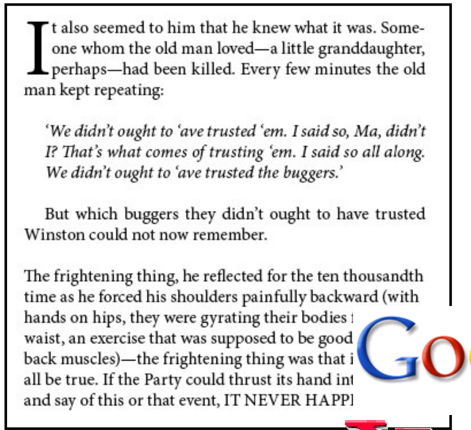
Keyword Search on Structured and Semi-Structured Data

Yi Chen
Wei Wang
Ziyang Liu
Xuemin Lin

Arizona State University, USA
University of New South Wales & NICTA, Australia

Traditional Data Access Methods

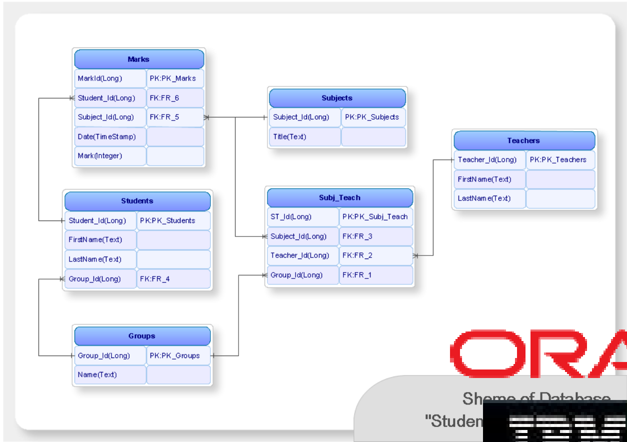


It also seemed to him that he knew what it was. Someone whom the old man loved—a little granddaughter, perhaps—had been killed. Every few minutes the old man kept repeating:

'We didn't ought to 'ave trusted 'em. I said so, Ma, didn't I? That's what comes of trusting 'em. I said so all along. We didn't ought to 'ave trusted the buggers.'

But which buggers they didn't ought to have trusted Winston could not now remember.

The frightening thing, he reflected for the ten thousandth time as he forced his shoulders painfully backward (with hands on hips, they were gyrating their bodies f waist, an exercise that was supposed to be good back muscles)—the frightening thing was that i all be true. If the Party could thrust its hand int and say of this or that event, IT NEVER HAPPI

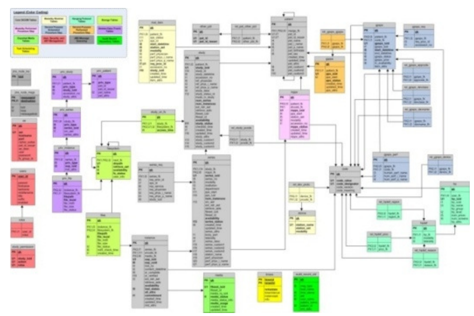


- Text documents:
  - Unstructured
  - Accessed by keywords
  - Limited search quality
  - Large user population
- Databases / XML data
  - Structured, with rich meta-data
  - Accessed by query languages
  - High search quality
  - Small user population that masters DB

The Challenges of Accessing Structured Data

- Query languages: long learning curves
- Schemas: Complex, evolving, or even unavailable.
- What about filling in query forms?
  - Limited access pattern.
  - Hard to design and maintain forms on dynamic and heterogeneous data!

select paper.title from conference c, paper p, author a1, author a2, write w1, write w2
where c.cid = p.cid AND p.pid = w1.pid AND p.pid = w2.pid AND w1.aid = a1.aid AND w2.aid = a2.aid AND a1.name = “John” AND a2.name = “Mary” AND c.name = SIGMOD



The usability of DB is severely limited unless easier ways to access databases are developed [Jagadish, SIGMOD 07].

Supporting Keyword Search on DB – Advantages /1

- Easy to use
  - The most important factor for the majority of users.
  - The same advantage of keyword search on text documents

Supporting Keyword Search on DB – Advantages /2

- Enabling interesting or unexpected discoveries
  - Relevant data pieces that are scattered but are collectively relevant to the query should be automatically assembled in the results
  - Larger scope for data inter-connection

Supporting Keyword Search on DB – Advantages /3

- Returning meaningful results by exploiting structural information.
- An unique opportunity in structured data

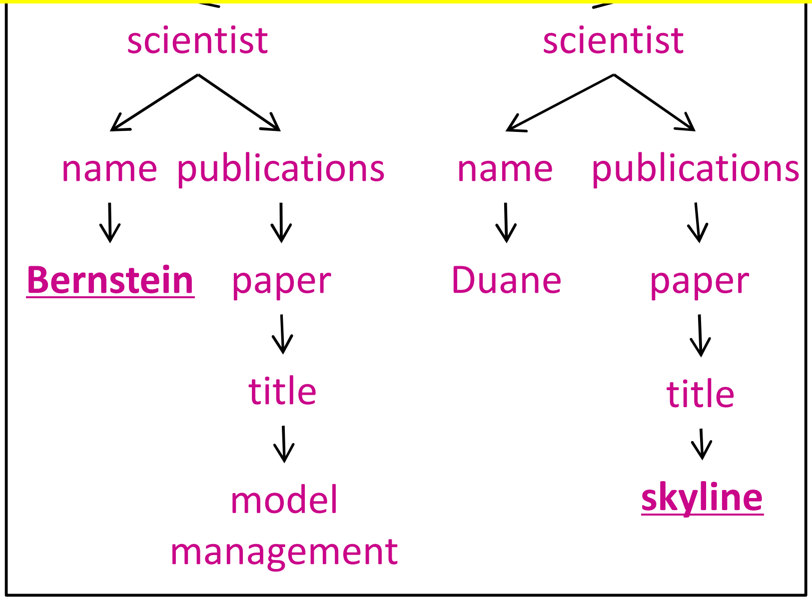
Query: “Bernstein, skyline”

Text Document

“Bernstein is a computer scientist.......... One of **Bernstein**’s colleagues, Duane, recently published a paper about **skyline** query processing.”

Structured Document

Such a result will have a low rank.



Supporting Keyword Search on DB – Summary of Advantages

- Increasing the DB usability
- Increasing the coverage and quality of keyword search



Supporting Keyword Search on DB – Challenges /1

Semantics: keyword queries are ambiguous

- How to infer the query semantics and find relevant answers?
- How to effectively rank the results in the order of their relevance?
- How to help users analyze results?
- How to evaluate the quality of search results?

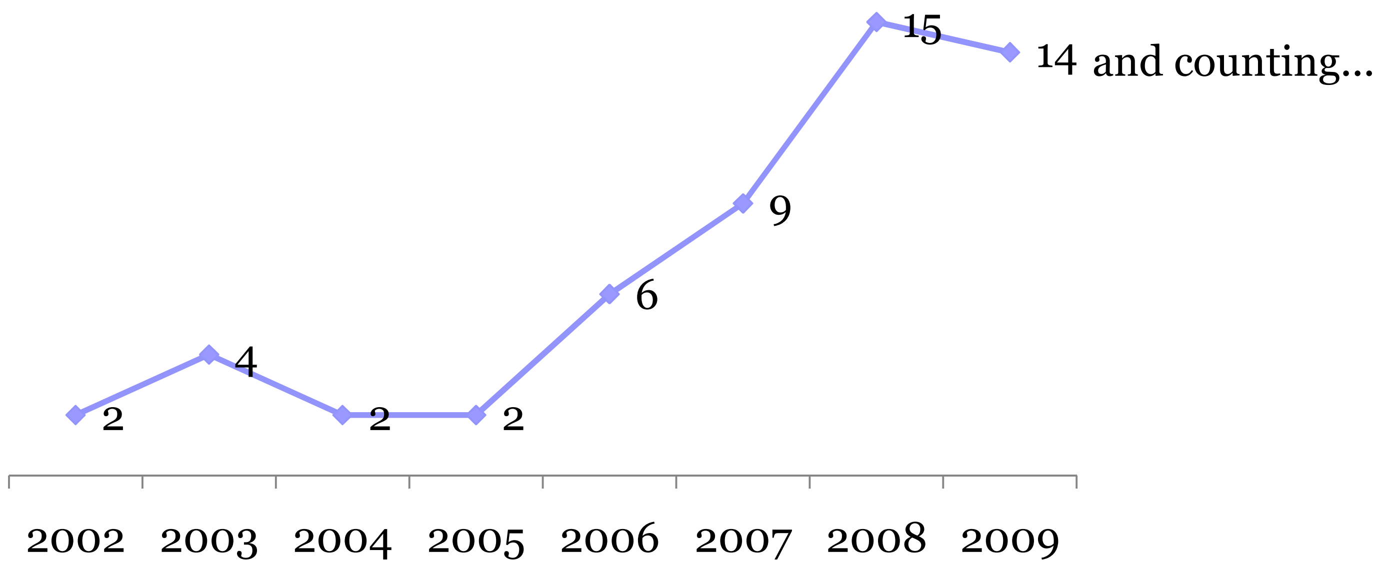
Supporting Keyword Search on DB – Challenges /2

Efficiency:

- Many problems in keyword search on DB are shown to be NP-hard.
  - Generating results, query segmentation, snippet generation, etc.,
- Large datasets
- How to generate (top-k) query results efficiently?

Keyword Search on DB: State-of-the Art

- Keyword search on DB has become a hot research direction, and attracted researchers in DB, IR, theory, etc
  - More than 50 research papers, from both research labs and universities in ***major database*** conferences/journals
  - Workshop about keyword search on DB (KEYS, June 28, 09)



Timeline /1

	2003	2004	2005	2006	2007	2008	2009
XML	XKeyword	MLCA	SLCA	Tree proximity	XSeek	MaxMatch	XReal
	XSEarch				SLCA 2	eXtract	RTF
	XRank				CVLCA	ELCA	
Nested Graphs /Workflows							WISE

Timeline /2

	Before 2002	2002	2003-2005	2006	2007	2008	2009
RDBMS/ Graph Result Generation	Proximity Search	BANKS 1	Discover 2	Preis	SPARK	BANKS 3	Community
	Information Unit	DBXplorer	BANKS 2	D&C	BLINKS	SUITS	RDMBS
		Discover		DP	Hetero-geneous	EASE	
RDBMS/ Graph Other Applications				IR Ranking	KDAP	QUnit	Form Search
					DB selection 1	SQAK	Frequent terms
						DB Selection 2	Data Clouds
						Query Cleaning	Minimal Group-by

XSeek Demo

xseek

texas men clothes eXtract Search

Data Sets Retailers View Data Snippet Size 11

Results 1-3 of 3 for keywords "**texas men clothes**" on data set "**Retailers**" with snippet size **11**. (**0.532** seconds)

Store : Brooks Brothers Clothing See Snippet of Google Desktop

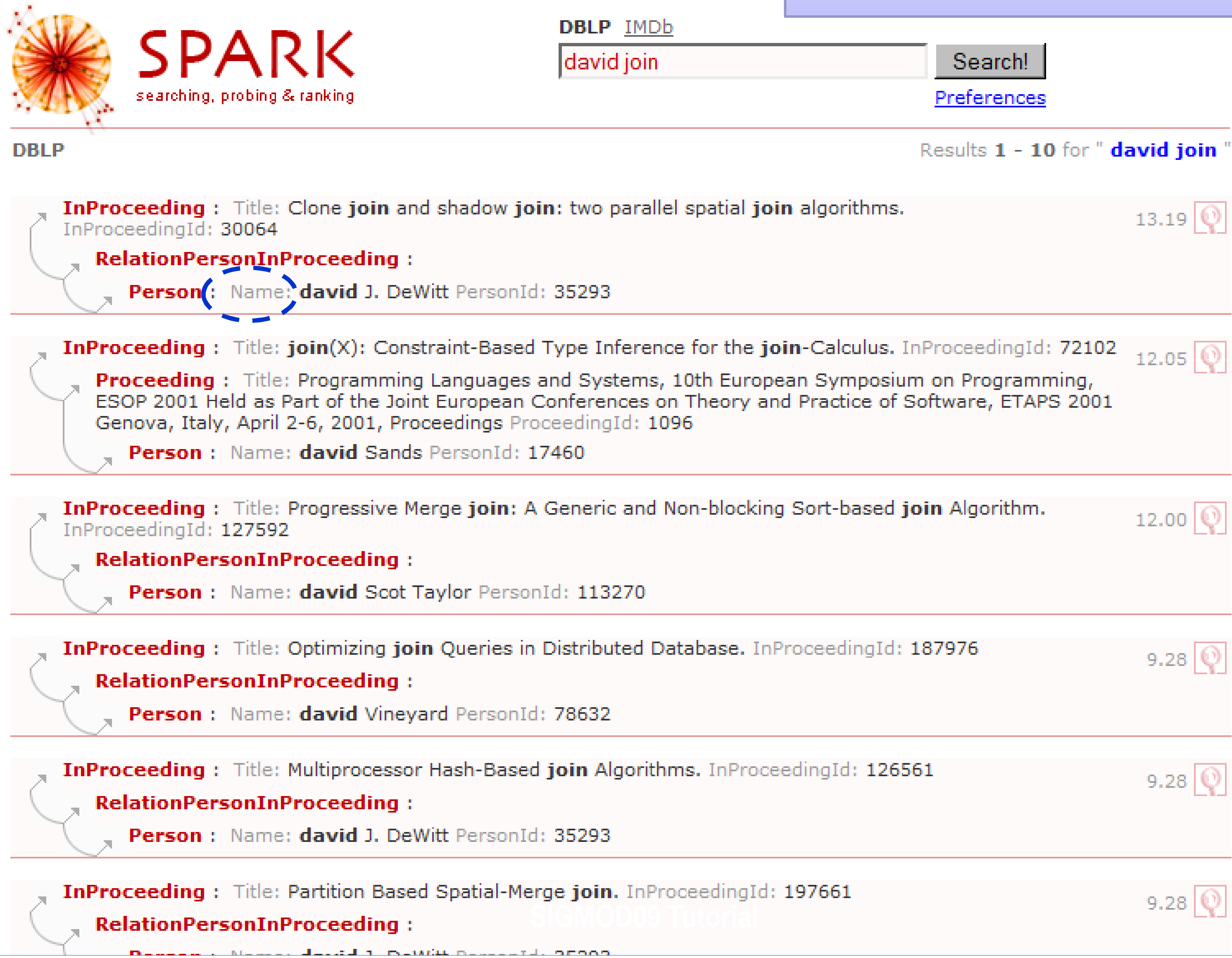
```
<Store>
  <name>Brooks Brothers Clothing</name>
  <state>Texas</state>
  <merchandise>
    <clothes>
      <category>pants</category>
    <clothes>
      <fitting>men</fitting>
      <situation>business</situation>
      <category>sweater</category>
    <clothes>
      <category>shirts</category>
```

Store : L.L. Bean See Snippet of Google Desktop

```
<Store>
  <name>L.L. Bean</name>
  <city>Houston</city>
  <state>Texas</state>
  <merchandise>
    <clothes>
      <fitting>men</fitting>
      <category>footwear</category>
    <clothes>
      <category>outwear</category>
    <clothes>
      <category>pants</category>
```

SPARK Demo /1

After seeing the query results, the user identifies that 'david' should be 'david J. Dewitt'.



SPARK Demo /2

The user is only interested in finding all join papers written by David J. Dewitt (i.e., not the 4th result)

SPARK
searching, probing & ranking

DBLP IMDb

Person:'David J. DeWitt' join | Search!

Preferences

DBLP

Results **1** - **10** for " **Person:'David J. DeWitt' join** "

InProceeding : Title: Multiprocessor Hash-Based **join** Algorithms. InProceedingId: 126561 — 15.15
RelationPersonInProceeding :
Person : Name: **David J. DeWitt** PersonId: 35293

InProceeding : Title: Partition Based Spatial-Merge **join.** InProceedingId: 197661 — 15.15
RelationPersonInProceeding :
Person : Name: **David J. DeWitt** PersonId: 35293

InProceeding : Title: Pointer-Based **join** Techniques for Object-Oriented Databases. InProceedingId: 111229 — 15.09
RelationPersonInProceeding :
Person : Name: **David J. DeWitt** PersonId: 35293

InProceeding : Title: Hash-Partitioned **join** Method Using Dynamic Destaging Strategy. InProceedingId: 127266 — 15.09
Proceeding : Title: Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings. ProceedingId: 1704
Person : Name: **David J. DeWitt** PersonId: 35293

InProceeding : Title: Tradeoffs in Processing Complex **join** Queries via Hashing in Multiprocessor Database Machines. InProceedingId: 127858 — 15.04
RelationPersonInProceeding :
Person : Name: **David J. DeWitt** PersonId: 35293

InProceeding : Title: Clone **join** and shadow **join**: two parallel spatial **join** algorithms. InProceedingId: 30064 — 15.04
RelationPersonInProceeding :
Person : Name: **David J. DeWitt** PersonId: 35293

SPARK Demo /3

SPARK
searching, probing & ranking
DBLP IMDb
@10 Person:'David J. DeWitt' 'join'
Search!
Preferences
DBLP
Results 1 - 10 for " @10 Person:'David J. DeWitt' 'join' "
InProceeding : Title: Multiprocessor Hash-Based join Algorithms. InProceedingId: 126561
15.15
RelationPersonInProceeding :
Person : Name: David J. DeWitt PersonId: 35293
InProceeding : Title: Partition Based Spatial-Merge join. InProceedingId: 197661
15.15
RelationPersonInProceeding :
Person : Name: David J. DeWitt PersonId: 35293
InProceeding : Title: Pointer-Based join Techniques for Object-Oriented Databases. InProceedingId: 111229
15.09
RelationPersonInProceeding :
Person : Name: David J. DeWitt PersonId: 35293
InProceeding : Title: Tradeoffs in Processing Complex join Queries via Hashing in Multiprocessor Database Machines. InProceedingId: 127858
15.04
RelationPersonInProceeding :
Person : Name: David J. DeWitt PersonId: 35293
InProceeding : Title: Clone join and shadow join: two parallel spatial join algorithms. InProceedingId: 30064
15.04
RelationPersonInProceeding :
Person : Name: David J. DeWitt PersonId: 35293
InProceeding : Title: A Performance Evaluation of Four Parallel join Algorithms in a Shared-Nothing Multiprocessor Environment. InProceedingId: 199103
15.04
RelationPersonInProceeding :

Overview of This Tutorial

- Outline the problem space and review typical approaches
  - Data Models: Trees, Graphs, Nested Graphs, Distributed Data
  - Problem Space:

Pre-processing
Database Selection
Query Cleaning

→

Query Processing
Result Generation
Ranking

→

Post-processing
Result Snippets
Result Clustering
Result Analysis/Evaluation

- Discuss future directions

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
  - Trees
  - Nested Graphs
  - Graphs
  - RDBMS
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

Part 1

Part 2

Result Definitions

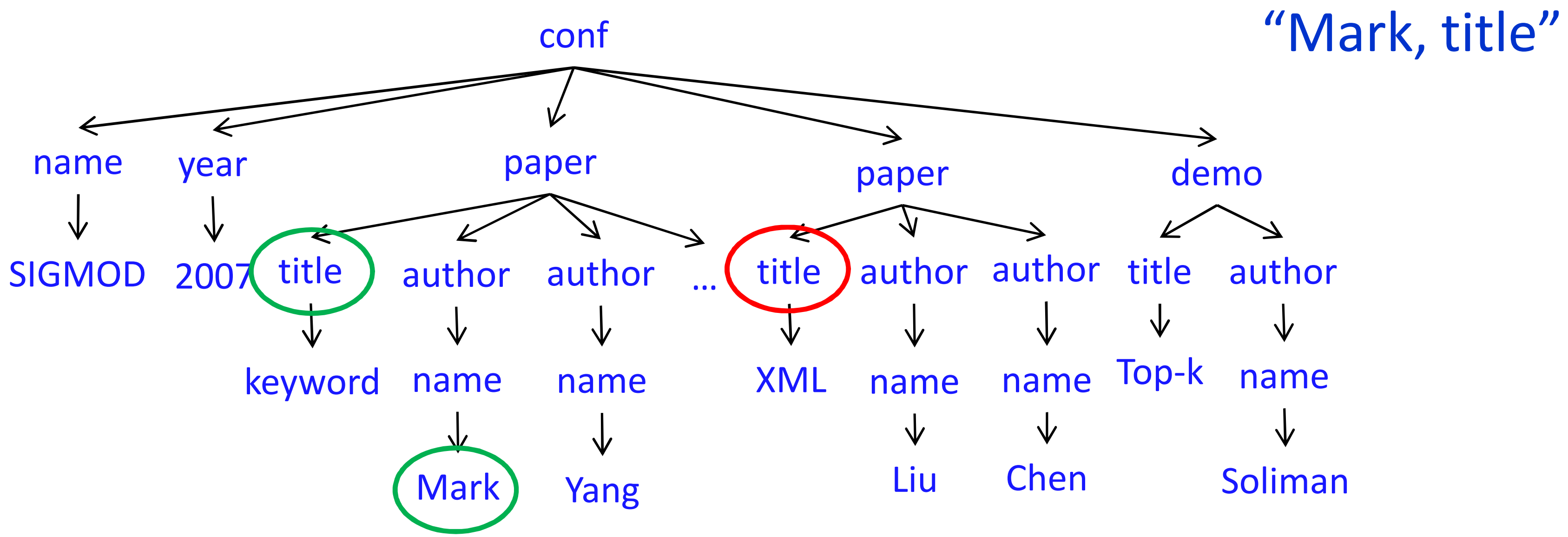
- Input:
  - Data: DB, XML, Web, Nested Graphs, etc.

