EDBT ICDT EDINBURGH 2022

Tutorial: Answering Unions of Conjunctive Queries with Ideal Time Guarantees

Nofar Carmeli

ENS, PSL University

Focus

- Data: general relational databases
- Tasks: enumeration & related tasks
- Queries: joins → CQs → UCQs
- Tractability: “ideal” time guarantees
- Goal: classify cases into tractable/not

Content

Join Queries	• Acyclicity • Complexity measures • Model subtleties • Lower bounds • Self-joins
Conjunctive Queries	• Handling projections • Free-connexity • Lower bounds
Unions of Conjunctive Queries	• Easy ∪ easy • Easy ∪ hard • Cheater’s Lemma • Linear partial time • Hard ∪ hard
Related Problems	• Ordered enumeration • Direct & ranked access • Direct access for UCQs • Connections between problems

Join Queries

Join Query Example

authors:

Author	Affiliation	Title
Shay Gershtein	Tel Aviv Univ.	On the Hardness…
Uri Avron	Tel Aviv Univ.	On the Hardness…
Florent Capelli	Univ. Lille	Linear programs…
Nicolas Crosetti	Univ. Lille	Linear programs…

schedule:

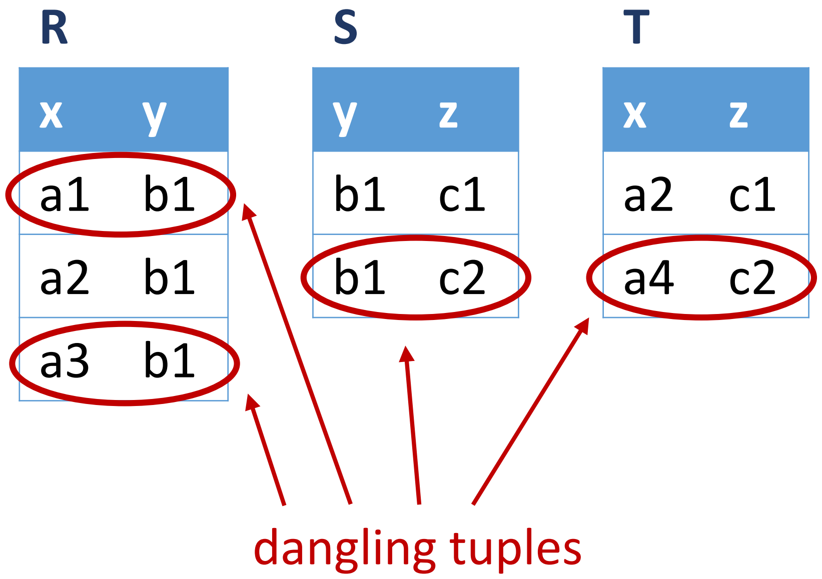
Title	Day
On the Hardness of…	Tue
Linear programs…	Tue
Answering Unions…	Tue
On an Information…	Wed

Author	Affiliation	Title	Day
Shay Gershtein	Tel Aviv Univ.	On the Hardness…	Tue
Uri Avron	Tel Aviv Univ.	On the Hardness…	Tue
Florent Capelli	Univ. Lille	Linear programs…	Tue
Nicolas Crosetti	Univ. Lille	Linear programs…	Tue

$$\mathrm{Q}_1(\mathrm{Author}, \mathrm{Affiliation}, \mathrm{Title}, \mathrm{Day}) \leftarrow \\ \mathrm{authors}(\mathrm{Author}, \mathrm{Affiliation}, \mathrm{Title}), \mathrm{schedule}(\mathrm{Title}, \mathrm{Day})$$

Challenges

- Many answers
- Many intermediate answers



R

x	y
a1	b1
a2	b1
a3	b1

S

y	z
b1	c1
b1	c2

T

x	z
a2	c1
a4	c2

dangling tuples

$$Q_1(x,y,z) \leftarrow R(x,y), S(y,z)$$

x	y	z
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

$$Q_2(x,y,z) \leftarrow R(x,y), S(y,z), T(x,z)$$

x	y	z
a2	b1	c1

Acyclicity

- A query that has a join tree is called acyclic

2. tree

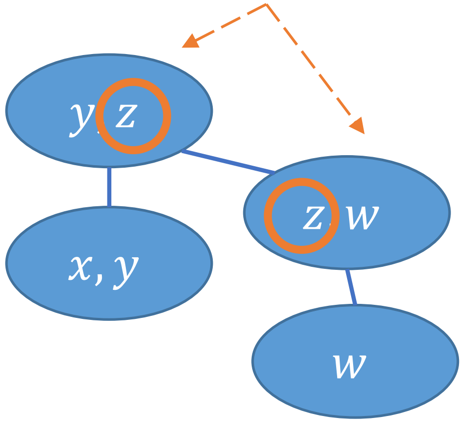
3. For every variable:
the nodes containing it form a subtree

1. a node for every atom

Query: $Q_1(x, y, z, w) \leftarrow R(x, y), S(y, z), T(z, w), U(w)$

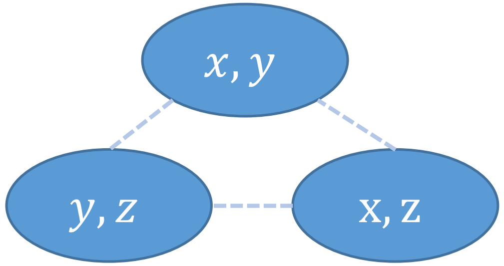
acyclic

Join Tree:



