

Object-Relational Indexing for General Interval Relationships

SSTD 2001, Redondo Beach, USA

Hans-Peter Kriegel, Marco Pötke, Thomas Seidl

 Database Group

Institute for Computer Science
University of Munich, Germany

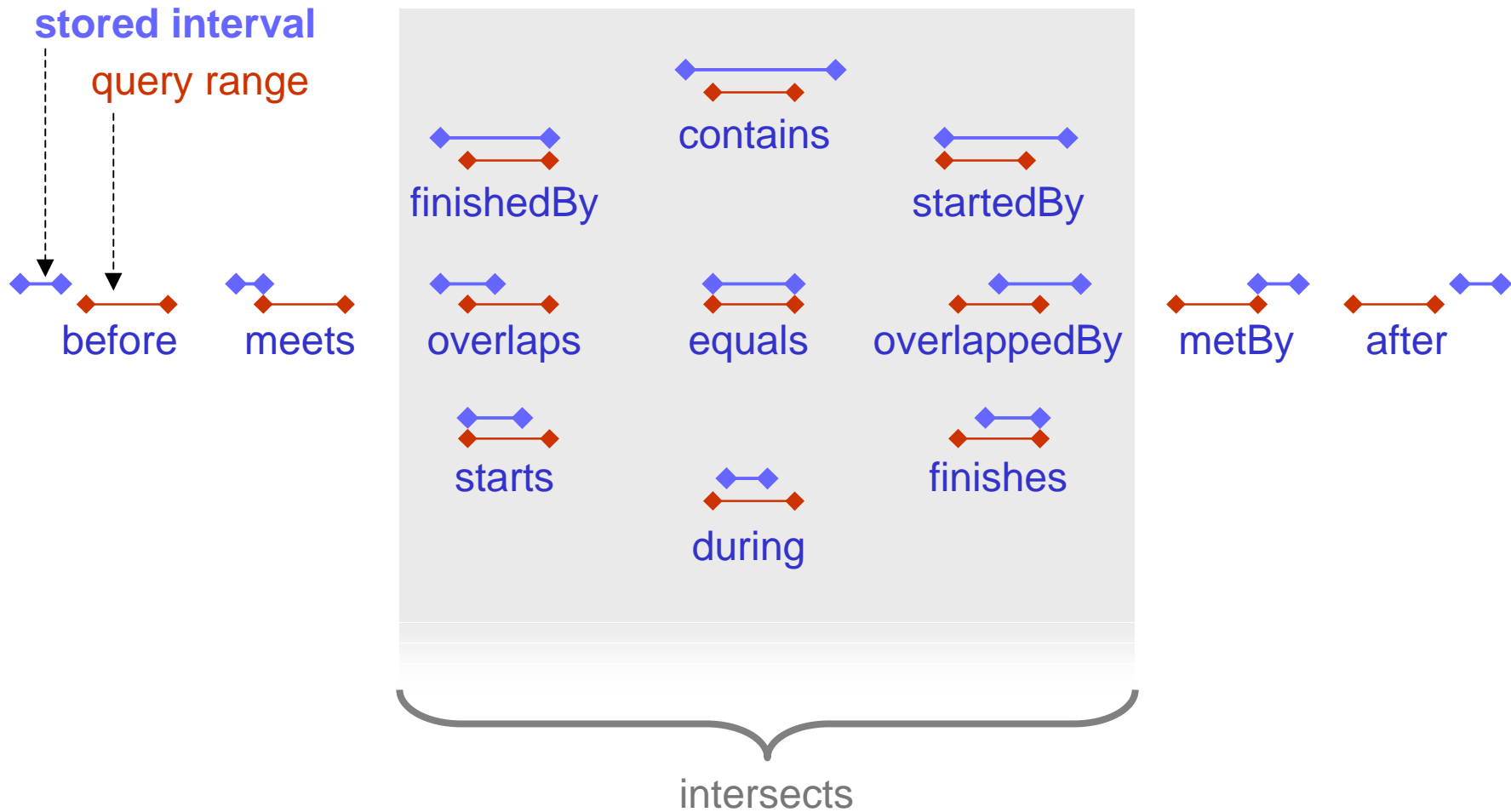


- 1. General Interval Relationships**
- 2. Relational Interval Tree**
- 3. Extended Query Processing**
 - Distinction of Node Classes
 - Closing Query Gaps
 - Extending Indexes
- 4. Experimental Evaluation**
 - Allen's Relationships
 - SQL:1999 Relationships

Thirteen Interval Relationships



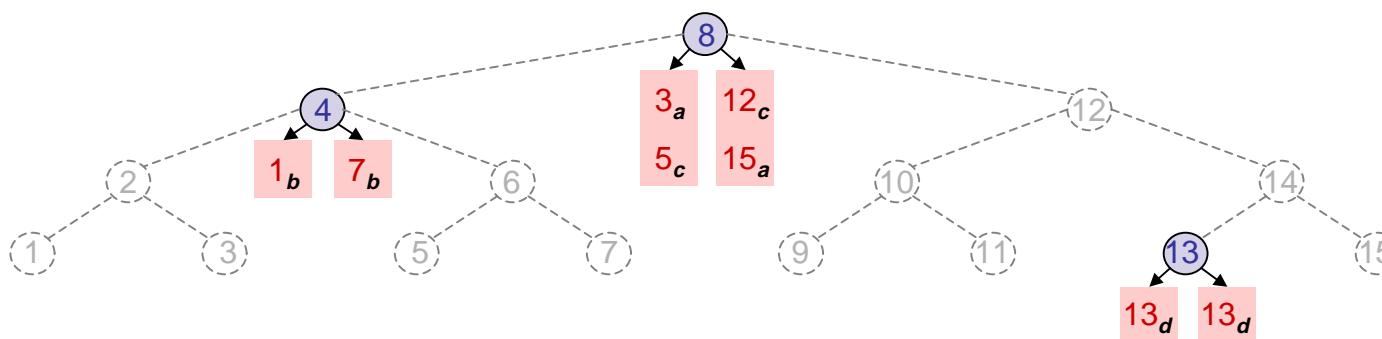
[Allen: CACM 1983]



Relational Interval Tree (again ...)



