Keyword Searching and Browsing in Databases using BANKS

Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan

Presenter: Monir Hajiaghayi

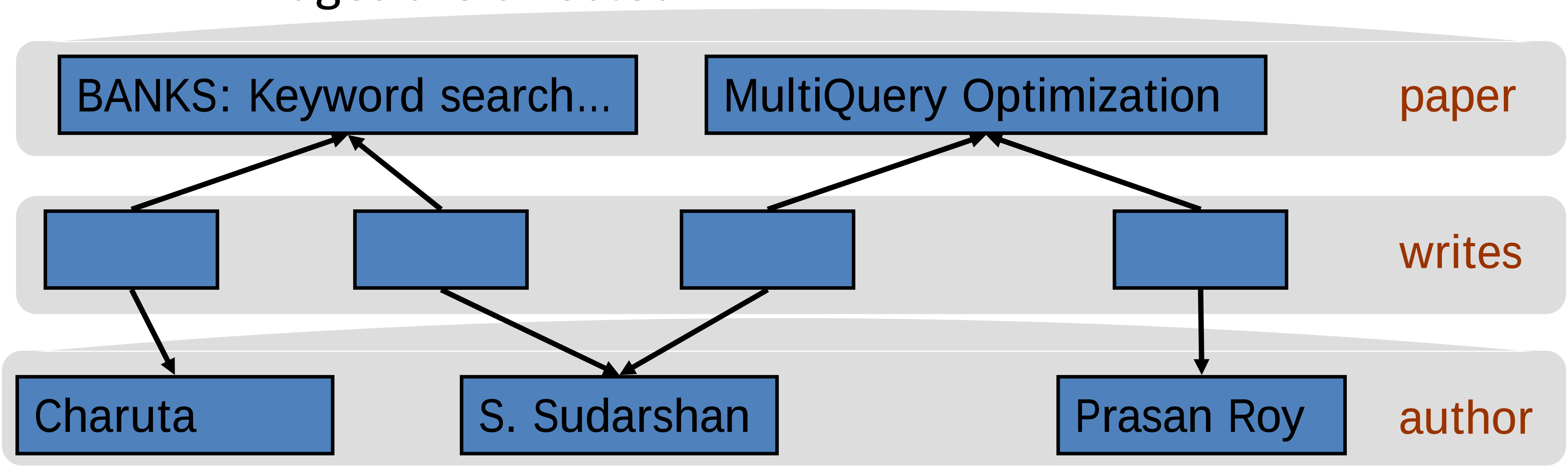
Discussion Leader : Ben Vandervalk

Motivation

- Web search engines are very successful
  - Simple and intuitive keyword query interface
- Database querying using keywords is desirable
  - Query languages, e.g.,SQL/QBE, are not appropriate for casual users
  - Form interfaces cumbersome, give limited views
- Examples of keyword queries on databases
  - e-store database: “camcorder panasonic”
  - Book store: “sudarshan databases”
- Differences from IR/Web Search
  - Normalization splits related data across multiple tuples
  - Answer to a query is a set of (closely) connected tuples that match all given keywords

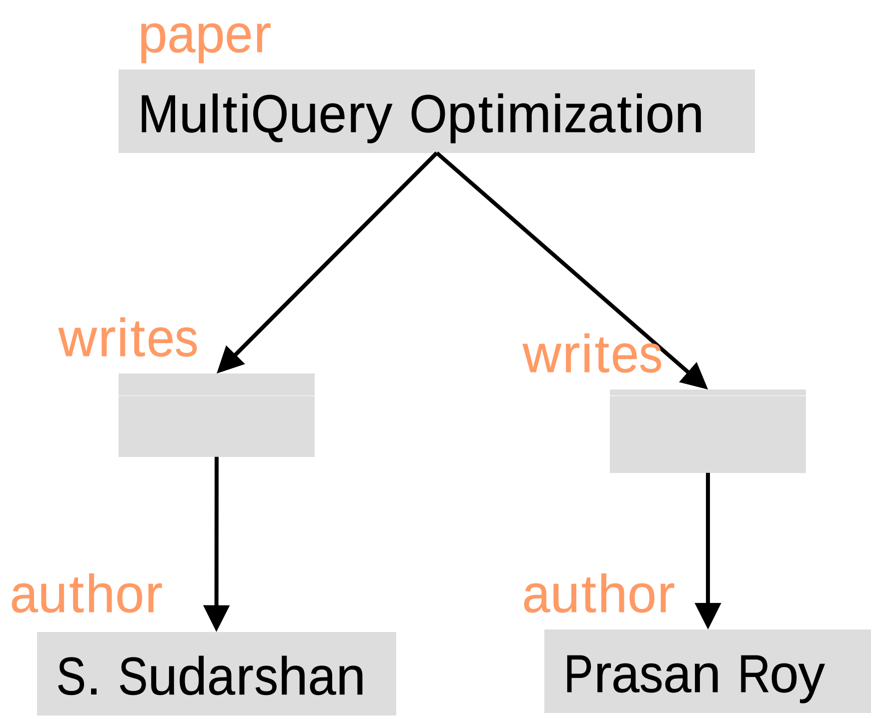
Basic Model

- Database: modeled as a graph
  - Nodes = tuples
  - Edges = references between tuples
    - foreign key, inclusion dependencies, etc.
    - Edges are directed



