DISCOVER: Keyword Search in Relational Databases

- *Vagelis Hristidis*
  University of California, San Diego
- Yannis Papakonstantinou
  University of California, San Diego

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

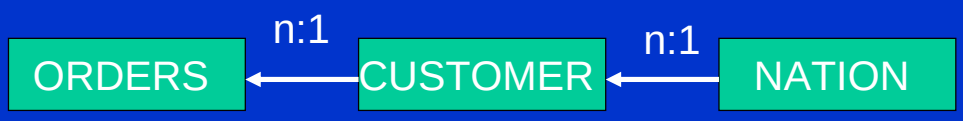
Result of Keyword Query

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

	ORDERKEY	CUSTKEY	TOTALPRICE	CLERK	...
o_1	1000105	12312	$5,000	John Smith	
o_2	1000111	12312	$3,000	Mike Miller	
o_3	1000125	10001	$7,000	Mike Miller	
o_4	1000110	10002	$8,000	Keith Brown	

CUSTOMER

	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

Example – Keyword Query

Query: "Smith, Miller"

ORDERS

	ORDERKEY	CUSTKEY	TOTALPRICE	CLERK	...
o_1	1000105	12312	$5,000	John Smith	
o_2	1000111	12312	$3,000	Mike Miller	
o_3	1000125	10001	$7,000	Mike Miller	
o_4	1000110	10002	$8,000	Keith Brown	

CUSTOMER

	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

Example – Keyword Query

Query: "Smith, Miller"

ORDERS

	ORDERKEY	CUSTKEY	TOTALPRICE	CLERK	...
o_1	1000105	12312	$5,000	John Smith	
o_2	1000111	12312	$3,000	Mike Miller	
o_3	1000125	10001	$7,000	Mike Miller	
o_4	1000110	10002	$8,000	Keith Brown	

CUSTOMER

	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

Results:

Size	Result
2	$o_1 \leftarrow c_1 \rightarrow o_2$

Example – Keyword Query

Query: "Smith, Miller"

ORDERS

	ORDERKEY	CUSTKEY	TOTALPRICE	CLERK	...
o_1	1000105	12312	$5,000	John Smith	
o_2	1000111	12312	$3,000	Mike Miller	
o_3	1000125	10001	$7,000	Mike Miller	
o_4	1000110	10002	$8,000	Keith Brown	

CUSTOMER

	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

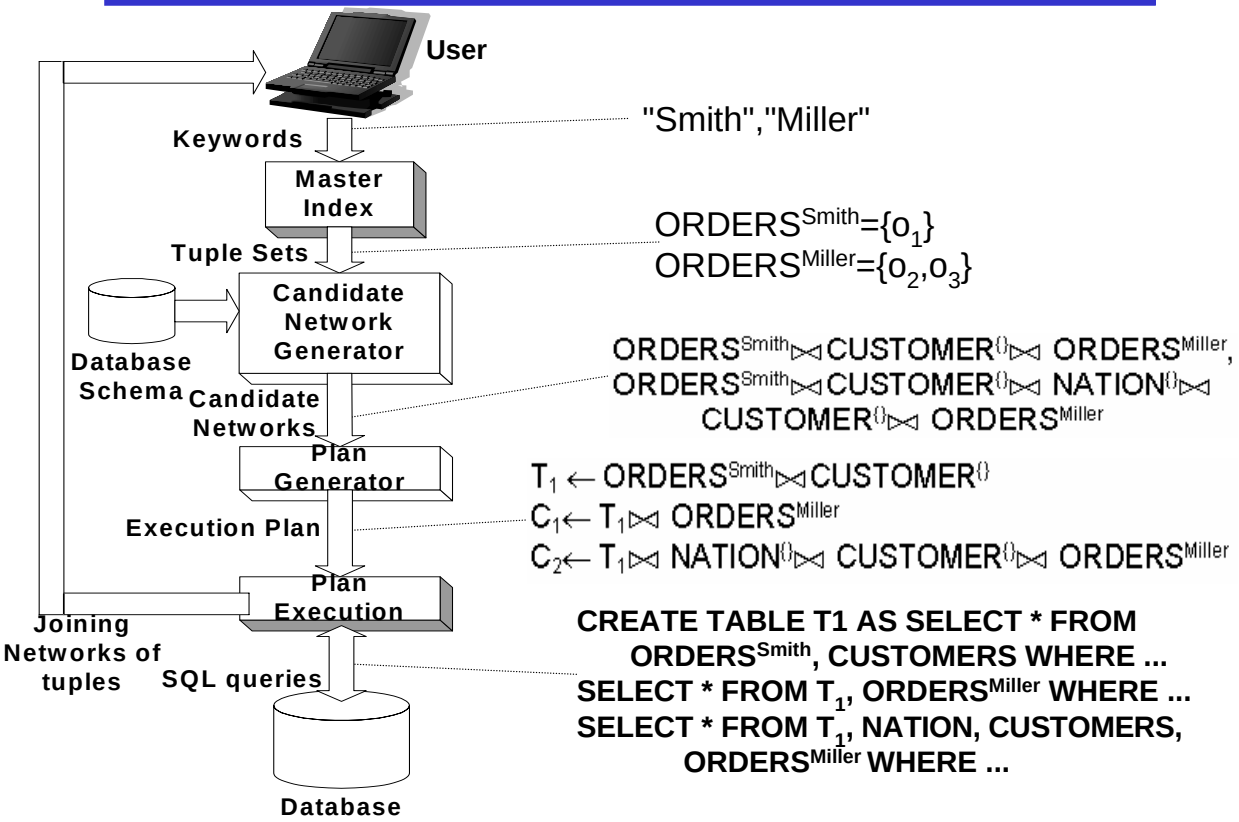
	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