	DB	XML	Web	Nested Graph
Node	tuple	element /attribute	webpage	object
Edge	foreign key	parent/ child	hyper-link	expansion / dataflow

  - Query **Q** = <k_1, k_2, ..., k_l>
- Output: "closely related" nodes that are "collectively relevant" to the query
  - The smallest trees covering all keywords.

Result Definition on XML & Trees /1

- In an XML tree, every two nodes are connected through their LCA.
- Not all connected trees are relevant, even if the size is small.
- The focus is defining query results to prune irrelevant subtrees.

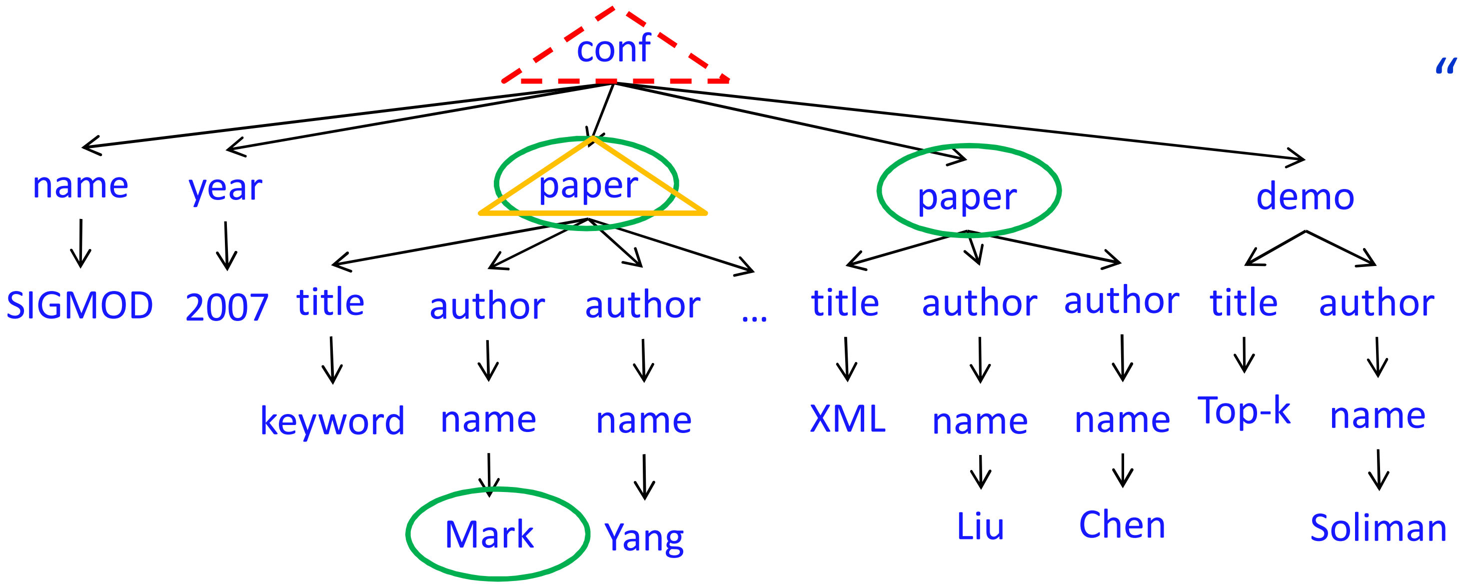


Result Definition on XML & Trees /2

- Typical approaches of result definition: pruning irrelevant matches based on
  - Tree structure: SLCA, ELCA, MLCA
  - Labels/Tags: XSEarch, CVLCA
  - Peer node comparisons: MaxMatch

Result Definition based on Tree Structure: SLCA [Xu et al. SIGMOD 05] & MLCA [Li et al. VLDB 04]

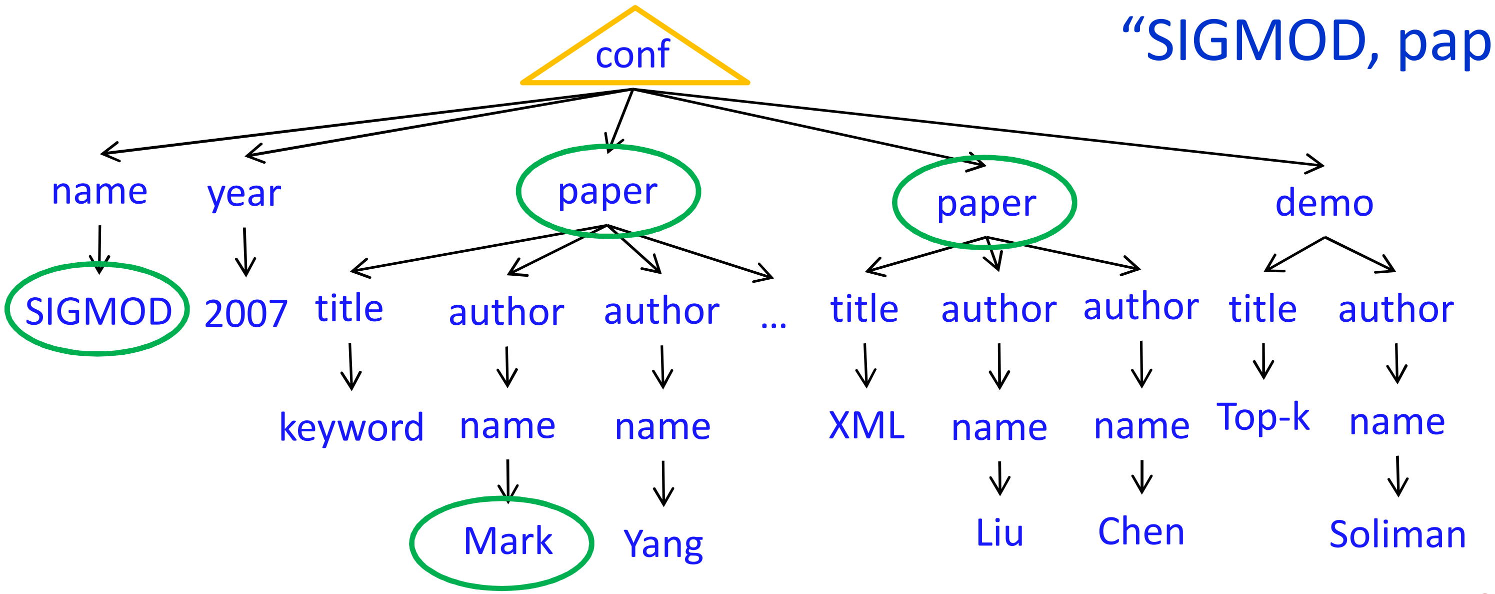
- 2-keyword queries
  - The shorter the distance b/w two nodes, the closer their relationship
  - For $Q=(K_1, K_2)$, with matches (M_{11}, M_{12}, M_2)
    If the LCA (M_{11}, M_2) is a descendant of LCA (M_{12}, M_2), then M_{11} is "***strictly closer***" to M_2 than M_{12}



"paper, Mark"

SLCA [Xu et al. SIGMOD 05] & MLCA [Li et al. VLDB 04]

- 3+-keyword queries:
  - SLCA: finding the subtrees with no proper subtree containing all keywords.
  - MLCA: finding a set of nodes, every pair is "closest".



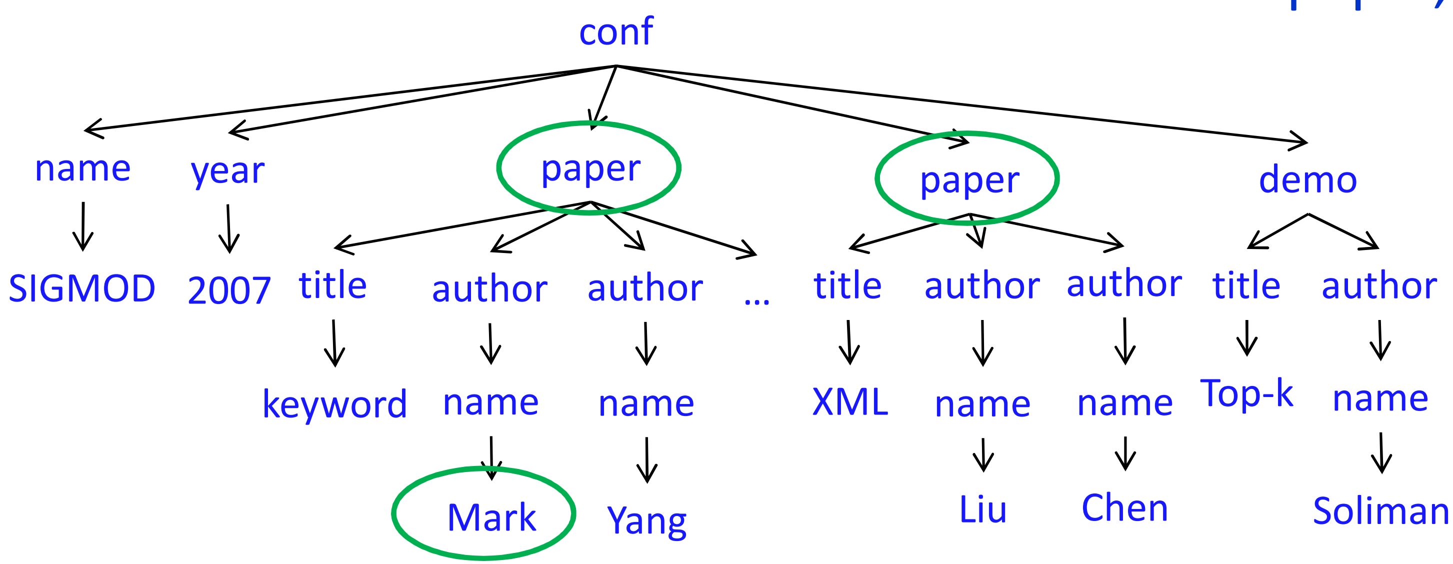
"SIGMOD, paper, Mark"

SLCA is a superset of MLCA.

Result Definition based on Labels: XSEarch [Cohen et al. VLDB 03]

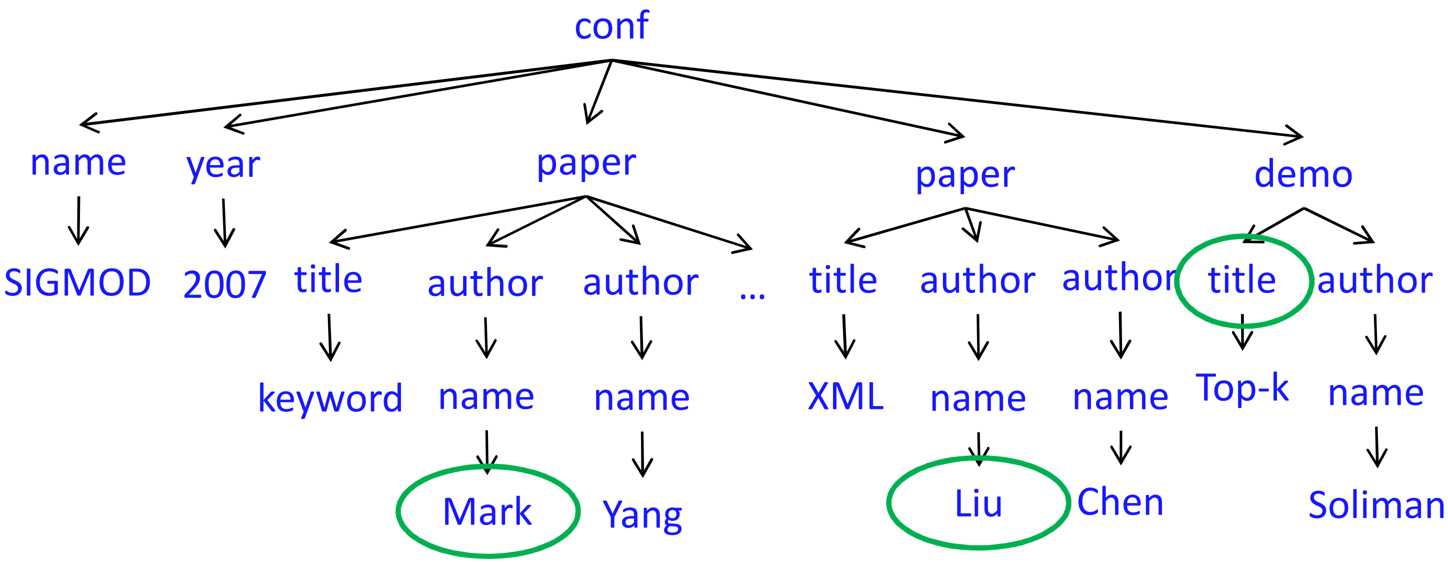
- 2-keyword queries:
  - Two nodes are interconnected if there's no two nodes with the same label on their path.
  - Intuitions: nodes with two same labels on their path are usually unrelated.

"paper, mark"



MLCA vs. XSEarch

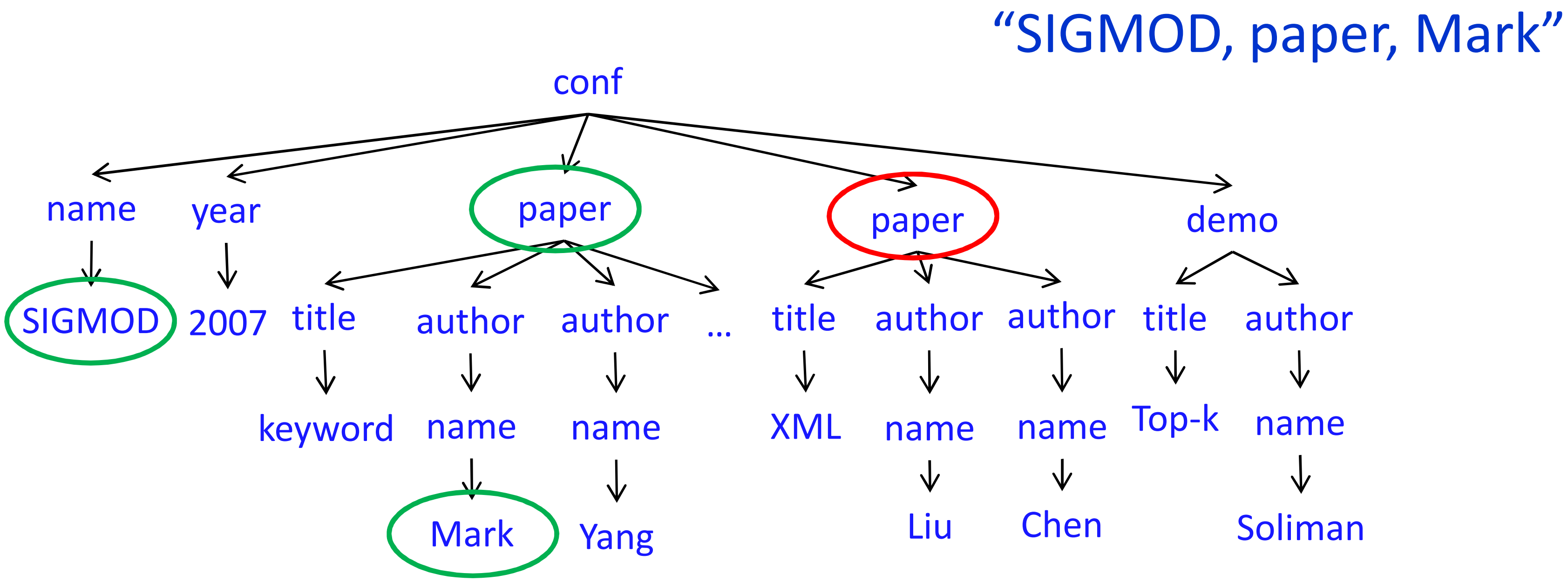
- MLCA and XSEarch use different inference of node relationships, and hence different results.



Closest, not interconnected.
Interconnected, not closest.

XSEarch [Cohen et al. VLDB 03]

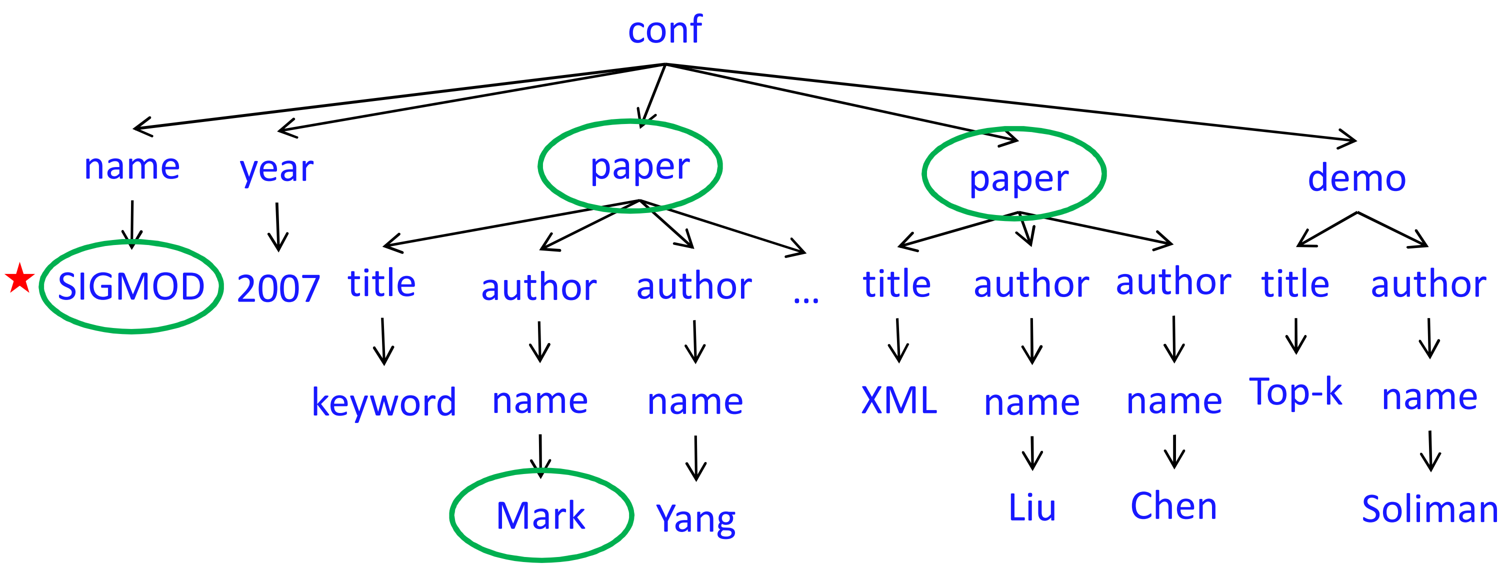
- 3+-keyword queries:
  - **All-pair Semantics**: every two keyword matches in a result are interconnected (MLCA also uses all-pair semantics)



XSEarch [Cohen et al. VLDB 03]

- 3+-keyword queries:
  - **Star Semantics:** each result has a "star" node, such that every other node is interconnected with it.

"SIGMOD, paper, Mark"

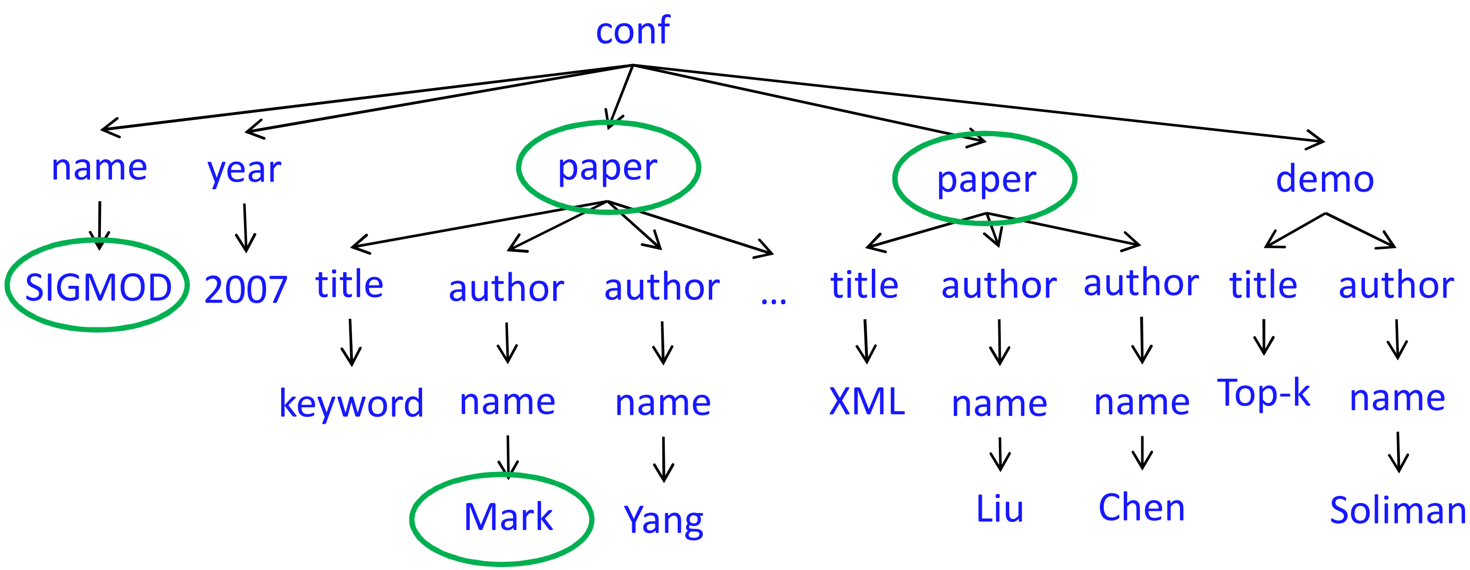


Relevant matches in Star semantics is a superset of those in all-pair semantics

Result Definition based on Peer Node Comparison: MaxMatch [Liu et al. VLDB 08]

- Intuition: pruning nodes with stronger siblings

“SIGMOD, paper, Mark”



Other Result Semantics on XML

- XReal [Bao et al. ICDE 09]
  - Inferring node types for result roots using data statistics
  - A result root node should
    - Be relevant to all keywords
    - Neither too low or too high
- Relaxed Tightest Fragments [Kong et al. EDBT 09]
  - An improvement of XSEarch aiming at reducing false negatives.

