DB4ML
Can we build more accessible ML Systems?

Maximilian Schleich

https://mjschleich.github.io

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

Machine Learning: A Success Story

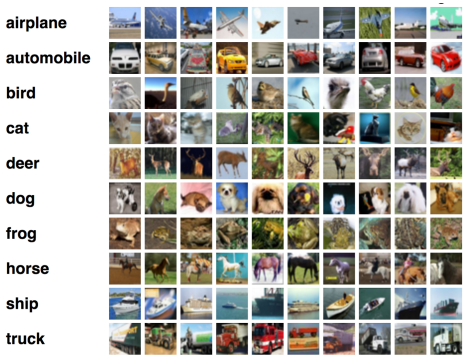
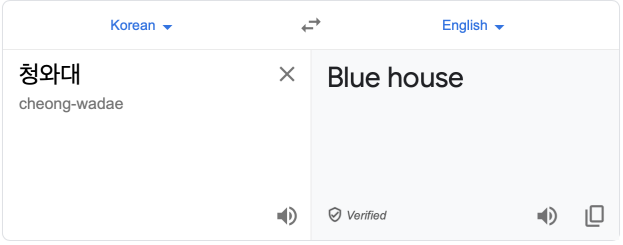
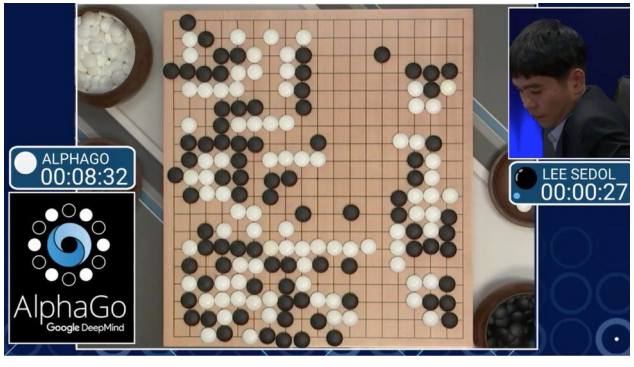


Image Classification

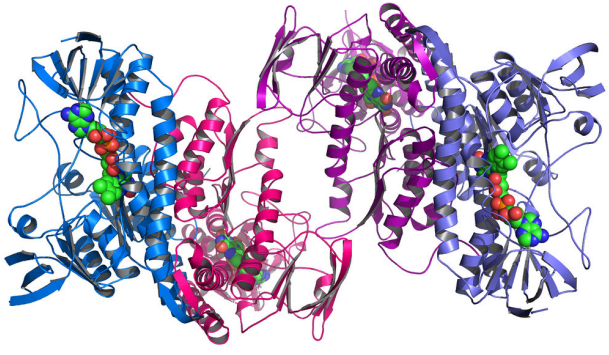
Autonomous Driving



Language Translation



Playing Go



Predicting Protein Structures

Applying Machine Learning ...

... is complicated:

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Does the model behave as expected?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Does the model behave as expected?

Why do I get this outcome?

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Does the model behave as expected?

Why do I get this outcome?

$\Rightarrow$ Only an ML expert can answer these questions

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Does the model behave as expected?

Why do I get this outcome?

$\Rightarrow$ Only an ML expert can answer these questions

... requires massive computing resources:

Applying Machine Learning ...

... is complicated:

Which model do I learn?

Which features do I need?

Do I need GPUs?

Do I use a dense or sparse tensor?

Does the model behave as expected?

Why do I get this outcome?

$\Rightarrow$ Only an ML expert can answer these questions

... requires massive computing resources:

GPT-3: Best-in-class natural language model

Size:	175 billion parameters
Training Time:	355 years on NVIDIA Tesla V100
Est. Training Cost:	$> \$4.6M^*$

*https://lambdalabs.com/blog/demystifying-gpt-3/

Current State Of Affairs

Machine Learning is a success story for the few,

... not the masses.

In Comparison, Data Management ...

... has comparable problems:

How to represent the data?

Which join algorithm to use?

How to encode domain knowledge?

In Comparison, Data Management ...

... has comparable problems:

How to represent the data?

Which join algorithm to use?

How to encode domain knowledge?

... is user-friendly and scalable

In Comparison, Data Management ...

... has comparable problems:

How to represent the data?

Which join algorithm to use?

How to encode domain knowledge?

... is user-friendly and scalable

Logical Query Specification

```
SELECT A, B
FROM R, S
WHERE R.A = S.A
```

In Comparison, Data Management ...

... has comparable problems:

How to represent the data?

Which join algorithm to use?

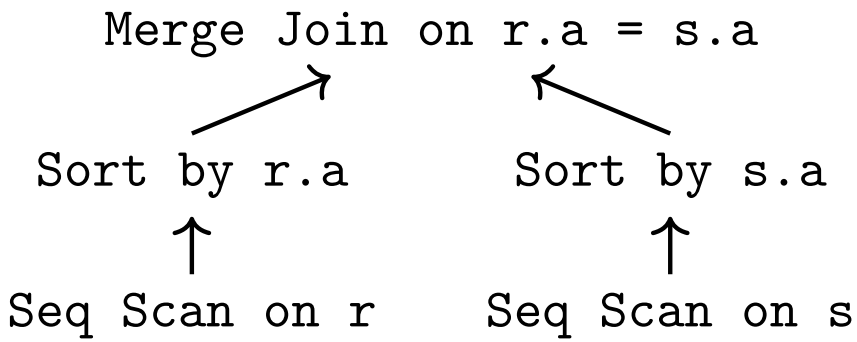
How to encode domain knowledge?

... is user-friendly and scalable

Logical Query Specification

```
SELECT A, B
FROM R, S
WHERE R.A = S.A
```

Physical Evaluation Plan



In Comparison, Data Management ...

... has comparable problems:

How to represent the data?

Which join algorithm to use?

How to encode domain knowledge?

... is user-friendly and scalable

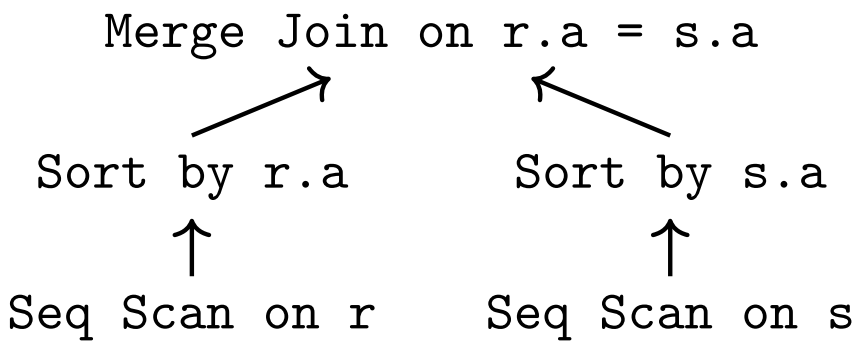
Logical Query Specification

```
SELECT A, B
FROM R, S
WHERE R.A = S.A
```

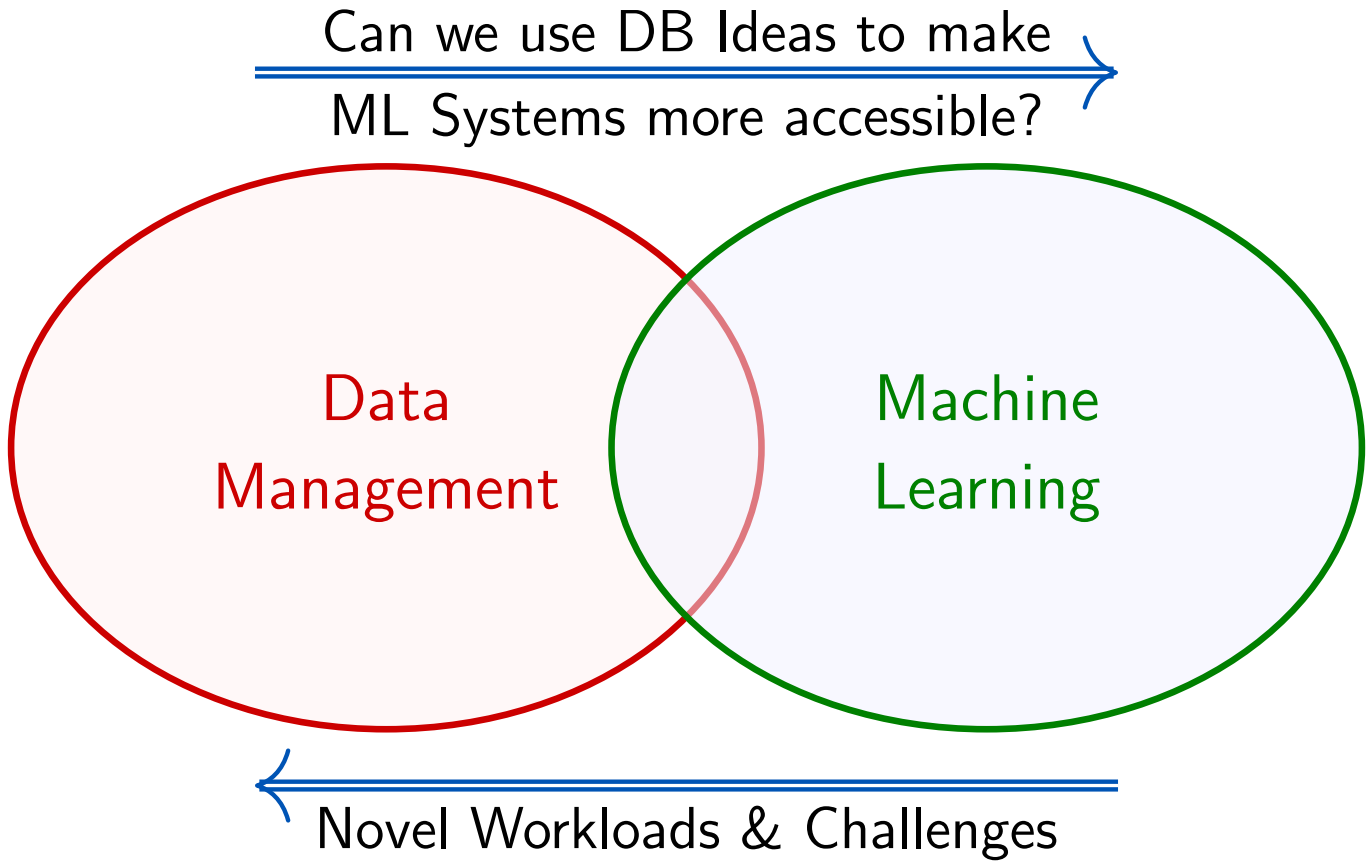
Query

Optimizer

Physical Evaluation Plan



Research Plan

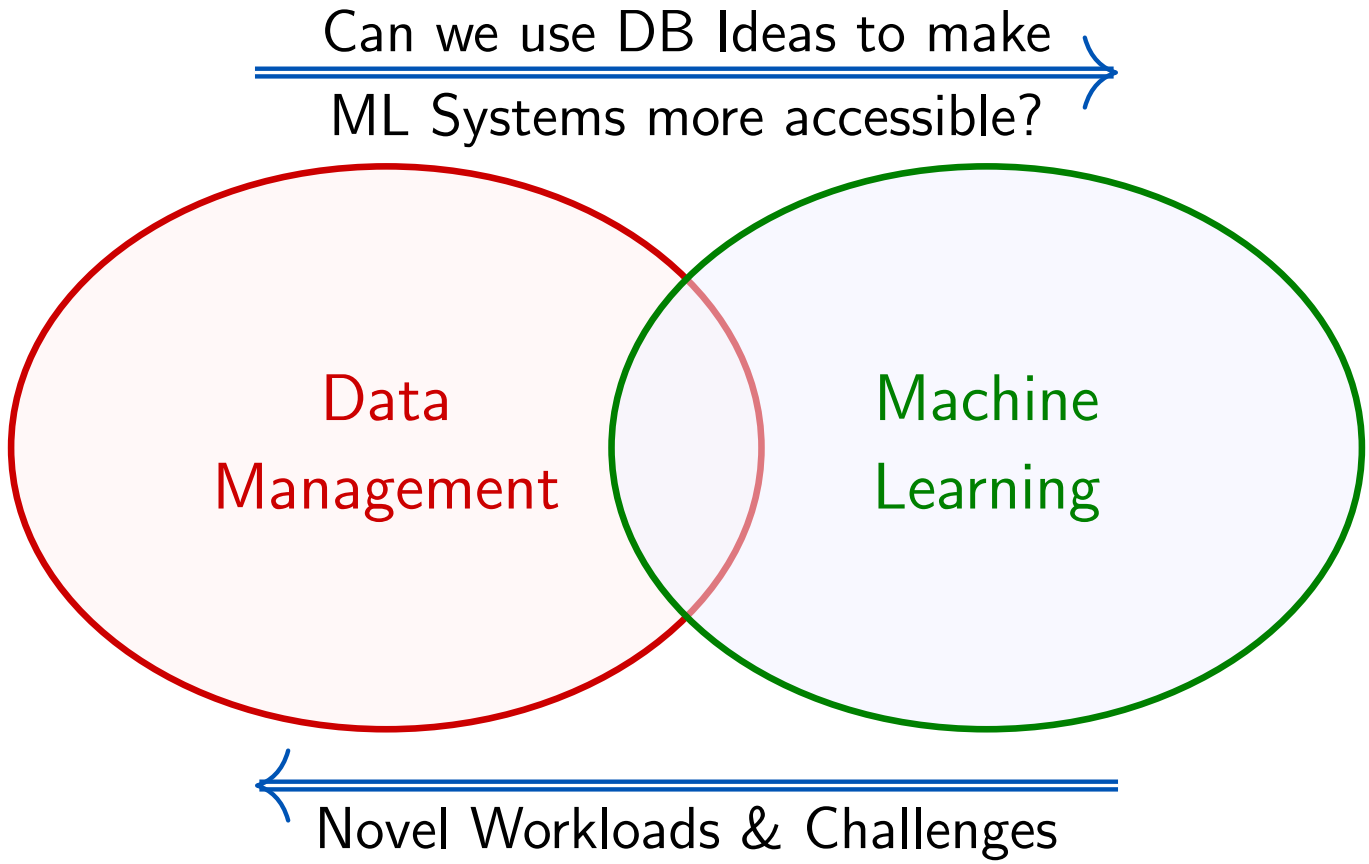


This talk:

Part 1: Learning over Relational Databases

Part 2: Explaining Prediction Outcomes

Research Plan



This talk:

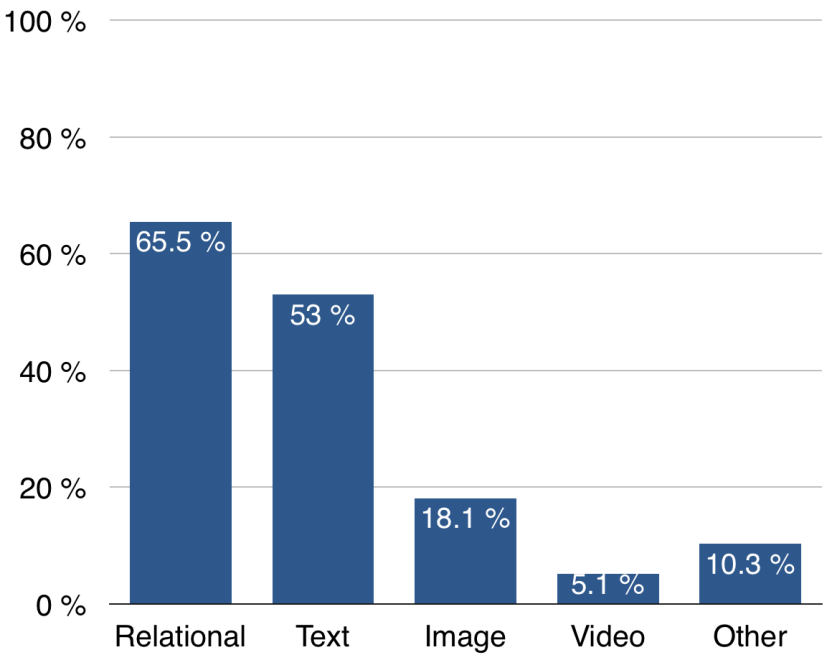
Part 1: Learning over Relational Databases

Part 2: Explaining Prediction Outcomes

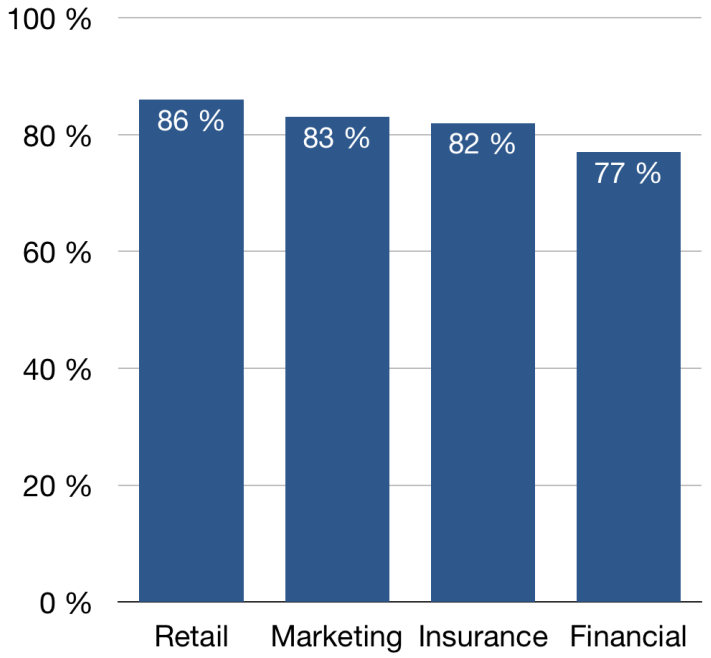
Motivation: Relational Data Is Ubiquitous

Kaggle Survey:

Most Data Scientists use Relational Data at Work!



Overall



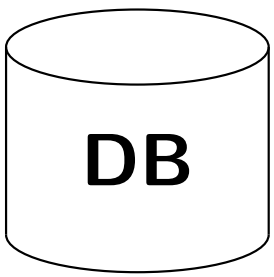
By Industry

Source: The State of Data Science & Machine Learning, Kaggle, October 2017
(survey of 16,000 ML practitioners)

Current State Of Affairs

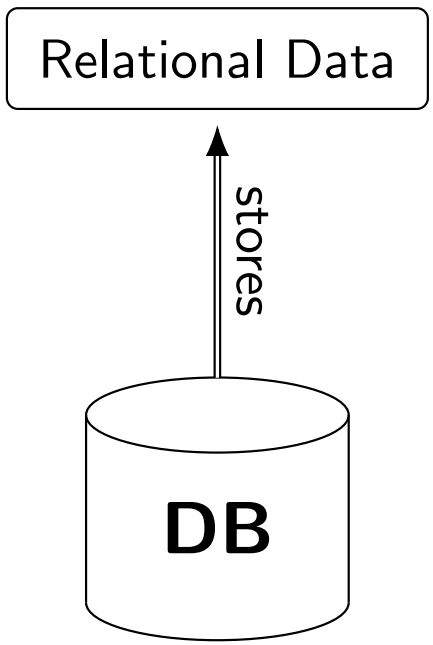
Data Scientist @ Retail Inc.

Current State Of Affairs



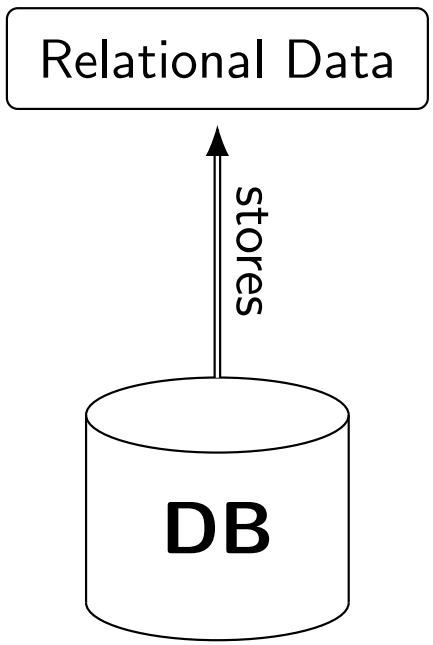
Data Scientist @ Retail Inc.

Current State Of Affairs



Data Scientist @ Retail Inc.

Current State Of Affairs



ML Tool

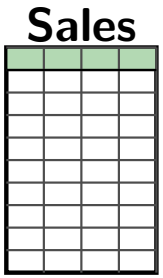
Data Scientist @ Retail Inc.

Current State Of Affairs

Current State Of Affairs

Current State Of Affairs

Current State Of Affairs



Relational Data

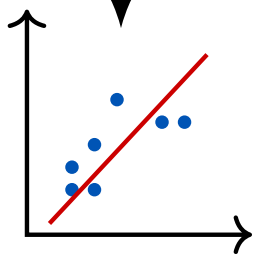
stores

DB

Let's predict sales for next month!

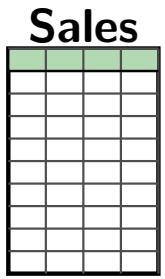
Training Dataset

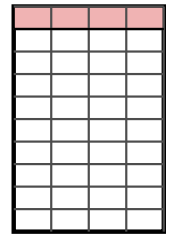
ML Tool

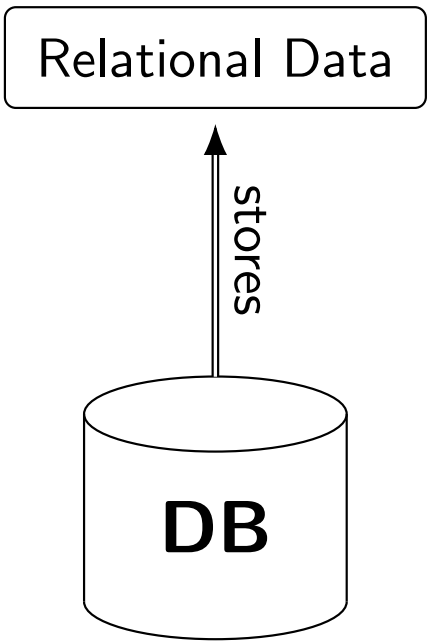


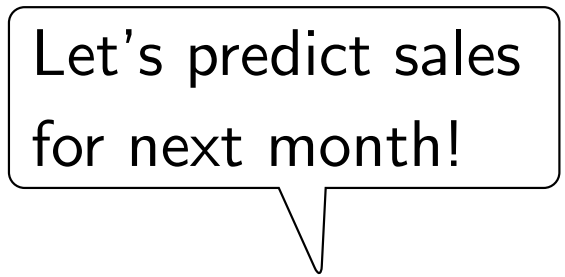
Data Scientist @ Retail Inc.

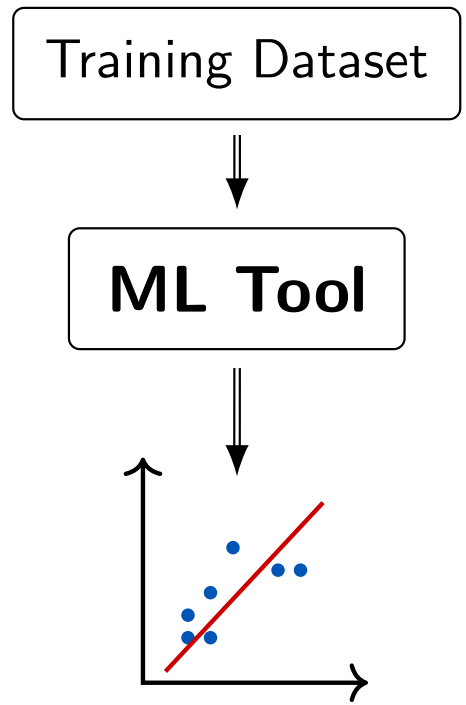
Current State Of Affairs





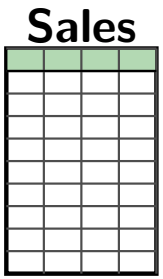


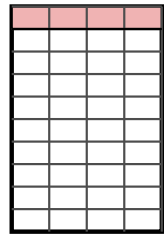


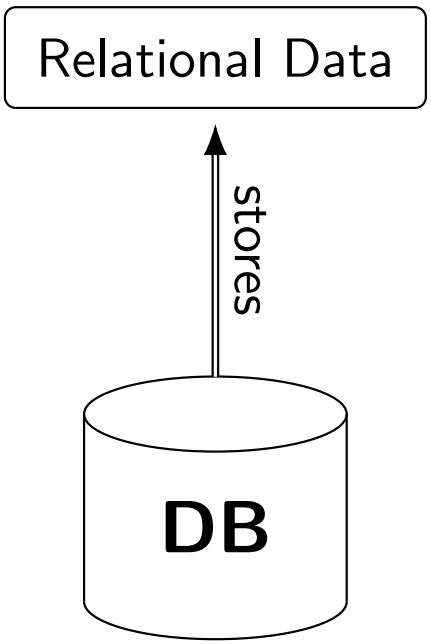


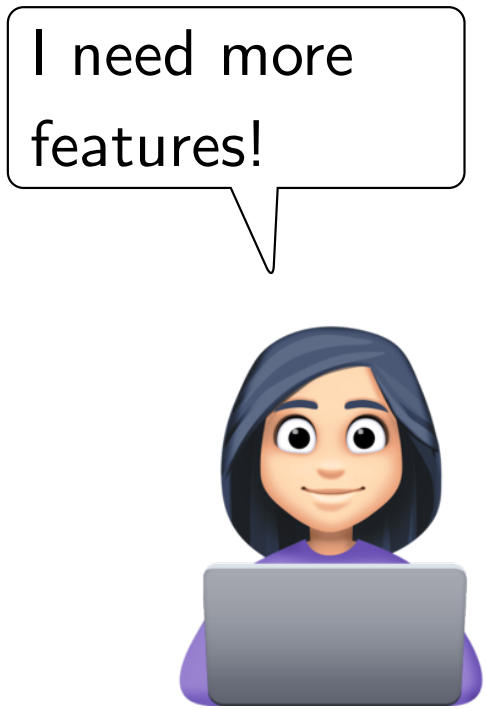
Data Scientist @ Retail Inc.

