LinCQA: Faster Consistent Query Answering with Linear Time Guarantees

Xiating Ouyang [1]

joint work with Zhiwei Fan [1,2] Paris Koutris [1] Jef Wijsen [3]

University of Wisconsin–Madison [1]

Meta [2]

University of Mons [3]

SIGMOD, Seattle WA, June 18–23 2023

1. Consistent Query Answering for Primary Keys

2. Acyclic Queries Revisited

3. Pair-pruning join tree (PPJT) & LinCQA

4. Experiments

1. Consistent Query Answering for Primary Keys

2. Acyclic Queries Revisited

3. Pair-pruning join tree (PPJT) & LinCQA

4. Experiments

Primary key constraint (violated)

- Metadata of `stackoverflow.com` as of 02/2021 from Stack Exchange Data Dump
- 551M rows, ~400 GB

Table	# of rows	inRatio	bSize	# of Attributes
Users	14M	0%	1	14
Posts	53M	0%	1	20
PostHistory	141M	0.001%	4	9
Badges	40M	0.58%	941	4
Votes	213M	30.9%	1441	6

inconsistencyRatio = # facts violating PK constraint / # of rows

blockSize = max. # facts with the same PK

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
     = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Data cleaning

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Data cleaning $2 \times 2 \times 2 \times 1$ repairs

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Data cleaning $2 \times 2 \times 2 \times 1$ repairs

What are the answers guaranteed to be returned on all repairs?

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Data cleaning $2 \times 2 \times 2 \times 1$ repairs

What are the answers $\boxed{\text{guaranteed to be returned on \underline{all} repairs}}$?

CS 703

Finding consistent answers

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id
```

Return all classes taught by a CS faculty

{CS 703, CS 787} . . .

Data cleaning $2 \times 2 \times 2 \times 1$ repairs

What are the answers guaranteed to be returned on all repairs?

CS 703

Consistent Answer

Finding consistent answers without enumeration

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id  AND
  (all f_id's for the same c_id
  appear in CS_Faculty)
```

The original query has a *first-order rewriting*

Finding consistent answers without enumeration

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id  AND
  (all f_id’s for the same c_id
  appear in CS_Faculty)
```

The original query has a first-order rewriting

Finding consistent answers without enumeration

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id  AND
  (all f_id’s for the same c_id
  appear in CS_Faculty)
```

The original query has a first-order rewriting

Finding consistent answers without enumeration

Course

c_id	f_id
CS 703	2
CS 703	5
CS 787	3
CS 787	5

CS_Faculty

f_id	f_name
2	Adam
2	Alice
5	Bob

```
SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id
      = CS_Faculty.f_id  AND
  (all f_id's for the same c_id
  appear in CS_Faculty)
```

The original query has a first-order rewriting

For which Q can the consistent answers be found efficiently?

Can we build a system to find the consistent answers?

For which Q can the consistent answers be found efficiently?

Can we build a system to find the consistent answers?

1 Consistent Query Answering for Primary Keys

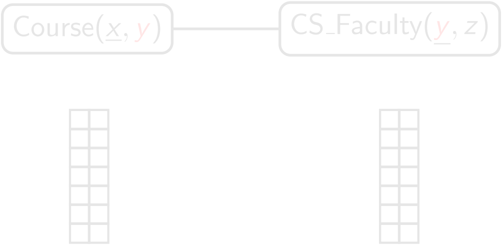
2 Acyclic Queries Revisited

3 Pair-pruning join tree (PPJT) & LinCQA

4 Experiments

Acyclic query evaluation

$$q() \text{ :- } \mathrm{Course}(\underline{x}, y), \mathrm{CS\_Faculty}(\underline{y}, z)$$

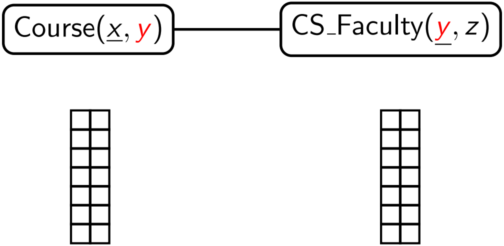


Course($\underline{x}$, y)

CS_Faculty($\underline{y}$, z)

Acyclic query evaluation

$$q() \text{ :- } \mathrm{Course}(\underline{x}, y), \mathrm{CS\_Faculty}(\underline{y}, z)$$

