

# Schema-Free XQuery

---

Yunyao Li, Cong Yu, H.V. Jagadish



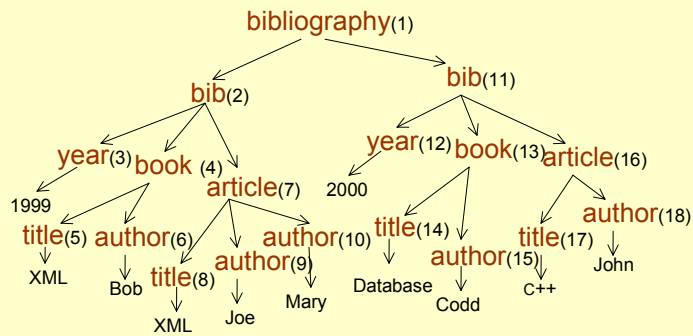
## Roadmap

---

- Introduction and Motivation
- Data Model
- Core algorithm
- Experimental Evaluation
- Conclusion and Future Work

# Example

---



**Query 1:** What are the titles and years of the publications, of which Mary is an author?

## Standard Solution - XQuery

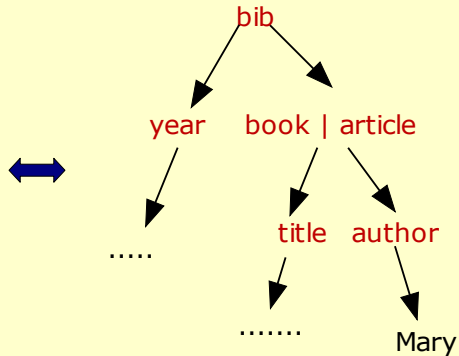
---

- **XQuery**
  - a structured query language
  - search for specific information within XML documents primarily based on path expression
  - require extensive knowledge of XML structure

# Standard Solution - XQuery

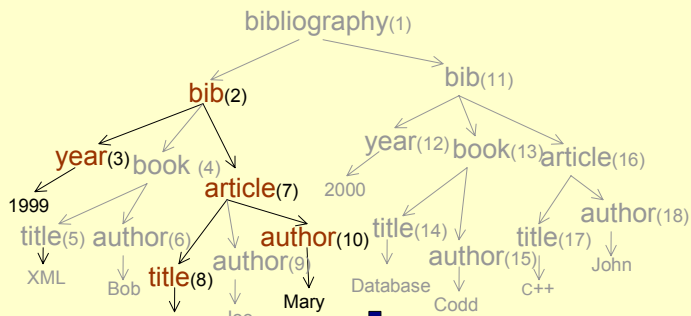
**Query 1:** What are the titles and years of the publications, of which Mary is an author?

```
for $b in doc("bib.xml")/bib
for $c in $b/book or $b/article
where $c/author = "Mary"
return
{<$c/title>
  <$b/year>}
```



# Standard Solution - XQuery

**Query 1:** What are the titles and years of the publications, of which Mary is an author?



```
<title>XML</title>
<year>1999</year>
```

The correct answer

## Standard Solution - XQuery

---

- XQuery
  - Flexible
  - Accurate
- XQuery is powerful, but .....

**The user may NOT KNOW the structure of bib.xml!**

## Solution 1 – Study Schema

---

- **Solution 1:**  
Ask the user to study the schema of bib.xml and write a query in XQuery



## Solution 1 – Study Schema

---

### More problems

- Data evolution: The document structure may change over time
- Heterogeneity: multiple XML documents with similar content but different structures

## Solution 2 – Keyword Search

---

- **Solution 2:**

Keyword Search – IR approach

- Discard all tags:

*“Mary”*

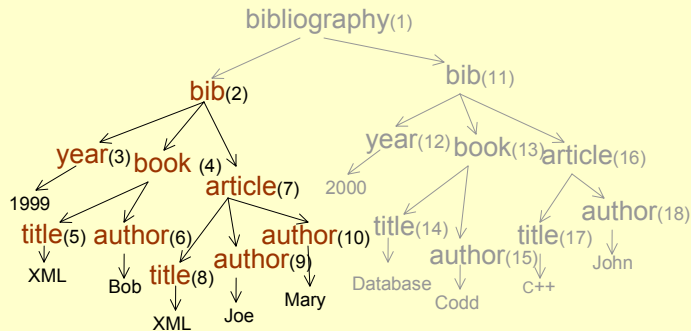
- Treat tags as keywords:

