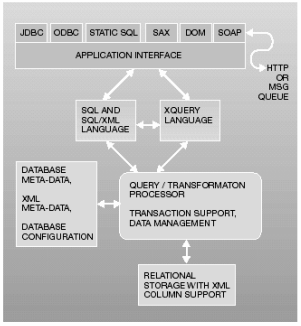
XMLデータベース

吉川正俊
（京都大学）

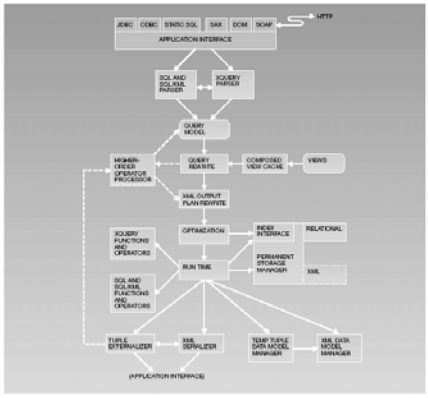
関係問合せとXML問合せをともにサポートするDBMSの構造

Figure 1 XML/relational database manager



XML programming with SQL/XML and XQuery by J. E. Funderburk, S. Malaika, B. Reinwald, IBM Systems Journal, Volume 41, Number 4, pp. 642-665 (2002)

Figure 4 Integrated database architecture

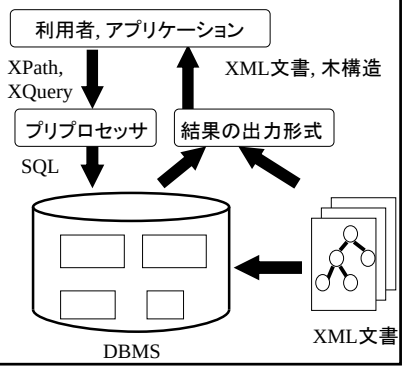


XML programming with SQL/XML and XQuery by J. E. Funderburk, S. Malaika, B. Reinwald, IBM Systems Journal, Volume 41, Number 4, pp. 642-665 (2002)

XML文書のRDBへの格納

XML文書のRDBへの格納

考慮すべき点
- XMLスキーマの有無
- XPath, XQueryからSQLへの問合せ変換
- 問合せの処理効率
- データ容量
- 更新に対する頑健性



XML文書を分解し，既存DBMSのスキーマに写像した上で検索する方法

- ◆構造写像アプローチ
  - XML文書の論理構造（DTD）を表現するデータベーススキーマを定義
- ◆モデル写像アプローチ
  - XMLデータモデルの構成要素（ノード，枝，経路など）を表現するデータベーススキーマを定義

×

- ◆RDB
- ◆OODB

XML文書を分解しデータベーススキーマに写像する方法 --- 関係データベースの場合

```
<emp>
        <name>Joe Doe</name>
        <age>30</age>
</emp>
```

構造写像アプローチ

モデル写像アプローチ

emp

name	age
Joe Doe	30

少数の文書構造（DTD）に従う大量のXML文書

node

id	parent	name	value
1	-	emp	
2	1	name	Joe Doe
3	1	age	30

DTDが不明または頻繁に更新される

考慮すべき点

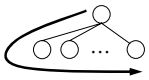
- XPath, XQueryからSQLへの問合せ変換
- 問合せの処理効率
- データ容量
- 更新に対する頑健性

木のノードラベリング手法

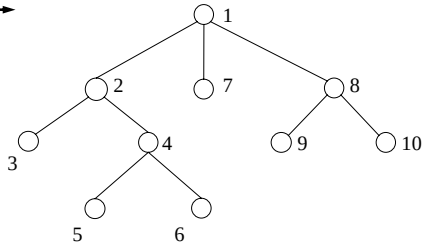
木のノードラベリング手法

- 設計目標
  - XPathの軸（特に親子，先祖/子孫関係）の高速計算
  - 短いラベル長
  - 更新に対する頑健性
  - XMLスキーマの考慮
- ラベルの種類
  - Dietzの番号付け
  - 接頭辞ラベル
  - k分木への埋め込み

