Formal Concept Analysis & Semantic Filesystems www.libferris.com

Ben Martin monkeyiq@users.sf.net

Synoptic

- What is SFS
- And what is FCA. . . maths boy?
- Why combine these
- Complexity issues
- Indexing, getting away with it!

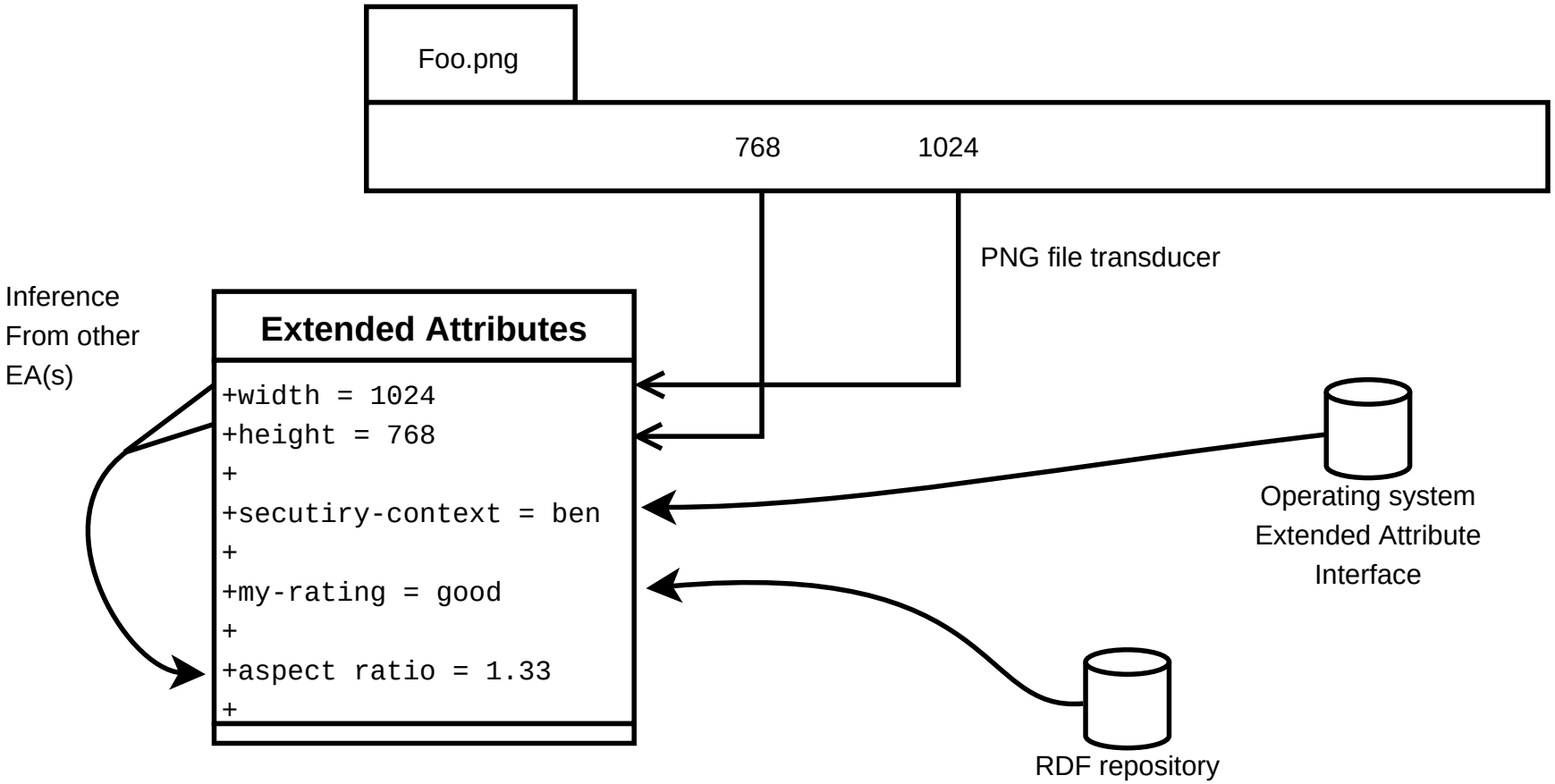
What is SFS - Semantic FileSystems?

- The notion of a semantic filesystem was originally published by David K. Gifford et al. in 1991.
- Virtual Filesystem (VFS)
- Metadata as Key-Value pairs on Files/Dirs
- Metadata extraction
- Index and Search
- Search results as a VFS

Novelties of libferris

- Metadata as RDF with Smushing
- Metadata tracking and single interface
- User address space - pg, xml, Firefox et al.
- Pluggable VFS, Metadata, Index, Creation
- POSet tagging & geotagging
- SVM support
- VFS = VXML = VFS

Novelties of libferris



Enough waffling about 'ferris

- What about this FCA thing then?

FCA - Formal Concept Analysis

- UML
- Natural clustering
- Formally defined by Bernhard Ganter & Rudolf Wille
- Initially mathematician oriented research field