Results:

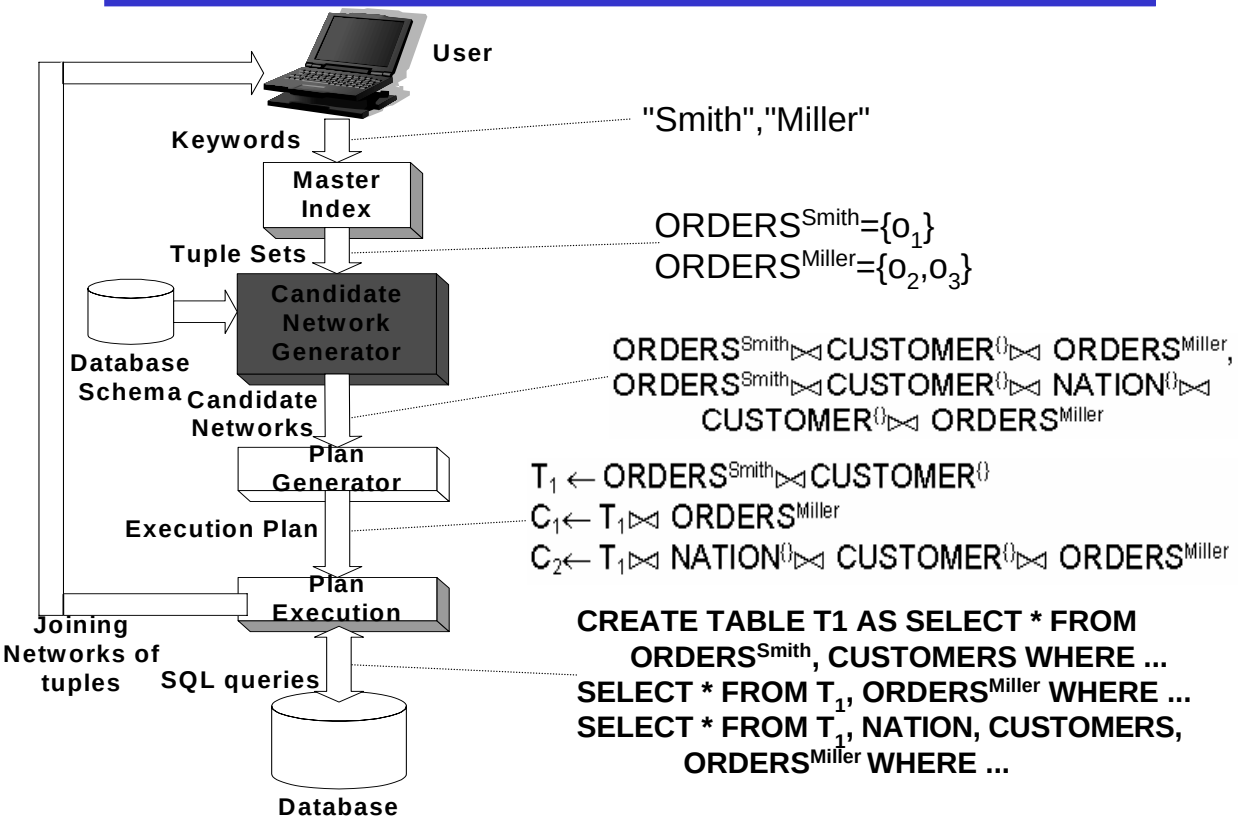
Size	Result
2	$o_1 \leftarrow c_1 \rightarrow o_2$
4	$o_1 \leftarrow c_1 \leftarrow n_1 \rightarrow c_2 \rightarrow o_3$

