Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

An Effective and Efficient Approach for Keyword-Based XML Retrieval

Xiaoguang Li, Jian Gong, Daling Wang, and Ge Yu

retold by Daryna Bronnykova

Why not use “google”?

Why are traditional search engines not suitable for searching XML docs?

There is no way to specify semantics in a keyword query executed with a traditional search engine.

Consider these queries (they share the same set of keywords):

1. Find title and year of publications, of which Mary is an author

2. Find year and author of publications with similar titles to a publication of which Mary is an author

(Queries Adapted from Li [2])

Why are traditional search engines not suitable for searching XML docs?

Traditional search engines index and return whole documents, not the specific parts of the documents.

For the query: bike + luxury

this document -> would be considered a relevant result

```
< item>
        <ID> ID0 </ID>
        <name> bike </name>
        <desc>lame</desc>
</item>
< item>
        <ID> ID1</ID>
        <name> car </name>
        <desc> luxury </desc>
</item>
```

XML document model – as a tree

tree (rooted, ordered, labeled)

edges = parent-child relationships between elements in XML document

inner nodes (elements) and leaf nodes (values)

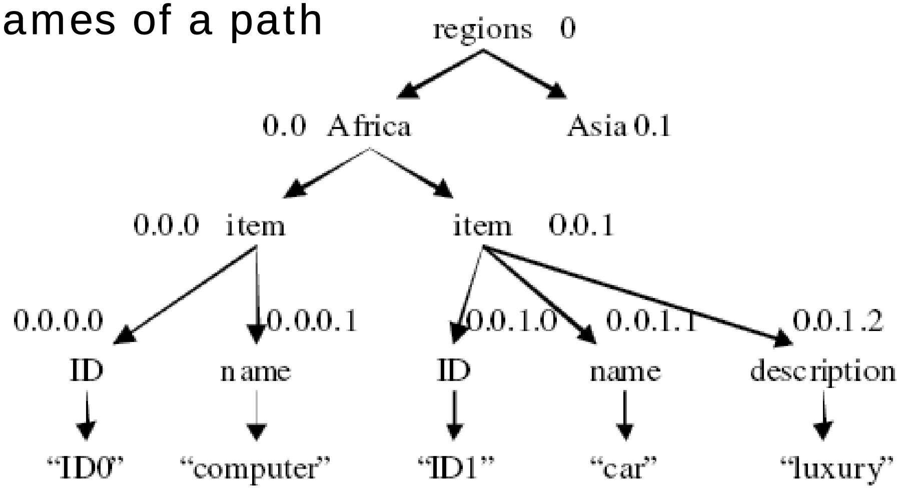
labels

path

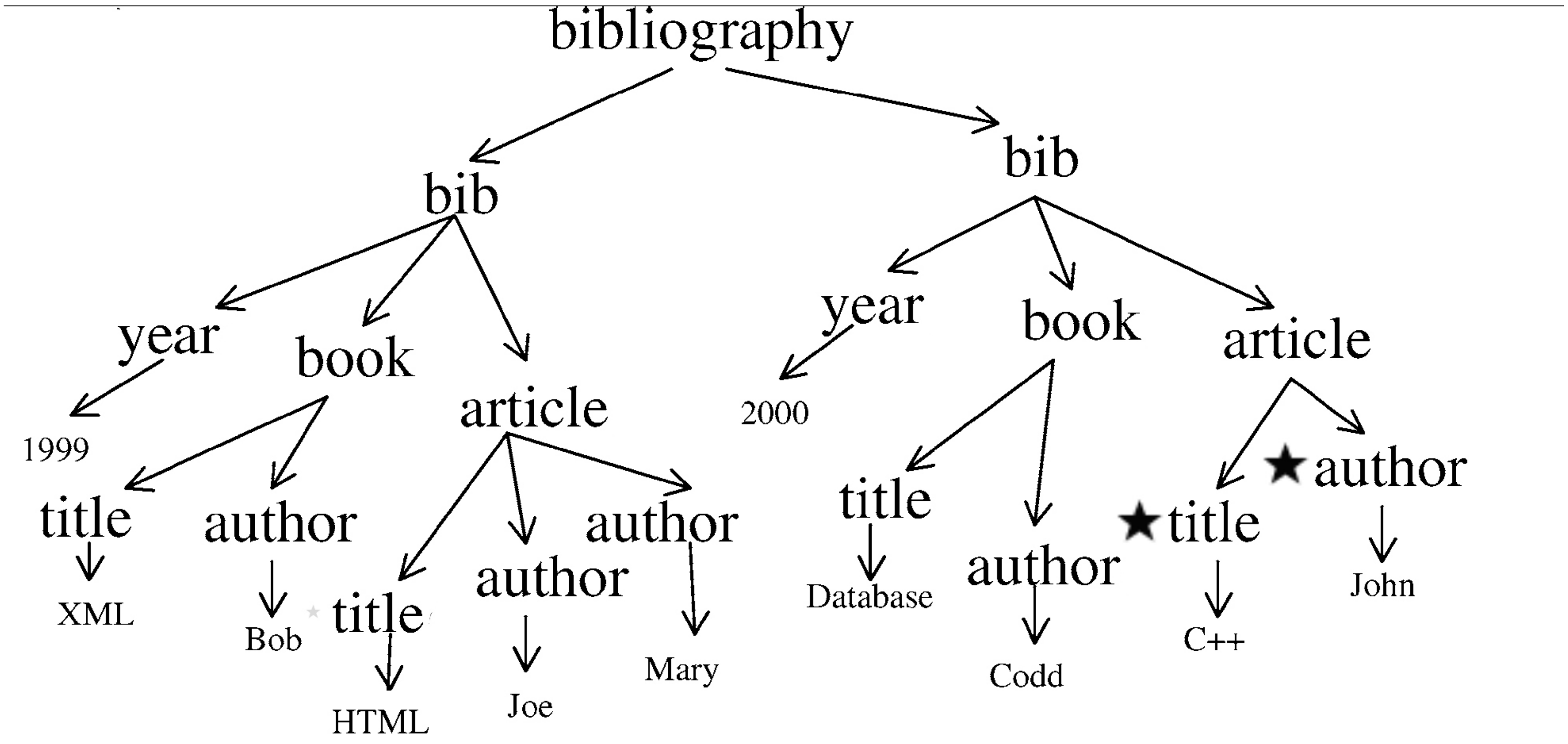
label path - a sequence of label names of a path

