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

- Feature extraction
- Query Engine
- Optimised join-aggregate queries
- Materialised result
- ML tool
- $\vec{\theta}^*$
- Gradient-descent Trainer
- Model reformulation
- Model

Complexity gap for some models:

$O(|DB| fhtw)$ vs. $O(|DB| n)$, 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.
In-Database vs. Out-of-database Analytics

Complexity gap for some models: $O(|DB|^{fhtw})$ vs. $O(|DB|^n)$, 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
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
Real-Time Analytics via Incremental View Maintenance (IVM)

Dataset → Analytics → Result
Real-Time Analytics via Incremental View Maintenance (IVM)

Dataset → Analytics → Result
Real-Time Analytics via Incremental View Maintenance (IVM)
Real-Time Analytics via Incremental View Maintenance (IVM)

Dataset → Analytics → Result

EXPENSIVE
Real-Time Analytics via Incremental View Maintenance (IVM)

Dataset → DELTA → Result

Analytics

EXPENSIVE
Real-Time Analytics via Incremental View Maintenance (IVM)
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 \((\mathcal{D}, +, *, 0, 1)\)
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 \((D, +, \ast, 0, 1)\)

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
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:
\( R(A, B), S(A, C, E), T(C, D) \)

\[
Q = \text{SELECT SUM}(1)
\text{FROM } R \text{ NATURAL JOIN } S
\text{ NATURAL JOIN } T
\]

How can we compute \( Q \)?
First Example: COUNT Aggregate

Naïve: compute the join and then SUM(1)

\[ Q = \text{SELECT } \text{SUM}(1) \]
\[ \text{FROM } R \text{ NATURAL JOIN } S \text{ NATURAL JOIN } T \]

Let all relations be of size \( N \)

Computing \( Q \) takes \( O(N^3) \) time!

Can we do better?
First Example: COUNT Aggregate

**Naïve:** compute the join and then \( \text{SUM}(1) \)

\[
Q = \text{SELECT SUM}(1) \\
\text{FROM R NATURAL JOIN S} \\
\text{NATURAL JOIN T}
\]

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

\[ Q = \text{SELECT SUM}(1) \text{ FROM } R \text{ NATURAL JOIN } S \text{ NATURAL JOIN } T \]
First Example: COUNT Aggregate

Push SUM past joins
to eliminate variables

\[ Q = \text{SELECT } \text{SUM}(1) \]
\[ \text{FROM } R \text{ NATURAL JOIN } S \text{ NATURAL JOIN } T \]

\[ Q = \text{SUM}(C_B \ast C_C) \]

\[ V_R = A, \text{SUM}(1) \text{ as } C_B \]
\[ V_{ST} = A, \text{SUM}(C_D \ast C_E) \text{ as } C_C \]

\[ V_T = C, \text{SUM}(1) \text{ as } C_D \text{ GROUP BY } C \]
\[ V_S = A, C, \text{SUM}(1) \text{ as } C_E \text{ GROUP BY } A, C \]

\[ \text{GROUP BY } A \]
\[ \text{GROUP BY } A \]

\[ R(A, B) \]
\[ \text{GROUP BY } C \]
\[ T(C, D) \]
\[ \text{GROUP BY } A, C \]
\[ S(A, C, E) \]
Push SUM past joins to eliminate variables

\[ Q = \text{SELECT SUM}(1) \]
\[ \text{FROM R NATURAL JOIN S NATURAL JOIN T} \]

Distributivity of $*$ over $\text{SUM}$ enables this query rewriting

\[ Q \text{ computed in } \mathcal{O}(N) \text{ time using a hierarchy of views!} \]
A Slightly Different Example: SUM Aggregate

SUM over products of B and C

\[ Q = \text{SELECT } \text{SUM}(B \times C) \]
\[ \text{FROM } R \text{ NATURAL JOIN } S \]
\[ \text{NATURAL JOIN } T \]
A Slightly Different Example: SUM Aggregate

