XML Databases: Modelling and Multidimensional Indexing

Rudolf Bayer

Sept. 3, 2001

XML Document as an XML Tree

<Paper>		
<Title>	MISTRAL	</Title>
<Authors>		
<FN>	Rudolf	</FN>
<LN>	Bayer	</LN>
<Affiliation>	Techn. Univ.	</Affiliation>
</Authors>		
<Authors>		
<FN>	Volker	</FN>
<LN>	Markl	</LN>
<Affiliation>	FORWISS	</Affiliation>
</Authors>		
<Keywords>	UB-tree	</Keywords>
<Keywords>	POT	</Keywords>
<Keywords>	Region	</Keywords>
<Abstract>	a piece of text	</Abstract>
<Text>	more text	</Text>
<BibRec>		
<Authors>		
<LN>	Fenk	</LN>
</Authors>		
<Authors>		
<LN>	Ramsak	</LN>
</Authors>		
<Title>	DW Queries	</Title>
</BibRec>		
</Paper>		

XML Basics

1. Every XML document is an ordered tree with labeled branches, many potential representations
2. The structure of the tree is described by a DTD
3. Parsing a document is trivial w.r.t. to wellformedness or conformance with a given DTD

DTD for the Document <Paper>

```
<DOCTYPE Paper [
        <!ELEMENT Paper (Title, Authors*, Keyword*,
                Abstract?, Text, BibRec*)>
        <!ELEMENT Title (#PCDATA)>
        <!ELEMENT Authors (FN?, LN, Affiliation?)>
        <!ELEMENT FN (#PCDATA)>
        <!ELEMENT LN (#PCDATA)>
        <!ELEMENT Affiliation (#PCDATA)>
        <!ELEMENT Keywords (#PCDATA)>
        <!ELEMENT Abstract (#PCDATA)>
        <!ELEMENT Text (#PCDATA)>
        <!ELEMENT BibRec (Author*, Title)>
        ]
```

Operations with Trees

- **store** in rel. DBMS
- **find** documents based on path search predicates
- **retrieve** parts of found documents
- **insert** and delete documents in a DB
- **modify** documents, i.e. delete and insert subtrees of a document

Note: in RDBMS we deal with complete tuples, in XML we deal with partial documents

Path Notation and Identical Paths

<Paper>
    <Authors> [1]
        <LN> Bayer
<Paper>
    <Authors> [2]
        <LN> Markl

Distinguish paths by repetition numbers, in document fixed by order of the text

→ Paths become unique within a document, use them as attributes in a universal relation XML-Rel

Universal Relation XML-Rel:

with Attribute Paths APi

Did AP1 AP2 AP3 ... Apk

Use path notation for attributes:

Paper/Authors[2]/LN Markl

Every document is exactly one tuple in XML-Rel with unbounded number of attributes

Document with

DTD	**and**	**Data Instance**		
Paper				
Title		MISTRAL		
Authors*				
FN		Rudolf	Volker	
LN		Bayer	Markl	
Affiliation		TU	FORWISS	
Keywords*		UB-tree	POT	Region
Abstract		a piece of text		
Text		more text		
BibRec*				
Authors*				
LN		Fenk	Ramsak	
Title		DW Queries		

DTD with Numbering per Level and Repetition Numbers

Paper			
1 Title	MISTRAL		
2 Authors*	[1]	[2]	
1 FN	Rudolf	Volker	
2 LN	Bayer	Markl	
3 Affiliation	TU	FORWISS	
3 Keywords*	[1]	[2]	[3]
	UB-tree	POT	Region
4 Abstract	a piece of text		
5 Text	more text		
6 BibRec*	[1]		
1 Authors*	[1]	[2]	
2 LN	Fenk	Ramsak	
2 Title	DW Queries		

Surrogate-Patterns and Path-Expressions: 1↔1

Surr-P	Path-Expressions	Examples of Paths
1	Title	
2.*	Authors*	Authors[1], Authors[2],…
2.*.1	Authors*/FN	Authors[1]/FN
2.*.2	Authors*/LN	Authors[1]/LN
2.*.3	Authors*/Affiliation	
3*	Keywords*	
4	Abstract	
5	Text	
6.*	BibRec*	
6.*.1.*	BibRec*/Authors*	BibRec[1]/Authors[1], …
6.*.2	BibRec*/Title	BibRec[1]/Title, …

Surrogate patterns reflect ordering!

