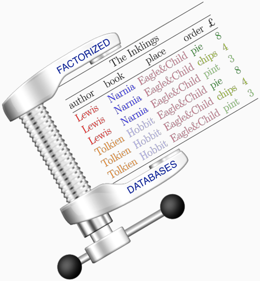
Incremental View Maintenance with Triple-Lock Factorization Benefits

`fdbresearch.github.io`

Milos Nikolic and Dan Olteanu (Oxford)

Université Libre de Bruxelles, December 2017

partially funded by the Wiener Anspach Foundation



Integrate analytics into relational data systems

In-Database Analytics Builds on Three Observations

1. Move the analytics and not the data
   - Small analytics code vs. large data: Avoid expensive data export/import in the software stack
   - Exploit database technology and the relational structure (schema, query, functional dependencies)
   - Build better models faster and using larger datasets

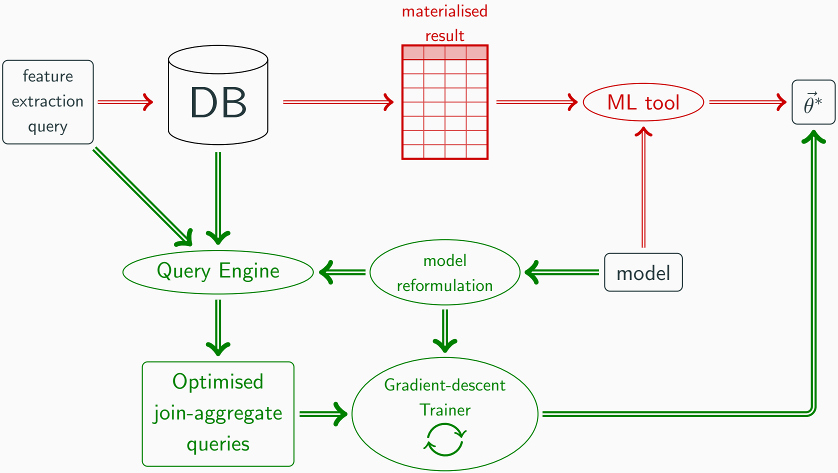
In-Database Analytics Builds on Three Observations

1. Move the analytics and not the data
   - Small analytics code vs. large data: Avoid expensive data export/import in the software stack
   - Exploit database technology and the relational structure (schema, query, functional dependencies)
   - Build better models faster and using larger datasets
2. Analytics code can be cast as join-aggregate queries
   - Many similar queries that massively share computation
   - Fixpoint computation needed for model convergence

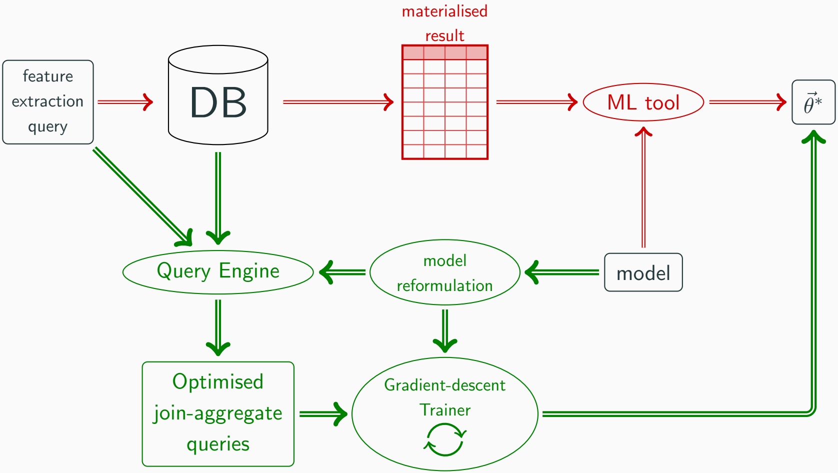
In-Database Analytics Builds on Three Observations

1. Move the analytics and not the data
   - Small analytics code vs. large data: Avoid expensive data export/import in the software stack
   - Exploit database technology and the relational structure (schema, query, functional dependencies)
   - Build better models faster and using larger datasets
2. Analytics code can be cast as join-aggregate queries
   - Many similar queries that massively share computation
   - Fixpoint computation needed for model convergence
3. State-of-the-art relational data systems not scalable enough
   - Highly redundant data representation and processing
   - Tractability map for queries and analytics mostly uncharted

In-Database vs. Out-of-database Analytics



In-Database vs. Out-of-database Analytics



Complexity gap for some models: $\mathcal{O}(|DB|^{fhtw})$ vs. $\mathcal{O}(|DB|^{n})$, where n is the number of relations in the database and $fhtw \ll n$ is the fractional hypertree width of the join of all database relations.

Software Prototypes

F@Oxford and inDBLearn@LogicBlox (now Infor) support:

- ridge linear regression
- polynomial regression
- factorisation machines
- logistic regression
- support vector machines
- principal component analysis
- decision trees
- frequent itemset
- ...

Datasets continuously evolve over time

In-Database Analytics over 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 up-to-date data models

Web Analytics

Sensor Networks

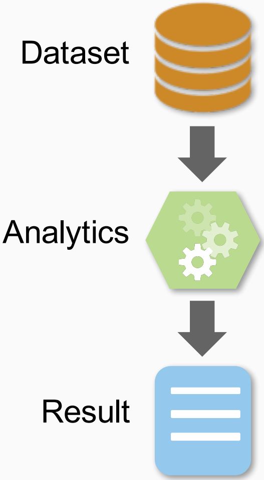
Cloud Monitoring

EVENTS → RUNTIME ENGINE → DECISION SUPPORT → ACTIONS

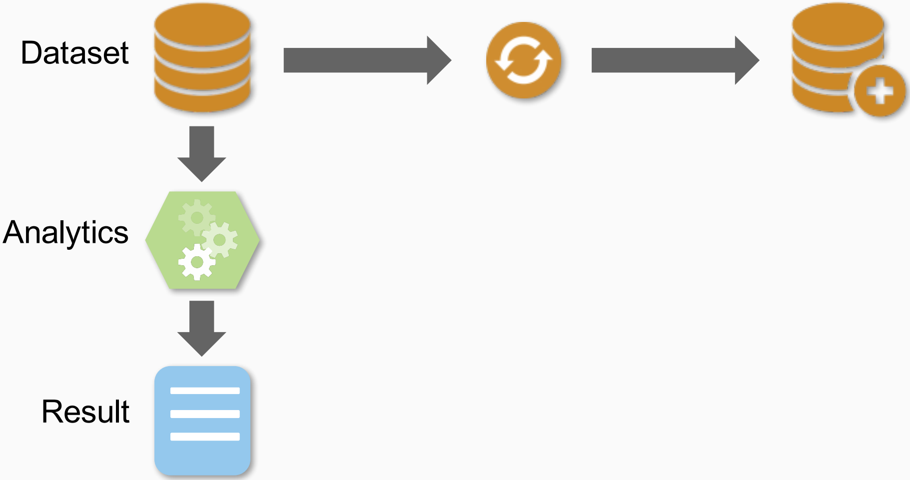
Continuously arriving data

Continuously evaluated views

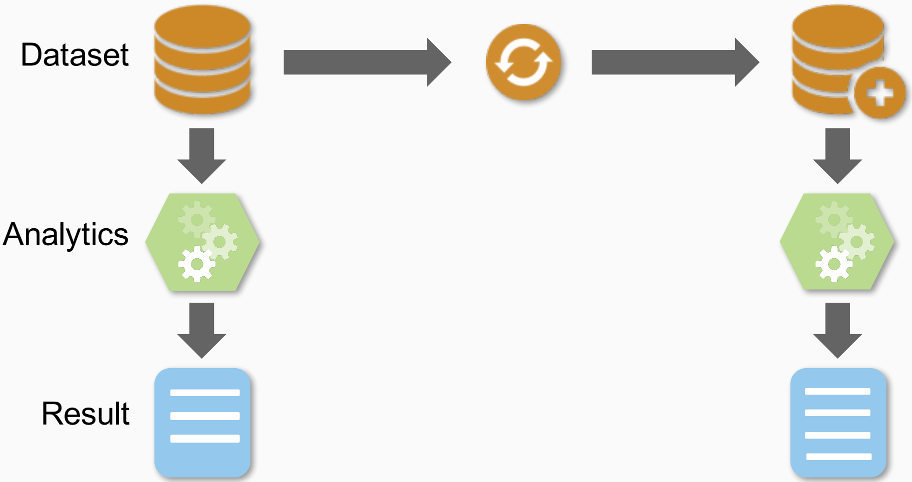
Real-Time Analytics via Incremental View Maintenance (IVM)