Smaller sizes usually denote tighter association between keywords

Architecture



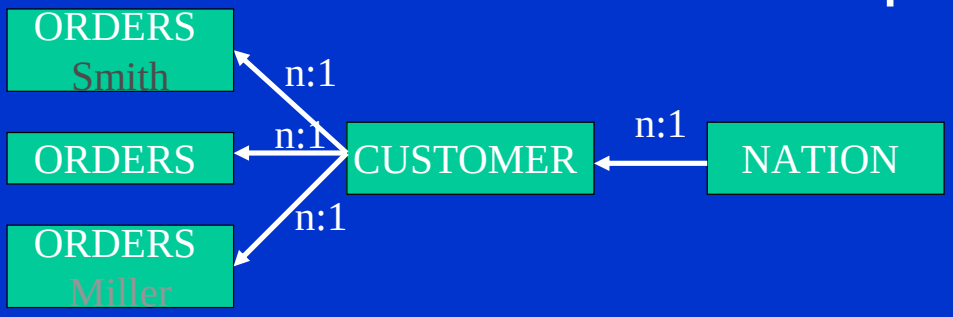
Architecture



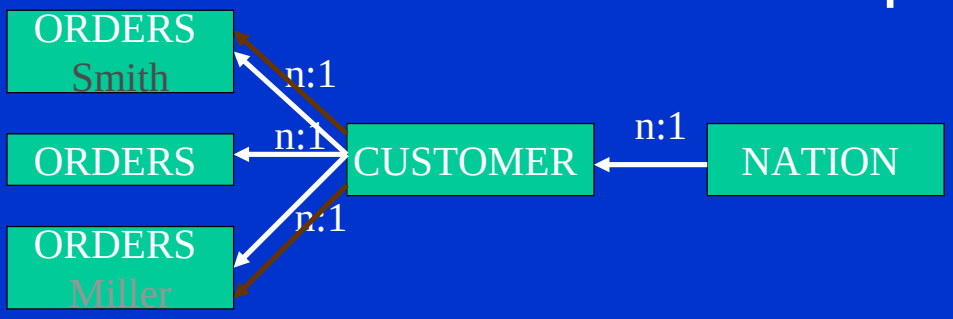
Candidate Networks Generator - Challenges

