Interactive keyword-based access to large-scale structured datasets

2nd Keystone Summer School
20 July 2016

Dr. Elena Demidova
University of Southampton

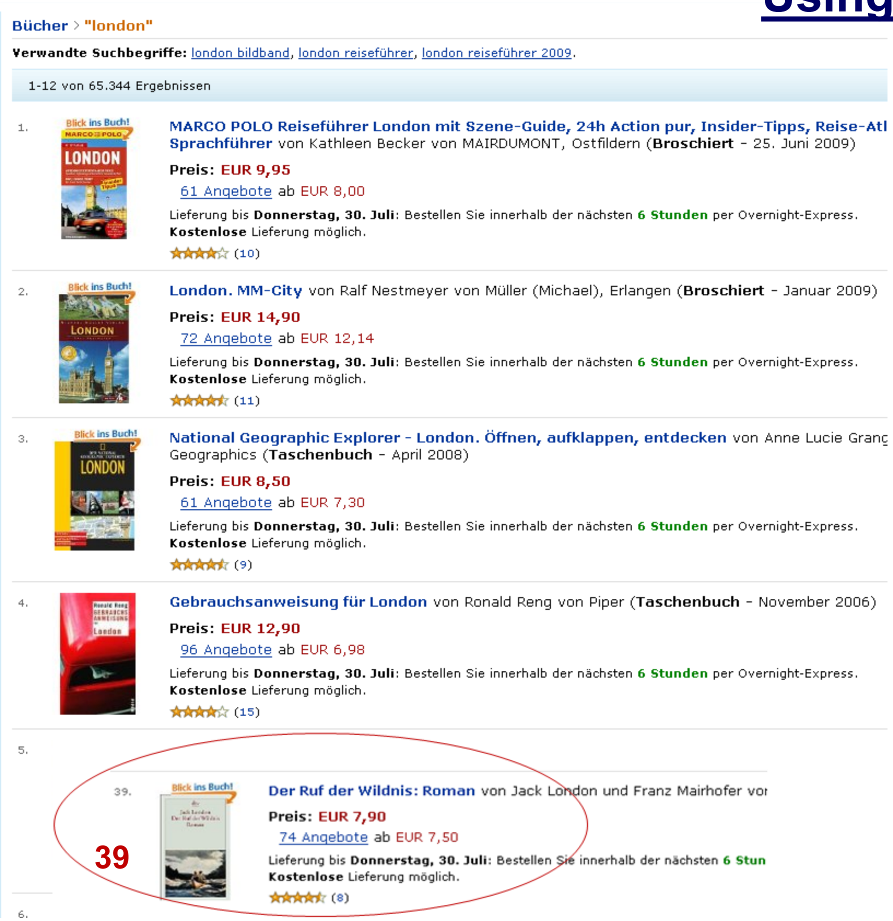
Overview

- **Keyword-based access to structured data**
  - **Usability and expressiveness**
- **Preparation of data for keyword-based access**
  - **Indexing structured data**
- **Interactive query construction**
  - **Building structured queries with user input**

Keyword-based access to structured data

Usability and expressiveness

Keyword-based access to structured data



Bücher > "london"

Verwandte Suchbegriffe: london bildband, london reiseführer, london reiseführer 2009.

1-12 von 65.344 Ergebnissen

1. MARCO POLO Reiseführer London mit Szene-Guide, 24h Action pur, Insider-Tipps, Reise-Atl Sprachführer von Kathleen Becker von MAIRDUMONT, Ostfildern (Broschiert - 25. Juni 2009)
Preis: EUR 9,95
61 Angebote ab EUR 8,00
Lieferung bis Donnerstag, 30. Juli: Bestellen Sie innerhalb der nächsten 6 Stunden per Overnight-Express. Kostenlose Lieferung möglich.
(10)

2. London. MM-City von Ralf Nestmeyer von Müller (Michael), Erlangen (Broschiert - Januar 2009)
Preis: EUR 14,90
72 Angebote ab EUR 12,14
Lieferung bis Donnerstag, 30. Juli: Bestellen Sie innerhalb der nächsten 6 Stunden per Overnight-Express. Kostenlose Lieferung möglich.
(11)

3. National Geographic Explorer - London. Öffnen, aufklappen, entdecken von Anne Lucie Granc Geographics (Taschenbuch - April 2008)
Preis: EUR 8,50
61 Angebote ab EUR 7,30
Lieferung bis Donnerstag, 30. Juli: Bestellen Sie innerhalb der nächsten 6 Stunden per Overnight-Express. Kostenlose Lieferung möglich.
(9)

4. Gebrauchsanweisung für London von Ronald Reng von Piper (Taschenbuch - November 2006)
Preis: EUR 12,90
96 Angebote ab EUR 6,98
Lieferung bis Donnerstag, 30. Juli: Bestellen Sie innerhalb der nächsten 6 Stunden per Overnight-Express. Kostenlose Lieferung möglich.
(15)

5.

39. Der Ruf der Wildnis: Roman von Jack London und Franz Mairhofer vo
Preis: EUR 7,90
74 Angebote ab EUR 7,50
Lieferung bis Donnerstag, 30. Juli: Bestellen Sie innerhalb der nächsten 6 Stun
Kostenlose Lieferung möglich.
(8)

39

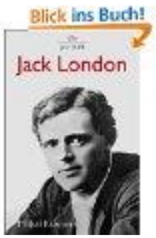
6.

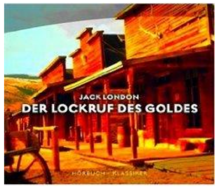
Using structure we could refine the results:

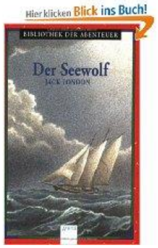
London… What do you mean?

A book title?

An author?







Access to structured data: Search vs. query

Example: DBLP as a relational database containing paper-author relations

Keyword query: $K = \{\text{Michelle}, \text{XML}\}$
Structured query: $Q = \boldsymbol{\sigma}_{\text{michelle} \in \text{name}}(\text{Author}) \bowtie \text{Write} \bowtie \boldsymbol{\sigma}_{\text{xml} \in \text{title}}(\text{Paper})$

TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

(b) Paper

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

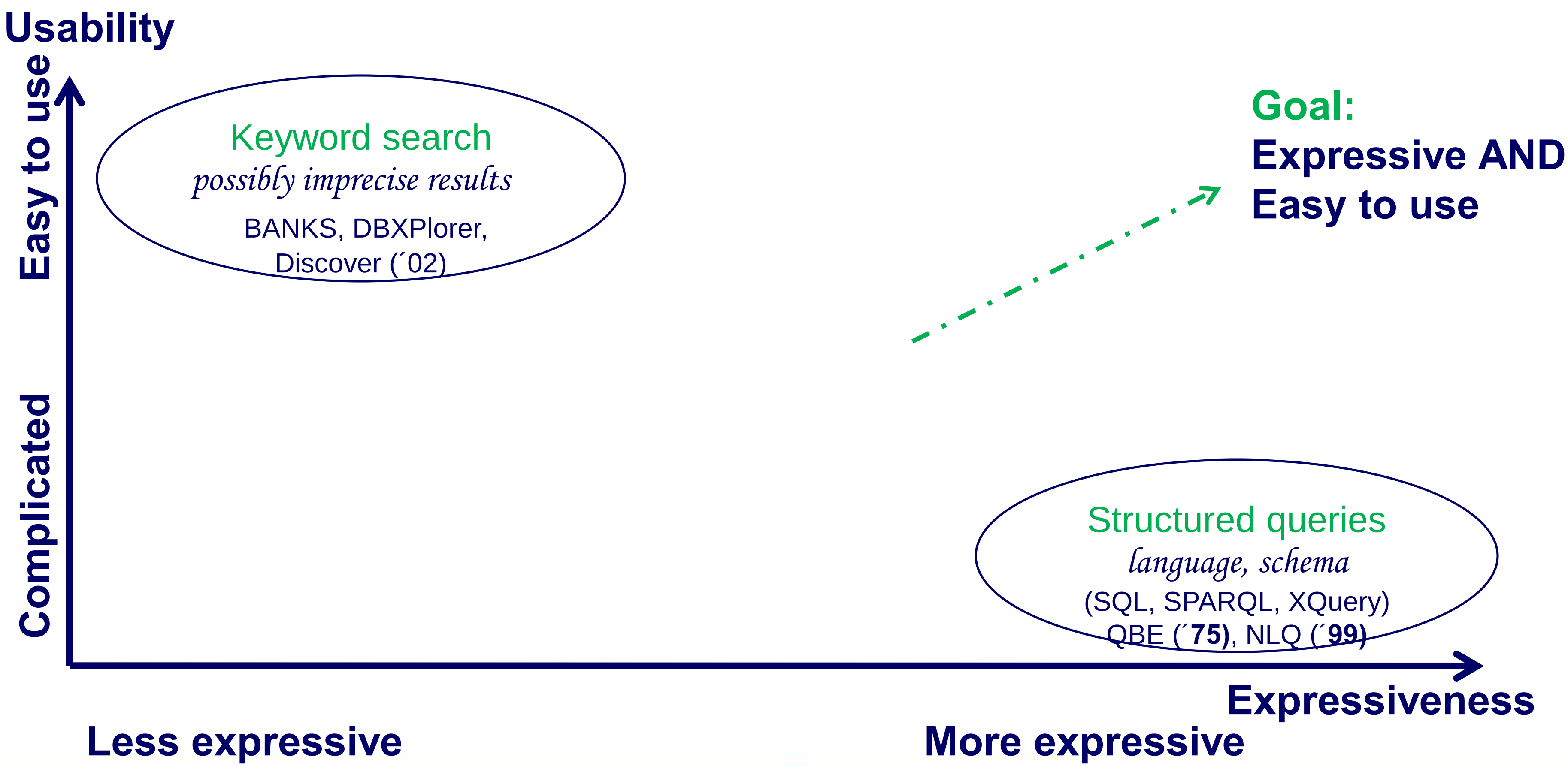
TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite

(Example from [Yu et. al 2009])

adapted from: [Tata et. al 2008]

Database queries: Expressiveness vs. usability



Database queries: Expressiveness vs. usability

- **Database queries:**
  - knowledge of database schema
  - knowledge of query language syntax
- **Keyword search:**
  - Easy-to-use but imprecise
  - Ambiguous: unclear information need
- **Keyword query interpretation:**
  - Automatically translate keyword query in a (most likely) structured query (-ies)

Preparation of data for keyword-based access

Indexing structured data at the example of relational databases

Keyword query semantics

A l-keyword query $K = \{k_1, k_2, \ldots, k_l\}$ –
a set of keywords of size l.

