Accessing Answers to Unions of Conjunctive Queries with Ideal Time Guarantees

Nofar Carmeli
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
  • Unions of conjunctive queries

• Other Evaluation Tasks
  • The tasks
  • Known complexity results
Plan

• **Enumeration**
  • Join queries
    • Self-joins
    • Conjunctive queries
    • Unions of conjunctive queries

• **Other Evaluation Tasks**
  • The tasks
  • Known complexity results
Example: Join Query

Q(\(E, P, R, D\)) \leftarrow \text{Problem}(\(D, R\)), \text{Office}(\(R, P\)), \text{Contact}(\(P, E\))

\{(\(E, P, R, D\))| (\(D, R\)) \in \text{Problem}, (\(R, P\)) \in \text{Office}, (\(P, E\)) \in \text{Contact}\}

<table>
<thead>
<tr>
<th>Email</th>
<th>Person</th>
<th>Room</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:nc@lirmm.fr">nc@lirmm.fr</a></td>
<td>Nofar</td>
<td>5/127</td>
<td>Missing board</td>
</tr>
<tr>
<td><a href="mailto:dc@lirmm.fr">dc@lirmm.fr</a></td>
<td>David</td>
<td>5/129</td>
<td>Moisture</td>
</tr>
</tbody>
</table>
Challenges

- Many answers
- Many intermediate answers

\[ Q_1(x, y, z) \leftarrow R(x, y), S(y, z) \]

\[
\begin{array}{ccc}
    x & y & z \\
    a1 & b1 & c1 \\
    a1 & b1 & c2 \\
    a2 & b1 & c1 \\
    a2 & b1 & c2 \\
    a3 & b1 & c1 \\
    a3 & b1 & c2 \\
\end{array}
\]

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

\[
\begin{array}{ccc}
    x & y & z \\
    a2 & b1 & c1 \\
\end{array}
\]

dangling tuples
Example: Algorithm

\[ Q(E, P, R, D) \leftarrow \text{Problem}(D, R), \text{Office}(R, P), \text{Contact}(P, E) \]
Example: Algorithm Fails

<table>
<thead>
<tr>
<th>Registration</th>
<th>Student</th>
<th>Exam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Anna</td>
<td>algorithms</td>
</tr>
<tr>
<td></td>
<td>Thomas</td>
<td>databases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Staff</th>
<th>Exam</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>algorithms</td>
<td>Pierre</td>
</tr>
<tr>
<td></td>
<td>databases</td>
<td>Marie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COI</th>
<th>Student</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Thomas</td>
<td>Pierre</td>
</tr>
<tr>
<td></td>
<td>Anne</td>
<td>Marie</td>
</tr>
</tbody>
</table>

\[ Q(S, E, P) \leftarrow \text{Registration}(S, E), \text{Staff}(E, P), \text{COI}(S, P) \]

Database

- Anna
- Thomas
- Pierre
- Marie

- algorithms
- databases

No query answers

Query

- S
- E
- P
Complexity Guarantees

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

• Possibly: output $\gg$ input
  (Polynomial number of answers)

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

• RAM model
• We allow log factors
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
Research Question

• Goal: Given a query, what is the most efficient algorithm?
• Type of results: Can we solve a task for a given query in a given time complexity?

Yes / No

algorithm
conditional lower bound
Acyclicity

• A query that has a join tree is called **acyclic**

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

**Join Tree:**
1. a node for every atom
2. tree
3. For every variable: the nodes containing it form a subtree

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

**Diagram:**
- **Acyclic**
- **Cyclic**
Given a join query $Q$, if $Q$ is acyclic, $Q \in \text{Enum}<\text{lin, const}>$; if $Q$ is cyclic, $Q \notin \text{Enum}<\text{lin, const}>$. * no self-joins, assuming sHyperclique or Zero-Clique
Acyclic Joins

• 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:
   \( r_{\text{parent}} \leftarrow r_{\text{parent}} \bowtie r_{\text{child}} \)

2. Root-to-leaf:
   \( r_{\text{child}} \leftarrow r_{\text{child}} \bowtie r_{\text{parent}} \)

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

for t1 in \( R_1 \):
  for t2 in \( R_2 \) matching t1:
    for t3 in \( R_3 \) matching t1, t2:
      for t4 in \( R_4 \) matching t1, t2, t3:
        output t1, t2, t3, t4
Given a join query Q,