[patent pending]

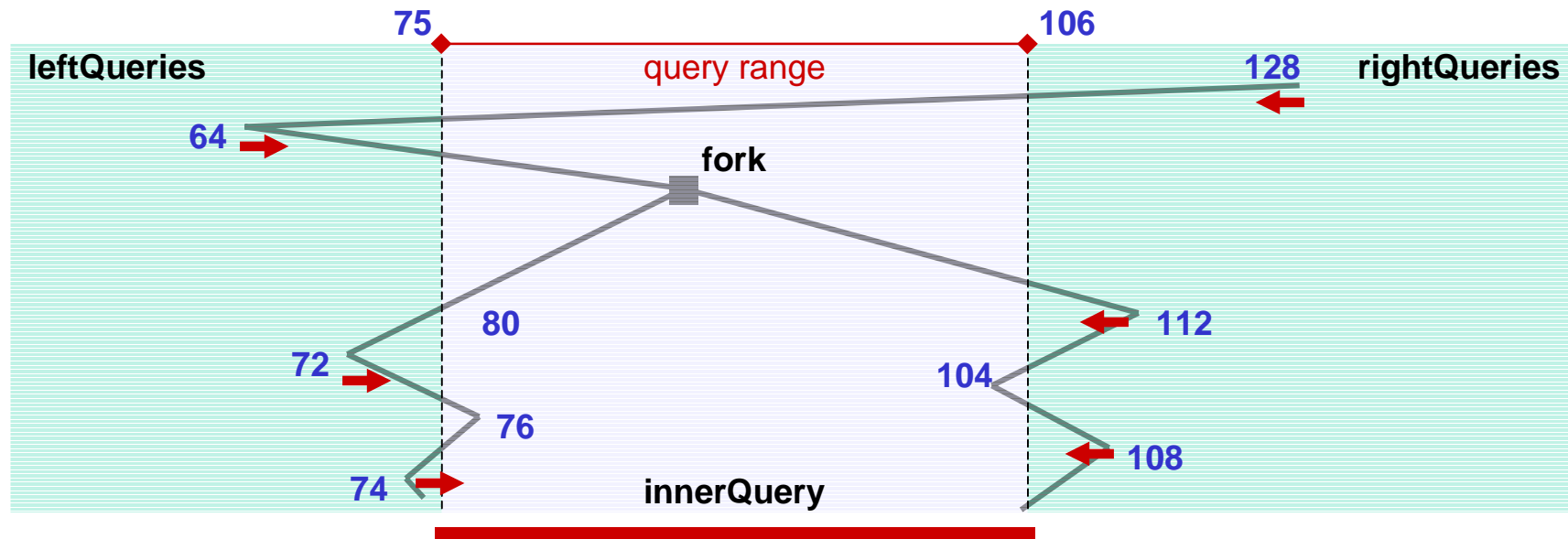


- relational storage by built-in B+-trees:

	<u>node</u>	<u>lower</u>	<u>id</u>
	4	1	b
	8	3	a
lowerIndex	8	5	c
(node,lower,id)	13	13	d

	<u>node</u>	<u>upper</u>	<u>id</u>
	4	7	b
	8	12	c
upperIndex	8	15	a
(node,upper,id)	13	13	d

Interval Intersection Query



```
SELECT id FROM upperIndex i, :leftQueries q  
  WHERE i.node = q.node AND i.upper >= :lower
```

UNION ALL

```
SELECT id FROM lowerIndex /*or upperIndex*/ i  
  WHERE i.node BETWEEN :lower AND :upper
```

UNION ALL

```
SELECT id FROM lowerIndex i, :rightQueries q  
  WHERE i.node = q.node AND i.lower <= :upper
```

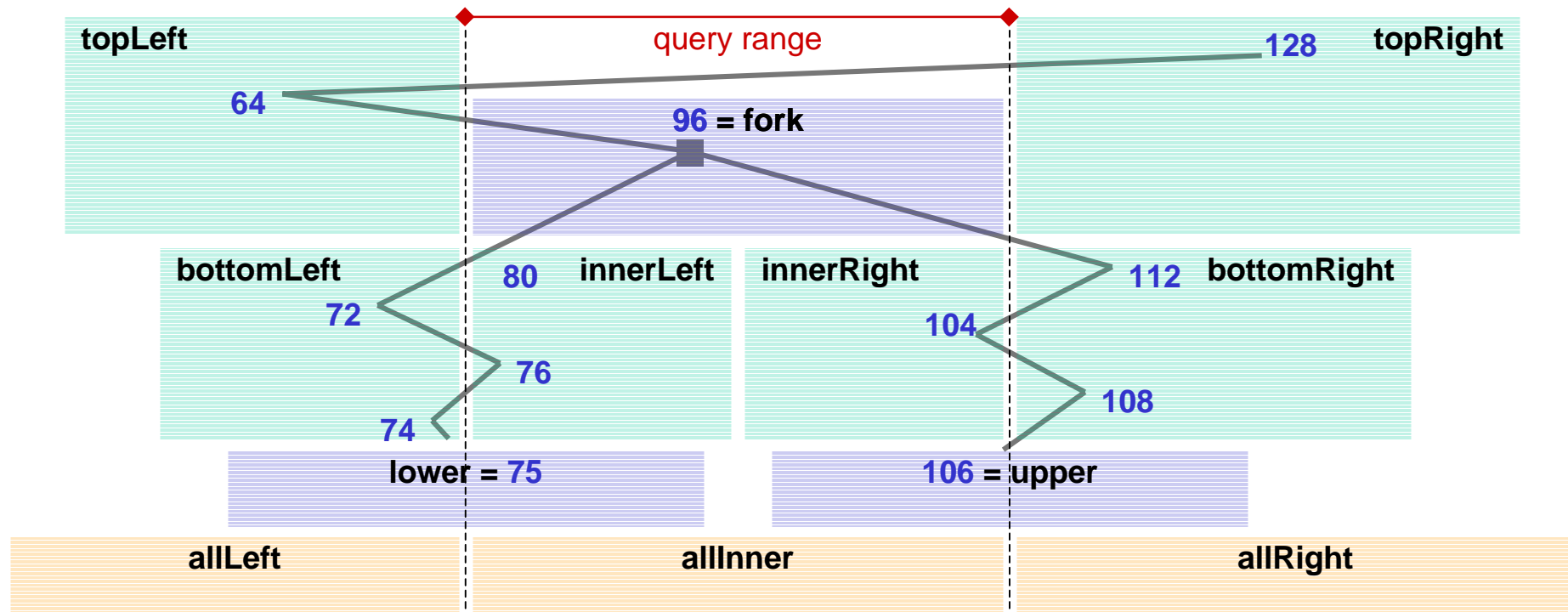
Three query classes:

leftQueries {64, 72, 74}

innerQuery [75-106]

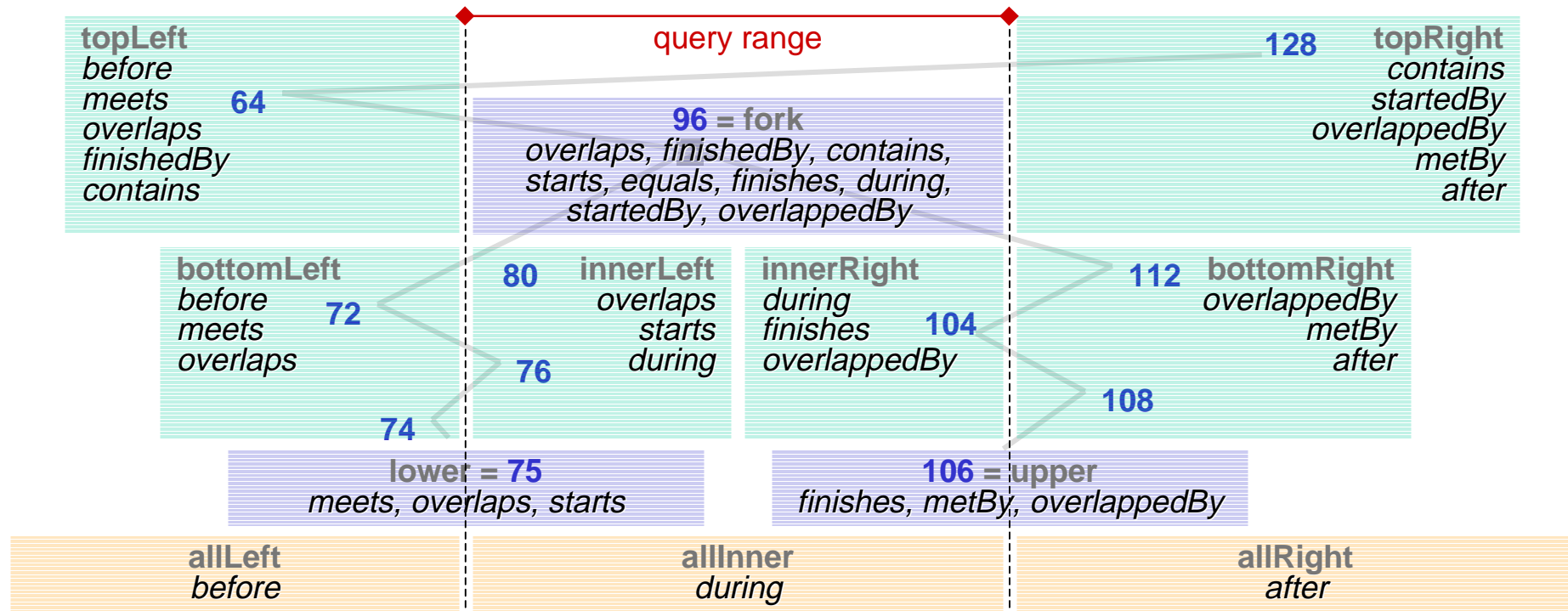
rightQueries {108, 112, 128}

Twelve Node Classes



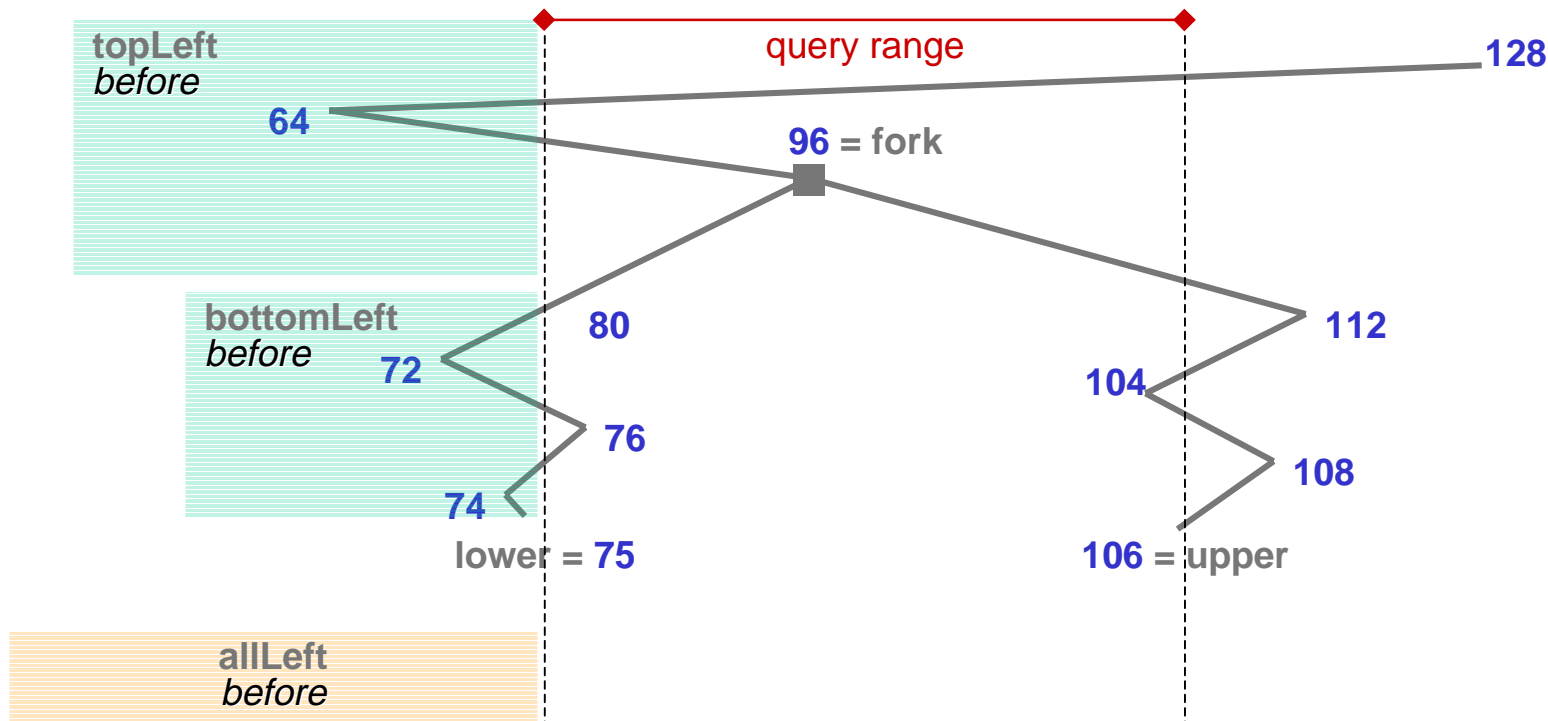
- **Traversal Classes** ($O(h)$ nodes): **topLeft, bottomLeft, innerLeft topRight, ...**
- **Singleton Classes** (single nodes): **lower, fork, upper**