K semantics (typically): search for interconnected tuples that jointly contain $\{k_1, k_2, \ldots, k_l\}$.

How can we find the tuples containing $\{k_1, k_2, \ldots, k_l\}$ in a database?

Full-text search on a specific database attribute

Full-text search on specific attribute is supported by major databases, e.g. using contains predicate:

***contains* (*R.A*, k_i) – the predicate selecting all tuples from a relation *R* that contain keyword k_i in the text attribute *R.A*.**

SELECT * FROM Author WHERE contains(Author.Name, „Michelle“);

String comparison operators (e.g. *like*):

SELECT * FROM Author WHERE Author.Name LIKE '%michelle%';

Problem: need to search in each attribute separately

DB indexing for keyword search

Inverted index using Lucene, Solr, Elasticsearch…

Granularity:

Tuple level:

TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

(b) Paper

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite

Dictionary		Postings		
Michelle	->	Author.a_3	Paper.p_1	...
XML	->	Paper.p_2	Paper.p_3	…

Attribute level:

Dictionary		Postings		
Michelle	->	Author.Name	Paper.Title	...
XML	->	Paper.Title	…	

Differences?

SQL full-text search vs. indexing

Built-in full-text search capabilities are database dependent

Contains predicate can use indexes but is neither flexible, nor not generally available

String comparison operators can require sequential scan (e.g. like operator if the prefix is undefined)

Each textual attribute needs to be queried separately

In the global full-text index, the list of attributes is immediately available

Index construction cost

Storage cost (depends on the index granularity)

Query construction as a way to improve query expressiveness

Building structured queries from keywords

DPLP example and definitions from: [Yu et. al 2009]

From keywords to structured queries: An example

K = {Michelle, XML}

1. Identify tuples / attributes containing keywords

$\sigma_{\text{michelle} \in \text{name}}(\text{Author})$: ***michelle***

$\sigma_{\text{xml} \in \text{title}}(\text{Paper})$: ***xml***

$\sigma_{\text{michelle} \in \text{title}}(\text{Paper})$: ***michelle***

TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

(b) Paper

2. Identify join paths to connect all keywords in the query

$Q = {}_{\text{michelle} \in \text{name}}(\text{Author}) \bowtie \text{Write} \bowtie \sigma_{\text{xml} \in \text{title}}(\text{Paper})$

Other paths?

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite

From keywords to structured queries: An example

K = {Michelle, XML}

$Q = {}_{michelle \in name}(Author) \bowtie Write \bowtie \sigma_{xml \in title}(Paper)$

TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

(b) Paper

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite

The translation *K* - > *Q* requires:

1. **Knowledge of the schema graph (tables, attributes, join paths)**
2. **Knowledge of keyword occurrences**
3. **Efficient algorithms**

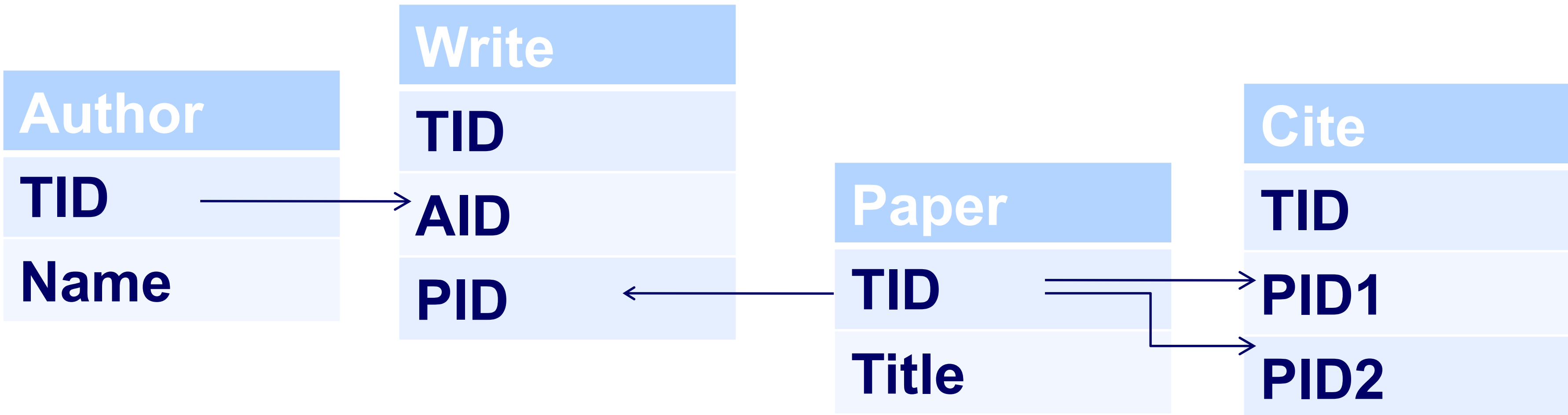
Definitions and notations: The schema graph

Schema graph: a directed graph G_s (V,E)

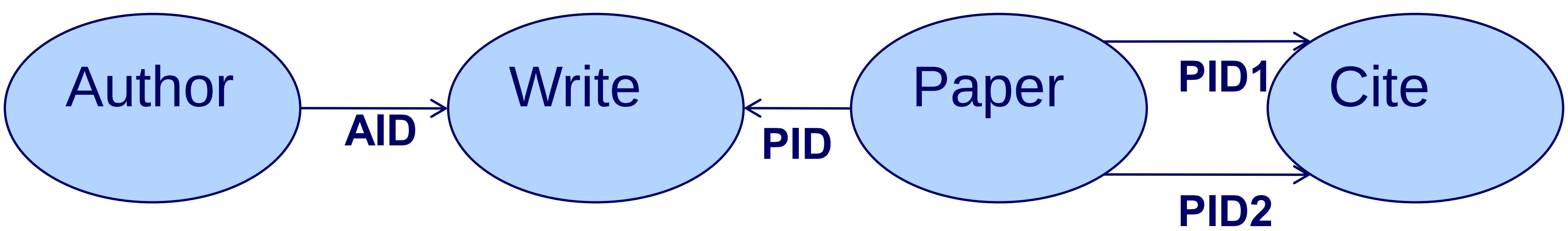
- V – the set of relation schemas $\{R_1, R_2, \dots, R_n\}$. An instance of a relation schema is a set of tuples (i.e. a database table).
- E - the set of edges R_i -> R_j between two relation schemas. An edge is a primary key to foreign key relation.
- TID – primary key attribute (i.e. tuple identifier).

Text attribute – an attribute allowing full-text search.

An example: The DBLP schema graph



A simplified representation of the schema graph:



Definitions and notations: The database graph

The *database graph*: a directed graph G_D (V_t, E_t) on the schema graph *Gs*.

V_t – the set of tuples {t_1, t_2, … , t_n}.

E_t - the set of edges between tuples.

Two tuples t_i and t_j are *connected* if there exists a foreign key (fk) reference t_i -> t_j or t_j -> t_i.

Two tuples t_i, t_j are *reachable* if there exists a sequence of connections between them, e.g. t_i -> t_1 , …. , t_n -> t_j.

The *distance* between two tuples dis(t_i, t_j) is the *minimum* number of connections between t_i, t_j.

An example: The DPLP database graph

TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

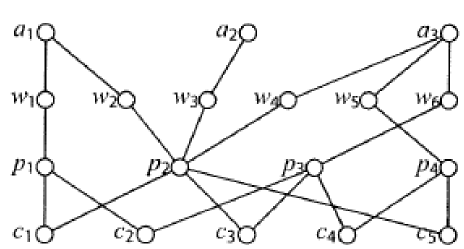
(b) Paper

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite



(e) Tuple Connections

The *distance* between two tuples dis(t_i, t_j) is the *minimum* number of connections between t_i, t_j.

dis (a1, p4)?

Interconnecting keywords: MTJNT

An answer to a l-keyword query is a Minimal Total Joining Network of Tuples (MTJNT).

JNT (Joining Network of Tuples) – a connected tree of tuples. Every two adjacent tuples t_i, t_j in JNT an be joined based on the fk-reference in the schema i.e. either R_i -> R_j or R_j -> R_i (ignoring direction).

TJNT (Total JNT) w.r.t. a l-keyword query K if it contains all keywords of K.

MTJNT (Minimal TJNT) if no tuple can be removed such that JNT remains total.

T_{max} – a size control parameter to define the maximum number of tuples in MTJNT.

