XMLデータベース技術に関する研究動向
― 研究の流れとXML索引付け ―

江田 毅晴
NTTサイバースペース研究所
清水 敏之
名古屋大学 情報科学研究科

DEWS2006 ミニサーベイ

2006/03/01

発表の流れ

1. 背景
   - XMLに対する検索
2. RDBへのマッピング
3. ノードラベリング手法
   - 構造ジョイン (Structural Join)
4. 索引付け
   - ノード索引 (Node Indexing)
   - 経路索引 (Path Indexing)
   - 系列索引 (Sequence-Based Indexing)
   - 全文索引 (Full-Text Index)
5. まとめ

XMLとは？

- Extensible Markup Language
  - W3Cによって1998年に標準化されたデータフォーマット
  - SGML互換
- 多数の関連仕様
  - XML Schema, XQuery, XSL, XInclude, etc.
- インターネットで使われることを想定

DTD (Document Type Definition)

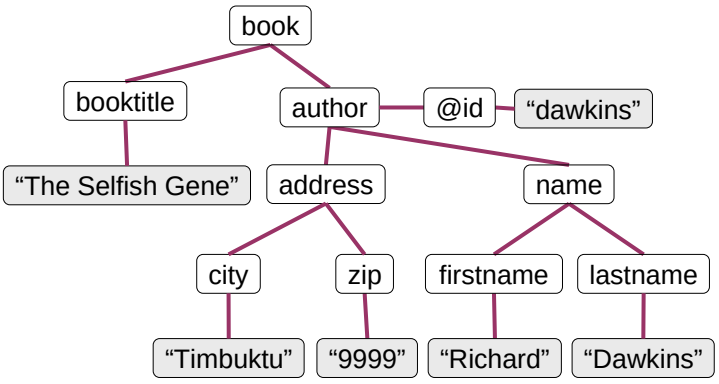
サンプル XML

```
<book>
 <booktitle>The Selfish Gene</booktitle>
 <author id = "dawkins">
    <name>
      <firstname>Richard</firstname>
      <lastname>Dawkins</lastname>
    </name>
    <address>
      <city>Timbuktu</city>
      <zip>9999</zip>
    </address>
 </author>
</book>
```

サンプル DTD

```
<!ELEMENT book (booktitle, author)>
<!ELEMENT article (title, author*, contactauthor)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF #IMPLIED>
<!ELEMENT monograph (title, author, editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author (name, address)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT name (firstname?, lastname)>
<!ELEMENT booktitle (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address  (city,zip)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```


サンプルXMLの木構造表現

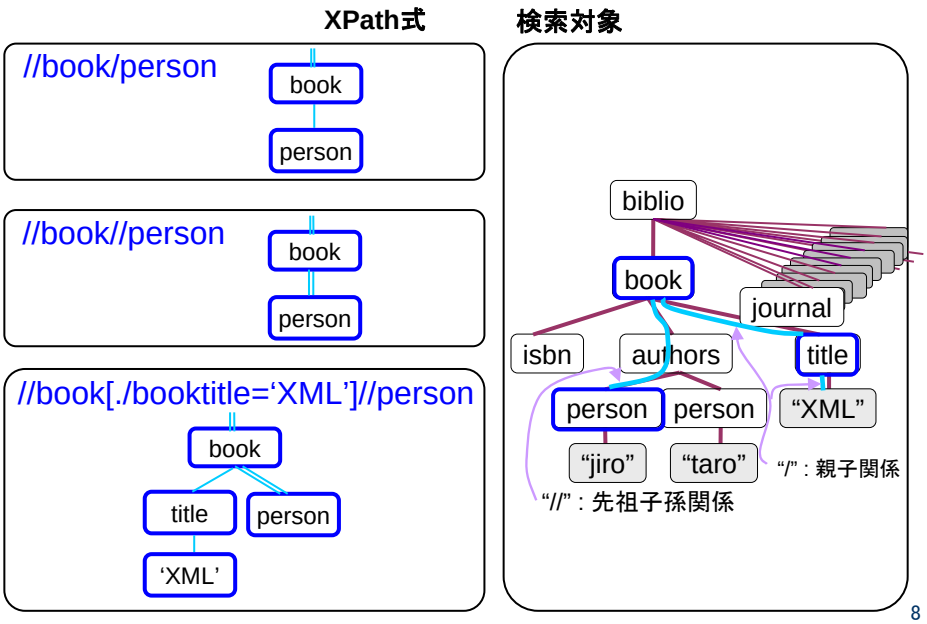


XMLデータの検索とは？

1. XMLパーサ (DOM, SAX, etc.)を使う
   - 手続き型言語による検索⇔簡単な検索も一定のコスト
   - 毎回全てをメモリに読む必要がある

2. XQuery (XPath) を使う (XSLT)
   - 宣言的問合せ言語による検索⇔簡単な検索は簡単に
     - XML⇔XQuery (RDB (Relation) ⇔ SQL)
     - FLWOR 構文
     - 問合せ最適化の余地
   - XPath – 木の位置指定とフィルタ条件からなるXQueryの構成要素
   - XMLデータベースとして必要な機能

XPathの例



XMLデータのRDBへのマッピング方法

RDBへマップする理由

- データサイズが大きいから
- 既にいろんな機能が提供されているから
  - API, トランザクション, etc.
- 蓄積された問合せ最適化技術

課題

- RDBとXMLのインピーダンスミスマッチ
- XPath検索のRDBでの実現

方法

- 構造写像アプローチ
  - DTD(XML Schema) を利用する
  - 情報をテーブルの属性として保持（インライニング）
- モデル写像アプローチ
  - DTD(XML Schema) を利用しない
  - 木構造としてテーブルにマッピング

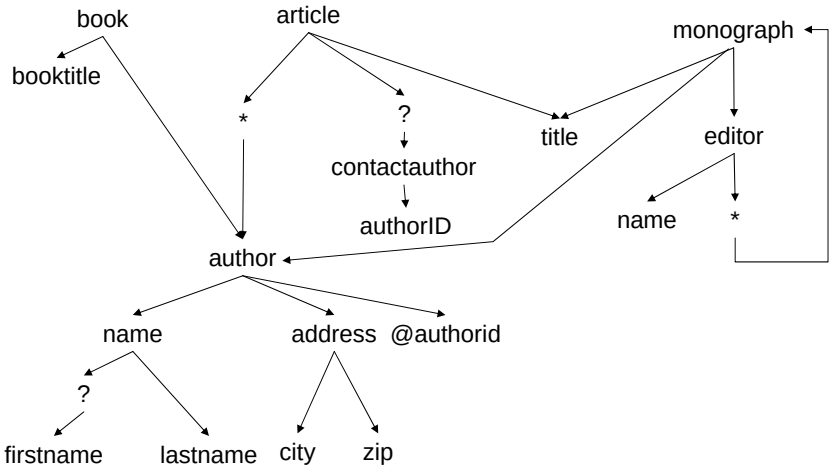
構造写像アプローチ [1]

考え方

- 要素をテーブルにする, or 属性にする（インライニング）
  - テーブルをどのタイミングで作成するか
- 木構造情報をどうやって保持するか

インライニング

- Basic
  - 要素をできるだけインライニングする
- Shared
  - 出来るだけ情報の重複を省く
- Hybrid

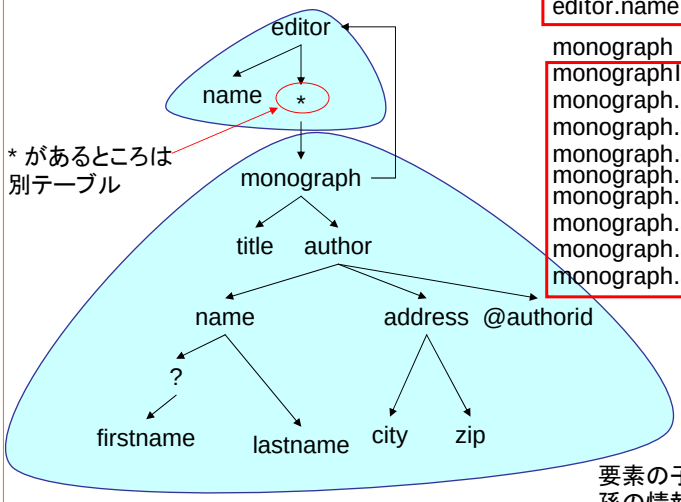


DTD グラフ
•DTDをグラフ表現したもの
•インスタンスのルートはどこからでも良い

Basic

editor スキーマ
editorID,
editor.parentID,
editor.name

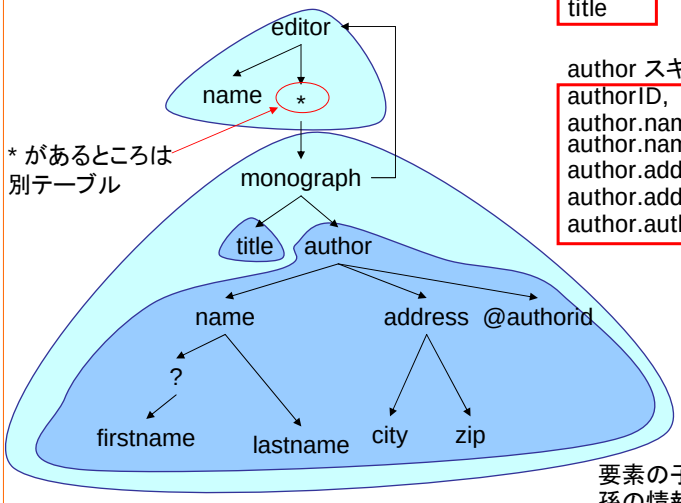
monograph スキーマ
monographID,
monograph.parentID,
monograph.title,
monograph.editor.name,
monograph.author.name.firstname,
monograph.author.name.lastname,
monograph.author.address.city,
monograph.author.address.zip,
monograph.author.@authorid



要素の子
孫の情報を
属性化する

editorノードグラフ
•DTDグラフを, editor ノードから展開したもの

Basic



* があるところは
別テーブル

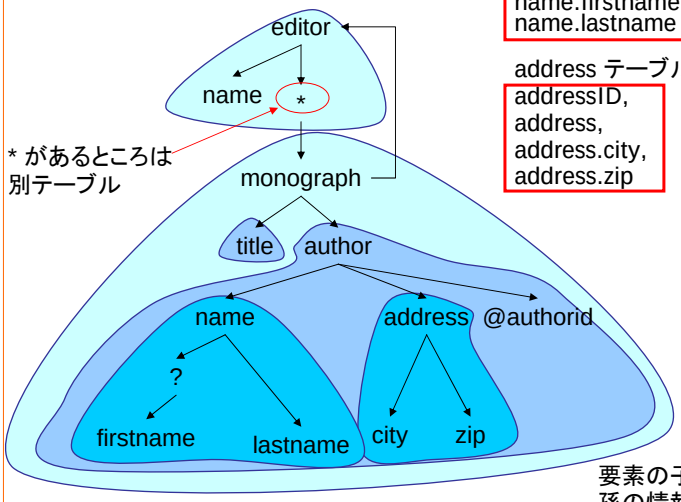