cyclic

Query: $Q_2(x, y, z) \leftarrow R(x, y), S(y, z), T(x, z)$

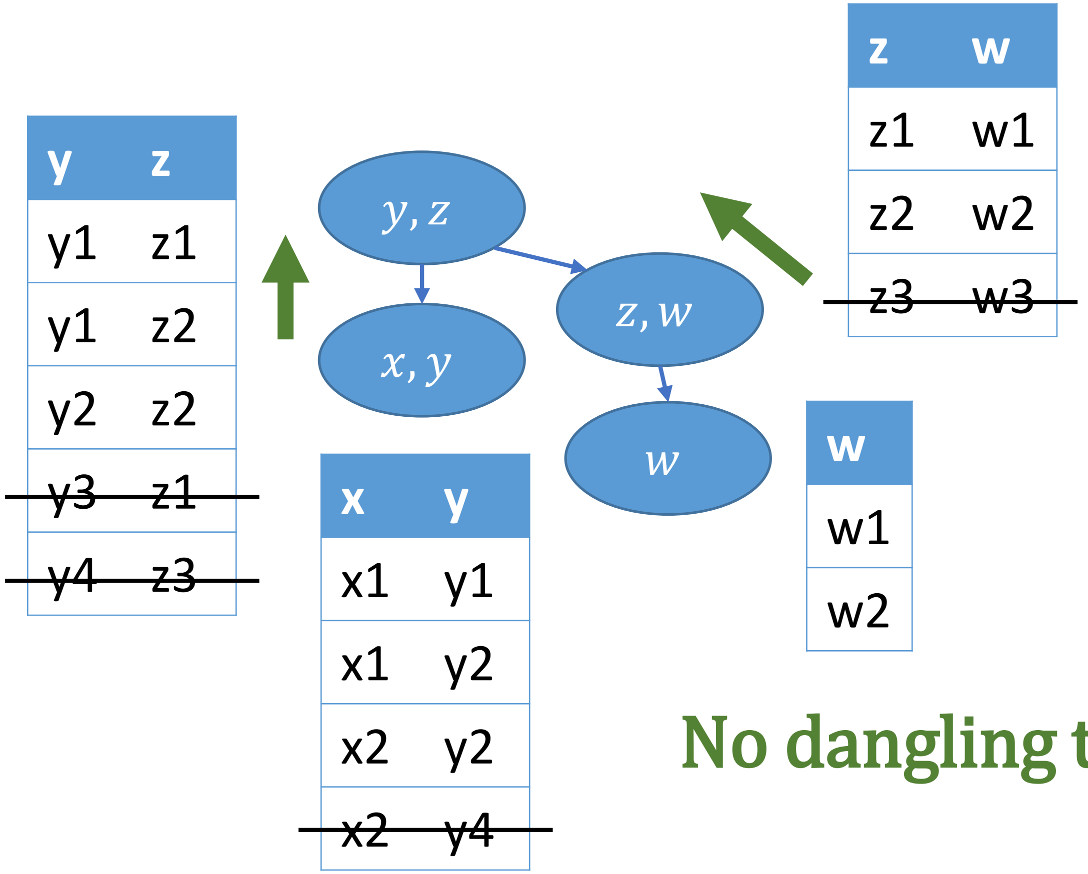


Acyclic Joins

[Yannakakis 81]

- An efficient algorithm for acyclic joins
  1. Find a join tree and set a root
  2. Remove dangling tuples
  3. Join

1. Leaf-to-root:
   $r_{parent} \leftarrow r_{parent} \ltimes r_{child}$
2. Root-to-leaf:
   $r_{child} \leftarrow r_{child} \ltimes r_{parent}$



y	z
y1	z1
y1	z2
y2	z2
~~y3~~	~~z1~~
~~y4~~	~~z3~~

x	y
x1	y1
x1	y2
x2	y2
~~x2~~	~~y4~~

z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

w
w1
w2

No dangling tuples!

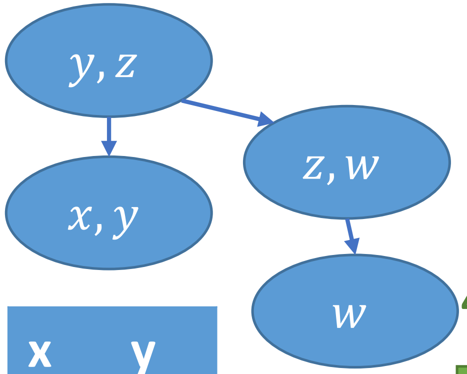
Acyclic Joins

- An efficient algorithm for acyclic joins
  1. Find a join tree and set a root
  2. Remove dangling tuples
  3. Join

y	z	x	w
y1	z1	x1	w1
y1	z2	x1	w2
y2	z2	x1	w2
y2	z2	x2	w2

1

y	z
y1	z1
y1	z2
y2	z2
~~y3~~	~~z1~~
~~y4~~	~~z3~~



2

x	y
x1	y1
x1	y2
x2	y2
~~x2~~	~~y4~~

3

z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

4

w
w1
w2

```
for t1 in relation 1:
    for t2 in relation 2 matching t1:
        for t3 in relation 3 matching t1,t2:
            for t4 in relation 4 matching t1,t2,t3:
                output t1,t2,t3,t4
```

Complexity Guarantees

- Data complexity
  - input = database
  - query size = constant

- Possibly: output $\gg$ input

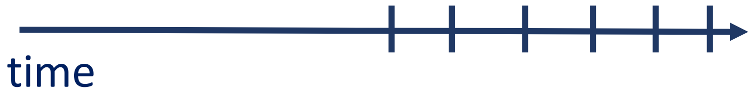
- Minimal requirements:
  - Linear time (to read input)
  - Constant time per answer (to print output)

Complexity Guarantees

- Worst-case-optimal total time [Atserias, Grohe, Marx; FOCS 08]
  - Linear in input + worst-case output

- Instance-optimal total time (also relevant)
  - Linear in input + output

- Enumeration ("ideal"; our focus)
  - Preprocessing linear in input
  - delay constant

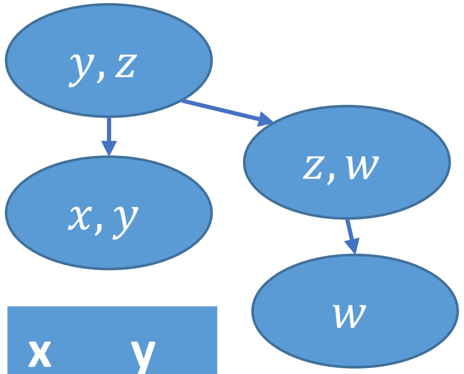


Acyclic Joins

[Yannakakis 81]

- An efficient algorithm for acyclic joins
  1. Find a join tree and set a root (constant time)
  2. Remove dangling tuples (linear preprocessing)
  3. Join (constant delay)

y	z
y1	z1
y1	z2
y2	z2
~~y3~~	~~z1~~
~~y4~~	~~z3~~



x	y
x1	y1
x1	y2
x2	y2
~~x2~~	~~y4~~

w
w1
w2

z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

RAM Model Subtleties

- Constant time in the RAM model, what does it mean?
- Assumptions:
  - Length of registers: $\theta(\log n)$
  - Basic operations in $O(1)$
  - Available memory: $O(n^c)$ / $O(n)$
  - Modified memory: everything / $O(n)$
  - Modified memory during enumeration: everything / … / $O(1)$
- Implications:
  - Domain values $\leq n^c$
  - Sorting the input in $O(n)$
    - Radix Sort handles k integers, each bounded by n^c, in time $O(ck + cn)$
  - If $O(n^c)$ available memory,
    - Lookup table with k elements: construction in $O(k)$, search in $O(1)$

"saves"
log factors

n = size of input database

RAM Model Subtleties

- Constant time in the RAM model, what does it mean?
- Assumptions:
  - Length of registers: $\theta(\log n)$
  - Basic operations in $O(1)$
  - Available memory: $\boldsymbol{O(n^c)}$ / $O(n)$
  - Modified memory: **everything** / $O(n)$
  - Modified memory during enumeration: **everything** / … / $O(1)$
- Implications:
  - Domain values $\leq n^c$
  - Sorting the input in $O(n)$
    - Radix Sort handles k integers, each bounded by n^c, in time $O(ck + cn)$
  - If $O(n^c)$ available memory,
    - Lookup table with k elements: construction in $O(k)$, search in $O(1)$
- **In this talk, assume the relaxed model**

“saves”
log factors

n = size of input database

Join Queries

When it doesn’t work out

Acyclic Joins

[Yannakakis 81]

