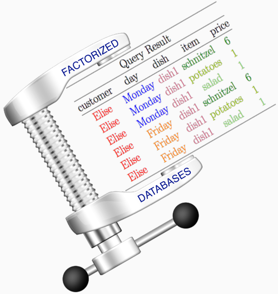
Incremental View Maintenance with Triple-Lock Factorization Benefits

Milos Nikolic and Dan Olteanu

Toronto, October 2017

RelationalAI



In-Database Analytics

- Integrate analytics into relational database engines
  - Mathematical optimization, statistics, ML, data mining
- Move the analytics, not the data
  - Avoid expensive data export/import
  - Exploit database technologies
  - Build better models using larger datasets

In-Database Analytics on Streaming Datasets

- Datasets continuously evolve over time
  - E.g.: data streams from sensors, social networks, apps
- Real-time analytics over streaming data
  - Users want fresh data models
  - Long-lived (continuous) queries provide up-to-date results

Challenges: In-Database Real-time Analytics

1. Analytics over relational databases
    - Combine different data sources to improve models
    - Common practice: join relations, then build models

      ⇒ **Inefficient:** high redundancy in computation and representation of join results

2. Low-latency processing
    - Naïve solution: re-compute query results as data changes

      ⇒ **Inefficient:** high-latency processing
    - Common practice: **IVM** (Incremental View Maintenance)

      For query Q, database $\mathcal{D}$, and change ΔD, compute (the hopefully cheaper) delta ΔQ:

      $$Q(\mathcal{D} + \Delta\mathcal{D}) = Q(\mathcal{D}) + \Delta Q(\mathcal{D}, \Delta\mathcal{D})$$

3. Support for complex analytics

Our Approach: Factorized IVM

- *"Concrete recipe on how to IVM the next analytic task you may face"* (anonymous SIGMOD'18 reviewer)
- Generalized aggregates over joins
  - Relations are functions mapping tuples to ring values
  - Computation described by application-specific rings
- Triple-lock factorization: keys, payloads, updates
  - Factorized Keys = Factorized Query Processing
  - Factorized Payloads = Avoid listing representation
  - Bulk updates decomposed into sums of joins of factors
- Prototype implemented on top of DBToaster
  - Performance: Up to 2 OOM faster than classical IVM and DBToaster and up to 4 OOM less memory than DBToaster

Talk Outline

Factorized Ring Computation

- Relations are modeled as factors
  - Functions mapping keys (tuples of values) to payloads (ring elements)

A	B	$\rightarrow$	R[A, B]
a_1	b_1	$\rightarrow$	r_1
a_2	b_1	$\rightarrow$	r_2

Finitely many tuples with non-zero payloads

r_1 and r_2 are elements from a ring

- Query language: Subset of FAQ
  - Operations: union, join, and variable marginalization
  - More expressiveness via application-specific rings
- Query evaluation: FDB/FAQ engine variation

Rings

- A ring $(\mathbf{D}, +, *, \mathbf{0}, \mathbf{1})$ is a set $\mathbf{D}$ with two binary ops:

Additive commutativity	$a + b = b + a$
Additive associativity	$(a + b) + c = a + (b + c)$
Additive identity	$\mathbf{0} + a = a + \mathbf{0} = a$
Additive inverse	$\exists -a \in \mathbf{D} : a + (-a) = (-a) + a = \mathbf{0}$
Multiplicative associativity	$(a * b) * c = a * (b * c)$
Multiplicative identity	$a * \mathbf{1} = \mathbf{1} * a = a$
Left and right distributivity	$a * (b + c) = a * b + a * c$ and $(a + b) * c = a * c + b * c$

- Examples: $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{R}^n$, matrix ring, polynomial ring

Factor Operations

Factors R, S, and T with payloads from a ring $(\mathbf{D}, +, *, \mathbf{0}, \mathbf{1})$:

A	B	$\rightarrow$	R[A, B]
a_1	b_1	$\rightarrow$	r_1
a_2	b_1	$\rightarrow$	r_2

A	B	$\rightarrow$	S[A, B]
a_2	b_1	$\rightarrow$	s_1
a_3	b_2	$\rightarrow$	s_2

B	C	$\rightarrow$	T[B, C]
b_1	c_1	$\rightarrow$	t_1
b_2	c_2	$\rightarrow$	t_2

Operations:

Union $\uplus$

A	B	$\rightarrow$	$(R \uplus S)[A, B]$
a_1	b_1	$\rightarrow$	r_1
a_2	b_1	$\rightarrow$	$r_2 + s_1$
a_3	b_2	$\rightarrow$	s_2

Join $\otimes$

A	B	C	$\rightarrow$	$((R \uplus S) \otimes T)[A, B, C]$
a_1	b_1	c_1	$\rightarrow$	$r_1 * t_1$
a_2	b_1	c_1	$\rightarrow$	$(r_2 + s_1) * t_1$
a_3	b_2	c_2	$\rightarrow$	$s_2 * t_2$

Marginalization $\bigoplus_A$

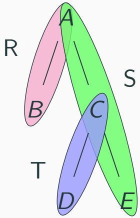
B	C	$\rightarrow$	$(\bigoplus_A (R \uplus S) \otimes T)[B, C]$

Example: Aggregate Computation

Compute COUNT over the natural join:
$R(A, B)$, $S(A, C, E)$, $T(C, D)$

Let all relations be of size N

View relations as factors mapping tuples to multiplicity from $\mathbb{Z}$



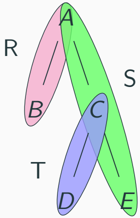
Join hypergraph

Example: Aggregate Computation

Compute COUNT over the natural join:
$R(A, B)$, $S(A, C, E)$, $T(C, D)$

Let all relations be of size N

View relations as factors mapping tuples to multiplicity from $\mathbb{Z}$



Join hypergraph

Naïve: compute the join and then COUNT

$$\mathrm{Q} = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (\mathrm{R} \otimes \mathrm{S} \otimes \mathrm{T})$$

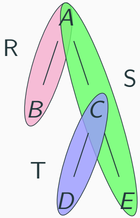
Takes $\mathcal{O}(N^3)$ time!

Example: Aggregate Computation

Compute COUNT over the natural join:
$R(A, B)$, $S(A, C, E)$, $T(C, D)$

Let all relations be of size N

View relations as factors mapping tuples to multiplicity from $\mathbb{Z}$



Join hypergraph

Naïve: compute the join and then COUNT

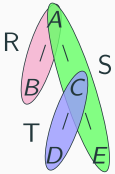
$$\mathrm{Q} = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (\mathrm{R} \otimes \mathrm{S} \otimes \mathrm{T})$$

Takes $\mathcal{O}(N^3)$ time!

Can we compute COUNT in $\mathcal{O}(N)$ time?

