Querying Complex Structured Databases

Cong Yu – Yahoo! Research
H. V. Jagadish – Univ. of Michigan

presented by **Arnab Nandi** – Univ. of Michigan

VLDB 2007, Vienna, Austria
September 27th, 2007

Querying Complex Structured Databases

–

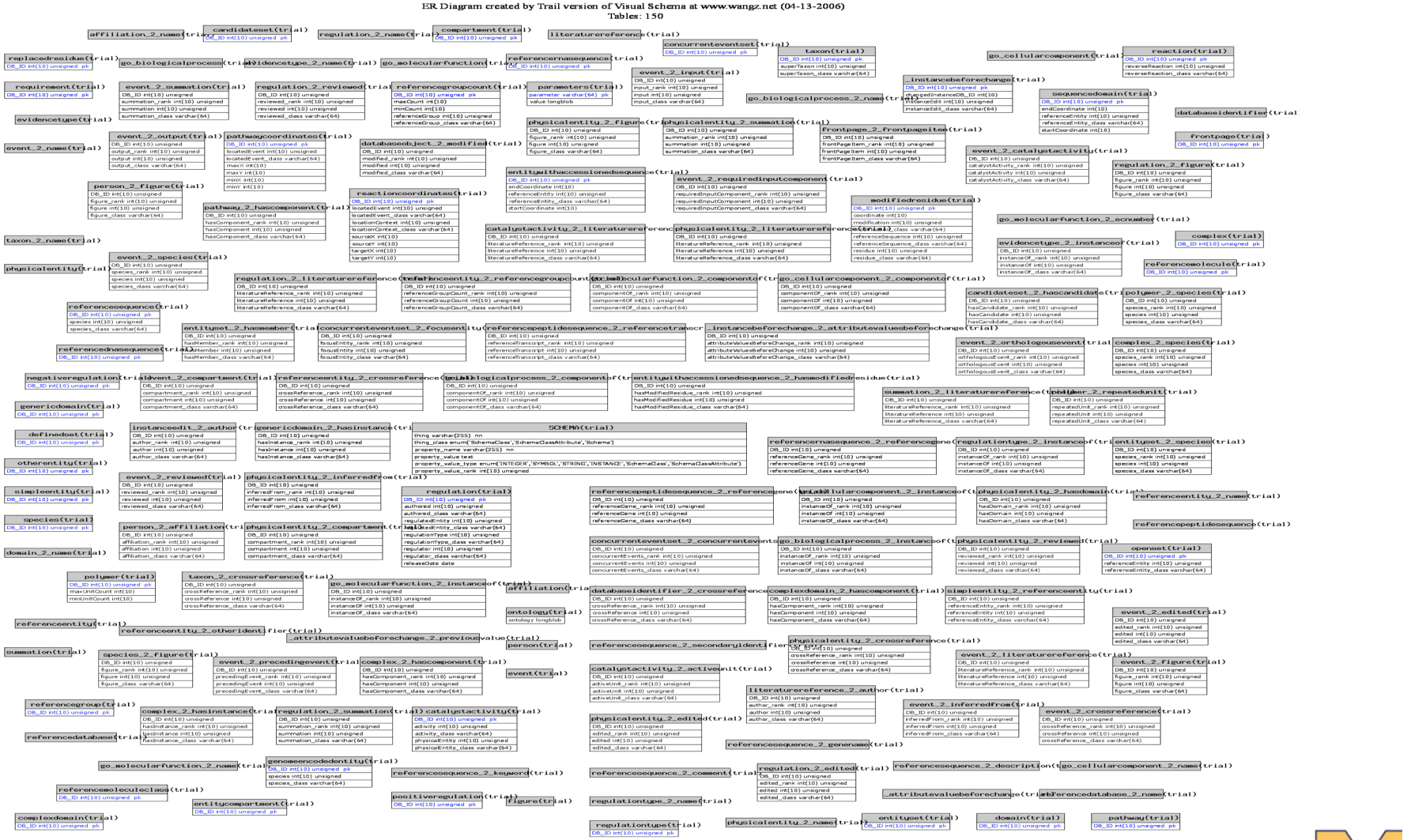
A step toward database usability

Cong Yu – Yahoo! Research
H. V. Jagadish – Univ. of Michigan

presented by **Arnab Nandi** – Univ. of Michigan

VLDB 2007, Vienna, Austria
September 27th, 2007

Databases Are Complex



University of Mic

Databases Are Complex

The *usability costs* are gradually becoming the bottleneck for many database applications!

Database Usability

- A recent focus of our research group to address the following challenges facing the “real” users of databases:

Database Usability

- A recent focus of our research group to address the following challenges facing the “real” users of databases:
  - Unknown/complicated query language
  - unknown/complex schema
  - lack of instantaneous feedback on results
  - lack of effective tracking of provenance
  - rigid process of database content creation

Database Usability

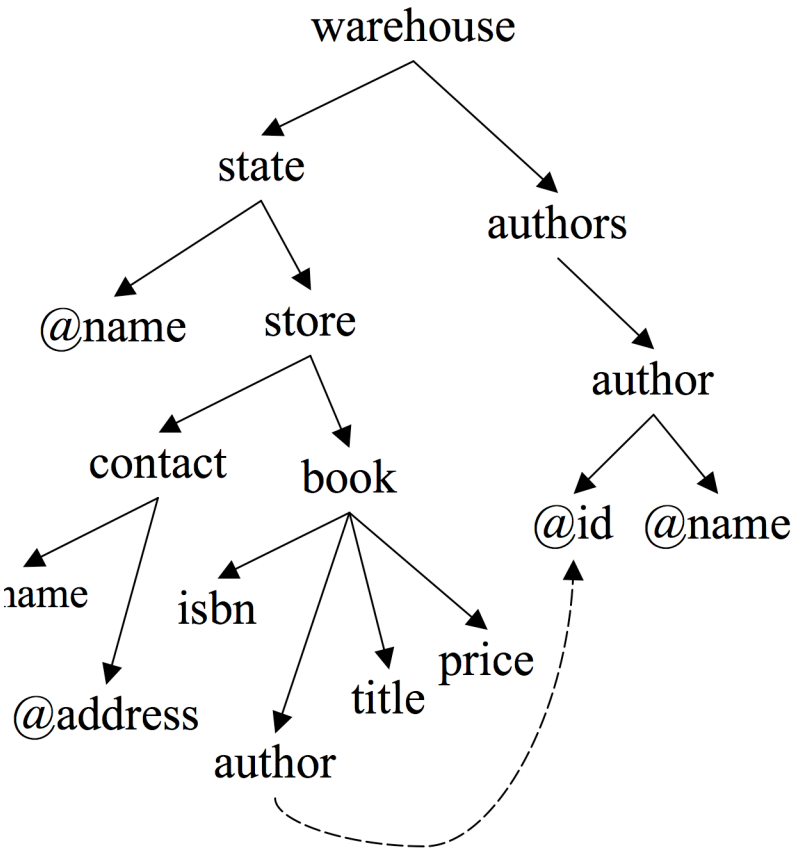
- A recent focus of our research group to address the following challenges facing the “real” users of databases:

Unknown/complex schema

Challenge of Query Language

- SQL/XQuery are too complicated for our users (i.e., biologists) to learn and use
  - Some of those users are even quite tech-savvy!
  - Still, they do not have the working knowledge of SQL/XQuery to pose the queries they have in mind
  - Simpler query interfaces is much more preferred

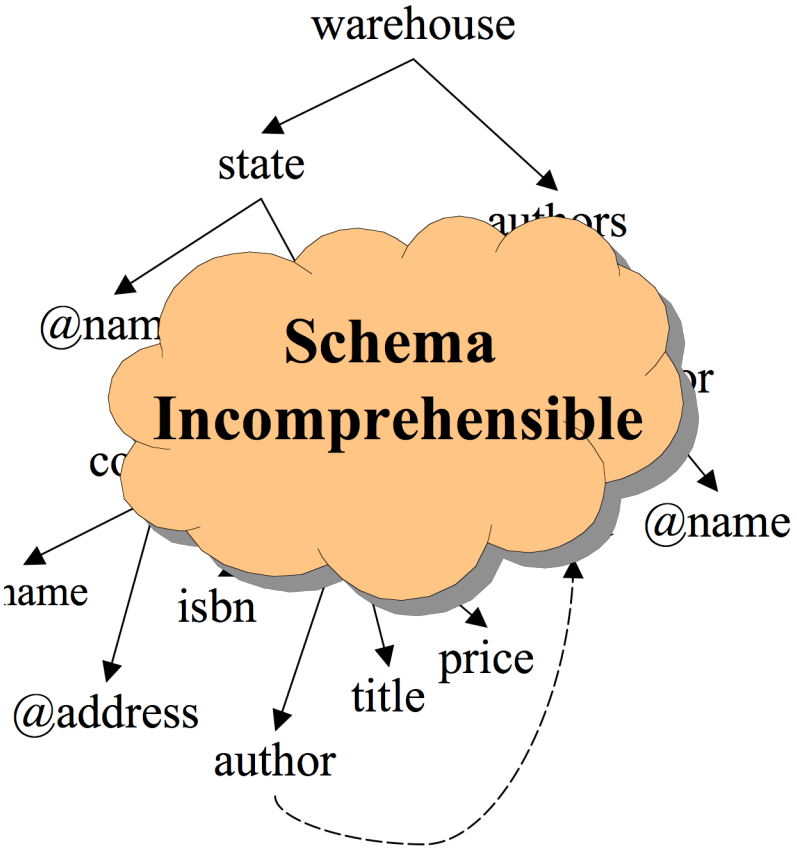
Challenge of Schema



XQuery:

```
for   $a in doc()//author,
      $s in doc()//store
let   $b in $s/book
where $s/contact/@name =
   "Amazon" and $b/author =
  $a/id
return { $a/name, count($b) }
```

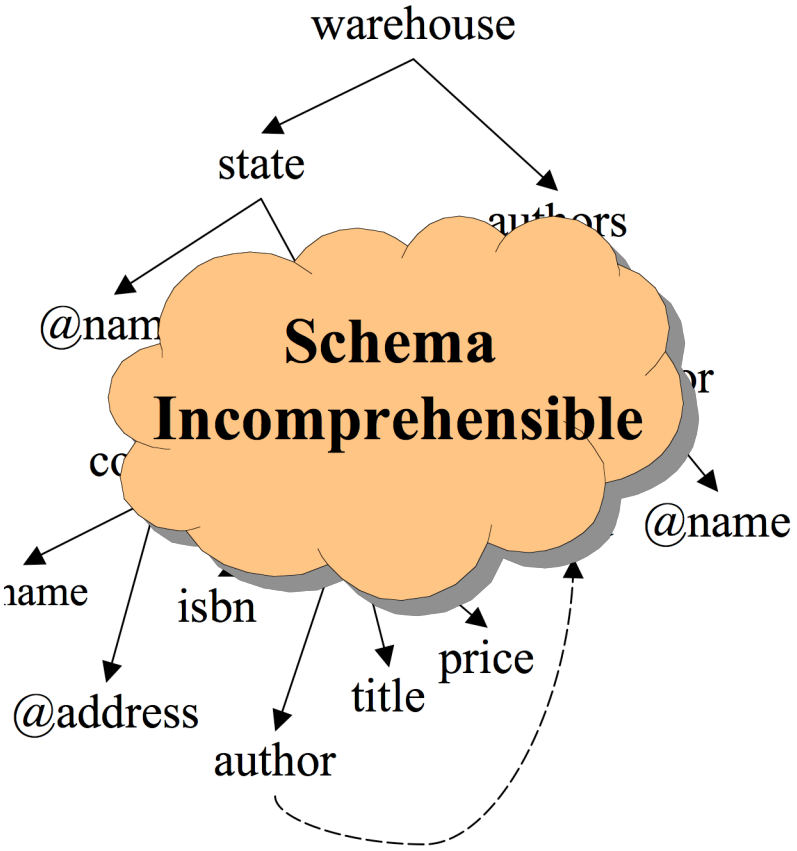
Challenge of Schema



XQuery:

```
for   $a in doc()//author,
      $s in doc()//store
let   $b in $s/book
where $s/contact/@name =
   "Amazon" and $b/author =
  $a/id
return { $a/name, count($b) }
```

Challenge of Schema



XQuery:

```
for   $
      $s
let   $
where
  “Am                    hor =
  $a/id
return { $a/name, count($b) }
```

Our Goal

- Establish a novel query model based on three principles:
  - Flexible requirement on **schema knowledge**
  - Maintaining the same **query result quality** as a structured query can achieve
  - Minimum increase in **query evaluation cost**

Our Goal

- Establish a novel query model based on three principles:
  - Flexible requirement on **schema knowledge**
  - Maintaining the same **query result quality** as a structured query can achieve
  - Minimum increase in **query evaluation cost**
- Built upon several previous works
  - Structure-free query models
  - Schema summarization

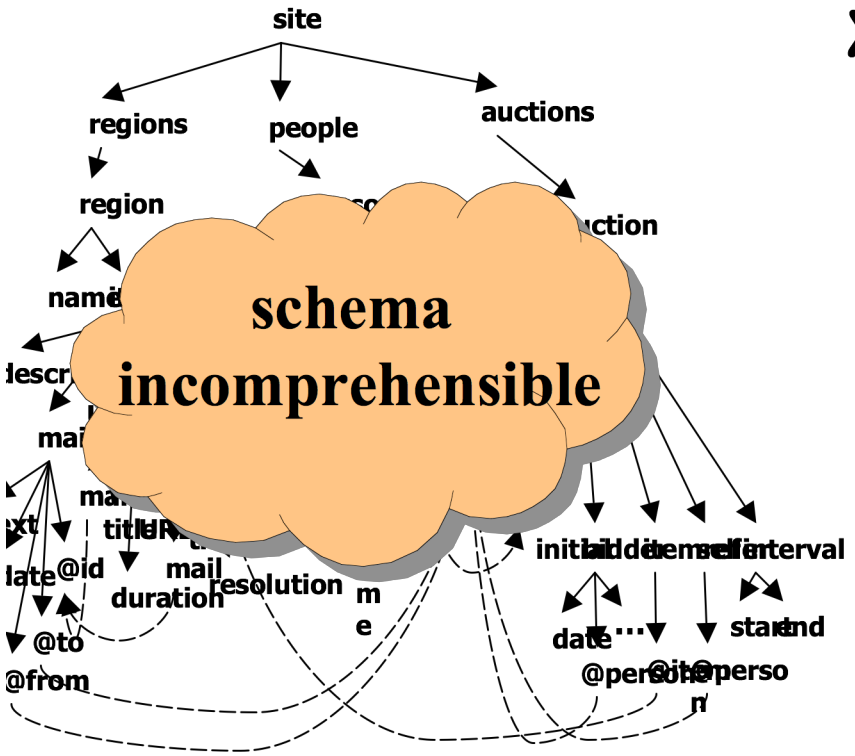
Outline

- Background and Motivation
- Issues with Current Approaches
- Schema-Based Matching Semantics
- Meaningful Summary Query Model
- Conclusion

Outline

- Background and Motivation
- Issues with Current Approaches
- Schema-Based Matching Semantics
- Meaningful Summary Query Model
- Conclusion

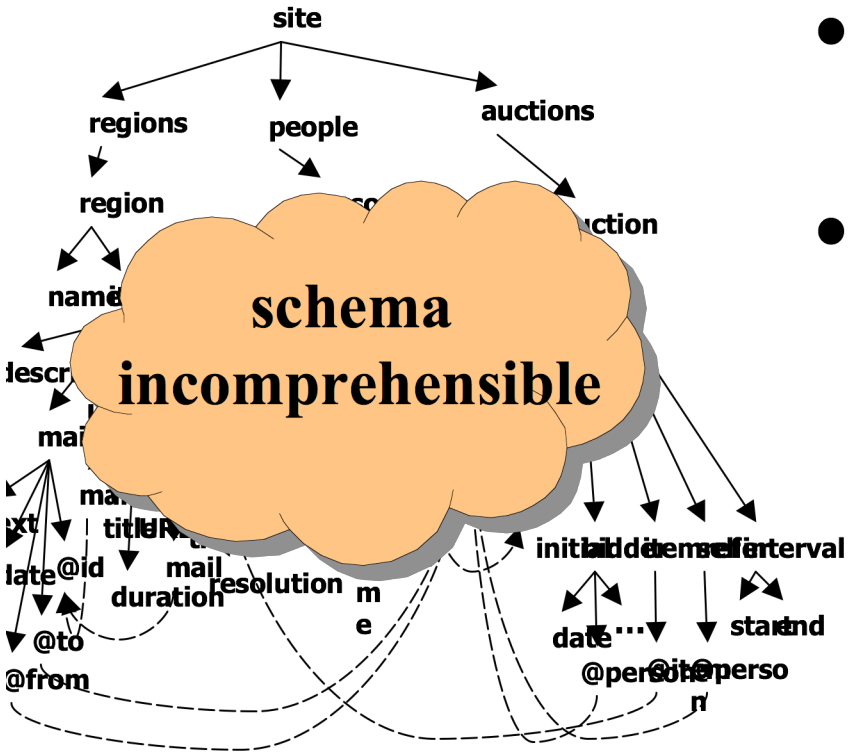
Structure-Free Query Model



XQuery:

```
for   $a in doc()//region/item,
      $p in doc()//person
let   $b in $p//watch
where $p//@name =
   "Peter" and $p/@auction =
  $b/id
   and $a/@id =
  $b/item/@itemref
return { $p/name, count($b) }
```

