Traditional Data Access Methods

- 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!

The usability of DB is severely limited unless easier ways to access databases are developed [Jagadish, SIGMOD 07].
Supporting Keyword Search on DB – Advantages

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

Seltzer is a developer of Berkeley DB.

Seltzer, Berkeley
Is Seltzer a student at UC Berkeley?

Wow.
Supporting Keyword Search on DB – Advantages

- 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)

![Graph showing the number of publications from 2002 to 2009.](image-url)
XSeek Demo

http://xseek.asu.edu/

**Store: Brooks Brothers Clothing**

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

**Store: L.L. Bean**

```
<Store>
  <name>L.L. Bean</name>
  <city>Houston</city>
  <state>Texas</state>
  <merchandise>
    <clothes>
      <category>men</category>
    </clothes>
    <category>footwear</category>
    <clothes>
      <category>outwear</category>
    </clothes>
    <clothes>
      <category>pants</category>
    </clothes>
  </merchandise>
</Store>
```
After seeing the query results, the user identifies that ‘david’ should be ‘david J. Dewitt’.
The user is only interested in finding all join papers written by David J. Dewitt (i.e., not the 4th result).
<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: Multiprocessor Hash-Based <code>join</code> Algorithms. InProceedingId: 125561</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: Partition Based Spatial-Merge <code>join</code>. InProceedingId: 197661</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: Pointer-Based <code>join</code> Techniques for Object-Oriented Databases. InProceedingId: 111229</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: Tradeoffs in Processing Complex <code>join</code> Queries via Hashing in Multiprocessor Database Machines. InProceedingId: 127858</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: Clone <code>join</code> and shadow <code>join</code>: two parallel spatial <code>join</code> algorithms. InProceedingId: 30064</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InProceeding</th>
<th>Title: A Performance Evaluation of Four Parallel <code>join</code> Algorithms in a Shared-Nothing Multiprocessor Environment. InProceedingId: 199103</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelationPersonInProceeding</td>
<td>Person: Name: David J. DeWitt PersonId: 35293</td>
</tr>
</tbody>
</table>
Overview of This Tutorial

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

<table>
<thead>
<tr>
<th>Pre-processing</th>
<th>Query Processing</th>
<th>Post-processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Selection</td>
<td>Result Generation</td>
<td>Result Snippets</td>
</tr>
<tr>
<td>Query Cleaning</td>
<td>Ranking</td>
<td>Result Clustering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Result Analysis/Evaluation</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th></th>
<th>DB</th>
<th>XML</th>
<th>Web</th>
<th>Nested Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>tuple</td>
<td>element/attribute</td>
<td>webpage</td>
<td>object</td>
</tr>
<tr>
<td>Edge</td>
<td>foreign key</td>
<td>parent/child</td>
<td>hyperlink</td>
<td>expansion/dataflow</td>
</tr>
</tbody>
</table>

- **Query** $\mathbf{Q} = <k_1, k_2, \ldots, k_l>$

- **Output:** “closely related” nodes that are “collectively relevant” to the query
  - The smallest trees covering all keywords.
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.
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 $M_{12}$
**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.

```
conf
  ├── name
  │    └── SIGMOD
  │        └── 2007
  │            ├── title
  │            │    └── keyword
  │            │        └── Mark
  │            └── year
  │                └── name
  │                    └── Yang
  └── paper
      ├── title
      │    └── XML
      │        └── name
      │            └── Chen
      └── author
          └── name
              └── Soliman

conf
  ├── name
  │    └── demo
  │        └── title
  │            └── name
  │                └── Liu
  └── paper
      ├── author
      │    └── name
      │        └── Chen
      └── author
          └── name
              └── Soliman
```
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)

“SIGMOD, paper, Mark”
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: $O(S_{\text{min}}kd\log S_{\text{max}})$
    - 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
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.
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

- 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 \( Q = <k_1, k_2, \ldots, k_l> \)

- **Outputs** are "closely related" objects that are "collectively relevant" to the query
  - Graph
  - RDBMS

- **Scoring/ranking methods**
  - To be covered in Sec 3.
Evolution of Query Result Definitions

- **Group Steiner Tree (GST)**
  - Dynamic Programming or Mixed Integer Programming
  - Lawler’s framework