Result Quality Evaluation

- Given various heuristics, which approach will have a better search quality?
- Stay tuned, our talk later will discuss evaluation metrics
  - Empirical benchmark
  - Axiomatic framework

Efficiency

- Achieving all these semantics take polynomial time.
  - SLCA: $O(S_{min} k d \log S_{max})$
    - Multi-way SLCA [Sun et al. WWW 07] further improves the efficiency.
- Materialized views are proposed for further speedup of computing SLCA [Liu et al. ICDE 08 (poster)]
  - Results can be efficiently computed from materialized views of subqueries.
- Nodes are usually encoded using Dewey labels.

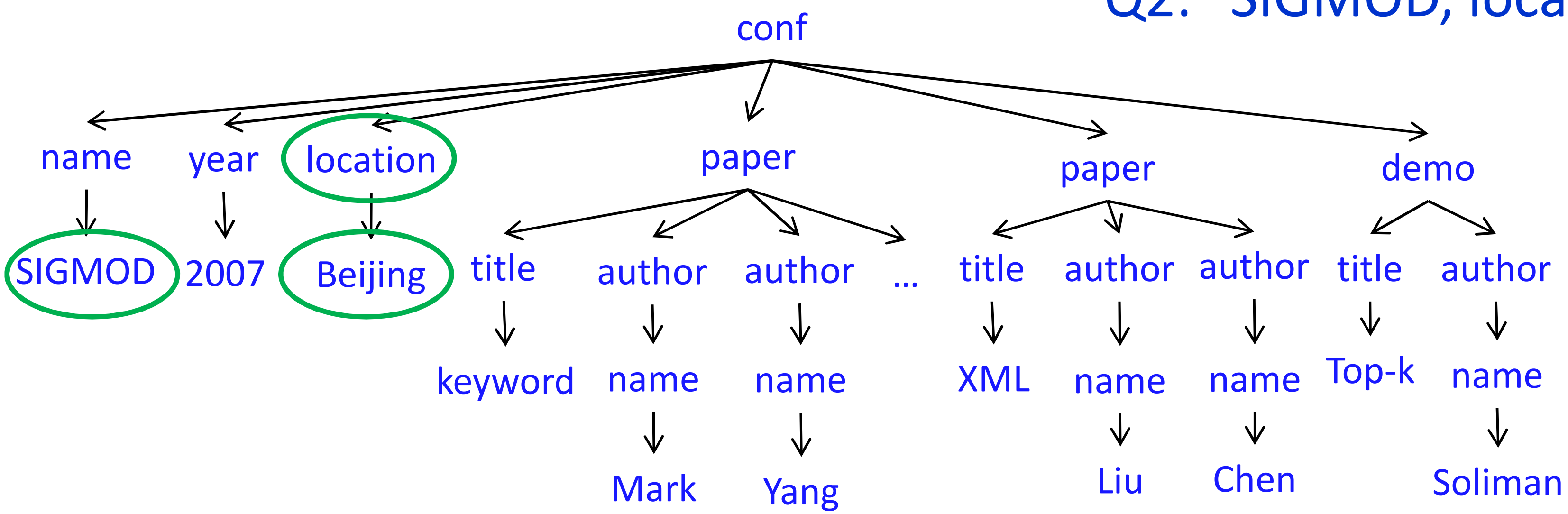
Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
  - Trees: Finding relevant matches; Finding relevant non-matches
  - Nested Graphs
  - Graphs
  - RDBMS
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

Relevant Non-matches /1 [Liu et al. SIGMOD 07]

- Besides keyword matches and the paths connecting them, other nodes may also be interested to the user.

Q1: "SIGMOD, Beijing"
Q2: "SIGMOD, location"



Similar relevant matches, different query semantics, and thus should have different query results

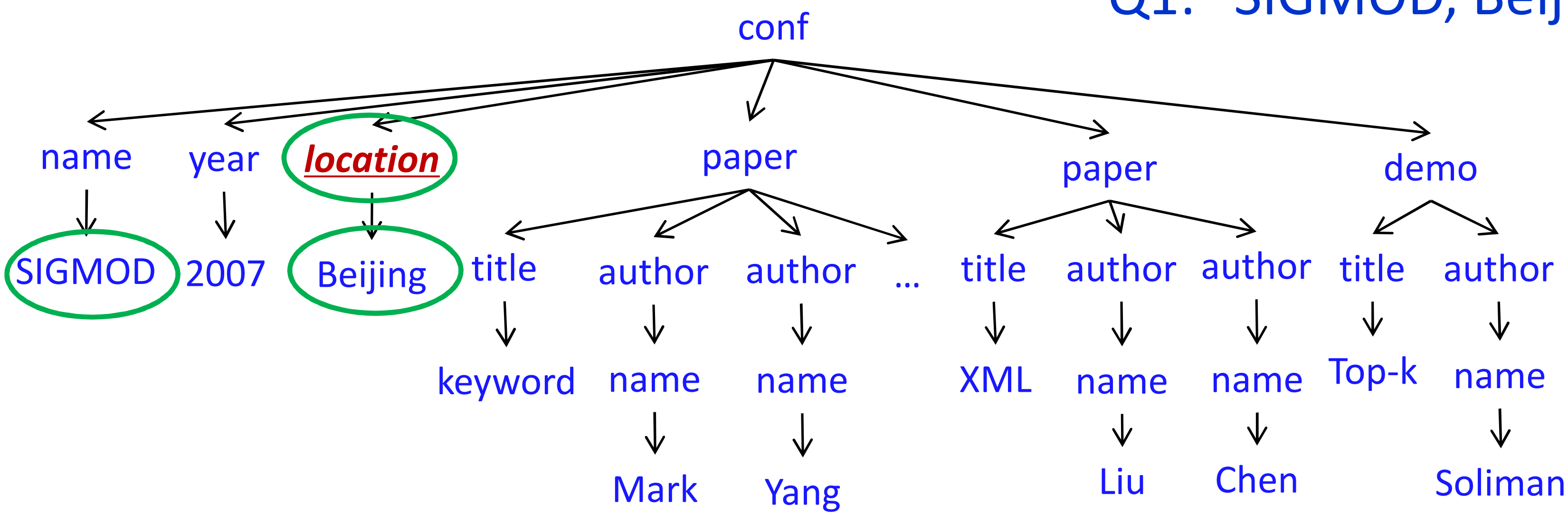
Relevant Non-matches /2 [Liu et al. SIGMOD 07]

- Similar as XQuery, Keywords can specify *predicates* or *return nodes*.
  - Q1: “SIGMOD, Beijing”
  - Q2: “SIGMOD, location”
- Return nodes may also be implicit.
  - Q1: “SIGMOD, Beijing” → return node = “conf”
- Information (subtree) of return nodes are potentially interesting, and considered as relevant non-matches.

Relevant Non-matches /3 [Liu et al. SIGMOD 07]

- Explicit return nodes: analyzing keyword match patterns
- Implicit return nodes: analyzing data semantics (entity, attribute) [Kimelfeld et al. SIGMOD 09 (demo)]

Q2: “SIGMOD, location”
Q1: “SIGMOD, Beijing”

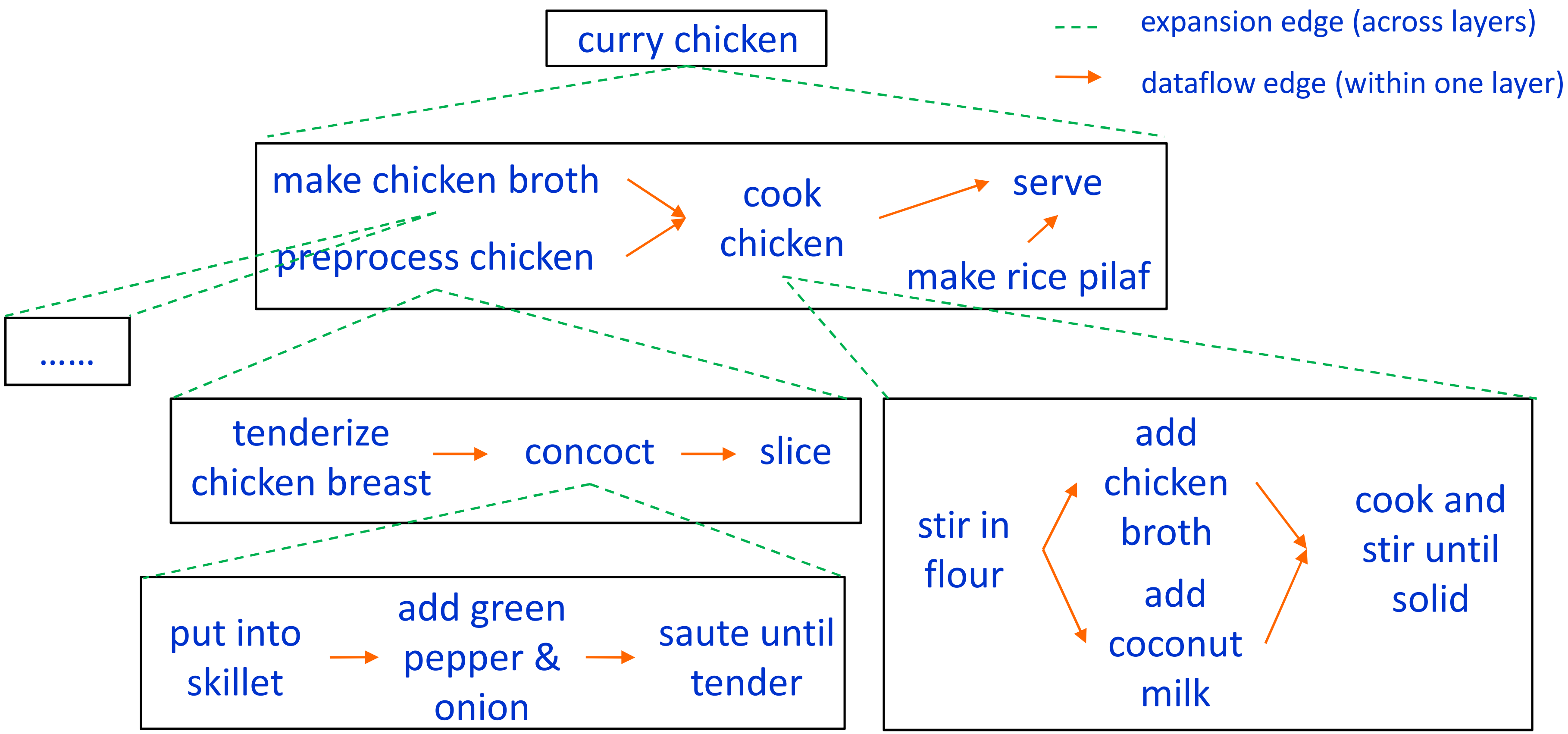


Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
  - Trees
  - Nested Graphs
  - Graphs
  - RDBMS
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

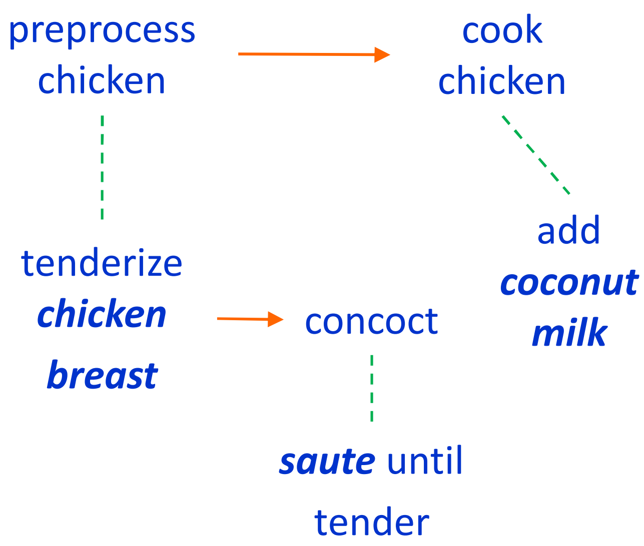
Searching Nested Graphs /1 [Shao et al. ICDE 09 (demo)]

- Multi-resolution data are used in workflows, spatial and temporal data.
- Workflows are widely used in scientific, business domains as well as in daily life.



Searching Nested Graphs /2 [Shao et al. ICDE 09 (demo)]

- Approaches for keyword search on graphs/trees (i.e. finding minimal trees) are not desirable



“chicken breast, coconut milk, saute”

- Not Informative: dataflows between tasks are lost.
  - do not capture the different semantics of edges in workflows
- Not self-contained: nodes in the result do not accomplish a task/goal.

Challenge: how to define desirable query results on nested graphs?

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
  - Trees
  - Nested Graphs
  - Graphs
  - RDBMS
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

Result Definitions for Graphs

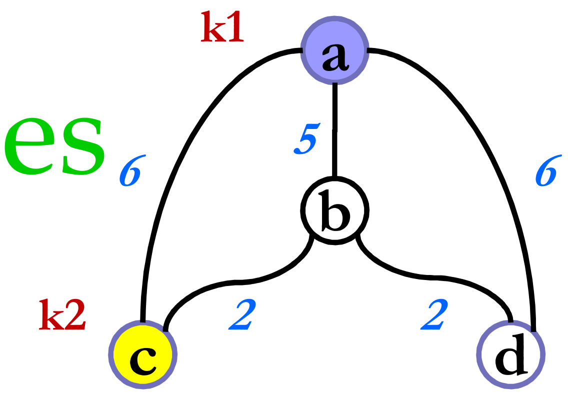
- Input:
  - Query **Q** = <k_1, k_2, ..., k_l>
- Outputs are "***closely related***" objects that are "***collectively relevant***" to the query
  - Graph — Schema-free
  - RDBMS — Schema-based
- Scoring/ranking methods
  - To be covered in Sec 3.

Evolution of Query Result Definitions

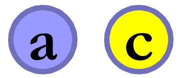
Schema-free

- Group Steiner Tree (GST)
  - Dynamic Programming or Mixed Integer Programming
  - Lawler's framework
- Approximate Group Steiner Tree
  - BANKS 1/2/3, BLINKS — O(l)-approximation to 1-GST
    - STAR [Kasneci et al, ICDE09] — O(log l)-approximation 1-GST
  - Distinct root semantics
- Subgraph-based
  - Community
  - EASE

Closely Related Nodes



a – c : 6

- Obtaining the graph
  - From DB, XML, Web, RDF, etc.
  - (***Un***)***directed*** (weighted) ***graph*** G = < V, E, **w**>
  - Matching/keyword nodes 

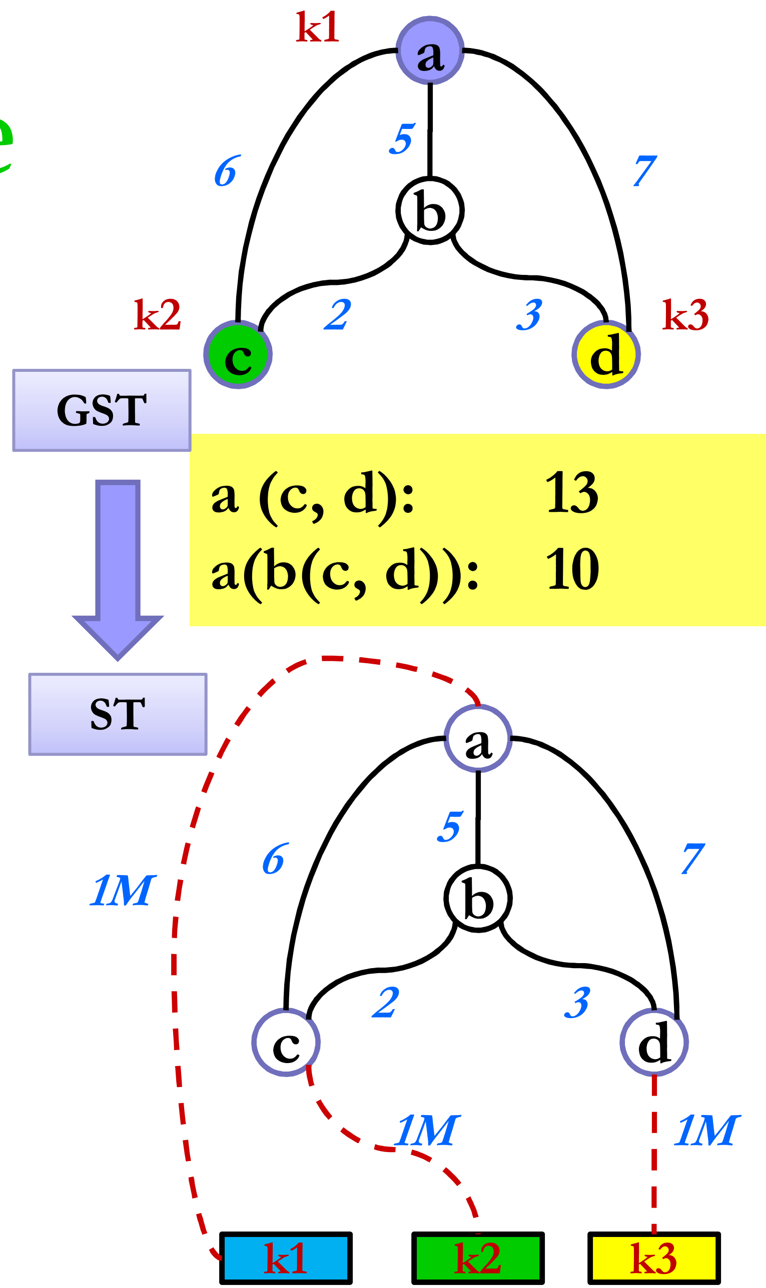
- If only two keywords
  - Shortest path !
  - k-shortest paths

Group Steiner Tree

Steine nodes ⓑ

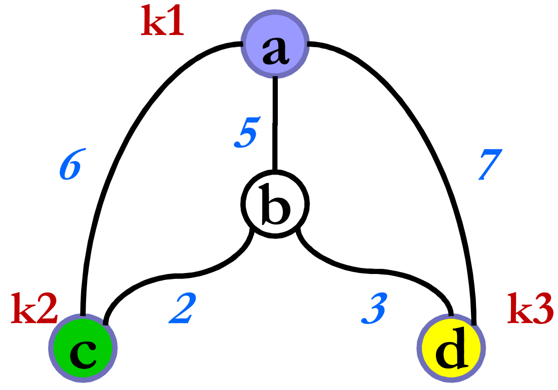
- Steiner Tree
  - A connected tree in G that ***spans*** a set of node S_i
  - S_i are collectively relevant to the query
- Group Steiner Tree [Li et al, WWW01]
  - Spanning from one node from each group
- top-1 GST = top-1 ST

☒NP-hard ☑Tractable for fixed l



Dynamic Programming for GST-1 [Ding et al, ICDE07]

- Recurrence equations
  - $T(n, \mathbf{Q}) = 0$
  - $T(v, \mathbf{Q}) = \min(T_g(v, \mathbf{Q}), T_m(v, \mathbf{Q}))$
  - $T_g(v, \mathbf{Q}) = \min_{(v,u)\in E} ((v, u) \oplus T(u, \mathbf{Q}))$
  - $Tm(v, \mathbf{Q}) = \min_{\mathbf{Q1} \subseteq \mathbf{Q}} (T(v, \mathbf{Q1}) \oplus T(v, \mathbf{Q} \setminus \mathbf{Q1}))$



a (c, d): 13
a(b(c, d)): 10

$T(a, 123) = \min(T_g(a, 123), T_m(a, 123))$

$T_g(a, 123) = \min(5+T(b, 23),$
$6+T(c, 23),$
$7+T(d, 23))$

$T_m(a, 123) = \min(T(a, 12)+T(a, 3),$
$T(a, 13)+T(a, 2),$
$T(a, 23)+T(a, 1))$

DP for GST-k

- Keep running GST-1 until k results are obtained ➔ approximate answer
- Complexities (GST-1, GST-k)
  - Time: $O(3^l n + 2^l((l+\log n)n + m))$
  - Space: $O(2^l n)$

