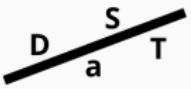


# Efficient Algorithms for Frequently Asked Questions

## 10. Dynamic Evaluation of Functional Aggregate Queries

---

Prof. Dan Olteanu

**DaST**        
Data • (Systems+Theory)

May 23, 2022

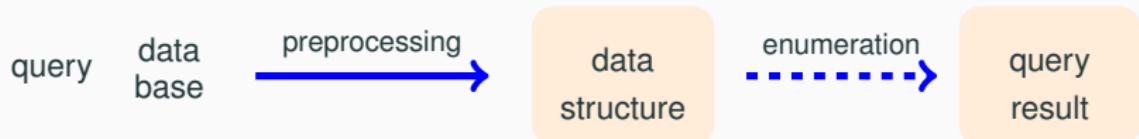


University of  
Zurich<sup>UZH</sup>

<https://lms.uzh.ch/url/RepositoryEntry/17185308706>

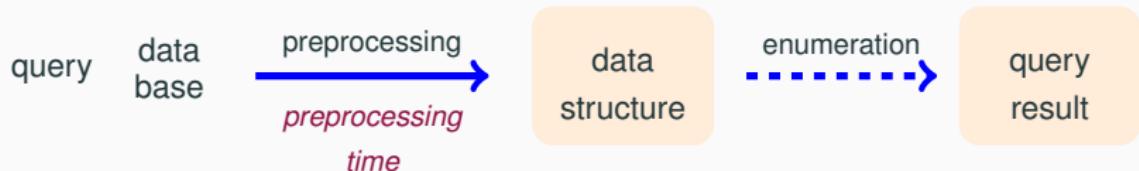
# Static and Dynamic Query Evaluation

## Static Query Evaluation



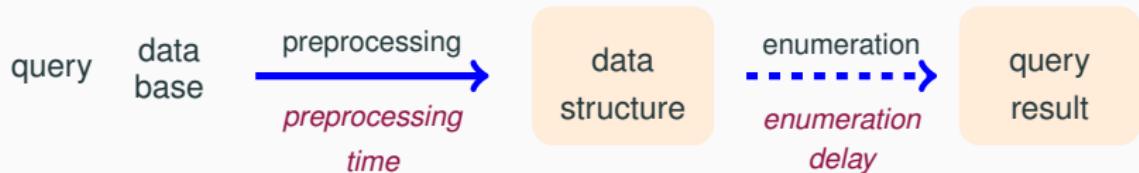
# Static and Dynamic Query Evaluation

## Static Query Evaluation



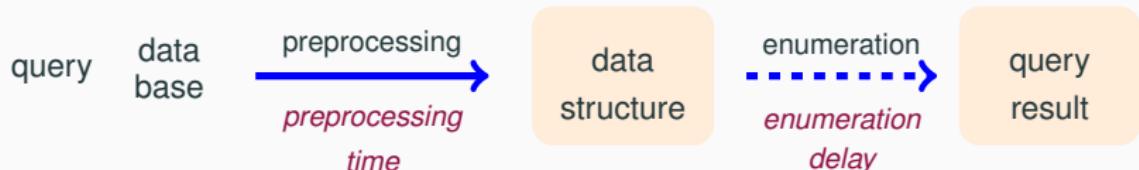
# Static and Dynamic Query Evaluation

## Static Query Evaluation



# Static and Dynamic Query Evaluation

## Static Query Evaluation

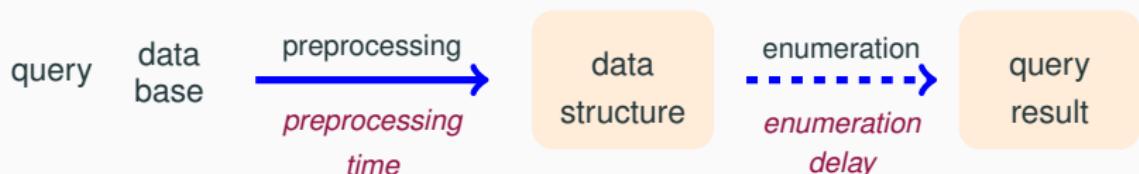


## Dynamic Query Evaluation

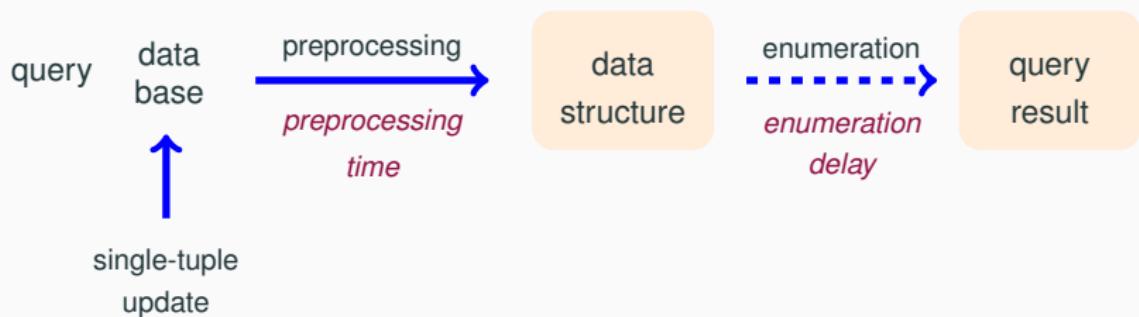


# Static and Dynamic Query Evaluation

## Static Query Evaluation

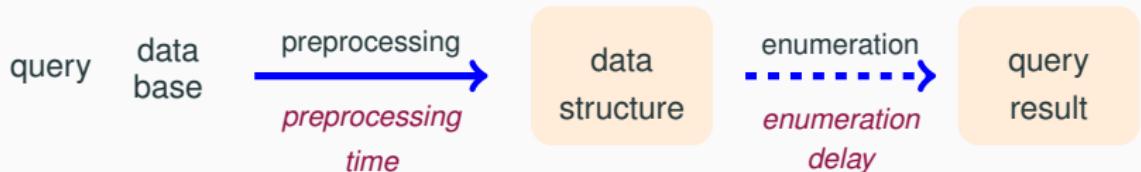


## Dynamic Query Evaluation

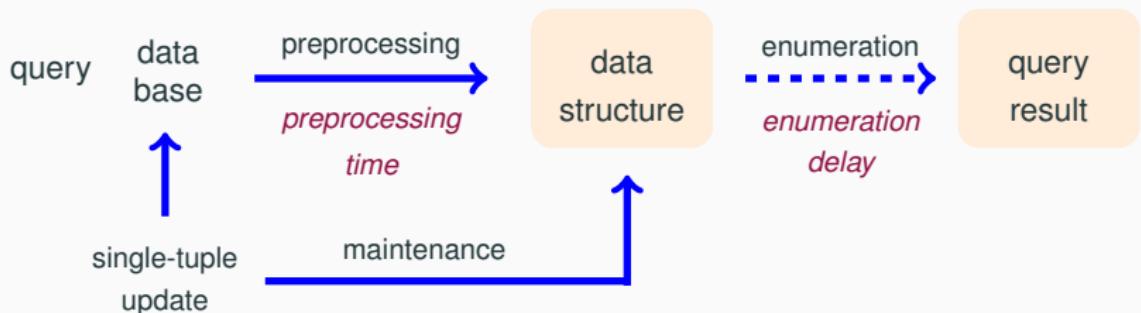


# Static and Dynamic Query Evaluation

## Static Query Evaluation

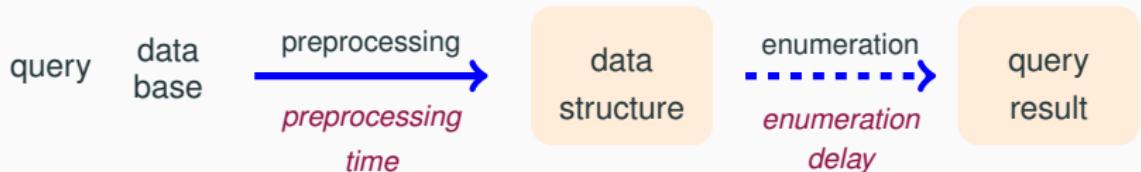


## Dynamic Query Evaluation

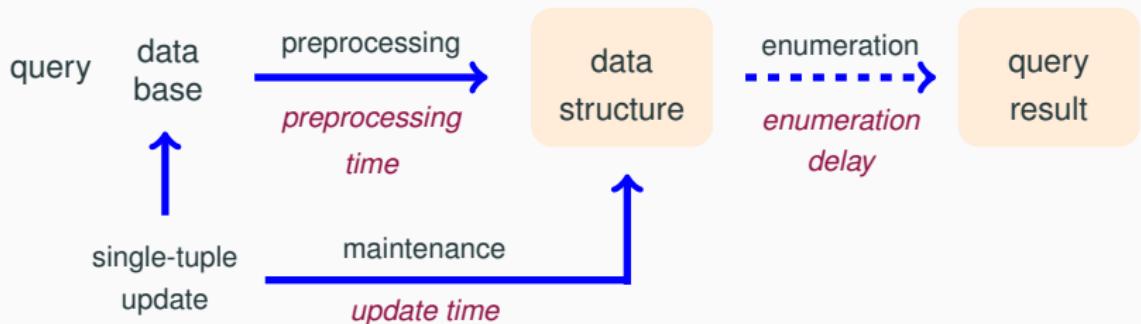


## Static and Dynamic Query Evaluation

## Static Query Evaluation

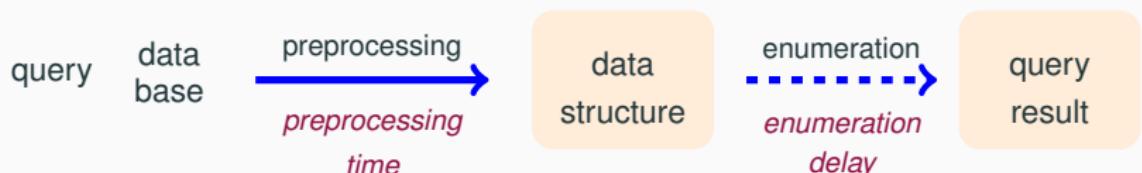


## Dynamic Query Evaluation

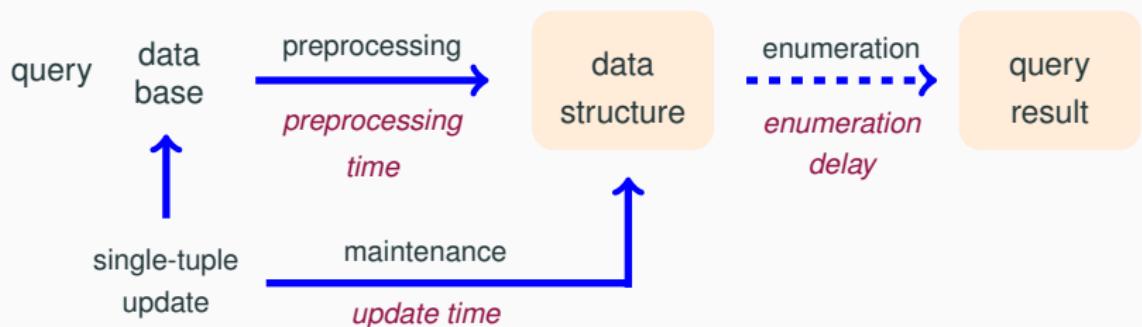


# Static and Dynamic Query Evaluation

## Static Query Evaluation

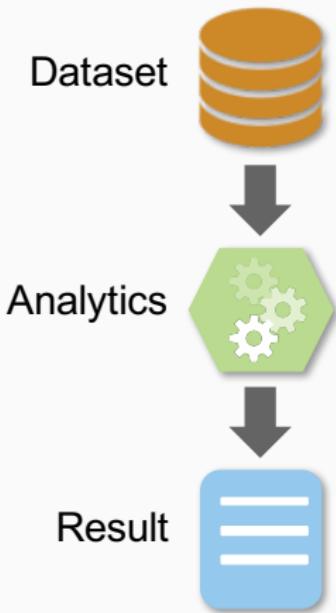


## Dynamic Query Evaluation

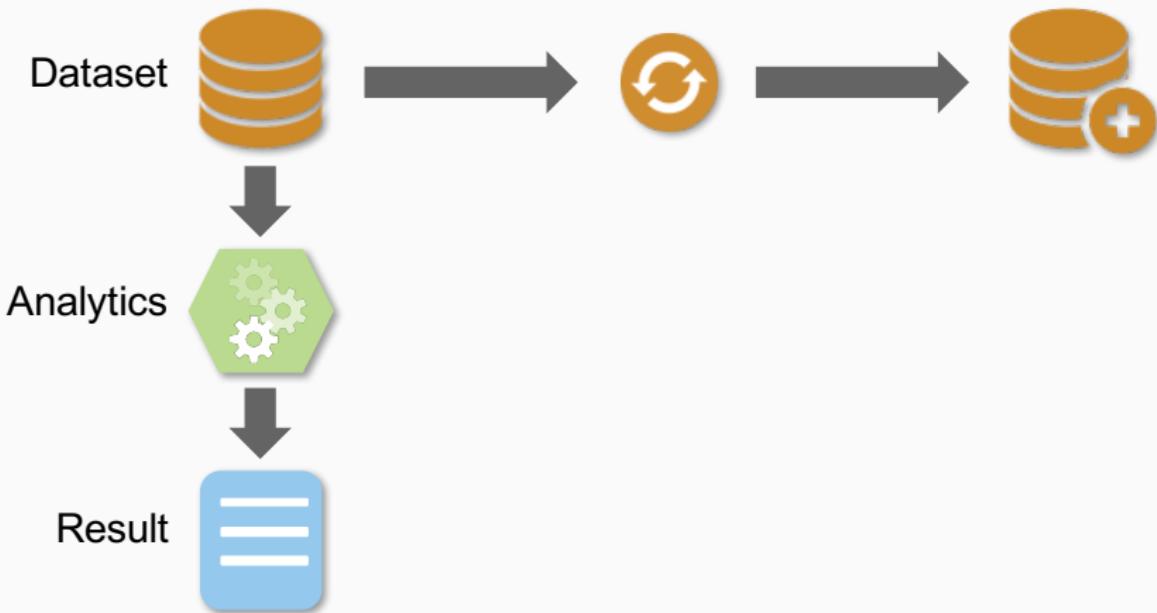


We are interested in the **trade-off** between:  
preprocessing time - enumeration delay - update time

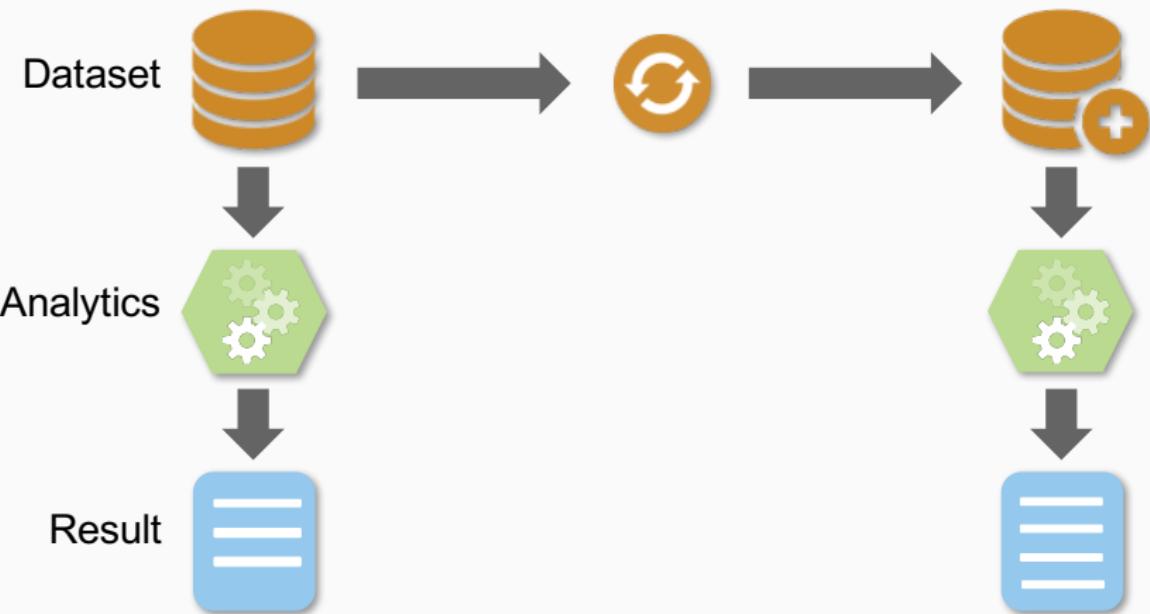
## Computation from Scratch vs Delta Computation



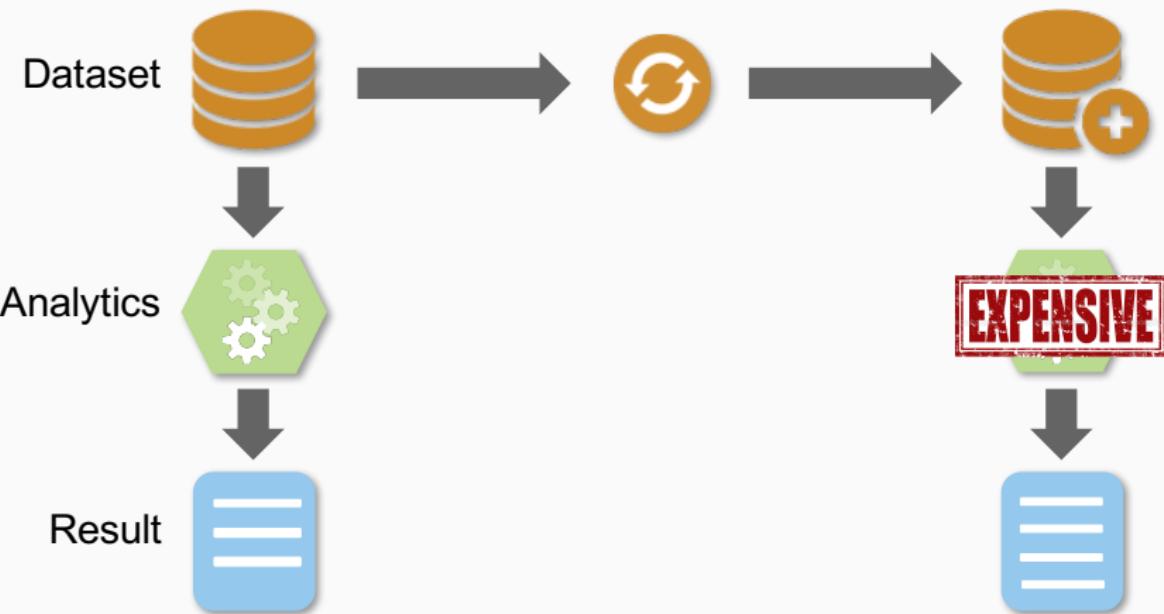
## Computation from Scratch vs Delta Computation



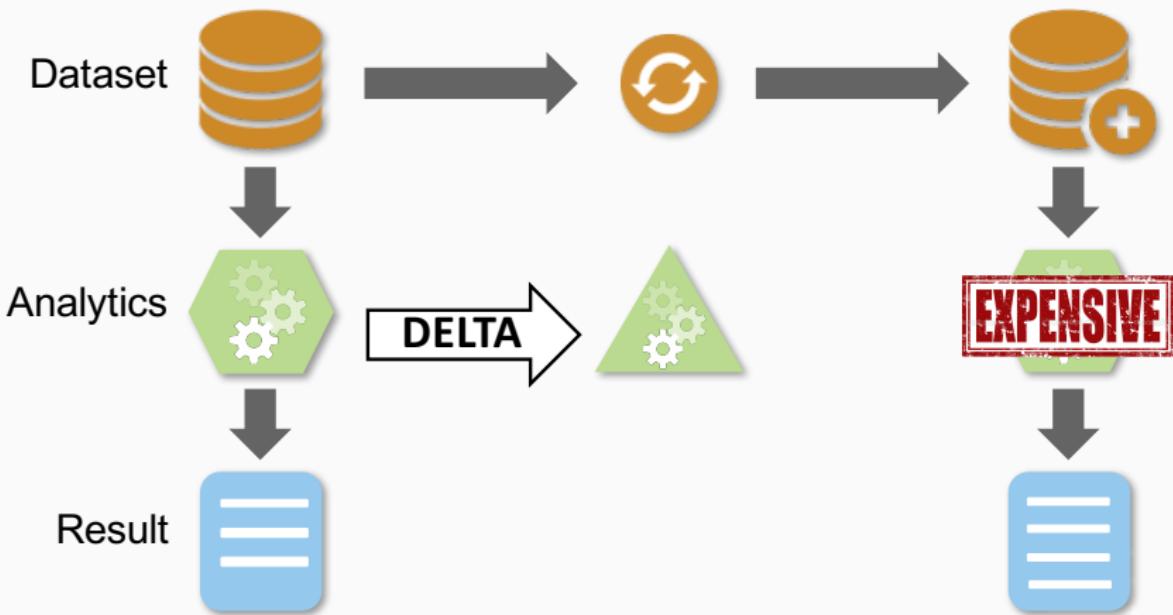
## Computation from Scratch vs Delta Computation