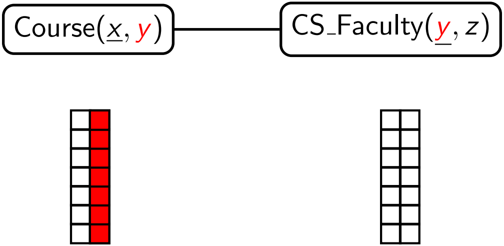


Course($\underline{x}$, y)

CS_Faculty($\underline{y}$, z)

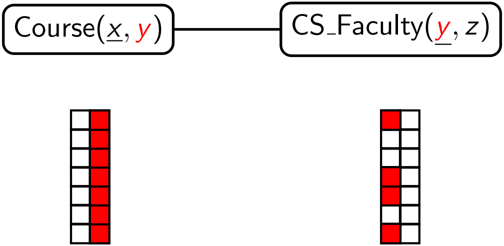
Acyclic query evaluation

$$q() \text{ :- } \mathrm{Course}(\underline{x}, y), \mathrm{CS\_Faculty}(\underline{y}, z)$$


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

Acyclic query evaluation

$$q() \text{ :- } \mathrm{Course}(\underline{x}, y), \mathrm{CS\_Faculty}(\underline{y}, z)$$


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

Yannakakis [VLDB'81]

The answer to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

Yannakakis [VLDB'81]   Our result

consistent answer

The ~~answer~~ to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

$\wedge$

with a pair-pruning join tree (PPJT)

Yannakakis [VLDB'81]   Our result

consistent answer

The ~~answer~~ to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

$\wedge$

with a pair-pruning join tree (PPJT)

non-Boolean $\leq_T^P$ **Boolean**

1 Consistent Query Answering for Primary Keys

2 Acyclic Queries Revisited

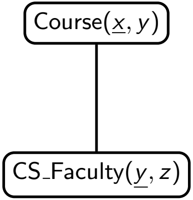
3 Pair-pruning join tree (PPJT) & LinCQA

4 Experiments

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

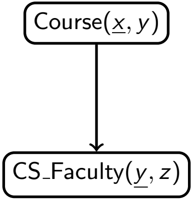
$\boxed{\text{the root of every subtree is \underline{unattacked} in the subtree}}$


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$



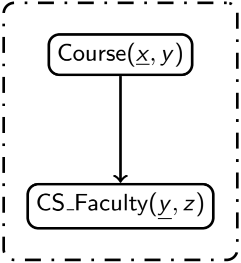
Course($\underline{x}, y$)

CS_Faculty($\underline{y}, z$)

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

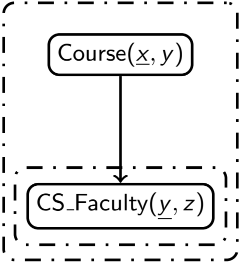
the root of every subtree is unattacked in the subtree


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

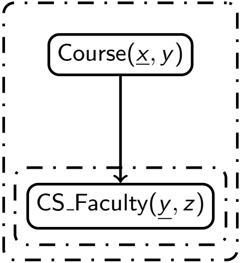
$$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$$


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

$$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$$


Course($\underline{x}$, y)
CS_Faculty($\underline{y}$, z)

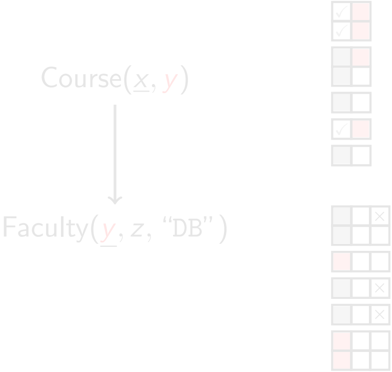
Every acyclic query has a join tree, but not every acyclic query has a PPJT

PPJT is a wide class

- star/snowflake schema (e.g. TPC-H)
- Every acyclic query in $\mathcal{C}_{\text{forest}}$ [FM, ICDT'05] has a PPJT

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"



$\mathrm{Course}(\underline{x}, y)$
$\mathrm{Faculty}(\underline{y}, z, \text{“DB”})$
✓
✓
✓
×
×
×

$\mathrm{Course}_{join}()$:- $\mathrm{Course}(x, y)$, $\neg\mathrm{Course}_{fkey}(x)$

