Motivation

• Keyword Search is the dominant information discovery method in documents
• Increasing amount of data stored in databases
Motivation

• Currently, information discovery in databases requires:
  – Knowledge of schema
  – Knowledge of a query language (eg: SQL)
  – Knowledge of the role of the keywords

• DISCOVER eliminates these requirements

Keyword Query - Semantics

Keywords are:
• in same tuple
• in same relation
• connected through primary-foreign key relationships

Score of result:
• distance of keywords within a tuple
• distance between keywords in terms of primary-foreign key connections
• weighted distance
Result of Keyword Query

Result is tree $T$ of tuples where:

- each edge corresponds to a primary-foreign key relationship
- every keyword contained in a tuple of $T$ (total)
- no tuple of $T$ is redundant (minimal)

Example - Schema

Subset of TPC-H schema
### Example - Data

**ORDERS**

<table>
<thead>
<tr>
<th>ORDERKEY</th>
<th>CUSTKEY</th>
<th>TOTALPRICE</th>
<th>CLERK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000105</td>
<td>12312</td>
<td>$5,000</td>
<td>John Smith</td>
</tr>
<tr>
<td>1000111</td>
<td>12312</td>
<td>$3,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000125</td>
<td>10001</td>
<td>$7,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000110</td>
<td>10002</td>
<td>$8,000</td>
<td>Keith Brown</td>
</tr>
</tbody>
</table>

**CUSTOMER**

<table>
<thead>
<tr>
<th>CUSTKEY</th>
<th>NAME</th>
<th>NATIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>12312</td>
<td>Brad Lou</td>
<td>01</td>
</tr>
<tr>
<td>10001</td>
<td>George Walters</td>
<td>01</td>
</tr>
<tr>
<td>10013</td>
<td>John Roberts</td>
<td>01</td>
</tr>
</tbody>
</table>

**NATION**

<table>
<thead>
<tr>
<th>NATIONKEY</th>
<th>NAME</th>
<th>REGIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>USA</td>
<td>N.America</td>
</tr>
</tbody>
</table>

### Example – Keyword Query

**Query:** "Smith, Miller"

**ORDERS**

<table>
<thead>
<tr>
<th>ORDERKEY</th>
<th>CUSTKEY</th>
<th>TOTALPRICE</th>
<th>CLERK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000105</td>
<td>12312</td>
<td>$5,000</td>
<td>John Smith</td>
</tr>
<tr>
<td>1000111</td>
<td>12312</td>
<td>$3,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000125</td>
<td>10001</td>
<td>$7,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000110</td>
<td>10002</td>
<td>$8,000</td>
<td>Keith Brown</td>
</tr>
</tbody>
</table>

**CUSTOMER**

<table>
<thead>
<tr>
<th>CUSTKEY</th>
<th>NAME</th>
<th>NATIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>12312</td>
<td>Brad Lou</td>
<td>01</td>
</tr>
<tr>
<td>10001</td>
<td>George Walters</td>
<td>01</td>
</tr>
<tr>
<td>10013</td>
<td>John Roberts</td>
<td>01</td>
</tr>
</tbody>
</table>

**NATION**

<table>
<thead>
<tr>
<th>NATIONKEY</th>
<th>NAME</th>
<th>REGIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>USA</td>
<td>N.America</td>
</tr>
</tbody>
</table>
Example – Keyword Query

Query: “Smith, Miller”

ORDERS

<table>
<thead>
<tr>
<th>ORDERKEY</th>
<th>CUSTKEY</th>
<th>TOTALPRICE</th>
<th>CLERK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000105</td>
<td>12312</td>
<td>$5,000</td>
<td>John Smith</td>
</tr>
<tr>
<td>1000111</td>
<td>12312</td>
<td>$3,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000125</td>
<td>10001</td>
<td>$7,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000110</td>
<td>10002</td>
<td>$8,000</td>
<td>Keith Brown</td>
</tr>
</tbody>
</table>

CUSTOMER

<table>
<thead>
<tr>
<th>CUSTKEY</th>
<th>NAME</th>
<th>NATIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>12312</td>
<td>Brad Lou</td>
<td>01</td>
</tr>
<tr>
<td>10001</td>
<td>George Walters</td>
<td>01</td>
</tr>
<tr>
<td>10013</td>
<td>John Roberts</td>
<td>01</td>
</tr>
</tbody>
</table>