Structure-Free Query Model



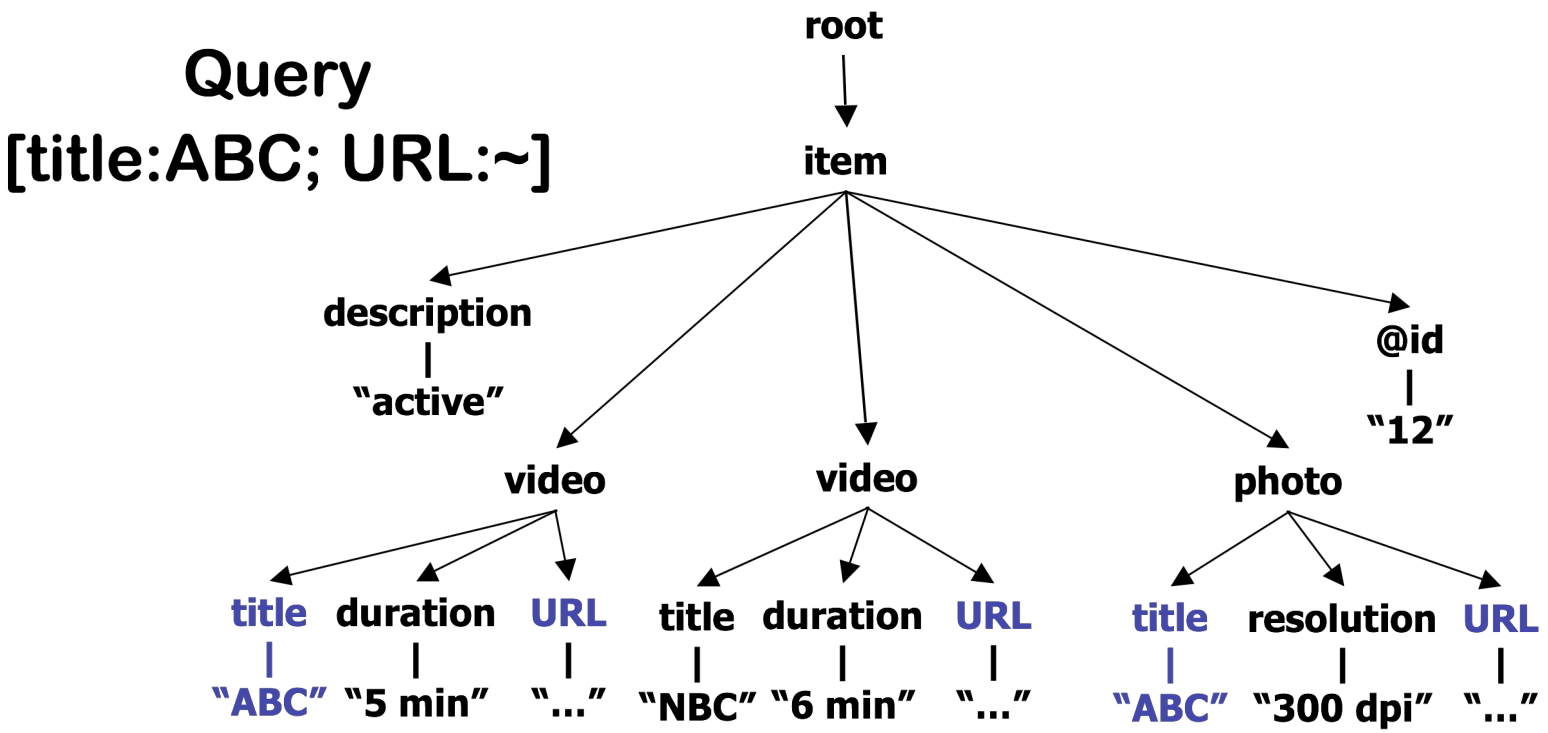
- Simple Keyword Search
  [name; peter; address; asia]
- Labeled Keyword Search
  [name:peter; address:asia]

A Simple Example

- XML: Lowest Common Ancestor
- Relational: Smallest Tuple Graph

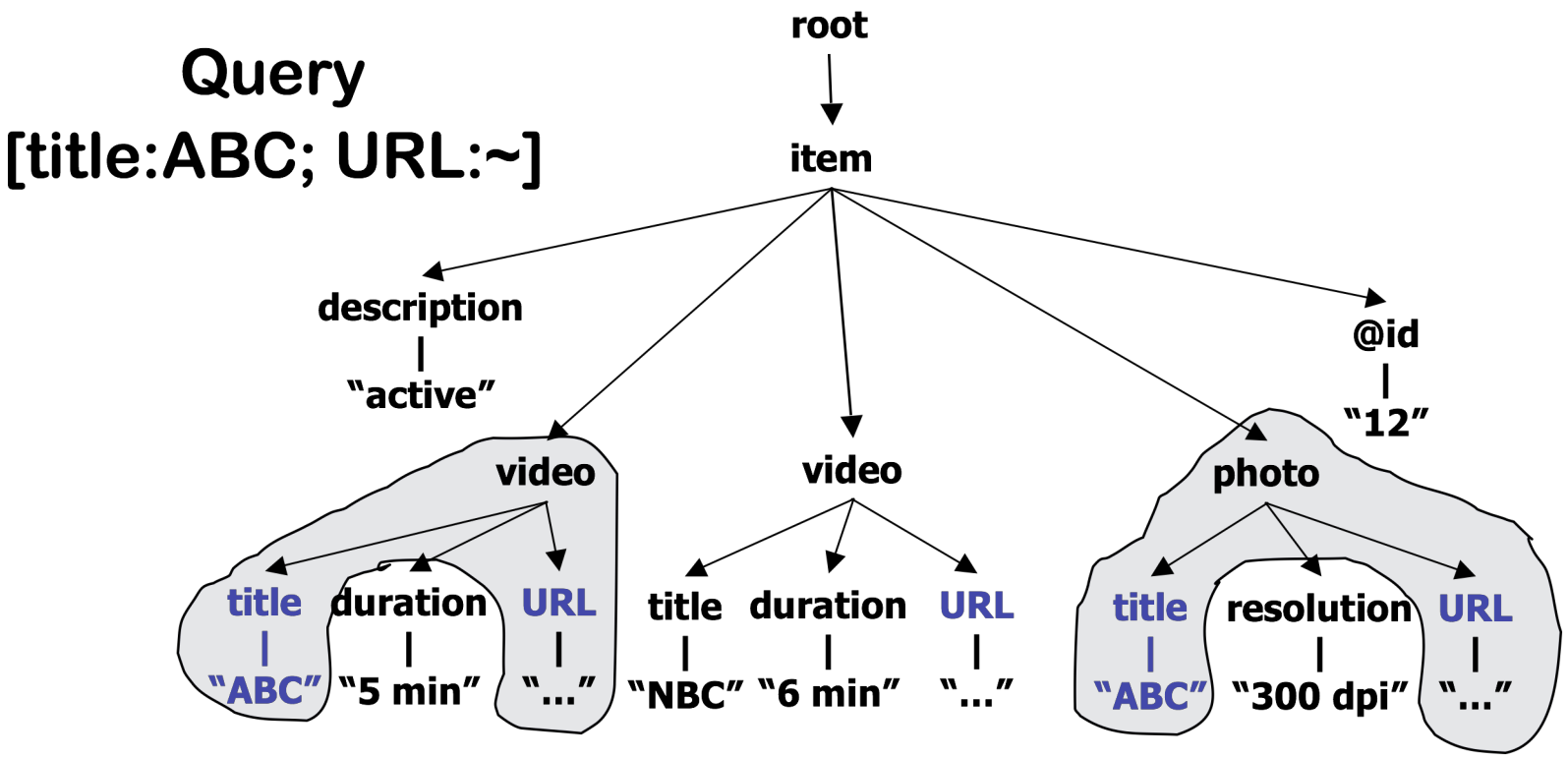
A Simple Example

- XML: Lowest Common Ancestor
- Relational: Smallest Tuple Graph



A Simple Example

- XML: Lowest Common Ancestor
- Relational: Smallest Tuple Graph

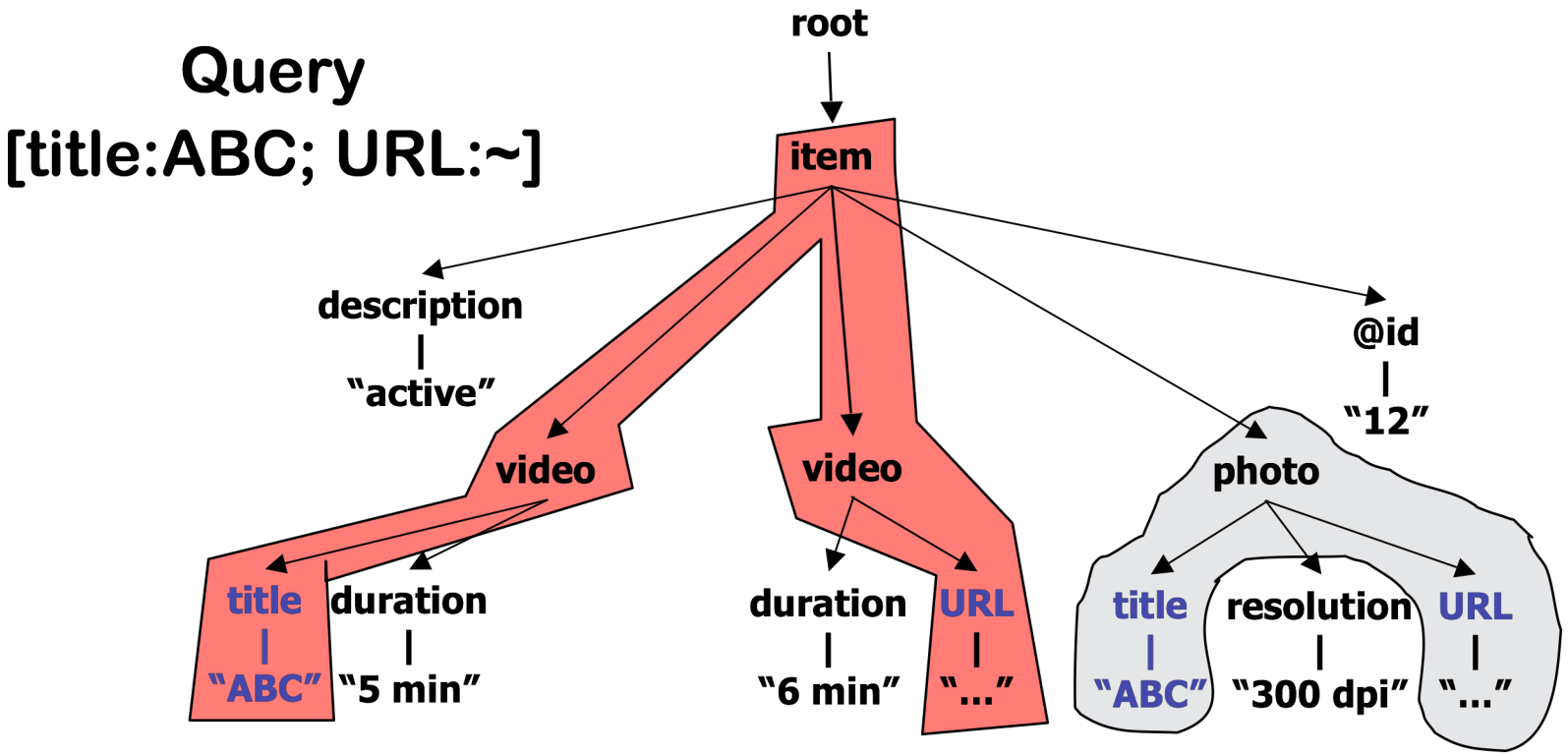


A Simple Example : Issues

Result accuracy is no longer 100%: depends on matching semantics

A Simple Example : Issues

Result accuracy is no longer 100%: depends on matching semantics



Problems with Current Structure-Free Query Models

- Rely on content-based matching semantics
  - High query evaluation cost due to the need to examine non-meaningful data fragments
  - Wrong matches caused by missing data nodes

Problems with Current Structure-Free Query Models

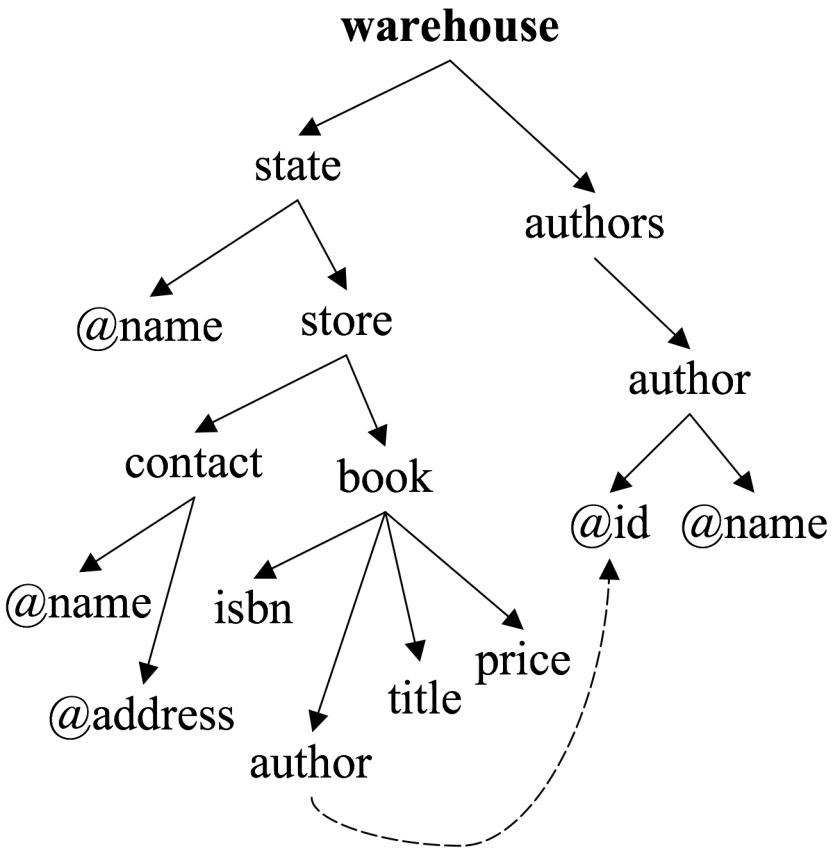
- Rely on content-based matching semantics
  - High query evaluation cost due to the need to examine non-meaningful data fragments
  - Wrong matches caused by missing data nodes
- Schema can become very complex
  - Related elements can be far apart
  - Lack of support for complex queries: e.g., joins, aggregations, etc.

Problems with Current Structure-Free Query Models

- Rely on content-based matching semantics
  - High query evaluation cost due to the need to examine non-meaningful data fragments
  - Wrong matches caused by missing data nodes
- Schema can become very complex
  - Related elements can be far apart
  - Lack of support for complex queries: e.g., joins, aggregations, etc.

