LinCQA: Faster Consistent Query Answering with Linear Time Guarantees

Xiating Ouyang

joint work with Zhiwei Fan, Paris Koutris, Jef Wijsen

University of Wisconsin–Madison

Meta

University of Mons

SIGMOD, Seattle WA, June 18–23 2023
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
Should I accept the CS PhD offer from UW?

Yes!

Can I go skiing when studying at UW?

Yes!

What about the 9 months of rain?

Which UW?
<table>
<thead>
<tr>
<th>Acronym</th>
<th>School</th>
<th>Great CS</th>
<th>Skiing</th>
<th>Rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>UW</td>
<td>U of WA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UW</td>
<td>U of WI</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Should I accept the CS PhD offer from UW? Yes!
<table>
<thead>
<tr>
<th>Acronym</th>
<th>School</th>
<th>Great CS</th>
<th>Skiing</th>
<th>Rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>UW</td>
<td>U of WA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UW</td>
<td>U of WI</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Should I accept the CS PhD offer from UW?  
Yes!
<table>
<thead>
<tr>
<th>Acronym</th>
<th>School</th>
<th>Great CS</th>
<th>Skiing</th>
<th>Rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>UW</td>
<td>U of WA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UW</td>
<td>U of WI</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Should I accept the CS PhD offer from UW? Yes!
Primary key constraint (violated)

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

<table>
<thead>
<tr>
<th>Table</th>
<th># of rows</th>
<th>inconsistencyRatio</th>
<th>blockSize</th>
<th># of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>14M</td>
<td>0%</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Posts</td>
<td>53M</td>
<td>0%</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>PostHistory</td>
<td>141M</td>
<td>0.001%</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Badges</td>
<td>40M</td>
<td>0.58%</td>
<td>941</td>
<td>4</td>
</tr>
<tr>
<td>Votes</td>
<td>213M</td>
<td>30.9%</td>
<td>1441</td>
<td>6</td>
</tr>
</tbody>
</table>

inconsistencyRatio = # facts violating PK constraint / # of rows
blockSize = max. # facts with the same PK
1 Consistent Query Answering for Primary Keys

2 Results

3 Techniques
Consistent Query Answering for Primary Keys

Results

Techniques
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

SELECT DISTINCT c_id
FROM Course, CS_Faculty
WHERE Course.f_id = CS_Faculty.f_id
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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

\[ Q(db) = \{ \text{CS 703, CS 787} \} \ldots \]
Finding consistent answers

**Course**

<table>
<thead>
<tr>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

**CS_Faculty**

<table>
<thead>
<tr>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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

\[ Q(db) = \{ \text{CS 703, CS 787} \} \ldots \]

Data cleaning

\[ Q(rep) \]
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c_id</td>
<td>f_id</td>
</tr>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f_id</td>
<td>f_name</td>
</tr>
<tr>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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

\[ Q(db) = \{\text{CS 703, CS 787}\} \ldots \]

Data cleaning

\[ 2 \times 2 \times 2 \times 1 \text{ repairs} \]

\[ Q(rep) \]
Finding consistent answers

Course

<table>
<thead>
<tr>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

CS_Faculty

<table>
<thead>
<tr>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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

\[Q(db) = \{\text{CS 703, CS 787}\} \ldots\]

Data cleaning

\[2 \times 2 \times 2 \times 1 \text{ repairs}\]

Which answers are guaranteed to be returned on all repairs?

\[Q(rep)\]
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c_id</td>
<td>f_id</td>
<td></td>
</tr>
<tr>
<td>CS 703</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

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

\[ Q(db) = \{ CS 703, CS 787 \} \ldots \]

Data cleaning

\[ 2 \times 2 \times 2 \times 1 \text{ repairs} \]

Which answers are guaranteed to be returned on all repairs?

\[ \bigcap_{\text{rep is a repair of } db} Q(rep) \]
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

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

\[ Q(\text{db}) = \{\text{CS 703, CS 787}\} \ldots \]

Data cleaning

2 × 2 × 2 × 1 repairs

Which answers are guaranteed to be returned on all repairs?