*“book article year title author Mary”*

## Solution 2 – Keyword Search

**Query 1:** What are the titles and years of the publications, of which Mary is an author?

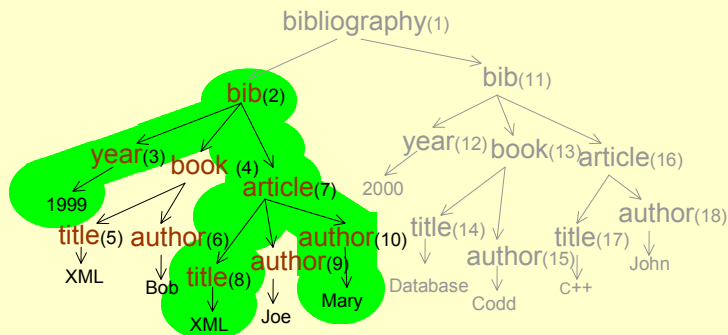
*“year title author Mary”*



## Solution 2 – Keyword Search

**Query 1:** What are the titles and years of the publications, of which Mary is an author?

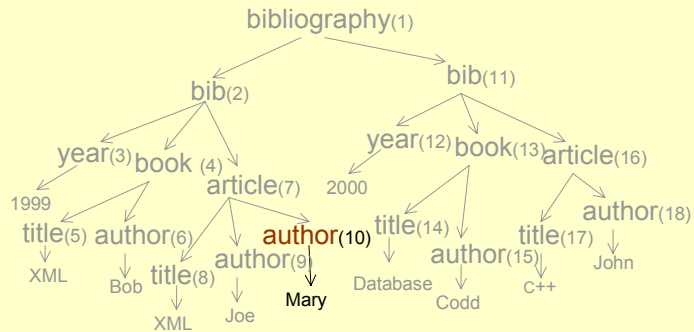
*“author Mary year title ”*



## Solution 2 – Keyword Search

**Query 2:** Find additional authors of the publications, of which Mary is an author?

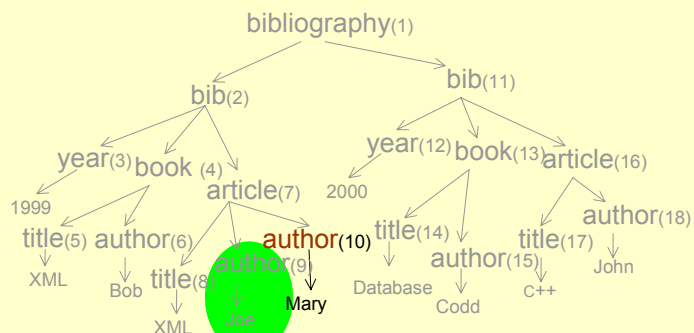
*“author Mary author”*



## Solution 2 – Keyword Search

**Query 2:** Find additional authors of the publications, of which Mary is an author?

*“author Mary author”*



## Solution 2 – Keyword Search

---

**Pro** - No knowledge of document structure required

**Con** - Most do not distinguish tag names from value

*Eg. "book" as a tag name, not as a value*

- Cannot express complex semantic knowledge

*Eg. Year < 2000*

- Cannot specify desired result

## Our Goal

---

- **Schema-Free XQuery**

- Enable users to query XML data by exploiting whatever partial knowledge of the schema they have: support wide range of queries - from regular XQuery to keyword search.
- Requires no exact knowledge of tag names
- Minimum extra effort required from the user

# Roadmap

---

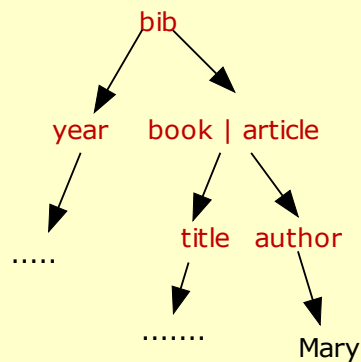
- Introduction and Motivation
- **Data Model**
- Core Algorithm
- Experimental Evaluation
- Conclusion and Future Work

# Query Focus

---

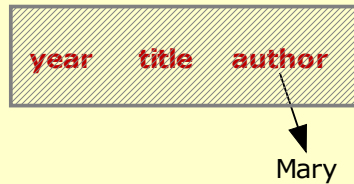
- Knowing the document structure, the user can specify in XQuery **HOW** the nodes are related in terms of structural relationship

```
for $b in doc("bib.xml")/bib
for $c in $b/book or $b/article
where $c/author = "Mary"
return
{<$c/title>
 <$b/year>}
```