Some Paths, Values Surrogates

Some Paths,	Values	Surrogates
Title	MISTRAL	1
Authors[1]/FN	Rudolf	2[1]1
Authors[1]/LN	Bayer	2[1]2
Authors[1]/Affiliation	Techn. Univ.	2[1]3
Authors[2]/FN	Volker	2[2]1
...		
Keywords[3]	Region	3[3]
Abstract	a piece of text	4
...		
BibRec[1]/Authors[2]/LN	Ramsak	6[1]1[2]1
BibRec[1]/Title	DW Queries	6[1]2

Note: from surrogates and values we can reconstruct the original document with the help of an additional surrogate to tag mapping, which is stored in an additional table

Mapping for XML-Rel:

XML-Rel with Attribute Paths APi

Did AP1 AP2 AP3 ... APk

is mapped to

XML-Quad

Did Attr-Path Value Surrogate

with candidate keys (Did, Attr-Path) or (Did, Surrrogate)

Decompose XML Quad into two relations

Type-Dim to replace Attribute-Paths by Surrogates

Attr-Path **Surrogate** **Type**

and relation

XML-Ind for XML Index

Did **Surrogate** **Value**

define view XML-Quad **as**
select Did, Surrogate, Attr-Path, Value
from Type-Dim T, XML-Ind X
where T.Surrogate = X.Surrogate

Relation XML-Ind for decomposed XML-Quad

Did	Surrogate	Value
7	1	MISTRAL
7	2[1]1	Rudolf
...		
	2[2]1	Volker
	2[2]2	Markl
	2[2]3	FORWISS
	3[1]	UB-Tree
	3[2]	POT
	3[3]	Region
	4	a piece of text
	5	more text
	6[1]1[1]2	Fenk
	6[1]1[2]2	Ramsak
7	6[1]2	DW Queries

Observations

- **lexicographic ordering of surrogates properly represents document order**
- **subtrees, like for Author[2] correspond to intervals of surrogates**
- **very compact representation of XML documents**
- **3-dimensional table**
- **most queries have 2 restrictions**

User Queries and DML Statements

select Paper/Title **from** XML-Rel
where Paper/Authors[1]/LN = ‘Bayer’

Note: if the repetition number of the author is not known, we write

where Paper/Authors[$i]/LN = ‘Bayer’

in order to instantiate the variable [$i] properly

Rewriting the XML-Rel Query

select Paper/Title
from XML-Rel
where Paper/Authors[1]/LN = 'Bayer'

is rewritten into 2 queries:

select Did **into** Did-Set
from XML-Quad
where Attr-Path = 'Paper/Authors[1]/LN' **and**
Value = 'Bayer' ;

select Value
from XML-Quad
where Attr-Path = 'Paper/Title' **and**
Did **in** Did-Set

Another Rewriting of the XML-Rel Query

select Paper/Title
from XML-Rel
where Paper/Authors[1]/LN = 'Bayer'

is rewritten into a single join-query:

select Q2.value
from XML-Quad Q1, XML-Quad Q2
where (Q1.Attr-Path = 'Paper/Authors[1]/LN' **and**
Q1.Value = 'Bayer'**)** **and**
(Q2.Did = Q1.Did **and**
Q2.Attr-Path = 'Paper/Title'**)**

General Join Rewriting of XML-Rel to XML-Quad

select APi, APj

from XML-Rel **where** APk = c1 **and** APl = c2

Is rewritten into:

select T3.Value, T4.Value
from XML-Quad T1, XML-Quad T2**,** XML-Quad T3, XML-Quad T4
where T1.Attr-Path = 'APk' **and** T1.Value = c1 **and**
T2.Attr-Path = 'APl' **and** T2.Value = c2 **and**
T3.Attr-Path = 'APi' **and**
T4.Attr-Path = 'APj' **and**
and T1.Did **=** T2.Did
and T2.Did **=** T3.Did
and T3.Did **=** T4.Did

Transformation of previous Query

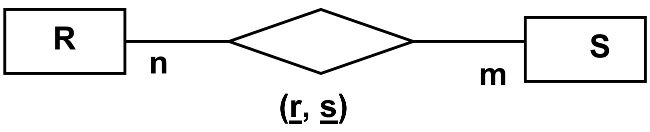
select T3.Value, T4.Value
from XML-Quad T1, XML-Quad T2**,** XML-Quad T3, XML-Quad T4
where T1.Attr-Path = 'APk' **and** T1.Value = c1 **and**
T2.Attr-Path = 'APl' **and** T2.Value = c2 **and**

T1.Did **=** T2.Did

T3.Attr-Path = 'APi' **and** T3.Did **=** T1.Did
T4.Attr-Path = 'APj' **and** T4.Did **=** T1.Did