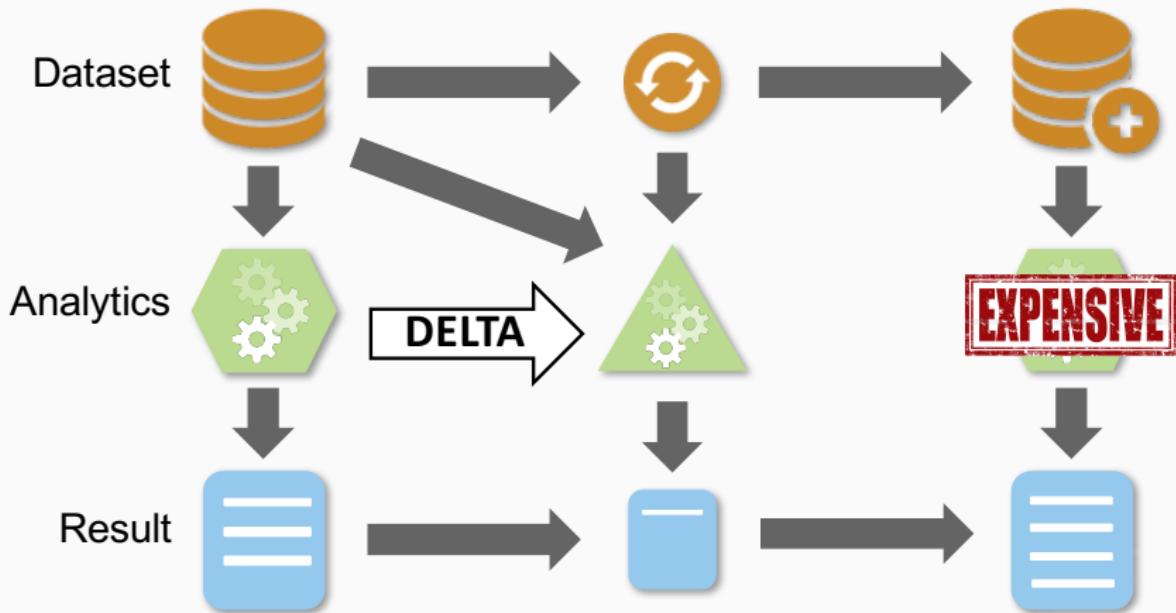
## Computation from Scratch vs Delta Computation



## Computation from Scratch vs Delta Computation



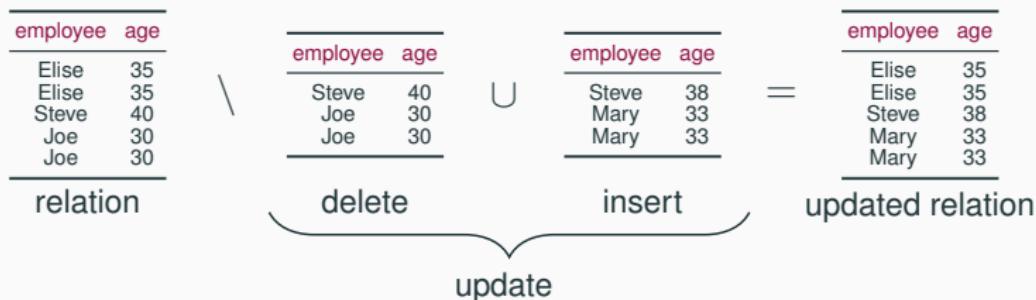
## Computation from Scratch vs Delta Computation



# **Representation of Updates**

# Traditional and Uniform Update Representation

- **Traditional representation:** Insertions and deletions are represented as separate tables:



- **Uniform representation:** Insertions and deletions are represented as a single factor over a **ring**. Here, we use the ring  $(\mathbb{Z}, +, *, 0, 1)$ :



## Example: Single-Tuple Updates to the Triangle Query

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

<i>R</i>		<i>S</i>		<i>T</i>	
<i>A</i>	<i>B</i>	<i>B</i>	<i>C</i>	<i>C</i>	<i>A</i>
$a_1$	$b_1$	2	$b_1$	$c_1$	2
$a_2$	$b_1$	3	$b_1$	$c_2$	1
				$c_1$	$a_1$
				$c_2$	$a_1$
				$c_2$	$a_2$

## Example: Single-Tuple Updates to the Triangle Query

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

R		S		T		R · S · T			
A	B	B	C	C	A	A	B	C	#
$a_1$	$b_1$	$b_1$	$c_1$	$c_1$	$a_1$	$c_1$	$b_1$	$c_1$	2
$a_2$	$b_1$	$b_1$	$c_2$	$c_2$	$a_1$	$c_2$	$a_1$	$c_2$	3
						$c_2$	$a_2$		3

$\frac{a_1 b_1 c_1 | 2 \cdot 2 \cdot 1 = 4}{a_2 b_1 c_2 | 3}$

## Example: Single-Tuple Updates to the Triangle Query

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

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

## Example: Single-Tuple Updates to the Triangle Query

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

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



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

## Example: Single-Tuple Updates to the Triangle Query

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

- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

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



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

A	B	#
$a_2$	$b_1$	-2



$$Q()$$

Ø	#
()	$4 + 6 + 9 = 19$

## Example: Single-Tuple Updates to the Triangle Query

- $$Q() = \sum_{a,b,c} R(a,b) \cdot S(b,c) \cdot T(c,a)$$
- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

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



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

A	B	#
$a_2$	$b_1$	-2



$$Q()$$

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

## Example: Single-Tuple Updates to the Triangle Query

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

- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

$R$		$S$		$T$		$R \cdot S \cdot T$		
$A$	$B$	$B$	$C$	$C$	$A$	$A$	$B$	$C$
$a_1$	$b_1$	$b_1$	$c_1$	$c_1$	$a_1$	$a_1$	$b_1$	$c_1$
$a_2$	$b_1$	$b_1$	$c_2$	$c_2$	$a_1$	$c_2$	$b_1$	$c_2$
$a_2$	$b_1$				$a_2$	$a_2$	$b_1$	$c_2$



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

$A$	$B$	#
$a_2$	$b_1$	-2

$$Q()$$

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

## Example: Single-Tuple Updates to the Triangle Query

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

- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R		S		T		R · S · T			
A	B	B	C	C	A	A	B	C	#
$a_1$	$b_1$	$b_1$	$c_1$	$c_1$	$a_1$	1	$a_1$	$b_1$	$c_1$
$a_2$	$b_1$	$b_1$	$c_2$	$c_2$	$a_1$	3	$a_1$	$b_1$	$c_2$
$a_2$	$b_1$			$c_2$	$a_2$	1	$a_2$	$b_1$	$c_2$



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

A	B	#
$a_2$	$b_1$	-2

$$Q()$$

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

## Example: Single-Tuple Updates to the Triangle Query

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

- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

$R$		$S$		$T$		$R \cdot S \cdot T$		
$A$	$B$	$B$	$C$	$C$	$A$	$A$	$B$	$C$
$a_1$	$b_1$	$b_1$	$c_1$	$c_1$	$a_1$	$c_1$	$b_1$	$c_1$
$a_2$	$b_1$	$b_1$	$c_2$	$b_1$	$a_1$	$c_2$	$b_1$	$c_2$
$a_2$	$b_1$				$a_2$	$c_2$	$b_1$	$c_2$

↑   ↓

$A$	$B$	#
$a_2$	$b_1$	-2

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

## Example: Single-Tuple Updates to the Triangle Query

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

- A single-tuple update is a factor mapping a tuple to a non-zero value (positive for insertions, negative for deletions)

R		S		T		R · S · T			
A	B	B	C	C	A	A	B	C	#
$a_1$	$b_1$	$b_1$	$c_1$	$c_1$	$a_1$	$c_1$	$b_1$	$c_1$	$2 \cdot 2 \cdot 1 = 4$
$\cancel{a_2}$	$\cancel{b_1}$	$\cancel{b_1}$	$\cancel{c_2}$	$\cancel{c_2}$	$\cancel{a_1}$	$\cancel{c_2}$	$\cancel{b_1}$	$\cancel{c_2}$	$2 \cdot 1 \cdot 3 = 6$
$\cancel{a_2}$	$\cancel{b_1}$	$\cancel{b_1}$	$\cancel{c_2}$	$\cancel{c_2}$	$\cancel{a_2}$	$\cancel{c_2}$	$\cancel{b_1}$	$\cancel{c_2}$	$3 \cdot 1 \cdot 3 = 9$
$\cancel{a_2}$	$\cancel{b_1}$	$\cancel{b_1}$	$\cancel{c_2}$	$\cancel{c_2}$	$\cancel{a_2}$	$\cancel{c_2}$	$\cancel{b_1}$	$\cancel{c_2}$	$1 \cdot 1 \cdot 3 = 3$



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

A	B	#
$a_2$	$b_1$	-2

$$Q()$$

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

# **Delta Queries**

## Delta Queries: Example

Consider the following FAQ query and an update  $\delta R$  to the factor  $R$

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

We derive the updated FAQ  $Q_{\text{new}}$

## Delta Queries: Example

Consider the following FAQ query and an update  $\delta R$  to the factor  $R$

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

We derive the updated FAQ  $Q_{\text{new}}$

$$Q_{\text{new}}(a, c) = \sum_b (R(a, b) + \delta R(a, b)) \cdot S(b, c)$$

## Delta Queries: Example

Consider the following FAQ query and an update  $\delta R$  to the factor  $R$

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

We derive the updated FAQ  $Q_{\text{new}}$

$$\begin{aligned} Q_{\text{new}}(a, c) &= \sum_b (R(a, b) + \delta R(a, b)) \cdot S(b, c) \\ &= \sum_b (R(a, b) \cdot S(b, c)) + (\delta R(a, b) \cdot S(b, c)) \end{aligned}$$

## Delta Queries: Example

Consider the following FAQ query and an update  $\delta R$  to the factor  $R$

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

We derive the updated FAQ  $Q_{\text{new}}$

$$\begin{aligned} Q_{\text{new}}(a, c) &= \sum_b (R(a, b) + \delta R(a, b)) \cdot S(b, c) \\ &= \sum_b (R(a, b) \cdot S(b, c)) + (\delta R(a, b) \cdot S(b, c)) \\ &= \underbrace{\sum_b (R(a, b) \cdot S(b, c))}_{Q(a, c)} + \underbrace{\sum_b (\delta R(a, b) \cdot S(b, c))}_{\delta Q(a, c)} \end{aligned}$$

$\delta Q$  defines the change in the query result after applying  $\delta R$  to the database

## Delta Queries: Example

Consider the following FAQ query and an update  $\delta R$  to the factor  $R$

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

We derive the updated FAQ  $Q_{\text{new}}$

$$\begin{aligned} Q_{\text{new}}(a, c) &= \sum_b (R(a, b) + \delta R(a, b)) \cdot S(b, c) \\ &= \sum_b (R(a, b) \cdot S(b, c)) + (\delta R(a, b) \cdot S(b, c)) \\ &= \underbrace{\sum_b (R(a, b) \cdot S(b, c))}_{Q(a, c)} + \underbrace{\sum_b (\delta R(a, b) \cdot S(b, c))}_{\delta Q(a, c)} \end{aligned}$$

$\delta Q$  defines the change in the query result after applying  $\delta R$  to the database

Next, we give rules to derive delta queries for general FAQs

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\varphi_{\text{new}}(\mathbf{x}) = (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2))$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus\end{aligned}$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \\ &\quad \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \\ &\quad \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

If  $\psi$  only occurs in  $\varphi_1$ , then  $\delta\varphi_2$  becomes  $\mathbf{0}$  and the formula gets simpler:

$$\varphi_{\text{new}}(\mathbf{x}) = (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2))$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \\ &\quad \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

If  $\psi$  only occurs in  $\varphi_1$ , then  $\delta\varphi_2$  becomes  $\mathbf{0}$  and the formula gets simpler:

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \mathbf{0})\end{aligned}$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \\ &\quad \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

If  $\psi$  only occurs in  $\varphi_1$ , then  $\delta\varphi_2$  becomes  $\mathbf{0}$  and the formula gets simpler:

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \mathbf{0}) \\ &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes \varphi_2(\mathbf{x}_2)\end{aligned}$$

## Delta Queries: Product Case

Consider an FAQ query  $\varphi = \varphi_1 \otimes \varphi_2$  and an update  $\delta\psi$  to a factor  $\psi$  in  $\varphi$

We derive the updated FAQ  $\varphi_{\text{new}}$ :

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \\ &\quad \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1)) \oplus (\delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

If  $\psi$  only occurs in  $\varphi_1$ , then  $\delta\varphi_2$  becomes  $\mathbf{0}$  and the formula gets simpler:

$$\begin{aligned}\varphi_{\text{new}}(\mathbf{x}) &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_2(\mathbf{x}_2)) \\ &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes (\varphi_2(\mathbf{x}_2) \oplus \mathbf{0}) \\ &= (\varphi_1(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1)) \otimes \varphi_2(\mathbf{x}_2) \\ &= \underbrace{(\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2))}_{\varphi(\mathbf{x})} \oplus \underbrace{((\delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2)))}_{\delta\varphi(\mathbf{x})}\end{aligned}$$

## Delta Queries: Rules

The following rules follow from the associativity, commutativity, and distributivity of ring operations

Query $\varphi(\mathbf{x})$	Delta query $\delta\varphi(\mathbf{x})$
$\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2)$	$\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)$

## Delta Queries: Rules

The following rules follow from the associativity, commutativity, and distributivity of ring operations

Query $\varphi(\mathbf{x})$	Delta query $\delta\varphi(\mathbf{x})$
$\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2)$	$\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)$
$\bigoplus_x \varphi_1(\mathbf{x}_1)$	$\bigoplus_x \delta\varphi_1(\mathbf{x}_1)$

## Delta Queries: Rules

The following rules follow from the associativity, commutativity, and distributivity of ring operations

Query $\varphi(\mathbf{x})$	Delta query $\delta\varphi(\mathbf{x})$
$\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2)$	$\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)$
$\bigoplus_x \varphi_1(\mathbf{x}_1)$	$\bigoplus_x \delta\varphi_1(\mathbf{x}_1)$
$\varphi_1(\mathbf{x}) \oplus \varphi_2(\mathbf{x})$	$\delta\varphi_1(\mathbf{x}) \oplus \delta\varphi_2(\mathbf{x})$

## Delta Queries: Rules

The following rules follow from the associativity, commutativity, and distributivity of ring operations

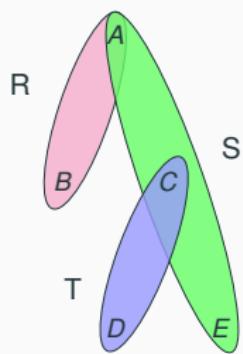
Query $\varphi(\mathbf{x})$	Delta query $\delta\varphi(\mathbf{x})$
$\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_2)$	$\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \varphi_2(\mathbf{x}_1) \oplus \delta\varphi_1(\mathbf{x}_1) \otimes \delta\varphi_2(\mathbf{x}_2)$
$\bigoplus_x \varphi_1(\mathbf{x}_1)$	$\bigoplus_x \delta\varphi_1(\mathbf{x}_1)$
$\varphi_1(\mathbf{x}) \oplus \varphi_2(\mathbf{x})$	$\delta\varphi_1(\mathbf{x}) \oplus \delta\varphi_2(\mathbf{x})$
$\psi'(\mathbf{x})$	$\delta\psi(\mathbf{x})$ when $\psi = \psi'$ and $\mathbf{0}$ otherwise

# **View Trees for Dynamic Query Evaluation**

## Example: Simple Sum Aggregate 1/3

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

How can we compute Q?



## Example: Simple Sum Aggregate 2/3

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

**Naïve Approach:** Compute the join and then take the sum

$$Q() = \sum_{a,b,c,d,e} V_{RST}(a, b, c, d, e)$$



$$V_{RST}(a, b, c, d, e) = R(a, b) \cdot V_{ST}(a, c, d, e)$$

$$\begin{array}{ccc} & & \\ R(a, b) & & V_{ST}(a, c, d, e) = T(c, d) \cdot S(a, c, e) \\ & \diagdown & \diagup \\ & T(c, d) & S(a, c, e) \end{array}$$

## Example: Simple Sum Aggregate 2/3

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

**Naïve Approach:** Compute the join and then take the sum

$$Q() = \sum_{a,b,c,d,e} V_{RST}(a,b,c,d,e)$$

$$V_{RST}(a, b, c, d, e) = R(a, b) \cdot V_{ST}(a, c, d, e)$$

$$R(a,b)$$

$$V_{ST}(a,$$

$$(c, d) \cdot S(a, c, e)$$

$$T(c, d)$$

$S(a, c, e)$

Let all relations be of size  $N$

Computation time:  $\mathcal{O}(N^3)$

## Example: Simple Sum Aggregate 2/3

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

**Naïve Approach:** Compute the join and then take the sum

$$Q() = \sum_{a,b,c,d,e} V_{RST}(a,b,c,d,e)$$

$$V_{RST}(a, b, c, d, e) = R(a, b) \cdot V_{ST}(a, c, d, e)$$

$$R(a,b)$$

$$V_{ST}(a, c, d, e)$$

$$T(c, d)$$

$S(a, c, e)$

Let all relations be of size  $N$

Computation time:  $\mathcal{O}(N^3)$

## Can we do better?

## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum

$$Q() = \sum_a V_R(a) \cdot V_{ST}(a)$$

$$V_R(a) = \sum_b R(a, b)$$

$$V_{ST}(a) = \sum_c V_T(c) \cdot V_S(a, c)$$

$$R(a, b)$$

$$V_T(c) = \sum_d T(c, d)$$

$$V_S(a, c) = \sum_e S(a, c, e)$$

$$T(c, d)$$

$$S(a, c, e)$$

## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum  
Join on & eliminate one variable at a time

$$Q() = \sum_a V_R(a) \cdot V_{ST}(a)$$
$$V_R(a) = \sum_b R(a, b)$$
$$V_{ST}(a) = \sum_c V_T(c) \cdot V_S(a, c)$$
$$R(a, b)$$
$$T(c, d)$$
$$S(a, c, e)$$

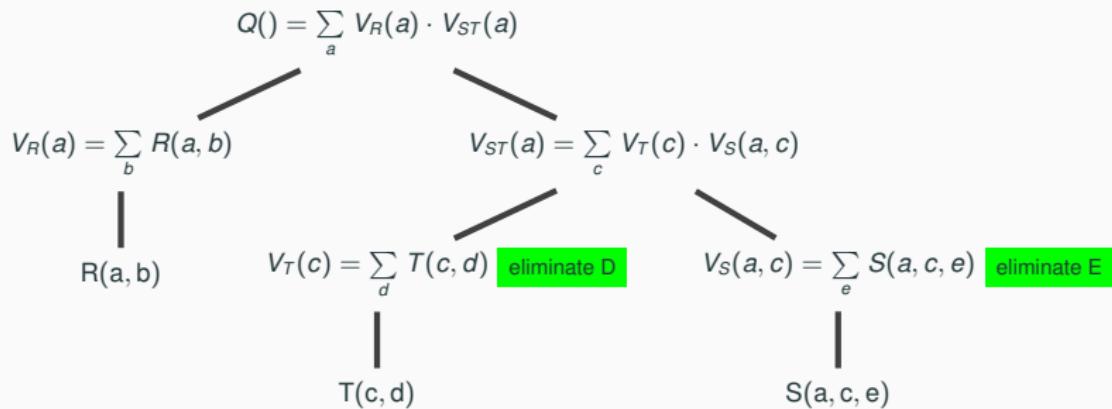
eliminate D

## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum  
Join on & eliminate one variable at a time