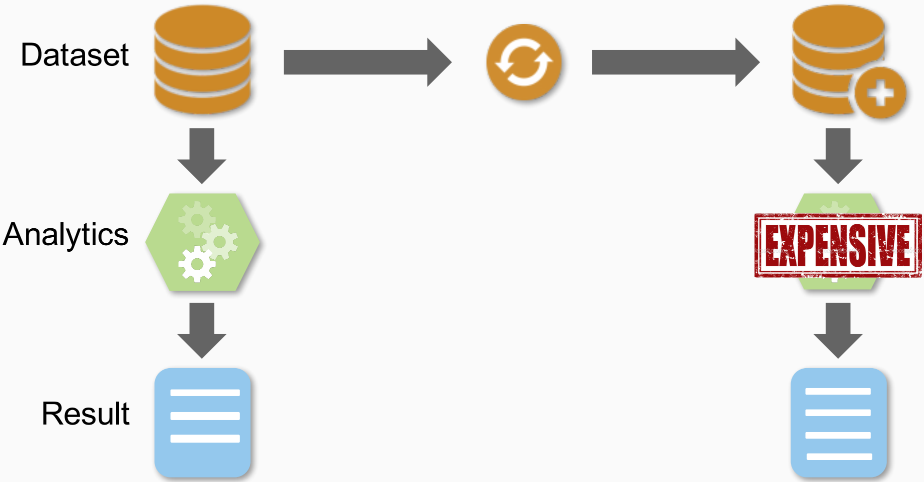
Real-Time Analytics via Incremental View Maintenance (IVM)



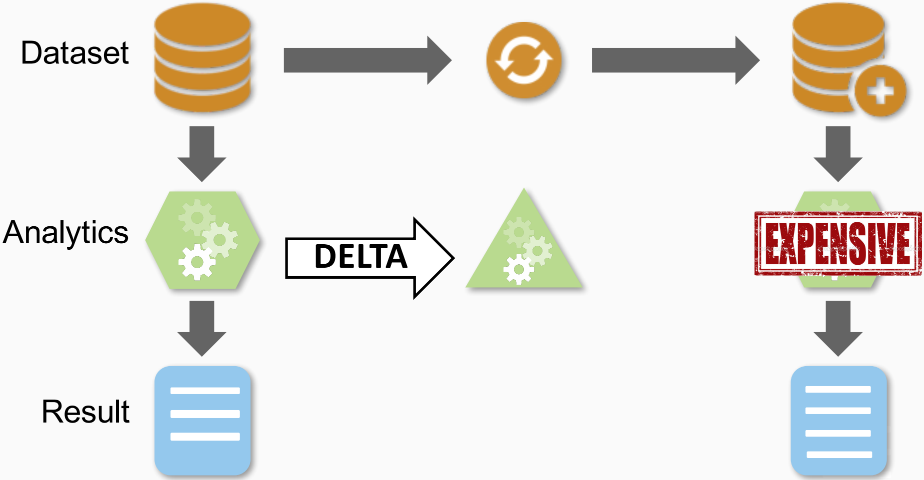
Real-Time Analytics via Incremental View Maintenance (IVM)



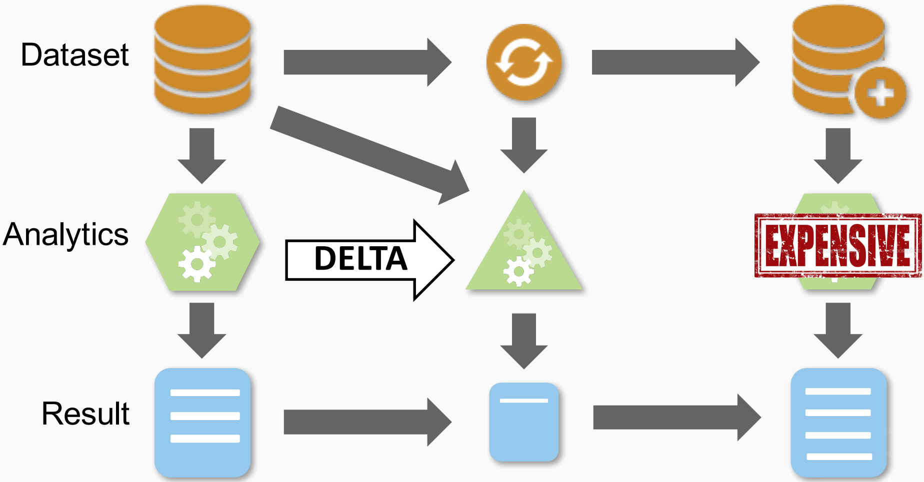
Real-Time Analytics via Incremental View Maintenance (IVM)



Real-Time Analytics via Incremental View Maintenance (IVM)



Real-Time Analytics via Incremental View Maintenance (IVM)



Unified Framework for Real-Time In-Database Analytics

Unified framework F-IVM for a host of tasks, e.g.,

- database join-aggregate queries
- gradient computation for least-squares regression models
- matrix chain multiplication

Unified Framework for Real-Time In-Database Analytics

Unified framework F-IVM for a host of tasks, e.g.,

- database join-aggregate queries
- gradient computation for least-squares regression models
- matrix chain multiplication

Key to unified computation:

- same in-database computation, coupled with
- task-specific rings $(\mathcal{D}, +, *, \mathbf{0}, \mathbf{1})$

Unified Framework for Real-Time In-Database Analytics

Unified framework F-IVM for a host of tasks, e.g.,

- database join-aggregate queries
- gradient computation for least-squares regression models
- matrix chain multiplication

Key to unified computation:

- same in-database computation, coupled with
- task-specific rings $(\mathcal{D}, +, *, \mathbf{0}, \mathbf{1})$

Key to performance: Triple-lock factorisation for

1. delta analytics, compiled to optimised C++ code
2. representation of the result
3. bulk updates via tensor decomposition techniques

Software Prototype

F-IVM@Oxford:

- Prototype implemented on top of DBToaster's backend
- Performance: Up to 2 OOM faster than classical IVM and DBToaster and up to 4 OOM less memory than DBToaster

"Concrete recipe on how to IVM the next analytic task you may face" (anonymous SIGMOD'18 reviewer)

Talk Outline

Why Real-Time In-Database Analytics?