If $l=O(1)$

$O(n\log n + m)$

$O(n)$

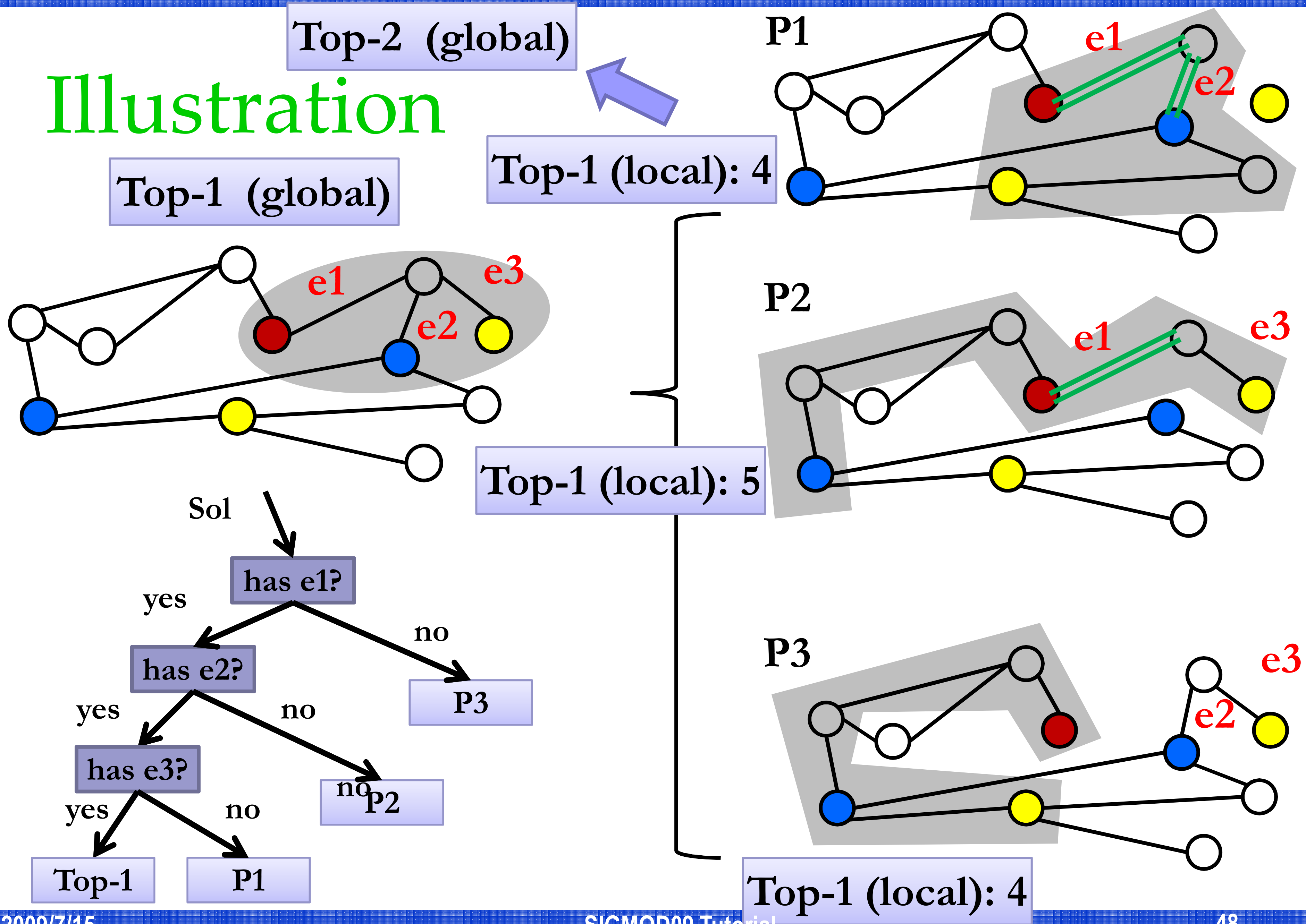
From top-1 to top-k Exactly

- Lawler's Framework [Lawler, 1972]
  - Discrete optimization problem ➔ Enumeration problem
  - Input
    - A way to partition the solution space
    - An algorithm to find top-1 solution in a (constraint) solution space
  - Output
    - Top-k solution in the entire solution space (with good running time properties)
  - c.f. [Cohen, et al. ICDE09] tutorial

Finding top-k GST [Kimelfeld et al, PODS06]

- Idea
  - Steiner tree can be found efficiently for fixed number of keywords
  - Apply Lawler's framework
    - Intricate technical details to find solution under inclusion constraints
- Algorithm:
  - Q.enqueue(ST(G))
  - While Q not empty
    - <T, I, E> = Q.dequeue()
    - $\{e_1, \ldots, e_k\}$ = edges(T) \ I
    - Generate k partitions ($E' = e_{k-i}$, $I' = \{e_1, \ldots, e_i\}$) and Queue.enqueue(CST(G), I', E')

Illustration



MIP [Talukdar *et al*, VLDB08]

- Top-1 Steiner Tree
  - Mixed Linear Programming (MIP) to find the minimum Steiner Tree rooted at *r*
    - Can also solve a constrained version of the problem
  - Call this procedure for each node *r* in the graph
- Applying Lawler's framework to obtain top-k Steiner Trees

- Approximate solutions for larger graph
  - Reduce *G* to *G'*, where only *m* shortest paths between every pair of keyword nodes are kept

Approximate GST-k

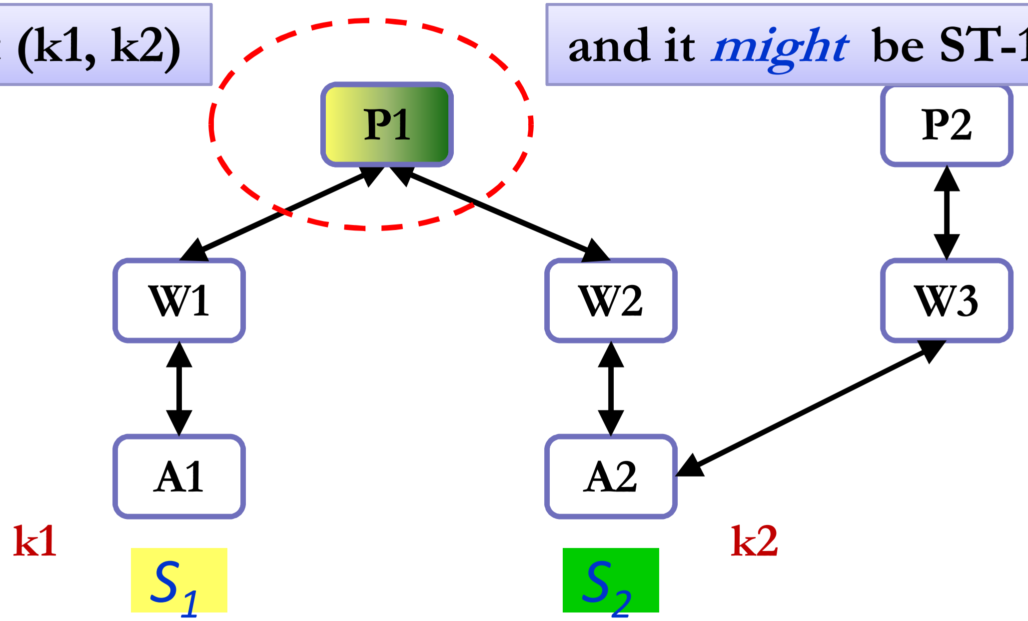
- BANKS1 [Bhalotia et al, ICDE02]
  - Result definition: Group Steiner Trees
- Approximate *ST-k*s using *ST*s
  - a ***backward*** expansion search algorithm
  - Run multiple Dijkstra's single-source-shortest-path algorithms iteratively until k answers are found ➔ equi-distance expansion
- No guarantee on the quality of its top-*k* results

P1 is the root of a ST wrt (k1, k2)

and it *might* be ST-1

Example

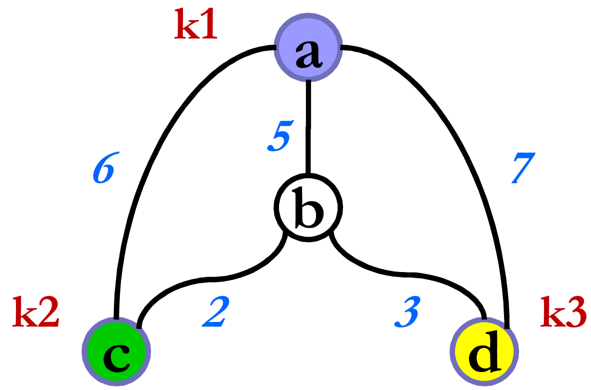
```
A: Author
W: Writes
P: Paper
```



- While (!quit)
  - Execute the iterator, I_j , whose output node, v_j, has the least "distance" from its source
  - v_j.reachable_from[label(I_j)] $\cup$= source(I_j)
  - If v is reachable from at least one source in every S_i
    - OutputHeap << GenResult(v_j) // result = Π(reachable sources)

      // current best result emitted when heap is full

BANKS2 [Kacholia et al, VLDB05]



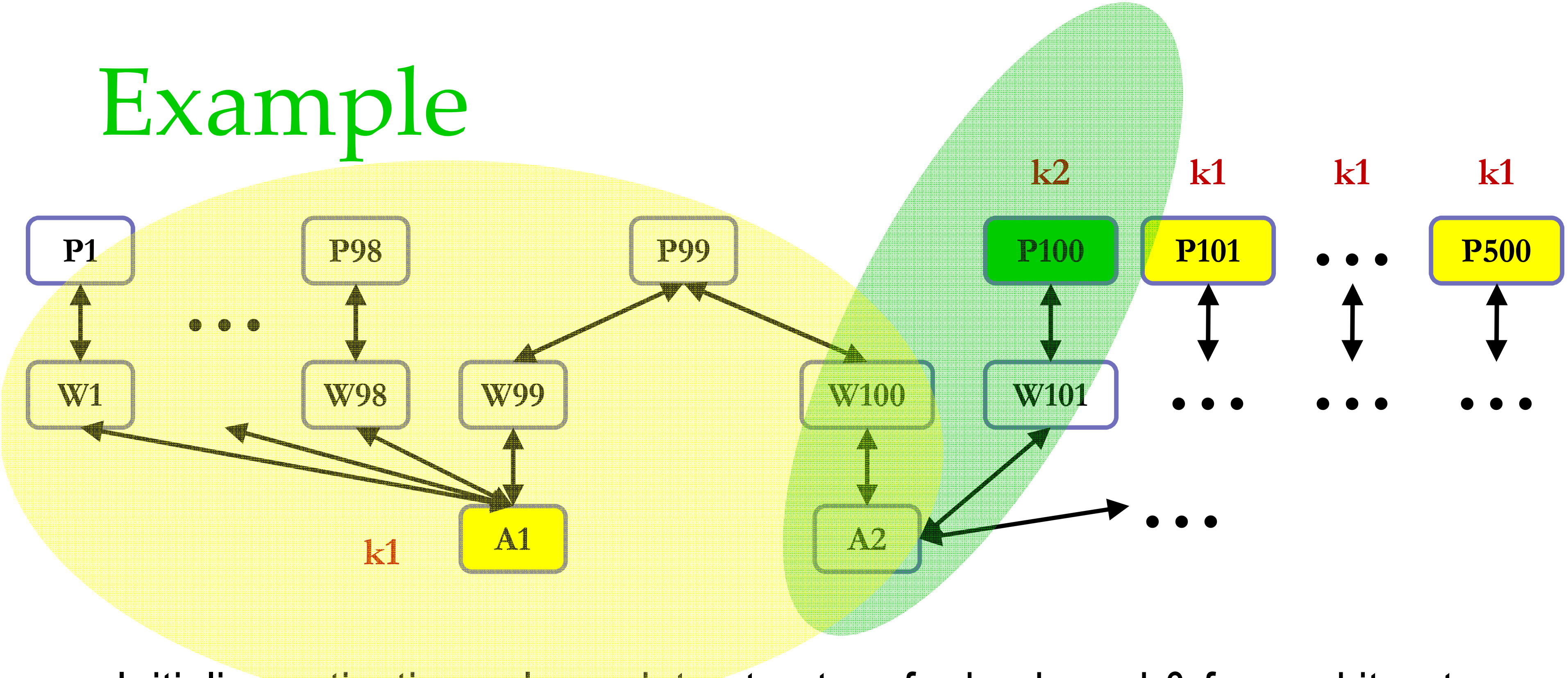
- ***Distinct root semantics***
  - Find trees rooted at *r* s.t it minimizes $cost(T_r) = \sum_i cost(r, match_i)$
  - A tree ➔ a set of paths
- Why?
  - Fits into backward expansion search algorithms (BANKS1) perfectly
  - Favors trees with small radii
- Algorithmic ideas:
  - bi-directional search + activation mechanism

a (c, d): 13

~~**a(b(c, d)): 10**~~

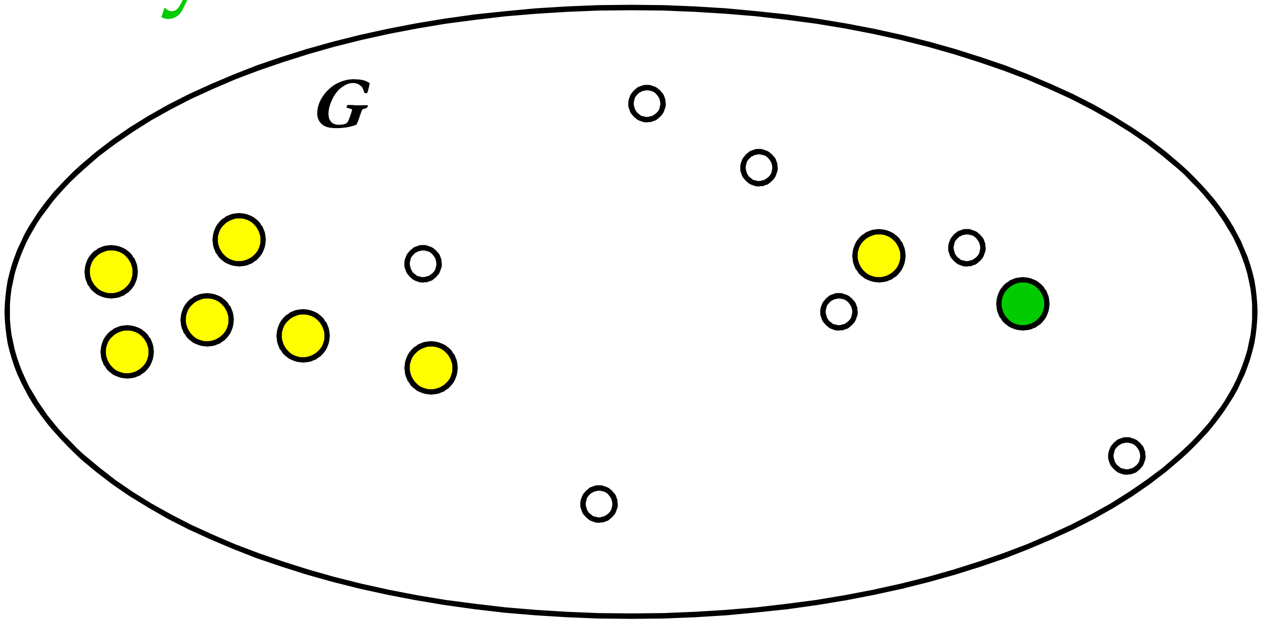
{a↪a, a↪b, a↪d}	0+7+8

Example



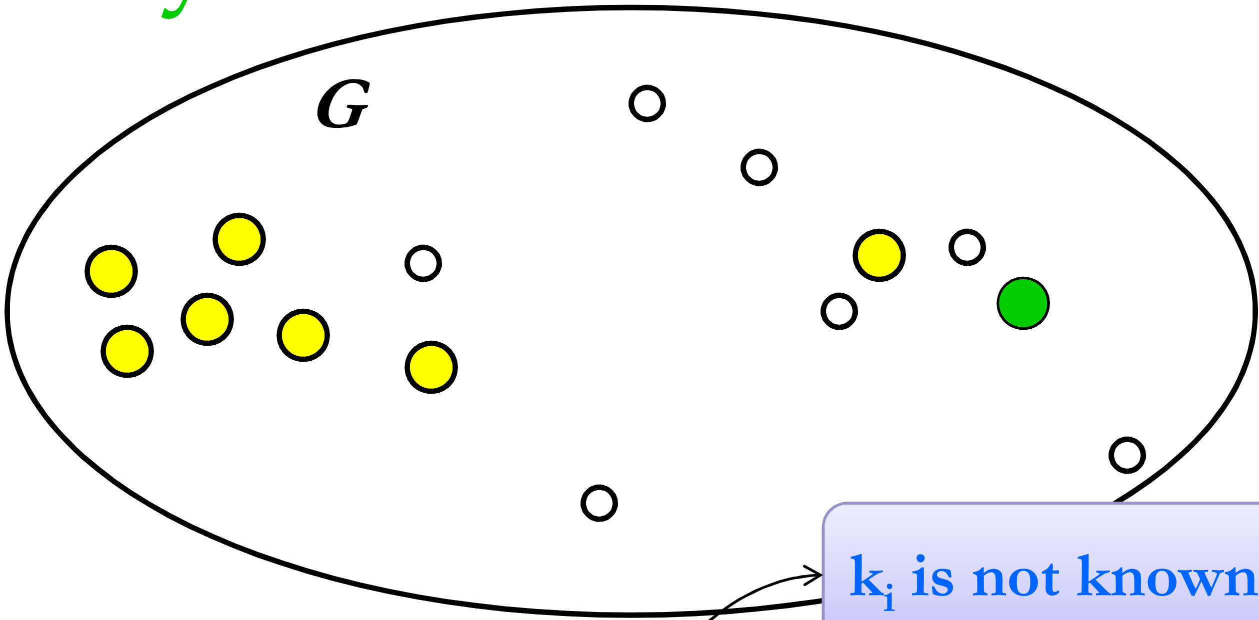
- Initialize activation values, data structure for backward & forward iterators
- While (!quit)
  - Explore the nodes with the highest activation value (consider both iterators)
  - Spread the activation to its neighbors
  - Update the min dist from *v* to each of the search terms (and other data structures)

Proximity Search [Goldman et al, VLDB98]



- Distinct root semantics
- Foreach root candidates r_i 
  - Cost(r_i) = Cost(r_i, k1) + Cost(r_i, k2)
  - Keep only the top-k min cost roots

Proximity Search [Goldman et al, VLDB98]



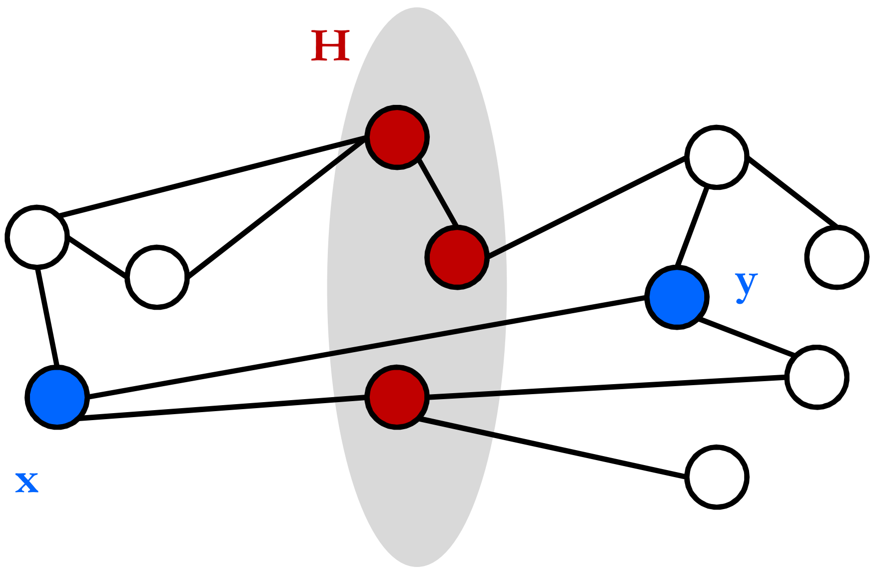
k_i is not known a priori

2 Choices:
(1) Index node-node distance, or
(2) Index node-keyword distance

- Distinct root semantics
- Foreach root candidates r_i
  - Cost(r_i) = Cost(r_i, k1) + Cost(r_i, k2)
  - Keep only the top-k min cost roots

Indexing Node-Node Min Distance

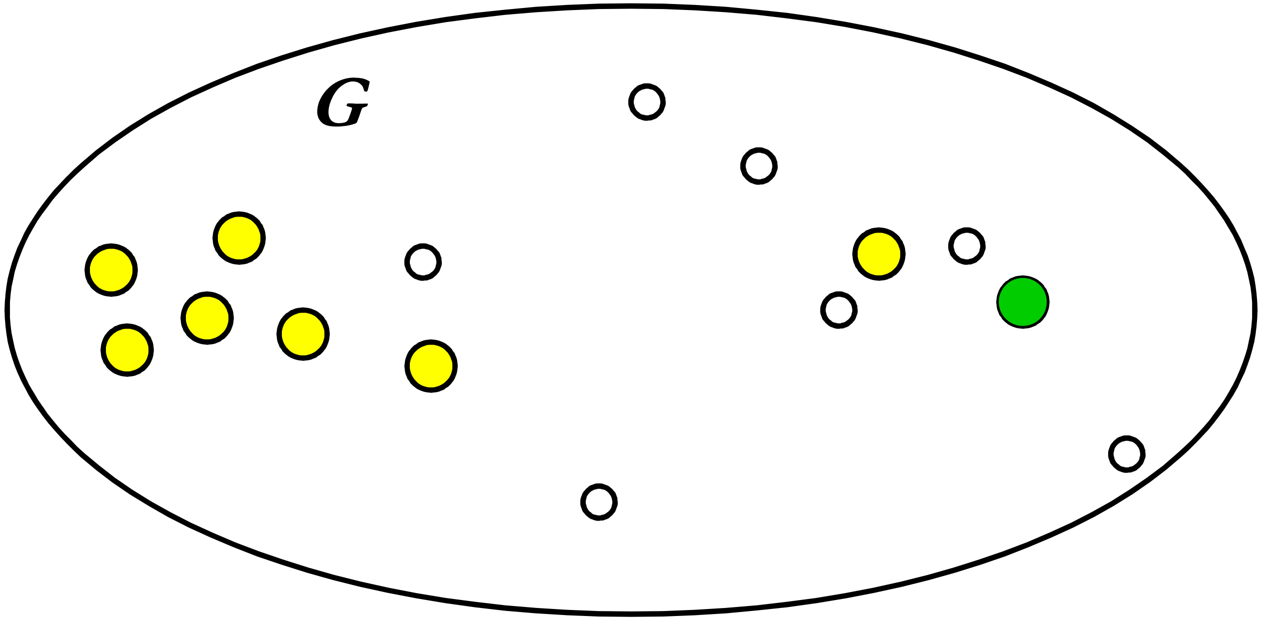
- $O(|V|^2)$ space is impractical
- Select hub nodes (H_i)
  - d*(u, v) records min distance between u and v *without* crossing any H_i



- Using the ***Hub Index***

$$d(x, y) = \min\left(d^*(x, y),\ d^*(x, A) + d^H(A, B) + d^*(B, y), \quad \forall A, B \in H\right)$$

SLINKS /1 [He et al, SIGMOD07]

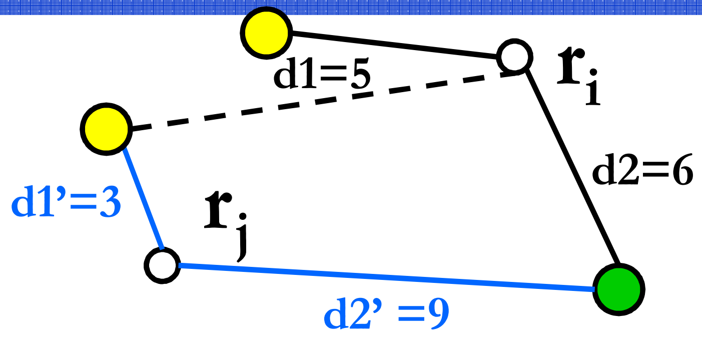


- Distinct root semantics
- Foreach root candidates r_i ○
  - Cost(r_i) = Cost(r_i, k1) + Cost(r_i, k2)
  - Keep only the top-k min cost roots

(2) Index node-keyword distance

Use Fagin's TA Alg.