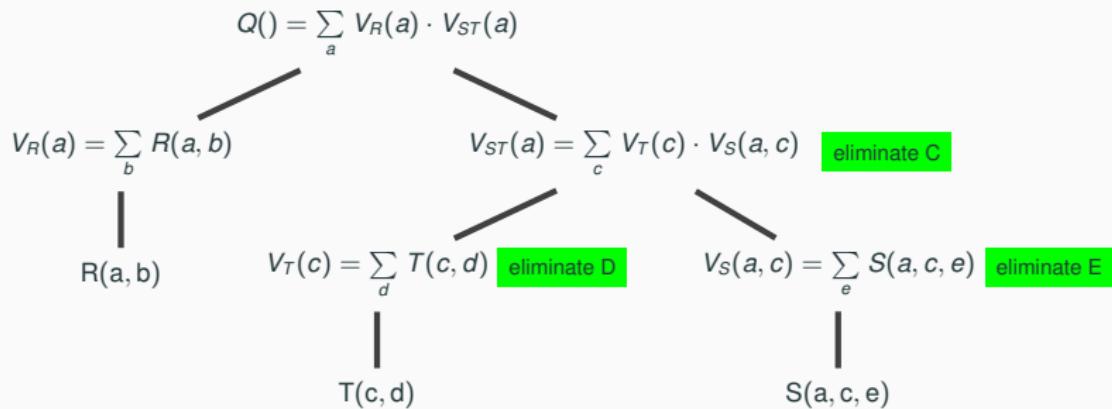


## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum  
Join on & eliminate one variable at a time

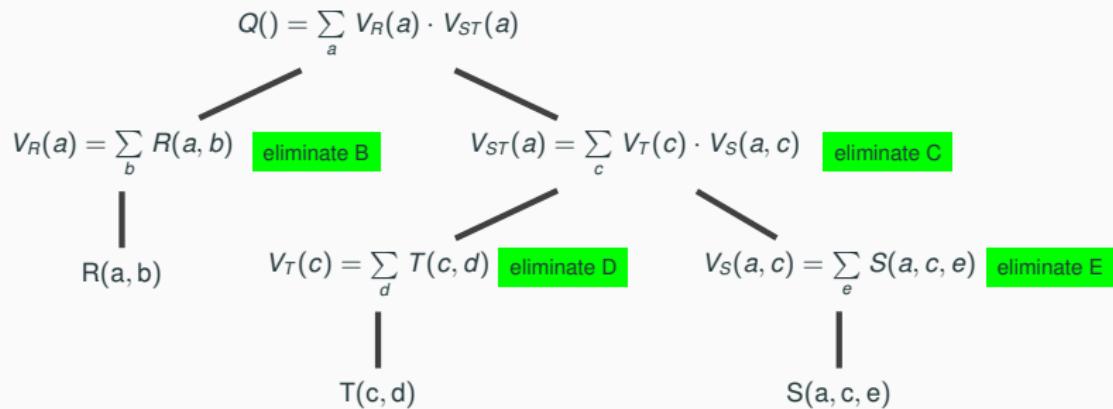


## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum  
Join on & eliminate one variable at a time

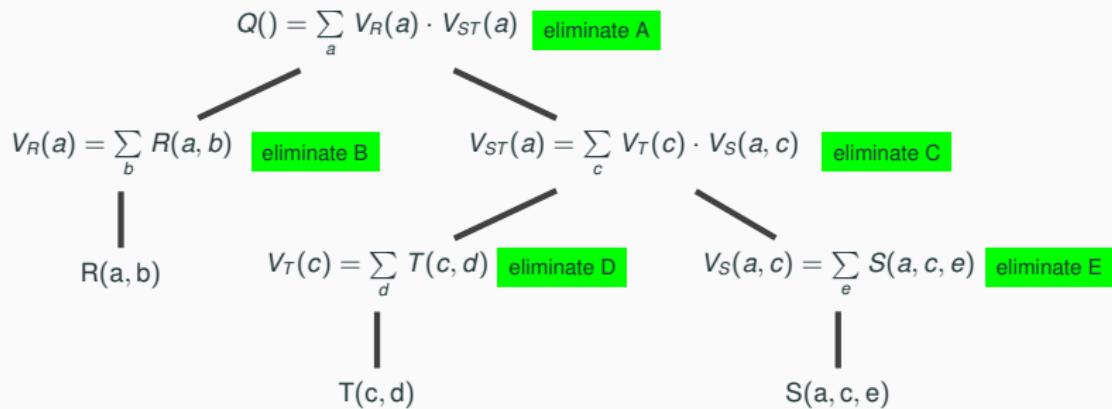


## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

Use distributivity of product over sum  
Join on & eliminate one variable at a time



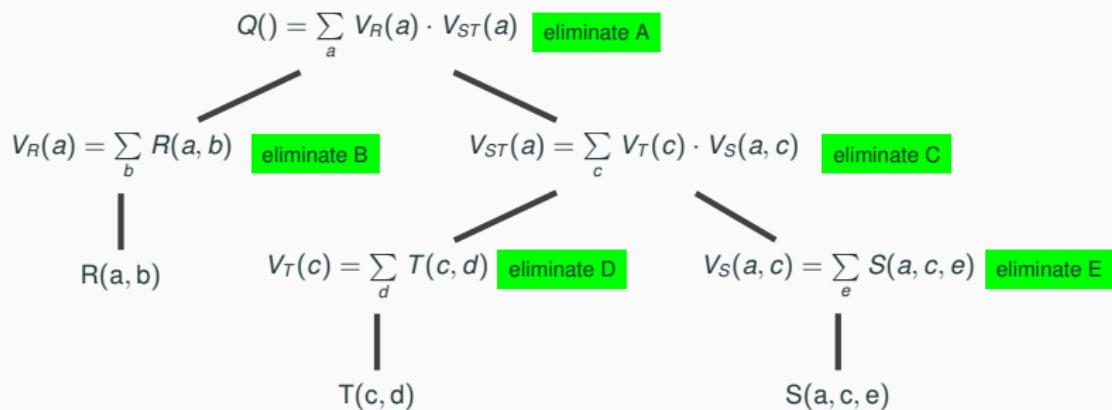
## Example: Simple Sum Aggregate 3/3

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

Push sum past product to marginalize variables early on

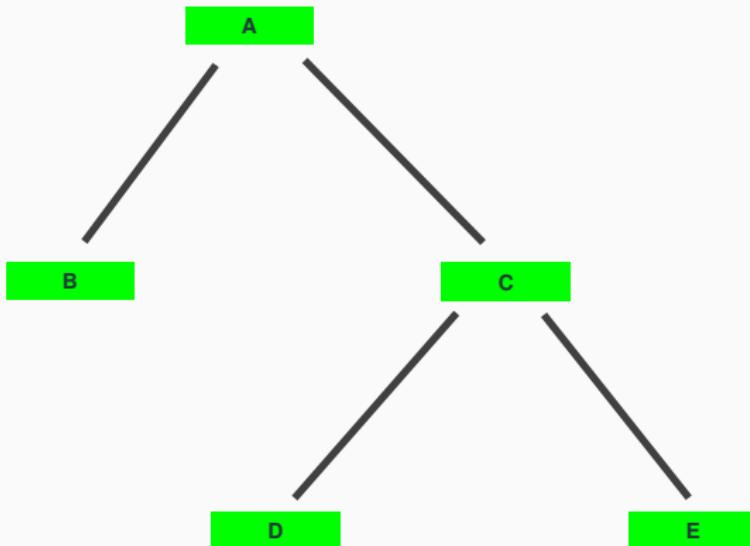
Use distributivity of product over sum  
Join on & eliminate one variable at a time

Computation time:  $\mathcal{O}(N)$



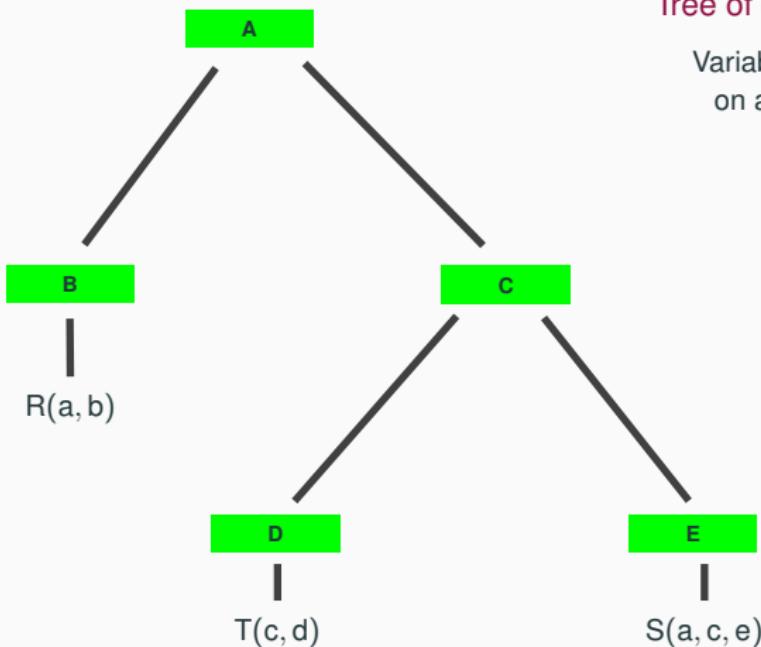
## Query Evaluation Plans using Variable Orders

Variable order for  $Q() = \sum_{a,b,c,d,e} R(a, b) \cdot S(a, c, e) \cdot T(c, d)$



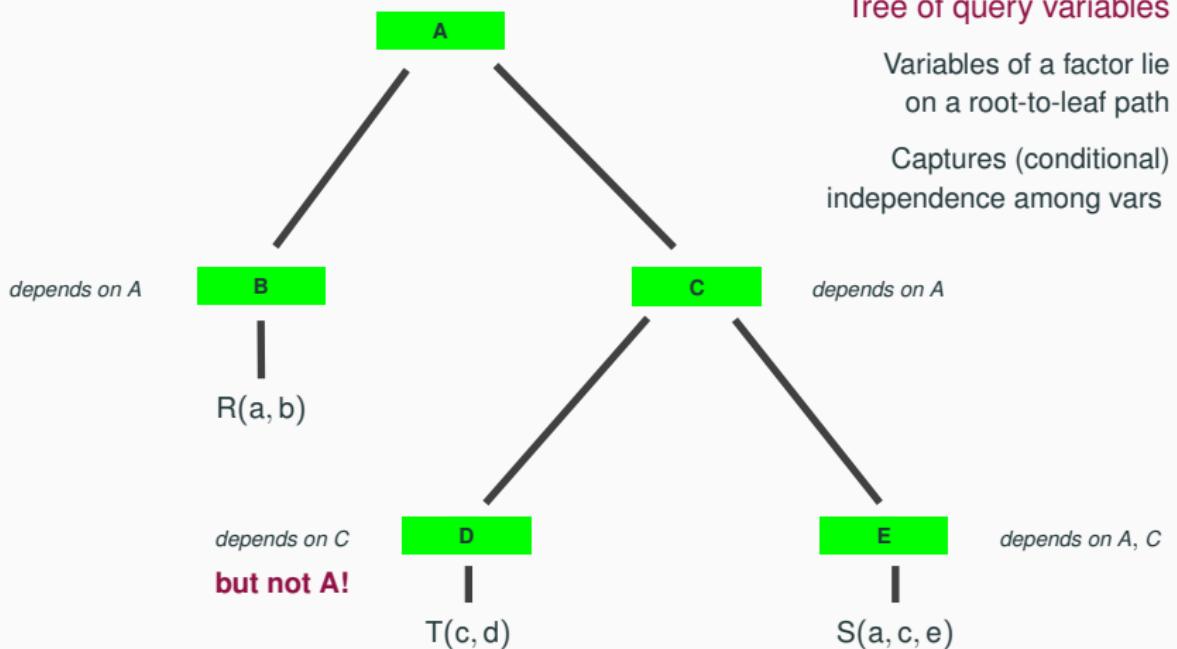
# Query Evaluation Plans using Variable Orders

Variable order for  $Q() = \sum_{a,b,c,d,e} R(a, b) \cdot S(a, c, e) \cdot T(c, d)$



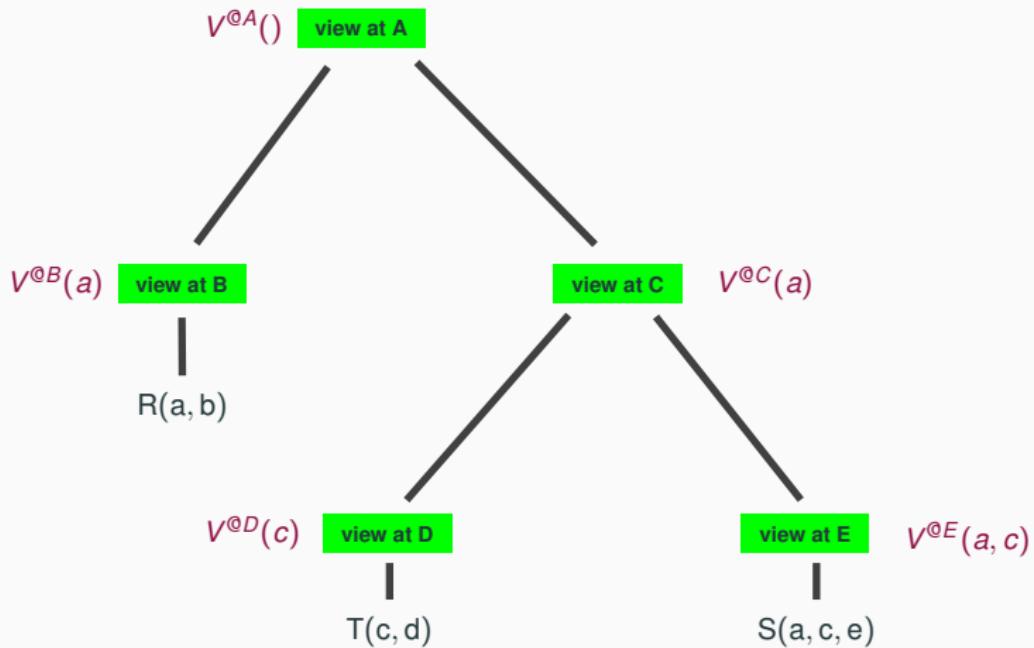
# Query Evaluation Plans using Variable Orders

Variable order for  $Q() = \sum_{a,b,c,d,e} R(a, b) \cdot S(a, c, e) \cdot T(c, d)$



## View Trees

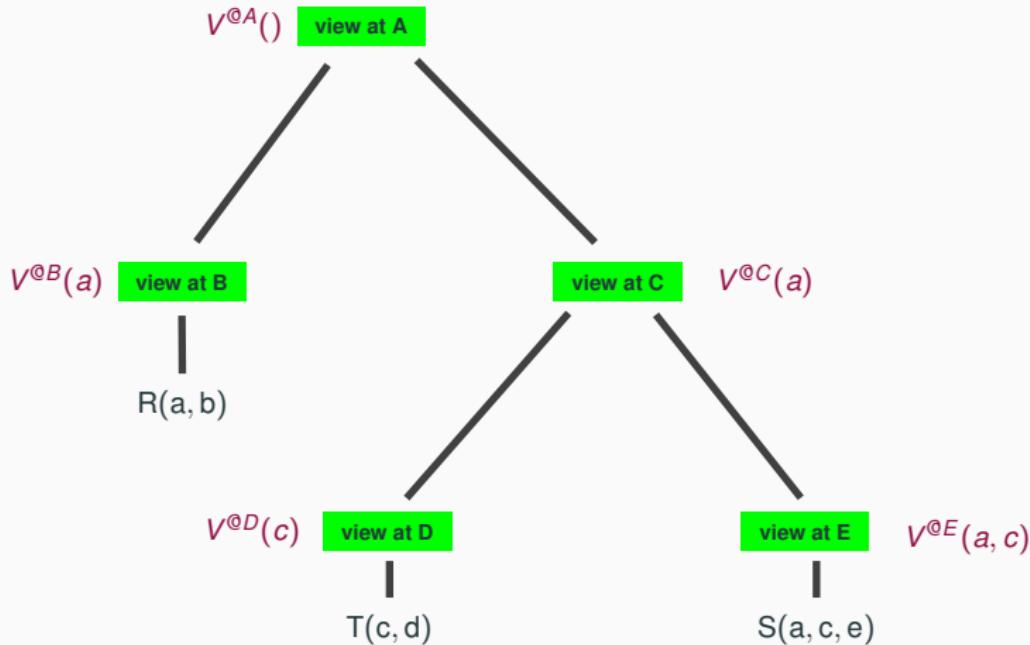
Create a **view** at each var  $X$   
with schema **depends(X)**



## View Trees

Create a **view** at each var  $X$   
with schema **depends(X)**

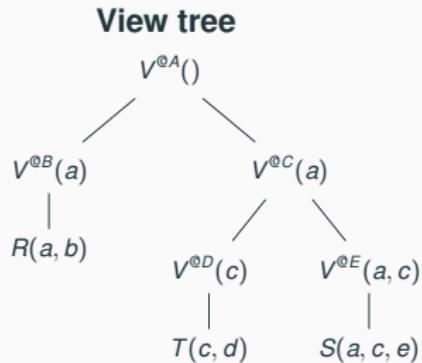
View at variable  $X$ :  
joins its child views  
aggregates away  $X$   
(if  $X$  is not a free var)



# Delta Propagation

Consider our running example

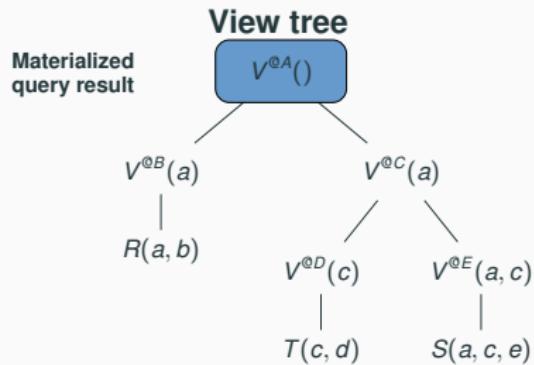
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

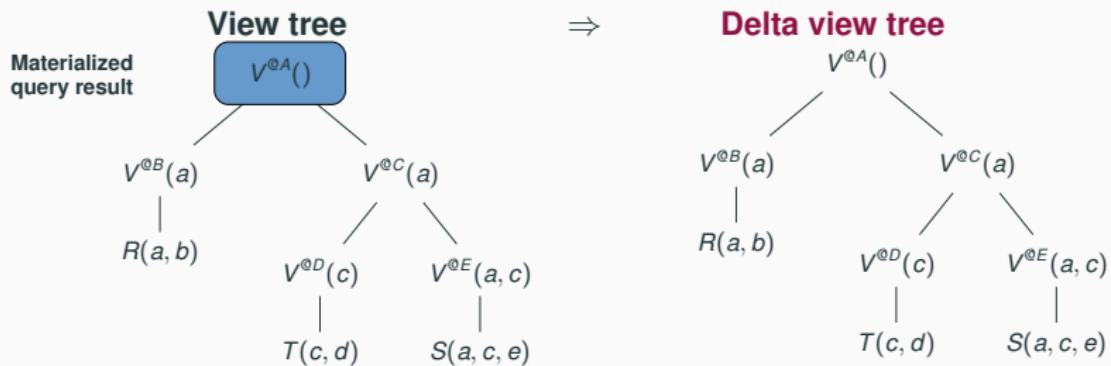
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

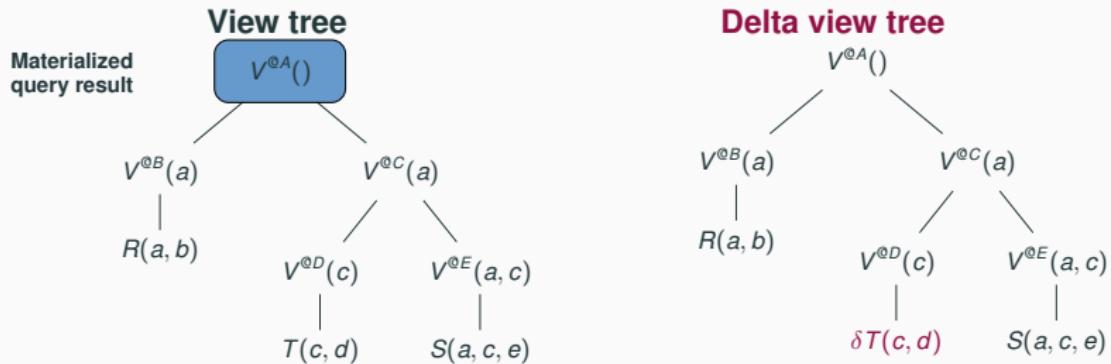
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