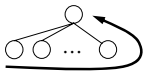
preorder



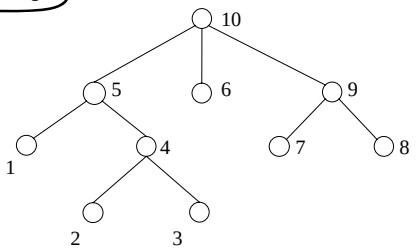
開始タグの出現順



postorder

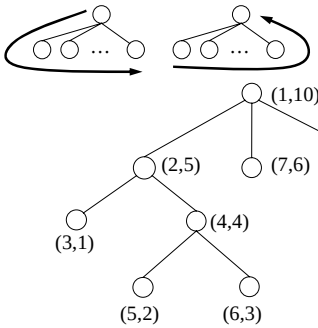


終了タグの出現順



Dietzの番号付けスキーム

◆(preorder, postorder)



ノード a はノード d の先祖
⇔
a.preorder < d.preorder
かつ
a.postorder > d.postorder

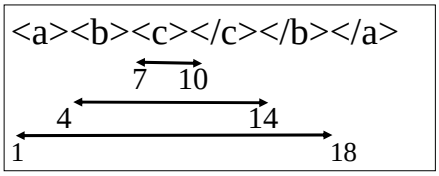
リージョン (region)

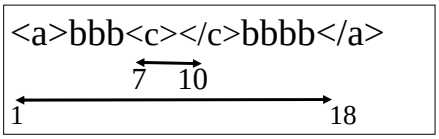
◆開始タグの先頭からのバイト数
と
終了タグの先頭からのバイト数
の対

◆それぞれの順序関係は，preorder, postorder と同じ

◆親子関係の判定は困難なので，レベル（根ノードからの距離）も合わせて記録することが多い

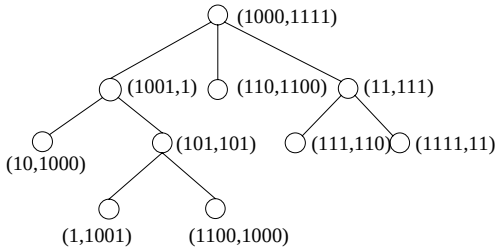
リージョン





VLEIコード [小林ら DBWS2003]

v0{0|1}* < v < v1{0|1}*
このコードによりpreorder, postorderの対で表現
ビット長を長くすることによりノード挿入に対処

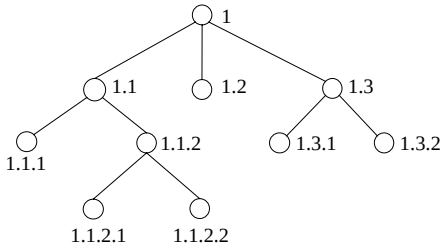


接頭辞ラベル

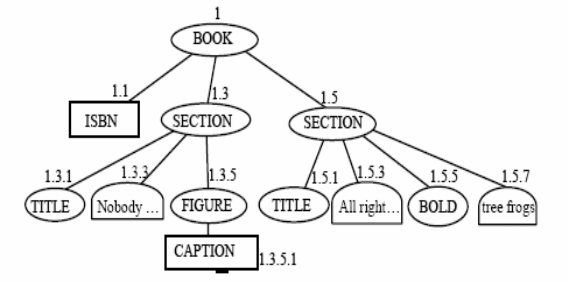
◆親のラベルは子のラベルの接頭辞

Dewey order

◆図書館で広く使われている分類法



ORDPATHノードラベル



ORDPATHノードラベル

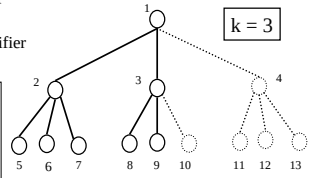
- 最初は奇数のみを用いる
- 挿入が生じた場合
  - 偶数をcaretとして用いる（偶数はレベルとして数えない）
  - 最後の数字は必ず奇数とする
  - 例： 3.5.5 と 3.5.7の間に挿入する場合は，3.5.6.1, 3.5.6.3, 3.5.6.5, … などを用いる

k分木への埋め込み

Unique identifier

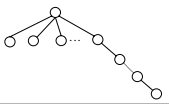
- UID (*Y.K.Lee, 96*)
  - each internal node supposedly has the same children
    - » maximal fan-out (= k)
  - enable computing the parent node
    from a child node's identifier

