DBAG 2005

Some Exciting Research Problems in XML Databases

Theo Härder
www.haerder.de

Extensions and optimizations in the layer model

Is the layer model suitable for other data types?

Labeling of XML tree nodes

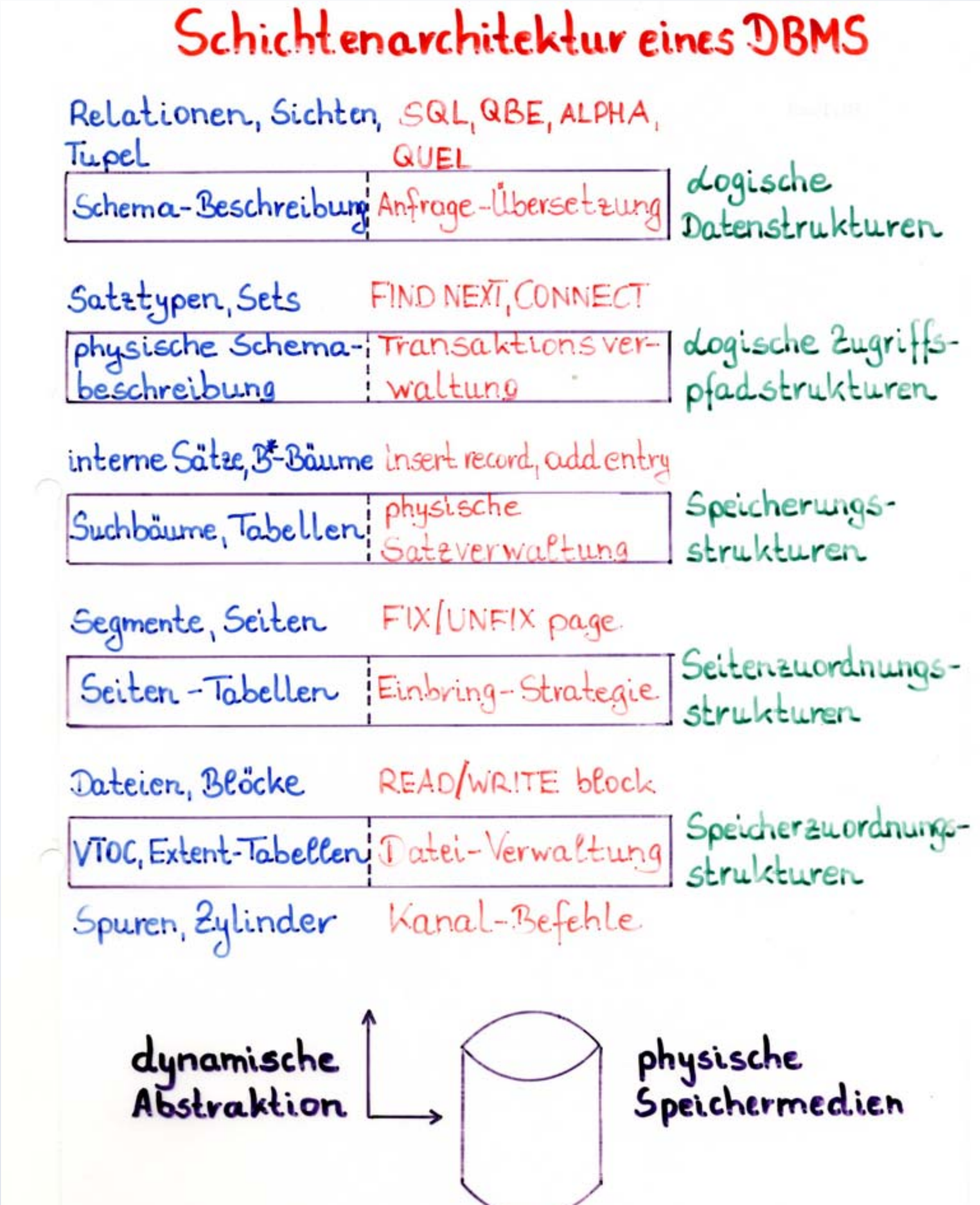
Which node services are required?

How to support fine-grained XML locking?

Architectural extensions for XDBMSs

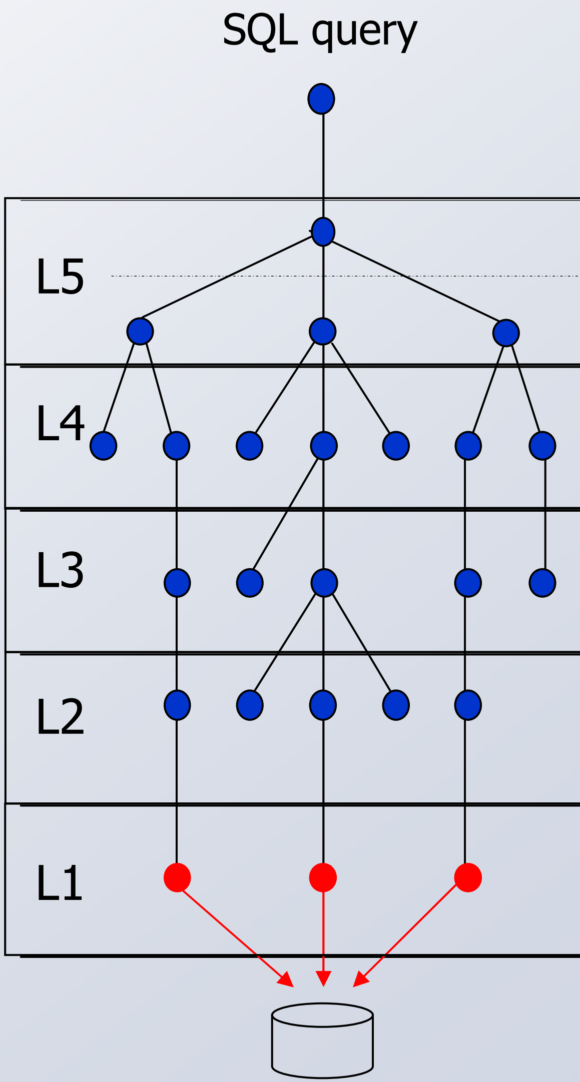
DBIS Datenbanken und Informationssysteme
Technische Universität Kaiserslautern

Datenbanksysteme in Büro, Technik und Wissenschaft, März 1985, Karlsruhe

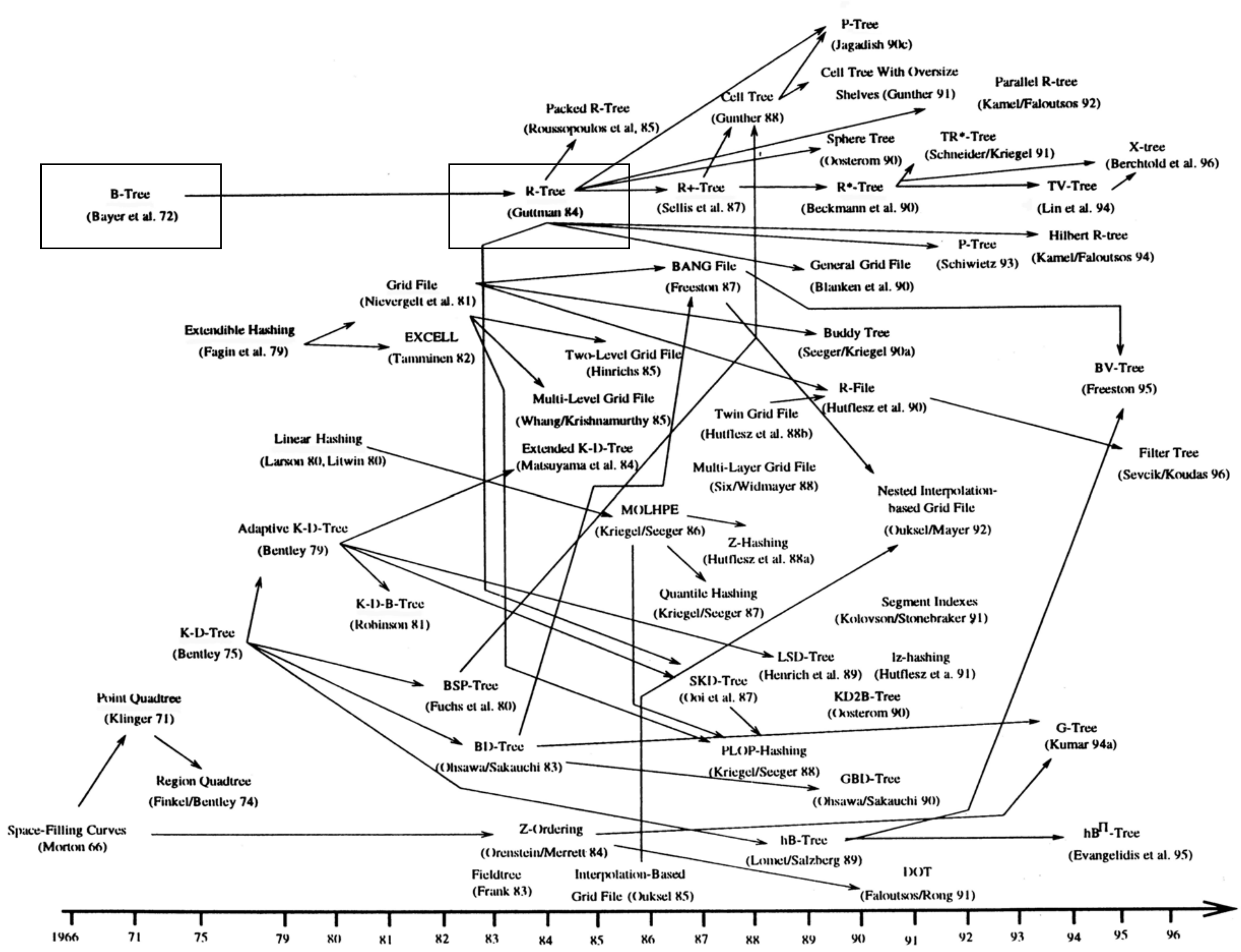


Extensions and Optimizations

- New storage types, LOBs
- Optimization by Moore's law
  - factor 10^4, page size from 2K to 8–32K
  - LRU + reference density (LRU-K)
- New access paths?
  - adequate and integrated access paths in addition to the ubiquitous B-tree?
  - most important performance improvement by fine-grained locking
- New algorithms
  - hash joins, arbitrary predicates, …
  - shared use of scans, reuse of results
  - adaptivity to resource unavailability
- Compilation and Optimization (SQL4)
  - cost-based optimizers (histograms)
  - dynamic QEPs
  - "gambling"



Genealogy of Access Paths



Is the Layer Model suitable for other data types?

- Layer model
  - designed for declarative and set-oriented evaluation of records
  - layer architecture is comparable to a set of "abstract machines" cooperating in a "use hierarchy"
  - hierarchical layers and information hiding guaranteed long-term system evolution
  - "self-*" properties require much more information flow across system layers

Important observation:
The invariants in database management determine the mapping steps of the supporting architecture

Reuse and adaptation of
- storage system
- access system
- data system?