Answer Model

- Rooted, directed tree connecting keyword nodes
  - May include internal nodes that contain no keywords
  - Root node has special significance
    - May be restricted to relations representing entities
    - Avoid relations representing relationships, e.g. “writes”
- Multiple answers may exist
  - Ranked by proximity + prestige



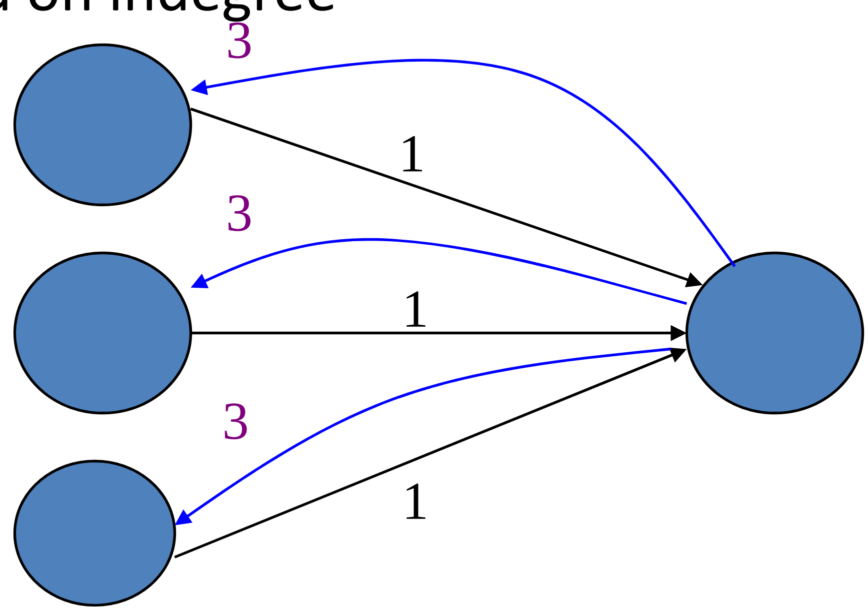
Eg> “Sudarshan Roy”

Discussion Question

- In the 90's, a board game came out called "Tribond". It was a trivia game where players had to find the "common link" between three items.
  - Example: What do "Bering", "Black", and "Baltic" have in common? (Answer: They are seas.)
- This is essentially what BANKS and DISCOVER systems do, but in the context of a relational database.
- What are the practical applications for finding the "common link" between a set of keywords in a database?

Relevance Calculation

- Proximity
  - Forward edges: foreign key → primary key
  - Weight of forward edge is based on schema
    - E.g. “cites” link weight greater than “writes” link weight
  - May need backward edges to form answer tree
    - Weight of backward edge u→v $\propto$ indegree of u
- Node prestige based on indegree



Discussion Question

- On WebCT, many of you commented that the assignment of forward/reverse edge weights was complicated and ad hoc.
  - What criteria should be used for assigning edge weights? Is there a good way to assign the weights automatically? Or should the weights be assigned manually, based on the particular schema?

Searching for the Best Answers

- We have to use not just the tree with the highest relevance score but also those with high scores

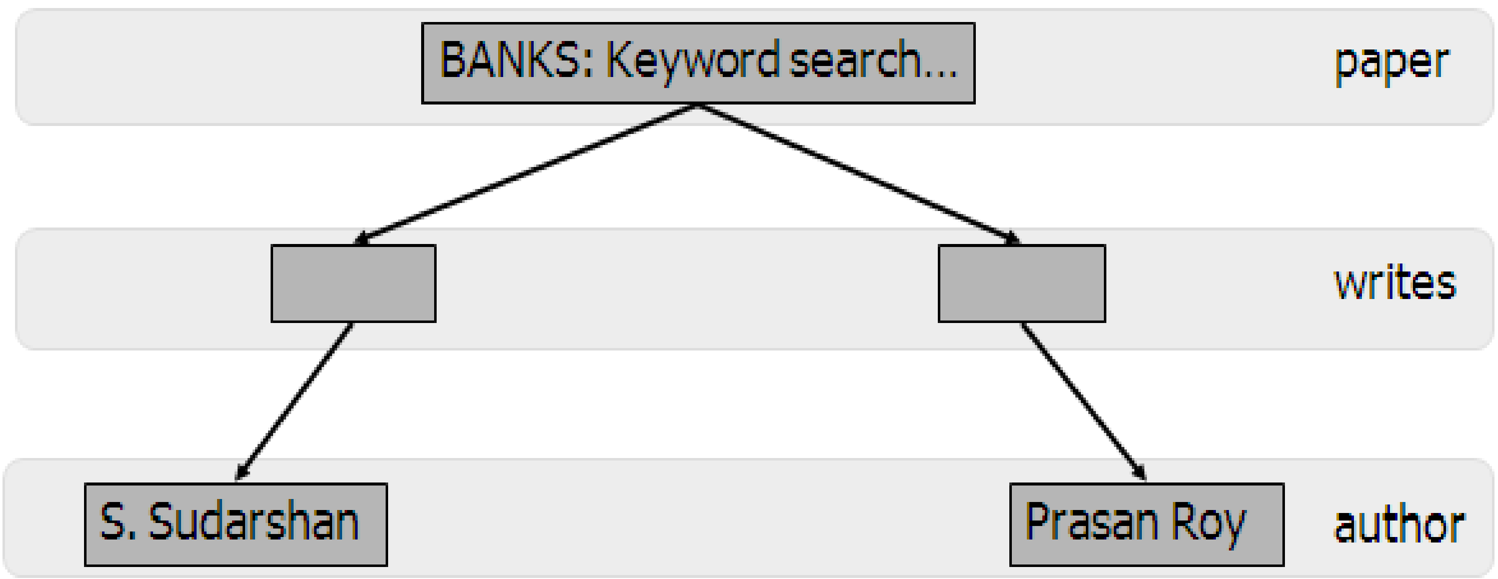
- Answers have to be generated incrementally so that the user are provided with the 'best' answers at the beginning