title スキーマ
titleID,
title

author スキーマ
authorID,
author.name.firstname,
author.name.lastname,
author.address.city,
author.address.zip,
author.authorid

要素の子
孫の情報を
属性化する

editorノードグラフ
•DTDグラフを, editor ノードから展開したもの

Basic



* があるところは
別テーブル

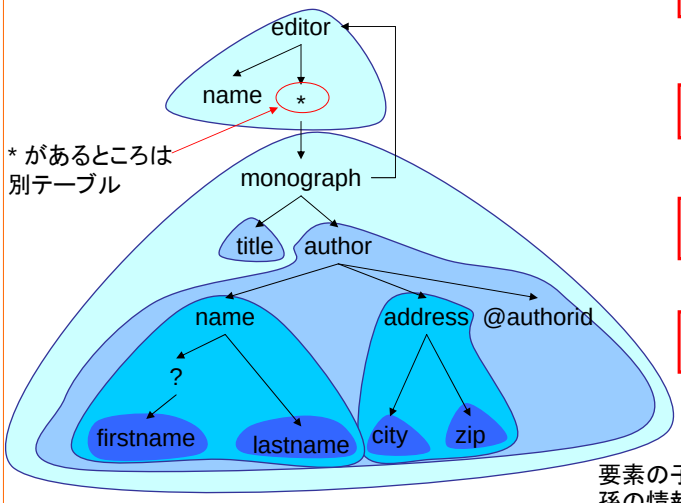
name テーブル
nameID,
name.firstname,
name.lastname

address テーブル
addressID,
address,
address.city,
address.zip

要素の子
孫の情報を
属性化する

editorノードグラフ
•DTDグラフを, editor ノードから展開したもの

Basic



* があるところは
別テーブル

firstname テーブル
firstnameID,
firstname

lastname テーブル
lastnameID,
lastname

city テーブル
cityID,
city

zip テーブル
zipID,
zip

要素の子
孫の情報を
属性化する

editorノードグラフ
•DTDグラフを, editor ノードから展開したもの

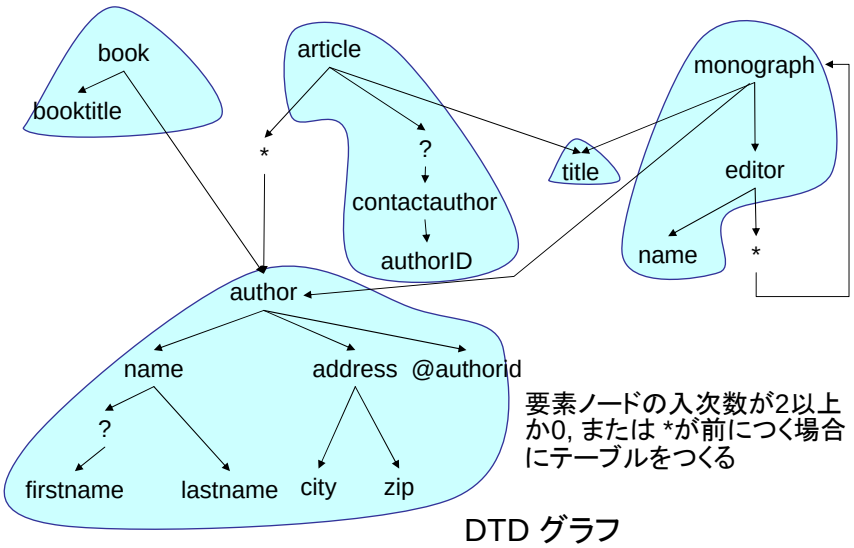
関係スキーマ(Basic)

book (bookID, book.booktitle, book.author.name.firstname,
book.author.name.lastname, book.author.address, author.authorid)
booktitle (booktitleID, booktitle)
article (articleID, article.contactauthor.authorid, article.title)
article.author (article.authorID, article.author.parentID,
article.author.name.firstname, article.author.name.lastname,
article.author.address, article.author.authorid)
contactauthor (contactauthorID, contctauthor.authorid)
title (titleID, title)
monograph (monographID, monograph.parentID, monograph.title,
monograph.editor.name, monograph.author.name.firstname,
monograph.author.name.lastname, monograph.author.address,
monograph.author.authorid)
editor (editorID, editor.parentID, editor.name)
editor.monograph (editr.monographID, editor.monograph.parentID,
editor.monograph.title, editor.monograph.author.name.firstname,
editor.monograph.author.name.lastname, editor.monograph.author.address,
editor.monograph.author.authorid)
author (authorID, author.name.firstname, author.name.lastname, author.address,
author.authorid)
name (nameID, name.firstname, name.lastname)
firstname (firstnameID, firstname)
lastname (lastnameID, lastname)
address (addressID, address)
型情報は省略

* の箇所は,
parentが必要

同じタグの情報のテーブルが複数作成されるため, UNIONのコストが大きくなる

Shared



要素ノードの入次数が2以上か0, または *が前につく場合にテーブルをつくる

DTD グラフ

関係スキーマ(Shared)

book
(bookID, book.booktitle.isroot, book.booktitle)
article
(articleID, article.contactauthor.isroot, article.contactauthor.authorid)
monograph
(monographID, monograph.parentID, monograph.parentCODE, monograph.editor.isroot, monograph.editor.name)
title
(titleID, title.parentID, title.parentCODE, title)
author
(authorID, author.parentID, author.parentCODE, author.name.isroot, author.name.firstname, author.name.lastname.isroot, author.name.lastname, author.address.isroot, author.address, author.authorid)

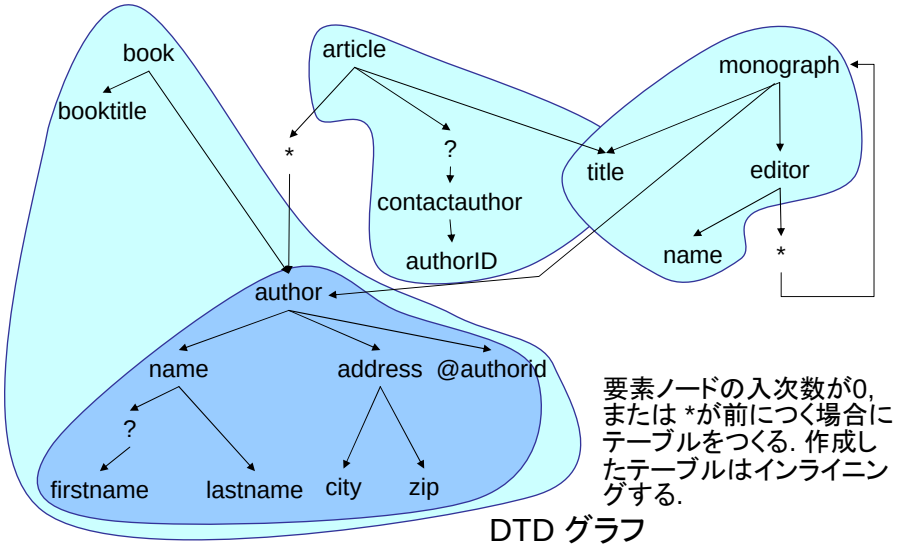
入次数が2以上および* の箇所は, parentが必要

isroot: 中間ノードからの開始判定用識別子

情報が分断されるため, ジョイン回数が増える

型情報は省略

Hybrid



要素ノードの入次数が0, または *が前につく場合にテーブルをつくる. 作成したテーブルはインライニングする.

