Trade-Offs in Incremental View Maintenance

Dan Olteanu (University of Zurich)

fdbresearch.github.io

Logic & Algorithms in DB Theory and AI
August 25, 2023

Acknowledgments

DaST IVM team

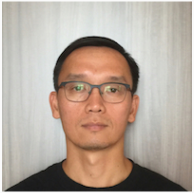
Ahmet

Haozhe

Johann

Milos

RelationalAI colleagues



Hung

ElSeidy

Henrik

Niko

Setting & Objective of this Lecture

Incremental View Maintenance (IVM)

- Well-established and longstanding research problem
- Confusing naming: incremental vs decremental

  Alternative common naming: *Fully dynamic*

Setting & Objective of this Lecture

Incremental View Maintenance (IVM)

- Well-established and longstanding research problem
- Confusing naming: incremental vs decremental

  Alternative common naming: *Fully dynamic*

Setting

- Fully dynamic algorithms (i.e., supports inserts and deletes)
- Single-tuple updates to relational databases
- Relational queries (non-recursive)

Setting & Objective of this Lecture

Incremental View Maintenance (IVM)

- Well-established and longstanding research problem
- Confusing naming: incremental vs decremental

  Alternative common naming: *Fully dynamic*

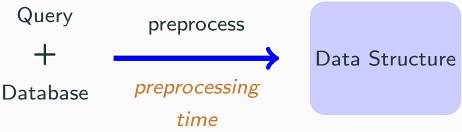
Setting

- Fully dynamic algorithms (i.e., supports inserts and deletes)
- Single-tuple updates to relational databases
- Relational queries (non-recursive)

Objective

- Overview of recent (and *very preliminary*) results on worst-case optimal IVM, trade-offs, and IVM for complex analytics

The Incremental View Maintenance Problem



Query

+

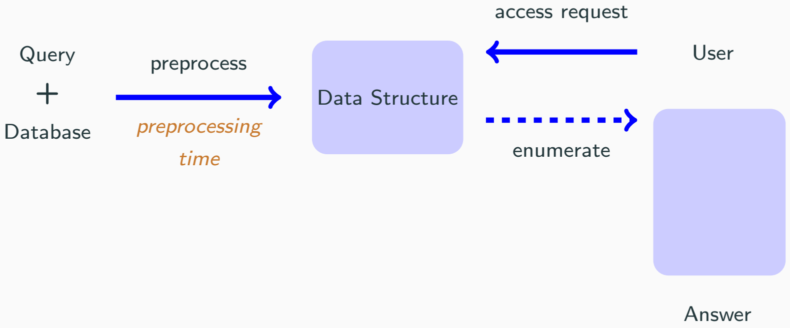
Database

preprocess

preprocessing time

Data Structure

The Incremental View Maintenance Problem



Query

+

Database

preprocess

preprocessing time

Data Structure

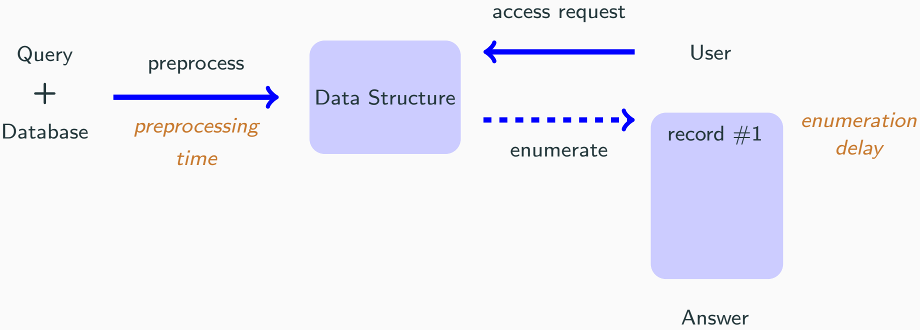
access request

User

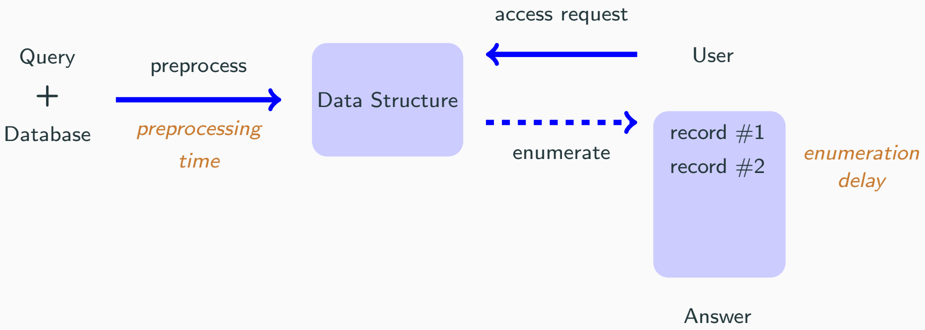
enumerate

Answer

The Incremental View Maintenance Problem



The Incremental View Maintenance Problem



Query

+

Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

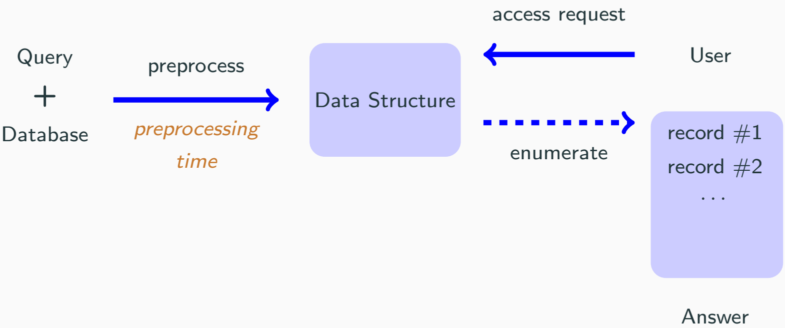
record #1

record #2

enumeration delay

Answer

The Incremental View Maintenance Problem



Query

+

Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

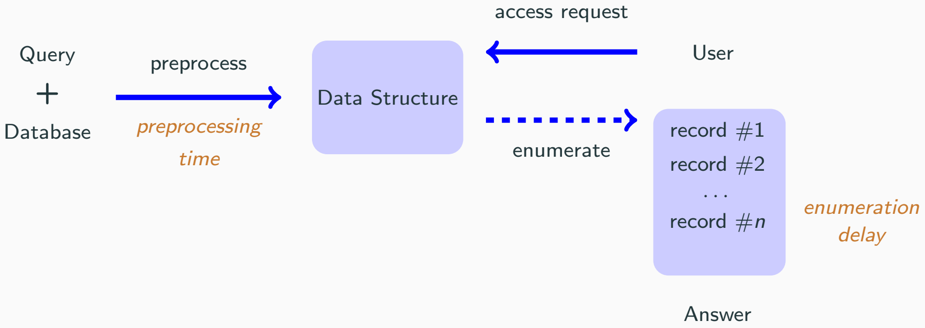
record #1

record #2

. . .

Answer

The Incremental View Maintenance Problem



Query + Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

record #1

record #2

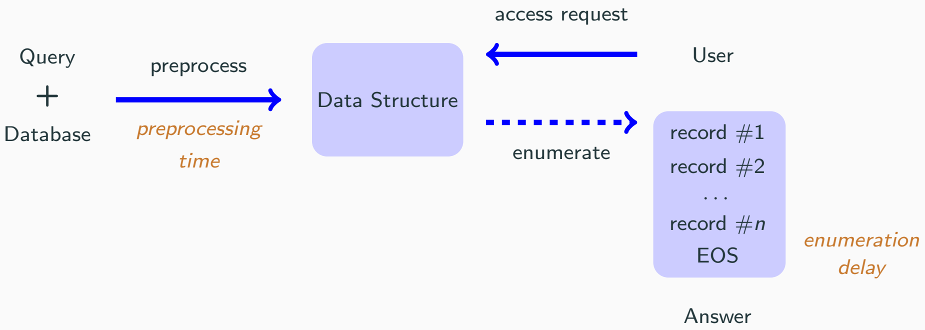
...

record #n

enumeration delay

Answer

The Incremental View Maintenance Problem



Query

+

Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

record #1

record #2

...

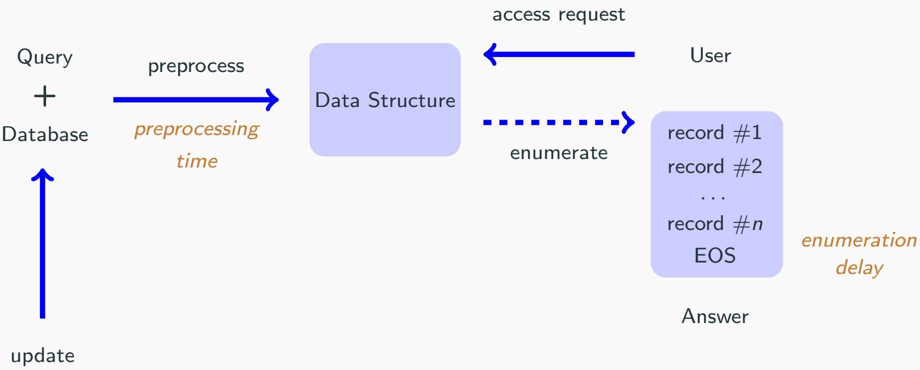
record $\#n$

EOS

enumeration delay

Answer

The Incremental View Maintenance Problem



Query + Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

record #1

record #2

...

record #n

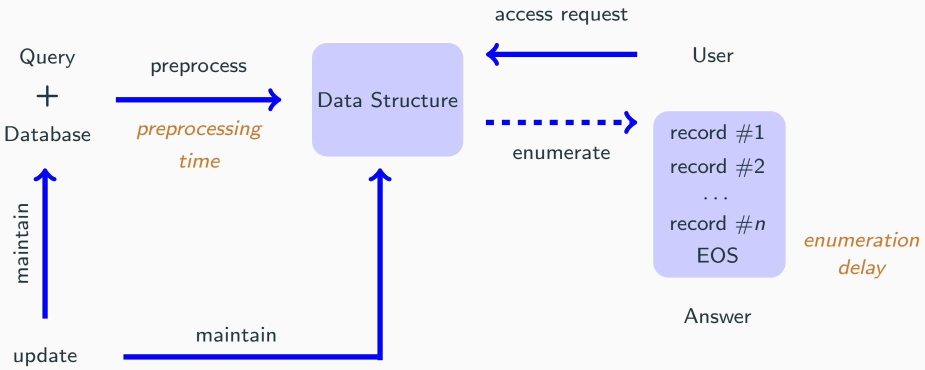
EOS

enumeration delay

Answer

update

The Incremental View Maintenance Problem



Query

+

Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

record #1

record #2

...

record #n

EOS

Answer

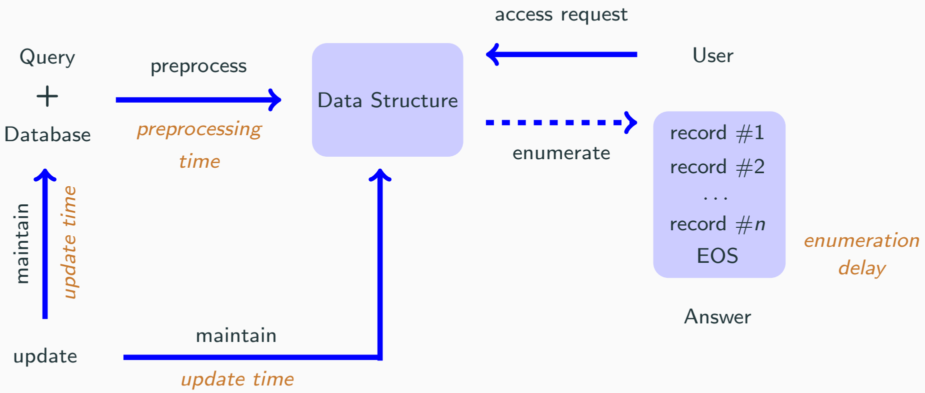
enumeration delay

maintain

update

maintain

The Incremental View Maintenance Problem



Query + Database

preprocess

preprocessing time

Data Structure

access request

User

enumerate

record #1

record #2

...

record $\#n$

EOS

Answer

enumeration delay

maintain

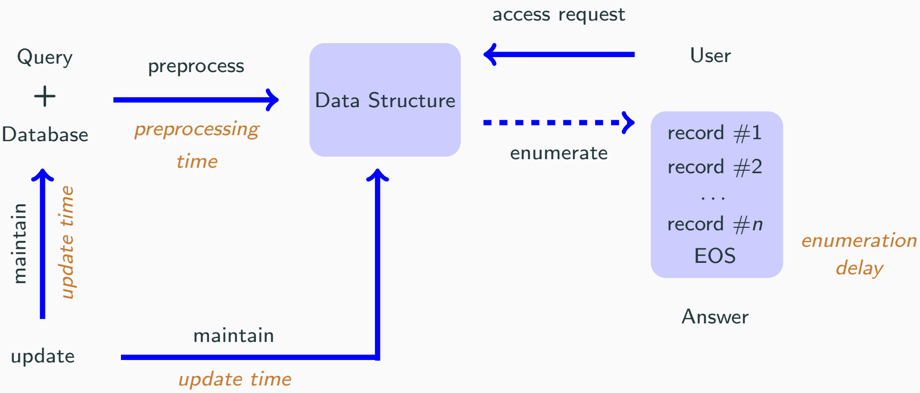
update time

update

maintain

update time

The Incremental View Maintenance Problem



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

Agenda

Part 1. Main IVM techniques by example

- The triangle count query

Part 2. Constant update time and enumeration delay

- The q-hierarchical queries

Part 3. Update time - enumeration delay trade-offs

- The hierarchical queries and beyond

Part 4. ML models under updates

- Covariance matrix and Chow-Liu trees

1. IVM Techniques By Example

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)

R

A	B	#
a_1	b_1	2
a_2	b_1	3

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)

R

A	B	#
a_1	b_1	2
a_2	b_1	3

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)

R

A	B	#
a_1	b_1	2
a_2	b_1	3

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$

R

A	B	$\#$
a_1	b_1	2
a_2	b_1	3

S

B	C	$\#$
b_1	c_1	2
b_1	c_2	1

T

C	A	$\#$
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	$\#$
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

↓

Q

$\emptyset$	$\#$
( )	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	$\#$
a_1	b_1	2
a_2	b_1	3

S

B	C	$\#$
b_1	c_1	2
b_1	c_2	1

T

C	A	$\#$
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	$\#$
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

$\uparrow$

$\downarrow$

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	$\#$
a_2	b_1	-2

Q

$\emptyset$	$\#$
$()$	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	$\#$
a_1	b_1	2
a_2	b_1	3

S

B	C	$\#$
b_1	c_1	2
b_1	c_2	1

T

C	A	$\#$
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	$\#$
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

↑

↓

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	$\#$
a_2	b_1	-2

Q

$\emptyset$	$\#$
$()$	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	#
a_1	b_1	2
~~a_2~~	~~b_1~~	~~3~~
a_2	b_1	1

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

↑

↓

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	#
a_2	b_1	-2

Q

$\emptyset$	#
( )	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	#
a_1	b_1	2
~~a_2~~	~~b_1~~	~~3~~
a_2	b_1	1

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
a_2	b_1	c_2	$3 \cdot 1 \cdot 3 = 9$

↑

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	#
a_2	b_1	-2

↓

Q

$\emptyset$	#
( )	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	#
a_1	b_1	2
~~a_2~~	~~b_1~~	~~3~~
a_2	b_1	1

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
~~a_2~~	~~b_1~~	~~c_2~~	~~$3 \cdot 1 \cdot 3 = 9$~~
a_2	b_1	c_2	$1 \cdot 1 \cdot 3 = 3$

↑

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	#
a_2	b_1	-2

↓

Q

$\emptyset$	#
( )	$4 + 6 + 9 = 19$

Background: Relations and Queries

- Relations are functions mapping tuples to elements from a ring (here, $\mathbb{Z}$)
- Triangle Count Query: $Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$
- A single-tuple update is a relation mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R

A	B	#
a_1	b_1	2
~~a_2~~	~~b_1~~	~~3~~
a_2	b_1	1

S

B	C	#
b_1	c_1	2
b_1	c_2	1

T

C	A	#
c_1	a_1	1
c_2	a_1	3
c_2	a_2	3

$R \cdot S \cdot T$

A	B	C	#
a_1	b_1	c_1	$2 \cdot 2 \cdot 1 = 4$
a_1	b_1	c_2	$2 \cdot 1 \cdot 3 = 6$
~~a_2~~	~~b_1~~	~~c_2~~	~~$3 \cdot 1 \cdot 3 = 9$~~
a_2	b_1	c_2	$1 \cdot 1 \cdot 3 = 3$

↑

$\delta R = \{(a_2, b_1) \mapsto -2\}$

A	B	#
a_2	b_1	-2

↓

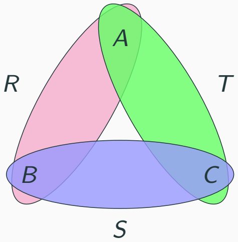
Q

$\emptyset$	#
~~( )~~	~~$4 + 6 + 9 = 19$~~
( )	$4 + 6 + 3 = 13$

The Triangle Count Query

The triangle count query Q returns the number of tuples in the join of R, S, and T:

$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$



Problem: Maintain Q under single-tuple updates to R, S, and T

Much Ado about Triangles

The Triangle Query Served as Milestone in Many Fields

- Worst-case optimal join algorithms *[Algorithmica 1997, SIGMOD R. 2013]*
- Parallel query evaluation *[Found. & Trends DB 2018]*
- Randomized approximation in static settings *[FOCS 2015]*
- Randomized approximation in data streams *[SODA 2002, COCOON 2005, PODS 2006, PODS 2016, Theor. Comput. Sci. 2017]*

Answering Queries under Updates

- Theoretical developments *[PODS 2017, ICDT 2018]*
- Systems developments *[F. & T. DB 2012, VLDB J. 2014, SIGMOD 2017, 2018]*
- Lower bounds *[STOC 2015, ICM 2018]*

Much Ado about Triangles

The Triangle Query Served as Milestone in Many Fields

- Worst-case optimal join algorithms *[Algorithmica 1997, SIGMOD R. 2013]*
- Parallel query evaluation *[Found. & Trends DB 2018]*
- Randomized approximation in static settings *[FOCS 2015]*
- Randomized approximation in data streams *[SODA 2002, COCOON 2005, PODS 2006, PODS 2016, Theor. Comput. Sci. 2017]*

Answering Queries under Updates

- Theoretical developments *[PODS 2017, ICDT 2018]*
- Systems developments *[F. & T. DB 2012, VLDB J. 2014, SIGMOD 2017, 2018]*
- Lower bounds *[STOC 2015, ICM 2018]*

Is there a **fully dynamic algorithm** that can maintain the **exact triangle count** in **worst-case optimal** time?

Naïve Maintenance

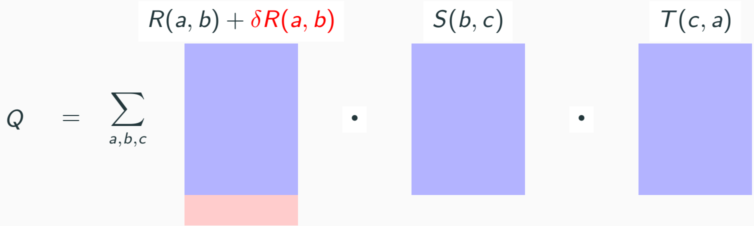
"Recompute from scratch"

$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

Naïve Maintenance

"Recompute from scratch"

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$

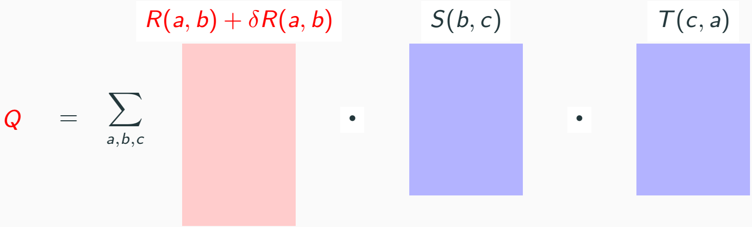


$$Q \quad = \quad \sum_{a,b,c} R(a,b) + \delta R(a,b) \quad \cdot \quad S(b,c) \quad \cdot \quad T(c,a)$$

Naïve Maintenance

"Recompute from scratch"

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$

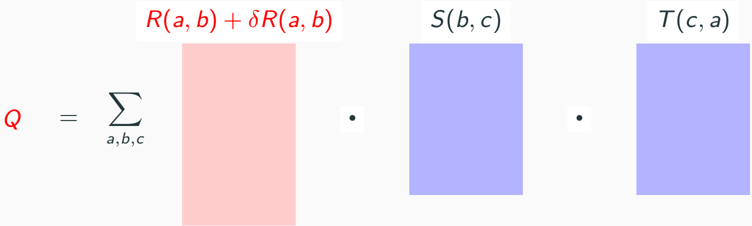


$$Q = \sum_{a,b,c} R(a,b) + \delta R(a,b) \cdot S(b,c) \cdot T(c,a)$$

Naïve Maintenance

"Recompute from scratch"

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



$$Q = \sum_{a,b,c} \big(R(a,b) + \delta R(a,b)\big) \cdot S(b,c) \cdot T(c,a)$$

- N is the database size
- Update time: $\mathcal{O}(N^{1.5})$ using worst-case optimal join algorithms

  [Algorithmica 1997, SIGMOD R. 2013, ICDT 2014]

  Slightly better using Strassen-like matrix multiplication
- Space: $\mathcal{O}(N)$ to store input relations

First-Order Incremental View Maintenance

"Compute the delta" *[Found. & Trends DB 2018]*

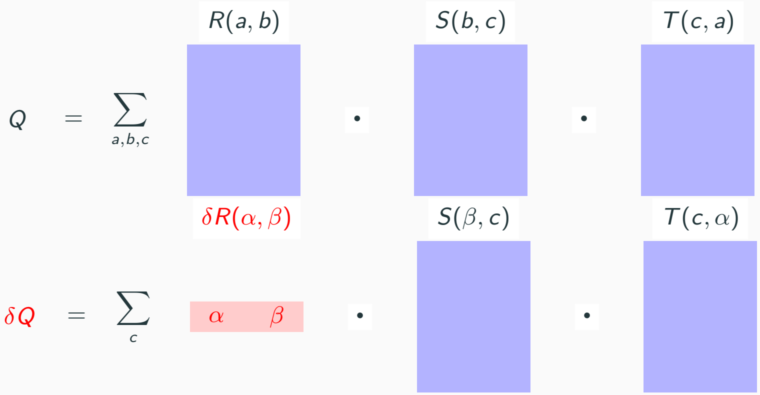
$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