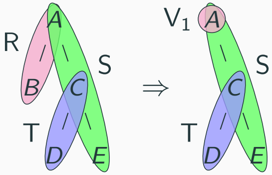
Example: Factorized Aggregate Computation

- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:



Example: Factorized Aggregate Computation

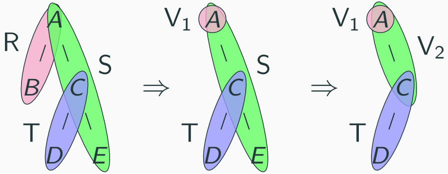
- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:



$V_1[A] = \bigoplus_B R[A, B]$ (marginalize B)

Example: Factorized Aggregate Computation

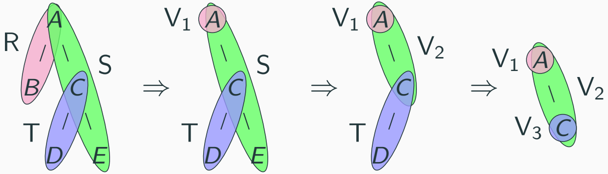
- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:



$V_1[A] = \bigoplus_B R[A, B]$ (marginalize B) $V_2[A, C] = \bigoplus_E S[A, C, E]$ (marginalize E)

Example: Factorized Aggregate Computation

- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:

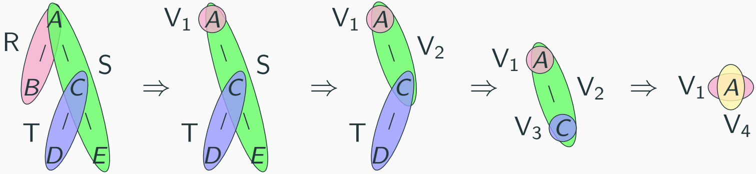


$V_1[A] = \bigoplus_B R[A, B]$ (marginalize B) $V_2[A, C] = \bigoplus_E S[A, C, E]$ (marginalize E)

$V_3[C] = \bigoplus_D T[C, D]$ (marginalize D)

Example: Factorized Aggregate Computation

- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:

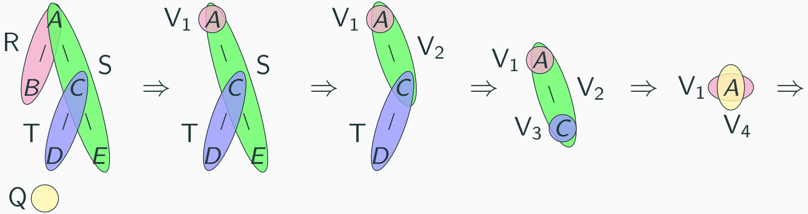


$V_1[A] = \bigoplus_B R[A,B]$ (marginalize B) $V_2[A,C] = \bigoplus_E S[A,C,E]$ (marginalize E)

$V_3[C] = \bigoplus_D T[C,D]$ (marginalize D) $V_4[A] = \bigoplus_C (V_2[A,C] \otimes V_3[C])$ (marginalize C) also re-use counts of E and D!

Example: Factorized Aggregate Computation

- Push COUNT past joins to eliminate variables
- Factorized computation à la InsideOut/FDB:



$V_1[A] = \bigoplus_B R[A, B]$ (marginalize B) $V_2[A, C] = \bigoplus_E S[A, C, E]$ (marginalize E)

$V_3[C] = \bigoplus_D T[C, D]$ (marginalize D) $V_4[A] = \bigoplus_C (V_2[A, C] \otimes V_3[C])$ (marginalize C) also re-use counts of E and D!

$Q = \bigoplus_A (V_1[A] \otimes V_4[A])$ (marginalize A)

Different Modeling of Relations

- Compute $SUM(C \cdot D)$ over the join $R(A, B)$, $S(A, C, E)$, $T(C, D)$
  - Let the domain of all variables be $\mathbb{R}$
- Model relations as factors with payloads from $\mathbb{R}$:
  - $\mathrm{R}[a, b] = 1$ iff $(a, b) \in R$, 0 otherwise
  - $\mathrm{S}[a, c, e] = c$ iff $(a, c, e) \in S$, 0 otherwise
  - $\mathrm{T}[c, d] = d$ iff $(c, d) \in T$, 0 otherwise

Different Modeling of Relations

- Compute $SUM(C \cdot D)$ over the join $R(A, B)$, $S(A, C, E)$, $T(C, D)$
  - Let the domain of all variables be $\mathbb{R}$
- Model relations as factors with payloads from $\mathbb{R}$:
  - $\mathrm{R}[a, b] = 1$ iff $(a, b) \in R$, 0 otherwise
  - $\mathrm{S}[a, c, e] = c$ iff $(a, c, e) \in S$, 0 otherwise
  - $\mathrm{T}[c, d] = d$ iff $(c, d) \in T$, 0 otherwise
- The factor Q expressing the sum is:

$$\mathrm{Q} = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (\mathrm{R}[A, B] \otimes \mathrm{S}[A, C, E] \otimes \mathrm{T}[C, D])$$

Factor payloads carry out the summation!

- Same as the COUNT query but with diff modeling and ring!

Modeling Relations as Factors

Eager modeling (as in previous examples)

- Assign payload $\mathrm{R}[t]$ to each tuple t of relation R
- Computed payloads might be discarded later on
  - $\mathrm{T}[c, d] = c \cdot d$ computed for every pair (c, d) in T, even for those c-values that do not exist in S
  - *Non-trivial cost with more complex rings!*

Modeling Relations as Factors

Eager modeling (as in previous examples)

- Assign payload $\mathrm{R}[t]$ to each tuple t of relation R
- Computed payloads might be discarded later on
  - $\mathrm{T}[c, d] = c \cdot d$ computed for every pair (c, d) in T, even for those c-values that do not exist in S
  - *Non-trivial cost with more complex rings!*

Lazy modeling

- Decompose payload computation into a product of functions of one variable: $f(c, d) = c \cdot d = f_C(c) \cdot f_D(c)$
- Use them to lift variable values to payloads on demand
  - E.g., after ensuring a C-value appears in both S and T

Factorized Computation with Lift Factors

- Factors R, S, and T with payloads from a ring $(\mathbf{D}, +, *, \mathbf{0}, \mathbf{1})$
  - Each tuple has the payload of $\mathbf{1} \in \mathbf{D}$
- Lift factors Λ_A, Λ_B, Λ_C, Λ_D, Λ_E map the domain of a variable to **D**

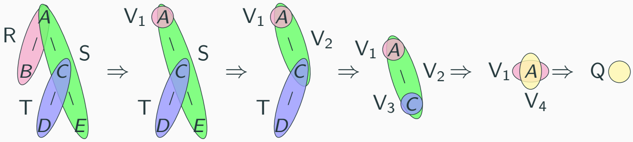
COUNT	all lift factors map to $1 \in \mathbb{Z}$
SUM(C*D)	$\Lambda_C[c] = c$ and $\Lambda_D[d] = d$; others map to $1 \in \mathbb{R}$

Factorized Computation with Lift Factors

- Factors R, S, and T with payloads from a ring $(\mathbf{D}, +, *, \mathbf{0}, \mathbf{1})$
  - Each tuple has the payload of $\mathbf{1} \in \mathbf{D}$
- Lift factors Λ_A, Λ_B, Λ_C, Λ_D, Λ_E map the domain of a variable to $\mathbf{D}$

  COUNT all lift factors map to $1 \in \mathbb{Z}$

  SUM(C*D) $\Lambda_C[c] = c$ and $\Lambda_D[d] = d$; others map to $1 \in \mathbb{R}$
- Lift values of a variable just before its marginalization