DTD グラフ

関係スキーマ(Hybrid)

book
(bookID, book.booktitle.isroot, book.booktitle, author.name.firstname, author.name.lastname, author.address, author.authorid)
article
(articleID, article.contactauthor.isroot, article.contactauthor.authorid, article.title.isroot, article.title)
monograph
(monographID, monograph.parentID, monograph.parentCODE, monograph.title, monograph.editor.isroot, monograph.editor.name, author.name.firstname, author.name.lastname, author.address, author.authorid)
author
(articleID, author.parentID, author.parentCODE, author.name.isroot, author.name.firstname.isroot, author.name.firstname, author.name.lastname.isroot, author.name.lastname, author.address.isroot, author.address, author.authorid)

* が前につく場合は, parent が必要

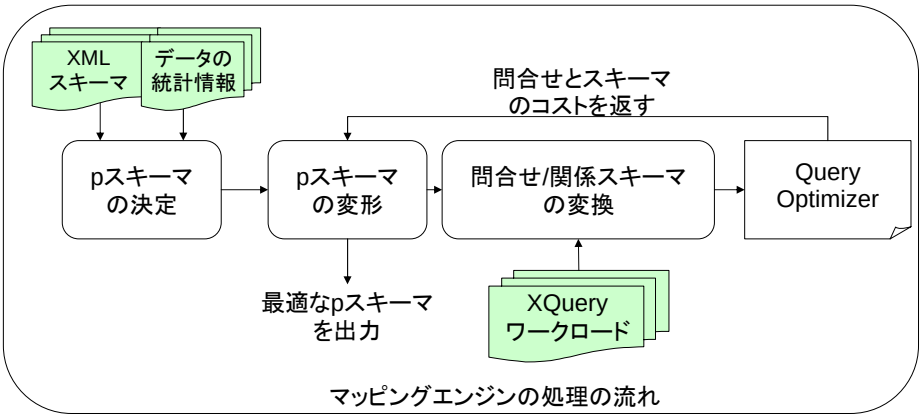
isroot: 中間ノードからの開始判定用識別子

テーブル数も少なくジョイン回数も少ない

型情報は省略

Lego DB [2]

- （XML Schema）+（データの統計情報）+（クエリワークロード）で関係スキーマを設計



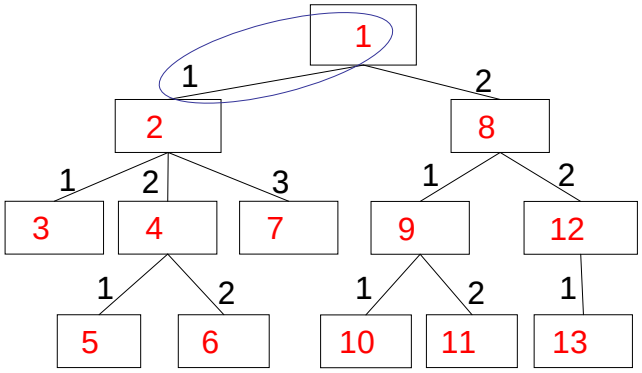
モデル写像アプローチ

ニーズはこちらの方が高い

XMLデータをラベル付木とみなしてマッピング

- エッジ アプローチ [3]
  - 全ての枝をテーブルとして保存
  - (source, ordinal, name, flag, target)
- XRel アプローチ [4]
  - 全てのノードを保存（枝は明示的には保存しない）
  - パス情報を別テーブルとして保存する

エッジアプローチ



1. ノードに識別子を与える
2. 枝に(識別子,順序)を対応付ける

エッジアプローチ

Edgeテーブル

key	source	ordinal	name	flag	target
1	1	Age	Int	*v1*
1	2	name	string	*v2*
1	3	address	string	*v3*
1	4	child	ref	**3**
1	5	child	ref	**4**
1	1	age	int	*v4*
..	..	..	..	

V_{int} テーブル

vid	value
v1	55
v4	38
v8	22
v13	7

V_{string} テーブル

vid	value
v2	Peter
v3	4711 Fruitdale Ave.
v5	Mary
v6	4711 Fruitdale Ave.

- パス式を処理するにはセルフジョインが必要
  //book//author//firstname

=> Edgeテーブルに対して2回ジョイン

XRel アプローチ

- パスとリージョンでノードを表現
  - パス： タグを連結したもの
  - リージョン： ノードの開始と終了のバイトオフセット値

ノードテーブル

docid	pathid	start	end
1	1	0	240
1	2	9	39
1	3	43	168
1	4	56	103
1	4	108	155
1	6	172	231

リージョン

属性テーブル

docid	pathid	start	end	value
1	5	57	57	NAIST
1	5	109	109	KAIST

テキストテーブル

docid	pathid	start	end	value
1	2	16	31	XML and...
1	4	84	94	Yamada ...
1	4	136	146	Sugita Ziro
1	6	181	221	XML ...

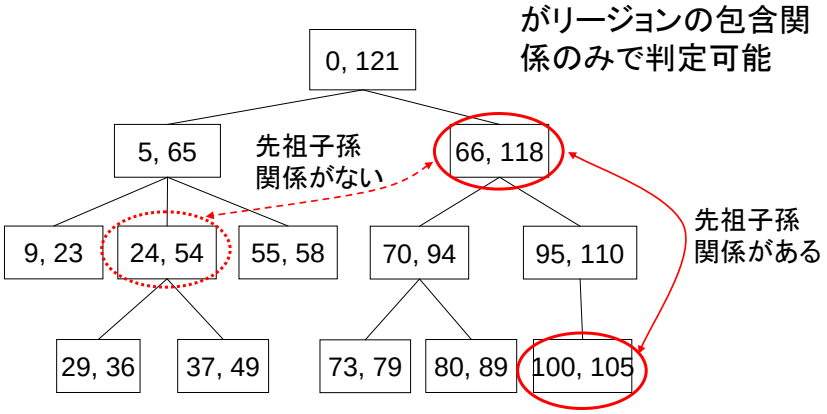
パステーブル

pathid	pathexpression
1	#/book
2	#/book#/title
3	#/book#/authors
4	#/book#/authors#/author
5	#/book#/authors#/author#/@affiliation
6	#/book#/summary

XPath処理はパステーブルを使う

リージョン (XRel)

ノードの先祖子孫関係がリージョンの包含関係のみで判定可能



検索時に中間ノードをスキップ可能

RDB マッピングから学んだこと

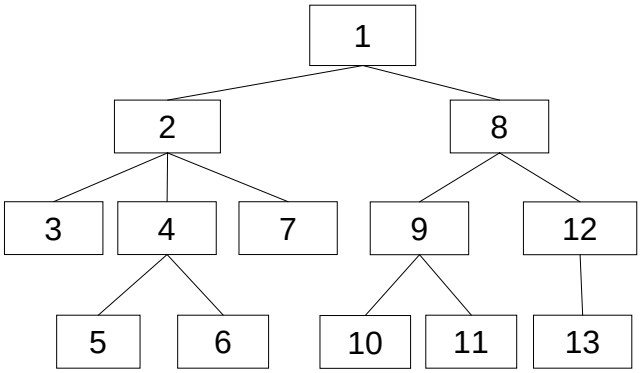
- エンコーディングは重要
  - → 様々なラベリングスキームの研究へ
    - 範囲, 接頭辞, k分木, 素数
- パス情報はXPath処理に有効
  - → 経路索引の研究へ
    - DataGuide, F&B Index, etc.