Keyword query answers: MTJNT examples

K **= {Michelle, XML}**

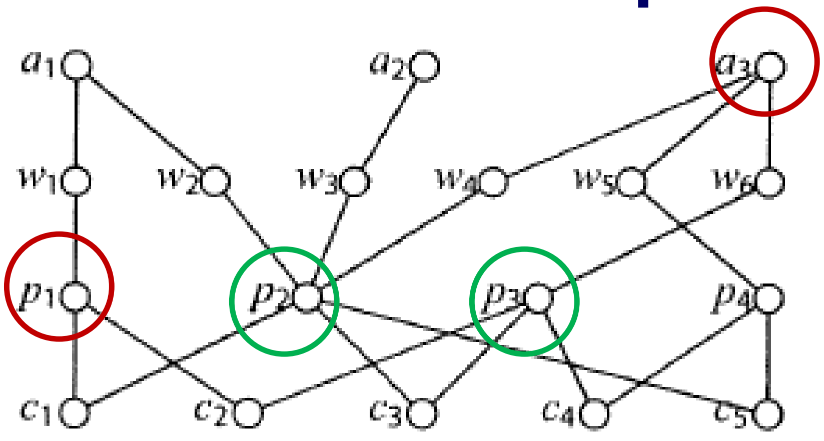
$T_{max} = 5$

contains (a₃, „Michelle“)

contains (p₁, „Michelle“)

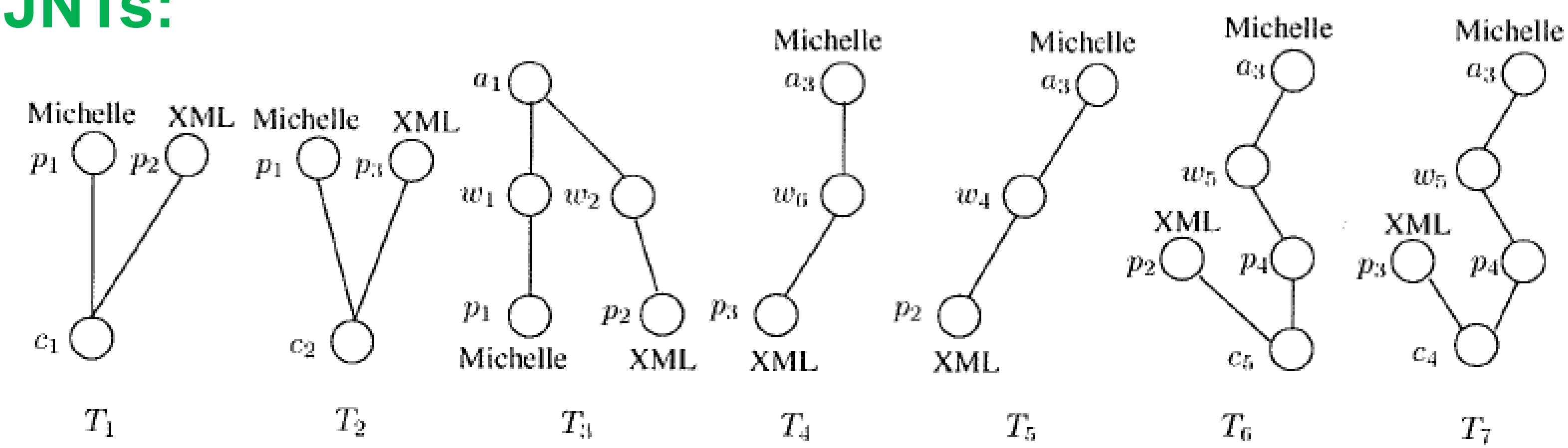
contains (p₂, „XML“)

contains (p₃, „XML“)



(c) Tuple Connections

MTJNTs:



MTJNT issues

Size and scalability:

The data graph is potentially very large, i.e. search is very costly

The search space increases exponentially by adding new data entries

Results semantics and presentation

The results are heterogeneous in terms of structure, i.e. difficult to present and understand

Aggregation / summarization is needed

Generation of structured queries:

Schema graph is much smaller

Structured queries naturally aggregate MTJNTs

Structured queries: Candidate Network (CN)

A ***keyword relation***: a subset $R_i\{K'\}$ of relation R_i that contains a subset K' of keywords from K *(and no other keywords from K)*. The subset can be empty $R_i\{\ \}$.

A ***Candidate Network (CN)*** is a connected tree of keyword relations. Every two adjacent keyword relations R_i, R_j in CN are joined based on the fk-reference in the schema G_s.

CN is total w.r.t. a l-keyword query K if its keyword relations jointly contain all keywords of K.

CN is minimal if no keyword relation can be removed such that CN remains total.

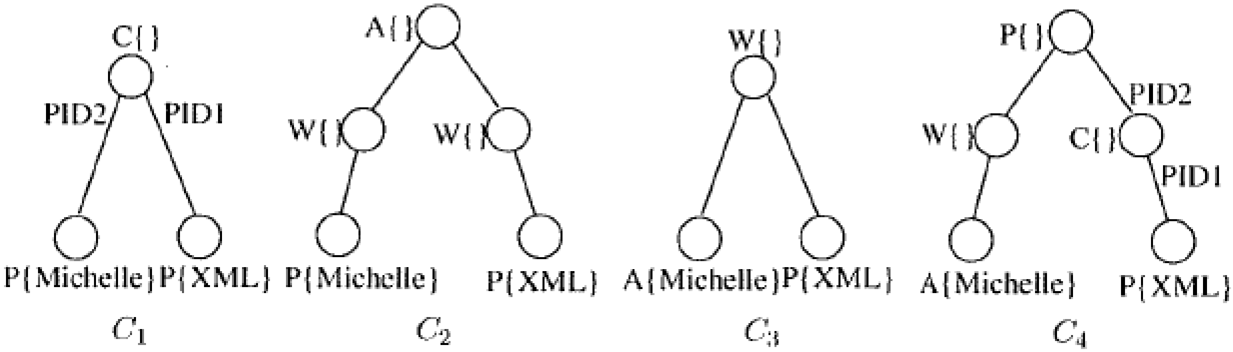
T_{max} – a size control parameter to define the maximum number of keyword relations in CN.

A CN can produce a set of possibly empty MTJNTs. One MTJNTs can be generated by exactly one CN.

CN examples

K = {Michelle, XML}, T_{max} = 5, P{Michelle}, P{XML}, A{Michelle}

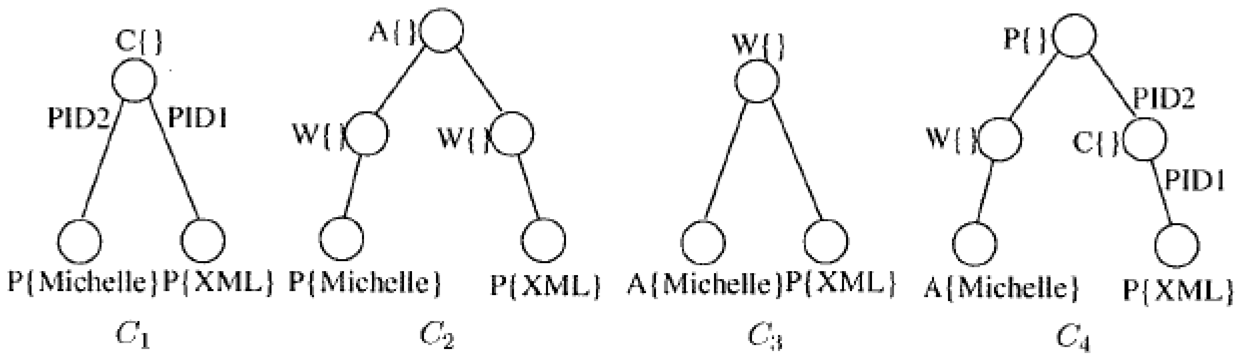
CNs:



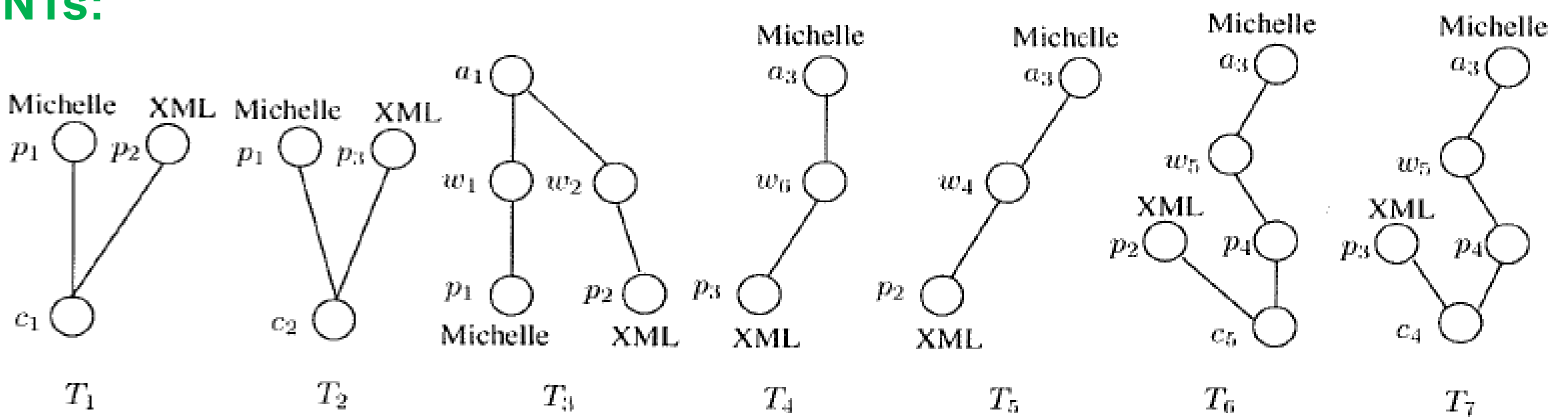
CN examples

K **= {Michelle, XML},** T_{max} **= 5, P{Michelle}, P{XML}, A{Michelle}**

CNs:



MTJNTs:



Which MTJNTs are generated by which CNs?