Factorized Ring Computation

Incremental View Maintenance

Applications

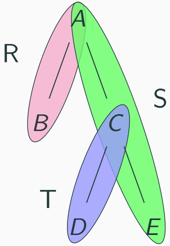
- Learning Linear Regression Models
- Factorized Representation of Conjunctive Query Results
- Matrix Chain Multiplication

First Example: COUNT Aggregate

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

```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```

How can we compute Q?

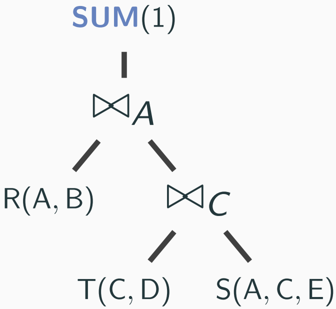


Join hypergraph

First Example: COUNT Aggregate

Naïve: compute the join and then SUM(1)

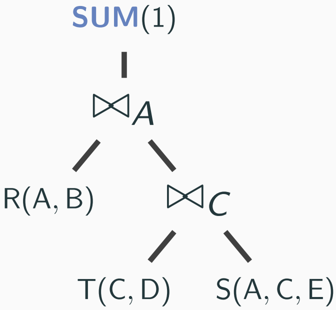
```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```



First Example: COUNT Aggregate

Naïve: compute the join and then SUM(1)

```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```



Let all relations be of size N

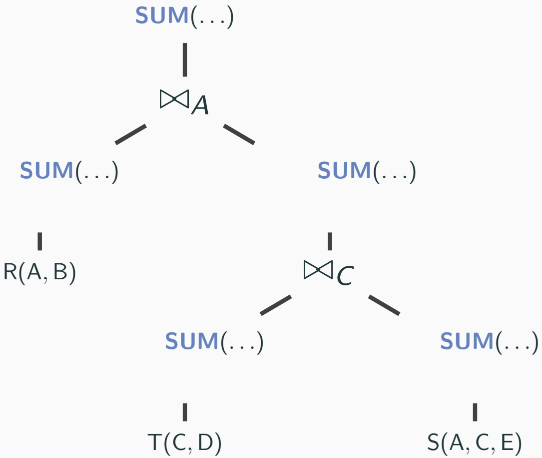
Computing Q takes $\mathcal{O}(N^3)$ time!

Can we do better?

First Example: COUNT Aggregate

Push SUM past joins
to eliminate variables

```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```



First Example: COUNT Aggregate

Push SUM past joins

to eliminate variables

```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```

$Q = \textbf{SUM}(C_B * C_C)$

$\bowtie_A$

$V_R = A, \textbf{SUM}(1) \text{ as } C_B$ **GROUP BY** A

R(A, B)

$V_{ST} = A, \textbf{SUM}(C_D * C_E) \text{ as } C_C$ **GROUP BY** A

$\bowtie_C$

$V_T = C, \textbf{SUM}(1) \text{ as } C_D$ **GROUP BY** C

T(C, D)

$V_S = A, C, \textbf{SUM}(1) \text{ as } C_E$ **GROUP BY** A, C

S(A, C, E)

First Example: COUNT Aggregate

Push SUM past joins
to eliminate variables

```
Q = SELECT SUM(1)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```

$Q = \textbf{SUM}(C_B * C_C)$

$\bowtie_A$

$V_R = A, \textbf{SUM}(1)$ as C_B
GROUP BY A

R(A, B)

$V_{ST} = A, \textbf{SUM}(C_D * C_E)$ as C_C
GROUP BY A

$\bowtie_C$

$V_T = C, \textbf{SUM}(1)$ as C_D
GROUP BY C

T(C, D)

$V_S = A, C, \textbf{SUM}(1)$ as C_E
GROUP BY A, C

S(A, C, E)

Distributivity of * over **SUM** enables this query rewriting

Q computed in $\mathcal{O}(N)$ time using a hierarchy of views!

A Slightly Different Example: SUM Aggregate

SUM over products of B and C

```
Q = SELECT SUM(B * C)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```

A Slightly Different Example: SUM Aggregate

SUM over products of B and C

```
Q = SELECT SUM(B * C)
    FROM R NATURAL JOIN S
           NATURAL JOIN T
```

$Q = \textbf{SUM}(S_B * S_C)$

$\bowtie_A$

$V_R = A, \textbf{SUM}(B)$ as S_B **GROUP BY** A

$R(A, B)$

$V_{ST} = A, \textbf{SUM}(S_D * S_E * C)$ as S_C **GROUP BY** A

$\bowtie_C$

$V_T = C, \textbf{SUM}(1)$ as S_D **GROUP BY** C

$T(C, D)$

$V_S = A, C, \textbf{SUM}(1)$ as S_E **GROUP BY** A, C

$S(A, C, E)$

Factorized evaluation

Reuse counts of D and E when joining on C

Multiply by C only after joining on C

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

$$Q = \textbf{SUM}(S_B * S_C * g_A(A))$$

$$\bowtie_A$$

$V_R = A, \textbf{SUM}(g_B(B))$ as S_B
GROUP BY A

R(A, B)

$V_{ST} = A, \textbf{SUM}(S_D * S_E * g_C(C))$ as S_C
GROUP BY A

$$\bowtie_C$$

$V_T = C, \textbf{SUM}(g_E(E))$ as S_D
GROUP BY C

T(C, D)

$V_S = A, C, \textbf{SUM}(g_D(D))$ as S_E
GROUP BY A, C

S(A, C, E)

More General Example: SUM Aggregate