ancestor

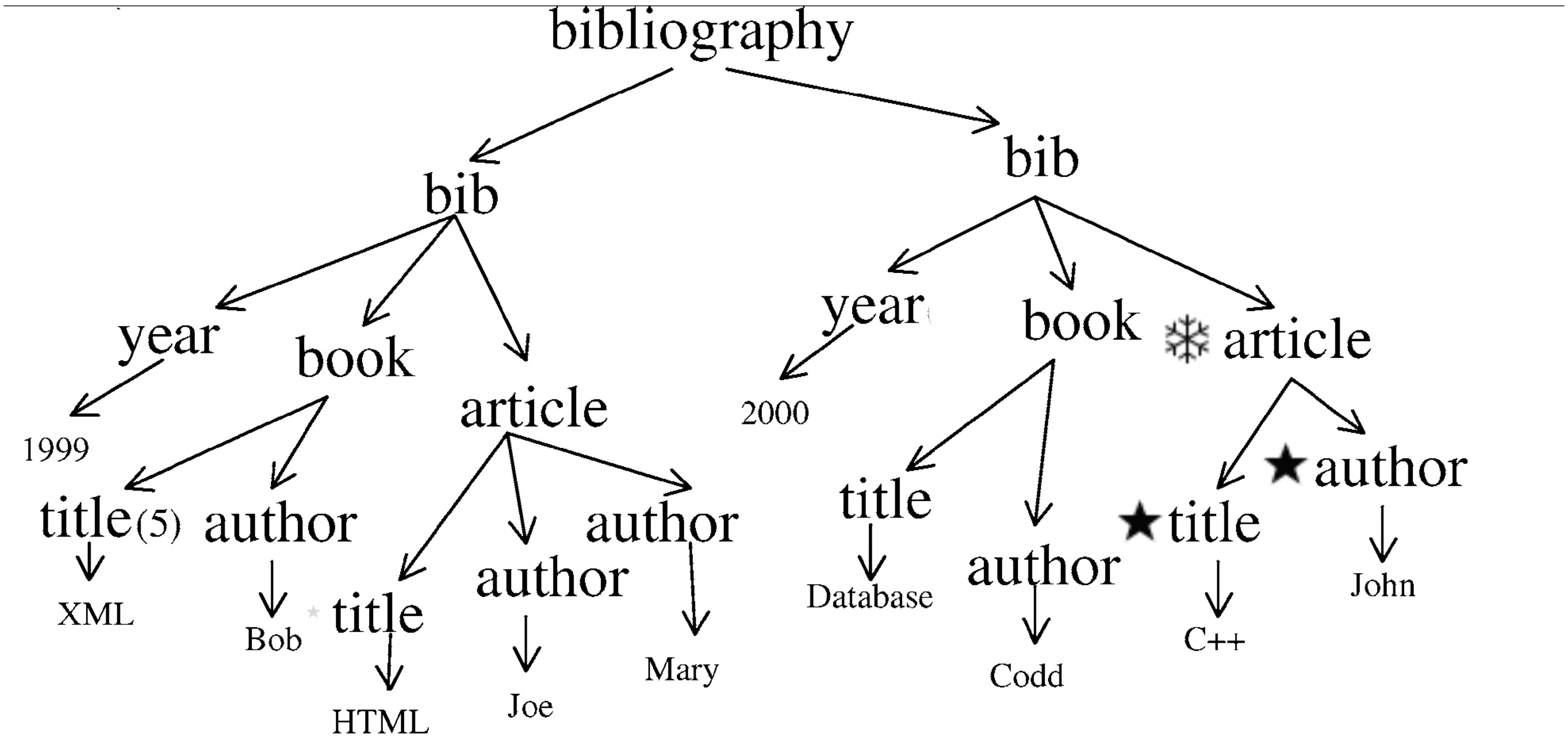
relational subtree



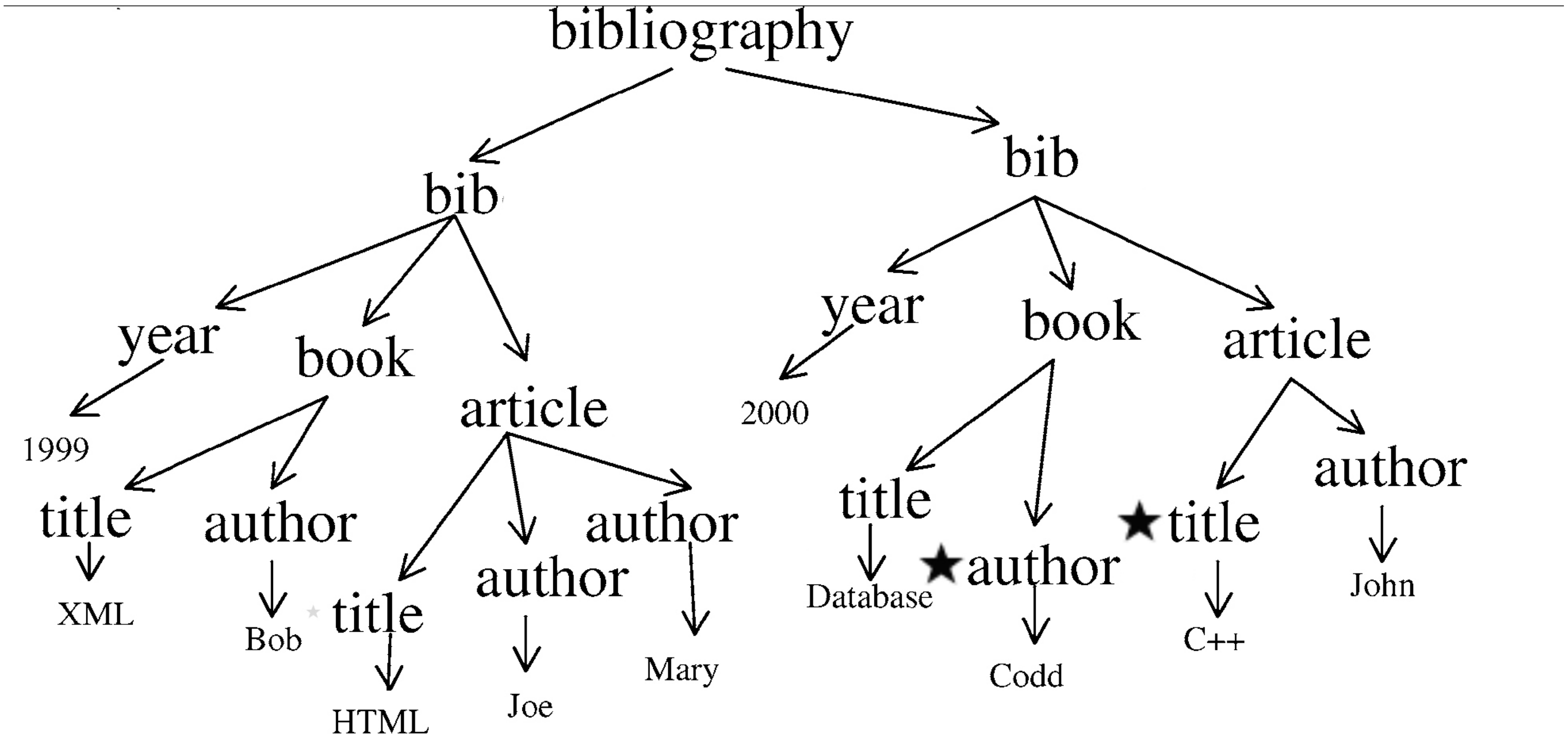
Lowest Common Ancestor (LCA)



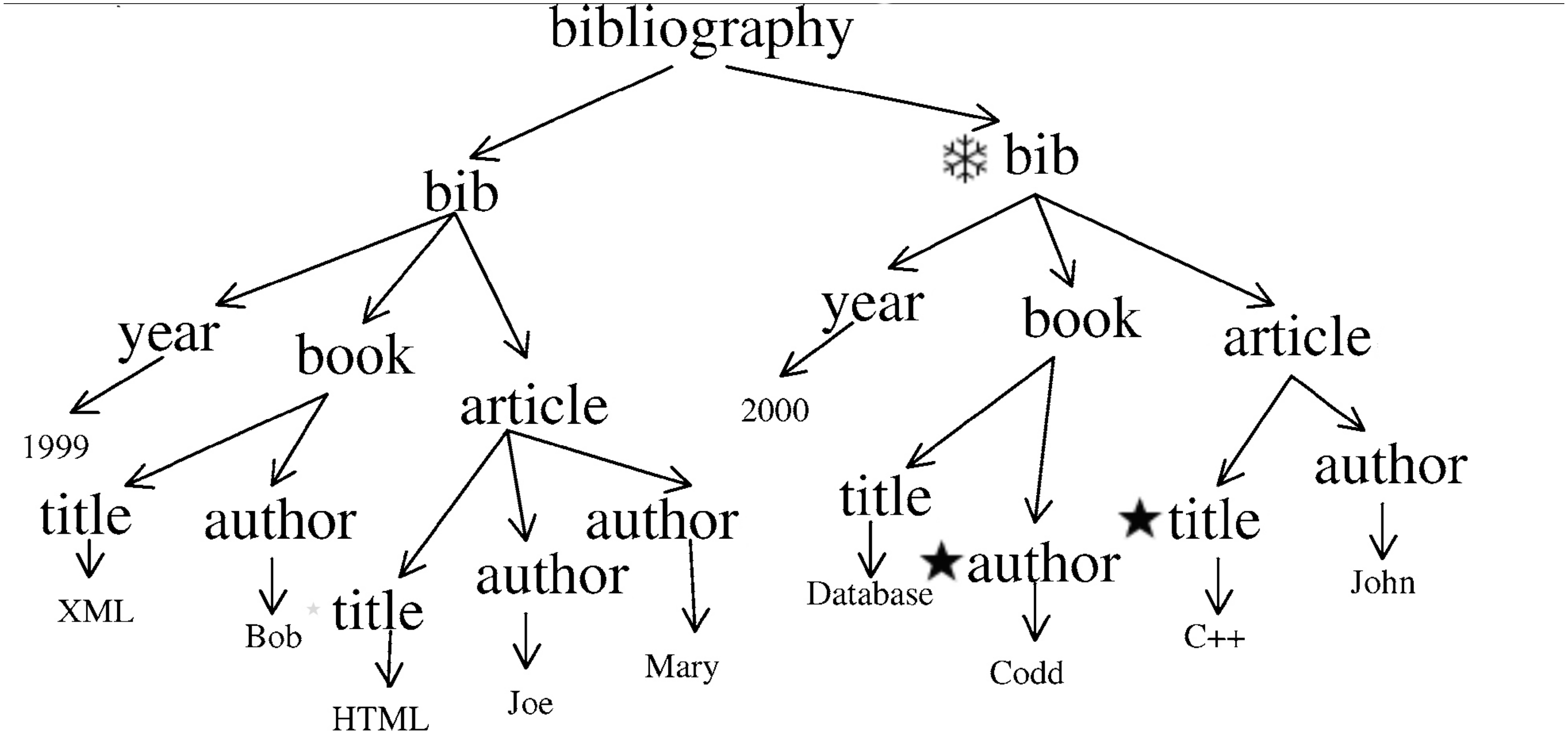
Lowest Common Ancestor (LCA)