\[ \bigcap_{\text{rep}} Q(\text{rep}) = \{\text{CS 703}\} \]

rep is a repair of db
Finding consistent answers

<table>
<thead>
<tr>
<th>Course</th>
<th>c_id</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>Adam</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Alice</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Bob</td>
</tr>
</tbody>
</table>

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

Q(db) = \{CS 703, CS 787\} …

Data cleaning

2 × 2 × 2 × 1 repairs

Which answers are guaranteed to be returned on all repairs?

\[
\bigcap \]

Q(rep) = \{CS 703\}

rep is a repair of db

Consistent Answer
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Course</th>
<th>f_id</th>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
<td></td>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
<td></td>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
<td></td>
<td>5</td>
<td>Bob</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

\[ Q'(db) = \bigcap \ Q(rep) \]

rep is a repair of db

The original query \( Q \) has a first-order rewriting \( Q' \)
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Course</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

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)

\[ Q'(db) = \bigcap_{\text{rep is a repair of } db} Q(\text{rep}) \]

The original query \( Q \) has a first-order rewriting \( Q' \)
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Course</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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)

\[ Q'(db) = \bigcap Q(rep) \]

rep is a repair of db

The original query \( Q \) has a first-order rewriting \( Q' \)
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Course</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Adam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Alice</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Bob</td>
</tr>
</tbody>
</table>

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)

\[ Q'(db) = \bigcap Q(rep) \]
rep is a repair of db

The original query \( Q \) has a first-order rewriting \( Q' \)
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Course</th>
<th>f_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 703</td>
<td>2</td>
</tr>
<tr>
<td>CS 703</td>
<td>5</td>
</tr>
<tr>
<td>CS 787</td>
<td>3</td>
</tr>
<tr>
<td>CS 787</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CS_Faculty</th>
<th>f_id</th>
<th>f_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Adam</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>

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)

\[ Q'(db) = \bigcap_{rep \text{ is a repair of } db} Q(rep) \]

The original query \( Q \) has a first-order rewriting \( Q' \)
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?
Consistent Query Answering for Primary Keys

Results

Techniques
Acyclic query evaluation

\[ q() \,:=\, \text{Course}(x, y), \text{CS\_Faculty}(y, z) \]
Acyclic query evaluation

\[ q() \rightarrow \text{Course}(x, y), \text{CS}\_\text{Faculty}(y, z) \]
Acyclic query evaluation

$q() \ :- \ Course(x, y), \ CS\_Faculty(y, z)$
Yannakakis [VLDB’81]

The answer to every Boolean acyclic query can be computed in $O(|db|)$. 
Yannakakis [VLDB’81] \hspace{1cm} \textbf{Our result}

consistent answer

The answer to every \textbf{Boolean} acyclic query can be computed in $O(|	ext{db}|)$.  
\wedge
with a pair-pruning join tree (PPJT)
Yannakakis [VLDB’81]  Our result

consistent answer

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

\land

with a pair-pruning join tree (PPJT)

\hspace{1cm}

\text{non-Boolean} \preceq_P^T \text{Boolean}
PPJT is a wide class

- ⊂ Selection, Projection, Join queries
- star/snowflake schema (e.g. TPC-H, TPC-DS)
- two distinct table join
- Every acyclic query in $C_{\text{forest}}$ [ICDT’05, SIGMOD’05] has a PPJT

- no self-joins! [PODS’18,20,22] [PODS’21]
- no aggregation (yet) [ICDE, 2022] [ICDT, 2022]
- no cyclic primary keys joins

\[
q() :\neg R(x, y), S(y, x)
\]

- no non-key to non-key joins

\[
q() :\neg R(x, z), S(y, z)
\]

Z. Fan, P. Koutris, X. Ouyang, J. Wijsen
LinCQA
SIGMOD 2023 13 / 22
PPJT is a wide class

+ ⊂ Selection, Projection, Join queries
+ star/snowflake schema (e.g. TPC-H, TPC-DS)
+ two distinct table join
+ Every acyclic query in $C_{\text{forest}}$ [ICDT’05, SIGMOD’05] has a PPJT