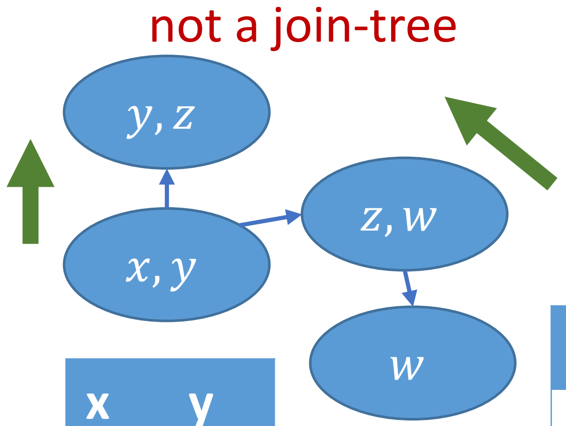
- An efficient algorithm for acyclic joins
    1. Find a join tree and set a root
    2. Remove dangling tuples
    3. Join

only works with join trees

1. Leaf-to-root:
   $r_{parent} \leftarrow r_{parent} \ltimes r_{child}$
2. Root-to-leaf:
   $r_{child} \leftarrow r_{child} \ltimes r_{parent}$

y	z
y1	z1
y1	z2
y2	z2
y3	z1
y4	z3

not a join-tree



z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

x	y
x1	y1
x1	y2
x2	y2
x2	y4

w
w1
w2

dangling tuples

Acyclic Joins

[Yannakakis 81]

- An efficient algorithm for acyclic joins
  1. Find a join tree and set a root
  2. Rem
  3. Joi

This approach fails for cyclic joins.
Is it possible with a different approach?

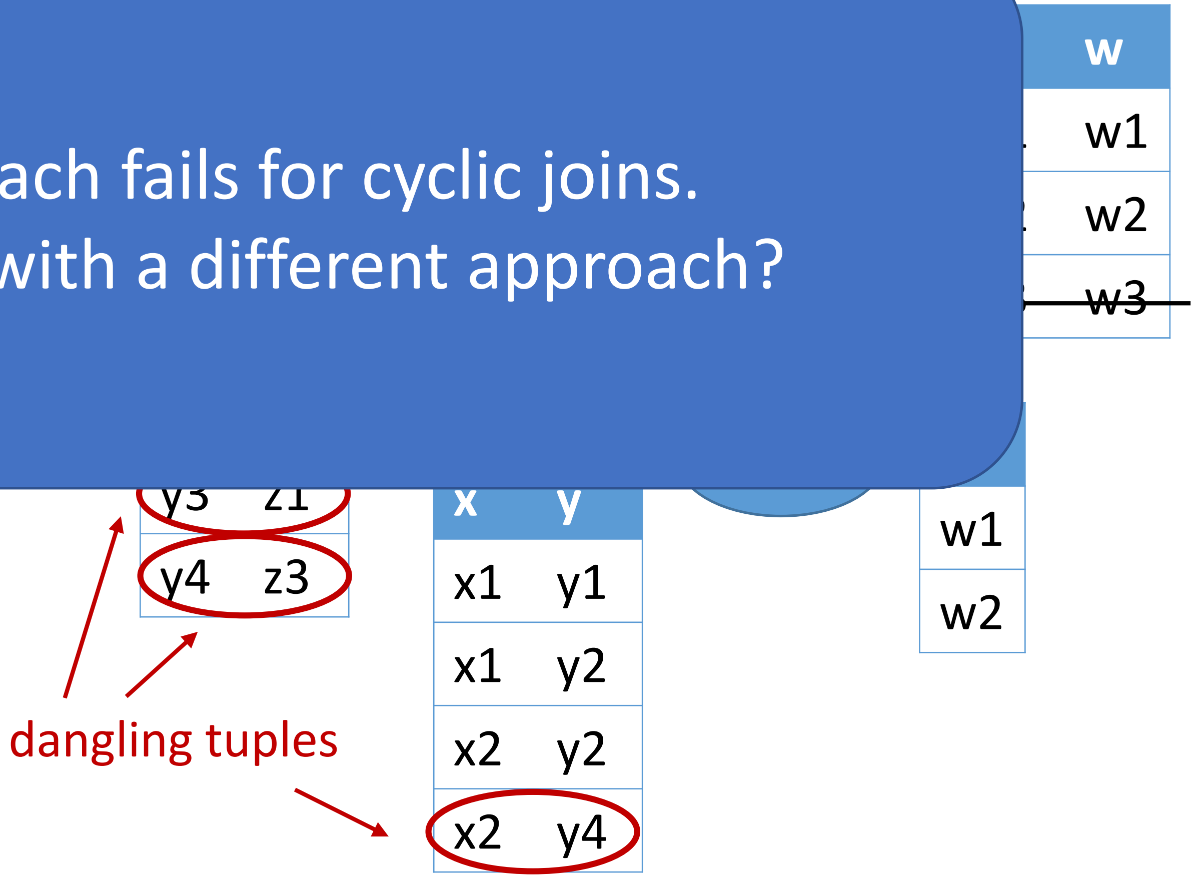
only wo

1. Leaf-t

   r_{paren}

2. Root-t

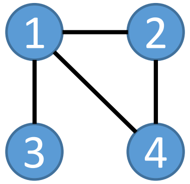
   $r_{child} \leftarrow r_{child} \ltimes r_{parent}$



Lower Bound: Cyclic Joins

[Brault-Baron 13]

Assumption: cannot detect triangles in a graph in linear time



edges

$\boldsymbol{Q}$

x	y	z
1	2	4

$\boldsymbol{R_1 = R_2 = R_3}$

1	2
1	3
1	4
2	4

Cyclic: $Q_2(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(x, z)$

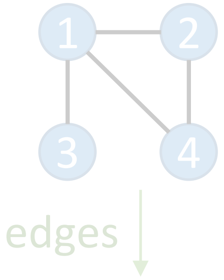
first answer in linear time $\Longrightarrow$ triangle in linear time $\Longrightarrow$ not possible

Lower Bound: Cyclic Joins

[Brault-Baron 13]

Assumption: k-Hypercliques cannot be found in time $O(m)$

m = number of edges of size $k-1$



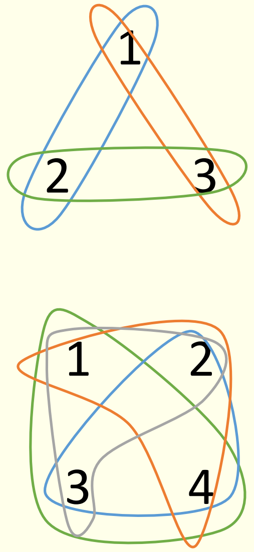
Q

x	y	z
1	2	4

$R_1 = R_2 = R_3$

1	2
1	3
1	4
2	4

k-Hypercliques



Cyclic: $Q_2(x,y,z) \leftarrow R_1(x,y), R_2(y,z), R_3(x,z)$

For every cyclic query, we can preform a similar reduction!