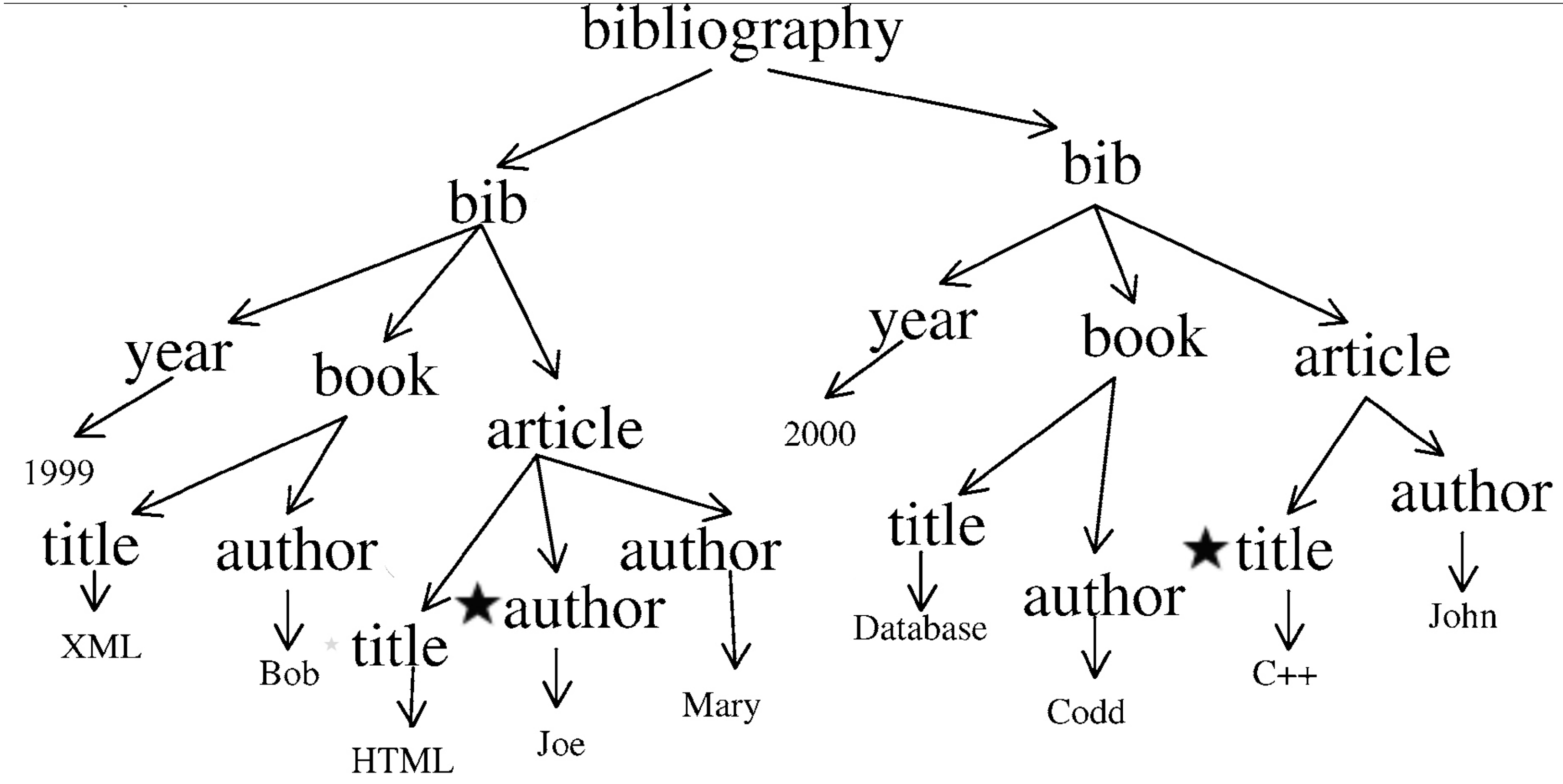
Lowest Common Ancestor (LCA)



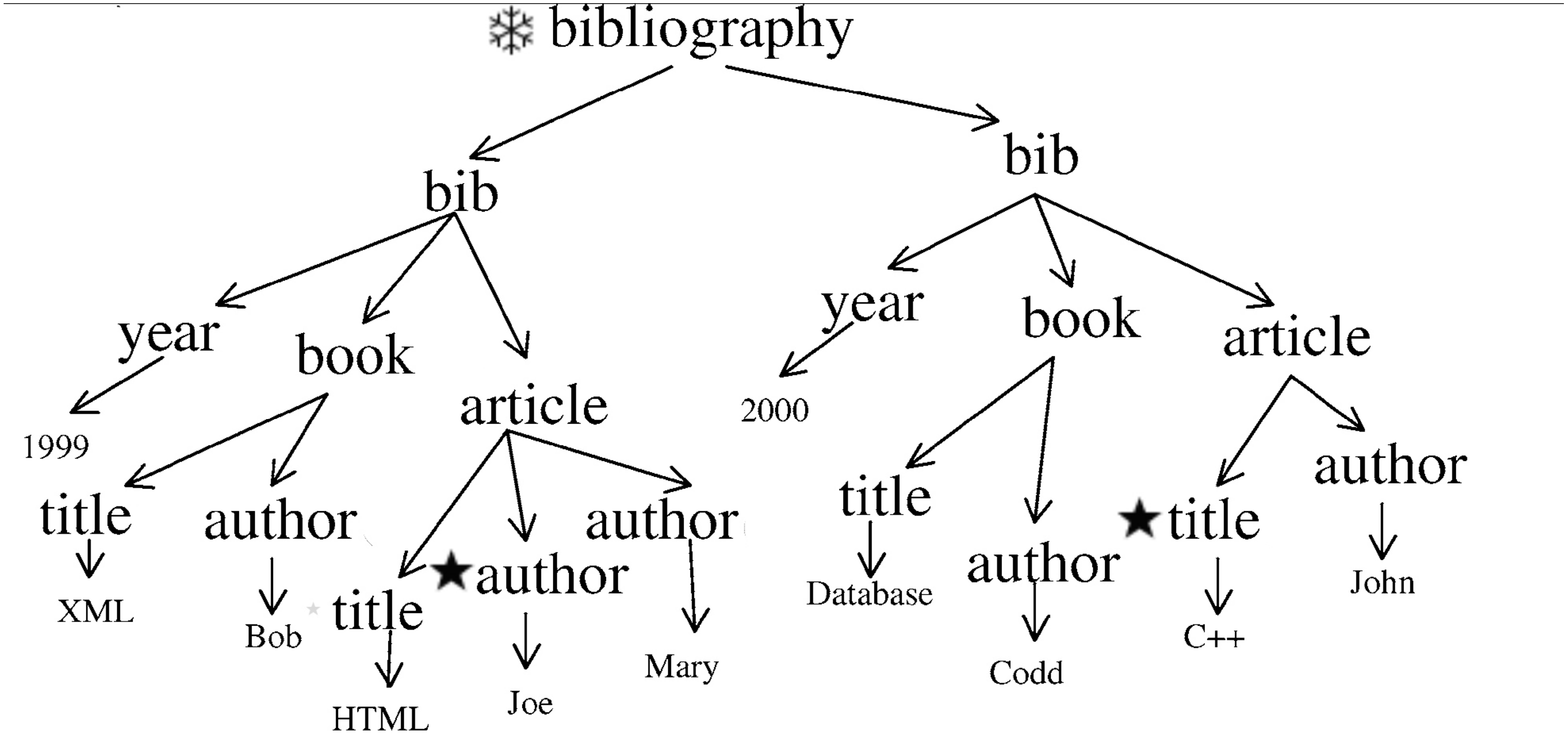
Lowest Common Ancestor (LCA)