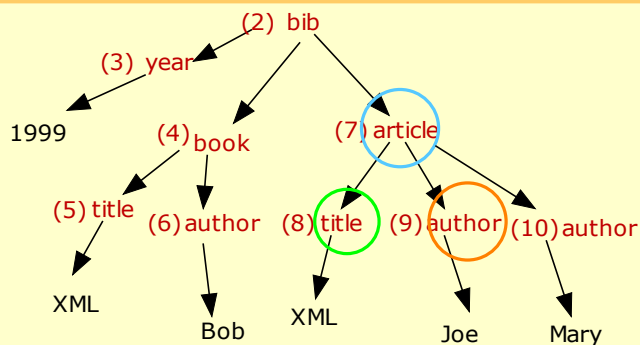


# Query Focus

- Without knowing the document structure, the user can still specify **WHICH** nodes should be meaningfully related

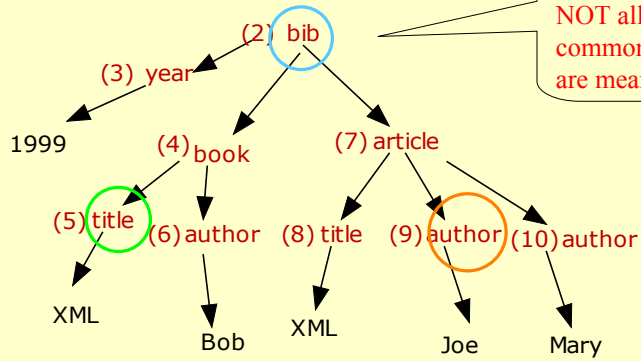


# Intuition



- A node in a XML document usually represent a real-world entity
- Two nodes are related to each other by their lowest common ancestor  
*article* is one of the most specific entities that contain entity *title* and *author*

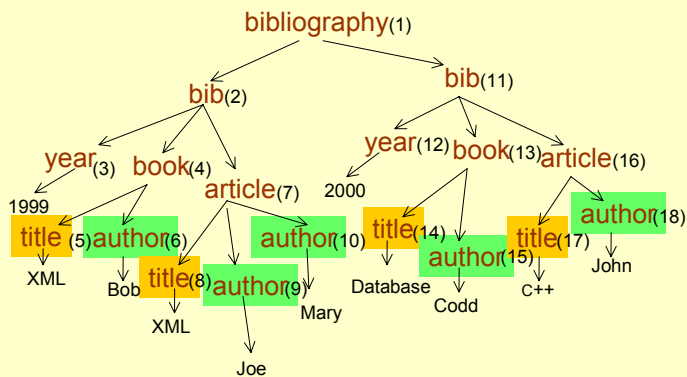
# Intuition



The entity represented by a lowest common ancestor node may not be the most specific type of entity that contains the types of entities each of the nodes represents

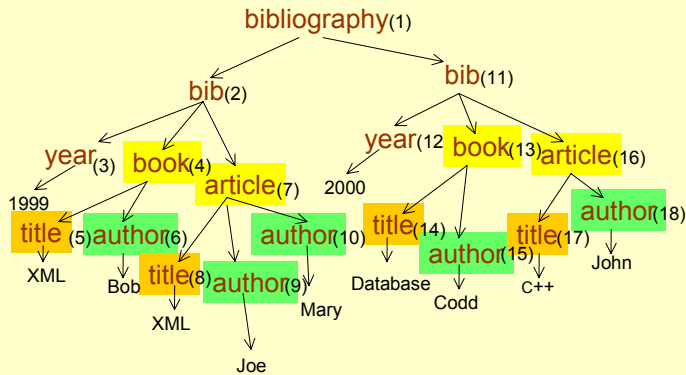
# Meaningful Lowest Common Ancestor

Given the set of nodes with tag *title* and the set of nodes with tag *author*



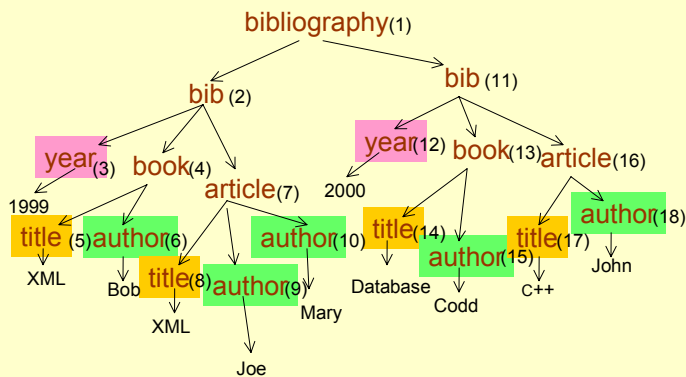
## Meaningful Lowest Common Ancestor

Given the set of nodes with tag title and the set of nodes with tag author, MLCA(title,author) are