SUM over products of B and C

\[ Q = \text{SELECT} \ \text{SUM}(B \times C) \]
\[ \text{FROM} \ R \ \text{NATURAL JOIN} \ S \ \text{NATURAL JOIN} \ T \]

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

\[ Q = \text{SELECT } \text{SUM} \left( g_A(A) \cdot g_B(B) \cdot g_C(C) \cdot g_D(D) \cdot g_E(E) \right) \]

\[ \text{FROM } R \text{ NATURAL JOIN } S \text{ NATURAL JOIN } T \]

\[ Q = \text{SUM}(S_B \cdot S_C \cdot g_A(A)) \]

\[ V_R = A, \ \text{SUM}(g_B(B)) \text{ as } S_B \]
\[ V_{ST} = A, \ \text{SUM}(S_D \cdot S_E \cdot g_C(C)) \text{ as } S_C \]
\[ \text{GROUP BY } A \]

\[ V_T = C, \ \text{SUM}(g_E(E)) \text{ as } S_D \]
\[ V_S = A, C, \ \text{SUM}(g_D(D)) \text{ as } S_E \]
\[ \text{GROUP BY } C \]
\[ \text{GROUP BY } A, C \]

\[ \text{GROUP BY } A \]

\[ R(A, B) \]
\[ T(C, D) \]
\[ S(A, C, E) \]
More General Example: SUM Aggregate

Q = SELECT SUM ( g_A(A) * g_B(B) * g_C(C) * g_D(D) * g_E(E) )
FROM R NATURAL JOIN S NATURAL JOIN T

Q = SUM(S_B * S_C * g_A(A))

Join on & eliminate one variable at a time

V_R = A, SUM(g_B(B)) as S_B
GROUP BY A

V_ST = A, SUM(S_D * S_E * g_C(C)) as S_C
GROUP BY A

V_T = C, SUM(g_E(E)) as S_D
GROUP BY C

eliminate D

T(C, D)

V_S = A, C, SUM(g_D(D)) as S_E
GROUP BY A, C

S(A, C, E)
More General Example: SUM Aggregate

Q = SELECT SUM ( \( g_A(A) \times g_B(B) \times g_C(C) \times g_D(D) \times g_E(E) \) ) FROM R NATURAL JOIN S NATURAL JOIN T

Q = SUM ( \( S_B \times S_C \times g_A(A) \) )

Join on & eliminate one variable at a time

\( V_R = A, \text{SUM}(g_B(B)) \) as \( S_B \)

GROUP BY \( A \)

\( V_{ST} = A, \text{SUM}(S_D \times S_E \times g_C(C)) \) as \( S_C \)

GROUP BY \( A \)

\( V_T = C, \text{SUM}(g_E(E)) \) as \( S_D \)

GROUP BY \( C \)

eliminate \( D \)

\( T(C, D) \)

\( V_S = A, C, \text{SUM}(g_D(D)) \) as \( S_E \)

GROUP BY \( A, C \)

eliminate \( E \)

\( S(A, C, E) \)
More General Example: SUM Aggregate

\[ Q = \text{SELECT} \ \text{SUM} \left( g_A(A) \times g_B(B) \times g_C(C) \times g_D(D) \times g_E(E) \right) \]
\[ \text{FROM} \ R \ \text{NATURAL JOIN} \ S \ \text{NATURAL JOIN} \ T \]

\[ Q = \text{SUM}(S_B \times S_C \times g_A(A)) \]

