

Object-Oriented XML Keyword Search

Huayu Wu, and Zhifeng Bao
National University of Singapore

Outline

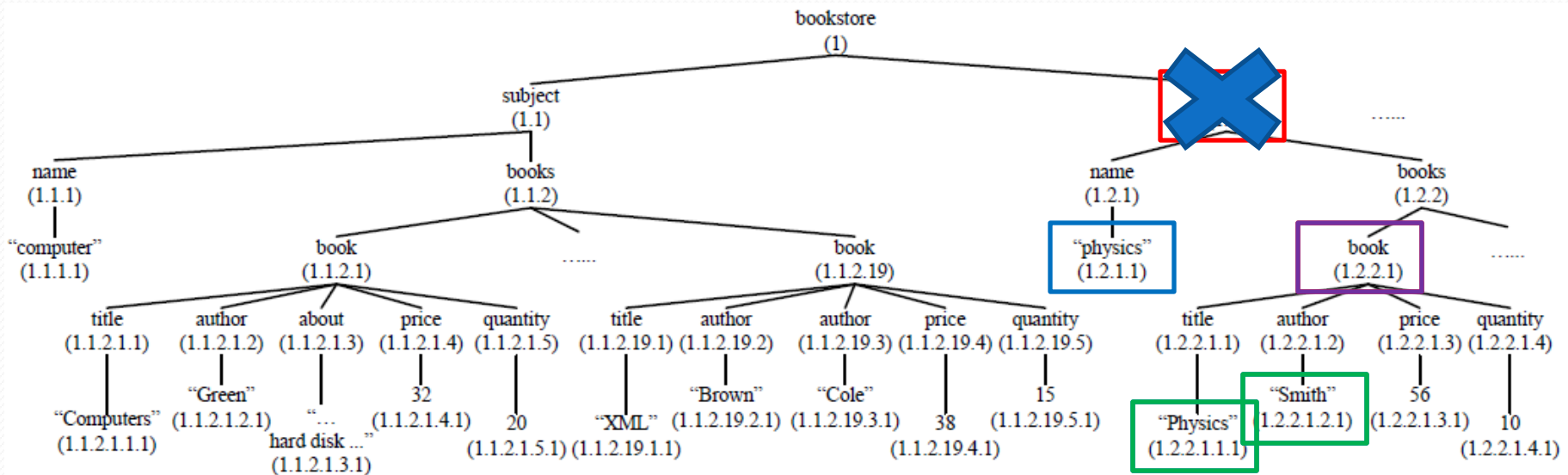
- Introduction
- Motivation
- Our Approach
- Experiments
- Conclusion

Introduction

- XML Keyword Search
 - Originated from Information Retrieval (**IR**)
 - Find **relevant information** based on a set of given keywords
 - How to define **relevant information** in XML data?
 - Tree-based approach: *Lowest Common Ancestor* (**LCA**) of query keywords.
 - Graph-based approach: *path* containing query keywords.
 - Since tree search is more **efficient** than graph search, the LCA-based approach attracts most research attention.
 - E.g., LCA, SLCA, MLCA, VLCA.....

Introduction (Cont.)

- Example: Smallest Lowest Common Ancestor (SLCA)



- Query: {Physics, Smith}
 - LCA answers: book(1.2.2.1), subject(1.2), ...
 - SLCA answers: book(1.2.2.1), ...
 - Many other improved LCA-based semantics

Motivation

- The existing works do not consider the **semantics of object and attribute** during LCA-based searching.
 - Such semantics is considered in some work to infer output information, after LCA-based searching, e.g., XSeek.
- Problems:
 - **Efficiency problem:**
 - **Example 1: {book, title}**. If every book has a title, there is no need to search for all book nodes and all title nodes to compute LCAs.
 - **Example 2: {book, title, XML}**. If there are 100 book titles, but only one of them has value of “XML”, it is redundant to search all 100 title nodes.

