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
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
     \[ \Rightarrow \text{Inefficient:} \text{ high redundancy in computation and representation of join results} \]

2. Low-latency processing
   - Naïve solution: re-compute query results as data changes
     \[ \Rightarrow \text{Inefficient:} \text{ high-latency processing} \]
   - Common practice: IVM (Incremental View Maintenance)
     For query \( Q \), database \( D \), and change \( \Delta D \), compute (the hopefully cheaper) delta \( \Delta Q \):
     \[
     Q(D + \Delta D) = Q(D) + \Delta Q(D, \Delta D)
     \]

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

Introduction

Factorized Ring Computation

Incremental View Maintenance

Applications

  Learning Linear Regression Models

  Factorized Representation of Conjunctive Query Results

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

<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>

$r_1$ 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
• A ring \((D, +, *, 0, 1)\) is a set \(D\) 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 \(\exists -a \in D : a + (-a) = (-a) + a = 0\)

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

<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>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>

Operations:

**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>$r_1 \ast t_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$c_1$</td>
<td>→</td>
<td>$(r_2 + s_1) \ast t_1$</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_2$</td>
<td>$c_2$</td>
<td>→</td>
<td>$s_2 \ast t_2$</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>
</table>
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 \( \mathbb{Z} \)
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 \( \mathbb{Z} \)

**Naïve:** compute the join and then COUNT

\[
Q = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (R \otimes S \otimes T)
\]

Takes \( O(N^3) \) 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 \( \mathbb{Z} \)

**Naïve:** compute the join and then COUNT

\[ Q = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (R \otimes S \otimes T) \]

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:

\[
\begin{align*}
Q & = A \bigoplus B\left[ A, B \right] \text{(marginalize B)} \\
V_3\left[ C \right] & = D \bigoplus T\left[ C, D \right] \text{(marginalize D)} \\
Q & = A \bigoplus \left( V_1\left[ A \right] \otimes V_4\left[ A \right] \right) \text{(marginalize A)} \\
V_2\left[ A, C \right] & = E \bigoplus S\left[ A, C, E \right] \text{(marginalize E)} \\
V_4\left[ A \right] & = C \bigoplus \left( V_2\left[ A, C \right] \otimes V_3\left[ C \right] \right) \text{(marginalize C)}
\end{align*}
\]
Example: Factorized Aggregate Computation

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

\[
V_1[A] = \bigoplus_B R[A, B] \quad \text{(marginalize B)}
\]
Example: Factorized Aggregate Computation

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

\[
V_1[A] = \bigoplus_B R[A, B] \quad \text{(marginalize B)} \quad V_2[A, C] = \bigoplus_E S[A, C, E] \quad \text{(marginalize E)}
\]
Example: Factorized Aggregate Computation

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

\[
V_1[A] = \bigoplus_B R[A, B] \quad \text{(marginalize B)} \quad V_2[A, C] = \bigoplus_E S[A, C, E] \quad \text{(marginalize E)}
\]
\[
V_3[C] = \bigoplus_D T[C, D] \quad \text{(marginalize D)}
\]
Example: Factorized Aggregate Computation

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

\[ V_1[A] = \bigoplus_B R[A, B] \quad \text{(marginalize } B) \]
\[ V_2[A, C] = \bigoplus_E S[A, C, E] \quad \text{(marginalize } E) \]
\[ V_3[C] = \bigoplus_D T[C, D] \quad \text{(marginalize } D) \]
\[ V_4[A] = \bigoplus_C (V_2[A, C] \otimes V_3[C]) \quad \text{(marginalize } C) \]
\[ \text{also re-use counts of } E \text{ and } D! \]
Example: Factorized Aggregate Computation

- Push COUNT past joins to eliminate variables
- Factorized computation \(\text{à la InsideOut/FDB:}\)

\[
\begin{align*}
V_1[A] &= \bigoplus_B R[A, B] \quad \text{(marginalize B)} \\
V_2[A, C] &= \bigoplus_E S[A, C, E] \quad \text{(marginalize E)} \\
V_3[C] &= \bigoplus_D T[C, D] \quad \text{(marginalize D)} \\
V_4[A] &= \bigoplus_C (V_2[A, C] \otimes V_3[C]) \quad \text{(marginalize C)} \\
Q &= \bigoplus_A (V_1[A] \otimes V_4[A]) \quad \text{(marginalize } A) \\
V_4[A] &= \bigoplus_{CD} (V_2[A, C] \otimes V_3[c]) \quad \text{(marginalize } C, D) \quad \text{also re-use counts of } E \text{ and } D!
\end{align*}
\]
Different Modeling of Relations