$\mathrm{Course}_{fkey}(x)$:- $\mathrm{Course}(x, y)$, $\neg\mathrm{Faculty}_{join}(y)$

$\boxed{\forall\mathrm{Child} : \mathrm{Root}_{fkey}(\vec{x}) \text{ :- } \mathrm{Root}(\underline{\vec{x}}, \vec{y}), \neg\mathrm{Child}_{join}(\vec{\alpha})}$

$\boxed{\mathrm{Child}_{join}(\vec{\alpha}) \text{ :- } \mathrm{Child}(\underline{\vec{u}}, \vec{v}), \neg\mathrm{Child}_{fkey}(\vec{u})}$

$\mathrm{Faculty}_{join}(y)$:- $\mathrm{Faculty}(y, z, w)$, $\neg\mathrm{Faculty}_{fkey}(y)$

$\mathrm{Faculty}_{fkey}(y)$:- $\mathrm{Faculty}(y, z, w)$, $w \neq$ “DB”

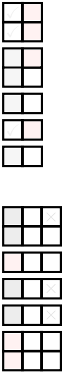
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$$\begin{tikzcd}
\text{Course}(\underline{x}, y) \arrow[d] \\
\text{Faculty}(\underline{y}, z, \text{“}\mathtt{DB}\text{”})
\end{tikzcd}$$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall \text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

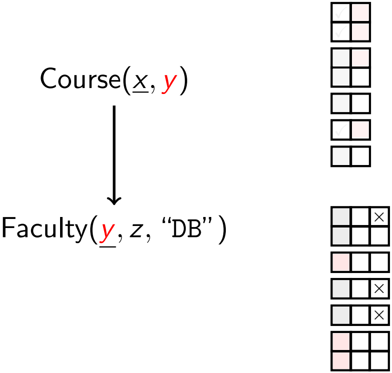
$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“}\mathtt{DB}\text{”}$

also expressible in SQL!
runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"



$\text{Course}(\underline{x}, y)$

$\text{Faculty}(\underline{y}, z, \text{``DB''})$

$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{``DB''}$

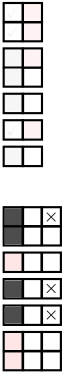
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is “bad”

$$\begin{tikzcd}
\text{Course}(\underline{x}, {\color{red}y}) \arrow[d] \\
\text{Faculty}(\underline{{\color{red}y}}, z, \text{“DB”})
\end{tikzcd}$$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}({\color{red}y})$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}({\color{red}\vec{\alpha}})}$

$\boxed{\text{Child}_{join}({\color{red}\vec{\alpha}}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}({\color{red}y}) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“DB”}$

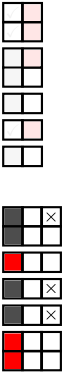
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is “bad”

$$\begin{tikzcd}
\text{Course}(\underline{x}, y) \arrow[d] \\
\text{Faculty}(\underline{y}, z, \text{“DB”})
\end{tikzcd}$$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

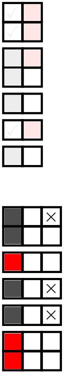
$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“DB”}$

also expressible in SQL!
runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is “bad”

$$\begin{tikzcd}
\text{Course}(\underline{x}, y) \arrow[d] \\
\text{Faculty}(\underline{y}, z, \text{“DB”})
\end{tikzcd}$$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

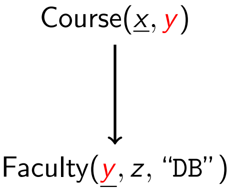
$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“DB”}$

also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

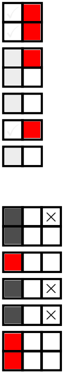
Remove a primary key if some tuple with this primary key is "bad"



$\text{Course}(\underline{x}, y)$

$\downarrow$

$\text{Faculty}(\underline{y}, z, \text{“DB”})$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

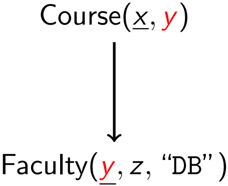
$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“DB”}$

also expressible in SQL!

