Trade-offs in Static and Dynamic Query Evaluation

Ahmet Kara, Milos Nikolic
Dan Olteanu, and Haozhe Zhang

fdbresearch.github.io

Dagstuhl Seminar 20071, Feb. 2020
Foundations of Composite Event Recognition
We are interested in the trade-off between: preprocessing time - enumeration delay - (update time)
We are interested in the trade-off between:

- preprocessing time
- enumeration delay
- (update time)
Static and Dynamic Query Evaluation

Static Query Evaluation

query → database → preprocessing → data structure → enumeration → query result

preprocessing time

data structure

enumeration delay
Static and Dynamic Query Evaluation

Static Query Evaluation

Query database → preprocessing time → data structure → enumeration delay → query result

Dynamic Query Evaluation

Query database → preprocessing time → data structure → enumeration delay → query result

We are interested in the trade-off between:
- preprocessing time
- enumeration delay
- update time
We are interested in the trade-off between:

- preprocessing time
- enumeration delay
- (update time)
We are interested in the trade-off between:

- preprocessing time
- enumeration delay
- single-tuple update maintenance

Static Query Evaluation

- Query
- Database
- Preprocessing
- Data Structure
- Enumeration
- Result

Dynamic Query Evaluation

- Query
- Database
- Preprocessing
- Data Structure
- Enumeration
- Result
We are interested in the trade-off between:

- preprocessing time
- enumeration delay
- update time

Static Query Evaluation

Dynamic Query Evaluation

Single-tuple maintenance update time
We are interested in the trade-off between:

preprocessing time - enumeration delay - (update time)
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

conjunctive
\(O(N^w)/O(1)\) [TODS ’15]

\[\begin{array}{c}
\text{log}_N \text{ delay} \\
\text{log}_N \text{ preprocessing time}
\end{array}\]

conjunctive

static width \(w = s^\uparrow\) [TODS ’15] or faqw [PODS ’16]
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

- **conjunctive**
  \( O(N^w)/O(1) \) [TODS '15]

- \((\alpha)-acyclic\)
  \( O(N)/O(N) \) [CSL '07]

- **acyclic**
  \( O(N)/O(N) \)

- **hierarchical**
  \( O(N^{1+(w-1)\varepsilon})/O(N^{1-\varepsilon}) \) 
  \( \varepsilon \in [0, 1] \)

static width \( w = s^{\uparrow} \) [TODS '15] or faqw [PODS '16]
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

- **conjunctive**
  - $O(N^w)/O(1)$
  - [TODS '15]

- **(α)-acyclic**
  - $O(N)/O(N)$
  - [CSL '07]

- **free-connex**
  - $O(N)/O(1)$
  - [CSL '07]

static width $w = s^\uparrow$ [TODS '15] or faqw [PODS '16]
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

- **conjunctive**
  \( \mathcal{O}(N^w)/\mathcal{O}(1) \) [TODS ’15]

- **(\( \alpha \))-acyclic**
  \( \mathcal{O}(N)/\mathcal{O}(N) \) [CSL ’07]

- **hierarchical**
  \[ \text{This work} \] [PODS ’20]

- **free-connex**
  \( \mathcal{O}(N)/\mathcal{O}(1) \) [CSL ’07]

\[ \text{static width } w = s^\uparrow \text{ [TODS’15] or } faqw \text{ [PODS’16]} \]
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

- **conjunctive**
  \( \mathcal{O}(N^w)/\mathcal{O}(1) \)  
  [TODS ’15]

- **(\(\alpha\))-acyclic**
  \( \mathcal{O}(N)/\mathcal{O}(N) \)  
  [CSL ’07]

- **hierarchical**
  \( \mathcal{O}(N^{1+(w-1)\varepsilon})/\mathcal{O}(N^{1-\varepsilon}) \)  
  \( \varepsilon \in [0, 1] \)

- **free-connex**
  \( \mathcal{O}(N)/\mathcal{O}(1) \)  
  [CSL ’07]

**static width** \( w = s^{\uparrow} [\text{TODS’15}] \) or \( \text{faqw} [\text{PODS’16}] \)
Landscape of Static Query Evaluation