Backward Expanding Search

- Incrementally computes search results
- Start at leaf nodes each containing a query keyword
- Run concurrent single source shortest path algorithm from each such node
    - Output a node whenever it is on the intersection of the sets of nodes reached from each keyword
- Answer trees may not be generated in relevance order
    - Insert answers to a small buffer (heap)
    - Output highest ranked answer from buffer to user when buffer is full

Searching for Best Answers

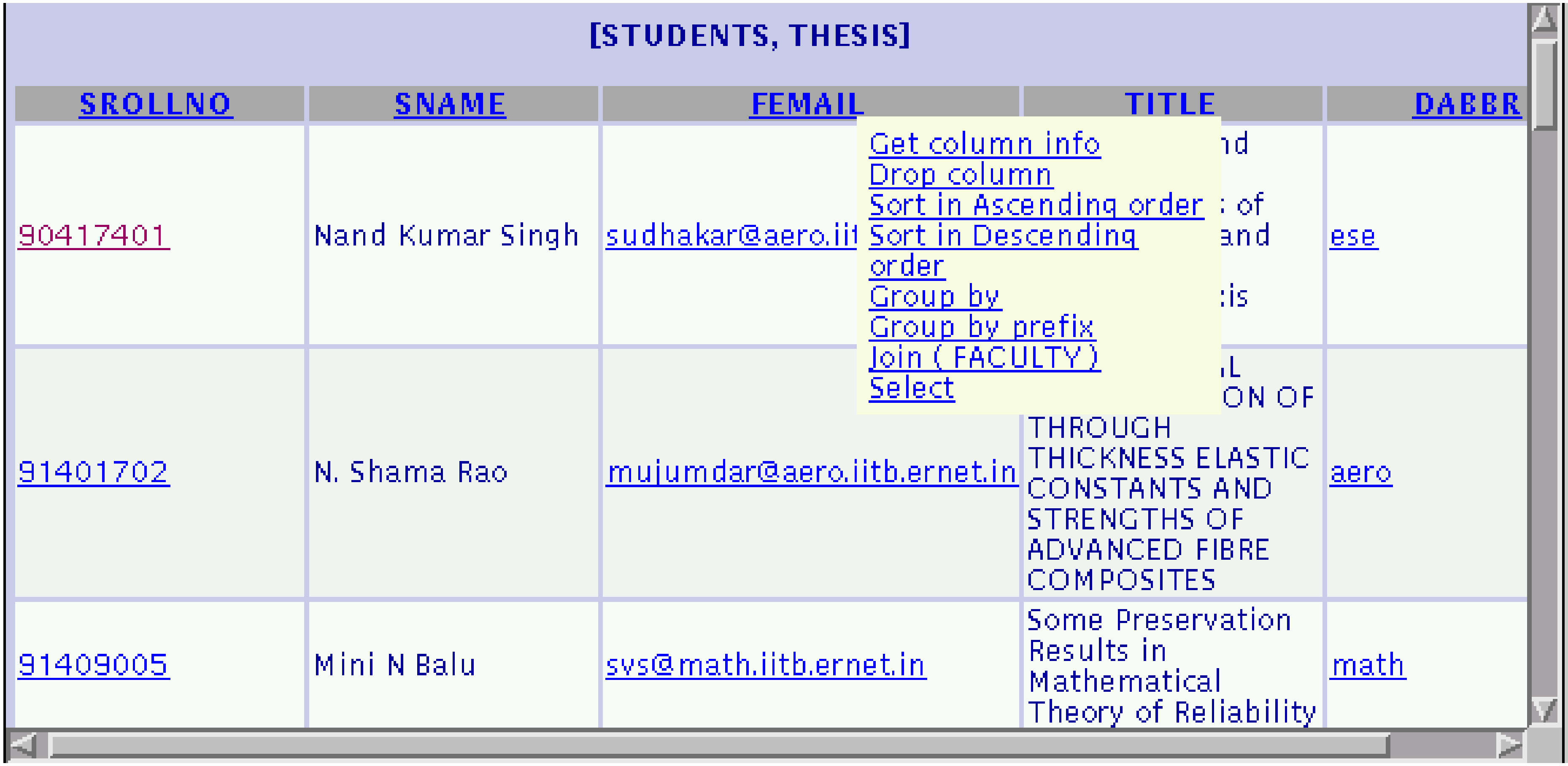
- Model (Query : Roy Sudarshan)