Join on & eliminate one variable at a time

\[ V_R = A, \ \text{SUM}(g_B(B)) \ \text{as} \ S_B \]
\[ \text{GROUP BY} \ A \]

\[ V_{ST} = A, \ \text{SUM}(S_D \times S_E \times g_C(C)) \ \text{as} \ S_C \]
\[ \text{GROUP BY} \ A \]

\[ V_T = C, \ \text{SUM}(g_E(E)) \ \text{as} \ S_D \]
\[ \text{GROUP BY} \ C \]

\[ V_S = A, C, \ \text{SUM}(g_D(D)) \ \text{as} \ S_E \]
\[ \text{GROUP BY} \ A, C \]

eliminate D

eliminate E
More General Example: SUM Aggregate

\[ Q = \text{SELECT SUM} \left( g_A(A) \cdot g_B(B) \cdot g_C(C) \cdot g_D(D) \cdot g_E(E) \right) \]
\[ \text{FROM R NATURAL JOIN S NATURAL JOIN T} \]

\[ Q = \text{SUM} (S_B \cdot S_C \cdot g_A(A)) \]

Join on & eliminate one variable at a time

- \( V_R = A, \text{SUM}(g_B(B)) \) as \( S_B \)
- \( V_{ST} = A, \text{SUM}(g_C(C) \cdot S_D \cdot S_E) \) as \( S_C \)

GROUP BY A

- eliminate B

R(A, B)

GROUP BY A

- eliminate C

C

GROUP BY C

- eliminate D

T(C, D)

GROUP BY C

- eliminate E

S(A, C, E)

GROUP BY A, C

14/46
More General Example: SUM Aggregate

\[ Q = \text{SELECT} \ \text{SUM} \left( g_A(A) \ast g_B(B) \ast g_C(C) \ast g_D(D) \ast g_E(E) \right) \]
\[ \text{FROM} \ \text{R NATURAL JOIN S NATURAL JOIN T} \]

\[ Q = \text{SUM} \left( S_B \ast S_C \ast g_A(A) \right) \]

Join on & eliminate one variable at a time

\[ V_R = A, \text{SUM} \left( g_B(B) \right) \text{ as } S_B \]
\[ \text{GROUP BY } \ A \]
\[ V_{ST} = A, \text{SUM} \left( S_D \ast S_E \ast g_C(C) \right) \text{ as } S_C \]
\[ \text{GROUP BY } \ A \]

\[ V_T = C, \text{SUM} \left( g_E(E) \right) \text{ as } S_D \]
\[ \text{GROUP BY } \ C \]
\[ V_S = A, C, \text{SUM} \left( g_D(D) \right) \text{ as } S_E \]
\[ \text{GROUP BY } \ A, C \]

eliminate A

eliminate B

eliminate C

eliminate D

eliminate E
One variable order for
\( R(A, B), S(A, C, E), T(C, D) \)
Query Evaluation Plans using **Variable Orders**

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

![Diagram of query variables]

Tree of query variables

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

depends on $A$

$B$

$R(A, B)$

depends on $C$

$D$

depends on $A$, $C$

$E$

$T(C, D)$

but not $A$!
Create a view at each var \( X \) with schema \( \text{depends}(X) \)

\[
\begin{align*}
V^{@A}[] & \quad \text{view at A} \\
V^{@B}[A] & \quad \text{view at B} \\
V^{@D}[C] & \quad \text{view at D} \\
V^{@C}[A] & \quad \text{view at C} \\
V^{@E}[A, C] & \quad \text{view at E}
\end{align*}
\]
Create a view at each var $X$ with schema \textit{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)

\[

\begin{align*}
V^{@A}[] & \quad \text{view at } A \\
V^{@B}[A] & \quad \text{view at } B \\
R(A, B) & \\
V^{@D}[C] & \quad \text{view at } D \\
T(C, D) & \\
V^{@C}[A] & \quad \text{view at } C \\
V^{@E}[A, C] & \quad \text{view at } E \\
S(A, C, E) & 
\end{align*}
\]
More General Example: SUM Aggregate

Q = SELECT SUM( g_A(A) * g_B(B) * g_C(C) * g_D(D) * g_E(E) )
FROM R NATURAL JOIN S NATURAL JOIN T

Q = SUM(S_B * S_C * g_A(A))