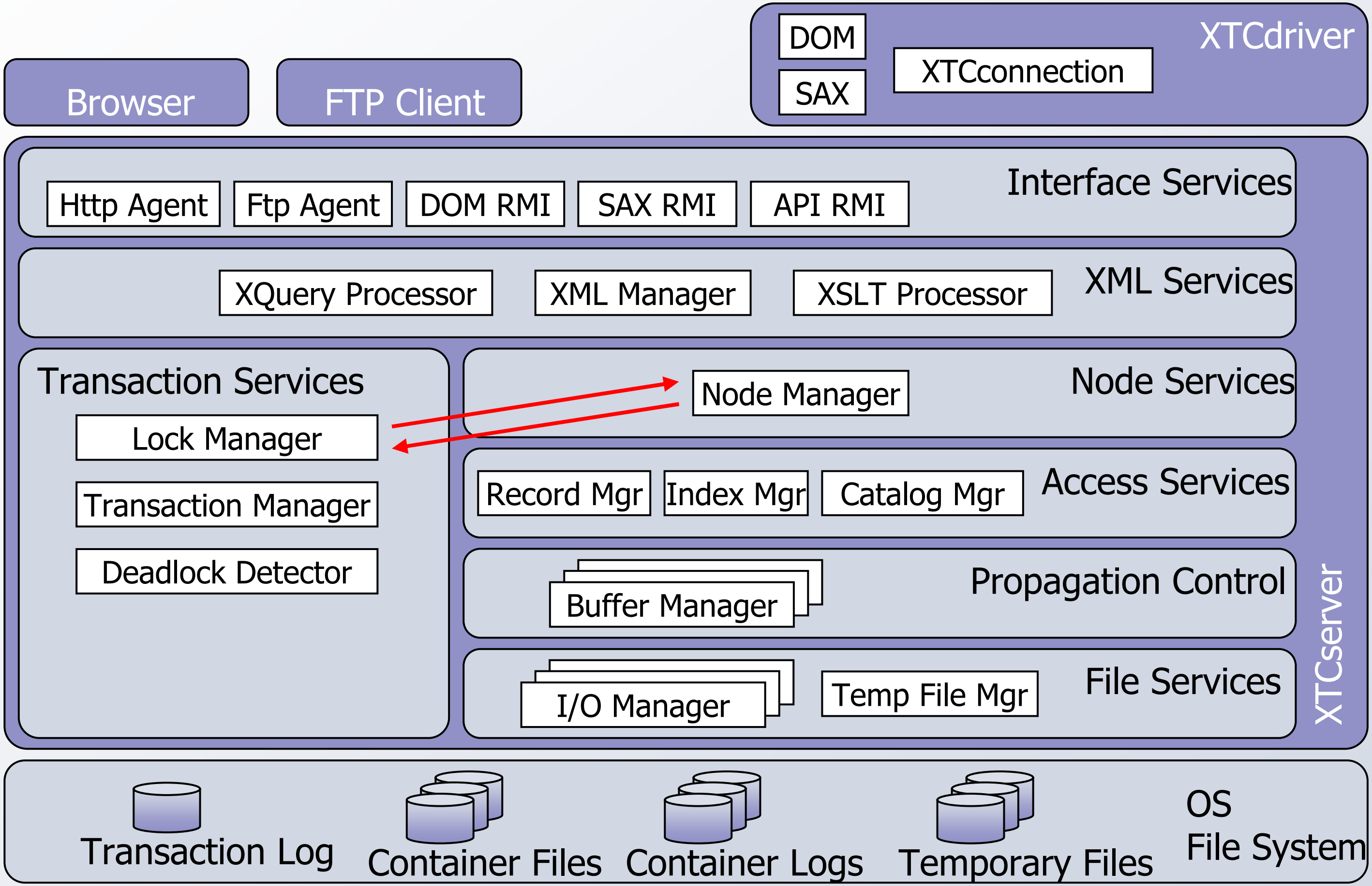
Example of an XML Document

```
<bib>
        <book year=“1994“ id=“1“>
                <title>TCP/IP Illustrated</title>
                <author>
                        <last>Stevens</last>
                        <first>W.</first>
                </author>
                <price>65.95</price>
        </book>
        <book year=“2000“ id=“2“>
                <title>Data on the Web</title>
                <author>
                        <last>Abiteboul</last>
                        <first>Serge</first>
                </author>
                <author>
                        <last>Buneman</last>
                        <first>Peter</first>
                </author>
                <author>
                        <last>Suciu</last>
                        <first>Dan</first>
                </author>
                <price>39.95</price>
        </book>
        <book year=“1999“ id=“3“>
                <title>The Economics of . . . </title>
                <editor>
                        <last>Gerbarg</last>
                        <first>Darcy</first>
                        <affiliation>CITI</affiliation>
                </editor>
                <price>129.95</price>
        </book>
</bib>
```

XML documents are to be stored in databases

- conceptual representation: trees with nodes and edges
- document order must be preserved / recoverable: node order matters!
- **LOBs** don't enable fine-granular management, no content-based search and no multi-user operation
- **mapping onto relational tables?**
  - many solutions: "shredding"
  - XML query language (e.g., XQuery, XPath, DOM, SAX) must be mapped to SQL
  - use of the SQL optimizer!
  - but: concurrency control (locking) very cumbersome, because a document is distributed over n tables

A Native XDBMS Architecture



- XTC – architectural overview
  - reuse of the layer model is possible, but needs substantial adjustments and, in particular, new functionality in the higher layers

Native XDBMSs

xtensions &
ptimizations

daptation to
L documents?

Labeling of
L tree nodes

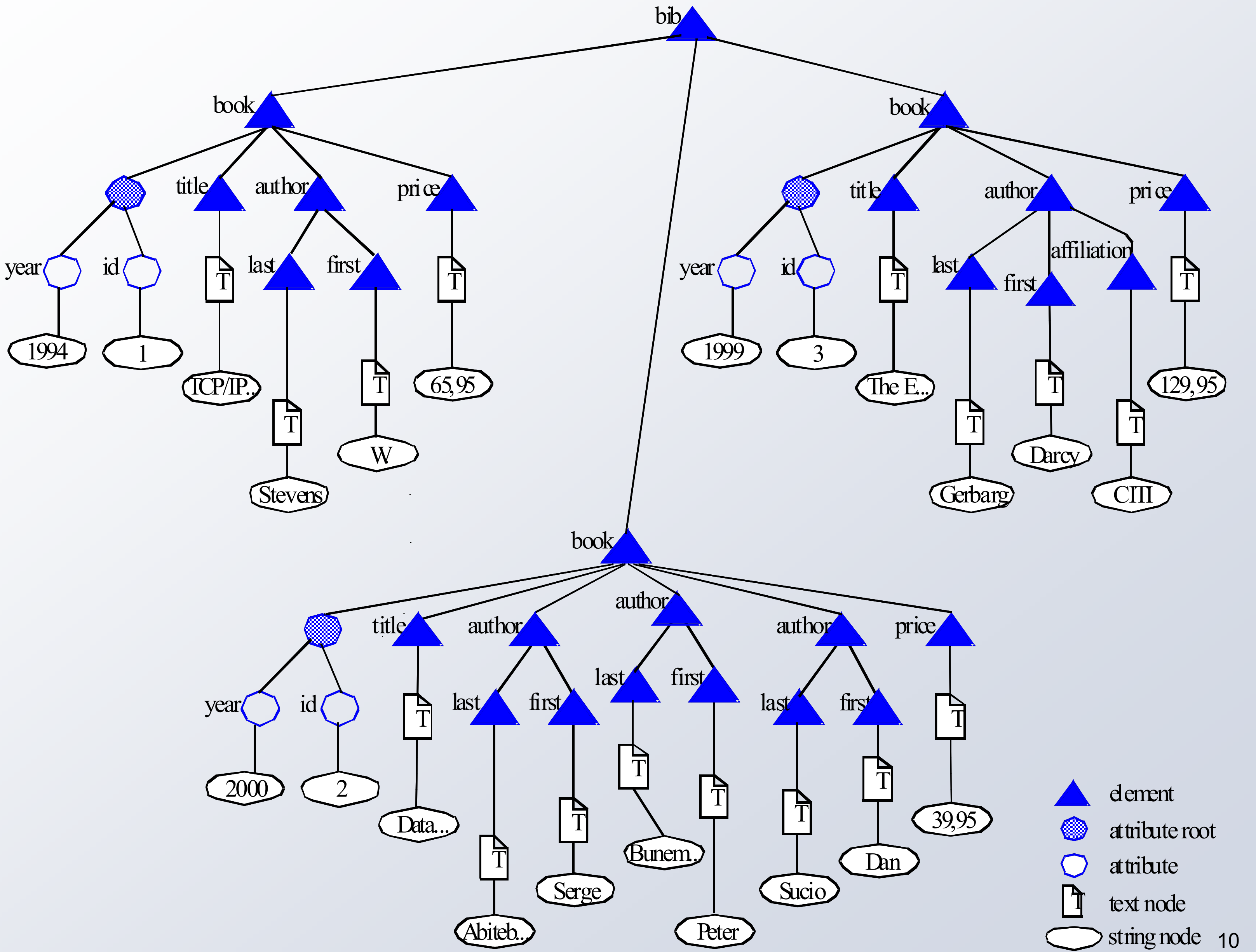
Node
services

pport of fine-
ained locking

new revolu-
ion ahead?

- Improved solution needed
  - fine-granular management and storage of the XML documents as native tree-like storage structures
  - navigational and direct access to all document nodes
  - indexing of nodes should accelerate declarative queries
  - modification of documents also required under multi-user operations (cooperative processing)
  - fine-granular locking: nodes, edges, and subtrees
- How to store and address tree nodes,
  which can be arbitrarily displaced by later insertions?
  - how do XML documents appear at the user level?
  - which storage structures are adequate?
  - which labeling scheme
    should be used for the nodes?

Example of a taDOM Tree: Conceptual Representation (View of the Lock Mgr)



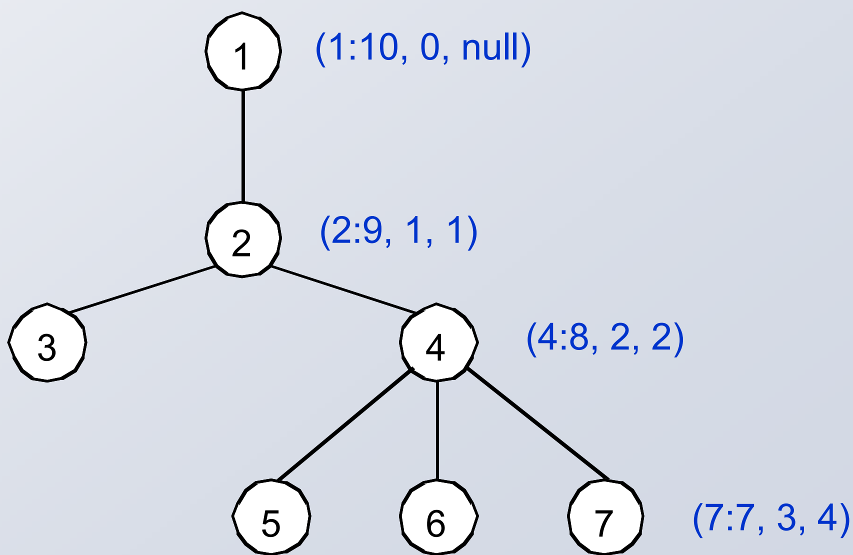
Node Labeling: Principal Approaches to a Solution

- representation of an XML document: ordered, labeled tree with nodes of type element, attribute, text (and attribute root, string)
- labeling scheme should be insensitive to insertions
- 13 different axes defined in XPath (sequence semantics)
- support of the most important axes required: **parent/child, ancestor/descendant, preceding-sibling/following-sibling**
- **two classes: range-based and prefix-based schemes**

Range-based Schemes

- positions of nodes marked by (DocNo, LeftPos:RightPos, LevelNo)
- LP and RP describe the labeling range in each node with its subtree; generated by a depth-first traversal of the tree
- ancestor-descendant containment (DocNo is omitted):
a node n1 (LP1:RP1, lv1) contains a node n2 (LP2:RP2, lv2),
iff LP1 < LP2 and RP1 > RP2.
- additional condition for parent-child containment: lv1 = lv2 - 1
- supporting preceding-sibling/following-sibling relationship?