- XML特有のジョイン（構造ジョイン）
  - 木構造関係の条件によるジョイン
  - 既存のRDBのジョインは最適でない

ラベリングスキーム

- 各ノードに木構造の情報を保持するラベルを付与
  - ラベルを付与して木構造を切断
  - 識別子の役割
  - 文書順の保持
  - 二つのノード間に 親子関係, 先祖子孫関係, 兄弟関係があるかラベルのみでの判定
- 例

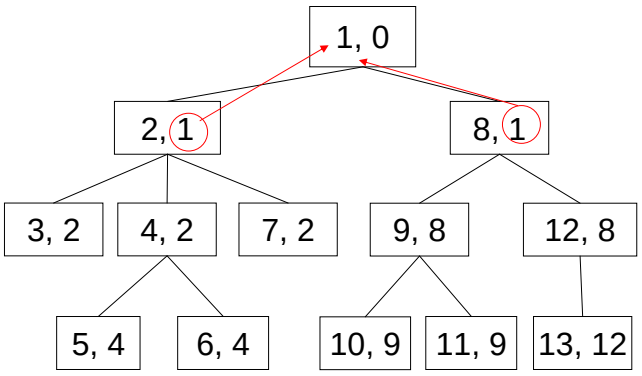
範囲, 接頭辞, k分木への埋め込み, 素数

深さ優先 (preorder) (＝文書順)



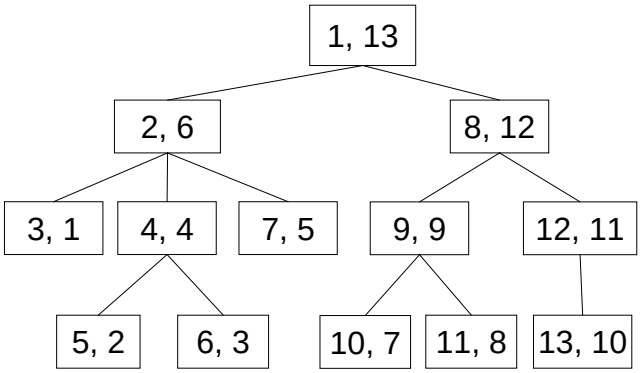
・識別子
・文書順

エッジアプローチ (preorder, parent)



・識別子
・文書順
・親子関係

範囲ラベル (preorder, postorder) [5]



- 識別子
- 文書順
- 先祖子孫関係

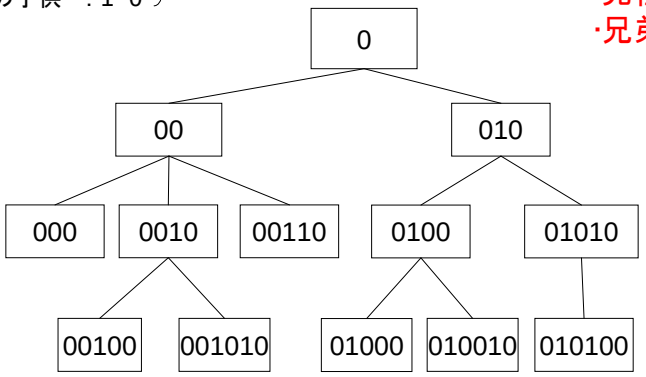
レベルを付与すると
- 親子関係

接頭辞ラベル [6]

1番目の子供 ：0
2番目の子供 ：10
3番目の子供 ：110
・・・
i番目の子供 ：$1^{i}0$

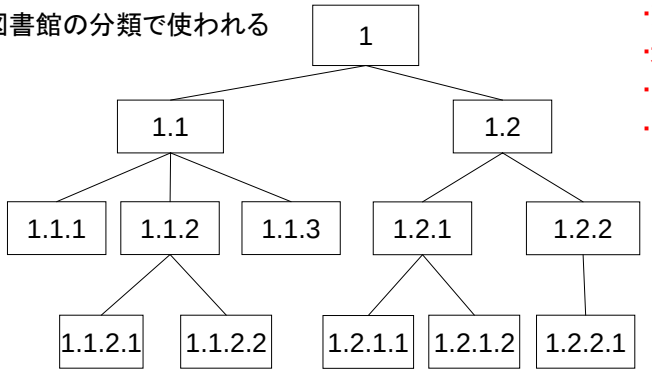
を親のラベルに結合

- 識別子
- 文書順
- 親子関係
- 先祖子孫関係
- 兄弟関係



Dewey Order (接頭辞ラベル) [7]

- “.”（ドット）がデリミタ
- 図書館の分類で使われる

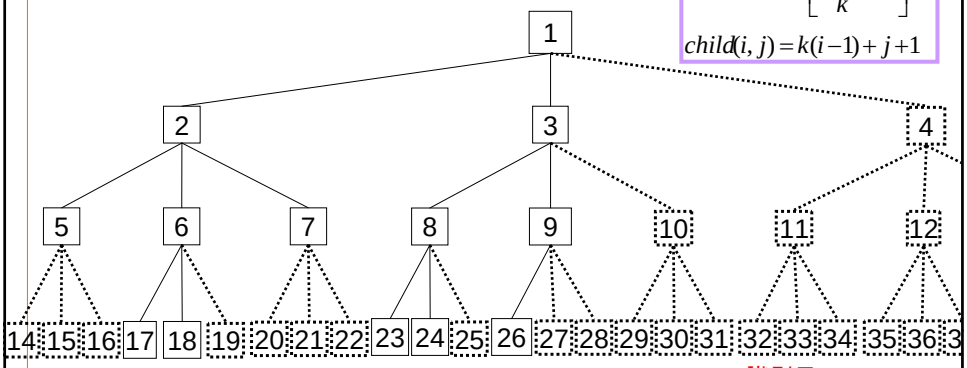


- 識別子
- 文書順
- 親子関係
- 先祖子孫関係
- 兄弟関係

k分木への埋めこみ [8]

$$parent(i) = \left\lfloor \frac{i-2}{k} + 1 \right\rfloor$$

$$child(i, j) = k(i-1) + j + 1$$



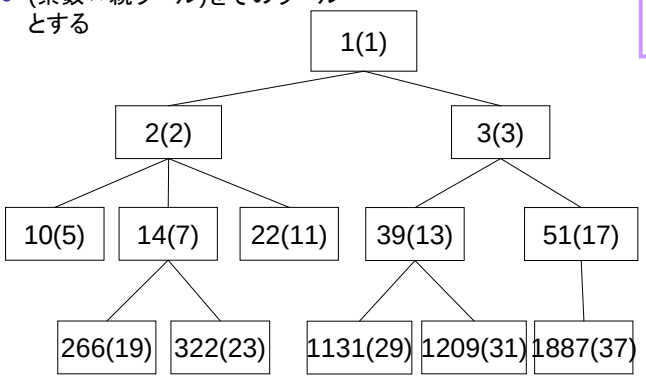
ゆがんだ木の場合，
ラベルが指数的に増える

- 識別子
- 文書順
- 親子関係
- 先祖子孫関係
- 兄弟関係

素数ラベル [9]

Top-down Prime

- 各ノードに素数を割り当てる
- (素数×親ラベル)をそのラベルとする



vi が vj の先祖
⇔
label(vj) が label(vi) で割り切れる

·識別子
·親子関係
·先祖子孫関係

更新があった場合，新たな素数を割り当てれば挿入可能

構造ジョイン

- 先祖子孫関係を保持するラベルを使った構造的な関係によるジョイン
  - //Book//person を高速化
    - 中間ノードの探索が必要ない

Structural Joins [10]

- Tree-merge
  - ソートマージを範囲情報を用いて改良．パイプライン可能．
- Stack-tree
  - 先祖候補を Stack に保持する．パイプライン可能．

Nested loop	Tree-merge	Stack-tree
O(n*m)	O(n*m)	O(n+m+out)

構造ジョインを行った場合の計算量

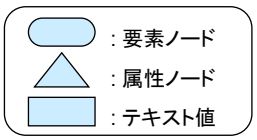
XML索引付け

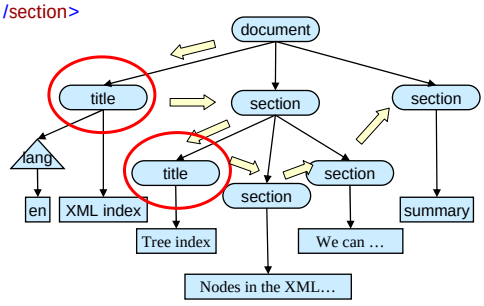
- XMLに対する問合せ処理の高速化
  - Native XMLDBの重要な要素
- 多くのXML索引が提案されている
  - 活発な研究分野

索引付けに対するモチベーション

- //title
  - 要素名がtitleであるノードを取得

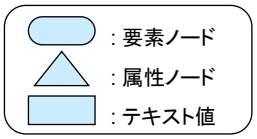
```
<?xml version="1.0"?>
<document>
  <title lang="en">XML index</title>
  <section>
    <title>Tree index</title>
    <section>Nodes in the XML are labeled</section>
    <section>We can get texts in the XML