V^B = A, SUM(g_B(B)) as S_B
GROUP BY A

V^C = A, SUM(S_D * S_E * g_C(C)) as S_C
GROUP BY A

V^D = C, SUM(g_E(E)) as S_D
GROUP BY C

V^E = A, C, SUM(g_D(D)) as S_E
GROUP BY A, C

R(A, B)

T(C, D)

S(A, C, E)
More General Example: SUM Aggregate

\[ Q = \text{SELECT} \ \text{SUM} \left( \ g_A(A) \ast g_B(B) \ast g_C(C) \ast g_D(D) \ast g_E(E) \right) \]
\[ \text{FROM} \ R \ \text{NATURAL JOIN} \ S \ \text{NATURAL JOIN} \ T \]
More General Example: SUM Aggregate

Q = SELECT SUM ( \text{ga}(A) \ast \text{gb}(B) \ast \text{gc}(C) \ast \text{gd}(D) \ast \text{ge}(E) )
FROM R NATURAL JOIN S NATURAL JOIN T

Imagine aggregate values are of type \mathcal{R}

\text{g}_{X} : \text{Dom}(X) \rightarrow \mathcal{R}

Can we evaluate Q using the query plan from before?
More General Example: SUM Aggregate

\[
Q = \text{SELECT SUM} ( g_A(A) \times g_B(B) \times g_C(C) \times g_D(D) \times g_E(E) ) \\
\text{FROM } R \ \text{NATURAL JOIN } S \ \text{NATURAL JOIN } T
\]

Imagine aggregate values are of type \( \mathcal{R} \)

\[ g_X : \text{Dom}(X) \rightarrow \mathcal{R} \]

Can we evaluate \( Q \) using the query plan from before?

Yes(!), but we need to:

- Define \( \times \) 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 \( \times \) over \( + \)
A ring \((R, +, *, 0, 1)\) is a set \(R\) with two binary ops:

- **Additive commutativity**: \(a + b = b + a\)
- **Additive associativity**: \((a + b) + c = a + (b + c)\)
- **Additive identity**: \(0 + a = a + 0 = a\)
- **Additive inverse**: \(\exists -a \in R : a + (-a) = (-a) + a = 0\)
- **Multiplicative associativity**: \((a * b) * c = a * (b * c)\)
- **Multiplicative identity**: \(a * 1 = 1 * a = a\)
- **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)

\[
\begin{array}{ccc}
A & B & \rightarrow & R[A, B] \\
\hline
a_1 & b_1 & \rightarrow & r_1 \\
a_2 & b_1 & \rightarrow & r_2 \\
\end{array}
\]

- Finitely many tuples with non-zero payloads

- 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

\[
Q = \text{SELECT } \text{SUM}( \text{g}_A(A) \times \text{g}_B(B) \times \text{g}_C(C) \times \text{g}_D(D) \times \text{g}_E(E) ) \text{ FROM } R \text{ NATURAL JOIN } S \text{ NATURAL JOIN } T
\]

In our formalism:

\[
Q = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E \left( R[A, B] \otimes S[A, C, E] \otimes T[C, D] \right)
\]

variable marginalization 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 \((\mathcal{R}, +, *, 0, 1)\):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>R[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>b_1</td>
<td>→</td>
<td>r_1</td>
</tr>
<tr>
<td>a_2</td>
<td>b_1</td>
<td>→</td>
<td>r_2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>S[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_2</td>
<td>b_1</td>
<td>→</td>
<td>s_1</td>
</tr>
<tr>
<td>a_3</td>
<td>b_2</td>
<td>→</td>
<td>s_2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>T[B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_1</td>
<td>c_1</td>
<td>→</td>
<td>t_1</td>
</tr>
<tr>
<td>b_2</td>
<td>c_2</td>
<td>→</td>
<td>t_2</td>
</tr>
</tbody>
</table>
Query Operators

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$R[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$S[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>→</td>
<td>$s_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>$T[B, C]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$t_1$</td>
</tr>
<tr>
<td>$b_2$</td>
<td>$c_2$</td>
<td>→</td>
<td>$t_2$</td>
</tr>
</tbody>
</table>