```
Q = SELECT SUM(gA(A) * gB(B) * gC(C) * gD(D) * gE(E))
    FROM R NATURAL JOIN S NATURAL JOIN T
```

$Q = \textbf{SUM}(S_B * S_C * g_A(A))$

Join on & eliminate one variable at a time

$\bowtie_A$

$V_R = A, \textbf{SUM}(g_B(B))$ as S_B **GROUP BY** A

R(A, B)

$V_{ST} = A, \textbf{SUM}(S_D * S_E * g_C(C))$ as S_C **GROUP BY** A

$\bowtie_C$

$V_T = C, \textbf{SUM}(g_E(E))$ as S_D **GROUP BY** C

eliminate D

T(C, D)

$V_S = A, C, \textbf{SUM}(g_D(D))$ as S_E **GROUP BY** A, C

S(A, C, E)

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

$Q = \mathbf{SUM}(S_B * S_C * g_A(A))$

$\bowtie_A$

$V_R = A, \mathbf{SUM}(g_B(B))$ as S_B
GROUP BY A

R(A, B)

$V_{ST} = A, \mathbf{SUM}(S_D * S_E * g_C(C))$ as S_C
GROUP BY A

$\bowtie_C$

$V_T = C, \mathbf{SUM}(g_E(E))$ as S_D
GROUP BY C

eliminate D

T(C, D)

$V_S = A, C, \mathbf{SUM}(g_D(D))$ as S_E
GROUP BY A, C

eliminate E

S(A, C, E)

Join on & eliminate one variable at a time

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

$Q = \textbf{SUM}(S_B * S_C * g_A(A))$

Join on & eliminate one variable at a time

$\bowtie_A$

$V_R = A, \textbf{SUM}(g_B(B))$ as S_B
GROUP BY A

R(A, B)

$V_{ST} = A, \textbf{SUM}(S_D * S_E * g_C(C))$ as S_C
GROUP BY A

eliminate C

$\bowtie_C$

$V_T = C, \textbf{SUM}(g_E(E))$ as S_D
GROUP BY C

eliminate D

T(C, D)

$V_S = A, C, \textbf{SUM}(g_D(D))$ as S_E
GROUP BY A, C

eliminate E

S(A, C, E)

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

Join on & eliminate one variable at a time

$Q = \mathbf{SUM}(S_B * S_C * g_A(A))$

$\bowtie_A$

$V_R = A, \mathbf{SUM}(g_B(B))$ as S_B
GROUP BY A

eliminate B

R(A, B)

$V_{ST} = A, \mathbf{SUM}(S_D * S_E * g_C(C))$ as S_C
GROUP BY A

eliminate C

$\bowtie_C$

$V_T = C, \mathbf{SUM}(g_E(E))$ as S_D
GROUP BY C

eliminate D

T(C, D)

$V_S = A, C, \mathbf{SUM}(g_D(D))$ as S_E
GROUP BY A, C

eliminate E

S(A, C, E)

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

Join on & eliminate one variable at a time

$$Q = \mathbf{SUM}(S_B * S_C * g_A(A))$$

eliminate A

$$\bowtie_A$$

$$V_R = A, \mathbf{SUM}(g_B(B)) \text{ as } S_B \quad \mathbf{GROUP\ BY}\ A$$

eliminate B

R(A, B)

$$V_{ST} = A, \mathbf{SUM}(S_D * S_E * g_C(C)) \text{ as } S_C \quad \mathbf{GROUP\ BY}\ A$$

eliminate C

$$\bowtie_C$$

$$V_T = C, \mathbf{SUM}(g_E(E)) \text{ as } S_D \quad \mathbf{GROUP\ BY}\ C$$

eliminate D

T(C, D)

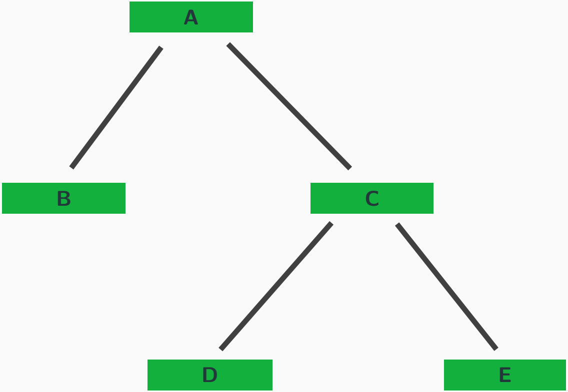
$$V_S = A, C, \mathbf{SUM}(g_D(D)) \text{ as } S_E \quad \mathbf{GROUP\ BY}\ A, C$$

eliminate E

S(A, C, E)

Query Evaluation Plans using Variable Orders

One variable order for
$R(A, B)$, $S(A, C, E)$, $T(C, D)$

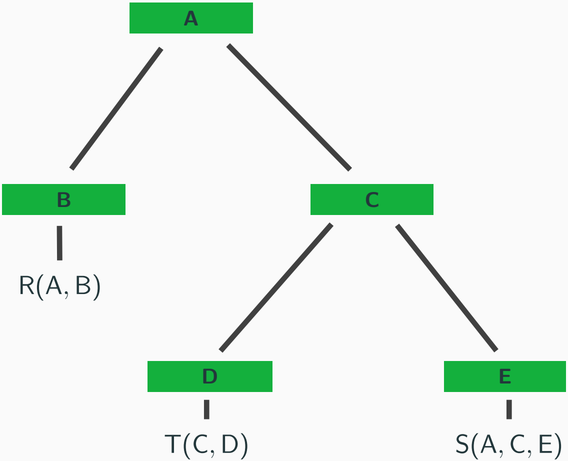


Query Evaluation Plans using Variable Orders

One variable order for
$R(A,B)$, $S(A,C,E)$, $T(C,D)$

Tree of query variables

Variables of a relation lie on a root-to-leaf path



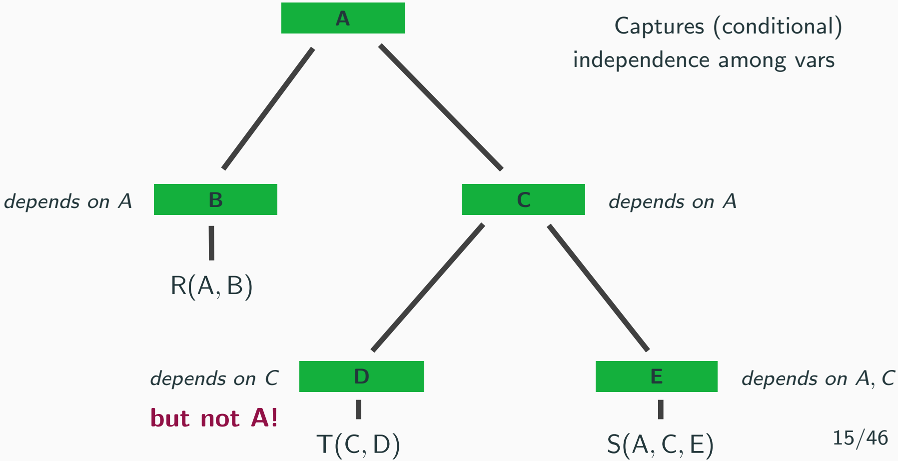
Query Evaluation Plans using Variable Orders

One variable order for
$R(A, B)$, $S(A, C, E)$, $T(C, D)$

Tree of query variables

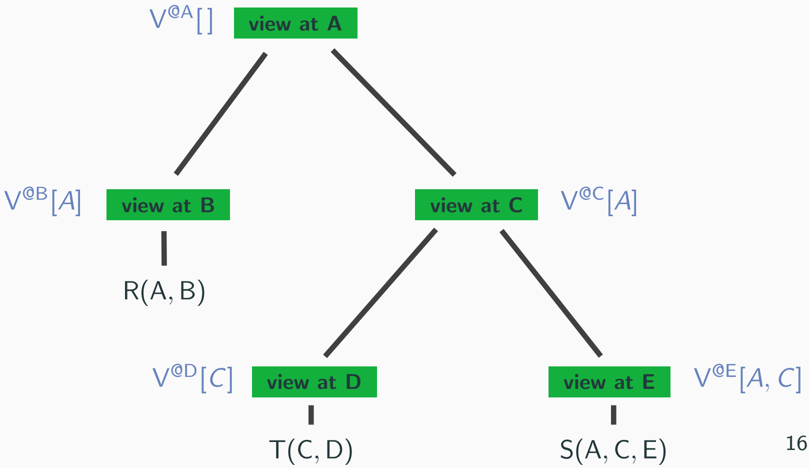
Variables of a relation lie on a root-to-leaf path

Captures (conditional) independence among vars



View Trees

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



View Trees

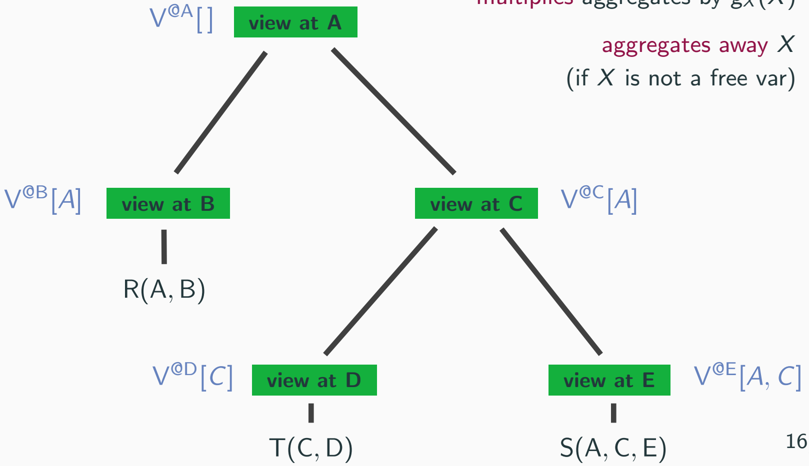
Create a view at each var X with schema depends(X)

View at variable X:

joins its child views

multiplies aggregates by $g_X(X)$

aggregates away X (if X is not a free var)



More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

$Q = \textbf{SUM}(S_B * S_C * g_A(A))$

$\bowtie_A$

$\mathrm{V}^{@\mathrm{B}} = A, \textbf{SUM}(g_B(B))$ as S_B
GROUP BY A

R(A, B)

$\mathrm{V}^{@\mathrm{C}} = A, \textbf{SUM}(S_D * S_E * g_C(C))$ as S_C
GROUP BY A

$\bowtie_C$

$\mathrm{V}^{@\mathrm{D}} = C, \textbf{SUM}(g_E(E))$ as S_D
GROUP BY C

T(C, D)

$\mathrm{V}^{@\mathrm{E}} = A, C, \textbf{SUM}(g_D(D))$ as S_E
GROUP BY A, C

S(A, C, E)

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

Imagine aggregate values are of type $\mathcal{R}$

$$g_X : \mathrm{Dom}(X) \to \mathcal{R}$$

Can we evaluate Q using the query plan from before?

More General Example: SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

Imagine aggregate values are of type $\mathcal{R}$

$$g_X : \mathsf{Dom}(X) \to \mathcal{R}$$

Can we evaluate Q using the query plan from before?

Yes(!), but we need to:

- Define $*$ and $+$ binary operators in $\mathcal{R}$
- Define zero in $\mathcal{R}$ (for initial values)
- Define one in $\mathcal{R}$ (e.g., if X is not used, $g_X(x) = 1$)
- Ensure distributivity of $*$ over $+$

Rings

- A ring $(\mathcal{R}, +, *, \mathbf{0}, \mathbf{1})$ is a set $\mathcal{R}$ 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	$\exists - a \in \mathcal{R} : a + (-a) = (-a) + a = \mathbf{0}$
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

Factorized Ring Computation

- Relations are functions
  - mapping keys (tuples) 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
  - Operations: union, join, and variable marginalization
  - More expressiveness via application-specific rings
- Query evaluation
  - using *view trees* shown before

More General SUM Aggregate