- **Range Classes** (range of nodes): **allLeft, allInner, allRight**

Contributions to Relationships



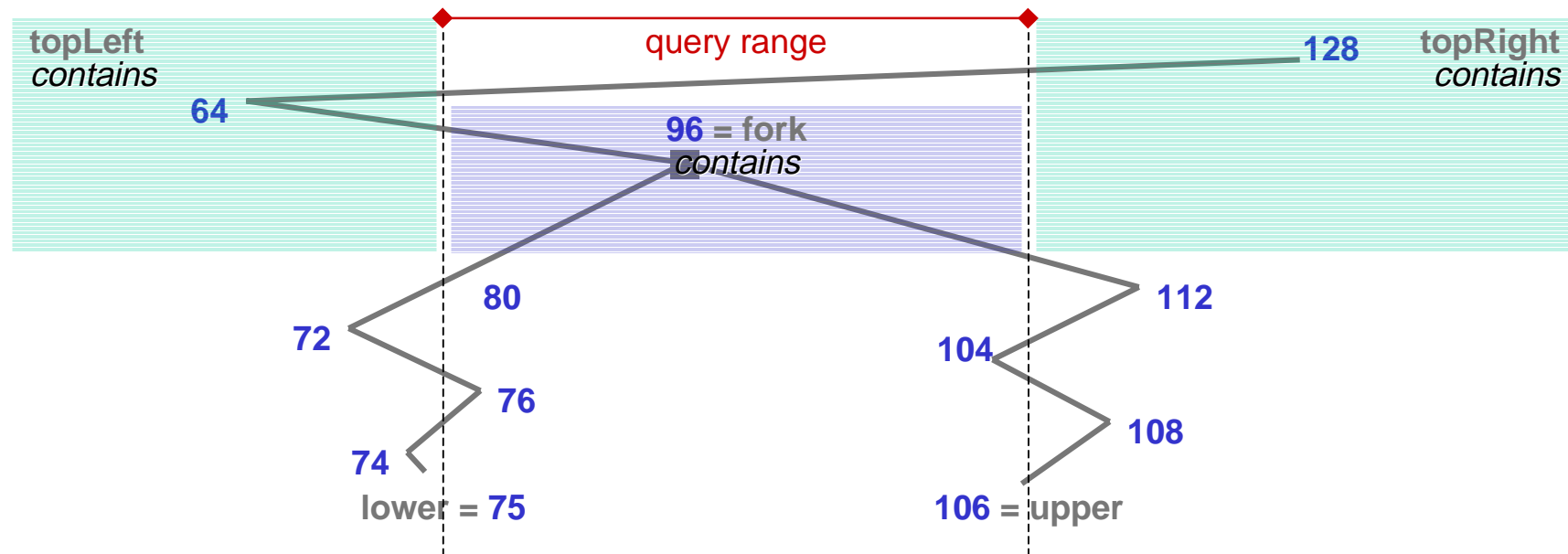
- **Traversal Classes** ($O(h)$ nodes): topLeft, bottomLeft, innerLeft topRight, ...
- **Singleton Classes** (single nodes): lower, fork, upper
- **Range Classes** (range of nodes): allLeft, allInner, allRight