**Union** $\uplus$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$(R \uplus S)[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_2 + s_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>→</td>
<td>$s_2$</td>
</tr>
</tbody>
</table>
Query Operators

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>R[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>S[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>T[B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>t₁</td>
</tr>
<tr>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>t₂</td>
</tr>
</tbody>
</table>

Union \(\uplus\)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>(R (\uplus) S)[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂ + s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

Marginalization \(\bigoplus\)

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>(\bigoplus A) (R (\uplus) S) ⊗ T[B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>t₁</td>
</tr>
<tr>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>t₂</td>
</tr>
</tbody>
</table>
Query Operators

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>R[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>S[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>T[B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>t₁</td>
</tr>
<tr>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>t₂</td>
</tr>
</tbody>
</table>

**Union** \(\uplus\)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>(R (\uplus) S)[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂ + s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

**Join** \(\otimes\)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>→</th>
<th>((R (\uplus) S) (\otimes) T)[A, B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>r₁ * t₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>(r₂ + s₁) * t₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>s₂ * t₂</td>
</tr>
</tbody>
</table>
Query Operators

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>R[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>S[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>T[B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>t₁</td>
</tr>
<tr>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>t₂</td>
</tr>
</tbody>
</table>

**Union** $\biguplus$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>(R $\biguplus$ S)[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>r₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>→</td>
<td>r₂ $+$ s₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>→</td>
<td>s₂</td>
</tr>
</tbody>
</table>

**Join** $\otimes$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>→</th>
<th>((R $\biguplus$ S) $\otimes$ T)[A, B, C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>r₁ $*$ t₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₁</td>
<td>c₁</td>
<td>→</td>
<td>(r₂ $+$ s₁) $*$ t₁</td>
</tr>
<tr>
<td>a₃</td>
<td>b₂</td>
<td>c₂</td>
<td>→</td>
<td>s₂ $*$ t₂</td>
</tr>
</tbody>
</table>
Query Operators

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$R[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$S[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$s_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>→</td>
<td>$s_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>$T[B, C]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$t_1$</td>
</tr>
<tr>
<td>$b_2$</td>
<td>$c_2$</td>
<td>→</td>
<td>$t_2$</td>
</tr>
</tbody>
</table>

**Union** $\uplus$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>$(R \uplus S)[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>→</td>
<td>$r_2 + s_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>→</td>
<td>$s_2$</td>
</tr>
</tbody>
</table>

**Join** $\otimes$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>→</th>
<th>$((R \uplus S) \otimes T)[A, B, C]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$r_1 * t_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$(r_2 + s_1) * t_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>→</td>
<td>$s_2 * t_2$</td>
</tr>
</tbody>
</table>

**Marginalization** $\bigoplus_A$

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>→</th>
<th>$(\bigoplus_A (R \uplus S) \otimes T)[B, C]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$r_1 * t_1 * g_A(a_1) + (r_2 + s_1) * t_1 * g_A(a_2)$</td>
</tr>
<tr>
<td>$b_2$</td>
<td>$c_2$</td>
<td>→</td>
<td>$s_2 * t_2 * g_A(a_3)$</td>
</tr>
</tbody>
</table>

for a given $g_A : \text{Dom}(A) \rightarrow \mathcal{R}$
General Query Form

\[ Q = \text{SELECT } X_1, \ldots, X_f, \text{ SUM}( g_{f+1}(X_{f+1}) \ast \cdots \ast g_m(X_m) ) \text{ FROM } R_1 \text{ NATURAL JOIN } \ldots \text{ NATURAL JOIN } R_n \text{ GROUP BY } X_1, \ldots, X_f \]

Expressed as Functional Aggregate Query:

\[ Q[X_1, \ldots, X_f] = \bigoplus X_{f+1} \ldots \bigoplus X_m \bigotimes_{i \in [n]} R_i[S_i] \]

where:

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

\[ Q(D + \delta D) = Q(D) + \delta Q(D, \delta D) \]

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>δR[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>−1</td>
</tr>
<tr>
<td>a₄</td>
<td>b₃</td>
<td>→</td>
<td>2</td>
</tr>
</tbody>
</table>

Tuples with positive/negative payloads denote insertions/deletions

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

- The query language is closed under taking deltas

\[
\delta(R \uplus S) = \delta R \uplus \delta S \\
\delta(R \otimes S) = (\delta R \otimes S) \uplus (R \otimes \delta S) \uplus (\delta R \otimes \delta S) \\
\delta(\bigoplus_A R) = \bigoplus_A \delta R
\]
Consider our running examples

Maintain the query result under updates to $T$

**View tree**

```
  V@A[]
   /  \
  V@B[A]  V@C[A]
  /  \
 /  \
V@D[C]  T[C, D]
  /  \
V@E[A, C]  S[A, C, E]
```
Consider our running examples

Maintain the query result under updates to $T$
Consider our running examples

Maintain the query result under updates to $T$

Materialized query result

View tree

$V^{@A}[ ]$

$V^{@B}[A]$

$V^{@C}[A]$

$V^{@D}[C]$

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

$R[A, B]$

$T[C, D]$

$S[A, C, E]$

$\Rightarrow$

Delta view tree

$V^{@A}[ ]$

$V^{@B}[A]$

$V^{@C}[A]$

$V^{@D}[C]$

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

$R[A, B]$

$T[C, D]$

$S[A, C, E]$
Delta Propagation

Consider our running examples

Maintain the query result under updates to $T$

**View tree**

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

**Delta view tree**

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

$$\delta V^{@A}[]$$

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

$\delta R[A, B]$

$V^{@C}[A]$

$V^{@D}[C]$

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

$T[C, D]$

$S[A, C, E]$

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

$$\delta V^{@A}[]$$

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

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

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

$\delta T[C, D]$

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

$S[A, C, E]$
Maintain the query result for updates to R and T

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

$$\left( \bigoplus_A V^{@B}[A] \otimes \delta S_A[A] \right) \otimes \left( \bigoplus_C V^{@D}[C] \otimes \delta S_C[C] \right) \otimes \bigoplus_E \delta S_E[E]$$

$$\delta S_A[A] \otimes (\bigoplus_C V^{@D}[C] \otimes \delta S_C[C]) \otimes \bigoplus_E \delta S_E[E]$$

$$\delta S_A[A] \otimes \delta S_C[C] \otimes \bigoplus_E \delta S_E[E]$$

$$\delta S_A[A] \otimes \delta S_C[C] \otimes \delta S_E[E]$$
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
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!)
Learning Linear Regression Models

- Find model parameters $\Theta$ best satisfying:

<table>
<thead>
<tr>
<th>Size (ft²)</th>
<th>#beds</th>
<th>Year</th>
<th>Region</th>
<th>Price (£)</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>4026</td>
<td>7</td>
<td>1925</td>
<td>1</td>
<td>3,450,000</td>
<td>3</td>
</tr>
<tr>
<td>1894</td>
<td>6</td>
<td>1948</td>
<td>1</td>
<td>2,750,000</td>
<td>2</td>
</tr>
<tr>
<td>5683</td>
<td>8</td>
<td>1935</td>
<td>0</td>
<td>6,000,000</td>
<td>4</td>
</tr>
<tr>
<td>4198</td>
<td>4</td>
<td>1908</td>
<td>0</td>
<td>4,600,000</td>
<td>1</td>
</tr>
<tr>
<td>2463</td>
<td>5</td>
<td>1928</td>
<td>1</td>
<td>3,250,000</td>
<td>2</td>
</tr>
</tbody>
</table>

- Iterative gradient computation:

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

- Matrices $X^T X$ and $X^T Y$ 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
Learn Learning Linear Regression Models over Joins