- If Q is acyclic, $Q \in \text{Enum}<\text{lin, const}>

- If Q is cyclic, $Q \notin \text{Enum}<\text{lin, const}>$*

* no self-joins, assuming sHyperclique or Zero-Clique

[BaganDurandGrandjean CSL’2007]
[Brault-Baron 2013]
[Bringmann, C, Mengel 2022]
Example: Algorithm Fails

\[ Q_\Delta(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(x, z) \]
\[ Q(S, E, P) \leftarrow Registration(S, E), Staff(E, P), COI(S, P) \]

No query answers
Example: Conditional Lower Bound

Assumption: cannot detect triangles in a graph in linear time

\[ Q \triangleq x, y, z \leftarrow R_1(x, y), R_2(y, z), R_3(x, z) \]

edges \((a, b)\) with \(a < b\)

\[
\begin{array}{c|c|c}
  x & y & z \\
  \\
  1 & 2 & 4 \\
\end{array}
\quad
\begin{array}{c|c}
  R_1 = R_2 = R_3 \\
  1 & 2 \\
  1 & 3 \\
  1 & 4 \\
  2 & 4 \\
\end{array}
\]

first answer in linear time \(\implies\) triangle in linear time \(\implies\) not possible

[Brault-Baron 13]
sHyperclique Hypothesis

• \((k, k - 1)\)-hyperclique: \(k\) vertices, each \(k - 1\) of them form an edge.

\[
\begin{array}{ccc}
1 & 2 & 3 \\
\end{array}
\]

• sHyperclique Hypothesis:
\(\forall k \geq 3\), deciding the existence of a \((k, k - 1)\)-hyperclique in a hypergraph with \(m\) edges cannot be done in time \(O(m)\).

• Lemma:
A cyclic hypergraph contains an induced \(k\)-cycle or an induced \((k, k - 1)\)-hyperclique for some \(k \geq 3\).
Dichotomy

- Given a join query $Q$,

  - If $Q$ is acyclic, $Q \in \text{Enum}<\text{lin, const}>
  
  - If $Q$ is cyclic, $Q \notin \text{Enum}<\text{lin, const}>^*$

* no self-joins, assuming sHyperclique or Zero-Clique

[BaganDurandGrandjean CSL’2007]
[Brault-Baron 2013]
[Bringmann, C, Mengel 2022]
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 $O(1)$

$n = \text{size of input database}$
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: $O(n^c) / O(n)$
  • Modified memory: everything $/ O(n)$
  • Modified memory during enumeration: everything $/ \ldots / O(1)$
• 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

$n = \text{size of input database}$
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
  • Unions of conjunctive queries

• Other Evaluation Tasks
  • The tasks
  • Known complexity results
• Given a join query $Q$, 

If $Q$ is acyclic, $Q \in \text{Enum}<\text{lin, const}>$

If $Q$ is cyclic, $Q \notin \text{Enum}<\text{lin, const}>$

* no self-joins, assuming $s$Hyperclique or Zero-Clique

[DaganDurandGrandjean CSL’2007]
[Brault-Baron 2013]
[Bringmann, C, Mengel 2022]
Example 1

\[ Q(s_1, s_2, \text{room, grade}) \leftarrow \text{Seating}(\text{room, } s_1), \text{Seating}(\text{room, } s_2), \text{Grade}(s_1, \text{grade}), \text{Grade}(s_2, \text{grade}) \]
Lower Bound: Cyclic Joins

Assumption: cannot detect triangles in a graph in linear time

Assumption: cannot detect triangles in a graph in linear time

\( Q_\Delta: \)

\[
\begin{align*}
Q_\Delta & \leftarrow R_1(x, y), R_2(y, z), R_3(x, z) \\
& \text{Cyclic: } Q_\Delta(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(x, z)
\end{align*}
\]

first answer in linear time \( \Rightarrow \) triangle in linear time \( \Rightarrow \) not possible

[Brault-Baron 13]
Lower Bound: Cyclic Joins

Assumption: cannot detect triangles in a graph in linear time

Construction:

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

First answer in linear time $\Rightarrow$ triangle in linear time $\Rightarrow$ not possible
Algorithm [C Segoufin PODS’2023]

Query

\[ Q(x, u, v, y) \leftarrow R(x, u), R(u, y), R(x, v), R(v, y) \]

Database

\[ R \]

\| a & b \\
\| b & c \\
\| b & d \\
\| c & e \\
\| d & e \\

\[ \alpha = \text{empty dictionary} \]

for answer \((x, u, y)\) to \(I\) :

output \((x, u, u, y)\)

for \(v\) in \(\alpha(x, y)\) :

output \((x, u, v, y)\)

output \((x, v, u, y)\)

\[ \alpha(x, y).\text{insert}(u) \]

Answers

\[ I \text{ answers} \]