$V_1[A] = \bigoplus_B (R[A, B] \otimes \Lambda_B[B])$ $\quad V_2[A, C] = \bigoplus_E (S[A, C, E] \otimes \Lambda_E[E])$

$V_3[C] = \bigoplus_D (T[C, D] \otimes \Lambda_D[D])$ $\quad V_4[A] = \bigoplus_C (V_2[A, C] \otimes V_3[C] \otimes \Lambda_C[C])$

$Q = \bigoplus_A (V_1[A] \otimes V_4[A] \otimes \Lambda_A[A]))$

Variable Orders

Variable order for a join query Q

- Rooted tree with one node per variable in Q
- Function *dep* maps each variable to a subset of its ancestors

- Properties:
  - The variables of a factor R lie along the same root-to-leaf path
    - $Y \in dep(X)$ if X and Y are variables of R and Y is ancestor of X
  - For every child B of A, $dep(B) \subseteq dep(A) \cup \{A\}$
- One variable order for the query $R(A, B), S(A, C, E), T(C, D)$

```
                      A          dep(A) = ∅
                     / \
dep(B) = {A}       B     C       dep(C) = {A}
                        / \
dep(D) = {C}           D   E     dep(E) = {A, C}
```

Captures conditional independence

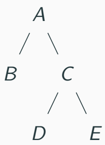
View Trees

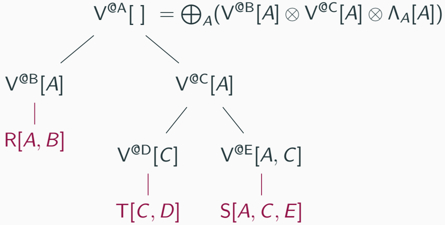
Variable orders guide query evaluation

- Create a factor view at each variable in the order
- $V^{@X}$ – view at variable X with schema $dep(X)$
  1. joins the views at its children
  2. lifts and marginalizes X if X is not a free (group-by) variable

Variable order ⇒ **View tree**

$dep(A) = \emptyset$
$dep(B) = \{A\}$
$dep(C) = \{A\}$
$dep(D) = \{C\}$
$dep(E) = \{A, C\}$





- Views can be materialized if needed

FAQs: Functional Aggregate Queries

We support a subset of FAQs:

$$\mathrm{Q}[X_1, \ldots, X_f] = \bigoplus_{X_{f+1}} \cdots \bigoplus_{X_m} \bigotimes_{i \in [n]} \mathrm{R_i}[\mathcal{S}_i] \bigotimes_{j \in [f+1,m]} \Lambda_{X_j}[X_j]$$

where:

- Factors $\mathrm{R_1}, \ldots, \mathrm{R_n}$ are defined over variables $X_1, \ldots, X_m$
- $X_1, \ldots, X_f$ are free variables
- Each factor $\mathrm{R_i}$ maps keys over schema $\mathcal{S}_i$ to payloads in a ring $(\mathbf{D}, +, *, \mathbf{0}, \mathbf{1})$

Talk Outline

Incremental Computation

- Maintain query results with changes in the underlying data

$$Q(\mathcal{D} + \Delta\mathcal{D}) = Q(\mathcal{D}) + \Delta Q(\mathcal{D}, \Delta\mathcal{D})$$

Fast "merge" operation

Smaller and faster delta query (ideally)

- Incremental View Maintenance (IVM) in databases
  - Often with limited query support and poor performance

Incremental View Maintenance with Factors

- Ring payloads simplify incremental computation
  - Updates are uniformly represented as factors

A	B	$\rightarrow$	$\delta R[A, B]$
a_1	b_1	$\rightarrow$	-1
a_4	b_3	$\rightarrow$	2

Tuples with positive/negative payloads denote insertions/deletions

  - Applying updates: $R_{new}[A, B] = R_{old}[A, B] \uplus \delta R[A, B]$

- The query language is closed under taking deltas

$$\delta(R \uplus S) = \delta R \uplus \delta S$$

$$\delta(R \otimes S) = (\delta R \otimes S) \uplus (R \otimes \delta S) \uplus (\delta R \otimes \delta S)$$

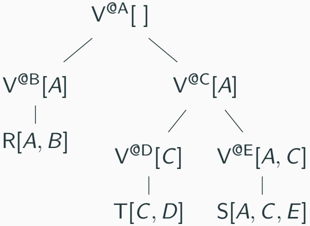
$$\delta(\bigoplus_A R) = \bigoplus_A \delta R$$