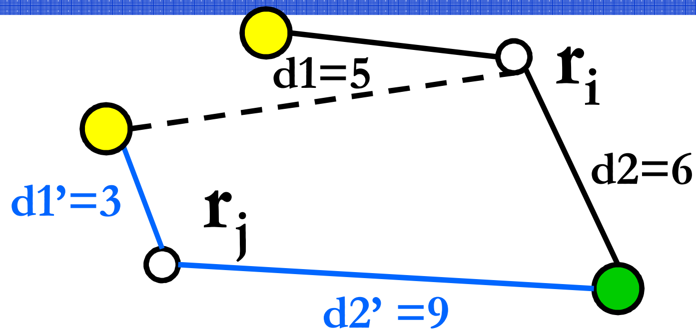
SLINKS /2



r	d1	d2
ri	5	6
rj	3	9

- Formulate it as a top-k problem
  - Each candidate root r_i has l attributes $d_1, d_2, \ldots, d_l$
    - $D_j = d(r_i, k_j)$
  - $\text{Score}(r_i) = r_i.d_1 + r_i.d_2 + \ldots + r_i.d_l$
- Input: for each d_j, sort r_i in increasing order
- Threshold Algorithm (TA)
  - While (less than k results)
    - Visit the next r from d_i's list (round-robin) // backward expansion using index
    - Find r's missing d_i values, if any // forward expansion using index
    - Maintain score lower bound, etc. // book-keeping

SLINKS → BLINKS



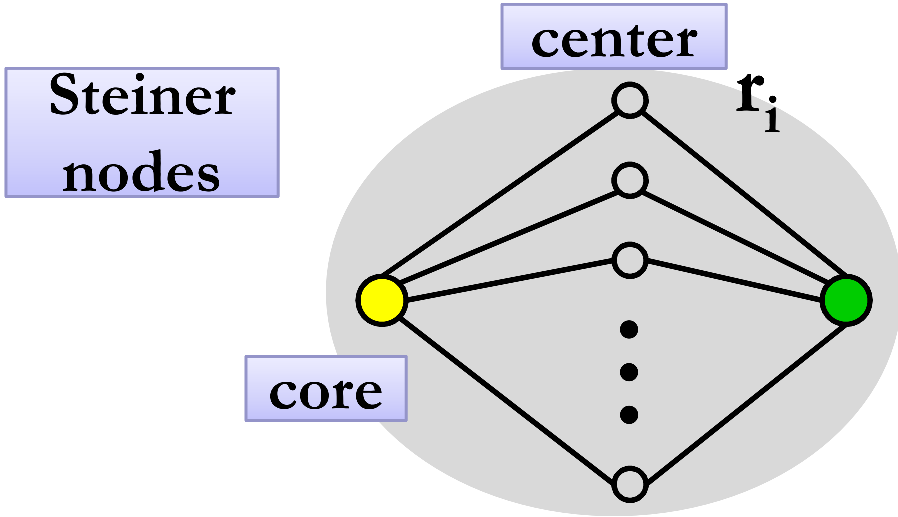
- SLINKS requires backward + forward indexes
  - Between nodes and keywords
  - Thus O(K*|V|) space ≈ $O(|V|^2)$ in practice
- BLINKS
  - Partition the graph into blocks
    - Portal nodes shared by blocks
  - Build intra-block, inter-block, and keyword-to-block indexes

Other Related Methods

- GST and its approximation
  - Information Unit [Li et al, WWW01]
    - Growing a forest of MSTs (minimum spanning trees)
  - BANKS3 [Dalvi et al, VLDB08]
    - Use graph clustering to handle external graphs
- Distinct root semantics
  - [Tran et al, ICDE09]
    - Considers more complex ranking functions

Community [Qin et al, ICDE09]

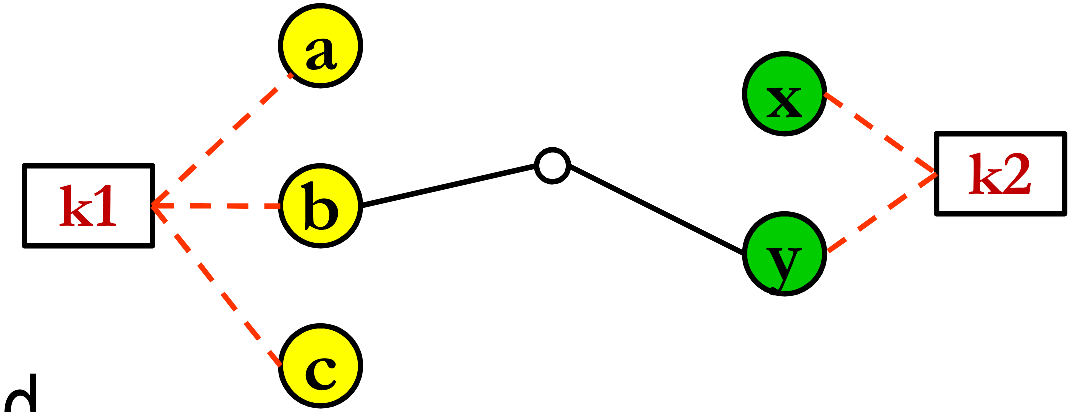
- Redundancy affects
  - Distinct root semantics
  - GST
- ***Community*** | R_{max}
  - Idea: GROUP BY (unique keyword nodes combinations)



Community-finding Algorithms

- Nested loop
  - Enumerate core node combinations
- Bottom-up search
  - BANKS 2, BLINKS (using index)
- Top-down search
  - Proximity search (using index)
- Polynormial delay enumeration
  - Backward search to find the best root
  - Partition the solution space and apply Lawler's method

Example

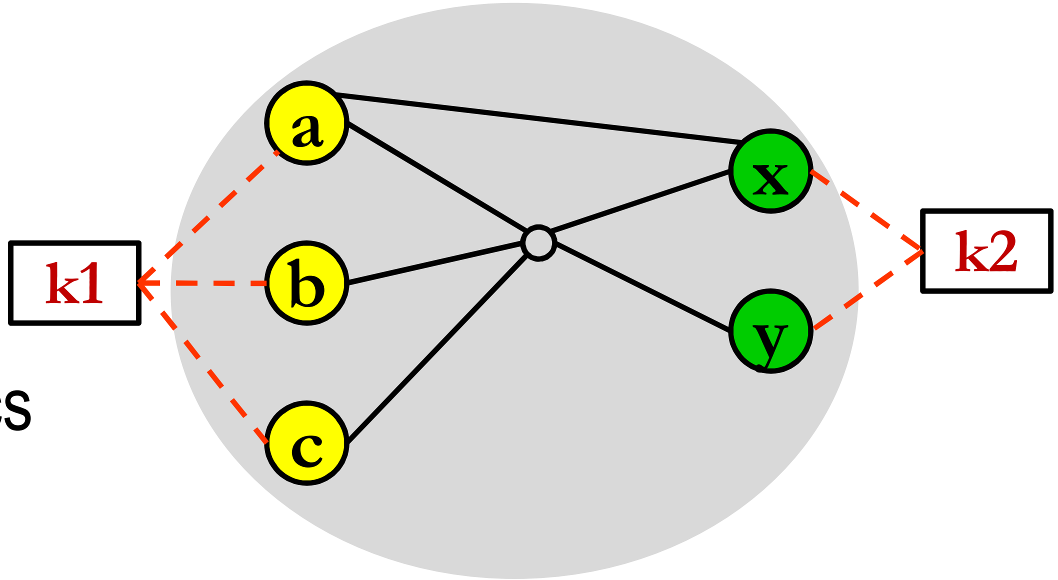


- Solution space
- 2 partitions generated
  - (b, ¬y)
  - (¬b, *)

c		
b		✓
a		
	x	y

EASE /1 [Li et al, SIGMOD08]

- Redundancy affects
  - GST
  - Distinct root semantics
  - Community



- ***Subgraphs*** as results | r

EASE /2

- r-Radius graph (r-G) ➔ r-Radius Steiner graph (r-SG), given **Q**
  - By removing useless nodes
  - Also introduced maximal r-G/r-SG
- Keyword query results are x-SGs that contain all/some the search keywords ($x \leq r$)
- Index (keyword pair ➔ (maximal r-Gs, *sim*))
  - *sim* is used to compute the final score
- TA-style algorithm to find top-k r-SGs

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
  - Trees
  - Nested Graphs
  - Graphs
  - RDBMS
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

Keyword Search for RDBMSs

Schema-based

- Running example
  - **A**uthor(aid, name)
  - **P**aper(pid, title)
  - **W**rites(aid, pid)

Schema Graph:

Author ← Writes → Paper

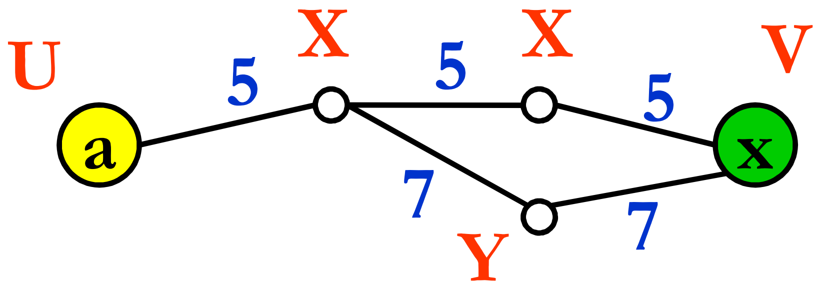
- Keyword queries as ***query interpretation***
  - "Widom XML" $\sigma_{\text{"widom"}}(A) \bowtie W \bowtie \sigma_{\text{"xml"}}(P)$
  - "XML Trio" $\sigma_{\text{"xml"}}(P) \bowtie W \bowtie A \bowtie W \bowtie \sigma_{\text{"trio"}}(P)$

$\sigma_{\text{"trio"}}(A) \bowtie W \bowtie \sigma_{\text{"xml"}}(P)$, …

Candidate Network (CN) $A^{trio} - W - P^{xml}$

What if "trio" is also a person's name?

Why CNs?



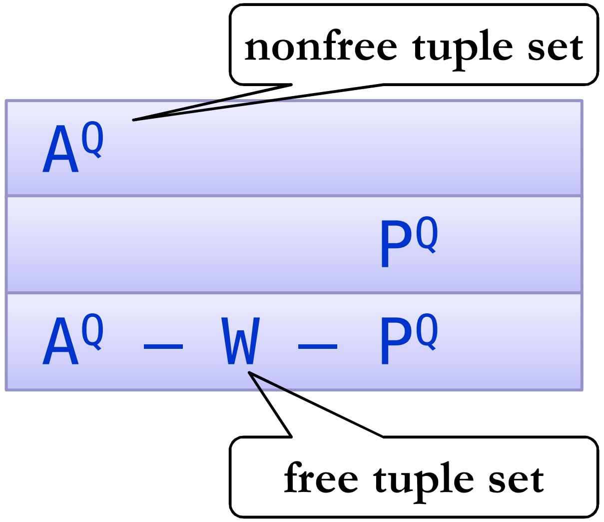
U – X – X – V	0
U – X – Y – V	19

- Advantages
  - Query driven
  - Compensate for normalization
  - Perspectives
- Differences with graph-based approaches
  - Reflect one's prior belief
    - Précis [Koutrika et al, ICDE06], Recommending CN [Yang et al, ICDE09], Interconnection Semantics [Cohen and Sagiv, ICDT05], Disambiguation: SUITS [Zhou et al, 2007]
  - Can leverage IR/other ranking principles
    - [Liu et al, SIGMOD06], SPARK [Yi et al, SIGMOD07]

DISCOVER [Hristidis et al, VLDB02]

- Consider enumerating all the *necessary* CNs
  - up to a size limit T_{max}
  - Minimum set of join expressions to execute
  - allow multiple occurrence of a relation as cmp'ed with DBXplorer [Agrawal et al, ICDE02]

$T_{max} = 3$



Query Processing

1. Construct non-free tuple sets
   - Via inverted index
2. Generate all the valid CNs
   - Breadth-first enumeration on the database schema graph
   - + pruning
3. Rewrite the list of CNs into an execution schedule
   - Usually top-k retrieval
   - ***Most algorithms differ here***

Schema Graph: $A^Q \leftarrow W \rightarrow P^Q$

Generating CNs

Not minimal

Non-promising

- Input
  - non-free tuple sets
- Output
  - all valid CNs no larger than T_{max}
- Method
  - Breadth-first search + pruning

1	A^Q
2	P^Q
3	$A^Q – W$
4	$W – P^Q$
5	$A^Q – W$ A^Q
6	$W – P^Q$ P^Q
…	…
9	$A^Q – W – P^Q$
…	…
12	$A^Q – W – P^Q$ $A^Q – W$
13	$W – P^Q$ $A^Q – W – P^Q$
…	…

DISCOVER2 [Hristidis et al, VLDB03]

1. Construct non-free tuple sets
2. Generate all the valid CNs
3. Execution algorithms optimized for top-*k* queries
   - Naïve → Sparse → Single pipelined/Global pipelined

Push top-k constraints inside !

Naive

- Naive
  - Retrieve top-k results from each CN
    - `ORDER BY + LIMIT`
  - Merge them to obtain top-k query result
  - Can be optimized to share computation

```
SELECT * FROM P, W, A
WHERE P.pid = W.pid AND P.aid = A.aid
    AND P.title MATCHES 'xml, trio'
    AND A.name MATCHES 'xml, trio'
ORDER BY score_p + score_a
LIMIT 2;
```

top-2

CN1 P^Q – W – A^Q

CN2 P^Q – W – A – W – P^Q

Result (CN1)	Score
P1-W1-A2	3.0
P2-W5-A3	2.3
…	…

Result (CN2)	Score
P2-W2-A1-W3-P7	1.0
P2-W9-A5-W6-P8	0.6
…	…

Naive → Sparse

- Sparse
  - Execute 1 CN at a time
    - start from the smallest CNs
  - Prune the rest of the CNs using the current top-k score & MPSs of the remaining CNs.

$$\text{score}(P1 - \dots - P1) \geq \text{score}(Px - \dots - Py)$$
$(x>1, y>1)$

top-2

CN1 $P^Q - W - A^Q$

Result (CN1)	Score
P1-W1-A2	3.0
P2-W5-A3	2.3
…	…

CN2 $P^Q - W - A - W - P^Q$

Result (CN2)	Score
P1-W?-A?-W?-P1	1.5

Best case scenario → Max Possible Score

- No need to execute CN2 !
- Requires "*score monotonicity*"

Pipelined /1

- Motivation
  - What if join result >> k ?
- Top-k optimization *within* a CN

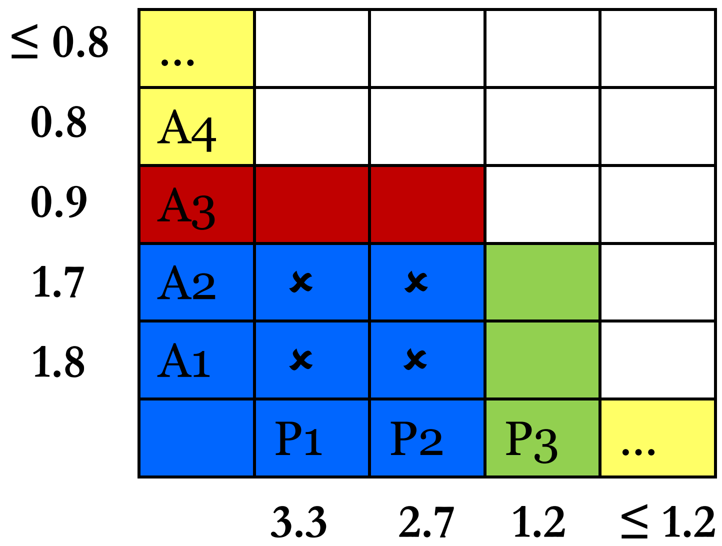
top-2

CN1 $P^Q - W - A^Q$

Result (CN1)	Score
P1-W1-A2	3.0
P2-W5-A3	2.3
…	…

MPS(P3 – W? – [A1, A2]) = (1.8+1.2) /3 = 1.0

MPS([P1, P2] – W? –A3) = (3.3+0.9) /3 = 1.4




```
SELECT * FROM P, W, A
WHERE P.pid = W.pid AND P.aid = A.aid
    AND P.pid in (P1, P2)
    AND A.pid = A3
```

Pipelined /2

top-2

CN1 P^Q – W – A^Q

- Motivation
  - What if join result >> k ?
- Top-k optimization ***within*** a CN

≤ 0.3	…				
0.3	A4	≤1.2	≤1.2		
0.9	A3	1.4	1.2	≤1.0	
1.7	A2	×	×	≤1.0	
1.8	A1	×	×	≤1.0	
		P1	P2	P3	…
		3.3	2.7	1.2	≤ 1.2

MPS(P3 – W? – [A1, A2]) = (1.8+1.2) /3 = 1.0

MPS([P1, P2] – W? –A3) = (3.3+0.9) /3 = 1.4

Result (CN1)	Score
P1-W8-A3	1.4
P2-W9-A3	1.2
…	…

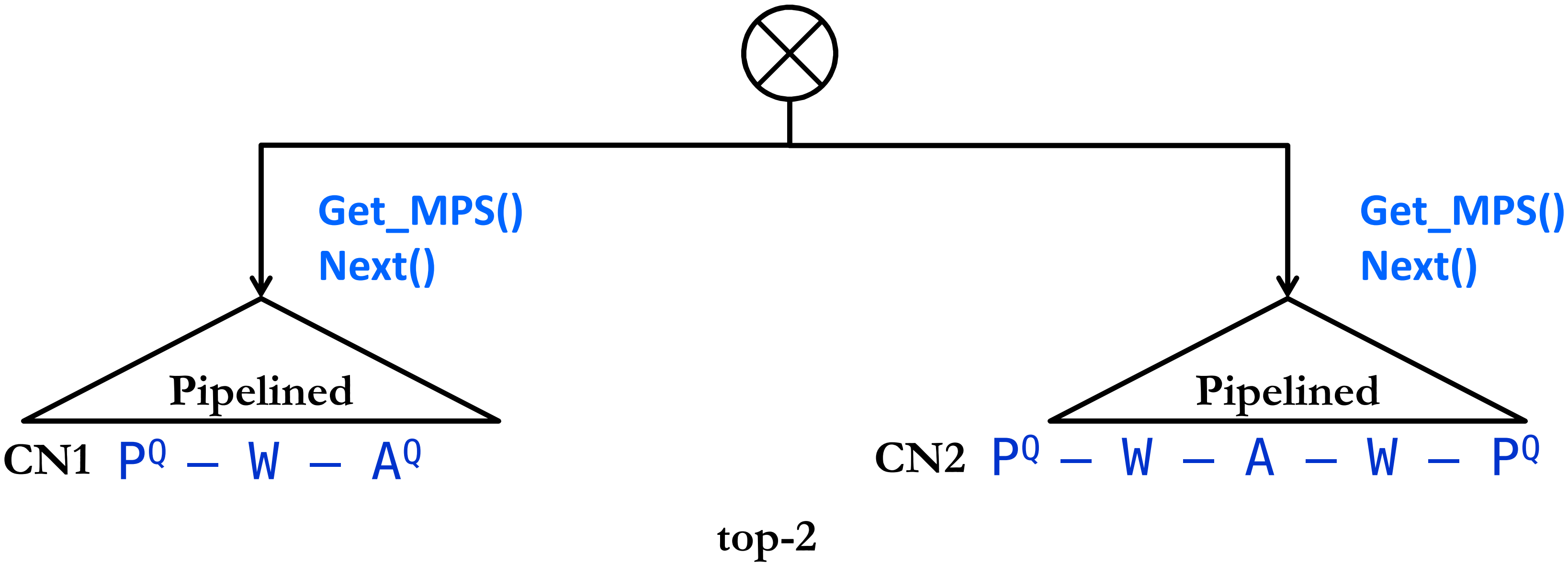
Can we stop?