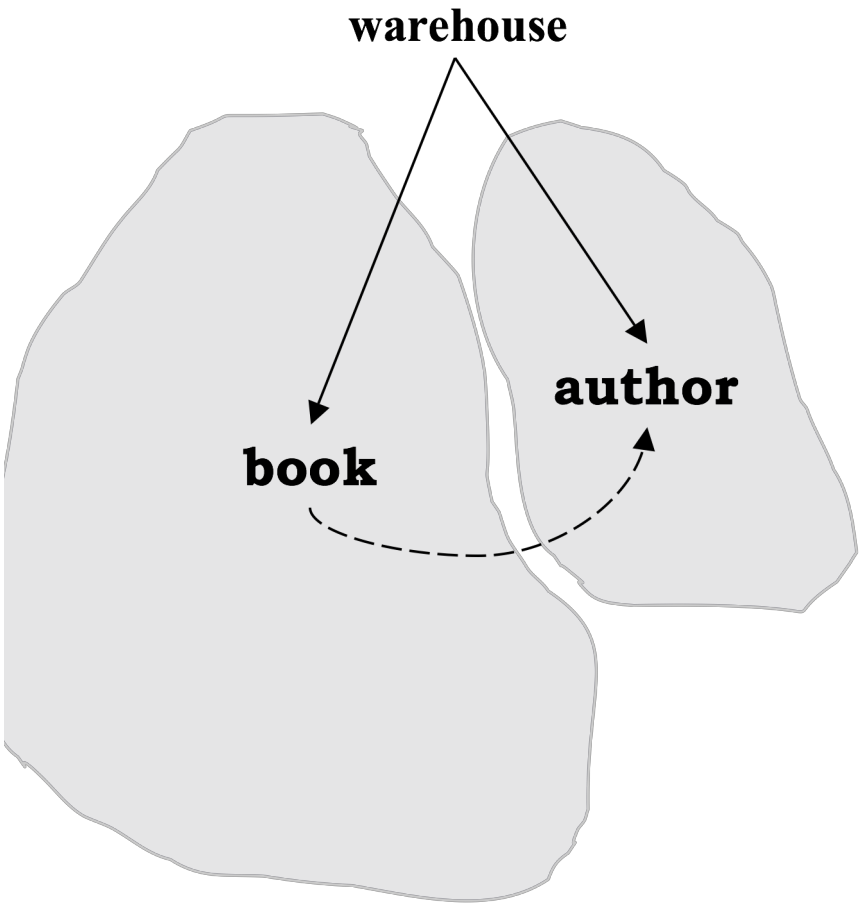
Solution: Schema summary-based query construction

Schema Summary



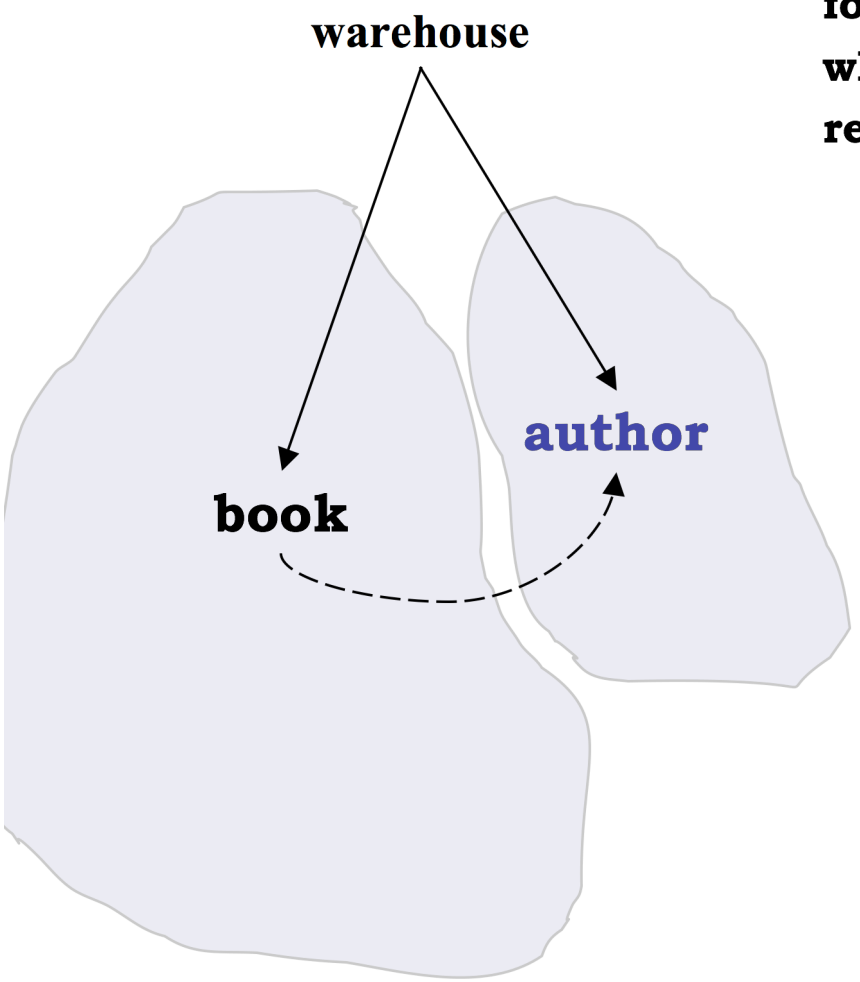
- For database with complex schemas, a much simpler schema that summarizes the database can be generated [YJ06]
- It can help reduce the human cost of query construction

Schema Summary



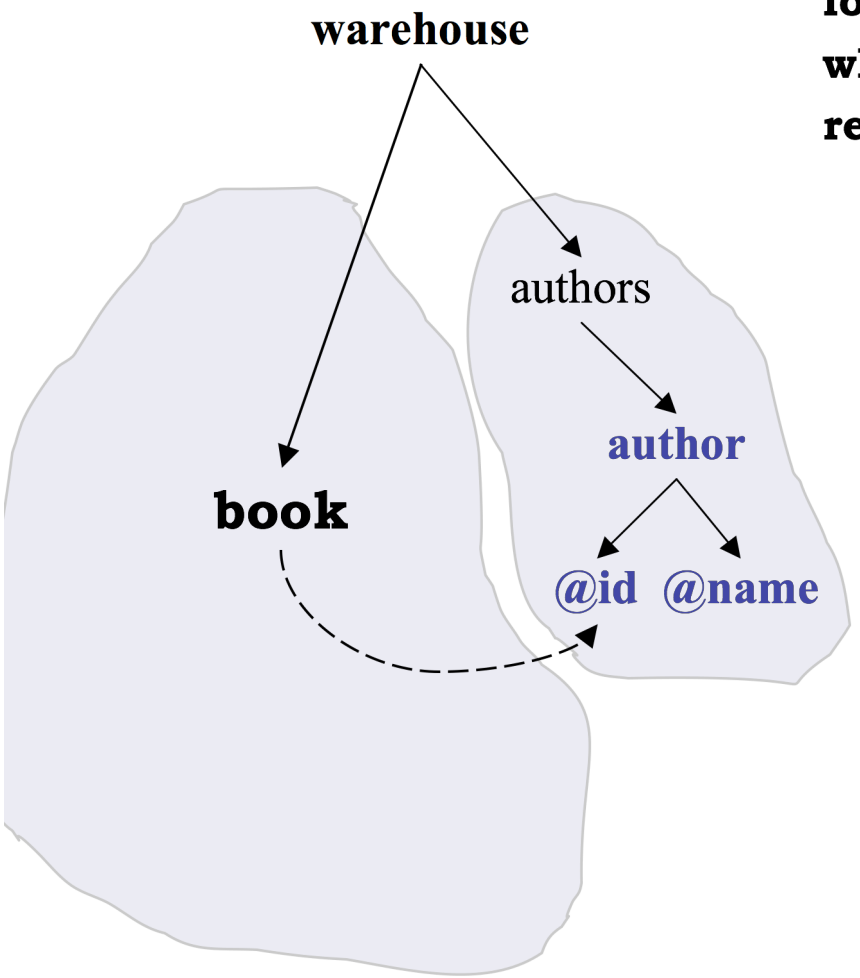
- For database with complex schemas, a much simpler schema that summarizes the database can be generated [YJ06]
- It can help reduce the human cost of query construction

Summary-Based Query Construction



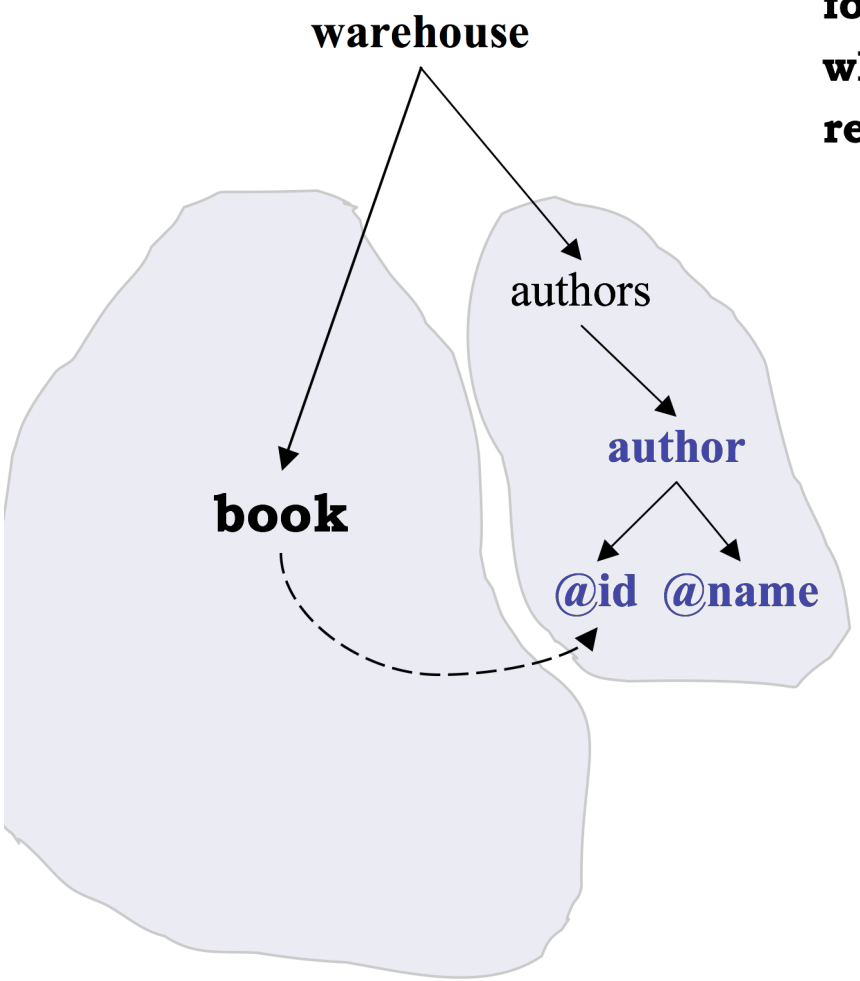
- Consider a simple query:
  for $a in doc()/warehouse/authors/author
  where $a/@name = “Jagadish”
  return $a/@id

Summary-Based Query Construction



- Consider a simple query:
  for $a in doc()/warehouse/authors/author
  where $a/@name = “Jagadish”
  return $a/@id

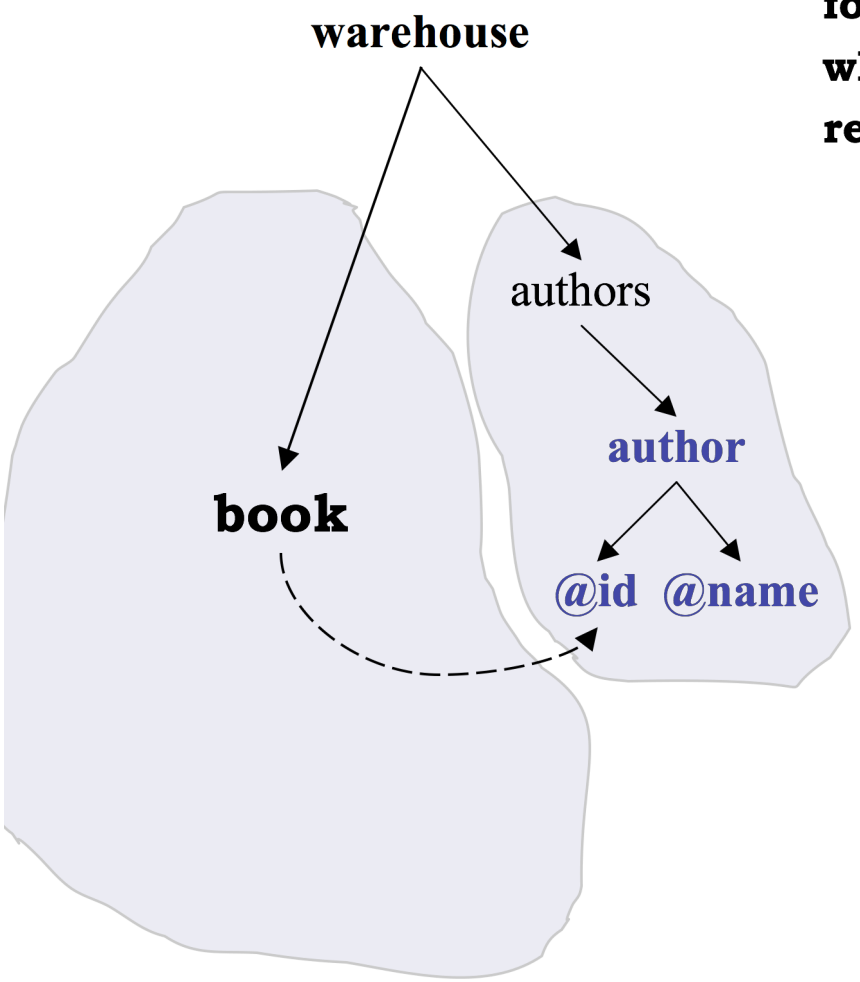
Summary-Based Query Construction



- Consider a simple query:
  for $a in doc()/warehouse/authors/author
  where $a/@name = “Jagadish”
  return $a/@id

- Users can construct a structured query by visiting the **summary**, instead of the entire schema

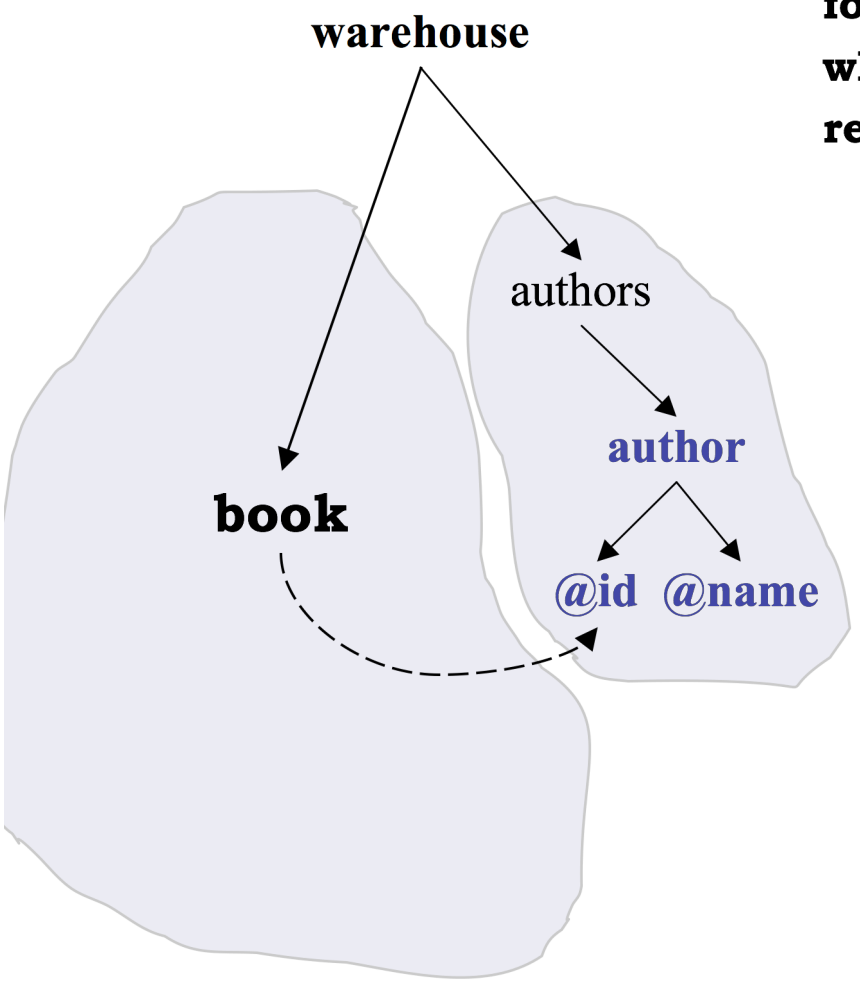
Summary-Based Query Construction



- Consider a simple query:
  for $a in doc()/warehouse/authors/author
  where $a/@name = “Jagadish”
  return $a/@id

- Users can construct a structured query by visiting the **summary**, instead of the entire schema
- Only relevant part of the schema needs to be visited!

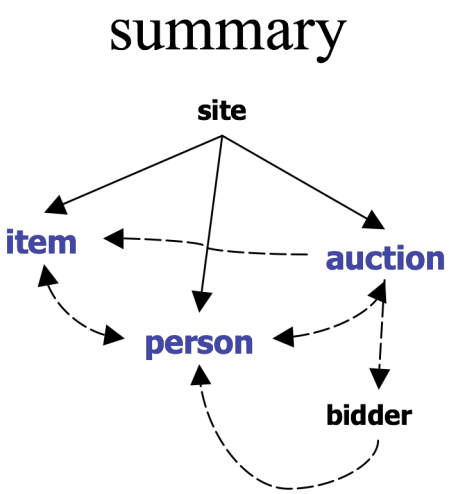
Summary-Based Query Construction



- Consider a simple query:
  for $a in doc()/warehouse/authors/author
  where $a/@name = “Jagadish”
  return $a/@id

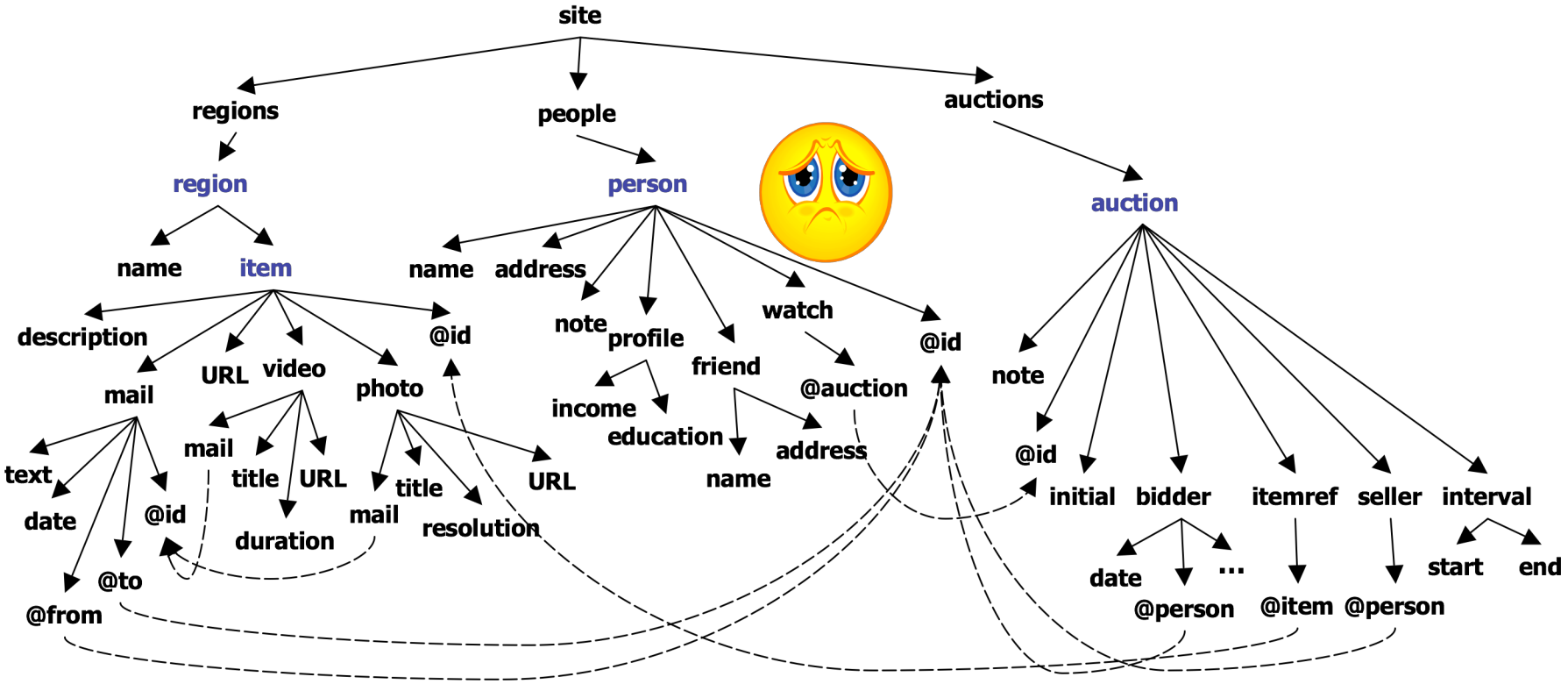
- Users can construct a structured query by visiting the **summary**, instead of the entire schema
- Only relevant part of the schema needs to be visited!
- Problems …

