Trade-offs in Static and Dynamic Query Evaluation

Ahmet Kara, Milos Nikolic
Dan Olteanu, and Haozhe Zhang

fdbresearch.github.io

KOCOON Workshop 2019, Arras
Static and Dynamic Query Evaluation

Static Query Evaluation

query \rightarrow \text{data base} \rightarrow \text{preprocessing} \rightarrow \text{data structure} \rightarrow \text{enumeration} \rightarrow \text{query result}

We are interested in the trade-off between:

- preprocessing time
- enumeration delay
- (update time)

single-tuple update maintenance 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 → data base → preprocessing → data structure → enumeration → query result

preprocessing time

enumeration delay

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 - (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
- 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]

$hierarchical$

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**
  - $O(N^w)/O(1)$  [TODS '15]
- **$(\alpha)$-acyclic**
  - $O(N)/O(N)$  [CSL '07]
- **acyclic**

static width $w = s^\uparrow$  [TODS '15] or $fwq$  [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**
  \[ O(N^w)/O(1) \]  
  [TODS ’15]

- **(α)-acyclic**
  \[ O(N)/O(N) \]  
  [CSL ’07]

- **hierarchical**
  \[ O(N)/O(N) \]  
  [PODS ’20]
  This work

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

\[ \log N \] time

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]

- **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]

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

- **Free-connex**
  \[ O(N)/O(1) \]
  [CSL ’07]

Preprocessing time $O(N^{1+(w-1)\epsilon})$  
Enumeration delay $O(N^{1-\epsilon})$

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**
  \( \mathcal{O}(N^{1+(w-1)\varepsilon})/\mathcal{O}(N^{1-\varepsilon}) \)  
  - \(\varepsilon \in [0, 1]\)
- **free-connex**
  \( \mathcal{O}(N)/\mathcal{O}(1) \)  
  - [CSL '07]

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

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

\[ \varepsilon \in [0, 1] \]

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

---

preprocessing time \( \mathcal{O}(N^{1+(w-1)\varepsilon}) \)

delay \( \mathcal{O}(N^{1-\varepsilon}) \)

---

enumeration delay \( \mathcal{O}(N^{1-\varepsilon}) \)
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

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

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

(\(\alpha\)-)acyclic

hierarchical
[PODS '20]
This work

free-connex

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

dynamic width \( \delta = \max_{\text{delta queries}} \text{static width} \) [PODS '20]
Preprocessing time/Update time/Enumeration delay

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

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

\( (*) \): amortized update time

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

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

\( \delta \): hierarchical
\[ w \leq 2, \delta = 1 \]
Landscape of Dynamic Query Evaluation

Preprocessing time/Update time/Enumeration delay

conjunctive
$O(N^w)/O(N^\delta)/O(1)$ [SIGMOD ‘18]

(α-)acyclic

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

$\delta_0$-hierarchical
$w = 1, \delta = 0$
[PODS ‘17]

free-connex

(*): amortized update time

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

dynamic width $\delta = \max_{\text{delta queries}}$ 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]

(\(\alpha\)-)acyclic

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

\((*)\): amortized update time

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

dynamic width \(\delta = \max_{\text{delta queries}}\) static width [PODS '20]
Contribution 1: Recovery of Prior Approaches

Recovers prior approach for conjunctive queries by setting $\varepsilon = 1$.

Recovers prior approach for $\delta_0$-hierarchical queries by setting $\varepsilon = 1$. 

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

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

\[ \delta \]

\[ \text{conjunctive} \]

\[ (1, 0, 1) \]
Contribution 1: Recovery of Prior Approaches

Recovers prior approach for conjunctive queries by setting $\epsilon = 1$.

Recovers prior approach for $\delta_0$-hierarchical queries by setting $\epsilon = 1$. 

### Diagram

- $\log_N$ preprocessing time
- $\log_N$ update time
- $\log_N$ delay
- $\delta$
- $w$

Points:
- $(1, 0, 1)$
- $\delta_0$-hierarchical ($w = 1, \delta = 0$)
Contribution 1: Recovery of Prior Approaches

- **log** $N$ delay
- **log** $N$ preprocessing time
- **log** $N$ update time
- **log** $N$ time

Recovers prior approach for conjunctive queries by setting $\delta = 0$.