Lowest Common Ancestor (LCA)



Lowest Common Ancestor (LCA)

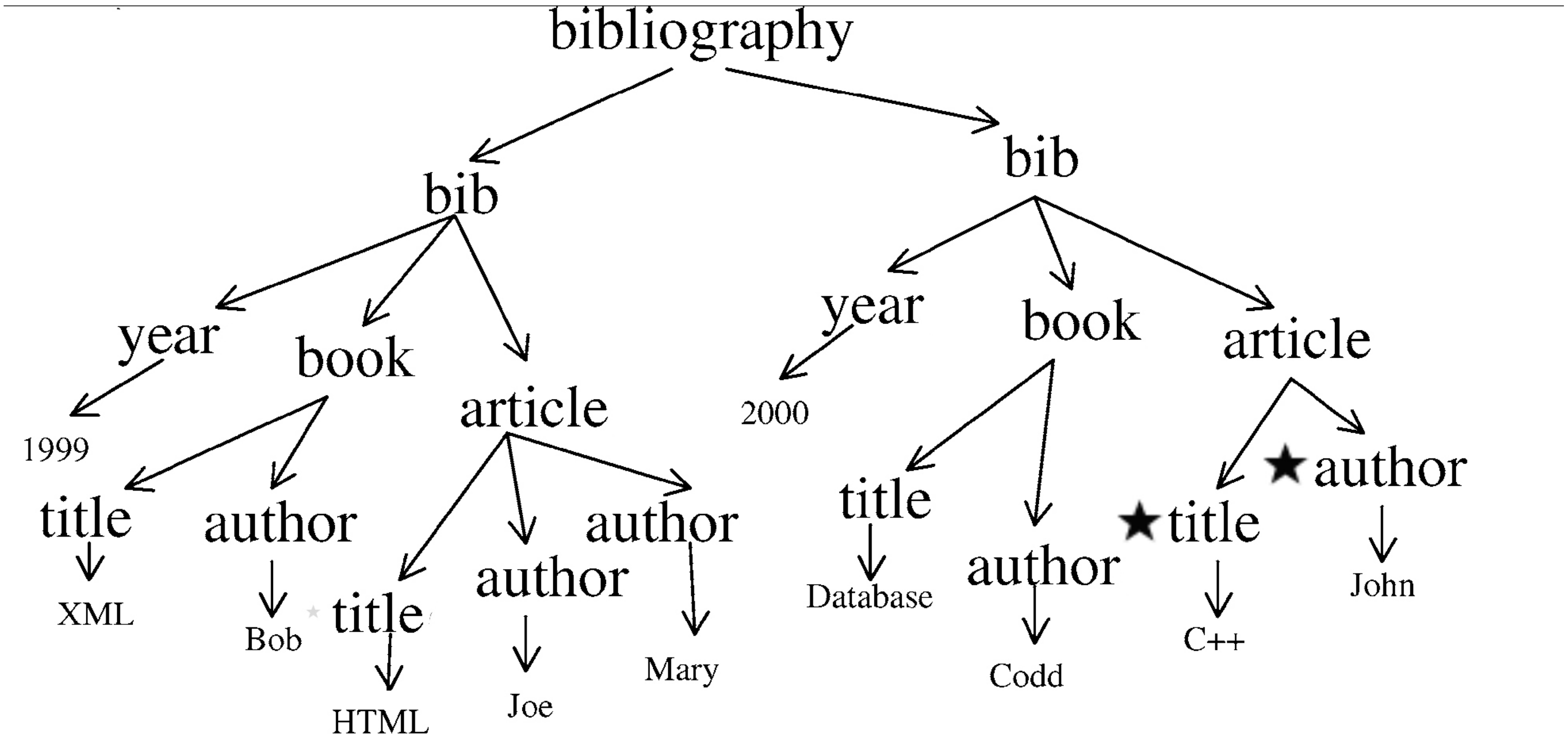


Yet even more definitions – meaningfully related nodes

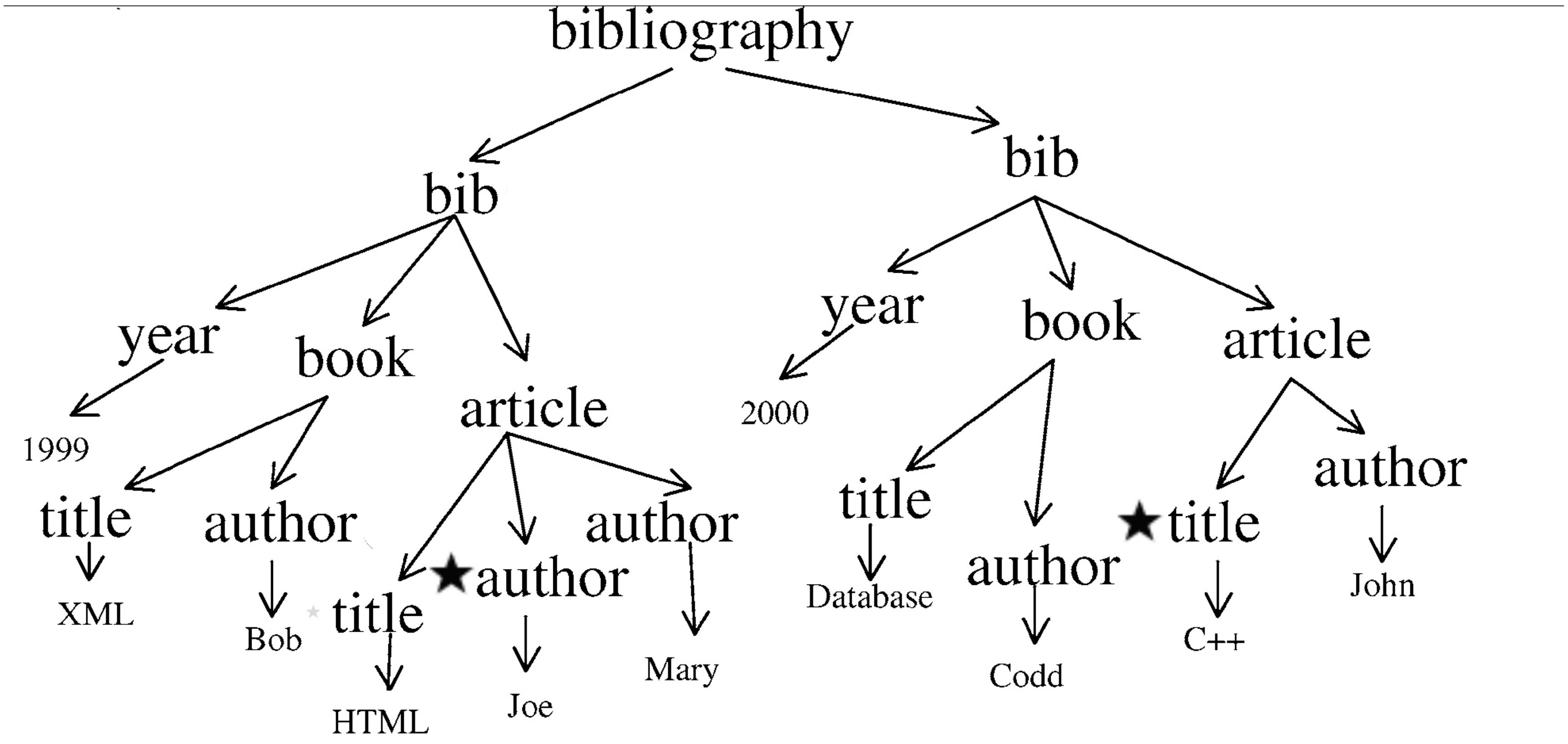
Two nodes *a* and *b* are related meaningfully if the following conditions are met:

1. There are no two nodes with the same label on the subtree
2. The only two nodes that can have the identical labels on this subtree are the nodes *a* and *b* themselves