\| a & b & c \\
\| a & b & d \\
\| b & c & e \\
\| b & d & e \\

\[ Q \text{ answers} \]

\| b & c & d & e \\
\| a & b & b & c \\
\| a & b & b & d \\
\| b & c & c & e \\
\| b & d & d & e \\

Image \(I\)
Examples: Full CQs

∈ \text{Enum}<\text{lin, const}>

∉ \text{Enum}<\text{lin, const}>

* assuming sTriangle
Examples: Full CQs

\[ E \quad E \quad E \quad E \]

\[ \in \text{Enum}<\text{lin, const}> \quad \notin \text{Enum}<\text{lin, const}> \]

\[ s\text{Triangle} \]

* assuming sTriangle
Hardness Proof

\[ R(x, y) \leftarrow E \]

\[
\begin{array}{cc}
1 & 2 \\
1 & 3 \\
1 & 4 \\
2 & 4 \\
\end{array}
\]

\[ R(y, z) \leftarrow E \]

\[
\begin{array}{cc}
1 & 2 \\
1 & 3 \\
1 & 4 \\
2 & 4 \\
\end{array}
\]

\[ R(x, w) \leftarrow E \]

\[
\begin{array}{cc}
1 & 2 \\
1 & 3 \\
1 & 4 \\
2 & 4 \\
\end{array}
\]

\[ R(w, z) \leftarrow = \]

\[
\begin{array}{cc}
1 & 1 \\
2 & 2 \\
3 & 3 \\
4 & 4 \\
\end{array}
\]
Hardness Proof

Works because Q is a core!
Hardness Proof Fails

### Graph Representation

- **$R(x, y) \leftarrow E$**
  - $(1, x)$, $(2, y)$
  - $(1, x)$, $(1, x)$
  - $(1, x)$, $(1, x)$
  - $(2, x)$, $(4, y)$

- **$R(y, z) \leftarrow E$**
  - $(1, y)$, $(2, z)$
  - $(1, y)$, $(1, y)$
  - $(1, y)$, $(1, y)$
  - $(2, y)$, $(4, z)$

- **$R(x, w) \leftarrow E$**
  - $(1, x)$, $(2, w)$
  - $(1, x)$, $(1, x)$
  - $(1, x)$, $(1, x)$
  - $(2, x)$, $(4, w)$

- **$R(w, z) \leftarrow E$**
  - $(1, w)$, $(1, z)$
  - $(2, w)$, $(2, z)$
  - $(3, w)$, $(3, z)$
  - $(4, w)$, $(4, z)$

### Union

<table>
<thead>
<tr>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1, x)$, $(2, y)$</td>
</tr>
<tr>
<td>$(1, x)$, $(1, x)$</td>
</tr>
<tr>
<td>$(1, x)$, $(1, x)$</td>
</tr>
<tr>
<td>$(2, x)$, $(4, y)$</td>
</tr>
<tr>
<td>$(1, y)$, $(2, z)$</td>
</tr>
<tr>
<td>$(1, y)$, $(1, y)$</td>
</tr>
<tr>
<td>$(1, y)$, $(1, y)$</td>
</tr>
<tr>
<td>$(2, y)$, $(4, z)$</td>
</tr>
<tr>
<td>$(1, w)$, $(1, z)$</td>
</tr>
<tr>
<td>$(2, w)$, $(2, z)$</td>
</tr>
<tr>
<td>$(3, w)$, $(3, z)$</td>
</tr>
<tr>
<td>$(4, w)$, $(4, z)$</td>
</tr>
<tr>
<td>$(1, x)$, $(2, w)$</td>
</tr>
<tr>
<td>$(1, x)$, $(3, w)$</td>
</tr>
<tr>
<td>$(1, x)$, $(4, w)$</td>
</tr>
<tr>
<td>$(2, x)$, $(4, w)$</td>
</tr>
</tbody>
</table>

...
Sufficient and Necessary Conditions

Let $Q$ be a full CQ.

- If $Q$ is a mirror, then $Q \in \text{Enum}\langle\text{lin, const}\rangle$

- If $Q$ has a cyclic core, then $Q \notin \text{Enum}\langle\text{lin, const}\rangle$ *

* assuming sHyperclique
Examples: Full CQs

Unlike the self-join-free case, may affect the complexity:
• reordering variables inside an atom

\[ E = E \]

\[ \in \text{Enum}<\text{lin, const}> \]

\[ \notin \text{Enum}<\text{lin, const}> \ast \]

* assuming sTriangle
Examples: Full CQs

Unlike the self-join-free case, may affect the complexity:
• reordering variables inside an atom
• introducing unary atoms