First-Order Incremental View Maintenance

"Compute the delta"

[Found. & Trends DB 2018]

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



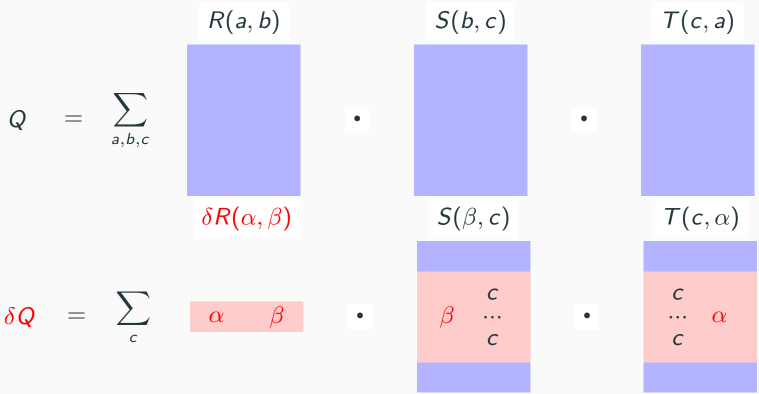
$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

$$\delta Q = \sum_{c} \delta R(\alpha,\beta) \cdot S(\beta,c) \cdot T(c,\alpha)$$

First-Order Incremental View Maintenance

"Compute the delta" *[Found. & Trends DB 2018]*

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

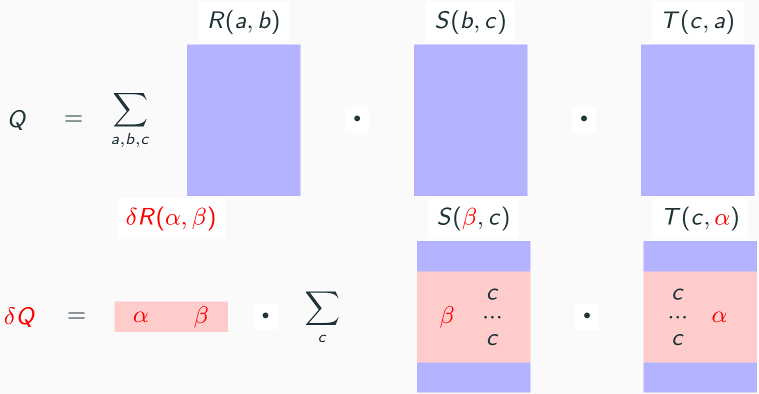
$$\delta Q = \sum_{c} \delta R(\alpha,\beta) \cdot S(\beta,c) \cdot T(c,\alpha)$$

First-Order Incremental View Maintenance

"Compute the delta"

[Found. & Trends DB 2018]

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



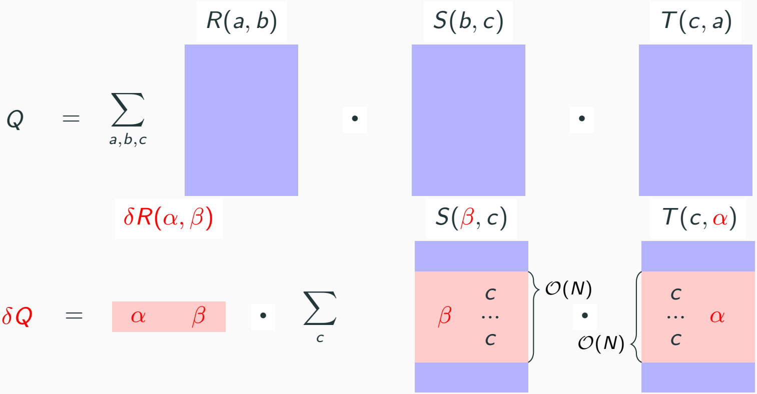
$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

$$\delta Q = \delta R(\alpha,\beta) \cdot \sum_{c} S(\beta,c) \cdot T(c,\alpha)$$

First-Order Incremental View Maintenance

"Compute the delta" *[Found. & Trends DB 2018]*

$\delta R = \{(\alpha, \beta) \mapsto m\}$



$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$

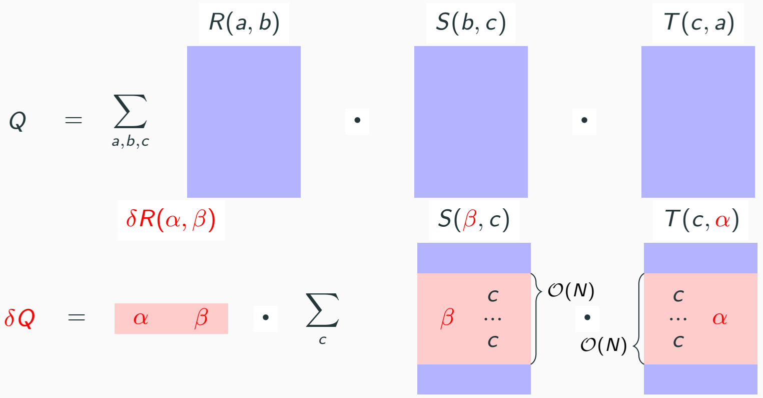
$\delta Q = \delta R(\alpha, \beta) \cdot \sum_{c} S(\beta, c) \cdot T(c, \alpha)$

First-Order Incremental View Maintenance

"Compute the delta"

[Found. & Trends DB 2018]

$\delta R = \{(\alpha, \beta) \mapsto m\}$

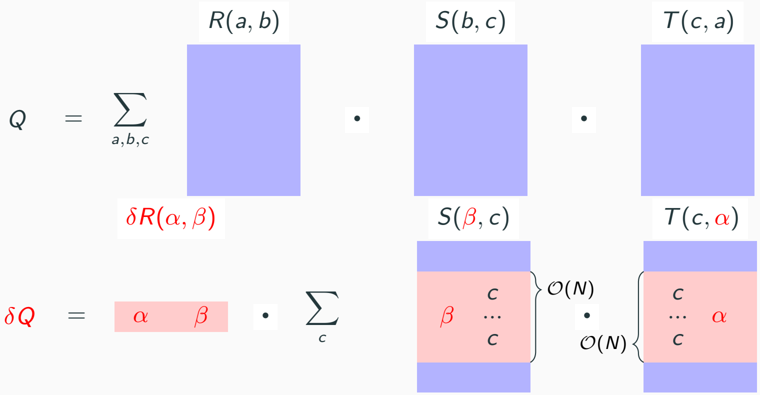


$$Q = Q + \delta Q$$

First-Order Incremental View Maintenance

"Compute the delta" *[Found. & Trends DB 2018]*

$\delta R = \{(\alpha, \beta) \mapsto m\}$



$R(a, b)$ $S(b, c)$ $T(c, a)$

$Q = \sum_{a,b,c}$ $\cdot$ $\cdot$

$\delta R(\alpha, \beta)$ $S(\beta, c)$ $T(c, \alpha)$

$\delta Q = \alpha \quad \beta \cdot \sum_{c}$ β c … c $\mathcal{O}(N)$ $\cdot$ $\mathcal{O}(N)$ c … c α

$Q = Q + \delta Q$

- Update time: $\mathcal{O}(N)$ to intersect C-values from S and T
- Space: $\mathcal{O}(N)$ to store input relations

Higher-Order Incremental View Maintenance

“Compute the delta using materialized views” *[VLDB J 2014]*

$$Q = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$

$$Q = \sum_{a,b,c} R(a,b) \cdot \underbrace{S(b,c) \cdot T(c,a)}_{V_{ST}(b,a) = \sum_{c} S(b,c) \cdot T(c,a)}$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

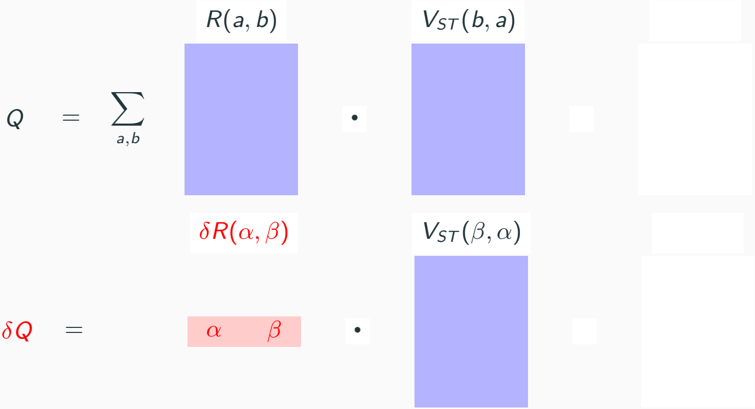
$$\delta R = \{(\alpha, \beta) \mapsto m\}$$

$$Q = \sum_{a,b} R(a, b) \cdot V_{ST}(b, a)$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

$\delta R = \{(\alpha, \beta) \mapsto m\}$



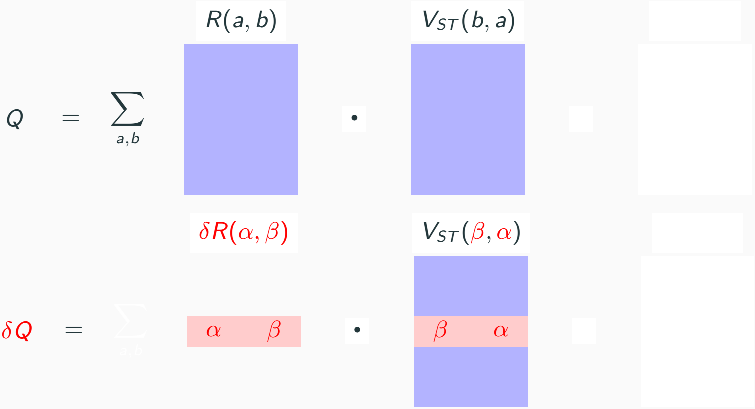
$$Q = \sum_{a,b} R(a, b) \cdot V_{ST}(b, a)$$

$$\delta Q = \delta R(\alpha, \beta) \cdot V_{ST}(\beta, \alpha)$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



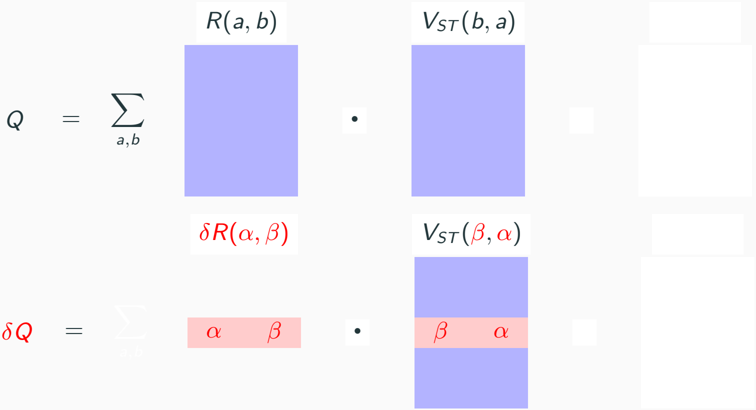
$$Q = \sum_{a,b} R(a, b) \cdot V_{ST}(b, a)$$

$$\delta Q = \sum_{a,b} \delta R(\alpha, \beta) \cdot V_{ST}(\beta, \alpha)$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



$$Q = \sum_{a,b} R(a,b) \cdot V_{ST}(b,a)$$

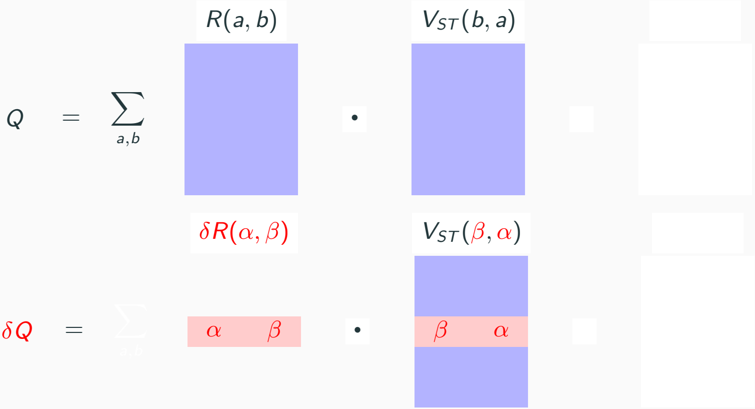
$$\delta Q = \sum_{a,b} \delta R(\alpha,\beta) \cdot V_{ST}(\beta,\alpha)$$

$$Q = Q + \delta Q$$

Higher-Order Incremental View Maintenance

"Compute the delta using materialized views" *[VLDB J 2014]*

$$\delta R = \{(\alpha, \beta) \mapsto m\}$$



$$Q = \sum_{a,b} R(a,b) \cdot V_{ST}(b,a)$$

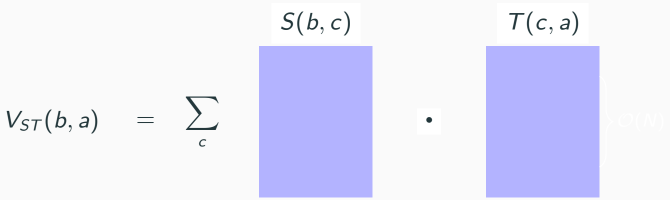
$$\delta Q = \sum_{a,b} \delta R(\alpha,\beta) \cdot V_{ST}(\beta,\alpha)$$

$$Q = Q + \delta Q$$

- Time for updates to R: $\mathcal{O}(1)$ to look up in V_{ST}

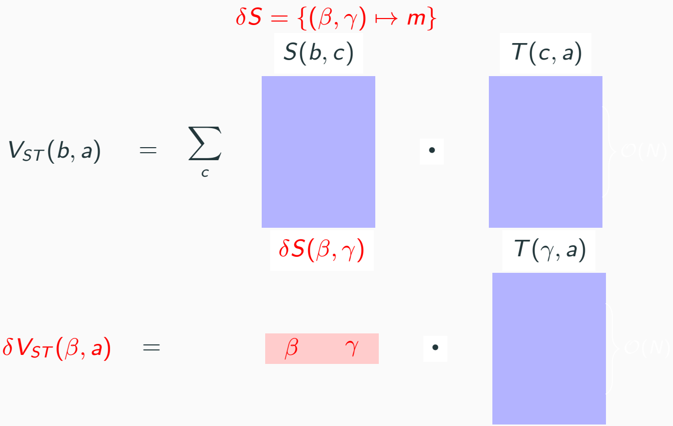
Higher-Order Incremental View Maintenance

Maintain V_{ST} under updates



Higher-Order Incremental View Maintenance

Maintain V_{ST} under updates



$\delta S = \{(\beta, \gamma) \mapsto m\}$

$S(b, c)$

$T(c, a)$

$V_{ST}(b, a) \quad = \quad \sum_{c}$

$\cdot$

$\}\mathcal{O}(N)$

$\delta S(\beta, \gamma)$

$T(\gamma, a)$

$\delta V_{ST}(\beta, a) \quad =$

$\beta \quad \gamma$

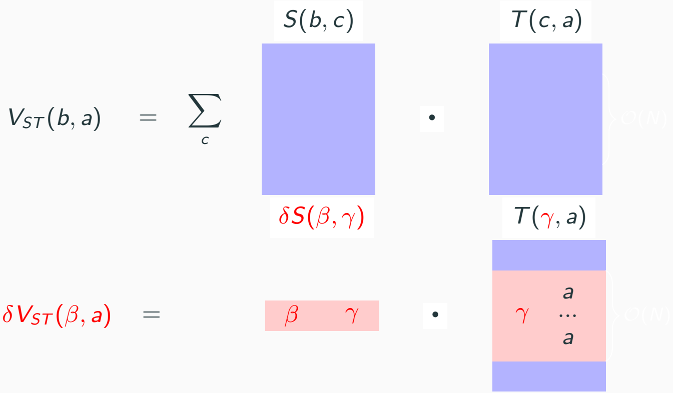
$\cdot$

$\}\mathcal{O}(N)$

Higher-Order Incremental View Maintenance

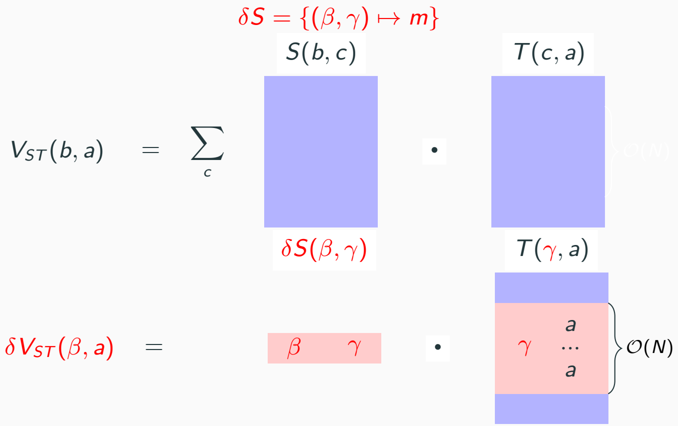
Maintain V_{ST} under updates

$\delta S = \{(\beta, \gamma) \mapsto m\}$



Higher-Order Incremental View Maintenance

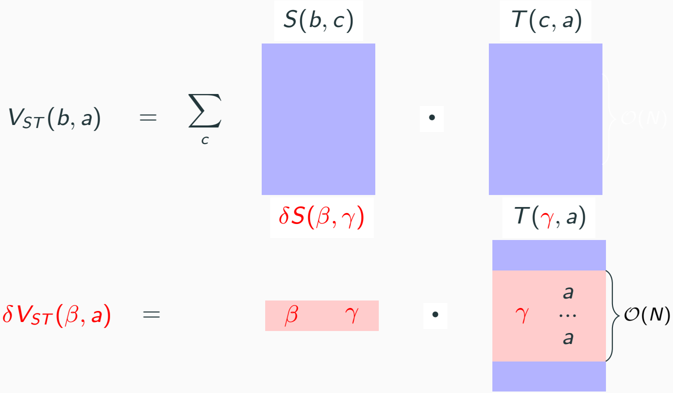
Maintain V_{ST} under updates


$\delta S = \{(\beta, \gamma) \mapsto m\}$
$S(b, c)$
$T(c, a)$
$V_{ST}(b, a) \quad = \quad \sum_{c}$
$\bullet$
$\delta S(\beta, \gamma)$
$T(\gamma, a)$
$\delta V_{ST}(\beta, a) \quad =$
β
γ
$\bullet$
γ
a
$\cdots$
a
$\mathcal{O}(N)$

Higher-Order Incremental View Maintenance

Maintain V_{ST} under updates

$$\delta S = \{(\beta, \gamma) \mapsto m\}$$



$$V_{ST}(b, a) = \sum_{c} S(b, c) \cdot T(c, a)$$

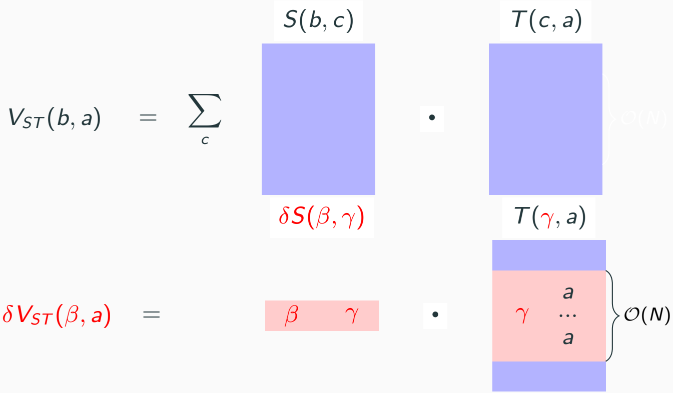
$$\delta V_{ST}(\beta, a) = \delta S(\beta, \gamma) \cdot T(\gamma, a)$$

$$V_{ST}(\beta, a) = V_{ST}(\beta, a) + \delta V_{ST}(\beta, a)$$

Higher-Order Incremental View Maintenance

Maintain V_{ST} under updates

$$\delta S = \{(\beta, \gamma) \mapsto m\}$$



$$V_{ST}(\beta, a) = V_{ST}(\beta, a) + \delta V_{ST}(\beta, a)$$

- Time for updates to S and T: $\mathcal{O}(N)$ to maintain V_{ST}
- Space: $\mathcal{O}(N^2)$ to store input relations and V_{ST}

Lower Bound for Maintaining the Triangle Count

The Boolean Triangle Detection Problem

Boolean Triangle Detection Query

$$Q_b = \bigvee_{a,b,c} R(a,b) \wedge S(b,c) \wedge T(c,a)$$

The Boolean Triangle Detection Problem

Boolean Triangle Detection Query

$$Q_b = \bigvee_{a,b,c} R(a,b) \wedge S(b,c) \wedge T(c,a)$$

Let **D** be the database instance and N the number of tuples in **D**.
For any $\gamma > 0$, there is no algorithm that incrementally maintains Q_b with

update time	enumeration delay
$\mathcal{O}(N^{\frac{1}{2}-\gamma})$	$\mathcal{O}(N^{1-\gamma})$

unless the Online Vector-Matrix-Vector Multiplication (OuMv) Conjecture fails.

Online Vector-Matrix-Vector Multiplication

The OuMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1, \mathbf{v}_1), \ldots, (\mathbf{u}_n, \mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.
- Goal: After seeing each pair $(\mathbf{u}_r, \mathbf{v}_r)$, output $\mathbf{u}_r^\top \mathbf{M} \mathbf{v}_r$

Online Vector-Matrix-Vector Multiplication

The OuMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1, \mathbf{v}_1), \ldots, (\mathbf{u}_n, \mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.
- Goal: After seeing each pair $(\mathbf{u}_r, \mathbf{v}_r)$, output $\mathbf{u}_r^\top \mathbf{M} \mathbf{v}_r$

The OuMv Conjecture [*STOC 2015*]

For any $\gamma > 0$, there is no algorithm that solves OuMv in time $\mathcal{O}(n^{3-\gamma})$.