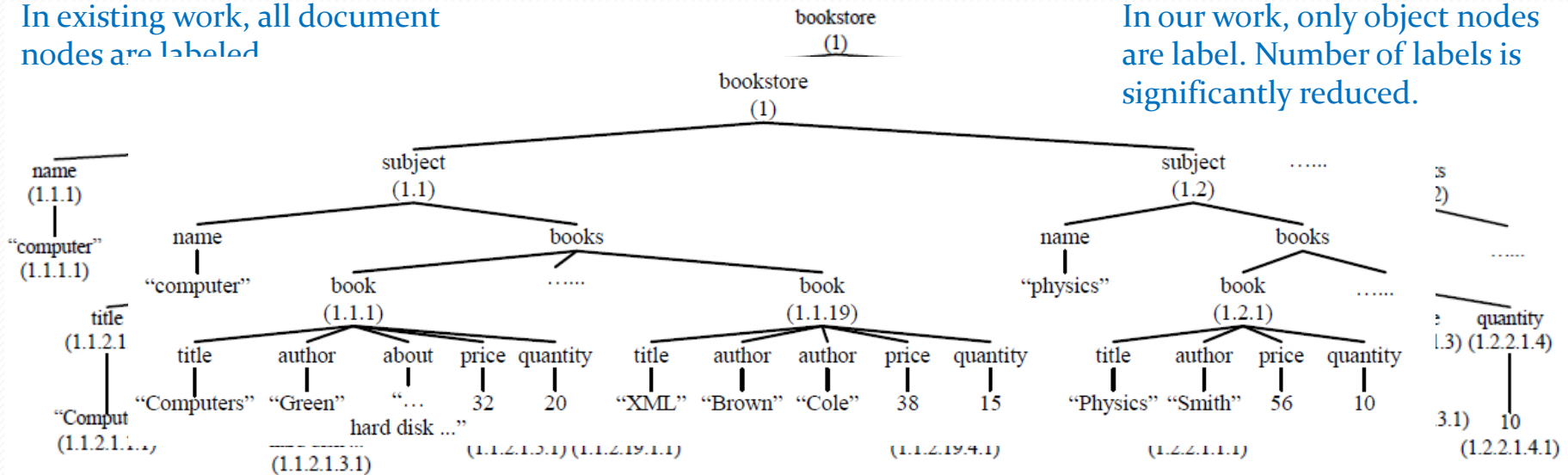
Motivation (Cont.)

- Problems (cont.):
 - Search quality problem:
 - **Example 3: {physics}**. Physics may refer to the subject of physics, or the book of physics. LCA-based computation mix up all interpretations, which is difficult for users to filter results based on his/her real search intention.
 - **Example 2: {book, price<50}, {book, about “hard disk”}**. The inverted list based LCA computation cannot efficiently support advanced search, e.g., range search, phrase search, etc.
 - Solution: Using **semantics of object** during LCA-based computation, to avoid these efficiency and search quality problems.

Our Approach

- Core idea: put semantics into LCA-based computation.
- Document labeling: Only **label objects**. Other nodes inherit the label of its lowest ancestor object.

In existing work, all document nodes are labeled



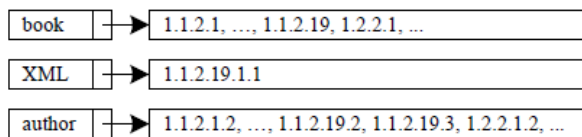
In our work, only object nodes are label. Number of labels is significantly reduced.

Our Approach (Cont.)

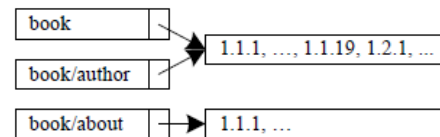
- Indices:

- Inverted lists

- Only for **non-value** nodes, and also **object-oriented**.



(a) Normal inverted list



(b) OO inverted list

- Object tables

- Values are stored in object tables, with the corresponding object labels

R_{book}

OO-Dewey	Title	About	Price	Quantity
1.1.1	Computers	... hard disk ...	32	20
...
1.1.19	XML	null	38	15
...
1.2.1	Physics	null	56	10
...