Delta Propagation

Consider our running example

Maintain the query result for updates to T

View tree

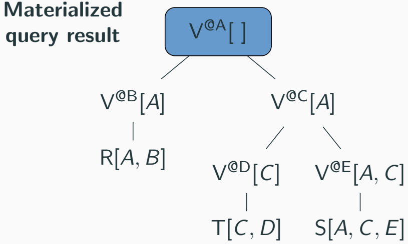


Delta Propagation

Consider our running example

Maintain the query result for updates to T

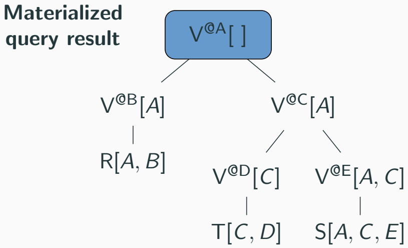
View tree

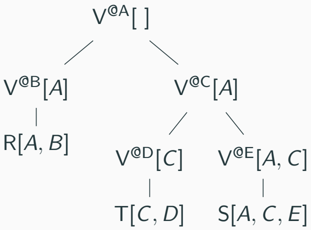


Consider our running example

Maintain the query result for updates to T

View tree ⇒ **Delta view tree**



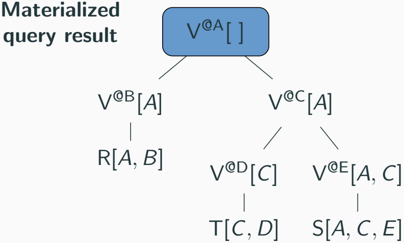


Delta Propagation

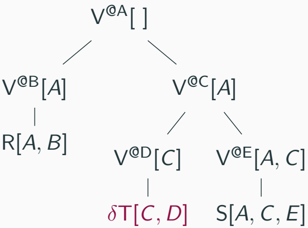
Consider our running example

Maintain the query result for updates to T

View tree



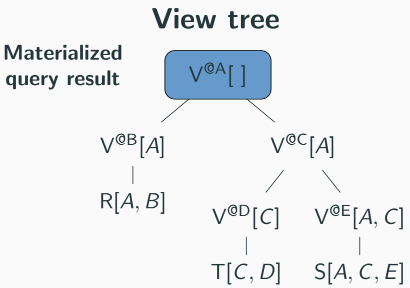
Delta view tree

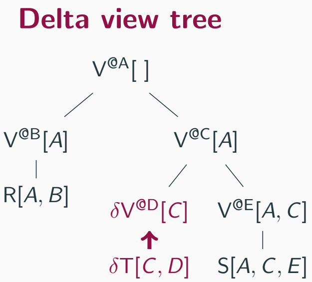


Delta Propagation

Consider our running example

Maintain the query result for updates to T



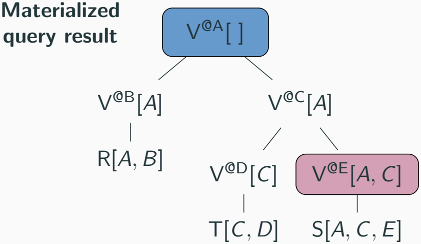


Delta Propagation

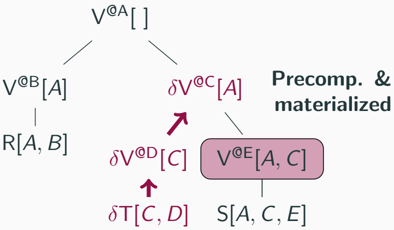
Consider our running example

Maintain the query result for updates to T

View tree



Delta view tree

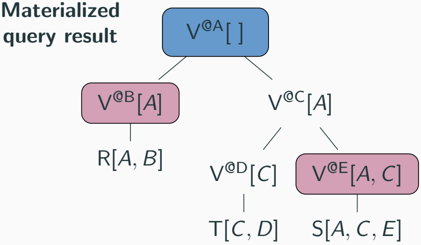


Delta Propagation

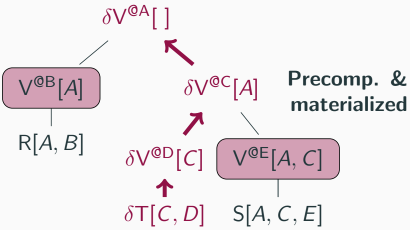
Consider our running example

Maintain the query result for updates to T

View tree



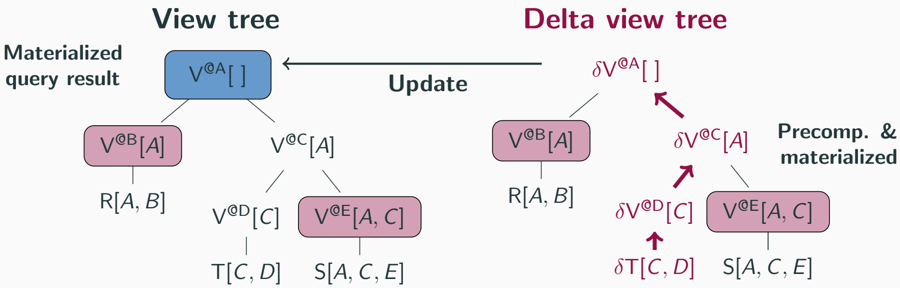
Delta view tree



Delta Propagation

Consider our running example

Maintain the query result for updates to T

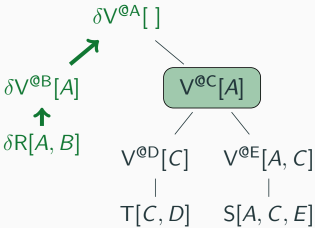


Updates to Multiple Factors

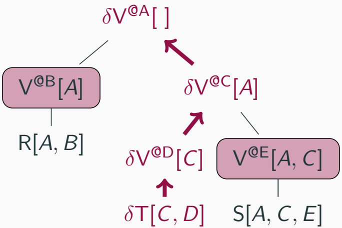
Maintain the query result for updates to R and T

- 2 propagation paths, 1 extra materialization
- Both paths need to maintain auxiliary views

Delta view tree (for R)



Delta view tree (for T)

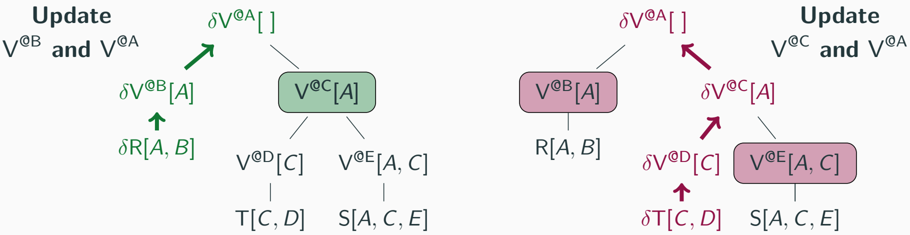


Updates to Multiple Factors

Maintain the query result for updates to R and T

- 2 propagation paths, 1 extra materialization
- Both paths need to maintain auxiliary views

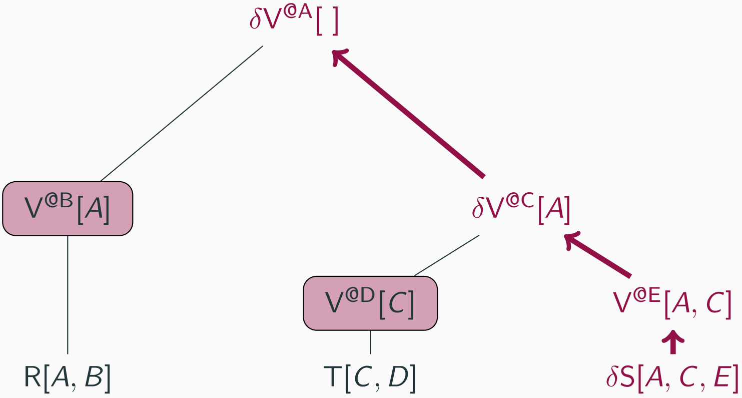
Delta view tree (for R) **Delta view tree (for T)**



Factorizable Bulk Updates

Assume update $\delta S[A, C, E]$ factorizes as $\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$.

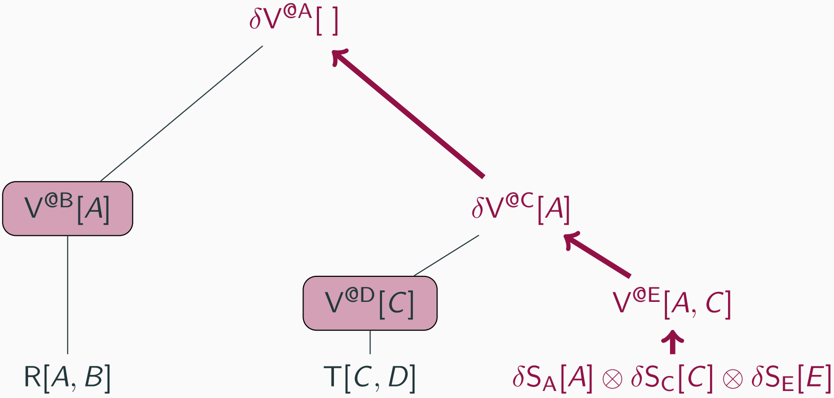
We may then factorize subsequent updates up the delta tree.



Factorizable Bulk Updates

Assume update $\delta S[A, C, E]$ factorizes as $\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$.

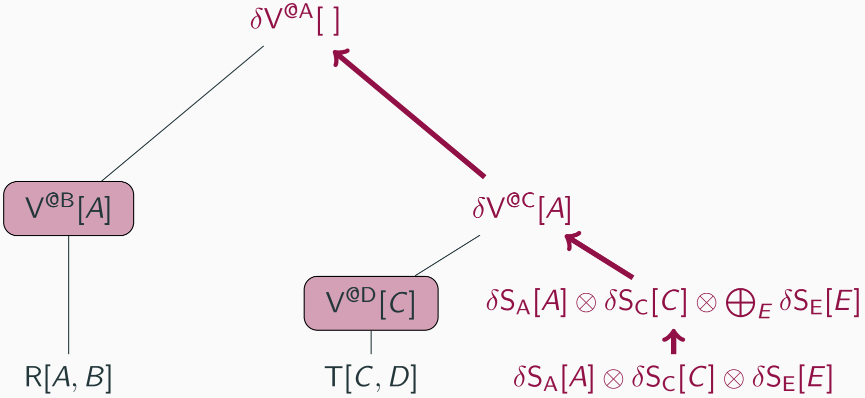
We may then factorize subsequent updates up the delta tree.