Preprocessing time/Enumeration delay

- **conjunctive**
  - $\mathcal{O}(N^w)/\mathcal{O}(1)$
  - [TODS '15]

- **(α)-acyclic**
  - $\mathcal{O}(N)/\mathcal{O}(N)$
  - [CSL '07]

- **hierarchical**
  - $\mathcal{O}(N^{1+(w-1)\epsilon})/\mathcal{O}(N^{1-\epsilon})$
  - $\epsilon \in [0, 1]$

- **free-connex**
  - $\mathcal{O}(N)/\mathcal{O}(1)$
  - [CSL '07]

**static width $w = s^\uparrow$ [TODS '15] or $faqw$ [PODS '16]**

**Figures:**
- **Left:** Logarithmic plot of preprocessing time vs. delay, showing regions for various query types.
- **Right:** Logarithmic plot of preprocessing time vs. enumeration delay, with specific cases for different query types.
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**

\( \mathcal{O}(N^w)/\mathcal{O}(N^\delta)/\mathcal{O}(1) \) [SIGMOD ’18]

\[ \delta = \text{max} \text{ static width} \] [PODS ’20]

static width \( w = s^{\uparrow} \) [TODS ’15] or faqw [PODS ’16]

dynamic width \( \delta = \text{max} \text{ static width} \) [PODS ’20]
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**
\[ O(N^w) / O(N^\delta) / O(1) \] [SIGMOD ’18]
triangle query \[ O(N^{1.5}) / O(N^{0.5})^* / O(1) \] [TODS ’20]

static width \( w = s^\uparrow \) [TODS ’15] or faqw [PODS ’16]
dynamic width \( \delta = \max_{\text{delta queries}} \) static width [PODS ’20]

\( ^* \): amortized update time
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**
- $O(N^w)/O(N^\delta)/O(1)$ [SIGMOD '18]
- Triangle query $O(N^{1.5})/O(N^{0.5^*})/O(1)$ [TODS '20]

**acyclic**
- Joins $O(N)/O(N)/O(1)$ [SIGMOD '17]

$w$ = static width

$\delta$ = dynamic width

$\epsilon \in [0, 1]$ for $\delta_1$-hierarchical

$(\alpha)$-acyclic

(*): amortized update time

$w$ = static width $s^\uparrow$ [TODS '15] or faqw [PODS '16]

$\delta$ = max static width [PODS '20]
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**
\[ O(N^w) / O(N^\delta) / O(1) \] [SIGMOD '18]

triangle query \[ O(N^{1.5}) / O(N^{0.5})^{*} / O(1) \] [TODS '20]

**\((\alpha-)acyclic\)**

**joins** \[ O(N) / O(N) / O(1) \] [SIGMOD '17]

**free-connex**
\[ O(N) / O(N) / O(1) \]
[SIGMOD '17]

static width \( w = s^\uparrow \) [TODS '15] or faqw [PODS '16]
dynamic width \( \delta = \max_{\text{delta queries}} \text{static width} \) [PODS '20]

(*) : amortized update time
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**
\[ O(N^w)/O(N^\delta)/O(1) \text{ [SIGMOD ’18]} \]
triangle query \[ O(N^{1.5})/O(N^{0.5})^*/O(1) \text{ [TODS ’20]} \]

\((\alpha\text{-})\text{acyclic}\)

\[ \text{joins } O(N)/O(N)/O(1) \text{ [SIGMOD ’17]} \]

**hierarchical**?
\[ \text{ [PODS ’20]} \]
This work

**free-connex**
\[ O(N)/O(N)/O(1) \text{ [SIGMOD ’17]} \]

static width \( w = s^{\uparrow} \text{ [TODS ’15]} \) or faqw \text{ [PODS ’16]} \]
dynamic width \( \delta = \max_{\text{delta queries}} \text{ static width} \text{ [PODS ’20]} \)

