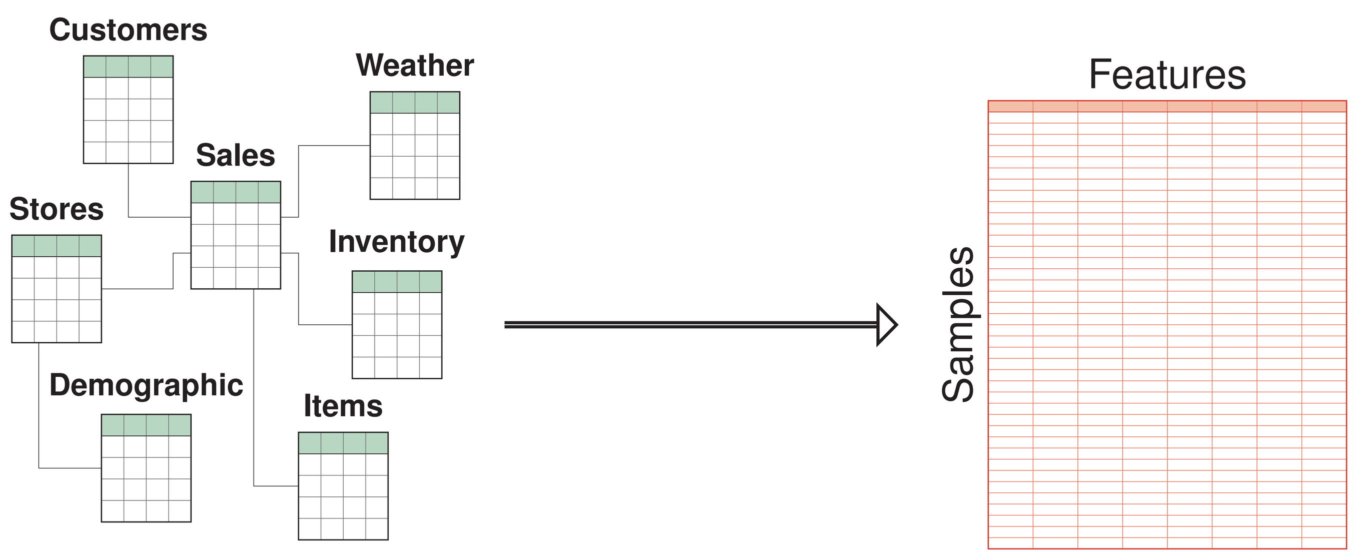
erc | UNIVERSITY OF OXFORD | UNIVERSITY OF MICHIGAN | relationalAI — AI for the enterprise

A Layered Aggregate Engine for Analytics Workloads

M. Schleich, D. Olteanu, M. Abo Khamis, H. Q. Ngo, X. Nguyen

relational.ai   fdbresearch.github.io

Current State of Affairs

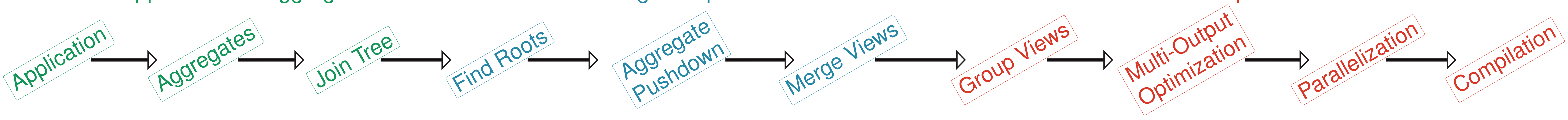


- Carefully crafted by domain experts
- Comes with relational structure
- Throws away relational structure
- Can be order-of-magnitude larger

Turn Analytics Problem into Database Problem!