MPS([P1, P2] – W? –A4) = (3.3+0.3) /3 = 1.2

Global Pipelined

Naive ➔ Sparse ➔ Pipelined
- *Be lazy!*
- *Utilize upper bound estimates*

- Run Pipelined on each CN in an interleaving way
  - Determined by CN's MPS



SPARK
[Luo et al, SIGMOD07]

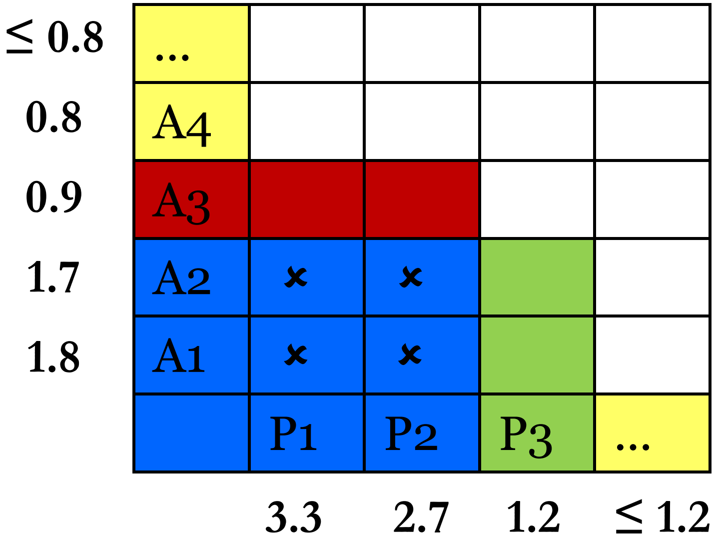
- Motivation
  - What if (# of red cells) >> k ?
- Skyline Sweeping
  - Perform 1 probe each time
  - Push "neighbors" to a heap based on their MPSs

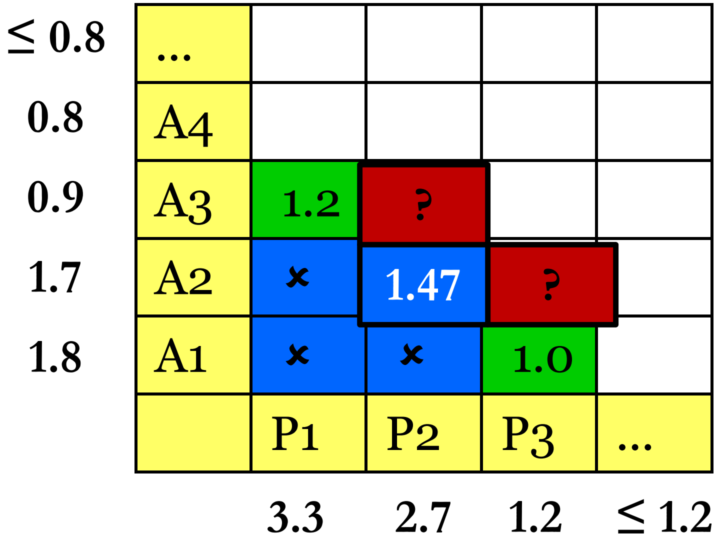
top-2

CN1 $P^Q - W - A^Q$

Temp Results	Score
P2-W7-A2	1.47

MPS(P2 – W? –A3) = 1.2
MPS(P3 – W? –A2) = 0.97

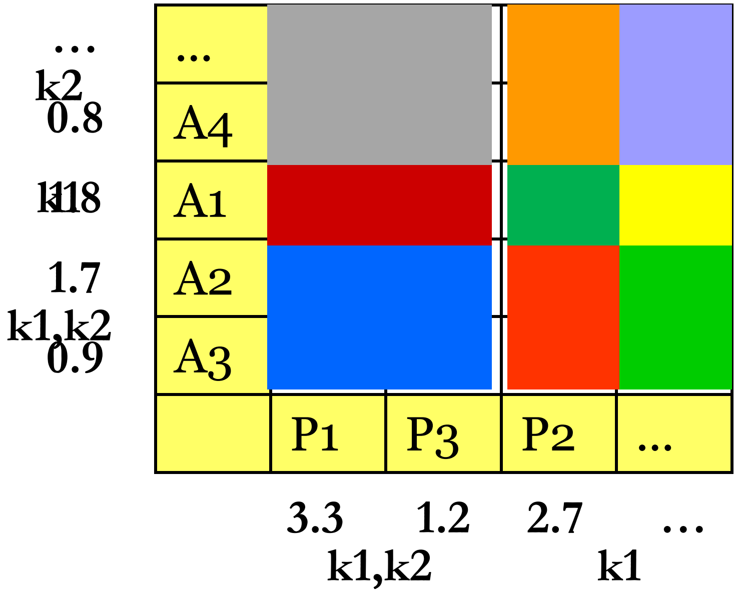




Block Pipeline

CN1 $P^Q - W - A^Q$

- Motivation
  - What if "score monotonicity" does not hold?
- Ideas
  - Find salient orderings s.t. we can derive a global score upper bounding function
  - Partition the search space into blocks s.t there is a tighter upper bounding function for each block



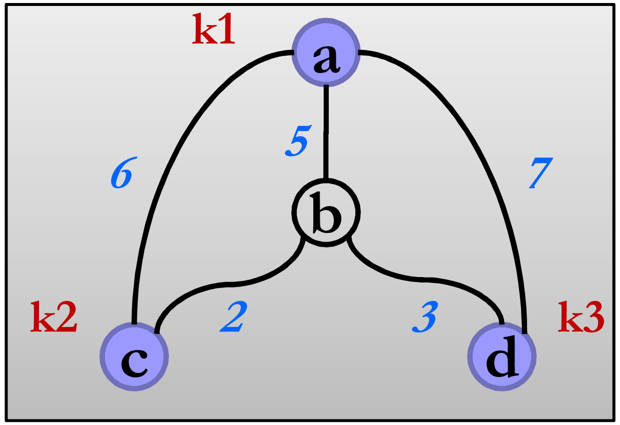
Using Semi-joins

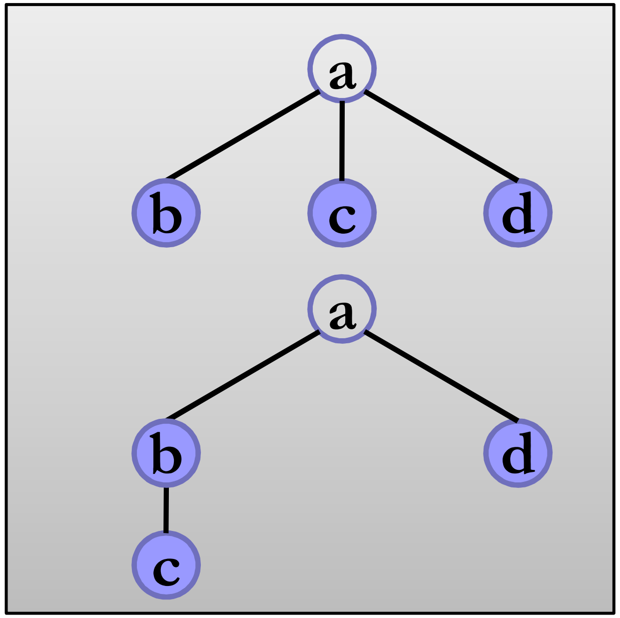
- Qin et al, "Keyword Search in Databases: The Power of RDBMS", ***SIGMOD 2009***
  - Tomorrow morning
  - Research Session 18: Keyword Search

Comparing Result Definitions

- Using schema?

	Schema-based	Schema-free
RDBMS	CN	
Graph		(Group) Steiner Tree, Distinct root semantics, Subgraph
XML	XSEarch, Entities, …	LCA and its variants





- Differences between def's
  - Bias
  - Computational complexity
  - Redundancy

Summary of Result Definition and Algorithms

- We have discussed result definition and query processing on three data models
  - Trees
  - Graphs
  - Nested Graphs
- The basis of query result is minimum Group Steiner tree, and later other variants (suitable in different data models)

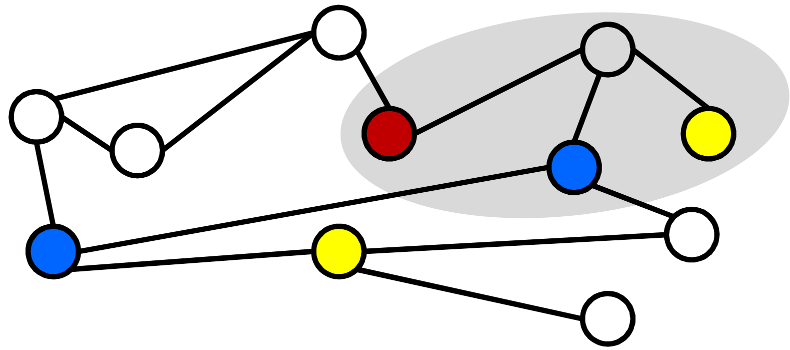
Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Search Distributed Databases
- Future Research Directions

Ranking Schemes

- Ranking is important for keyword search
  - On the Web
  - On databases
- Illustrate existing ranking schemes
  - Simple → IR-based + other factors considered

Proximity /1



- Total proximity
  - Group Steiner tree
- Proximity to root/center
  - Distinct root semantics

Proximity /2

- Proximity between keyword nodes
  - EASE:

$$sim(k_i, k_j) = \frac{1}{|C_{k_i} \cup C_{k_j}|} \cdot \sum_{\substack{n_i \in C_{k_i} \\ n_j \in C_{k_j}}} \sum_{n_i \leftrightsquigarrow n_j} \frac{1}{(|n_i \leftrightsquigarrow n_j| + 1)^2}$$

  - XRank: $p(n, k1, k2, \ldots, k_l) = |w|$
    - w is the smallest text window in n that contains all search keywords

Assigning Node Weights /1

- Based on graph structure
  - BANKS
    - Nodes: $\text{in-degree}_*(n)$
    - Edges : $\min(s(R(u), R(v)), \text{in-degree}_v(u) s(R(u), R(v)))$
  - PageRank-like methods
    - XRank [Guo et al, SIGMOD03]
    - ObjectRank [Balmin et al, VLDB04] : considers both Global ObjectRank and Keyword-specific ObjectRank

Assigning Node Weights /2

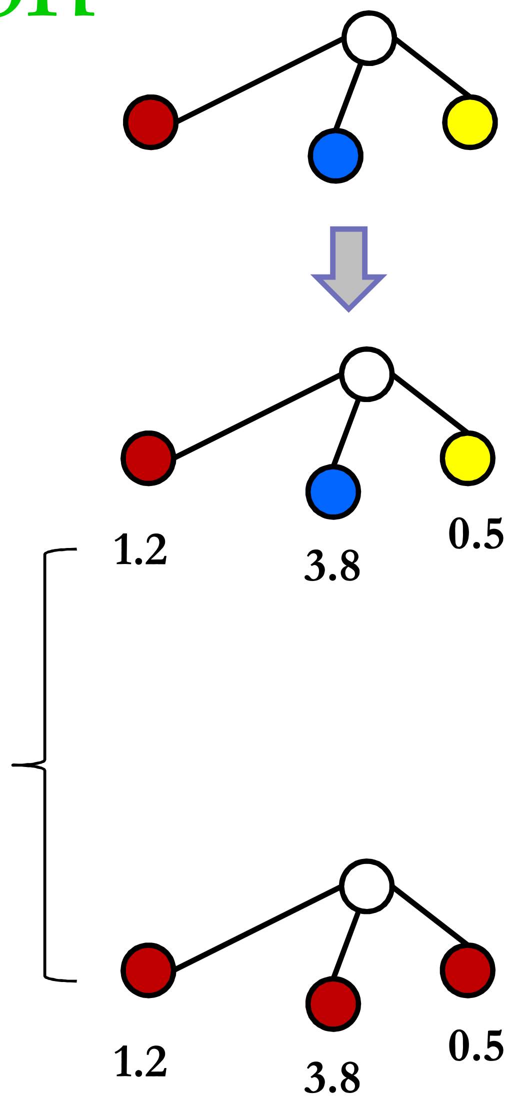
- TF*IDF based: $Score(n,Q) = \sum_{w \in Q \cap n} \frac{1+\ln(1+\ln(tf))}{(1-s)+s \cdot dl / avdl} \cdot \ln \frac{N+1}{df}$
  - Discover/EASE
  - [Liu et al, SIGMOD06]
    - $ndl = ((1-s)+s \cdot dl/avdl) \cdot (1+\ln avdl)$
      $nidf^g = \ln \frac{N^g}{df^g+1}$
  - SPARK
    - but *not* at the node level

Score Aggregate Function

- Combine s($node_i$) into a final score for ranking
  - BANKS: agg(edge) * agg(node)$^{\lambda}$
  - DISCOVER: $\sum_n$ s(n) / size_normalization
  - [Liu et al, SIGMOD06]:

$$\max S_i \cdot \left(1 + \ln\left(1 + \ln \frac{\sum S_i}{\max S_i}\right)\right)$$

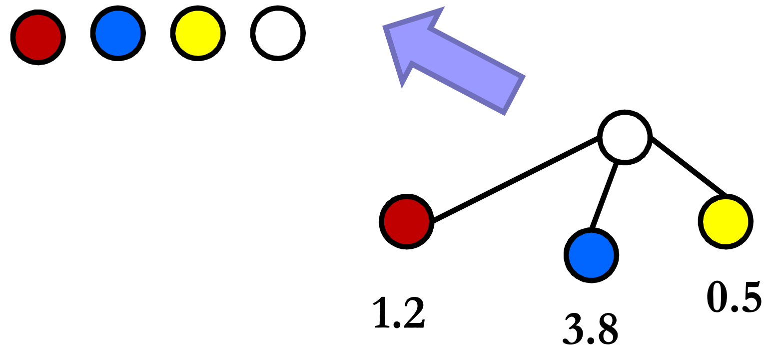
same score?

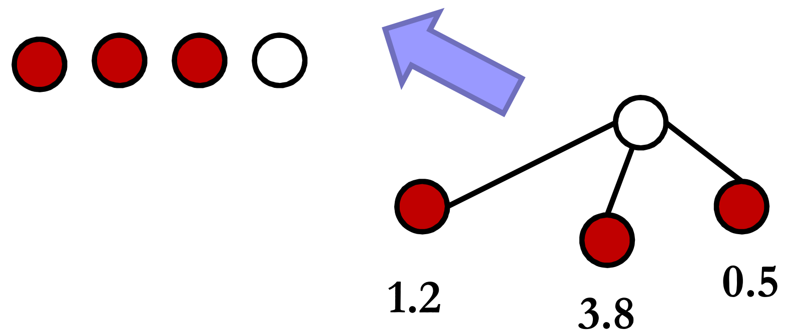


- Problem
  - Raw tf values are not well attenuated

Holistic Ranking

- SPARK
  - Each results in a CN is deemed as a virtual document
  - Calculate tf and idf on the virtual document level



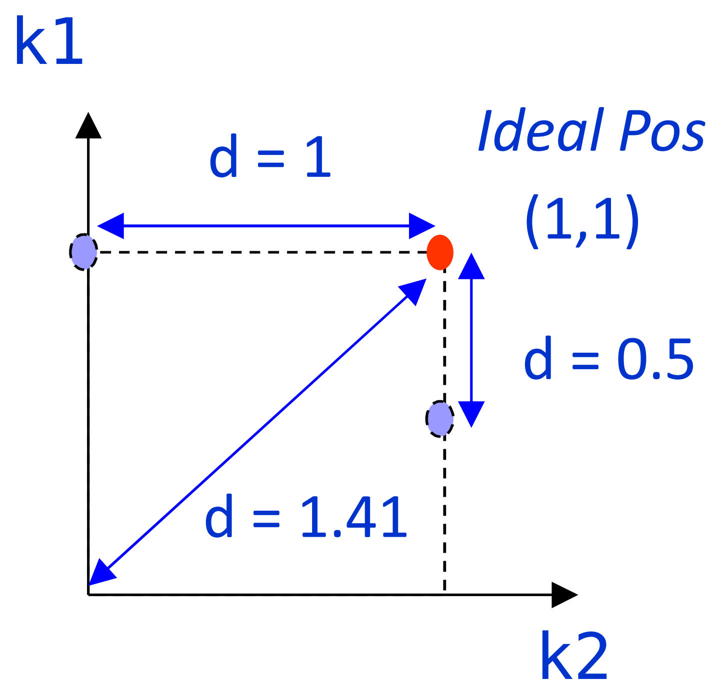


CN Scores

- Prefer small results
  - Discover 2
    - $size(CN)$
  - [Liu et al, SIGMOD06]
    - $(1-s)+s\cdot(size(T)/avgsize)$
  - SPARK
    - $(1+s_1-s_1|CN|)\cdot(1+s_2-s_2|CN^{nf}|)$
  - Prune CNs
    - By experts, query log, materialized views
    - Constraints: Précis, Interconnection semantics

Completeness Factor

- SPARK
  - $$1 - \left(\frac{\sum_{i=1}^{m} (1 - T.i)^p}{m} \right)^{\frac{1}{p}}$$
  - Tune between AND- and OR- semantics
  - Based on *Extended Boolean Model:* Measure Lp distance to the idea position
- SUITS: $$\left(\frac{T.uniqKeywords}{|Q|} \right)^p$$



L_2 distance

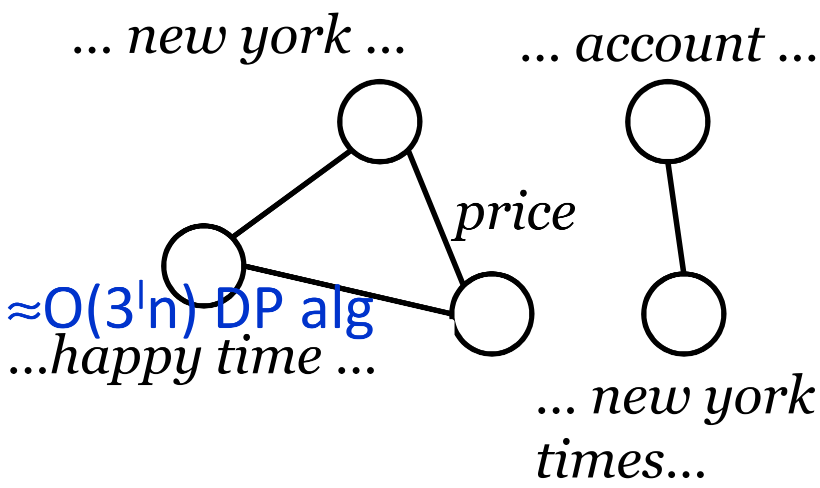
Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Search Distributed Databases
- Future Research Directions

Query Cleaning [Pu et al, VLDB08]

new york time price

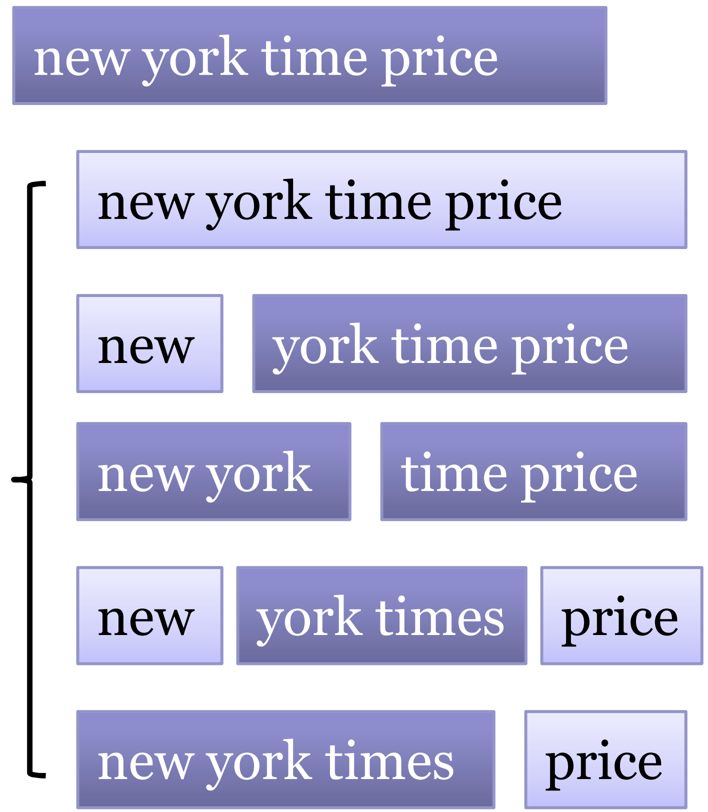
- Motivations
  - Query may contain typos
  - Query may contain phrases
  - Speed up query processing
- Input
  - A keyword query
  - Database
- Output
  - Corrected and segmented query



new york **times** | price

Cleaning Algorithm

- Cleaning Algorithm
  - Expand each token into possible variants and construct a candidate space
  - Find an optimal segmentation that maximizes a segmentation score (error-aware)
    - A dynamic programming algorithm for the static case; also incremental version of the DP algorithm



Also relevant: Query autocompletion [Li et al, SIGMOD09, Chaudhuri et al, SIGMOD09]

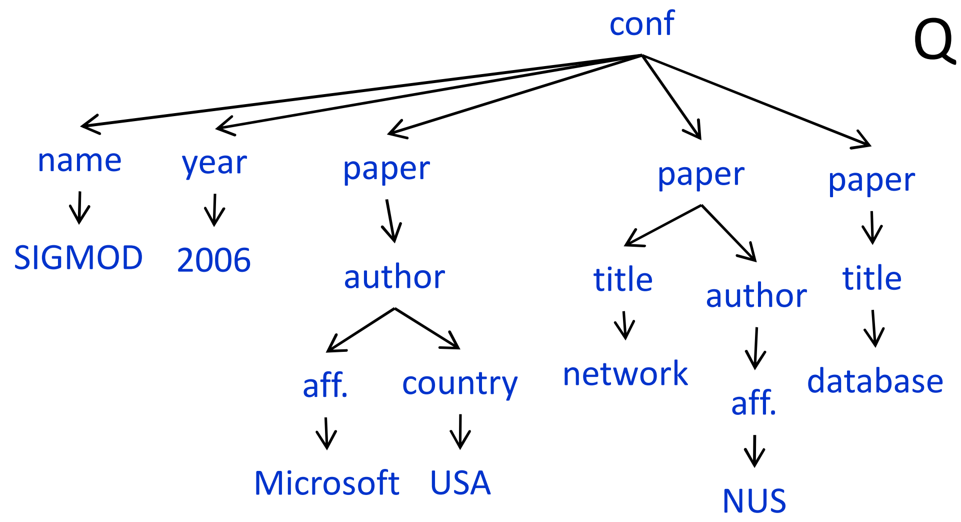
Roadmap

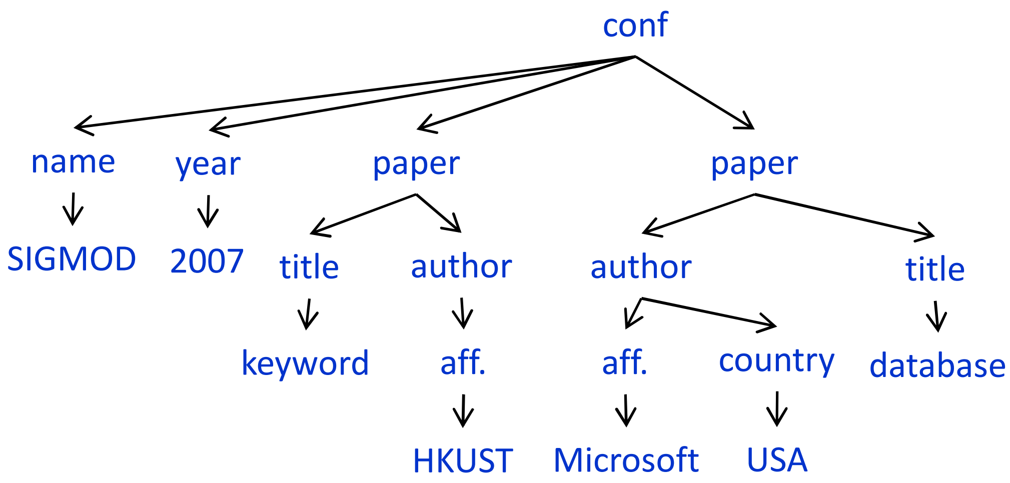
- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
  - Result Snippets
  - Mining Interesting Terms
  - Table Analysis
  - Result Evaluation
- Search Distributed Databases
- Future Research Directions

Result Analysis / Evaluation

- Result Snippets
  - Complement ranking schemes and help user pick relevant results quickly.
- Mining Interesting Terms
  - Help user formulate new queries.
- Table Analysis
  - Finding tuple clusters that are relevant to a keyword query.
- Result Evaluation
  - A useful guide for users to pick the most desirable search engine.

