related Works
- Summary of our Contribution
- Property of our Indices
  - Node Labeling Scheme
  - Text Fragments in the Index
  - Order of the Search Keys
  - Text Compression in the Index
- Query Processing
- Experiments
- **Conclusions, Future Works**

Conclusions

- We proposed XICS for accelerating structural and full-text searches for XML documents
  - STB-tree
    - Node identifier
  - COB-tree
    - Pair of text fragment and node identifier
- Query processing algorithm using XICS
- Experiments on index size and query processing time

Future Works

- Introduce schema information
  - Utilize schema information for index construction or query processing
- Introduce data statistics and query workloads
- Improve the join operation
- Consider document updates
  - Design dynamic index that is efficient for node insertion or deletion