- no self-joins! [PODS’18,20,22] [PODS’21]
- no aggregation (yet) [ICDE, 2022] [ICDT, 2022]
- no cyclic primary keys joins

$$q() \; :\; R(x, y), S(y, x)$$

- no non-key to non-key joins

$$q() \; :\; R(x, z), S(y, z)$$
SELECT
    DISTINCT Posts.Id, 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
Original query + primary key info \(\xrightarrow{\text{LinCQA}}\) Query rewriting
## Setup & Baselines

<table>
<thead>
<tr>
<th>System</th>
<th>Target class</th>
<th>Interm. output</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAVSAT</td>
<td>*</td>
<td>SAT formula</td>
<td>SQL Server &amp; MaxHS</td>
</tr>
<tr>
<td>Conquer</td>
<td>$C_{\text{forest}}$</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
<tr>
<td>Improved Conquesto</td>
<td>SJF FO</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
<tr>
<td>LinCQA</td>
<td>PPJT</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
</tbody>
</table>
## Stackoverflow data

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

<table>
<thead>
<tr>
<th>Table</th>
<th># of rows</th>
<th>inconsistencyRatio</th>
<th>blockSize</th>
<th># of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>14M</td>
<td>0%</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Posts</td>
<td>53M</td>
<td>0%</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>PostHistory</td>
<td>141M</td>
<td>0.001%</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Badges</td>
<td>40M</td>
<td>0.58%</td>
<td>941</td>
<td>4</td>
</tr>
<tr>
<td>Votes</td>
<td>213M</td>
<td>30.9%</td>
<td>1441</td>
<td>6</td>
</tr>
</tbody>
</table>
**Experiments on Stackoverflow**

\[
Q_1 : \text{Posts} \bowtie \text{Votes} \quad Q_2 : \text{Users} \bowtie \text{Badges} \quad Q_3 : \text{Users} \bowtie \text{Posts} \\
Q_4 : \text{Users} \bowtie \text{Posts} \bowtie \text{Comments} \\
Q_5 : \text{Posts} \bowtie \text{PostHistory} \bowtie \text{Votes} \bowtie \text{Comments}
\]

---

**Graph**

- **Original Query**
- **LinCQA**
- **Conquer**
- **FastFO**
- **CAvSAT**

### Results

<table>
<thead>
<tr>
<th>Query</th>
<th># poss.</th>
<th># cons.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>27578</td>
<td>145</td>
</tr>
<tr>
<td>Q2</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>Q3</td>
<td>38320</td>
<td>38320</td>
</tr>
<tr>
<td>Q4</td>
<td>3925</td>
<td>3925</td>
</tr>
<tr>
<td>Q5</td>
<td>1250</td>
<td>1245</td>
</tr>
</tbody>
</table>

---

Z. Fan, P. Koutris, X. Ouyang, J. Wijsen
Experiments on Stackoverflow

\[ Q_1 : \text{Posts} \bowtie \text{Votes} \quad Q_2 : \text{Users} \bowtie \text{Badges} \quad Q_3 : \text{Users} \bowtie \text{Posts} \]

\[ Q_4 : \text{Users} \bowtie \text{Posts} \bowtie \text{Comments} \quad Q_5 : \text{Posts} \bowtie \text{PostHistory} \bowtie \text{Votes} \bowtie \text{Comments} \]

\begin{tabular}{c|c|c|c|c|c}
  & \text{Original Query} & \text{LinCQA} & \text{Conquer} & \text{FastFO} & \text{CAvSAT} \\
\hline
Q1 & N/A & 10 & 1 & 10 & N/A \\
Q2 & N/A & 0 & 1 & 10 & N/A \\
Q3 & N/A & 10 & 2 & 10 & N/A \\
Q4 & N/A & 100 & 101 & 102 & N/A \\
Q5 & N/A & 103 & N/A & Time Out & N/A \\
\end{tabular}

\begin{tabular}{c}
\# poss. & 27578 & 145 & 38320 & 3925 & 1250 \\
\# cons. & 27578 & 145 & 38320 & 3925 & 1245 \\
\end{tabular}
1 Consistent Query Answering for Primary Keys

