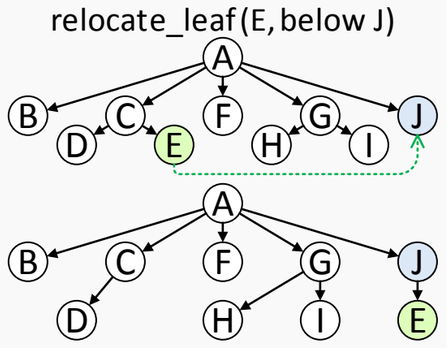
Indexing Highly Dynamic Hierarchical Data

Jan Finis, Robert Brunel, Alfons Kemper, Thomas Neumann, Norman May, Franz Färber
{finis, brunel, kemper, neumann}@in.tum.de {norman.may,franz.faerber}@sap.com

Challenge: Complex Updates on Hierarchical Data

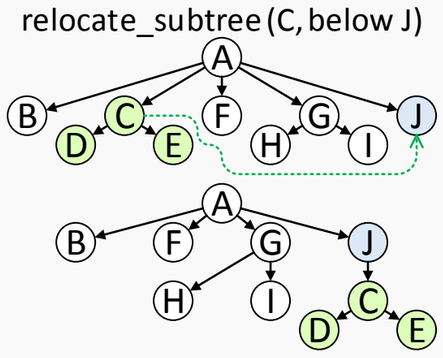
Leaf

relocate_leaf(E, below J)



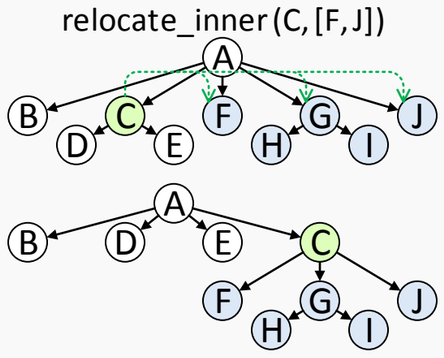
Subtree

relocate_subtree(C, below J)



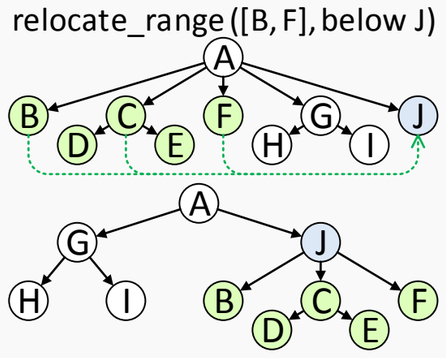
Inner Node

relocate_inner(C, [F, J])



Range

relocate_range([B, F], below J)

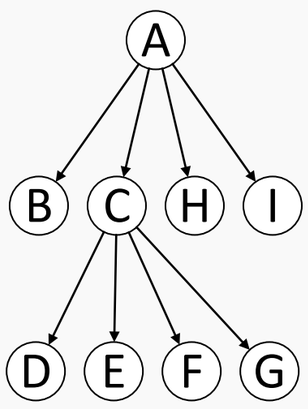


Asymptotic Runtime

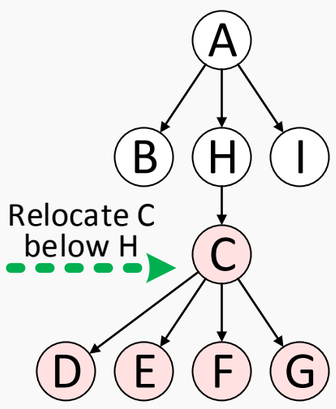
Scheme		Update Operations: leaf	subtree	inner node	range	skew[u]
naïve	Adjacency	1	1	c	c	1
	Linked	1	1	c	c	1
contain.	NI	n	n	n	n	n
	Dyn-NI	1	s	1	s	u
	Dyn-NI-Parent	1	s	c	s	u
	Dyn-NI-Level	1	s	s	s	u
path	Dewey	l*f*	l(*f* + *s*)	l(*f* + *s*)	l(*f* + *s*)	l(*f* + *s*)
	Dyn-Dewey	l	ls	ls	ls	l+*u*
index	AO-Tree	1	log *n*	log *n*	log *n*	1
	BO-Tree[B]	1	B $\log_B n$	B $\log_B n$	B $\log_B n$	1
	O-List[B]	1	*s*/B+B	*s*/B+B	*s*/B+B	*u*/B^2
	DeltaNI	log *u*	log *u*	log *u*	log *u*	log *u*