- simple example



label template (LP:RP, lv, P, LP)

xtensions &
ptimizations

daptation to
L documents?

Labeling of
L tree nodes

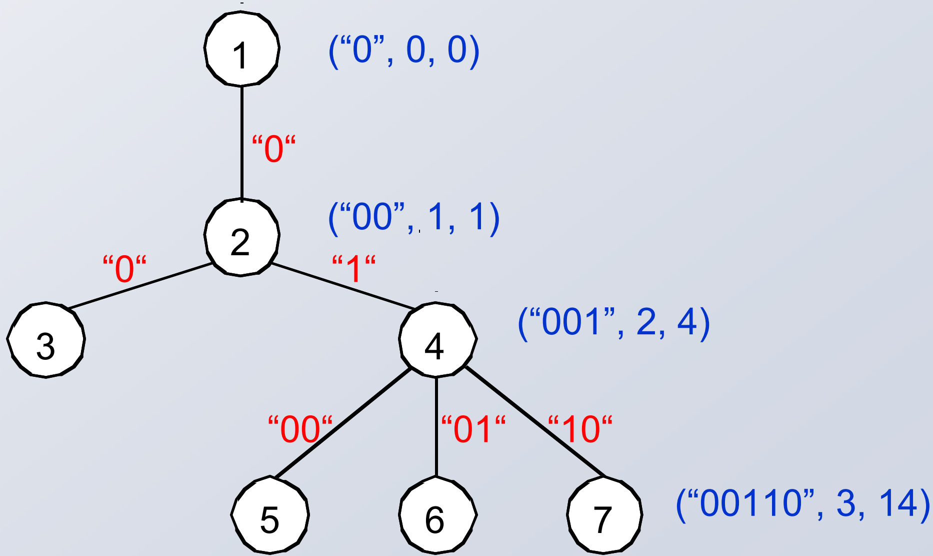
Node
services

pport of fine-
ained locking

new revolu-
ion ahead?

Prefix-Based Schemes

- each node is encoded with a unique string S such that
  - S(v) is before S(u) in lexicographic order iff node v is before node u in the document order
  - S(v) is a prefix of S(u) iff node v is the ancestor of node u
- simple example:
  - assign to the outgoing edges of each node a set of prefix-free binary strings in lexicographical order from left to right
  - the label of each node is the concatenation of the parent's label and the string assigned to its incoming edge
  - record the level of a node
  - add the edge string length esl to each node descriptor to derive the ancestor label



label template (S, lv, esl)

Node Labeling Scheme

- labels must
  - be immutable for the lifetime of the nodes
  - preserve the document order, when inserting new nodes
  - easily reveal the level and the ID for all ancestor nodes
- DeweyID consists of several divisions separated by dots
  - overflow mechanism: even division values

    d_1 = 1.3.17.2.2.3.4.9 d_2 = 1.3.17.2.3.7

  - level determination

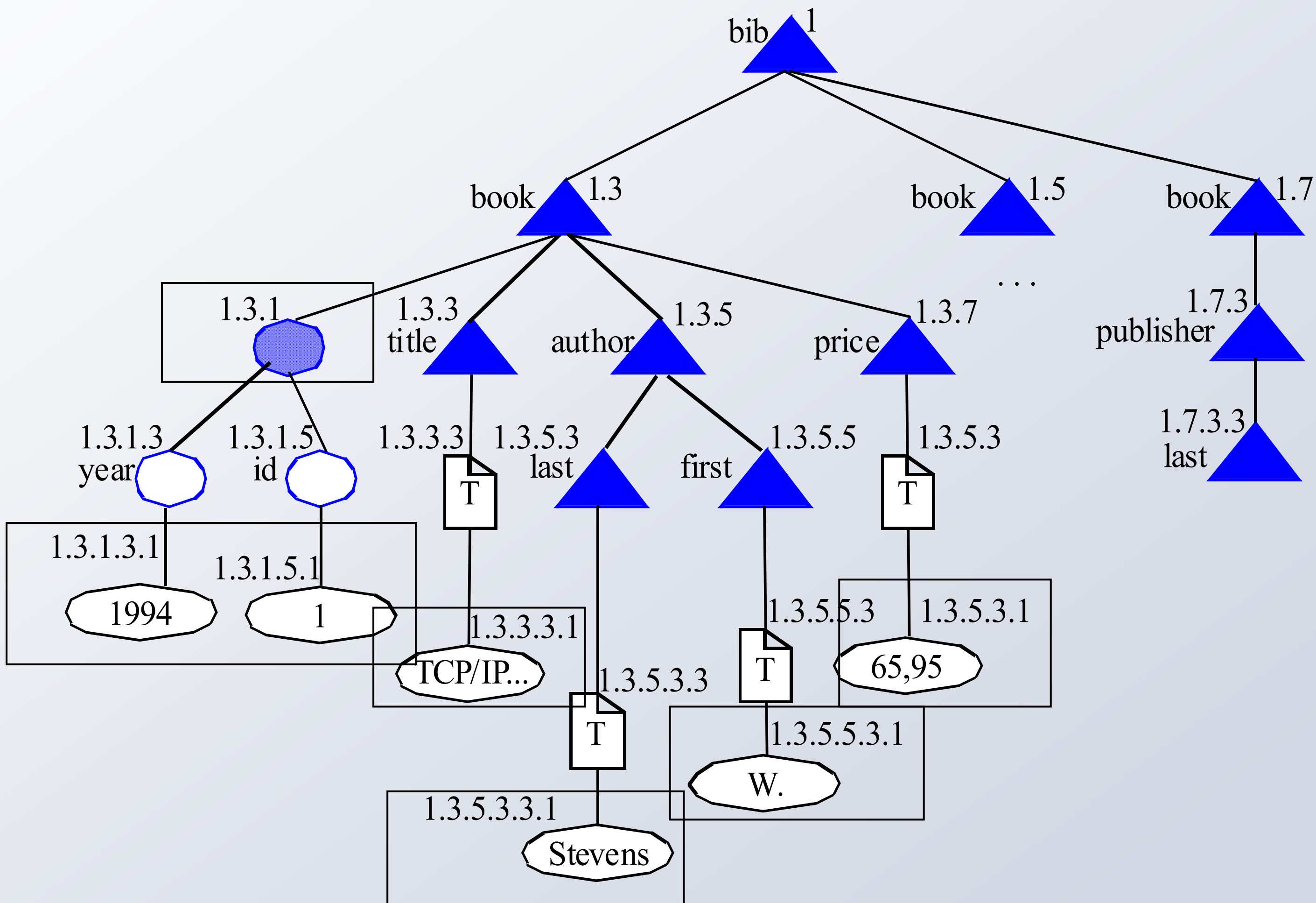
    d_1 = 1.3.17.2.2.3.4.9

  - ancestor IDs: a_0 = 1; a_1 = 1.3; a_2 = 1.3.17; a_3 = 1.3.17.2.2.3
  - ordering d_2 ? d_1

    d_1 < d_2 : 1.3.17.2.2.3.4.9 < 1.3.17.2.3.7

Initial Assignment of DeweyIds

- assignment of division values is affected by parameter *distance* (= 2)
- on initial loading, only odd division values are assigned

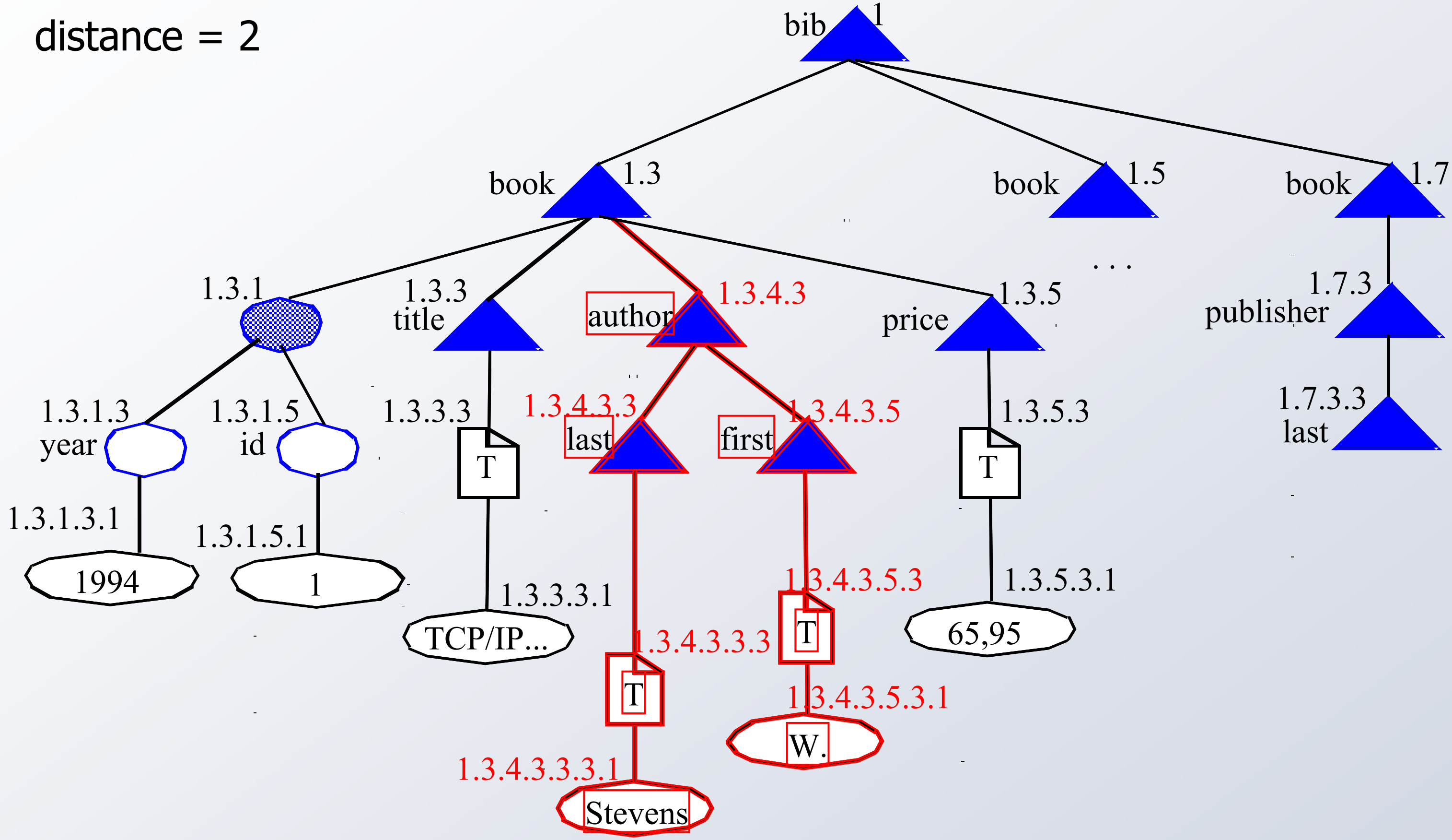


xtensions & ptimizations
daptation to L documents?
Labeling of L tree nodes
Node services
pport of fine- ained locking
new revolu- ion ahead?