1. Exploit structure in the data
   - Algebraic structure: Factorized aggregate computation
   - Combinatorial structure: Query complexity measures
2. Sharing computation and data access
   - Aggregates decomposed into directional views over join tree
   - Share data access across views
3. Specialization for workload and data
   - Generate code specific to the query batch and dataset
   - Improve cache locality for hot data
4. Parallelization
   - Task and domain parallelism

The Layers of LMFAO: Layered Multiple Functional Aggregate Optimization

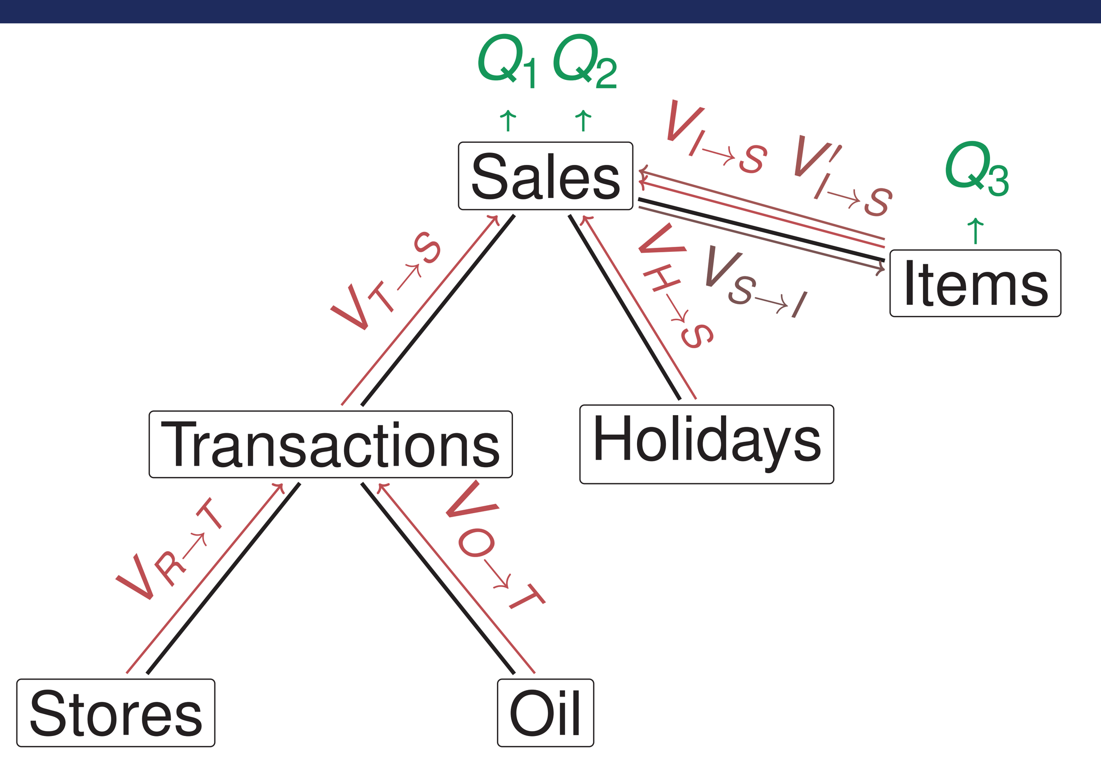


Aggregates required per Analytics Workload

Workload	Query Batch	# Queries
Linear Regression Covariance Matrix	SUM(X_i*X_j) SUM(X_i) GROUP BY X_j COUNT(*) GROUP BY X_i, X_j	814
Decision Tree (Regression, 1 Node)	VARIANCE(Y) WHERE $X_j = c_j$	3,141
Mutual Information Chow-Liu Trees	COUNT(*) GROUP BY X_i COUNT(*) GROUP BY X_i, X_j	56
Data Cubes	SUM(M) GROUP BY $X_1, \dots, X_d$	40