NATION

<table>
<thead>
<tr>
<th>NATIONKEY</th>
<th>NAME</th>
<th>REGIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>USA</td>
<td>N.America</td>
</tr>
</tbody>
</table>

Results:

Size  Result
2       \( o_1 \leftarrow c_1 \rightarrow o_2 \)

Example – Keyword Query

Query: “Smith, Miller”

ORDERS

<table>
<thead>
<tr>
<th>ORDERKEY</th>
<th>CUSTKEY</th>
<th>TOTALPRICE</th>
<th>CLERK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000105</td>
<td>12312</td>
<td>$5,000</td>
<td>John Smith</td>
</tr>
<tr>
<td>1000111</td>
<td>12312</td>
<td>$3,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000125</td>
<td>10001</td>
<td>$7,000</td>
<td>Mike Miller</td>
</tr>
<tr>
<td>1000110</td>
<td>10002</td>
<td>$8,000</td>
<td>Keith Brown</td>
</tr>
</tbody>
</table>

CUSTOMER

<table>
<thead>
<tr>
<th>CUSTKEY</th>
<th>NAME</th>
<th>NATIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>12312</td>
<td>Brad Lou</td>
<td>01</td>
</tr>
<tr>
<td>10001</td>
<td>George Walters</td>
<td>01</td>
</tr>
<tr>
<td>10013</td>
<td>John Roberts</td>
<td>01</td>
</tr>
</tbody>
</table>

NATION

<table>
<thead>
<tr>
<th>NATIONKEY</th>
<th>NAME</th>
<th>REGIONKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>USA</td>
<td>N.America</td>
</tr>
</tbody>
</table>

Results:

Size  Result
2       \( o_1 \leftarrow c_1 \rightarrow o_2 \)

4       \( o_1 \leftarrow c_1 \leftarrow n_1 \rightarrow c_2 \rightarrow o_3 \)

Smaller sizes usually denote tighter association between keywords.
CREATE TABLE T1 AS SELECT * FROM ORDERSSmith, CUSTOMERS WHERE ...
SELECT * FROM T1, ORDERSMiller WHERE ...
SELECT * FROM T1, NATION, CUSTOMERS, ORDERSMiller WHERE ...

"Smith", "Miller"

ORDERSSmith = \{o1\}  
ORDERSMiller = \{o2, o3\}
Candidate Networks Generator - Challenges

- A keyword may appear in multiple tuples
- \# candidate networks can be too big (sometimes unbounded)

Candidate Network - Example
Candidate Network - Example

CN1: $O^{Smith} \leftarrow C \rightarrow O^{Miller}$

size=2

CN2: $O^{Smith} \leftarrow C \leftarrow N \rightarrow C \rightarrow O^{Miller}$

size=4
Candidate Network - Example

ORDERS
Miller
ORDERS
ORDERS
CUSTOMER
n:1
n:1
n:1
n:1
NATION

CN3: OSmith ← C → OMiller → C size=3

------------------------------

c1 – o – c2

c1 ≡ c2, because primary to foreign key from CUSTOMER to ORDERS

Pruning Condition: R^K→S←R^L

Candidate Network - Example

ORDERS
Smith
ORDERS
ORDERS
CUSTOMER
n:1
n:1
n:1
n:1
NATION

CN4: OSmith ← C → O ← C → OMiller size=4

------------------------------

c1 – o – c2

c1 ≡ c2, because primary to foreign key from CUSTOMER to ORDERS

Pruning Condition: R^K→S←R^L
Candidate Networks Generator - Algorithm

- Traverse tuple set graph breadth first
- $Q \leftarrow$ tuple sets containing keyword $k_1$
- For each network $n$ of tuple sets in $Q$ do
  - If $pruning\_condition(n)$ drop $n$
  - else if $is\_CN(n)$ output $n$
  - else expand $n$ by one tuple set to all possible directions in tuple set graph and insert expansions to $Q$
    
    [eg: if $n$ is $O^{Smith} \leftarrow C$ then we add to $Q$
    $O^{Smith} \leftarrow C \rightarrow O^{Miller}$, $O^{Smith} \leftarrow C \rightarrow O$, $O^{Smith} \leftarrow C \leftarrow N$ ]

Candidate Networks Generator is Complete and Non-Redundant

- Prove that the set of Candidate Networks generated is
  - Complete: All solutions generated by a CN
  - Non-redundant: There is database instance, where by removing a CN a solution is lost
Size of Candidate Networks may be Unbounded