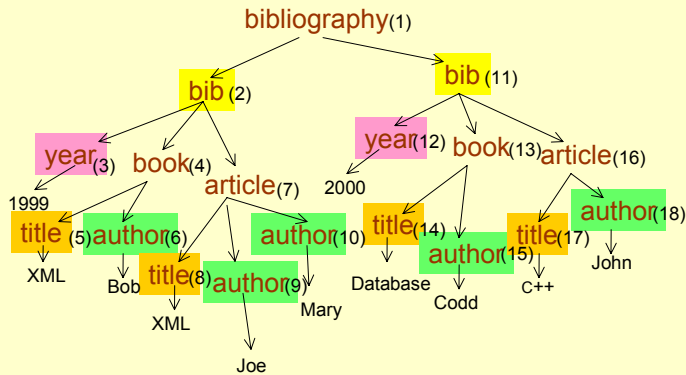
## Meaningful Lowest Common Ancestor

Given the sets of nodes with tag name title, year and author



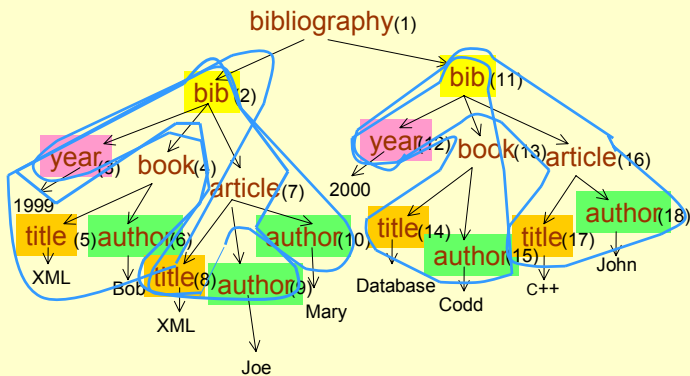
## Meaningful Lowest Common Ancestor

Given the sets of nodes with tag name title, year and author  
MLCA(title, year, author) are



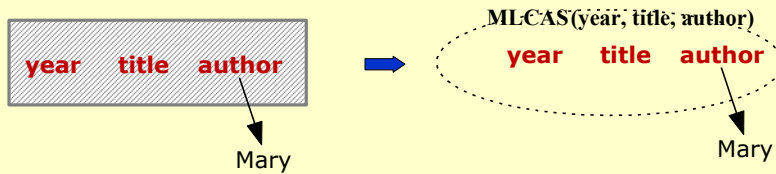
## Meaningful Lowest Common Ancestor Structure

Given the sets of nodes with tag name title, year and author ,  
MLCAS of these nodes are



## Proposal

- The MLCAS (Meaningful Lowest Common Ancestor Structure) is the most specific XML structure in which the nodes are related.
- Our proposal: nodes within a MLCAS are meaningfully related.



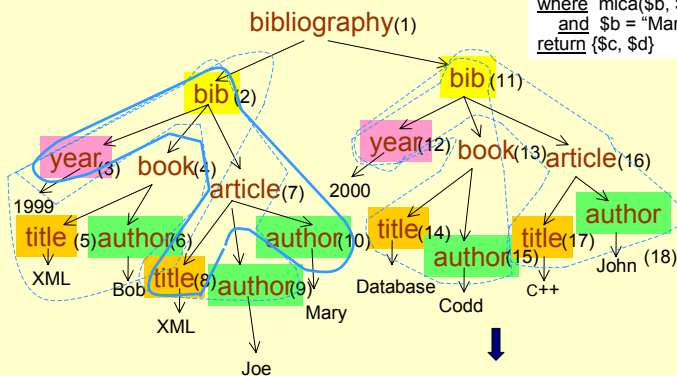
## MLCAS-embedded XQuery

**Query 1:** What are the titles and years of the publications, of which Mary is an author?

```

for $b in doc("bib.xml")//author
  $c in doc("bib.xml")// title
  $d in doc("bib.xml")//year
where mca($b, $c, $d)
and $b = "Mary"
return {$c, $d}