Factorizable Bulk Updates

Assume update $\delta S[A, C, E]$ factorizes as $\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$.

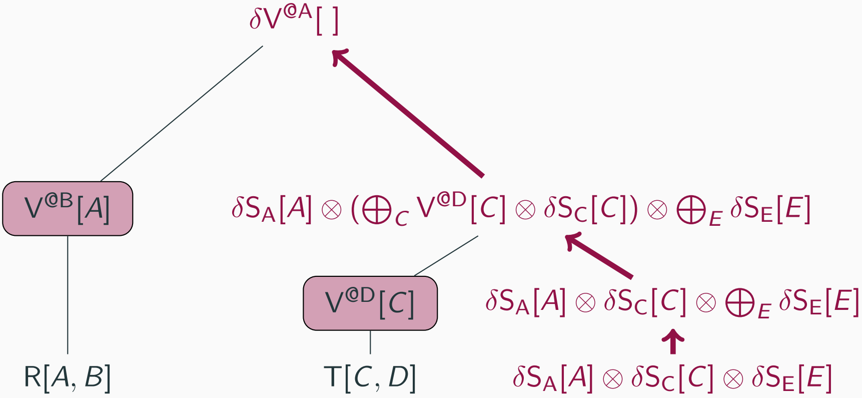
We may then factorize subsequent updates up the delta tree.



Factorizable Bulk Updates

Assume update $\delta S[A, C, E]$ factorizes as $\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$.

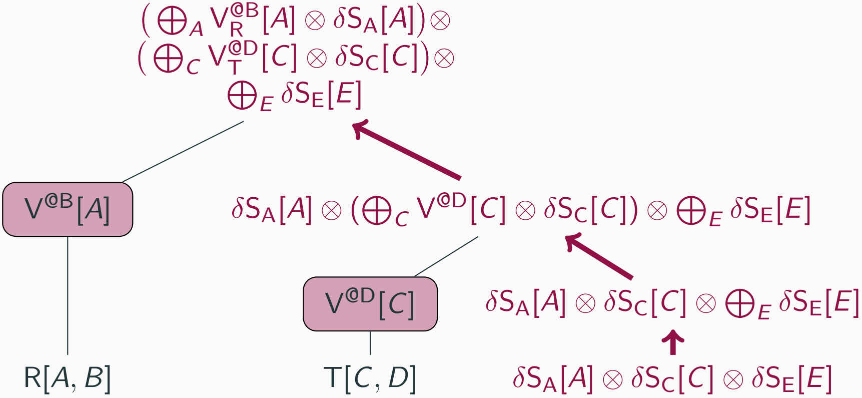
We may then factorize subsequent updates up the delta tree.



Factorizable Bulk Updates

Assume update $\delta S[A, C, E]$ factorizes as $\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$.

We may then factorize subsequent updates up the delta tree.



Talk Outline

Introduction

Factorized Ring Computation

Incremental View Maintenance

Applications

- Learning Linear Regression Models
- Factorized Representation of Conjunctive Query Results
- Matrix Chain Multiplication

Applications

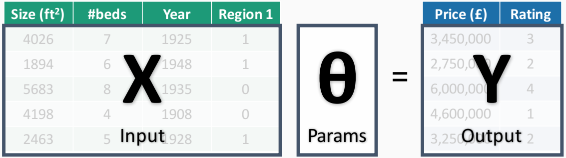
Aggregates over joins with task-specific rings can capture a host of problems

- learning regression models
- factorized representation of results of conjunctive queries
- matrix chain multiplication
- group-by aggregation (we've seen this already)
- inference in PGMs etc.

Next: zoom in the first three problems above

Learning Linear Regression Models

- Find model parameters Θ best satisfying:



- Iterative gradient computation:

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

- Matrices $\mathbf{X^T}\,\mathbf{X}$ and $\mathbf{X^T}\,\mathbf{Y}$ computed once for all iterations
  - Compute $SUM(X_i \cdot X_j)$ for each pair (X_i, X_j) of variables
  - We assume in this talk that all variables are continuous

Learning Linear Regression Models over Joins

Compute $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ when $\mathbf{X}$ is the join of input relations

- Naïve: compute the join, then $\mathcal{O}(m^2)$ sums over the join result ($m = \#$query variables)
- Factorized: compute one optimized join-aggregate query
  - Using our running query

$$\mathrm{Q} = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (\, \mathrm{R}[A,B] \otimes \mathrm{S}[A,C,E] \otimes \mathrm{T}[C,D] \\ \Lambda_A[A] \otimes \Lambda_B[B] \otimes \Lambda_C[C] \otimes \Lambda_D[D] \otimes \Lambda_E[E]$$

but a different payload ring and different lift factors!

Linear Regression Ring

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

$$\Big(\text{COUNT}, \quad \text{vector of } SUM(X_i), \quad \text{matrix of } SUM(X_i \cdot X_j) \Big)$$

$$a +^{\mathbf{D}} b = (c_a + c_b, \mathbf{s}_a + \mathbf{s}_b, \mathbf{Q}_a + \mathbf{Q}_b)$$

$$a *^{\mathbf{D}} b = (c_a c_b, c_b \mathbf{s}_a + c_a \mathbf{s}_b, c_b \mathbf{Q}_a + c_a \mathbf{Q}_b + \mathbf{s}_a \mathbf{s}_b^T + \mathbf{s}_b \mathbf{s}_a^T)$$

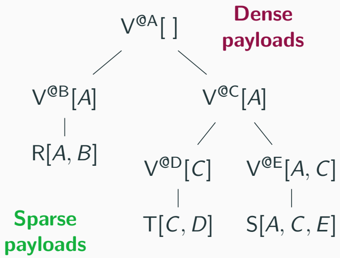
$$\mathbf{0} = (0, \mathbf{0}_{m \times 1}, \mathbf{0}_{m \times m})$$

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

Lift factor for variable X_j

$\Lambda_{X_j}[x] = (1, \mathbf{s}, \mathbf{Q})$ where

- $\mathbf{s}$ has all 0s except $s_j = x$
- $\mathbf{Q}$ has all 0s except $Q_{j,j} = x^2$



Performance: Learning Linear Regression Models over Joins

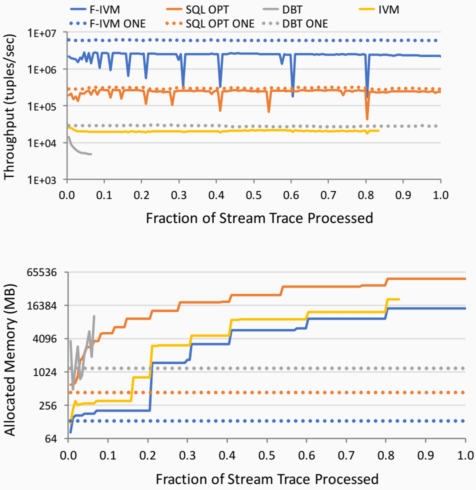
Streaming dataset with 5 relations

The natural join has 43 variables

Matrix with 946 distinct aggregates

Comparing IVM strategies on a common system

- F-IVM (9 views)
- SQL-OPT (9 views)
- DBToaster (3, 425 views)
- IVM (951 views)



Relational Data Ring

- Set of factors over $\mathbf{D}$ with $\uplus$ and $\otimes$ forms a ring of factors
  - Factor $\mathbf{0}$ maps every tuple to $\mathbf{0} \in \mathbf{D}$
  - Factor $\mathbf{1}$ maps the empty tuple to $\mathbf{1} \in \mathbf{D}$, others to $\mathbf{0} \in \mathbf{D}$
- Payload: Factors over $\mathbf{D} = \mathbb{Z}$ with the same schema!