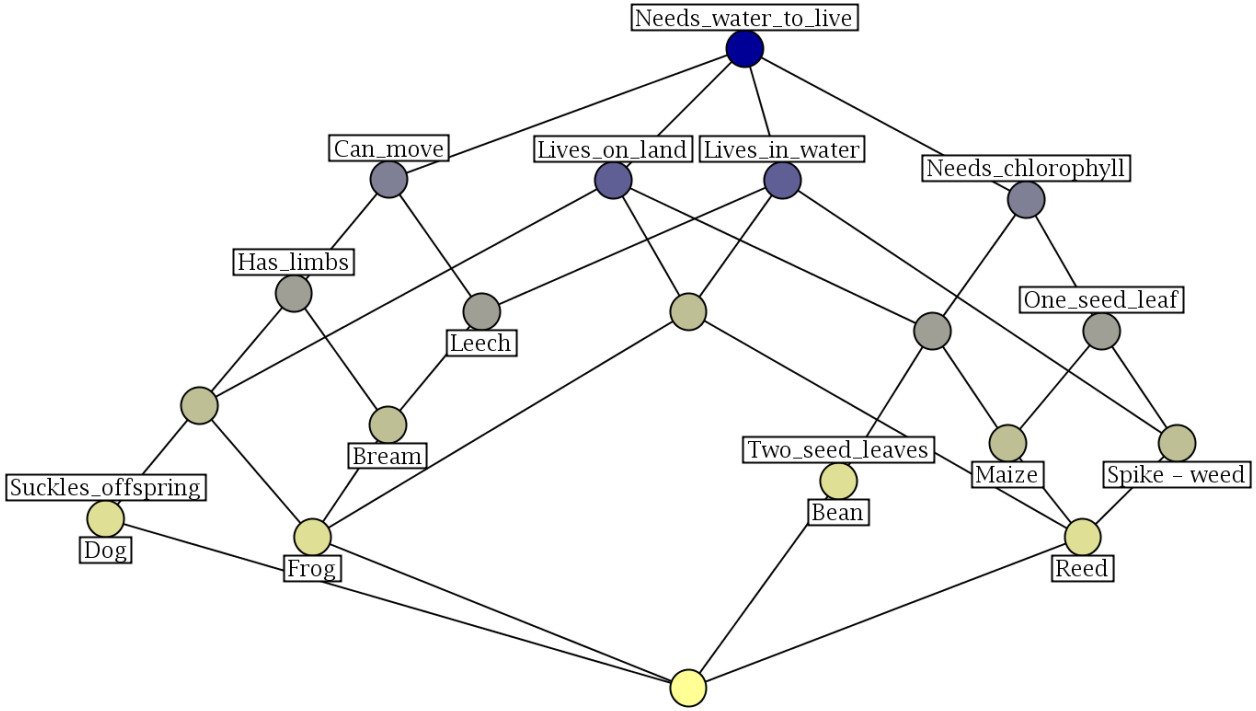
FCA - In a nutshell

- Set of Objects, Set of Attributes
- Objects may / may not have attributes
- Galois connection - binary relation between two sets
- Minimal Lattice to capture relation

Creatures!

	Needs water to live	Lives in water	Lives on land	needs chlorophyll	two seed leaves	one seed leaf	can move	has limbs	suckles offspring
Leech	×	×					×		
Bream	×	×					×	×	
Frog	×	×	×				×	×	
Dog	×		×				×	×	×
Spike - weed	×	×		×		×			
Reed	×	×	×	×		×			
Bean	×		×	×	×				
Maize	×		×	×		×			

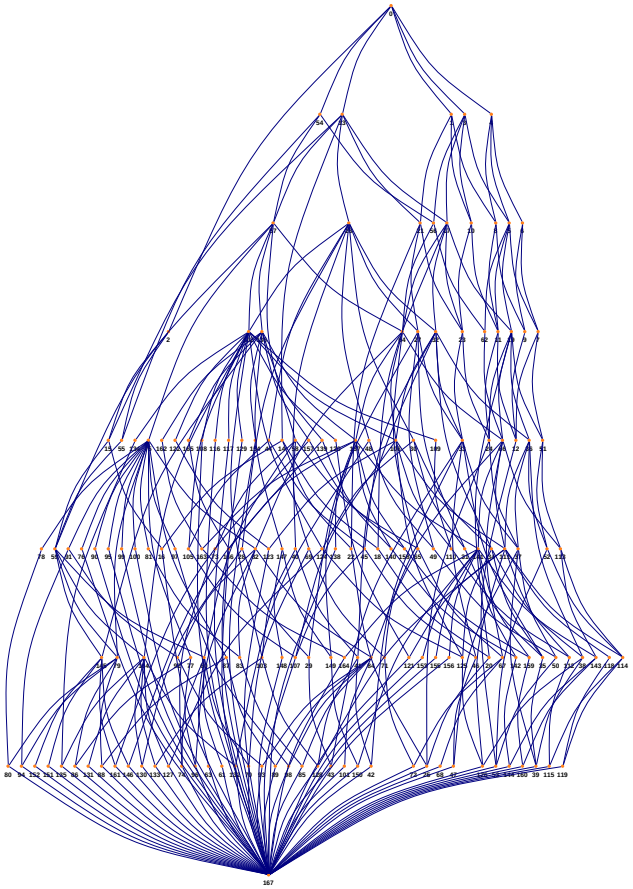
Creatures and FCA



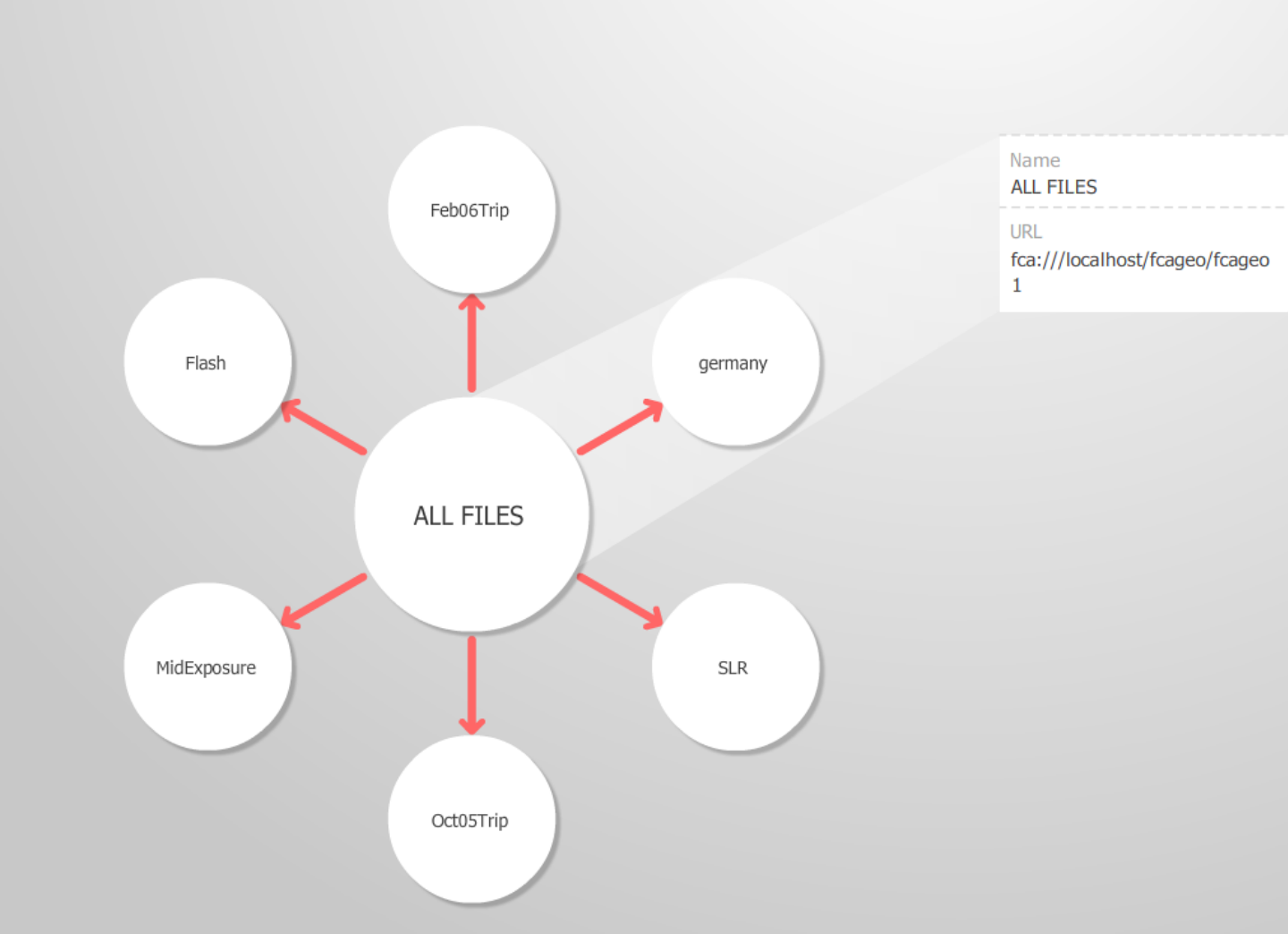
Why combine SFS and FCA

- FCA is formally defensible
- Conventional "desktop search" has limitations
  - Over-specified query
  - Search vs. Navigation
  - Ordered single containment hierarchy