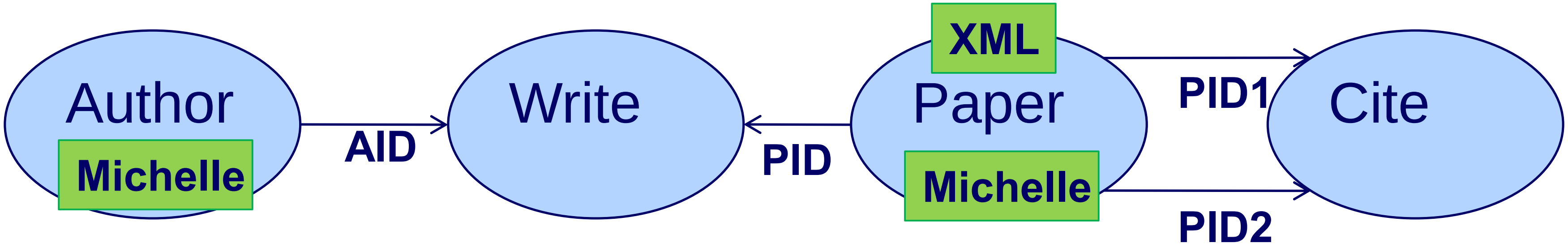
CN generation algorithms

Given are:

1. Keyword query $K = \{k_1, k_2, \ldots, k_l\}$
2. Schema graph G_s
3. The nodes of G_s containing each keyword k_i in K

The Problem: Find the path(s) connecting all $\{k_1, k_2, \ldots, k_l\}$ in G_s (i.e. the structured query(-ies))

Example: K = {Michelle, XML}



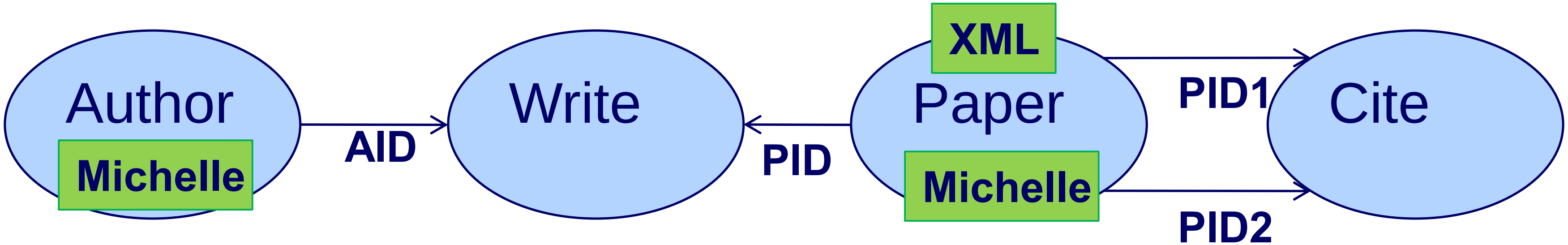
CN generation algorithms

Complexity: similar to the Steiner tree problem - find the shortest interconnect for a given set of objects: NP-complete.

Approximation algorithms:

Iteratively explore the schema graph to construct the paths

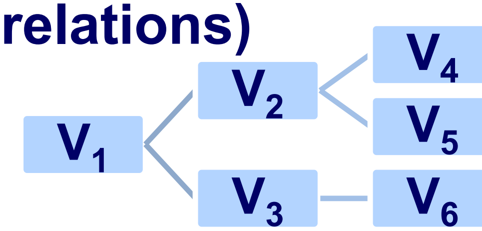
BFS/DFS



Data structures?

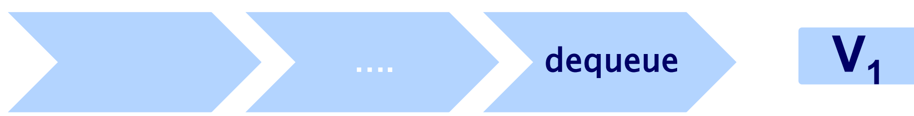
Search algorithms and data structures: BFS

Search on the schema graph G_s (with keyword relations)

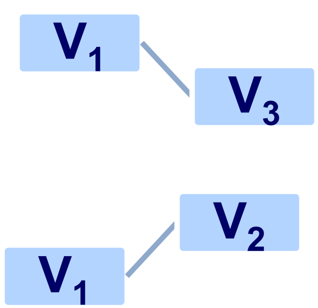


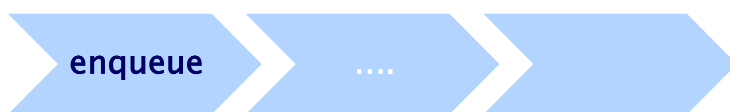
Breadth-First-Search (BFS): queue

Step i:



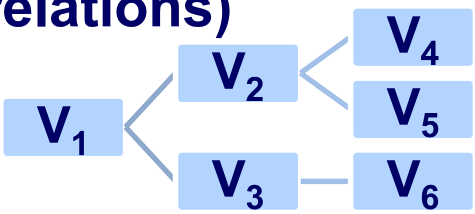
Step i+1:





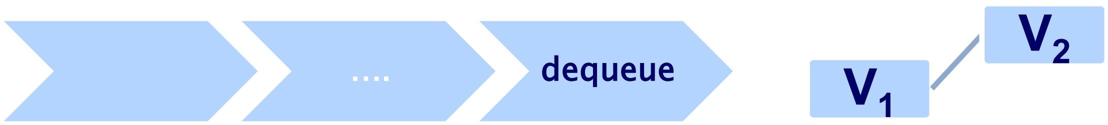
Search algorithms and data structures: BFS

Search on the schema graph G_s (with keyword relations)

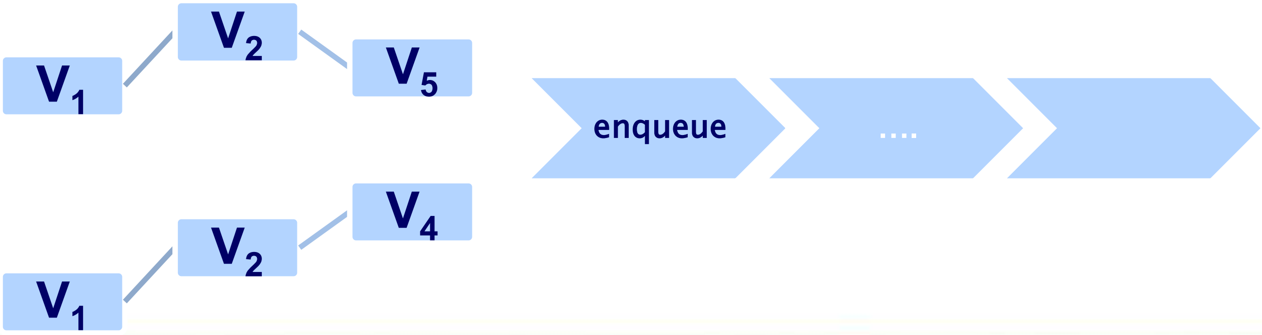


Breadth-First-Search (BFS): queue

Step j:



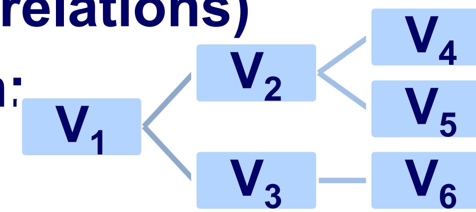
Step j+1:



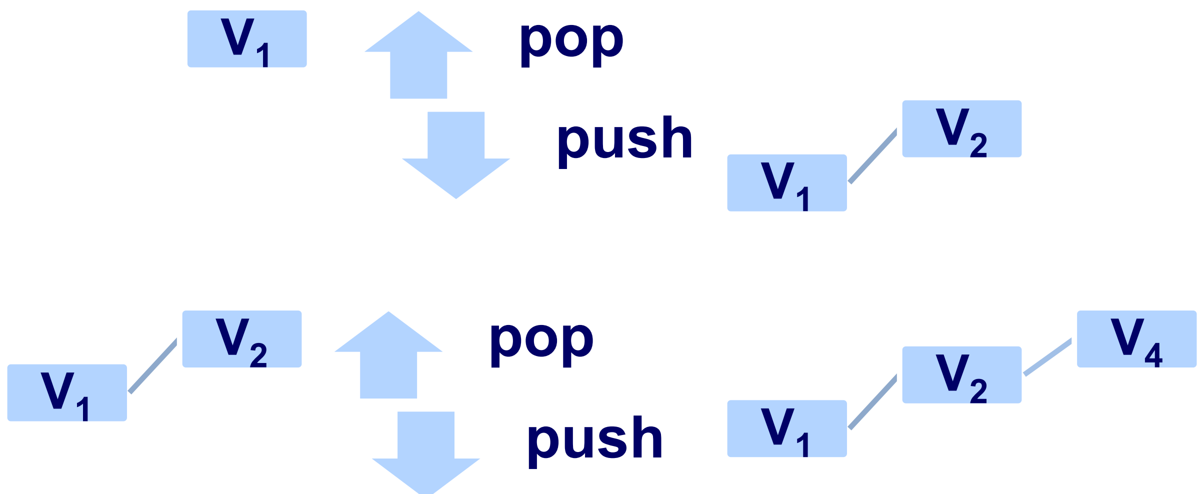
Search algorithms and data structures: DFS

Search on the schema graph G_s (with keyword relations)

Depth First Search (DFS) – for top-k generation:



Stack



algorithm from [Hristidis et. al. 2002]

CN generation algorithm (BFS-based): Discover

Algorithm 1 Discover-CNGen (Q, Tmax, G_S)

Notation: here Q is a keyword query!