* assuming sTriangle
Examples: Full CQs

Unlike the self-join-free case, may affect the complexity:

- reordering variables inside an atom
- introducing unary atoms

\[ \in \text{Enum}_{\text{lin, const}} \]

\[ \notin \text{Enum}_{\text{lin, const}} \] * assuming sTriangle

\[ \not\in \text{Enum}_{\text{lin, const}} \] *
Examples: Full CQs

Unlike the self-join-free case, may affect the complexity:
• reordering variables inside an atom
• introducing unary atoms
• introducing ‘spikes’
Examples: Full CQs

\[x \leq 2 + 2 \leq 2 + \text{number of simple solutions found}\]

\[\in \text{Enum}\langle \text{lin, const} \rangle\]

\[\notin \text{Enum}\langle \text{lin, const} \rangle\]

* assuming sTriangle
Examples: Full CQs

# non-triangle solutions \( \leq |E_0|^2 \)
Vertex-Unbalanced Triangle Detection

- An $\alpha$-unbalanced tripartite graph has vertex sets $|V_1| = n$ and $|V_2| = |V_3| = \Theta(n^\alpha)$

- Hypothesis: $\forall$ constant $\alpha \in (0,1]$, it is not possible to test the existence of a triangle in an $\alpha$-unbalanced tripartite graph in time $O(n^{1+\alpha})$.

Remark: this hypothesis is also connected to UCQs [Bringmann, C; 22]
Hypotheses

**sTriangle**: The existence of a triangle in an undirected graph with $m$ edges cannot be decided in time $O(m)$

**Triangle**: The existence of a triangle in an undirected graph with $n$ nodes cannot be decided in time $O(n^2)$

**VUTD (Vertex-Unbalanced Triangle Detection) [Bringmann, C; 22]**:

$\forall \alpha \in (0,1]$ the existence of a triangle in a tripartite graph with $|V_1| = n$ and $|V_2| = |V_3| = \Theta(n^\alpha)$ cannot be decided in time $O(n^{1+\alpha})$

**sHyperclique**: $\forall k \geq 3$ the existence of a $k$-hyperclique in a $(k - 1)$-uniform hypergraph with $m$ edges cannot be decided in time $O(m)$

**Hyperclique**: $\forall k \geq 3$ the existence of a $k$-hyperclique in a $(k - 1)$-uniform hypergraph with $n$ nodes cannot be decided in time $O(n^{k-1})$
Examples: Full CQs

# non-triangle solutions $\leq |E_0|^2$

\[ \in \text{Enum}<\text{lin},\text{const}> \]

\[ \notin \text{Enum}<\text{lin},\text{const}> * \]

* assuming sTriangle
** assuming VUTD
Examples: Full CQs

\[ \in \text{Enum<lin, const>} \]

\[ \notin \text{Enum<lin, const>} \]

Examples: Full CQs

\[ \in \text{Enum<lin, const>} \]

\[ \notin \text{Enum<lin, const>} \]

* assuming sTriangle

** assuming VUTD
Plan

- Enumeration
  - Join queries
    - Self-joins
  - Conjunctive queries
    - Unions of conjunctive queries
- Other Evaluation Tasks
  - The tasks
  - Known complexity results
Example: Query

**Problem**

<table>
<thead>
<tr>
<th>Description</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moisture</td>
<td>5/129</td>
</tr>
<tr>
<td>Broken ceiling</td>
<td>Cafeteria</td>
</tr>
<tr>
<td>Missing board</td>
<td>5/127</td>
</tr>
</tbody>
</table>

**Office**

<table>
<thead>
<tr>
<th>Room</th>
<th>Person</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/127</td>
<td>Nofar</td>
<td>9590</td>
</tr>
<tr>
<td>5/127</td>
<td>Nofar</td>
<td>9591</td>
</tr>
<tr>
<td>5/128</td>
<td>Florent</td>
<td>6548</td>
</tr>
<tr>
<td>5/128</td>
<td>Guillaume</td>
<td>6548</td>
</tr>
<tr>
<td>5/129</td>
<td>David</td>
<td>7544</td>
</tr>
<tr>
<td>5/129</td>
<td>Akira</td>
<td>7544</td>
</tr>
</tbody>
</table>

**Contact**

<table>
<thead>
<tr>
<th>Person</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nofar</td>
<td><a href="mailto:nc@lirmm.fr">nc@lirmm.fr</a></td>
</tr>
<tr>
<td>Florent</td>
<td><a href="mailto:ft@lirmm.fr">ft@lirmm.fr</a></td>
</tr>
<tr>
<td>Guillaume</td>
<td><a href="mailto:gpk@lirmm.fr">gpk@lirmm.fr</a></td>
</tr>
<tr>
<td>David</td>
<td><a href="mailto:dc@lirmm.fr">dc@lirmm.fr</a></td>
</tr>
</tbody>
</table>