Result Snippets on XML [Huang et al. SIGMOD 08]





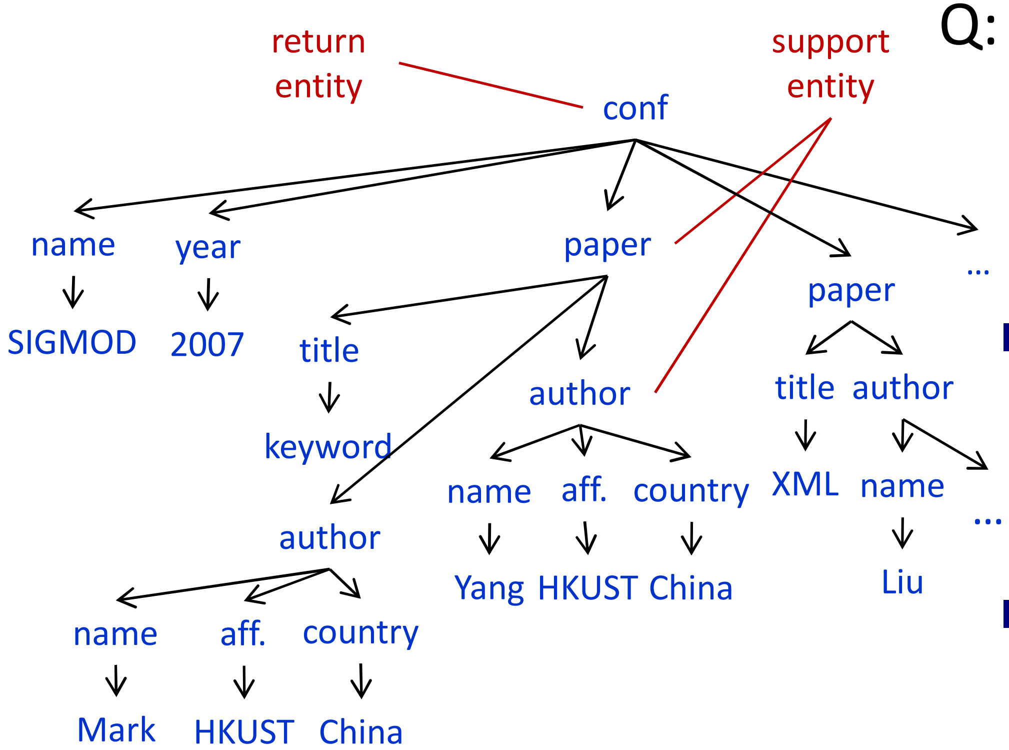
Q: "Sigmod, conf"

- From the snippets, we know
  - The two results are about "SIGMOD 06" and "SIGMOD 07".
  - Feature different hot topics and different institution / countries that have significant contribution.

- What are good snippets?

- How to generate them?

Distinguishable Snippets [Huang et al. SIGMOD 08]



Q: "Sigmod, conf"

What is the key of an XML search result?

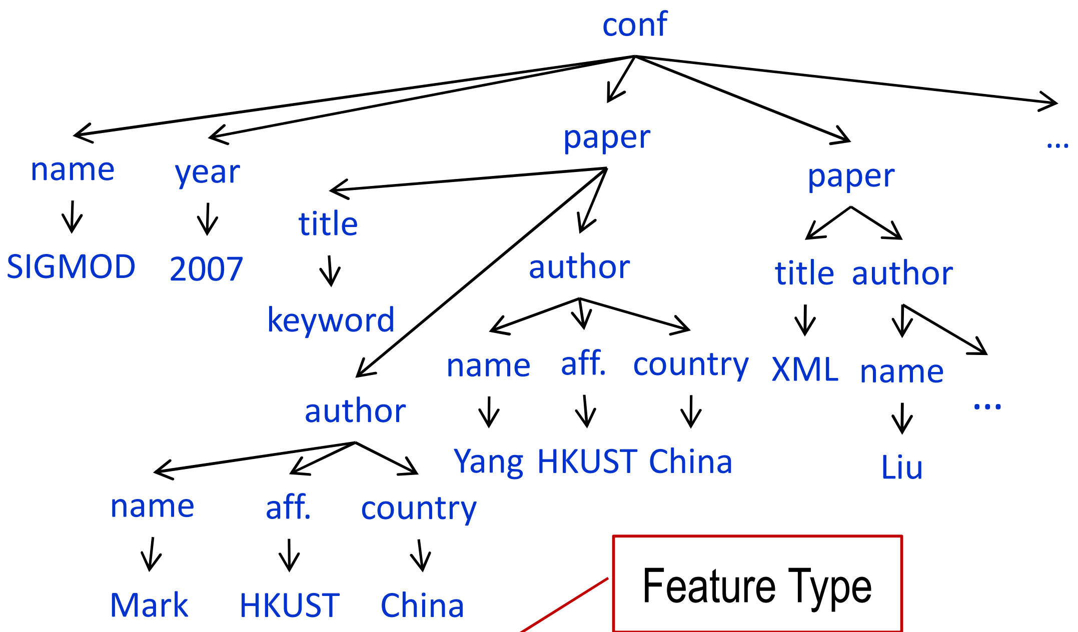
- Two types of entities:
  - Return entities
  - Support entities
- Key of a query result = keys of return entities

IList: a ranked list of information items to be included in snippets

Adding keywords, and the key of the query result to IList.

IList: **SIGMOD**, conf, **2007**

Representative Snippets [Huang et al. SIGMOD 08]



statistics

Author: country: USA: 84

Author: country: China: 17

Author: country: Singapore: 7

Paper: title: database: 21

Paper: title: keyword: 6

Paper: title: ranking: 3

Author: aff.: Microsoft: 35

Author: aff.: HKUST: 9

Feature Type

- Feature: (entity, attribute, value)
  - e.g., (paper, title, XML)
- Dominant features: features that have more occurrences than the other features of the same type.

Adding the dominant features of query result to IList

IList: SIGMOD, conf, 2007, **USA, Microsoft, database, keyword, HKUST**

Result Snippets on XML [Huang et al. SIGMOD 08]

- Small snippet:
  - Goal: selecting data instances, such that as many items in IList can be included in the snippet as possible with a size bound.
  - NP-hard.
  - Heuristic algorithms are proposed .

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
  - Result Snippets
  - Mining Interesting Terms
  - Table Analysis
  - Result Evaluation
- Search Distributed Databases
- Future Research Directions

Mining Interesting Terms [Tao et al. EDBT 09, Koutrika et al. EDBT 09]

- Snippets: generated for each individual result to help users choose most relevant ones.

- Mining Interesting Terms: returning interesting non-keyword terms in all query results, to help user better understand the results and issue new queries.
  - For query "art" on a course database, it is helpful to return the interesting words that are related to "art".
    - E.g., "Performance", "Renaissance", "Byzantine"

Data Cloud [Koutrika et al. EDBT 09]

- Input: Query and results
- Output: Top-k ranked non-keyword terms in the results.
- Terms in results are ranked by several factors
  - Term frequency
  - Inverse Document Frequency
  - Rank of the result in which a term appears

Frequent Co-occurring Terms[Tao et al. EDBT 09]

- Can we avoid generating all results first?
- Input: Query
- Output: Top-k ranked non-keyword terms in the results.
- Capable of computing top-k terms efficiently without even generating results.
- Terms in results are ranked by frequency.
  - Tradeoff of quality and efficiency.

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
  - Result Snippets
  - Mining Interesting Terms
  - Table Analysis
  - Result Evaluation
- Search Distributed Databases
- Future Research Directions

Table Analysis[Zhou et al. EDBT 09]

- In some application scenarios, a user may be interested in a group of tuples jointly matching a set of query keywords.

- Given a keyword query with a set of specified attributes,
  - Cluster tuples based on (subsets) of specified attributes so that each cluster has all keywords covered
  - Output results by clusters, along with the shared specified attribute values

Table Analysis [Zhou et al. EDBT 09]

- Input:
  - Keywords: "pool, motorcycle, American food"
  - Interesting attributes specified by the user: month state
- Goal: cluster tuples so that each cluster has the same value of month and/or state and contains query keywords
- Output

	Month	State	City	Event	Description
December Texas	Dec	TX	Houston	US Open **Pool**	Best of 19, ranking
	Dec	TX	Dallas	Cowboy's dream run	**Motorcycle**, beer
	Dec	TX	Austin	SPAM Museum party	Classical **American food**
* Michigan	Oct	MI	Detroit	**Motorcycle** Rallies	Tournament, round robin
	Oct	MI	Flint	Michigan **Pool** Exhibition	Non-ranking, 2 days
	Sep	MI	Lansing	**American Food** history	The best food from USA

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
  - Result Snippets
  - Mining Interesting Terms
  - Table Analysis
  - Result Evaluation: Empirical vs Formal
- Search Distributed Databases
- Future Research Directions

INEX - INitiative for the Evaluation of XML Retrieval

- Benchmarks for DB: TPC, for IR: TREC
- A large-scale campaign for the evaluation of document-oriented XML retrieval systems.
  - Document oriented XML

```
< article >
  < title > Structured Document Retrieval < /title >
  < author >
    < firstAuthor > Tom < /firstAuthor >
    < secondAuthor > John < /secondAuthor >
  < /author >
  < chapter >
    < title > Introduction to Xpath < /title >
    < paragraph > ... < /paragraph >
   ...
  < /chapter >
   ...
< /article >
```

- Search quality is evaluated by large-scale user studies.

  http://inex.is.informatik.uni-duisburg.de/

Axiomatic Framework

- Formalize broad intuitions as a collection of simple axioms and evaluate strategies based on the axioms.
- It has been successful in many areas, e.g. mathematical economics, clustering, location theory, collaborative filtering, etc

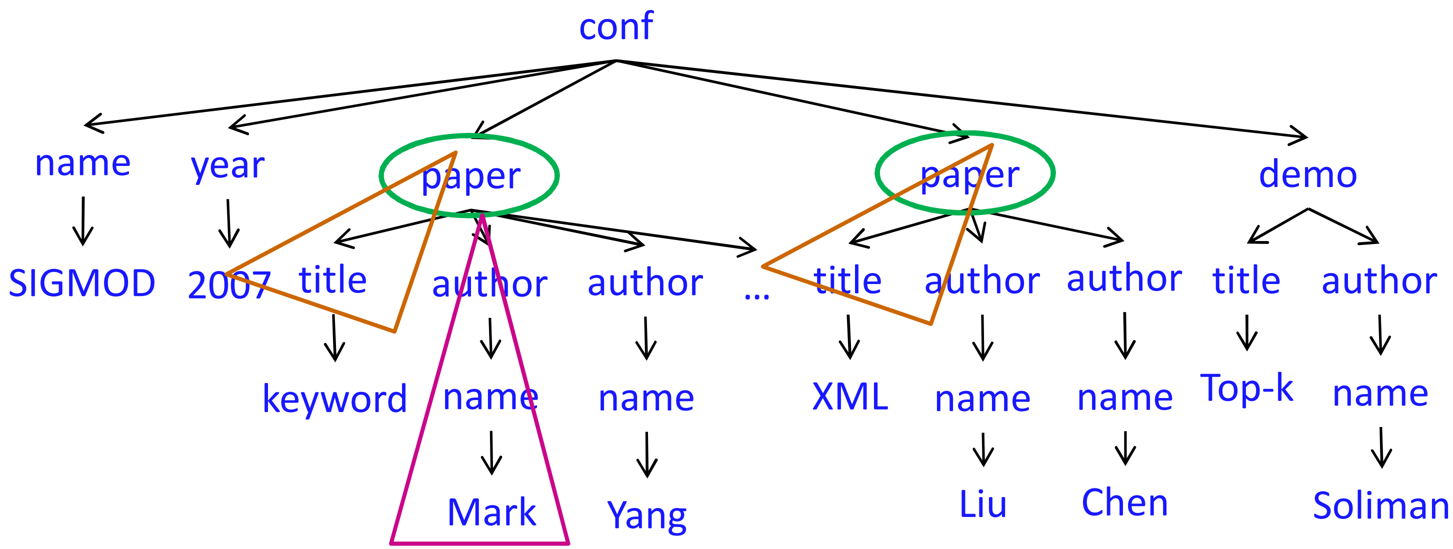
Axioms [Liu et al. VLDB 08]

Axioms for XML keyword search have been proposed for identifying relevant keyword matches

- Assuming "AND" semantics
- Some abnormal behaviors can be clearly observed when examining results of two similar queries or one query on two similar documents produced by the same search engine.
- Four axioms
  - Data Monotonicity
  - Query Monotonicity
  - Data Consistency
  - Query Consistency

Example: Query Monotonicity / Consistency

Q2: “paper, title,” ***Mark***”

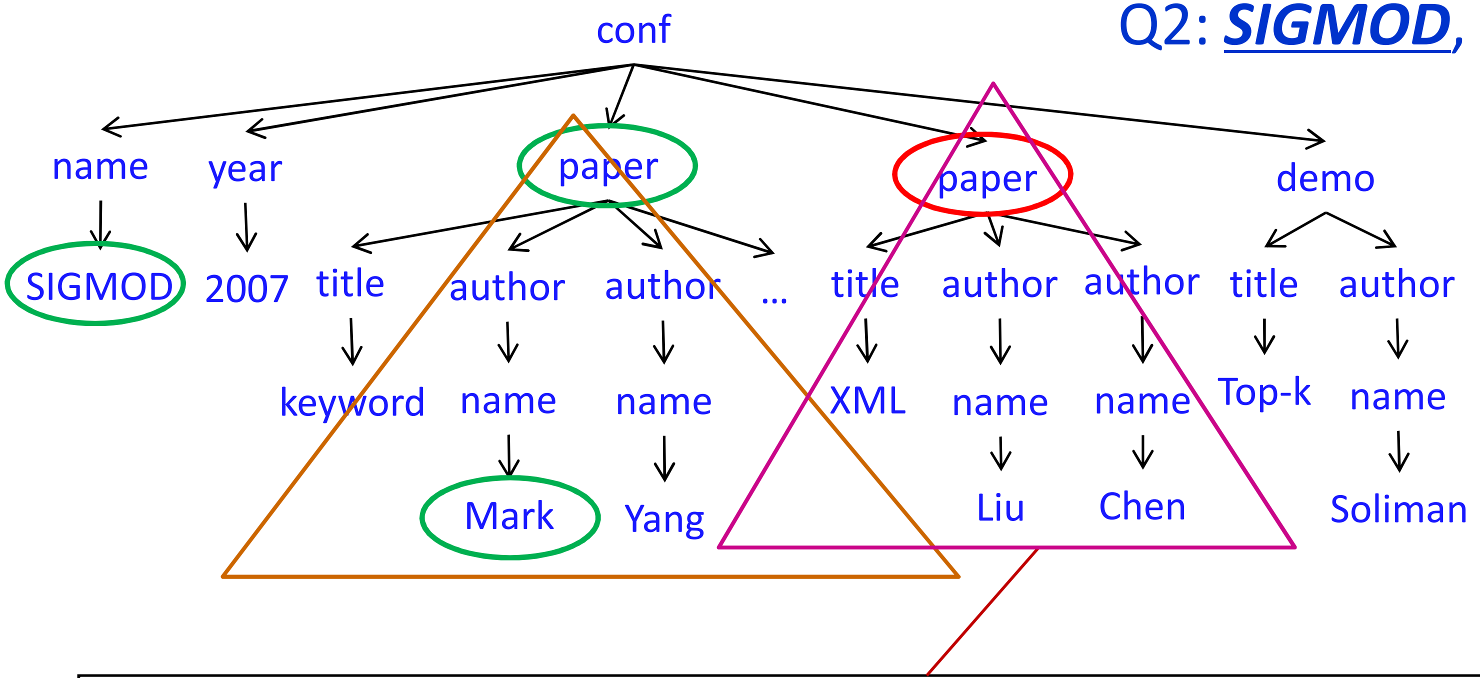


Query Monotonicity: the # of query results does not increase after adding a query keyword.

Query Consistency: the new result subtree contains the new query keyword.

Example: Violation of Query Consistency

Q1: paper, Mark
Q2: ***SIGMOD***, paper, Mark

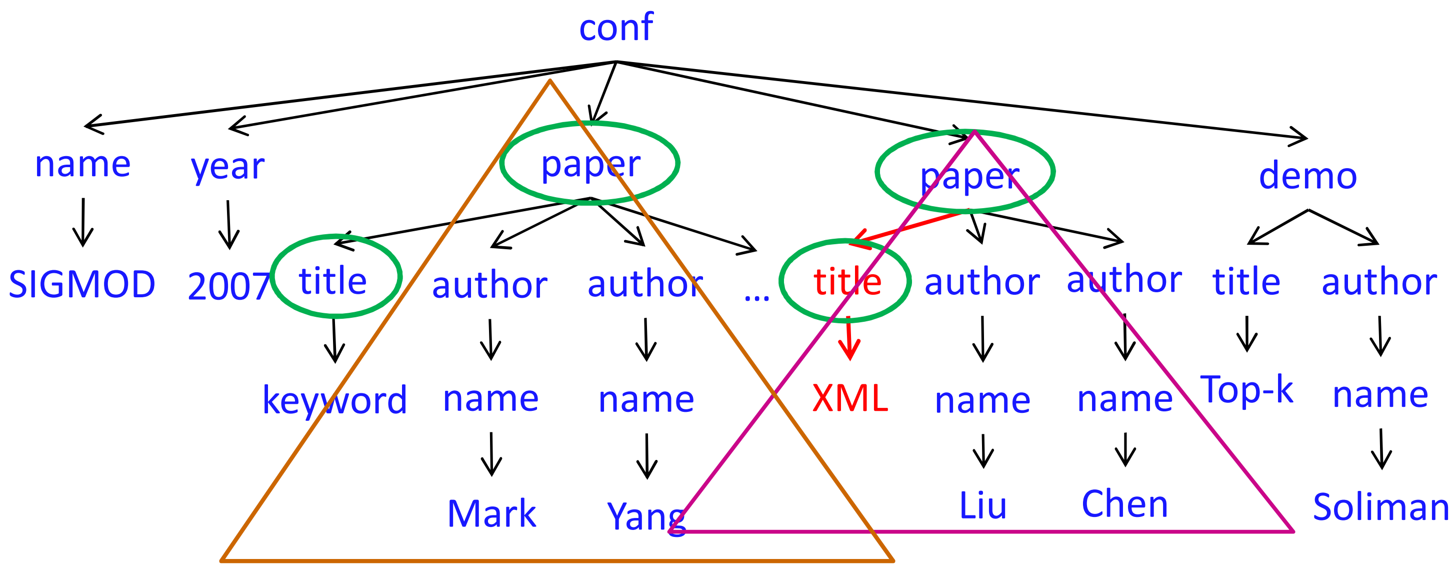


An XML keyword search engine that considers this subtree as relevant for the new query ***violates query consistency*** .

Query Consistency: the new result subtree contains the new query keyword.