    document</section>
  </section>
  <section>summary</section>
</document>
```

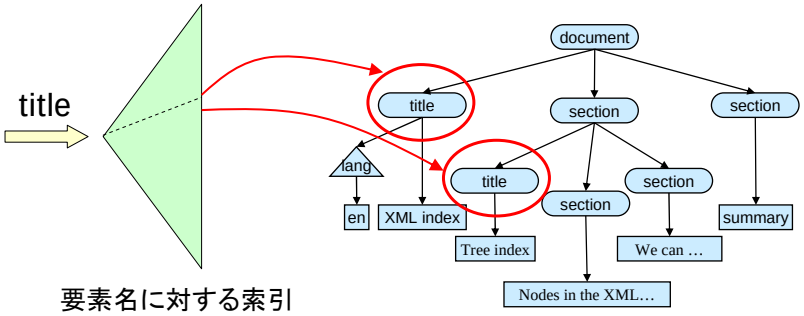




索引付けに対するモチベーション

- //title
  - 要素名が“title”であるノードを取得



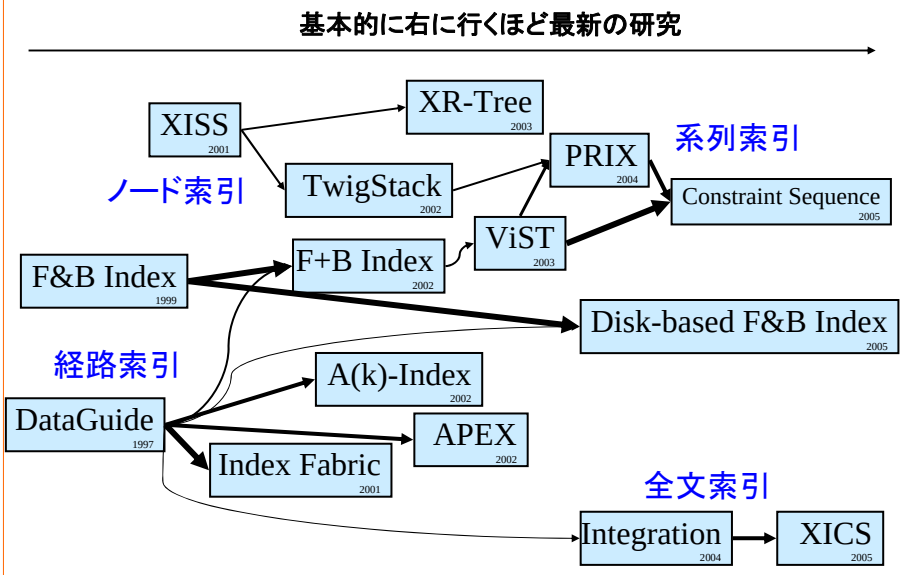


要素名に対する索引

XML索引分類

- ノード索引 (Node Indexing)
  - ノードを単位とした索引
- 経路索引 (Path Indexing)
  - 経路を用いた索引
- 系列索引 (Sequence-Based Indexing)
  - XMLを系列とみなして処理する索引
- 全文索引 (Full-Text Index)
  - XMLに含まれるテキスト値の全文検索もサポートする索引

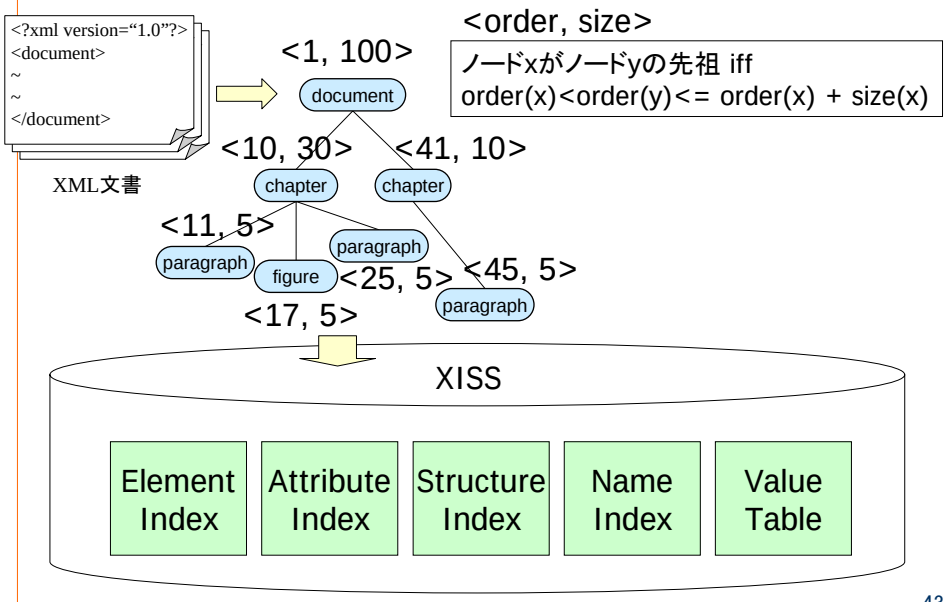
XML索引付け研究マップ



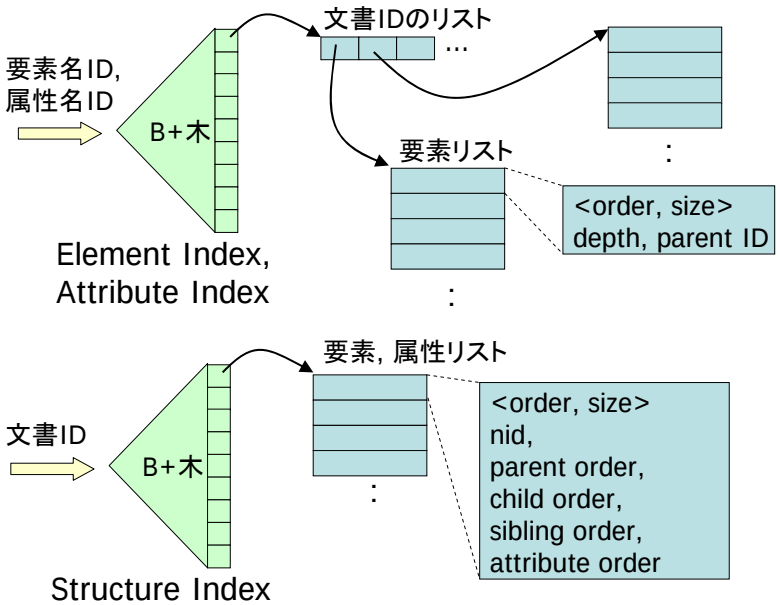
ノード索引 (Node Indexing)

- ノードを単位として索引を構築
  - タグ名・属性名
  - ノード同士の関係(先祖/子孫関係，親子関係など) は構造ジョインによって求める
- 主な手順
  - ノードに対して識別子を付与
  - ノード識別子をエントリに含む索引を構築
    - 識別子以外の情報も含める
- 研究事例
  - XISS [11]

XISS

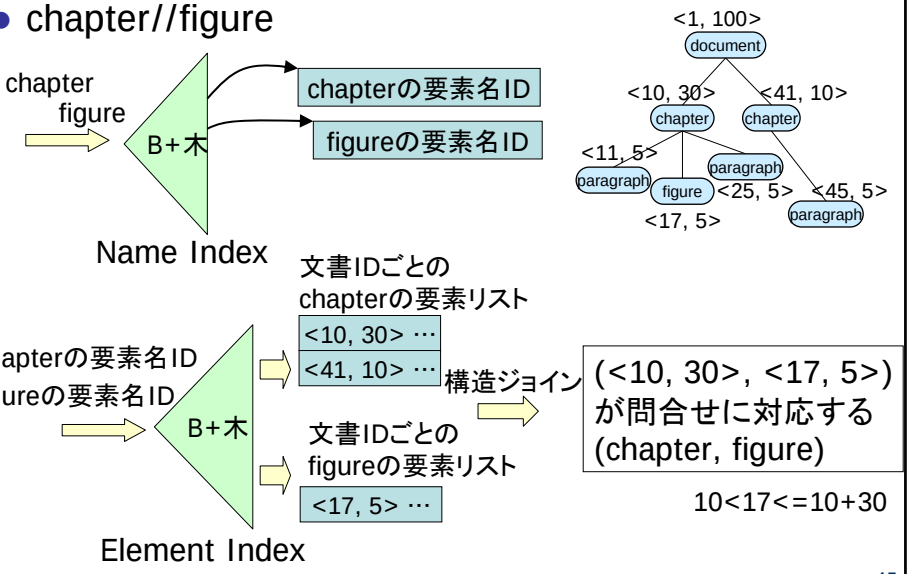


XISS 索引詳細



XISS 問合せ処理例

- chapter//figure



構造ジョイン高速化のための索引

- XR-Tree [12]
  - XML 文書中のノードに対して、(start, end) で表現される識別子を付与し、B+木を拡張した索引を構築
- TwigStack [13]
  - XB-Tree

ノード索引の長所と短所

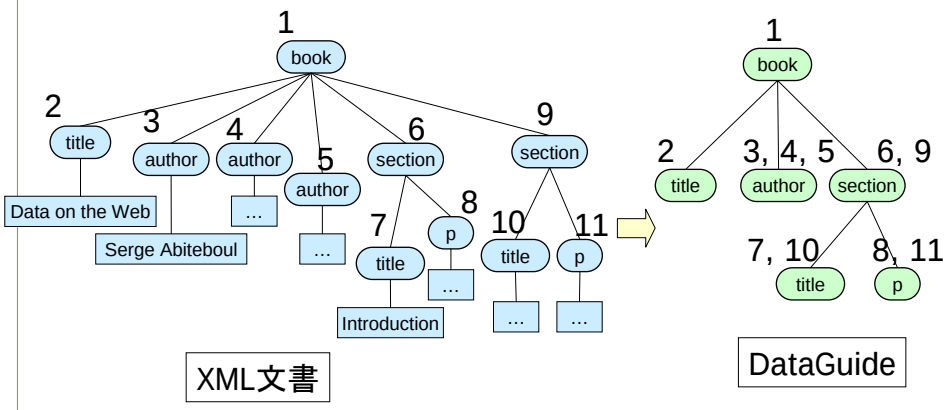
- 長所
  - 結合操作を行うことで広い範囲の問合せに対応
    - 先祖/子孫関係，親子関係だけでなく兄弟関係なども
- 短所
  - ステップごとに結合操作が必要
  - 問合せ中に出現するタグ名・属性名を持つノードを全て取得する

経路索引 (Path Indexing)

- XML中の経路を用いて索引を構築
  - 経路情報を効率的に処理する
  - 構造要約 (Structural Summary)
- 主な手順
  - ノードごとに経路を解析
  - 同じ経路を持つノードをまとめる
- 研究事例
  - DataGuide [14]

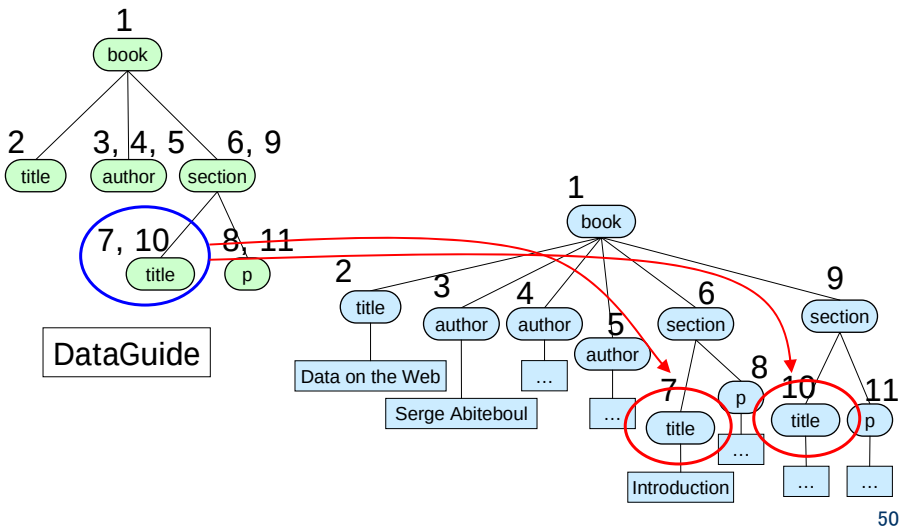
DataGuide

- 半構造データに対しての構造の要約



DataGuide 問合せ処理例

- /book/section/title

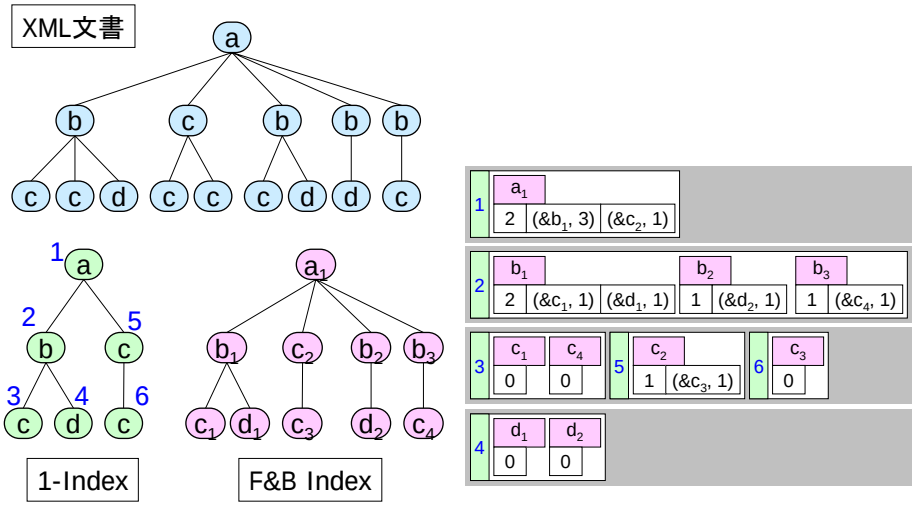


経路索引の研究事例

- Index Fabric [15]
  - 頻繁に問合せに現れるパターンを索引に加える
- A(k)-Index [16]
  - 局所的な経路情報のみを用いる
- APEX [17]
  - 問合せ頻度を考慮し、部分的な経路を保持することで更新にも強い索引
- F&B Index [18]
  - 枝分れ経路に対して索引を生成する
- F+B Index [19]
  - 枝分れ経路に対して索引を生成する
  - 索引サイズを抑えるために、どの枝分れ経路に対して索引付けするか制限する
- Disk-based F&B Index [20]
  - F&B indexのディスクへの格納

Disk-based F&B Index

- 効率的にクエリ処理可能なデータのクラスタリング



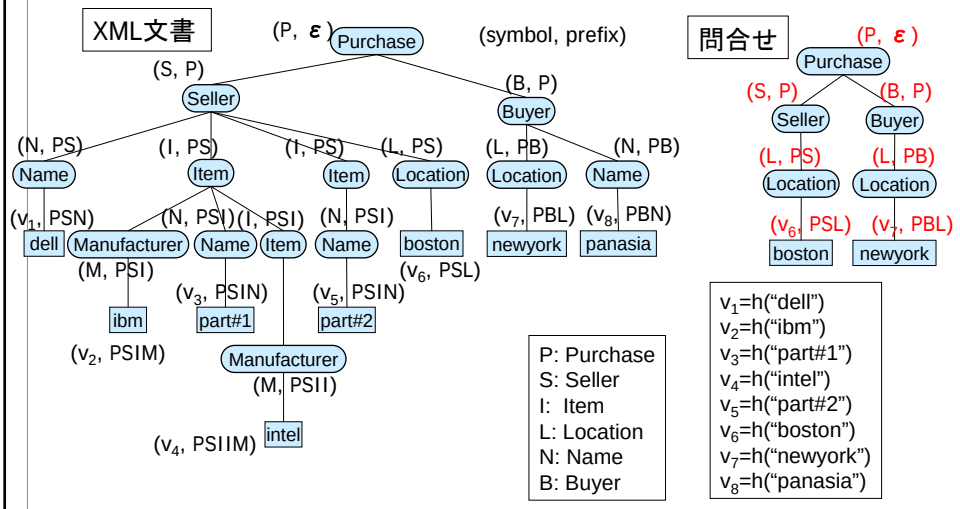
経路索引の長所と短所

- 長所
  - 索引付けした経路に対する問合せに関しては高速に処理できる