Problem with Complex Queries



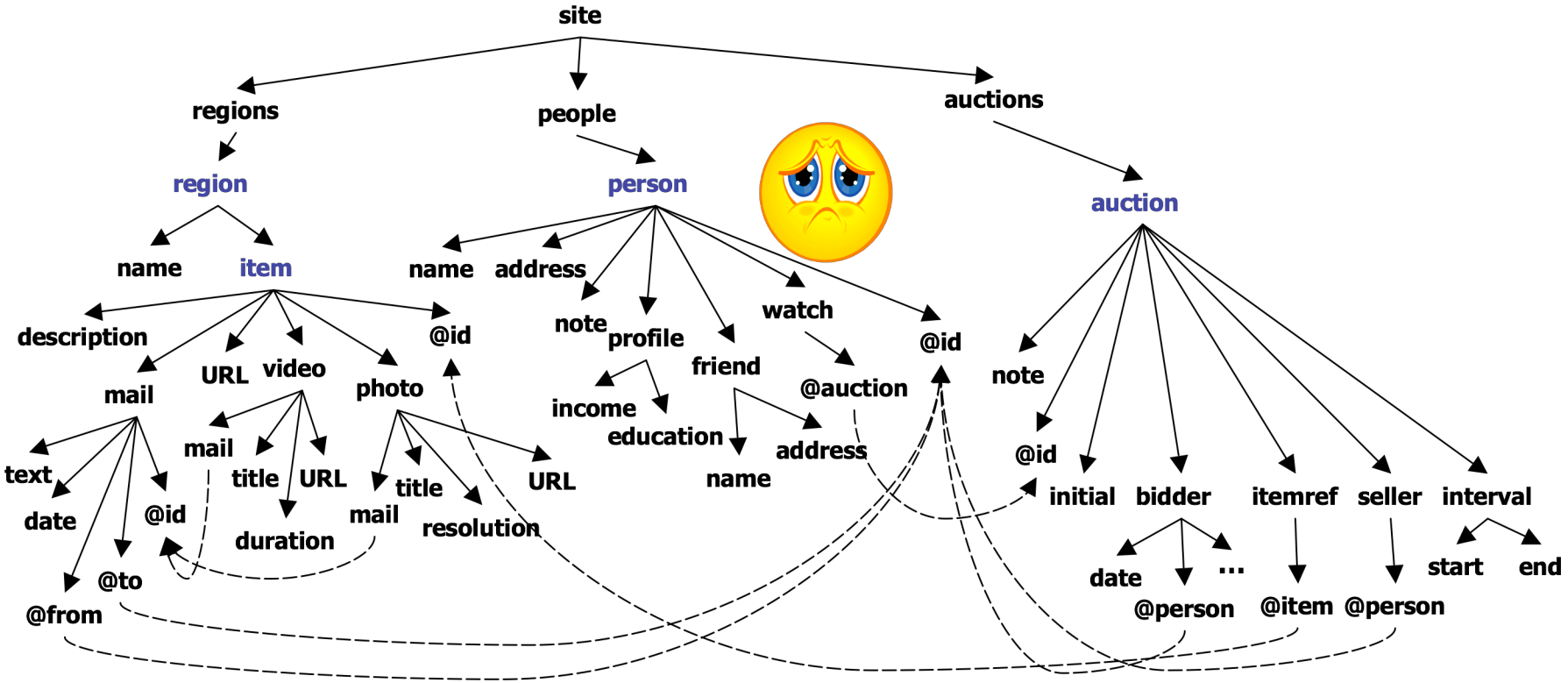
- Query: retrieve auction that are *sold* by the person named “peter” in “chicago” and *contain* items that are “antiques” in region “asia”

Problem with Complex Queries



- Query: retrieve <u>auction</u> that are *sold* by the <u>person</u> named “peter” in “chicago” and *contain* <u>items</u> that are “antiques” in <u>region</u> “asia”

Problem with Complex Queries



Constructing complex queries requires the expansion of the summary into the entire schema!

Our Solution

- Content-based matching semantics
- Curse of query and schema complexity

Our Solution

- Conte tics
- Curse of query and schema complexity

Schema-Based Matching Semantics

Our Solution

- Conte tics
- Curs plexity

Schema-Based Matching Semantics

Summary-Based Query Model

Outline

- Background and Motivation
- Issues with Current Approaches
- Schema-Based Matching Semantics
- Meaningful Summary Query Model
- Conclusion

Schema-Based Matching Semantics

- Based on *labeled keywords* as the basic query condition

Schema-Based Matching Semantics

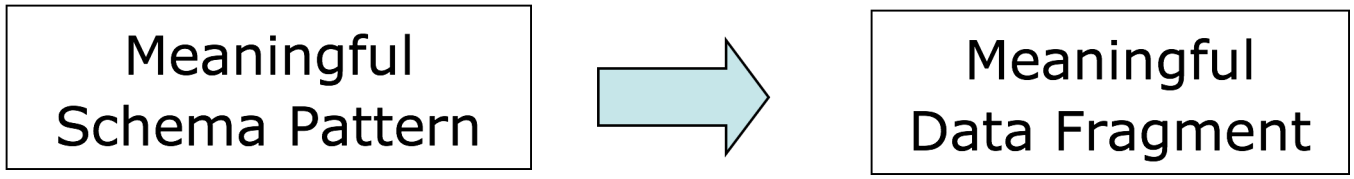
- Based on *labeled keywords* as the basic query condition
- Two components:

Meaningful Schema Pattern

- Meaningful Schema Pattern
  - A schema sub-graph
  - Satisfying certain semantic conditions: Basic / Related-Entity / Non-Redundant

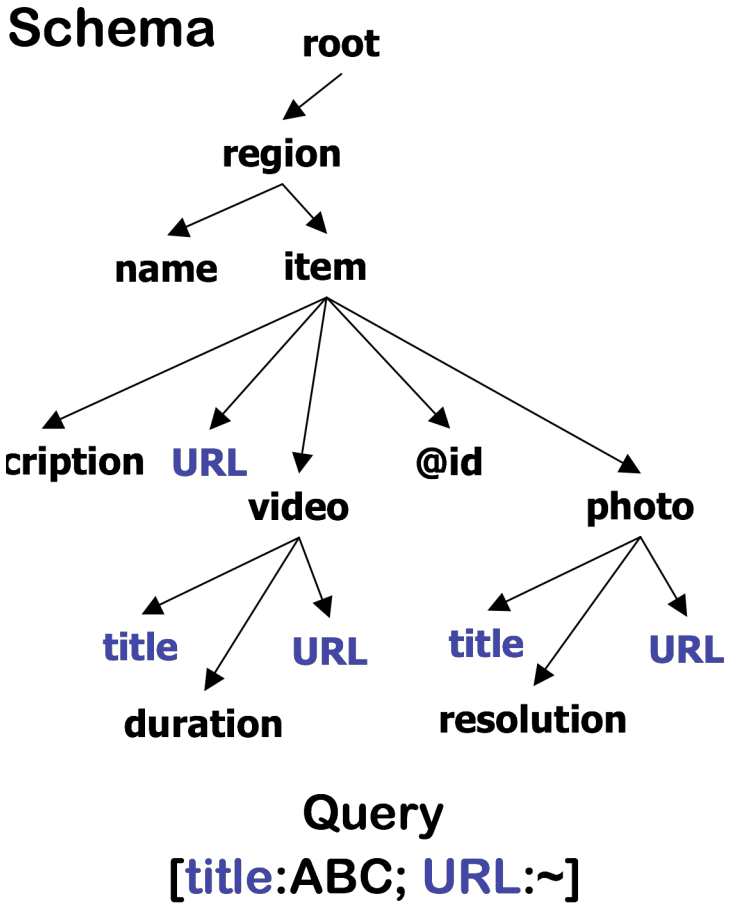
Schema-Based Matching Semantics

- Based on *labeled keywords* as the basic query condition
- Two components:



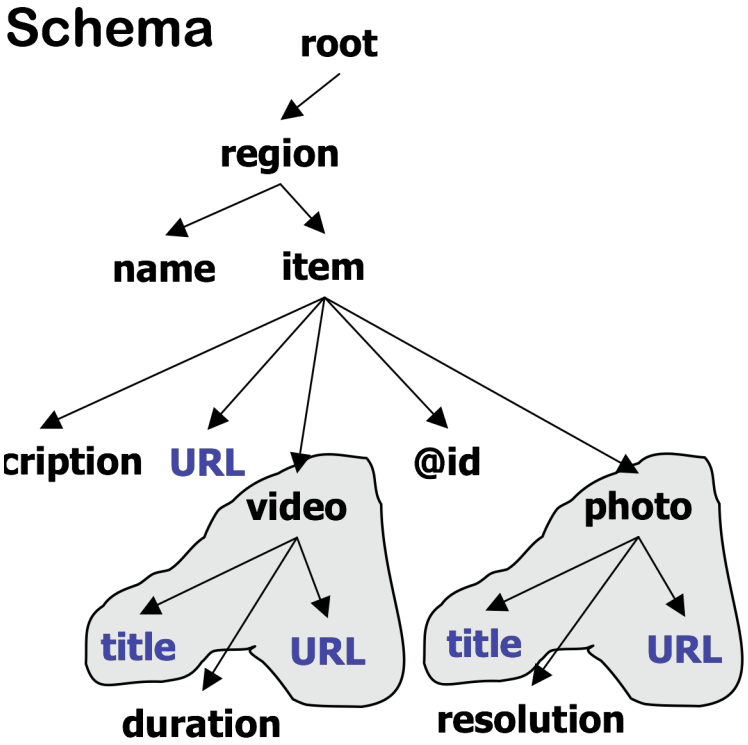
- Meaningful Data Fragment
  - A data subtree
  - Result of querying the database with the meaningful schema pattern

Basic Matching Semantics



- Inspired by content-based semantics
- A schema pattern is meaningful if:
  - Each label in the query is represented
  - All elements in the schema pattern are necessary
- Not accurate enough because not all schema information is leveraged!

Basic Matching Semantics

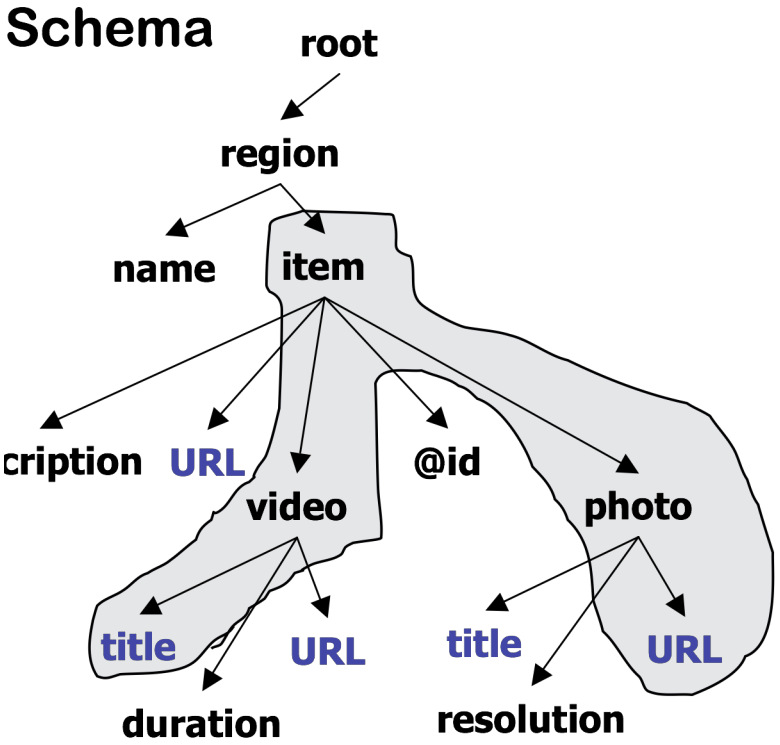


Query

[title:ABC; URL:~]

- Inspired by content-based semantics
- A schema pattern is meaningful if:
  - Each label in the query is represented
  - All elements in the schema pattern are necessary
- Not accurate enough because not all schema information is leveraged!

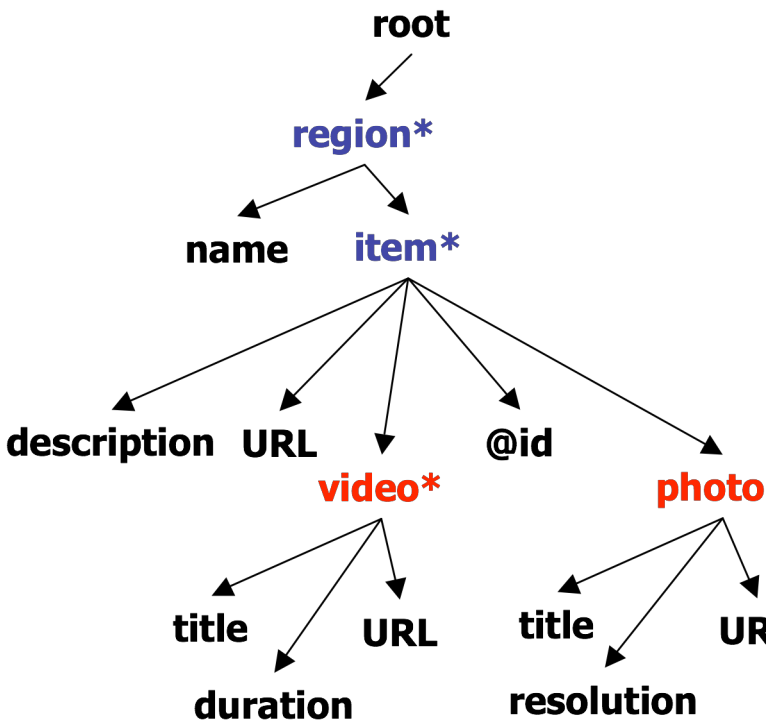
Basic Matching Semantics



- Inspired by content-based semantics
- A schema pattern is meaningful if:
  - Each label in the query is represented
  - All elements in the schema pattern are necessary
- Not accurate enough because not all schema information is leveraged!