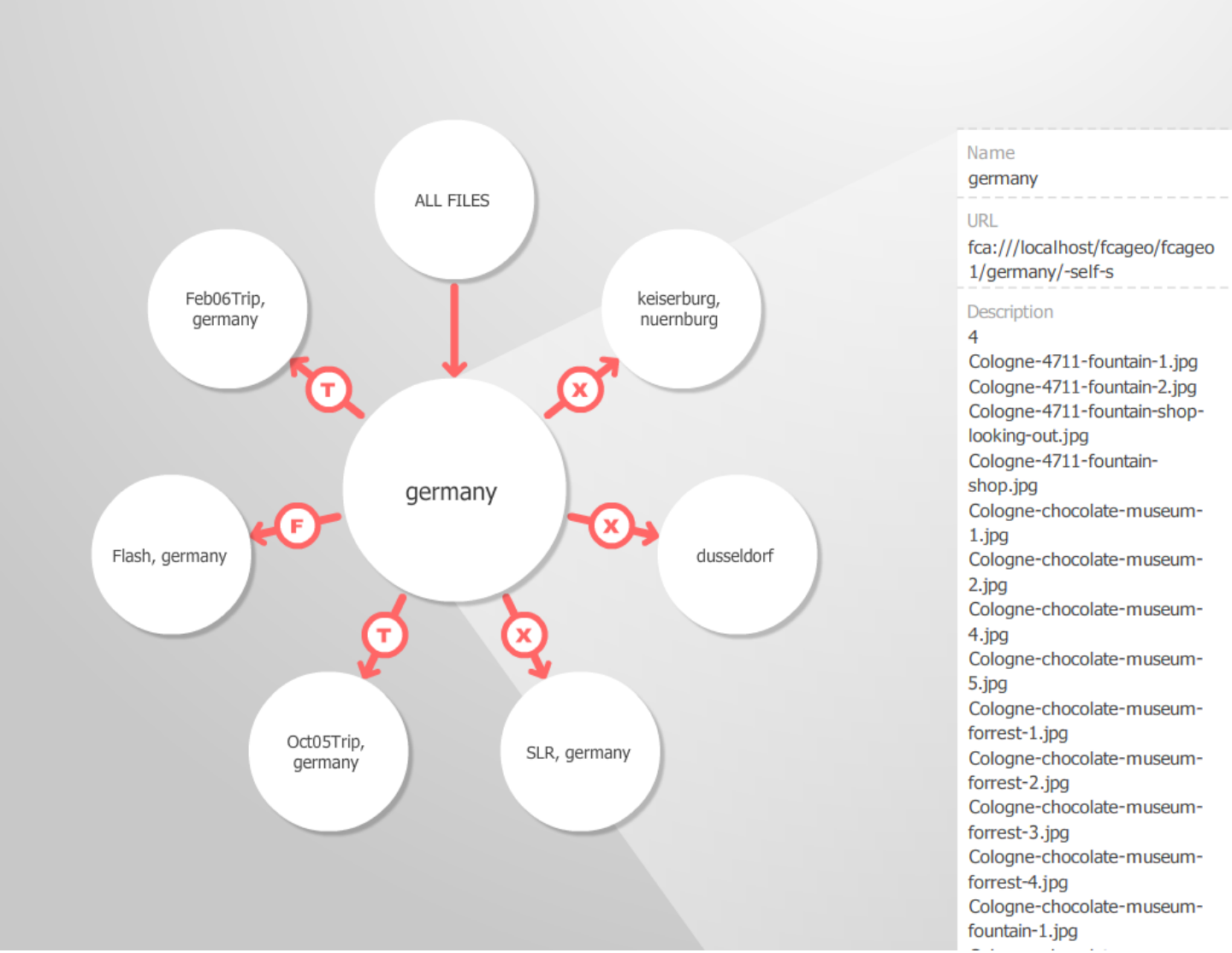
Lattice as the Interface?