\((*)\): amortized update time
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**

\[ \mathcal{O}(N^w)/\mathcal{O}(N^\delta)/\mathcal{O}(1) \] \[ \text{[SIGMOD '18]} \]

triangle query \( \mathcal{O}(N^{1.5})/\mathcal{O}(N^{0.5})^*/\mathcal{O}(1) \) \[ \text{[TODS '20]} \]

**\( (\alpha-)\)acyclic**

**joins** \( \mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1) \) \[ \text{[SIGMOD '17]} \]

**hierarchical**

\[ \mathcal{O}(N^{1+(w-1)\varepsilon})/\mathcal{O}(N^{\delta\varepsilon})^*/\mathcal{O}(N^{1-\varepsilon}) \] \[ \varepsilon \in [0, 1] \]

**free-connex**

\( \mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1) \) \[ \text{[SIGMOD '17]} \]

static width \( w = s^\uparrow \) \[ \text{TODS '15} \] or faqw \[ \text{PODS '16} \]

dynamic width \( \delta = \max_{\text{delta queries}} \) static width \[ \text{PODS '20} \]

\( (*) \): amortized update time
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**

\( \mathcal{O}(N^w)/\mathcal{O}(N^\delta)/\mathcal{O}(1) \) \[SIGMOD '18\]

triangle query \( \mathcal{O}(N^{1.5})/\mathcal{O}(N^{0.5})^{*}/\mathcal{O}(1) \) \[TODS '20\]

\( (\alpha-)\text{acyclic} \)

joins \( \mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1) \) \[SIGMOD '17\]

**hierarchical**

\( \mathcal{O}(N^{1+(w-1)\varepsilon})/\mathcal{O}(N^{\delta\varepsilon})^{*}/\mathcal{O}(N^{1-\varepsilon}) \) \( \varepsilon \in [0, 1] \)

\( \delta_0\text{-hierarchical} \)

\( w = 1, \delta = 0 \) \[PODS '17\]

\( \delta_1\text{-hierarchical} \)

\( w \leq 2, \delta = 1 \) \[PODS '20\]

**free-connex**

\( \mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1) \) \[SIGMOD '17\]

\( \delta \geq 1 \)

\( (*) \): amortized update time

static width \( w = s^\uparrow \) \[TODS '15\] or \( faqw \) \[PODS '16\]

dynamic width \( \delta = \max \text{ static width} \) \[PODS '20\]
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

**conjunctive**
\[ O(N^w)/O(N^\delta)/O(1) \] [SIGMOD ’18]

triangle query \( O(N^{1.5})/O(N^{0.5})^*/O(1) \) [TODS ’20]

**\((\alpha-)\)acyclic**

joins \( O(N)/O(N)/O(1) \) [SIGMOD ’17]

**hierarchical**
\[ O(N^{1+(w-1)\varepsilon})/O(N^{\delta\varepsilon})^*/O(N^{1-\varepsilon}) \]
\( \varepsilon \in [0, 1] \)

**\(\delta_1\)-hierarchical**
\( w \leq 2, \delta = 1 \)

**\(\delta_0\)-hierarchical**
\( w = 1, \delta = 0 \)
[PODS ’17]

**free-connex**
\( O(N)/O(N)/O(1) \)
[SIGMOD ’17]

**\(\delta_1\)-hierarchical**
\( w \leq 2, \delta = 1 \)

static width \( w = s^{\uparrow} \) [TODS ’15] or faqw [PODS ’16]
dynamic width \( \delta = \max_{\text{delta queries}} \) static width [PODS ’20]

\((*)\): amortized update time
1. Recovery of Prior Approaches

- \( \log_N \text{update time} \)
- \( \log_N \text{preprocessing time} \)
- \( \log_N \text{delay} \)

 Conjunctive \( \delta \)

\[ \delta = 0 \] for hierarchical

\[ w = \frac{1}{\delta} \] for \( \delta \)-hierarchical

- Preprocessing time: \( O\left( N^{1 + (w - 1) \epsilon} \right) \)
- Amortized update time: \( O\left( N^{\delta \epsilon} \right) \)
- Enumeration delay: \( O\left( N^{1 - \epsilon} \right) \)
1. Recovery of Prior Approaches

logₙ update time

logₙ preprocessing time

δ

conjunctive

δ₀-hierarchical (w = 1, δ = 0)