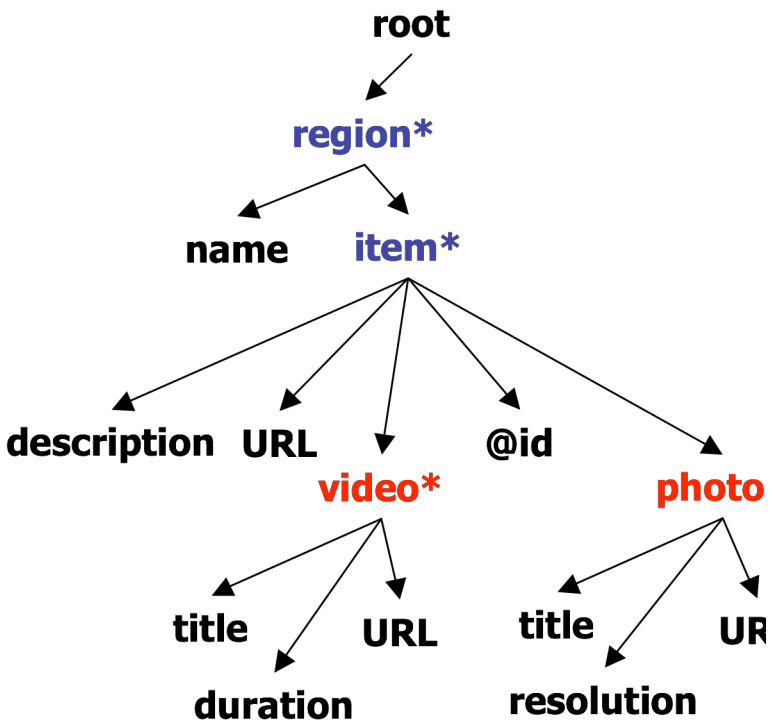
Schema Hints

- Element repeatability
  - Repeatable elements are *entities*
  - Non-repeatable elements are *attributes* of their "parent" entities



Schema Hints

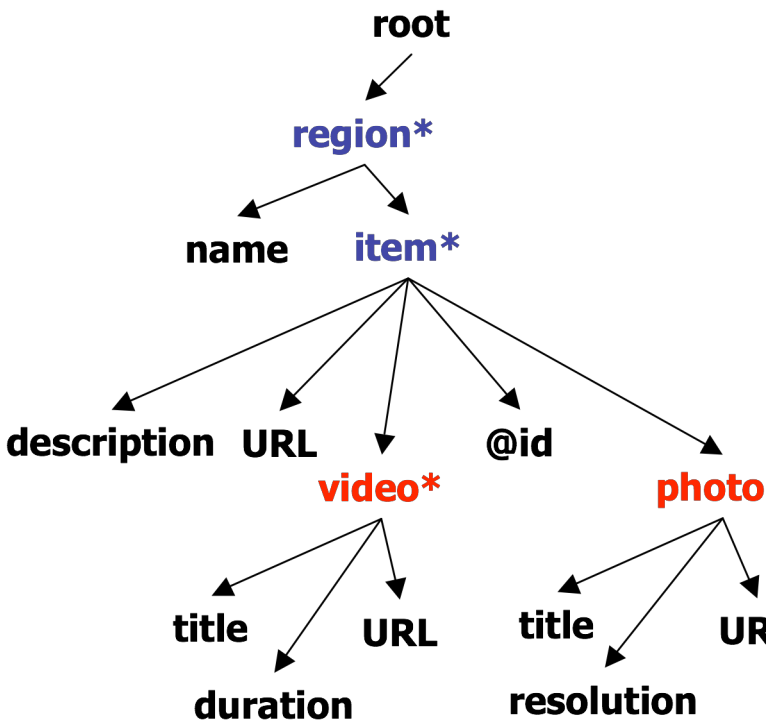
Attributes of the same entity are meaningfully related



Schema Hints

Attributes of the same entity are meaningfully related

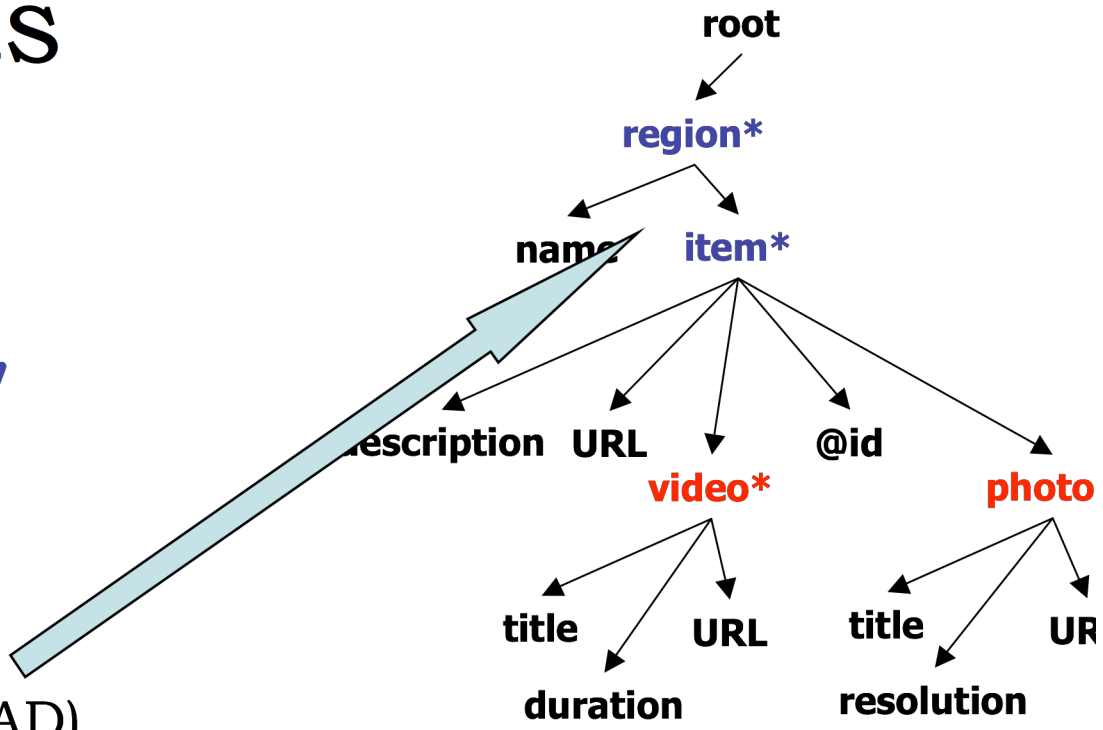
- Relationship between entity elements



Schema Hints

Attributes of the same entity are meaningfully related

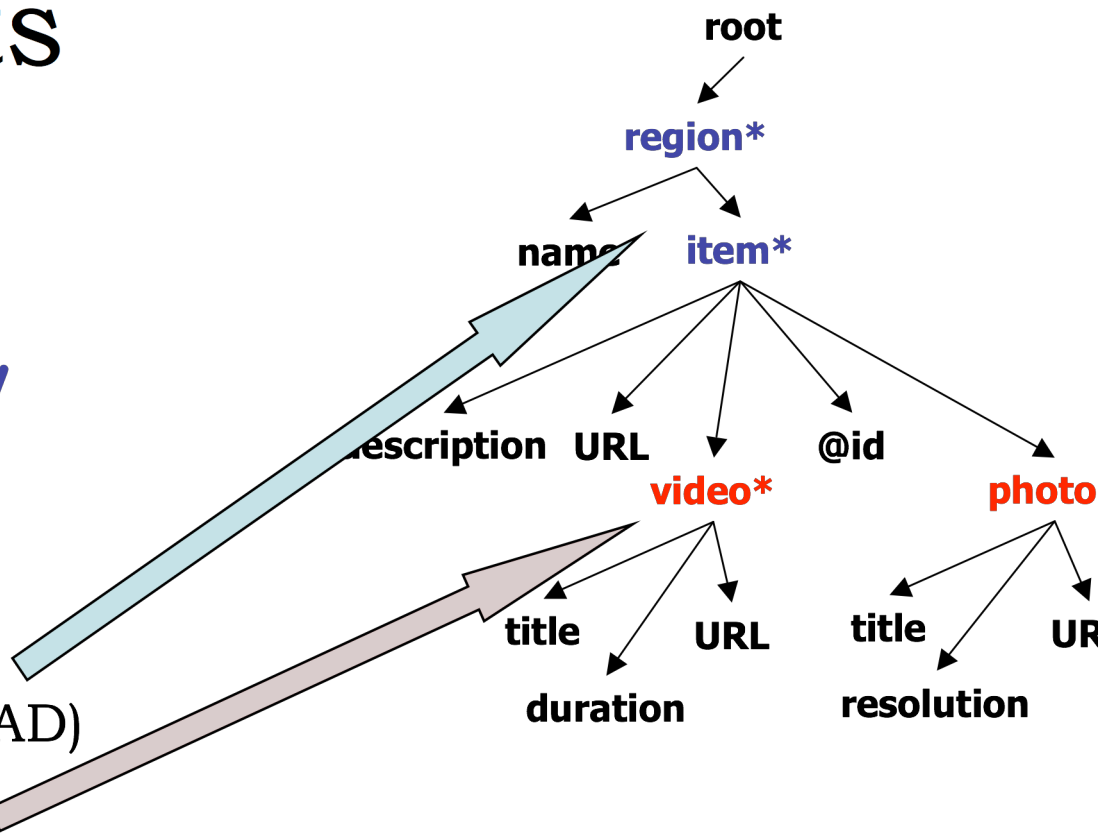
- Relationship between entity elements
  - Ancestor-Descendant (AD)



Schema Hints

Attributes of the same entity are meaningfully related

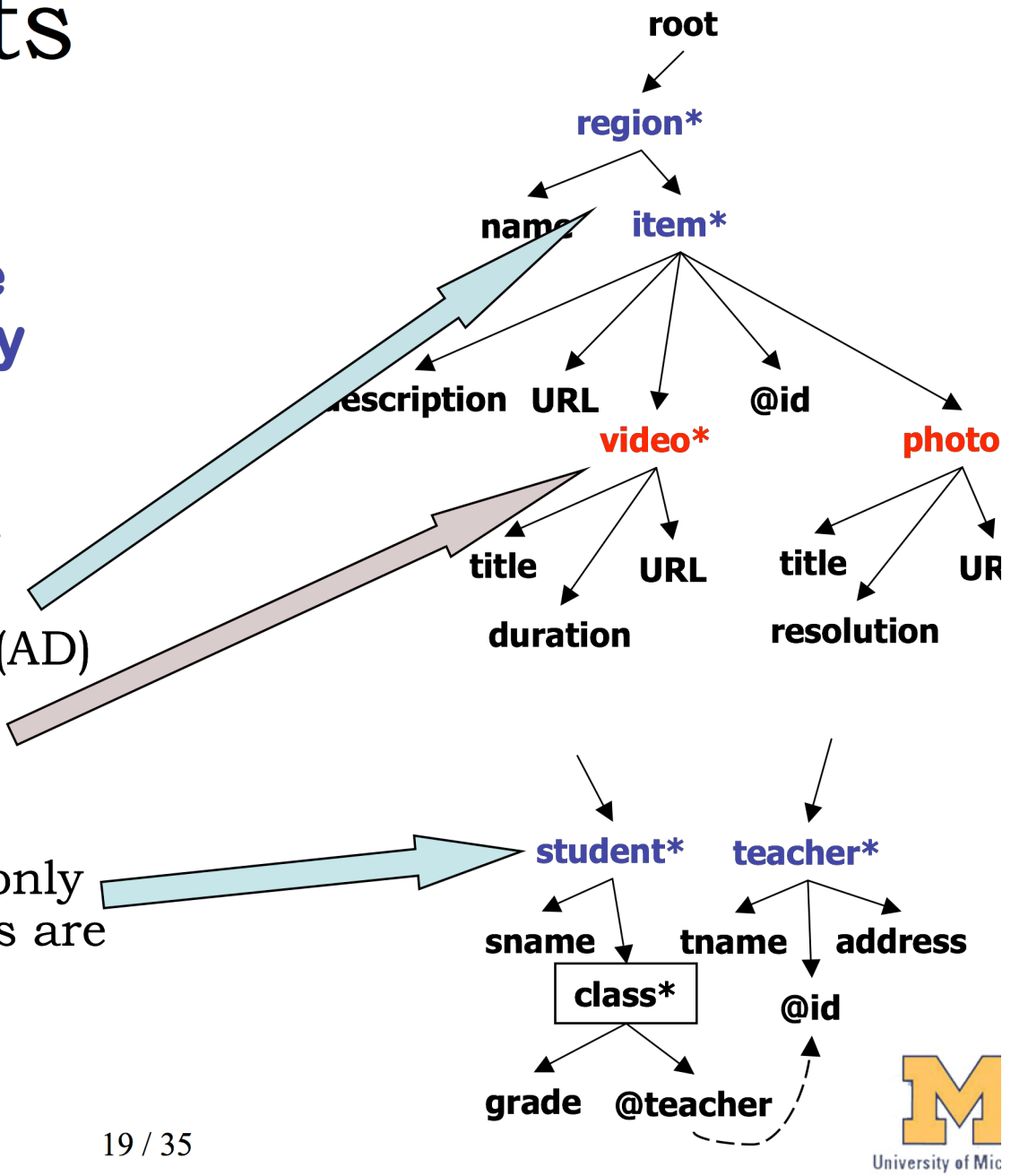
- Relationship between entity elements
  - Ancestor-Descendant (AD)
  - Sibling with common ancestor (SIB-A)



Schema Hints

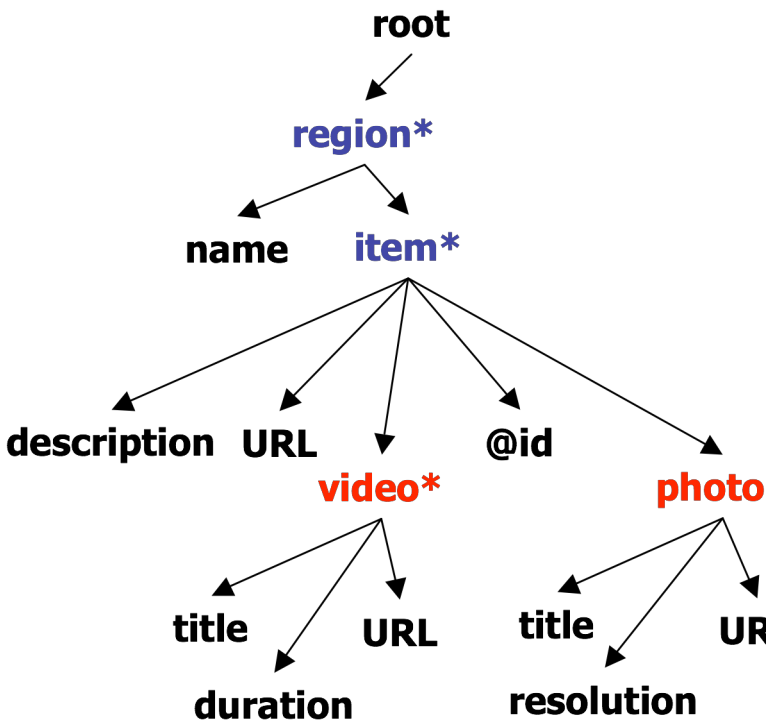
Attributes of the same entity are meaningfully related

- Relationship between entity elements
  - Ancestor-Descendant (AD)
  - Sibling with common ancestor (SIB-A)
  - Sibling with common descendant (SIB-D) – only occur when value links are present

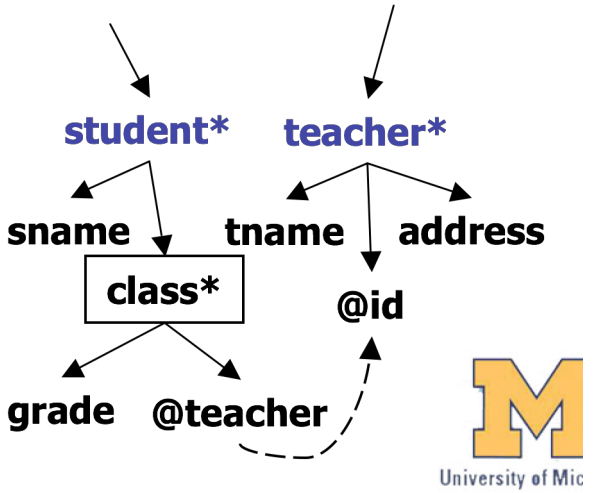


Schema Hints

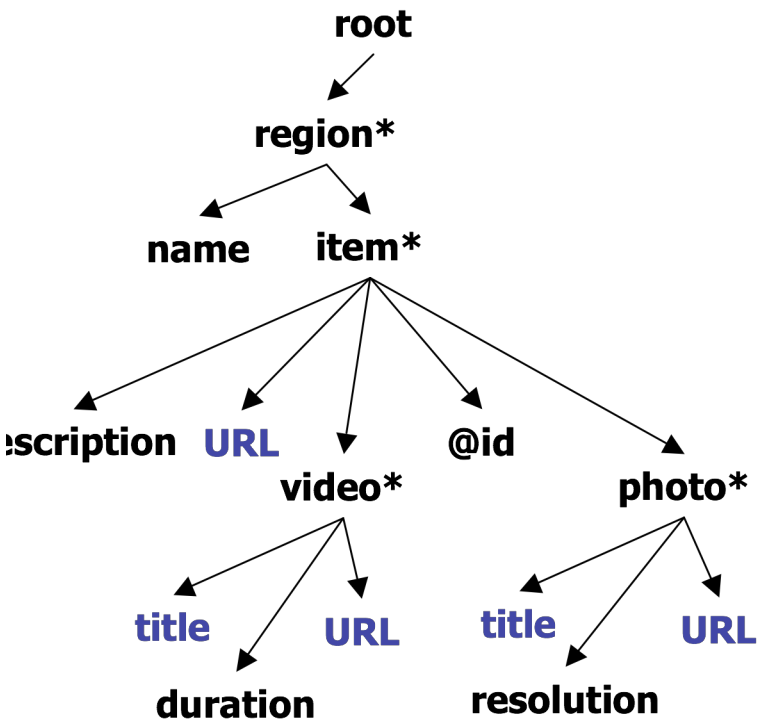
Attributes of the same entity are meaningfully related



Entities with AD and SIB-D relationships are meaningfully related



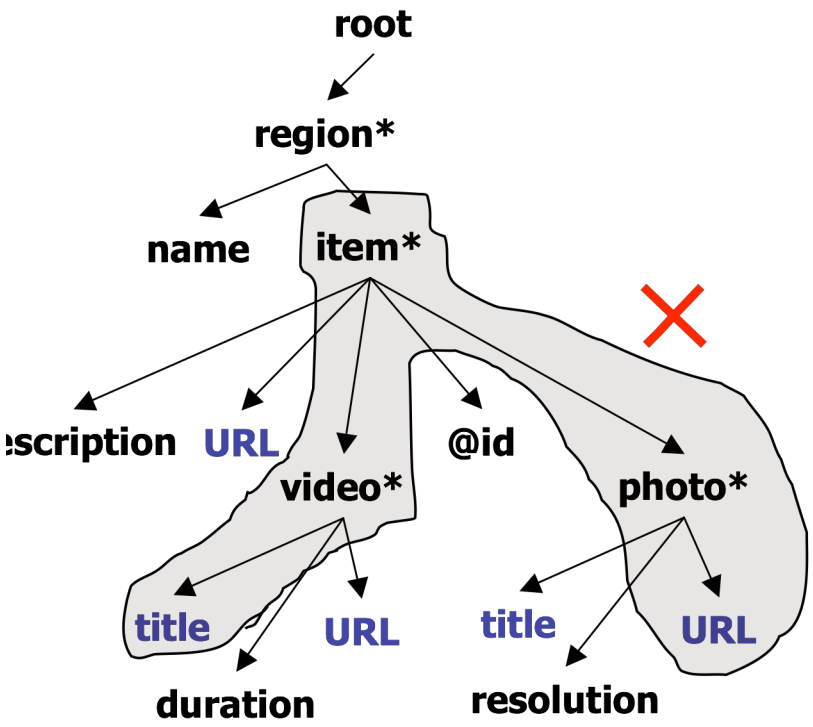
Advanced Matching Semantics



Query
[title:ABC; URL:~]