Recovers prior approach for $\delta_0$-hierarchical queries by setting $\epsilon = 1$.
Contribution 1: Recovery of Prior Approaches

Recovers prior approach for conjunctive queries by setting $\varepsilon = 1$.

Recovers prior approach for $\delta_0$-hierarchical queries by setting $\varepsilon = 1$.

$$\log_N \text{update time}$$

$$\log_N \text{preprocessing time}$$

$$(1, 0, 1)$$

$$(w = 1, \delta = 0)$$

$\delta_0$-hierarchical

conjunctive

$$\delta$$

$w$

preprocessing time $O(N^{1+(w-1)\varepsilon})$

amortized update time $O(N^{\delta \varepsilon})$

enumeration delay $O(N^{1-\varepsilon})$
Contribution 1: Recovery of Prior Approaches

- Recovers prior approach for **conjunctive** queries by setting $\varepsilon = 1$.
- Recovers prior approach for **$\delta_0$-hierarchical** queries by setting $\varepsilon = 1$. 

![Diagram showing log_N preprocessing time, log_N update time, and log_N delay axes.](image)
First approach that allows sublinear amortized update time and sublinear enumeration delay for hierarchical queries.
Contribution 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 $O(N^{0.5-\gamma})$, and enumeration delay $O(N^{0.5-\gamma})$ for any $\gamma > 0$, unless the OMv Conjecture (*) fails.

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

- For any $\delta_1$-hierarchical query, there is no algorithm that admits
  preprocessing time  \hspace{1cm} amortized update time  \hspace{1cm} enumeration delay
  arbitrary  \hspace{1cm} $O(N^{0.5-\gamma})$  \hspace{1cm} $O(N^{0.5-\gamma})$
  for any $\gamma > 0$, unless the OMv Conjecture (*) fails.

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

\ (*) OMv Conjecture: Online Matrix-Vector Multiplication Problem cannot be solved in sub-cubic time.

\[
\begin{align*}
\log_N \text{preprocessing time} & \\
\log_N \text{update time} & \\
\log_N \text{delay} & \\
\delta = 1 & \\
(1, 0, 1) & \\
(1.5, 0.5, 0.5) & \\
\end{align*}
\]
Contribution 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 $O(N^{1+\epsilon})$  
amortized update time $O(N^{\epsilon})$
enumeration delay $O(N^{1-\epsilon})$.

  $\implies$ For $\epsilon = 0.5$, this is weak Pareto optimal, unless OMv Conjecture fails.

(*) OMv Conjecture: Online Matrix-Vector Multiplication Problem cannot be solved in sub-cubic time.
\[ \delta = w - 1 \] or \[ \delta = w \] for hierarchical queries.

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

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

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

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

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

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

\[ \implies \text{Preprocessing can be simulated by executing } N \text{ single-tuple updates.} \]

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

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

\[ \implies \text{Complexity gap of } \mathcal{O}(N^{\varepsilon}) \text{ between single-tuple updates and bulk updates.} \]
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 $\text{atoms}(X) \cap \text{atoms}(Y) = \emptyset$

**Example:**

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

$\mathcal{F} \subseteq \{A, B, C, D, F, G\}$

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

---

Not hierarchical

$\mathcal{F} \subseteq \{A, B, C, D, F, G\}$

$Q(\mathcal{F}) = R(A), S(A, B), T(B)$
A hierarchical query is $\delta_0$-hierarchical if for any bound variable $X$ and atom $R(X) \in \text{atoms}(X)$: $\text{free}(\text{atoms}(X)) \subseteq X$.

\[
\delta_0\text{-hierarchical Q}(A, B, C) = R(A, B, D), S(A, B), T(A, C, F), U(A, C, G)
\]

$\delta_0$-hierarchical
A hierarchical query is $\delta_0$-hierarchical if for any bound variable $X$ and atom $R(X) \in \text{atoms}(X)$: $\text{free}(\text{atoms}(X)) \subseteq X$.