Online Vector-Matrix-Vector Multiplication

The OuMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1, \mathbf{v}_1), \ldots, (\mathbf{u}_n, \mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.
- Goal: After seeing each pair $(\mathbf{u}_r, \mathbf{v}_r)$, output $\mathbf{u}_r^\top \mathbf{M} \mathbf{v}_r$

The OuMv Conjecture [*STOC 2015*]

For any $\gamma > 0$, there is no algorithm that solves OuMv in time $\mathcal{O}(n^{3-\gamma})$.

The OuMv Conjecture is implied by the OMv Conjecture [*STOC 2015*]

The OMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n Boolean column-vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ of size n arriving one after the other
- Goal: After seeing each vector $\mathbf{v}_r$, output $\mathbf{M}\mathbf{v}_r$

Online Vector-Matrix-Vector Multiplication

The OuMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n pairs $(\mathbf{u}_1, \mathbf{v}_1), \dots, (\mathbf{u}_n, \mathbf{v}_n)$ of Boolean column-vectors of size n arriving one after the other.
- Goal: After seeing each pair $(\mathbf{u}_r, \mathbf{v}_r)$, output $\mathbf{u}_r^\top \mathbf{M} \mathbf{v}_r$

The OuMv Conjecture [*STOC 2015*]

For any $\gamma > 0$, there is no algorithm that solves OuMv in time $\mathcal{O}(n^{3-\gamma})$.

The OuMv Conjecture is implied by the OMv Conjecture [*STOC 2015*]

The OMv problem:

- Input: An $n \times n$ Boolean matrix $\mathbf{M}$ and n Boolean column-vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of size n arriving one after the other
- Goal: After seeing each vector $\mathbf{v}_r$, output $\mathbf{M}\mathbf{v}_r$

The OMv Conjecture

For any $\gamma > 0$, there is no algorithm that solves OMv in time $\mathcal{O}(n^{3-\gamma})$.

Proof Idea

- Assume there is an algorithm $\mathcal{A}$ that can maintain Triangle Detection Query Q_b with

amortized update time	enumeration delay
$\mathcal{O}(N^{\frac{1}{2}-\gamma})$	$\mathcal{O}(N^{1-\gamma})$

  for some $\gamma > 0$.

- We design an algorithm $\mathcal{B}$ that uses the oracle $\mathcal{A}$ to solve OuMv in subcubic time in n. $\Longrightarrow$ **Contradicts the OuMv Conjecture!**

Proof Idea

- Assume there is an algorithm $\mathcal{A}$ that can maintain Triangle Detection Query Q_b with

amortized update time	enumeration delay
$\mathcal{O}(N^{\frac{1}{2}-\gamma})$	$\mathcal{O}(N^{1-\gamma})$

  for some $\gamma > 0$.

- We design an algorithm $\mathcal{B}$ that uses the oracle $\mathcal{A}$ to solve OuMv in subcubic time in n. $\Longrightarrow$ **Contradicts the OuMv Conjecture!**

Algorithm $\mathcal{B}$

- Relation S encodes the matrix $\mathbf{M}$: $S(i, j) = \mathbf{M}[i, j]$
- In each round $r \in [n]$:
  - Relation R encodes the vector $\mathbf{u}_r$: $R(\mathrm{a}, i) = \mathbf{u}_r[i]$, for constant a
  - Relation T encodes the vector $\mathbf{v}_r$: $T(j, \mathrm{a}) = \mathbf{v}_r[j]$, for constant a
  - Then $\mathbf{u}_r^\top \mathbf{M} \mathbf{v}_r = Q_b$
  - Check whether $Q_b = 1$ using algorithm $\mathcal{A}$.

Example Encoding for u, M, and v

$\mathbf{u}^\top$

0	1	0

$\mathbf{M}$

0	1	0
1	1	0
1	0	1

$\mathbf{v}$

1
0
0

$\mathbf{u}^\top\mathbf{M}\mathbf{v}$

1

R

A	B	val
a	2	1

S

B	C	val
2	1	1
3	1	1
1	2	1
2	2	1
3	3	1

T

C	A	val
1	a	1

Q_b

$\emptyset$	val
( )	1

Proof Sketch: Algorithm $\mathcal{B}$

(1) For $i, j \in [n]$: $S(i, j) = \mathbf{M}[i, j]$ ($\leq n^2$ insertions)

Proof Sketch: Algorithm $\mathcal{B}$

(1) For $i, j \in [n]$: $S(i, j) = \mathbf{M}[i, j]$ ($\leq n^2$ insertions)

(2) In each round $r \in [n]$:

- Delete all tuples in R and T ($\leq 2n$ deletions)

Proof Sketch: Algorithm $\mathcal{B}$

(1) For $i, j \in [n]$: $S(i, j) = \mathbf{M}[i, j]$ ($\leq n^2$ insertions)

(2) In each round $r \in [n]$:

- Delete all tuples in R and T ($\leq 2n$ deletions)
- Insert into R and T:
  For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$ ($\leq 2n$ insertions)

Proof Sketch: Algorithm $\mathcal{B}$

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$ ($\leq n^2$ insertions)

(2) In each round $r \in [n]$:

- Delete all tuples in R and T ($\leq 2n$ deletions)
- Insert into R and T:
  For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$ ($\leq 2n$ insertions)
- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

$$\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1 \Leftrightarrow \exists i, j \in [n] : \mathbf{u}_r[i] = 1, \mathbf{M}[i,j] = 1, \mathbf{v}_r[j] = 1$$

Proof Sketch: Algorithm $\mathcal{B}$

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$ ($\leq n^2$ insertions)

(2) In each round $r \in [n]$:

- Delete all tuples in R and T ($\leq 2n$ deletions)
- Insert into R and T:
  For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$ ($\leq 2n$ insertions)
- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

$$\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1 \Leftrightarrow \exists i, j \in [n] : \mathbf{u}_r[i] = 1, \mathbf{M}[i,j] = 1, \mathbf{v}_r[j] = 1$$

$\mathcal{B}$ constructs a database of size $N = \mathcal{O}(n^2)$.

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i, j) = \mathbf{M}[i, j]$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$

$$\mathcal{O}(\underbrace{n^2}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{3-2\gamma})$$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$

$$\mathcal{O}(\underbrace{n^2}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{3-2\gamma})$$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

$$\mathcal{O}(\underbrace{4n}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{2-2\gamma})$$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$

$$\mathcal{O}(\underbrace{n^2}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{3-2\gamma})$$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

$$\mathcal{O}(\underbrace{4n}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{2-2\gamma})$$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

$$\mathcal{O}(\underbrace{(n^2)^{1-\gamma}}_{\text{delay}}) = \mathcal{O}(n^{2-2\gamma})$$

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$

$$\mathcal{O}(\underbrace{n^2}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{3-2\gamma})$$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

$$\mathcal{O}(\underbrace{4n}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{2-2\gamma})$$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

$$\mathcal{O}(\underbrace{(n^2)^{1-\gamma}}_{\text{delay}}) = \mathcal{O}(n^{2-2\gamma})$$

For n rounds: $\mathcal{O}(n(n^{2-2\gamma} + n^{2-2\gamma})) = \mathcal{O}(n^{3-2\gamma})$

Proof Sketch: Time Analysis

Recall $\mathcal{A}$ needs $\mathcal{O}((n^2)^{\frac{1}{2}-\gamma})$ update time and $\mathcal{O}((n^2)^{1-\gamma})$ delay

(1) For $i, j \in [n]$: $S(i,j) = \mathbf{M}[i,j]$

$$\mathcal{O}(\underbrace{n^2}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{3-2\gamma})$$

(2) In each round $r \in [n]$:

- Delete all tuples in R and T
- Insert into R and T: For $i, j \in [n]$: $R(\mathtt{a}, i) = \mathbf{u}_r[i]$ and $T(j, \mathtt{a}) = \mathbf{v}_r[j]$

$$\mathcal{O}(\underbrace{4n}_{\#\text{updates}} \cdot \underbrace{(n^2)^{\frac{1}{2}-\gamma}}_{\text{update time}}) = \mathcal{O}(n^{2-2\gamma})$$

- Check $Q_b = 1$: This holds if and only if $\mathbf{u}_r^T \mathbf{M} \mathbf{v}_r = 1$

$$\mathcal{O}(\underbrace{(n^2)^{1-\gamma}}_{\text{delay}}) = \mathcal{O}(n^{2-2\gamma})$$

For n rounds: $\mathcal{O}(n(n^{2-2\gamma} + n^{2-2\gamma})) = \mathcal{O}(n^{3-2\gamma})$

Overall time: $\mathcal{O}(n^{3-2\gamma} + n^{3-2\gamma}) = \mathcal{O}(n^{3-2\gamma}) \Rightarrow$ Contradicts OuMv Conjecture!

Closing the Complexity Gap

Closing the Complexity Gap

Complexity bounds for the maintenance of the triangle count

Known Upper Bound

Update Time: $\mathcal{O}(N)$

Space: $\mathcal{O}(N)$

Known Lower Bound

Update time: not $\mathcal{O}(N^{\frac{1}{2}-\gamma})$ for any $\gamma > 0$

under the OuMv Conjecture

Closing the Complexity Gap

Complexity bounds for the maintenance of the triangle count

Known Upper Bound

Update Time: $\mathcal{O}(N)$

Space: $\mathcal{O}(N)$

Can the triangle count be maintained with sublinear update time?

Known Lower Bound

Update time: not $\mathcal{O}(N^{\frac{1}{2}-\gamma})$ for any $\gamma > 0$

under the OuMv Conjecture

Closing the Complexity Gap

Complexity bounds for the maintenance of the triangle count

Known Upper Bound

Update Time: $\mathcal{O}(N)$

Space: $\mathcal{O}(N)$

Can the triangle count be maintained with sublinear update time?

Yes: IVM$^{\varepsilon}$

Amortized update time:

$\mathcal{O}(N^{\frac{1}{2}})$

This is worst-case optimal

Known Lower Bound

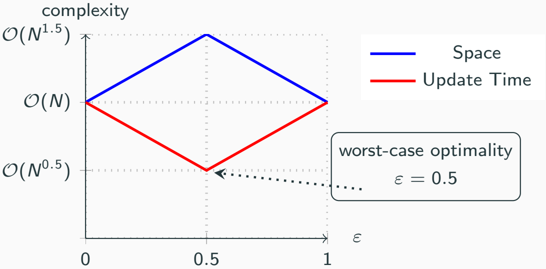
Update time: not $\mathcal{O}(N^{\frac{1}{2}-\gamma})$ for any $\gamma > 0$

under the OuMv Conjecture

IVM$^{\varepsilon}$ Exhibits a Time-Space Tradeoff

Given $\varepsilon \in [0, 1]$, IVM$^{\varepsilon}$ maintains the triangle count with

- $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$ amortized update time
- $\mathcal{O}(N^{1+\min\{\varepsilon,1-\varepsilon\}})$ space
- $\mathcal{O}(N^{\frac{3}{2}})$ preprocessing time
- $\mathcal{O}(1)$ answer time.



(Linear space possible with a slightly more involved argument)

Inside IVM$^{\varepsilon}$

Main Techniques used in IVM$^{\varepsilon}$

- Compute the delta like in first-order IVM

- Materialize views like in higher-order IVM

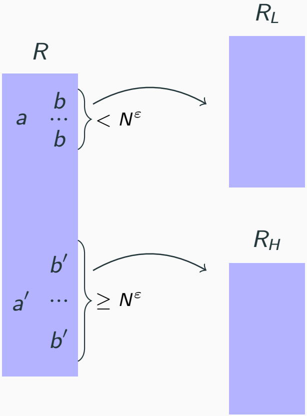
- New ingredient: Use adaptive processing based on data skew

  $\Longrightarrow$ Treat *heavy* values differently from *light* values

Heavy/Light Partitioning of Relations

Partition R based on A into

- a light part $R_L = \{t \in R \mid |\sigma_{A=t.A}| < N^{\varepsilon}\}$,
- a heavy part $R_H = R \backslash R_L$!



Derived Bounds

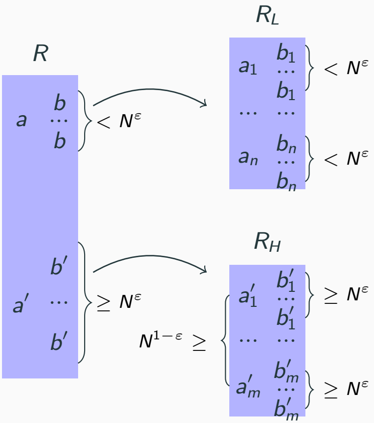
from light part:
for all A-values a, $|\sigma_{A=a} R_L| < N^{\varepsilon}$

from heavy part:
for all A-values a, $|\sigma_{A=a} R_H| \geq N^{\varepsilon}$
and $|\pi_A R_H| \cdot N^{\varepsilon} \leq N$
$\Longrightarrow |\pi_A R_H| \leq N^{1-\varepsilon}$

Heavy/Light Partitioning of Relations

Partition R based on A into

- a light part $R_L = \{t \in R \mid |\sigma_{A=t.A}| < N^{\varepsilon}\}$,
- a heavy part $R_H = R \backslash R_L$!



Derived Bounds

from light part:
for all A-values a, $|\sigma_{A=a}R_L| < N^{\varepsilon}$

from heavy part:
$|\pi_A R_H| \leq N^{1-\varepsilon}$, since
for all A-values a, $|\sigma_{A=a}R_H| \geq N^{\varepsilon}$
and $|\pi_A R_H| \cdot N^{\varepsilon} \leq N$

Heavy/Light Partitioning of Relations

Likewise, partition

- $S = S_L \cup S_H$ based on B, and
- $T = T_L \cup T_H$ based on C!

Q is the **sum** of skew-aware queries

$$Q = \sum_{a,b,c} R_U(a,b) \cdot S_V(b,c) \cdot T_W(c,a), \text{ for } U, V, W \in \{L, H\}.$$

Adaptive Maintenance Strategy

Given an update $\delta R_* = \{(\alpha, \beta) \mapsto m\}$, compute the delta for each of the following skew-aware queries using a different strategy:

$$Q_{*LL} = \sum_{a,b,c} R_*(a,b) \cdot S_L(b,c) \cdot T_L(c,a)$$

$$Q_{*HH} = \sum_{a,b,c} R_*(a,b) \cdot S_H(b,c) \cdot T_H(c,a)$$

$$Q_{*LH} = \sum_{a,b,c} R_*(a,b) \cdot S_L(b,c) \cdot T_H(c,a)$$

$$Q_{*HL} = \sum_{a,b,c} R_*(a,b) \cdot S_H(b,c) \cdot T_L(c,a)$$

Adaptive Maintenance Strategy

Given an update $\delta R_* = \{(\alpha, \beta) \mapsto m\}$, compute the delta for each of the following skew-aware queries using a different strategy:

$$\delta Q_{*LL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_L(c, \alpha)$$

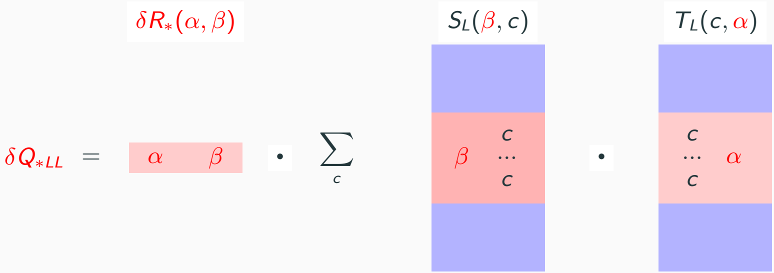
$$\delta Q_{*HH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_H(c, \alpha)$$

$$\delta Q_{*LH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_H(c, \alpha)$$

$$\delta Q_{*HL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_L(c, \alpha)$$

Adaptive Maintenance Strategy

$$\delta Q_{*LL} = \delta R_*(\alpha,\beta) \cdot \sum_c S_L(\beta,c) \cdot T_L(c,\alpha)$$



$\delta R_*(\alpha,\beta)$

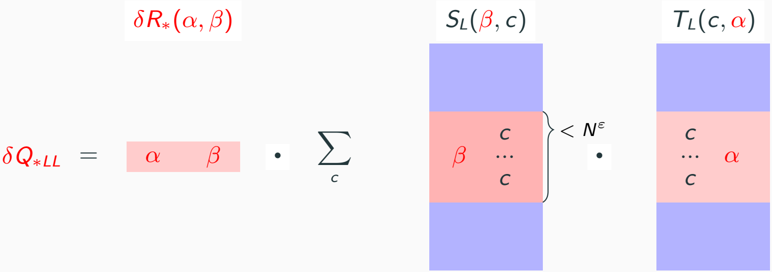
$S_L(\beta,c)$

$T_L(c,\alpha)$

δQ_{*LL} $=$ α β $\bullet$ $\sum_c$ β c $\cdots$ c $\bullet$ c $\cdots$ c α

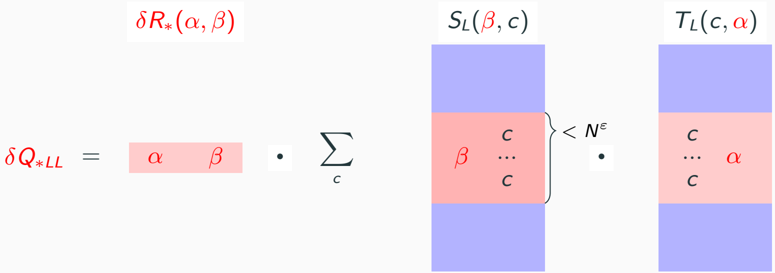
Adaptive Maintenance Strategy

$$\delta Q_{*LL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_L(c, \alpha)$$



Adaptive Maintenance Strategy

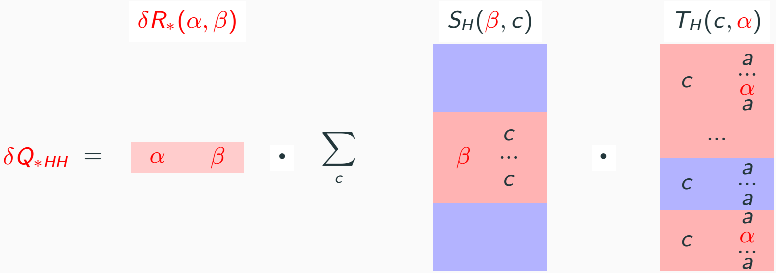
$$\delta Q_{*LL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_L(c, \alpha)$$



Update time: $\mathcal{O}(N^{\varepsilon})$ to intersect the lists of C-values from S_L and T_L

Adaptive Maintenance Strategy

$$\delta Q_{*HH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_H(c, \alpha)$$



$\delta R_*(\alpha, \beta)$

$S_H(\beta, c)$

$T_H(c, \alpha)$

$\delta Q_{*HH} =$ α β $\cdot \sum_c$ β c $\cdots$ c $\cdot$

c a $\cdots$ α a

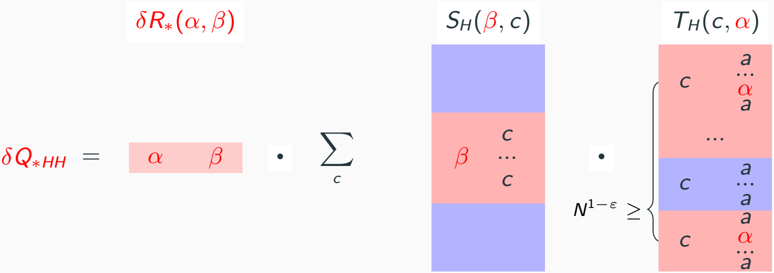
$\cdots$

c a $\cdots$ a

c a α $\cdots$ a

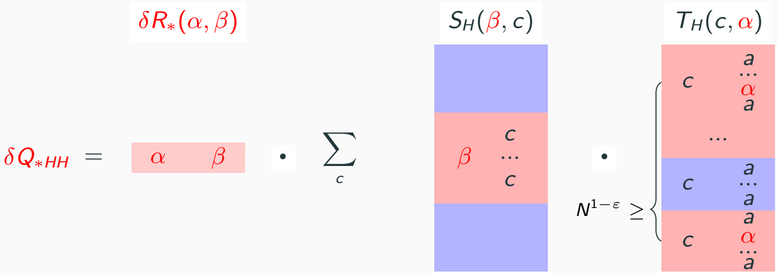
Adaptive Maintenance Strategy

$$\delta Q_{*HH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_H(c, \alpha)$$



Adaptive Maintenance Strategy

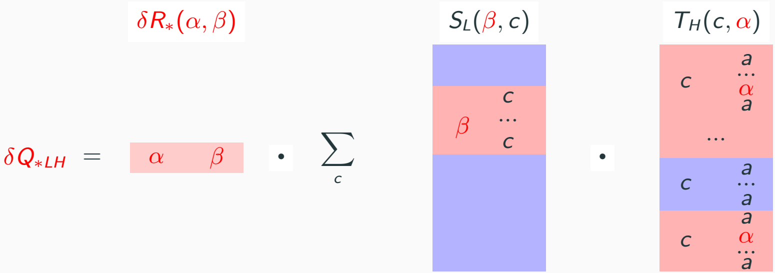
$$\delta Q_{*HH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_H(c, \alpha)$$



Update time: $\mathcal{O}(N^{1-\varepsilon})$ to intersect the lists of C-values from S_H and T_H

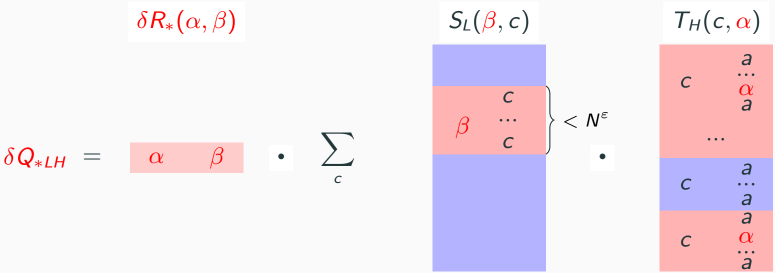
Adaptive Maintenance Strategy

$$\delta Q_{*LH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_H(c, \alpha)$$