**Conjunctive query**

\[
\{(E, P, R, D, N) \mid (D, R) \in \text{Problem}, (R, P, N) \in \text{Office}, (P, E) \in \text{Contact}\}
\]

**Join query:**

\[
Q(E, P, R, D, N) \leftarrow \text{Problem}(D, R), \text{Office}(R, P, N), \text{Contact}(P, E)
\]

**Email**

<table>
<thead>
<tr>
<th>Email</th>
<th>Person</th>
<th>Room</th>
<th>Description</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:nc@lirmm.fr">nc@lirmm.fr</a></td>
<td>Nofar</td>
<td>5/127</td>
<td>Missing board</td>
<td>9590</td>
</tr>
<tr>
<td><a href="mailto:nc@lirmm.fr">nc@lirmm.fr</a></td>
<td>Nofar</td>
<td>5/127</td>
<td>Missing board</td>
<td>9591</td>
</tr>
<tr>
<td><a href="mailto:dc@lirmm.fr">dc@lirmm.fr</a></td>
<td>David</td>
<td>5/129</td>
<td>Moisture</td>
<td>7544</td>
</tr>
</tbody>
</table>
Handling Projection

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

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

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>y1</td>
<td>z1</td>
<td>w1</td>
</tr>
<tr>
<td>x1</td>
<td>y1</td>
<td>z2</td>
<td>w2</td>
</tr>
<tr>
<td>x1</td>
<td>y2</td>
<td>z2</td>
<td>w2</td>
</tr>
<tr>
<td>x2</td>
<td>y2</td>
<td>z2</td>
<td>w2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>z</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>z1</td>
<td>w1</td>
</tr>
<tr>
<td>y1</td>
<td>z2</td>
<td>w2</td>
</tr>
<tr>
<td>y2</td>
<td>z2</td>
<td>w2</td>
</tr>
<tr>
<td>y3</td>
<td>z1</td>
<td>w1</td>
</tr>
<tr>
<td>y4</td>
<td>z3</td>
<td>w2</td>
</tr>
</tbody>
</table>

[Diagram showing join tree and data tuples]

\[ x, y \]
\[ y, z \]
\[ z, w \]
\[ w \]
Handling Projection

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

\[ Q_1(y, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), R_4(w) \]
\[ Q_2(x, y, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), R_4(w) \]
An acyclic CQ has a graph with:

A free-connex CQ also requires:

1. a node for every atom
   possibly also subsets
2. tree
   the nodes containing it form a subtree
3. for every variable:
4. a subtree with exactly the free variables

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

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

[Bagan, Durand, Grandjean; CSL 07]
Eliminating Projection

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

• Given a conjunctive query Q,

If Q is acyclic free-connex, $Q \in \text{Enum}\langle\text{lin, const}\rangle$

If Q is acyclic not free-connex, $Q \notin \text{Enum}\langle\text{lin, const}\rangle$*

If Q is cyclic, $Q \notin \text{Enum}\langle\text{lin, const}\rangle$**

* no self-joins, assuming sBMM
** no self-joins, assuming sHyperclique
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

\begin{align*}
R_1 & = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 2 \end{pmatrix} \\
R_2 & = \begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix} \\
Q & = \begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}
\end{align*}

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

\[
O(n^2) \text{ preprocessing} + O(1) \text{ delay} = O(n^2) \text{ total} \implies \text{not possible}
\]

Intractability cause: free-path \( x \rightarrow y \rightarrow z \)
Hypotheses

**sBMM:** Boolean matrices cannot be multiplied in linear time in the number of the 1 entries

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

**sTriangle:** The existence of a triangle in an undirected graph with $m$ edges cannot be decided in time $O(m)$

**Triangle:** The existence of a triangle in an undirected graph with $n$ nodes cannot be decided in time $O(n^2)$

**VUTD (Vertex-Unbalanced Triangle Detection):**

$\forall \alpha \in (0,1]$ the existence of a triangle in a tripartite graph with $|V_1| = n$ and $|V_2| = |V_3| = \Theta(n^\alpha)$ cannot be decided in time $O(n^{1+\alpha})$

**sHyperclique:** $\forall k \geq 3$ the existence of a $k$-hyperclique in a $(k - 1)$-uniform hypergraph with $m$ edges cannot be decided in time $O(m)$