UB-Trees: Multidimensional Indexing

- geographic databases (GIS)
- Data-Warehousing: Star Schema
- all relational databases with n:m relationships



- mobile, location based applications
- XML

Typical Queries on XML-Quad

Did Attr-Path Value Surrogate

find documents written by Markl

Leads to two restrictions:

Attr-Path = ‚Author[$i]/LN‘ **and** Value = ‚Markl‘

Retrieve Title of found documents with Did = k

Again two restrictions:

Attr-Path = ‚Title‘ **and** Did = k

Change spelling errror from ‚Rudolph‘ to ‚Rudolf‘ in Did 1274

Did = 1274 **and** Attr-Path = Author[1]/FN

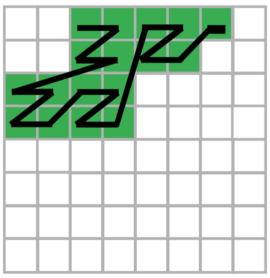
→ Suitable for mulitdimensional indexing!!

Basic Idea of UB-Tree

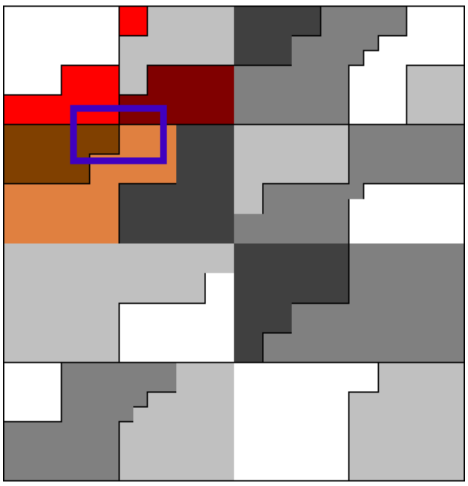
- linearize multidimensional space by space filling curve, e.g. Z-curve or Hilbert
- use Z-address as key to store objects in B-Tree

→ ***Response time for query is proportional to size of the answer!***

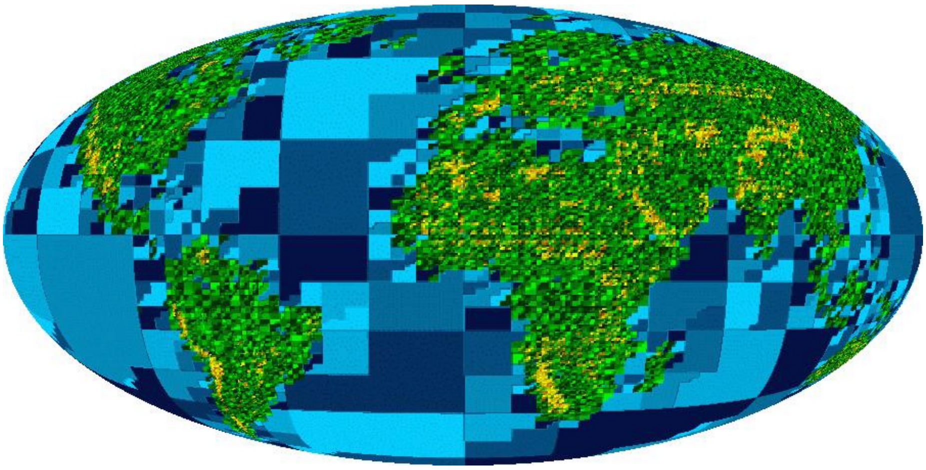
UB-Tree: Regions and Query-Box



Z-region
[0.1 : 1.1.1]



World as self balancing UB-Tree



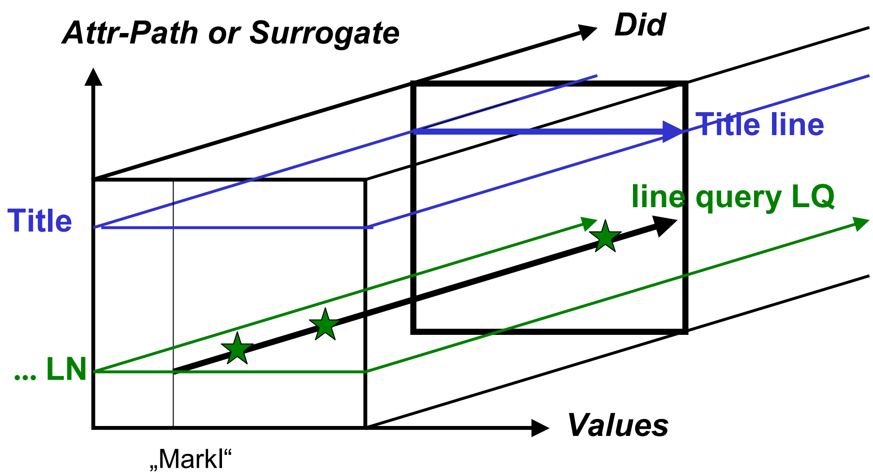
QB1: ***select*** Paper/Title ***where***
Paper/Authors[1]/LN = 'Markl'