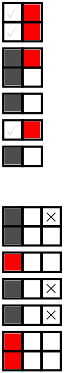
runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"



$\text{Course}(\underline{x}, y)$
$\text{Faculty}(\underline{y}, z, \text{"DB"})$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{"DB"}$

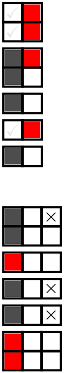
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$$\begin{tikzcd}
\text{Course}(\underline{x}, {\color{red}y}) \arrow[d] \\
\text{Faculty}(\underline{{\color{red}y}}, z, \text{"DB"})
\end{tikzcd}$$



$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}({\color{red}y})$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}({\color{red}\vec{\alpha}})}$

$\boxed{\text{Child}_{join}({\color{red}\vec{\alpha}}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}({\color{red}y}) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$

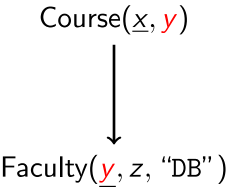
$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{"DB"}$

also expressible in SQL!

runs in $O(N)$

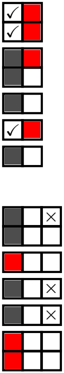
From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"



Course($\underline{x}$, y)

Faculty($\underline{y}$, z, "DB")



$\text{Course}_{join}()$:- $\text{Course}(x, y)$, $\neg\text{Course}_{fkey}(x)$

$\text{Course}_{fkey}(x)$:- $\text{Course}(x, y)$, $\neg\text{Faculty}_{join}(y)$

$\boxed{\forall \text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Faculty}_{join}(y)$:- $\text{Faculty}(y, z, w)$, $\neg\text{Faculty}_{fkey}(y)$

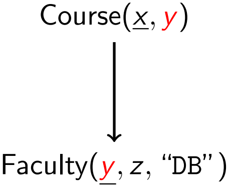
$\text{Faculty}_{fkey}(y)$:- $\text{Faculty}(y, z, w)$, $w \neq$ "DB"

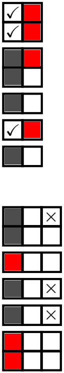
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"


$\text{Course}(\underline{x}, y)$
$\text{Faculty}(\underline{y}, z, \text{"DB"})$



$$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$$

$$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$$

$$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$$

$$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$$

$$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$$

$$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{"DB"}$$

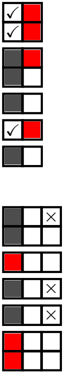
also expressible in SQL!

runs in $O(N)$

From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$$\begin{tikzcd}
\text{Course}(\underline{x}, y) \arrow[d] \\
\text{Faculty}(\underline{y}, z, \text{“DB”})
\end{tikzcd}$$



$$\text{Course}_{join}() \text{ :- } \text{Course}(x, y), \neg\text{Course}_{fkey}(x)$$

$$\text{Course}_{fkey}(x) \text{ :- } \text{Course}(x, y), \neg\text{Faculty}_{join}(y)$$

$$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$$

$$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$$

$$\text{Faculty}_{join}(y) \text{ :- } \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y)$$

$$\text{Faculty}_{fkey}(y) \text{ :- } \text{Faculty}(y, z, w), w \neq \text{“DB”}$$

also expressible in SQL!
runs in $O(N)$

From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

A1	A2
a	b
x	y
...	...

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output (a, b), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

...

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

A1	A2
a	b
x	y
...	...

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output (a, b), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

...

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

A1	A2
a	b
x	y
...	...

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output (a, b), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

...

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

A1	A2
a	b
x	y
. . .	. . .

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output (a, b), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

. . .

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

A1	A2
a	b
x	y
...	...

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output (a, b), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

...

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert\mathrm{OUT}_{\mathrm{inconsistent}}\rvert)$	$O(N \cdot \lvert\mathrm{OUT}\rvert)$
full q (`SELECT *`)	$O(N + \lvert\mathrm{OUT}_{\mathrm{consistent}}\rvert)$	$O(N + \lvert\mathrm{OUT}\rvert)$

Consistent answers of common join queries can be computed with no asymptotic overhead

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \mathrm{OUT}_{\mathrm{inconsistent}} \rvert)$	$O(N \cdot \lvert \mathrm{OUT} \rvert)$
full q (`SELECT *`)	$O(N + \lvert \mathrm{OUT}_{\mathrm{consistent}} \rvert)$	$O(N + \lvert \mathrm{OUT} \rvert)$

Consistent answers of common join queries can be computed with no asymptotic overhead