```



```

<title>XML</title>
<year>1999</year>

```

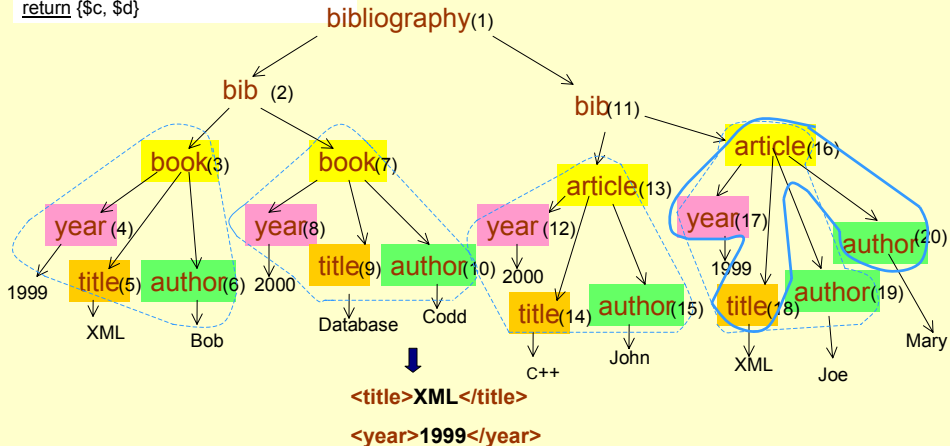
# MLCAS-embedded XQuery

```

for $b in doc("bib.xml")//author
  $c in doc("bib.xml")//title
  $d in doc("bib.xml")//year
where mlca($b, $c, $d)
and $b = "Mary"
return {$c, $d}

```

Document structure changed,  
the same query still applies



# MLCAS Transformation

**Query 1:** What are the titles and years of the publications, of which Mary is an author?

```

for $b in mlcas doc("bib.xml")//author
  $c in mlcas doc("bib.xml")//title
  $d in mlcas doc("bib.xml")//year
where $b = "Mary"
return $c, $d

```

↓ Query transformation

```

for $b in doc("bib.xml")//author
  $c in doc("bib.xml")//title
  $d in doc("bib.xml")//year
where mlca($b, $c, $d)
and $b = "Mary"
return $c, $d

```

# Term expansion

---

User may not know the exact tag names

```
for $b in mlcas doc("bib.xml")//expand(writer)
  $c in mlcas doc("bib.xml")//title
  $d in mlcas doc("bib.xml")//year
where $b = "Mary"
return $c, $d
```

↓ term expansion

```
for $b in doc("bib.xml")//author
  $c in doc("bib.xml")//title
  $d in doc("bib.xml")//year
where mlca($b, $c, $d)
  and $b = "Mary"
return $c, $d
```

# Roadmap

---

- Introduction and Motivation
- Data Model
- **Core algorithm**
- Experimental Evaluation
- Conclusion and Future Work

# Core Algorithm - Overview

## Basic Idea:

Lists of Input nodes ordered by their position in XML document tree



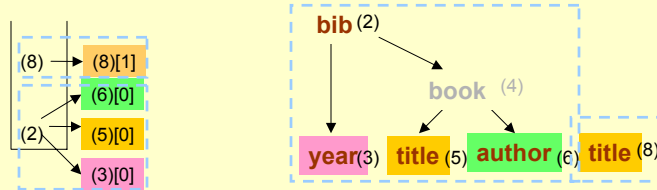
Merge the input nodes into trees rooted at their MLCAs



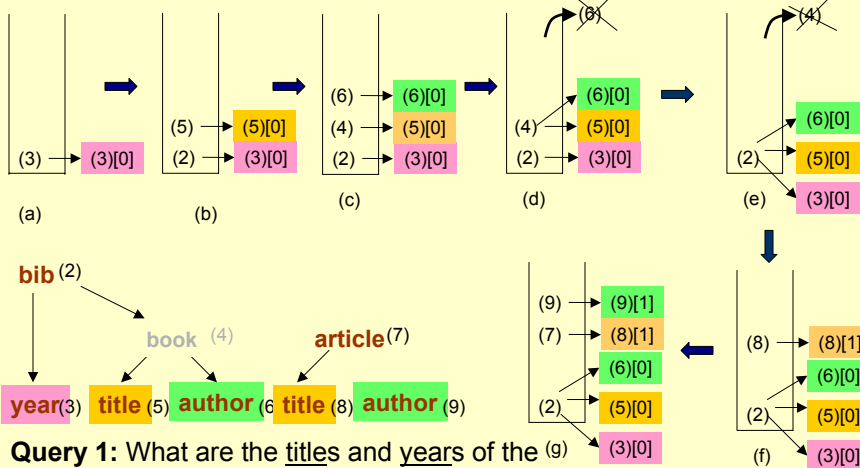
Merge nodes in each such tree into MLCASs

## Main data structure:

A stack with the head of each stack node being a descendant of the head of stack node below it.