```
Q = SELECT SUM( gA(A) * gB(B) * gC(C) * gD(D) * gE(E) )
    FROM R NATURAL JOIN S NATURAL JOIN T
```

In our formalism:

$$Q = \underbrace{\bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E}_{\textit{variable marginalization}} \underbrace{(\ R[A,B] \otimes S[A,C,E] \otimes T[C,D]\ )}_{\textit{natural joins}}$$

Intuition: Relation payloads carry out the summation!

Marginalization of X applies g_X, sums payloads, projects away X

Join multiplies payloads of matching tuples

Query Operators

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

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

Query Operators

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

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

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

Query Operators

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

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

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

Query Operators

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

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

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

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

Union $\uplus$

A	B	$\rightarrow$	$(\mathrm{R} \uplus \mathrm{S})[A, B]$
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	$\rightarrow$	$((\mathrm{R} \uplus \mathrm{S}) \otimes \mathrm{T})[A, B, C]$
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	$\rightarrow$	$s_2 * t_2$

Query Operators

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

A	B	→	R[A, B]
a_1	b_1	→	r_1
a_2	b_1	→	r_2

A	B	→	S[A, B]
a_2	b_1	→	s_1
a_3	b_2	→	s_2

B	C	→	T[B, C]
b_1	c_1	→	t_1
b_2	c_2	→	t_2

Union $\uplus$

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

Join $\otimes$

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

Query Operators

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

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

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 $\bigoplus_A$

for a given
$g_A : \mathrm{Dom}(A) \rightarrow \mathcal{R}$

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

General Query Form