Input: an l-keyword query $Q = \{k_1, k_2, \cdots, k_l\}$, the size control parameter Tmax, the schema graph G_S.

Output: the set of CNs $\mathcal{C} = \{C_1, C_2, \cdots\}$.

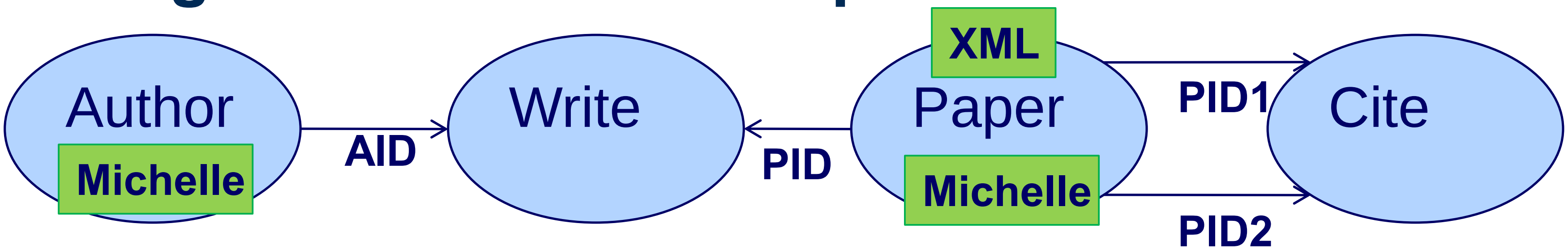
```
Q ← ∅; C ← ∅
for all R_i ∈ V(G_S), K' ⊆ Q do
    Q.enqueue(R_i{K'})
while Q ≠ ∅ do
    T ← Q.dequeue()
    if T is minimal and total and T does not satisfy Rule-1 then
        C ← C ⋃ {T}; continue
    if the size of T < Tmax then
        for all R_i ∈ T do
            for all (R_i, R_j) ∈ E(G_S) or (R_j, R_i) ∈ E(G_S) do
                T' ← T ⋃ (R_i, R_j)
                if T' does not satisfy Rule-2 or Rule-3 then
                    Q.enqueue(T')
return C;
```

Rule 1: duplicate elim.

Rule 2: minimality

Rule 3: avoid cycles

CN generation: An example



Keyword relations:
P{Michelle}, P{XML}, A{Michelle}
…

Algorithm 1 Discover-CNGen (Q, Tmax, G_S)

Input: an l-keyword query $Q = \{k_1, k_2, \cdots, k_l\}$, the size control parameter Tmax, the schema graph G_S.
Output: the set of CNs $\mathcal{C} = \{C_1, C_2, \cdots\}$.

$\mathcal{Q} \leftarrow \emptyset; \mathcal{C} \leftarrow \emptyset$
for all $R_i \in V(G_S)$, $K' \subseteq Q$ **do**
  $\mathcal{Q}.enqueue(R_i\{K'\})$
while $\mathcal{Q} \neq \emptyset$ **do**
  $T \leftarrow \mathcal{Q}.dequeue()$
  if T is minimal and total **and** T does not satisfy Rule-1 **then**
    $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{T\}$; **continue**
  if the size of T < Tmax **then**
    for all $R_i \in T$ **do**
      for all $(R_i, R_j) \in E(G_S)$ **or** $(R_j, R_i) \in E(G_S)$ **do**
        $T' \leftarrow T \bigcup (R_i, R_j)$
        if T' does not satisfy Rule-2 or Rule-3 **then**
          $\mathcal{Q}.enqueue(T')$
return $\mathcal{C}$;

enqueue: P{Michelle}, P{XML}, A{Michelle}
dequeue: T_1 <- A{Michelle}
expand: T_2 <- A{Michelle} $\bowtie$ **W{}**
enqueue: T_2
…
dequeue: T_2 <- A{Michelle} $\bowtie$ **W{}**
expand: T_3 <- A{Michelle} $\bowtie$ **W{}** $\bowtie$ **P{XML}**
enqueue: T_3
…
dequeue: T_3, check if T_3 is minimal and total, add T_3 to the result

CN generation: Complexity and optimizations

Complexity factors:

- **Size of the schema graph G_s – the number of nodes and edges**
- **Maximum number of joins (T_{max})**
- **Size of the keyword query (l)**

The number of CNs grows exponentially with these factors.

Algorithm optimizations:

- **Avoid generation of duplicate CNs by defining the expansion order**
- **Generate only the top-k CNs**
- **…**

CN and MTJNT ranking factors

Ranking can be performed at CN and MTJNT levels

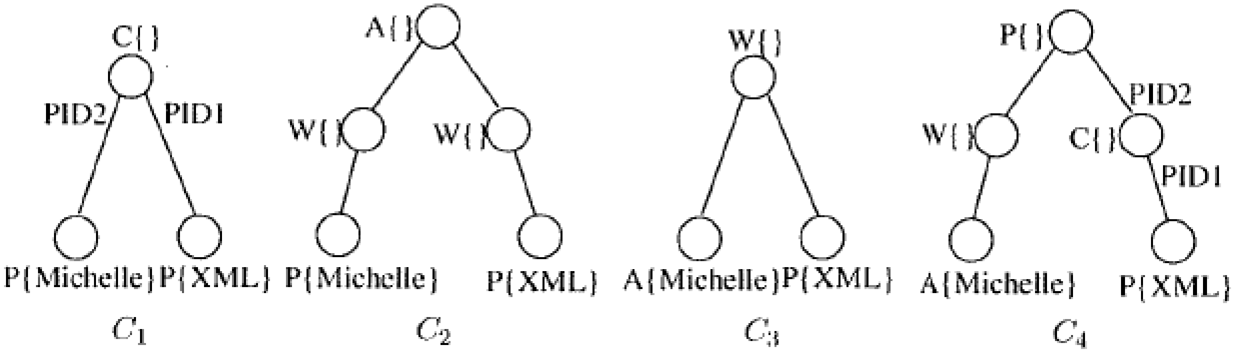
Typical ranking factors include:

- Size of the CN / tuple tree – preference to the short paths
- IR-Style factors
  - Frequency-based keyword weights
  - Keyword selectivity (IDF)
  - Length normalizations
- Global attribute weight in a database (PageRank / ObjectRank)

Typically, the factors are combined

Ranking query interpretations: An example

Rank the following CNs using the size factor:



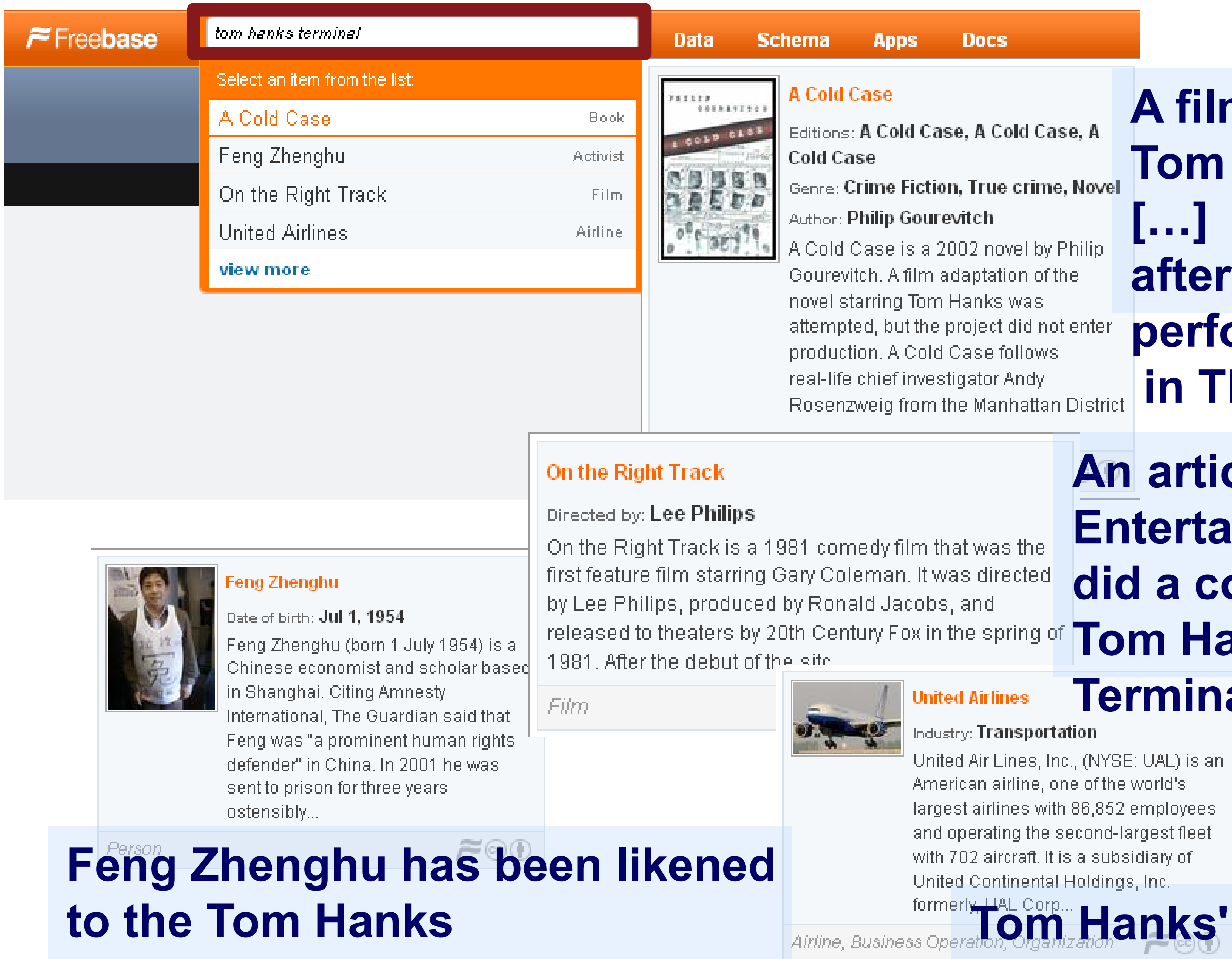
Interactive query construction

Building structured queries with user input

Query construction for large scale databases

- Freebase:
  - 22 millions entities, more than 350 millions facts
  - more than 7,500 relational tables
  - about 100 domains
  - Wikipedia, MusicBrainz, …
  - part of the LOD cloud

- **Goal:**
  - **Enable efficient and scalable query construction solutions for large scale data**



A film adaptation starring Tom Hanks was attempted […] after the actor's performances in The Terminal (2004)

An article in Entertainment Weekly did a comparison to the Tom Hanks film The Terminal

Feng Zhenghu has been likened to the Tom Hanks character in The Terminal

Tom Hanks' character Viktor Navorski is stuck at New York's JFK airport in the United terminal in The Terminal

Structured MQL query for „Tom Hanks Terminal“

```
[{
  "!pd:/film/actor/film": [{
     "name": "Tom Hanks"
     "type": "/film/actor"}],
  "film":[{
    "name" : "The Terminal"
    "type" : "/film/film"}],
  "character":{
    "name" : null }
  "type": "/film/performance"
 }]
```

http://www.freebase.com/query

Requires prior knowledge of:

✓**Schema: above 1000 entity types** (relational tables)

✓**Specialized query language: MQL**

Query interpretation techniques

- Automatic keyword query interpretation:
  - Automatically translate keyword query in the (most likely) structured query (-ies)
  - No one size fits all – no perfect ranking for every query and every user
  - If ranking fails, navigation cost can be inacceptable
    - too many interpretations / search results
- Interactive query refinement
  - Goal: Enable users to incrementally refine a keyword query into the intended interpretation on the target database in a minimal number of interactions

Query interpretation

A query interpretation consists of:

- A set of **keyword interpretations** ***I*** that map a keyword to a value of an attribute (also interpretations as an attribute or table name are possible)

$\sigma_{2001 \in year}(Movie){:}\,2001$ $\sigma_{cruise \in name}(Actor){:}\,cruise$

$\sigma_{hanks \in name}(Actor){:}\,hanks$

- A **query template** ***T***

$T = \sigma_{? \in name}(Actor) \bowtie Acts \bowtie \sigma_{? \in year}(Movie) \bowtie Acts \bowtie \sigma_{? \in name}(Actor)$

Query interpretation

Dictionary	Postings			
hanks	actor.name	director.name	....	
cruise	actor.name	company.name	location.name	...

K="hanks cruise 2001"

$\sigma_{\text{hanks} \in \text{name}}(\text{Actor})$: *hanks*

$\sigma_{\text{cruise} \in \text{name}}(\text{Actor})$: *cruise*

$\sigma_{2001 \in \text{year}}(\text{Movie})$: *2001*

+

$$T = \sigma_{? \in \text{name}}(\text{Actor}) \bowtie \text{Acts} \bowtie \sigma_{? \in \text{year}}(\text{Movie}) \bowtie \text{Acts} \bowtie \sigma_{? \in \text{name}}(\text{Actor})$$

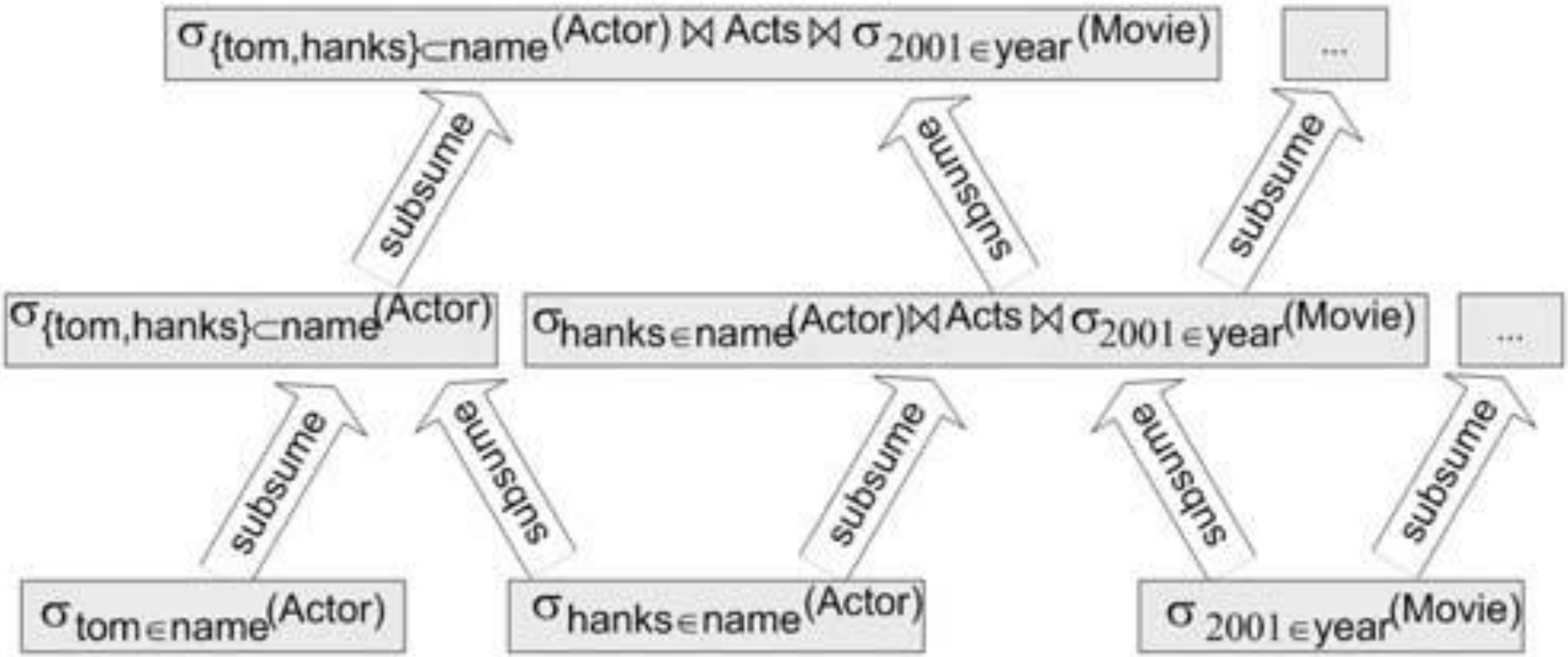
=

$$Q = \underbrace{\underbrace{\sigma_{\text{hanks} \in \text{name}}(\text{Actor}) \bowtie \text{Acts} \bowtie \sigma_{2001 \in \text{year}}(\text{Movie})}_{\text{partial interpretation of } K\text{, sub-query of } Q} \bowtie \text{Acts} \bowtie \sigma_{\text{cruise} \in \text{name}}(\text{Actor})}_{\text{complete interpretation of } K \text{ (structured query)}}$$

- ➢ interpretation space of *K*

Query hierarchy

K = "Tom Hanks 2001"



Query construction options (QCO)

Idea: use partial interpretations (sub-queries) as user interaction items (QCO)

Problem: large number of queries – and sub-queries (QCOs)

$\sigma_{2001 \in year}(Movie)$: *2001*

$\sigma_{hanks \in name}(Actor)$: *hanks*

$\sigma_{cruise \in name}(Actor)$: *cruise*

$Q' = \sigma_{hanks \in name}(Actor) \bowtie Acts \bowtie \sigma_{2001 \in year}(Movie)$