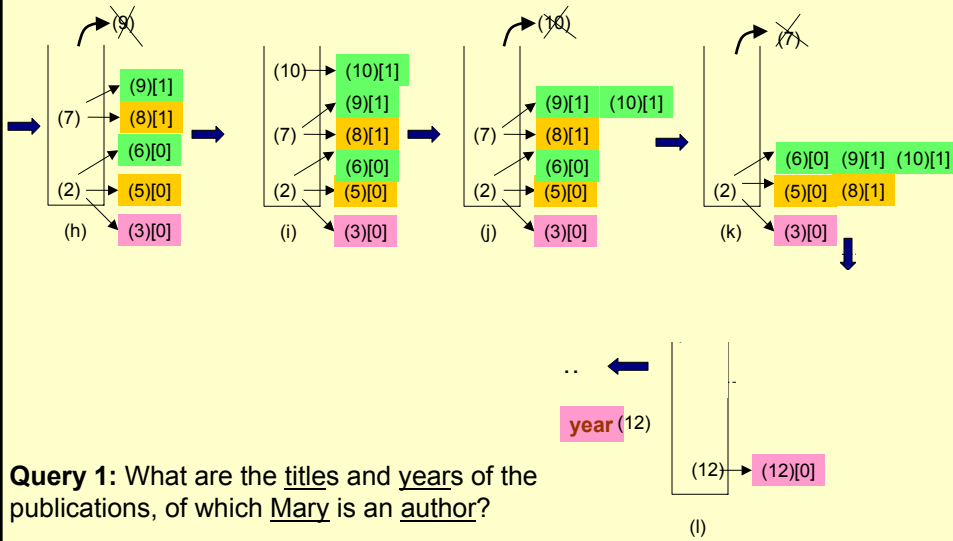


# Core Algorithm - Example

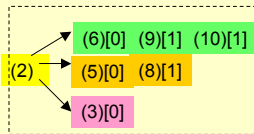


**Query 1:** What are the titles and years of the publications, of which Mary is an author?

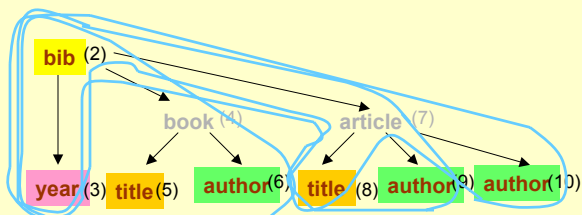
# Core Algorithm - Example



# Core Algorithm - Example



↓ Merge output stack node into MLCASs



# Roadmap

---

- Introduction and Motivation
- Data Model
- Core Algorithm
- **Experimental Evaluation**
- Conclusion and Future Work

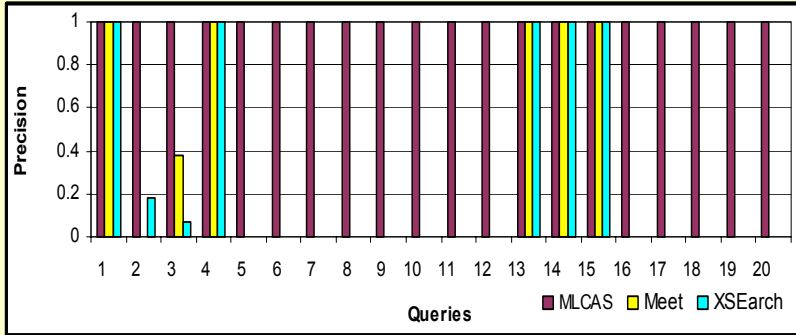
# Experiment Setup

---

- **Running Environment:**
  - Hardware: P III 800MHZ PC machine, 512MB RAM, 120GB Hard drive
  - Software: Windows 2000 Professional OS
- **Dataset:**
  - Xmark (50MB)
  - Publication list collection: collection of personal home pages of personal publication list
    - Similar content, different structures (6 schema families)
  - DBLP