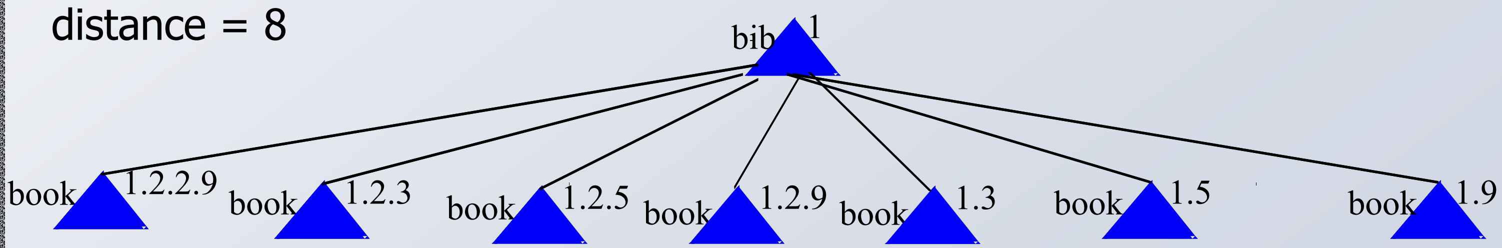
DeweyIDs: Insertion of a Subtree

Extensions & optimizations
Adaptation to XML documents?
Labeling of XML tree nodes
Node services
Support of fine-grained locking
A new revolution ahead?

distance = 2



worst-case considerations:
distance = 8



Benefits of DeweyID Use

- Existing DeweyIDs allow the assignment of new IDs without the need to reorganize the IDs of nodes present. Relabeling only in case of violations of implementation restrictions
- The DeweyID of each ancestor node can be determined in a very simple way
- Comparison of two DeweyIDs delivers the order of the respective nodes in the left-most depth-first stored document.
- Checking whether node d1 is an ancestor of d2 only requires to check whether DeweyID of d1 is a prefix of DeweyID of d2.
- High distance values reduce the probability of reorganization. They have to be balanced against increased storage space

but: DeweyIDs may become very long

Encoding of DeweyIDs

xtensions &
ptimizations

daptation to
L documents?

Labeling of
L tree nodes

Node
services

pport of fine-
ined locking

new revolu-
ion ahead?

- Fixed length field

TL	L_0	O_0	L_1	O_1	. . .	L_k	O_k

TL = total length
L_i =length of i-th division value
O_i = value of i-th division

$L_i = 6$: $L_{Oi} < 64$: $O_i < 2^{64}$ bits   encoding for $O_i = 7$ needs 6+3 bits

- Fixed- and variable-length length fields

TL	L_{f0}	L_{v0}	O_0	L_{f1}	. . .	L_{vk}	O_k

l_f = length of L_{fi}
L_{fi} = length of L_{vi}
L_{vi} = length of the i-th division

length of $L_{vi} < 2^{Lfi}$: value of $O_i < 2^{Lvi}$

$l_f = 2$: $O_i < 2^{16}$

$l_f = 3$: $O_i < 2^{256}$

but penalty for small division values: encoding for $O_i = 7$ needs 3+2+3 bits

Encoding of DeweyIDs (2)

k-based representation

- m = log (k + 1)
- reserve one code of length m to represent the separator "."
- interpret a sequence of m-bit codes as a number with base k

k = 3: "0": 00, "1": 01, "2": 10, ".": 11

1.7.11 : TL 01 11 10 01 11 01 00 10

$1*3^0$ $2*3^1 + 1*3^0$ $1*3^2 + 0*3^1 + 2*3^0$

good space efficiency: O_i = 7 needs 6 bits, but no adaptation to value distribution

is there a better k: k = 1 or k = 7?

k = 7: "0": 000, "1": 001, "2": 010, "3": 011, …., ".": 111

1.7.11 : TL 001 111 001 000 111 001 100

$1*7^0$ $1*7^1 + 0*7^0$ $1*7^1 + 4*7^0$

encoding of O_i = 7 needs 9 bits

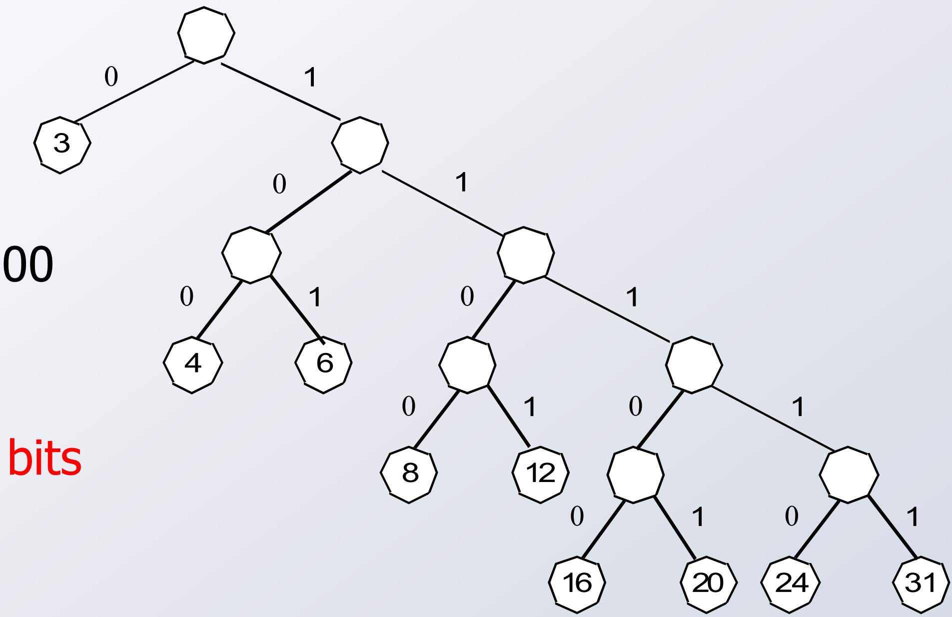
Encoding of DeweyIDs (3)

- Huffman Codes

TL	C_0	O_0	C_1	O_1	. . .	C_k	O_k

1.7.11: TL 0001 0111 1000100

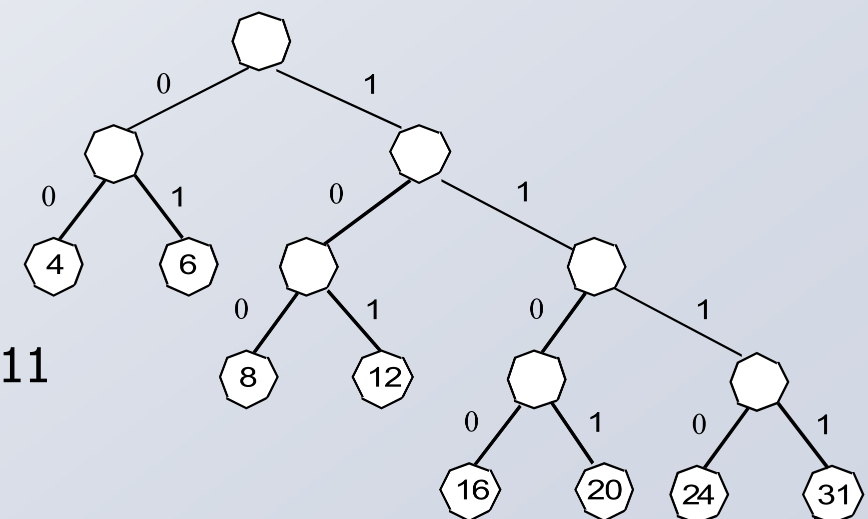
encoding of O_i = 7 needs 4 bits



- Degrees of freedom
  range weights and
  length assignments

1.7.11: TL 000001 000111 001011

encoding of O_i = 7 needs 6 bits



Characteristics of XML Documents Considered

xtensions & ptimizations
daptation to L documents?
Labeling of L tree nodes
Node services
pport of fine-ained locking
new revolu-ion ahead?

file name	description	size (bytes)	number of XML nodes	attributes	max. depth	⌀–depth	max. fanout	⌀–fan-ou of elems
1) treebank_e.xml	Encoded DB of English records of Wall Street Journal	86082517	2437666	1	**38**	**8.97**	**56385**	**2.33**
2) psd7003.xml	DB of protein sequences	716853016	21305818	1290647	**9**	**6.2**	**262527**	**3.99**
3) customer.xml	Customers from TPC-H benchmark	515660	13501	1	**5**	**3.92**	**1501**	**8.99**
4) ebay.xml	Ebay auction data	35562	156	0	7	4.76	12	5.0
5) lineitem.xml	Line items from TPC-H benchmark	32295475	1022976	1	5	3.96	60176	17.0
6) mondial-3.0.xml	Geographical DB of diverse sources	1784825	22423	47423	8	5.25	955	4.43
7) nasa.xml	Astronomical data	25050288	476646	56317	10	6.62	2435	2.79
8) orders.xml	Orders from TPC-H Benchmark	5378845	150001	1	5	3.93	15001	10.0
9) SwissProt.xml	DB of protein sequences	114820211	2977031	2189859	7	4.9	50000	6.75
10) uwm.xml	Courses of a University Website	2337522	66729	6	7	4.83	2112	4.21

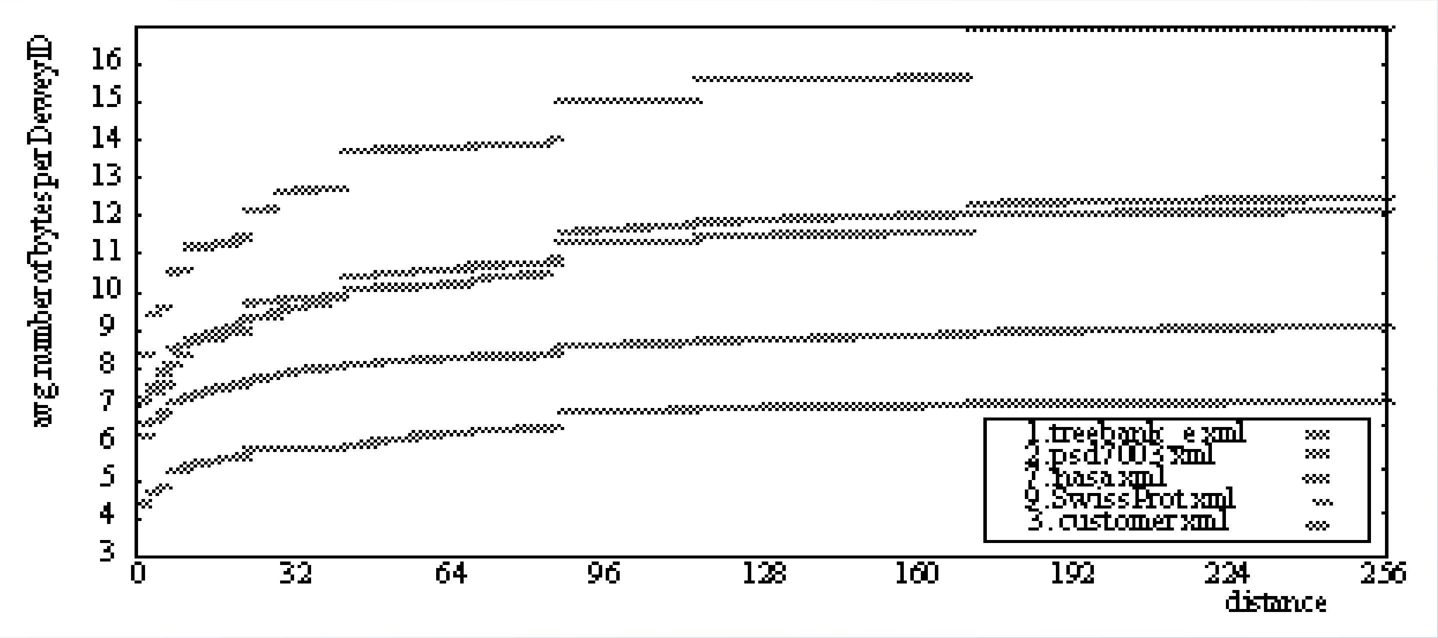
Encoding of DeweyIDs (4)