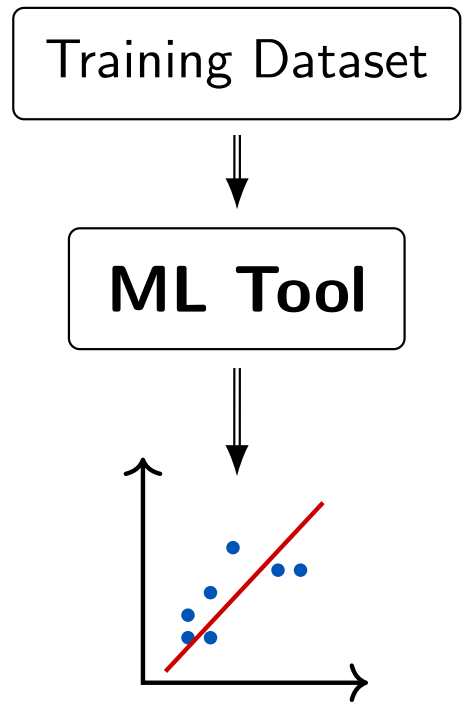
Current State Of Affairs





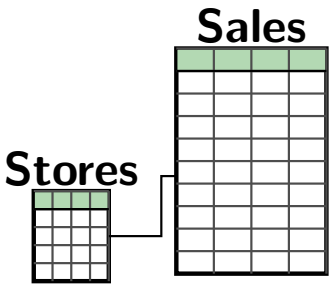


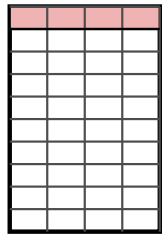


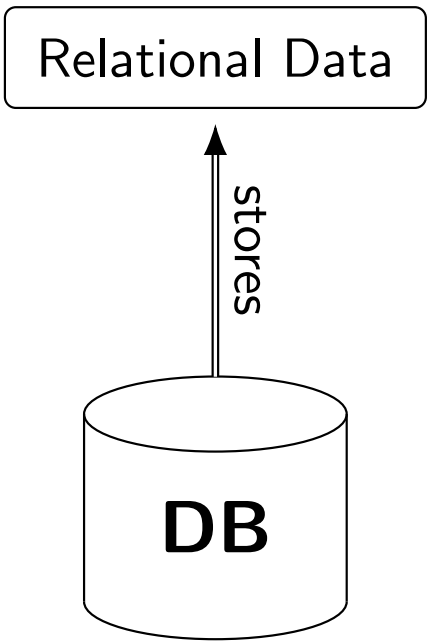


Data Scientist @ Retail Inc.

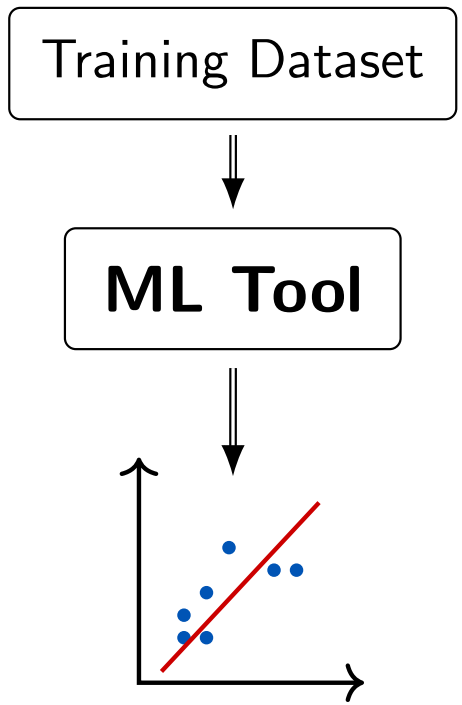
Current State Of Affairs



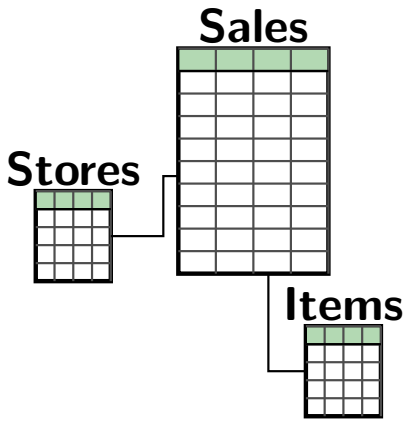




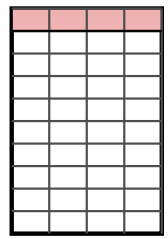
Data Scientist @ Retail Inc.



Current State Of Affairs



extract



Relational Data

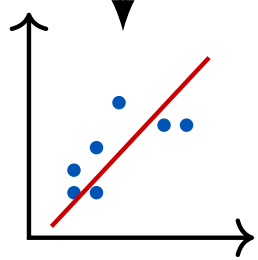
stores

DB

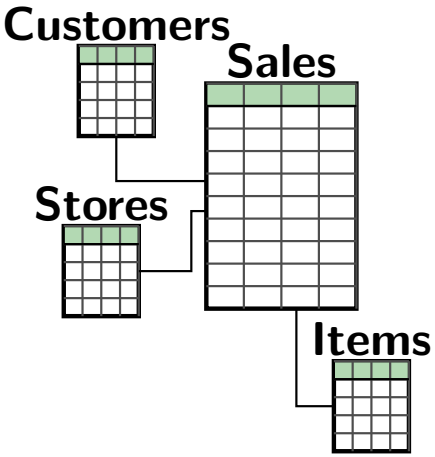
Data Scientist @ Retail Inc.

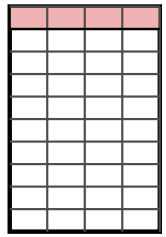
Training Dataset

ML Tool



Current State Of Affairs





Relational Data

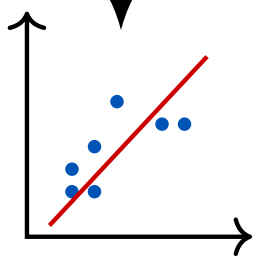
stores

DB

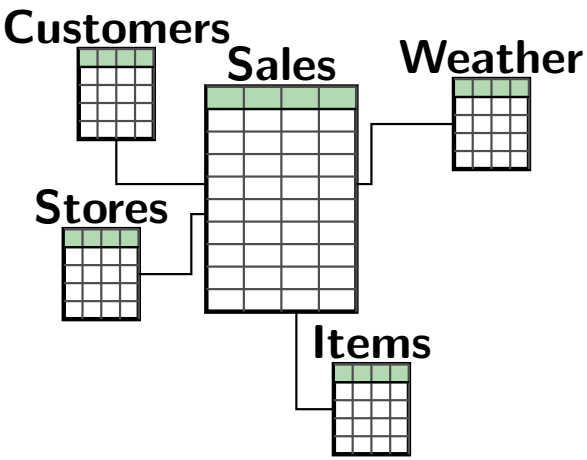
Data Scientist @ Retail Inc.

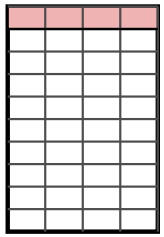
Training Dataset

ML Tool



Current State Of Affairs





Relational Data

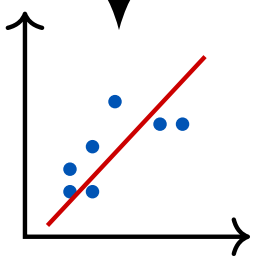
stores

DB

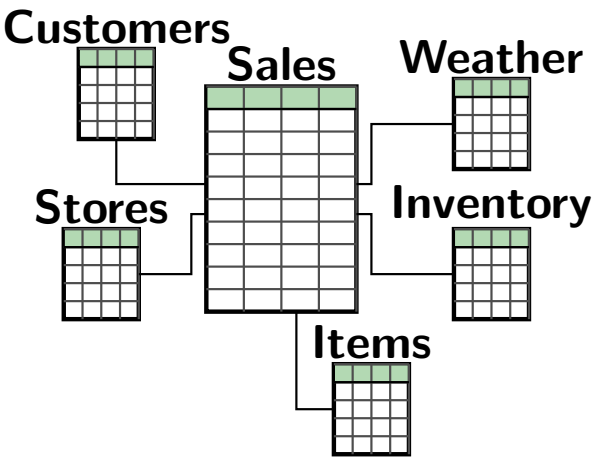
Data Scientist @ Retail Inc.

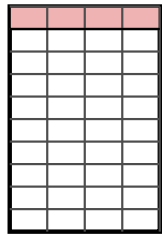
Training Dataset

ML Tool



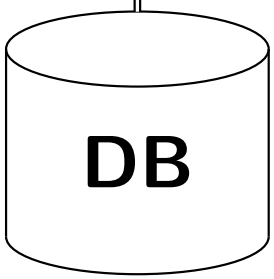
Current State Of Affairs





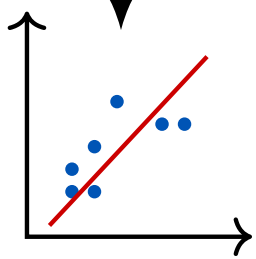
Relational Data

stores



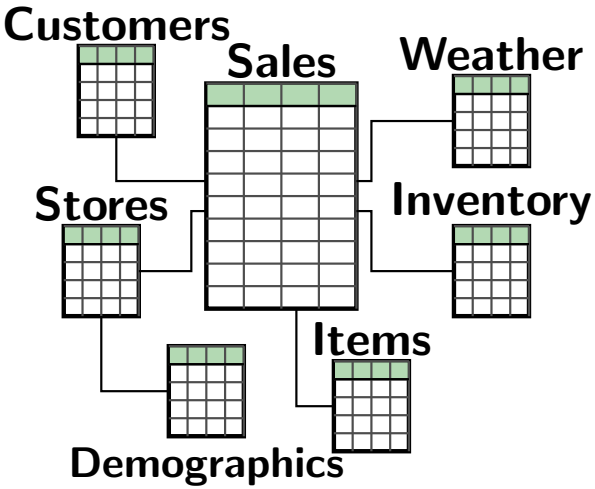
Training Dataset

ML Tool

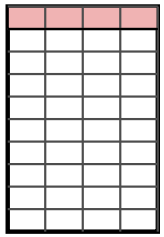


Data Scientist @ Retail Inc.

Current State Of Affairs

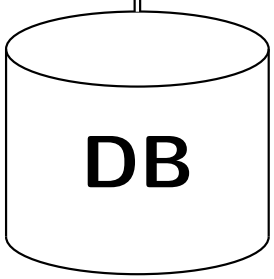


extract



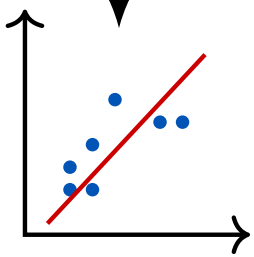
Relational Data

stores



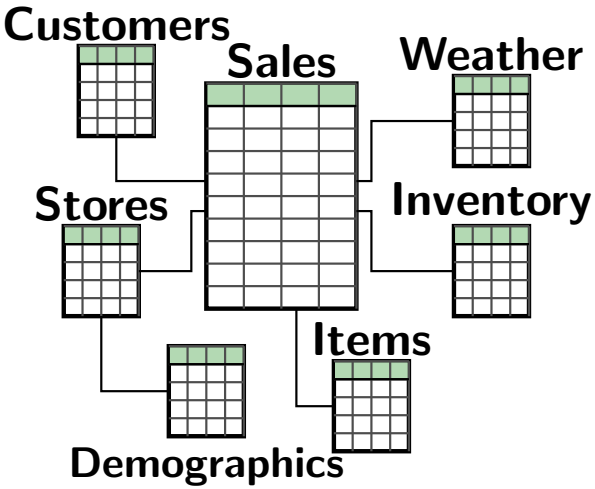
Training Dataset

ML Tool

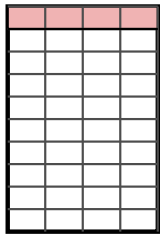


Data Scientist @ Retail Inc.

Current State Of Affairs



extract



Relational Data

stores

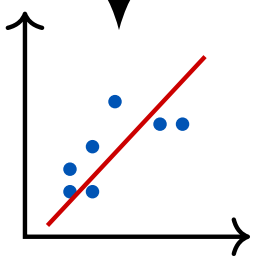
DB

ML Tool requires single table input

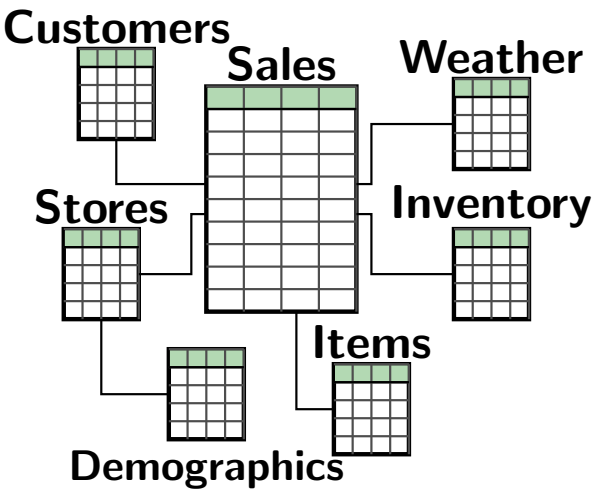
Data Scientist @ Retail Inc.