**Hyperclique:** $\forall k \geq 3$ the existence of a $k$-hyperclique in a $(k - 1)$-uniform hypergraph with $n$ nodes cannot be decided in time $O(n^{k-1})$
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
    • **Unions of conjunctive queries**
• Other Evaluation Tasks
  • The tasks
  • Known complexity results
Example: Union of CQs

\[ Q_1(\text{post}, p_2, p_3) \leftarrow \text{Posts}(\text{post}, p_1), \text{Followers} (p_1, p_2), \text{Friends} (p_2, p_3) \]
\[ \bigcup \]
\[ Q_2(\text{post}, p_1, p_2) \leftarrow \text{Posts}(\text{post}, p_1), \text{Followers} (p_1, p_2) \]

<table>
<thead>
<tr>
<th>Post</th>
<th>Person 1</th>
<th>Person 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazing vacation</td>
<td>Bob</td>
<td>Carol</td>
</tr>
<tr>
<td>Amazing vacation</td>
<td>Alice</td>
<td>Bob</td>
</tr>
<tr>
<td>Angry post</td>
<td>Carol</td>
<td>Dafni</td>
</tr>
<tr>
<td>Angry post</td>
<td>Bob</td>
<td>Carol</td>
</tr>
</tbody>
</table>

Post: due to \( Q_1 \) or \( Q_2 \)
Person 1: due to \( Q_1 \)
Person 2: due to \( Q_2 \)
Cases for UCQs

- All CQs are Easy
- Some Easy, Some Hard
- All CQs are Hard
Easy $\cup$ Easy Is Always Easy

Generated (lookup):

\[\text{a b c d} \quad \text{x}\]

Queue:

\[\text{a c b d x}\]

Output:

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

while $B$.hasNext():
    print $B$.next()

prints $A \setminus B$

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

prints $B$
Cases for UCQs

[Ü, Kröll; PODS 19]

All CQs are Easy

All CQs are Hard

Some Easy, Some 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

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

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

Intractability cause: free-path $x \rightarrow y \rightarrow z$
Why this isn’t hard

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

\[ \cup \]

\( Q_2(a, b, c) \leftarrow R_1(a, b), R_2(b, c) \)

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

The hardness results do not hold within a union
Example: Tractable Union

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

\[ Q_2(a, b, c) \leftarrow R_1(a, b), R_2(b, c) \]

\[ Q_1^+(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), Q_2(x, y, z) \]

\[ \in \text{Enum}\langle\text{lin, const}\rangle \]

Step | Output | Side Effect
--- | --- | ---
1 | Solve \( Q_2 \) | Find \( R_1 \Join R_2 \)
2 | Solve \( Q_1^+ \) | \( Q_1 \)
Cheater’s Lemma

If an enumeration problem can be solved with:

- Usually constant delay
- Almost no duplicates

Then*, it is $\in \text{Enum}^{<\text{lin},\text{const}>}$

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

* using polynomial space

[C, Kröll; PODS 19]
Complexity Measures

- (Instance-optimal) linear total time
  - Total time $O(n + N)$

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

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

$n = \text{input size}, \ N = \text{output size}$
Cases for UCQs

All CQs are Easy
- always easy

Some Easy, Some Hard
- sometimes hard
- sometimes easy

All CQs are Hard
- sometimes hard
- sometimes easy

[C, Kröll; PODS 19]
Hard $\cup$ Hard = Easy  

- Example: CQs with **isomorphic bodies**.

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

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Output</th>
<th>Side Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$Q_1'$</td>
<td>$\subseteq Q_1$</td>
</tr>
<tr>
<td>2</td>
<td>$Q_2^+$</td>
<td>$Q_2$</td>
</tr>
<tr>
<td>3</td>
<td>$Q_1^+$</td>
<td>$Q_1$</td>
</tr>
</tbody>
</table>

[\text{C, Kröll; PODS 19}]
Dichotomy for Unions of 2 CQs

- Given a union of two conjunctive queries Q,

If Q has an acyclic free-connex union extension,
\[ Q \in \text{Enum}\langle\text{lin, const}\rangle \]

Otherwise,
\[ Q \notin \text{Enum}\langle\text{lin, const}\rangle \]*

* no self-joins, assuming VUTD

There exists a family of UCQs with no free-connex union extensions s.t.
VUTD hypothesis holds \(\Leftrightarrow\) no query of the family is in \(\text{Enum}\langle\text{lin, const}\rangle\)
Example: Intractable Union (Assuming VUTD)

\[ Q_1(x, y, w) \leftarrow R_1(x, z), R_2(z, y), R_3(y, w) \]
\[ Q_2(x, y, w) \leftarrow R_1(x, t_1), R_2(t_2, y), R_3(w, t_3) \]

\[ \text{VUTD (Vertex-Unbalanced Triangle Detection)}: \]
\[ \forall \alpha \in (0, 1] \text{ the existence of a triangle in a tripartite graph} \]
\[ \text{with } |V_1| = n \text{ and } |V_2| = |V_3| = \Theta(n^\alpha) \text{ cannot be decided in time } O(n^{1+\alpha}) \]