Joins Queries

- Given a join query Q,

If Q is acyclic, it is solvable
in linear preprocessing and constant delay

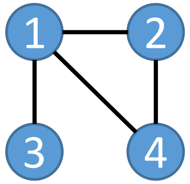
If Q is cyclic, a first answer
cannot be found in linear time *

* assuming hardness of k-hyperclique detection
* assuming no self-joins

Lower Bound: Cyclic Joins

[Brault-Baron 13]

Assumption: cannot detect triangles in a graph in linear time



edges

$\boldsymbol{Q}$

x	y	z
1	2	4

$\boldsymbol{R_1} = \boldsymbol{R_2} = \boldsymbol{R_3}$

1	2
1	3
1	4
2	4

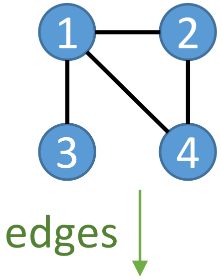
Cyclic: $Q_2(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(x, z)$

first answer in linear time $\Longrightarrow$ triangle in linear time $\Longrightarrow$ not possible

Lower Bound: Cyclic Joins

[Brault-Baron 13]

Assumption: cannot detect triangles in a graph in linear time



$\boldsymbol{Q}$

x	y	z	u	v
1	2	4	⊥	⊥

$\boldsymbol{R_1}$

1	2
1	3
1	4
2	4

$\boldsymbol{R_2}$

1	2	⊥
1	3	⊥
1	4	⊥
2	4	⊥

$\boldsymbol{R_3}$

⊥	1	2
⊥	1	3
⊥	1	4
⊥	2	4

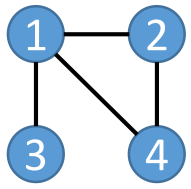
Cyclic: $Q_2(x, y, z, \boldsymbol{u}, \boldsymbol{v}) \leftarrow R_1(x, y), R_2(y, z, \boldsymbol{v}), R_3(\boldsymbol{u}, x, z)$

first answer in linear time $\Longrightarrow$ triangle in linear time $\Longrightarrow$ not possible

Lower Bound: Cyclic Joins

[Brault-Baron 13]

Assumption: cannot detect triangles in a graph in linear time



edges

With self-joins, cannot assign a different relation to each atom

Q

x	y	z	u	v
1	2	4	⊥	⊥

R_1

1	2
1	3
1	4
2	4

R_2

1	2	⊥
1	3	⊥
1	4	⊥
2	4	⊥

R_3

⊥	1	2
⊥	1	3
⊥	1	4
⊥	2	4

Cyclic: $Q_2(x, y, z, \boldsymbol{u}, \boldsymbol{v}) \leftarrow R_1(x, y), R_2(y, z, \boldsymbol{v}), \boldsymbol{R_2}(\boldsymbol{u}, x, z)$

first answer in linear time $\Longrightarrow$ triangle in linear time $\Longrightarrow$ not possible

Self-Joins

- Lower bounds do not apply with self-joins
- Can they be easier?
  - Yes! [Berkholz, Gerhardt, Schweikardt; SIGLOG News 20]

- Example: [**C**, Segoufin; 22]

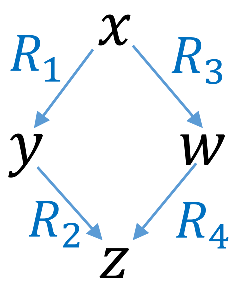
$$Q_1(x, y, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(x, w), R_4(w, z)$$

No Constant delay

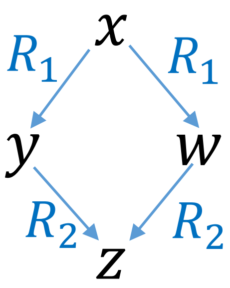
$$Q_2(x, y, z, w) \leftarrow R_1(x, y), R_2(y, z), R_1(x, w), R_2(z, w)$$

Constant delay

Q_1:



Q_2:



Conjunctive Queries

(Introducing Projections)

CQ Example

authors:

Author	Affiliation	Title
Shay Gershtein	Tel Aviv Univ.	On the Hardness…
Uri Avron	Tel Aviv Univ.	On the Hardness…
Florent Capelli	Univ. Lille	Linear programs…
Nicolas Crosetti	Univ. Lille	Linear programs…

schedule:

Title	Day
On the Hardness of…	Tue
Linear programs…	Tue
Answering Unions…	Tue
On an Information…	Wed

Affiliation	Day
Tel Aviv Univ.	Tue
Univ. Lille	Tue

$$Q_1(\text{Affiliation}, \text{Day}) \leftarrow \text{authors}(\text{Author}, \text{Affiliation}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$$

Handling Projection

works

$$Q_1(y, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), R_4(w)$$

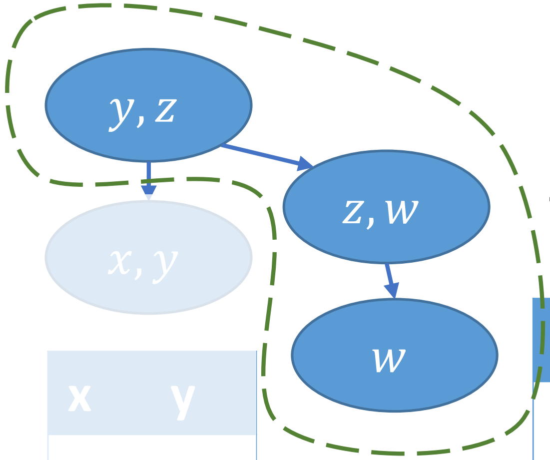
Solution:

1. Find a join tree
2. Remove dangling tuples
3. **Ignore existential variables**
4. Join

~~x~~	y	z	w
~~x1~~	y1	z1	w1
~~x1~~	y1	z2	w2
~~x1~~	y2	z2	w2
~~x2~~	y2	z2	w2

→

y	z	w
y1	z1	w1
y1	z2	w2
y2	z2	w2



y	z
y1	z1
y1	z2
y2	z2
~~y3~~	~~z1~~
~~y4~~	~~z3~~

x	y
x1	y1
x1	y2
x2	y2
~~x2~~	~~y4~~

z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

w
w1
w2

Handling Projection

works

$$Q_1(y,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w), R_4(w)$$

doesn't work

$$Q_2(x,y,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w), R_4(w)$$

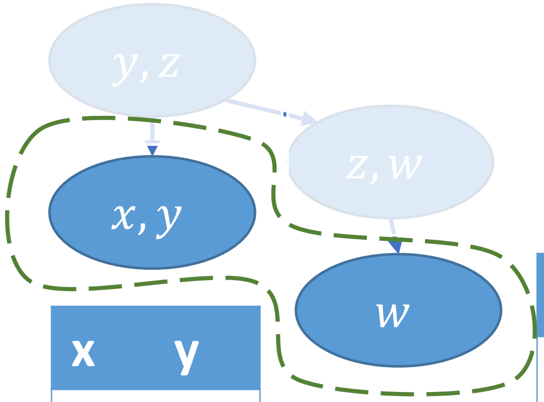
Solution:

1. Find a join tree
2. Remove dangling tuples
3. **Ignore existential variables**
4. Join

x	y	~~z~~	w
x1	y1	~~z1~~	w1
x1	y1	~~z2~~	w2
x1	y2	~~z2~~	w2
x2	y2	~~z2~~	w2

→

x	y	w
x1	y1	w1
x1	y1	w2
x1	y2	w2
x2	y2	w2



y	z
y1	z1
y1	z2
y2	z2
~~y3~~	~~z1~~
~~y4~~	~~z3~~

z	w
z1	w1
z2	w2
~~z3~~	~~w3~~

x	y
x1	y1
x1	y2
x2	y2
~~x2~~	~~y4~~

w
w1
w2

Definitions

[Bagan, Durand, Grandjean; CSL 07]

An acyclic CQ has a graph with:

A free-connex CQ also requires:

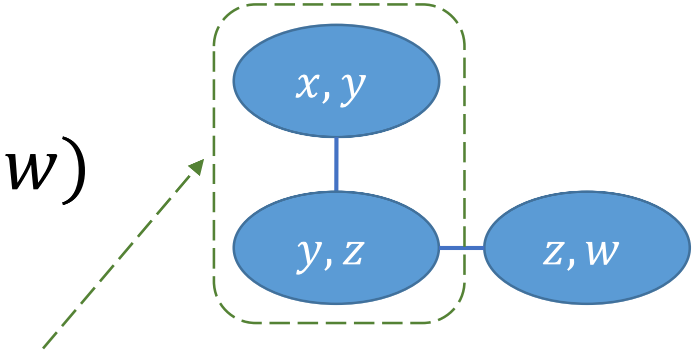
1. a node for every atom
possibly also subsets

2. tree

3. for every variable:
the nodes containing it form a subtree

free – connex **acyclic**

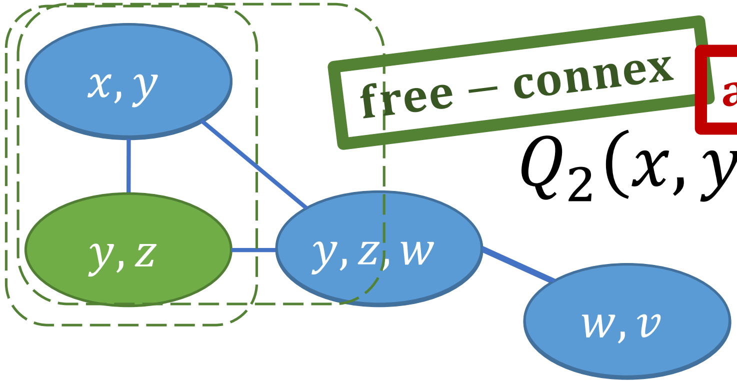
$$Q_1(x,y,z) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w)$$



4. a subtree with exactly the free variables

free – connex **acyclic**

$$Q_2(x,y,z) \leftarrow R_1(x,y), R_2(y,z,w), R_3(w,v)$$



Eliminating Projection

given a **free-connex** CQ and an input DB,
we can construct **in linear time**
an equivalent **full acyclic** CQ and input DB

Conjunctive Queries (CQs)

[Brault-Baron 13]

[Bagan, Durand, Grandjean; CSL 07]

- Given a conjunctive query Q,

If Q is free-connex, Q ∈ <lin,const>

If Q is acyclic not free-connex, Q ∉ <lin,const>*

If Q is cyclic, Q ∉ <lin,const>**

* no self-joins, assuming hardness of matrix multiplication

** no self-joins, assuming hardness of k-hyperclique detection

Lower Bound: acyclic non-free-connex

Hard due to duplicates

Q	
1	1

R_1	
1	1
1	2
1	3

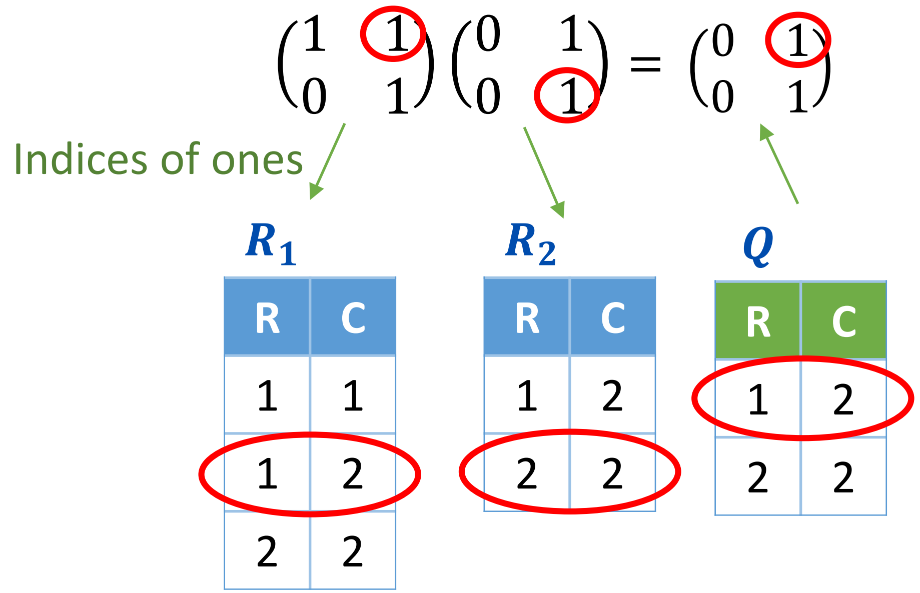
R_2	
1	1
2	1
3	1

Acyclic non-free-connex: $Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$

Lower Bound: acyclic non-free-connex

[Bagan, Durand, Grandjean; CSL 07]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$



$$\begin{pmatrix}1 & 1\\0 & 1\end{pmatrix}\begin{pmatrix}0 & 1\\0 & 1\end{pmatrix} = \begin{pmatrix}0 & 1\\0 & 1\end{pmatrix}$$

Indices of ones

$\boldsymbol{R_1}$

R	C
1	1
1	2
2	2

$\boldsymbol{R_2}$

R	C
1	2
2	2

$\boldsymbol{Q}$

R	C
1	2
2	2

Intractability cause:

$x - y - z$

Acyclic non-free-connex: $Q(x,z) \leftarrow R_1(x,y), R_2(y,z)$

$O(n^2)$ preprocessing + $O(1)$ delay = $O(n^2)$ total $\implies$ not possible

Unions of Conjunctive Queries

(Introducing Unions)

UCQ Example

authors:

Author	Affiliation	Title
Shay Gershtein	Tel Aviv Univ.	On the Hardness…
Uri Avron	Tel Aviv Univ.	On the Hardness…
Florent Capelli	Univ. Lille	Linear programs…
Nicolas Crosetti	Univ. Lille	Linear programs…

schedule:

Title	Day
On the Hardness of…	Tue
Linear programs…	Tue
Answering Unions…	Tue
On an Information…	Wed

talks:

Speaker	Affiliation	Title
Nofar Carmeli	ENS, PSL	Answering Unions…
Hung Ngo	RelationalAI	On an Information…