Browsing through BANKS

- BANKS system provides
  - A rich interface to browse data stored in a relational database
  - Automatically generates browsable views of database relations and query results
  - Schema browsing and data browsing
  - A hyperlink to the referenced tuple

Example of Browsing in BANKS



DISCOVER: Keyword Search in Relational Databases

- *Vagelis Hristidis*

  University of California, San Diego

- Yannis Papakonstantinou

  University of California, San Diego

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

(definition of “document” in databases)

Keywords are:

- in same tuple
- in same relation
- in tuples 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
- IR-style score of result tree

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)

Discussion Question

- In BANKS/DISCOVER "search hits" are not documents but rather trees of connected tuples.
- The BANKS paper shows an example result for the keyword search "soumen sunita":

Are these results easy for the user to understand? How could the results be displayed/navigated so that the system is more intuitive for the user?

Table = PAPER

PAPERID	TITLE	YEAR
ChakrabartiSD98	Mining Surprising Patterns Using Temporal Description Length.	

Table = WRITES

NAME	PAPERID
Soumen Chakrabarti	ChakrabartiSD98

Table = AUTHOR

NAME	URL
Soumen Chakrabarti	

Table = WRITES

NAME	PAPERID
Sunita Sarawagi	ChakrabartiSD98

Table = AUTHOR

NAME	URL
Sunita Sarawagi	

Example - Schema

Subset of TPC-H schema

Example - Data

	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

Smith Miller

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

	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”

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

	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

Smith Miller

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

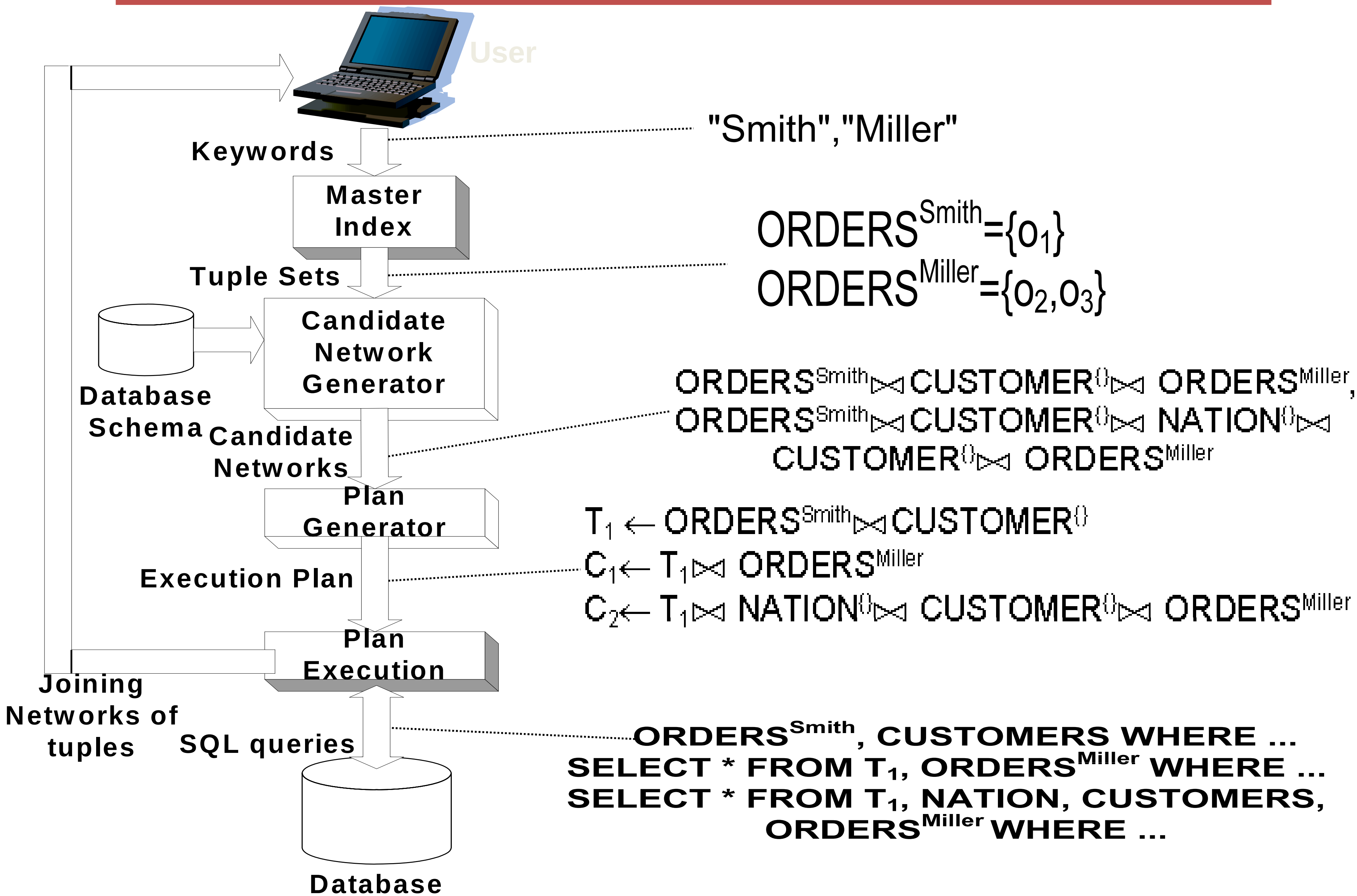
	CUSTKEY	NAME	NATIONKEY	...
c_1	12312	Brad Lou	01	
c_2	10001	George Walters	01	
c_3	10013	John Roberts	01	