$\delta_0$-hierarchical

$$Q(A, B, C) = R(A, B, D), S(A, B), T(A, C, F), U(A, C, G)$$

Hierarchical but not $\delta_0$-hierarchical

$$Q(A) = S(A, B), T(B)$$
\(\delta_1\)-Hierarchical Queries

- The query is not \(\delta_0\)-hierarchical.
- For any bound variable \(X\) and atom \(R(X) \in atoms(X)\): there is an atom \(S(Y) \in atoms(X)\) such that \(\text{free}(atoms(X)) \subseteq X \cup Y\).

\[
\delta_1\text{-hierarchical}
\]

\[
Q(A, D, E, G) = R(A, B, D), S(A, B, E), T(A, C, F), U(A, C, G)
\]
The query is not $\delta_0$-hierarchical.

For any bound variable $X$ and atom $R(X) \in \text{atoms}(X)$: there is an atom $S(Y) \in \text{atoms}(X)$ such that $\text{free}(\text{atoms}(X)) \subseteq X \cup Y$.

$\delta_1$-Hierarchical Queries

$\delta_1$-hierarchical

$Q(A, D, E, G) = R(A, B, D), S(A, B, E), T(A, C, F), U(A, C, G)$

not $\delta_1$-hierarchical

$Q(D, G) = R(A, B, D), S(A, B, E), T(A, C, F), U(A, C, G)$
Static Query Evaluation - Example

Simple $\delta_1$-hierarchical query

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

Diagram: R and S with B, A, C nodes.
**Static Query Evaluation - Example**

**Simple \(\delta_1\)-hierarchical query**

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

![Diagram](image)

**Lower bound [CSL’07]**

There is no algorithm that admits

- preprocessing time: \(O(N)\)
- enumeration delay: \(O(1)\)

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

Simple $\delta_1$-hierarchical query

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

**Diagram:**
- Graphical representation of the query $Q(B, C)$ with sets $R$ and $S$.

**Graphs:**
- Two graphs showing the relationship between log$_N$ preprocessing time and log$_N$ delay.
- The solid line represents our approach with preprocessing time $O(N^{1+\varepsilon})$ and enumeration delay $O(N^{1-\varepsilon})$.
- The dashed line represents a known approach with different time complexities.

**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**
- Lower bound [CSL '07]:
  - There is no algorithm that admits $O(N)$ preprocessing time and $O(1)$ enumeration delay 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?
Dynamic Query Evaluation - Example

Simple $\delta_1$-hierarchical query

\[ Q(A) = R(A, B), S(B) \]
Dynamic Query Evaluation - Example

Simple $\delta_1$-hierarchical query

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

For this query, there is no algorithm that admits

- preprocessing time \(O(N)\)
- amortized update time \(O(N^{0.5-\gamma})\)
- enumeration delay \(O(N^{0.5-\gamma})\)

for any \(\gamma > 0\), unless the OMv Conjecture fails.
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.

Lower bound

For this 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.

Open question

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

(∗): Weak Pareto optimality by OMv Conjecture
Dynamic Query Evaluation - Example

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.
**Dynamic Query Evaluation - Example**

Simple $\delta_1$-hierarchical query

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

![Diagram of the query and parameters](image)

**Open question**

Is there an algorithm that admits sub-linear (amortized) update time and sub-linear enumeration delay?
Dynamic Query Evaluation - Example

Simple $\delta_1$-hierarchical query

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

$(1, 0, 1)$

$(1.0, 0.5, 0.5)$ optimal

$(\ast)$: Weak Pareto optimality by OMv Conjecture

- Preprocessing time $O(N^1)$
- Amortized update time $O(N^\epsilon)$
- Enumeration delay $O(N^{1-\epsilon})$
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 sub-linear enumeration delay.
- Maintains $\delta_1$-queries with weak Pareto optimal update time and delay.

Ongoing Work

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