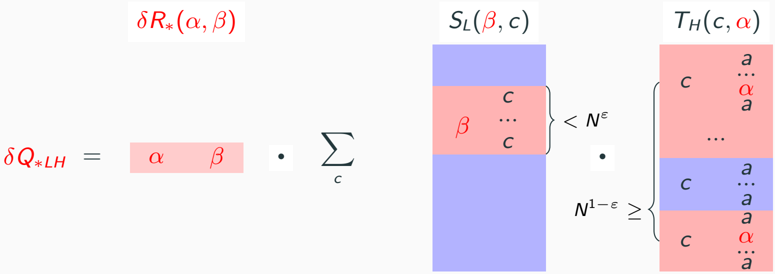
Adaptive Maintenance Strategy

$$\delta Q_{*LH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_H(c, \alpha)$$



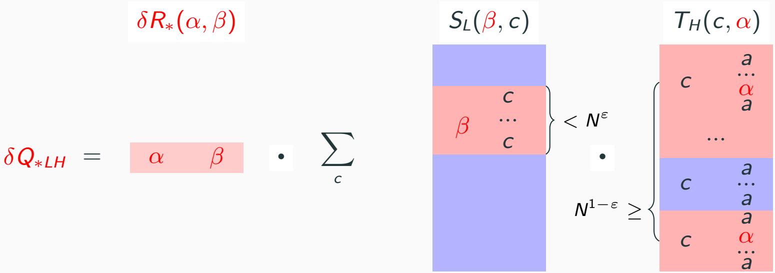
Adaptive Maintenance Strategy

$$\delta Q_{*LH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_H(c, \alpha)$$



Adaptive Maintenance Strategy

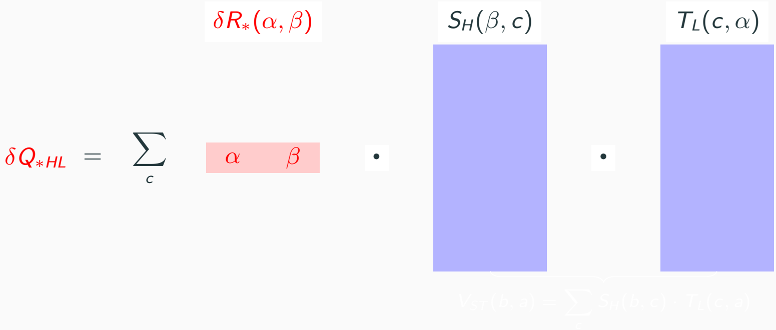
$$\delta Q_{*LH} = \delta R_*(\alpha, \beta) \cdot \sum_c S_L(\beta, c) \cdot T_H(c, \alpha)$$



Update time: $\mathcal{O}(N^{\min\{\varepsilon, 1-\varepsilon\}})$ to intersect the lists of C-values from S_L and T_H

Adaptive Maintenance Strategy

$$\delta Q_{*HL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_L(c, \alpha)$$



$\delta R_*(\alpha, \beta)$

$S_H(\beta, c)$

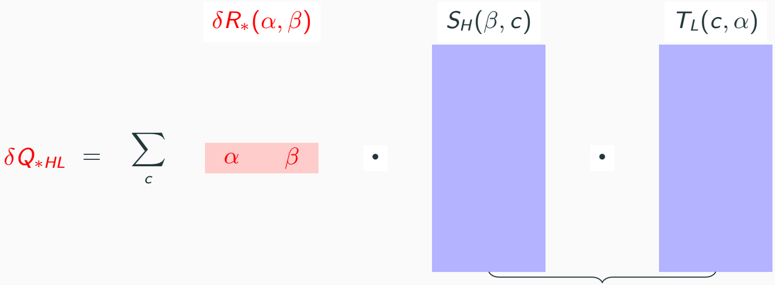
$T_L(c, \alpha)$

$$\delta Q_{*HL} = \sum_c \quad \alpha \quad \beta \quad \cdot \quad \cdot$$

$$V_{ST}(b, a) = \sum_c S_H(b, c) \cdot T_L(c, a)$$

Adaptive Maintenance Strategy

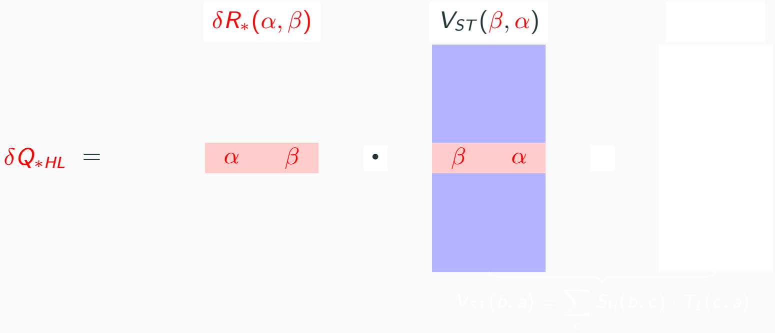
$$\delta Q_{*HL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_L(c, \alpha)$$



$$V_{ST}(b, a) = \sum_c S_H(b, c) \cdot T_L(c, a)$$

Adaptive Maintenance Strategy

$$\delta Q_{*HL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_L(c, \alpha)$$



δQ_{*HL} =

$\delta R_*(\alpha, \beta)$

α β

•

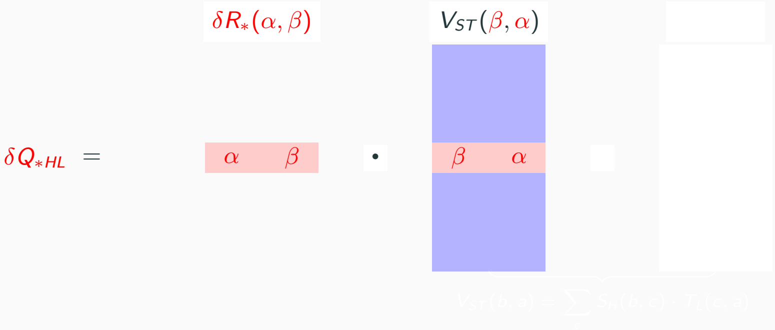
$V_{ST}(\beta, \alpha)$

β α

$$V_{ST}(b, a) = \sum_c S_H(b, c) \cdot T_L(c, a)$$

Adaptive Maintenance Strategy

$$\delta Q_{*HL} = \delta R_*(\alpha, \beta) \cdot \sum_c S_H(\beta, c) \cdot T_L(c, \alpha)$$



$\delta R_*(\alpha, \beta)$

$V_{ST}(\beta, \alpha)$

$\delta Q_{*HL} =$ α β • β α

$V_{ST}(b, a) = \sum_c S_H(b, c) \cdot T_L(c, a)$

Update time: $\mathcal{O}(1)$ to look up in V_{ST}, assuming V_{ST} is already materialized

Summary of Adaptive Maintenance Strategies

Maintenance for an update $\delta R_* = \{(\alpha, \beta) \mapsto m\}$:

Skew-aware View	Evaluation from left to right	Time
$\sum_{a,b,c} R_*(a,b) \cdot S_L(b,c) \cdot T_L(c,a)$	$\delta R_*(\alpha,\beta) \cdot \sum_{c} S_L(\beta,c) \cdot T_L(c,\alpha)$	$\mathcal{O}(N^{\varepsilon})$
$\sum_{a,b,c} R_*(a,b) \cdot S_H(b,c) \cdot T_H(c,a)$	$\delta R_*(\alpha,\beta) \cdot \sum_{c} T_H(c,\alpha) \cdot S_H(\beta,c)$	$\mathcal{O}(N^{1-\varepsilon})$
$\sum_{a,b,c} R_*(a,b) \cdot S_L(b,c) \cdot T_H(c,a)$	$\delta R_*(\alpha,\beta) \cdot \sum_{c} S_L(\beta,c) \cdot T_H(c,\alpha)$	$\mathcal{O}(N^{\varepsilon})$
	or	
	$\delta R_*(\alpha,\beta) \cdot \sum_{c} T_H(c,\alpha) \cdot S_L(\beta,c)$	$\mathcal{O}(N^{1-\varepsilon})$
$\sum_{a,b,c} R_*(a,b) \cdot S_H(b,c) \cdot T_L(c,a)$	$\delta R_*(\alpha,\beta) \cdot V_{ST}(\beta,\alpha)$	$\mathcal{O}(1)$

Overall update time: $\mathcal{O}(N^{\max(\varepsilon, 1-\varepsilon)})$

Auxiliary Materialized Views

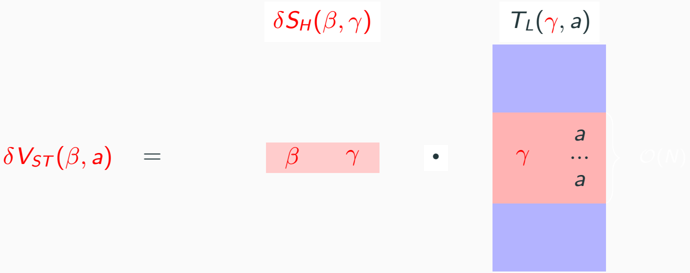
$$V_{RS}(a,c) = \sum_{b} R_H(a,b) \cdot S_L(b,c)$$

$$V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$$

$$V_{TR}(a,c) = \sum_{a} T_H(c,a) \cdot R_L(a,b)$$

Maintenance of Auxiliary Views

Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta S_H = \{(\beta,\gamma) \mapsto m\}$

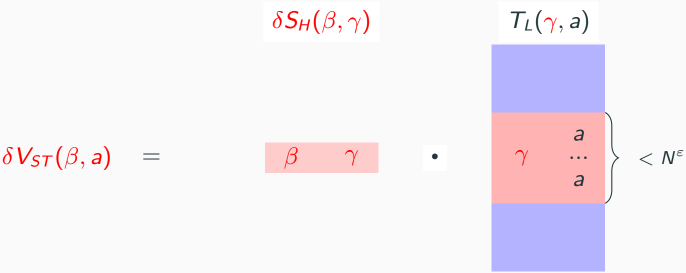


$\delta S_H(\beta,\gamma)$ $T_L(\gamma,a)$

$\delta V_{ST}(\beta,a) \quad = \quad \beta \quad \gamma \quad \bullet \quad \gamma \quad \begin{matrix} a \\ \cdots \\ a \end{matrix} \quad \Big\} \; \mathcal{O}(N)$

Maintenance of Auxiliary Views

Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta S_H = \{(\beta,\gamma) \mapsto m\}$



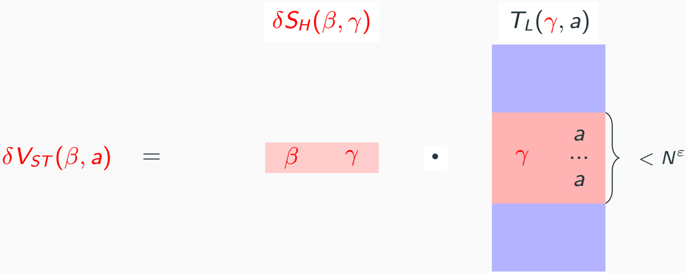
$\delta S_H(\beta,\gamma)$

$T_L(\gamma,a)$

$\delta V_{ST}(\beta,a) \quad = \quad \begin{array}{cc} \beta & \gamma \end{array} \quad \bullet \quad \left.\begin{array}{cc} & a \\ \gamma & \cdots \\ & a \end{array}\right\} < N^{\varepsilon}$

Maintenance of Auxiliary Views

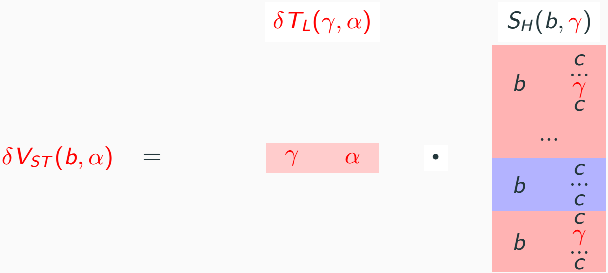
Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta S_H = \{(\beta,\gamma) \mapsto m\}$



Update time: $\mathcal{O}(N^{\varepsilon})$ to iterate over a-values paired with γ from T_L

Maintenance of Auxiliary Views

Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta T_L = \{(\gamma,\alpha) \mapsto m\}$



$\delta T_L(\gamma,\alpha)$

$S_H(b,\gamma)$

$\delta V_{ST}(b,\alpha) \quad =$

$\gamma \quad \alpha$

$\bullet$

$b \quad c, \dots, \gamma, c$

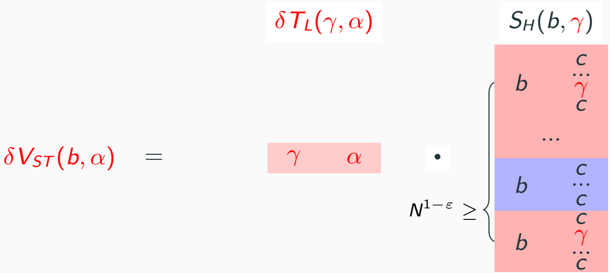
$\dots$

$b \quad c, \dots, c$

$b \quad c, \gamma, \dots, c$

Maintenance of Auxiliary Views

Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta T_L = \{(\gamma,\alpha) \mapsto m\}$



$\delta V_{ST}(b,\alpha) \quad = \quad$

$\delta T_L(\gamma,\alpha)$

γ	α

$\bullet$

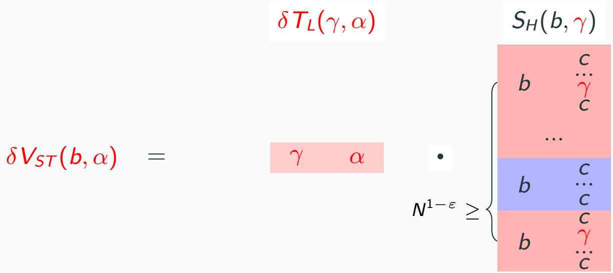
$S_H(b,\gamma)$

b	c $\vdots$ γ c
	$\dots$
b	c $\vdots$ c
b	c γ $\vdots$ c

$N^{1-\varepsilon} \geq \{$

Maintenance of Auxiliary Views

Maintain $V_{ST}(b,a) = \sum_{c} S_H(b,c) \cdot T_L(c,a)$ under update $\delta T_L = \{(\gamma,\alpha) \mapsto m\}$



$\delta V_{ST}(b,\alpha)$ $=$

$\delta T_L(\gamma,\alpha)$

γ α

$\bullet$

$S_H(b,\gamma)$

b c $\cdots$ γ c

$\cdots$

b c $\cdots$ c

b c γ $\cdots$ c

$N^{1-\varepsilon} \geq$

Update time: $\mathcal{O}(N^{1-\varepsilon})$ to iterate over b-values paired with γ from S_H

Maintenance of Auxiliary Views: Summary

$$V_{RS}(a, c) = \sum_{b} R_H(a, b) \cdot S_L(b, c)$$

$$V_{ST}(b, a) = \sum_{c} S_H(b, c) \cdot T_L(c, a)$$

$$V_{TR}(a, c) = \sum_{a} T_H(c, a) \cdot R_L(a, b)$$

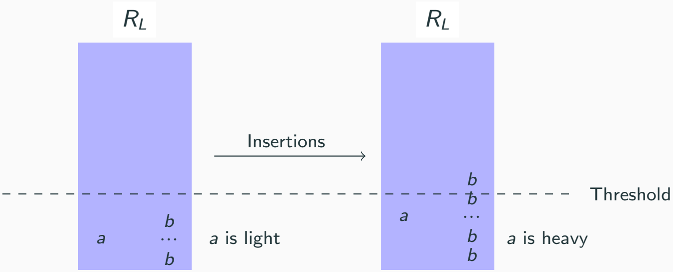
Maintenance Complexity

- Time: $\mathcal{O}(N^{\max\{\varepsilon, 1-\varepsilon\}})$
- Space: $\mathcal{O}(N^{1+\min\{\varepsilon, 1-\varepsilon\}})$

Updates can change frequencies of values & heavy/light threshold

Rebalancing Partitions

Updates can change the frequencies of values in the relation parts

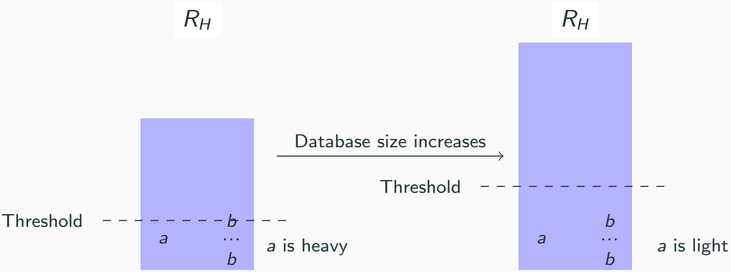


Minor Rebalancing

- Transfer $\mathcal{O}(N^{\varepsilon})$ tuples from one to the other part of the same relation
- Time complexity: $\mathcal{O}(N^{\varepsilon+\max\{\varepsilon,1-\varepsilon\}})$

Rebalancing Partitions

Updates can change the heavy-light threshold!



Major Rebalancing

- Recompute partitions and views from scratch
- Time complexity: $\mathcal{O}(N^{1+\max\{\varepsilon,1-\varepsilon\}})$

Amortization of Rebalancing Times

- Both forms of rebalancing require superlinear time

Amortization of Rebalancing Times

- Both forms of rebalancing require superlinear time
- The rebalancing times amortize over sequences of updates
  - Amortized minor rebalancing time: $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$
  - Amortized major rebalancing time: $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$

$$\dots\ \textit{update}\quad \overset{\mathcal{O}(N^{\varepsilon+\max\{\varepsilon,1-\varepsilon\}})}{\textit{minor}}\quad \underbrace{\textit{update}\ \dots\ \textit{update}}_{\Omega(N^{\varepsilon})}\quad \overset{\mathcal{O}(N^{\varepsilon+\max\{\varepsilon,1-\varepsilon\}})}{\textit{minor}}\quad \textit{update}\ \dots$$

$$\dots\ \textit{update}\quad \overset{\mathcal{O}(N^{1+\max\{\varepsilon,1-\varepsilon\}})}{\textit{major}}\quad \underbrace{\textit{update}\ \dots\ \textit{update}}_{\Omega(N)}\quad \overset{\mathcal{O}(N^{1+\max\{\varepsilon,1-\varepsilon\}})}{\textit{major}}\quad \textit{update}\ \dots$$

Amortization of Rebalancing Times

- Both forms of rebalancing require superlinear time
- The rebalancing times amortize over sequences of updates
  - Amortized minor rebalancing time: $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$
  - Amortized major rebalancing time: $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$
- Overall amortized rebalancing time: $\mathcal{O}(N^{\max\{\varepsilon,1-\varepsilon\}})$

$$\dots\quad \textit{update}\quad \overset{\mathcal{O}(N^{\varepsilon+\max\{\varepsilon,1-\varepsilon\}})}{\textit{minor}}\quad \underbrace{\textit{update}\ \dots\ \textit{update}}_{\Omega(N^{\varepsilon})}\quad \overset{\mathcal{O}(N^{\varepsilon+\max\{\varepsilon,1-\varepsilon\}})}{\textit{minor}}\quad \textit{update}\ \dots$$

$$\dots\quad \textit{update}\quad \overset{\mathcal{O}(N^{1+\max\{\varepsilon,1-\varepsilon\}})}{\textit{major}}\quad \underbrace{\textit{update}\ \dots\ \textit{update}}_{\Omega(N)}\quad \overset{\mathcal{O}(N^{1+\max\{\varepsilon,1-\varepsilon\}})}{\textit{major}}\quad \textit{update}\ \dots$$

Follow-up Work & Open Questions

Follow-up work

- TODS 2020
  - Triangle queries with different free variables
  - Strong and weak Pareto optimality
- APOCS 2021
  - Extend the triangle counting algorithm to k-clique counting
  - Parallel batch-dynamic triangle count algorithm based on the (sequential single-tuple dynamic) triangle count algorithm
- ICDT 2021
  - Update time-approximation quality trade-off for triangle counting
  - Complexity of triangle counting based on the arboricity of the data graph

Follow-up Work & Open Questions

Follow-up work

- TODS 2020
  - Triangle queries with different free variables
  - Strong and weak Pareto optimality
- APOCS 2021
  - Extend the triangle counting algorithm to k-clique counting
  - Parallel batch-dynamic triangle count algorithm based on the (sequential single-tuple dynamic) triangle count algorithm
- ICDT 2021
  - Update time-approximation quality trade-off for triangle counting
  - Complexity of triangle counting based on the arboricity of the data graph

Open questions

- Worst-case optimal (and beyond) maintenance and the update-space trade-off for functional aggregate queries
- Single-tuple updates versus batch updates

References i

[Algorithmica 1997] Noga Alon, Raphael Yuster, and Uri Zwick. *Finding and counting given length cycles.*

[SODA 2002] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. *Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs.*

[COCOON 2005] Hossein Jowhari and Mohammad Ghodsi. *New Streaming Algorithms for Counting Triangles in Graphs.*

[PODS 2006] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. *Counting Triangles in Data Streams.*

[Found. & Trends DB 2012] Rada Chirkova and Jun Yang. *Materialized Views.*

[SIGMOD R. 2013] Hung Q Ngo, Christopher Ré, and Atri Rudra. *Skew strikes back: new developments in the theory of join algorithms.*

References ii

[ICDT 2014] Todd L. Veldhuizen. *Leapfrog Triejoin: A Simple, Worst-Case Optimal Join Algorithm.*

[VLDB J. 2014] Christoph Koch, Yanif Ahmad, Oliver Kennedy, Milos Nikolic, Andres Nötzli, Daniel Lupei, and Amir Shaikhha. *DBToaster: Higher-Order Delta Processing for Dynamic, Frequently Fresh Views.*

[FOCS 2015] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. *Approximately Counting Triangles in Sublinear Time.*

[STOC 2015] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. *Unifying and Strengthening Hardness for Dynamic Problems via the Online Matrix-Vector Multiplication Conjecture.*

[PODS 2016] Andrew McGregor, Sofya Vorotnikova, and Hoa T Vu. *Better Algorithms for Counting Triangles in Data Streams.*

[PODS 2017] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. *Answering Conjunctive Queries Under Updates.*

References iii

[SIGMOD 2017] Muhammad Idris, Martín Ugarte, and Stijn Vansummeren. *The Dynamic Yannakakis Algorithm: Compact and Efficient Query Processing Under Updates.*

[Theor. Comput. Sci. 2017] Graham Cormode and Hossein Jowhari. *A Second Look at Counting Triangles in Graph Streams (Corrected).*

[Found. & Trends DB 2018] Paraschos Koutris, Semih Salihoglu, and Dan Suciu. *Algorithmic Aspects of Parallel Data Processing.*

[ICDT 2018] Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. *Answering UCQs Under Updates and in the Presence of Integrity Constraints.*

[ICM 2018] Virginia Vassilevska Williams. *On Some Fine-Grained Questions in Algorithms and Complexity.*

[SIGMOD 2018] Nikolic, Milos, and Dan Olteanu. *Incremental view maintenance with triple lock factorization benefits.*

References iv

[ICDT 2019] Ahmet Kara, Milos Nikolic, Hung Q. Ngo, Dan Olteanu, and Haozhe Zhang. *Counting Triangles under Updates in Worst-Case Optimal Time.*

[APOCS 2021] Laxman Dhulipala, Quanquan C. Liu, Julian Shun, Shangdi Yu. *Parallel Batch-Dynamic k-Clique Counting.*

[ICDT 2021] Shangqi Lu, Yufei Tao. *Towards Optimal Dynamic Indexes for Approximate (and Exact) Triangle Counting.*

2. Constant Update Time & Enumeration Delay

Queries with Constant Update Time & Delay

Q: Which queries admit

constant update time and enumeration delay in the worst-case?