NATION

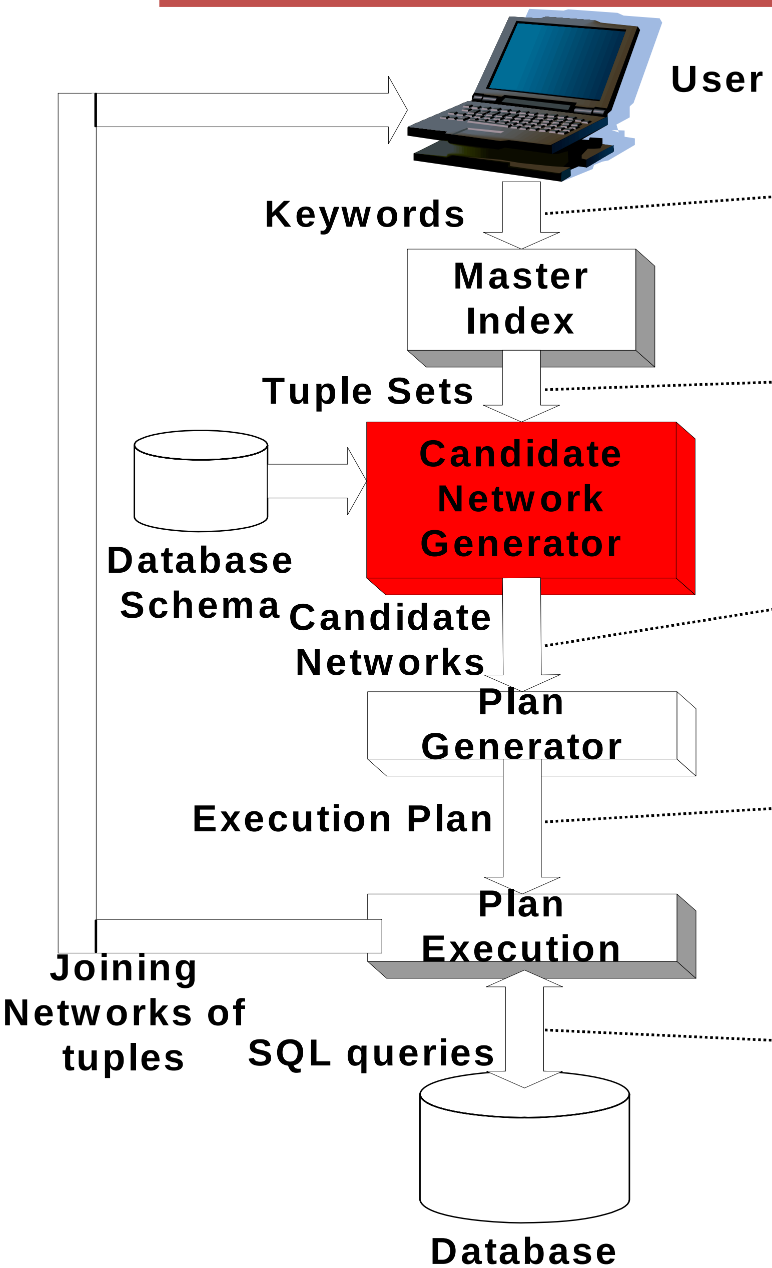
	NATIONKEY	NAME	REGIONKEY
n_1	01	USA	N.America

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

Architecture



Architecture



"Smith","Miller"

$ORDERS^{Smith}=\{o_1\}$
$ORDERS^{Miller}=\{o_2,o_3\}$

$ORDERS^{Smith} \bowtie CUSTOMER^{\{\}} \bowtie ORDERS^{Miller}$,
$ORDERS^{Smith} \bowtie CUSTOMER^{\{\}} \bowtie NATION^{\{\}} \bowtie CUSTOMER^{\{\}} \bowtie ORDERS^{Miller}$