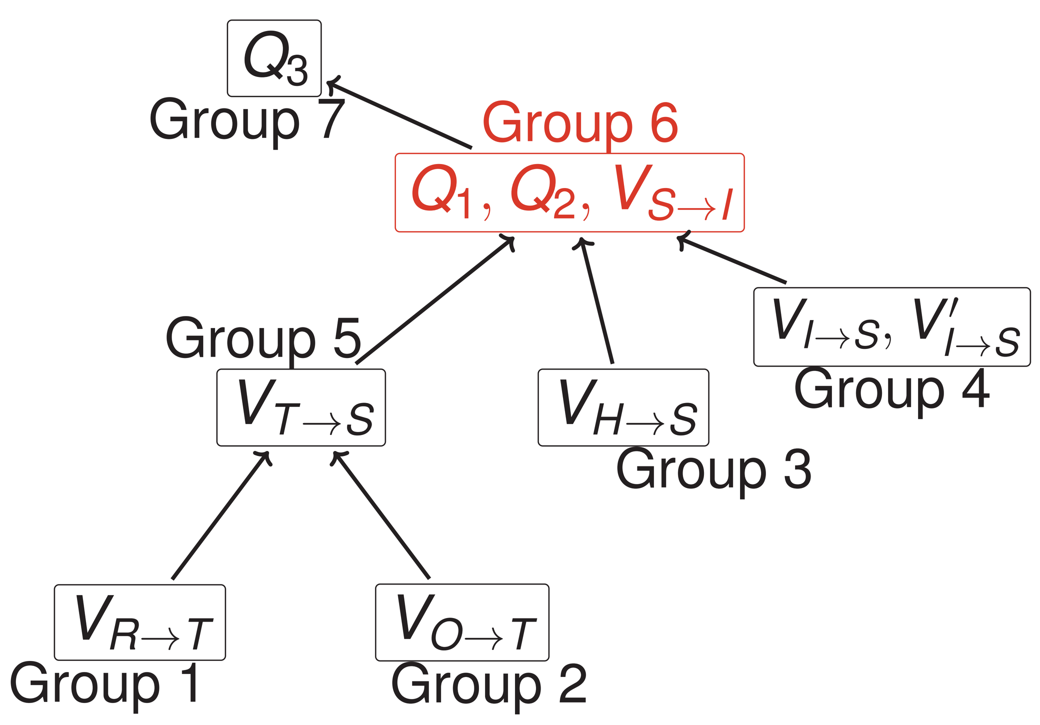
(# Queries shown for Retailer dataset)

Logical Optimization



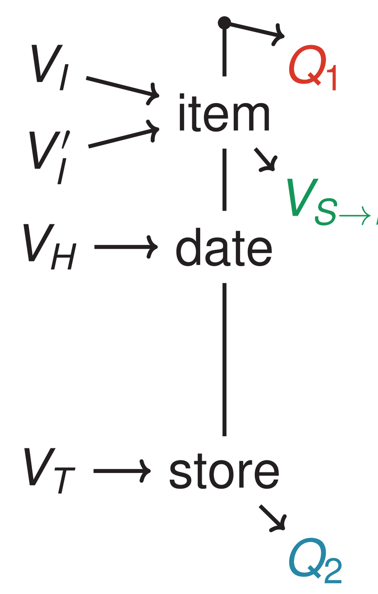
1. Find Roots: For each query, decide its output (root) node
2. Break down each query into directional views over the join tree
3. Reuse partial-aggregates and Merge Views with same group-by attributes

Dependency Graph & View Groups



- Create Dependency Graph and Group Views computed over same relation
- View Group is computational unit, computed in one pass over relation
- Task and Domain Parallelism