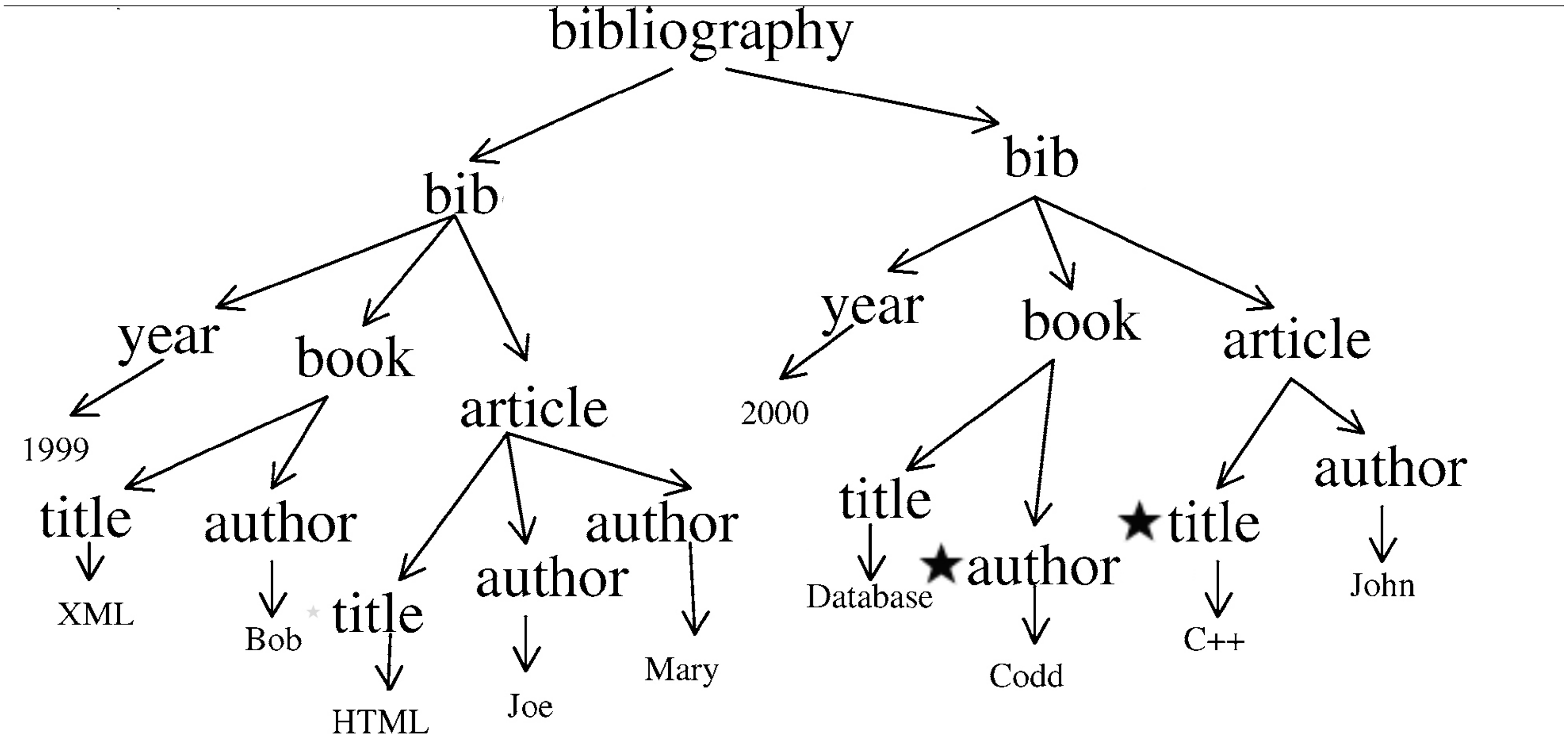
Example of meaningfully related nodes



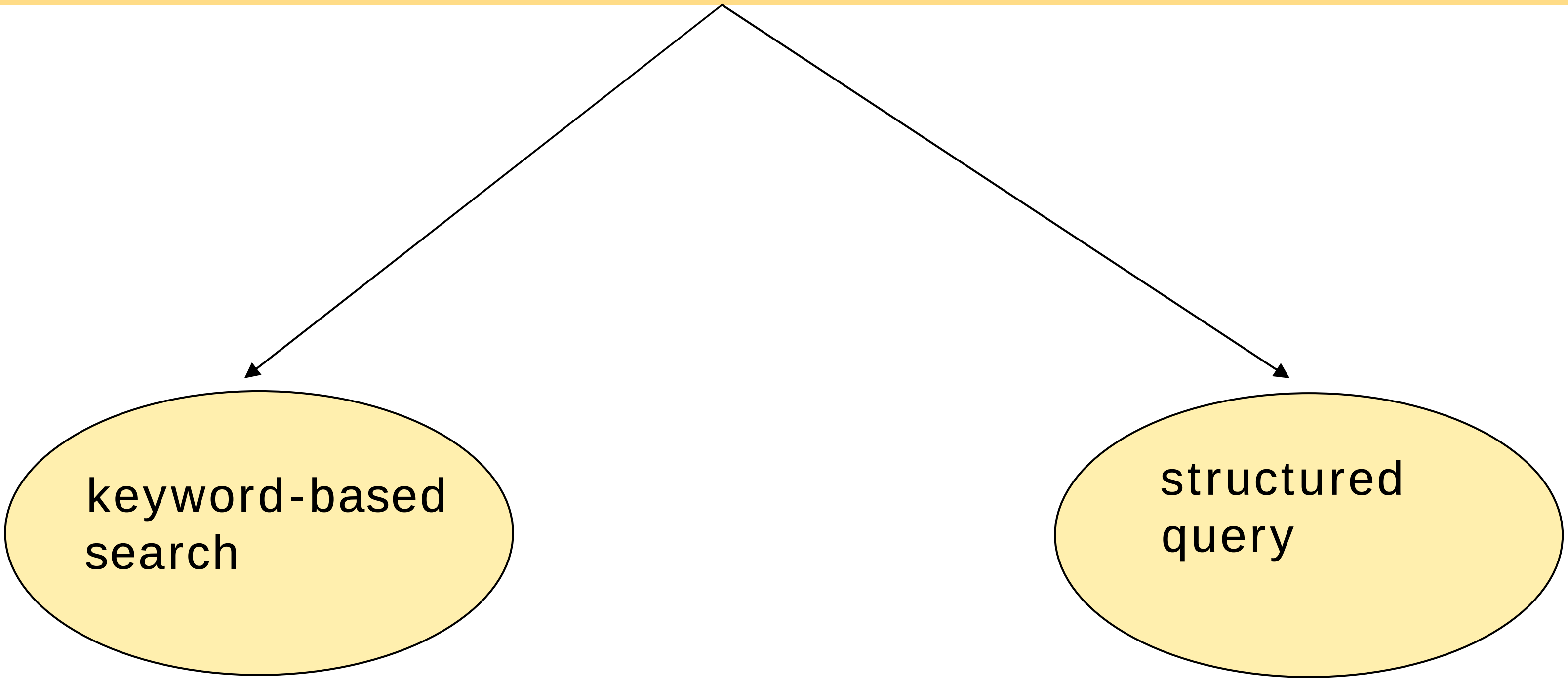
Example of two nodes, which are not meaningfully related and do not satisfy the rule



Example of two nodes that satisfy the rule, but are not meaningfully related



Types of search on XML documents



keyword-based search

structured query

XSEarch (2003)

Schema-free XQuery (2004)

XRank

XQuery

structured query search

(–)

requires knowledge of the XML document structure (partial or full);

requires user to possess basic knowledge of query language;

problems associated with retrieving data from multiple documents (dif. schemas – dif queries or use of translation)

(+)

high precision of the results;

queries can convey sophisticated semantic meaning;

keyword-based search

(–)

hard to express semantic meaning in a query (e.g. no way to refer in a query to the tags deliberately);

no way to specify which part of the document should be returned, often returns large number of

undesired results.

(+)

no knowledge of structure is necessary;

no special knowledge is needed, anyone can execute search.

Schema-Free XQuery

the base of the approach is the concept of Lowest Common Ancestor

authors define on the top of it *Meaningful Lowest Common Ancestor Structure* (MLCAS);

MLCAS serves for the purpose of identifying the nodes, which are meaningfully related to each other;

extends XQuery with the mlcas function, which provides an implementation of this approach.