Queries with Constant Update Time & Delay

Q: Which queries admit

constant update time and enumeration delay in the worst-case?

A: Q-hierarchical queries [PODS 2017]

Queries with Constant Update Time & Delay

Q: Which queries admit

constant update time and enumeration delay in the worst-case?

A: Q-hierarchical queries [PODS 2017]

A: Queries that become q-hierarchical under functional dependencies [ICDE 2009, VLDBJ 2023, RelationalAI]

Queries with Constant Update Time & Delay

Q: Which queries admit

constant update time and enumeration delay in the worst-case?

A: Q-hierarchical queries [PODS 2017]

A: Queries that become q-hierarchical under functional dependencies [ICDE 2009, VLDBJ 2023, RelationalAI]

A: Queries with free access patterns $Q(out|in)$ whose fractures are q-hierarchical [ICDT 2023]

Queries with Constant Update Time & Delay

Q: Which queries admit

constant update time and enumeration delay in the worst-case?

A: Q-hierarchical queries [PODS 2017]

A: Queries that become q-hierarchical under functional dependencies [ICDE 2009, VLDBJ 2023, RelationalAI]

A: Queries with free access patterns $Q(out|in)$ whose fractures are q-hierarchical [ICDT 2023]

A: Queries that become q-hierarchical under rewritings using q-hierarchical views and specific enumeration order [UZH 2023]

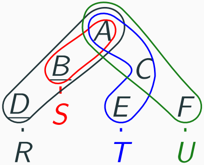
Hierarchical Queries

A query is hierarchical if for any two variables, their sets of atoms in the query are either disjoint or one is contained in the other

[VLDB 2004]

hierarchical

$$Q(b,d) = \sum_{a,c,e,f} R(a,b,d) \cdot S(a,b) \cdot T(a,c,e) \cdot U(a,c,f)$$



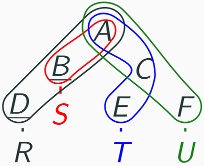
Hierarchical Queries

A query is hierarchical if for any two variables, their sets of atoms in the query are either disjoint or one is contained in the other

[VLDB 2004]

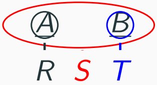
hierarchical

$$Q(b,d) = \sum_{a,c,e,f} R(a,b,d) \cdot S(a,b) \cdot T(a,c,e) \cdot U(a,c,f)$$



not hierarchical

$$Q(a,b) = R(a) \cdot S(a,b) \cdot T(b)$$



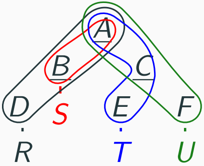
Q-Hierarchical Queries

A query is q-hierarchical if it is hierarchical and the free variables dominate the bound variables

[PODS 2017]

q-hierarchical

$$Q(a,b,c) = \sum_{d,e,f} R(a,b,d) \cdot S(a,b) \cdot T(a,c,e) \cdot U(a,c,f)$$



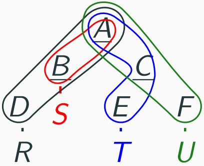
Q-Hierarchical Queries

A query is *q*-hierarchical if it is hierarchical and the free variables dominate the bound variables

[PODS 2017]

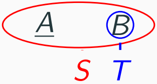
q-hierarchical

$$Q(a,b,c) = \sum_{d,e,f} R(a,b,d) \cdot S(a,b) \cdot T(a,c,e) \cdot U(a,c,f)$$



hierarchical but not *q*-hierarchical

$$Q(a) = \sum_{b} S(a,b) \cdot T(b)$$



Dichotomy for Q-Hierarchical Queries

Let Q be any conjunctive query without self-joins and D a database.

- If Q is **q-hierarchical**, then the query answer admits **O(1)** single-tuple updates and enumeration delay.
- If Q is **not q-hierarchical**, then there is **no algorithm with** $O(|D|^{1/2-\gamma})$ update time and enumeration delay for any $\gamma > 0$, unless the OMv conjecture fails.

[PODS 2017]

Queries under Functional Dependencies

Rewriting queries under functional dependencies [ICDE 2009]

- Given: Query Q and set Σ of functional dependencies
- Replace the set of variables of each atom in Q by its closure under Σ called Σ-reduct

  Under $\Sigma = \{x \to y, y \to z\}$, the closure of $\{x\}$ is $\{x, y, z\}$
- If the Σ-reduct is q-hierarchical, then Q admits constant update time and enumeration delay [VLDB J 2023]

Maintenance of Q-Hierarchical Queries

How to achieve constant update time and enumeration delay?

Recipe: [PODS 2017]

- Construct a factorized representation of the query answer [ICDT 2012]
- Such factorizations admit constant-delay enumeration
- Apply updates directly on the factorization

F-IVM system [https://github.com/fdbresearch/FIVM] [SIGMOD 2018]

- Factorize the query answer as a tree of views
- Materialize the views to speed up updates and enumeration

Example: Query Rewriting

$$Q(w, x, y, z) = R(w, x) \cdot S(x, y) \cdot T(y, z)$$

Assume the functional dependencies: $X \rightarrow Y$ and $Y \rightarrow Z$

Q is not q-hierarchical, but its rewriting under FDs is:

$$Q'(w, x, y, z) = R'(w, x, y, z) \cdot S'(x, y, z) \cdot T'(y, z)$$

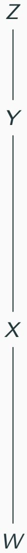
Example: Variable Order

$$Q'(w,x,y,z) = R'(w,x,y,z) \cdot S'(x,y,z) \cdot T'(y,z)$$

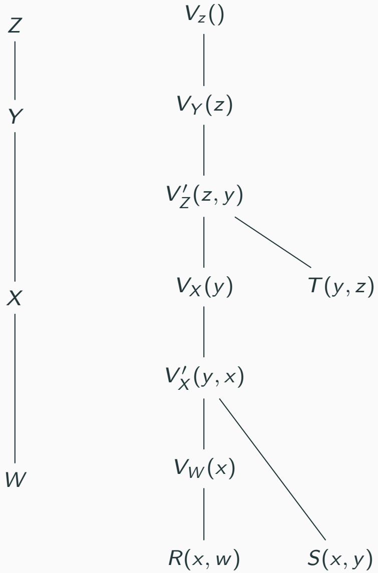
Top-down construction of variable order for Q':

- Z and Y are first as they dominate X and W
- Then X, which dominates W
- Finally W

We use this variable order also for Q



Example: View Tree



View tree construction:

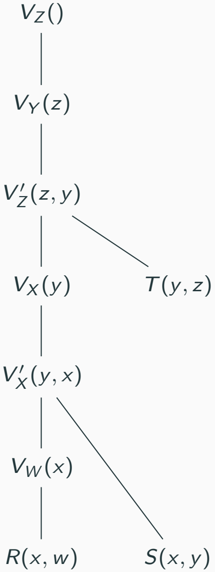
- Place relations at leaves
- Create parent view to join children

$$V'_Z(z,y) = T(y,z) \cdot V_X(y)$$
$$V'_X(y,x) = S(x,y) \cdot V_W(x)$$

- Aggregate away variables not needed for further joins

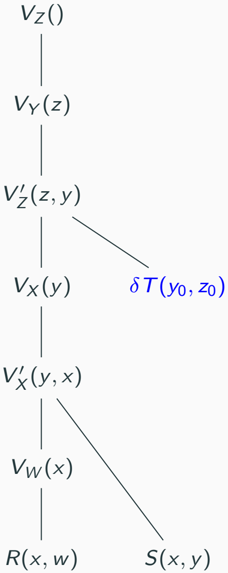
$$V_Z() = \sum_z V_Y(z)$$
$$V_Y(z) = \sum_y V'_Z(y,z)$$
$$V_X(y) = \sum_x V'_X(x,y)$$
$$V_W(x) = \sum_w R'(x,w)$$

Example: Single-Tuple Update to T



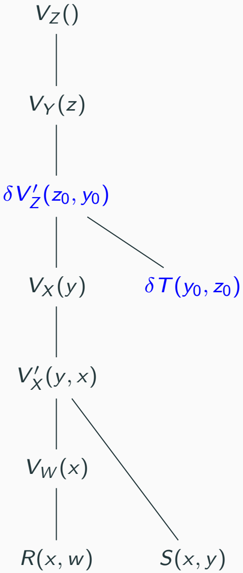
Single-tuple update to T

Example: Single-Tuple Update to T



Single-tuple update to T

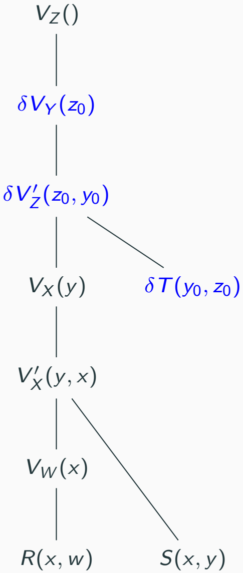
Example: Single-Tuple Update to T



Single-tuple update to T

$$\delta V'_Z(z_0, y_0) = \delta T(y_0, z_0) \cdot V_X(y_0)$$

Example: Single-Tuple Update to T

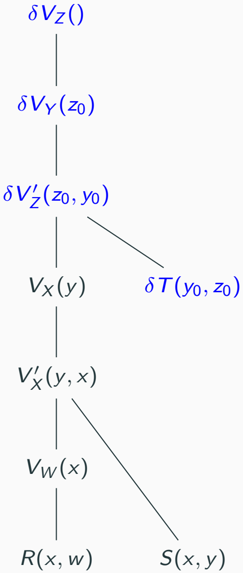


Single-tuple update to T

$$\delta V'_Z(z_0, y_0) = \delta T(y_0, z_0) \cdot V_X(y_0)$$

$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

Example: Single-Tuple Update to T



Single-tuple update to T

$$\delta V'_Z(z_0, y_0) = \delta T(y_0, z_0) \cdot V_X(y_0)$$

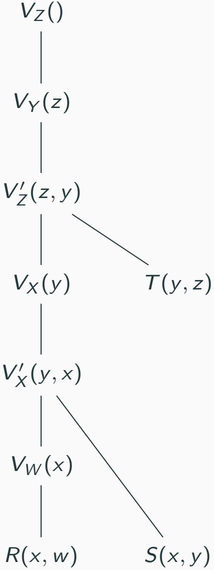
$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

$$\delta V_Z() = \sum_{z_0} \delta V_Y(z_0) = \delta V_Y(z_0)$$

For each updated view/relation A: $A := A + \delta A$

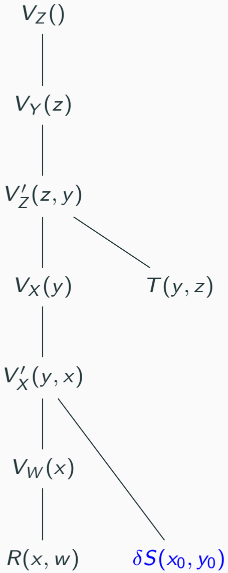
Each view update takes $O(1)$ time

Example: Single-Tuple Update to S



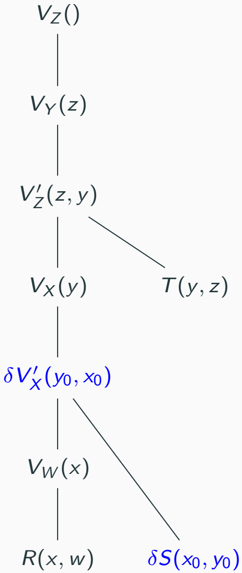
Single-tuple update to S

Example: Single-Tuple Update to S



Single-tuple update to S

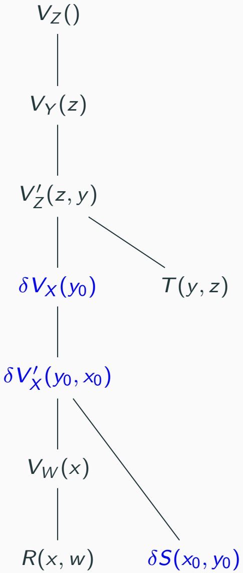
Example: Single-Tuple Update to S



Single-tuple update to S

$$\delta V'_X(y_0,x_0) = \delta S(x_0,y_0) \cdot V_W(x_0)$$

Example: Single-Tuple Update to S

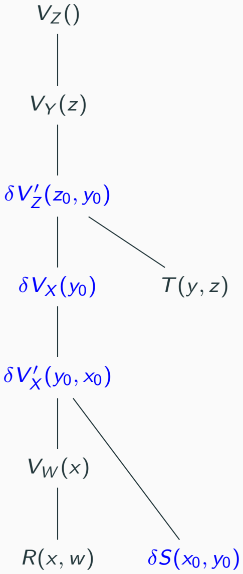


Single-tuple update to S

$$\delta V'_X(y_0, x_0) = \delta S(x_0, y_0) \cdot V_W(x_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

Example: Single-Tuple Update to S



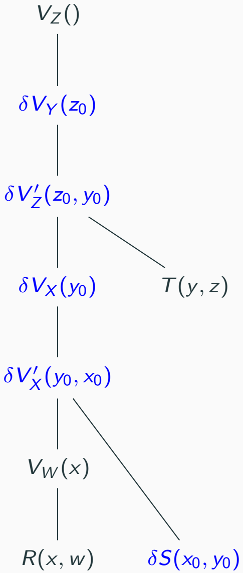
Single-tuple update to S

$$\delta V'_X(y_0, x_0) = \delta S(x_0, y_0) \cdot V_W(x_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

Example: Single-Tuple Update to S



$V_Z()$

$\delta V_Y(z_0)$

$\delta V'_Z(z_0, y_0)$

$\delta V_X(y_0)$ $T(y, z)$

$\delta V'_X(y_0, x_0)$

$V_W(x)$

$R(x, w)$ $\delta S(x_0, y_0)$

Single-tuple update to S

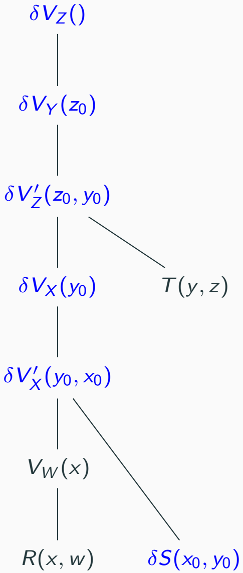
$$\delta V'_X(y_0, x_0) = \delta S(x_0, y_0) \cdot V_W(x_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

Example: Single-Tuple Update to S



$\delta V_Z()$

$\delta V_Y(z_0)$

$\delta V'_Z(z_0, y_0)$

$\delta V_X(y_0)$ $T(y, z)$

$\delta V'_X(y_0, x_0)$

$V_W(x)$

$R(x, w)$ $\delta S(x_0, y_0)$

Single-tuple update to S

$$\delta V'_X(y_0, x_0) = \delta S(x_0, y_0) \cdot V_W(x_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

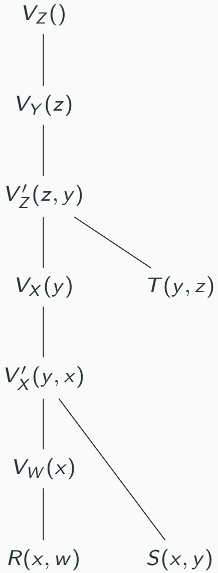
$$\delta V_Z() = \sum_{z_0} \delta V_Y(z_0) = \delta V_Y(z_0)$$

For each updated view/relation A: $A := A + \delta A$

Each view update takes $O(1)$ time

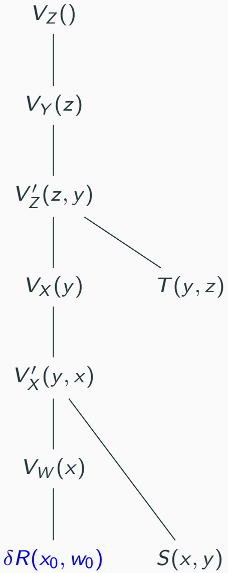
Example: Single-Tuple Update to R

Single-tuple update to R



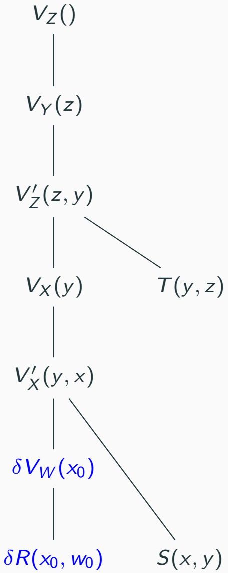
Example: Single-Tuple Update to R

Single-tuple update to R



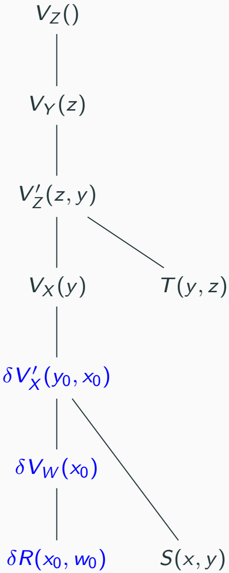
Example: Single-Tuple Update to R

Single-tuple update to R



$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0, w_0) = \delta R(x_0, w_0)$$

Example: Single-Tuple Update to R

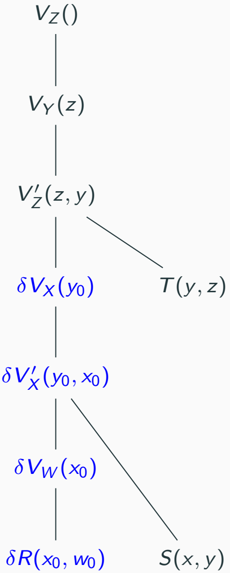


Single-tuple update to R

$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0,w_0) = \delta R(x_0,w_0)$$

$$\delta V'_X(y_0,x_0) : \delta V_W(x_0)\cdot S(x_0,y) \overset{x\to y}{=} \delta V_W(x_0)\cdot S(x_0,y_0)$$

Example: Single-Tuple Update to R



Single-tuple update to R

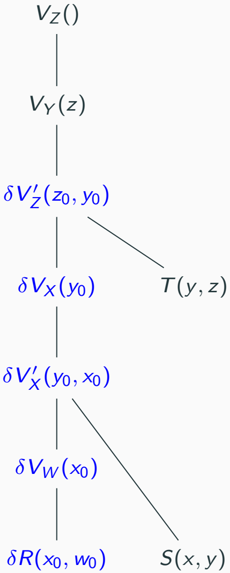
$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0, w_0) = \delta R(x_0, w_0)$$

$$\delta V'_X(y_0, x_0) : \delta V_W(x_0) \cdot S(x_0, y) \overset{x \rightarrow y}{=} \delta V_W(x_0) \cdot S(x_0, y_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

Example: Single-Tuple Update to R

Single-tuple update to R



$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0, w_0) = \delta R(x_0, w_0)$$

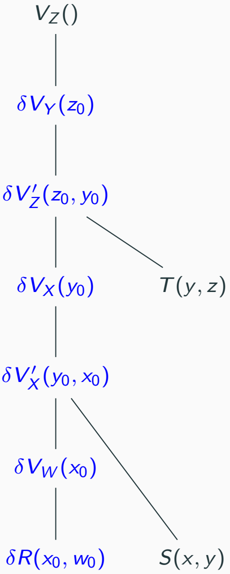
$$\delta V'_X(y_0, x_0) : \delta V_W(x_0) \cdot S(x_0, y) \overset{x \to y}{=} \delta V_W(x_0) \cdot S(x_0, y_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

Example: Single-Tuple Update to R

Single-tuple update to R



$V_Z()$

$\delta V_Y(z_0)$

$\delta V'_Z(z_0, y_0)$

$\delta V_X(y_0)$

$T(y, z)$

$\delta V'_X(y_0, x_0)$

$\delta V_W(x_0)$

$\delta R(x_0, w_0)$

$S(x, y)$

$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0, w_0) = \delta R(x_0, w_0)$$

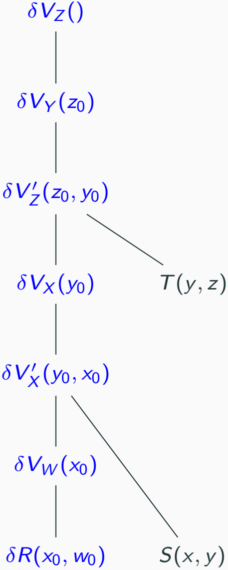
$$\delta V'_X(y_0, x_0) : \delta V_W(x_0) \cdot S(x_0, y) \overset{x \to y}{=} \delta V_W(x_0) \cdot S(x_0, y_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

Example: Single-Tuple Update to R



$\delta V_Z()$

$\delta V_Y(z_0)$

$\delta V'_Z(z_0, y_0)$

$\delta V_X(y_0)$

$T(y, z)$

$\delta V'_X(y_0, x_0)$

$\delta V_W(x_0)$

$\delta R(x_0, w_0)$

$S(x, y)$

Single-tuple update to R

$$\delta V_W(x_0) = \sum_{w_0} \delta R(x_0, w_0) = \delta R(x_0, w_0)$$

$$\delta V'_X(y_0, x_0) : \delta V_W(x_0) \cdot S(x_0, y) \overset{x \to y}{=} \delta V_W(x_0) \cdot S(x_0, y_0)$$

$$\delta V_X(y_0) = \sum_{x_0} \delta V'_X(y_0, x_0) = \delta V'_X(y_0, x_0)$$

$$\delta V'_Z(z_0, y_0) : \delta V'_X(y_0) \cdot T(y_0, z) \overset{y \to z}{=} \delta V'_X(y_0) \cdot T(y_0, z_0)$$