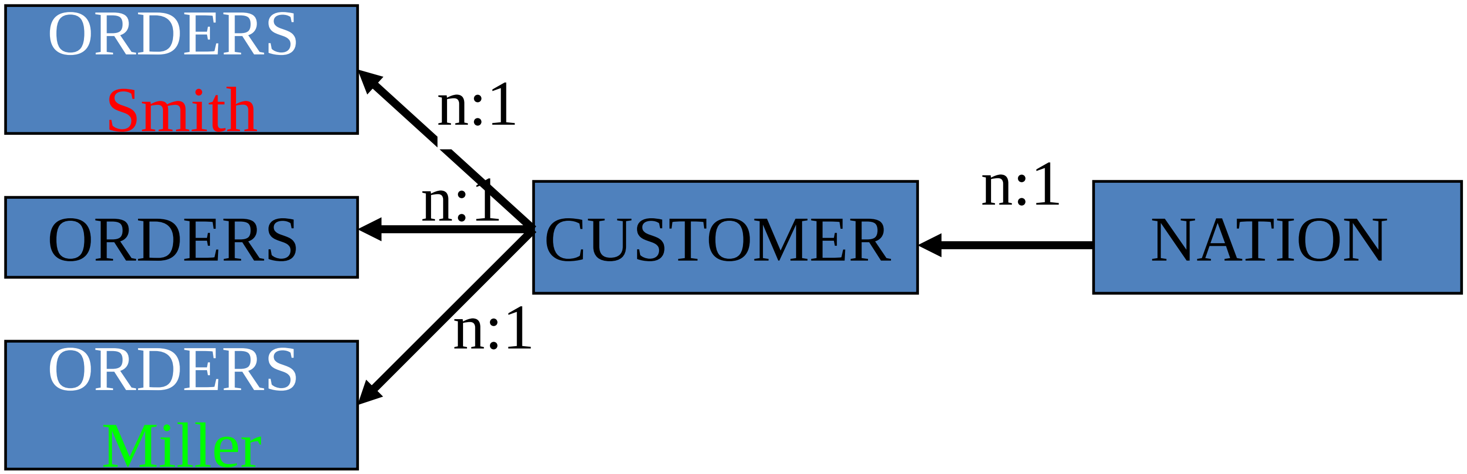
$T_1 \leftarrow ORDERS^{Smith} \bowtie CUSTOMER^{\{\}}$
$C_1 \leftarrow T_1 \bowtie ORDERS^{Miller}$
$C_2 \leftarrow T_1 \bowtie NATION^{\{\}} \bowtie CUSTOMER^{\{\}} \bowtie ORDERS^{Miller}$

CREATE TABLE T1 AS SELECT * FROM $ORDERS^{Smith}$, CUSTOMERS WHERE ...
SELECT * FROM T_1, $ORDERS^{Miller}$ WHERE ...
SELECT * FROM T_1, NATION, CUSTOMERS, $ORDERS^{Miller}$ WHERE ...

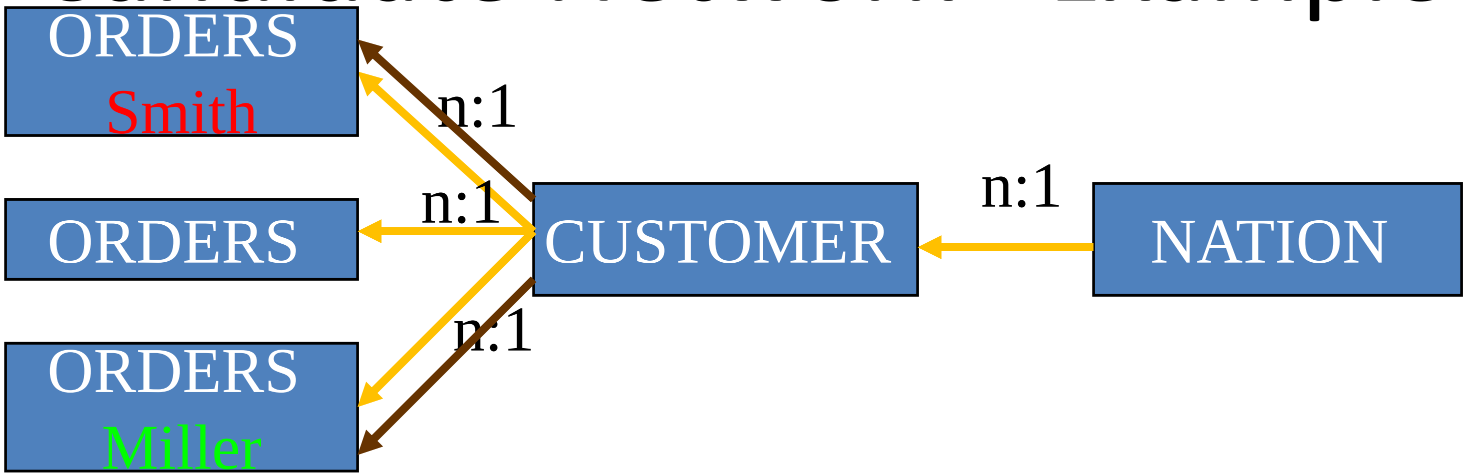
Candidate Networks Generator - Definition

- *Candidate Network* is a connected graph of tuple sets, where:
  - each edge has corresponding edge in schema graph
  - each keyword contained in at least one tuple set
  - there are no redundant tuple sets (with no keyword or not helping connect other keyword relations)