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



UIDの問題点

- enumerate virtual nodes
- identifiers increasing rate: max_fanout $^{tree\_height}$
  ⇒ 最悪の場合



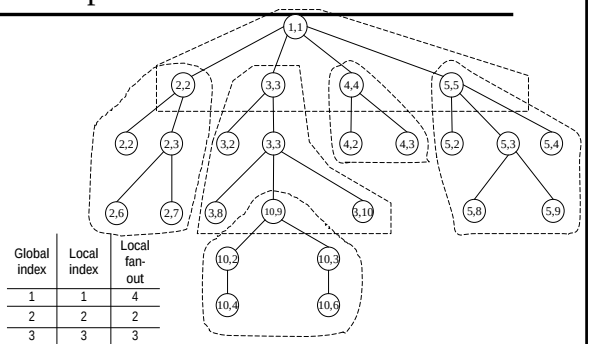
- ⇒ 実際に，以下の小さいXML文書でも表現できない．

Data set	size	# element & attributes	max fan-out	height
Data set I (data on *20 persons*)	7.6KB	201	11	11
Love's Labor's Lost (*plays grouped, collection added*)	210KB	5057	434	7

再帰的識別番号 (recursive UID)：目的と設計原理

- 目的
  - overcome the node number limitation
  - while preserve order determination property
- 設計原理
  - Nodes subsets managed in two levels
  - Global level:
    - » these subsets identifiers are recognized.
  - Local level:
    - » nodes in each subset are managed
  - Global parameters: used in both levels.
    - » small enough
    - » be comfortably loaded into the main memory

Example of 2-level UID



Global index	Local index	Local fan-out
1	1	4
2	2	2
3	3	3
4	4	2
5	5	3
10	9	2

Parent - child determination

- Suppose: $\kappa = 4$ and K is given

1. From (2,7,0) compute the parent node
   - look at line 2 of K: local fan-out = 2
   - local index = $\lfloor (7-2) / 2 + 1 \rfloor = 3 > 1$

   $\Rightarrow$ (2, 3, 0)
2. From (10, 9, 1) - a root
   - upper UID local area's index = $\lfloor (10-2) / 4 + 1 \rfloor = 3$
   - look at line 3 of K: local fan-out = 3
   - local index = $\lfloor (9-2) / 3 + 1 \rfloor = 3 > 1$
   - $\Rightarrow$ (3, 3, 0)

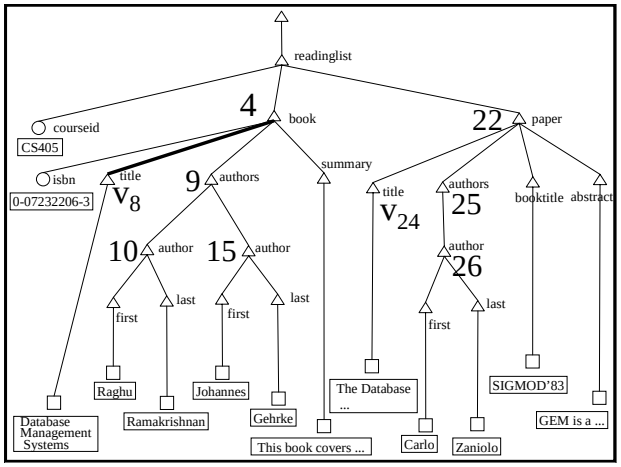
モデル写像アプローチ

モデルアプローチに基づくRDBへの格納

- XMLスキーマの存在とは独立
  - どのような整形式 XML文書も格納可能
- 問題
  - どのような関係データベーススキーマを設計するか？
    - » 枝アプローチ
    - » 経路アプローチ
  - XML問合せからSQLへの変換をどのように行うか？

枝アプローチ

- XML木の枝の情報を格納するために関係を一つ設け，各枝を関係の組として格納する.
- 各ノードに唯一の識別子が付与されていることを前提とする



枝アプローチの関係データベーススキーマ

Edge

source	ordinal	name	target
...	...	...	...
4	2	title	v_8
...	...	...	...
9	1	author	10
9	2	author	15
...	...	...	...
22	1	title	v_{24}
...	...	...	...
25	1	author	26
...	...	...	...