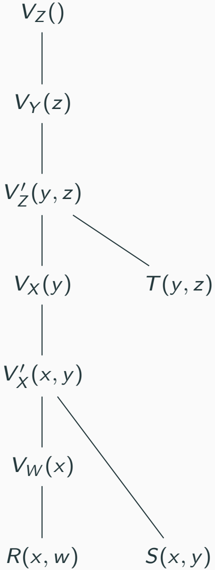
$$\delta V_Y(z_0) = \sum_{y_0} \delta V'_Z(z_0, y_0) = \delta V'_Z(z_0, y_0)$$

$$\delta V_Z() = \sum_{z_0} \delta V_Y(z_0) = \delta V_Y(z_0)$$

For each updated view/relation A: $A := A + \delta A$

Each view update takes $O(1)$ time

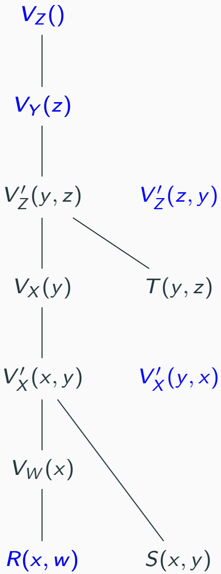
Example: Enumeration of Query Answers



Enumeration for $Q(z,y,x,w)$ with constant delay

- Top-down in the view tree
- Views calibrated for variables underneath
- Guaranteed to get matching tuples in views below

Example: Enumeration of Query Answers



Enumeration for $Q(z, y, x, w)$ **with constant delay**

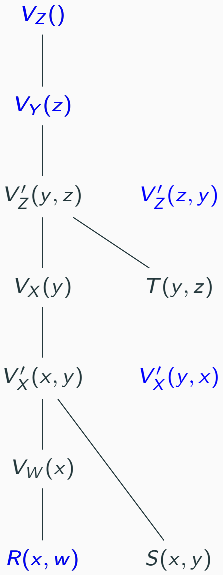
- Top-down in the view tree
- Views calibrated for variables underneath
- Guaranteed to get matching tuples in views below

Enumeration from the join:

$$\mathbf{1}_{V_Z} \cdot \mathbf{1}_{V_Y(z)} \cdot \mathbf{1}_{V'_Z(z,y)} \cdot \mathbf{1}_{V'_X(y,x)} \cdot T(z,y) \cdot S(x,y) \cdot R(x,w)$$

with variable order: $Z - Y - X - W$

Example: Enumeration of Query Answers



Enumeration for $Q(z, y, x, w)$ with constant delay

- Top-down in the view tree
- Views calibrated for variables underneath
- Guaranteed to get matching tuples in views below

Enumeration from the join:

$$\mathbf{1}_{V_Z} \cdot \mathbf{1}_{V_Y(z)} \cdot \mathbf{1}_{V'_Z(z,y)} \cdot \mathbf{1}_{V'_X(y,x)} \cdot T(z,y) \cdot S(x,y) \cdot R(x,w)$$

with variable order: $Z - Y - X - W$

- Is $V_Z()$ empty? If yes, stop.
- Iterate over z's in $V_Y(z)$
- For each z, iterate over y's in index $V'_Z(z,y)$
- For each y, iterate over x's in index $V'_X(y,x)$
- Iterate over $T(z,y)$, $S(x,y)$, $R(x,w)$

Open Questions

- Can we achieve worst-case optimality per single-tuple update beyond the *q*-hierarchical queries?

Open Questions

- Can we achieve worst-case optimality per single-tuple update beyond the *q*-hierarchical queries?

- In practice, *average* constant time might be enough.

  Which queries admit average constant time for single-tuple updates?

Open Questions

- Can we achieve worst-case optimality per single-tuple update beyond the q-hierarchical queries?

- In practice, *average* constant time might be enough.

  Which queries admit average constant time for single-tuple updates?

- What is the complexity trade-off between update time and enumeration delay if we drop:

  the "q" property?

  the hierarchical property?

References i

[VLDB 2004] Nilesh N. Dalvi, Dan Suciu. *Efficient Query Evaluation on Probabilistic Databases.*

[ICDE 2009] Dan Olteanu, Jiewen Huang, Christoph Koch. *SPROUT: Lazy vs. Eager Query Plans for Tuple-Independent Probabilistic Databases.*

[ICDT 2012] Dan Olteanu, Jakub Zavodny. *Factorised representations of query results: size bounds and readability.*

[PODS 2017] Christoph Berkholz, Jens Keppeler, Nicole Schweikardt. *Answering Conjunctive Queries under Updates.*

[SIGMOD 2018] Milos Nikolic, Dan Olteanu. *Incremental View Maintenance with Triple Lock Factorization Benefits.*

References ii

[ICDT 2023] Ahmet Kara, Milos Nikolic, Dan Olteanu, Haozhe Zhang. *Conjunctive Queries with Free Access Patterns Under Updates.*

[VLDBJ 2023] Ahmet Kara, Milos Nikolic, Dan Olteanu, Haozhe Zhang. *F-IVM: Analytics over Relational Databases under Updates.* (To appear)

[UZH 2023] Johann Schwabe. *CaVieR: CAscading VIEw tRees.* MSc thesis, University of Zurich

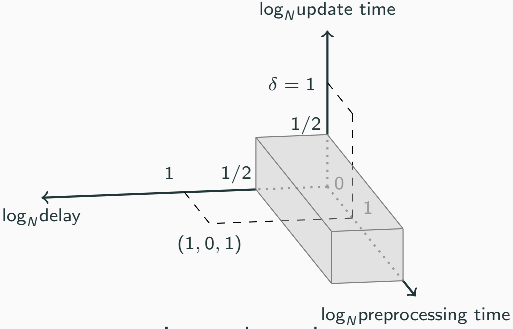
3. Beyond “Q”

Simplest Hierarchical Query without “Q” Property

$$Q(a) = \sum_b R(a, b) \cdot S(b)$$

Simplest Hierarchical Query without “Q” Property

$$Q(a) = \sum_b R(a,b) \cdot S(b)$$



Lower bound

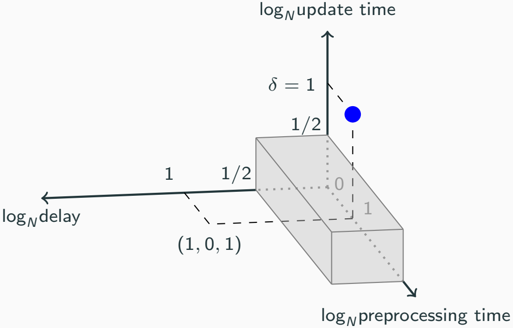
For this query, there is no algorithm that admits

preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{1/2-\gamma})$	$\mathcal{O}(N^{1/2-\gamma})$

for any $\gamma > 0$, unless the OMv Conjecture fails [PODS 2017]

Simplest Hierarchical Query without “Q” Property

$$Q(a) = \sum_b R(a, b) \cdot S(b)$$

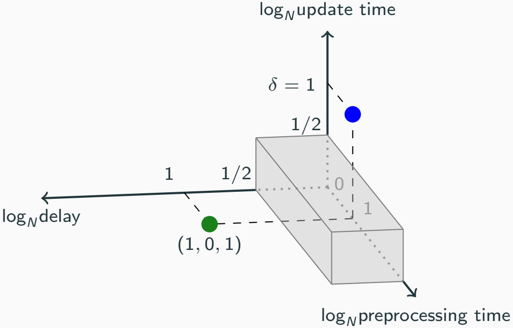


Known approach: Eager update, quick enumeration

- Preprocessing: Materialize the result.
- Upon update: Maintain the materialized result.
- Enumeration: Enumerate from materialized result.

Simplest Hierarchical Query without “Q” Property

$$Q(a) = \sum_b R(a, b) \cdot S(b)$$

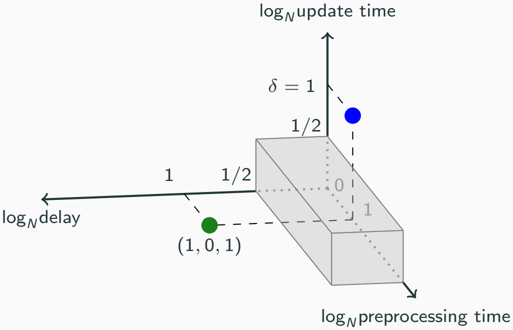


Known approach: Lazy update, heavy enumeration

- Preprocessing: Eliminate dangling tuples
- Upon update: Update only base relations
- Enumeration: Eliminate dangling tuples and enumerate from R

Simplest Hierarchical Query without “Q” Property

$$Q(a) = \sum_b R(a,b) \cdot S(b)$$

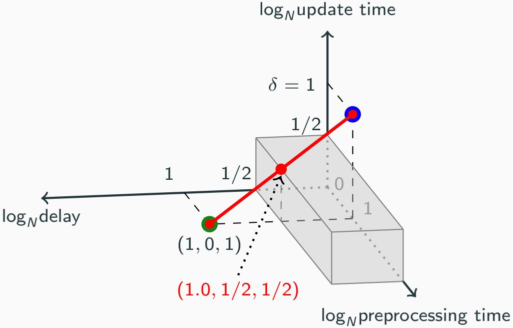


Yet, there is an algorithm that admits

sub-linear update time and sub-linear enumeration delay

Simplest Hierarchical Query without "Q" Property

$$Q(a) = \sum_b R(a, b) \cdot S(b)$$



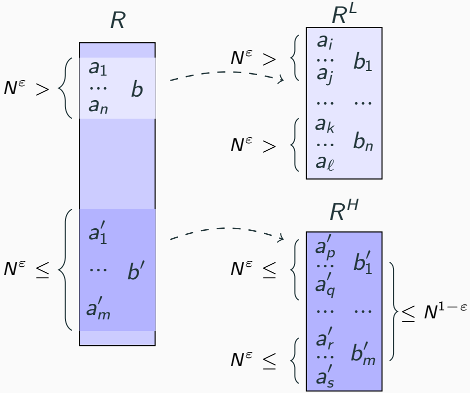
Weak Pareto optimality

Relation Partitioning

$$Q(a) = \sum_b R(a,b) \cdot S(b)$$

Partition R based on the values b into

- a light part $R^L = \{(a,b) \in R \mid |\sigma_{B=b} R| < N^\varepsilon\}$
- a heavy part $R^H = R - R^L$

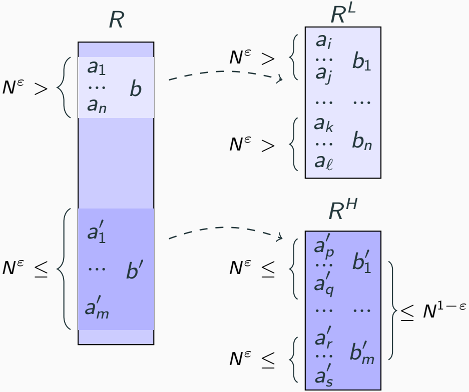


Relation Partitioning

$$Q(a) = \sum_b R(a,b) \cdot S(b)$$

Partition R based on the values b into

- a light part $R^L = \{(a,b) \in R \mid |\sigma_{B=b}R| < N^\varepsilon\}$
- a heavy part $R^H = R - R^L$



$$Q(a) = Q_L(a) + Q_H(a)$$
$$Q_L(a) = \sum_b R^L(a,b) \cdot S(b)$$
$$Q_H(a) = \sum_b R^H(a,b) \cdot S(b)$$

Light Case

$$Q_L(a) = \sum_b R^L(a, b) \cdot S(b)$$

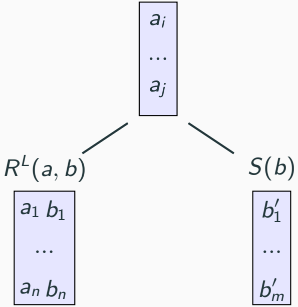
Materialize the result

Light Case

$$Q_L(a) = \sum_b R^L(a,b) \cdot S(b)$$

Materialize the result

$$Q_L(a) = \sum_b R^L(a,b) \cdot S(b)$$



a_i

$\dots$

a_j

$R^L(a,b)$

$a_1\ b_1$

$\dots$

$a_n\ b_n$

$S(b)$

b'_1

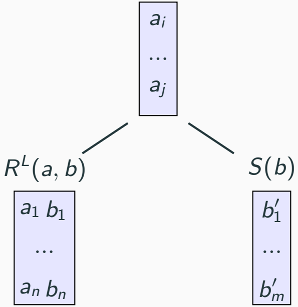
$\dots$

b'_m

Preprocessing in the Light Case

$$Q_L(a) = \sum_b R^L(a, b) \cdot S(b)$$

$$Q_L(a) = \sum_b R^L(a, b) \cdot S(b)$$



a_i

...

a_j

$R^L(a, b)$

$a_1 \; b_1$

...

$a_n \; b_n$

$S(b)$

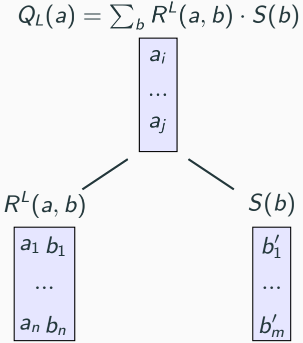
b'_1

...

b'_m

- Q_L can be computed in time $\mathcal{O}(N)$

Enumeration in the Light Case



$Q_L(a) = \sum_b R^L(a,b) \cdot S(b)$

a_i

$\dots$

a_j

$R^L(a,b)$

$a_1\ b_1$

$\dots$

$a_n\ b_n$

$S(b)$

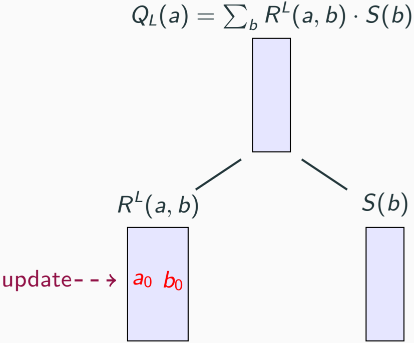
b'_1

$\dots$

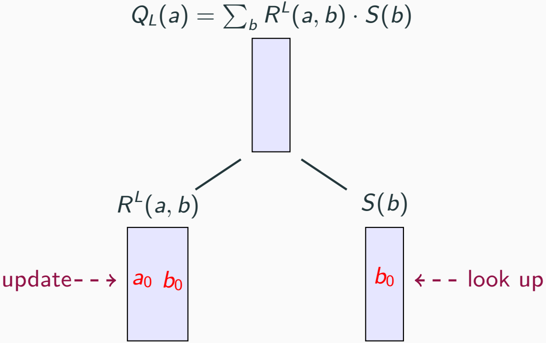
b'_m

- Q_L allows constant-time lookups and constant-delay enumeration

Updates in the Light Case



Updates in the Light Case



$Q_L(a) = \sum_b R^L(a,b) \cdot S(b)$

$R^L(a,b)$

$S(b)$

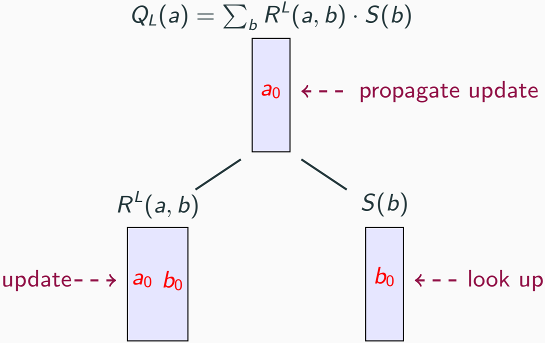
update

a_0 b_0

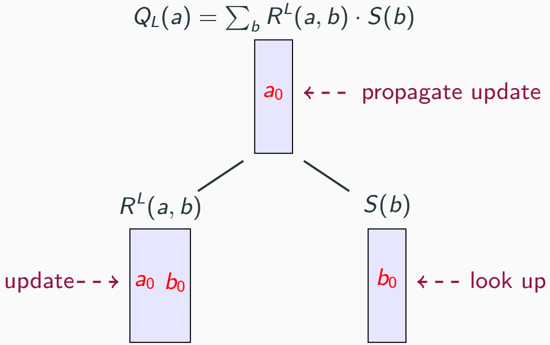
b_0

look up

Updates in the Light Case

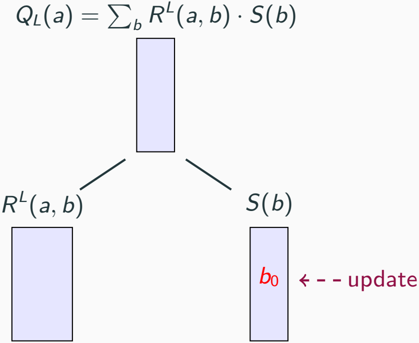


Updates in the Light Case



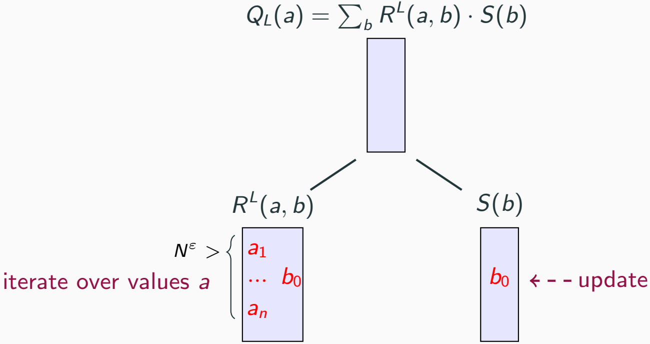
- Updates to R^L: $\mathcal{O}(1)$

Updates in the Light Case



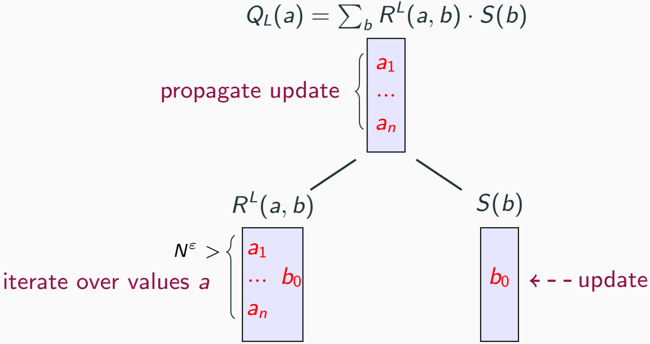
- Updates to R^L: $\mathcal{O}(1)$

Updates in the Light Case



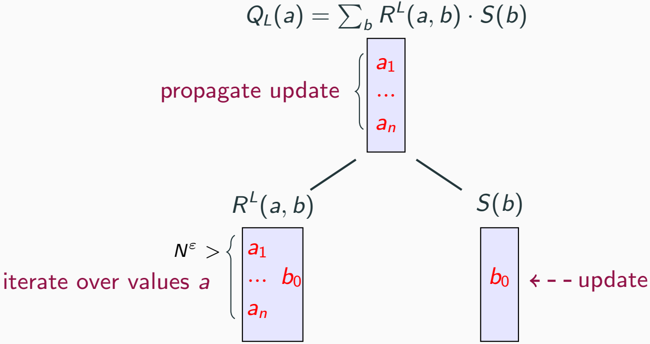
- Updates to R^L: $\mathcal{O}(1)$

Updates in the Light Case



- Updates to R^L: $\mathcal{O}(1)$

Updates in the Light Case



- Updates to R^L: $\mathcal{O}(1)$
- Updates to S: $\mathcal{O}(N^\varepsilon)$

Heavy Case

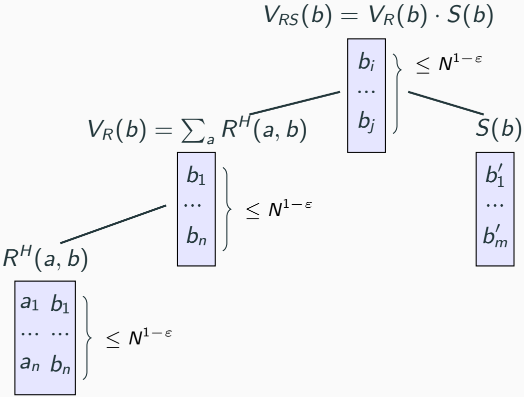
$$Q_H(a) = \sum_b R^H(a,b) \cdot S(b)$$

Materialize the b values in the join result

Heavy Case

$$Q_H(a) = \sum_b R^H(a,b) \cdot S(b)$$

Materialize the b values in the join result



$V_{RS}(b) = V_R(b) \cdot S(b)$

b_i … b_j $\Big\} \leq N^{1-\varepsilon}$

$V_R(b) = \sum_a R^H(a,b)$

b_1 … b_n $\Big\} \leq N^{1-\varepsilon}$

$S(b)$

b'_1 … b'_m

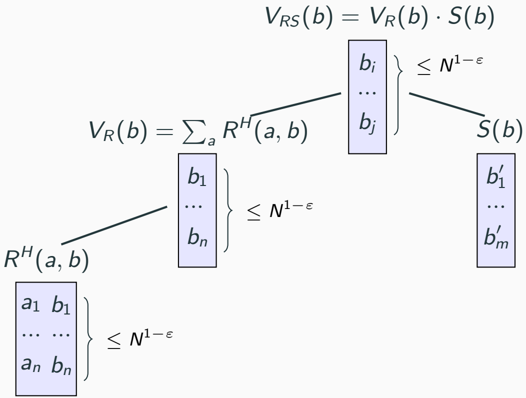
$R^H(a,b)$

a_1 b_1 … … a_n b_n $\Big\} \leq N^{1-\varepsilon}$

Preprocessing in the Heavy Case

$$Q_H(a) = \sum_b R^H(a, b) \cdot S(b)$$

Materialize the b values in the join result



$V_{RS}(b) = V_R(b) \cdot S(b)$

b_i $\cdots$ b_j $\leq N^{1-\varepsilon}$

$V_R(b) = \sum_a R^H(a, b)$

b_1 $\cdots$ b_n $\leq N^{1-\varepsilon}$

$S(b)$

b'_1 $\cdots$ b'_m

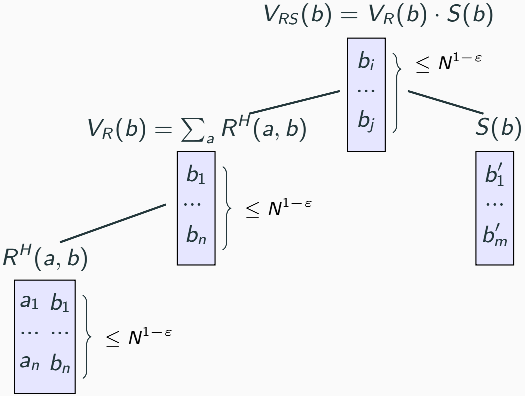
$R^H(a, b)$

a_1 b_1 $\cdots$ $\cdots$ a_n b_n $\leq N^{1-\varepsilon}$

- V_{RS} can be computed in time $\mathcal{O}(N^{1-\varepsilon})$ and has at most $N^{1-\varepsilon}$ values