2 Results

3 Techniques
From PPJT to **FO**-rewriting

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

\[
\begin{align*}
\text{Course}(x, y) & \\
\implies & \\
\text{Faculty}(y, z, “DB”) & \\
\end{align*}
\]

\[
\begin{align*}
\text{Course}_{\text{join}}() & := \text{Course}(x, y), \neg\text{Course}_{fkey}(x) \\
\text{Course}_{fkey}(x) & := \text{Course}(x, y), \neg\text{Faculty}_{\text{join}}(y) \\
\forall \text{Child} : \text{Root}_{fkey}(\vec{x}) & := \text{Root}(\vec{x}, \vec{y}), \neg\text{Child}_{\text{join}}(\vec{\alpha}) \\
\text{Child}_{\text{join}}(\vec{\alpha}) & := \text{Child}(\vec{u}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u}) \\
\text{Faculty}_{\text{join}}(y) & := \text{Faculty}(y, z, w), \neg\text{Faculty}_{fkey}(y) \\
\text{Faculty}_{fkey}(y) & := \text{Faculty}(y, z, w), w \neq “DB” \\
\end{align*}
\]

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($x, y$)

Faculty($y, z, “DB”)"

\[\text{Course}_\text{join}(x) \leftarrow \text{Course}(x, y), \neg \text{Course}_\text{fkey}(x)\]

\[\text{Course}_\text{fkey}(x) \leftarrow \text{Course}(x, y), \neg \text{Faculty}_\text{join}(y)\]

\[\forall \text{Child} : \text{Root}_\text{fkey}(x) \leftarrow \text{Root}(x, y), \neg \text{Child}_\text{join}(\vec{\alpha})\]

\[\text{Child}_\text{join}(\vec{\alpha}) \leftarrow \text{Child}(\vec{u}, \vec{v}), \neg \text{Child}_\text{fkey}(\vec{u})\]

\[\text{Faculty}_\text{join}(y) \leftarrow \text{Faculty}(y, z, w), \neg \text{Faculty}_\text{fkey}(y)\]

\[\text{Faculty}_\text{fkey}(y) \leftarrow \text{Faculty}(y, z, w), w \neq “DB”\]

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”

Course($x$, $y$)

Faculty($y$, $z$, “DB”)

Course$_{join}$() :- Course($x$, $y$), ¬Course$_{fkey}$(x)

Course$_{fkey}$(x) :- Course($x$, $y$), ¬Faculty$_{join}$(y)

$\forall$Child : Root$_{fkey}$($\bar{x}$) :- Root($\bar{x}$, $\bar{y}$), ¬Child$_{join}$($\bar{\alpha}$)

Child$_{join}$($\bar{\alpha}$) :- Child($\bar{u}$, $\bar{v}$), ¬Child$_{fkey}$($\bar{u}$)

Faculty$_{join}$(y) :- Faculty($y$, $z$, $w$), ¬Faculty$_{fkey}$(y)

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

also expressible in SQL!
runs in $O(N)$
Remove a primary key if some tuple with this primary key is “bad”

Course\((x, y)\)

Faculty\((y, z, “DB”)\)

Course\_join() :- Course\((x, y)\), \neg Course\_fkey\((x)\)

Course\_fkey(x) :- Course\((x, y)\), \neg Faculty\_join(y)\)

\(\forall\text{Child} : \text{Root}\_fkey(x) :- \text{Root}(x, y), \neg \text{Child}\_join(\vec{\alpha})\)

Child\_join(\vec{\alpha}) :- Child(\vec{u}, \vec{v}), \neg Child\_fkey(\vec{u})

Faculty\_join(y) :- Faculty(y, z, w), \neg Faculty\_fkey(y)