```
Q = SELECT X_1, ..., X_f , SUM( g_{f+1}(X_{f+1}) * ··· * g_m(X_m) )
    FROM R_1 NATURAL JOIN ... NATURAL JOIN R_n
    GROUP BY X_1, ..., X_f
```

Expressed as Functional Aggregate Query:

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

where:

- Relations $\mathrm{R_1}, \ldots, \mathrm{R_n}$ are defined over variables $X_1, \ldots, X_m$
- $X_1, \ldots, X_f$ are free variables
- $\mathrm{R_i}$ maps keys over schema $\mathcal{S}_i$ to payloads in a ring $(\mathcal{R}, +, *, \mathbf{0}, \mathbf{1})$
- Aggregations $\bigoplus_{X_{f+1}}, \ldots, \bigoplus_{X_m}$ use functions $\mathrm{g_{f+1}}, \ldots, \mathrm{g_m}$

Talk Outline

Why Real-Time In-Database Analytics?

Factorized Ring Computation

Incremental View Maintenance

Applications

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

Incremental Computation

- Maintain query results under updates to the input relations

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

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

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

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

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

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

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

Consider our running examples

Maintain the query result under updates to T

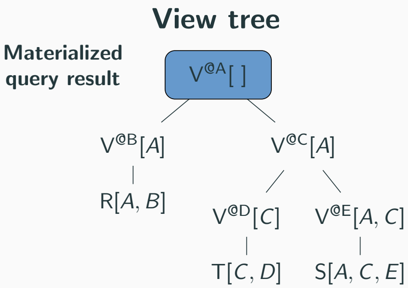
View tree

- $V^{@A}[\,]$
  - $V^{@B}[A]$
    - $R[A, B]$
  - $V^{@C}[A]$
    - $V^{@D}[C]$
      - $T[C, D]$
    - $V^{@E}[A, C]$
      - $S[A, C, E]$

Delta Propagation

Consider our running examples

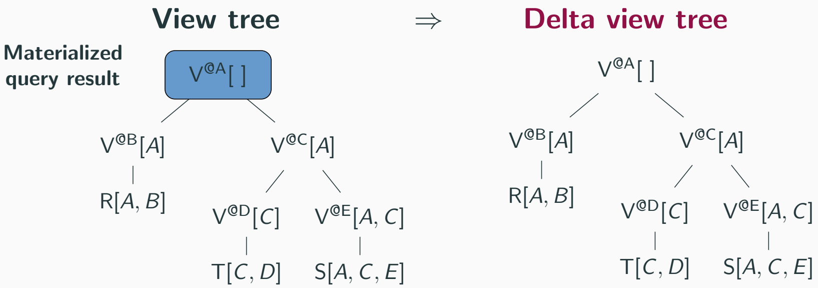
Maintain the query result under updates to T



Delta Propagation

Consider our running examples

Maintain the query result under updates to T

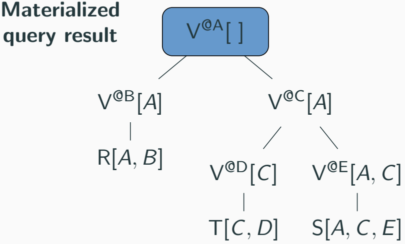


Delta Propagation

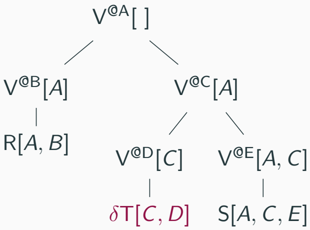
Consider our running examples

Maintain the query result under updates to T

View tree

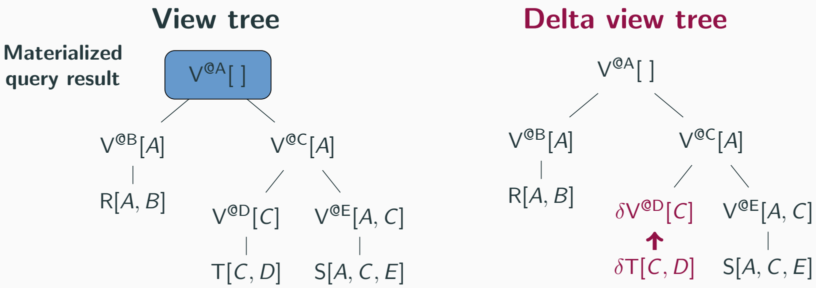


Delta view tree



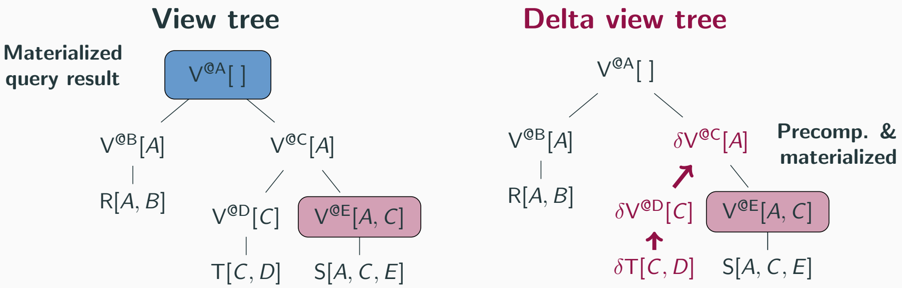
Consider our running examples

Maintain the query result under updates to T



Consider our running examples

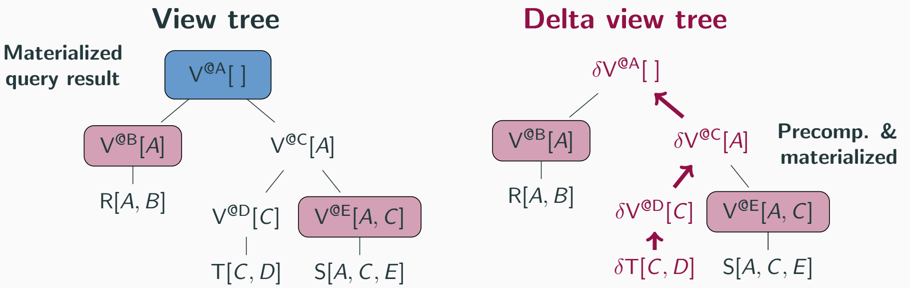
Maintain the query result under updates to T



Delta Propagation

Consider our running examples

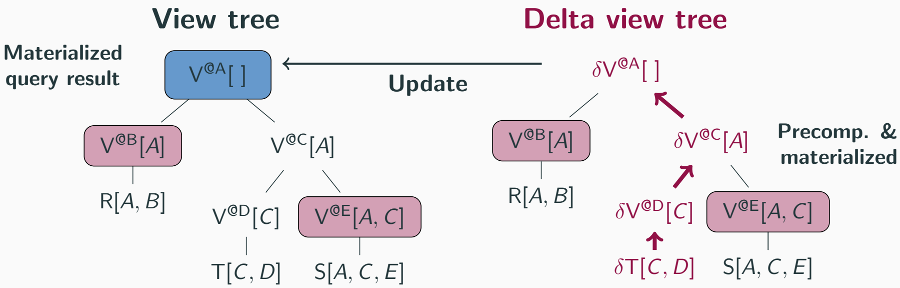
Maintain the query result under updates to T



Delta Propagation

Consider our running examples

Maintain the query result under updates to T

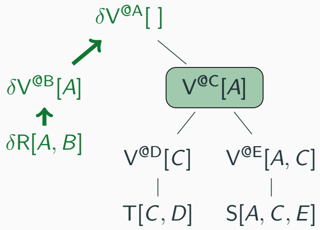


Updates to Multiple Relations

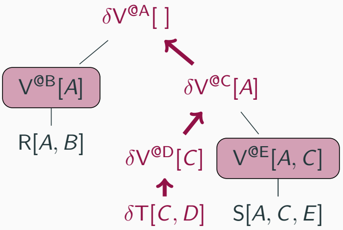
Maintain the query result for updates to R and T

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

Delta view tree (for R)



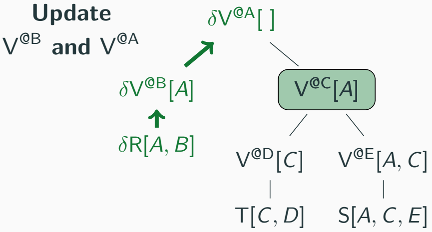
Delta view tree (for T)



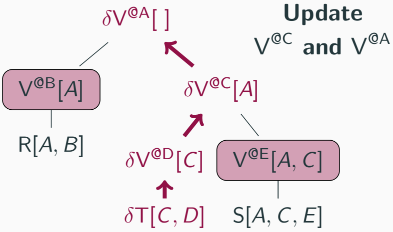
Maintain the query result for updates to R and T

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

Delta view tree (for R)



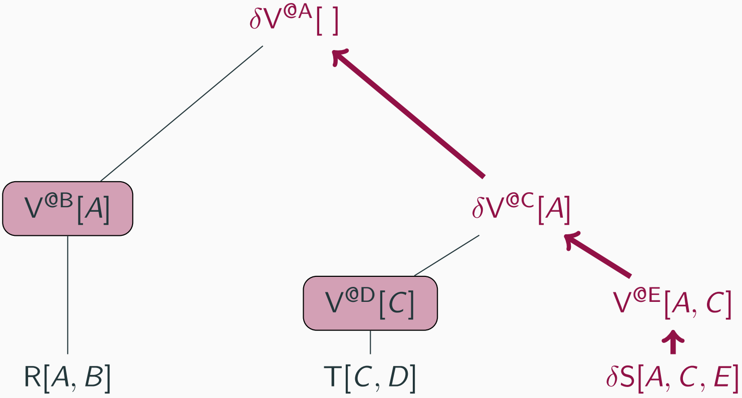
Delta view tree (for T)



Factorizable Bulk Updates

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

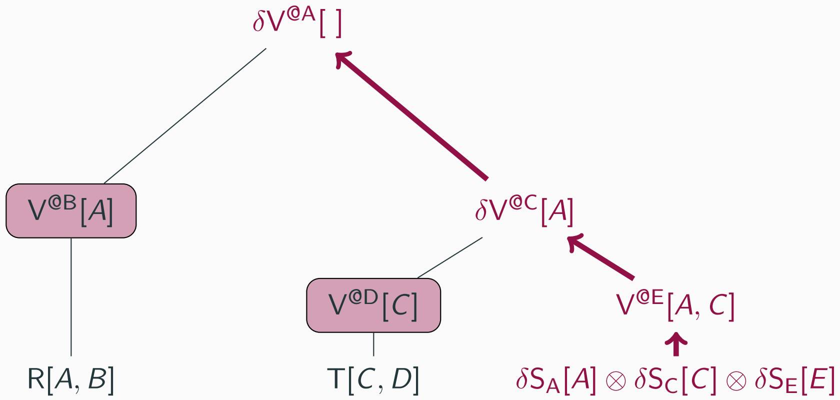
We may then factorize subsequent updates up the delta tree



Factorizable Bulk Updates

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

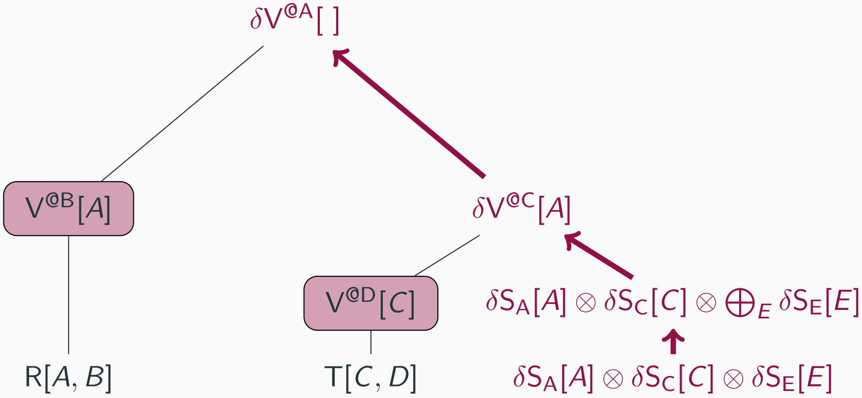
We may then factorize subsequent updates up the delta tree



Factorizable Bulk Updates

Assume update $\delta\mathsf{S}[A, C, E]$ factorizes as $\delta\mathsf{S}_\mathsf{A}[A] \otimes \delta\mathsf{S}_\mathsf{C}[C] \otimes \delta\mathsf{S}_\mathsf{E}[E]$

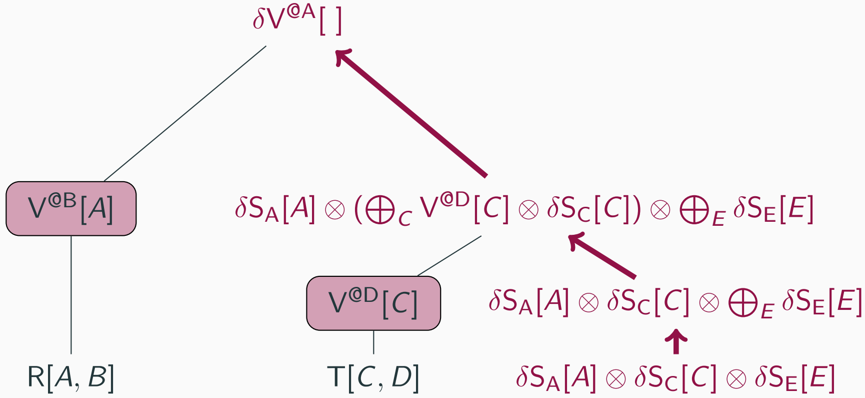
We may then factorize subsequent updates up the delta tree



Factorizable Bulk Updates

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

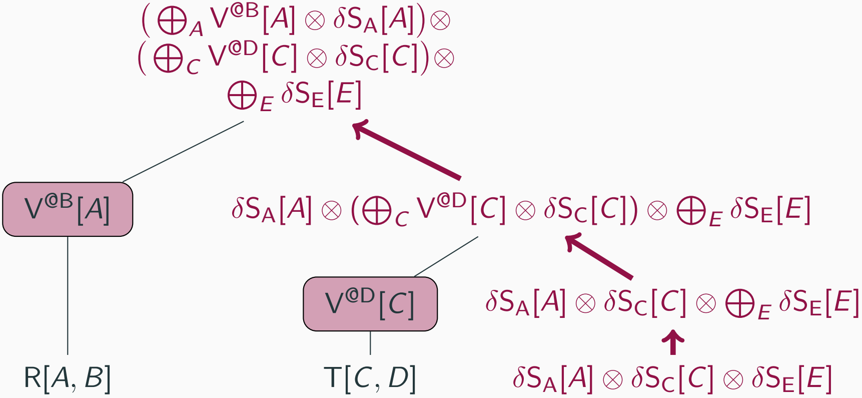
We may then factorize subsequent updates up the delta tree



Factorizable Bulk Updates

Assume update $\delta\mathrm{S}[A, C, E]$ factorizes as $\delta\mathrm{S}_\mathrm{A}[A] \otimes \delta\mathrm{S}_\mathrm{C}[C] \otimes \delta\mathrm{S}_\mathrm{E}[E]$

We may then factorize subsequent updates up the delta tree



Talk Outline

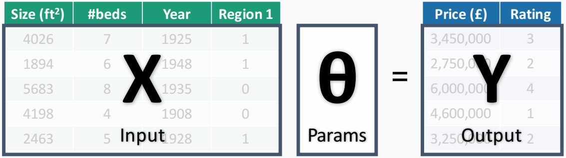
Applications