A	B	$\rightarrow$	R[A, B]
a_1	b_1	$\rightarrow$	C: $c_1 \rightarrow 1$, $c_2 \rightarrow 1$
a_2	b_1	$\rightarrow$	C: $c_3 \rightarrow 1$

Keep results of conjunctive queries in payloads

Evaluating Conjunctive Queries using Relational Payloads

- Consider the conjunctive query:

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

- Compute Q using factors with relational payloads

$$\mathrm{Q} = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (\mathrm{R}[A, B] \otimes \mathrm{S}[A, C, E] \otimes \mathrm{T}[C, D] \\ \Lambda_A[A] \otimes \Lambda_B[B] \otimes \Lambda_C[C] \otimes \Lambda_D[D] \otimes \Lambda_E[E])$$

- Lift factors:

$$\Lambda_X[x] = \begin{cases} \begin{array}{|l} \mathrm{X} \\ \hline \mathrm{x} \to 1 \end{array} & \text{if } X \text{ is a free variable} \\ \begin{array}{|l} \\ \hline () \to 1 \end{array} & \textit{otherwise} \end{cases}$$

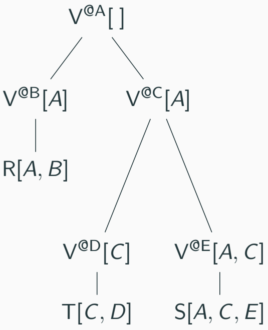
Listing Representation of Conjunctive Query Results

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

A	B	$\rightarrow$ R[A,B]
a_1	b_1	$\rightarrow$ $() \rightarrow 1$
a_1	b_2	$\rightarrow$ $() \rightarrow 1$
a_2	b_3	$\rightarrow$ $() \rightarrow 1$
a_3	b_4	$\rightarrow$ $() \rightarrow 1$

A	C	E	$\rightarrow$ S[A,C,E]
a_1	c_1	e_1	$\rightarrow$ $() \rightarrow 1$
a_1	c_1	e_2	$\rightarrow$ $() \rightarrow 1$
a_1	c_2	e_3	$\rightarrow$ $() \rightarrow 1$
a_2	c_2	e_4	$\rightarrow$ $() \rightarrow 1$

C	D	$\rightarrow$ T[C,D]
c_1	d_1	$\rightarrow$ $() \rightarrow 1$
c_2	d_2	$\rightarrow$ $() \rightarrow 1$
c_2	d_3	$\rightarrow$ $() \rightarrow 1$
c_3	d_4	$\rightarrow$ $() \rightarrow 1$



Listing Representation of Conjunctive Query Results

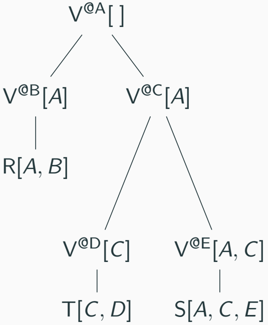
$$Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D)$$

A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1	→ $() \to 1$
a_1	c_1	e_2	→ $() \to 1$
a_1	c_2	e_3	→ $() \to 1$
a_2	c_2	e_4	→ $() \to 1$

C	D	→ T[C,D]
c_1	d_1	→ $() \to 1$
c_2	d_2	→ $() \to 1$
c_2	d_3	→ $() \to 1$
c_3	d_4	→ $() \to 1$

A	→ V@B[A]
a_1	→ B: $b_1 \to 1$, $b_2 \to 1$
a_2	→ B: $b_3 \to 1$
a_3	→ B: $b_4 \to 1$



Listing Representation of Conjunctive Query Results

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

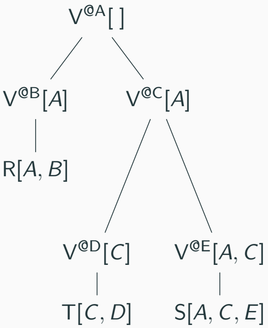
A	B	→ R[A,B]
a_1	b_1 →	$() \to 1$
a_1	b_2 →	$() \to 1$
a_2	b_3 →	$() \to 1$
a_3	b_4 →	$() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1 →	$() \to 1$
a_1	c_1	e_2 →	$() \to 1$
a_1	c_2	e_3 →	$() \to 1$
a_2	c_2	e_4 →	$() \to 1$

C	D	→ T[C,D]
c_1	d_1 →	$() \to 1$
c_2	d_2 →	$() \to 1$
c_2	d_3 →	$() \to 1$
c_3	d_4 →	$() \to 1$

A →	V@B[A]
a_1 →	B $b_1 \to 1$ $b_2 \to 1$
a_2 →	B $b_3 \to 1$
a_3 →	B $b_4 \to 1$

C →	V@D[C]
c_1 →	D $d_1 \to 1$
c_2 →	D $d_2 \to 1$ $d_3 \to 1$
c_3 →	D $d_4 \to 1$



Listing Representation of Conjunctive Query Results

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

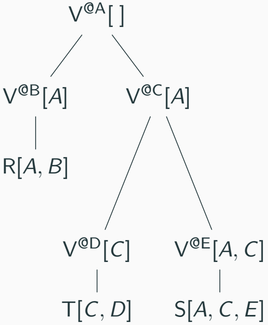
A	B	→ R[A,B]
a_1	b_1	→ () → 1
a_1	b_2	→ () → 1
a_2	b_3	→ () → 1
a_3	b_4	→ () → 1

A	C	E	→ S[A,C,E]
a_1	c_1	e_1	→ () → 1
a_1	c_1	e_2	→ () → 1
a_1	c_2	e_3	→ () → 1
a_2	c_2	e_4	→ () → 1

C	D	→ T[C,D]
c_1	d_1	→ () → 1
c_2	d_2	→ () → 1
c_2	d_3	→ () → 1
c_3	d_4	→ () → 1

A	→ $V^{@B}$[A]
a_1 →	B: $b_1 \rightarrow 1$, $b_2 \rightarrow 1$
a_2 →	B: $b_3 \rightarrow 1$
a_3 →	B: $b_4 \rightarrow 1$

C	→ $V^{@D}$[C]
c_1 →	D: $d_1 \rightarrow 1$
c_2 →	D: $d_2 \rightarrow 1$, $d_3 \rightarrow 1$
c_3 →	D: $d_4 \rightarrow 1$