Training Dataset

ML Tool

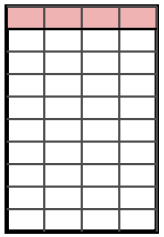


Current State Of Affairs



Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



Relational Data

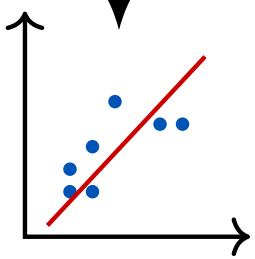
stores

DB

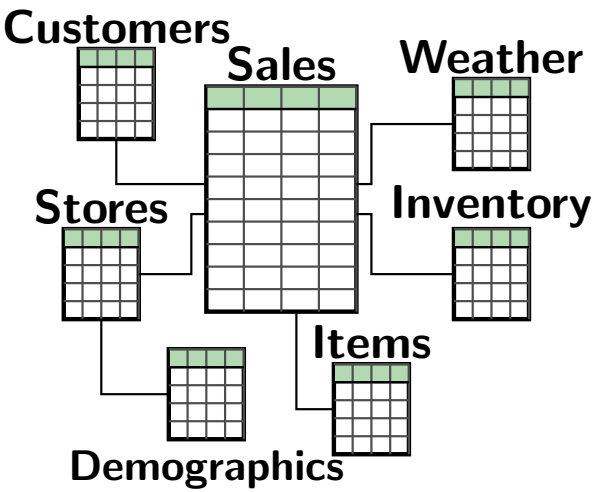
Data Scientist @ Retail Inc.

Training Dataset

ML Tool



Current State Of Affairs



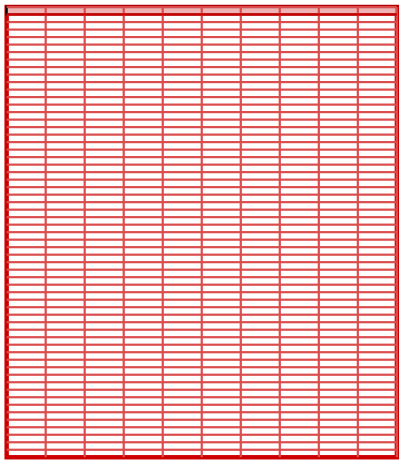
Relational Data

stores

DB

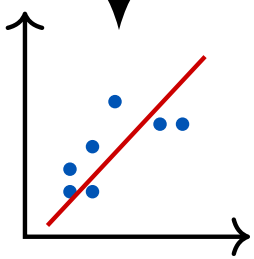
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



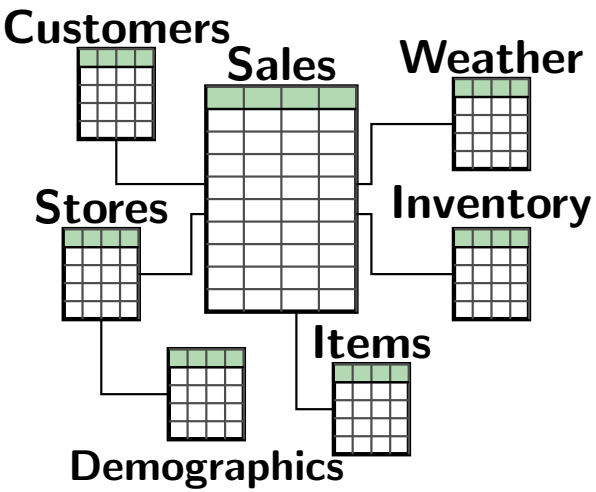
Training Dataset

ML Tool



Data Scientist @ Retail Inc.

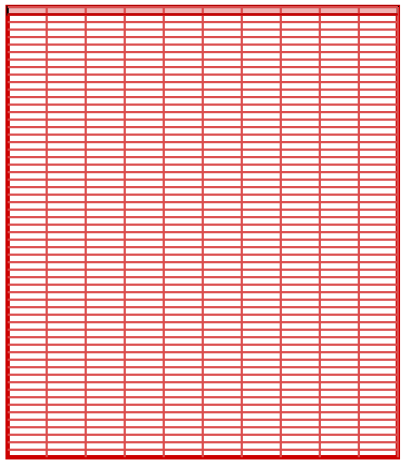
Current State Of Affairs



Relational Data

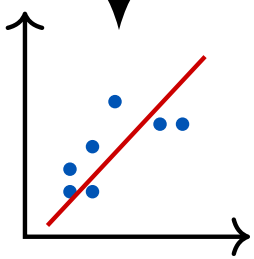
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



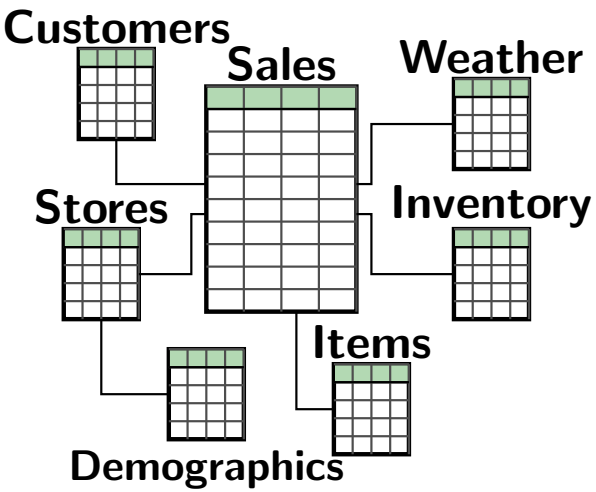
Training Dataset

ML Tool



Limitations:

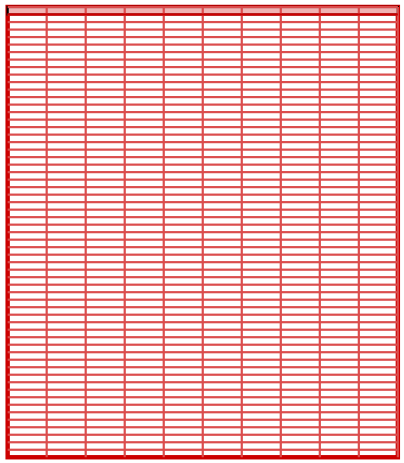
Current State Of Affairs



Relational Data

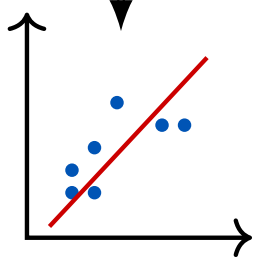
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



Training Dataset

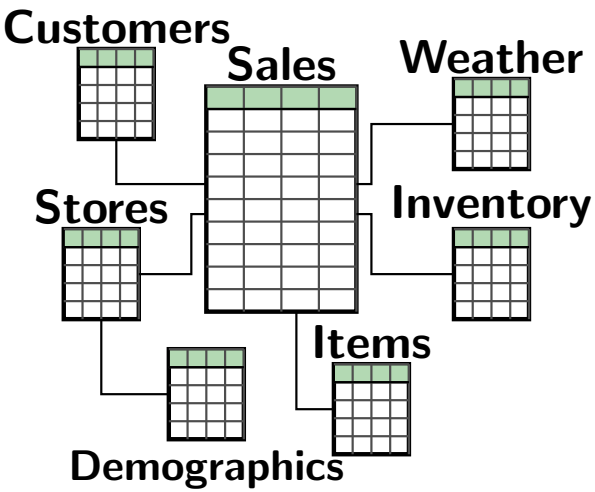
ML Tool



Limitations:

1. Materialize Feature Extraction Query

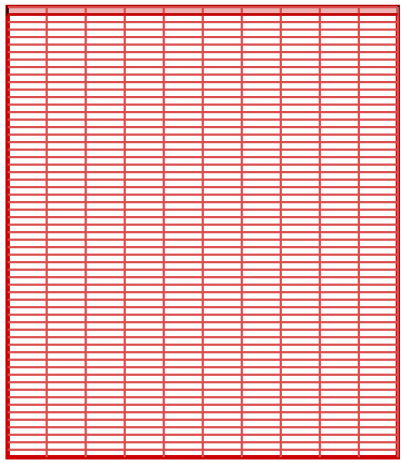
Current State Of Affairs



Relational Data

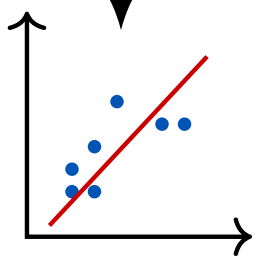
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



Training Dataset

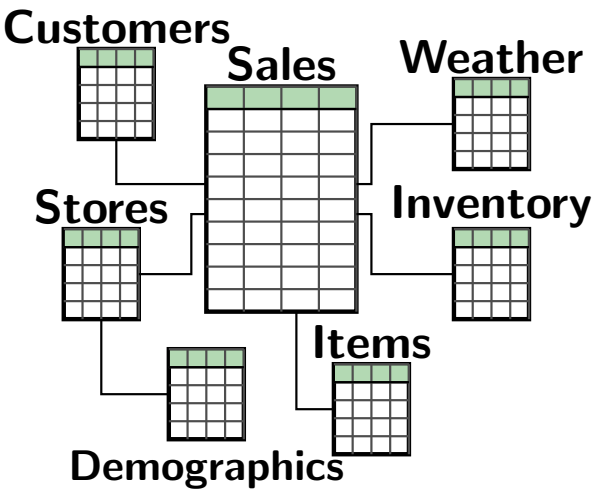
ML Tool



Limitations:

1. Materialize Feature Extraction Query
2. Export / import data

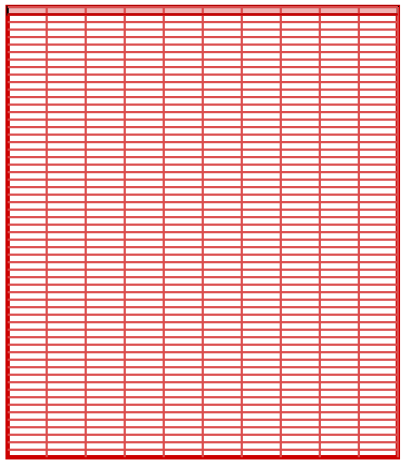
Current State Of Affairs



Relational Data

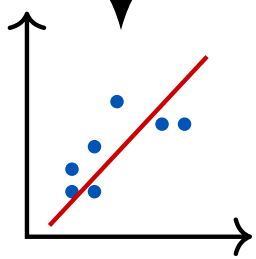
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



Training Dataset

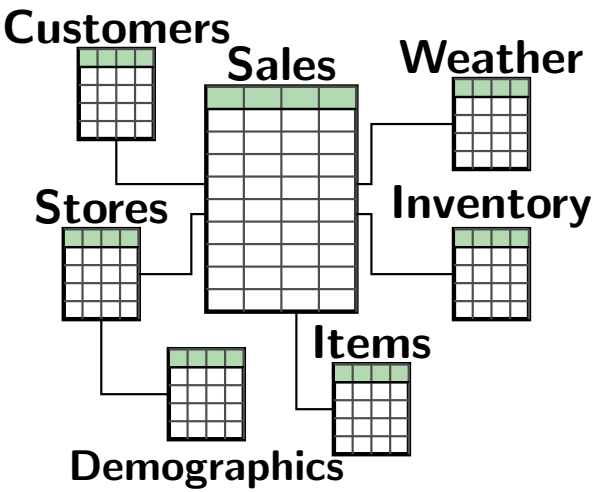
ML Tool



Limitations:

1. Materialize Feature Extraction Query
2. Export / import data
3. One-hot encode training data

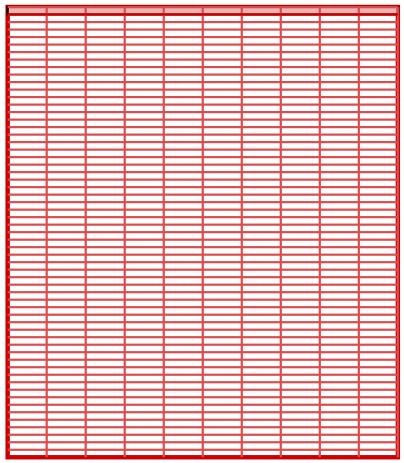
Current State Of Affairs



Relational Data

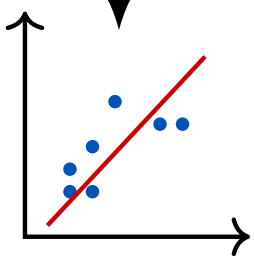
Feature Extraction Query

$$\text{Sales} \bowtie \text{Stores} \bowtie \text{Items} \bowtie \text{Customers} \bowtie \text{Weather} \bowtie \text{Inventory} \bowtie \text{Demographics}$$