Huffman code	L_i	value range of O_i
0	3	1-7
100	4	8-23
101	6	24-87
1100	8	88-343
1101	12	344-4439
11100	16	4440-69975
11101	20	69976-1118551
11110	24	1118552-17895767
11111	31	17895768-2147483646

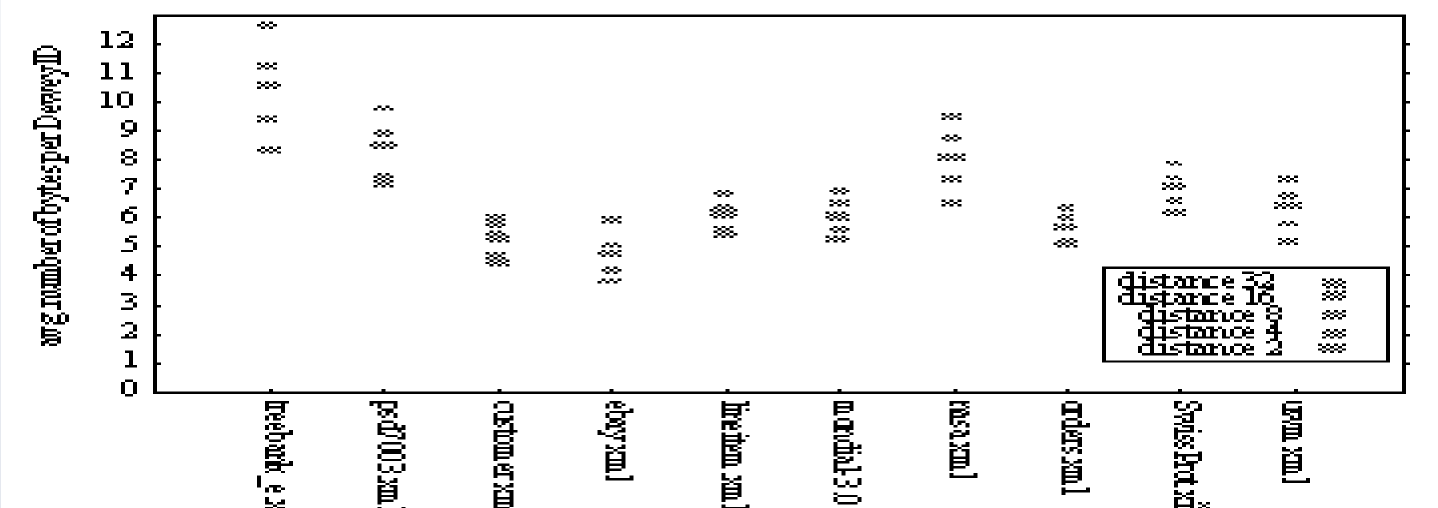
Optimization potential

- cut prefix 1.
  this may change the optimal Huffman code assignment
- virtualize taDOM extensions
- apply prefix compression for DeweyIDs used as keys and pointers

Avg. Sizes of DeweyIDs Grouped by the Documents Avg. Depth



influence of the distance parameter

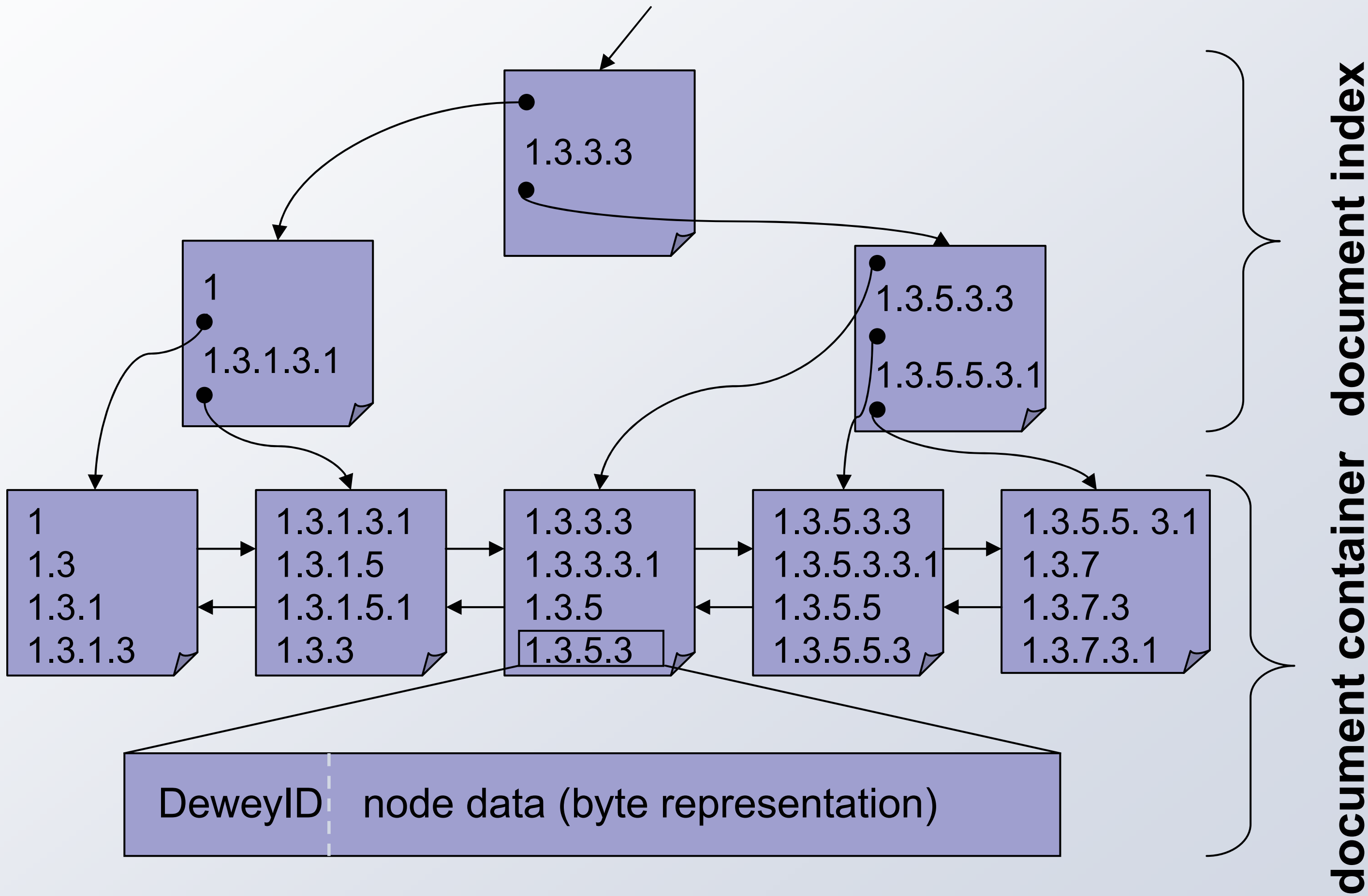


Levels of Abstraction

abstraction level	XDBS			RDBS
processing model	navigational node oriented	stream based node oriented	declarative sequence based	declarative set valued
language model/ interface	DOM	SAX	XQuery	SQL
logical access model	node services		XML query algebra	relational algebra
physical access model	Scan, FC-Impl., NS-Impl., PA-Impl., …		physical algebra operators	physical algebra operators
storage model	XTC document index, XTC element index			storage structures

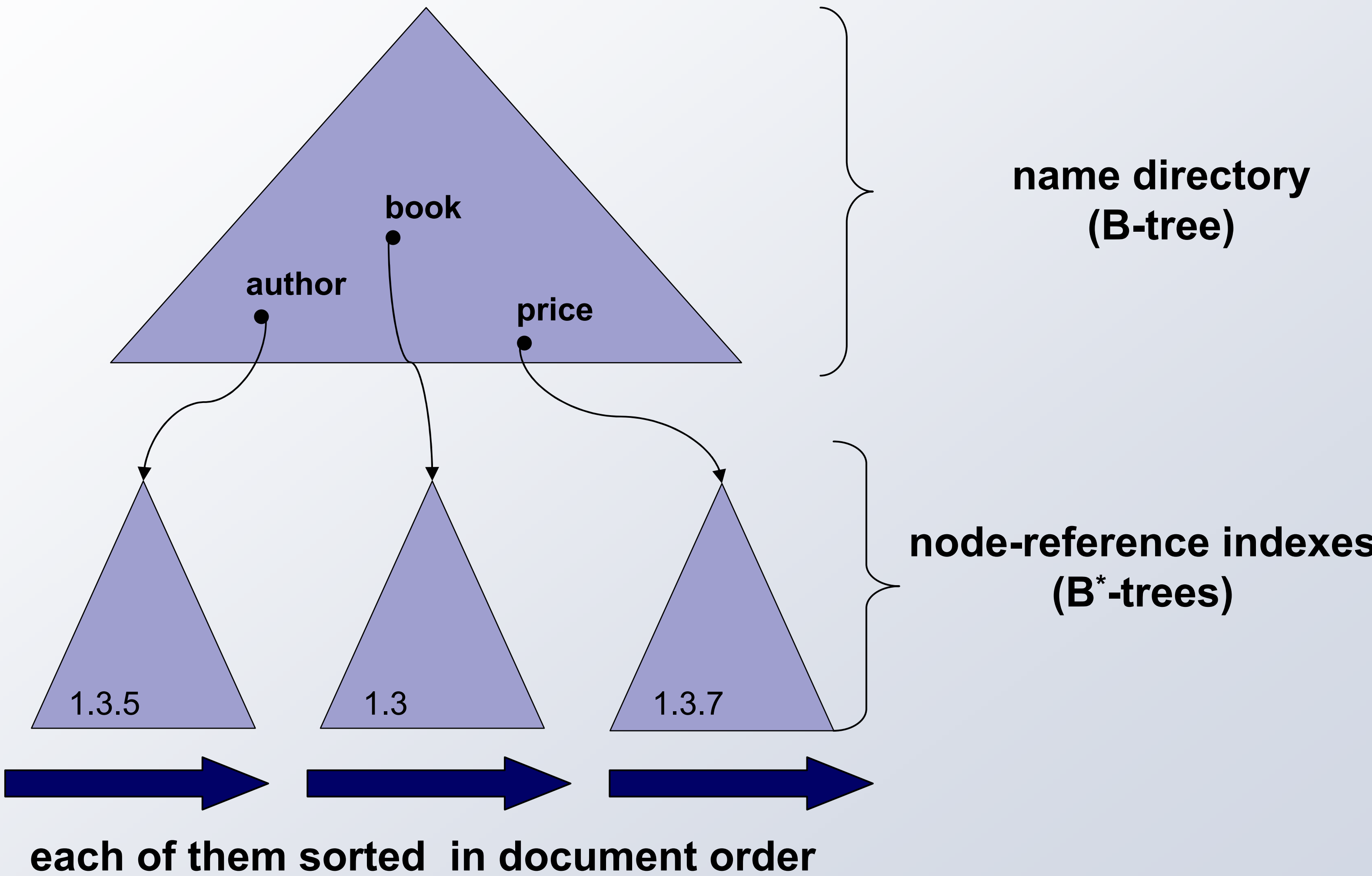
xtensions &
ptimizations
daptation to
L documents?
Labeling of
L tree nodes
Node
services
pport of fine-
ained locking
new revolu-
ion ahead?