A	C	→ $V^{@E}$[A,C]
a_1	c_1	→ () → 2
a_1	c_2	→ () → 1
a_2	c_2	→ () → 1

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

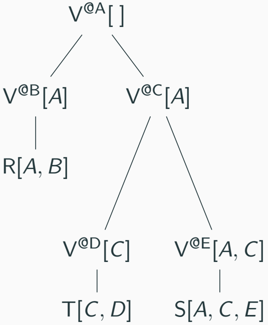
A	B	→ R[A,B]
a_1	b_1 →	$() \to 1$
a_1	b_2 →	$() \to 1$
a_2	b_3 →	$() \to 1$
a_3	b_4 →	$() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1 →	$() \to 1$
a_1	c_1	e_2 →	$() \to 1$
a_1	c_2	e_3 →	$() \to 1$
a_2	c_2	e_4 →	$() \to 1$

C	D	→ T[C,D]
c_1	d_1 →	$() \to 1$
c_2	d_2 →	$() \to 1$
c_2	d_3 →	$() \to 1$
c_3	d_4 →	$() \to 1$

A →	V@B[A] (B)
a_1 →	$b_1 \to 1$
	$b_2 \to 1$
a_2 →	$b_3 \to 1$
a_3 →	$b_4 \to 1$

C →	V@D[C] (D)
c_1 →	$d_1 \to 1$
c_2 →	$d_2 \to 1$
	$d_3 \to 1$
c_3 →	$d_4 \to 1$



A →	V@C[A] (C D)
a_1 →	$c_1\ d_1 \to 2$
	$c_2\ d_2 \to 1$
	$c_2\ d_3 \to 1$
a_2 →	$c_2\ d_2 \to 1$
	$c_2\ d_3 \to 1$

A	C	→ V@E[A,C]
a_1	c_1 →	$() \to 2$
a_1	c_2 →	$() \to 1$
a_2	c_2 →	$() \to 1$

Listing Representation of Conjunctive Query Results

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

A B → R[A,B]

A	B	→	R[A,B]
a_1	b_1	→	$() \to 1$
a_1	b_2	→	$() \to 1$
a_2	b_3	→	$() \to 1$
a_3	b_4	→	$() \to 1$

A C E → S[A,C,E]

A	C	E	→	S[A,C,E]
a_1	c_1	e_1	→	$() \to 1$
a_1	c_1	e_2	→	$() \to 1$
a_1	c_2	e_3	→	$() \to 1$
a_2	c_2	e_4	→	$() \to 1$

C D → T[C,D]

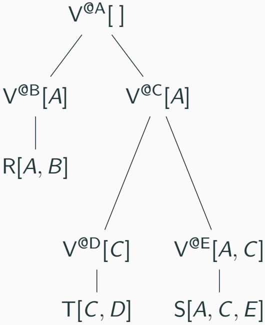
C	D	→	T[C,D]
c_1	d_1	→	$() \to 1$
c_2	d_2	→	$() \to 1$
c_2	d_3	→	$() \to 1$
c_3	d_4	→	$() \to 1$

A → V@B[A]

A	→	B
a_1	→	$b_1 \to 1$
		$b_2 \to 1$
a_2	→	$b_3 \to 1$
a_3	→	$b_4 \to 1$

C → V@D[C]

C	→	D
c_1	→	$d_1 \to 1$
c_2	→	$d_2 \to 1$
		$d_3 \to 1$
c_3	→	$d_4 \to 1$



() → V@A[ ]

()	→	A	B	C	D	
()	→	a_1	b_1	c_1	d_1	→ 2
		a_1	b_1	c_2	d_2	→ 1
		a_1	b_1	c_2	d_3	→ 1
		a_1	b_2	c_1	d_1	→ 2
		a_1	b_2	c_2	d_2	→ 1
		a_1	b_2	c_2	d_3	→ 1
		a_2	b_3	c_2	d_2	→ 1
		a_2	b_3	c_2	d_3	→ 1

A → V@C[A]

A	→	C	D	
a_1	→	c_1	d_1	→ 2
		c_2	d_2	→ 1
		c_2	d_3	→ 1
a_2	→	c_2	d_2	→ 1
		c_2	d_3	→ 1

A C → V@E[A,C]

A	C	→	V@E[A,C]
a_1	c_1	→	$() \to 2$
a_1	c_2	→	$() \to 1$
a_2	c_2	→	$() \to 1$

Factorized Representation of Conjunctive Query Results

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

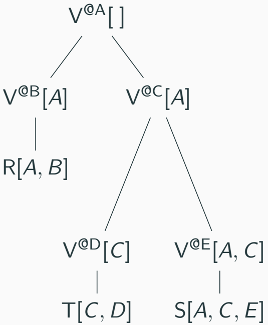
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1	→ $() \to 1$
a_1	c_1	e_2	→ $() \to 1$
a_1	c_2	e_3	→ $() \to 1$
a_2	c_2	e_4	→ $() \to 1$

C	D	→ T[C,D]
c_1	d_1	→ $() \to 1$
c_2	d_2	→ $() \to 1$
c_2	d_3	→ $() \to 1$
c_3	d_4	→ $() \to 1$

A	→ V@B[A] (B)
a_1	→ $b_1 \to 1$, $b_2 \to 1$
a_2	→ $b_3 \to 1$
a_3	→ $b_4 \to 1$

C	→ V@D[C] (D)
c_1	→ $d_1 \to 1$
c_2	→ $d_2 \to 1$, $d_3 \to 1$
c_3	→ $d_4 \to 1$



A	C	→ V@E[A,C]
a_1	c_1	→ $() \to 2$
a_1	c_2	→ $() \to 1$
a_2	c_2	→ $() \to 1$

Factorized Representation of Conjunctive Query Results

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

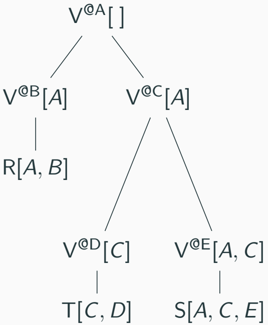
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1	→ $() \to 1$
a_1	c_1	e_2	→ $() \to 1$
a_1	c_2	e_3	→ $() \to 1$
a_2	c_2	e_4	→ $() \to 1$

C	D	→ T[C,D]
c_1	d_1	→ $() \to 1$
c_2	d_2	→ $() \to 1$
c_2	d_3	→ $() \to 1$
c_3	d_4	→ $() \to 1$

A	→ $V^{@B}$[A]
a_1 →	B: $b_1 \to 1$, $b_2 \to 1$
a_2 →	B: $b_3 \to 1$
a_3 →	B: $b_4 \to 1$

C	→ $V^{@D}$[C]
c_1 →	D: $d_1 \to 1$
c_2 →	D: $d_2 \to 1$, $d_3 \to 1$
c_3 →	D: $d_4 \to 1$



A	→ $V^{@C}$[A]
a_1 →	C: $c_1 \to 2$, $c_2 \to 2$
a_2 →	C: $c_2 \to 2$

A	C	→ $V^{@E}$[A,C]
a_1	c_1	→ $() \to 2$
a_1	c_2	→ $() \to 1$
a_2	c_2	→ $() \to 1$