Affiliation	Day
Tel Aviv Univ.	Tue
Univ. Lille	Tue
Affiliation	**Day**
ENS, PSL Univ.	Tue
RelationalAI	Wed

$$Q_3 = Q_1 \cup Q_2$$

$$Q_1(\mathrm{Affiliation}, \mathrm{Day}) \leftarrow \mathrm{authors}(\mathrm{Author}, \mathrm{Affiliation}, \mathrm{Title}), \mathrm{schedule}(\mathrm{Title}, \mathrm{Day})$$
$$Q_2(\mathrm{Affiliation}, \mathrm{Day}) \leftarrow \mathrm{talks}(\mathrm{Speaker}, \mathrm{Affiliation}, \mathrm{Title}), \mathrm{schedule}(\mathrm{Title}, \mathrm{Day})$$

Cases for UCQs

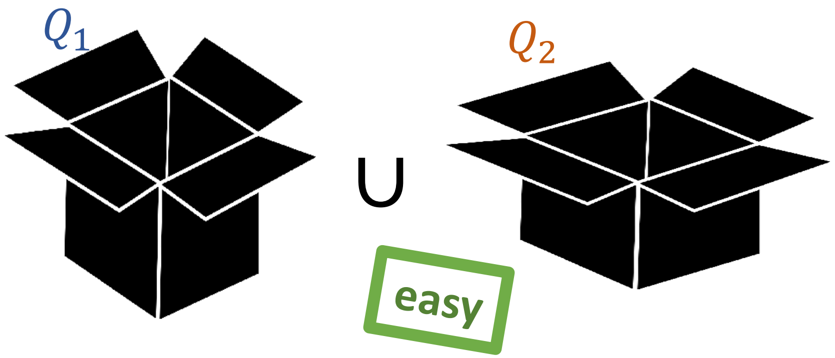
All CQs are Easy

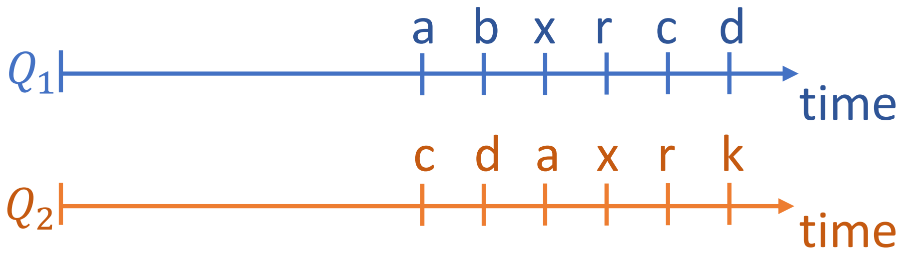
always easy

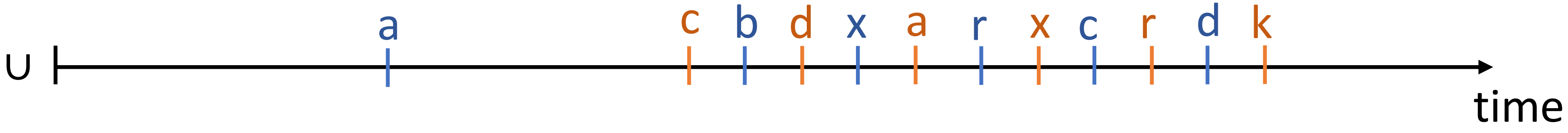
Some Easy, Some Hard

All CQs are Hard

Easy ∪ Easy Is Always Easy







Generated (lookup):

a b c d

x

Queue:

a
c
b
d
x

Output:

…

Enumeration: union of easy CQs

[Durand, Strozecki; CSL 11]

```
while $A$.hasNext():
    $a$ = $A$.next()
    if $a \notin B$:
        print $a$
    else:
        print $B$.next()
while $B$.hasNext():
    print $B$.next()
```

prints $A \setminus B$

prints B

$A \setminus B$ and B are a partition of $A \cup B$

Cases for UCQs

[**C**, Kröll; PODS 19]

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

Lower Bound: acyclic non-free-connex

[Bagan, Durand, Grandjean; CSL 07]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Indices of ones

$\boldsymbol{R_1}$

R	C
1	1
1	2
2	2

$\boldsymbol{R_2}$

R	C
1	2
2	2

$\boldsymbol{Q}$

R	C
1	2
2	2

Intractability cause:

$x - y - z$

Acyclic non-free-connex: $Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$

$O(n^2)$ preprocessing + $O(1)$ delay = $O(n^2)$ total $\implies$ not possible

Why this isn't hard

not free connex

hard part

$$Q_1(x,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w)$$

$$\cup$$

$$Q_2(x',y',z') \leftarrow R_1(x',y'), R_2(y',z')$$

Q_1		
1	2	⊥
2	2	⊥

Q_2		
1	1	2
1	2	2
2	2	2

$O(n^3)$ solutions:
The computation does not contradict the assumption

R_1	
1	1
1	2
2	2

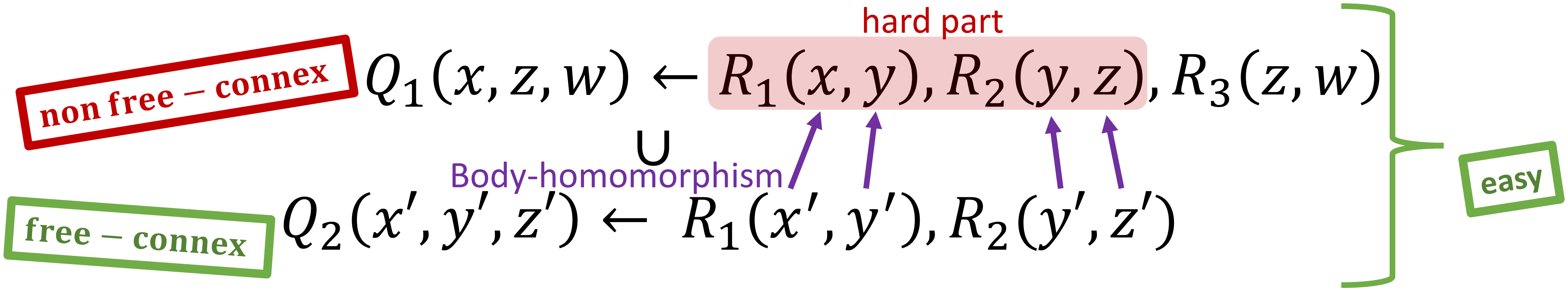
R_2	
1	2
2	2

R_3	
2	⊥

The hardness results do not hold within a union

Hard ∪ Easy Can Be Easy

[**C**, Kröll; PODS 19]



hard part

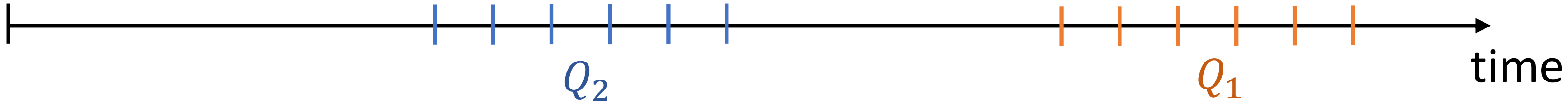
non free − connex $Q_1(x,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w)$