$R_{\text{book/author}}$

OO-Dewey	Value
1.1.1	Green
...	...
1.1.19	Brown
1.1.19	Cole
...	...
1.2.1	Smith
...	...

- Other indices to **map attribute type to object class**, and to **map values to the belonged object and attribute instances**.

Our Approach (Cont.)

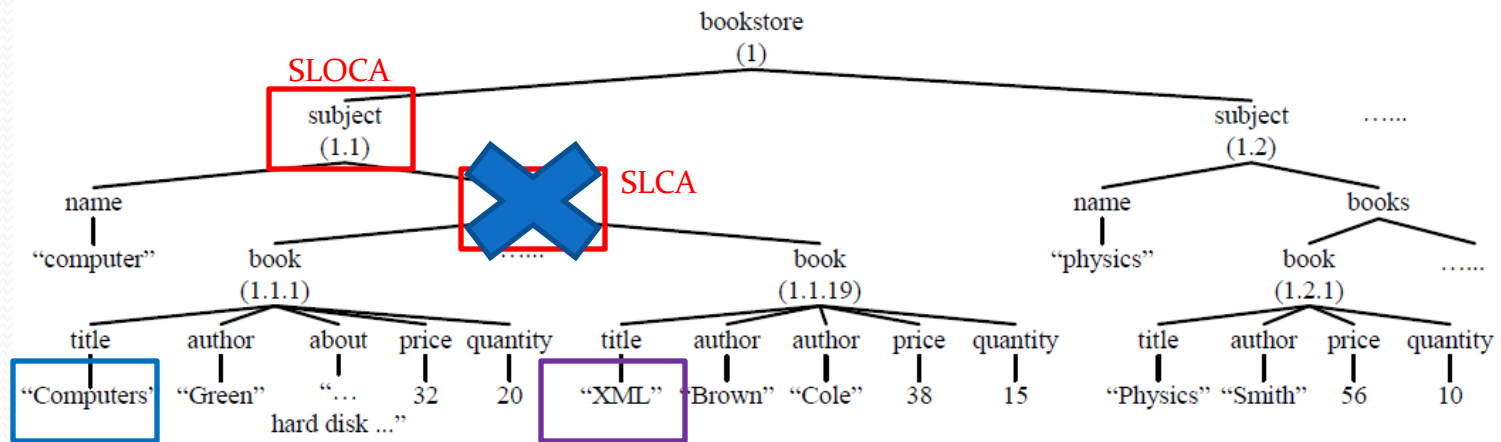
- OO keyword query processing
 - Step 1: Keyword partitioning
 - Create **one partition for each object** involved. Associate each value keyword to the corresponding attribute keyword and finally to the corresponding object partition. Associate unattached attributes keyword to object partitions.
 - **Ambiguous** keyword query leads different partitioning results.
 - Example:
 - {subject, physics} => {(subject, name/physics)}
 - {Brown, Cole} => {(book, author/Brown, author/Cole)} or {(book, author/Brown), (book, author/Cole)}

Our Approach (Cont.)

- OO keyword query processing
 - Step 2: Inverted list filtering for each partition
 - Case (a), if the partition contains only object keyword, take the **inverted list** for that object.
 - Case (b), if the partition contains both object and attribute keywords, but no value, take the **intersection** of inverted lists for the attribute keywords.
 - Case (c), if the partition contains object, attribute and value keywords, take the **selection** from the object table.
 - Example: {book, XML, subject, name} => {(book, title/XML), (subject, name)}.
 - The inverted list for **Partition 1** is the selected labels from book table, based on title = “XML”.
 - The inverted list for **Partition 2** is the inverted list for subject/name.

Our Approach (Cont.)

- OO keyword query processing
 - Step 3: SLOCA computation
 - Similar to the SLCA computation, but make sure the resulting nodes are object nodes.