(1, 0, 1)

logₙ delay

logₙ preprocessing time
1. Recovery of Prior Approaches

\[ \log_N \text{update time} \]

\[ \log_N \text{preprocessing time} \]

\[ \delta \]

\[ (1, 0, 1) \]

\[ \delta_0 \text{-hierarchical} \ (w = 1, \delta = 0) \]

\[ \text{conjunctive} \]
1. Recovery of Prior Approaches

- **log**\(_N\) preprocessing time
- **log**\(_N\) update time
- **log**\(_N\) delay

δ\(_0\)-hierarchical \((w = 1, \delta = 0)\)

- conjunctive
- hierarchical

Preprocessing time: \(O(N^{1+(w-1)\epsilon})\)

Amortized update time: \(O(N^{\delta\epsilon})\)

Enumeration delay: \(O(N^{1-\epsilon})\)
1. Recovery of Prior Approaches

- \( \log N \) preprocessing time
- \( \log N \) delay
- \( \log N \) update time

\( \delta \)-hierarchical
\( \delta_0 \)-hierarchical \( (w = 1, \delta = 0) \)
conjunctive

- Preprocessing time: \( O(N^{1+(w-1)\epsilon}) \)
- Amortized update time: \( O(N^\delta \epsilon) \)
- Enumeration delay: \( O(N^{1-\epsilon}) \)
First approach with sublinear amortized update time and enumeration delay for hierarchical queries.
3. Optimality for $\delta_1$-Hierarchical Queries

For any $\delta_1$-hierarchical query, there is no algorithm that admits
preprocessing time amortized update time enumeration delay
arbitrary $O(N^{0.5-\gamma})$ $O(N^{0.5-\gamma})$
for any $\gamma > 0$, unless the OMv Conjecture (*) fails.

(*) Online Matrix-Vector Multiplication cannot be solved in sub-cubic time.
3. Optimality for $\delta_1$-Hierarchical Queries

- For any $\delta_1$-hierarchical query, there is no algorithm that admits preprocessing time amortized update time enumeration delay
  arbitrary $\mathcal{O}(N^{0.5-\gamma})$ $\mathcal{O}(N^{0.5-\gamma})$
  for any $\gamma > 0$, unless the OMv Conjecture (*) fails.

- Our approach maintains any $\delta_1$-hierarchical query with preprocessing time amortized update time enumeration delay
  $\mathcal{O}(N^{1+\varepsilon})$ $\mathcal{O}(N^\varepsilon)$ $\mathcal{O}(N^{1-\varepsilon})$.

(*) Online Matrix-Vector Multiplication cannot be solved in sub-cubic time.
3. Optimality for \( \delta_1 \)-Hierarchical Queries

- For any \( \delta_1 \)-hierarchical query, there is no algorithm that admits
  preprocessing time \( O(N^{0.5-\gamma}) \)
  amortized update time \( O(N^{0.5-\gamma}) \)
  enumeration delay \( O(N^{0.5-\gamma}) \)
for any \( \gamma > 0 \), unless the OMv Conjecture (*) fails.

- Our approach maintains any \( \delta_1 \)-hierarchical query with
  preprocessing time \( O(N^{1+\varepsilon}) \)
  amortized update time \( O(N^\varepsilon) \)
  enumeration delay \( O(N^{1-\varepsilon}) \).

\[ \Rightarrow \] For \( \varepsilon = 0.5 \), this is weakly Pareto optimal, unless OMv Conjecture fails.

(*): Online Matrix-Vector Multiplication cannot be solved in sub-cubic time.
4. Single-Tuple vs Bulk Updates

\( \delta = w - 1 \) or \( \delta = w \) for hierarchical queries.

**Case \( \delta = w - 1 \)**

Time to insert \( N \) tuples: \( \mathcal{O}(N \cdot N^{(w-1)\varepsilon}) = \mathcal{O}(N^{1+(w-1)\varepsilon}) \).

\[ \Rightarrow \text{Preprocessing can be simulated by executing } N \text{ single-tuple updates.} \]
4. Single-Tuple vs Bulk Updates

\[ \delta = w - 1 \] or \[ \delta = w \] for hierarchical queries.