$\cup$

Body-homomorphism

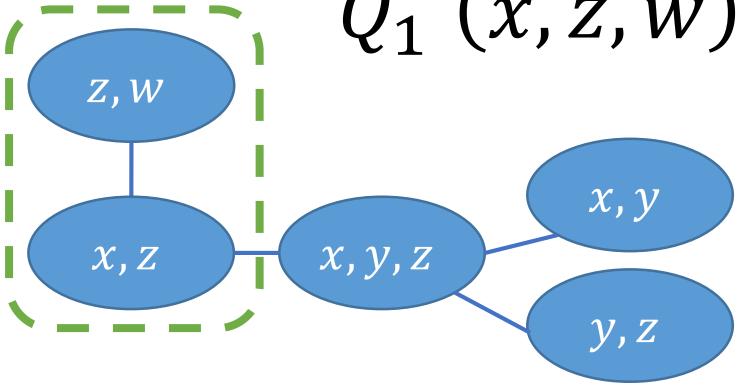
free − connex $Q_2(x',y',z') \leftarrow R_1(x',y'), R_2(y',z')$

easy



$$Q_1{}'(x,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w), Q_2(x,y,z)$$

free − connex



Cheater's Lemma

[**C**, Kröll; PODS 19]

If an enumeration problem can be solved with:

- Usually constant delay
- Almost no duplicates

constant number of linear delay steps

constant number of duplicates per answer

Then*, it is easy

Can be solved in:
linear preprocessing,
constant delay,
no duplicates

* assuming using polynomial space

Complexity Measures

[**C**, Kröll; TODS 21]

- Linear total time
  - Total time $O(n+m)$

- Linear partial time
  - Time before the ith answer is $O(n+i)$

- Linear preprocessing and constant delay
  - Time before the first answer $O(n)$
  - Time between successive answers $O(1)$

equivalent
assuming using
polynomial space

n = input size, m = output size

Cases for UCQs

[**C**, Kröll; PODS 19]

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

sometimes hard

sometimes easy

Hard ∪ Hard Can Be Easy

[**C**, Kröll; PODS 19]

- CQs with **isomorphic bodies**.

$$Q_1(x,z,w,u) \leftarrow \overset{\text{hard part}}{R_1(x,y), R_2(y,z)}, R_3(z,w), R_4(w,u)$$

$$Q_2(x,y,z,u) \leftarrow R_1(x,y), R_2(y,z), \underset{\text{hard part}}{R_3(z,w), R_4(w,u)}$$

	Step	Output	Side Effect
1	Solve $Q_2{}'$	$\subseteq Q_2$	Find $R_1 \bowtie R_2$
2	Solve Q_1^+	Q_1	Find $R_3 \bowtie R_4$
3	Solve Q_2^+	Q_2	

Related Problems

Example

$$Q(\text{Session}, \text{AttendanceA}, \text{AttendanceB}) \\ \leftarrow \text{runA}(\text{Session}, \text{AttendanceA}), \text{runB}(\text{Session}, \text{AttendanceB})$$

runA:

Session	AttendanceA
Tutorial	123
Optimization	85
Learning	123
Streaming	74

runB:

Session	AttendanceB
Tutorial	32
Optimization	71
Learning	78
Streaming	29

Session	AttendanceA	AttendanceB
Tutorial	123	29
Optimization	85	78
Learning	123	71
Streaming	74	32

- What can we learn before the enumeration is done?
- The answer order may be important
  - Random (sampling without repetitions)
  - Sorted (lexicographically / by sum of weights)

Ordered Enumeration

- With arbitrary order:
  - Free-connex: linear preprocessing, constant delay
- With order guarantees:
  - Free-connex: linear preprocessing, **logarithmic** delay

Random (sampling without repetitions)
[**C,** Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]
Sorted (lexicographically / by sum of weights)
[Tziavelis, Gatterbauer, Riedewald; VLDB 21]

Algorithm for Random Enumeration

[**C,** Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]

- Find the number N of answers

6

- Find a random permutation of 1,…,N

1 5 3 2 6 4

- Direct access to answers

a1 a5 a3 a2 a6 a4

answers
a1
a2
a3
a4
a5
a6

Direct Access
+
Binary Search

Modified Fisher-Yates Shuffle
[Durstenfeld 1964]

Direct Access
[Brault-Baron 2013]

Direct and Ranked Access

Example: get the median number of total attendees in a session

runA:

Session	AttendanceA
Tutorial	123
Optimization	85
Learning	131
Streaming	74

runB:

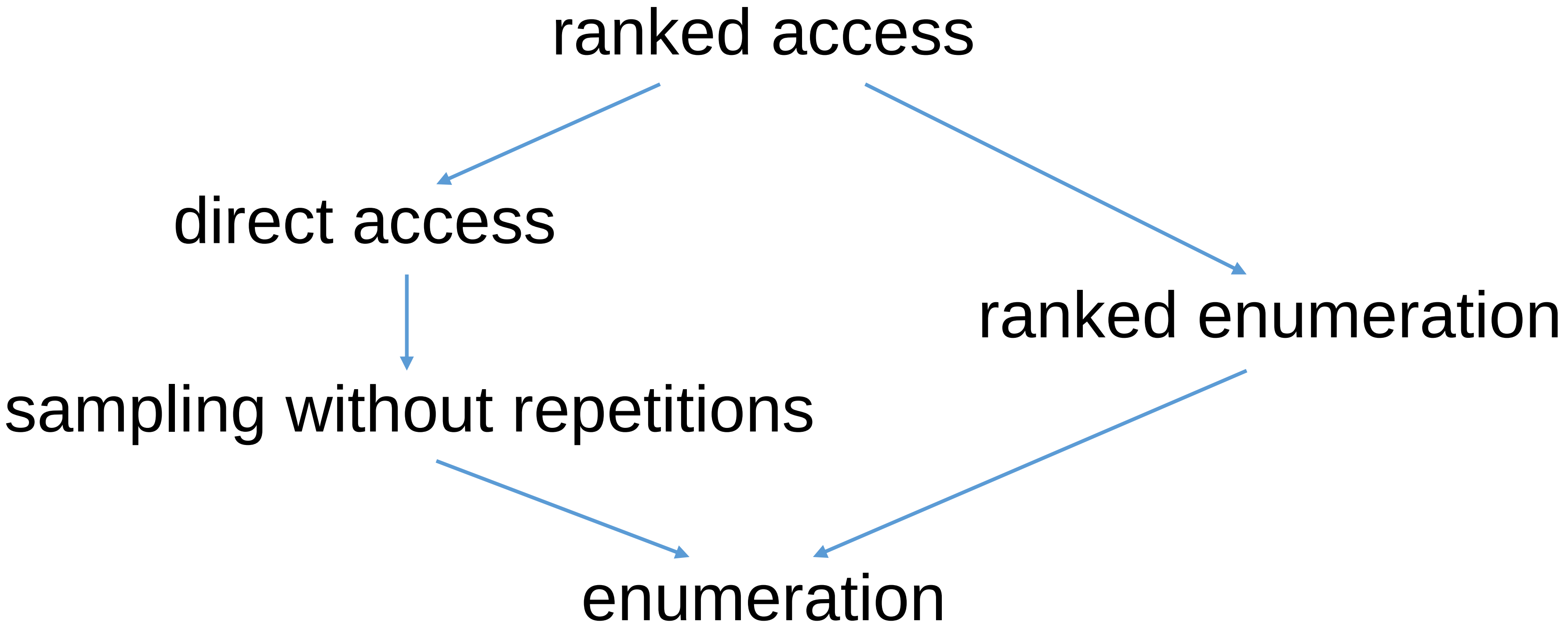
Session	AttendanceB
Tutorial	32
Optimization	71
Learning	78
Streaming	29