Factorized Representation of Conjunctive Query Results

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

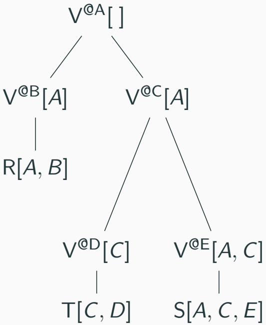
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

A	C	E	→ S[A,C,E]
a_1	c_1	e_1	→ $() \to 1$
a_1	c_1	e_2	→ $() \to 1$
a_1	c_2	e_3	→ $() \to 1$
a_2	c_2	e_4	→ $() \to 1$

C	D	→ T[C,D]
c_1	d_1	→ $() \to 1$
c_2	d_2	→ $() \to 1$
c_2	d_3	→ $() \to 1$
c_3	d_4	→ $() \to 1$

A	→ $V^{@B}$[A]
a_1	→ B: $b_1 \to 1$, $b_2 \to 1$
a_2	→ B: $b_3 \to 1$
a_3	→ B: $b_4 \to 1$

C	→ $V^{@D}$[C]
c_1	→ D: $d_1 \to 1$
c_2	→ D: $d_2 \to 1$, $d_3 \to 1$
c_3	→ D: $d_4 \to 1$



()	→ $V^{@A}$[ ]
()	→ A: $a_1 \to 8$, $a_2 \to 2$

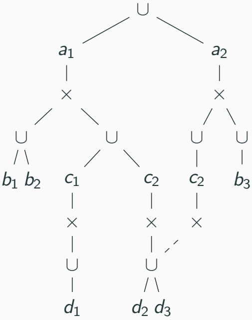
A	→ $V^{@C}$[A]
a_1	→ C: $c_1 \to 2$, $c_2 \to 2$
a_2	→ C: $c_2 \to 2$

A	C	→ $V^{@E}$[A,C]
a_1	c_1	→ $() \to 2$
a_1	c_2	→ $() \to 1$
a_2	c_2	→ $() \to 1$

Factorized Representation of Conjunctive Query Results

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

Factorized Join

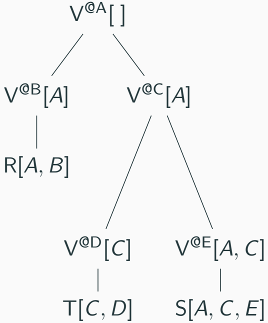


$A \rightarrow V^{@B}[A]$

A	B	
a_1	b_1	1
	b_2	1
a_2	b_3	1
a_3	b_4	1

$C \rightarrow V^{@D}[C]$

C	D	
c_1	d_1	1
c_2	d_2	1
	d_3	1
c_3	d_4	1



$() \rightarrow V^{@A}[\,]$

()	A	
$()$	a_1	8
	a_2	2

$A \rightarrow V^{@C}[A]$

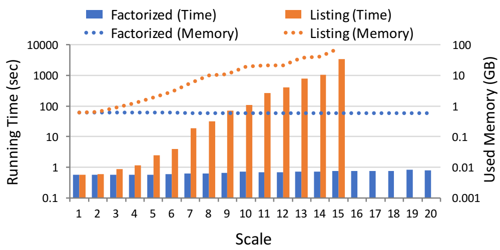
A	C	
a_1	c_1	2
	c_2	2
a_2	c_2	2

$A\ C \rightarrow V^{@E}[A,C]$

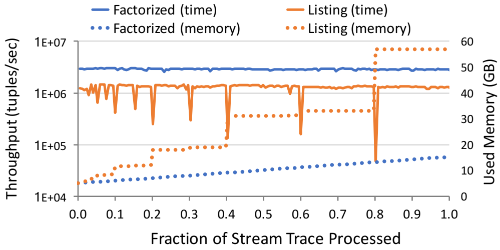
A C	()	
$a_1\ c_1$	$()$	2
$a_1\ c_2$	$()$	1
$a_2\ c_2$	$()$	1

Performance: Maintenance of Conjunctive Query Results

Star schema



Snowflake schema



Matrix Chain Multiplication

Input: Matrices $\boldsymbol{A}_i$ of size of $p_i \times p_{i+1}$ over some ring $\mathbf{D}$ ($i \in [n]$)

- Modeled as factors $\mathsf{A_i}[X_i, X_{i+1}]$ with payloads carrying matrix values in $\mathbf{D}$

Problem: Compute their product matrix of size $p_1 \times p_{n+1}$

$$\mathsf{A}[X_1, X_{n+1}] = \bigoplus_{X_2} \cdots \bigoplus_{X_n} \bigotimes_{i \in [n]} \mathsf{A_i}[X_i, X_{i+1}] \bigotimes_{j \in [2,n]} \Lambda_{X_j}[X_j]$$

where each lift view $\Lambda_{X_j}[X_j]$ maps any key to payload $\mathbf{1} \in \mathbf{D}$.

Factorized Matrix Updates

Matrix changes

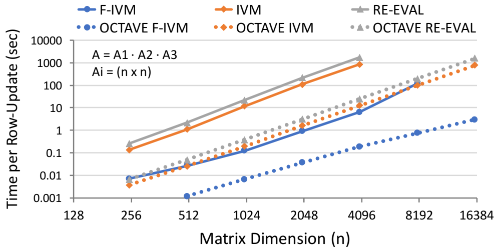
- Single-value change $\Rightarrow$ vector outer product
  $\delta \mathsf{A_i}[X_i, X_{i+1}] = \mathsf{u}[X_i] \otimes \mathsf{v}[X_{i+1}]$
- Several-values change $\Rightarrow$ sum of vector outer products
  $\delta \mathsf{A_i}[X_i, X_{i+1}] = \uplus_{k \in [r]} \mathsf{u_k}[X_i] \otimes \mathsf{v_k}[X_{i+1}]$

Time complexity for multiplication of n matrices of size $p \times p$:

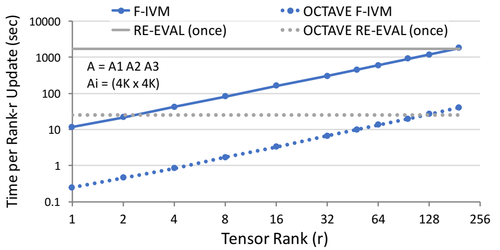
- Evaluation or IVM: $O(p^3)$
- IVM *with* factorized updates: $O(p^2)$

Performance: Matrix Chain Multiplication

Update to A_2 expressed as vector outer product



Update to A_2 expressed as sum of r vector outer products



Summary: Factorized Incremental View Maintenance

- Framework for unified IVM of in-database analytics
  - Captures many application scenarios
- Based on 3 shades of factorization
  - Factorized query evaluation
    - Exploits conditional independence among query variables
  - Factorized representation of query results
    - Enables succinct result representation
  - Factorized updates
    - Exploits low-rank tensor decomposition of updates
- Performance: Up to 2 OOM faster and 4 OOM less memory than state-of-the-art IVM techniques
- *Our IVM framework can accommodate any ring*

As My Girl Beyoncé Would Say..

Thank you!

The Triangle Query

$$Q_{\triangle}[\,] = \bigoplus_A \bigoplus_B \bigoplus_C R[A,B] \otimes S[B,C] \otimes T[C,A]$$