efficient | mostly efficient | inefficient

Problem: Labeling schemes cannot handle complex updates efficiently!



Key	Parent	PSL	NI	GapNI	Ordpath
A	null	0;8;0	[0,17]	[0,1700]	1
B	A	1;0;1	[1,2]	[100,200]	1.1
C	A	2;4;1	[3,12]	[300,1200]	1.3
D	C	3;0;2	[4,5]	[400,500]	1.3.1
E	C	4;0;2	[6,7]	[600,700]	1.3.3
F	C	5;0;2	[8,9]	[800,900]	1.3.5
G	C	6;0;2	[10,11]	[1000,1100]	1.3.7
H	A	7;0;1	[13,14]	[1300,1400]	1.5
I	A	8;0;1	[15,16]	[1500,1600]	1.7

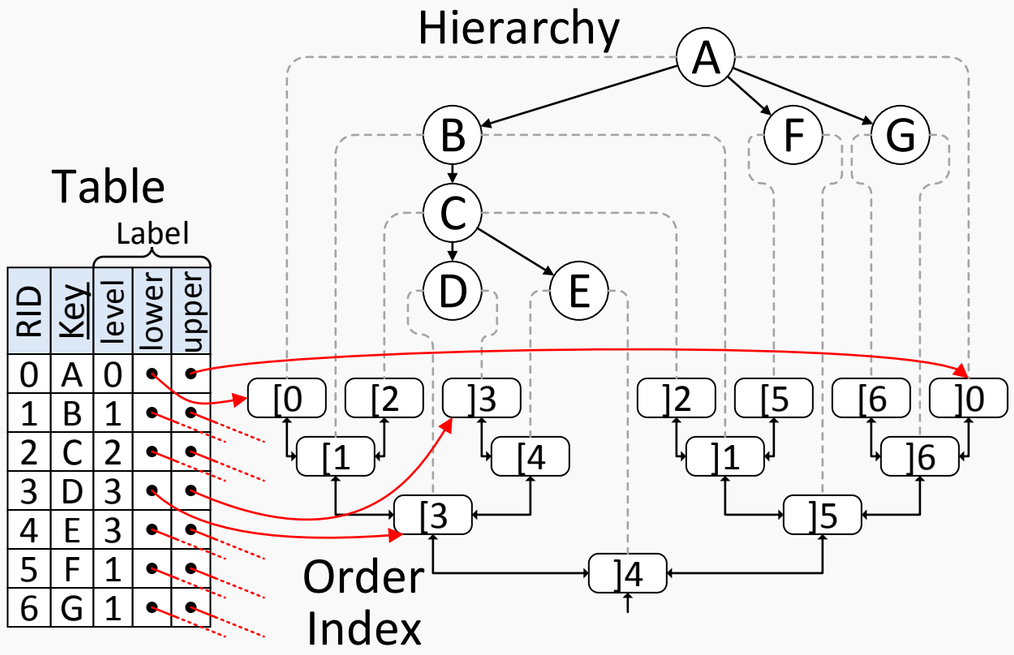


Key	Parent	PSL	NI	GapNI	Ordpath
A	null	0;8;0	[0,17]	[0,1700]	1
B	A	1;0;1	[1,2]	[100,200]	1.1
C	H	3;4;2	[4,13]	[1309,1390]	1.5.1
D	C	4;0;3	[5,6]	[1318,1327]	1.5.1.1
E	C	5;0;3	[7,8]	[1336,1345]	1.5.1.3
F	C	6;0;3	[9,10]	[1354,1363]	1.5.1.5
G	C	7;0;3	[11,12]	[1372,1381]	1.5.1.7
H	A	2;5;1	[3,14]	[1300,1400]	1.5
I	A	8;0;1	[15,16]	[1500,1600]	1.7