Candidate Network - Example

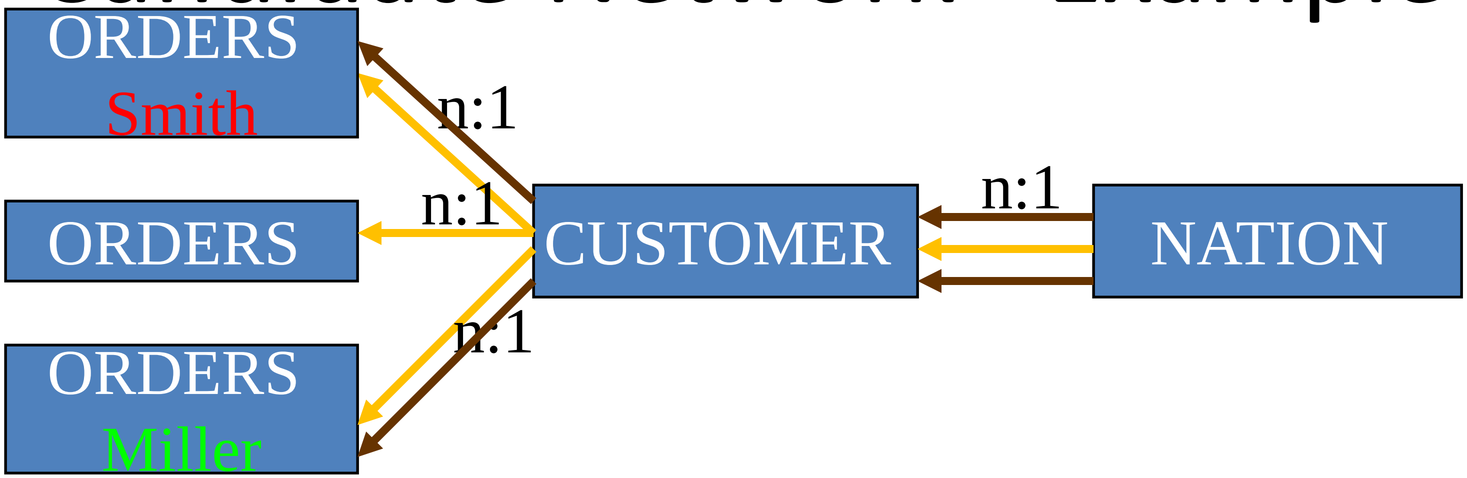


Candidate Network - Example



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

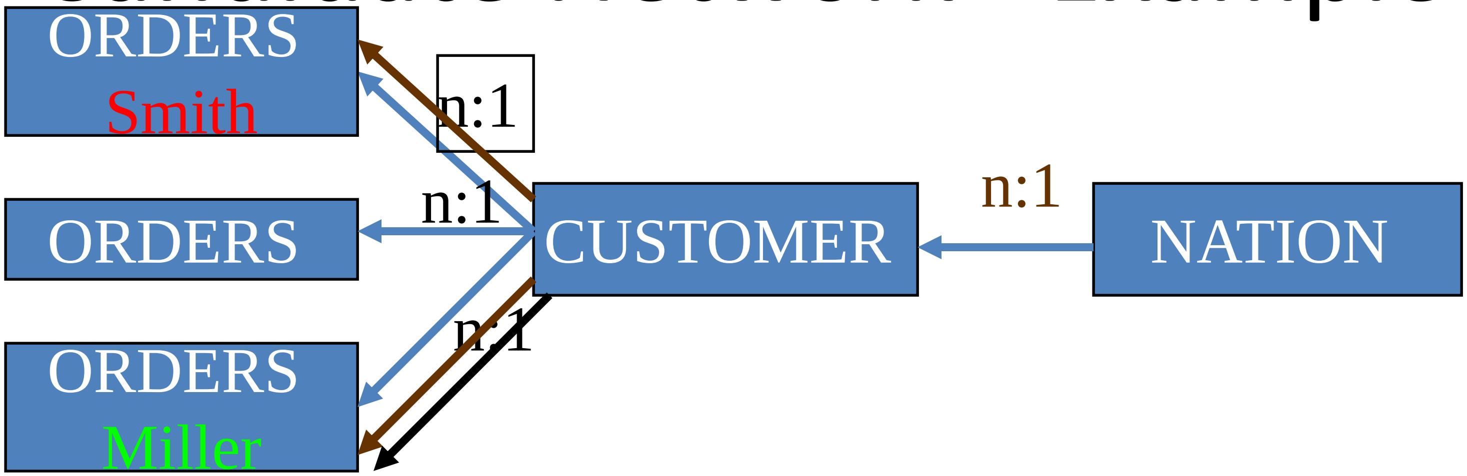
Candidate Network - Example



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

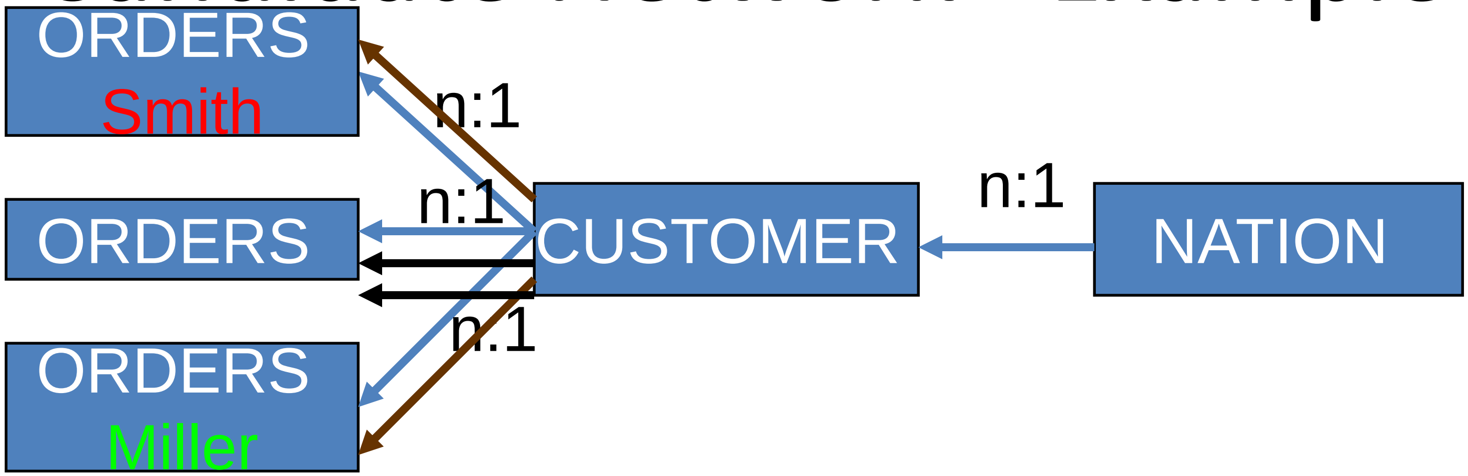
CN2: $O^{Smith} \leftarrow C \leftarrow N \rightarrow C \rightarrow O^{Miller}$ size=4

Candidate Network - Example



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

Candidate Network - Example



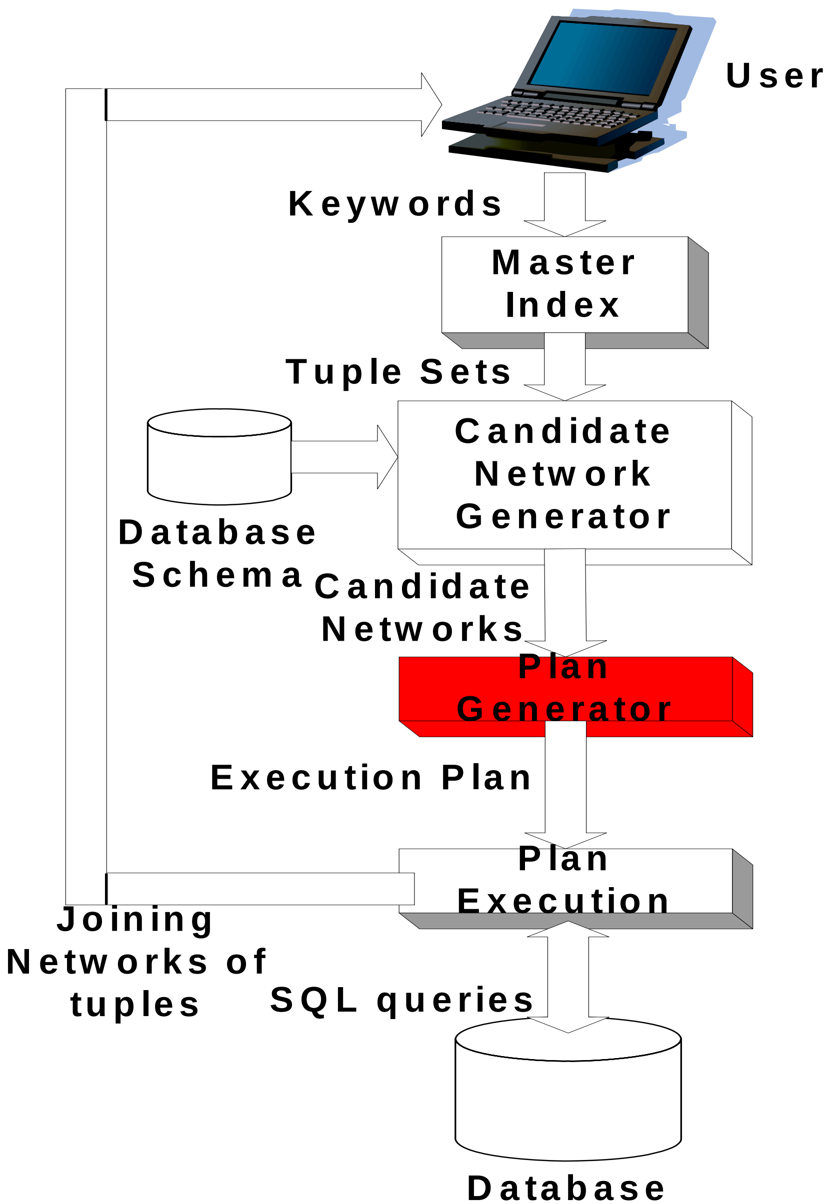
~~CN4: $O^{Smith} \leftarrow \mathbf{C} \rightarrow \mathbf{O} \leftarrow \mathbf{C} \rightarrow O^{Miller}$~~ size=4

$c_1 - o - c_2$

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

<u>Pruning Condition:</u> $R^K \rightarrow S \leftarrow R^L$

Architecture



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

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

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

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

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

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

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

Discussion Question

- BANKS and DISCOVER share the same goal of enabling keyword searches on relational databases. What are the key differences between the BANKS approach and the DISCOVER approach?
- If you wanted to add keyword search to your database, which system would you rather use?

Thank You

Any Question??