Li [2]

XSEarch

comprehensible syntax for queries, suitable for all users but also with extra possibilities for more advanced users to specify how keywords are related;

has two types of algorithms for off- and online computation;

has a full-fledged system for ranking the answers;

based on the tree representation of the XML document develop their own heuristic for defining relationships between nodes, key concept – define meaningfully related sets of nodes.

Cohen [1]

The Approach, which is described in my article

based on the concept of Lowest Common Ancestor (LCA) developed the LCA of Label Path structure (PLCA);

adapted the LCA rule for checking whether the nodes are meaningfully related (PLCA rule);

use Dewey indexing for nodes;

introduced a new type of index structure - PN-inverted index.

Li [3]

Some more definitions

“A *prefix path lp'* of lp is a sub-sequence of lp, where lp' has the same beginning with lp and
|lp'|≤|lp| ”

For the label path regions.Africa.item.name, which of the following will be the prefix paths (according to the definition):

regions.Africa,

regions.Africa.item,

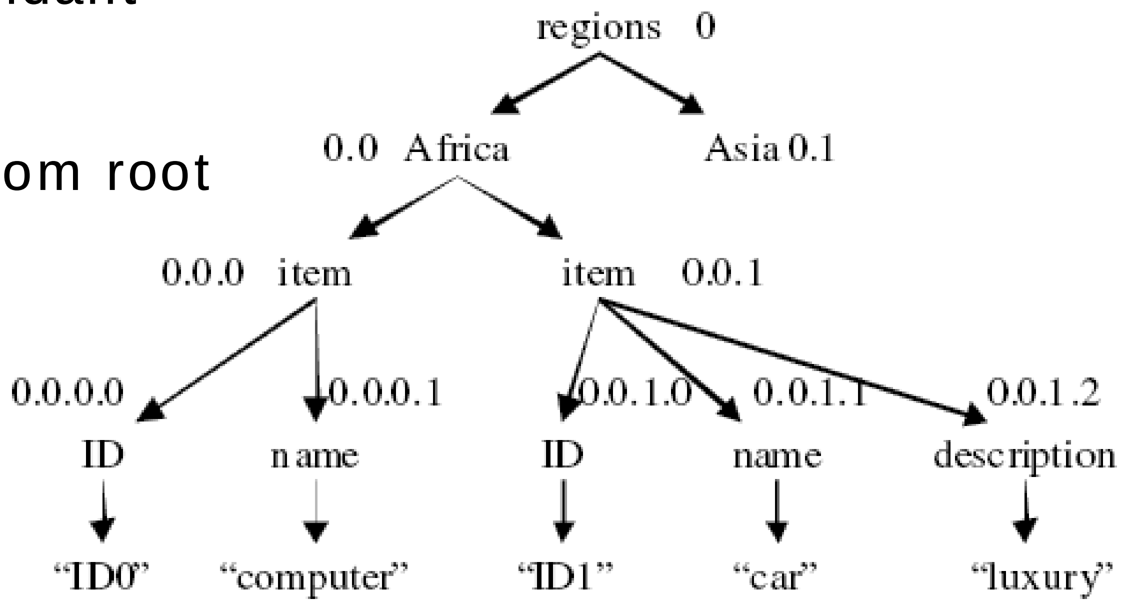
regions.Africa.item.ID?

Node encoding

Dewey encoding:

captures the relationship of ancestor and descendant information;

reflects the depth from root to the node.



LCA of two label paths

Label path k is Common path of label paths a and b, if it is the prefix path for both a and b and it is longer than any other label path k', which is also the prefix path of both a and b.

Then the last label in k is the LCA of a and b, denoted as

k = PLCA (a,b)

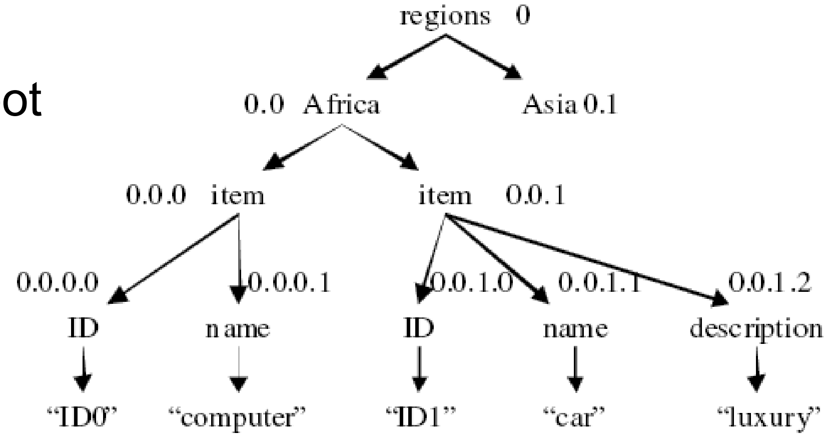
PLCA rule

nodes u1 and u2

lp1 and lp2 – label paths from root to the nodes u1 and u2

common path lp

u=LCA(u1,u2)

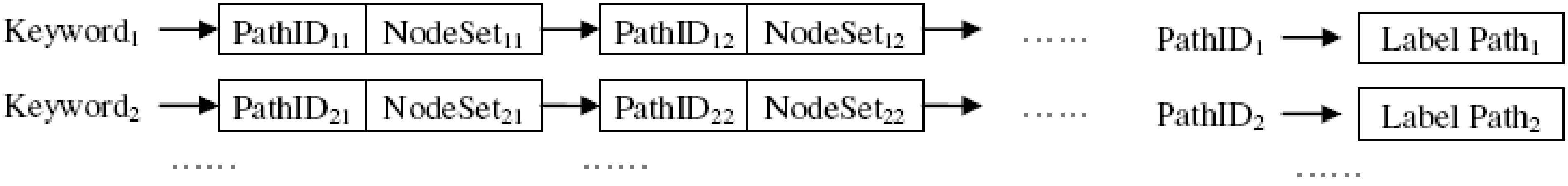


u1 and u2 are meaningfully related if:

|lp| = | Dewey id of u |

there are no two distinct labels with the same name (other than ends of lp1 and lp2) in the set (lp1-lp)U(lp2-lp)

PN-Inverted Index



Keyword$_1$ → PathID$_{11}$ | NodeSet$_{11}$ → PathID$_{12}$ | NodeSet$_{12}$ → ……

Keyword$_2$ → PathID$_{21}$ | NodeSet$_{21}$ → PathID$_{22}$ | NodeSet$_{22}$ → ……

…… ……

PathID$_1$ → Label Path$_1$

PathID$_2$ → Label Path$_2$

……

(a) Enhanced inverted list

(b) Path index

store for a keyword ID’s of all nodes, which contain this keyword + label paths from the root to these nodes;

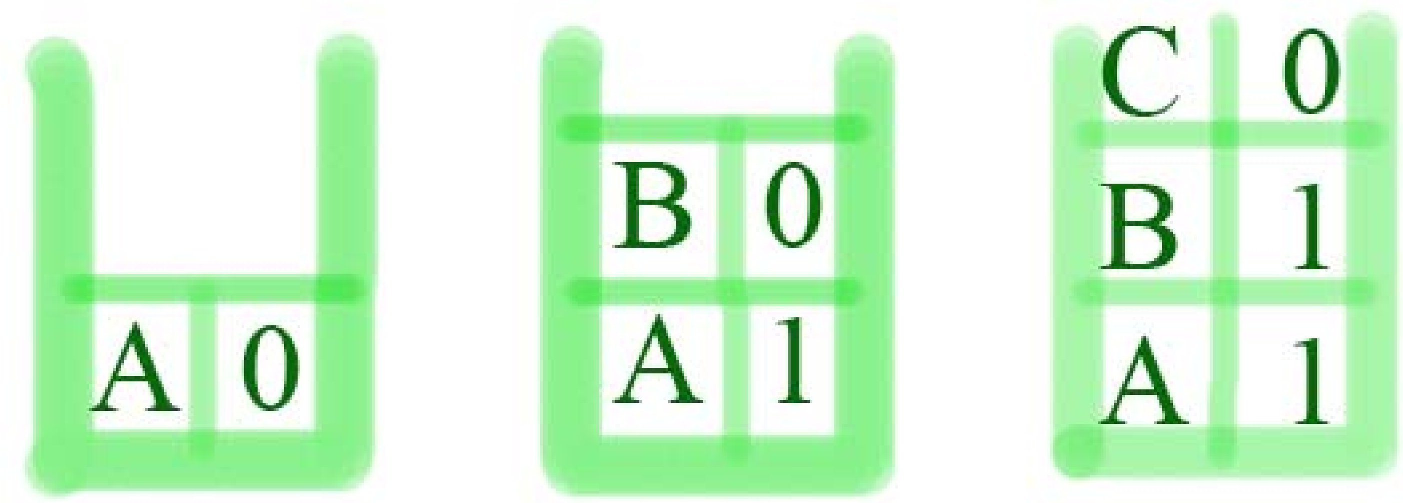
nodes that have the same label path are grouped together to form the NodeSet;

store ID of label path (instead of the path itself), using Path index(b) structure for mapping between the paths and their ID’s