Rewriting results in a 2-dimensinal restriction,

i.e. a line query LQ

where Attr-Path = ‚Paper/Authors[1]/LN'

and Values = ‚Markl'



Assumptions: for the analysis of queries

- 10^6 documents with 10^4 B each
- this results in a DB of 10 GB with
- 10^6 pages

→ ***Each dimension of a 3 dimensional cube spans about 100 pages, i.e. D = 100***

i.e. the number of pages „skewered“ by LQ

Algorithm and Complexity for QB1:

for each hit ***h*** on line query LQ there is a document ***d(h)***

finding all hits is ***O(D)***

d(h) corresponds to a plane slice ***s(h)***

intersecting ***O(D^2)*** pages

Title correspond to a line through ***s(h)***

therefore only ***O(D)***

pages must be fetched from ***s(h)***

Complexity per retrieved hit: ***O(D) ~ 1 sec/hit***

→ ***Response time = size of answer * 1 sec/hit***

Query QB2: *"Get Titles and Authors of papers, in which papers coauthored by Markl and Ramsak are cited"*

select D1/Paper/Title, D1/Paper/Authors*

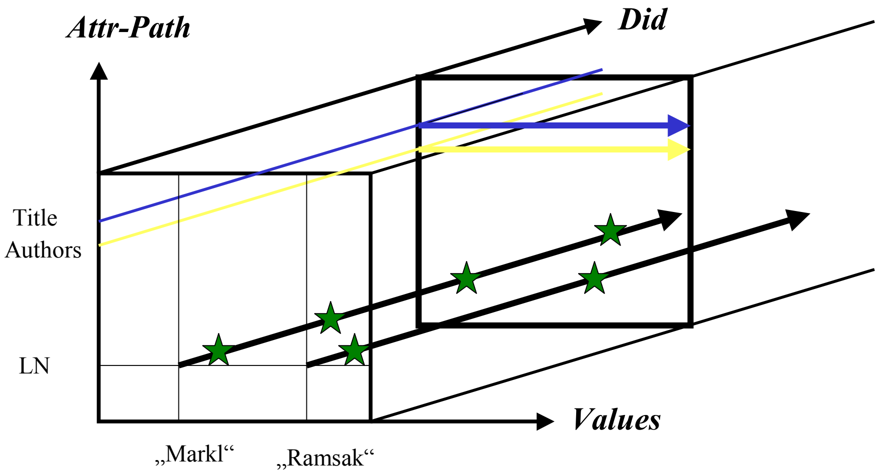
from Documents D1, Documents D2

where D1/Paper/BibRec[$i]/Authors[$j]/LN = 'Markl' ***and***

D2/Paper/BibRec[$k]/Authors[$m]/LN = 'Ramsak' ***and***

D1/Paper/BibRec[$i] = D2/Paper/BibRec[$k]

Note: *the last (join) condition is checked on the surrogates of D1...BibRec and D2...BibRec (note that several variables: Paper, $i and BibRec are instantiated by this) and the projection list is retrieved via a generalized Tetris algorithm*



Classification and Complexity of Queries

- **n** no restriction,
- **i** interval of breadth β ,
- **c** constant,
- D^3 = P = number of pages in DB

Assumption: P = 10^6 pages = 8 GB,

D = 100,
300 pages per second from disk,
time estimates in seconds,
restrictions 10% or less.

Restr	Query-Box	# of pages	Time	
n n n	universe	D^3	< 3000	
i n n	slice	βD^3 or D^2	< 300 or	30 for plane
i i n	pillar	$\beta_1 \beta_2 D^3$ or D	< 30 or	1 for line
i i i	box	$\beta_1 \beta_2 \beta_3 D^3$	< 3	
c n n	plane	D^2	< 30	
c i n	stripe	$\beta_1 D^2$	< 3	
c i i	rectangle	$\beta_1 \beta_2 D^2$	< 1	
c c n	line	D	< 1	
c c i	line interval	β_1 D	< 1	
c c c	point	const	< 1	