- \( Q_2 \) can’t help \( Q_1 \) : it doesn’t provide \( z \)
- Construction: assigns large vertex set to \( z \), small vertex sets to \( x \) and \( y \), constant \( \bot \) to \( w \)
- Answers:
  - Ignore answers to \( Q_2 \) (there are \( O(n^{2\alpha}) \) such answers)
  - Check whether answers to \( Q_1 \) form an edge (if so, triangle detected)
Beyond 2 CQs: almost open problem

- Example:
  \[ Q_1(x_1, x_2, x_3), Q_2(x_1, x_2, z), Q_3(x_1, x_3, z), Q_4(x_2, x_3, z) \leftarrow R_1(x_1, z), R_2(x_2, z), R_3(x_3, z) \]

- \( Q_1 \) hard: reduce from matrix multiplication to \( x_1, z, x_2 \)
- Others easy: free-connex acyclic

- Cannot use matrix multiplication reduction (others have too many answers)
- Can reduce from 4-clique: detection in \( O(n^3) \) time
Beyond 2 CQs: open problem

- Example: \( Q_1(x_1, x_2, x_3, x_4), Q_2(x_1, x_2, x_3, z), Q_3(x_1, x_2, x_4, z), \)
  \( Q_4(x_1, x_3, x_4, z), Q_5(x_2, x_3, x_4, z) \leftarrow \)
  \( R_1(x_1, z), R_2(x_2, z), R_3(x_3, z), R_4(x_4, z) \)

- \( Q_1 \) hard: reduce from matrix multiplication to \( x_1, z, x_2 \)
- Others easy: free-connex

- Cannot use matrix multiplication reduction (others have too many answers)
- Cannot reduce from 5-clique (it is not a valid assumption that we can't solve the \((k + 1)\)-clique problem in time \( O(n^k) \) for large \( k \) values).
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
  • Unions of conjunctive queries

• Other Evaluation Tasks
  • The tasks
  • Known complexity results
Overview of Tasks

- Sampling
- Random-ordered enumeration
- Ranked enumeration
- Top k
Quantile Computation via Ranked Access

What is the median monthly cost of an employee?

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

- Given \( i \), returns the \( i \)th answer or “out of bound”.
- Ranked Access: user-specified order
Goal: efficient ranked access

input: database instance

problem: query + order

$Q(x, y, z) \leftarrow R(x, z), S(z, y)$
Lexicographic $x > y > z$

preprocessing

index

The 57th answer is $(c_1, c_2, c_3)$

answer

access

data structure
Overview of Tasks

- ranked access
- quantile computation
- random-ordered enumeration
- sampling
- ranked enumeration
- enumeration
- top k
- enumeration
Definition: Access Tasks

- Given $i$, returns the $i^{th}$ answer or “out of bound”.
- Ranked Access: user-specified order
- Direct Access: no constraints on the ordering used
Overview of Tasks

- ranked access
- direct access
- quantile computation
- counting
- random-ordered enumeration
- sampling
- ranked enumeration
- top k

* with log time per answer after linear preprocessing
Counting via Direct Access

• Assumption: the number of answers is bounded by a polynomial
• Direct Access returns “out of bound” if needed
  • Allows checking if $|\text{answers}| > k$
• Binary search for $|\text{answers}|$
  • Requires $O(\log(|\text{answers}|))$ calls for Direct Access
  • If $|\text{answers}|$ is polynomial, $\log(|\text{answers}|) = O(\log(\text{input}))$
  • This takes $O(\log(\text{input}) \cdot \text{cost(access)})$ time
Overview of Tasks

- Ranked access
  - Quantile computation
  - Direct access
    - Counting
    - Random-ordered enumeration
  - Sampling
  - Enumeration
    - Top k

* with log time per answer after linear preprocessing
Random-Ordered Enumeration via Direct Access

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

1) Find the number $N$ of answers

   6

2) Find a random permutation of $1,\ldots,N$

   5  6  4  2  1  3

3) Direct access to answers

   |   |   |   |   |   |   |
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |
   |   |   |   |   |   |   |