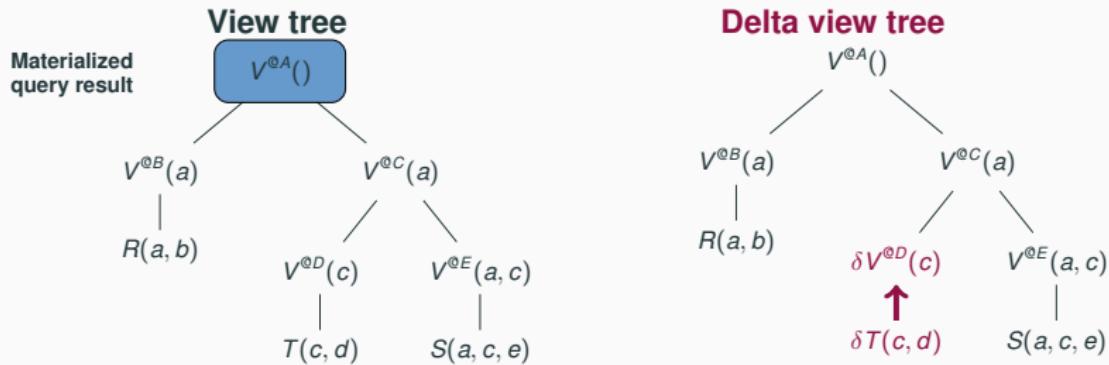
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

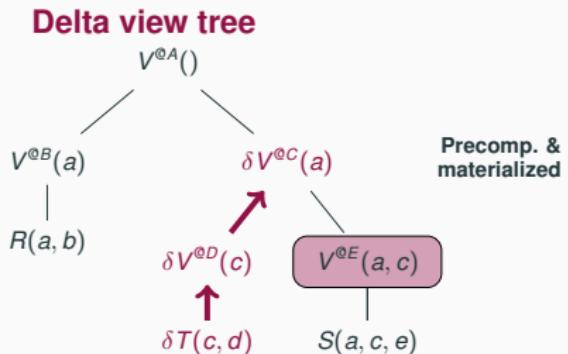
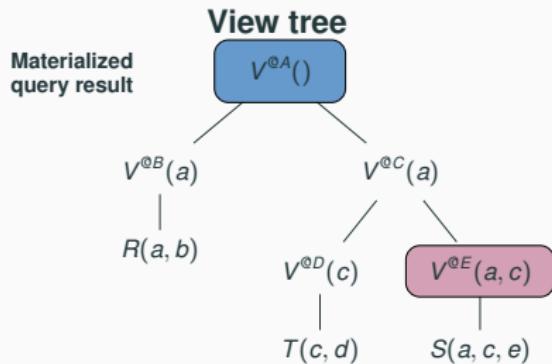
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

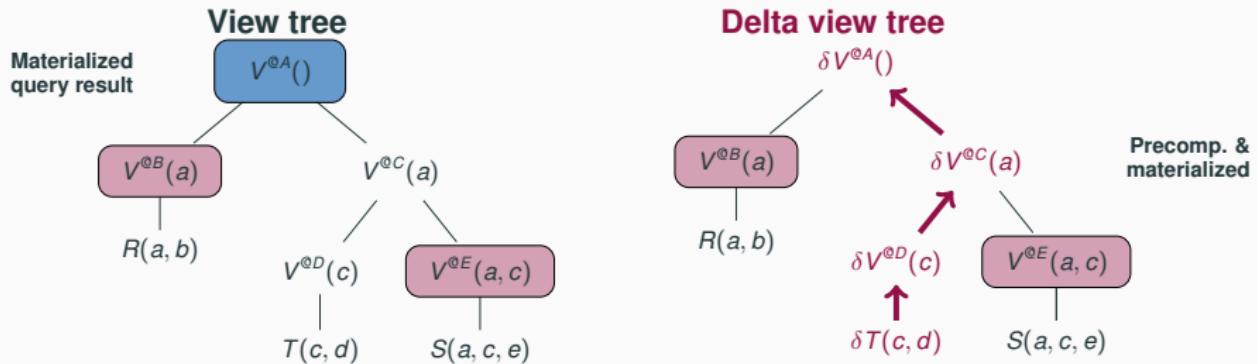
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

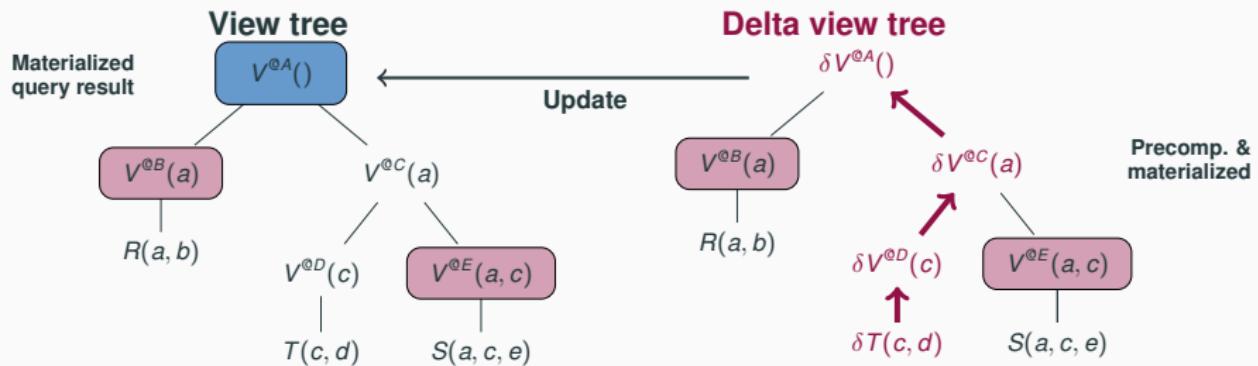
Maintain the query result under updates to  $T$



# Delta Propagation

Consider our running example

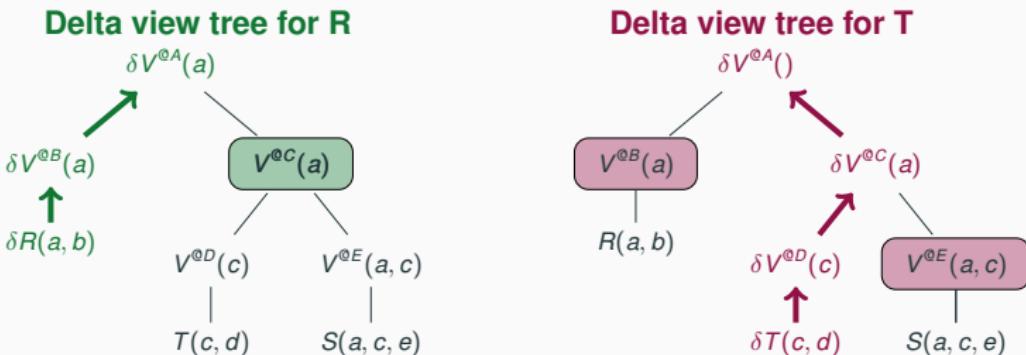
Maintain the query result under updates to  $T$



# Updates to Multiple Relations

Maintain the query result for updates to  $R$  and  $T$

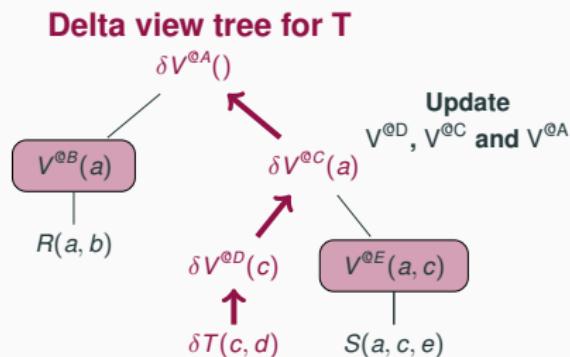
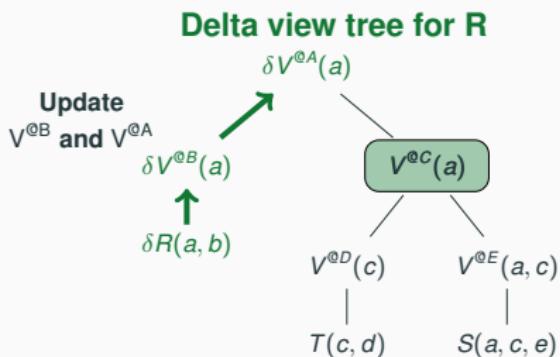
- Two delta propagation paths
- Both paths need to maintain auxiliary views



# Updates to Multiple Relations

Maintain the query result for updates to  $R$  and  $T$

- Two delta propagation paths
- Both paths need to maintain auxiliary views



# Landscape of Dynamic Query Evaluation

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay

**conjunctive**

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

static width  $w = \text{fhtw}$

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

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay

**conjunctive**

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

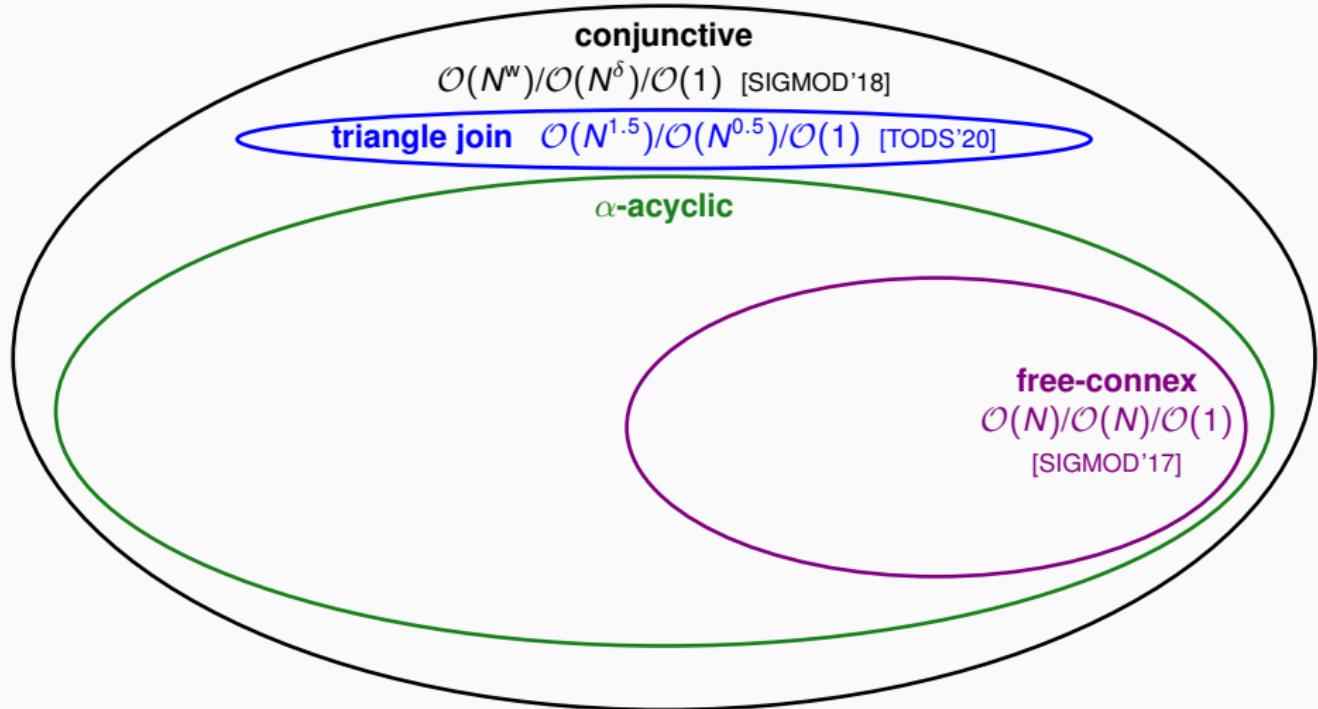
**triangle join**  $\mathcal{O}(N^{1.5})/\mathcal{O}(N^{0.5})/\mathcal{O}(1)$  [TODS'20]

static width  $w = \text{fhtw}$

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

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay

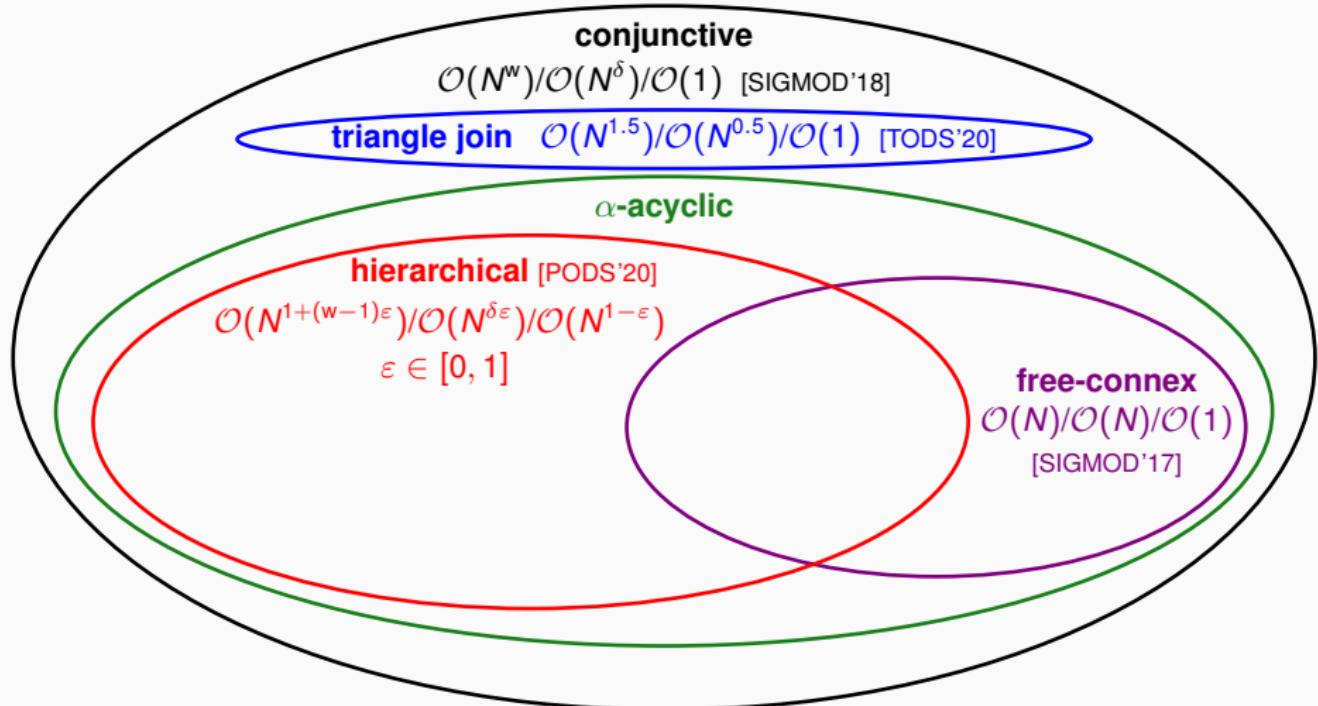


static width  $w = \text{fhtw}$

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

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay

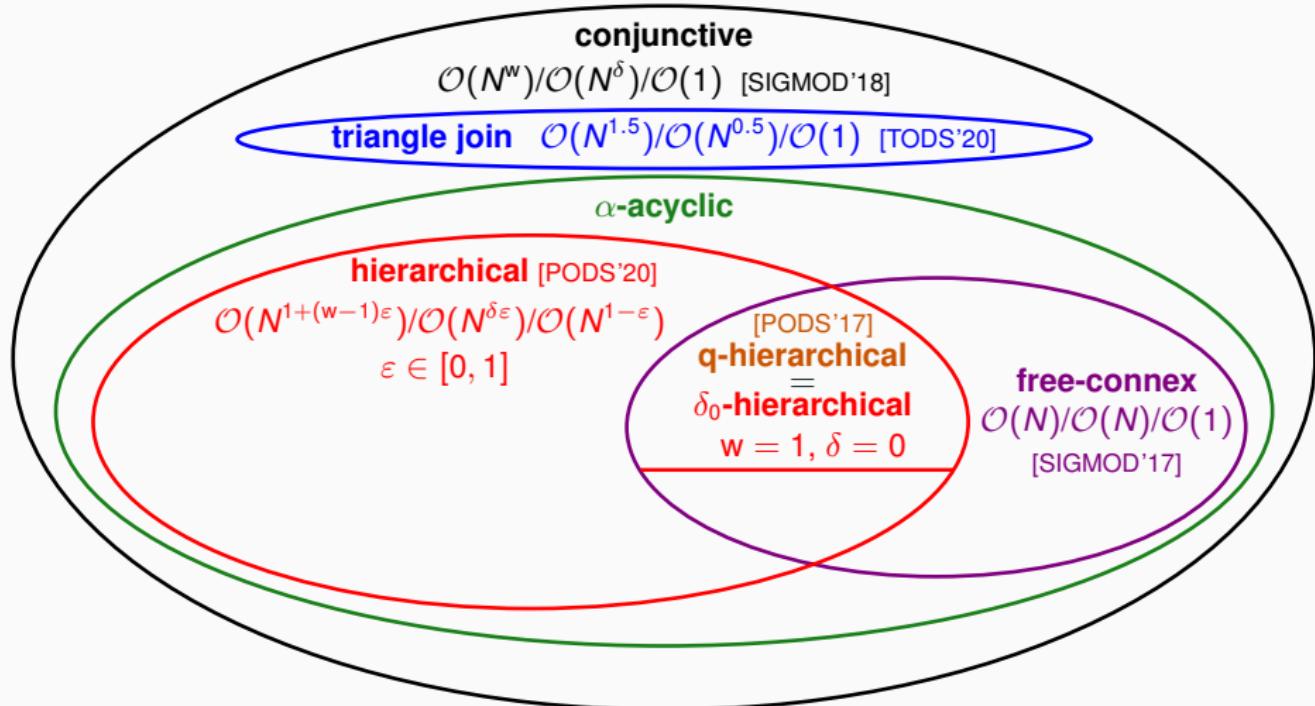


static width  $w = \text{fhtw}$

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

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay

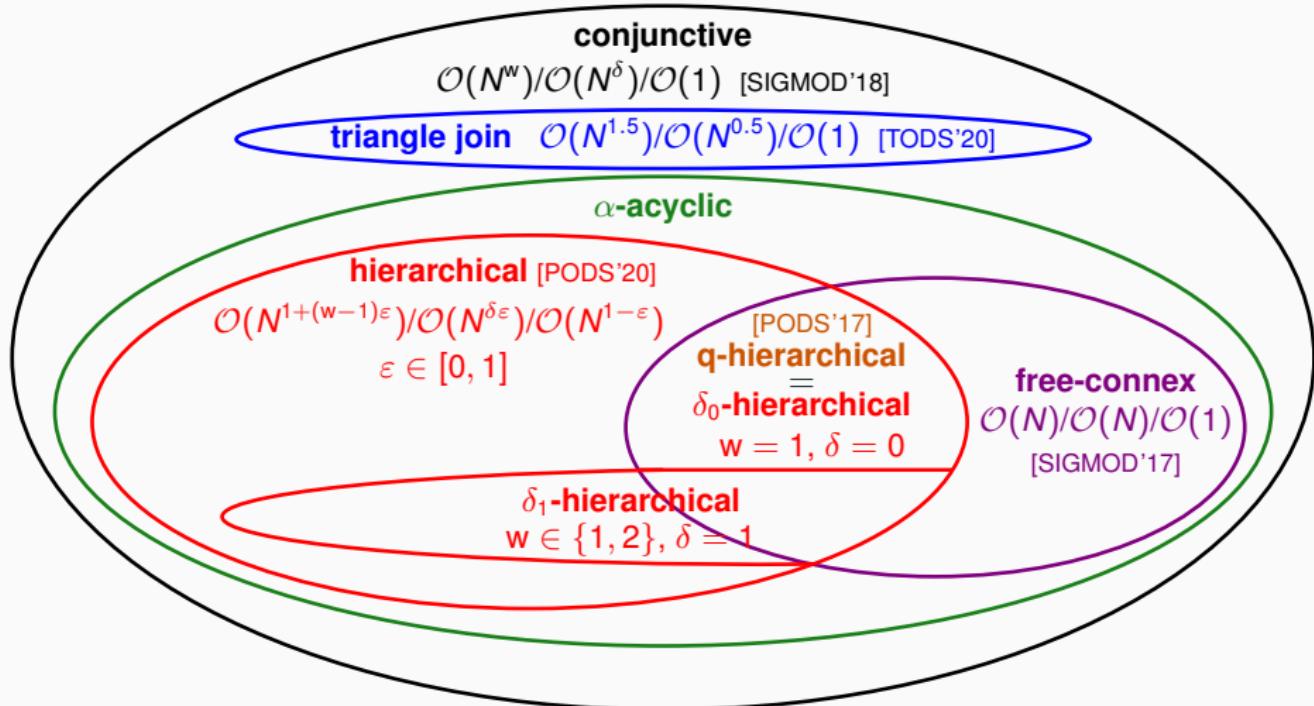


static width  $w = \text{fhtw}$

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

# Landscape of Dynamic Query Evaluation (Partial)

Preprocessing time/Update time/Enumeration delay



static width  $w = \text{fhtw}$

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

# Hierarchical Queries

## Hierarchical Queries

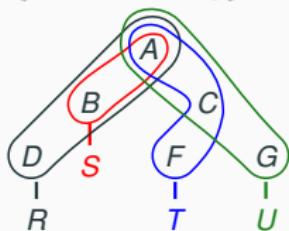
A query is **hierarchical** if for any two variables  $X, Y$ :