Faculty\_fkey(y) :- Faculty(y, z, w), w \neq “DB”

also expressible in SQL!
runs in \(O(N)\)
From PPJT to \textbf{FO}-rewriting

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

\begin{align*}
\text{Course}(x, y) & \quad \text{Faculty}(y, z, \text{“DB”}) \\
\text{Course}(x, y) & \quad \neg \text{Course}_{fkey}(x) \\
\text{Course}_{fkey}(x) & \quad \neg \text{Faculty}_{join}(y) \\
\forall \text{Child} : \text{Root}_{fkey}(x) & \quad \neg \text{Child}_{join}(\vec{\alpha}) \\
\text{Child}_{join}(\vec{\alpha}) & \quad \neg \text{Child}_{fkey}(\vec{\alpha}) \\
\text{Faculty}_{join}(y) & \quad \neg \text{Faculty}_{fkey}(y) \\
\text{Faculty}_{fkey}(y) & \quad w \neq \text{“DB”}
\end{align*}

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\(_{(x, y)}\) \rightarrow \text{Faculty\(_{(y, z, "DB")}\)}

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

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

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

\(\text{Child\(_{\text{join\(_{\alpha\)}} \text{ :- Child\(_{(\bar{u}, \bar{v})}, \neg Child\text{fkey\(_{( \bar{u} )}\)}}\)}\)}\)

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

Faculty\(_{\text{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”

Course\( (x, y) \)

Faculty\( (y, z, “DB”) \)

\[
\text{Course}_{\text{join}}() := \text{Course}(x, y), \neg\text{Course}_{\text{fkey}}(x)
\]

\[
\text{Course}_{\text{fkey}}(x) := \text{Course}(x, y), \neg\text{Faculty}_{\text{join}}(y)
\]

\[
\forall \text{Child : Root}_{\text{fkey}}(\vec{x}) := \text{Root}(\vec{x}, \vec{y}), \neg\text{Child}_{\text{join}}(\vec{\alpha})
\]

\[
\text{Child}_{\text{join}}(\vec{\alpha}) := \text{Child}(\vec{u}, \vec{v}), \neg\text{Child}_{\text{fkey}}(\vec{u})
\]

\[
\text{Faculty}_{\text{join}}(y) := \text{Faculty}(y, z, w), \neg\text{Faculty}_{\text{fkey}}(y)
\]

\[
\text{Faculty}_{\text{fkey}}(y) := \text{Faculty}(y, z, w), w \neq “DB”
\]

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”

Course\((x, y)\) 

Faculty\((y, z, “DB”)\)

\[
\begin{align*}
\text{Coursejoin}(\bar{\alpha}) & : \text{Course}\,(x, y), \neg\text{Course}_f\text{key}\,(x) \\
\text{Course}_f\text{key}(x) & : \text{Course}\,(x, y), \neg\text{Faculty}\text{join}(y) \\
\forall\text{Child} : \text{Root}_f\text{key}(\bar{x}) & : \text{Root}\,(\bar{x}, \bar{y}), \neg\text{Child}\text{join}(\bar{\alpha}) \\
\text{Child}_f\text{key}(\bar{u}, \bar{v}) & : \text{Child}\,(\bar{u}, \bar{v}), \neg\text{Child}_f\text{key}(\bar{u}) \\
\text{Faculty}_f\text{key}(y) & : \text{Faculty}\,(y, z, w), \neg\text{Faculty}_f\text{key}(y) \\
\text{Faculty}(y, z, w, “DB”) & \neq \text{“DB”} \\
\end{align*}
\]

also expressible in SQL! runs in \(O(N)\)
From PPJT to \( \textbf{FO} \)-rewriting

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

\[
\text{Course}(x, y) \\
\downarrow
\text{Faculty}(y, z, \text{“DB”})
\]

\[
\begin{align*}
\text{Course}_\text{join}(x) & : \text{Course}(x, y), \neg \text{Course}_\text{fkey}(x) \\
\text{Course}_\text{fkey}(x) & : \text{Course}(x, y), \neg \text{Faculty}_\text{join}(y) \\
\forall \text{Child} : \text{Root}_\text{fkey}(x) & : \text{Root}(x, y), \neg \text{Child}_\text{join}(\bar{\alpha})
\end{align*}
\]

\[
\begin{align*}
\text{Child}_\text{join}(\bar{\alpha}) & : \text{Child}(\bar{u}, \bar{v}), \neg \text{Child}_\text{fkey}(\bar{u}) \\
\text{Faculty}_\text{join}(y) & : \text{Faculty}(y, z, w), \neg \text{Faculty}_\text{fkey}(y) \\
\text{Faculty}_\text{fkey}(y) & : \text{Faculty}(y, z, w), w \neq \text{“DB”}
\end{align*}
\]