Training Dataset

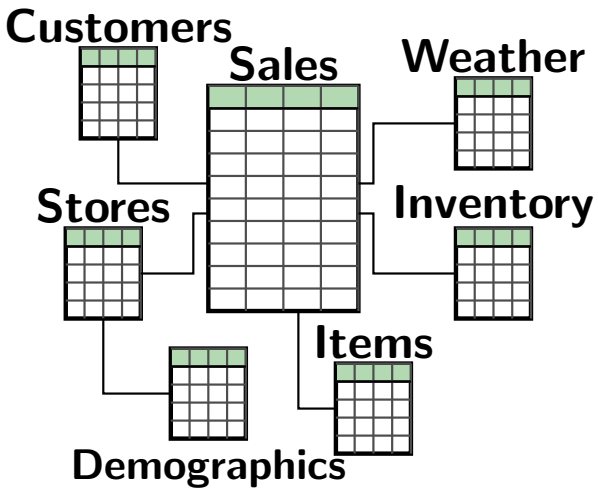
ML Tool



Limitations:

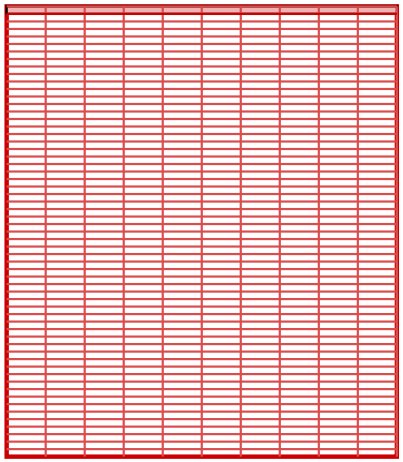
1. Materialize Feature Extraction Query
2. Export / import data
3. One-hot encode training data
4. Throws away relational structure

Tight Integration Of DB And ML Tasks



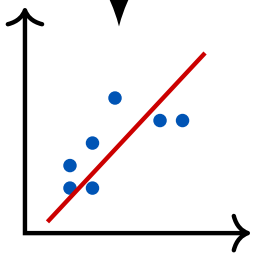
Relational Data

Feature Extraction Query

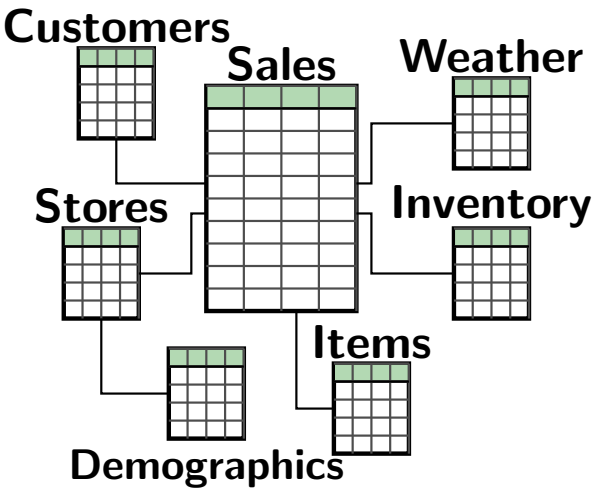


Training Dataset

ML Tool



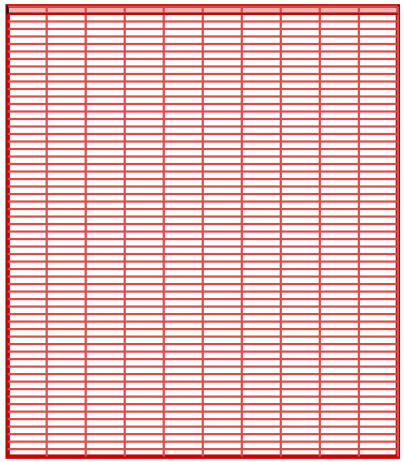
Tight Integration Of DB And ML Tasks



Relational Data

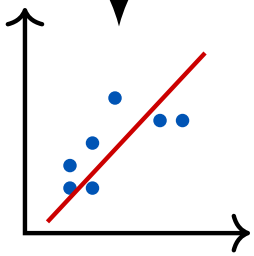
Batch of Aggregate Queries

Feature Extraction Query

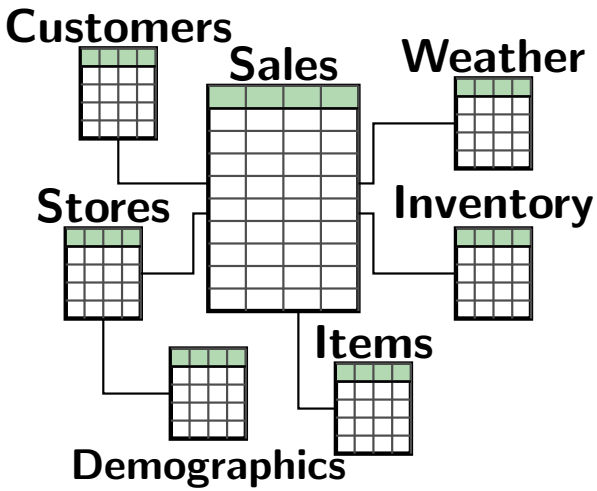


Training Dataset

ML Tool

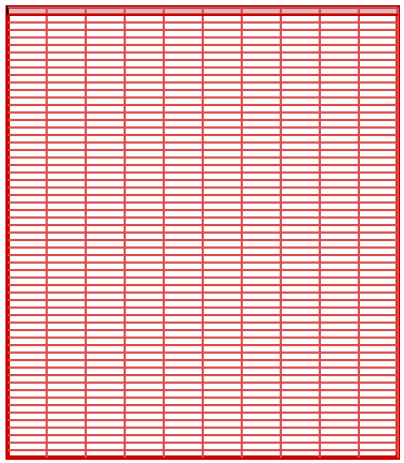


Tight Integration Of DB And ML Tasks



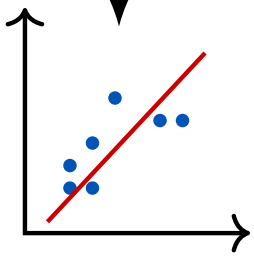
Relational Data

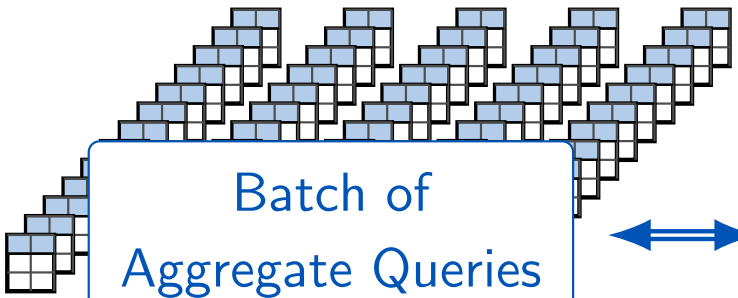
Feature Extraction Query



Training Dataset

ML Tool

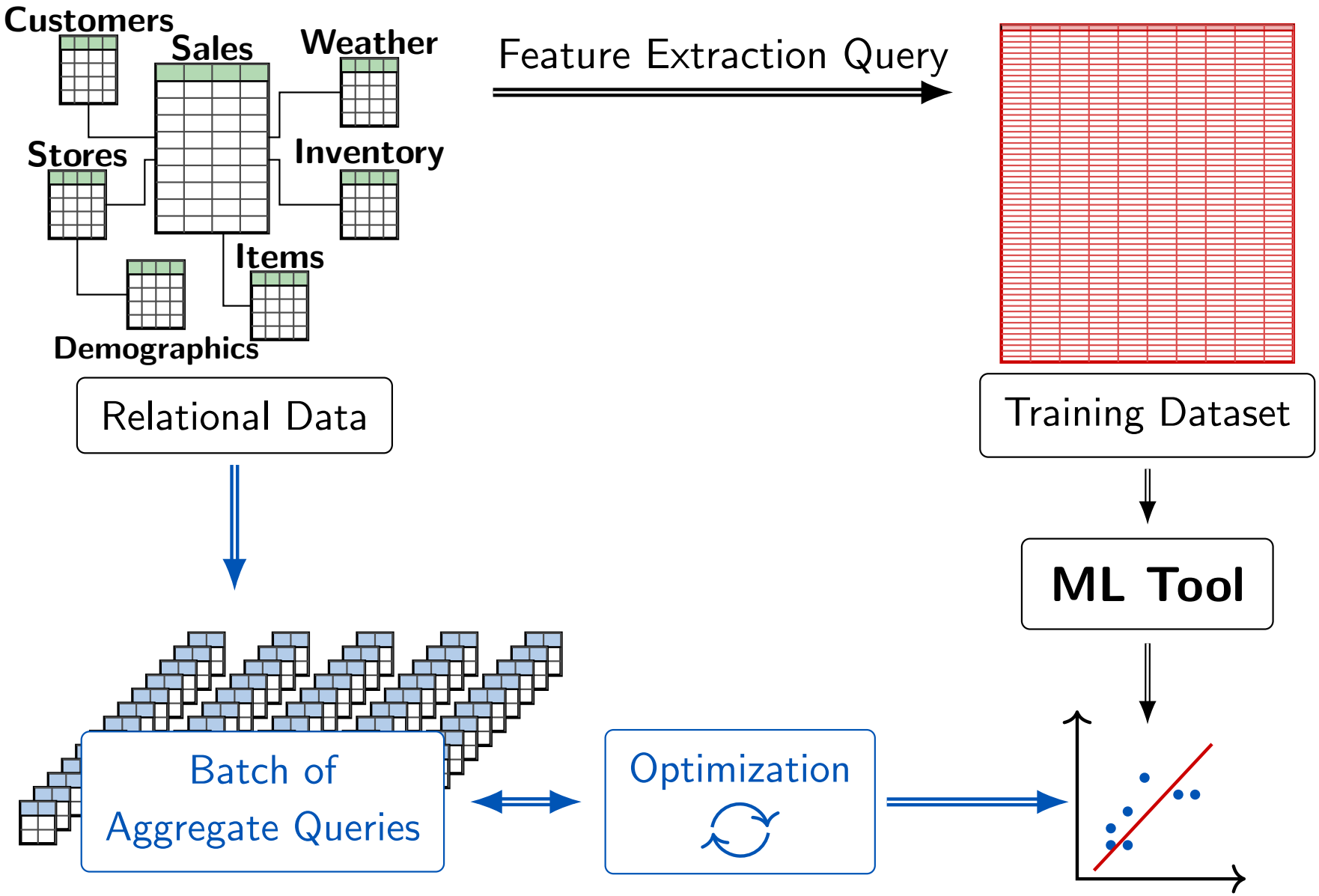




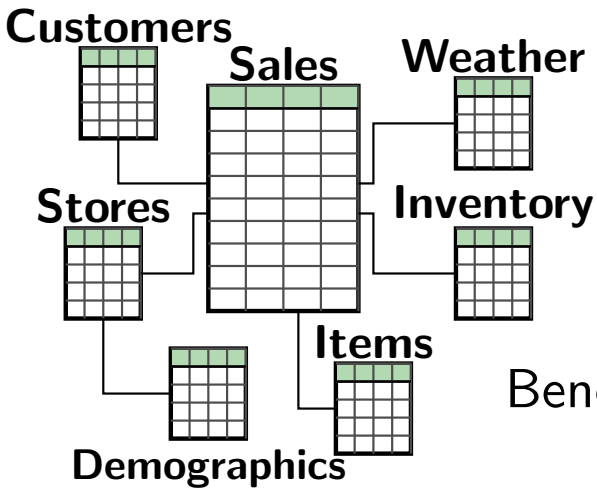
Batch of Aggregate Queries

Optimization

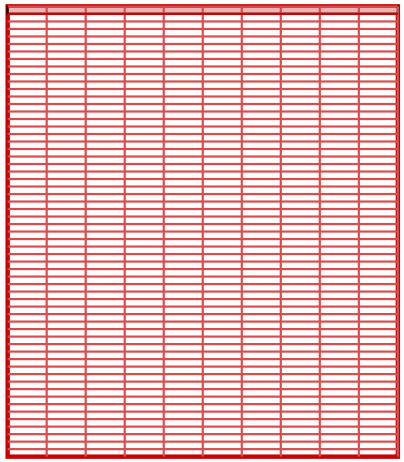
Tight Integration Of DB And ML Tasks



Tight Integration Of DB And ML Tasks



Feature Extraction Query

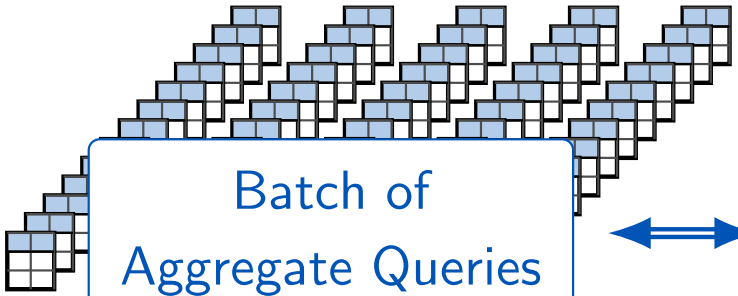


Benefits:

Relational Data

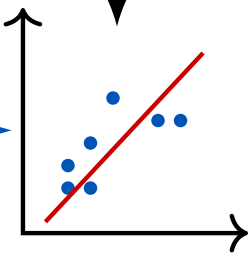
Training Dataset

ML Tool

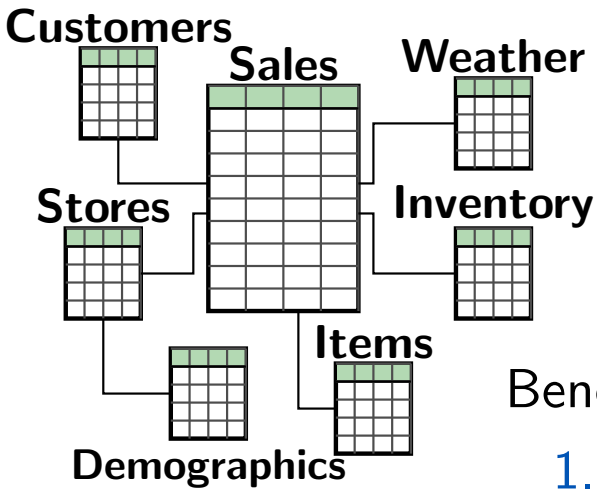


Batch of Aggregate Queries

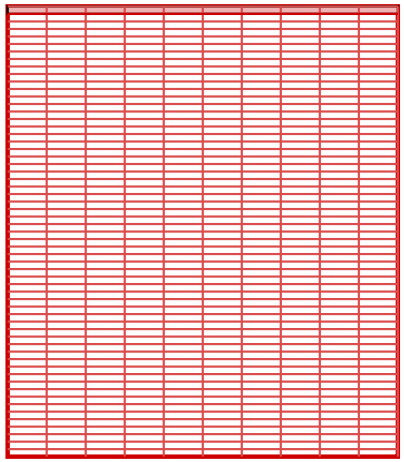
Optimization



Tight Integration Of DB And ML Tasks



Feature Extraction Query



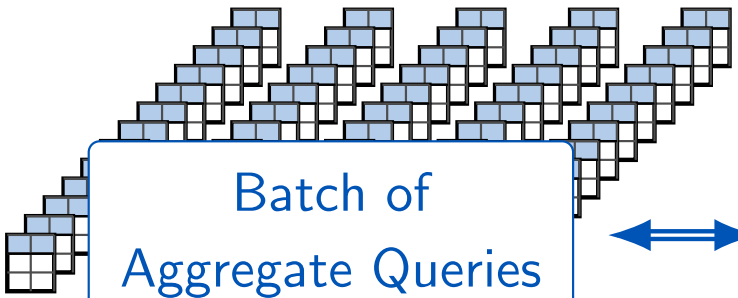
Benefits:

1. Integrated evaluation of DB and ML tasks

Relational Data

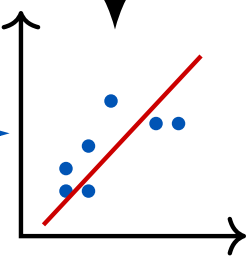
Training Dataset

ML Tool

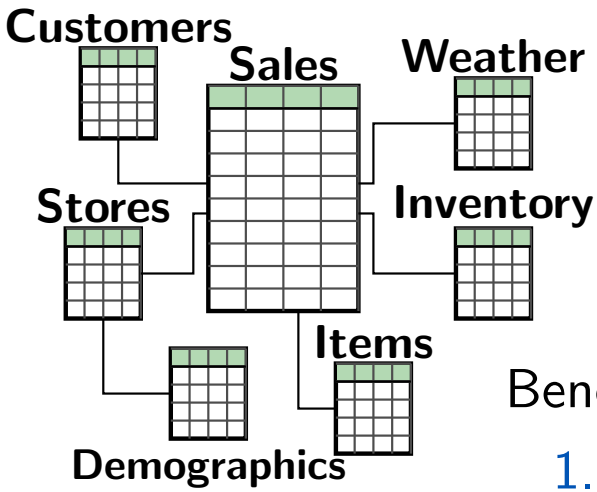


Batch of Aggregate Queries

Optimization

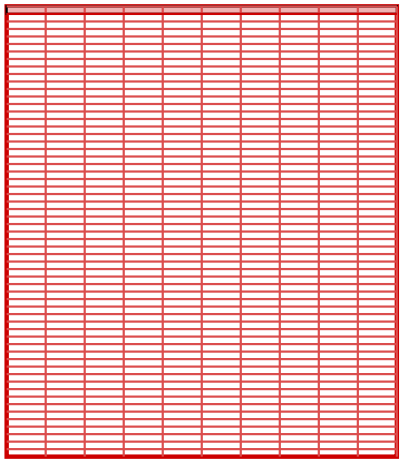


Tight Integration Of DB And ML Tasks



Relational Data

Feature Extraction Query

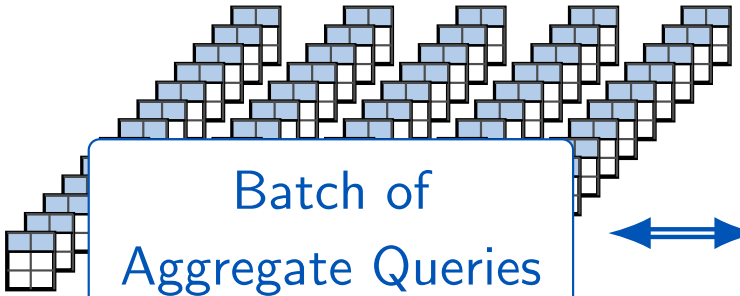


Training Dataset

Benefits:

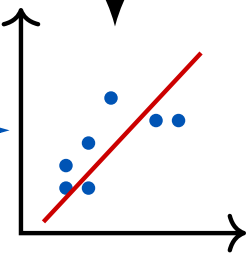
1. Integrated evaluation of DB and ML tasks
2. Exploit Relational Structure

ML Tool

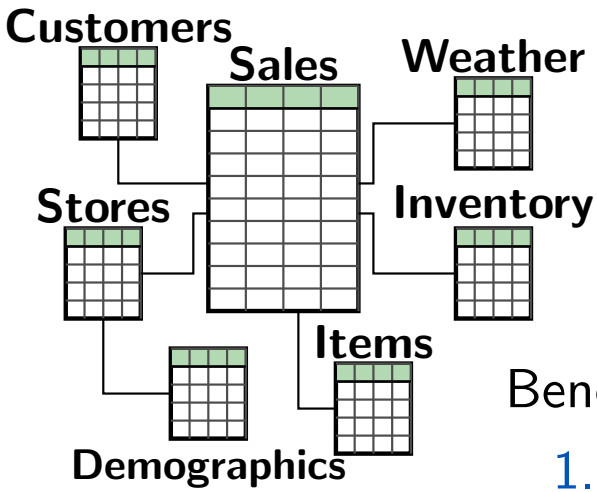


Batch of Aggregate Queries

Optimization

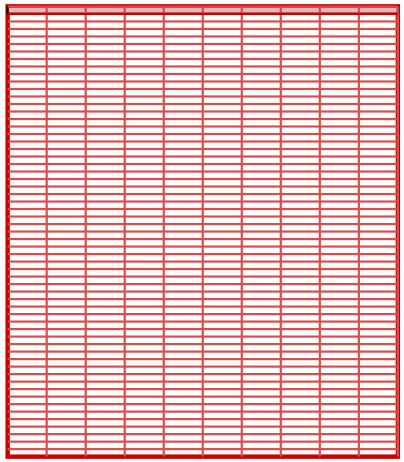


Tight Integration Of DB And ML Tasks



Relational Data

Feature Extraction Query

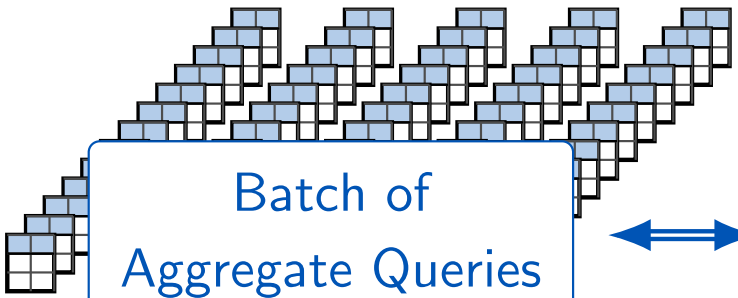


Training Dataset

Benefits:

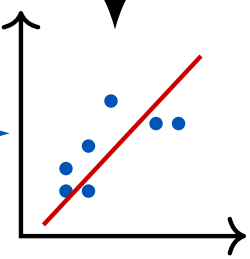
1. Integrated evaluation of DB and ML tasks
2. Exploit Relational Structure
3. Exploit DB Optimizations
   - Push ML tasks past joins