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

( $\partial(X)$  = the hyperedges containing  $X$ )

hierarchical

$$Q(\mathcal{F}) = \sum_{\mathcal{V} \setminus \mathcal{F}} R(a, b, d) \cdot \color{red}{S}(a, b) \cdot \color{blue}{T}(a, c, f) \cdot \color{green}{U}(a, c, g)$$
$$\mathcal{V} = \{a, b, c, d, e, f, g\}, \mathcal{F} \subseteq \mathcal{V}$$



## Hierarchical Queries

A query is **hierarchical** if for any two variables  $X, Y$ :

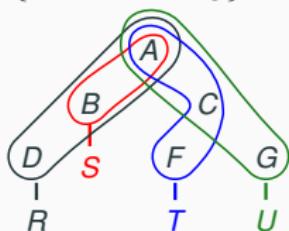
$$\partial(X) \subseteq \partial(Y) \text{ or } \partial(X) \supseteq \partial(Y) \text{ or } \partial(X) \cap \partial(Y) = \emptyset$$

( $\partial(X)$  = the hyperedges containing  $X$ )

hierarchical

$$Q(\mathcal{F}) = \sum_{\mathcal{V} \setminus \mathcal{F}} R(a, b, d) \cdot \color{red}{S}(a, b) \cdot \color{blue}{T}(a, c, f) \cdot \color{green}{U}(a, c, g)$$

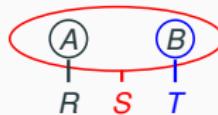
$$\mathcal{V} = \{a, b, c, d, e, f, g\}, \mathcal{F} \subseteq \mathcal{V}$$



not hierarchical

$$Q(\mathcal{F}) = \sum_{\mathcal{V} \setminus \mathcal{F}} R(a) \cdot \color{red}{S}(a, b) \cdot \color{blue}{T}(b)$$

$$\mathcal{V} = \{a, b\}, \mathcal{F} \subseteq \mathcal{V}$$

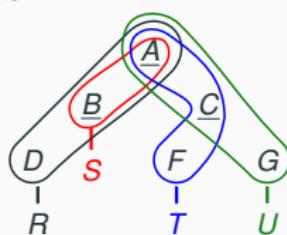


## $\delta_0$ -Hierarchical Queries

A hierarchical query is  $\delta_0$ -hierarchical if  
all free variables dominate the bound variables

$\delta_0$ -hierarchical

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



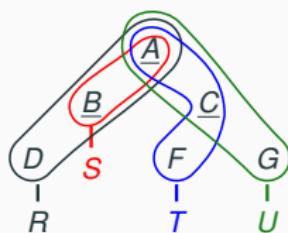
## $\delta_0$ -Hierarchical Queries

A hierarchical query is  $\delta_0$ -hierarchical if

all free variables dominate the bound variables

$\delta_0$ -hierarchical

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



hierarchical but not  
 $\delta_0$ -hierarchical

$$Q(a) = \sum_b \textcolor{red}{S}(a, b) \cdot \textcolor{blue}{T}(b)$$



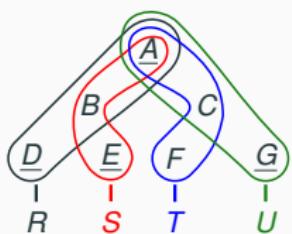
## $\delta_1$ -Hierarchical Queries

- For any bound variable  $X$  and any hyperedge  $S_1 \in \partial(X)$ , there is at most one other hyperedge  $S_2$  so that all free variables dominated by  $X$  are in  $S_1 \cup S_2$ .
- The query is not  $\delta_0$ -hierarchical

$\delta_1$ -hierarchical

$$Q(a, d, e, g) = \sum_{b,c,f} R(a, b, d) \cdot \textcolor{red}{S}(a, b, e) \cdot$$

$$\textcolor{blue}{T}(a, c, f) \cdot \textcolor{green}{U}(a, c, g)$$

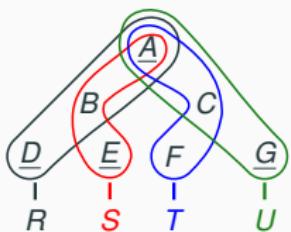


## $\delta_1$ -Hierarchical Queries

- For any bound variable  $X$  and any hyperedge  $S_1 \in \partial(X)$ , there is at most one other hyperedge  $S_2$  so that all free variables dominated by  $X$  are in  $S_1 \cup S_2$ .
- The query is not  $\delta_0$ -hierarchical

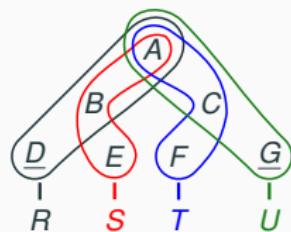
$\delta_1$ -hierarchical

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

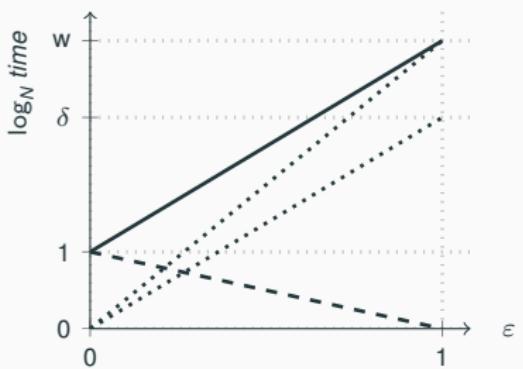


hierarchical but not  $\delta_1$ -hierarchical

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

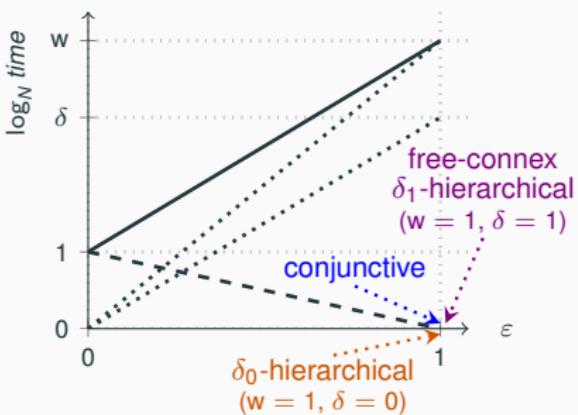


# Trade-Offs for Hierarchical Queries [PODS'20]



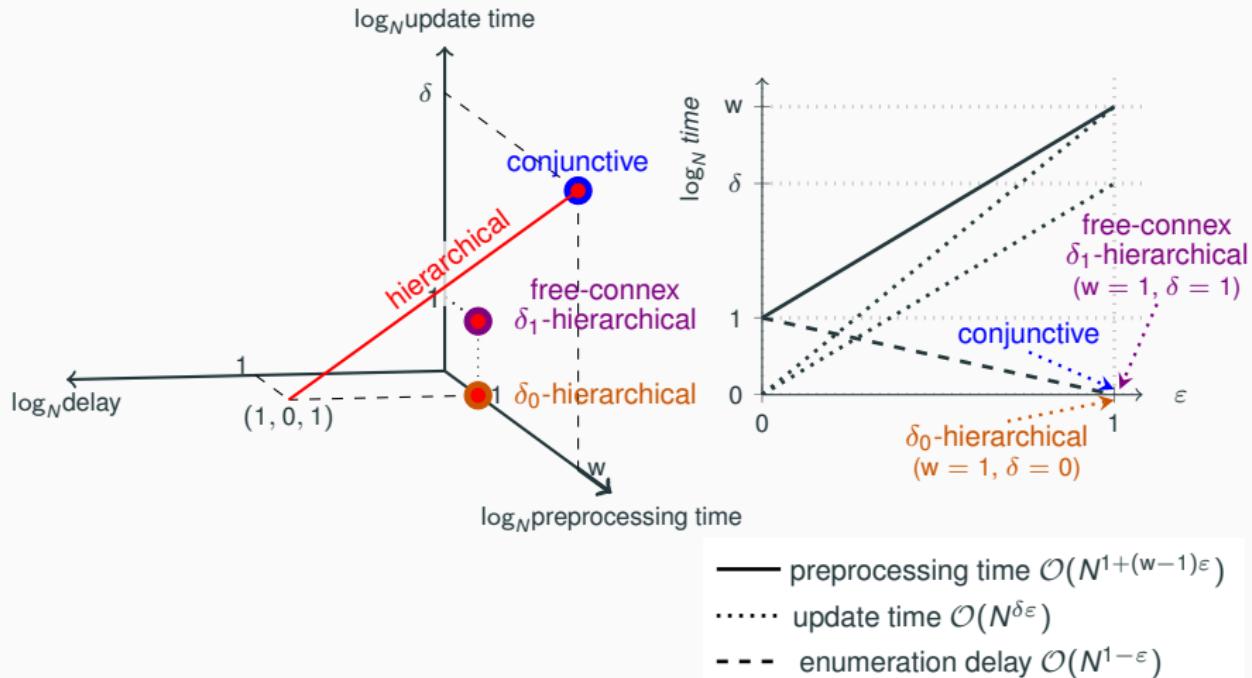
- preprocessing time  $\mathcal{O}(N^{1+(w-1)\varepsilon})$
- update time  $\mathcal{O}(N^{\delta\varepsilon})$
- - - enumeration delay  $\mathcal{O}(N^{1-\varepsilon})$

# Trade-Offs for Hierarchical Queries [PODS'20]

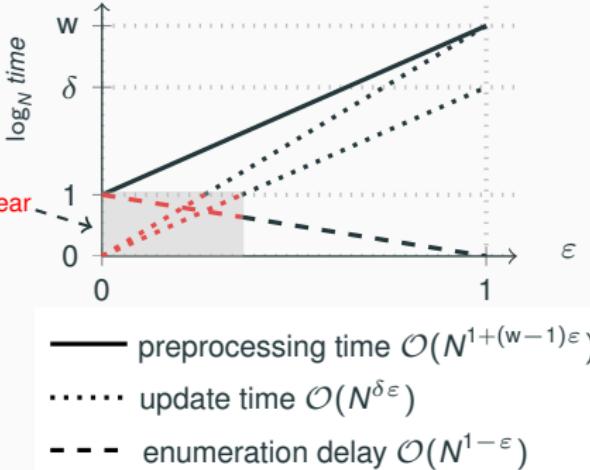
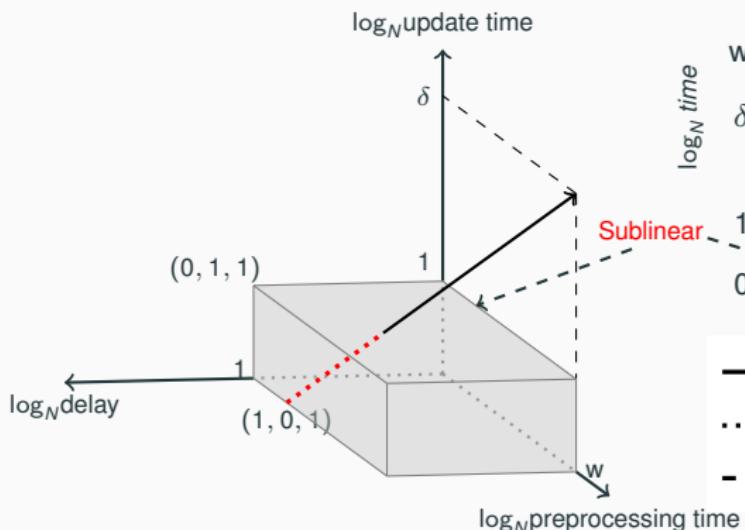


- preprocessing time  $\mathcal{O}(N^{1+(w-1)\varepsilon})$
- update time  $\mathcal{O}(N^{\delta\varepsilon})$
- - - enumeration delay  $\mathcal{O}(N^{1-\varepsilon})$

# Trade-Offs for Hierarchical Queries [PODS'20]



# Sublinear Update Time and Delay [PODS'20]



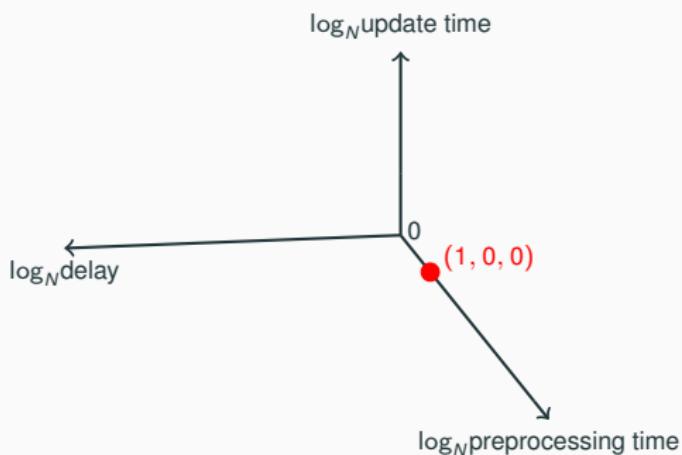
Hierarchical queries admit sublinear update time and enumeration delay

# **Dynamic Evaluation of $\delta_0$ -Hierarchical Queries**

## Dynamic Evaluation of $\delta_0$ -Hierarchical Queries [PODS'17]

Any  $\delta_0$ -hierarchical query can be maintained under single-tuple updates with

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



Using the **Online Matrix-Vector Multiplication (OMv)** Conjecture we can show:

The  $\delta_0$ -hierarchical queries are precisely the queries that admit  
**constant update time** and **constant enumeration delay**

### OMv Problem

We are given an  $n \times n$  Boolean matrix  $\mathbf{M}$  and receive  $n$  column vectors of size  $n$ , denoted by  $\mathbf{v}_1, \dots, \mathbf{v}_n$ , one by one; after seeing each vector  $\mathbf{v}_i$ , we output the product  $\mathbf{M}\mathbf{v}_i$  before we see the next vector  $\mathbf{v}_{i+1}$ .

Using the **Online Matrix-Vector Multiplication (OMv)** Conjecture we can show:

The  $\delta_0$ -hierarchical queries are precisely the queries that admit  
**constant update time** and **constant enumeration delay**

### OMv Problem

We are given an  $n \times n$  Boolean matrix  $\mathbf{M}$  and receive  $n$  column vectors of size  $n$ , denoted by  $\mathbf{v}_1, \dots, \mathbf{v}_n$ , one by one; after seeing each vector  $\mathbf{v}_i$ , we output the product  $\mathbf{M}\mathbf{v}_i$  before we see the next vector  $\mathbf{v}_{i+1}$ .

### OMv Conjecture [STOC'15]

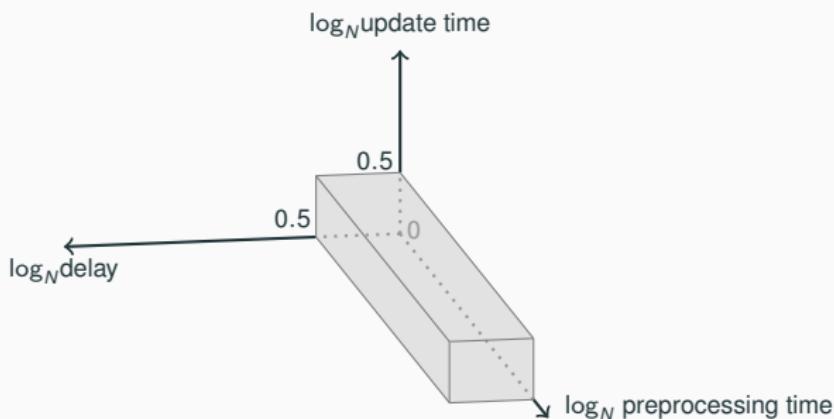
For any  $\gamma > 0$ , there is no algorithm that solves OMv in time  $\mathcal{O}(n^{3-\gamma})$ .

## Dichotomy by $\delta_0$ -Hierarchical Queries 2/2

- For any query that is not  $\delta_0$ -hierarchical, there is no algorithm that maintains the query under single-tuple updates with

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

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



## Dichotomy by $\delta_0$ -Hierarchical Queries 2/2

- For any query that is not  $\delta_0$ -hierarchical, there is no algorithm that maintains the query under single-tuple updates with

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

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

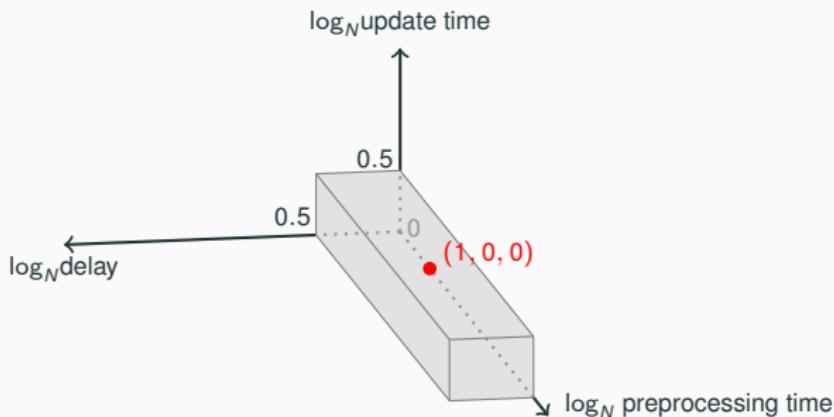
- Any  $\delta_0$ -hierarchical query can be maintained under single-tuple updates with

preprocessing time	update time	enumeration delay
--------------------	-------------	-------------------

$\mathcal{O}(N)$

$\mathcal{O}(1)$

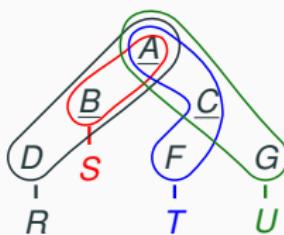
$\mathcal{O}(1)$



## Example: Dynamic Evaluation of a Simple $\delta_0$ -Hierarchical Query

Consider the following  $\delta_0$ -hierarchical query

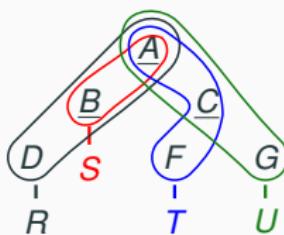
$$Q(a, b, c) = \sum_{d, e, f, g} R(a, b, d) \cdot S(a, b) \cdot T(a, c, f) \cdot U(a, c, g)$$



## Example: Dynamic Evaluation of a Simple $\delta_0$ -Hierarchical Query

Consider the following  $\delta_0$ -hierarchical query

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

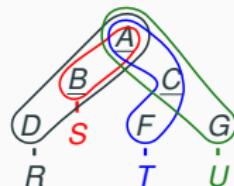


We construct in **linear time** a view tree that

- allows for **constant-delay** enumeration of the result of  $Q$ , and
- can be maintained in **constant time** under updates to all input factors.