- Solution 1:
  join, sort, access the middle
- Solution 2:
  count, ranked enumeration until the middle
- Solution 3:
  count, ranked access to the middle

Session	AttendanceA	AttendanceB	SUM
Learning	123	71	194
Optimization	85	78	163
Tutorial	123	29	152
Streaming	74	32	106

- Direct Access: simulate an answers array
- Ranked Access: simulate a **sorted** answers array

Connection between problems



ranked access

direct access

ranked enumeration

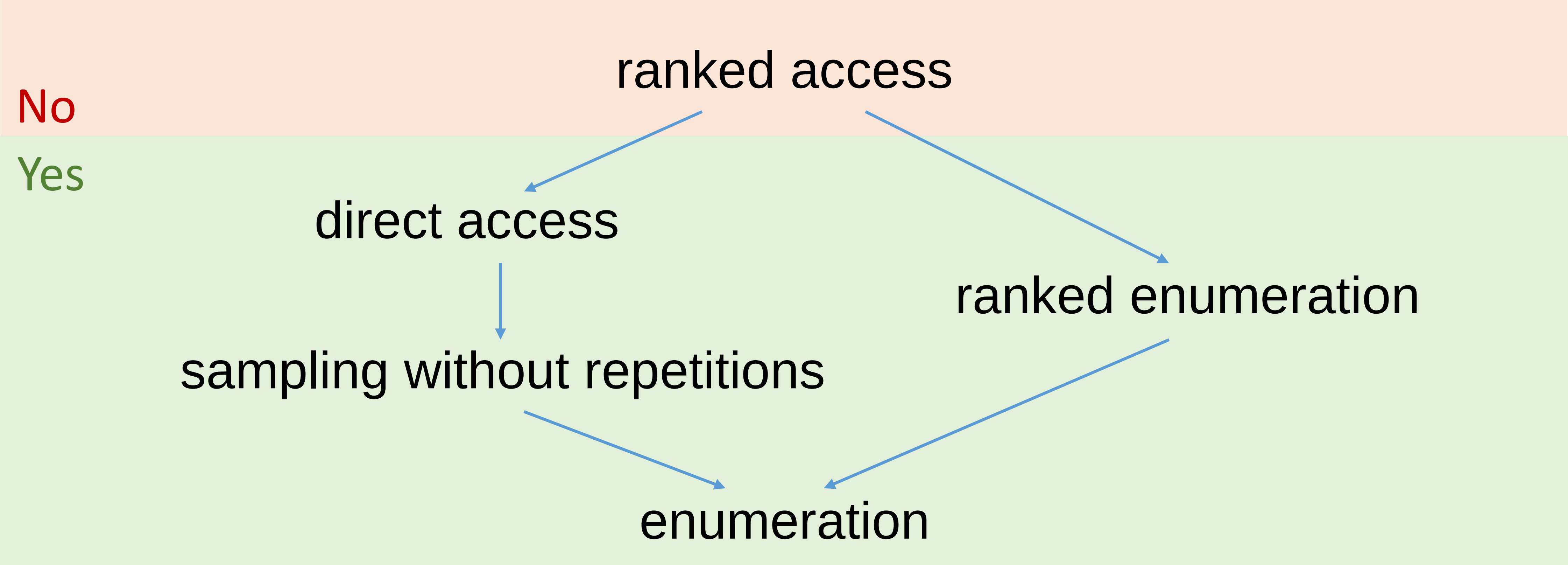
sampling without repetitions

enumeration

Connection between problems

[**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

Can be solved efficiently* for **all free-connex** CQs?



No

Yes

ranked access

direct access

ranked enumeration

sampling without repetitions

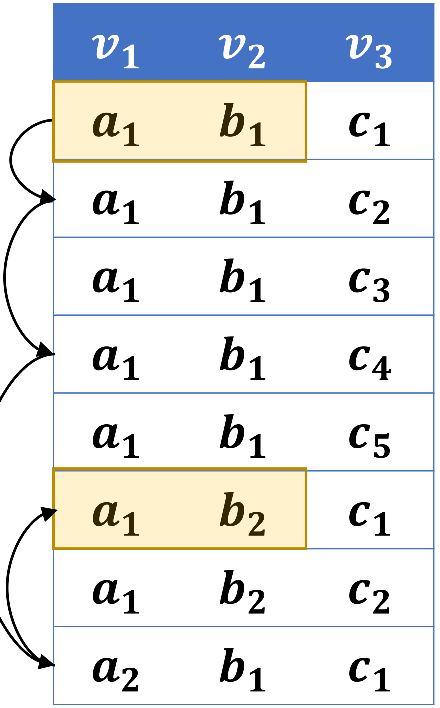
enumeration

* with **polylog** time per answer after linear preprocessing

CQ Enumeration & Lex. Access

[**C**, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]

binary search for next different $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ values



v_1	v_2	v_3
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_1	c_3
a_1	b_1	c_4
a_1	b_1	c_5
a_1	b_2	c_1
a_1	b_2	c_2
a_2	b_1	c_1

Enumerate

$$Q_1(v_1, v_2) \leftarrow R(v_1, v_3), S(v_3, v_2)$$

Not free-connex

using

Lexicographic access

$$Q_2(v_1, v_2, v_3) \leftarrow R(v_1, v_3), S(v_3, v_2)$$

Disruptive trio

Log number of direct-access calls between answers

Q_1 has no enumeration with polylog delay $\Rightarrow$ Q_2 has no lexicographic access with polylog access time

Related Work

- Other settings
  - Sparsity [Schweikardt, Segoufin, Vigny; PODS 18]
  - Functional dependencies [**C**, Kröll; ICDT 18]
  - Dynamic data [Berkholz, Keppeler, Schweikardt; ICDT 18]
  - Theta-joins [Idris, Ugarte, Vansummeren, Voigt, Lehner; VLDB 18]
  - Signed CQs [Brault-Baron 13]
- Other surveys
  - Written tutorial [Berkholz, Gerhardt, Schweikardt; SIGLOG News 20]
  - Tutorial talk and paper [Durand; PODS 20]
  - Surveys [Segoufin; STACS 14] [Segoufin; SIGMOD Rec. 15]

Focus

- Data: general relational databases
- Tasks: enumeration & related tasks
- Queries: joins → CQs → UCQs
- Tractability: “ideal” time guarantees
- Goal: classify cases into tractable/not