XTC Document Index



prefix compression works!

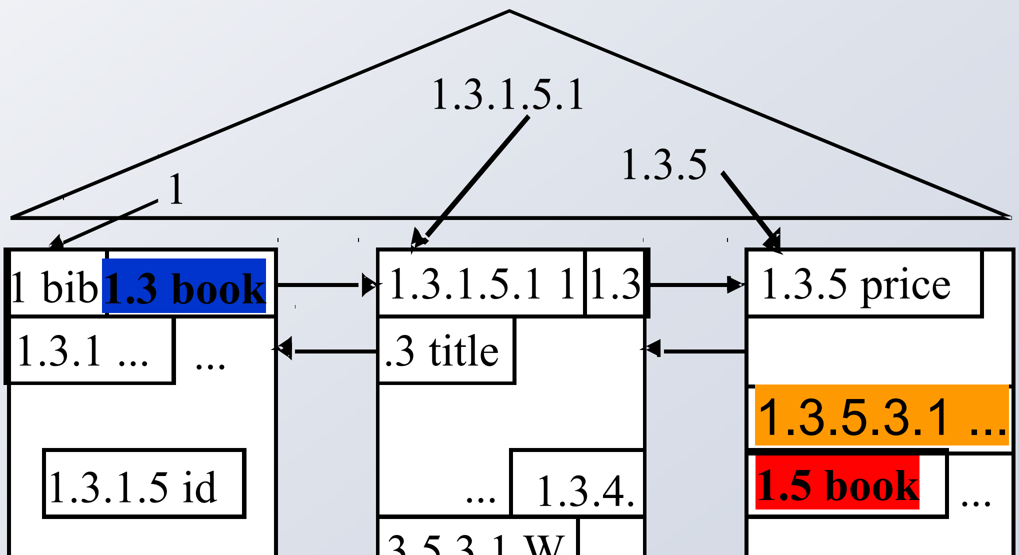
XTC Element Index



xtensions & ptimizations
daptation to L documents?
Labeling of L tree nodes
Node services
pport of fine-ained locking
new revolu-ion ahead?

Example: DeweyID Support for a Navigational Operation

- Expressiveness of DeweyIDs allows derivation of ancestors
- Navigational axes
  - parent
  - first/last child
  - next/previous sibling
- Context node: 1.5
  - previous sibling without scanning the container pages



XQuery – XPath

- Problem: XQuery is really complex
  - order-preserving joins, implicit grouping, result construction …
- therefore at the moment "only" XPath

```
doc("sample.xml")//autor
[count
   (parent::buch/preceding-sibling::
    element())>3]/vname[../nname = "Adams"]
```

- still complex
- concentration on path steps

```
Axis::name_test
```

basic processing units, sequence of path steps,
evaluation order: bottom-up, top-down, starting in the middle

XPath Axes

1. **parent**
2. **child**
3. **ancestor**
4. **descendant**
5. **previous-sibling**
6. **following-sibling**
7. **previous**
8. **following**
9. attribute
10. namespace
11. self
12. ancestor-or-self
13. descendant-or-self

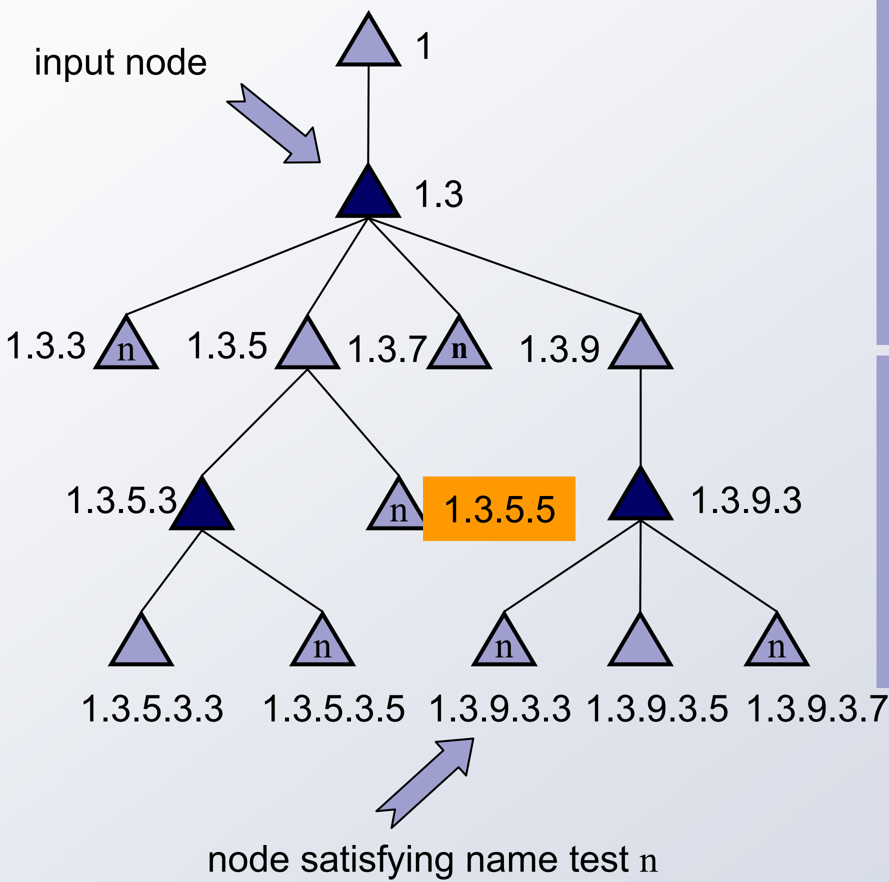
Axis Operators

- For each of the 8 relevant XPath axes an own operator
- Input
  - name test
  - duplicate-free sequence of DeweyIDs in document order
- Output
  - duplicate-free sequence of DeweyIDs in document order on specified axis: each referenced node satisfies the name test
- Chaining of axis operators to evaluate XPath expressions of the form

$$\texttt{axis}_1\texttt{::name\_test}_1\texttt{/…/axis}_n\texttt{::name\_test}_n$$

Child Axis

input sequence: dark nodes



observe level information

„probing“ of the parent DeweyIDs

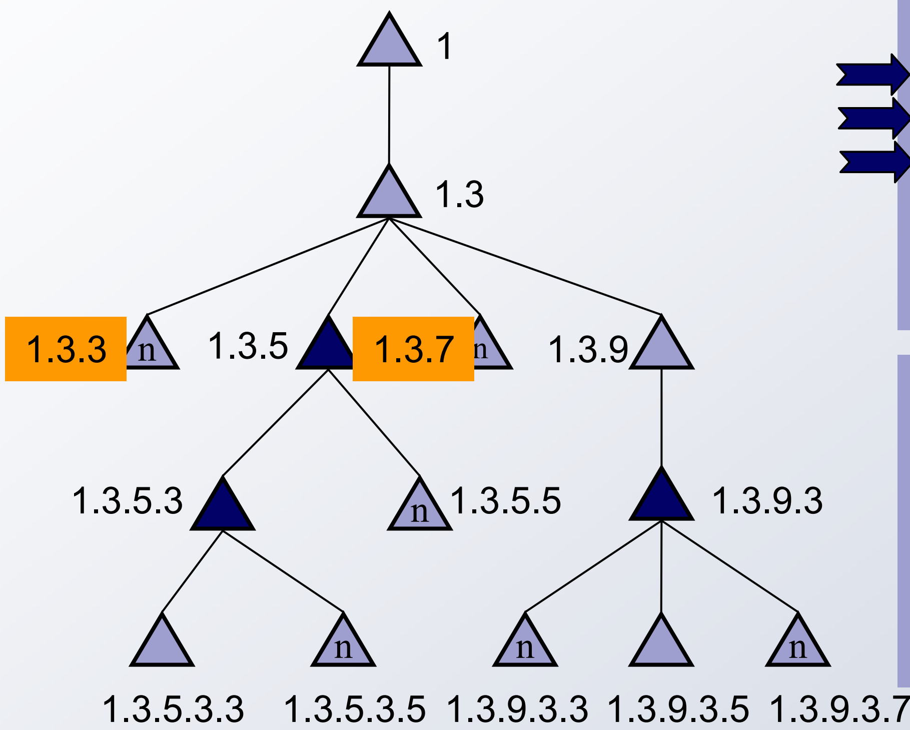
HT(1.3) HT(1.3.5.3) HT(1.3.9.3)	1.3.3 1.3.5.3.5 1.3.5.5 1.3.7 1.3.9.3.3 1.3.9.3.7

result

1.3.3
1.3.5.3.5
1.3.7
1.3.9.3.3
1.3.9.3.7

Descendant Axis

document order avoids duplicates in the result



merge method

input	descendents
1.3.5	1.3.3
1.3.5.3	1.3.5.3.5
1.3.9.3	1.3.5.5
	1.3.7
	1.3.9.3.3
	1.3.9.3.7

result

1.3.5.3.5
1.3.5.5
1.3.9.3.3
1.3.9.3.7

Following-Sibling Axis

xtensions &
ptimizations

daptation to
L documents?

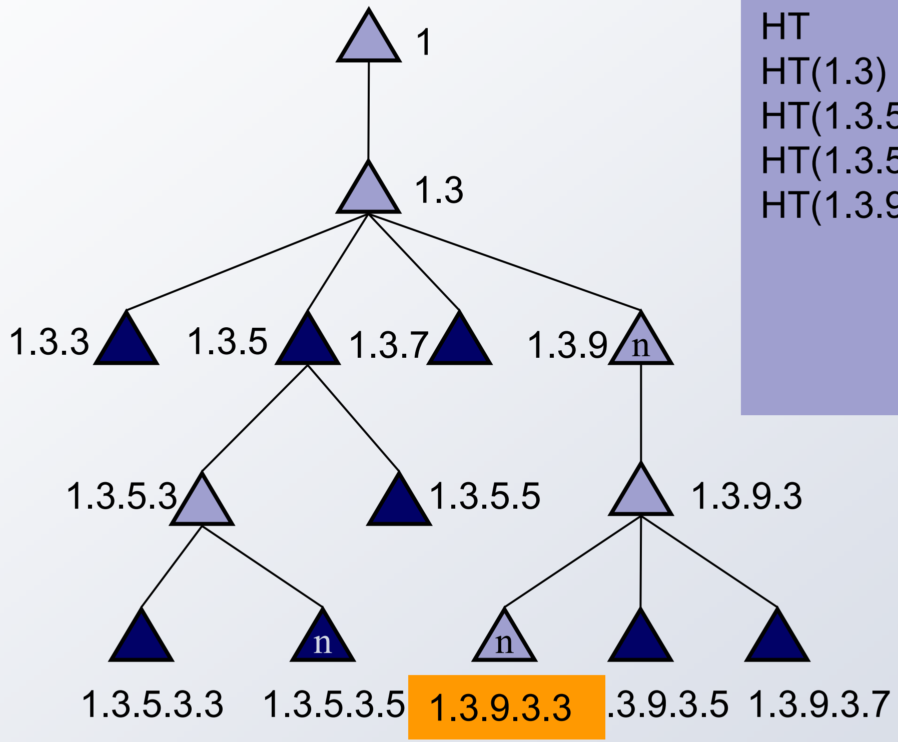
Labeling of
L tree nodes

Node
services

pport of fine-
ained locking

new revolu-
ion ahead?

all processing steps avoid duplicates: O(n)