## Example: Dynamic Evaluation of a Simple $\delta_0$ -Hierarchical Query

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$V_R(a, b) = \sum_d R(a, b, d)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

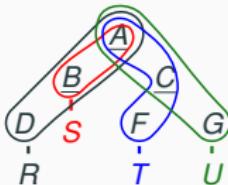
$$R(a, b, d)$$

$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Preprocessing

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$V_R(a, b) = \sum_d R(a, b, d)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

$$S(a, b)$$

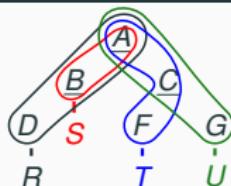
$$T(a, c, f)$$

$$U(a, c, g)$$

- Each view is the result of marginalizing a variable in a child view or joining two child views over the same schema  
 $\Rightarrow$  Each view can be computed in time  $\mathcal{O}(N)$

## Example: Enumeration

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$V_R(a, b) = \sum_d R(a, b, d)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

$$R(a, b, d)$$

$$T(a, c, f)$$

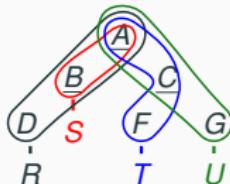
$$U(a, c, g)$$

The result of  $Q$  can be enumerated with constant delay:

- Iterate over the  $A$ -values in  $V_{RSTU}$ ;
- For each such  $A$ -value  $a$ , iterate over the  $B$ -values paired with  $a$  in  $V_{RS}$ ;
- For each such  $B$ -value  $b$ , iterate over the  $C$ -values  $c$  paired with  $b$  in  $V_{TU}$ ;
- Output  $(a, b, c)$ .

## Example: Updates

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$V_R(a, b) = \sum_d R(a, b, d)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

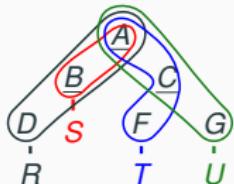
$$R(a, b, d)$$

$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Updates

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$V_R(a, b) = \sum_d R(a, b, d)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

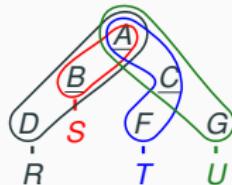
$$\delta R(a', b', d')$$

$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Updates

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

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

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$\delta V_R(a', b') = \delta R(a', b', d') \quad S(a, b)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

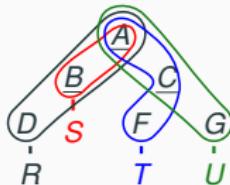
$$\delta R(a', b', d')$$

$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Updates

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



View tree

$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$V'_{RS}(a) = \sum_b V_{RS}(a, b)$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

$$\delta V_{RS}(a', b') = \delta V_R(a', b') \cdot S(a', b')$$

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$\delta V_R(a', b') = \delta R(a', b', d') \quad S(a, b)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

$$\delta R(a', b', d')$$

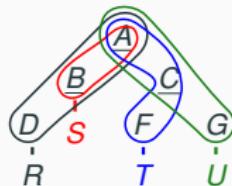
$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Updates

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

View tree



$$V_{RSTU}(a) = V'_{RS}(a) \cdot V'_{TU}(a)$$

$$\delta V'_{RS}(a') = \delta V_{RS}(a', b')$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

$$\delta V_{RS}(a', b') = \delta V_R(a', b') \cdot S(a', b')$$

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$\delta V_R(a', b') = \delta R(a', b', d') \quad S(a, b)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

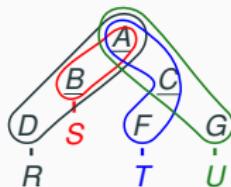
$$\delta R(a', b', d')$$

$$T(a, c, f)$$

$$U(a, c, g)$$

## Example: Updates

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



View tree

$$\delta V_{RSTU}(a') = \delta V'_{RS}(a') \cdot V'_{TU}(a')$$

$$\delta V'_{RS}(a') = \delta V_{RS}(a', b')$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

$$\delta V_{RS}(a', b') = \delta V_R(a', b') \cdot S(a', b')$$

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$\delta V_R(a', b') = \delta R(a', b', d') \quad S(a, b)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

$$V_U(a, c) = \sum_g U(a, c, g)$$

$$\delta R(a', b', d')$$

$$T(a, c, f)$$

$$U(a, c, g)$$

- Updates to  $R$ : Computation of each delta view requires at most one lookup  
 $\Rightarrow$  Update time:  $\mathcal{O}(1)$

## Example: Updates

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

View tree

$$\delta V_{RSTU}(a') = \delta V'_{RS}(a') \cdot V'_{TU}(a')$$

$$\delta V'_{RS}(a') = \delta V_{RS}(a', b')$$

$$V'_{TU}(a) = \sum_c V_{TU}(a, c)$$

$$\delta V_{RS}(a', b') = \delta V_R(a', b') \cdot S(a', b')$$

$$V_{TU}(a, c) = V_T(a, c) \cdot V_U(a, c)$$

$$\delta V_R(a', b') = \delta R(a', b', d') \quad S(a, b)$$

$$V_T(a, c) = \sum_f T(a, c, f)$$

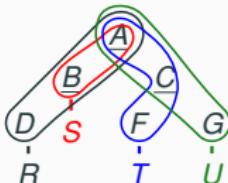
$$V_U(a, c) = \sum_g U(a, c, g)$$

$$\delta R(a', b', d')$$

$$T(a, c, f)$$

$$U(a, c, g)$$

- Updates to  $R$ : Computation of each delta view requires at most one lookup  
 $\Rightarrow$  Update time:  $\mathcal{O}(1)$
- Updates to the other factors: analogous

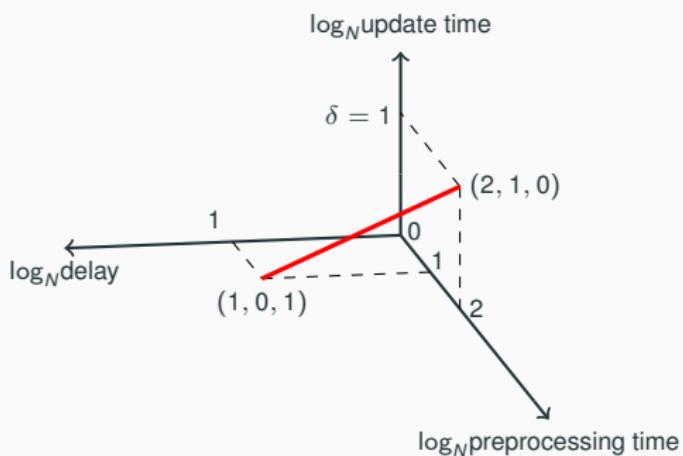


# **Dynamic Evaluation of $\delta_1$ -Hierarchical Queries**

# Dynamic Evaluation of $\delta_1$ -Hierarchical Queries [PODS'20]

Any  $\delta_1$ -hierarchical query can be maintained under single-tuple updates with

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

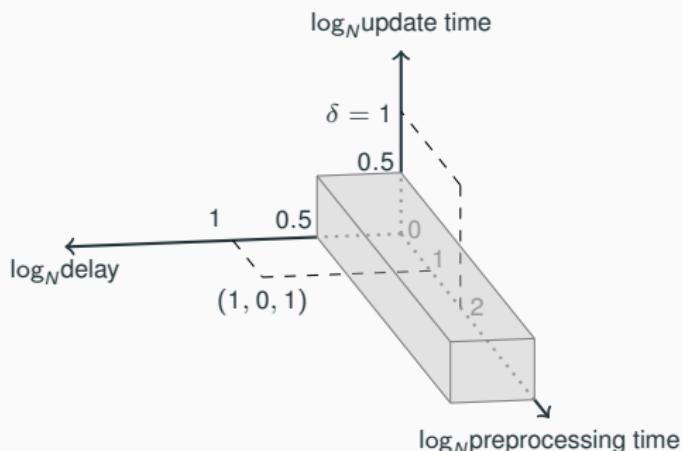


## Optimality for $\delta_1$ -Hierarchical Queries

- For any  $\delta_1$ -hierarchical query, there is no algorithm that maintains the query under single-tuple updates with

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

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



## Optimality for $\delta_1$ -Hierarchical Queries

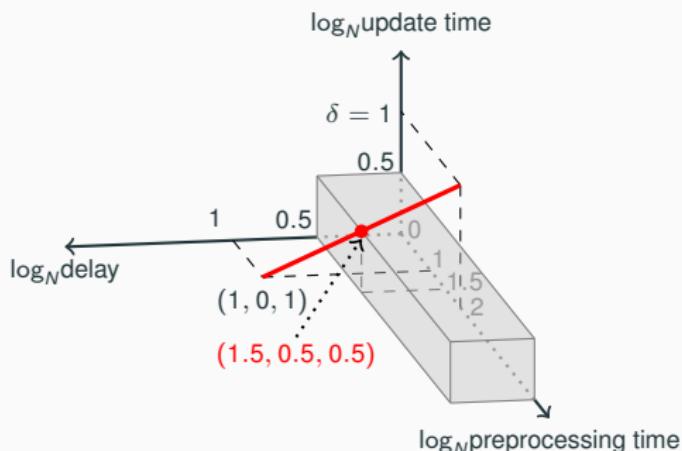
- For any  $\delta_1$ -hierarchical query, there is no algorithm that maintains the query under single-tuple updates with

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

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

- Any  $\delta_1$ -hierarchical query can be maintained under single-tuple updates with

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



## Optimality for $\delta_1$ -Hierarchical Queries

- For any  $\delta_1$ -hierarchical query, there is no algorithm that maintains the query under single-tuple updates with

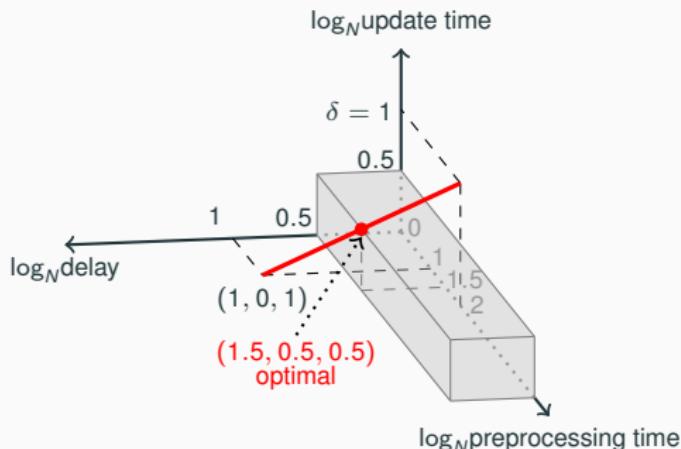
preprocessing time	update time	enumeration delay
arbitrary	$\mathcal{O}(N^{0.5-\gamma})$	$\mathcal{O}(N^{0.5-\gamma})$

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

- Any  $\delta_1$ -hierarchical query can be maintained under single-tuple updates with

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

$\implies$  For  $\varepsilon = 0.5$ , this is weakly Pareto optimal, unless OMv Conjecture fails



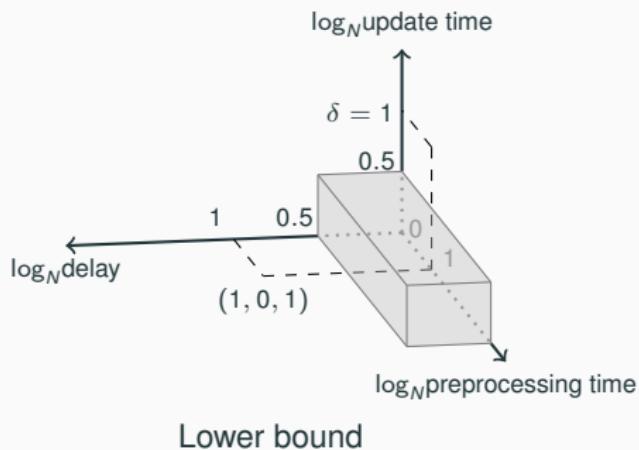
## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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



## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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



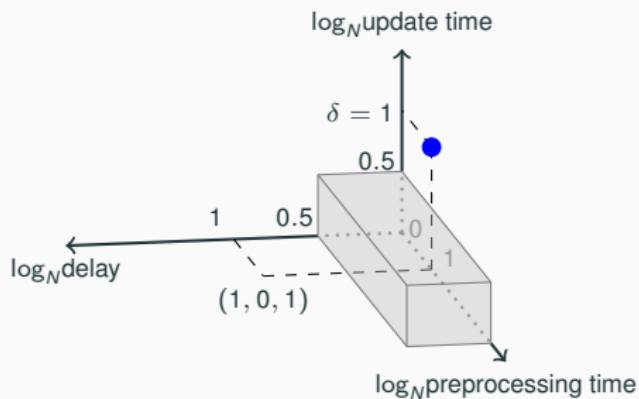
For this query, there is no algorithm that admits

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

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

## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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

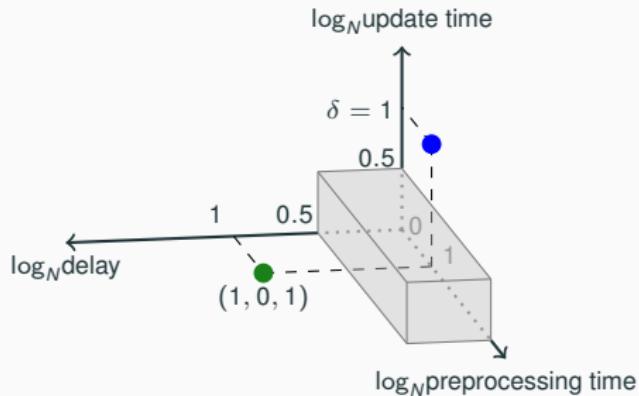


Known approach: Eager update, quick enumeration

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

## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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

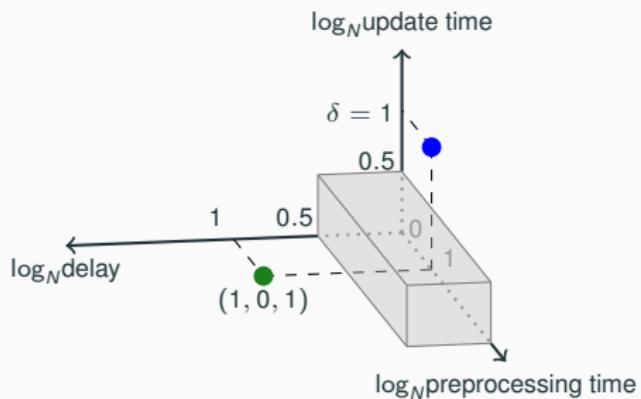


Known approach: Lazy update, heavy enumeration

- Preprocessing: Eliminate dangling tuples
- Upon update: Update only input factors
- Enumeration: Eliminate dangling tuples and enumerate from  $R$

## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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

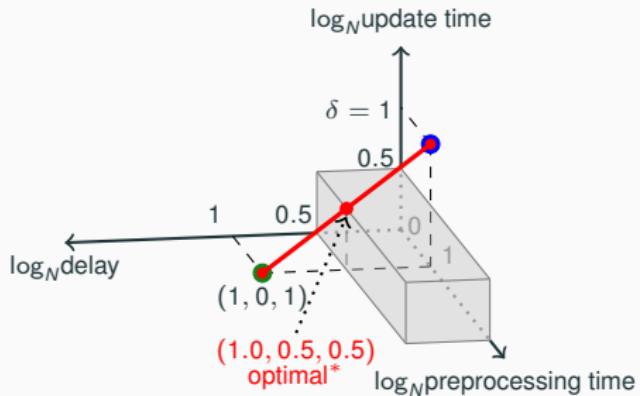


### Question

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

## Example: Dynamic Evaluation of a Simple $\delta_1$ -Hierarchical Query

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



(\*): Weak Pareto optimality by OMv Conjecture

The query  $Q$  can be maintained with

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

## Factor Partitioning

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



## Factor Partitioning

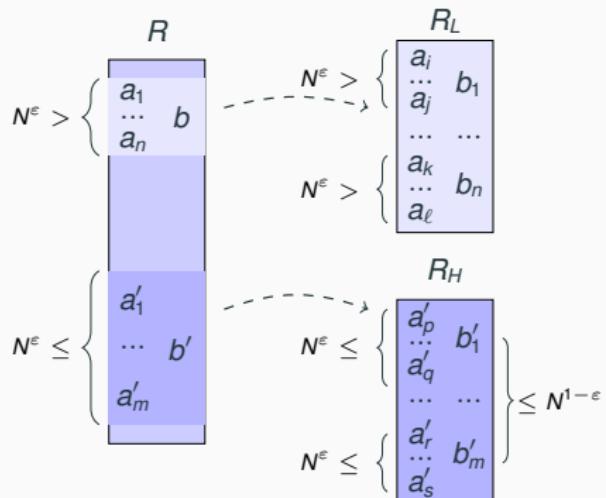
$$Q(a) = \sum_b R(a, b) \cdot S(b)$$



Partition  $R$  based on the B-values into a **light part**  $R_L$  and a **heavy part**  $R_H$ :

$$R_L(a, b) = \begin{cases} R(a, b) & \text{if } \text{degree}(b) < N^\varepsilon \\ 0 & \text{otherwise} \end{cases} \quad R_H(a, b) = \begin{cases} R(a, b) & \text{if } \text{degree}(b) \geq N^\varepsilon \\ 0 & \text{otherwise} \end{cases}$$

$\text{degree}(b)$ : number  $A$ -values  $a'$  such that  $R(a', b) \neq 0$



# Factor Partitioning

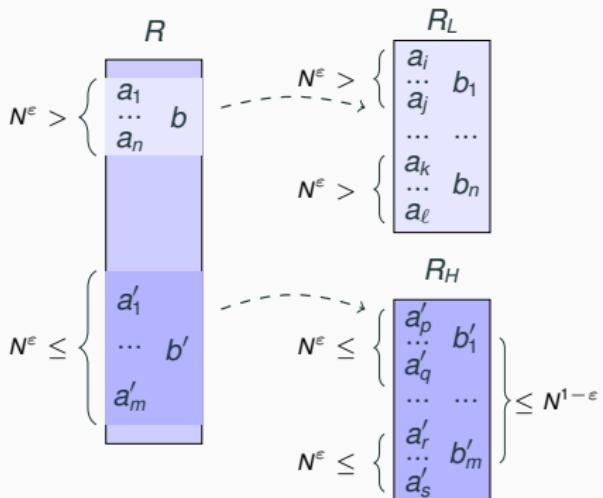
$$Q(a) = \sum_b R(a, b) \cdot S(b)$$



Partition  $R$  based on the B-values into a **light part**  $R_L$  and a **heavy part**  $R_H$ :

$$R_L(a, b) = \begin{cases} R(a, b) & \text{if } \text{degree}(b) < N^\varepsilon \\ 0 & \text{otherwise} \end{cases} \quad R_H(a, b) = \begin{cases} R(a, b) & \text{if } \text{degree}(b) \geq N^\varepsilon \\ 0 & \text{otherwise} \end{cases}$$

$\text{degree}(b)$ : number  $A$ -values  $a'$  such that  $R(a', b) \neq 0$



$$Q(a) = Q_L(a) + Q_H(a)$$

where

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

# Light Case

## Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$

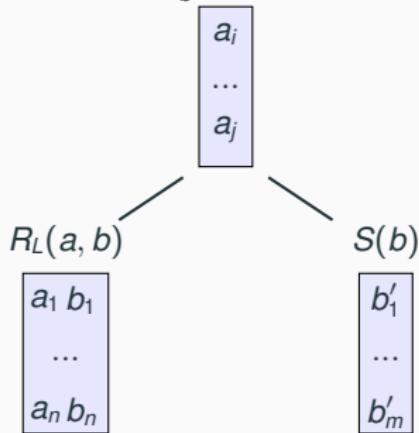
*Materialize the result*

## Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$

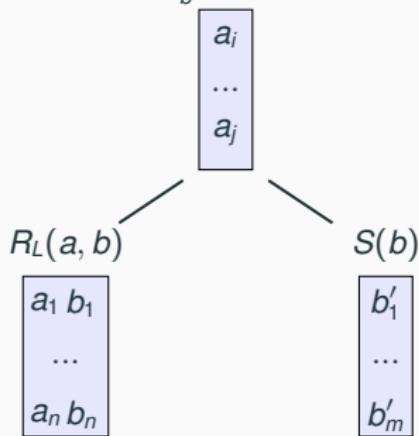
Materialize the result

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



## Preprocessing in the Light Case

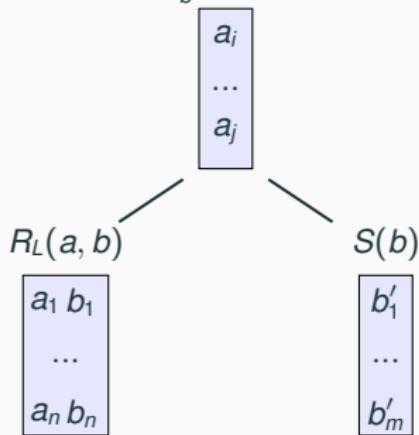
$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