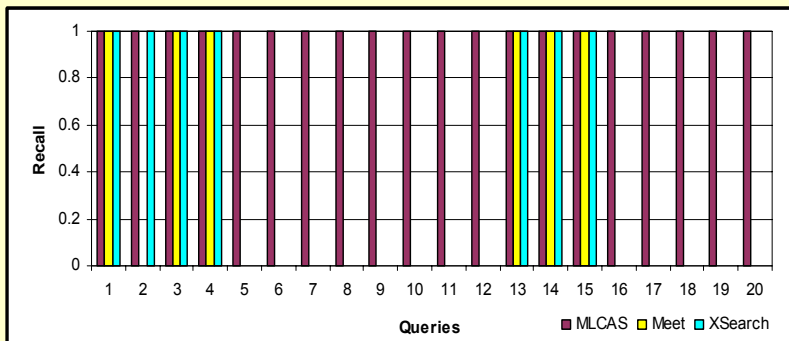
# Search Quality - XMark

Precision of different search strategies



# Search Quality - XMark

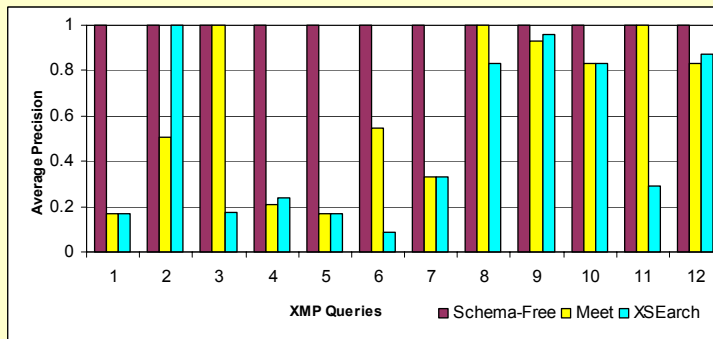
Recall of different search strategies



## Search Quality – Publication Collection

---

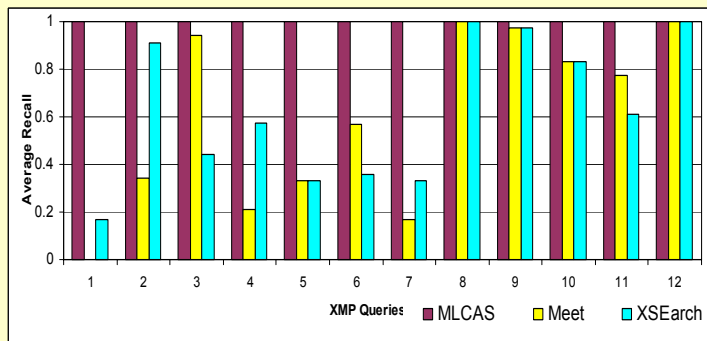
Precision of different search strategies



## Search Quality – Publication Collection

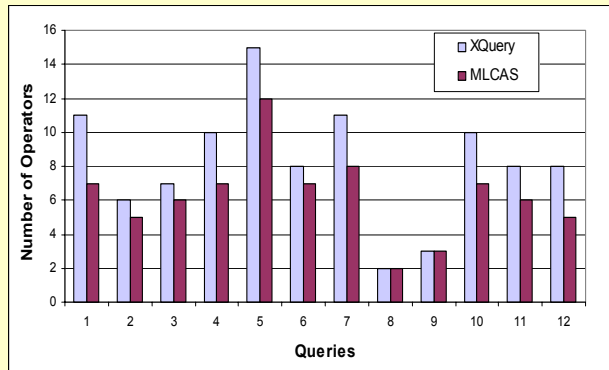
---

Recall of different search strategies



# Performance

Average number of operators



# Performance

Performance (seconds) of XQuery, MLCAS, and COMPOSE for the "XMP" queries on DBLP

<i>Query</i>	<i>XQuery</i>	<i>MLCAS</i>	<i>COMPOSE</i>
1	530	1533	25621
2	290	1173	DNF
3	527	1421	DNF
4	479	1665	26591
5	1132	2518	DNF
6	371	1116	DNF
7	552	1469	24590
8	240	243	241
9	237	240	240
10	367	1456	24536
11	511	1321	DNF
12	473	1088	DNF

**MLCAS vs. COMPOSE:** up to 16 times speed up

**MLCAS vs. XQuery:** less than 4 times overhead

# Roadmap

---

- Introduction and Motivation
- Data Model
- Core algorithm
- Experimental Evaluation
- Conclusion and Future Work

# Conclusion

---

- Schema-Free XQuery enable users to take full advantage of XQuery without requiring full knowledge of document structure
- Partial schema knowledge can be exploited to full advantage
- Robustness against schema changes - potentially of value to data integration, or data evolution
- High search quality with performance overhead less than 4.

## Future Work

---

- Ontology-based term expansion
- Expand MLCAS operator to enable queries involving attributes and references
- Investigate optimization techniques, and ranking and relaxing techniques
- Database selection

Any Questions?