Value

vid	value
...	...
v_8	Database Management Systems
...	...
v_{24}	The Database Language GEM
...	...

枝アプローチの問合せ変換

- 経路式の長さに比例した結合が必要

```
XPath:  /readinglist/book/title
SQL:    SELECT e3.target
        FROM Edge e1, Edge e2, Edge e3
        WHERE e1.name = "readinglist"
        AND e2.name = "book"
        AND e3.name = "title"
        AND e1.target = e2.source
        AND e2.target = e3.source
```

- XPathの // を処理するためには，再帰結合が必要

経路アプローチ

経路アプローチ

XML文書内の経路を文字列として
関係データベースに格納

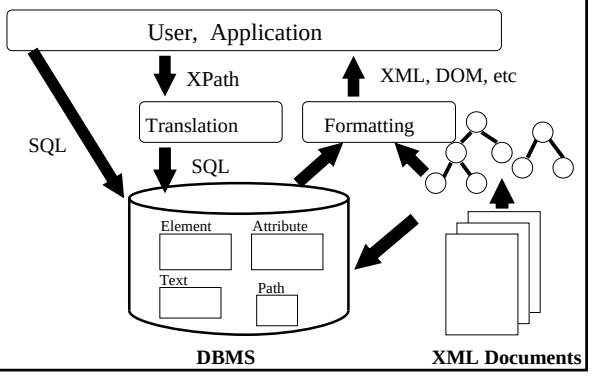
- DTDの情報は利用しない
  - どのような整形式 XML文書も格納可能
- XPathで表される経路に関する複雑な条件
  - SQLの文字列パターン照合と整数の比較に変換
    - » 複雑な結合処理は不要
  - 通常の DBMS で提供されている索引を利用可（B+木，R木）

経路アプローチ

- XML木の経路を格納単位とする
- ノードのラベル付け
  - XML木のトポロジーの保存
  - 問合せ，更新処理の高速化
- XPathで表される経路に関する複雑な条件
  - SQLの文字列パターン照合と整数の比較に変換
    - » 複雑な結合処理は不要
  - 通常の DBMS で提供されている索引を利用可（B+木，R木）

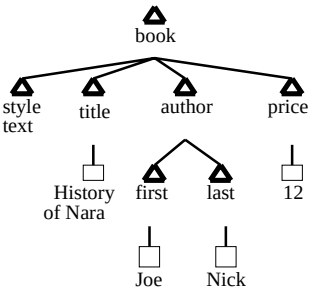
XRel

Storing XML Documents in RDB



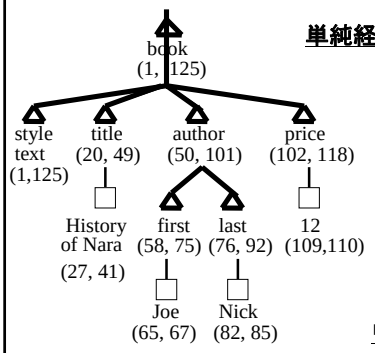
XML文書

```
<book style='text'>
  <title>History of Nara</title>
  <author>
    <first>Joe</first>
    <last>Nick</last>
  </author>
  <price>12</price>
</book>
```



Element △ Attribute △ Text □

XML文書の分解，表現の単位



単純経路式

/book
/book/@style
/book/title
/book/author
/book/author/first
/book/author/last
/book/price

リージョン : (Start, End)

XML文書の関係への格納方法

Path

pathname	pathID
/book	1
/book/title	2
/book/author	3
…	.