ML Tool

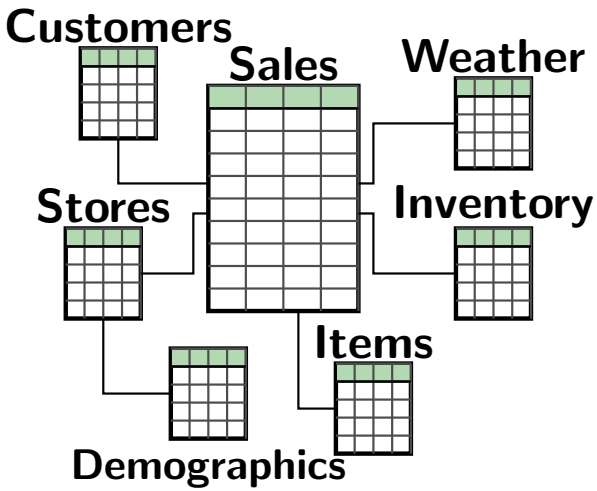


Batch of Aggregate Queries

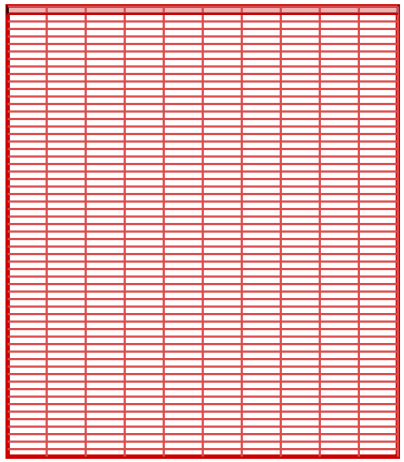
Optimization



Tight Integration Of DB And ML Tasks



Feature Extraction Query

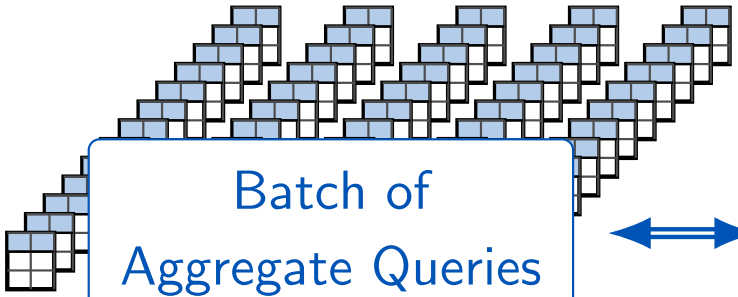


Relational Data

Training Dataset

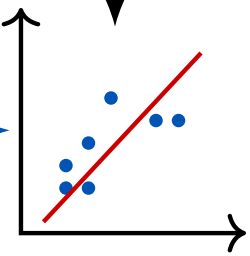
Structure Aware Learning

ML Tool

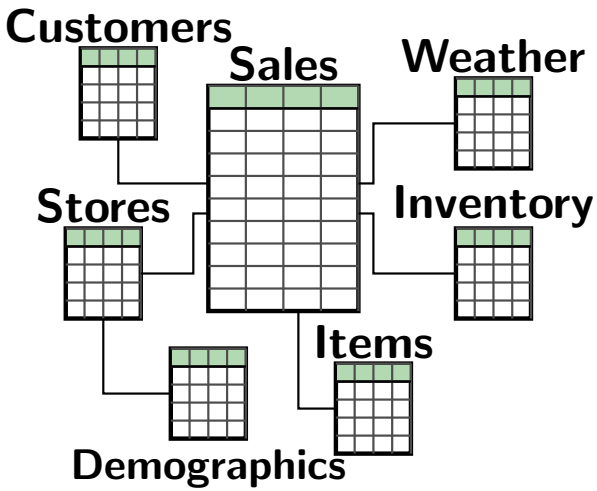


Batch of Aggregate Queries

Optimization

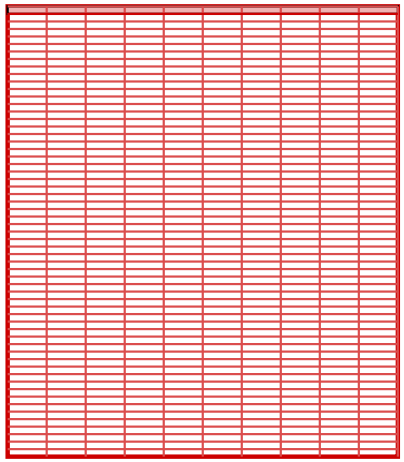


Tight Integration Of DB And ML Tasks



Relational Data

Feature Extraction Query

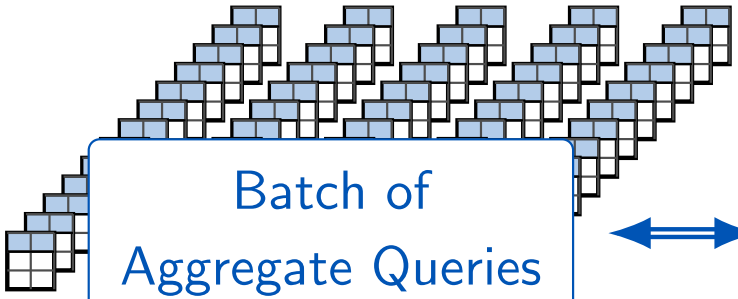


Training Dataset

Structure Agnostic Learning

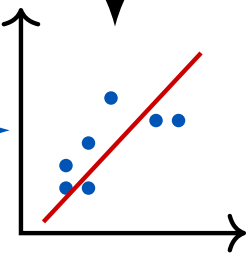
Structure Aware Learning

ML Tool



Batch of Aggregate Queries

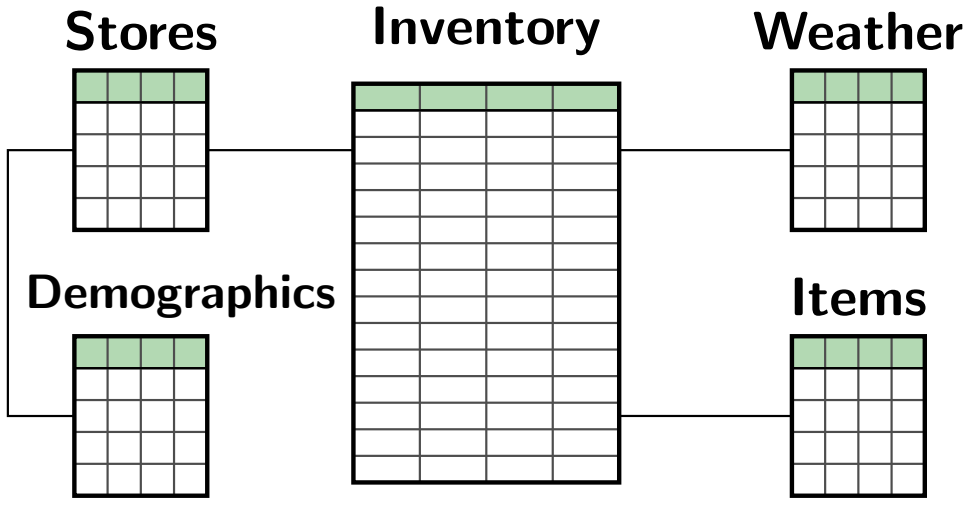
Optimization



Does It Pay Off?

Structure-Aware Learning can be **faster** than computing the **Feature Extraction Query**!

Example: A Retail Use Case



Relation	Cardinality	Attributes	File Size (CSV)
Inventory	84,055,817	4	2 GB
Items	5,618	5	129 KB
Stores	1,317	15	139 KB
Demographics	1,302	16	161 KB
Weather	1,159,457	8	33 MB
Join	84,055,817	44	23GB

The join result is 10× larger than the input database!

Structure-Aware versus Structure-Agnostic Learning

Train linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow	
	Time	**Size**
Database	–	2.1 GB
Join	152.06 secs	23 GB
Export	351.76 secs	23 GB
Shuffling	5,488.73 secs	23 GB
Query batch	–	–
Grad Descent	7,249.58 secs	–
Total time	13,242.13 secs	

Intel i7-4770, 3.4GHz, 32GB, 8 cores [SIGMOD'19]

TensorFlow: 1 epoch with 100K tuple batch **Size:** File Size in CSV

Structure-Aware versus Structure-Agnostic Learning

Train linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow		Structure Aware	
	Time	**Size**	**Time**	**Size**
Database	–	2.1 GB	–	2.1 GB
Join	152.06 secs	23 GB	–	–
Export	351.76 secs	23 GB	–	–
Shuffling	5,488.73 secs	23 GB	–	–
Query batch	–	–	6.08 secs	37 KB
Grad Descent	7,249.58 secs	–	0.05 secs	–
Total time	13,242.13 secs		6.13 secs	

Intel i7-4770, 3.4GHz, 32GB, 8 cores [SIGMOD'19]

TensorFlow: 1 epoch with 100K tuple batch **Size:** File Size in CSV

Models Under Consideration

So far:

- (Robust) Linear regression [SIGMOD'16, PODS'19]
- Polynomial regression [SigmodRec'16]
- Factorization machines [PODS'18, TODS'20, DEEM'18]
- Classification/regression trees [SIGMOD'19]
- Mutual information [SIGMOD'19]
- Chow Liu trees [SIGMOD'19]
- k-means clustering [AISTATS'20]
- PCA [TODS'20]
- SVM [PODS'19]

Ongoing: Sum-product networks, Boosting regression trees, Random forests, QR-decomposition, SVD

All these cases can benefit from structure-aware computation

How To Achieve This Performance Improvement?

Idea 1: Turn Data Intensive ML Computation into DB Problem

Idea 2: Exploit Problem Structure to Lower Complexity

Idea 3: DB Optimizations to Lower Constant Factors

How To Achieve This Performance Improvement?

Idea 1: **Turn Data Intensive ML Computation into DB Problem**

Idea 2: Exploit Problem Structure to Lower Complexity

Idea 3: DB Optimizations to Lower Constant Factors

Through DB Glasses, Everything Is A Batch Of Queries

Workload	Query Batch	# Queries
Linear Regression Covariance Matrix	$\texttt{SUM}(X_i * X_j)$ $\texttt{SUM}(X_i)\ \texttt{GROUP BY}\ X_j$ $\texttt{SUM(1) GROUP BY}\ X_i, X_j$	814
Decision Tree (Regression, 1 Node)	$\texttt{VARIANCE}(Y)\ \texttt{WHERE}\ X_j = c_j$	3,141
Rk-Means	$\texttt{SUM(1) GROUP BY}\ X_j$ $\texttt{SUM(1) GROUP BY}\ C_1, \ldots, C_k$	41

(# Queries shown for Retailer dataset with 39 attributes)

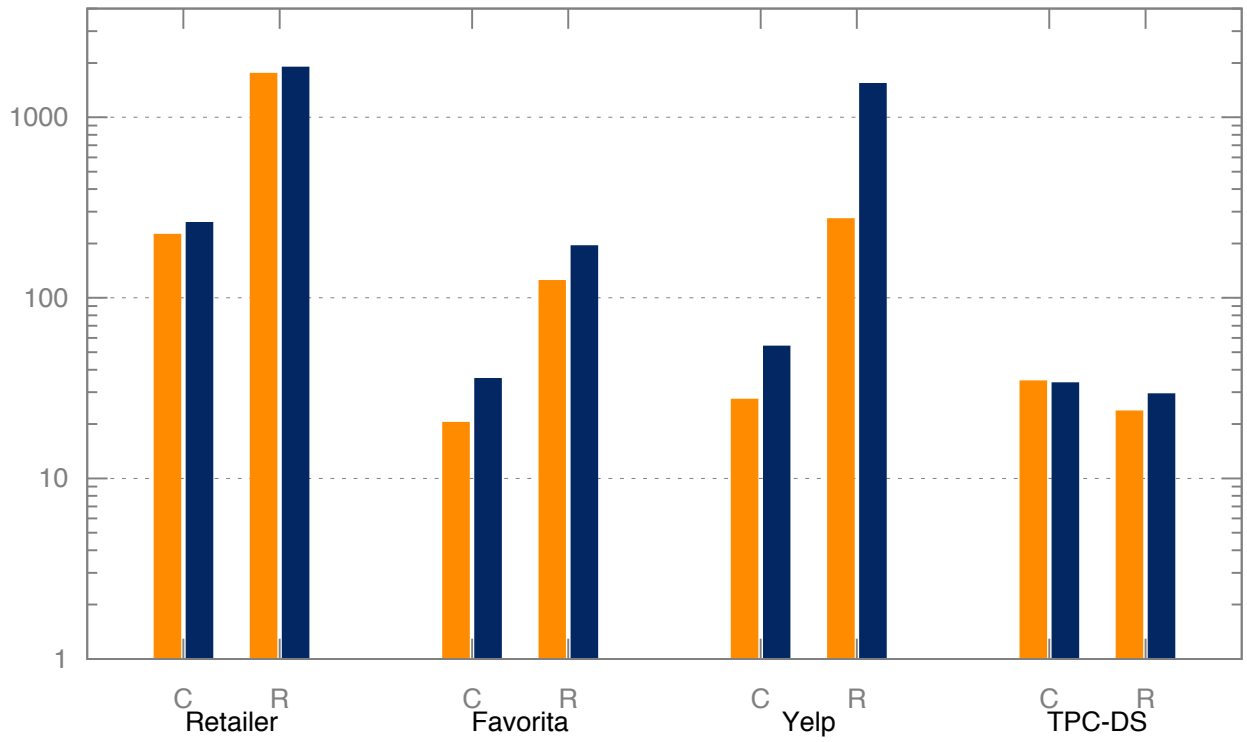
AMPLE opportunities for sharing computation in a batch

Natural Attempt:

Use Existing DB System to Compute Query Batch

Existing DBMSs Are NOT Designed For Query Batches

Relative Speedup for **Our Approach** over **DBX** and **MonetDB**



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

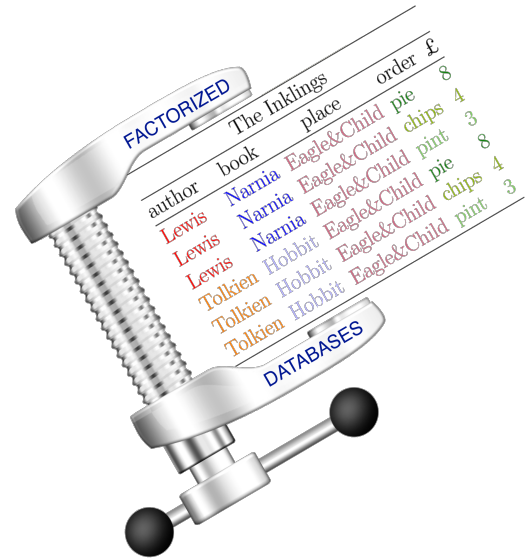
Multi-Query Optimization techniques are not designed for Aggregate Queries

How To Achieve This Performance Improvement?

Idea 1: Turn Data Intensive ML Computation into DB Problem

Idea 2: **Exploit Problem Structure to Lower Complexity**

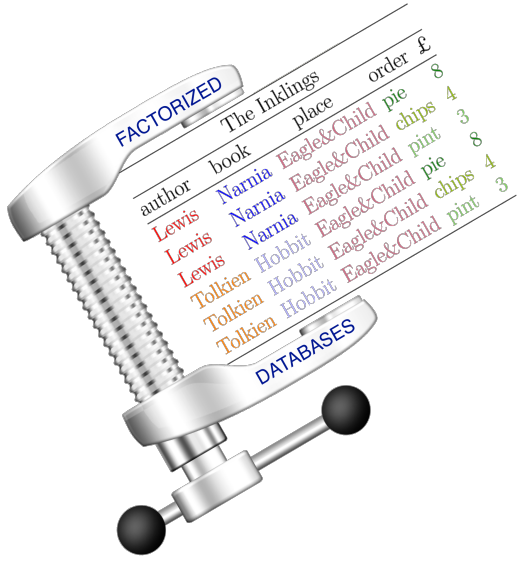
Idea 3: Use DB Optimizations to Lower the Constant Factors



Factorized Query Evaluation

Factorized query evaluation unifies three powerful ideas:

1. worst-case optimal joins, [Veldhuizen'14, NRR'13]
   - e.g., Leapfrog Trie-Join, Generic Join
2. query-plans defined by hypertree decompositions,
   - For this talk, hypertree decompositions = join trees
3. systematically pushing aggregates past join.