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

Candidate Network - Example

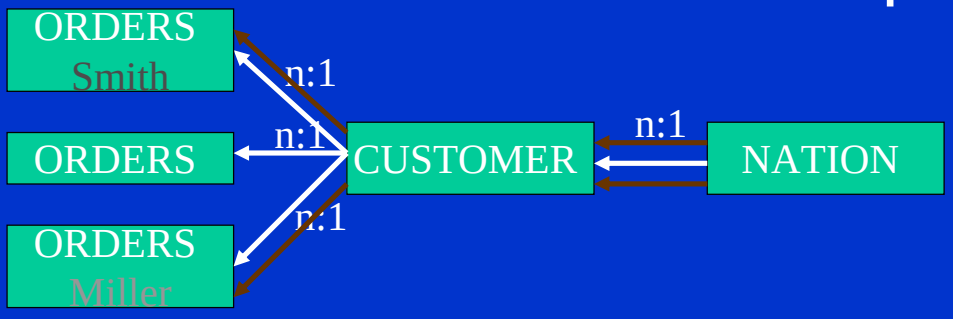


Candidate Network - Example



CN1: $O^{Smith} \leftarrow C \rightarrow O^{Miller}$ size=2

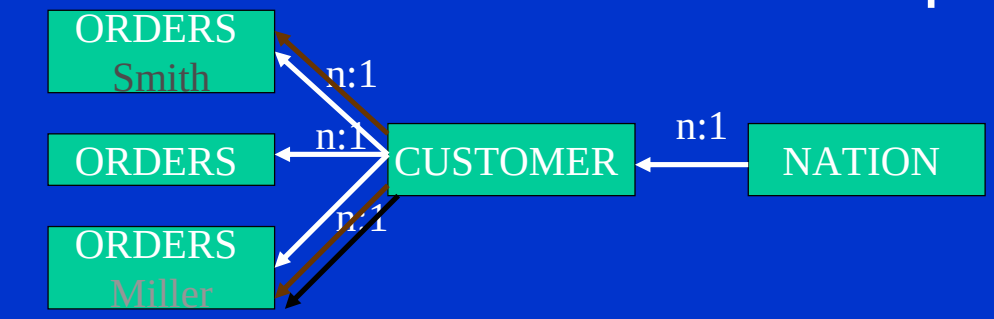
Candidate Network - Example



CN1: $O^{Smith} \leftarrow C \rightarrow O^{Miller}$ size=2

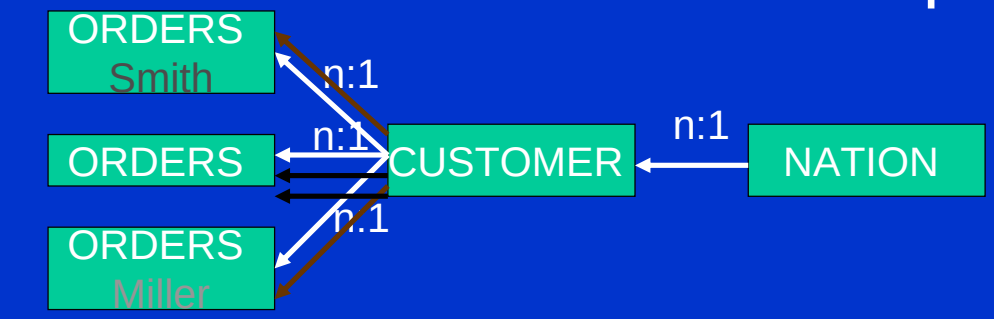
CN2: $O^{Smith} \leftarrow C \leftarrow N \rightarrow C \rightarrow O^{Miller}$ size=4

Candidate Network - Example



CN3: $O^{Smith} \leftarrow C \rightarrow O^{Miller} \leftarrow C$ size=3

Candidate Network - Example



CN4: $O^{Smith} \leftarrow C \rightarrow O \leftarrow C \rightarrow O^{Miller}$ size=4

$c_1 - o - c_2$

$c_1 \equiv c_2$, because primary to foreign key from CUSTOMER to ORDERS

Pruning Condition: $R^K \rightarrow S \leftarrow R^L$

Candidate Networks Generator - Algorithm

- Traverse tuple set graph breadth first
- Q ← 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

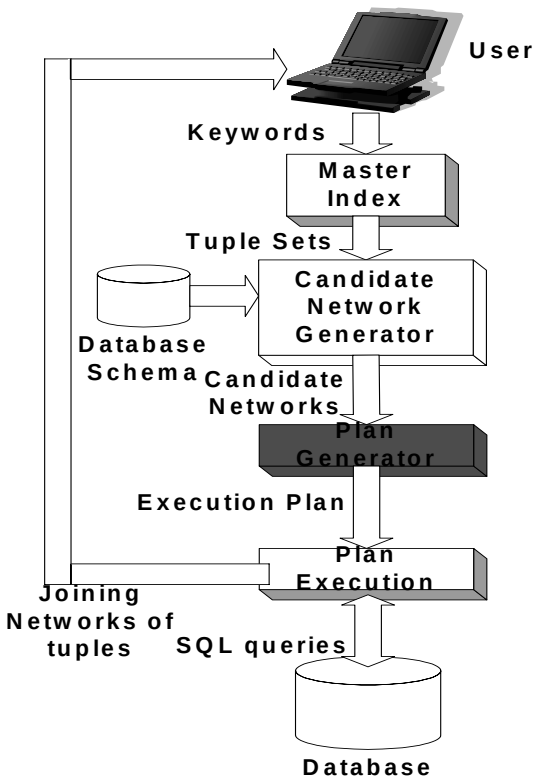
Size of Candidate Networks may be Unbounded

- 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 $\leftarrow O^{Smith} \triangleright\triangleleft C \triangleright\triangleleft O^{Miller}$

  CN2 $\leftarrow O^{Smith} \triangleright\triangleleft C \triangleright\triangleleft N \triangleright\triangleleft C \triangleright\triangleleft O^{Miller}$

Reuse Common Subexpressions - Example

- Execution Plan

  $CN1 \leftarrow O^{Smith} \bowtie C \bowtie O^{Miller}$

  $CN2 \leftarrow O^{Smith} \bowtie C \bowtie N \bowtie C \bowtie O^{Miller}$
- Optimized Execution Plan