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

- **Naïve**: compute the join, then $O(m^2)$ sums over the join result ($m = \# \text{query variables}$)

- **Factorized**: compute one optimized join-aggregate query
  
  - Using our running query

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

  but a different payload ring and different functions $g_X$!
Linear Regression Ring

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

\[
\left( \text{COUNT}, \quad \text{vector of } \text{SUM}(X_i), \quad \text{matrix of } \text{SUM}(X_i \cdot X_j) \right)
\]

\[
a + \mathcal{R} \ b = (c_a + c_b, s_a + s_b, Q_a + Q_b)
\]

\[
a \ast \mathcal{R} \ b = (c_a c_b, c_b s_a + c_a s_b, c_b Q_a + c_a Q_b + s_a s_b^T + s_b s_a^T)
\]

\[
0 = (0, 0_{m \times 1}, 0_{m \times m})
\]

\[
1 = (1, 0_{m \times 1}, 0_{m \times m})
\]

Function $g_{X_j}$ for variable $X_j$

\[
g_{X_j}(x) = (1, s, Q) \quad \text{where}
\]

$s$ has all 0s except $s_j = x$

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

Sparse payloads

Dense payloads
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...

**IF YOU LIKED IT**

**THEN YOU SHOULDA PUT A RING ON IT**
Thank you!
The Triangle Query

\[ Q_{\Delta}[\ ] = \bigoplus_A \bigoplus_B \bigoplus_C R[A, B] \otimes S[B, C] \otimes T[C, A] \]
Relational Data Ring

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$\rightarrow$</th>
<th>$\mathcal{R}[A, B]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$\rightarrow$</td>
<td>$c$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$c_1 \rightarrow 1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$c_2 \rightarrow 1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$\rightarrow$</td>
<td>$c$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$c_3 \rightarrow 1$</td>
</tr>
</tbody>
</table>

Keep results of conjunctive queries in 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

\[ Q = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E ( R[A, B] \times S[A, C, E] \times T[C, D] ) \]

• Lift (aggregate) functions:

\[ g_X(x) = \begin{cases} 
 X & \text{if } X \text{ is a free variable} \\
 x \rightarrow 1 \\
 (\) \rightarrow 1 & \text{otherwise}
\end{cases} \]
Listing Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Listing Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Listing Representation of Conjunctive Query Results

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→ R[A,B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₂</td>
<td>b₃</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₃</td>
<td>b₄</td>
<td>() → 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>E</th>
<th>→ S[A,C,E]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>e₁</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>e₂</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₁</td>
<td>c₂</td>
<td>e₃</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₂</td>
<td>c₂</td>
<td>e₄</td>
<td>() → 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>→ T[C,D]</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>d₁</td>
<td>() → 1</td>
</tr>
<tr>
<td>c₂</td>
<td>d₂</td>
<td>() → 1</td>
</tr>
<tr>
<td>c₂</td>
<td>d₃</td>
<td>() → 1</td>
</tr>
<tr>
<td>c₃</td>
<td>d₄</td>
<td>() → 1</td>
</tr>
</tbody>
</table>
Listing Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Listing Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Listing Representation of Conjunctive Query Results

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

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

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

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

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

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

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

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

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

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

\[
\begin{array}{c}
c_1 \quad \rightarrow \quad (\rightarrow 1) \\
c_2 \quad \rightarrow \quad (\rightarrow 1) \\
c_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

A \rightarrow R[A, B]

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

A \rightarrow S[A, C, E]

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
a_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

A \rightarrow T[C, D]

\[
\begin{array}{c}
c_1 \quad \rightarrow \quad (\rightarrow 1) \\
c_2 \quad \rightarrow \quad (\rightarrow 1) \\
c_3 \quad \rightarrow \quad (\rightarrow 1)
\end{array}
\]

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

\[
\begin{array}{c}
a_1 \quad \rightarrow \quad (\rightarrow 1) \\
a_2 \quad \rightarrow \quad (\rightarrow 1) \\
\end{array}
\]
**Listing Representation of Conjunctive Query Results**