- 短所
  - それ以外の問合せに関しては結合操作を必要とするか処理できない

系列索引 (Sequence-Based Indexing)

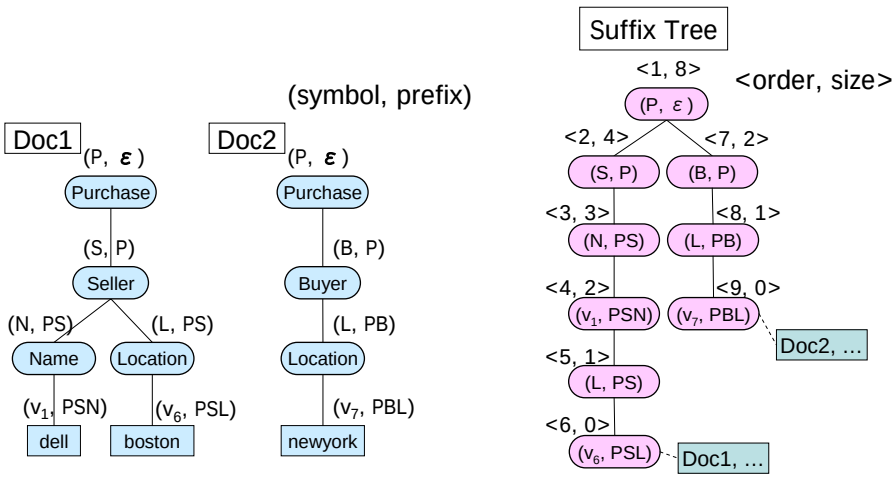
- XML文書と問合せをともに系列で表現し，部分系列一致により問合せを扱う
- 主な手順
  - ノードに対してラベルを付与
  - ラベルを特定の規則に従ってならべた系列を構築
  - 系列を基にして索引を構築
- 研究事例
  - ViST [21]
    - 系列索引の先駆け

ViST

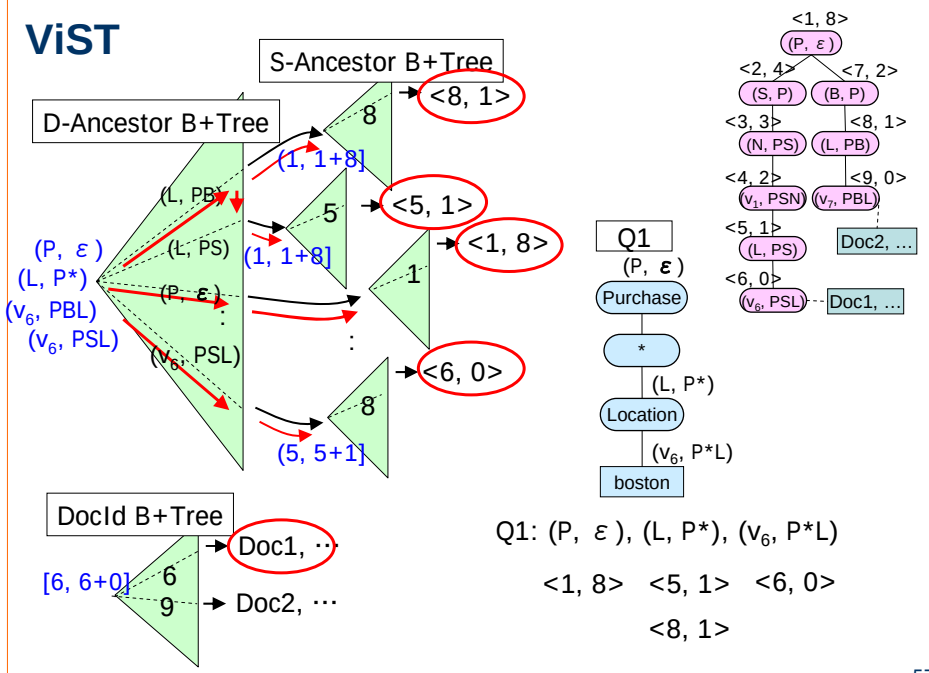


(P, ε), (S, P), (N, PS), (v_1, PSN), (I, PS), (M, PSI), (v_2, PSIM), (N, PSI), (v_3, PSIN), (I, PSI), (M, PSII), (v_4, PSIIM), (I, PS), (N, PSI), (v_5, PSIN), (L, PS), (v_6, PSL), (B, P), (L, PB), (v_7, PBL), (N, PB), (v_8, PBN)

ViST



Doc1: (P, ε), (S, P), (N, PS), (v_1, PSN), (L, PS), (v_6, PSL)
Doc2: (P, ε), (B, P), (L, PB), (v_7, PBL)



系列索引の研究事例

- PRIX [22]
  - Prüfer sequences を用いてXMLを系列化する
- Constraint Sequence [23]
  - 統計的な情報を用いることによりパフォーマンスに優れた系列化を提案

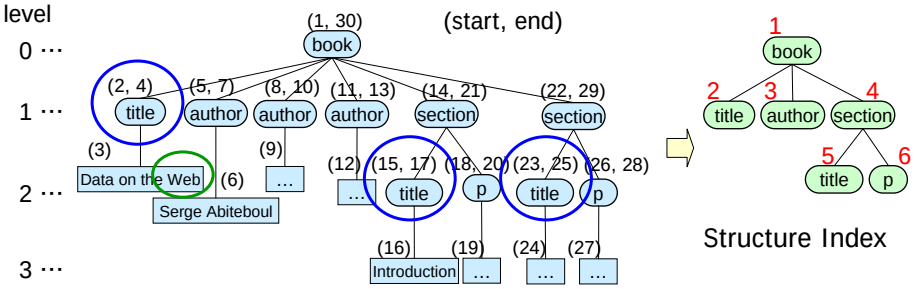
系列索引の長所と短所

- 長所
  - 枝分れ問合せに結合操作なしで答えることができる
  - 構造と内容を同じように扱う
  - “*” や “//” を扱うことができる
- 短所
  - 部分系列マッチと構造マッチが必ずしも一致しない場合がある
    - 兄弟ノードの順序

全文索引

- XMLに含まれるテキストの全文検索もサポート
- 研究事例
  - Integration [24]
    - 構造索引と転置リストを統合
  - XICS [25]
    - B+木を用いてテキスト保持やキー順序に工夫

Integration



要素ノード

要素名 → title

<docid, start, end, level, indexid>

<1, 2, 4, 1, 2>
<1, 15, 17, 2, 5>
<1, 23, 25, 2, 5>

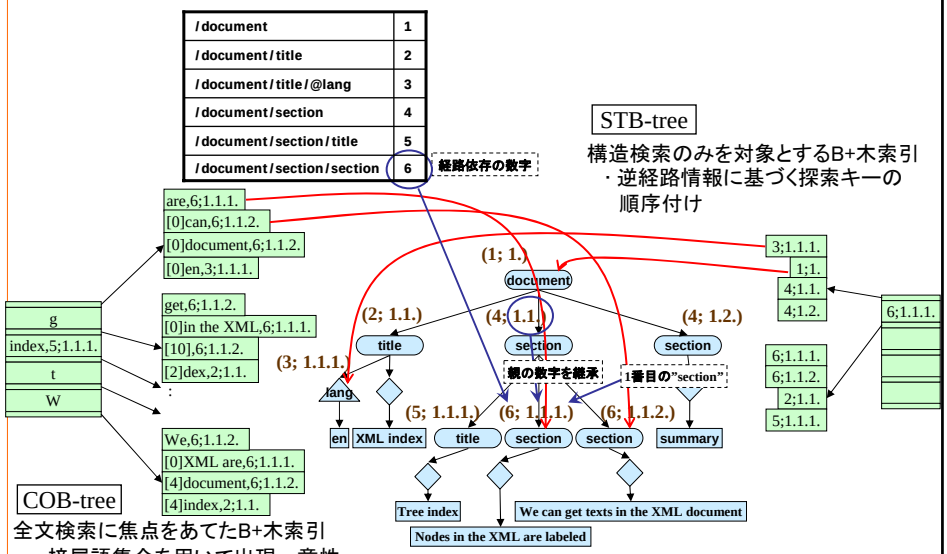
キーワードの出現

キーワード → “Web”

<docid, start, level, indexid>

<1, 3, 2, 2>

XICS



STB-tree

構造検索のみを対象とするB+木索引
・逆経路情報に基づく探索キーの順序付け

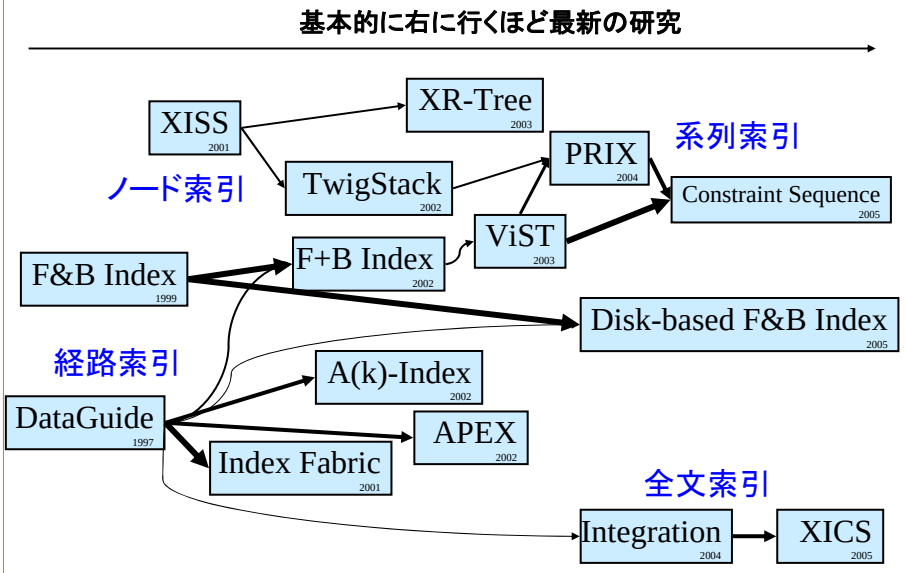
COB-tree

全文検索に焦点をあてたB+木索引
・接尾語集合を用いて出現一意性を考慮したテキスト保持
・差分情報を利用した索引圧縮

ノード識別子が付与されたXML

・経路情報を含んだノード識別子

XML索引付け研究マップ



まとめ

- XMLデータベース技術に関する研究動向
  - RDBへのマッピング
    - 構造写像アプローチ, モデル写像アプローチ
  - ノードラベリング手法
    - 範囲, 接頭辞, k分木, 素数
  - 索引付け
    - ノード索引, 経路索引, 系列索引, 全文索引
- 研究課題
  - 経路索引の問合せ補間問題と問合せ最適化
  - 大規模XMLデータに対するアクセス制御
  - ディスクへの格納まで考えた索引
  - 全文検索索引
    - XMLをデータ的に扱う際には1文字単位で検索できるべき

参考文献 (1/4)