Enumeration in the Heavy Case

$$Q_H(a) = \sum_b R^H(a,b) \cdot S(b)$$



- V_{RS} contains at most $N^{1-\varepsilon}$ values b
- For each value b in V_{RS}, the values a in R^H paired with b admit constant enumeration delay

Enumeration of Distinct Tuples from Union

- $V_{RS}(b)$ contains at most $N^{1-\varepsilon}$ values
- For each value b in V_{RS}, the values a in R^H paired with b admit constant enumeration delay
- Yet: For two distinct b_1 and b_2, the sets of values a in $R^H(a, b_1)$ and $R^H(a, b_2)$ may not be disjoint

  $\Longrightarrow$ Enumerating all the values a in $R^H(a, b_1)$ and $R^H(a, b_2)$ can lead to duplicates

Enumeration of Distinct Tuples from Union

- $V_{RS}(b)$ contains at most $N^{1-\varepsilon}$ values
- For each value b in V_{RS}, the values a in R^H paired with b admit constant enumeration delay
- Yet: For two distinct b_1 and b_2, the sets of values a in $R^H(a, b_1)$ and $R^H(a, b_2)$ may not be disjoint

  $\Longrightarrow$ Enumerating all the values a in $R^H(a, b_1)$ and $R^H(a, b_2)$ can lead to duplicates

Union Algorithm [CSL 2011]

- The distinct values a can be enumerated with $\mathcal{O}(N^{1-\varepsilon})$ delay

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay

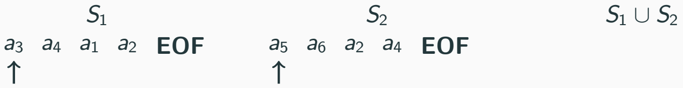
S_1	S_2	$S_1 \cup S_2$
a_3 a_4 a_1 a_2 **EOF**	a_5 a_6 a_2 a_4 **EOF**	

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

a_3 a_4 a_1 a_2 **EOF**

$\uparrow$

S_2

a_5 a_6 a_2 a_4 **EOF**

$\uparrow$

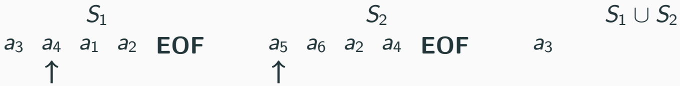
$S_1 \cup S_2$

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

a_3 a_4 a_1 a_2 **EOF**

S_2

a_5 a_6 a_2 a_4 **EOF**

$S_1 \cup S_2$

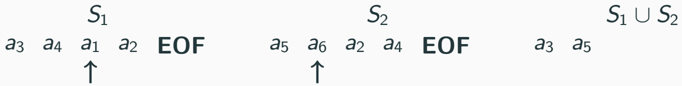
a_3

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

$a_3 \quad a_4 \quad a_1 \quad a_2 \quad$ **EOF**

S_2

$a_5 \quad a_6 \quad a_2 \quad a_4 \quad$ **EOF**

$S_1 \cup S_2$

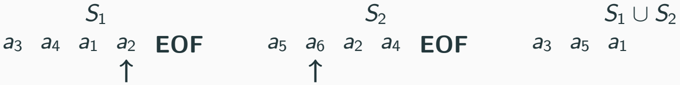
$a_3 \quad a_5$

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

a_3 a_4 a_1 a_2 **EOF**

$\uparrow$

S_2

a_5 a_6 a_2 a_4 **EOF**

$\uparrow$

$S_1 \cup S_2$

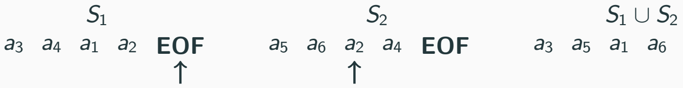
a_3 a_5 a_1

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

$a_3 \quad a_4 \quad a_1 \quad a_2$ **EOF**

$\uparrow$

S_2

$a_5 \quad a_6 \quad a_2 \quad a_4$ **EOF**

$\uparrow$

$S_1 \cup S_2$

$a_3 \quad a_5 \quad a_1 \quad a_6$

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay

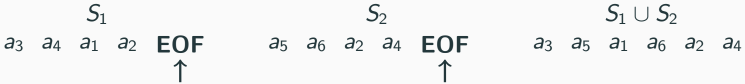
S_1	S_2	$S_1 \cup S_2$
$a_3 \quad a_4 \quad a_1 \quad a_2 \quad \underset{\uparrow}{\textbf{EOF}}$	$a_5 \quad a_6 \quad a_2 \quad \underset{\uparrow}{a_4} \quad \textbf{EOF}$	$a_3 \quad a_5 \quad a_1 \quad a_6 \quad a_2$

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay



S_1

$a_3 \quad a_4 \quad a_1 \quad a_2 \quad \textbf{EOF}$

$\uparrow$

S_2

$a_5 \quad a_6 \quad a_2 \quad a_4 \quad \textbf{EOF}$

$\uparrow$

$S_1 \cup S_2$

$a_3 \quad a_5 \quad a_1 \quad a_6 \quad a_2 \quad a_4$

The Union Algorithm: Example

Enumeration of the distinct tuples in the union of two sets

- Both sets allow lookup time ℓ and enumeration delay d

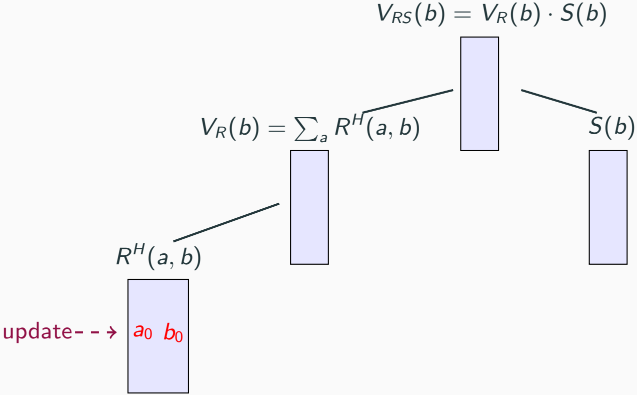
$\Longrightarrow$ The union of the sets can be enumerated with $\mathcal{O}(\ell + d)$ delay

S_1	S_2	$S_1 \cup S_2$
a_3 a_4 a_1 a_2 **EOF**	a_5 a_6 a_2 a_4 **EOF**	a_3 a_5 a_1 a_6 a_2 a_4
$\uparrow$		

Generalization: Enumeration from the union of n sets

- Each set allows lookup time ℓ and enumeration delay d
- The union of the sets can be enumerated with $\mathcal{O}(n(\ell + d))$ delay

Updates in the Heavy Case



$V_{RS}(b) = V_R(b) \cdot S(b)$

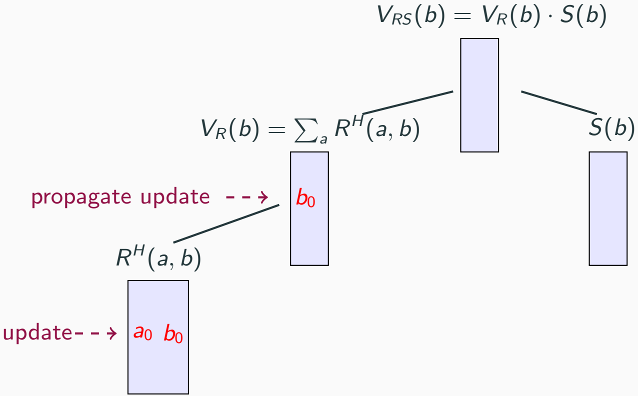
$V_R(b) = \sum_a R^H(a, b)$

$S(b)$

$R^H(a, b)$

update $\dashrightarrow$ a_0 b_0

Updates in the Heavy Case



$V_{RS}(b) = V_R(b) \cdot S(b)$

$V_R(b) = \sum_a R^H(a, b)$

$S(b)$

propagate update

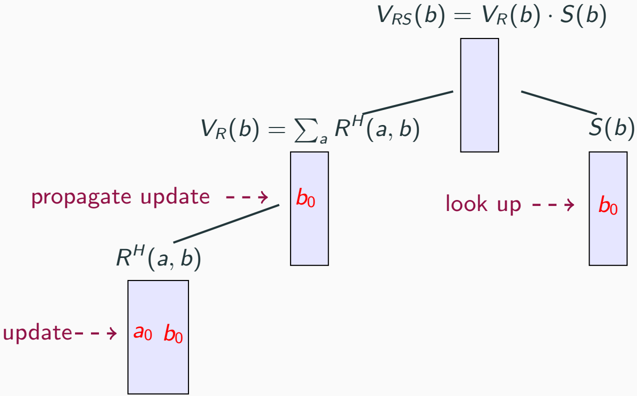
b_0

$R^H(a, b)$

update

a_0 b_0

Updates in the Heavy Case



$V_{RS}(b) = V_R(b) \cdot S(b)$

$V_R(b) = \sum_a R^H(a,b)$

$S(b)$

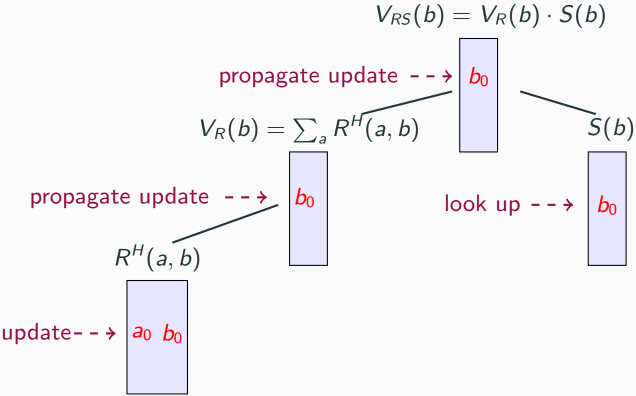
propagate update - - → b_0

look up - - → b_0

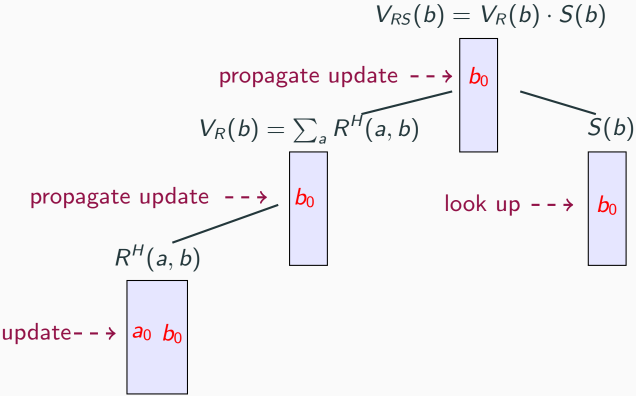
$R^H(a,b)$

update- - → a_0 b_0

Updates in the Heavy Case

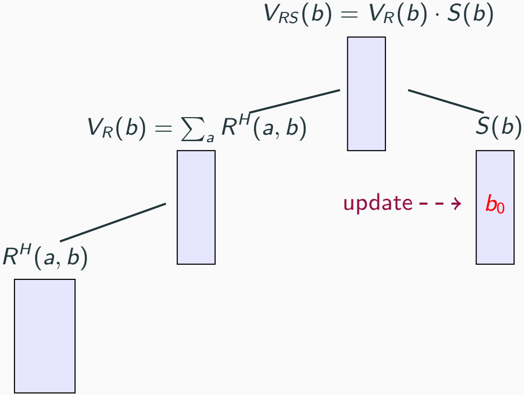


Updates in the Heavy Case



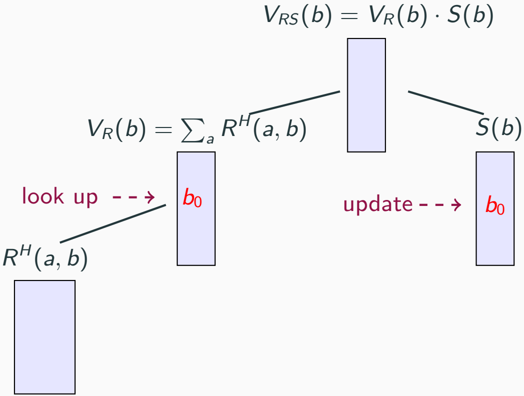
- Updates to R^H: $\mathcal{O}(1)$

Updates in the Heavy Case



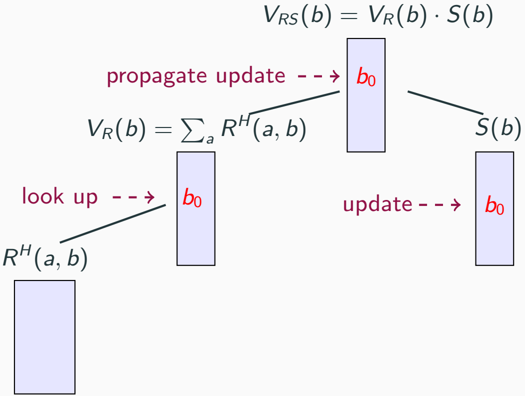
- Updates to R^H: $\mathcal{O}(1)$

Updates in the Heavy Case



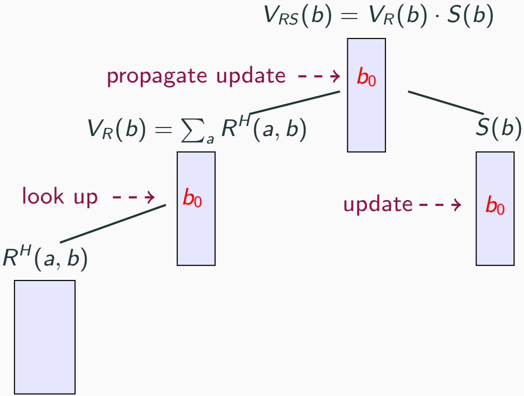
- Updates to R^H: $\mathcal{O}(1)$

Updates in the Heavy Case



- Updates to R^H: $\mathcal{O}(1)$

Updates in the Heavy Case



- Updates to R^H: $\mathcal{O}(1)$
- Updates to S: $\mathcal{O}(1)$

Summing Up

$$Q(a) = R(a, b) \cdot S(b)$$

Preprocessing Time

light case	heavy case	overall
$\mathcal{O}(N)$	$\mathcal{O}(N^{1-\varepsilon})$	$\mathcal{O}(N)$

Enumeration Delay

light case	heavy case	overall
$\mathcal{O}(1)$	$\mathcal{O}(N^{1-\varepsilon})$	$\mathcal{O}(N^{1-\varepsilon})$

Update Time

light case	heavy case	overall
$\mathcal{O}(N^{\varepsilon})$	$\mathcal{O}(1)$	$\mathcal{O}(N^{\varepsilon})$

Are there more queries

with the same

weak Pareto optimality

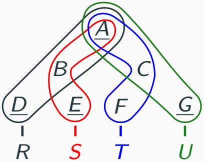
as our previous example?

δ_1-Hierarchical Queries

- For any bound variable X and any atom α of X, there is at most one other atom β so that all free variables dominated by X are covered by α and β together
- The query is hierarchical and not q-hierarchical

δ_1-hierarchical

$$Q(a,d,e,g) = R(a,b,d) \cdot S(a,b,e) \cdot T(a,c,f) \cdot U(a,c,g)$$

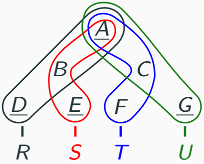


δ_1-Hierarchical Queries

- For any bound variable X and any atom α of X, there is at most one other atom β so that all free variables dominated by X are covered by α and β together
- The query is hierarchical and not q-hierarchical

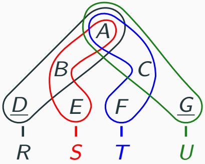
δ_1-hierarchical

$Q(a, d, e, g) = R(a, b, d) \cdot S(a, b, e) \cdot$
$T(a, c, f) \cdot U(a, c, g)$



hierarchical but not δ_1-hierarchical

$Q(d, g) = R(a, b, d) \cdot S(a, b, e) \cdot$
$T(A, C, F) \cdot U(a, c, g)$



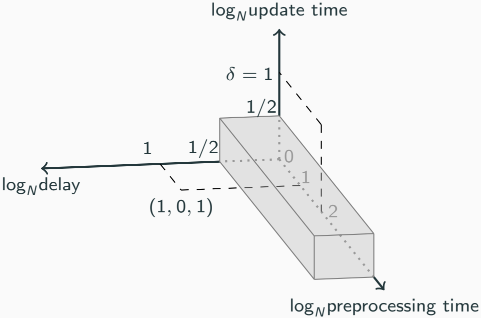
Optimality for δ_1-Hierarchical Queries

- For any δ_1-hierarchical query, there is no algorithm that admits

preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{1/2-\gamma})$	$\mathcal{O}(N^{1/2-\gamma})$

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

(*) Online Matrix-Vector Multiplication cannot be solved in sub-cubic time



Optimality for δ_1-Hierarchical Queries

- For any δ_1-hierarchical query, there is no algorithm that admits

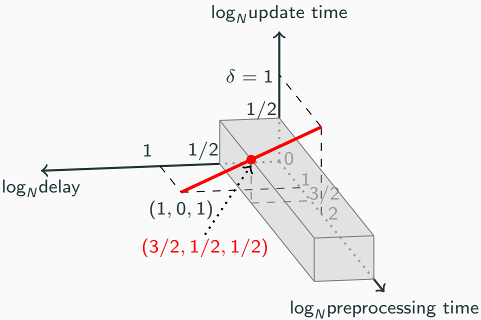
preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{1/2-\gamma})$	$\mathcal{O}(N^{1/2-\gamma})$

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

- Any δ_1-hierarchical query can be maintained with

preprocessing time	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



Optimality for δ_1-Hierarchical Queries

- For any δ_1-hierarchical query, there is no algorithm that admits

preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{1/2-\gamma})$	$\mathcal{O}(N^{1/2-\gamma})$

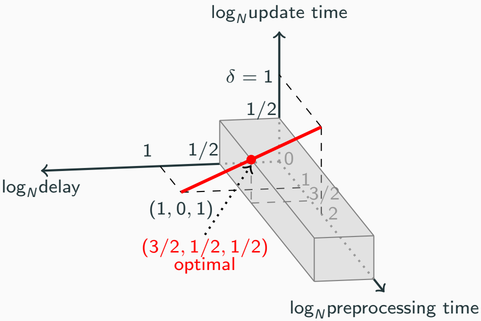
  for any $\gamma > 0$, unless the OMv Conjecture (*) fails

- Any δ_1-hierarchical query can be maintained with

preprocessing time	update time	enumeration delay
$\mathcal{O}(N^{1+\varepsilon})$	$\mathcal{O}(N^{\varepsilon})$	$\mathcal{O}(N^{1-\varepsilon})$

$\Longrightarrow$ For $\varepsilon = 1/2$, this is weakly Pareto optimal, unless OMv Conjecture fails

(*) Online Matrix-Vector Multiplication cannot be solved in sub-cubic time



Trade-Offs Beyond δ_1-Hierarchical

We can define syntactically classes of δ_i-hierarchical queries ($i \in \mathbb{N}$)

- with $\mathcal{O}(N^{i\varepsilon})$ update time and $\mathcal{O}(N^{1-\varepsilon})$ enumeration delay.