Our framework can capture a host of problems using task-specific rings

- Gradient computation for learning regression models
- Factorized representation of results of conjunctive queries
- Matrix chain multiplication
- Group-by aggregation over joins (we've seen this already)

Next: zoom in the first problem above
(Ask me about the other ones!)

- Find model parameters Θ best satisfying:



- Iterative gradient computation:

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

- Matrices $\mathbf{X}^\mathbf{T}\,\mathbf{X}$ and $\mathbf{X}^\mathbf{T}\,\mathbf{Y}$ computed once for all iterations
  - Compute $SUM(X_i \cdot X_j)$, $SUM(X_i)$, and $SUM(1)$ for variables X_i and X_j
  - We assume in this talk that all variables are continuous

Learning Linear Regression Models over Joins

Compute $\mathbf{X}^\mathsf{T}\mathbf{X}$ where $\mathbf{X}$ is the join of the 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

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

but a different payload ring and different functions g_X!

Linear Regression Ring

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

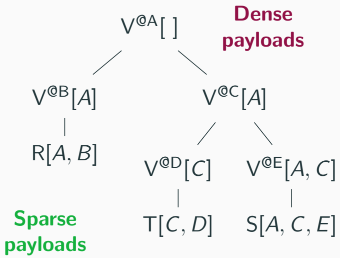
$$\Big(\ \text{COUNT}, \quad \text{vector of } SUM(X_i), \quad \text{matrix of } SUM(X_i \cdot X_j)\ \Big)$$

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

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

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

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



Function g_{X_j} for variable X_j

$g_{X_j}(x) = (1, \mathbf{s}, \mathbf{Q})$ 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

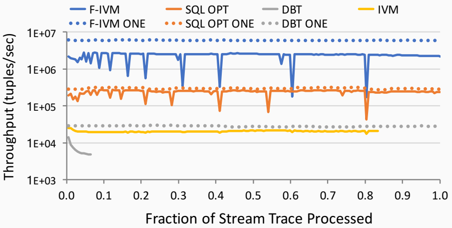
Streaming dataset with 5 relations

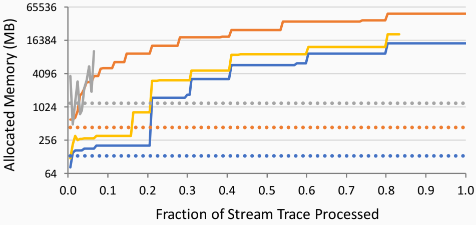
The natural join has 43 variables

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)





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

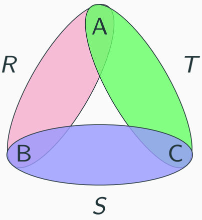
Our IVM framework can accommodate any ring

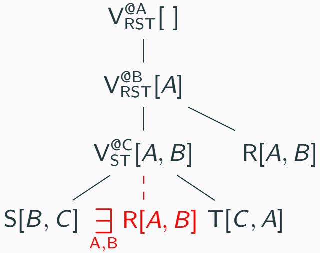
As My Girl Beyoncé Repeatedly Said..

Thank you!

The Triangle Query

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





Relational Data Ring

- Set of relations over $\mathcal{R}$ with $\uplus$ and $\otimes$ forms a ring of relations
  - Relation $\mathbf{0}$ maps every tuple to $\mathbf{0} \in \mathcal{R}$
  - Relation $\mathbf{1}$ maps the empty tuple to $\mathbf{1} \in \mathcal{R}$, others to $\mathbf{0} \in \mathcal{R}$
- Payload: Relations over $\mathcal{R} = \mathbb{Z}$ with the same schema!

A	B	$\rightarrow$	R[A, B]
a_1	b_1	$\rightarrow$	$\begin{array}{\|l} C \\ \hline c_1 \rightarrow 1 \\ c_2 \rightarrow 1 \end{array}$
a_2	b_1	$\rightarrow$	$\begin{array}{\|l} C \\ \hline c_3 \rightarrow 1 \end{array}$

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 relations with relational payloads

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

- Lift (aggregate) functions:

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

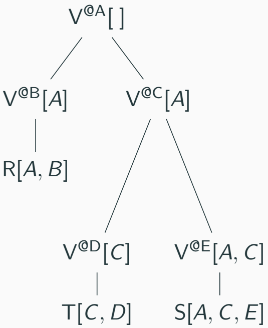
Listing Representation of Conjunctive Query Results

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

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

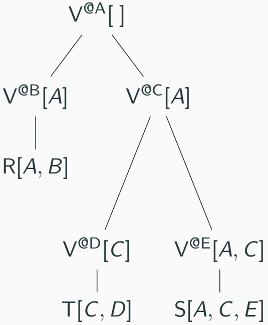
$$Q(A,B,C,D) = R(A,B), S(A,C,E), T(C,D)$$

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

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

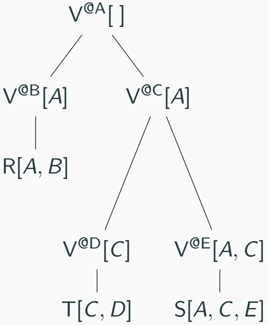
A	B	→ R[A,B]
a_1	b_1 →	$() \to 1$
a_1	b_2 →	$() \to 1$
a_2	b_3 →	$() \to 1$
a_3	b_4 →	$() \to 1$

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]
$a_1 \to$	B: $b_1 \to 1$, $b_2 \to 1$
$a_2 \to$	B: $b_3 \to 1$
$a_3 \to$	B: $b_4 \to 1$

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



Listing Representation of Conjunctive Query Results

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

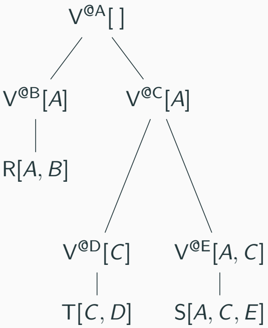
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

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	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

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

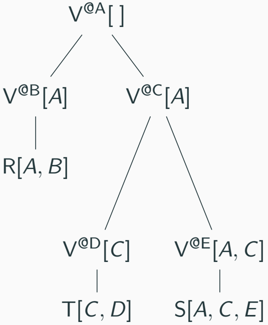
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

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 D $c_1\ d_1 \to 2$ $c_2\ d_2 \to 1$ $c_2\ d_3 \to 1$