$\Omega(s)$

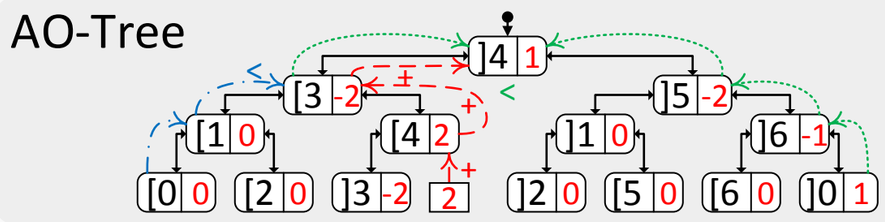
Solution: Order Indexes

Concept: Encode nested intervals by positions in an ordered data structure

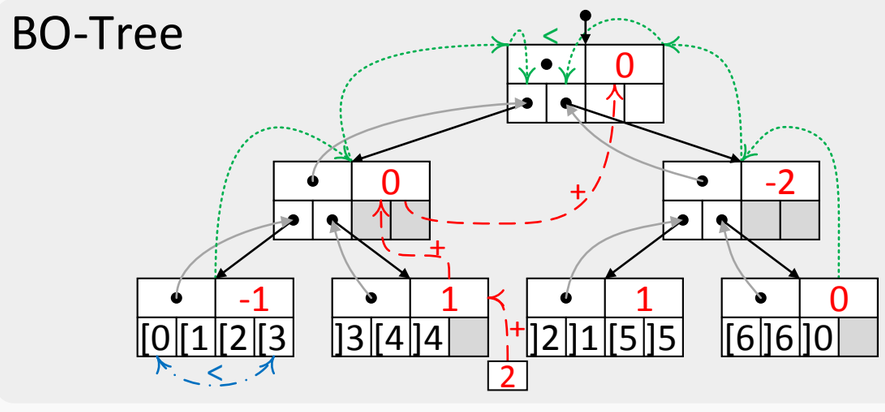


Flavors: AO-Tree, BO-Tree, Order List

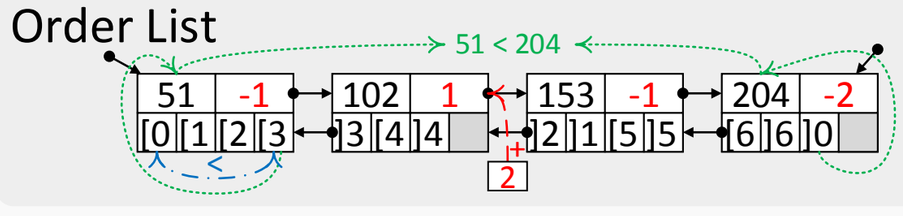
AO-Tree



BO-Tree

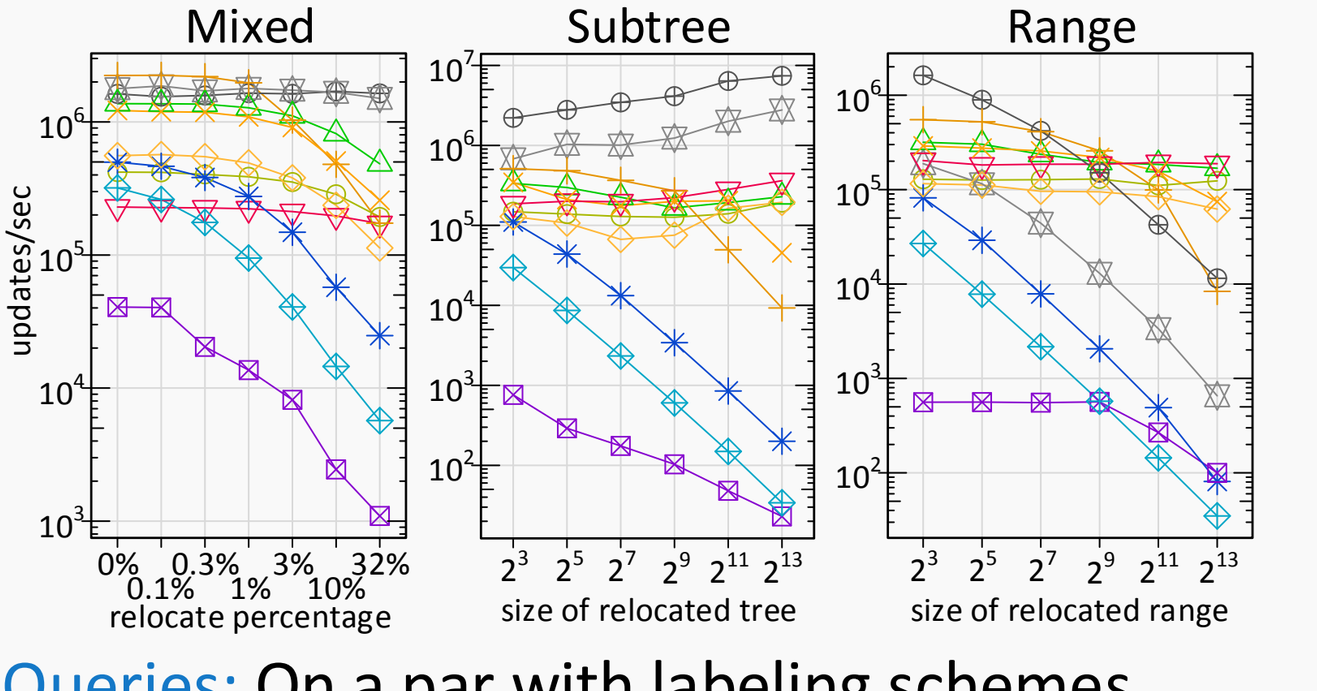