„probing“ of predecessor nodes

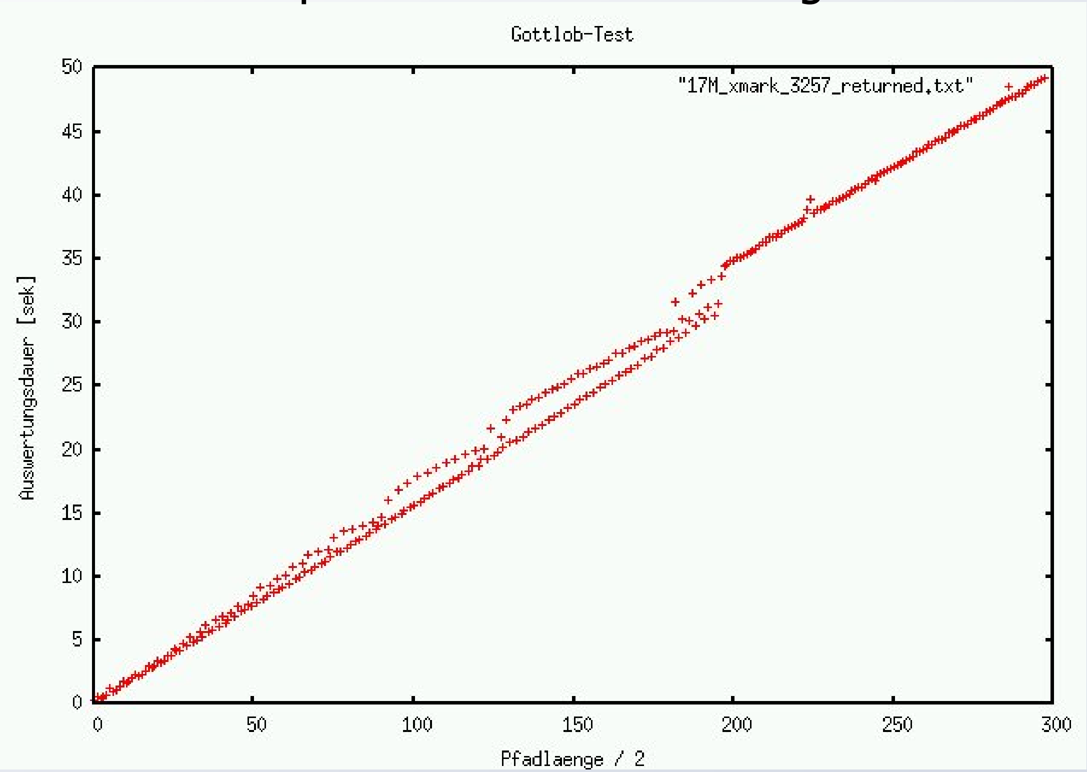
HT	following-siblings
HT(1.3) = 1.3.3	1.3.5.3.5
HT(1.3.5.3) = 1.3.5.3.3	1.3.9
HT(1.3.5) = 1.3.5.5	1.3.9.3.3
HT(1.3.9.3) = 1.3.9.3.5	

Test Case

early duplicate elimination avoids exponential behavior of algorithms

test document:
`Xmark`

`3257 elements each`



query

```
doc(...)/descendant::item/child::name
/parent::item/child::name/...
/parent::item/child::name
```

Concurrency Control – Node Lock Compatibilities

- Node locks and compatibility matrix

Compatibility matrix

	-	IR	NR	LR	SR	IX	CX	SU	SX
IR	+	+	+	+	+	+	+	-	-
NR	+	+	+	+	+	+	+	-	-
LR	+	+	+	+	+	+	-	-	-
SR	+	+	+	+	+	-	-	-	-
IX	+	+	+	+	-	+	+	-	-
CX	+	+	+	-	-	+	+	-	-
SU	+	+	+	+	+	-	-	-	-
SX	+	-	-	-	-	-	-	-	-

Read locks

lock	effect
IR (intention read)	Read lock on non-direct child node
NR (node read)	Read lock on the node
LR (level read)	Read lock on context node and all direct-child nodes
SR (subtree read)	Read lock on entire subtree

Write locks

lock	effect
SX (exclusive)	Write lock on entire subtree
CX (child excl.)	Write lock on direct child node
IX (intent. excl.)	Write lock on non-direct child node
SU (update)	Read lock with intended update operation on entire subtree

Concurrency Control – Node Read Lock

xtensions &
ptimizations

daptation to
. documents?

abeling of
L tree nodes

Node
services

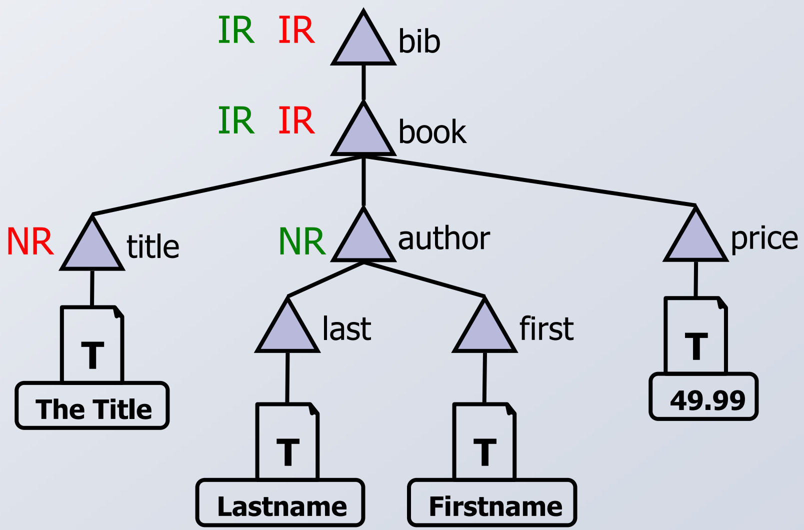
pport of fine-
ained locking

new revolu-
ion ahead?

- NR
  - Requested for reading the context node
  - Requires IR locks on the ancestor path

Transaction T_1 is reading <title>

Transaction T_2 is reading <author>

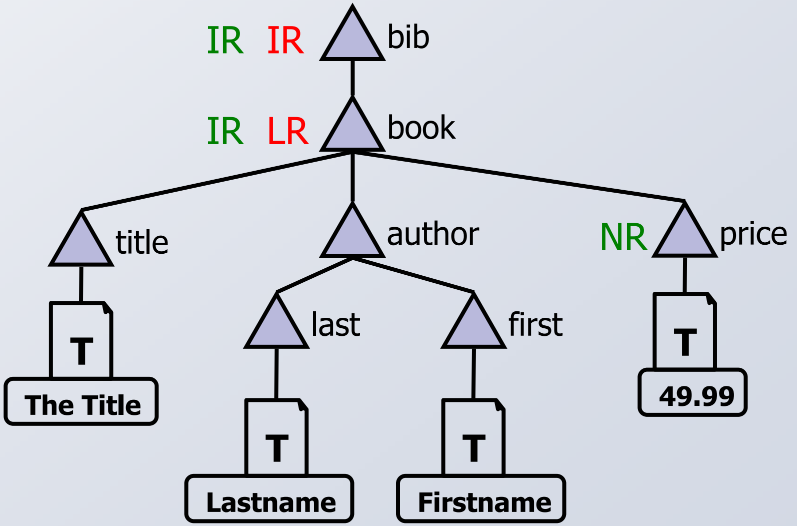


Concurrency Control – Level Read Lock

- LR
  - Requested for reading the context node and all nodes located in the level below (all direct-child nodes)
  - Requires NR locks on the ancestor path

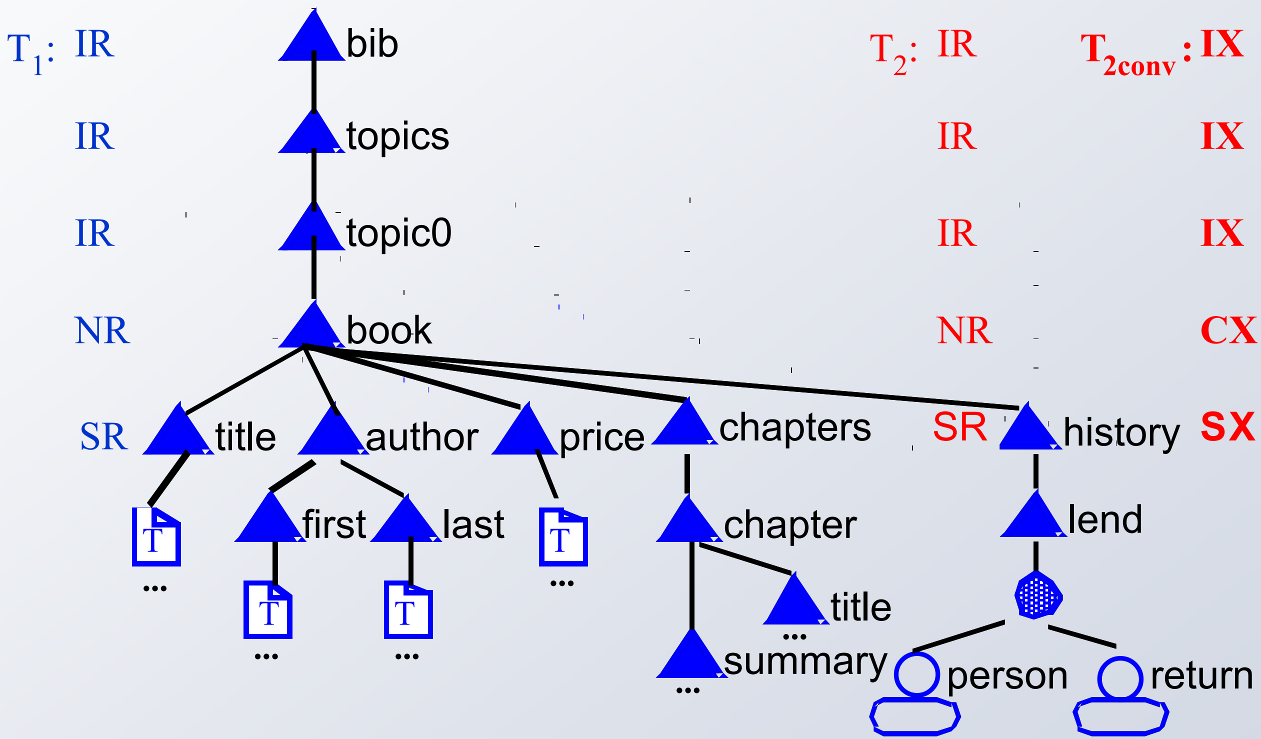
Transaction T_1 is reading <book> and all direct-child nodes (<title>, <author>, and <price>)

Transaction T_2 is reading <price>



Concurrency Control – Lock Depth

T_1 reads this book with lock depth = 4:
options SR on bib, topics, topic0, book, or on each children of book



T2 reads the history subtree (SR)

and decides to attach a new lend subtree

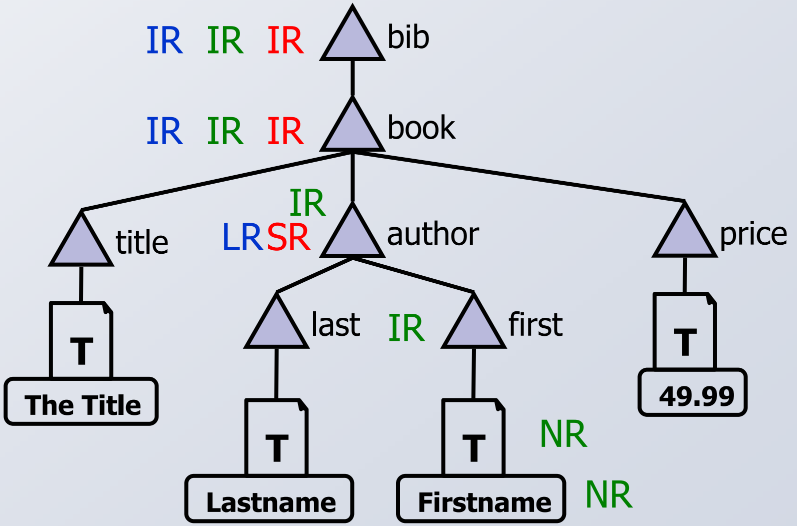
Concurrency Control – Subtree Read lock

- SR
  - Requested for reading the context node and all nodes located in the subtree below
  - Requires IR locks on the ancestor path