- Related-Entity (RE):
  - Satisfy Basic Semantics
  - Every attribute element belong to some entity element
  - If there is more than one entities, any two entities are either AD related or SIB-D related

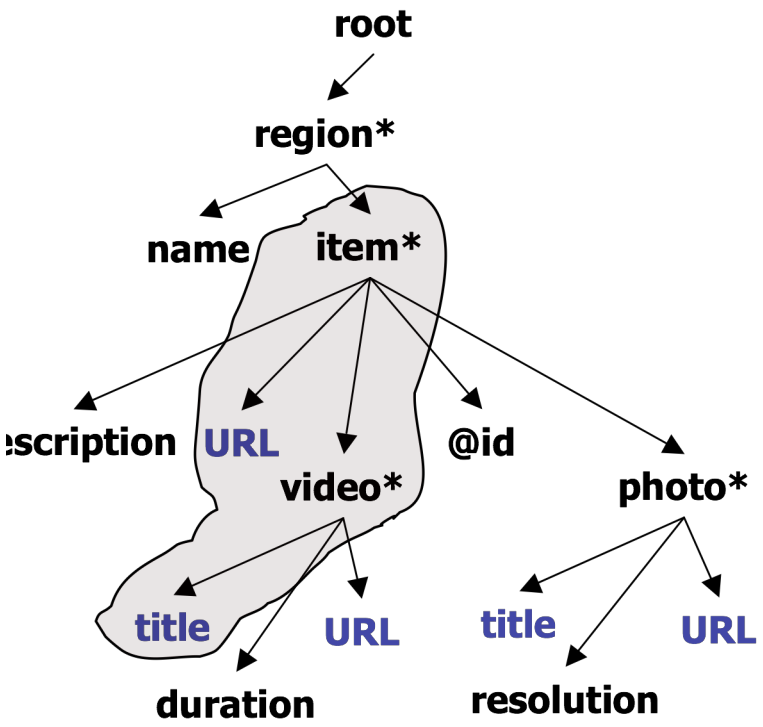
Advanced Matching Semantics



Query
[title:ABC; URL:~]

- Related-Entity (RE):
  - Satisfy Basic Semantics
  - Every attribute element belong to some entity element
  - If there is more than one entities, any two entities are either AD related or SIB-D related

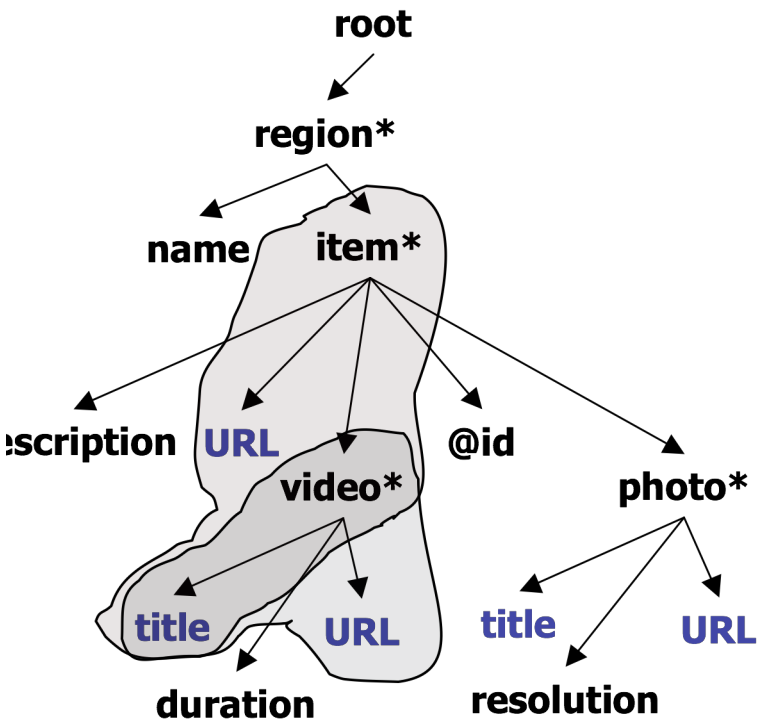
Advanced Matching Semantics



Query
[title:ABC; URL:~]

- Related-Entity (RE):
  - Satisfy Basic Semantics
  - Every attribute element belong to some entity element
  - If there is more than one entities, any two entities are either AD related or SIB-D related

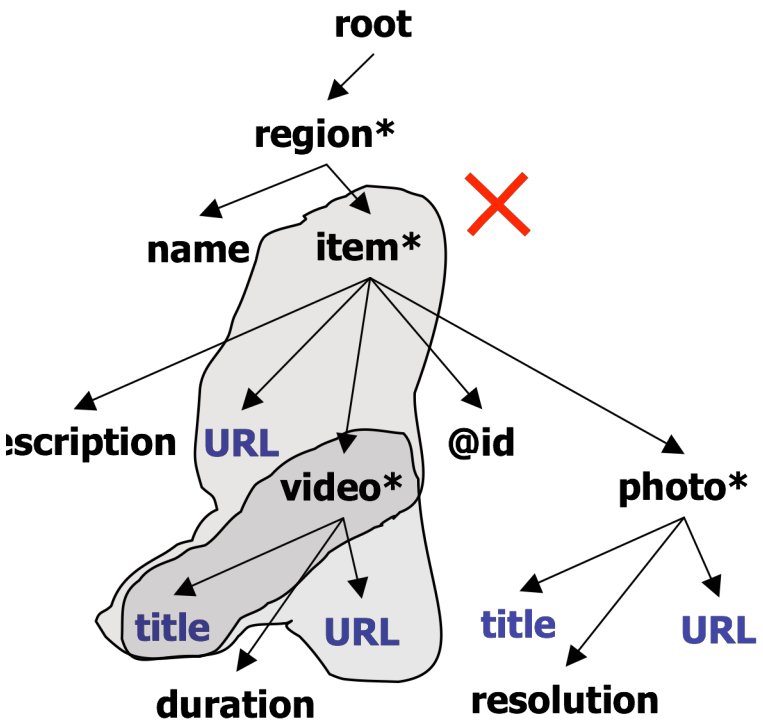
Advanced Matching Semantics



Query
[title:ABC; URL:~]

- Related-Entity (RE):
  - Satisfy Basic Semantics
  - Every attribute element belong to some entity element
  - If there is more than one entities, any two entities are either AD related or SIB-D related

Advanced Matching Semantics



Query
[title:ABC; URL:~]

- Related-Entity (RE):
  - Satisfy Basic Semantics
  - Every attribute element belong to some entity element
  - If there is more than one entities, any two entities are either AD related or SIB-D related
- Non-Redundant (NR):
  - Satisfy RE semantics
  - No other schema pattern satisfies RE Semantics, and yet contains a strict subset of entity elements

Summary of Schema-Based Semantics

Summary of Schema-Based Semantics

- Benefits
  - Avoid data specific erroneous matches
  - Significantly improve query performance

Summary of Schema-Based Semantics

- Benefits
  - Avoid data specific erroneous matches
  - Significantly improve query performance
- Limitations
  - Not enough flexibility in query semantics

Summary of Schema-Based Semantics

- Benefits
  - Avoid data specific erroneous matches
  - Significantly improve query performance
- Limitations
  - Not enough flexibility in query semantics
- Need to incorporate “some” schema information into the query!

Outline

- Background and Motivation
- Issues with Current Approaches
- Schema-Based Matching Semantics
- Meaningful Summary Query Model
- Conclusion

MSQ Model Overview

- Syntax extended from XQuery

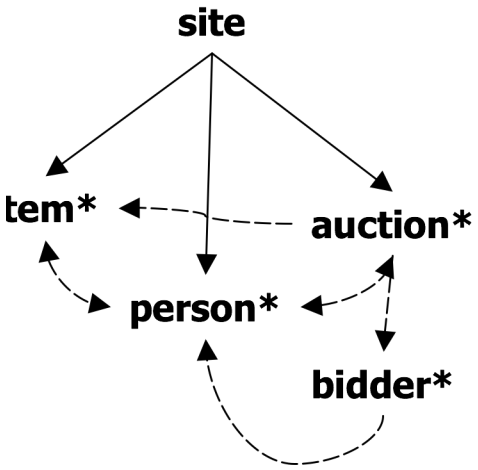
MSQ Model Overview

- Syntax extended from XQuery
- Leverages *schema summary* (as partial schema information) for the basic structure of the query

MSQ Model Overview

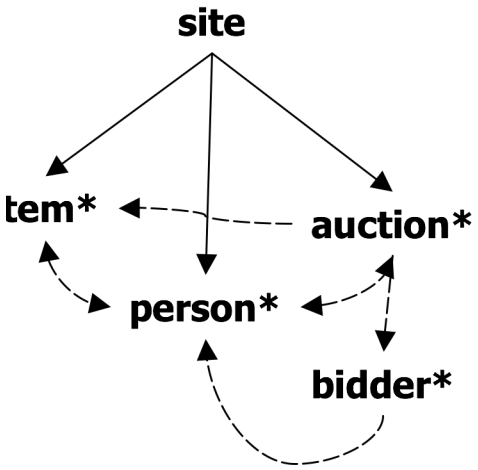
- Syntax extended from XQuery
- Leverages *schema summary* (as partial schema information) for the basic structure of the query
- Leverages *schema-based matching semantics* for fetching information from hidden schemas

Example MSQ Query



- Query: retrieve auction that are *sold* by the person named “peter” in “chicago” and *contain* items that are “antiques” in region “asia”

Example MSQ Query

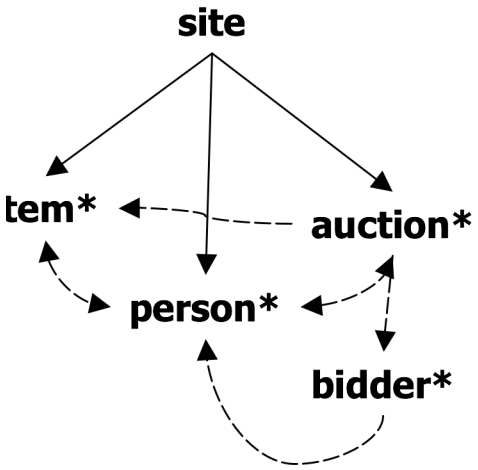



```
for     $a  in /site//auction
        $i  in /site//item
        $p  in /site//person
where       ($a, $p,        ) and
             ($a, $i,           )
return  $a
```

Basic Query Structure

- Query: retrieve auction that are *sold* by the person named “peter” in “chicago” and *contain* items that are “antiques” in region “asia”

Example MSQ Query



Structure-Free Conditions

```
set   $c1 = item[region:asia; item:antiques],
      $c2 = person[name:peter; address:chicago]
for   $a  in /site//auction.MF(),
      $i  in /site//item.MF($c1),
      $p  in /site//person.MF($c2)
where MR($a, $p,"sold" ) and
      MR($a, $i, "contains")
return $a
```

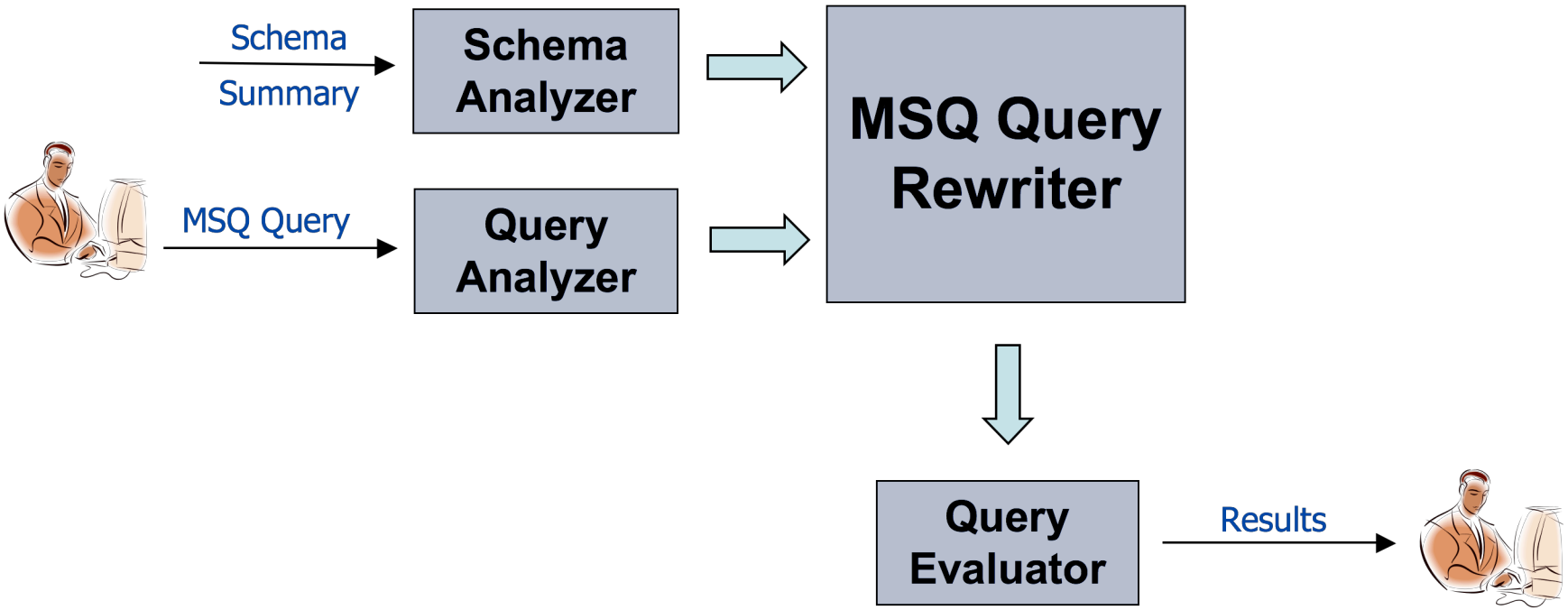
Basic Query Structure

- Query: retrieve auction that are *sold* by the person named "peter" in "chicago" and *contain* items that are "antiques" in region "asia"
- MF: Meaningful Fragment
- MR: Meaningful Relationship

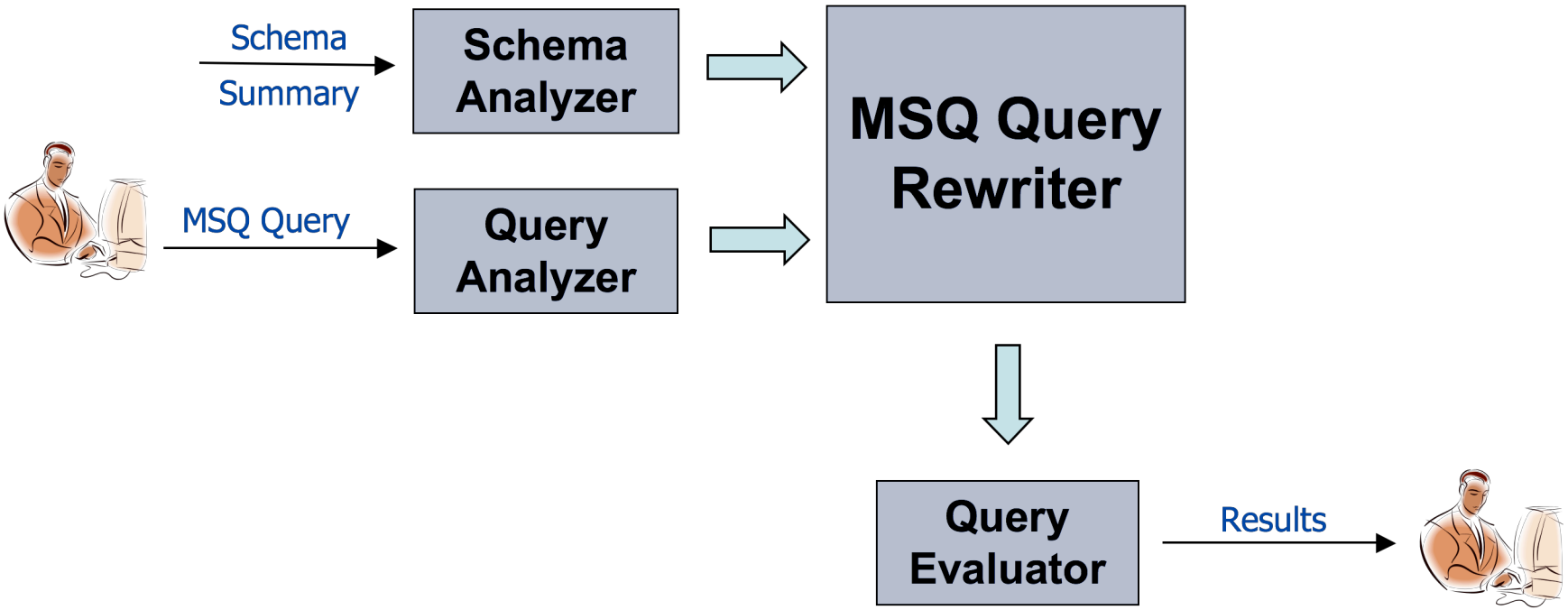
Benefits of MSQ Query Model

- The user only establishes the overall structure of the query (conceptual simplicity)
  - Express the details as labeled keyword conditions
  - Let the system automatically figure out the exact structured query
- The user can still express complex queries (e.g., with aggregate conditions)
- Schema knowledge can be easily injected into the query if available