a_2 →	C D $c_2\ d_2 \to 1$ $c_2\ d_3 \to 1$

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]

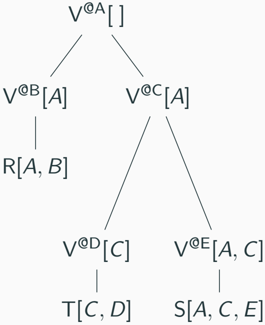
C	D	→ T[C,D]
c_1	d_1	$() \to 1$
c_2	d_2	$() \to 1$
c_2	d_3	$() \to 1$
c_3	d_4	$() \to 1$

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	$\to 2$
	a_1	b_1	c_2	d_2	$\to 1$
	a_1	b_1	c_2	d_3	$\to 1$
	a_1	b_2	c_1	d_1	$\to 2$
	a_1	b_2	c_2	d_2	$\to 1$
	a_1	b_2	c_2	d_3	$\to 1$
	a_2	b_3	c_2	d_2	$\to 1$
	a_2	b_3	c_2	d_3	$\to 1$

A → V@C[A]

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	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]

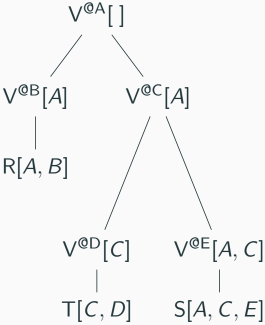
C	D	→	T[C,D]
c_1	d_1	→	$() \to 1$
c_2	d_2	→	$() \to 1$
c_2	d_3	→	$() \to 1$
c_3	d_4	→	$() \to 1$

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	→	$c_1\ d_1 \to 2$ $c_2\ d_2 \to 1$ $c_2\ d_3 \to 1$
a_2	→	$c_2\ d_2 \to 1$ $c_2\ d_3 \to 1$

A C → V@E[A,C]

A	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

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

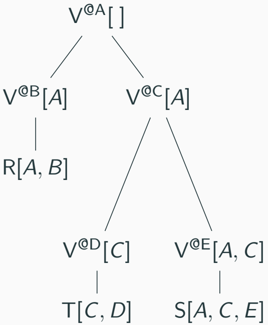
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

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	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

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

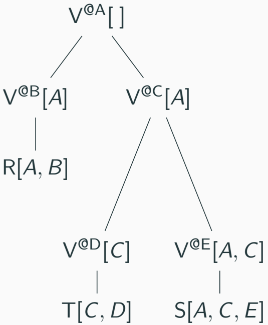
A	B	→ R[A,B]
a_1	b_1	→ $() \to 1$
a_1	b_2	→ $() \to 1$
a_2	b_3	→ $() \to 1$
a_3	b_4	→ $() \to 1$

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

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

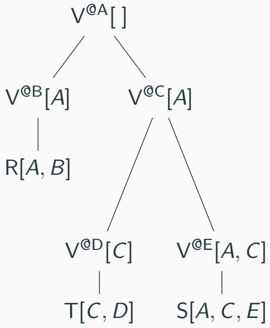
A	B	→ R[A,B]
a_1	b_1	$() \to 1$
a_1	b_2	$() \to 1$
a_2	b_3	$() \to 1$
a_3	b_4	$() \to 1$

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



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

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

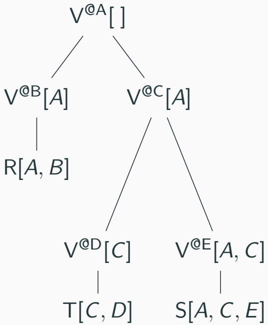
A	C	→ $\mathrm{V}^{@E}$[A,C]
a_1	c_1	$() \to 2$
a_1	c_2	$() \to 1$
a_2	c_2	$() \to 1$

Factorized Representation of Conjunctive Query Results

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

Constant Delay Enumeration

foreach a **in** $V^{@A}$
  foreach b **in** $V^{@B}$
    foreach c **in** $V^{@C}$
      foreach d **in** $V^{@D}$
        output (a,b,c,d)



() → V@A[ ]

	A
() →	$a_1 \to 8$
	$a_2 \to 2$

A → V@B[A]

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

A → V@C[A]

	C
$a_1 \to$	$c_1 \to 2$
	$c_2 \to 2$
$a_2 \to$	$c_2 \to 2$

C → V@D[C]

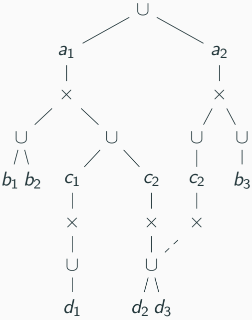
	D
$c_1 \to$	$d_1 \to 1$
$c_2 \to$	$d_2 \to 1$
	$d_3 \to 1$
$c_3 \to$	$d_4 \to 1$

A C → V@E[A,C]

$a_1\ c_1 \to$	$() \to 2$
$a_1\ c_2 \to$	$() \to 1$
$a_2\ c_2 \to$	$() \to 1$

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

Factorized Join

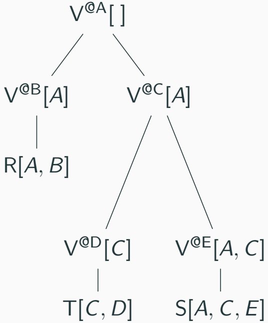


$A \to V^{@B}[A]$

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

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

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



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

	A
$() \to$	$a_1 \to 8$
	$a_2 \to 2$

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

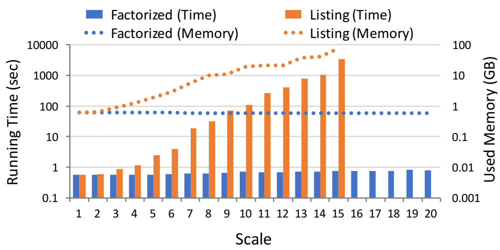
	C
$a_1 \to$	$c_1 \to 2$
	$c_2 \to 2$
$a_2 \to$	$c_2 \to 2$

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

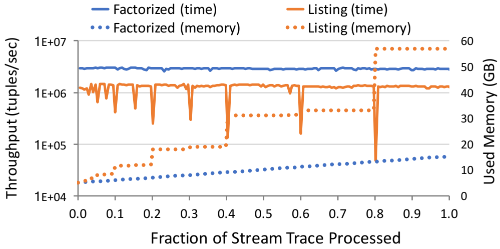
$a_1\ c_1 \to$	$() \to 2$
$a_1\ c_2 \to$	$() \to 1$
$a_2\ c_2 \to$	$() \to 1$

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 $\mathcal{R}$ ($i \in [n]$)

- Modeled as relations $\mathsf{A_i}[X_i, X_{i+1}]$ with payloads carrying matrix values in $\mathcal{R}$

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

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

where each lift function $g_{X_j}(X_j)$ maps any key to payload $\mathbf{1} \in \mathcal{R}$

Factorized Matrix Updates

Matrix changes

Single-value change $\Rightarrow$ vector outer product
$\delta \mathsf{A_i}[X_i, X_{i+1}] = \mathsf{u}[X_i] \otimes \mathsf{v}[X_{i+1}]$

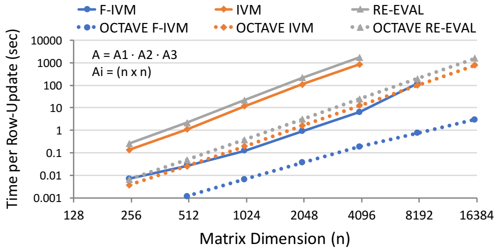
Several-values change $\Rightarrow$ sum of vector outer products
$\delta \mathsf{A_i}[X_i, X_{i+1}] = \uplus_{k \in [r]} \mathsf{u_k}[X_i] \otimes \mathsf{v_k}[X_{i+1}]$

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 outer product



Update to A_2 expressed as sum of r outer products