1. Consistent Query Answering for Primary Keys

2. Acyclic Queries Revisited

3. Pair-pruning join tree (PPJT) & LinCQA

4. Experiments

Setup & Baselines

System	Target class	Interm. output	Backend
CAvSAT	*	SAT formula	SQL Server & MaxHS
Conquer	$\mathcal{C}_{\text{forest}}$	SQL	SQL Server
Improved Conquesto	SJF **FO**	SQL	SQL Server
LinCQA	PPJT	SQL	SQL Server

CloudLab

Stackoverflow data

- Metadata of `stackoverflow.com` as of 02/2021 from Stack Exchange Data Dump
- 551M rows, 400 GB

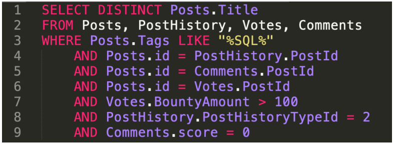
Table	# of rows	inRatio	bSize	# of Attributes
Users	14M	0%	1	14
Posts	53M	0%	1	20
PostHistory	141M	0.001%	4	9
Badges	40M	0.58%	941	4
Votes	213M	30.9%	1441	6

Experiments on Stackoverflow

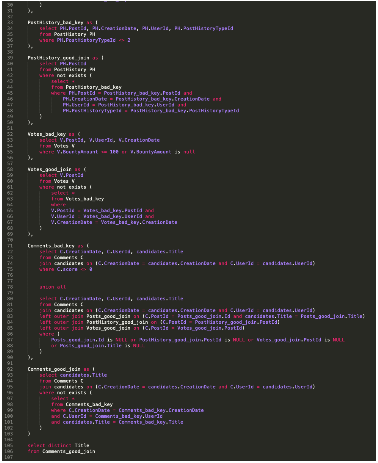
Q_1 : Posts $\bowtie$ Votes Q_2 : Users $\bowtie$ Badges Q_3 : Users $\bowtie$ Posts

Q_4 : Users $\bowtie$ Posts $\bowtie$ Comments

Q_5 : Posts $\bowtie$ PostHistory $\bowtie$ Votes $\bowtie$ Comments



```sql
SELECT DISTINCT Posts.Title
FROM Posts, PostHistory, Votes, Comments
WHERE Posts.Tags LIKE "%SQL%"
    AND Posts.id = PostHistory.PostId
    AND Posts.id = Comments.PostId
    AND Posts.id = Votes.PostId
    AND Votes.BountyAmount > 100
    AND PostHistory.PostHistoryTypeId = 2
    AND Comments.score = 0
```



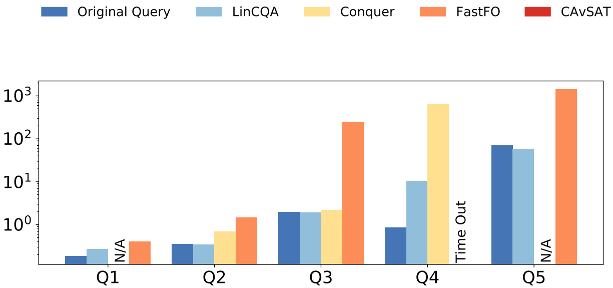
```sql
        )
    ),

    PostHistory_bad_key as (
        select PH.PostId, PH.CreationDate, PH.UserId, PH.PostHistoryTypeId
        from PostHistory PH
        where PH.PostHistoryTypeId <> 2
    ),

    PostHistory_good_join as (
        select PH.PostId
        from PostHistory PH
        where not exists (
            select *
            from PostHistory_bad_key
            where PH.PostId = PostHistory_bad_key.PostId and
                PH.CreationDate = PostHistory_bad_key.CreationDate and
                PH.UserId = PostHistory_bad_key.UserId and
                PH.PostHistoryTypeId = PostHistory_bad_key.PostHistoryTypeId
        )
    ),

    Votes_bad_key as (
        select V.PostId, V.UserId, V.CreationDate
        from Votes V
        where V.BountyAmount <= 100 or V.BountyAmount is null
    ),

    Votes_good_join as (
        select V.PostId
        from Votes V
        where not exists (
            select *
            from Votes_bad_key
            where
            V.PostId = Votes_bad_key.PostId and
            V.UserId = Votes_bad_key.UserId and
            V.CreationDate = Votes_bad_key.CreationDate
        )
    ),

    Comments_bad_key as (
        select C.CreationDate, C.UserId, candidates.Title
        from Comments C
        join candidates on (C.CreationDate = candidates.CreationDate and C.UserId = candidates.UserId)
        where C.score <> 0


        union all

        select C.CreationDate, C.UserId, candidates.Title
        from Comments C
        join candidates on (C.CreationDate = candidates.CreationDate and C.UserId = candidates.UserId)
        left outer join Posts_good_join on (C.PostId = Posts_good_join.Id and candidates.Title = Posts_good_join.Title)
        left outer join PostHistory_good_join on (C.PostId = PostHistory_good_join.PostId)
        left outer join Votes_good_join on (C.PostId = Votes_good_join.PostId)
        where (
            Posts_good_join.Id is NULL or PostHistory_good_join.PostId is NULL or Votes_good_join.PostId is NULL
            or Posts_good_join.Title is NULL
        )
    ),

    Comments_good_join as (
        select candidates.Title
        from Comments C
        join candidates on (C.CreationDate = candidates.CreationDate and C.UserId = candidates.UserId)
        where not exists (
            select *
            from Comments_bad_key
            where C.CreationDate = Comments_bad_key.CreationDate
            and C.UserId = Comments_bad_key.UserId
            and candidates.Title = Comments_bad_key.Title
        )
    )

    select distinct Title
    from Comments_good_join