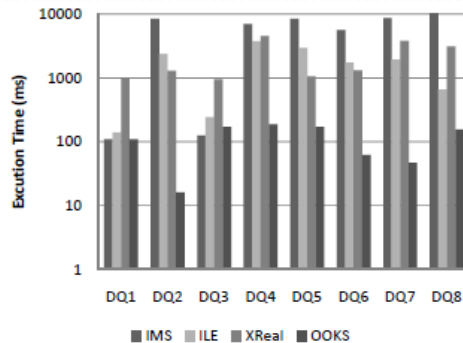


Our Approach (Cont.)

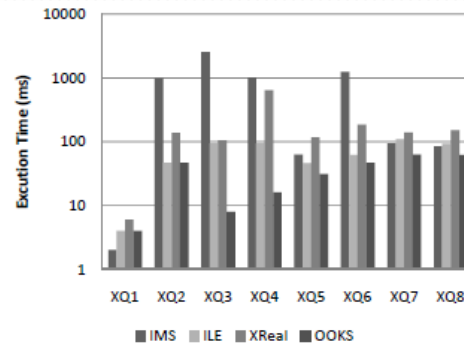
- OO keyword query processing
 - Step 4: Result return
 - Identifying return information
 - Based on the closeness among object keywords in the document schema
 - Extracting values
 - From object tables
 - Supports **advanced search**, e.g., range search, phrase search, because of the power of SQL in table selection.

Experiments

- Experimental results:
 - Efficiency:
 - Compared to three existing SLCA implementation and IR-styled keyword search algorithms



(a) DBLP data

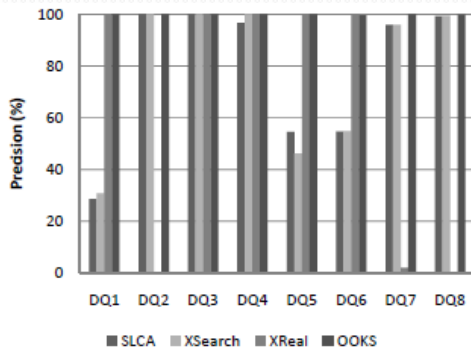


(b) XMark data

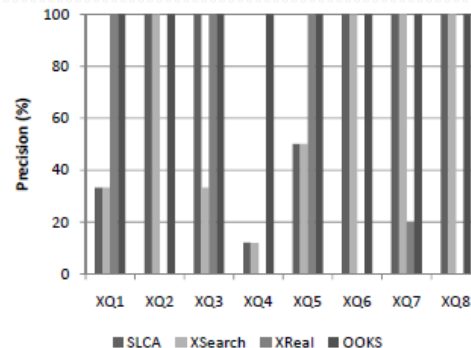
- Better performance, because after keyword partitioning, the searching is on **object level**, and reduce a lot of computation.

Experiments (Cont.)

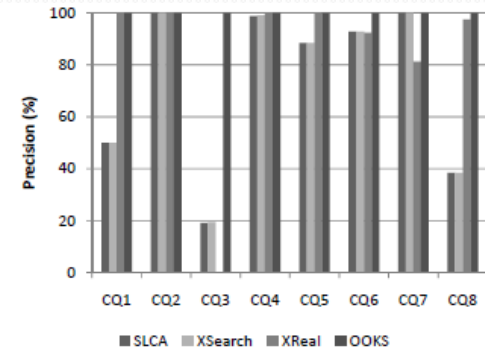
- Experimental results:
 - Search quality:
 - Compared with SLCA, XSearch, XReal, they are well-known XML keyword search algorithms.



(a) DBLP data




(b) XMark data



(c) Course data

Conclusion

- We propose to use the **semantics of object** in XML keyword search.
 - Step 1: Associate values and attributes to objects, for XML keyword queries.
 - Step 2: Filter the inverted list for each object partition.
 - Step 3: Perform LCA-based computation in object level.
 - Step 4: Return result with object tables.
- We show by experiments that the Object-oriented approach can improve both the query processing efficiency and the quality of keyword search.



Thank you
Q & A