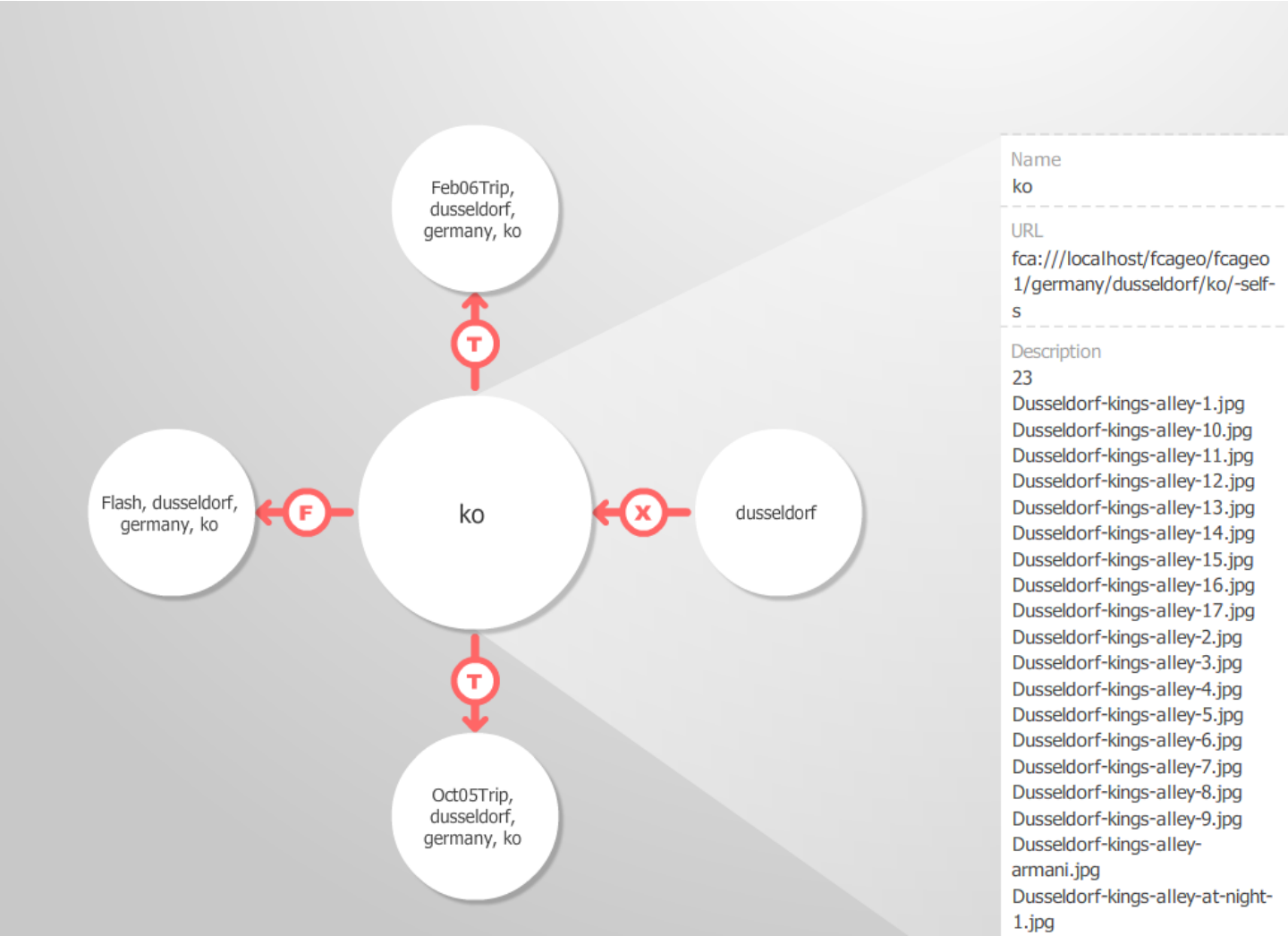
Lattice as the Context



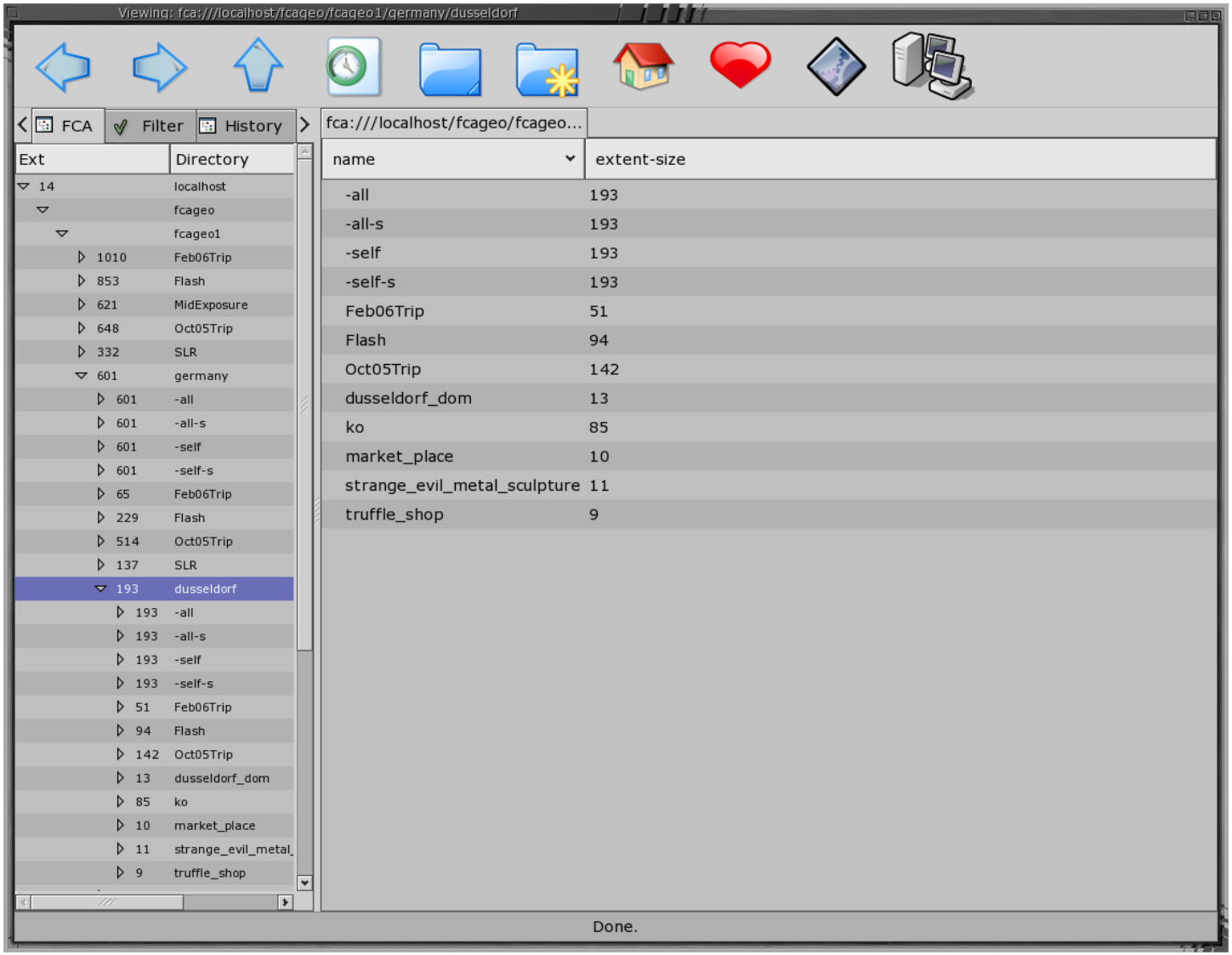
Lattice as the Context



Lattice as the Context



Lattice as Filesystem



Issues

- Binary Relation → Lattice
  - Nodes - Naive implementation 2^n
  - Edges - Naive implementation C^2
- Files matching a node in the Lattice
  ...Similar to a multitag search
- DS&A
-