Multi-Output Optimization & Code Compilation



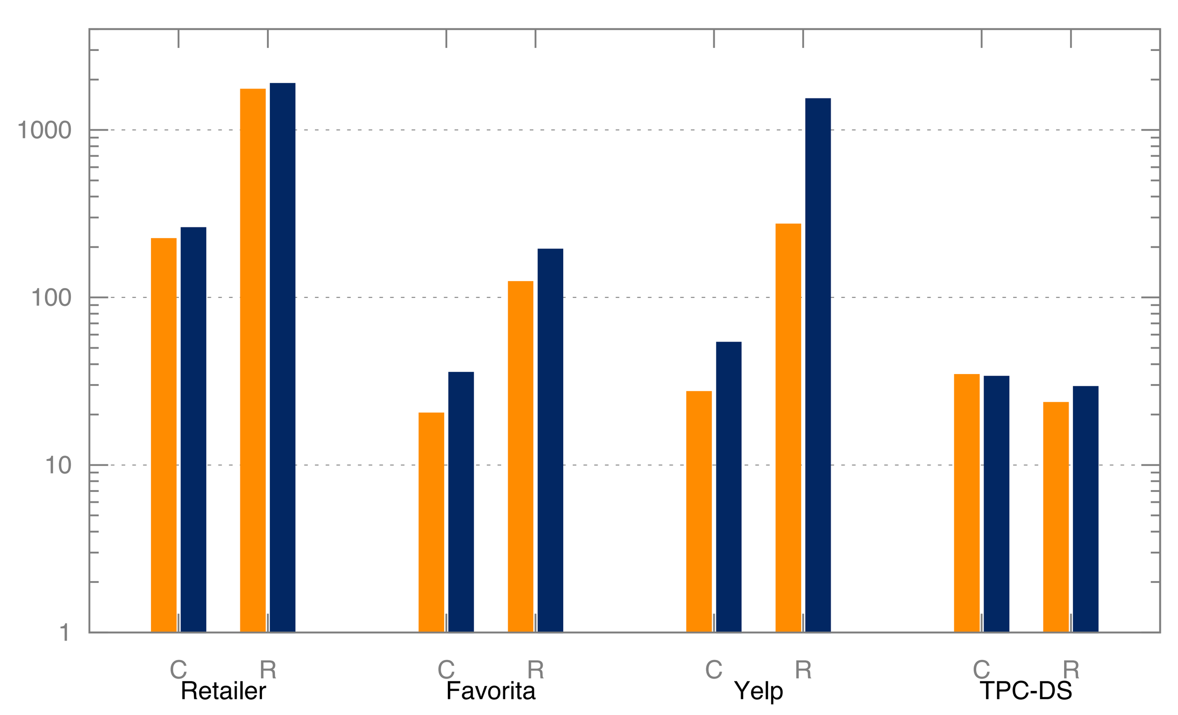
$$
\begin{array}{l}
\alpha_0 = 0;\\
\text{foreach } i \in \pi_{\text{item}}(S \bowtie_{\text{item}} V_I \bowtie_{\text{item}} V_I')\\
\quad \alpha_1 = V_I(i); \quad \alpha_2 = g(i); \quad \alpha_3 = 0;\\
\quad \text{foreach } d \in \pi_{\text{date}}(\sigma_{\text{item}=i} S \bowtie_{\text{date}} V_H \bowtie_{\text{date}} V_T)\\
\quad\quad \alpha_4 = V_H(d); \quad \alpha_5 = 0;\\
\quad\quad \text{foreach } c \in \pi_{\text{color}} \sigma_{\text{item}=i} V_I' \;:\; \alpha_5 \mathrel{+}= h(d,c) \cdot V_I'(i,c);\\
\quad\quad \alpha_6 = 0; \quad \alpha_7 = \alpha_2 \cdot \alpha_5 \cdot \alpha_4;\\
\quad\quad \text{foreach } s \in \pi_{\text{store}}(\sigma_{\text{item}=i,\text{date}=d} S \bowtie_{\text{store}} \sigma_{\text{date}=d} V_T)\\
\quad\quad\quad \alpha_8 = V_T(d,s); \quad \alpha_9 = 0; \quad \alpha_{10} = |\sigma_{\text{item}=i,\text{date}=d,\text{store}=s} S|;\\
\quad\quad\quad \text{foreach } u \in \pi_{\text{units}} \sigma_{\text{item}=i,\text{date}=d,\text{store}=s} S : \alpha_9 \mathrel{+}= f(u);\\
\quad\quad\quad \alpha_6 \mathrel{+}= \alpha_8 \cdot \alpha_9; \quad \alpha_{11} = \alpha_7 \cdot \alpha_8 \cdot \alpha_{10};\\
\quad\quad\quad \text{if } Q_2(s) \text{ then } Q_2(s) \mathrel{+}= \alpha_{11} \text{ else } Q_2(s) = \alpha_{11};\\
\quad\quad \alpha_3 \mathrel{+}= \alpha_4 \cdot \alpha_6;\\
\quad \alpha_0 \mathrel{+}= \alpha_1 \cdot \alpha_3 \quad V_{S\to I}(i) = \alpha_3 \cdot \alpha_2;\\
Q_1 = \alpha_0;
\end{array}
$$