Query Algorithm

```
if (!stack.isEmpty) stack[top].childCount ++;
    childCount = 0; label ← getCurrentNodeLabel();
    stack.push(label, childCount);
```

Example: <A <B <C> key />/>/>



A 0

B 0
A 1

C 0
B 1
A 1

Query Algorithm

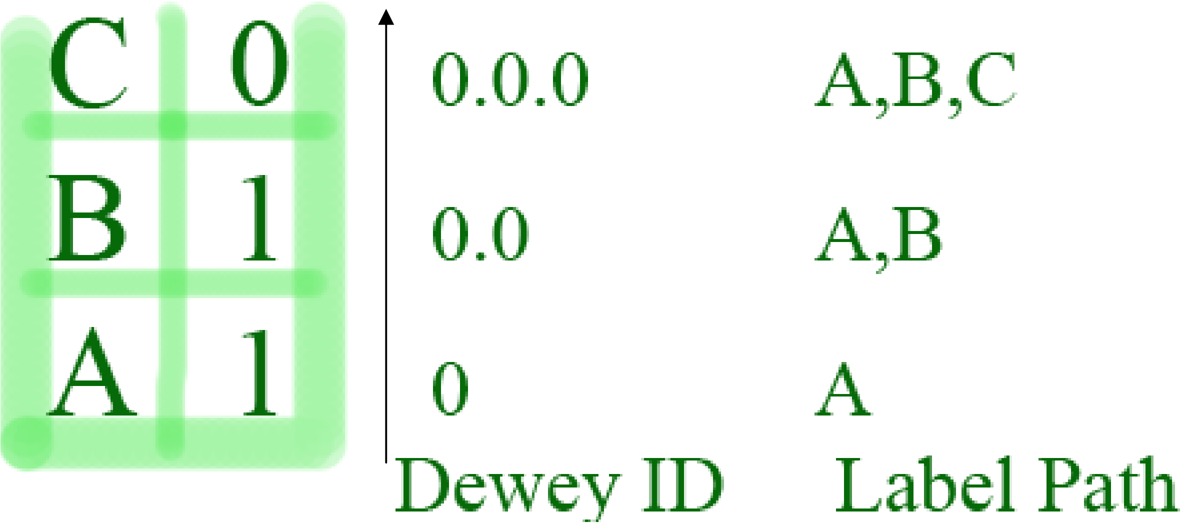
```
if (node belongs to NV)
    curID ← getIDfromStack();
    curLabelPath ← getLabelPathfromStack();
    curPathID ← pathIndex.getPathID(curLabelPath);
    for each word in the text
        invertedIndex.addEntry(word, curID, curPathID)
```

getIDfromStack() and getLabelPathfromStack() functions traverse the stack from bottom to top to compute the Dewey id and the label path of current node, respectively

addEntry function clusters nodes to the corresponding keyword

Query Algorithm

Example: <A <B <C> key />/>/>



C	0
B	1
A	1

Dewey ID	Label Path
0.0.0	A,B,C
0.0	A,B
0	A

Query evaluation

"all-pairs" approach for defining a set of meaningfully related nodes – each pair in a set should be meaningfully related

Naïve implementation of query evaluation

the search engine looks up each of the *k* keywords;

returns *k* nodes sets (each corresponding to one keyword);

calculates the Cortesian product of the these k sets;

checks whether they are meaningfully related (for each pair) based on the PLCA rule.

Efficient Query Evaluation

if two nodes are not meaningfully related, they will result in irrelevant answer in any combination with other nodes, therefore such nodes should be pruned as early as possible

Similarities with Schema-free XQuery

both approaches are based on the LCA concept;

build up very similar structures based on the LCA rule to check whether the nodes are meaningfully related to each other;

use index encodings for nodes, which captures depth and path from the root.

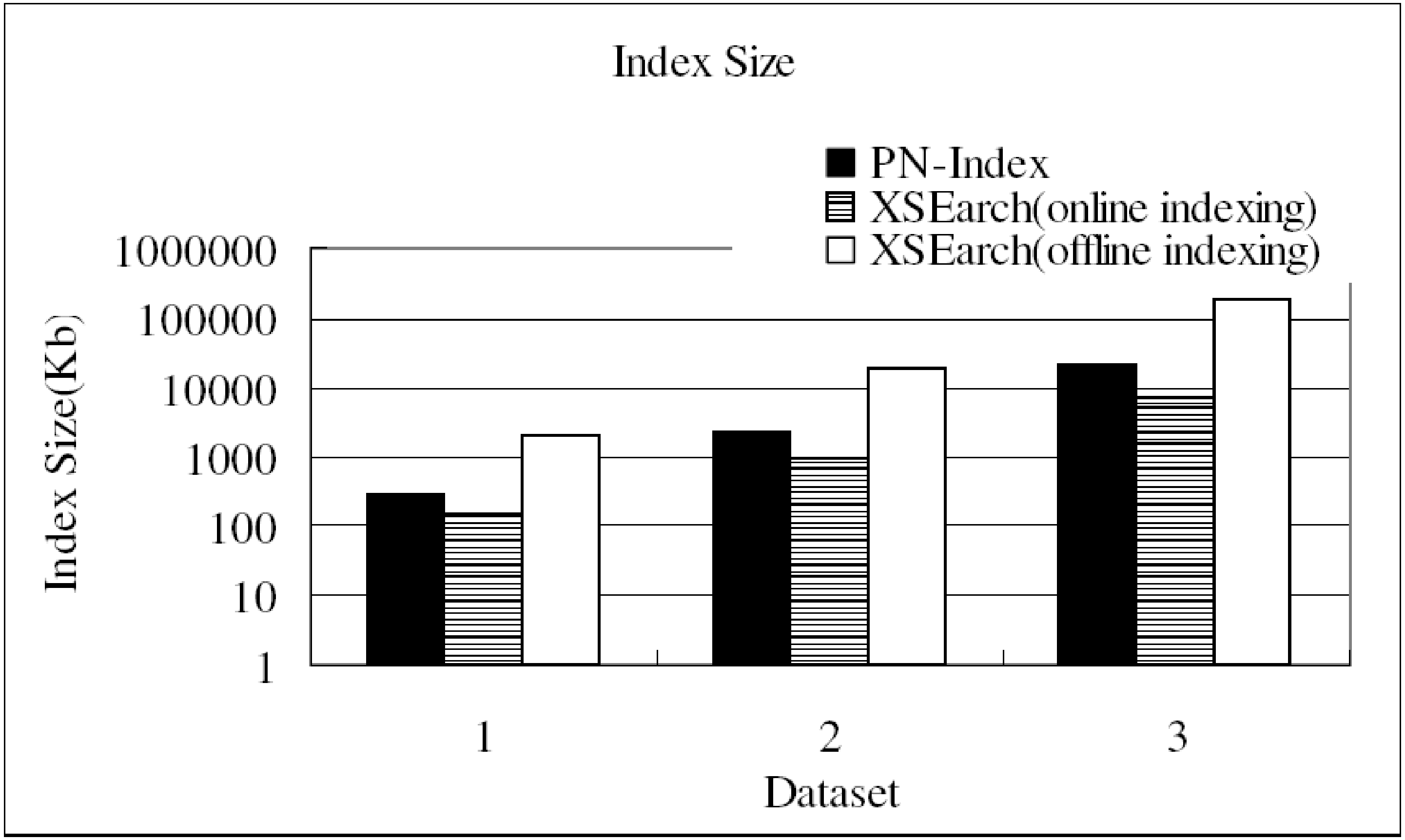
Similarities with XSEarch

both have comprehensive syntax for queries, suitable and convenient for naïve users;