Evaluating MSQ Queries



Evaluating MSQ Queries



Analyzing MSQ Query

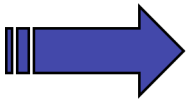
MSQ Query [User Specify]

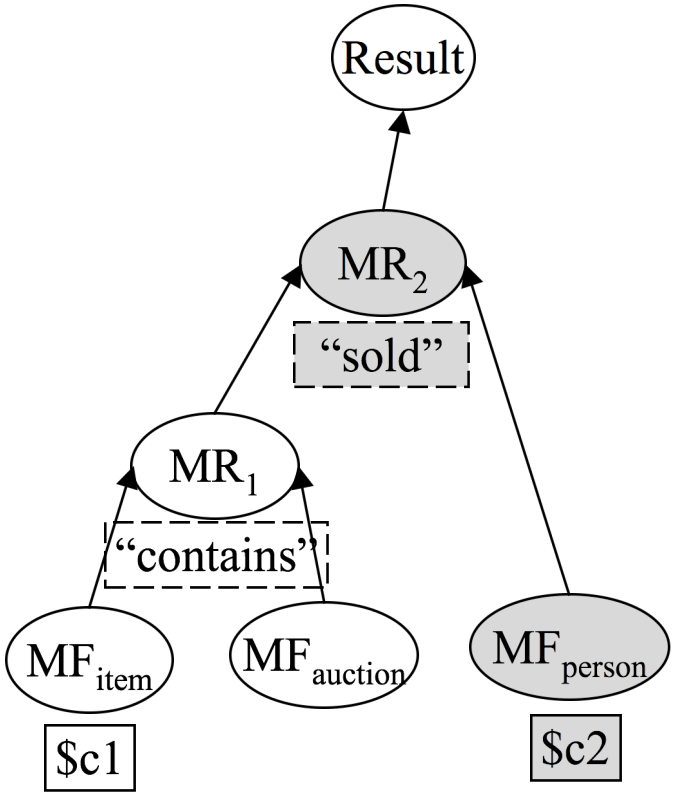
```
set $c1 = item[region:asia;
                item:antiques],
    $c2 = person[name:peter;
                  address:chicago]
for  $a  in /site//auction.MF(),
     $i  in /site//item.MF($c1),
     $p  in /site//person.MF($c2)
where  MR($a, $p, "sold") and
       MR($a, $i, "contains")
return $a
```

Analyzing MSQ Query

MSQ Query [User Specify]

```
set $c1 = item[region:asia;
                item:antiques],
    $c2 = person[name:peter;
                 address:chicago]
for  $a  in /site//auction.MF(),
     $i  in /site//item.MF($c1),
     $p  in /site//person.MF($c2)
where  MR($a, $p, "sold") and
       MR($a, $i, "contains")
return $a
```

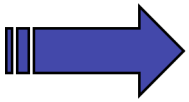


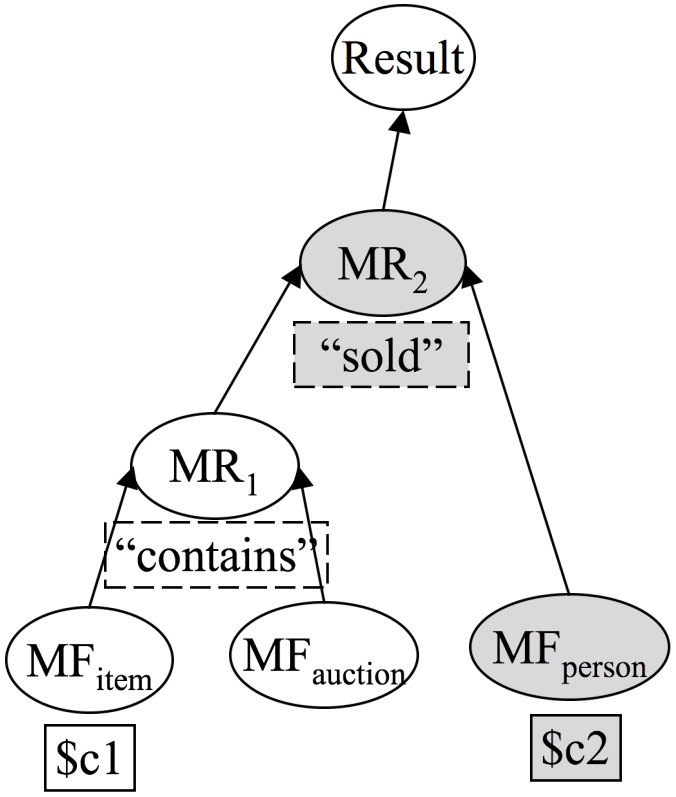


Analyzing MSQ Query

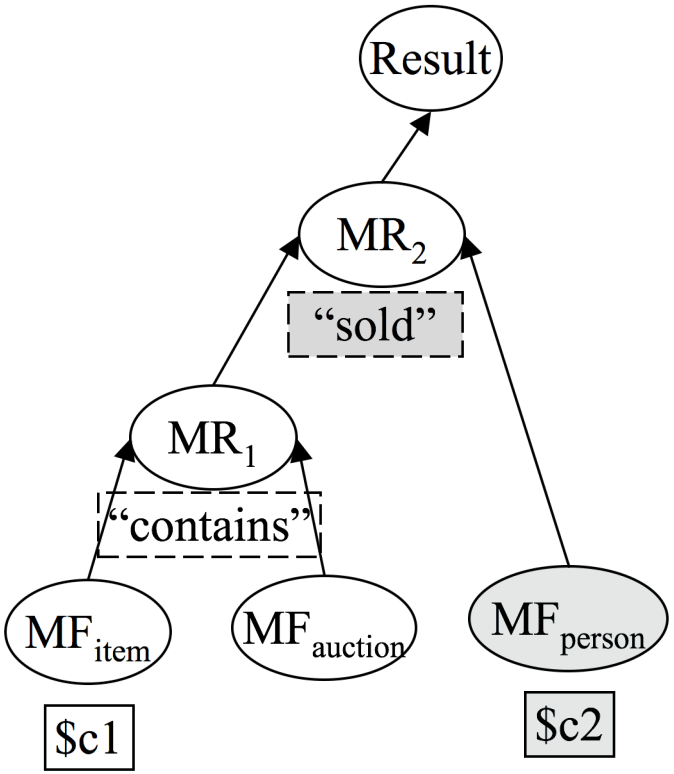
MSQ Query [User Specify]

```
set $c1 = item[region:asia;
                item:antiques],
    $c2 = person[name:peter;
                 address:chicago]
for  $a  in /site//auction.MF(),
     $i  in /site//item.MF($c1),
     $p  in /site//person.MF($c2)
where MR($a, $p, "sold") and
      MR($a, $i, "contains")
return $a
```



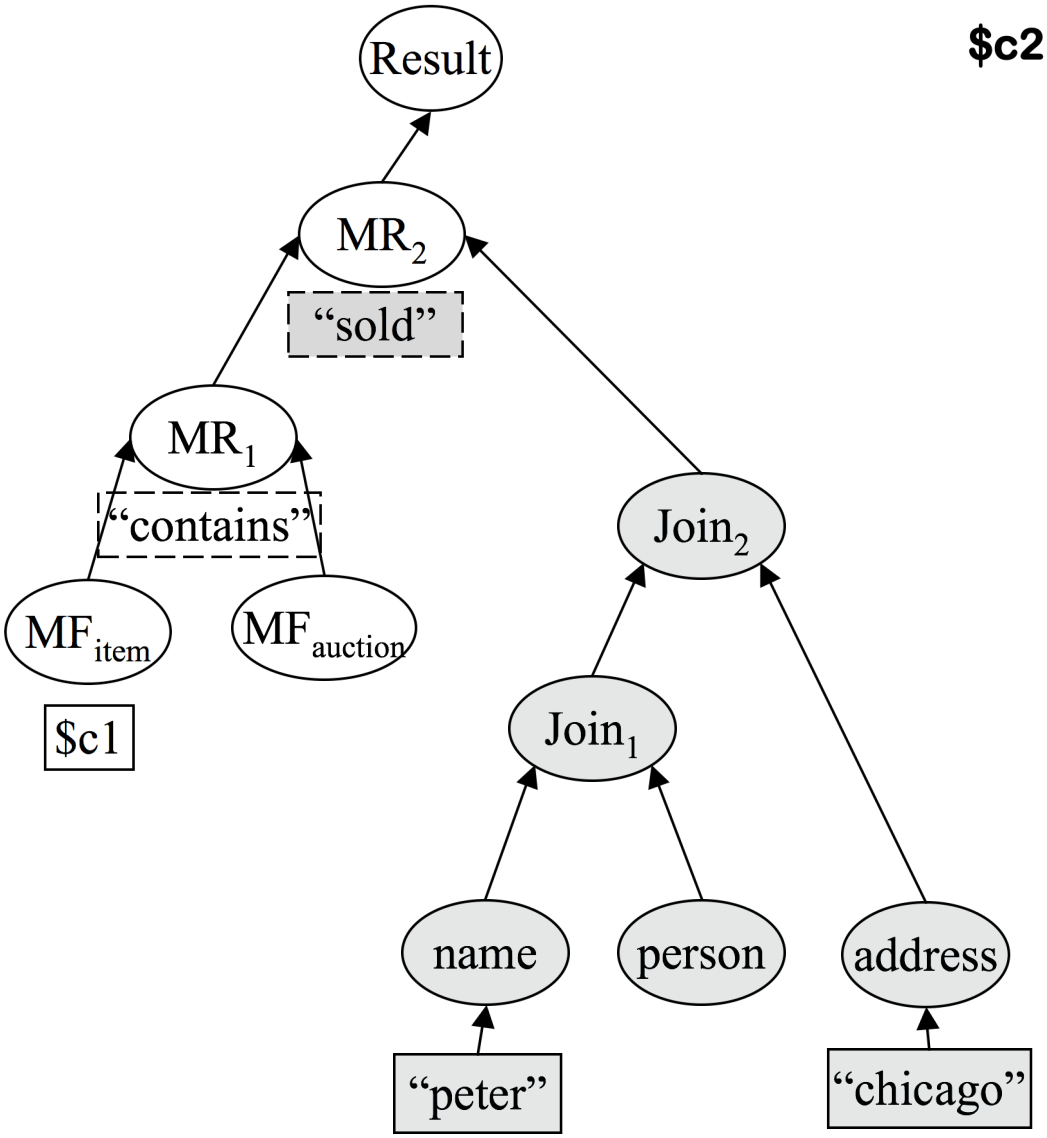


MSQ Query Rewriting Algorithm



$c2 = person[name:peter;
address:chicago]

MSQ Query Rewriting Algorithm



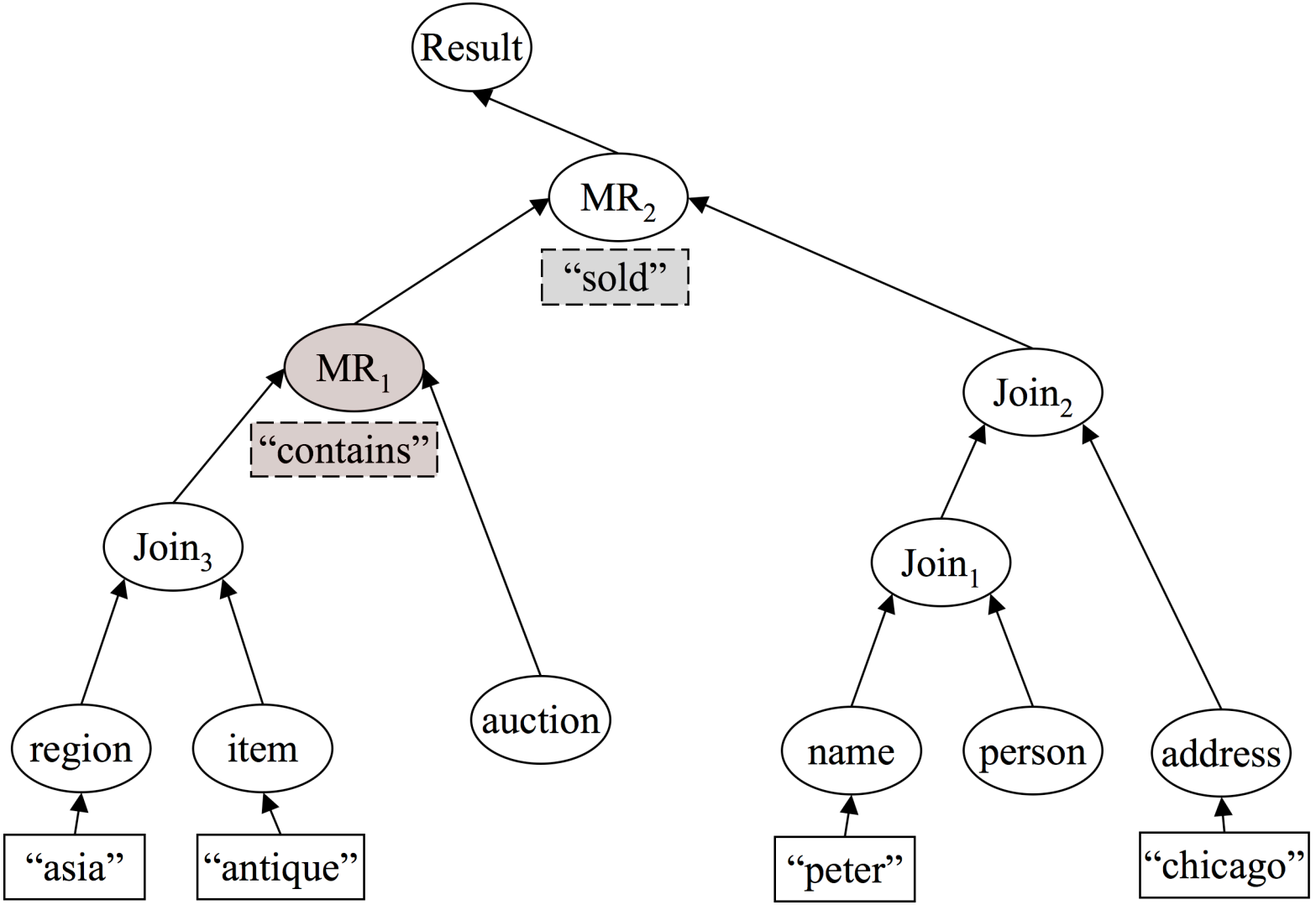
$c2 = person[name:peter; address:chicago]