[1] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton, “Relational databases for querying XML documents: Limitations and opportunities,” In *VLDB*, pages 302-314, September 1999.

[2] P. Bohannon, J. Freire, P. Roy, and J. Simeon, “From XML schema to relations: A cost-based approach to XML storage” In *ICDE*, page 64, 26 February - 1 March 2002.

[3] D. Florescu and D. Kossmann, “Storing and querying XML data using an RDMBS,” IEEE Data Engineering Bulletin, vol.22, no.3, pages 27-34, 1999.

[4] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura, “XRel: A path-based approach to storage and retrieval of XML documents using relational databases,” ACM Transactions on Internet Technology, vol.1, no.1, pages 110-141, 2001.

[5] T. Grust, “Accelerating XPath location steps,” In *ACM SIGMOD*, pages 109-120, June 2002.

[6] E. Cohen, H. Kaplan, and T. Milo, “Labeling Dynamic XML Trees,” In *PODS*, pages 271-281, June 2002.

参考文献 (2/4)

[7] I. Tatarinov, S. Viglas, K. S. Beyer, J. Shanmugasundaram, E. J. Shekita, and C. Zhang, “Storing and querying ordered XML using a relational database system,” In *ACM SIGMOD*, pages 204-215, June 2002.

[8] Y. K. Lee, S. Yoo, K. Yoon, and P. B. Berra, “Index structures for structured documents,” In *ICDL*, pages 91-99, March 1996.

[9] X. Wu, M. L. Lee, and W. Hsu, “A prime number labeling scheme for dynamic ordered XML trees,” In *ICDE*, pages 66-78, March 2004.

[10] S. A. Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu, “Structural joins: A primitive for efficient XML query pattern matching,”In *ICDE*, page 141, 26 February - 1 March 2002.

[11] Q. Li and B. Moon, “Indexing and querying XML data for regular path expressions,” In *VLDB*, pages 361-370, September 2001.

[12] H. Jiang, H. Lu, W. Wang, and B. C. Ooi, “XR-Tree: Indexing XML data for efficient structural joins,” In *ICDE*, pages 253-264, March 2003.

[13] N. Bruno, D. Srivastava, and N. Koudas, “Holistic twig joins: optimal XML pattern matching,” In *ACM SIGMOD*, pages 310-321, June 2002.