Conventional Indexing

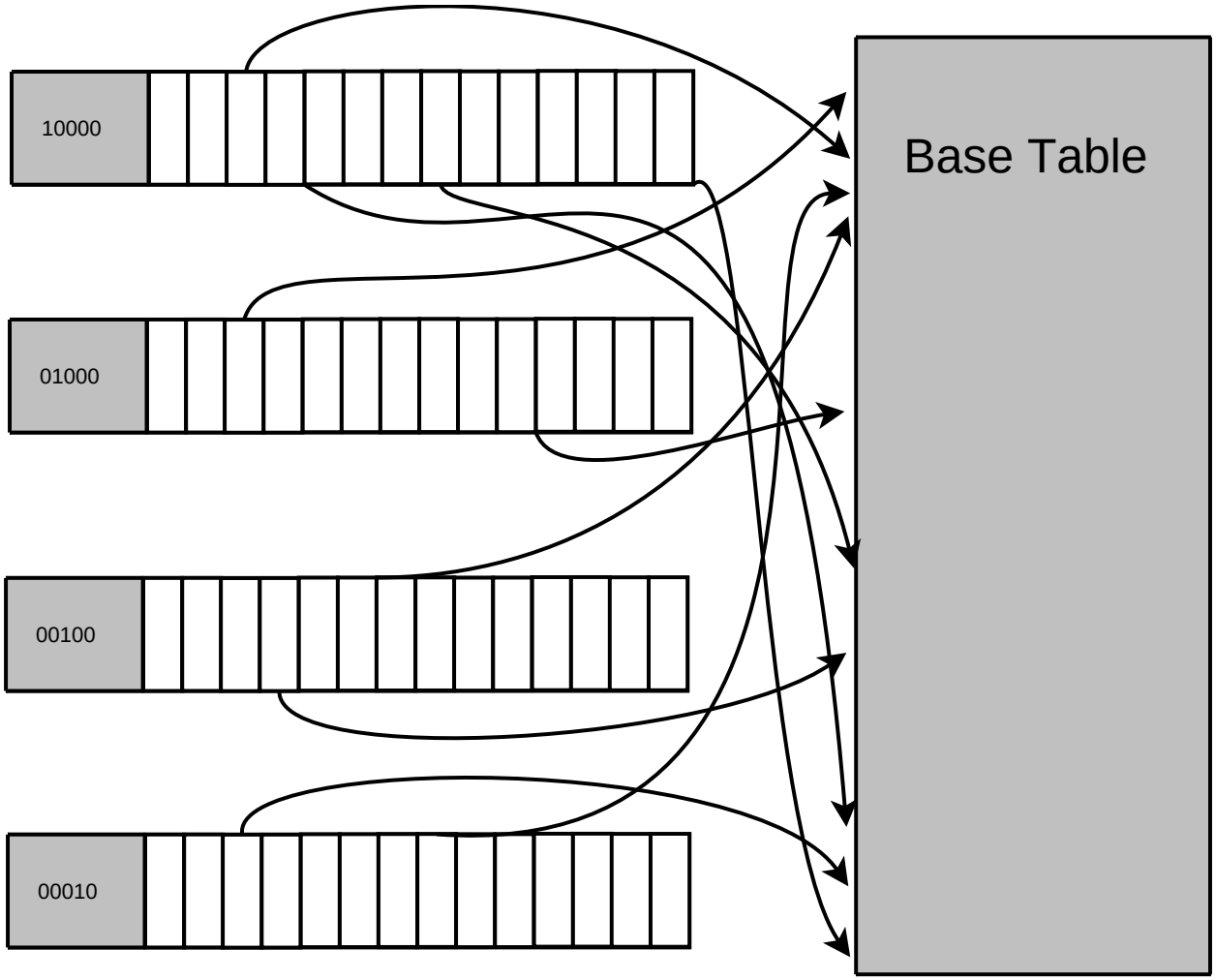
- Inverted File
- B-Tree and Hash

ineffective

Searching, What for?

Attribute	Columns involved	SQL predicate (f_x)
a_1	c_1	*size* $<=$ *4096*
a_2	c_1	*size* $<=$ *1Mb*
a_3	c_2	*modified* $<=$ *this week*
a_4	c_2	*modified* $<=$ *yesterday*
a_5	c_2	*modified* $<=$ *today*
a_6	c_3	*accessed* $<=$ *last week*
a_7	c_3	*accessed* $<=$ *yesterday*
a_8	c_3	*accessed* $<=$ *today*
a_9	c_4	*file-owner* $=$ *ben*
a_{10}	c_4	*file-owner* $=$ *peter*
a_{11}	c_4	*file-owner* $=$ *foo*
...	...	...

Inverted File

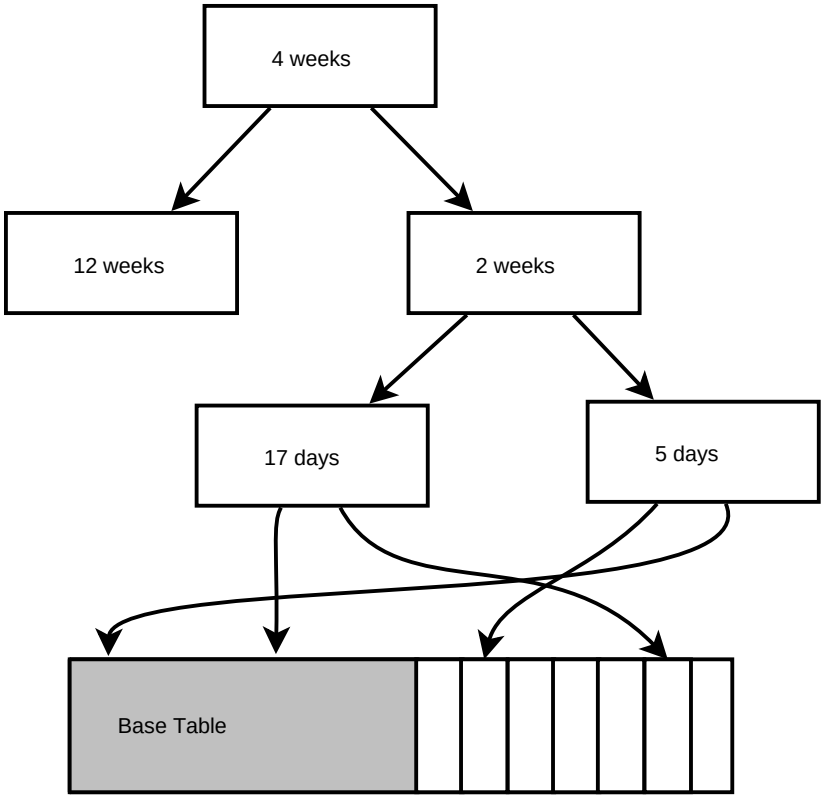


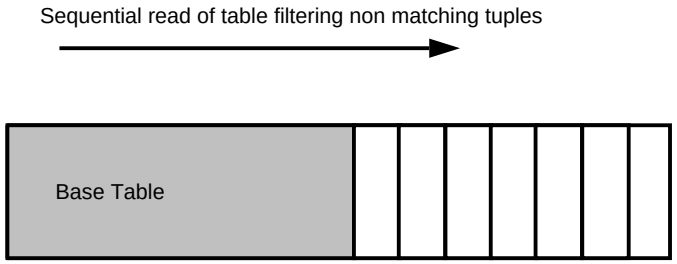
B-Tree or Hash

The numbers game...

- The estimated ratio of matching tuples → *selectivity*.
- 100×estimated tuple count/size of base table

B-Tree or Hash





Two or more predicates

- Predicate with best selectivity & index chosen
- Others as filters on fetched tuples. Index ignored for them.