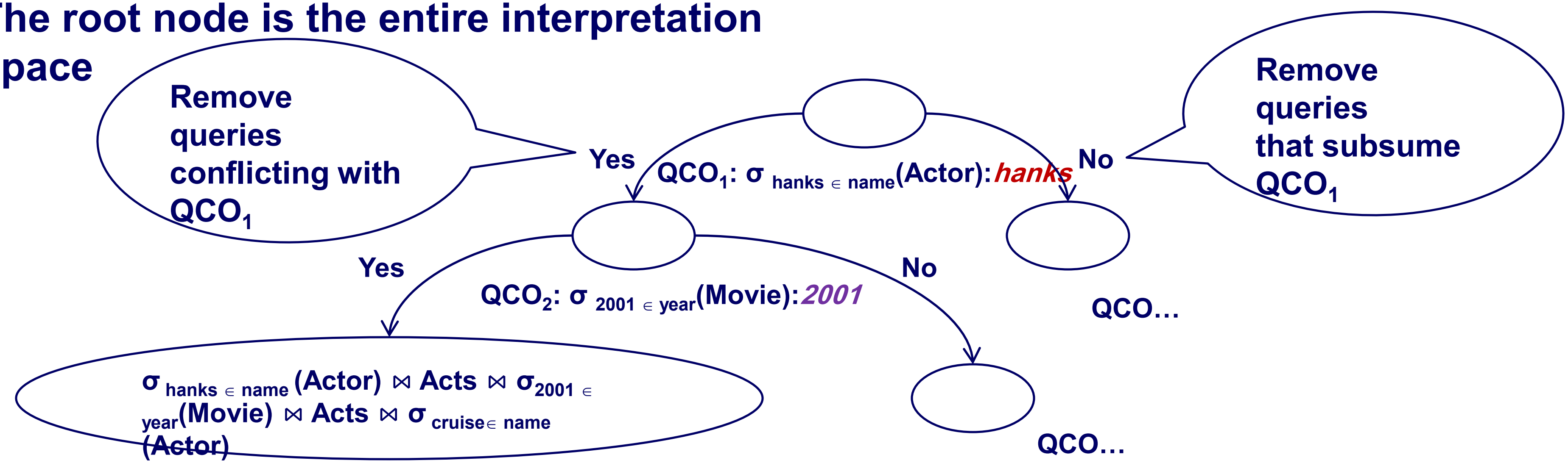
$\sigma_{2001 \in year}(Movie) \bowtie Acts \bowtie \sigma_{cruise \in name}(Actor)$

How to select a QCO to present to the user?

Query construction plan (*QCP*) as a binary tree

Idea: use sub-query relations to organize the options in a (binary) tree structure

The root node is the entire interpretation space



A leaf node is a single complete query interpretation

Problem: How to find an optimal QCP?

Defining a cost function for QCP

Idea: define a cost function

Take query probability into account

Construction of the most likely queries should not incur much cost

$$Cost(QCP) = \sum_{leaf \in QCP} depth(leaf) \times P(leaf)$$

Given a keyword query *K*, how to compute the probability of leaf nodes (i.e. complete query interpretations of *K*)?

K (a keyword query) = {hanks, 2001, cruise}

Q (a leaf node of QCP) =

$\sigma_{hanks \in name}(Actor) \bowtie Acts \bowtie \sigma_{2001 \in year}(Movie) \bowtie Acts \bowtie \sigma_{cruise \in name}(Actor)$

P(leaf) = P(*Q*|*K*): the conditional probability that, given *K*, *Q* is the user intended complete interpretation of *K*.

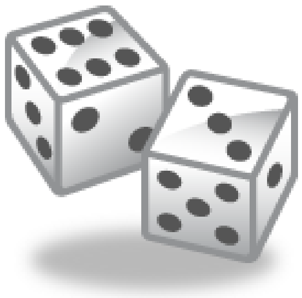
Query interpretation: assumptions

Assumption 1 (Keyword Independence): Assume that the interpretation of each keyword in a keyword query is independent from the other keywords.

Assumption 2 (Keyword Interpretation Independence): Assume that the probability of a keyword interpretation is independent from the part of the query interpretation the keyword is not interpreted to.

Probability of a query interpretation

$$P(Q \mid K) = P\left(I, T \mid K\right)$$



- *I* is the set of keyword interpretations $\{A_i : k_i\}$ in *Q*

$\sigma_{2001 \in year}(Movie){:}\,2001$ $\quad\quad$ $\sigma_{cruise \in name}(Actor){:}\,cruise$

$\sigma_{hanks \in name}(Actor){:}\,hanks$

- *T* is the template of *Q*

$T = \sigma_{? \in name}(Actor) \bowtie Acts \bowtie \sigma_{? \in year}(Movie) \bowtie Acts \bowtie \sigma_{? \in name}(Actor)$

$$P(Q \mid K) \propto \left(\prod_{k_i \in K} P\left(A_i : k_i \mid A_i\right) \right) \times P(T)$$

Estimates for $P(T)$ and $P(A_i{:}k_i \mid A_i)$?

Probability of a keyword interpretation

- We model the formation of a query interpretation as a random process.
- For an attribute A_i, this process randomly picks one of its instances a_j and randomly picks a keyword k_i from that instance to form the expression $\sigma_{k_i} \in A_i$.
- Then, the probability of $P(\sigma_{k_i} \in A_i | \sigma_? \in A_i)$ is the probability that $\sigma_{k_i} \in A_i$ is formed through this random process.

Example:

$T = \boxed{\sigma_{? \in name}(Actor)} \bowtie Acts \bowtie \sigma_{? \in year}(Movie) \bowtie Acts \bowtie \sigma_{? \in name}(Actor)$

$Q = \boxed{\sigma_{hanks \in name}(Actor)} \bowtie Acts \bowtie \sigma_{2001 \in year}(Movie) \bowtie Acts \bowtie \sigma_{cruise \in name}(Actor)$

$P(\boxed{\sigma_{hanks \in name}(Actor)} | \boxed{\sigma_? \in name (Actor)})$

Probability of a keyword interpretation

$P(\sigma_{k_i} \in A_i | \sigma_? \in A_i)$ can be estimated using Attribute Term Frequency (ATF):

$$ATF(k_i, A_i) = (TF(k_i, A_i)+\alpha) / (N_{A_i} + \alpha * B)$$

$ATF(k_i, A_i)$ - the normalized keyword frequency of k_i in A_i

N_{A_i} – the number of keywords in A_i

α - a smoothing parameter (typically $\alpha = 1$: Laplace smoothing)

B – the vocabulary size

Probability of a query template

$$P(T) = (\#occurences(T) + \alpha) \,/\, (N + \alpha * B)$$

#occurences(T) - number of queries in the log using *T* as a template

N - total number of queries in the log

α - smoothing parameter, typically set to 1

B – a constant

When the query log is absent or is not sufficient, we assume that all query templates are equally probable.

Challenges in query interpretation

- Inefficient QCOs
  - Too many keyword interpretations
  - A keyword interpretation subsumes a small proportion of the I-space, more general QCOs are needed

- Very large interpretation space
  - The number of subgraphs of the schema graph grows very sharply with the size of the schema graph. The occurrences of keywords are more numerous in a larger database. Too many query interpretations.
  - Existing query interpretation approaches rely on a completely materialized interpretation space. This is no longer feasible.
  - Need to enable incremental materialization of the interpretation space

Query construction algorithm

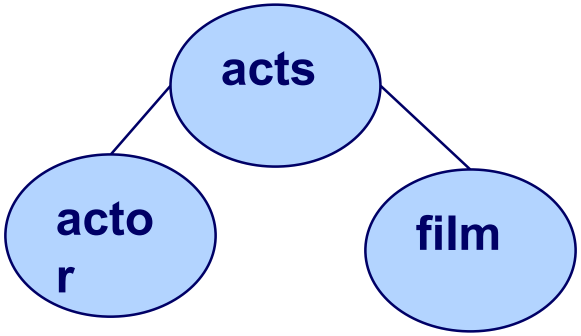
- Query hierarchy can become very large
- Use greedy algorithms
- Expand query hierarchy incrementally
- Use a threshold to restrict the size of the top level
- Select the QCO to be presented to the user based on Information Gain (IG)
- IG can be computed using probability of query interpretation

Query-based QCOs

- **Keyword as schema terms or attribute values**
  - **actor.name: hanks (Hanks is in the actor’s name)**

hanks →	**actor.name**	**director.name**	**….**	
terminal →	**film.name**	**company.name**	**location.name**	**…**