   [C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]
Fisher-Yates Shuffle

Place 1, ..., n in array
For i in 1, ..., n:
  choose j randomly from \{i, ..., n\}
  replace i and j

\[
\begin{array}{ccccc}
  3 & 2 & 3 & 4 & 5 \\
  i & i & ij & i & ij
\end{array}
\]
Fisher-Yates Shuffle

**Constant delay variant:**

place 1, ..., n in array *(lazy initialization)*
for i in 1, ..., n:
choose j randomly from {i, ..., n}
replace i and j
print $a[i]$
Overview of Tasks

- ranked access
  - quantile computation
  - counting
  - direct access
    - random-ordered enumeration
    - sampling
    - enumeration
      - ranked enumeration
      - top k
        - with log time per answer after linear preprocessing
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
  • Unions of conjunctive queries

• Other Evaluation Tasks
  • The tasks
  • Known complexity results
Can be solved efficiently* for all unions of free-connex CQs?

[\text{C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20}]

* with log time per answer after linear preprocessing
Example: Difficult Counting

\[ Q_1(x, y, z) \leftarrow R(x, y), S(y, z) \]
\[ Q_2(x, y, z) \leftarrow S(y, z), T(x, z) \]

- \( Q_1 \cap Q_2(x, y, z) \leftarrow R(x, y), S(y, z), T(x, z) \) cyclic
  - Cannot determine whether \(|Q_1 \cap Q_2| > 0\) in linear time, assuming sTriangle
- \(|Q_1 \cap Q_2| = |Q_1| + |Q_2| - |Q_1 \cup Q_2|\)
  - can be computed in linear time
Can be solved efficiently* for all unions of free-connex CQs?

[Yes with log time per answer after linear preprocessing]

[No]

quantile computation

direct access

random-ordered enumeration

sampling

counting

enumeration

ranked access

ranked enumeration

top k

~Yes (log in expectation)

* with log time per answer after linear preprocessing

[C, Zeevi, Berkholz, Kimelfeld, Schweikardt; PODS 20]
Can be solved efficiently* for all free-connex CQs?

* with log time per answer after linear preprocessing
Can be solved efficiently* for all free-connex CQs?

* with log time per answer after linear preprocessing

[Brault-Baron 2013]
Can be solved efficiently* for all free-connex CQs?

For lexicographic orders:

C, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21

Tziavelis, Gatterbauer, Riedewald; VLDB 21

Yes

No

ranked access

direct access

ranked enumeration

top k

counting

random-ordered enumeration

sampling

enumeration

quantile computation

* with log time per answer after linear preprocessing
Can be solved efficiently* for all free-connex CQs?

For sum of weights orders:

[\textbf{No}]

- ranked access
  - quantile computation
  - direct access
  - counting
  - random-ordered enumeration
  - sampling
  - ranked enumeration

[\textbf{Yes}]

- top k
- enumeration

* with log time per answer after linear preprocessing

\textbf{[C, Tziavelis, Gatterbauer, Kimelfeld, Riedewald; PODS 21]}
\textbf{[Tziavelis, Gatterbauer, Riedewald; VLDB 21]}
Given a conjunctive query $Q$,

If $Q$ is acyclic with an atom containing all free variables, $Q \in \Sigma_{\text{WeightAccess}}<\text{lin,log}>

Otherwise, $Q \notin \Sigma_{\text{WeightAccess}}<\text{lin,log}>$

* no self-joins, assuming 3SUM and sHyperclique
Hardness

**3SUM hypothesis**
given 3 sets of integers $|A| = |B| = |C| = n$,
deciding $\exists a \in A, b \in B, c \in C$ s.t. $a + b + c = 0$
cannot be done in time $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$

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

$\notin \Sigma \text{WeightAccess} < n^{2-\varepsilon}, n^{1-\varepsilon} >$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$z$</td>
<td>$w$</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0</td>
<td>$b_1$</td>
<td>$a_1 + b_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0</td>
<td>$b_1$</td>
<td>$a_2 + b_1$</td>
</tr>
<tr>
<td>0</td>
<td>$b_1$</td>
<td>$a_1 + b_1$</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$b_2$</td>
<td>$a_2 + b_1$</td>
<td></td>
</tr>
<tr>
<td>$a_1$</td>
<td>0</td>
<td>$b_2$</td>
<td>$a_1 + b_2$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0</td>
<td>$b_2$</td>
<td>$a_2 + b_2$</td>
</tr>
</tbody>
</table>

Binary search for $-c \ (\forall c)$
(log number of access calls)
Plan

• Enumeration
  • Join queries
    • Self-joins
  • Conjunctive queries
  • Unions of conjunctive queries

• Other Evaluation Tasks
  • The tasks
  • Known complexity results