**Case \( \delta = w - 1 \)**

Time to insert \( N \) tuples: \( \mathcal{O}(N \cdot N^{(w-1)\varepsilon}) = \mathcal{O}(N^{1+(w-1)\varepsilon}) \).

\[ \Rightarrow \] Preprocessing can be simulated by executing \( N \) single-tuple updates.

**Case \( \delta = w \)**

Time to insert \( N \) tuples: \( \mathcal{O}(N \cdot N^{w\varepsilon}) = \mathcal{O}(N^{1+(w-1)\varepsilon+\varepsilon}) \).

\[ \Rightarrow \] Complexity gap of \( \mathcal{O}(N^\varepsilon) \) between single-tuple updates and bulk updates.
A query is **hierarchical** if for any two variables $X$, $Y$: 

$$\text{atoms}(X) \subseteq \text{atoms}(Y) \text{ or } \text{atoms}(X) \supseteq \text{atoms}(Y) \text{ or } \text{atoms}(X) \cap \text{atoms}(Y) = \emptyset$$

Hierarchical \( Q(\mathcal{F}) = R(A, B, D), S(A, B), \) 
\( T(A, C, F), U(A, C, G) \)

\( \mathcal{F} \) any set of variables
Hierarchical Queries

A query is hierarchical if for any two variables $X$, $Y$:
$\text{atoms}(X) \subseteq \text{atoms}(Y)$ or $\text{atoms}(X) \supseteq \text{atoms}(Y)$ or $\text{atoms}(X) \cap \text{atoms}(Y) = \emptyset$

hierarchical
$Q(\mathcal{F}) = R(A, B, D), S(A, B), T(A, C, F), U(A, C, G)$
$\mathcal{F}$ any set of variables

not hierarchical
$Q(\mathcal{F}) = R(A), S(A, B), T(B)$
$\mathcal{F}$ any set of variables
A hierarchical query is $\delta_0$-hierarchical if all free variables dominate the bound ones.

\[ Q(A, B, C) = R(A, B, D), S(A, B), T(A, C, F), U(A, C, G) \]
A hierarchical query is \( \delta_0\)-hierarchical if all free variables dominate the bound ones.

\[
\delta_0\text{-hierarchical}
\]

\[
Q(A, B, C) = R(A, B, D), S(A, B), T(A, C, F), U(A, C, G)
\]

\[
\text{hierarchical but not } \delta_0\text{-hierarchical}
\]

\[
Q(A) = S(A, B), T(B)
\]
For any bound variable $X$ and any atom using $X$, we need at most one further atom to cover all free variables dominated by $X$.

The query is not $\delta_0$-hierarchical.
For any bound variable $X$ and any atom using $X$, we need at most one further atom to cover all free variables dominated by $X$.

The query is not $\delta_0$-hierarchical.
Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$
Trade-Off in Static Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$

Lower bound [CSL '07]

There is no algorithm that admits

- preprocessing time $\mathcal{O}(N)$
- enumeration delay $\mathcal{O}(1)$

unless Boolean Matrix Multiplication can be solved in quadratic time.
Trade-Off in Static Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$

- Known approach: Eager preprocessing, quick enumeration
  - Preprocessing: Materialize the result.
  - Enumeration: Enumerate from materialized result.

Open question

- Is there an algorithm that admits sub-quadratic preprocessing time and sub-linear enumeration delay?
Trade-Off in Static Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$

Known approach: Lazy preprocessing, heavy enumeration

- Preprocessing: Eliminate dangling tuples.
- Enumeration: For each $B$-value, enumerate distinct $C$-values.
Trade-Off in Static Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$

Open question

Is there an algorithm that admits sub-quadratic preprocessing time and sub-linear enumeration delay?
Simple $\delta_1$-hierarchical query

$$Q(B, C) = R(A, B), S(A, C)$$

**Known Approach: Eager Preprocessing, Quick Enumeration**
- **Preprocessing:** Materialize the result.
- **Enumeration:** Enumerate from materialized result.