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

<table>
<thead>
<tr>
<th>A B → R[A,B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁ b₁ → () → 1</td>
</tr>
<tr>
<td>a₁ b₂ → () → 1</td>
</tr>
<tr>
<td>a₂ b₃ → () → 1</td>
</tr>
<tr>
<td>a₃ b₄ → () → 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A C E → S[A,C,E]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁ c₁ e₁ → () → 1</td>
</tr>
<tr>
<td>a₁ c₁ e₂ → () → 1</td>
</tr>
<tr>
<td>a₁ c₂ e₃ → () → 1</td>
</tr>
<tr>
<td>a₂ c₂ e₄ → () → 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C D → T[C,D]</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁ d₁ → () → 1</td>
</tr>
<tr>
<td>c₂ d₂ → () → 1</td>
</tr>
<tr>
<td>c₂ d₃ → () → 1</td>
</tr>
<tr>
<td>c₃ d₄ → () → 1</td>
</tr>
</tbody>
</table>

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

\[ \begin{array}{cccc}
A & B & C & D \\
\hline
a₁ b₁ c₁ d₁ & → & 2 \\
a₁ b₁ c₂ d₂ & → & 1 \\
a₁ b₁ c₂ d₃ & → & 1 \\
a₁ b₂ c₁ d₁ & → & 2 \\
a₁ b₂ c₂ d₂ & → & 1 \\
a₁ b₂ c₂ d₃ & → & 1 \\
a₂ b₃ c₂ d₂ & → & 1 \\
a₂ b₃ c₂ d₃ & → & 1 \\
\end{array} \]

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

\[ \begin{array}{c}
A \rightarrow V^{@B}[A] \\
V^{@B}[A] \\
V^{@C}[A] \\
V^{@A}[ ] \\
V^{@D}[C] \\
V^{@E}[A, C] \\
T[C, D] \\
S[A, C, E] \\
\end{array} \]

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

\[ \begin{array}{c}
A \rightarrow V^{@E}[A,C] \\
A \rightarrow V^{@E}[A,C] \\
\end{array} \]

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

\[ \begin{array}{cccc}
A & B & C & D \\
\hline
a₁ b₁ c₁ d₁ & → & 2 \\
a₁ b₁ c₂ d₂ & → & 1 \\
a₁ b₁ c₂ d₃ & → & 1 \\
a₁ b₂ c₁ d₁ & → & 2 \\
a₁ b₂ c₂ d₂ & → & 1 \\
a₁ b₂ c₂ d₃ & → & 1 \\
a₂ b₃ c₂ d₂ & → & 1 \\
a₂ b₃ c₂ d₃ & → & 1 \\
\end{array} \]
Factorized Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Factorized Representation of Conjunctive Query Results

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

\[ \begin{align*}
A & \rightarrow R[AB] \\
A & \rightarrow V^{\theta B}[A] \\
A & \rightarrow V^{\theta C}[A] \\
A & \rightarrow V^{\theta E}[A, C] \\
C & \rightarrow V^{\theta D}[C] \\
C & \rightarrow V^{\theta E}[A, C] \\
\end{align*} \]
Factorized Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
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 \( (a, b, c, d) \)
Factorized Representation of Conjunctive Query Results

\[ Q(A, B, C, D) = R(A, B), S(A, C, E), T(C, D) \]
Performance: Maintenance of Conjunctive Query Results

Star schema

Snowflake schema
Matrix Chain Multiplication

Input: Matrices $A_i$ of size of $p_i \times p_{i+1}$ over some ring $\mathcal{R}$ ($i \in [n]$)

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

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

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

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

Matrix changes

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

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

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

- **Evaluation or IVM**: \( O(p^3) \)
- **IVM with factorized updates**: \( O(p^2) \)
Update to $A_2$ expressed as outer product

Update to $A_2$ expressed as sum of $r$ outer products