'Closing Query Gaps': *before*



- report all but a few entries from nodes before **lower** (**allLeft**)
- **with gaps:** skip false entries from **topLeft** and **bottomLeft** (**multiple ranges**)
- **idea:** close the $O(h)$ gaps (**single range**), only small scan overhead
`SELECT id FROM upperIndex i WHERE i.node < :lower AND i.upper < :lower`

'Extending Indexes': *contains*



- $O(h)$ range queries with *blocked output* from **topLeft** and **topRight**
- for entries at **fork**, test both bounds **lower** and **upper** → **join!**
- **idea:** extend indexes by opposite interval bounds:
 - *lowerUpperIndex* (node, lower, upper, id)
 - *upperLowerIndex* (node, upper, lower, id)
- **no join required:** ... WHERE i.node = :fork AND i.lower < :lower AND i.upper > :upper

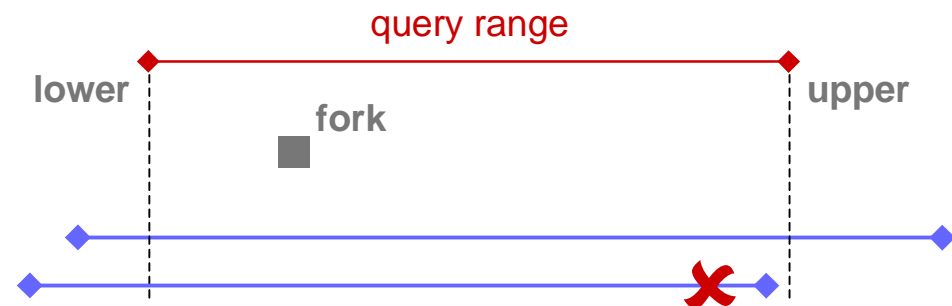
Fork Node Scan: *contains*



- **question:** *which* extended index should we use for the fork node scan?

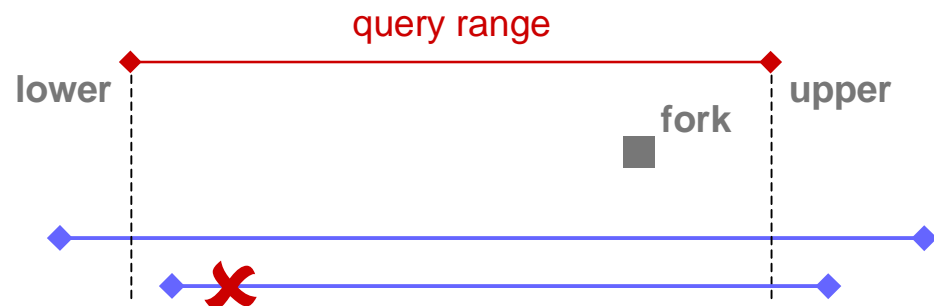
- $|fork - lower| < |fork - upper|$:

upper bounds are more selective
→ use *upperLowerIndex*

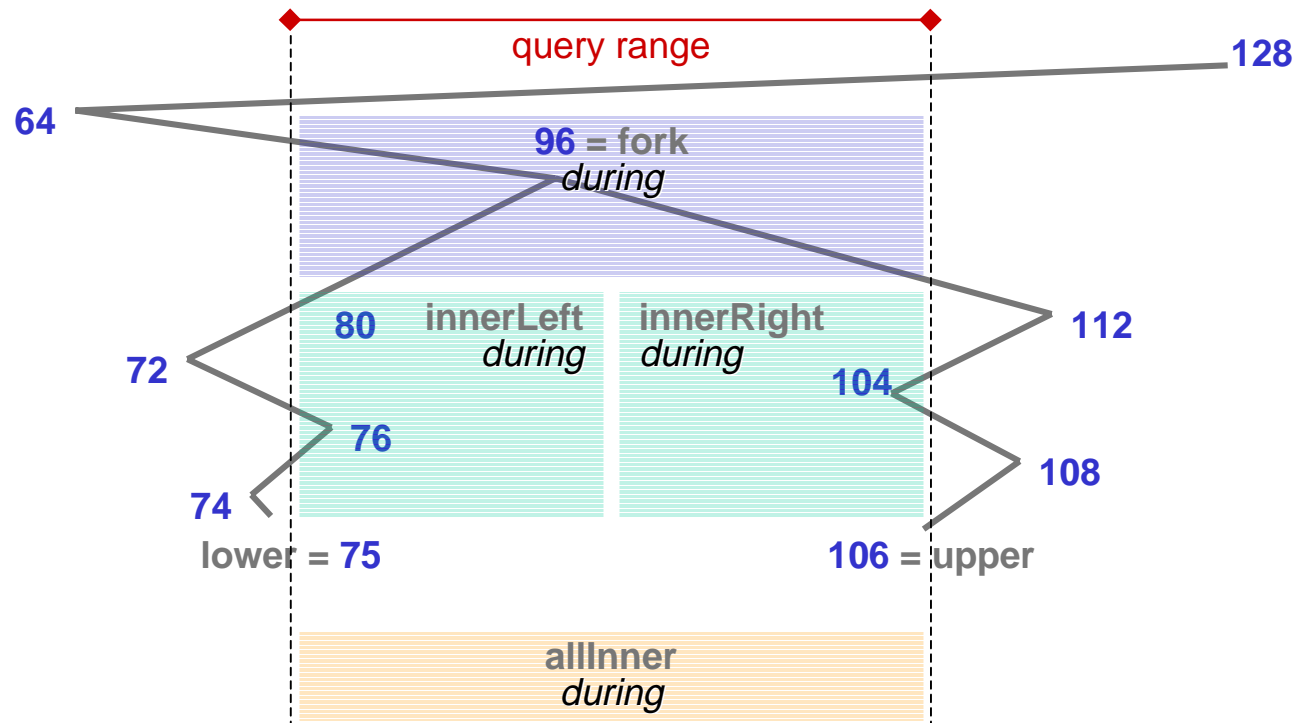


- $|fork - lower| > |fork - upper|$:

lower bounds are more selective
→ use *lowerUpperIndex*



Another Example: *during*

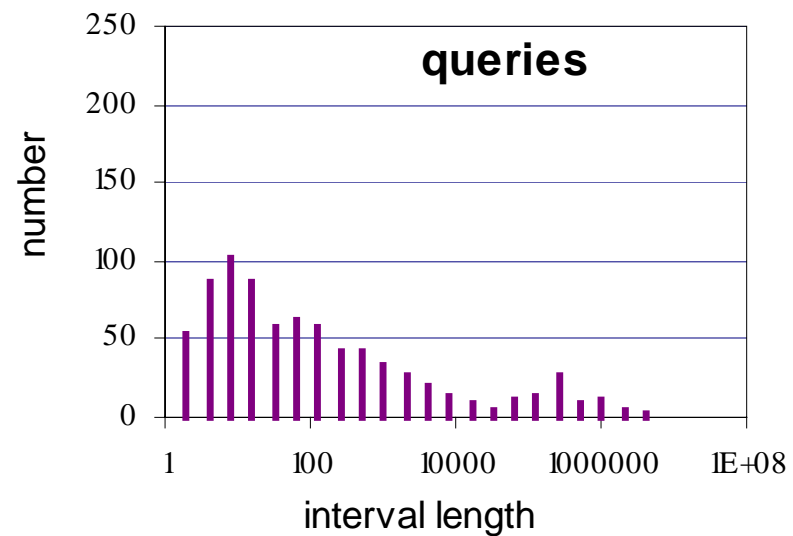
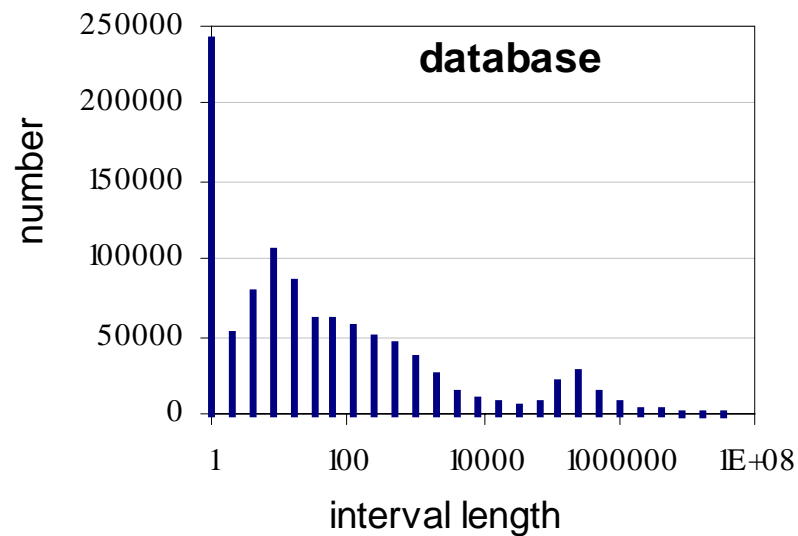


- all entries from nodes between **lower** and **upper** (**allInner**) except some entries from **innerLeft**, **innerRight**, **fork**
- **again:** $O(h)$ closed gaps yields a **single range** but only a **small overhead**
- **again:** **extended indexes** prevent from expensive join processing at fork