- Compute $\text{SUM}(C \cdot D)$ 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 $\mathbb{R}$:
  - $R[a, b] = 1$ iff $(a, b) \in R$, 0 otherwise
  - $S[a, c, e] = c$ iff $(a, c, e) \in S$, 0 otherwise
  - $T[c, d] = d$ iff $(c, d) \in T$, 0 otherwise
Different Modeling of Relations

- Compute $\text{SUM}(C \cdot D)$ 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 $\mathbb{R}$:
  - $R[a, b] = 1$ iff $(a, b) \in R$, 0 otherwise
  - $S[a, c, e] = c$ iff $(a, c, e) \in S$, 0 otherwise
  - $T[c, d] = d$ iff $(c, d) \in T$, 0 otherwise
- The factor $Q$ expressing the sum is:
  \[
  Q = \bigoplus_A \bigoplus_B \bigoplus_C \bigoplus_D \bigoplus_E (R[A, B] \otimes S[A, C, E] \otimes T[C, D])
  \]
  Factor payloads carry out the summation!
- Same as the COUNT query but with diff modeling and ring!
Eager modeling (as in previous examples)

- Assign payload $R[t]$ to each tuple $t$ of relation $R$
- Computed payloads might be discarded later on
  - $T[c, d] = c \cdot d$ 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 $R[t]$ to each tuple $t$ of relation $R$
- Computed payloads might be discarded later on
  - $T[c, d] = c \cdot d$ 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 $(\mathbb{D}, +, \times, 0, 1)$
  - Each tuple has the payload of $1 \in \mathbb{D}$

- Lift factors $\Lambda_A, \Lambda_B, \Lambda_C, \Lambda_D, \Lambda_E$ map the domain of a variable to $\mathbb{D}$
  - All lift factors map to $1 \in \mathbb{Z}$
  - $\text{COUNT} \quad \Lambda_C[c] = c$ 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 $(\mathbb{D}, +, \times, 0, 1)$
  - Each tuple has the payload of $1 \in \mathbb{D}$

- Lift factors $\Lambda_A, \Lambda_B, \Lambda_C, \Lambda_D, \Lambda_E$ map the domain of a variable to $\mathbb{D}$
  - $\text{COUNT}$ all lift factors map to $1 \in \mathbb{Z}$
  - $\text{SUM}(C*D)$ $\Lambda_C[c] = c$ and $\Lambda_D[d] = d$; others map to $1 \in \mathbb{R}$

- Lift values of a variable just before its marginalization

$$V_1[A] = \bigoplus_B (R[A, B] \otimes \Lambda_B[B])$$
$$V_3[C] = \bigoplus_D (T[C, D] \otimes \Lambda_D[D])$$
$$Q = \bigoplus_A (V_1[A] \otimes V_4[A] \otimes \Lambda_A[A])$$

$$V_2[A, C] = \bigoplus_E (S[A, C, E] \otimes \Lambda_E[E])$$
$$V_4[A] = \bigoplus_C (V_2[A, C] \otimes V_3[C] \otimes \Lambda_C[C])$$
Variable Orders

**Variable order for a join query** $Q$

- Rooted tree with one node per variable in $Q$
- Function $\text{dep}$ maps each variable to a subset of its ancestors

**Properties:**

- The variables of a factor $R$ lie along the same root-to-leaf path
  - $Y \in \text{dep}(X)$ if $X$ and $Y$ are variables of $R$ and $Y$ is ancestor of $X$
- For every child $B$ of $A$, $\text{dep}(B) \subseteq \text{dep}(A) \cup \{A\}$

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

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

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

\[
\begin{align*}
\text{dep}(A) &= \emptyset & A \\
\text{dep}(B) &= \{A\} & / \quad \backslash \\
\text{dep}(C) &= \{A\} & B \quad C \\
\text{dep}(D) &= \{C\} & / \quad \backslash \\
\text{dep}(E) &= \{A, C\} & D \quad E
\end{align*}
\]

\[
\Rightarrow
\]

View tree

\[
\begin{align*}
V^{@A}[ ] &= \bigoplus_A (V^{@B}[A] \otimes V^{@C}[A] \otimes \land_A[A]) \\
V^{@B}[A] \quad \bigtriangleup \quad V^{@C}[A] \quad \bigtriangleup \quad V^{@D}[C] \quad \bigtriangleup \quad V^{@E}[A, C] \\
\quad \bigtriangleup \quad R[A, B] \quad \bigtriangleup \quad \bigtriangleup \quad T[C, D] \quad \bigtriangleup \quad \bigtriangleup \quad S[A, C, E]
\end{align*}
\]

- Views can be materialized if needed
We support a subset of FAQs:

\[ Q[X_1, \ldots, X_f] = \bigoplus X_{f+1} \cdots \bigoplus X_m \otimes_{i \in [n]} R_i[S_i] \otimes_{j \in [f+1, m]} \Lambda X_j[X_j] \]

where:

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

Introduction

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 with changes in the underlying data