- **Approximate Group Steiner Tree**
  - BANKS 1/2/3, BLINKS
  - STAR [Kasneci et al, ICDE09]
  - Distinct root semantics

- **Subgraph-based**
  - Community
  - EASE
Closely Related Nodes

- **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

- **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**
  - $\Box$ NP-hard
  - $\checkmark$ Tractable for fixed l
Dynamic Programming for GST-1 [Ding et al, ICDE07]

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

\[
T(a, 123) = \min(T_g(a, 123), T_m(a, 123))
\]

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

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

- \( a (c, d): 13 \)
- \( a(b(c, d)): 10 \)
DP for GST-k

- Keep running GST-1 until k results are obtained ➔ approximate answer

- Complexities (GST-1, GST-k)
  - Time: $O(3^l n + 2^l((l+\log n)n + m))$ → $O(n \log n + m)$
  - Space: $O(2^l n)$ → $O(n)$

If $l = O(1)$
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\} = \text{edges}(T) \setminus I
    - Generate k partitions (E' = e_{k-i}, I' = \{e_1, \ldots, e_i\}) and Queue.enqueue(CST(G), I', E')
Illustration

Top-2 (global)

Top-1 (global)

Top-1 (local): 4

P1

Top-1 (local): 5

P2

Top-1 (local): 4

P3

Sol

has e1?

yes

has e2?

yes

has e3?

yes

Top-1

no

no

P3

P1

P2
**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-ks using STs
  - a \textit{backward} expansion search algorithm
  - Run multiple Dijkstra’s single-source-shortest-path algorithms iteratively until k answers are found \(\Rightarrow\) equi-distance expansion
- No guarantee on the quality of its top-k results
Example

- **While (!quit)**
  - Execute the iterator, $I_j$, whose output node, $v_j$, has the least "distance" from its source
  - $v_j$.reachable_from[label($I_j$)] ∪ = source($I_j$)
  - If $v$ is reachable from at least one source in every $S_i$
    - OutputHeap << GenResult($v_j$)  // result = $\Pi$(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 \( \text{cost}(T_r) = \sum_i \text{cost}(r, \text{match}_i) \)
  - A tree \( \rightarrow \) 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
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 $r_i$
  - $\Box$ Cost($r_i$) = Cost($r_i$, k1) + Cost($r_i$, k2)
  - $\Box$ Keep only the top-k min cost roots
**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

$k_i$ is not known a priori

2 Choices:
1. Index node-node distance, or
2. Index node-keyword distance
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**
  - $d(x, y) = \min (d^*(x, y), d^*(x, A) + d^H(A, B) + d^*(B, y), \forall A, B \in H)$
Distinct root semantics

Foreach root candidates $r_i$

- $\text{Cost}(r_i) = \text{Cost}(r_i, k1) + \text{Cost}(r_i, k2)$
- Keep only the top-k min cost roots

(2) Index node-keyword distance

Use Fagin’s TA Alg.
SLINKS /2

- 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
    \[ \approx O(|V|^2) \text{ in practice} \]
- 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)

i.e., the set of core nodes
Community-finding Algorithms

- Nested loop
  - Enumerate core node combinations

- Bottom-up search
  - BANKS 2, BLINKS (using index)

- Top-down search
  - Proximity search (using index)

- Polynomial 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, *)
EASE /1 [Li et al, SIGMOD08]

- Redundancy affects
  - GST
  - Distinct root semantics
  - Community

- **Subgraphs** as results | r
EASE /2

- r-Radius graph (r-G) $\Rightarrow$ 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 $\Rightarrow$ (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

- Running example
  - Author(aid, name)
  - Paper(pid, title)
  - Writes(aid, pid)

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

What if “trio” is also a person’s name?

Candidate Network (CN)

A $\bowtie W \bowtie P_{xml}$
Why CNs?