**Known Approach: Lazy Preprocessing, Heavy Enumeration**
- **Preprocessing:** Eliminate dangling tuples.
- **Enumeration:** For each $B$-value, enumerate distinct $C$-values.

**Lower Bound** [CSL '07]
There is no algorithm that admits preprocessing time, enumeration delay $O(N)$ $O(1)$ unless Boolean Matrix Multiplication can be solved in quadratic time.

**Open Question**
Is there an algorithm that admits sub-quadratic preprocessing time and sub-linear enumeration delay?

**Graphical Representation**
- **Preprocessing Time:** $O(N^{1+\epsilon})$
- **Enumeration Delay:** $O(N^{1-\epsilon})$
Simple $\delta_1$-hierarchical query

$$Q(A) = R(A, B), S(B)$$
Simple $\delta_1$-hierarchical query

\[ Q(A) = R(A, B), S(B) \]

For this query, there is no algorithm that admits

- Preprocessing time: arbitrary
- Amortized update time: $O(N^{0.5-\gamma})$
- Enumeration delay: $O(N^{0.5-\gamma})$

for any $\gamma > 0$, unless the OMv Conjecture fails.
Trade-Off in Dynamic Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(A) = R(A, B), S(B)$$

Known approach: Eager update, quick enumeration

- **Preprocessing:** Materialize the result.
- **Upon update:** Maintain the materialized result.
- **Enumeration:** Enumerate from materialized result.
Simple $\delta_1$-hierarchical query

$$Q(A) = R(A, B), S(B)$$

Known approach: Lazy update, heavy enumeration

- **Preprocessing:** Eliminate dangling tuples.
- **Upon update:** Update only base relations.
- **Enumeration:** Eliminate dangling tuples and enumerate from $R$. 
Trade-Off in Dynamic Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(A) = R(A, B), S(B)$$

Known approach: Eager update, quick enumeration
Preprocessing: Materialize the result.
Upon update: Maintain the materialized result.
Enumeration: Enumerate from materialized result.

Known approach: Lazy update, heavy enumeration
Preprocessing: Eliminate dangling tuples.
Upon update: Update only base relations.
Enumeration: Eliminate dangling tuples and enumerate from $R$.

Lower bound
For this query, there is no algorithm that admits
preprocessing time \( O(N^{0.5}) \)
amortized update time \( O(N^{0.5}) \)
enumeration delay \( O(N^{1-\epsilon}) \)
for any \( \epsilon > 0 \), unless the OMv Conjecture fails.

Open question
Is there an algorithm that admits
sub-linear (amortized) update time and sub-linear enumeration delay?
Trade-Off in Dynamic Query Evaluation: Example

Simple $\delta_1$-hierarchical query

$$Q(A) = R(A, B), S(B)$$

- **Logarithmic update time**
- **Logarithmic preprocessing time**
- **Amortized update time** $O(N^{1-\varepsilon})$
- **Enumeration delay** $O(N^{1-\varepsilon})$

**Known approach:**
1. **Eager update, quick enumeration**
   - **Preprocessing:** Materialize the result.
   - **Upon update:** Maintain the materialized result.
   - **Enumeration:** Enumerate from the materialized result.

2. **Lazy update, heavy enumeration**
   - **Preprocessing:** Eliminate dangling tuples.
   - **Upon update:** Update only base relations.
   - **Enumeration:** Eliminate dangling tuples and enumerate from $R$.

**Open question:** Is there an algorithm that admits sub-linear (amortized) update time and sub-linear enumeration delay?

- Weak Pareto optimality by OMv Conjecture
Conclusion

Benefits of Our Approach

- Allows to tune the trade-off between preprocessing time, update time, and enumeration delay.
- Recovers existing results as specific points.
- Maintains hierarchical queries with sub-linear amortized update time and enumeration delay.
- Maintains $\delta_1$-queries with weakly Pareto optimal update time and delay.

Ongoing Work

- Extension of our approach to
  - conjunctive queries,
  - aggregate queries, and
  - enumeration in desired order.
- System prototype.