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

## Enumeration in the Light Case

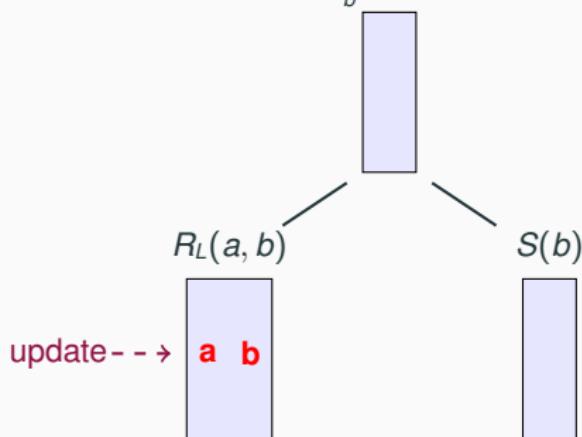
$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



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

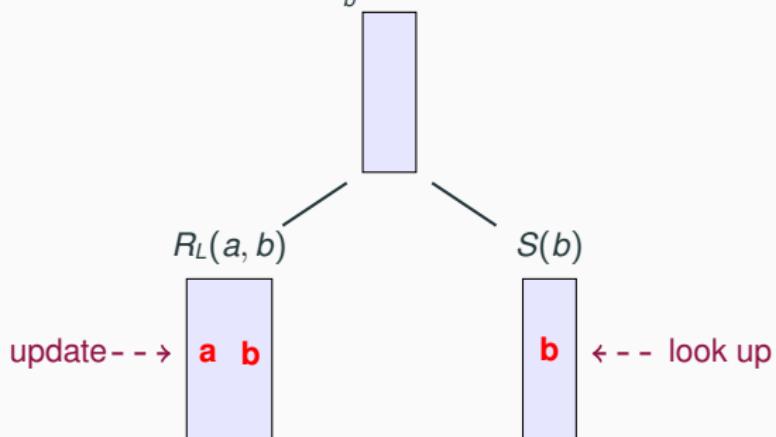
## Updates in the Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



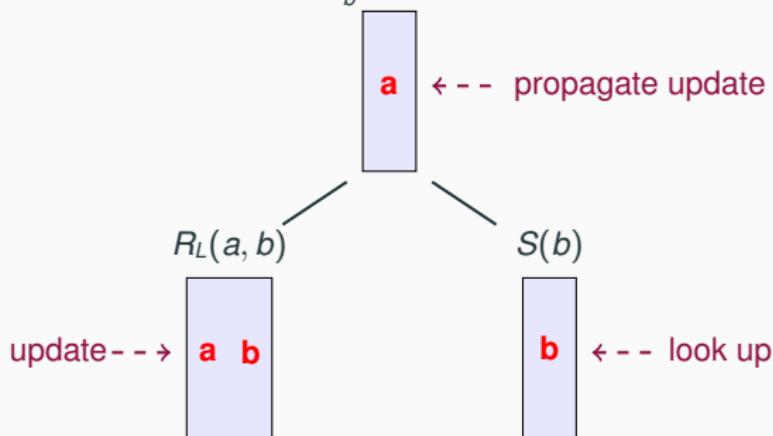
## Updates in the Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



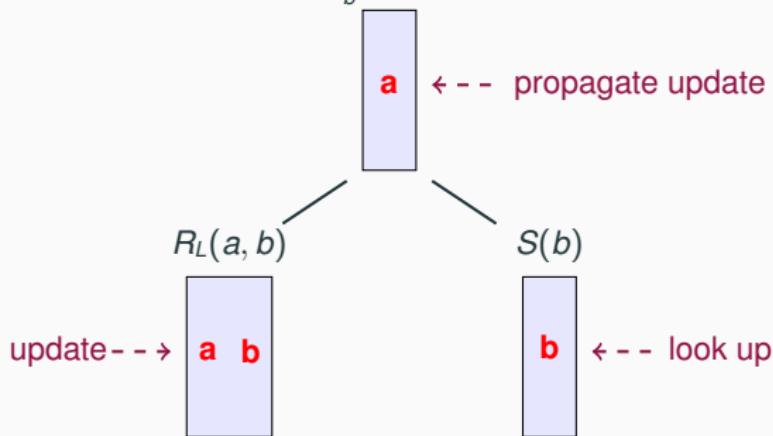
## Updates in the Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



## Updates in the Light Case

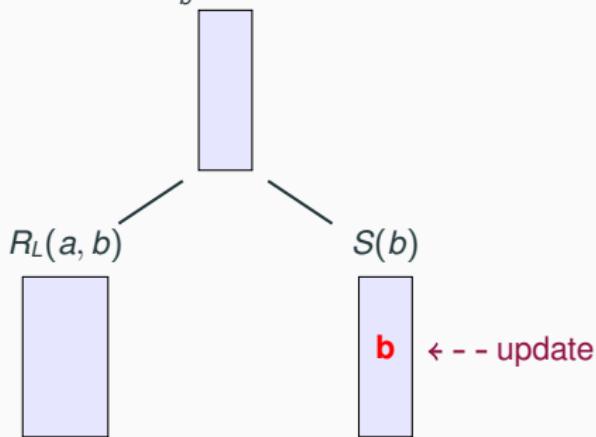
$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



- Updates to  $R_L$ :  $\mathcal{O}(1)$

## Updates in the Light Case

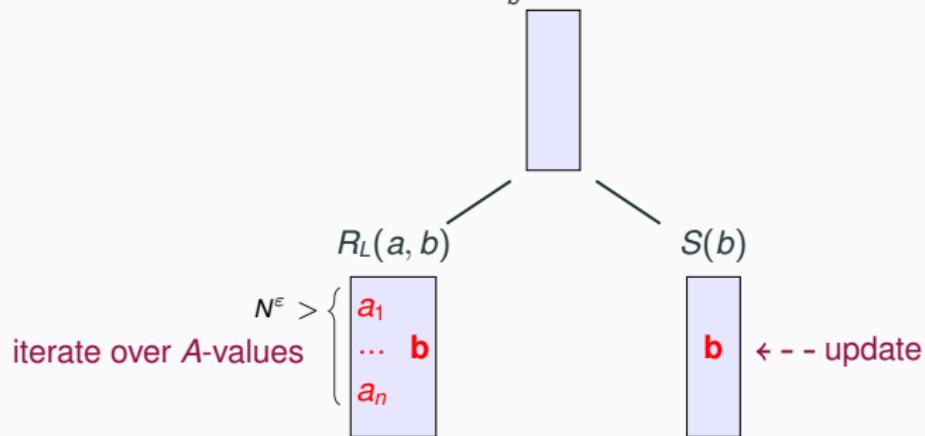
$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



- Updates to  $R_L$ :  $\mathcal{O}(1)$

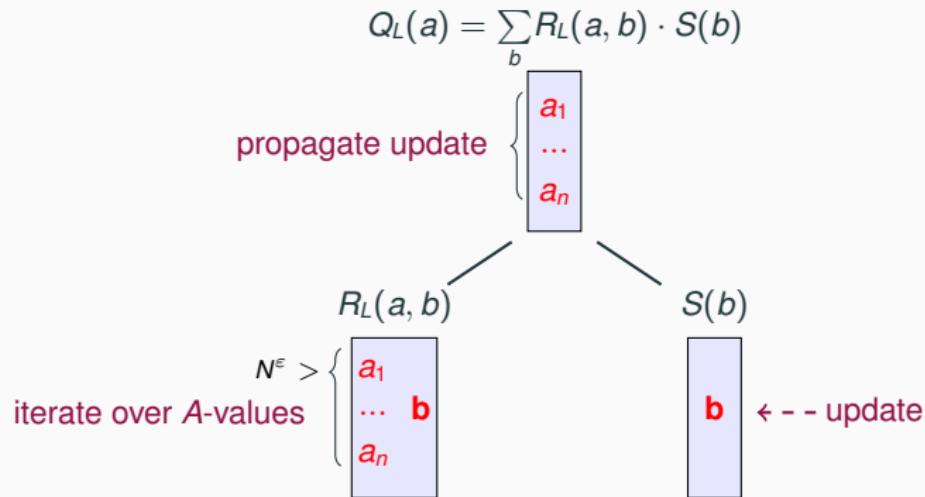
## Updates in the Light Case

$$Q_L(a) = \sum_b R_L(a, b) \cdot S(b)$$



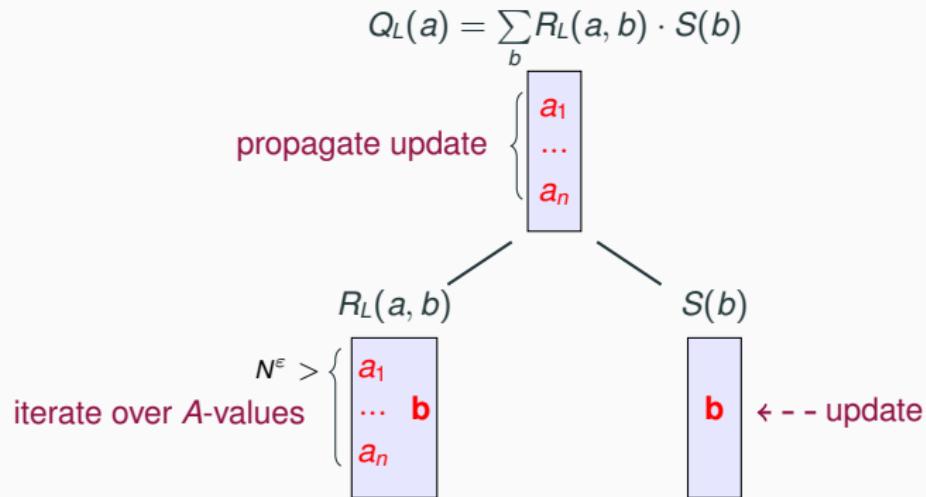
- Updates to  $R_L$ :  $\mathcal{O}(1)$

## 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

## Heavy Case

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

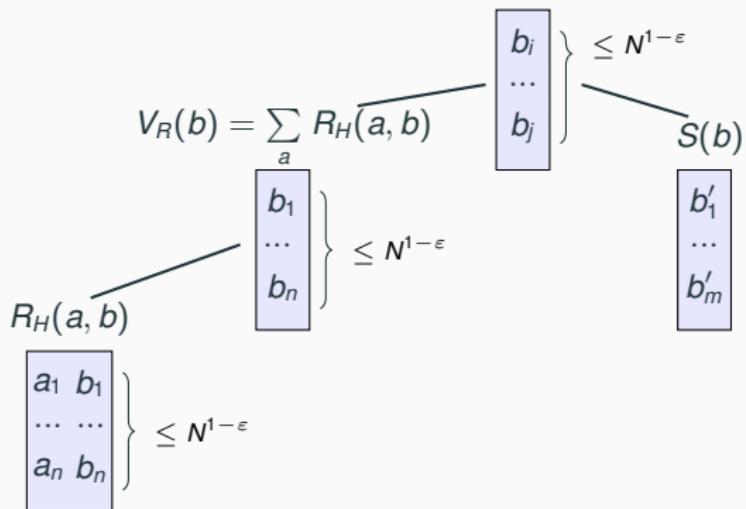
*Materialize the projection of the join result onto B*

## Heavy Case

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

Materialize the projection of the join result onto B

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



## Preprocessing in the Heavy Case

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

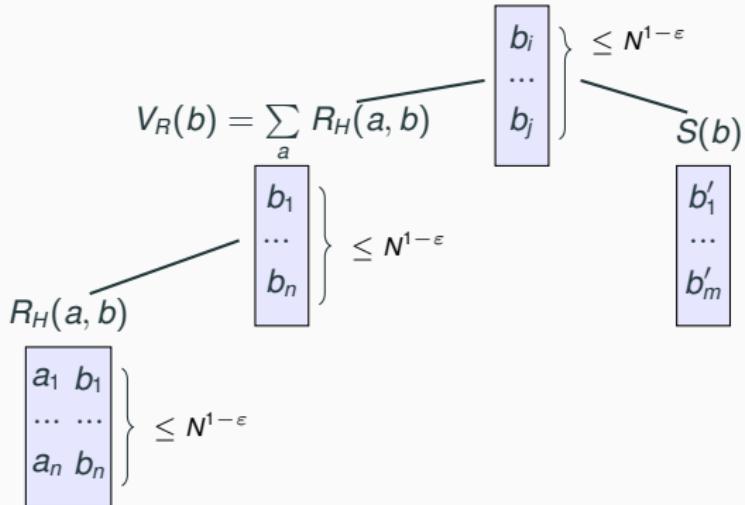
$$V_{RS}(b) = V_R(b) \cdot S(b)$$
$$V_R(b) = \sum_a R_H(a, b)$$
$$R_H(a, b)$$
$$\left\{ \begin{matrix} a_1 & b_1 \\ \dots & \dots \\ a_n & b_n \end{matrix} \right\} \leq N^{1-\varepsilon}$$
$$\left\{ \begin{matrix} b_1 \\ \dots \\ b_n \end{matrix} \right\} \leq N^{1-\varepsilon}$$
$$\left\{ \begin{matrix} b_i \\ \dots \\ b_j \end{matrix} \right\} \leq N^{1-\varepsilon}$$
$$S(b)$$
$$\left\{ \begin{matrix} b'_1 \\ \dots \\ b'_m \end{matrix} \right\} \leq N^{1-\varepsilon}$$

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

## Enumeration in the Heavy Case

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

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

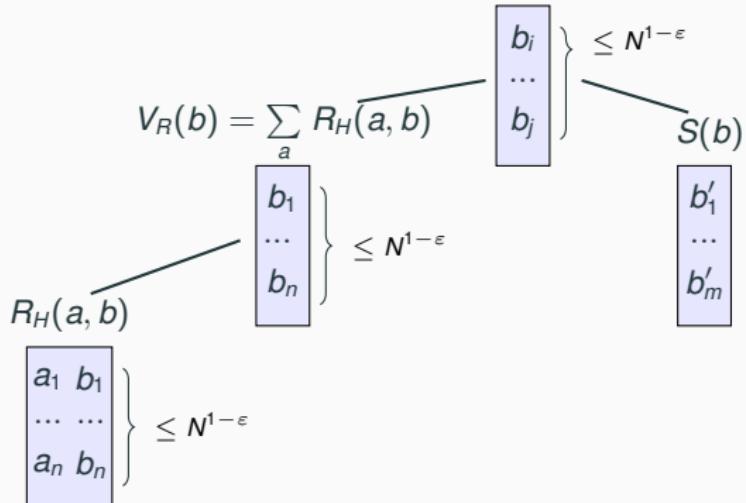


- $V_{RS}$  contains at most  $N^{1-\varepsilon}$   $B$ -values
- For each  $B$ -value  $b$  in  $V_{RS}$ , the  $A$ -values in  $R_H$  paired with  $b$  admit constant enumeration delay

## Enumeration in the Heavy Case

$$Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$$

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



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

Next: How can the distinct  $A$ -values in  $Q_H$  be enumerated with  $\mathcal{O}(N^{1-\varepsilon})$  delay

## Enumeration of Distinct $A$ -Values

- Properties of the data structure for  $Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$ :
  - $V_{RS}(b) = V_R(b) \cdot (b)$  contains at most  $N^{1-\varepsilon}$   $B$ -values
  - For each  $B$ -value  $b$  in  $V_{RS}$ , the  $A$ -values in  $R_H$  paired with  $b$  admit constant enumeration delay

## Enumeration of Distinct $A$ -Values

- Properties of the data structure for  $Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$ :
  - $V_{RS}(b) = V_R(b) \cdot (b)$  contains at most  $N^{1-\varepsilon}$   $B$ -values
  - For each  $B$ -value  $b$  in  $V_{RS}$ , the  $A$ -values in  $R_H$  paired with  $b$  admit constant enumeration delay
- Attention:** For two distinct  $b_1$  and  $b_2$ , the  $A$ -values in  $R_H$  paired with  $b_1$  and those paired with  $b_2$  might not be disjoint  
⇒ Enumerating first the  $A$ -values paired with  $b_1$  and then those paired with  $b_2$  (or vice-versa) can lead to duplicates in the output

## Enumeration of Distinct $A$ -Values

- Properties of the data structure for  $Q_H(a) = \sum_b R_H(a, b) \cdot S(b)$ :
  - $V_{RS}(b) = V_R(b) \cdot (b)$  contains at most  $N^{1-\varepsilon}$   $B$ -values
  - For each  $B$ -value  $b$  in  $V_{RS}$ , the  $A$ -values in  $R_H$  paired with  $b$  admit constant enumeration delay
- Attention:** For two distinct  $b_1$  and  $b_2$ , the  $A$ -values in  $R_H$  paired with  $b_1$  and those paired with  $b_2$  might not be disjoint  
⇒ Enumerating first the  $A$ -values paired with  $b_1$  and then those paired with  $b_2$  (or vice-versa) can lead to duplicates in the output
- Using the **union algorithm** [CSL'11], we can enumerate the distinct  $A$ -values with  $\mathcal{O}(N^{1-\varepsilon})$  delay.

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.

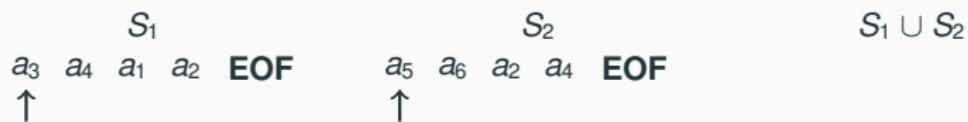
$S_1$	$S_2$	$S_1 \cup S_2$
$a_3 \ a_4 \ a_1 \ a_2 \ \text{EOF}$	$a_5 \ a_6 \ a_2 \ a_4 \ \text{EOF}$	

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



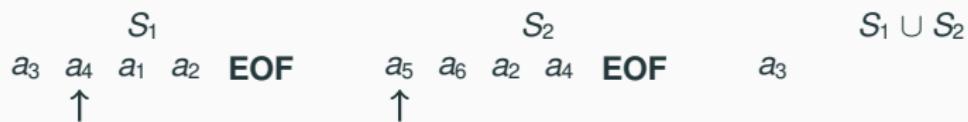
$a_3$  is not included in  $S_2$ , so output from  $S_1$  and move first pointer

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



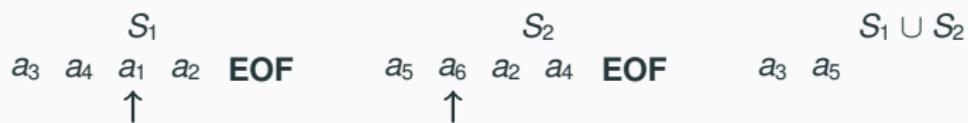
$a_4$  is included in  $S_2$ , so output from  $S_2$  and move both pointers

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



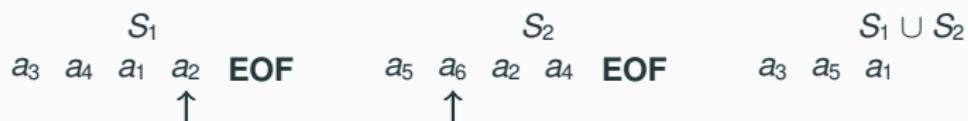
$a_1$  is not included in  $S_2$ , so output from  $S_1$  and move first pointer

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



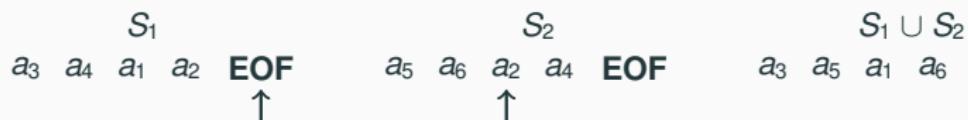
$a_2$  is included in  $S_2$ , so output from  $S_2$  and move both pointers

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



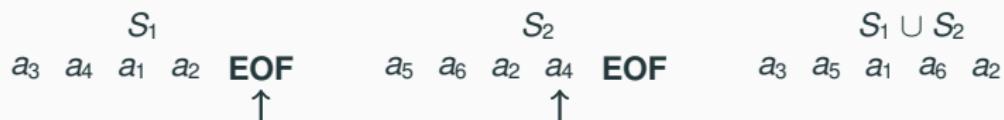
$S_1$  is exhausted, so enumerate from  $S_2$

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.



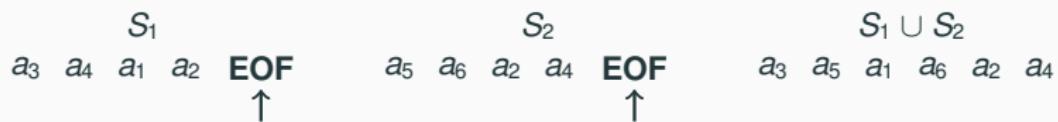
$S_1$  is exhausted, so enumerate from  $S_2$

## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.