- **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 $T_{\text{max}}$
  - Minimum set of join expressions to execute
- allow multiple occurrence of a relation as cmp’ed with DBXplorer [Agrawal et al, ICDE02]
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*
Generating CNs

- **Input**
  - non-free tuple sets

- **Output**
  - all valid CNs no larger than $T_{\text{max}}$

- **Method**
  - Breadth-first search + pruning

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A^Q$</td>
</tr>
<tr>
<td>2</td>
<td>$P^Q$</td>
</tr>
<tr>
<td>3</td>
<td>$A^Q - W$</td>
</tr>
<tr>
<td>4</td>
<td>$W - P^Q$</td>
</tr>
<tr>
<td>5</td>
<td>$A^Q - W$</td>
</tr>
<tr>
<td>6</td>
<td>$W - P^Q$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$A^Q - W - P^Q$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>$A^Q - W - P^Q$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>$W - P^Q$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A^Q - W - P^Q$</td>
</tr>
</tbody>
</table>

| Not minimal |
| Non-promising |
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

   \[\text{Push top-k constraints inside!}\]
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;

<table>
<thead>
<tr>
<th>Result (CN1)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-W1-A2</td>
<td>3.0</td>
</tr>
<tr>
<td>P2-W5-A3</td>
<td>2.3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result (CN2)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2-W2-A1-W3-P7</td>
<td>1.0</td>
</tr>
<tr>
<td>P2-W9-A5-W6-P8</td>
<td>0.6</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
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.

\[
\text{score}(P_1 - \ldots - P_1) \geq \text{score}(P_x - \ldots - P_y) \quad (x>1, y>1)
\]

<table>
<thead>
<tr>
<th>CN1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-W1-A2</td>
<td>3.0</td>
</tr>
<tr>
<td>P2-W5-A3</td>
<td>2.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CN2</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-W?-A?-W?-P1</td>
<td>1.5</td>
</tr>
</tbody>
</table>

**Result (CN1)**

**Result (CN2)**

**Best case scenario** ➔ **Max Possible Score**

- No need to execute CN2!
- Requires “score monotonicity”
Pipelined /1

- Motivation
  - What if join result $\gg k$?

- Top-k optimization within a CN

### Motivation

What if join result $\gg k$?

### Top-k optimization within a CN

#### Result (CN1)

<table>
<thead>
<tr>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-W1-A2</td>
</tr>
<tr>
<td>P2-W5-A3</td>
</tr>
</tbody>
</table>

#### MPS

- MPS($P_3 - W? - [A_1, A_2]$) = $(1.8 + 1.2) / 3 = 1.0$
- MPS($[P_1, P_2] - W? - A_3$) = $(3.3 + 0.9) / 3 = 1.4$

#### Query

```
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
```
**Motivation**

- What if join result $\gg k$?

**Top-k optimization within a CN**

\[
\begin{array}{c|c|c|c}
\text{Result (CN1)} & \text{Score} \\
\hline
\text{P1-W8-A3} & 1.4 \\
\text{P2-W9-A3} & 1.2 \\
\ldots & \ldots \\
\hline
\text{MPS([P1, P2] – W? – A4)} = (3.3+0.3) / 3 = 1.2
\end{array}
\]

Can we stop?

\[
\begin{array}{c|c|c|c|c}
A4 & \leq 1.2 & \leq 1.2 \\
A3 & 1.4 & 1.2 & \leq 1.0 \\
A2 & \times & \times & \leq 1.0 \\
A1 & \times & \times & \leq 1.0 \\
P1 & P2 & P3 & \ldots \\
3.3 & 2.7 & 1.2 & \leq 1.2
\end{array}
\]
Global Pipelined

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

Naive $\Rightarrow$ Sparse $\Rightarrow$ Pipelined
- Be lazy!
- Utilize upper bound estimates

CN1: $P^Q \rightarrow W \rightarrow A^Q$

CN2: $P^Q \rightarrow W \rightarrow A \rightarrow W \rightarrow P^Q$

Get_MPS() Next() Get_MPS() Next()

top-2
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

<table>
<thead>
<tr>
<th>Temp Results</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2-W7-A2</td>
<td>1.47</td>
</tr>
</tbody>
</table>

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

- 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

![Diagram showing partitioning of search space into blocks]
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?