---

# Thank you!

---

## Search Quality

---

- **Comparison Study:**

- Meet:

- Keyword search
- Answer: sub-tree rooted with the closest ancestor node of keywords

- XSearch:

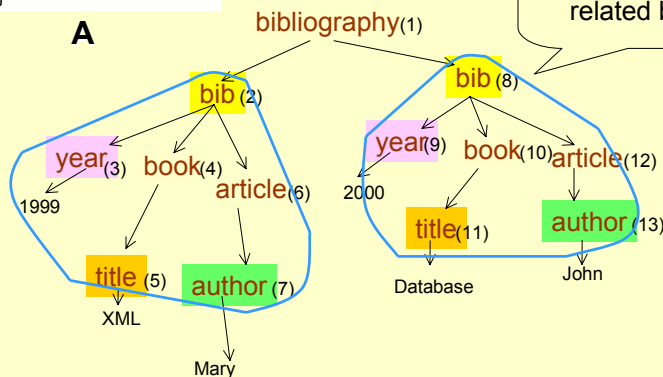
- Distinguish tag name & value
- (tag name, value) pairs
- Two nodes are meaningfully related if they are interconnected
- **All-pair answer:** all-pair of nodes are interconnected
- **Star-pair answer:** all nodes are interconnected with one node

# Related Work

- Keyword Search
  - Relational Database
    - DBXplorer (Agrawal et al. ICDE 2002)
    - DISCOVER (Hristidis et al. VLDB 2002)
    - Keyword Search in Graph Databases (Goldman et. al. VLDB 98)
    - BANKS (Bhalotia, G., et al., ICDE. 2002)
  - XML documents
    - Meet (Schmidt, A., et al. ICDE, 2001)
    - Xkeyword (Hristidis et al. ICDE, 2003)
    - XRANK (Guo, L., et al. SIGMOD, 2003)
  - Extend structure query language with keyword search
    - XIRQL ( Fuhr, N., et al, SIGIR, 2000)
    - LOREL(Quass, D., et al.,1995)
    - Florescu, D., et al., 2000
- Other
  - XSEarch (Cohen, S., et al, VLDB, 2003)

# MLCAS can go wrong

```
for $b in document("bib.xml")//author
  $c in document("bib.xml")// title
  $d in document("bib.xml")// year
where exist mlcas($b, $c, $d)
and $b = "Mary"
return {$c, $d}
```

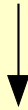


## Representation of tree node

---

- Let the set of nodes in a tree be  $N$ .
- Each node is represented by  $(docID, startPos, endPos, Level)$

**a**  $(docID, StartPos1, EndPos1, Level1)$



**b**  $(docID, StartPos2, EndPos2, Level2)$

## MCCAS-extended Xquery

---

- User-given XQuery:

```
FOR var1 in expr1(tag1)
  ...
  varm in exprm(tagm)
LET varm+1 := exprm+1(tagm+1)
  ...
  varn := exprn(tagn)
WHERE conditionExpr(var1, ..., varn)
RETURN returnExpr(var1, ..., varn)
SYSTEM_HELP_ON
```

1. Tag names should be either included in the FOR or LET clause.

2. Turn the system\_help on

## MCCAS-extended XQuery (cont.)

---

- Recursively apply the following transform operator to the original Xquery statement queryX: transform(queryX)
  - transform(queryX) = queryX, if there is no FLWR statement inside queryX
  - else

## MCCAS-extended XQuery (cont.)

---

1. FOR clause:  
FOR var<sub>1</sub> in transform(expr<sub>1</sub>(tag<sub>1</sub>))  
...  
var<sub>m</sub> in transform(expr<sub>m</sub>(tag<sub>m</sub>))
2. LET clause: apply transform operator to each expressions  
LET transform(expr<sub>m+1</sub>(tag<sub>m+1</sub>))  
...  
var<sub>n</sub> := transform(expr<sub>n</sub>(tag<sub>n</sub>))

## MCCAS-extended XQuery (cont.)

---

### 3. WHERE clause

a. if  $m > 1$

```
WHERE mccas(var1, ..., varm)
      AND transform(conditionExpr(var1, ..., varn))
```

b. if  $m \leq 1$

```
WHERE transform(conditionExpr(var1, ..., varn))
```

### 4. RETURN clause

- RETURN transform(returnExpr((var<sub>1</sub>, ..., var<sub>n</sub>)))

## Differences in tag names?

---

- This is a well-understood problem
  - WordNet® - an online lexical reference system. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept.
  - Ontology mapping tools: Chimaera, PROMPT, GLUE, etc
- Tag names tend to be domain specific, which makes tag name mapping even simpler

## Queries didn't run

---

- **User case query 4**

For each author in the bibliography, list the author's name and the titles of all books by that author, grouped inside a "result" element

- Timber doesn't support some & satisfies.

- **User case query 8**

Find books in which the name of some element ends with the string "or" and the same element contains the string "Suciu" somewhere in its content. For each such book, return the title and the qualifying element.

- Timber doesn't support end-with