Consider this...

$$size <= 4096$$

and

$$\textit{modified} <= \textit{this week}$$

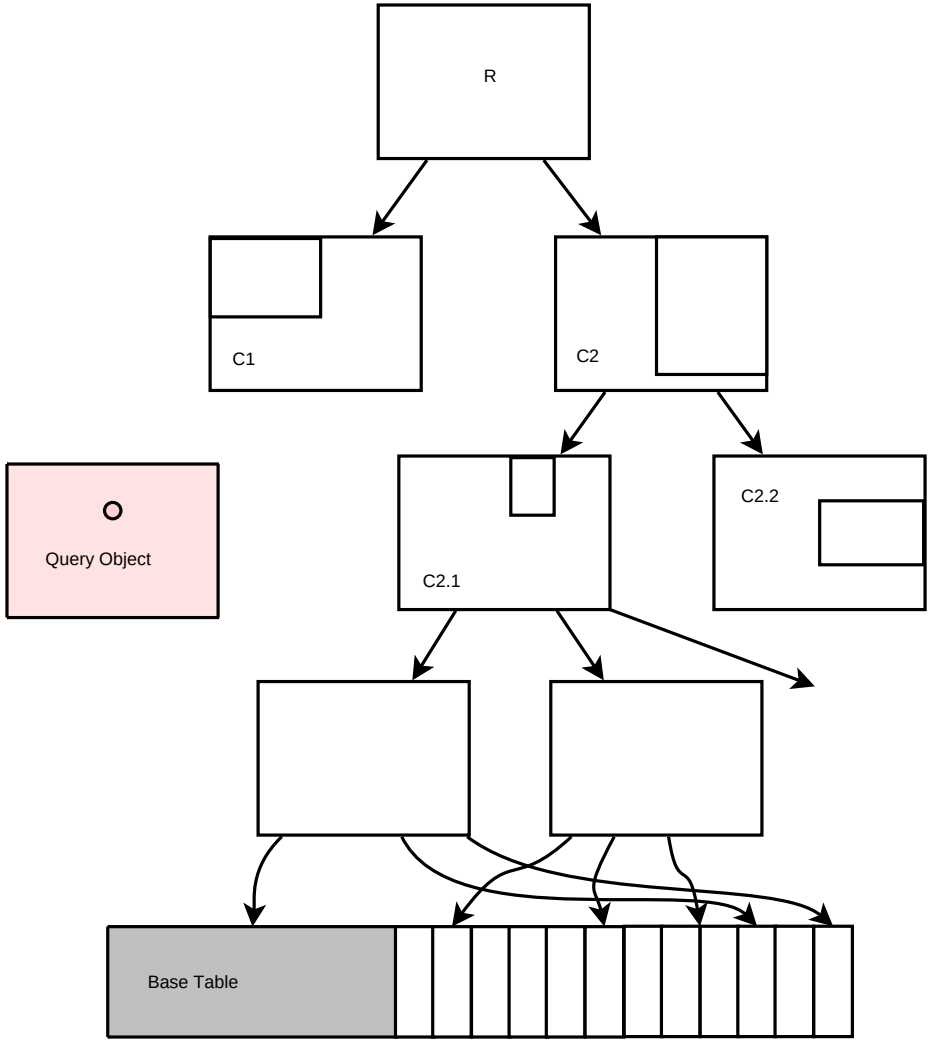
OR this...

location(Germany)
and
Camera(D90)

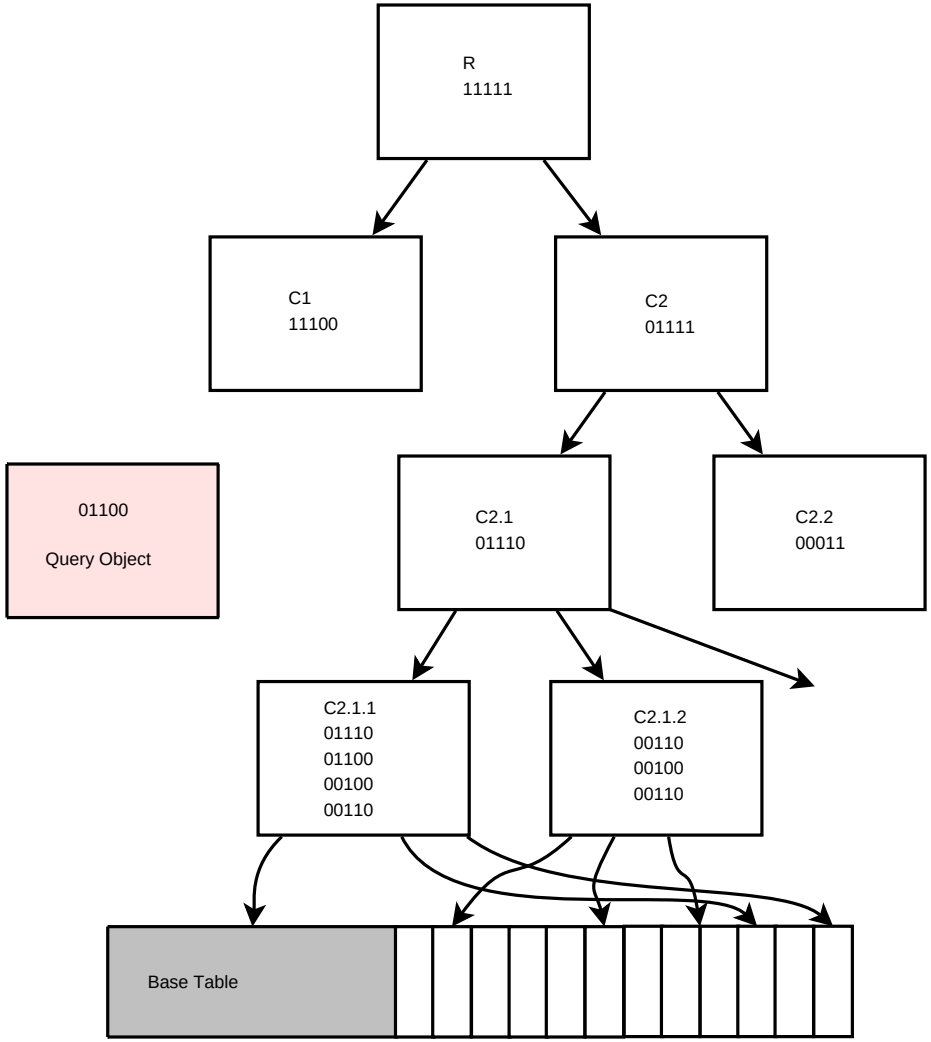
Spatial Indexing

- R-Tree based thinking
- Containment & Unification
- GiST - picksplit

R-Tree



RD-Tree


R
11111
C1
11100
C2
01111
01100
Query Object
C2.1
01110
C2.2
00011
C2.1.1
01110
01100
00100
00110
C2.1.2
00110
00100
00110
Base Table

FCA queries that Spatial Indexing helps with

- Subset Query: seeks all objects for which the query object is a subset
- Overlap Query: seeks objects which have more than a given number of attributes in common with the query

Hamming weight, overlaps and Files in a Lattice Node

Normal query	$a < 10$ and $a < 20$ and not $(a < 30)$ and not $(a < 40)$
Simple translation	rd-tree contains 10,20 and not rd-tree contains 30 ...
Custom translation	rd-tree contains 10,20 and hamming-weight(rd-tree) = 2

Empirical Testing

Time Machine Clockwork by Pierre J., April 29, 2007.

Used under creative commons share alike: http://www.flickr.com/photos/7969902@N07/476909988/

UCI Mushroom: Selectivity

Column	Value	Selectivity (count)	Selectivity (% of table)
bruises	NO	10080	59.9
bruises	BRUISES	6752	40.1
capshape	KNOBBED	1680	10.0
capshape	CONVEX	7592	45.1
capshape	FLAT	6584	39.1
capshape	BELL	904	5.4
capshape	SUNKEN	64	0.4
capshape	CONICAL	8	0.05

UCI Mushroom: Performance

Test type	Cold cache	Hot cache	Sequential scans
B-Tree only	30	18	4
RD-Tree index simple query translation	10	4	1
RD-Tree optimized query translation	1.6	0.4	0

Figure 1: Times are in seconds.