  $Temp \leftarrow O^{Smith} \bowtie C$

  $CN1 \leftarrow Temp \bowtie O^{Miller}$

  $CN2 \leftarrow Temp \bowtie N \bowtie C \bowtie O^{Miller}$

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

$$\frac{\text{frequency}^a}{\log^b(\text{size})}, 0 \le a,b \le 1$$

Tuning of Greedy Algorithm

$$\frac{\text{frequency}^a}{\log^b(\text{size})}, 0 \le a,b \le 1$$

- a: frequency factor
  - ➢ favors reusability
- b: size factor
  - ➢ favors small intermediate results
- a=1
- $0 \le b \le 0.3$

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*

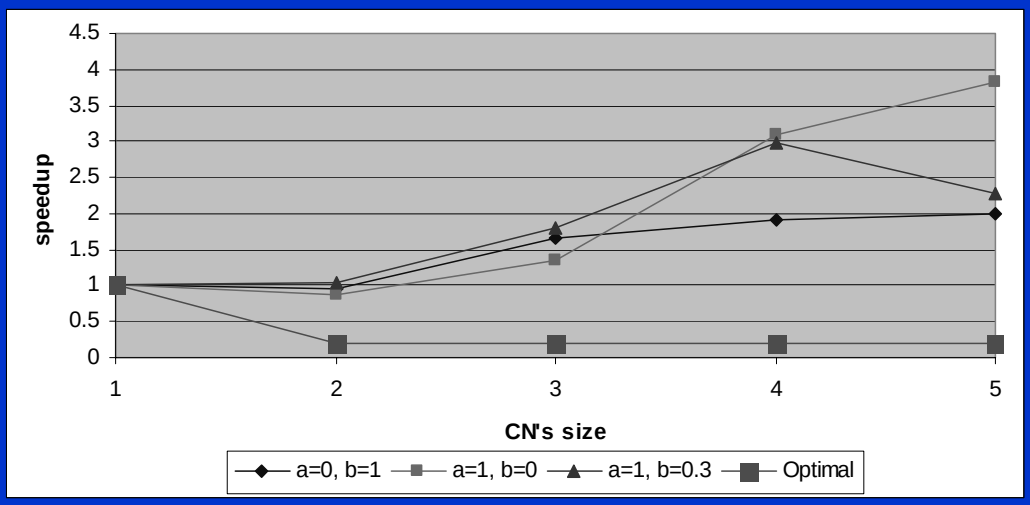
Experimentation: Pruning Capabilities of CN Generator

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

maxCNSize	*netw w/ keyw*	*CNs*	*Tuple Sets*
1	4.54	1	2.96
2	25.78	2.12	2.96
3	168.88	3.7	2.96
4	2961	6.4	2.96
5	51045	11.45	2.96

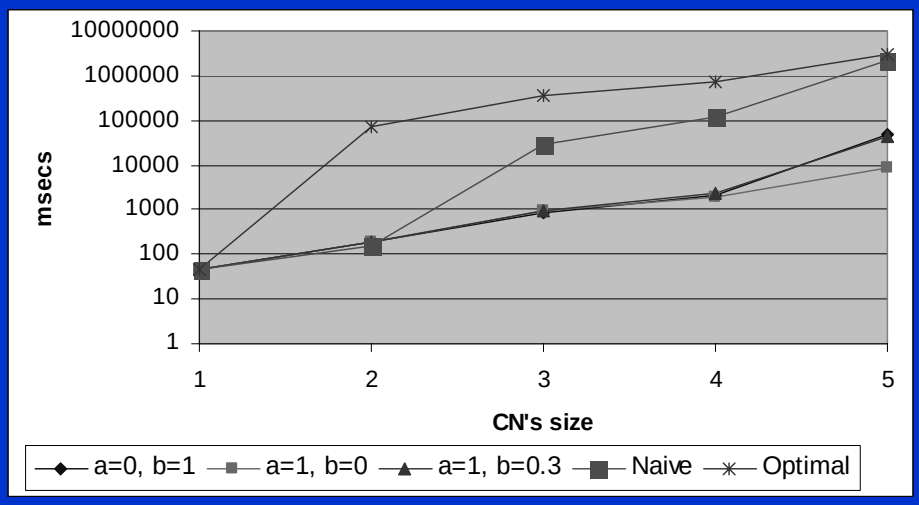
Experiments - Speedup by using common subexpressions

- # keywords = 3
- TPC-H dataset



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

Future Work

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