参考文献 (3/4)

[14] R. Goldman and J. Widom, “Dataguides: Enabling query formulation and optimization in semistrucutred databases,” In *VLDB*, pages 436-445, August 1997.

[15] B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon, “A fast index for semistructured data,” In *VLDB*, pages 341-350, September 2001.

[16] R. Kaushik, P. Shenoy, P. Bohannon, E. Gudes, “Exploiting local similarity for indexing paths in graph-structured data,” In *ICDE*, pages 129-140, 26 February - 1 March 2002.

[17] C. Chung, J. Min, and S. Kyuseok, “APEX: An adaptive path index for XML data,” In *ACM SIGMOD*, pages 121-132, June 2002.

[18] S. Abiteboul, P. Buneman, and D. Suciu, “Data on the web: from relations to semistructured data and XML,” Morgan Kaufmann Publishers, Los Altos, 1999.

[19] R. Kaushik, P. Bohannon, J. Naughton, and H. Korth, “Covering indexes for branching path queries,” In *ACM SIGMOD*, pages 133?144, June 2002.

参考文献 (4/4)

[20] W. Wang, H. Wang, H. Lu, H. Jiang, X. Lin, J. Li, “Efficient processing of XML path queries using the disk-based F&B index,” In *VLDB*, pages 145-156, 30 August - 2 September 2005.

[21] H. Wang, S. Park, W. Fan, and P. S. Yu, “ViST: A dynamic index method for querying XML data by tree structures,” In *ACM SIGMOD*, pages 110-121, June 2003.

[22] P. R. Raw and B. Moon, “PRIX: Indexing and querying XML using prüfer sequences,” In *ICDE*, pages 288-300, March 2004.

[23] H. Wang and X. Meng, “On the sequencing of tree structures for XML indexing,” In *ICDE*, pages 372-383, April 2005.

[24] R. Kaushik, R. Krishnamurthy, J. F. Naughton, and R. Ramakrishnan, “On the integration of structure indexes and inverted lists,” In *ACM SIGMOD*, pages 779-790, June 2004.

[25] T. Shimizu and M. Yoshikawa, “Full-text and structural XML indexing on B+ Tree,” In *DEXA*, pages 451-460, August 2005.