Element

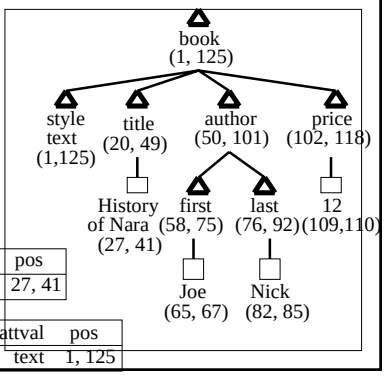
docID	pathID	pos
#1	2	20, 49

Text

docID	pathID	value	pos
#1	2	History	27, 41

Attribute

docID	pathID	attname	attval	pos
#1	1	style	text	1, 125



検索方法~問合せの書き換え~

/book/title
//last
//author[first='Joe']

Path

pathname	pathID
/book	1
/book/title	2
/book/author	3
…	.

- XPathの基本は / または // と要素型をつないだ列
- データベースでは XML 文書を単なる文字列として格納
- 書き換えについては, SQL のもつ文字列照合を行う述語を利用

問合せ変換例(1) ~単純経路式~

/book/title

SELECT e.docID, e.position
FROM Element e, Path p
WHERE e.pathID = p.pathID
AND p.pathname = "/book/title"

Element

docID	pathID	pos
#1	1	1, 125
#1	2	20, 49
#1	3	50, 101
.	.	…

Path

pathname	pathID
/book	1
/book/title	2
/book/author	3
…	.

問合せ変換例(2) ~あいまい経路式~

//last

SELECT e.docID, e.position
FROM Element e, Path p
WHERE e.pathID = p.pathID
AND p.pathname like “%/last”

⇧
文字列照合を行う述語

Element

docID	pathID	pos
#1	1	1, 125
#1	2	20, 49
…	…	…
#1	5	76, 92

Path

pathname	pathID
/book	1
/book/title	2
…	.
/book/author/last	5

問合せ変換例(3)
~制約条件付きのあいまい経路式~

//author[first=‘Joe’]

SELECT e.docID, e.position
FROM Element e, Text t,
Path p1, Path p2
WHERE p1.pathname
like “%/author”
AND p2.pathname
like “%/author/first”
AND t.value = “Joe”
AND t.pathID = p2.pathID
AND e.pathID = p1.pathID
AND e.docID = t.docID
AND e.pos.contain(t.pos)

Element

docID	pathID	pos
#1	1	1, 125
#1	2	20, 49
#1	3	50, 101

Text

docID	pathID	value	pos
#1	4	Joe	65, 67

Path

pathname	pathID
/book/author	3
/book/author/first	4

Microsoft SQL Server 2005

Primary XML index (infosettab)

主キー

ORDPATH	TAG	NODE_ TYPE	VALUE	PATH_ ID
1	1 (BOOK)	1 (Element)	Null	#1
1.1	2 (ISBN)	2 (Attribute)	'1-55860-438-3'	#2#1
1.3	3 (SECTION)	1 (Element)	Null	#3#1
1.3.1	4 (TITLE)	1 (Element)	'Bad Bugs'	#4#3#1
1.3.3	10 (TEXT)	4 (Value)	'Nobody loves Bad bugs.'	#10#3#1
1.3.5	5 (FIGURE)	1 (Element)	Null	#5#3#1
1.3.5.1	6 (CAPTION)	2 (Attribute)	'Sample bug'	#6#3#1
1.5	3 (SECTION)	1 (Element)	Null	#3#1
1.5.1	4 (TITLE)	1 (Element)	'Tree frogs'	#4#3#1
1.5.3	10 (TEXT)	4 (Value)	'All right-thinking people'	#10#3#1
1.5.5	7 (BOLD)	1 (Element)	'love '	#7#3#1
1.5.7	10 (TEXT)	4 (Value)	'tree frogs'	#10#3#1

問合せの変換

/BOOK[@ISBN = “1-55860-438-3”]/SECTION

⇩

引数のノードを根とする部分文書を返す

```
SELECT SerializeXML (N2.ID, N2.ORDPATH)
FROM infosettab N1
        JOIN infosettab N2 ON (N1.ID = N2.ID)
WHERE N1.PATH_ID = PATH_ID(/BOOK/@ISBN)
AND N1.VALUE = '1-55860-438-3'
AND N2.PATH_ID = PATH_ID(/BOOK/SECTION)
AND Parent (N1.ORDPATH) = Parent (N2.ORDPATH)
```

引数のORDPATHの親ノードのORDPATHを返す

引数の経路の経路IDを返す