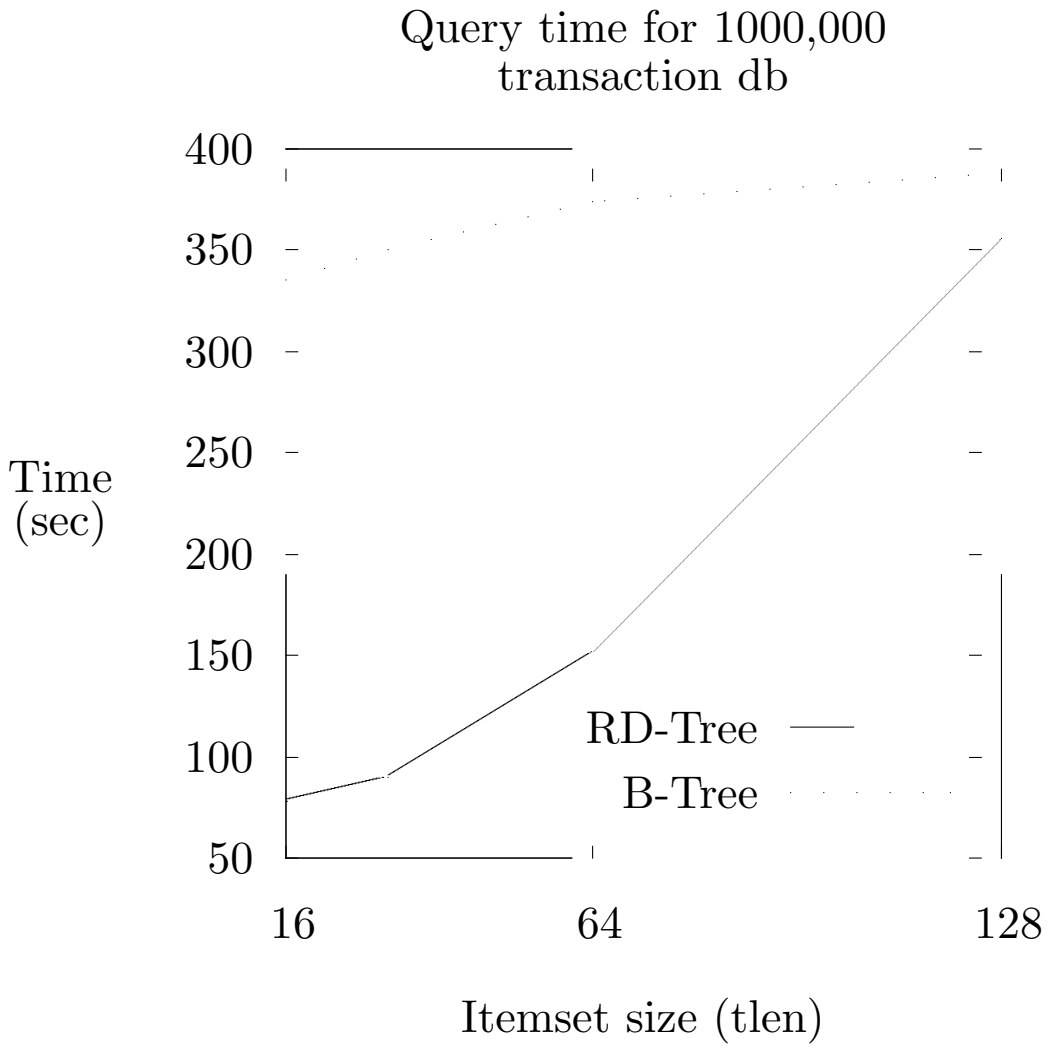
Stats for UCI covtype dataset

- 581,012 tuples
- 54 columns
- A total of 378 queries for chosen FCA application
- Primary table is 987Mb