```

Experiments on Stackoverflow

Q_1 : Posts $\bowtie$ Votes $\quad Q_2$: Users $\bowtie$ Badges $\quad Q_3$: Users $\bowtie$ Posts

Q_4 : Users $\bowtie$ Posts $\bowtie$ Comments

Q_5 : Posts $\bowtie$ PostHistory $\bowtie$ Votes $\bowtie$ Comments



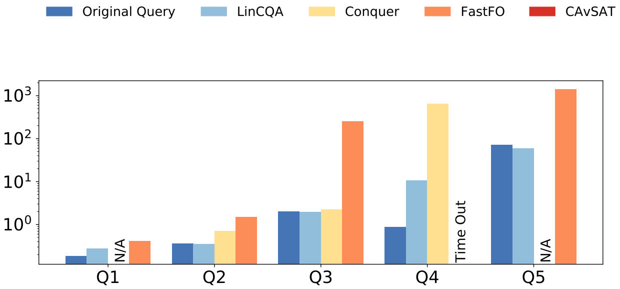
# poss.	27578	145	38320	3925	1250
# cons.	27578	145	38320	3925	1245

Concluding remarks

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert\mathrm{OUT}_{\mathrm{inconsistent}}\rvert)$	$O(N \cdot \lvert\mathrm{OUT}\rvert)$
full q (`SELECT *`)	$O(N + \lvert\mathrm{OUT}_{\mathrm{consistent}}\rvert)$	$O(N + \lvert\mathrm{OUT}\rvert)$

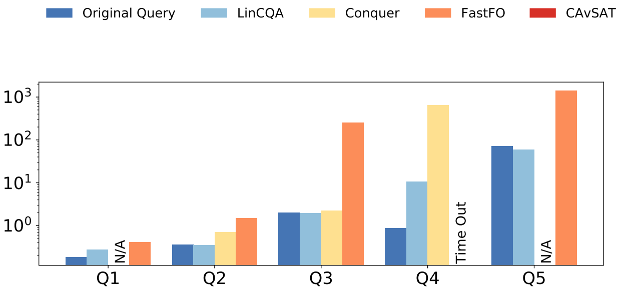
Concluding remarks

Acyclic q	PPJT	Yannakakis [VLDB’81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \mathrm{OUT}_{\mathrm{inconsistent}} \rvert)$	$O(N \cdot \lvert \mathrm{OUT} \rvert)$
full q (`SELECT *`)	$O(N + \lvert \mathrm{OUT}_{\mathrm{consistent}} \rvert)$	$O(N + \lvert \mathrm{OUT} \rvert)$



Concluding remarks

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \mathrm{OUT}_{\mathrm{inconsistent}} \rvert)$	$O(N \cdot \lvert \mathrm{OUT} \rvert)$
full q (`SELECT *`)	$O(N + \lvert \mathrm{OUT}_{\mathrm{consistent}} \rvert)$	$O(N + \lvert \mathrm{OUT} \rvert)$



Thank you!