Q_1: SUM (f(units)) $V_{S\to I}$: SUM (f(units) $\cdot$ g(item)) GROUP BY item

Q_2: SUM (g(item) $\cdot$ h(date, color)) GROUP BY store

Aggregate Experiments

Relative Speedup over DBX and MonetDB

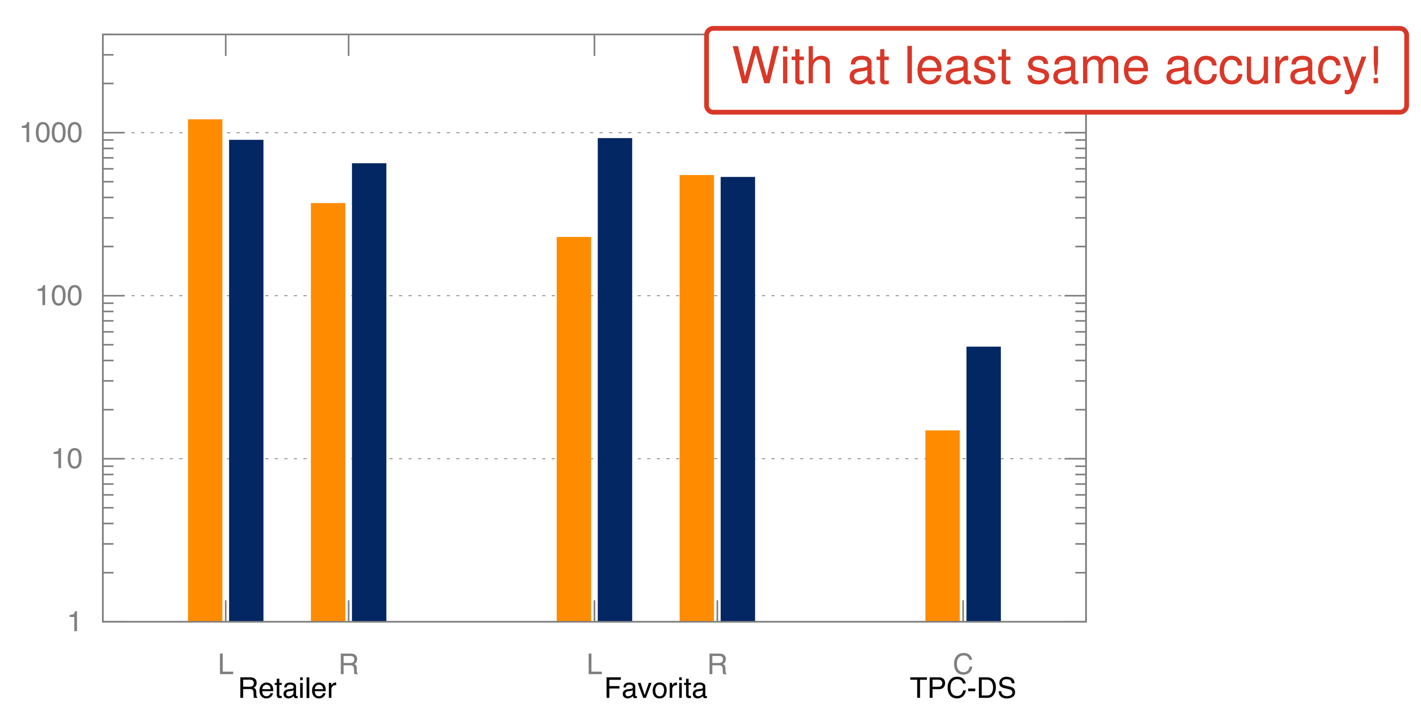


C = Covariance Matrix; R = Regression Tree Node; AWS d2.xlarge (4 vCPUs, 32GB)

Machine Learning Experiments

Relative Speedup over TensorFlow and MADlib

With at least same accuracy!



L = Linear Regression; R = Regression Tree; C = Classification Tree; Intel i7-4770 (8 CPUs, 32GB)