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

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 with Factors

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>→</th>
<th>( \delta R[A, B] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>( b_1 )</td>
<td>→</td>
<td>( -1 )</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>( b_3 )</td>
<td>→</td>
<td>( 2 )</td>
</tr>
</tbody>
</table>

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**

\[
\begin{align*}
V^@A[ & ] \\
V^@B[A] & V^@C[A] \\
R[A, B] & \\
V^@D[C] & V^@E[A, C] & \\
\end{align*}
\]
Consider our running example

Maintain the query result for updates to $T$
Consider our running example

Maintain the query result for updates to $T$

**View tree**

Materialized query result

$V^@A[ ]$

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

$V^@C[A]$  

$V^@E[A, C]$  

$T[C, D]$  

**Delta view tree**

$V^@A[ ]$

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

Consider our running example

Maintain the query result for updates to $T$

**View tree**

Materialized query result

$V^{@A}[ ]$

$V^{@B}[A]$

$V^{@C}[A]$

$V^{@D}[C]$

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

$R[A, B]$

$T[C, D]$

$S[A, C, E]$

**Delta view tree**

$\delta T[C, D]$

$\delta T[C, D]$

$S[A, C, E]$
Consider our running example

Maintain the query result for updates to $T$

**View tree**

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

**Delta view tree**

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

Materialized query result

$V^{\oplus_A}[ ]$

$V^{\oplus_B}[A]$

$V^{\oplus_C}[A]$

$V^{\oplus_D}[C]$

$V^{\oplus_E}[A, C]$

$R[A, B]$

$T[C, D]$

$S[A, C, E]$

Delta view tree

$\delta V^{\oplus_A}[ ]$

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

$\delta V^{\oplus_D}[C]$

$\delta T[C, D]$

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

$V^{\oplus_E}[A, C]$

$S[A, C, E]$

Precomp. & materialized
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

\[
\begin{align*}
\delta & V^{@A}[ ] \\
\delta & V^{@B}[A] \\
\delta & R[A, B] \\
V^{@C}[A] \\
V^{@D}[C] & \quad V^{@E}[A, C] \\
T[C, D] & \quad S[A, C, E] \\
\end{align*}
\]

Delta view tree (for \( R \))

\[
\begin{align*}
\delta & V^{@A}[ ] \\
\delta & V^{@B}[A] \\
\delta & V^{@C}[A] \\
V^{@D}[C] & \quad V^{@E}[A, C] \\
R[A, B] & \quad S[A, C, E] \\
\delta & T[C, D] \\
\end{align*}
\]

Delta view tree (for \( T \))
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 $R$)

```
Update $V^{@B}$ and $V^{@A}$

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

$V^{@C}[A]$

$V^{@D}[C]$
```

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

```
Update $V^{@C}$ and $V^{@A}$

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

```

Update $V^{@B}$ and $V^{@A}$
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.
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.

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

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

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

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

\begin{align*}
\left( \bigoplus_A V_{R}^{@B}[A] \otimes \delta S_A[A] \right) & \otimes \\
\left( \bigoplus_C V_{T}^{@D}[C] \otimes \delta S_C[C] \right) & \otimes \\
\bigoplus_E \delta S_E[E] & \\
\delta S_A[A] & \otimes \left( \bigoplus_C V_{T}^{@D}[C] \otimes \delta S_C[C] \right) \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]
\end{align*}
Talk Outline

Introduction

Factorized Ring Computation

Incremental View Maintenance

Applications

Learning Linear Regression Models

Factorized Representation of Conjunctive Query Results

Matrix Chain Multiplication
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 $\Theta$ best satisfying:

<table>
<thead>
<tr>
<th>Size (ft$^2$)</th>
<th>#beds</th>
<th>Year</th>
<th>Region 1</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 $SUM(X_i \cdot X_j)$ for each pair $(X_i, X_j)$ of variables
  - We assume in this talk that all variables are continuous
Compute $X^TX$ when $X$ is the join of input relations

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

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

but a different payload ring and different lift factors!
Set of triples $D = (\mathbb{Z}, \mathbb{R}^m, \mathbb{R}^{m \times m})$

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

\[
a +^D b = (c_a + c_b, sa + sb, Q_a + Q_b)
\]

\[
a \ast^D b = (c_ac_b, cbsa + ca sb, cb Q_a + ca Q_b + sa sb^T + sb sa^T)
\]

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

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

Lift factor for variable $X_j$

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

\[
s \text{ has all 0s except } s_j = x
\]

\[
Q \text{ has all 0s except } Q_{j,j} = x^2
\]
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 $D$ with $\cup$ and $\otimes$ forms a ring of factors
  - Factor $0$ maps every tuple to $0 \in D$
  - Factor $1$ maps the empty tuple to $1 \in D$, others to $0 \in D$
- Payload: Factors over $D = \mathbb{Z}$ with the same schema!