- δ_0-hierarchical $=$ Q-hierarchical

[LMCS 2023]

Trade-Offs Beyond δ_i-Hierarchical

Any hierarchical query can be maintained with

preprocessing time	update time	enumeration delay
$\mathcal{O}(N^{1+(\mathrm{w}-1)\varepsilon})$	$\mathcal{O}(N^{\delta\varepsilon})$	$\mathcal{O}(N^{1-\varepsilon})$

where

- static width w = the fractional hypertree width for CQs

- dynamic width $\delta =^* \max_{\text{delta queries}}$ static width

[PODS 2020]

Trade-Offs Beyond δ_i-Hierarchical

Any hierarchical query can be maintained with

preprocessing time	update time	enumeration delay
$\mathcal{O}(N^{1+(\mathrm{w}-1)\varepsilon})$	$\mathcal{O}(N^{\delta\varepsilon})$	$\mathcal{O}(N^{1-\varepsilon})$

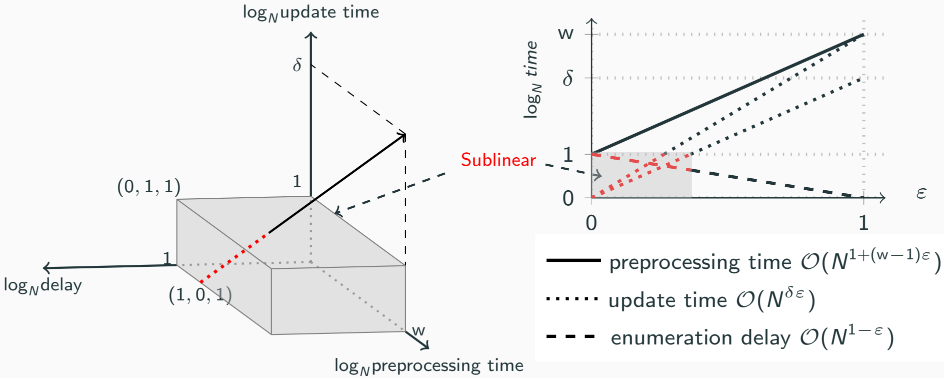
where

- static width w = the fractional hypertree width for CQs
- dynamic width $\delta =^{*} \max_{\text{delta queries}}$ static width

[PODS 2020]

Open question: Lower bounds for hierarchical queries

Sublinear Update Time and Delay



Hierarchical queries admit sublinear update time and enumeration delay

Trade-Offs Beyond Hierarchical

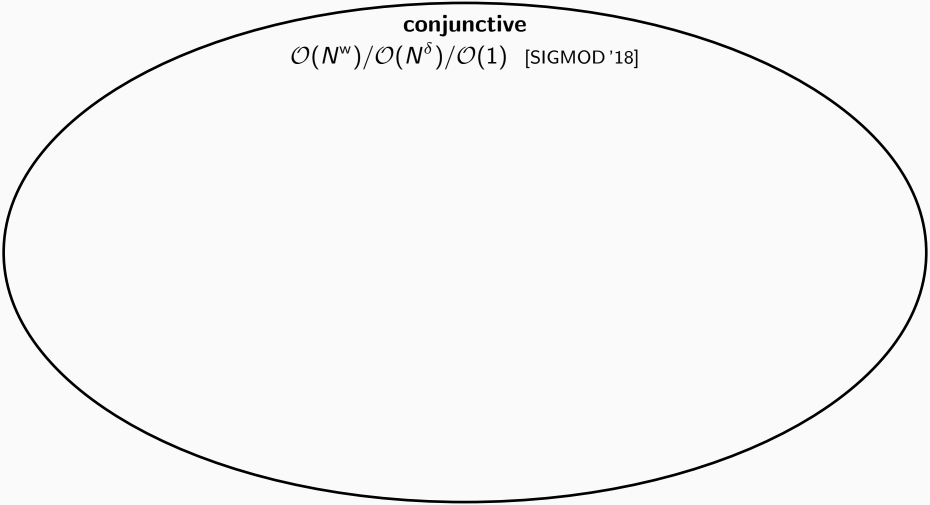
- No nice closed-form expression for complexities seem possible

- For some α-acyclic queries, trade-offs seem not possible

- First steps already made for α-acyclic queries [CSL 2023]

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay

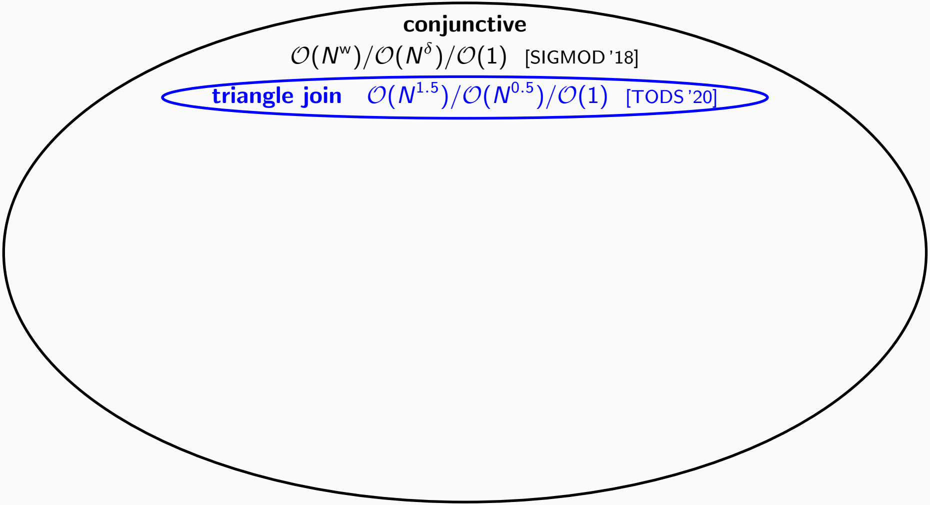


conjunctive

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

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay



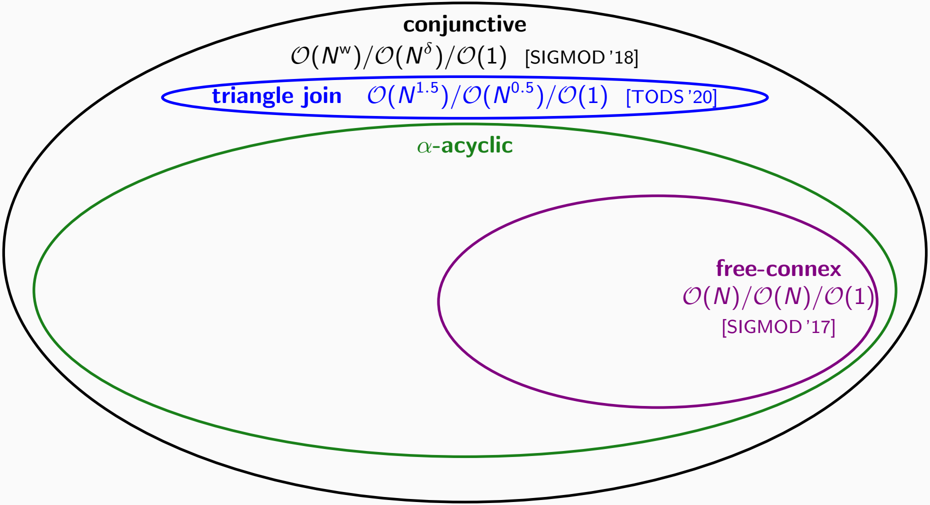
conjunctive

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

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

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay



conjunctive

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

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

α**-acyclic**

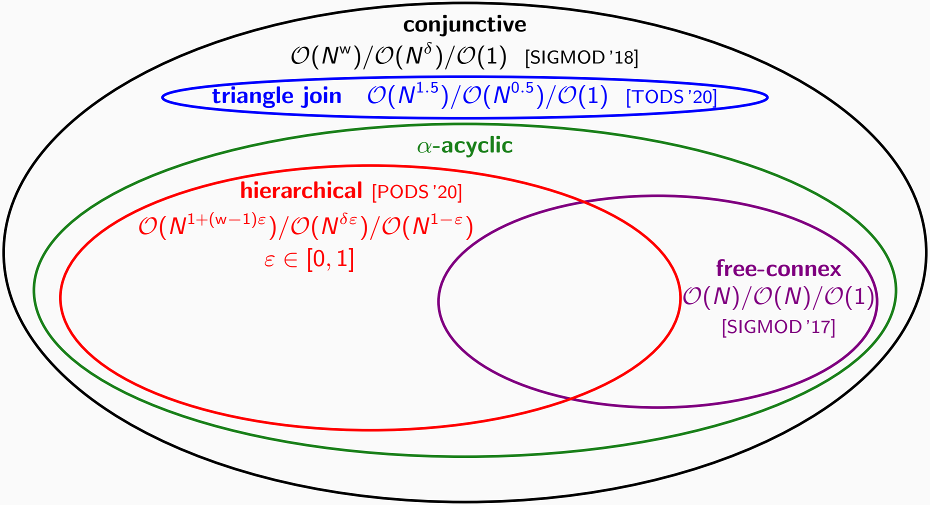
free-connex

$\mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1)$

[SIGMOD '17]

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay



conjunctive

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

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

α-acyclic

hierarchical [PODS '20]

$\mathcal{O}(N^{1+(\mathsf{w}-1)\varepsilon})/\mathcal{O}(N^{\delta\varepsilon})/\mathcal{O}(N^{1-\varepsilon})$

$\varepsilon \in [0,1]$

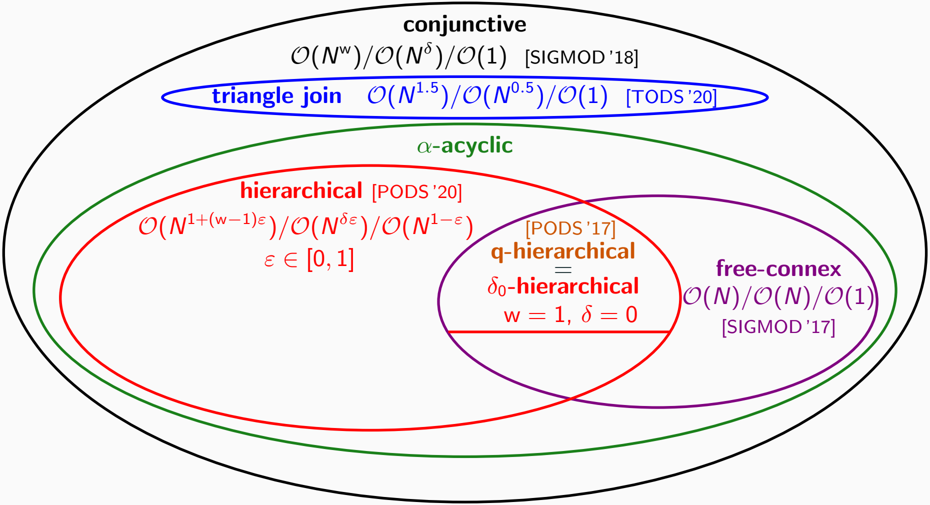
free-connex

$\mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1)$

[SIGMOD '17]

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay



conjunctive

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

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

α-acyclic

hierarchical [PODS'20]

$\mathcal{O}(N^{1+(\mathsf{w}-1)\varepsilon})/\mathcal{O}(N^{\delta\varepsilon})/\mathcal{O}(N^{1-\varepsilon})$

$\varepsilon \in [0,1]$

[PODS'17]

q-hierarchical

$=$

δ_0-hierarchical

$\mathsf{w} = 1,\ \delta = 0$

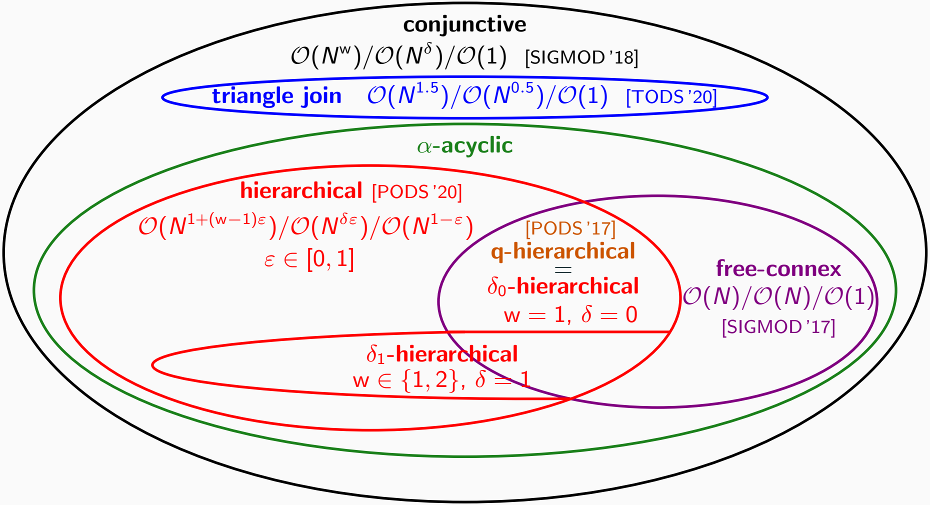
free-connex

$\mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1)$

[SIGMOD'17]

IVM Landscape (Partial)

Preprocessing time/Update time/Enumeration delay



conjunctive

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

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

α-acyclic

hierarchical [PODS '20]

$\mathcal{O}(N^{1+(\mathrm{w}-1)\varepsilon})/\mathcal{O}(N^{\delta\varepsilon})/\mathcal{O}(N^{1-\varepsilon})$

$\varepsilon \in [0,1]$

[PODS '17]

q-hierarchical

$=$

δ_0-hierarchical

$\mathrm{w} = 1,\ \delta = 0$

δ_1-hierarchical

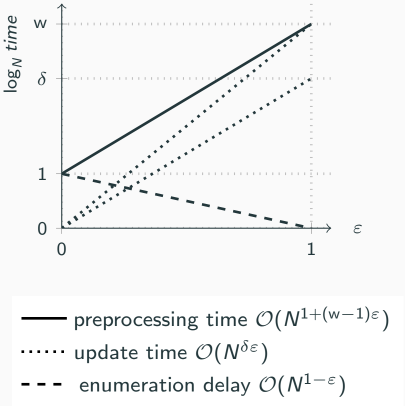
$\mathrm{w} \in \{1,2\},\ \delta = 1$

free-connex

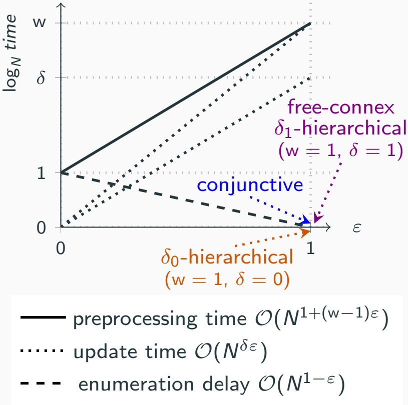
$\mathcal{O}(N)/\mathcal{O}(N)/\mathcal{O}(1)$

[SIGMOD '17]

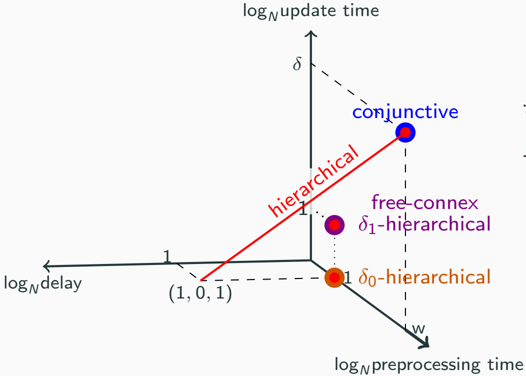
Recovery of Prior Results

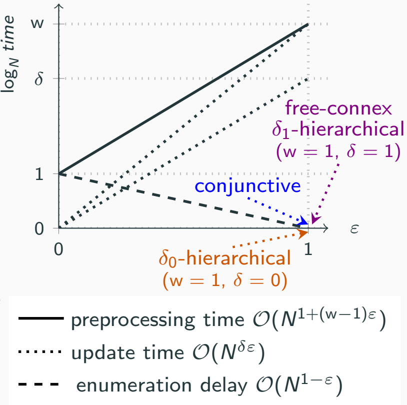


Recovery of Prior Results



Recovery of Prior Results





References i

[CSL 2011] Arnaud Durand, Yann Strozecki. *Enumeration Complexity of Logical Query Problems with Second-order Variables.* CSL 2011

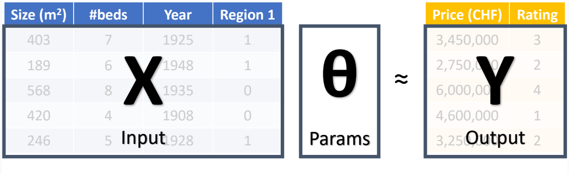
[CSL 2023] Ahmet Kara, Milos Nikolic, Dan Olteanu, Haozhe Zhang. *Evaluation Trade-Offs for Acyclic Conjunctive Queries.*

[LMCS 2023] Ahmet Kara, Milos Nikolic, Dan Olteanu, Haozhe Zhang. *Trade-offs in Static and Dynamic Evaluation of Hierarchical Queries.*

4. Maintaining ML Models over Evolving Relational Data

Maintain Models under Updates

1. Polynomial Regression: Find parameters Θ best satisfying



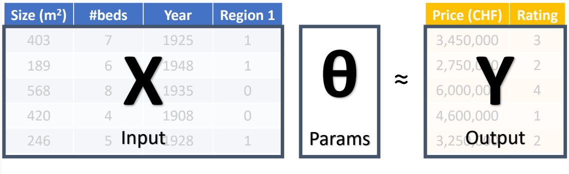
Size (m²)	#beds	Year	Region 1
403	7	1925	1
189	6	1948	1
568	8	1935	0
420	4	1908	0
246	5	1928	1

Price (CHF)	Rating
3,450,000	3
2,750,000	2
6,000,000	4
4,600,000	1
3,250,000	2

- Features **X** and labels **Y** are given by database joins

Maintain Models under Updates

1. Polynomial Regression: Find parameters Θ best satisfying



Size (m²)	#beds	Year	Region 1
403	7	1925	1
189	6	1948	1
568	8	1935	0
420	4	1908	0
246	5	1928	1

Price (CHF)	Rating
3,450,000	3
2,750,000	2
6,000,000	4
4,600,000	1
3,250,000	2

- Features **X** and labels **Y** are given by database joins
- Solved using iterative gradient computation:

$$\mathbf{\Theta}_{i+1} = \mathbf{\Theta}_i - \alpha\, \mathbf{X}^{\mathbf{T}}(\mathbf{X}\, \mathbf{\Theta}_i - \mathbf{Y}) \quad \text{(repeat until convergence)}$$

2. Chow-Liu Trees: based on pairwise mutual information

Approach for both: Maintain the Covariance Matrix $[\mathbf{X}\ \mathbf{Y}]^{\mathbf{T}}\,[\mathbf{X}\ \mathbf{Y}]$

[SIGMOD 2018 & 2020, VLDB J 2023]

Covariance Matrix Defined by Queries

Covariance matrix $[\mathbf{X}\ \mathbf{Y}]^{\mathbf{T}}\,[\mathbf{X}\ \mathbf{Y}]$ can be expressed in SQL

```
Q = SELECT SUM(1 *1), SUM(1 *X_1), ... SUM(1 *X_n), SUM(1 *Y),
           SUM(X_1*1), SUM(X_1 *X_1), ... SUM(X_1*X_n), SUM(X_1*Y),
           ...
           SUM(X_n*1), SUM(X_n*X_1), ... SUM(X_n*X_n), SUM(X_n*Y)
           SUM(Y *1), SUM(Y *X_1), ... SUM(Y *X_n), SUM(Y *Y)
      FROM R1 JOIN R2 JOIN ... JOIN Rn
```

Covariance Matrix Defined by Queries

Covariance matrix $[\mathbf{X}\ \mathbf{Y}]^T [\mathbf{X}\ \mathbf{Y}]$ can be expressed in SQL

```
Q = SELECT [ SUM(1 *1),  SUM(1 *X1),  ... SUM(1 *Xn),  SUM(1 *Y),
             SUM(X1*1),  SUM(X1*X1),  ... SUM(X1*Xn),  SUM(X1*Y),
             ...
             SUM(Xn*1),  SUM(Xn*X1),  ... SUM(Xn*Xn),  SUM(Xn*Y),
             SUM(Y *1),  SUM(Y *X1),  ... SUM(Y *Xn),  SUM(Y *Y) ]
      FROM R1 JOIN R2 JOIN ... JOIN Rn
```

We compute and maintain under data updates:

- COUNT $=$ SUM(1) $=$ database join size
- vector of SUM($\mathbf{X}_i$) for feature/label $\mathbf{X}_i$
- matrix of SUM($\mathbf{X}_i \cdot \mathbf{X}_j$) for features/label $\mathbf{X}_i$ and $\mathbf{X}_j$

The Covariance Ring

Covariance Ring has the support:

- Set of triples $(\mathbb{Z}, \mathbb{R}^m, \mathbb{R}^{m\times m})$

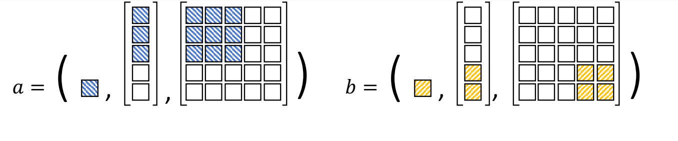
  $$\Big(\text{ COUNT}, \quad \text{vector of SUM}(\mathbf{X}_i), \quad \text{matrix of SUM}(\mathbf{X}_i \cdot \mathbf{X}_j)\Big)$$

- Neutral elements for sum and product operations:

$$\mathbf{0} = (0, \mathbf{0}_{m\times 1}, \mathbf{0}_{m\times m})$$
$$\mathbf{1} = (1, \mathbf{0}_{m\times 1}, \mathbf{0}_{m\times m})$$

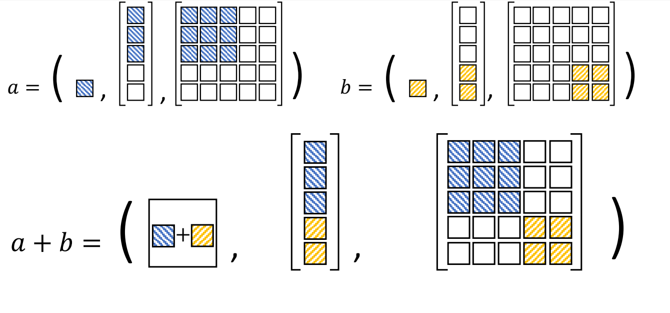
The Covariance Ring

Covariance Ring has the sum and product operations:



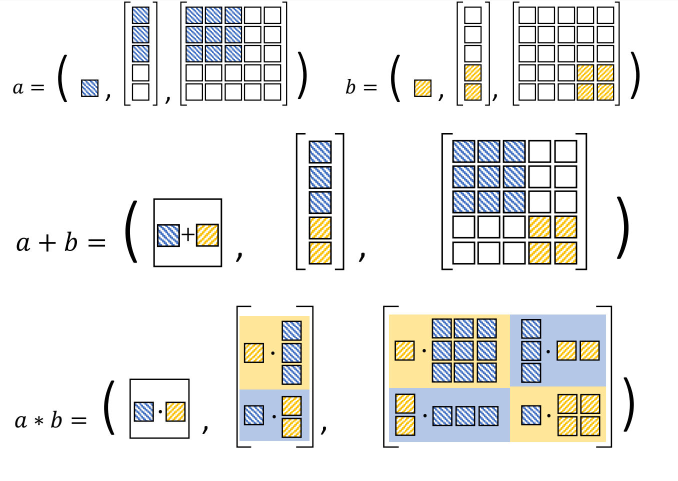
The Covariance Ring

Covariance Ring has the sum and product operations:



The Covariance Ring

Covariance Ring has the sum and product operations:



References i

[SIGMOD 2018] Milos Nikolic, Dan Olteanu. *Incremental View Maintenance with Triple Lock Factorization Benefits.*

[SIGMOD 2020] Milos Nikolic, Haozhe Zhang, Ahmet Kara, Dan Olteanu. *F-IVM: Learning over Fast-Evolving Relational Data.*

[VLDBJ 2023] Ahmet Kara, Milos Nikolic, Dan Olteanu, Haozhe Zhang. *F-IVM: Analytics over Relational Databases under Updates.* (To appear)

Thank You!