Example: Data Consistency / Monotonicity

“paper, title”

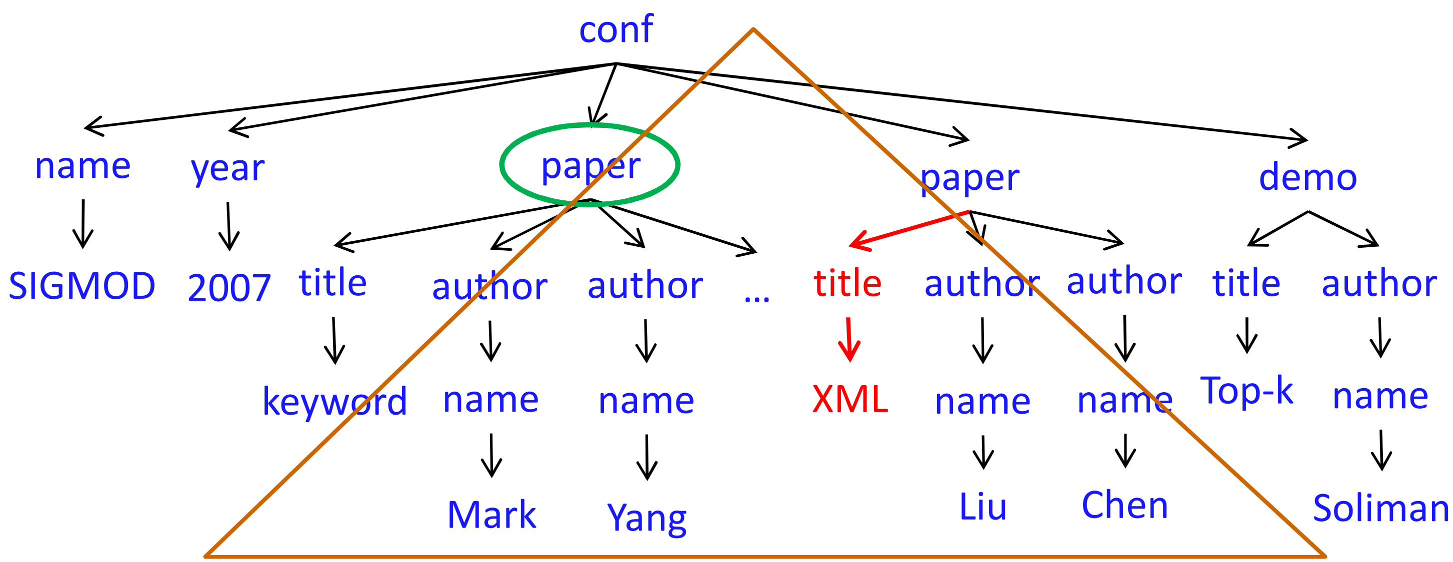


Data Monotonicity: the # of query results doesn’t decrease after inserting a new data node.

Data Consistency: each new result subtree contains the new data node.

Example: Violation of Data Monotonicity

"SIGMOD, Mark, Liu, title"



An XML keyword search engine that outputs an empty result on the updated data ***violates data monotonicity.***

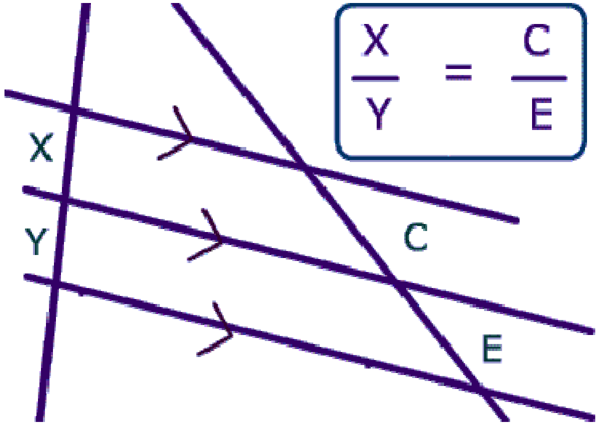
Data Monotonicity: the # of query results doesn't decrease after inserting a new data node.

This set of axioms is non-trivial, but indeed satisfiable [Liu et al VLDB 08]

Empirical vs. Formal Evaluation

- Benchmark
  - The ultimate evaluation
  - Costly – needs large data sets, query sets, and users.

- Axioms
  - Cost-effective
  - Theoretical and objective
  - Guiding the design
  - Complement empirical studies



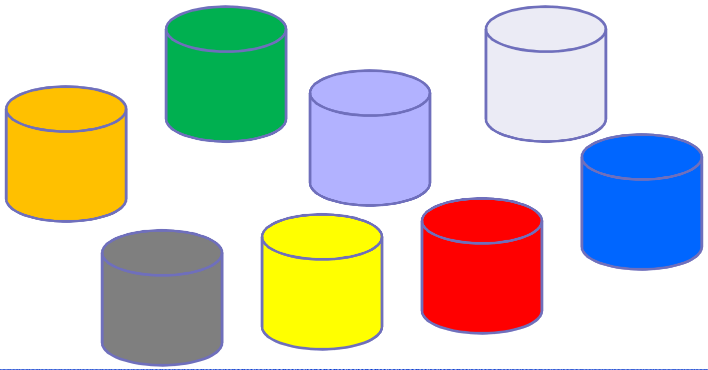
Roadmap

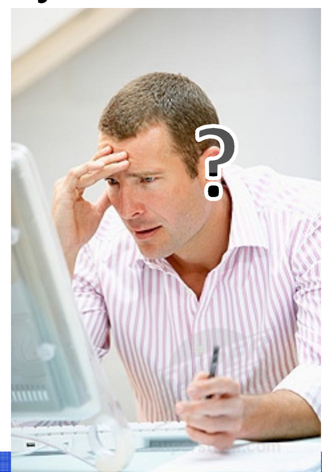
- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
- Searching Distributed Databases
- Future Research Directions

Database Selection [Yu et al. SIGMOD 07]

- Input:
  - a query
  - multiple databases, each of which that can provide results to the query.
- Output: names of databases that are likely to generate top-K results

- Intuition: Pushing top-K query processing at database level instead of issuing the query to all databases, only issue it to high-quality databases





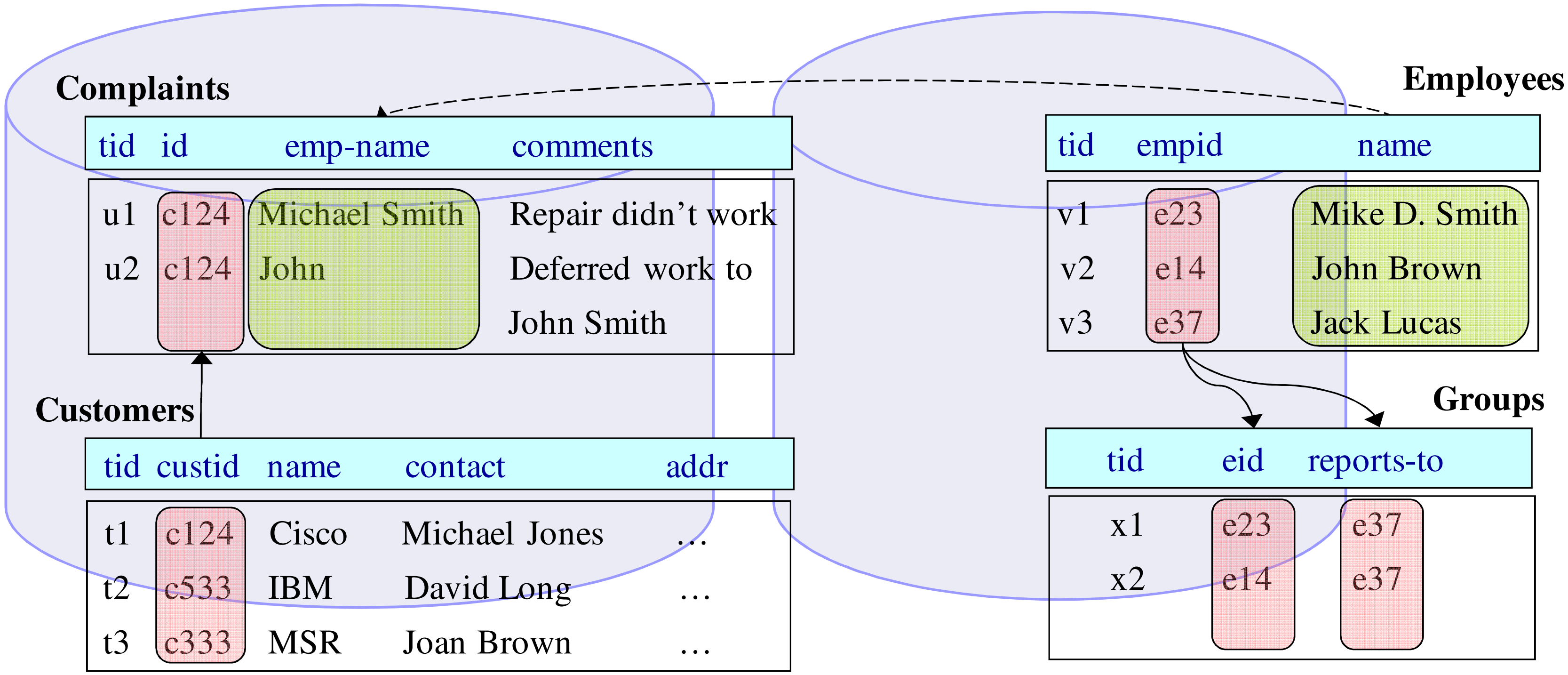
Database Selection [Yu et al. SIGMOD 07]

- Goal: Database score = sum score of top k results on this database
  - Impossible to precisely evaluate w/o generating query results.
- Approximation: database score = sum of score of top k connections of every pair of keywords
  - Score of a connection = length of path
- Algorithms are proposed to compute the relationship matrix between every two keywords in a database.

Kite [Sayyadan et al. ICDE 07]

- Input:
  - A query
  - Multiple databases, each of which may NOT provide results to the query
- Output: Results that contain all query keywords composed from multi-databases.
- Intuition: Pushing keyword search from the level of multi-relations to multi-databases, where the relationships among databases can be discovered.

Kite [Sayyadan et al. ICDE 07]



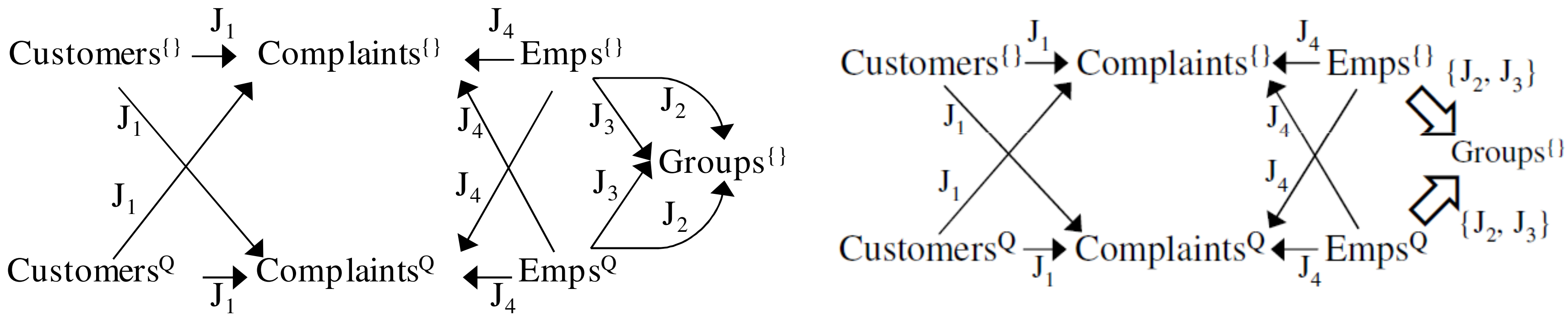
Query: **[Cisco Jack Lucas]**

Challenges:

- Automatically inferring meaningful joins across databases
- Supporting approximate/similarity joins

Kite [Sayyadan et al. ICDE 07]

- Challenge: tables in multiple databases usually involve a large number of joins, making the number of CNs huge.
  - Condense multiple relationships among two tables as one.



- Lazily expand condensed CN when they are promising to provide top k results

Roadmap

- Motivation and Challenges
- Query Result Definition and Algorithms
- Ranking
- Query Preprocessing
- Result Analysis and Evaluation
  - Result Snippets
  - Mining Interesting Terms
  - Table Analysis
  - Result Evaluation: Empirical vs Formal
- Search Distributed Databases
- Future Research Directions

Expressive Power vs. Complexity

- *Where is the right balance and how to achieve it?*
- Related work
  - Supporting aggregate queries: KDAP [Wu et al, SIGMOD07], SQAK [Tata and Lohman, SIGMOD08]
  - Forms [Jayapandian and Jagadish, VLDB08], [Chu et al, SIGMOD09]
  - Natural language queries [Li et al, SIGMOD07]
  - Formulate queries interactively: ExQueX [Kimelfeld et al, SIGMOD09]

Evaluation and Benchmarking

- *How to evaluate a system?*
- Related work
  - Pooling in IR
  - Benchmarking: INEX
  - Axiomatic approaches

Efficiency and Deployment

- *I want this keyword feature in my application/database. Where can I get it?*
- Related work
  - Algorithmic approaches to scale to large databases with complex schema
  - DB + IR, rank-aware query optimization

Search Quality Improvement

- *What can we learn from IR / Web Search?*
- Related work
  - (Pseudo-) Relevance feedback and query refinement: SUITS [Zhou et al, 2007]
  - Result post processing and presentation: eXtract [Huang et al, VLDB08], TreeCluster [Peng et al, 2006], Visualization [many eyes]
  - Ranking
  - Personalization

Diverse Data Models

- *How to accommodate & serve different data models?*
- Related work
  - Querying (and integrating) heteogenous data: [Talukdar et al, VLDB08], Wolfram Alpha, Google Squared.
  - Data Warehouses [Wu et al, SIGMOD07], Spatial Databases [De Felipe et al, ICDE08] [Zhang et al, ICDE 2009],Workflow [Shao et al, ICDE09]
  - INEX-related work
  - Querying extracted data
  - Graph data: bio-DB [Guo et al, ICDE07], RDB and Linked Data [Tran et al, ICDE09], NAGA [Kasneci et al, SIGMOD08]

Thank you!

Questions?

Reference /1

Agrawal, S., Chaudhuri, S., and Das, G. (2002). DBXplorer: A system for keyword-based search over relational databases. In ICDE, pages 5-16.

Al-Khalifa, S., Yu, C., and Jagadish, H. V. (2003). Querying structured text in an xml database. In SIGMOD Conference, pages 4-15.

Amer-Yahia, S. and Shanmugasundaram, J. (2005). XML full-text search: Challenges and opportunities. In VLDB, page 1368.

Bao, Z., Ling, T. W., Chen, B., and Lu, J. (2009). Effective xml keyword search with relevance oriented ranking. In ICDE, pages 517-528.

Bhalotia, G., Nakhe, C., Hulgeri, A., Chakrabarti, S., and Sudarshan, S. (2002). Keyword Searching and Browsing in Databases using BANKS. In ICDE, pages 431-440.

Chaudhuri, S., Kaushik, R. (2009) Extending autocompletion to tolerate errors. In SIGMOD, pages 707-718.

Cohen, S., Mamou, J., Kanza, Y., and Sagiv, Y. (2003). XSEarch: A semantic search engine for XML. In VLDB, pages 45-56.

Dalvi, B. B., Kshirsagar, M., and Sudarshan, S. (2008). Keyword search on external memory data graphs. PVLDB, 1(1):1189-1204.

Ding, B., Yu, J. X., Wang, S., Qin, L., Zhang, X., and Lin, X. (2007). Finding top-k min-cost connected trees in databases. In ICDE, pages 836-845.

Goldman, R., Shivakumar, N., Venkatasubramanian, S., and Garcia-Molina, H. (1998). Proximity search in databases. In VLDB, pages 26-37.

Golenberg, K., Kimelfeld, B., and Sagiv, Y. (2008). Keyword proximity search in complex data graphs. In SIGMOD, pages 927-940.

Reference /2

Guo, L., Shanmugasundaram, J., and Yona, G. (2007). Topology search over biological databases. In ICDE, pages 556-565.

Guo, L., Shao, F., Botev, C., and Shanmugasundaram, J. (2003). XRANK: Ranked keyword search over XML documents. In SIGMOD.

He, H., Wang, H., Yang, J., and Yu, P. S. (2007). BLINKS: Ranked keyword searches on graphs. In SIGMOD, pages 305-316.

Hristidis, V., Hwang, H., and Papakonstantinou, Y. (2008). Authority-based keyword search in databases. ACM Trans. Database Syst., 33(1):1-40.

Hristidis, V. and Papakonstantinou, Y. (2002). Discover: Keyword search in relational databases. In VLDB.

Hristidis, V., Papakonstantinou, Y., and Balmin, A. (2003). Keyword proximity search on xml graphs. In ICDE, pages 367-378.

Huang, Yu., Liu, Z. and Chen, Y. (2008). Query Biased Snippet Generation in XML Search. In *SIGMOD*.

Jagadish, H. V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A., and Yu, C. (2007). Making database systems usable. In SIGMOD, pages 13-24.

Jayapandian, M. and Jagadish, H. V. (2008). Automated creation of a forms-based database query interface. PVLDB, 1(1):695-709.

Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., and Karambelkar, H. (2005). Bidirectional expansion for keyword search on graph databases. In VLDB, pages 505-516.

Reference /3

Kimelfeld, B. and Sagiv, Y. (2006). Finding and approximating top-k answers in keyword proximity search. In PODS, pages 173-182.

Kong, L., Gilleron, R., and Lemay, A. (2009). Retrieving Meaningful Relaxed Tightest Fragments for XML Keyword Search. In *EDBT*.

Koutrika, G., Zadeh, Z.M., and Garcia-Molina, H. (2009). Data Clouds: Summarizing Keyword Search Results over Structured Data. In *EDBT*.

Li, G., Ooi, B. C., Feng, J., Wang, J., and Zhou, L. (2008). EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In SIGMOD.

Li, G., Ji, S., Li, C., Feng, J. (2009). Efficient type-ahead search on relational data: a TASTIER approach. In SIGMOD, pages 695-706.

Li, W.-S., Candan, K. S., Vu, Q., and Agrawal, D. (2001). Retrieving and organizing web pages by "information unit". In WWW, pages 230-244.

Li, Y., Chaudhuri, I., Yang, H., Singh, S., and Jagadish, H. V. (2007). Danalix: a domain-adaptive natural language interface for querying xml. In SIGMOD, pages 1165-1168.

Li, Y., Yu, C., and Jagadish, H. V. (2004). Schema-free XQuery. In VLDB.

Liu, B. and Jagadish, H. V. (2009). A spreadsheet algebra for a direct data manipulation query interface. In ICDE, pages 417-428.

Liu, F., Yu, C., Meng, W., and Chowdhury, A. (2006). Effective keyword search in relational databases. In SIGMOD, pages 563-574.

Reference /4

Liu, Z. and Chen, Y. (2007). Identifying Meaningful Return Information for XML Keyword Search. In *SIGMOD*.

Liu, Z. and Chen, Y. (2008). Reasoning and identifying relevant matches for xml keyword search. PVLDB, 1(1):921-932.

Liu, Z. and Chen, Y. (2008). Answering Keyword Queries on XML Using Materialized Views. In *ICDE (poster)*.

Luo, Y., Lin, X., Wang, W., and Zhou, X. (2007). SPARK: Top-k keyword query in relational databases. In SIGMOD, pages 115-126.

Nandi, A. and Jagadish, H. V. (2009). Qunits: queried units in database search. In CIDR. Pu, K. Q. and Yu, X. (2008). Keyword query cleaning. PVLDB, 1(1):909-920.

Qin, L., Yu, J. X., Chang, L., and Tao, Y. (2009). Querying communities in relational databases. In ICDE, pages 724-735.

Qin, L., Yu J. X., Chang, L. (2009) Keyword search in databases: the power of RDBMS. In SIGMOD, pages 681-694.

Sayyadian, M., LeKhac, H., Doan, A., and Gravano, L. (2007). Efficient keyword search across heterogeneous relational databases. In ICDE, pages 346-355.

Shao, Q., Sun, P., and Chen, Y. 2009. WISE: A Workflow Information Search Engine. In *ICDE* (demo).

Simitsis, A., Koutrika, G., and Ioannidis, Y. (2008). Preis: from unstructured keywords as queries to structured databases as answers. The VLDB Journal, 17(1):117-149.

Su, Q. and Widom, J. (2005). Indexing relational database content offline for efficient keyword-based search. In IDEAS, pages 297-306.

Sun, C., Chan, C.-Y., and Goenka, A. (2007). Multiway SLCA-based keyword search in XML data. In WWW.

Reference /5

Tao, Y., and Yu, J.X. (2009). Finding Frequent Co-occurring Terms in Relational Keyword Search. In *EDBT*.

Talukdar, P. P., Jacob, M., Mehmood, M. S., Crammer, K., Ives, Z. G., Pereira, F., and Guha, S. (2008). Learning to create data-integrating queries. PVLDB, 1(1):785-796.

Tata, S. and Lohman, G. M. (2008). SQAK: doing more with keywords. In SIGMOD, pages 889-902.

Vagelis Hristidis, L. G. and Papakonstantinou, Y. (2003). Efficient ir-style keyword search over relational databases. In VLDB.

Wu, P., Sismanis, Y., and Reinwald, B. (2007). Towards keyword-driven analytical processing. In SIGMOD, pages 617-628.

Xu, Y. and Papakonstantinou, Y. (2005). Efficient keyword search for smallest LCAs in XML databases. In SIGMOD.

Xu, Y. and Papakonstantinou, Y. (2008). Efficient lca based keyword search in xml data. In EDBT '08: Proceedings of the 11th international conference on Extending database technology, pages 535-546, New York, NY, USA. ACM.

Yu, B., Li, G., Sollins, K., Tung, A.T.K. (2007). Effective Keyword-based Selection of Relational Databases. In *SIGMOD*.

Zhou, B., and Pei, J. (2009). Answering Aggregate Keyword Queries on Relational Databases Using Minimal Group-bys. In *EDBT*.

Zhou, X., Zenz, G., Demidova, E., and Nejdl, W. (2007). SUITS: Constructing structured data from keywords. Technical report, L3S Research Center.