Identify MSP

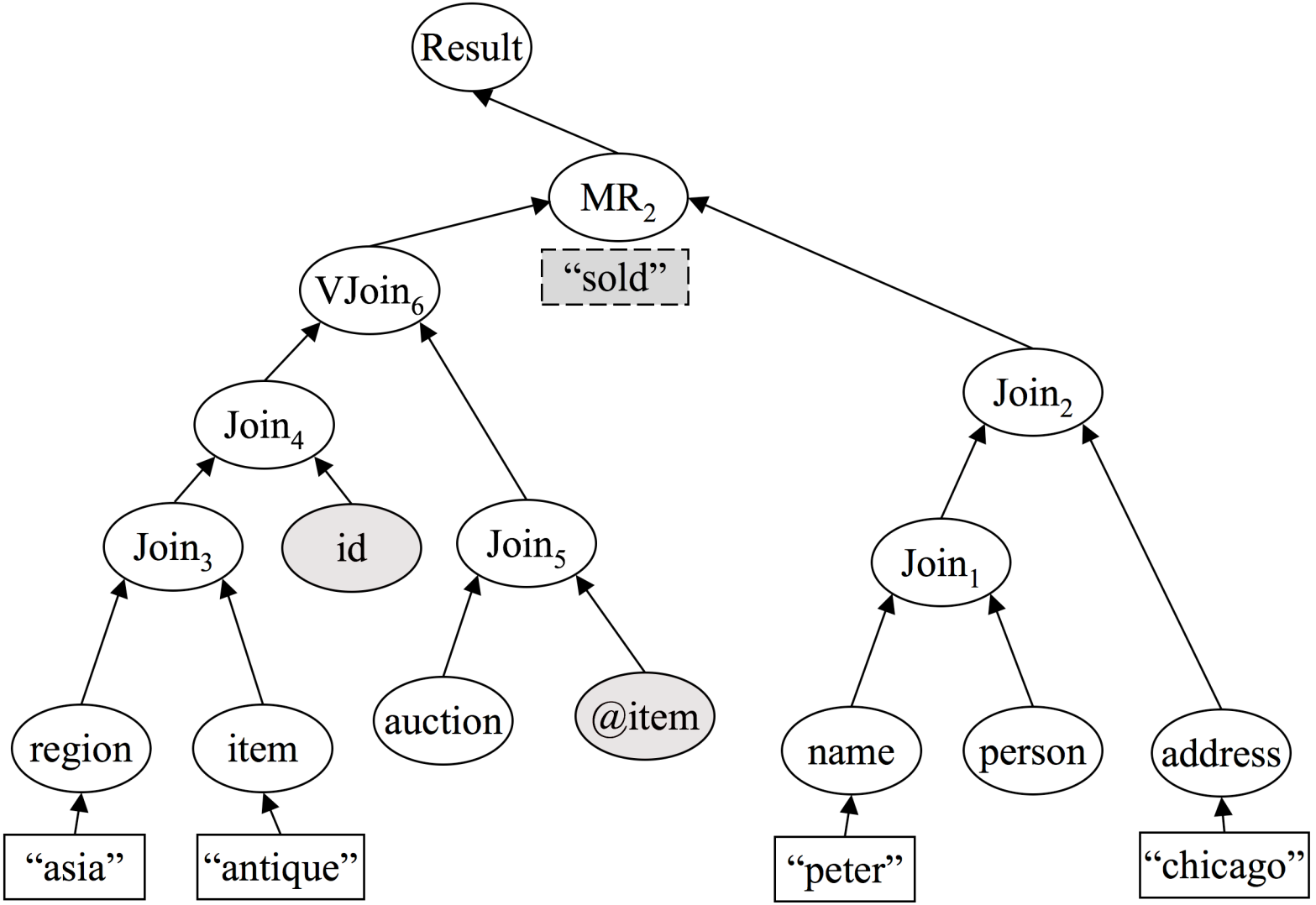
person*

name **address**

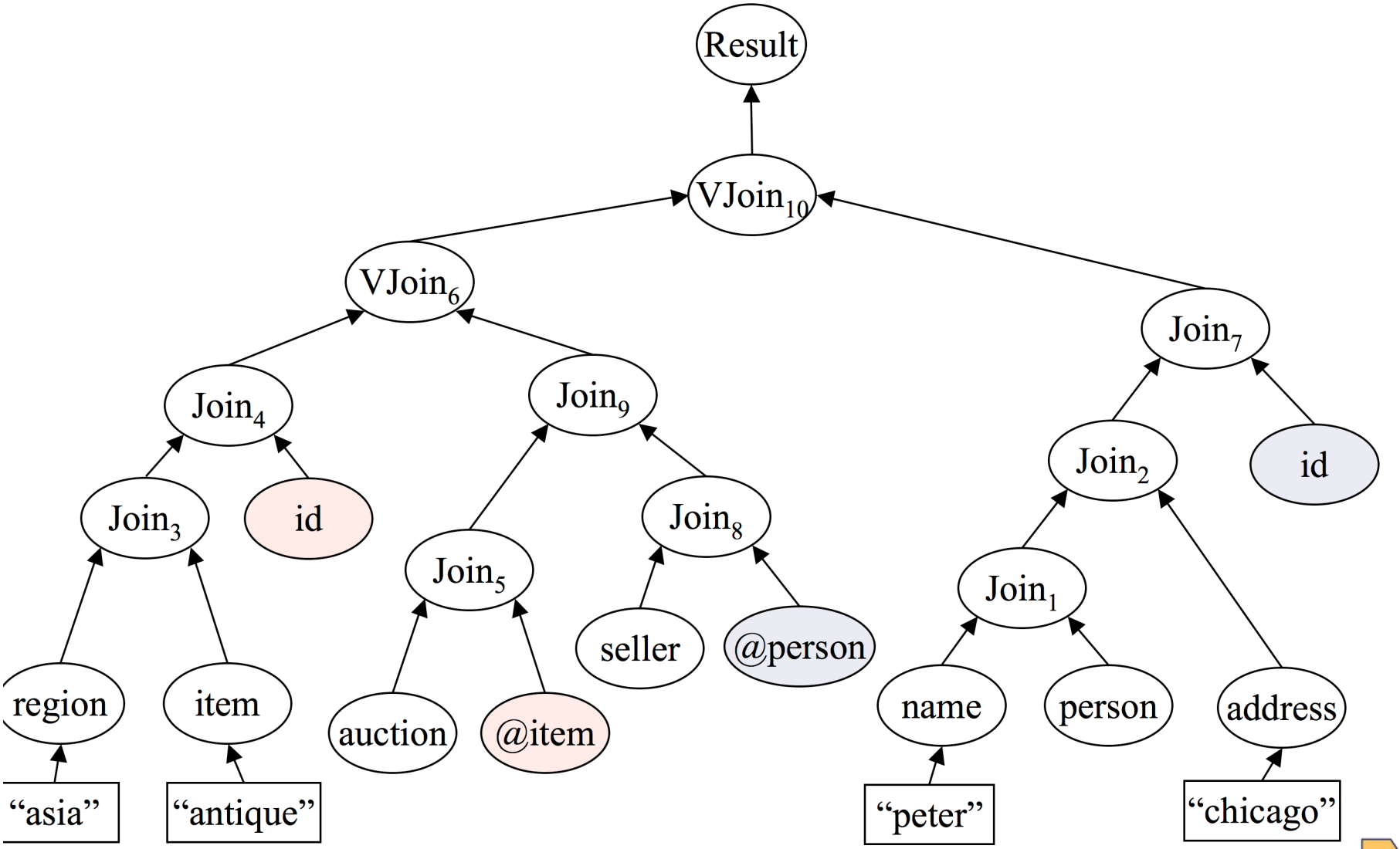
MSQ Query Rewriting Algorithm



MSQ Query Rewriting Algorithm



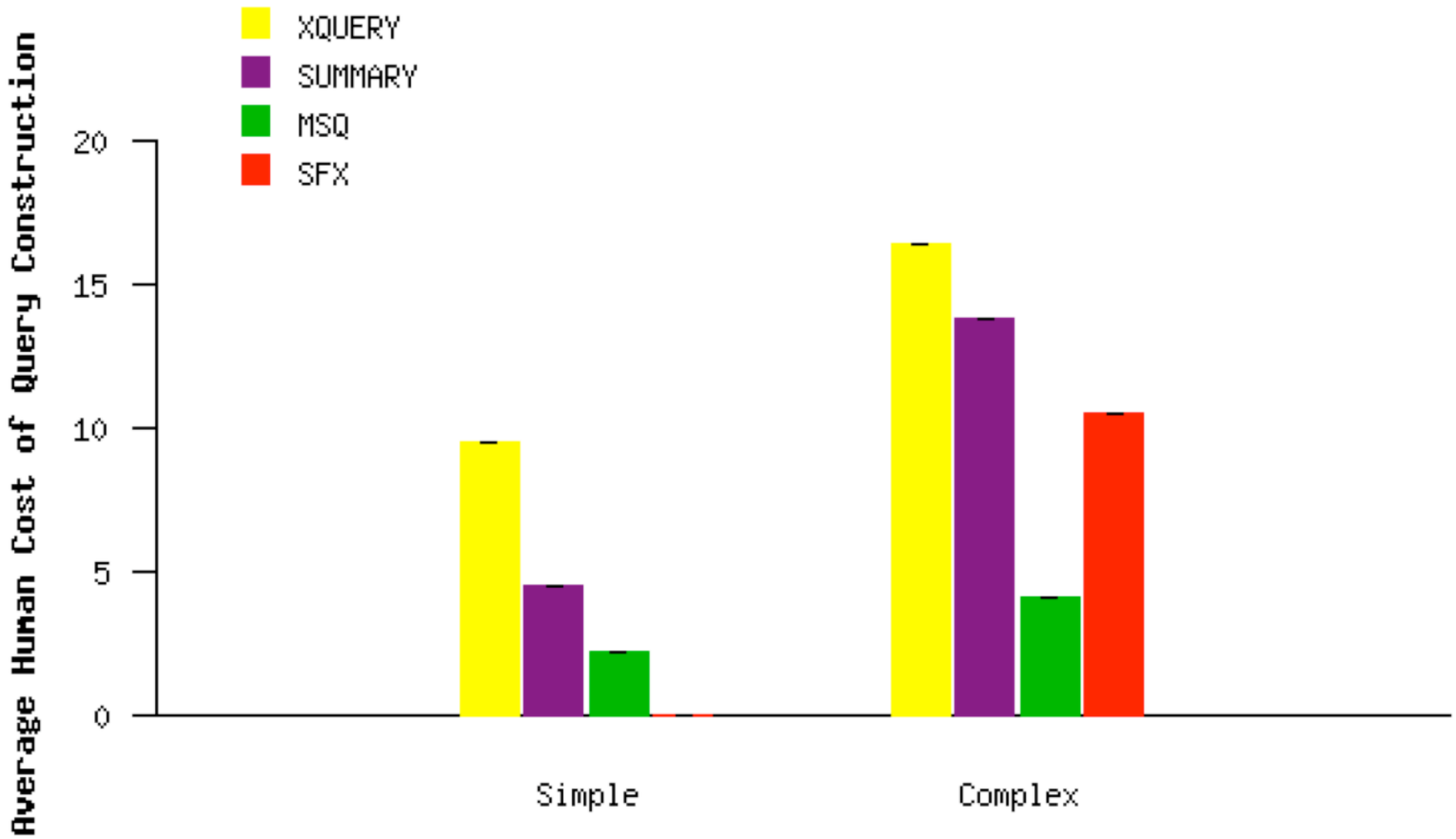
Final Evaluation Tree



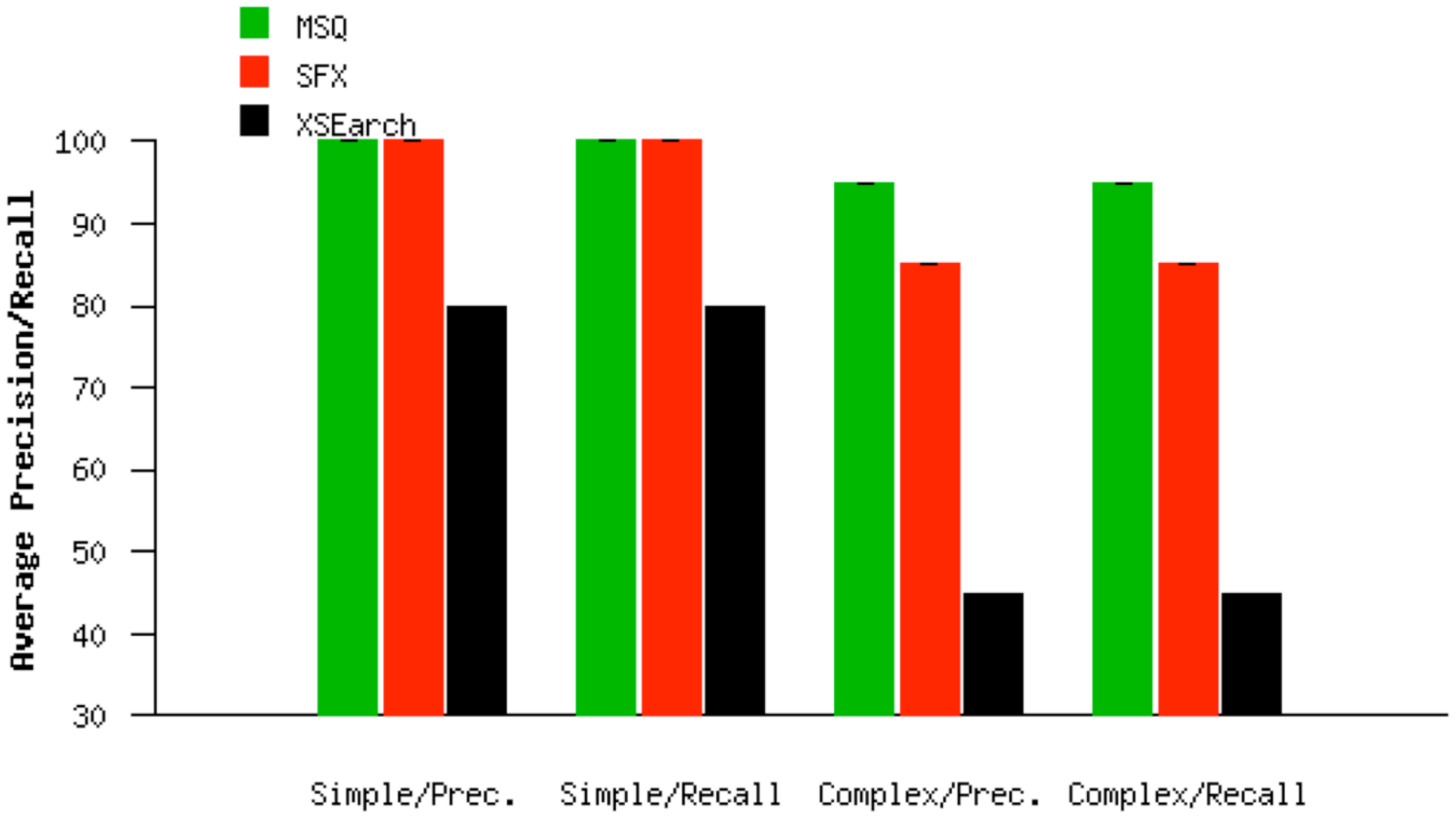
Experimental Evaluation

- XMark and MiMI datasets
  - 20 queries for XMark, 52 queries for MiMI
- Compare with four alternative strategies
  - XQuery (XQUERY)
  - Summary-Based Exploration (SUMMARY)
  - Labeled Keyword (XSEarch)
  - Schema-Free XQuery (SFX)
- Divide queries into *simple* and *complex*

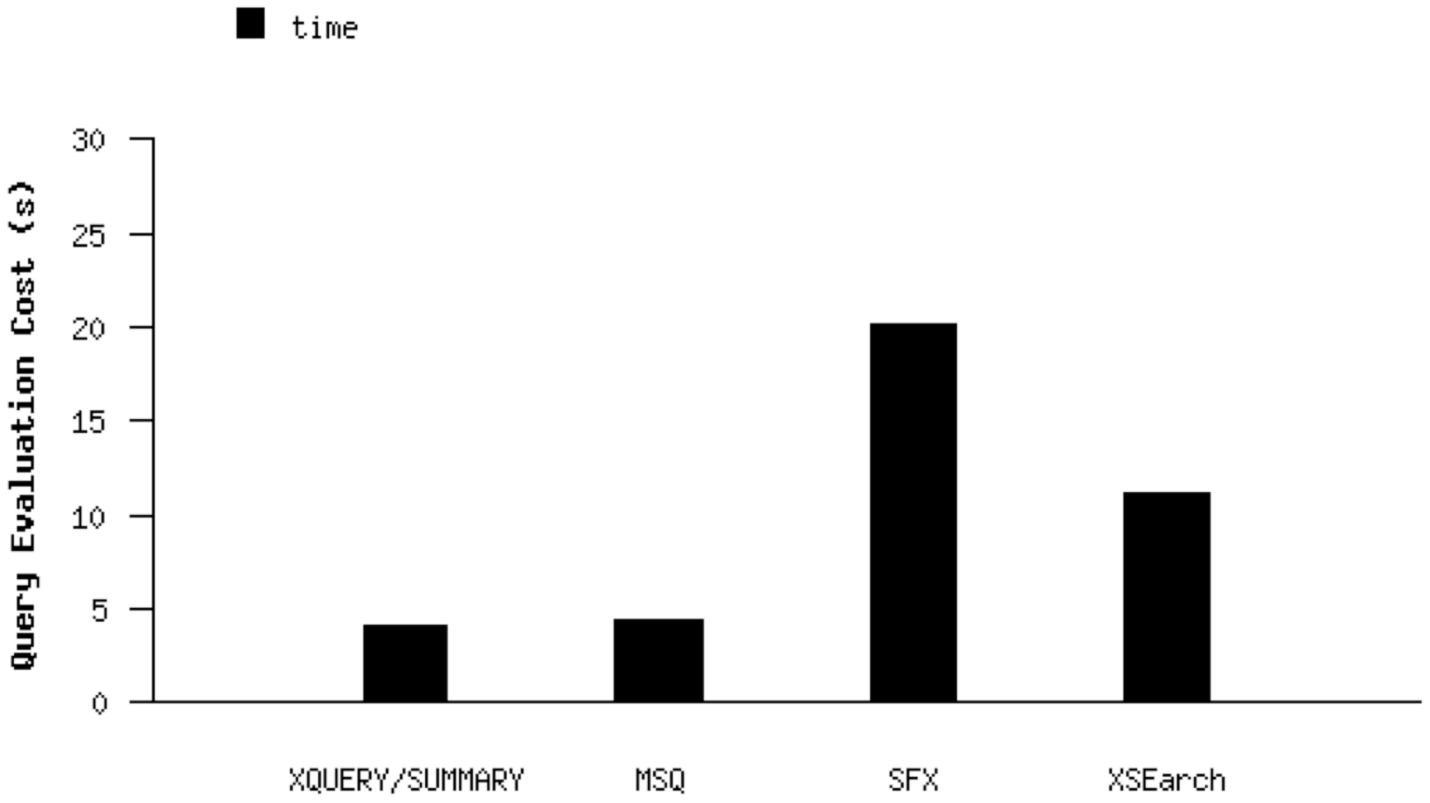
Human Cost of Querying



Result Quality (Precision & Recall)



Query Performance



Conclusion

- We proposed a novel query model called Meaningful Summary Query (MSQ)
  - Leverages schema-based semantics to improve query performance while maintaining result quality
  - Enables ordinary users to query on the schema summary directly

Questions ?

http://www.eecs.umich.edu/db/usable