Order List



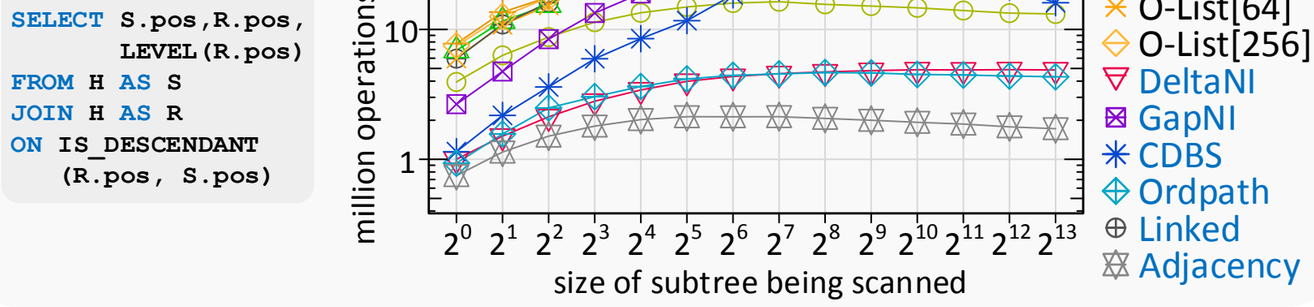
Queries	level([4)	before([0,[3)	before([3,]0)

Efficient Updates



Performance

Updates: Complex updates facilitated efficiently



Queries: On a par with labeling schemes

Query:

```
SELECT S.pos,R.pos,
       LEVEL(R.pos)
FROM H AS S
JOIN H AS R
ON IS_DESCENDANT
   (R.pos, S.pos)
```