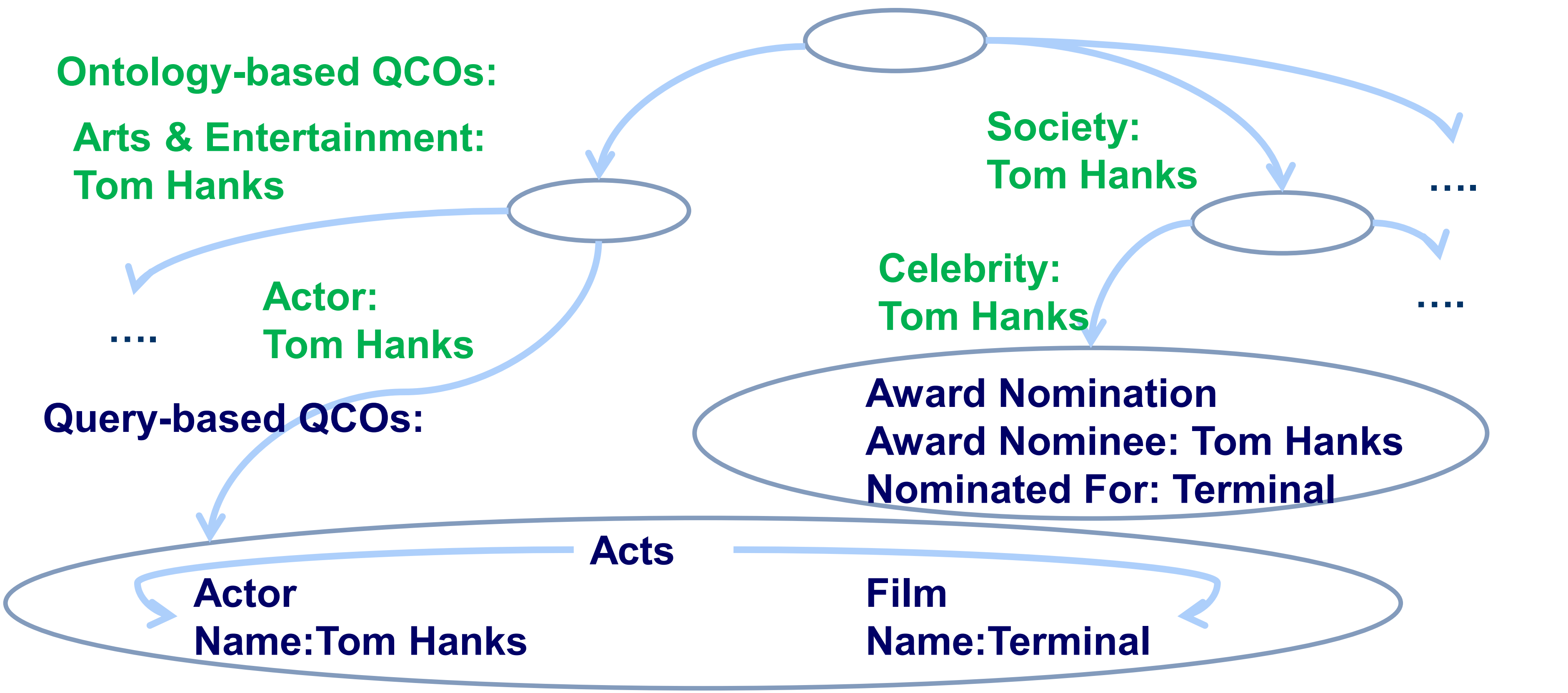
- **Joins using pk-fk relationships in the schema graph**
  - **actor.name: hanks – acts – film.name: terminal**



Ontology-based QCOs

- Freebase domain hierarchy
  - Arts & Entertainment, Society
- External ontologies
  - E.g. YAGO+F mapping between YAGO and Freebase
  - Person, Location, Object

FreeQ query hierarchy example



The arrows represent sub-query relationship

A measure of QCO efficiency

Entropy of the query interpretation space:

$$H(\zeta) = -\sum_{I \in \zeta} P(I) \times log_2 P(I)$$

Expected information gain of a QCO as entropy reduction:

$$IG(O) = H(\zeta) - H(\zeta|O) = H(O)$$

Entropy of O computed using P(O):

$$H(O) = -P(O) log_2 P(O) - P(\neg O) log_2 P(\neg O)$$

Probability estimation for QCOs

Probability of a QCO using probabilities of the subsumed query interpretations:

$$P(O) = \sum_{I \in \zeta(O)} P(I)$$

Estimation of QCO probability using materialized part of the query hierarchy:

$$P(o) = \frac{\sum_{\zeta(s) \subset \zeta(o)} P(s)}{\sum_{\zeta(s) \subset \zeta(o)} P(s) + \sum_{\zeta(s) \cap \zeta(o) = \varnothing} P(s)}$$

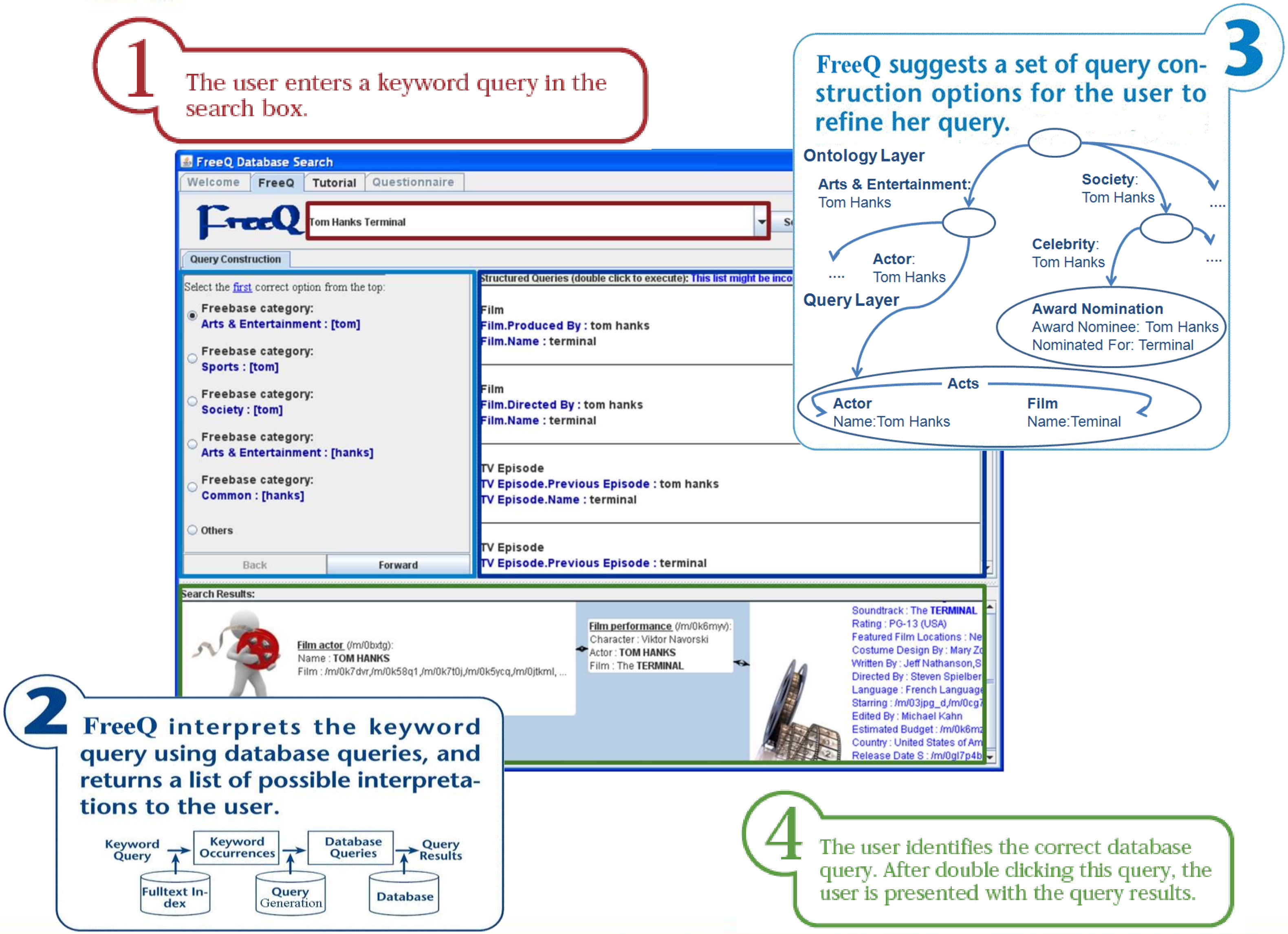
Efficient hierarchy traversal

- ✓ **Query initialization:**
  - ✓ **Path indexing: for each table, index all paths leading to keywords within radius *r*/2 (bi-directional):**
  - ✓ **Is independent of keyword query length**

$$T * avg(E_t)^{r/2}$$

- ✓ **User interaction:**
  - ✓ **Use path index to materialize QCOs and query interpretations incrementally by BF-k and DF-k**
  - ✓ **Start expansion with the most probable QCOs**
  - ✓ **Thresholds, time limits**

[Demidova et. al 2013]



Discussion

- ✓ Interactive query construction can enable efficient and scalable query solutions for large scale data
- ✓ It can involve ontologies to summarize and enrich database schema using abstract concepts (e.g. using YAGO ontology)
- ✓ Query interpretation space on large scale data can and should be materialized incrementally

References

[Yu et. al 2009] Jeffrey Xu Yu, Lu Qin, Lijun Chang. Keyword Search in Databases. Synthesis Lectures on Data Management. Morgan & Claypool Publishers. 2009. (Chapter 2.)

[Qin et. al 2009] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. Keyword search in databases: the power of RDBMS. In Proc. of the 2009 ACM SIGMOD

[Hristidis et. al 2002] Vagelis Hristidis and Yannis Papakonstantinou. Discover: keyword search in relational databases. In Proc. of VLDB 2002.

[Demidova et. al 2012] Elena Demidova, Xuan Zhou, Wolfgang Nejdl: A Probabilistic Scheme for Keyword-Based Incremental Query Construction. IEEE Trans. Knowl. Data Eng. 24(3): 426-439 (2012).

[Demidova et. al 2013] Elena Demidova, Xuan Zhou, and Wolfgang Nejdl. 2013. Efficient query construction for large scale data. In *Proc. of the ACM SIGIR* 2013.

Further reading

[Tata et. al 2008] Sandeep Tata and Guy M. Lohman. SQAK: doing more with keywords. In Proc. of the 2008 ACM SIGMOD.

[Nandi et. al 2009] Nandi, A., Jagadish, H.V.: Qunits: queried units in database search. In CIDR (2009).

[Jayapandian et. al 2008] Magesh Jayapandian and H. V. Jagadish. 2008. Expressive query specification through form customization. In Proc. of the EDBT 2008.

[Chu et. al 2009] Eric Chu, Akanksha Baid, Xiaoyong Chai, AnHai Doan, and Jeffrey Naughton. 2009. Combining keyword search and forms for ad hoc querying of databases. In Proc. of the 2009 ACM SIGMOD.

Questions, Comments?

Dr. Elena Demidova
Web and Internet Science Group
University of Southampton

e.demidova@soton.ac.uk