Transaction T_1 is reconstructing
<author>
<last>Lastname</last>
<first>Firstname</first>
</author>

Transaction T_2 is reading the value of the text node in <first>

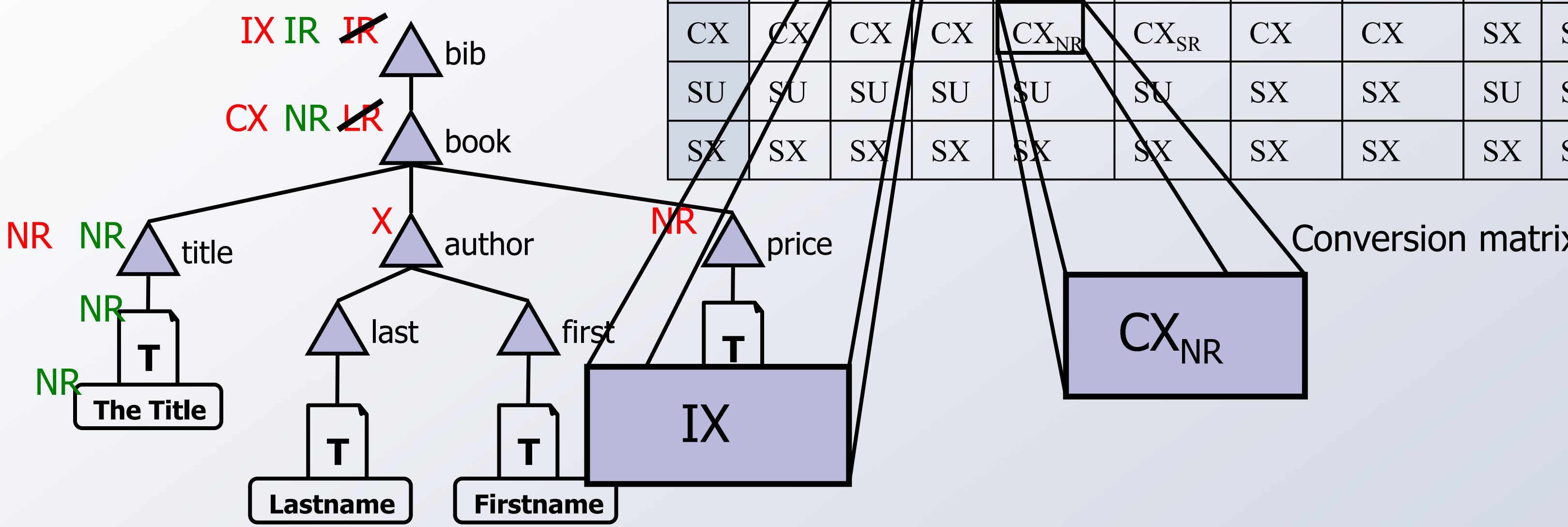
Transaction T_3 is reading all direct-child nodes of <author> (<last> and <first>)



Concurrency Control – Node Lock Conversion

- Lock conversion in the ancestor path of author

	-	IR	NR	LR	SR	IX	CX	SU	S
IR	IR	IR	NR	LR	SR	IX	CX	SU	S
NR	NR	NR	NR	LR	SR	IX	CX	SU	S
LR	LR	LR	LR	LR	SR	IX_{NR}	CX_{NR}	SU	S
SR	SR	SR	SR	SR	SR	IX_{SR}	CX_{SR}	SR	S
IX	IX	IX	IX	IX_{NR}	IX_{SR}	IX	CX	SX	S
CX	CX	CX	CX	CX_{NR}	CX_{SR}	CX	CX	SX	S
SU	SU	SU	SU	SU	SU	SX	SX	SU	S
SX	SX	SX	SX	SX	SX	SX	SX	SX	S



Transaction T_1 is reading <book> and all its direct-child nodes

Transaction T_2 is reading <book>, the first child node <title> and its value

Transaction T_1 is deleting <author> and its entire subtree

Compatibility Matrix of taDOM3+

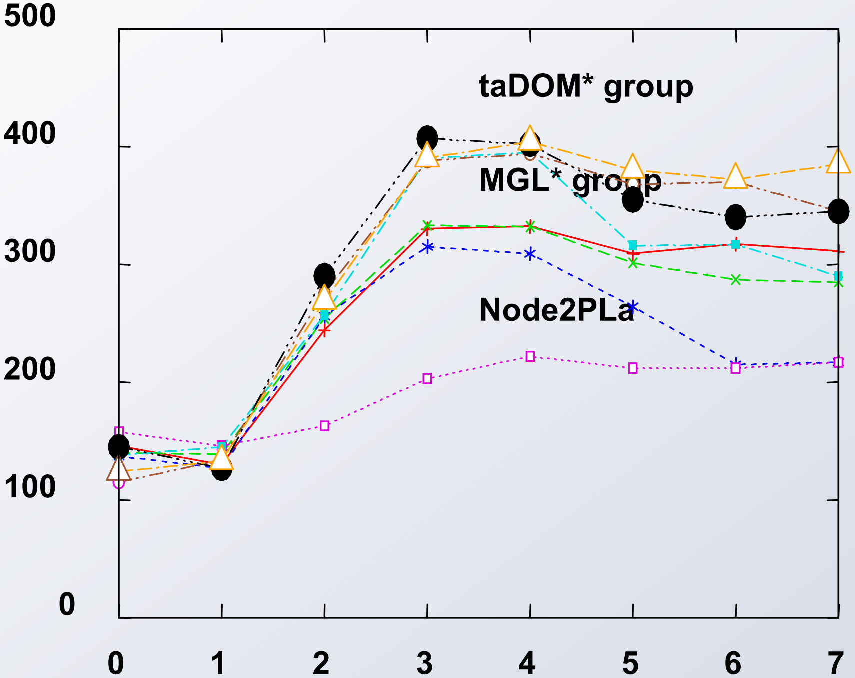
xtensions & ptimizations
daptation to L documents?
Labeling of L tree nodes
Node services
pport of fine-ained locking
new revolu-ion ahead?

	-	IR	NR	LR	SR	IX	NRIX	CX	NRCX	NU	NX	SU	SX
IR	IR	IR	NR	LR	SR	IX	NRIX	CX	NRCX	NU	NX	SU	SX
NR	NR	NR	NR	LR	SR	NRIX	NRIX	NRCX	NRCX	NR	NX	SU	SX
LR	LR	LR	LR	LR	SR	$NRIX_{NR}$	$NRIX_{NR}$	$NRCX_{NR}$	$NRCX_{NR}$	NU_{NR}	NX_{NR}	SU	SX
SR	SR	SR	SR	SR	SR	$NRIX_{SR}$	$NRIX_{SR}$	$NRCX_{SR}$	$NRCX_{SR}$	NU_{SR}	NX_{SR}	SR	SX
IX	IX	IX	NRIX	$NRIX_{NR}$	$NRIX_{SR}$	IX	NRIX	CX	NRCX	NX	NX	SX	SX
NRIX	NRIX	NRIX	NRIX	$NRIX_{NR}$	$NRIX_{SR}$	NRIX	NRIX	NRCX	NRCX	NX	NX	SX	SX
CX	CX	CX	NRCX	$NRCX_{NR}$	$NRCX_{SR}$	CX	NRCX	CX	NRCX	NX	NX	SX	SX
NRCX	NRCX	NRCX	NRCX	$NRCX_{NR}$	$NRCX_{SR}$	NRCX	NRCX	NRCX	NRCX	NX	NX	SX	SX
NU	NU	NU	NU	NU_{NR}	NU_{SR}	NX	NX	NX	NX	NU	NX	SU	SX
NX	NX	NX	NX	NX_{NR}	NX_{SR}	NX	NX	NX	NX	NX	NX	SX	SX
SU	SU	SU	SU	SU	SU	SX	SX	SX	SX	SU	SX	SU	SX
SX	SX	SX	SX	SX	SX	SX	SX	SX	SX	SX	SX	SX	SX

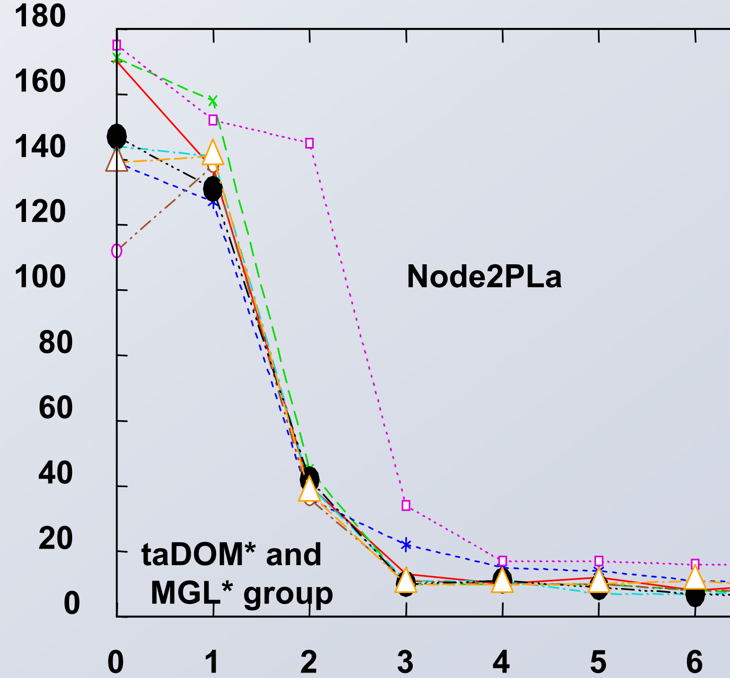
DBIS
Datenbanken und Informationssysteme

Contest of Lock Protocols

- benchmark: 3 groups of lock protocols
  - *-2PL and MGL* groups
  - taDOM* group: taDOM2, taDom2+, taDOM3, taDOM3+
  - meta-synchronization allows identical runtime env.



transaction throughput



deadlocks

Levels of Abstraction – The Next Steps

abstraction level	XDBS			RDBS
processing model	navigational node oriented	stream based node oriented	declarative sequence based	declarative set valued
language model/ interface	DOM	SAX	XQuery	SQL
logical access model	node services		XML query algebra	relational algebra
physical access model	Scan, FC-Impl., NS-Impl., PA-Impl., …		physical algebra operators	physical algebra operators
storage model	XTC document index, XTC element index			storage structures

xtensions &
ptimizations

daptation to
L documents?

Labeling of
L tree nodes

Node
services

pport of fine-
ained locking

new revolu-
ion ahead?

Horizontal Distribution of XDBMS Services

- Similar to RDBMSs
  - distributed DBMS (SN/SD)
  - federated DBMS
  - Multi-DBMS, …
- Vertical distribution of XDBMS services also possible (DB caching of XML documents)

Revolution in the DBMS Architecture?

- (O)RDBMS architecture is exhausted
  - Extenders, DataBlades, …
  - extensibility infrastructure, …
- XML services require substantial changes and new services in the upper layers
  - but: VITA, streaming, external files, …
  - cooperation of individual architectures in a DBMS ecosystem?