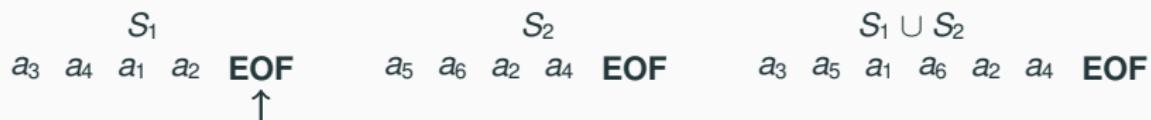


## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.

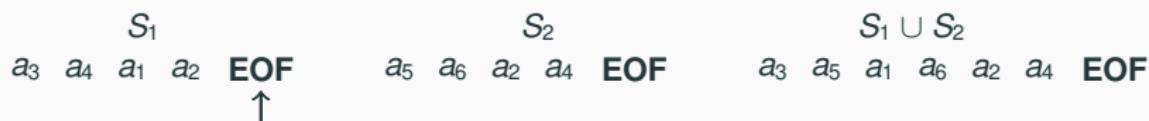


## Union Algorithm

### Enumeration of the Union of Two Sets

Assume, both sets allow lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the two sets can be enumerated with  $\mathcal{O}(\ell + d)$  delay.

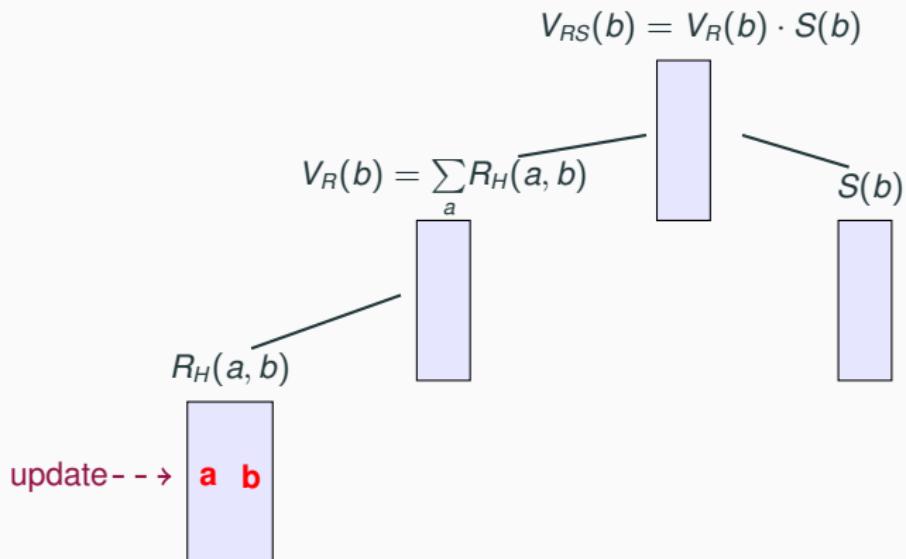


### Generalization: Enumeration of the Union of $n > 2$ Sets

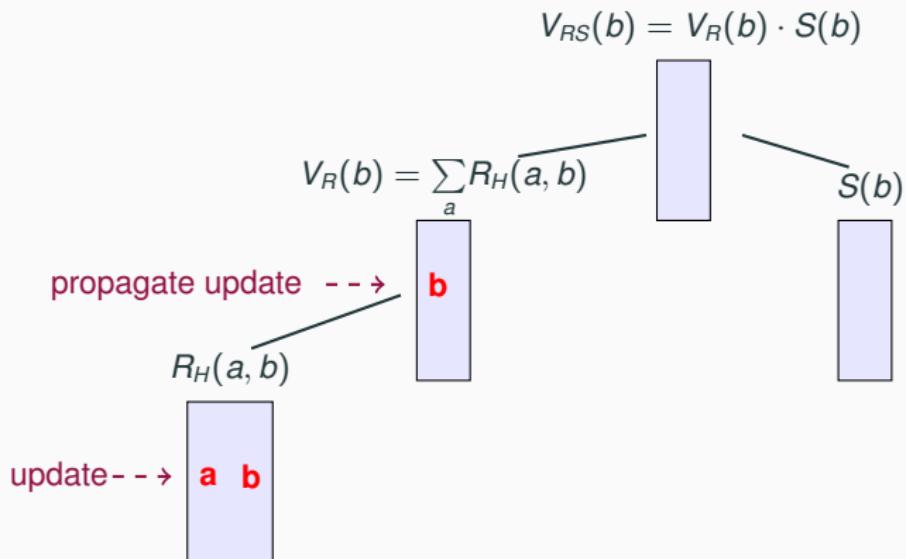
Assume, each set allows lookup time  $\ell$  and enumeration delay  $d$ .

⇒ The distinct elements in the union of the sets can be enumerated with  $\mathcal{O}(n(\ell + d))$  delay.

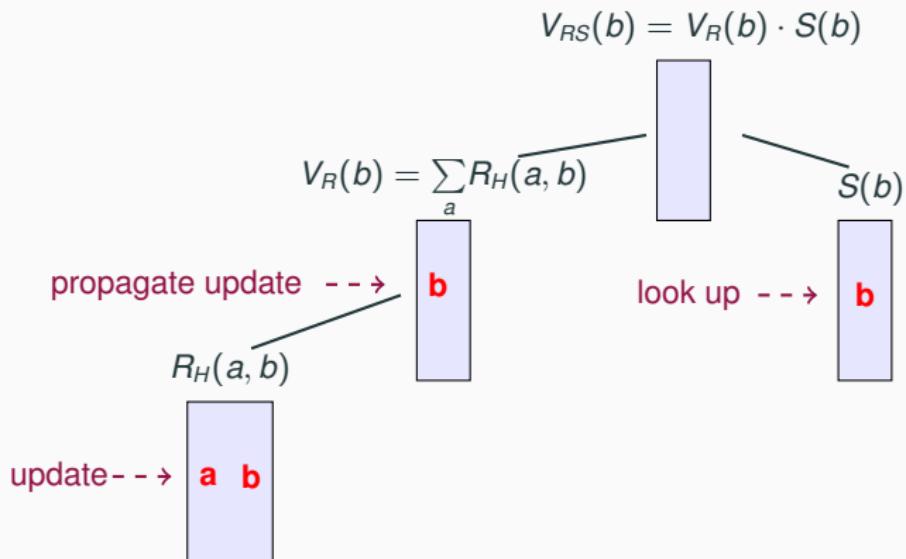
## Updates in the Heavy Case



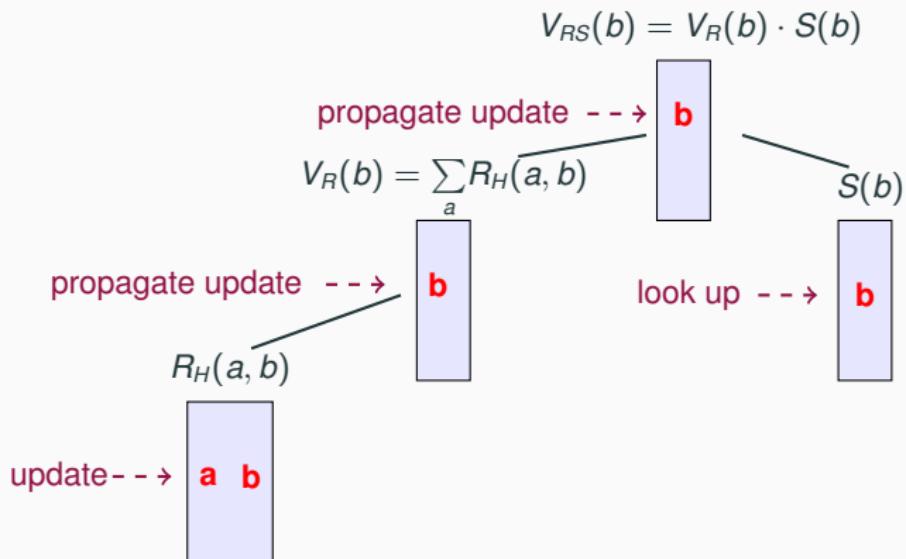
## Updates in the Heavy Case



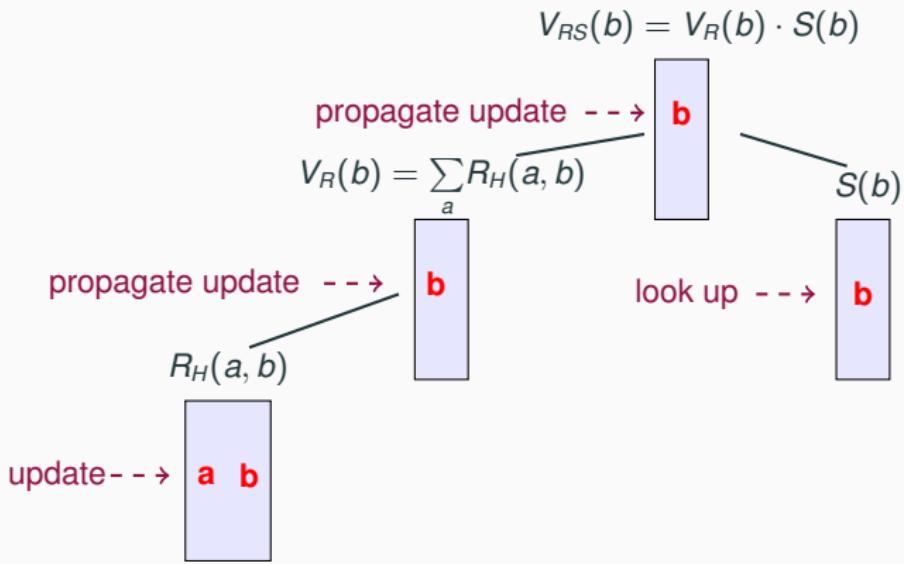
## Updates in the Heavy Case



## Updates in the Heavy Case

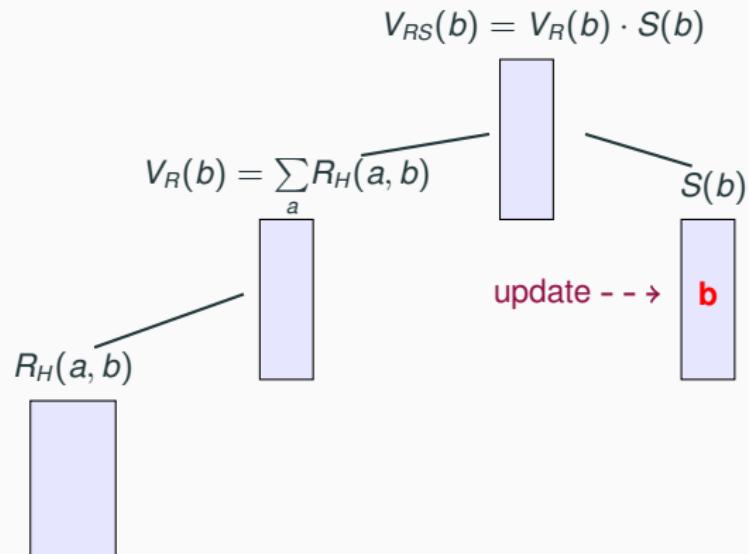


## Updates in the Heavy Case



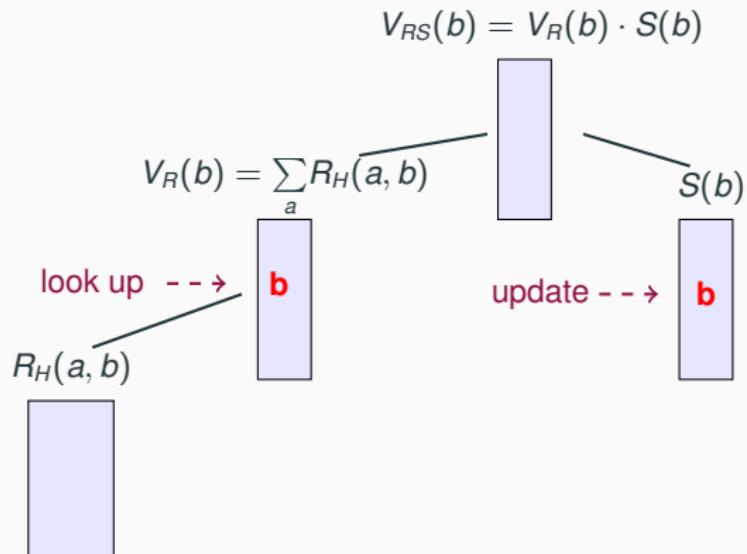
- Updates to  $R_H$ :  $\mathcal{O}(1)$

## Updates in the Heavy Case



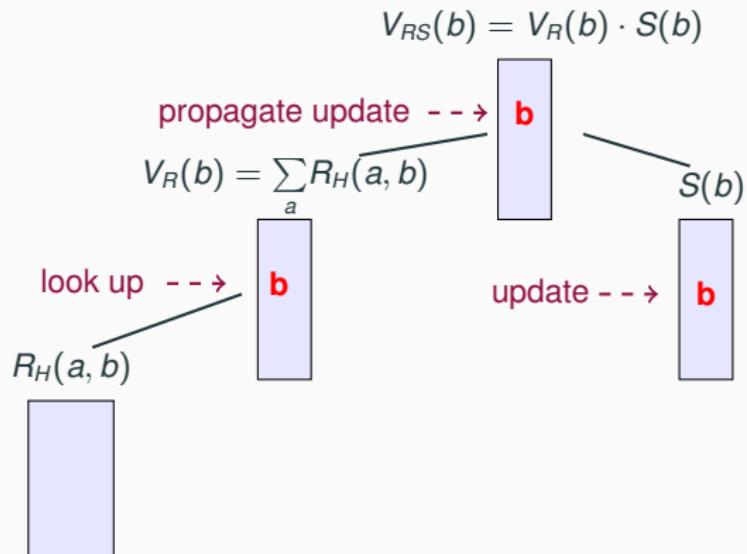
- Updates to  $R_H$ :  $\mathcal{O}(1)$

## Updates in the Heavy Case



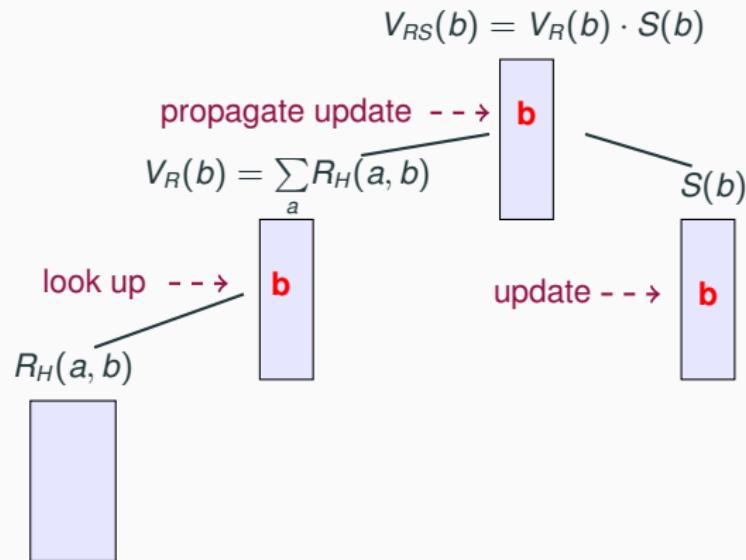
- Updates to  $R_H$ :  $\mathcal{O}(1)$

## 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  $S$ :  $\mathcal{O}(1)$

# **Example: Summing Up**

## Summing Up

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



Preprocessing Time

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

## Summing Up

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



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

## Summing Up

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



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

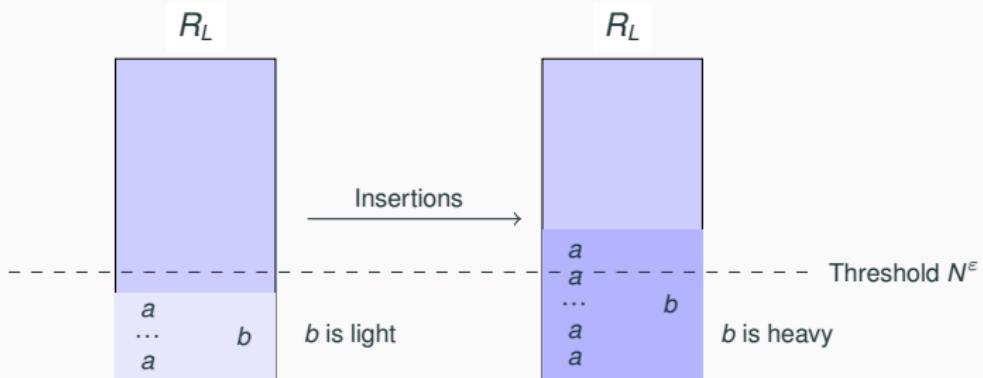
Update Time

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

# Rebalancing Partitions

## Rebalancing Partitions

Updates can change the frequencies of values in the factor parts!

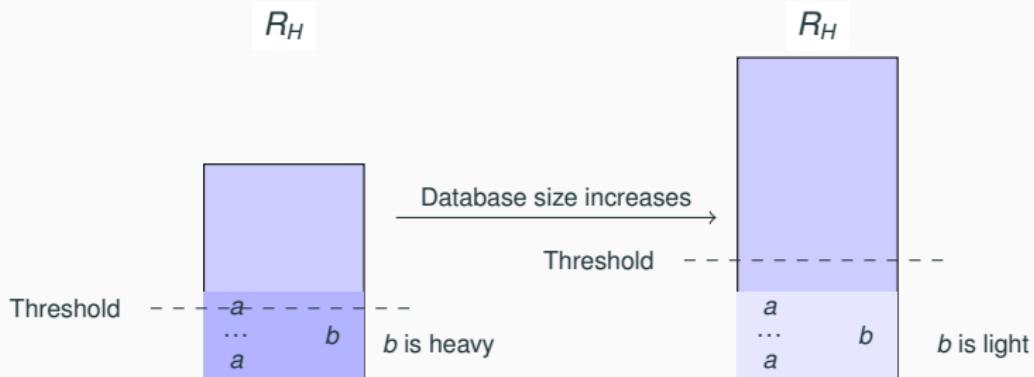


### Minor Rebalancing

- Transfer  $\mathcal{O}(N^\varepsilon)$  tuples from one to the other part of the same factor!
- Time complexity:  $\mathcal{O}(N^{2\varepsilon})$

## Rebalancing Partitions

Updates can change the heavy-light threshold!



### Major Rebalancing

- Recompute partitions and views from scratch!
- Time complexity:  $\mathcal{O}(N)$

## Amortization of Rebalancing Times

- Major rebalancing can require linear and minor rebalancing super-linear time

## Amortization of Rebalancing Times

- Major rebalancing can require linear and minor rebalancing super-linear time
- The rebalancing times amortize over sequences of updates.
  - Amortized minor rebalancing time:  $\mathcal{O}(N^\varepsilon)$
  - Amortized major rebalancing time:  $\mathcal{O}(1)$
- Overall amortized rebalancing time:  $\mathcal{O}(N^\varepsilon)$

$\mathcal{O}(N^{2\varepsilon})$                                      $\mathcal{O}(N^{2\varepsilon})$   
... update **minor** update ... update **minor** update ...  
 $\underbrace{\qquad\qquad\qquad}_{\Omega(N^\varepsilon)}$

$\mathcal{O}(N)$                                      $\mathcal{O}(N)$   
... update **major** update ... update **major** update ...  
 $\underbrace{\qquad\qquad\qquad}_{\Omega(N)}$

## Amortization of Rebalancing Times

- Major rebalancing can require linear and minor rebalancing super-linear time
- The rebalancing times amortize over sequences of updates.
  - Amortized minor rebalancing time:  $\mathcal{O}(N^\varepsilon)$
  - Amortized major rebalancing time:  $\mathcal{O}(1)$
- Overall amortized rebalancing time:  $\mathcal{O}(N^\varepsilon)$
- Rebalancing time can be de-amortized using standard techniques

$\mathcal{O}(N^{2\varepsilon})$                                      $\mathcal{O}(N^{2\varepsilon})$   
... update **minor** update ... update **minor** update ...  
 $\underbrace{\qquad\qquad\qquad}_{\Omega(N^\varepsilon)}$

$\mathcal{O}(N)$                                      $\mathcal{O}(N)$   
... update **major** update ... update **major** update ...  
 $\underbrace{\qquad\qquad\qquad}_{\Omega(N)}$