<table>
<thead>
<tr>
<th></th>
<th>Schema-based</th>
<th>Schema-free</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDBMS</td>
<td>CN</td>
<td></td>
</tr>
<tr>
<td>Graph</td>
<td>(Group) Steiner Tree, Distinct root semantics, Subgraph</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>XSEarch, Entities, ...</td>
<td>LCA and its variants</td>
</tr>
</tbody>
</table>

- 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:
    \[
    \text{sim}(k_i, k_j) = \frac{1}{|C_{k_i} \cup C_{k_j}|} \cdot \sum_{n_i \in C_{k_i}} \sum_{n_j \in C_{k_j}} \frac{1}{(|n_i \leftrightarrow n_j| + 1)^2}
    \]
  - XRank: \[p(n, k_1, k_2, \ldots, k_l) = |w|\]
    - \(w\) is the smallest text window in \(n\) that contains all search keywords

SIGMOD09 Tutorial 86 2009/7/15
Assigning Node Weights /1

- Based on graph structure
  - BANKS
    - Nodes: $\text{in-degree}_n(n)$
    - 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:**
  
  \[
  \text{Score}(n, Q) = \sum_{w \in Q \cap n} \frac{1 + \ln(1 + \ln(tf))}{(1 - s) + s \cdot dl / avdl} \cdot \ln \frac{N + 1}{df}
  \]

- Discover/EASE
- [Liu et al, SIGMOD06]
  
  \[
  ndl = \left(1 - s\right) + s \cdot dl / avdl \right) \cdot \left(1 + \ln \text{avdl}\right)
  \]
  \[
  nidf^g = \ln \frac{N^g}{df^g + 1}
  \]

- **SPARK**
  
  - but *not* at the node level
Score Aggregate Function

- Combine $s(node_i)$ into a final score for ranking
  - BANKS: $agg(edge) \times agg(node)^\lambda$
  - DISCOVER: $\sum_n s(n) / size\_normalization$
  - [Liu et al, SIGMOD06]:
    \[
    \max S_i \cdot \left( 1 + \ln \left( 1 + \ln \frac{\sum S_i}{\max S_i} \right) \right)
    \]

- 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
- **SUITES**: \[ \left( \frac{T.uniqKeywords}{|Q|} \right)^{\tau} \]

Ideal Pos: \((1,1)\), d = 1
L2 distance: d = 1.41

**SUITS**:

\[ \left( \frac{T.uniqKeywords}{|Q|} \right)^{\tau} \]
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]

- **Motivations**
  - Query may contain typos
  - Query may contain phrases
  - Speed up query processing

- **Input**
  - A keyword query
  - Database

- **Output**
  - Corrected and segmented query

Input:
- A keyword query: "new york time price"
- Database: "... new york... account...
- Output: "... new york times..."

Motivations:
- Speed up query processing
- Query may contain typos
- Query may contain phrases

Algorithm:
- \( O(3^n) \) DP alg
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.
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]

- 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
Small snippet:

- Goal: selecting data instances, such that as many items in ILList 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**

<table>
<thead>
<tr>
<th>Month</th>
<th>State</th>
<th>City</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>TX</td>
<td>Houston</td>
<td>US Open <strong>Pool</strong></td>
<td>Best of 19, ranking</td>
</tr>
<tr>
<td>Dec</td>
<td>TX</td>
<td>Dallas</td>
<td>Cowboy’s dream run</td>
<td><strong>Motorcycle</strong>, beer</td>
</tr>
<tr>
<td>Dec</td>
<td>TX</td>
<td>Austin</td>
<td>SPAM Museum party</td>
<td>Classical <strong>American food</strong></td>
</tr>
<tr>
<td>Oct</td>
<td>MI</td>
<td>Detroit</td>
<td><strong>Motorcycle</strong> Rallies</td>
<td>Tournament, round robin</td>
</tr>
<tr>
<td>Oct</td>
<td>MI</td>
<td>Flint</td>
<td>Michigan <strong>Pool</strong> Exhibition</td>
<td>Non-ranking, 2 days</td>
</tr>
<tr>
<td>Sep</td>
<td>MI</td>
<td>Lansing</td>
<td><strong>American Food</strong> history</td>
<td>The best food from USA</td>
</tr>
</tbody>
</table>

* Michigan
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

```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

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 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]

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) heterogeneous 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 [Kasnci et al, SIGMOD08]
Thank you!

Questions?


Reference /4