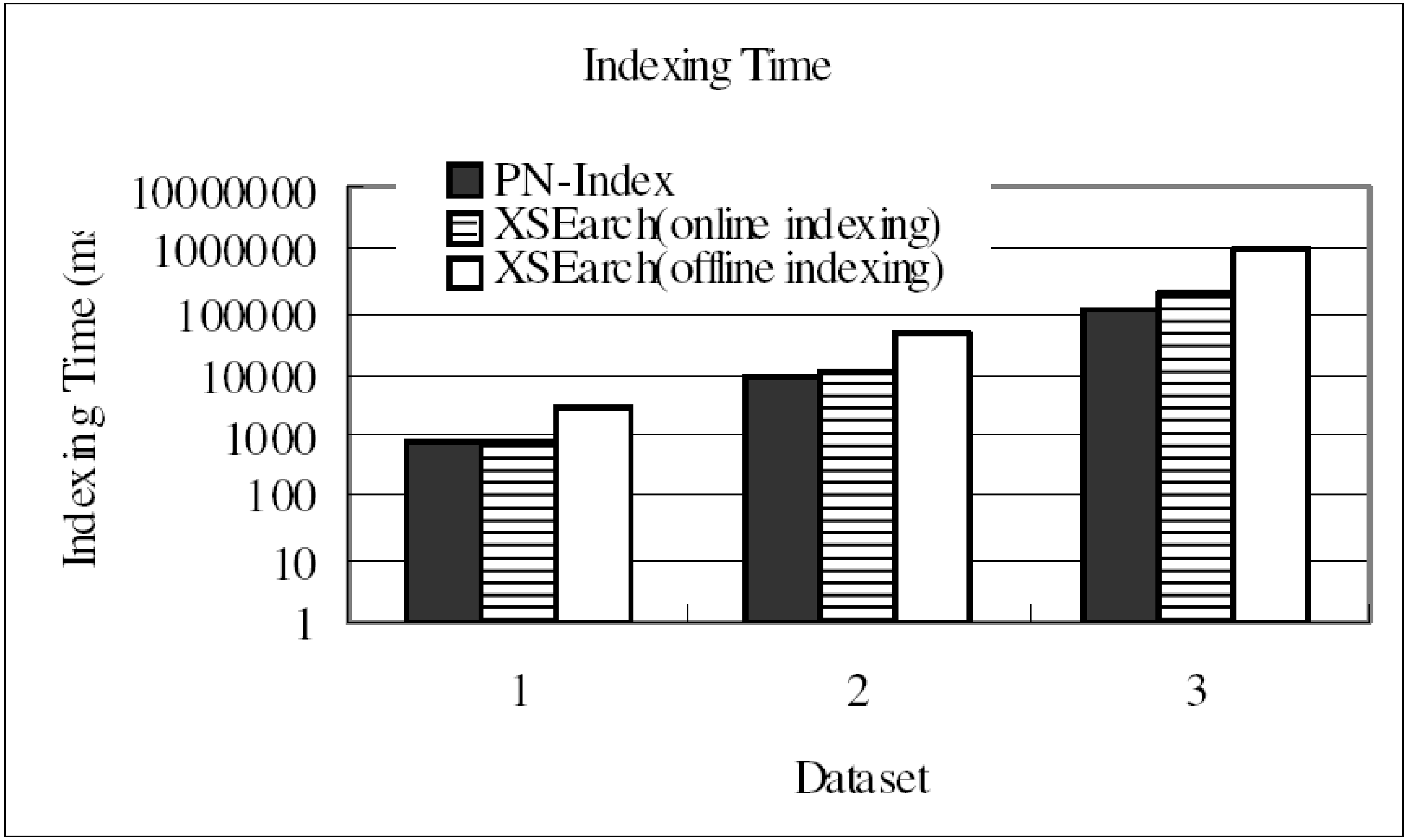
relational subtree concept;

check against the result elements satisfy each word in the query, as well as whether or not they are meaningfully related.

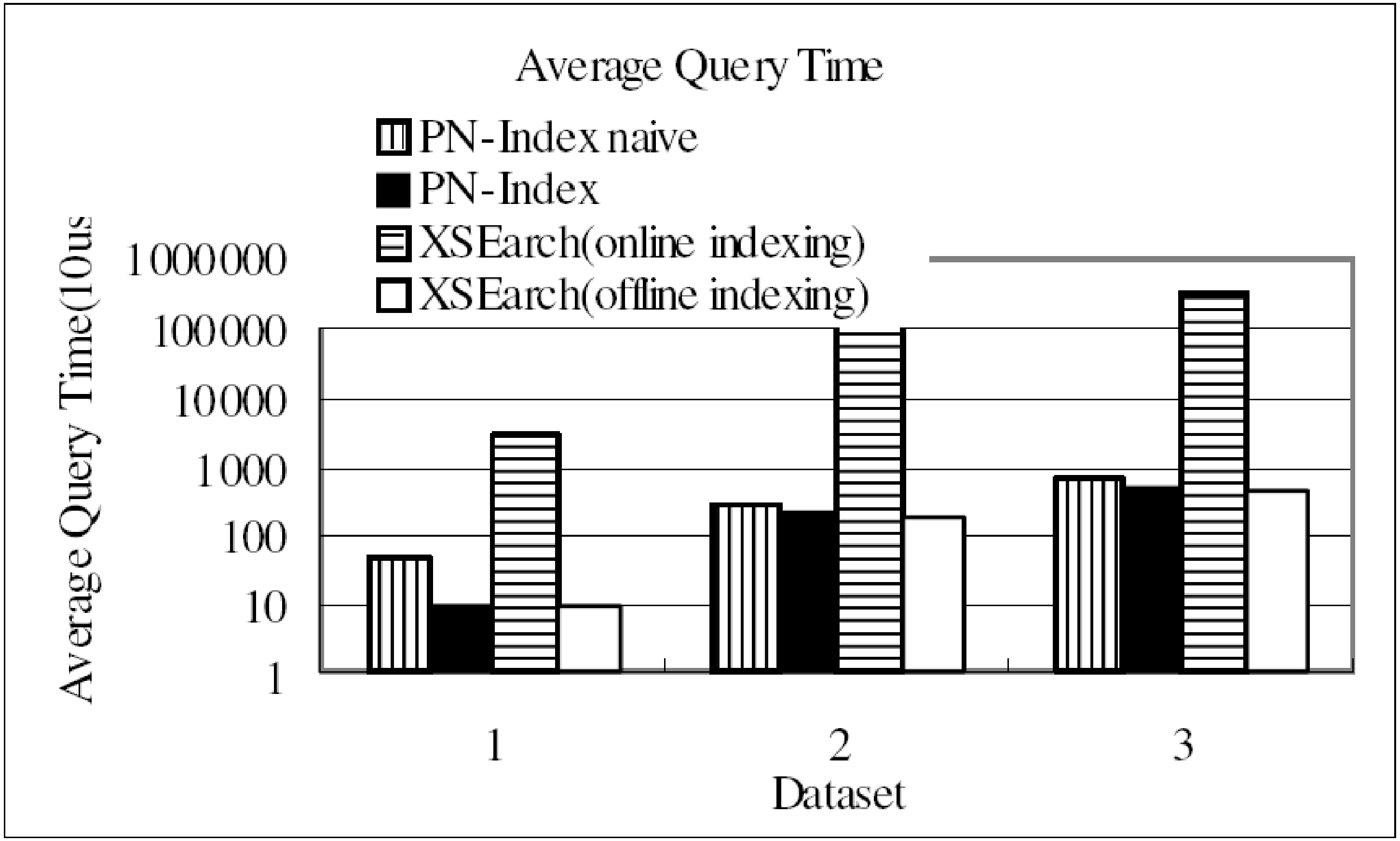
Performance



Performance



Performance



References

1. S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv. XSearch: a semantic search engine for xml. Proc. of VLDB, 2003
2. Y. Li, C. Yu, H. V. Jagadish. Schema-free XQuery. Proc. of VLDB, 2004
3. X.Li, J.Gong,D.Wang, G. Yu An Effective and Efficient Approach for Keyword-Based XML Retrieval