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>R[A, B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
<td>$\begin{array}{l} c \ c_1 \rightarrow 1 \ c_2 \rightarrow 1 \end{array}$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_1$</td>
<td>$\begin{array}{l} c \ c_3 \rightarrow 1 \end{array}$</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 factors with relational payloads

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

\[
\Lambda_A[A] \otimes \Lambda_B[B] \otimes \Lambda_C[C] \otimes \Lambda_D[D] \otimes \Lambda_E[E]
\]

Lift factors:

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

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

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

\[ \begin{align*}
A & \rightarrow R[A,B] \\
\begin{array}{c} \rightarrow \end{array} & (\rightarrow 1) \\
\begin{array}{c} \rightarrow \end{array} & (\rightarrow 1) \\
C & \rightarrow V^{\oplus B}[A] \\
\begin{array}{c} \rightarrow \end{array} & (\rightarrow 1) \\
V^{\oplus A}[ ] & \\
V^{\oplus B}[A] & V^{\oplus C}[A] \\
R[A, B] & \\
V^{\oplus D}[C] & V^{\oplus E}[A, C] \\
\end{align*} \]

\[ \begin{align*}
A & \rightarrow V^{\oplus B}[A] \\
\begin{array}{c} \rightarrow \end{array} & (\rightarrow 1) \\
C & \rightarrow V^{\oplus D}[C] \\
\begin{array}{c} \rightarrow \end{array} & (\rightarrow 1) \\
D & \rightarrow T[C, D] \\
C & \rightarrow S[A, C, E] \\
\end{align*} \]
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]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
</tr>
<tr>
<td>a₂</td>
<td>b₃</td>
</tr>
<tr>
<td>a₃</td>
<td>b₄</td>
</tr>
</tbody>
</table>

A C E → S[A,C,E]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>e₁</td>
</tr>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>e₂</td>
</tr>
<tr>
<td>a₁</td>
<td>c₂</td>
<td>e₃</td>
</tr>
<tr>
<td>a₂</td>
<td>c₂</td>
<td>e₄</td>
</tr>
</tbody>
</table>

A → V@B[A]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₃</td>
<td>b₄</td>
<td>() → 1</td>
</tr>
</tbody>
</table>

A C → V@E[A,C]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₁</td>
<td>c₂</td>
<td>() → 1</td>
</tr>
</tbody>
</table>

C D → T[C,D]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>

C → V@D[C]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>c₃</td>
<td>d₄</td>
<td>() → 1</td>
</tr>
</tbody>
</table>

V@A[ ]

V@B[A]

V@C[A]

R[A, B]

V@D[C]

V@E[A, C]

S[A, C, E]

A C → V@E[A,C]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>c₁</td>
<td>() → 2</td>
</tr>
<tr>
<td>a₁</td>
<td>c₂</td>
<td>() → 1</td>
</tr>
<tr>
<td>a₂</td>
<td>c₂</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**

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>a₁</td>
<td>b₂</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₃</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b₄</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
</tbody>
</table>

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c₁</td>
<td>d₁</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>c₂</td>
<td>d₂</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>c₂</td>
<td>d₃</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
<tr>
<td>c₃</td>
<td>d₄</td>
<td>→</td>
<td>( ) → 1</td>
<td></td>
</tr>
</tbody>
</table>

\[ R[A, B] \]

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

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

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

\[ \text{B} \rightarrow b₁ → 1 \]

\[ b₂ → 1 \]

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

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

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

\[ \text{D} \rightarrow d₁ → 1 \]

\[ d₂ → 1 \]

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

\[ \text{T}[C, D] \]

\[ S[A, C, E] \]

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

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

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>a₁ b₁ c₁ d₁ → 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁ b₁ c₂ d₂ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁ b₁ c₂ d₃ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁ b₂ c₁ d₁ → 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁ b₂ c₂ d₂ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₁ b₂ c₂ d₃ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₂ b₃ c₂ d₂ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a₂ b₃ c₂ d₃ → 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Factorized Representation of Conjunctive Query Results

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

\[ \begin{align*}
A & \rightarrow R[A, B] \\
A & \rightarrow V^{\oplus B}[A] \\
B & \rightarrow V^{\oplus B}[A] \\
C & \rightarrow V^{\oplus D}[C] \\
D & \rightarrow T[C, D] \\
C & \rightarrow V^{\oplus E}[A, C] \\
A & \rightarrow V^{\oplus E}[A, C]
\end{align*} \]
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

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

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

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

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

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

Matrix changes

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

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

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

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

IF YOU LIKED IT
THEN YOU SHOULD PUT A RING ON IT
Thank you!
\[ Q_\Delta[ ] = \bigoplus_A \bigoplus_B \bigoplus_C R[A, B] \otimes S[B, C] \otimes T[C, A] \]