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\((x, y)\)

Faculty\((y, z, “DB”)\)

\[\text{Course}_{\text{fkey}}(x) \iff \text{Course}(x, y), \neg \text{Course}_{\text{fkey}}(x)\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

\[\text{Child}_{\text{join}}(\bar{\alpha}) \iff \text{Child}(\bar{u}, \bar{v}), \neg \text{Child}_{\text{fkey}}(\bar{u})\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

\[\text{Faculty}_{\text{join}}(y) \iff \text{Faculty}(y, z, w), \neg \text{Faculty}_{\text{fkey}}(y)\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

\[\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \iff \text{Root}(x, y), \neg \text{Child}_{\text{join}}(\bar{\alpha})\]

also expressible in SQL!

runs in \(O(N)\)
Remove a primary key if some tuple with this primary key is “bad”

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”

\[
\text{Course}(x, y) \\
\downarrow \\
\text{Faculty}(y, z, “DB”) \\
\]

\[
\begin{align*}
\text{Course}_{\text{fkey}}(x) & : \text{Course}(x, y), \neg \text{Course}_{\text{fkey}}(x) \\
\text{Course}_{\text{join}}() & : \text{Course}(x, y), \neg \text{Faculty}_{\text{join}}(y) \\
\forall \text{Child} : \text{Root}_{\text{fkey}}(\vec{x}) & : \text{Root}(\vec{x}, \vec{y}), \neg \text{Child}_{\text{join}}(\vec{\alpha}) \\
\text{Child}_{\text{join}}(\vec{\alpha}) & : \text{Child}(\vec{u}, \vec{v}), \neg \text{Child}_{\text{fkey}}(\vec{u}) \\
\text{Faculty}_{\text{join}}(y) & : \text{Faculty}(y, z, w), \neg \text{Faculty}_{\text{fkey}}(y) \\
\text{Faculty}_{\text{fkey}}(y) & : \text{Faculty}(y, z, w), w \neq “DB”
\end{align*}
\]

also expressible in SQL! runs in \(O(N)\)
From Boolean to non-Boolean

SELECT DISTINCT \textbf{A1}, \textbf{A2} FROM T WHERE A3 = 42

Step 1 Evaluate directly

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Step 2 Reduce to \textbf{Boolean} (using PPJT)

SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = \textcolor{red}{a} AND A2 = \textcolor{red}{b}

if \textit{yes}, then output (\textcolor{red}{a}, \textcolor{red}{b}), otherwise continue

SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = \textcolor{red}{x} AND A2 = \textcolor{red}{y}

...
From Boolean to non-Boolean

SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42

Step 1  Evaluate directly

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

...
## Concluding remarks

<table>
<thead>
<tr>
<th>Acyclic $q$</th>
<th>PPJT</th>
<th>Yannakakis [VLDB’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean $q$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean $q$</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>full $q$ (SELECT *)</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

Thank you!

Xiating Ouyang

xouyang@cs.wisc.edu
Concluding remarks

<table>
<thead>
<tr>
<th>Acyclic q</th>
<th>PPJT</th>
<th>Yannakakis [VLDB’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean q</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean q</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>full q (SELECT *)</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

- *Original Query LinCQA Conquer FastFO CAvSAT*

Q1
Q2
Q3
Q4
Q5

Time Out
N/A

**Diagram:**
- **Xiating Ouyang**
- **xouyang@cs.wisc.edu**

Z. Fan, P. Koutris, X. Ouyang, J. Wijsen

LinCQA

SIGMOD 2023 22 / 22
### Concluding remarks

<table>
<thead>
<tr>
<th>Acyclic q</th>
<th>PPJT</th>
<th>Yannakakis [VLDB’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean q</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean q</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>full q (SELECT *)</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

![Graph with time out](image)

**Thank you!**

Xiating Ouyang  
xouyang@cs.wisc.edu