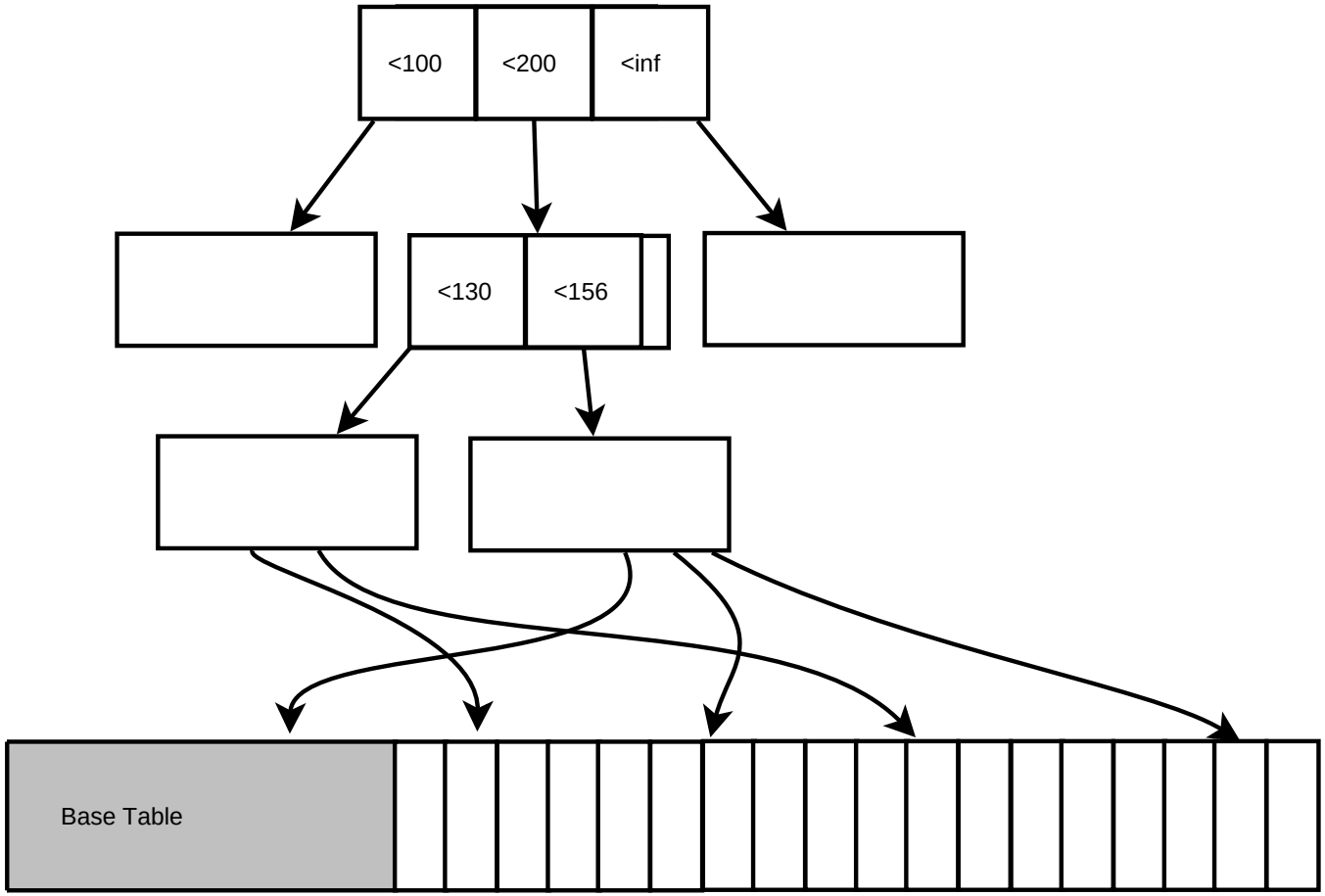
Performance on UCI covtype dataset

Test type	Cold cache (mm:sec)	Sequential scans
B-Tree only	56:16	90
RD-Tree index simple query translation	0:42	0

IBM Synthetic data generator... 1,000,000 transactions



GiST



Penalty() & Picksplit()

- Penalty: How well does this key fit the page?
- Picksplit: Break a page in two
-

Asymmetric

- Propagate smaller keys upwards
- Two ways:
  - Replace Guttman
  - Guttman followed by FCA

Replace Guttman

1. Using the **standard** RD-Tree method select the initial **left** L **and right** R sets as new parent keys.

2. For all sets yet to be distributed, preallocate any set which is a non strict subset of either parent key $\{L, R\}$.

3. For all unallocated keys

   (a) Test if the current key is a subset of either updated parent key, if so then allocate that key to the child page of the respective parent key.
   (b) Attempt to **minimize expansion** of either parent key, when expansion has to occur **prefer** to expand the **right parent's bounding set**.
   (c) If both parents would have to expand the same amount to cater for a key then distribute as per the normal RD-Tree method.

4. If either page is drastically under full shuffle keys into it from the other page. A page is under full if it is less than 10% utilized. Do not expand the under full page's bounding set by more than x elements. For our testing $x = 1$.

FCA, tree, FCA, a loop?

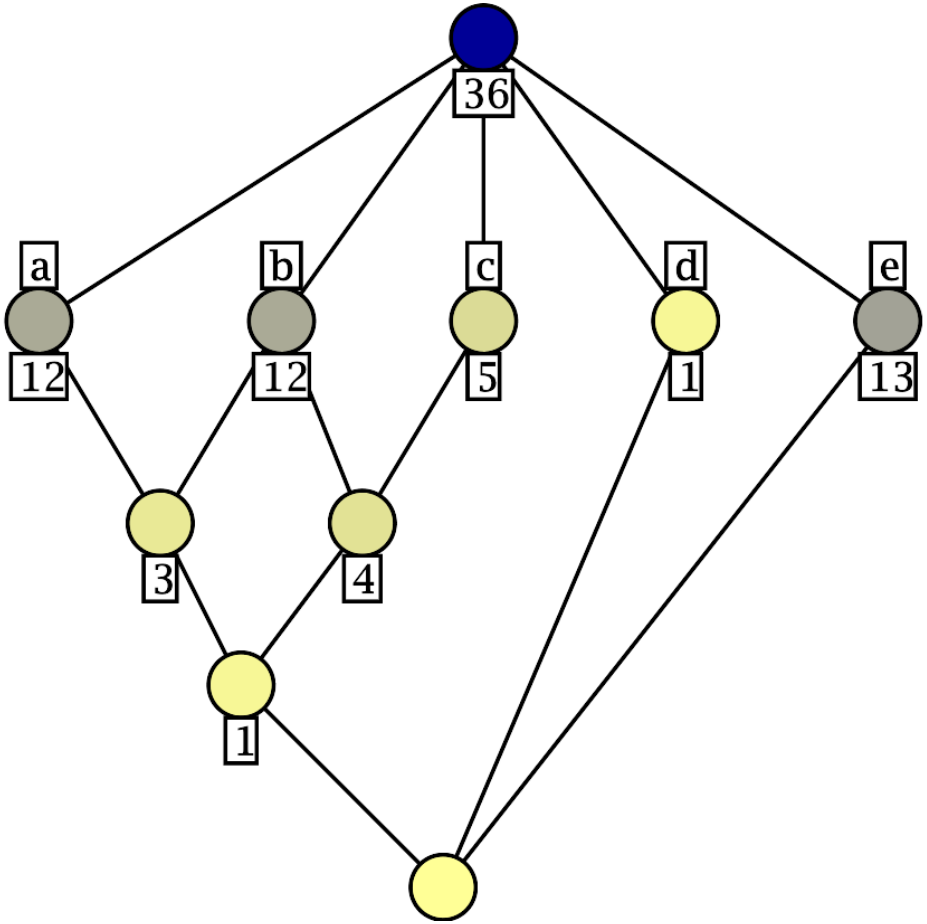


Figure 2: The numbers below the nodes indicate how many keys match that node or any connected below it.

Performance: UCI covtype

Index	tree depth	index size (Mb)	leaf node count	internal node count	mean leaf free
RD-NC	17	66.1	3883	4582	5247
Asym-NC	14	52.2	3581	3100	4799
Shuf-NC	12	47.1	3353	2680	4539
RD-Comp	7	37.7	3846	977	4946
Shuf-Comp	7	33.1	3321	915	4565
GuttFCA-30p-Comp	6	34.3	3652	741	4681
GuttFCA-50p-Comp	6	34.3	3648	739	4677
GuttFCAR-30p-Comp	6	34.3	3652	741	4681
GuttFCAR-50p-Comp	6	34.3	3648	739	4677

IBM Synth data generator

Index	tree depth	index size (Mb)	leaf node count	internal node count	mean leaf free
rd16					
`RD-Comp`	3	63.5	8033	95	3950
`Shuf-Comp`	3	64.9	8233	77	4051
`GuttFCA-30p-Comp`	3	63.9	8097	80	3982
`GuttFCA-50p-Comp`	3	64.5	8175	77	4021
rd32					
`RD-Comp`	4	67.1	8462	131	3749
`Shuf-Comp`	4	69.5	8789	103	3910
`GuttFCA-30p-Comp`	4	68.2	8620	107	3827
`GuttFCA-50p-Comp`	3	69.5	8797	100	3913

Whats left TODO?

- Wizards - All Y'all don't want to learn FCA?
- More speed improvements
- Embedded target - PG → BerkeleyDB + libsoldb
- Of course, libferris road continues
  - Mounting more + more Web
  - Extracting more
  - Improved XQuery integration
  - Ontologies, Nepomuk, RDF
  - Already on maemo, maybe other Embedded?
- Accept patches . . . Profit!?

Questions... Tomatos?

Leader of the gang (No.33) by PhotoGraham, December 1, 2006.
Used under creative commons share alike: http://www.flickr.com/photos/photograham/311323441/