- Size is unbounded iff schema graph G has one of the following properties:
  - There is a node of G that has at least two incoming edges.
    [eg: PARTSUPP→LINEITEM←ORDERS]
  - G has a directed cycle.
    [eg: ancestor schemas]

Architecture
Execution Plan - Challenges

- Generated SQL queries are expensive due to joins
- Reusability opportunities

Execution Plan

- Each CN corresponds to a SQL statement
- CN1: $O^{Smith} \leftarrow C \rightarrow O^{Miller}$
  CN2: $O^{Smith} \leftarrow C \leftarrow N \rightarrow C \rightarrow O^{Miller}$
- Execution Plan
  CN1 $\leftarrow O^{Smith} \triangleright \C \triangleright O^{Miller}$
  CN2 $\leftarrow O^{Smith} \triangleright \C \triangleright N \triangleright C \triangleright O^{Miller}$
Reusable Common Subexpressions - Example

- Execution Plan
  CN1 ← O^{Smith} < C < O^{Miller}
  CN2 ← O^{Smith} < C < N < C < O^{Miller}

- Optimized Execution Plan
  Temp ← O^{Smith} < C
  CN1 ← Temp < O^{Miller}
  CN2 ← Temp < N < C < O^{Miller}

Optimal Reuse of Common Subexpressions is NP-Complete

- Simple Cost Model: each join has cost 1
- Prove that finding Optimal Common Subexpressions is NP-Complete.
  **Proof:** Reduce string compression problem
Cost Model and Greedy Optimization Algorithm

- Actual Cost Model: cost of a join is size of result
- Greedy algorithm:
  In each iteration build intermediate result of size 1 (1 join) that maximizes

\[ \frac{\text{frequency}^a}{\log^b(\text{size})}, 0 \leq a, b \leq 1 \]

Tuning of Greedy Algorithm

\[ \frac{\text{frequency}^a}{\log^b(\text{size})}, 0 \leq a, b \leq 1 \]

- a: frequency factor
  - favors reusability
- b: size factor
  - favors small intermediate results
- a = 1
- 0 ≤ b ≤ 0.3
Related Work

- **DBXplorer. S. Agrawal et al. ICDE 2002**
  - Similar three step architecture
  - Incomplete solutions (relations are not re-used)
  - Non-pruning Candidate Network generator
  - No common subexpression reusability
- **BANKS. G. Bhalotia et al. ICDE 2002**
  - Database viewed as graph
  - Steiner tree problem approximations
- **Proximity searching in databases. R. Goldman et al. VLDB 1998**
  - Database viewed as graph
  - No schema info
  - hub nodes

Performance and Tuning of the frequency/size ratio

TPC-H Dataset

**Variables**

- Max CN size
- # keywords
- frequency factor $a$
- size factor $b$
Experimentation: Pruning Capabilities of CN Generator

- #keywords = 2
- TPC-H schema
- Randomly insert keywords
- Keyword in relation R with $\Pr(R) = a \cdot \log(\text{size}(R))$
- Select $a$ such that $0.01 \leq \Pr(R) \leq 0.1$

<table>
<thead>
<tr>
<th>maxCNSize</th>
<th>netw w/ keyw</th>
<th>CNs</th>
<th>Tuple Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.54</td>
<td>1</td>
<td>2.96</td>
</tr>
<tr>
<td>2</td>
<td>25.78</td>
<td>2.12</td>
<td>2.96</td>
</tr>
<tr>
<td>3</td>
<td>168.88</td>
<td>3.7</td>
<td>2.96</td>
</tr>
<tr>
<td>4</td>
<td>2961</td>
<td>6.4</td>
<td>2.96</td>
</tr>
<tr>
<td>5</td>
<td>51045</td>
<td>11.45</td>
<td>2.96</td>
</tr>
</tbody>
</table>

Experiments - Speedup by using common subexpressions
- #keywords = 3
- TPC-H dataset

Experiments using common subexpressions
Experiments – Execution Times

- # keywords = 2
- Each added keyword in 50 tuples in 2 relations

Current & Future Work

Current Work
- XKeyword is system for efficient keyword search in XML databases
- XKeyword is dedicated system and uses materialized views to speedup execution
- Specialized UI for summarizing results
- Demo on DBLP dataset available at [www.db.ucsd.edu/XKeyword](http://www.db.ucsd.edu/XKeyword)

Future Work
- Investigate other proximity semantics
- More efficient Master Index
- Compare different result presentation methods