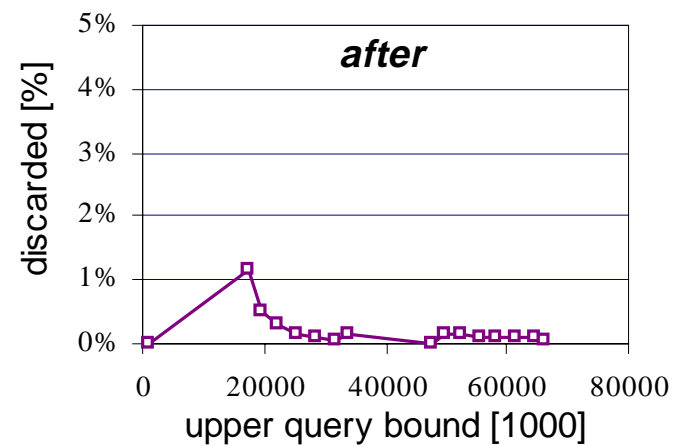
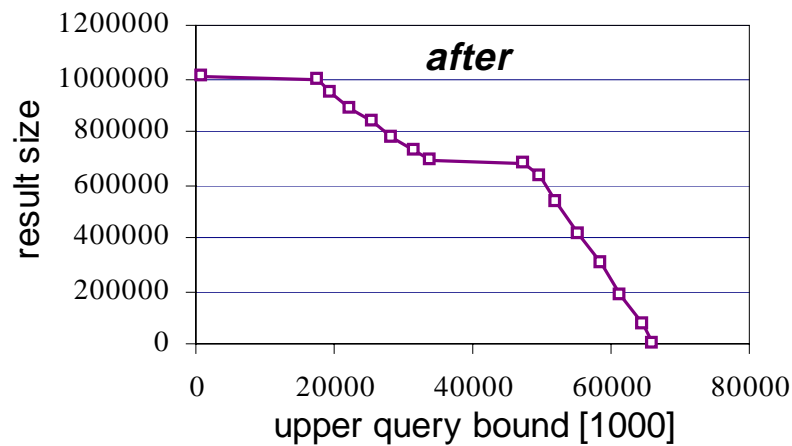
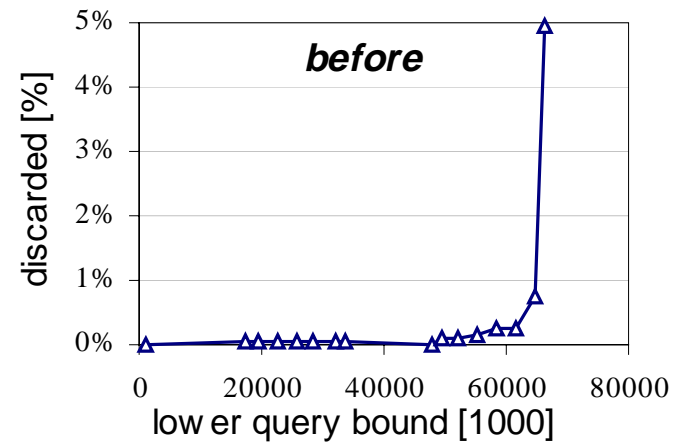
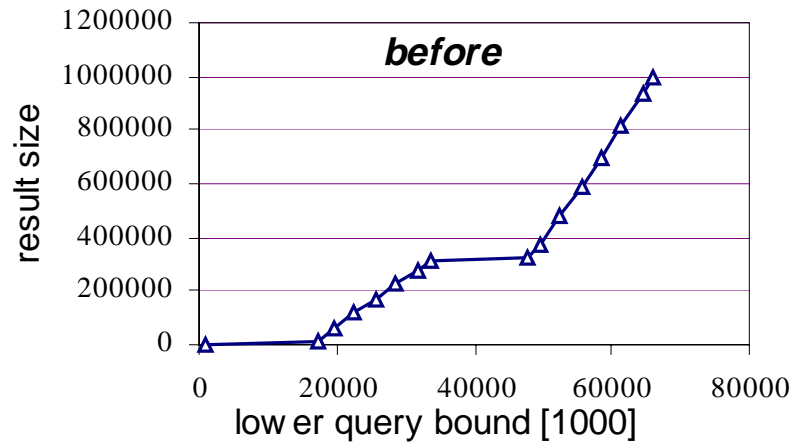
Experimental Evaluation



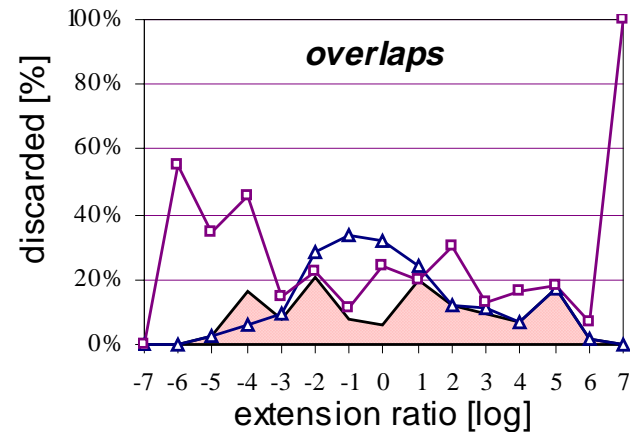
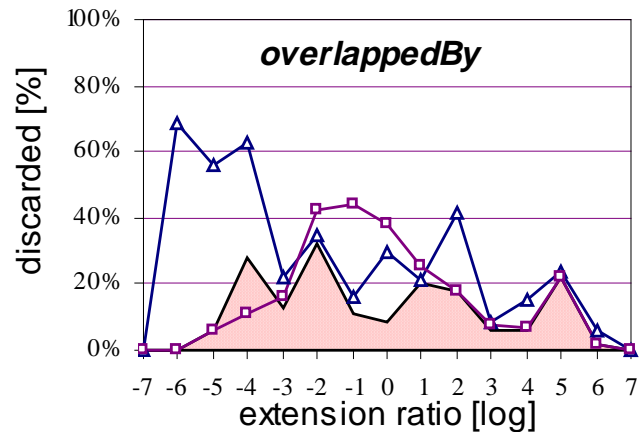
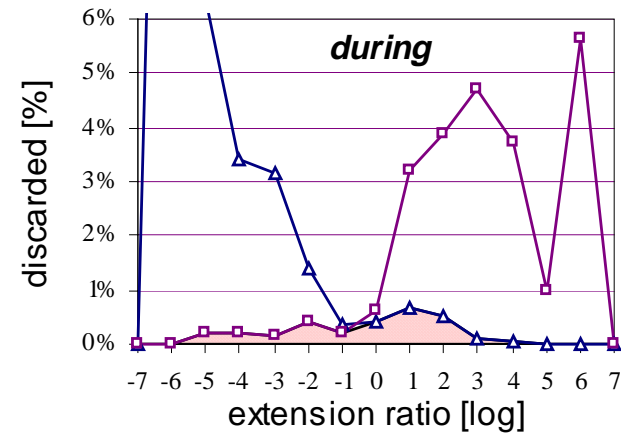
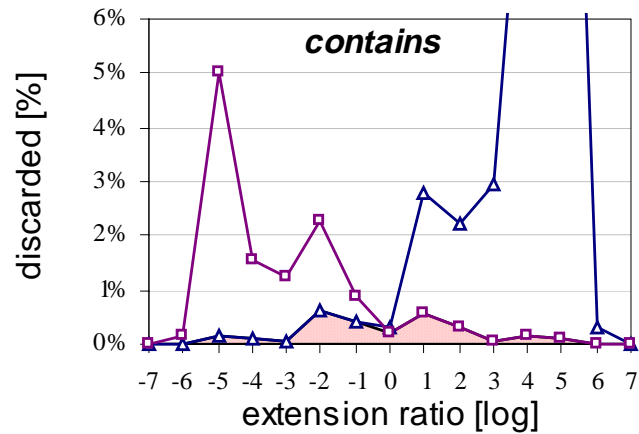
- experiments on an Oracle8i server
- 1,000,000 session periods from weblog click-streams
- characteristics of data and queries:



Discarded Entries (allLeft, allRight)



Discarded Entries (at the fork node)



optimized
 fork query on upperLowerIndex
 fork query on lowerUpperIndex

SQL:1999 Relationships



[ANSI/ISO/IEC: 9075-1999]

p **OVERLAPS** $q \equiv \text{overlaps}(p,q) \vee \text{overlappedBy}(p,q) \vee$
 $\text{starts}(p,q) \vee \text{startedBy}(p,q) \vee$
 $\text{finishes}(p,q) \vee \text{finishedBy}(p,q) \vee$
 $\text{contains}(p,q) \vee \text{during}(p,q) \vee$
 $\text{equals}(p,q)$

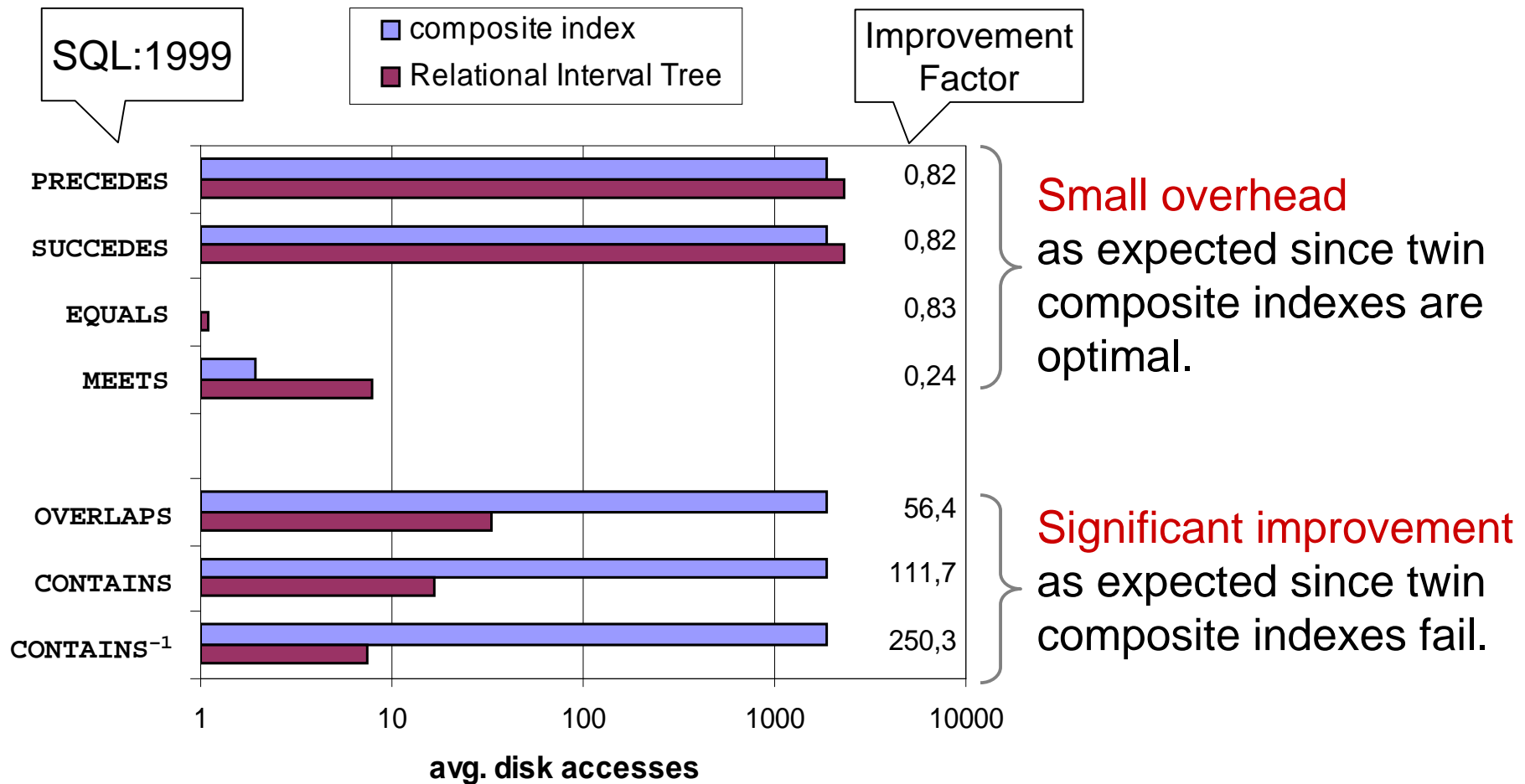
p **PRECEDES** $q \equiv \text{before}(p,q)$

p **SUCCEEDS** $q \equiv \text{after}(p,q)$

p **MEETS** $q \equiv \text{meets}(p,q) \vee \text{metBy}(p,q)$

p **CONTAINS** $q \equiv \text{contains}(p,q)$

Comparison to Composite Indexes





- Support for general interval relationships
 - original 13 relationships from Allen
 - derived 7 relationships from SQL:1999
- Extension of Relational Interval Tree
 - fine-grained distinction of node classes
 - single range scans by closing gaps
 - extended relational indexes to avoid joins
- Good Performance
 - experimental evaluation on 1,000,000 weblog sessions
 - improvement factor of up to 250 (**CONTAINS**)



Still questions?