Details:
Factorized Databases, Sigmod Records, 2016

Factorized Query Evaluation

Join Query Q with $|Q| =$ # relations, $N \geq$ size of relations

- **Computing Join Query Q:** [NPRR'12]

$$O(N^{\rho^*(Q)})$$

  - $\rho* =$ fractional edge cover number of Q

Factorized Query Evaluation

Join Query Q with $|Q| =$ # relations, $N \geq$ size of relations

- **Computing Join Query Q:** [NPRR'12]

$$O(N^{\rho^*(Q)})$$

  - $\rho* =$ fractional edge cover number of Q

- **Linear Regression Model* over Q:** [PODS'18, TODS'20]

$$O(N^{fhtw(Q)})$$

  - $fhtw =$ fractional hypertree width of Q

$$1 \leq \underbrace{fhtw(Q) \leq \rho^*(Q)}_{\text{gap up to } |Q|-1} \leq |Q|$$

* Assuming domain size of categorical features is $<< N$

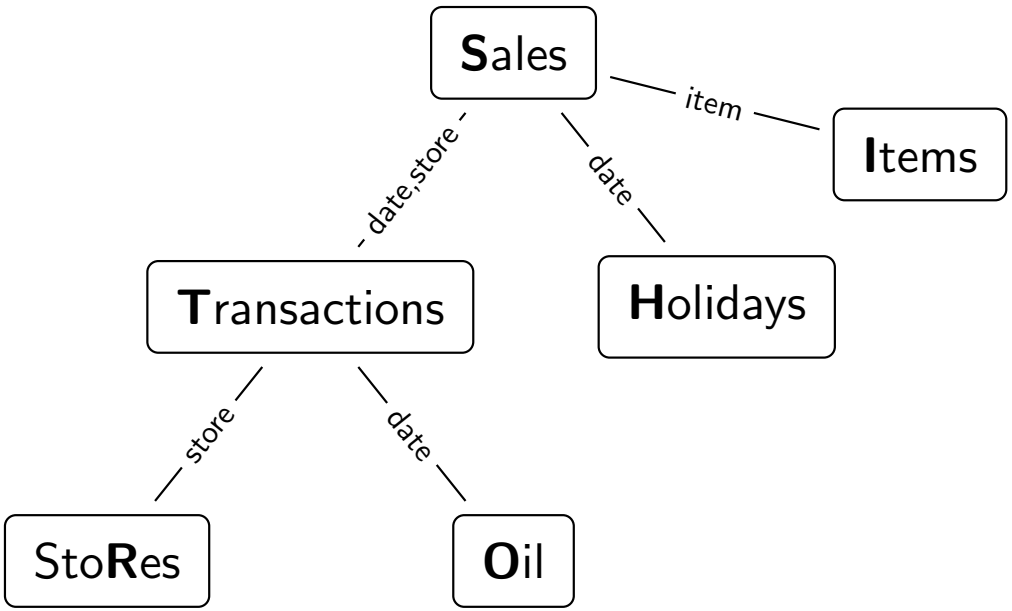
Factorized Query Evaluation: Example

Q_1: `SELECT SUM (units)` `FROM D`

Q_2: `SELECT color, SUM (units` $\cdot\, g($`item`$)$`)` `FROM D GROUP BY color`

Q_3: `SELECT store, SUM (`$g($`item`$) \cdot h($`date`$)$`)` `FROM D GROUP BY store`

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



Favorita Kaggle Dataset: Units Sales for different store, date, item.

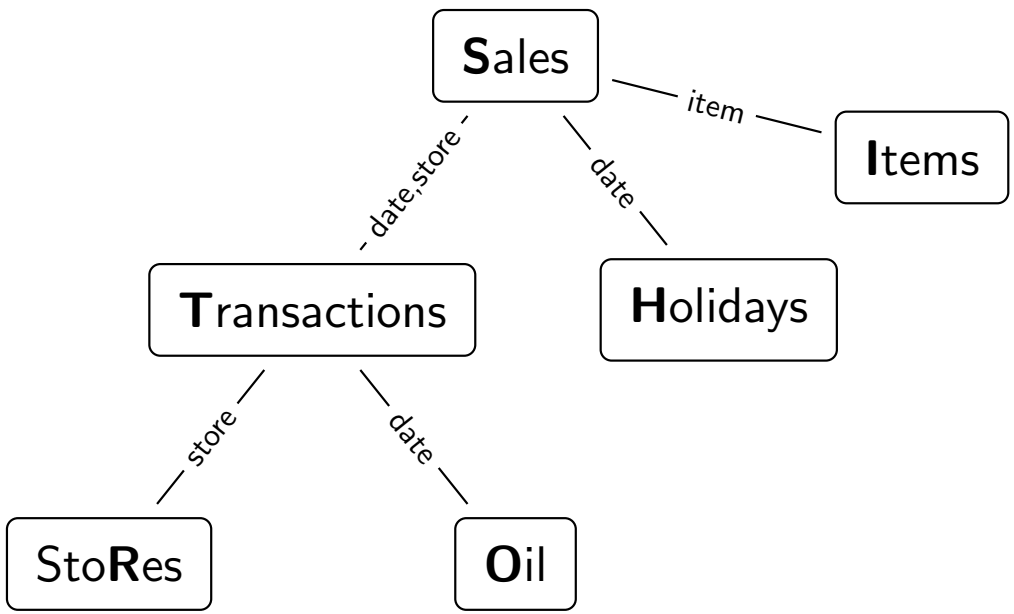
Factorized Query Evaluation: Example

Q_1: `SELECT SUM (units) FROM D`

Q_2: `SELECT color, SUM (units` $\cdot\, g($`item`$)$`) FROM D GROUP BY color`

Q_3: `SELECT store, SUM (`$g($`item`$)\cdot h($`date`$)$`) FROM D GROUP BY store`

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



Aggregate Pushdown: Break down query into views [Yannakakis'81]

Similar to **Variable Elimination** and **Message Passing** [Pearl'82]

Factorized Query Evaluation: Example

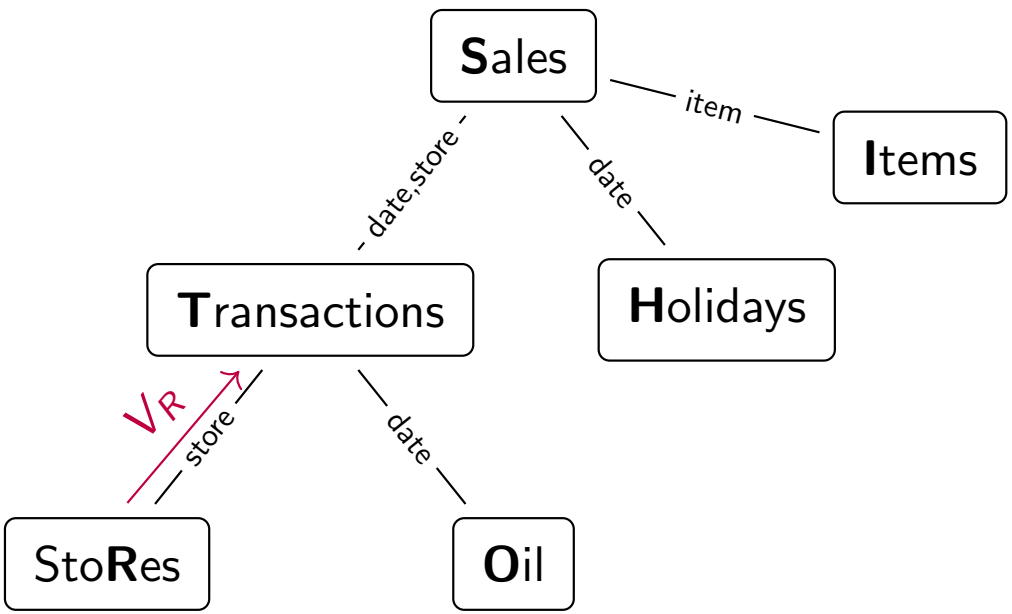
Q_1: `SELECT SUM (units` $*$ c_R`)` `FROM D`

Q_2: `SELECT color, SUM (units` $\cdot$ g`(item)) FROM D GROUP BY color`

Q_3: `SELECT store, SUM (`g`(item)` $\cdot$ h`(date)) FROM D GROUP BY store`

... where $D = S \bowtie T \bowtie V_R \bowtie I \bowtie O \bowtie H$

V_R : `store,SUM(1) AS` c_R
`FROM R`



Aggregate Pushdown: Break down query into views [Yannakakis'81]

Similar to **Variable Elimination** and **Message Passing** [Pearl'82]

Factorized Query Evaluation: Example

Q_1: SELECT SUM ($\texttt{units} * c_R * c_O$) FROM D

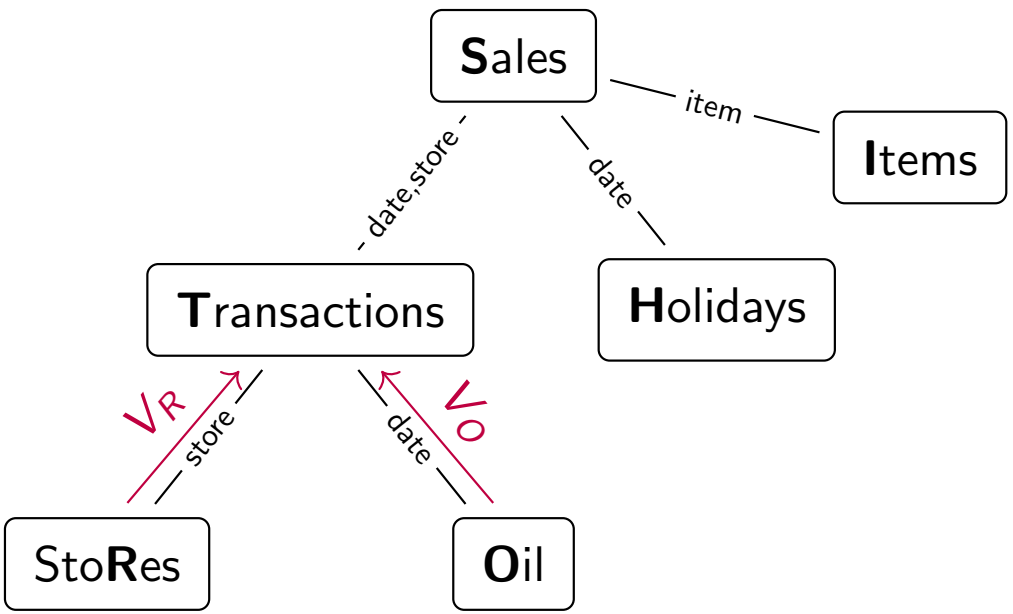
Q_2: SELECT color, SUM ($\texttt{units} \cdot g(\texttt{item})$) FROM D GROUP BY color

Q_3: SELECT store, SUM ($g(\texttt{item}) \cdot h(\texttt{date})$) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie V_R \bowtie I \bowtie V_O \bowtie H$

V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O



Aggregate Pushdown: Break down query into views [Yannakakis'81]

Similar to **Variable Elimination** and **Message Passing** [Pearl'82]

Factorized Query Evaluation: Example

Q_1: SELECT SUM ($\texttt{units} * c_T$) FROM D

Q_2: SELECT color, SUM ($\texttt{units} \cdot g(\texttt{item})$) FROM D GROUP BY color

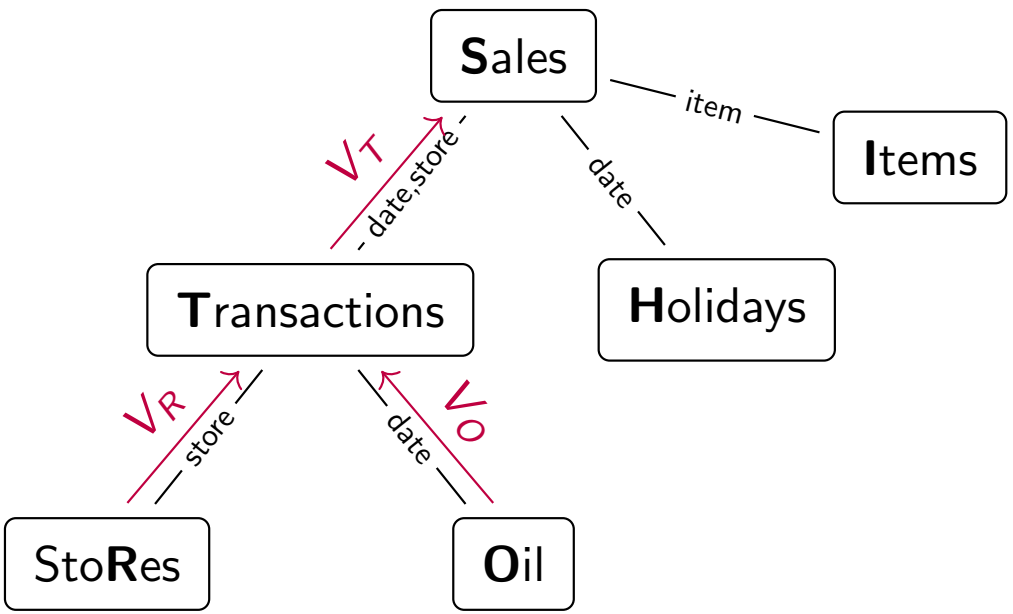
Q_3: SELECT store, SUM ($g(\texttt{item}) \cdot h(\texttt{date})$) FROM D GROUP BY store

... where $D = S \bowtie V_T \bowtie I \bowtie H$

V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O



Aggregate Pushdown: Break down query into views [Yannakakis'81]

Similar to **Variable Elimination** and **Message Passing** [Pearl'82]

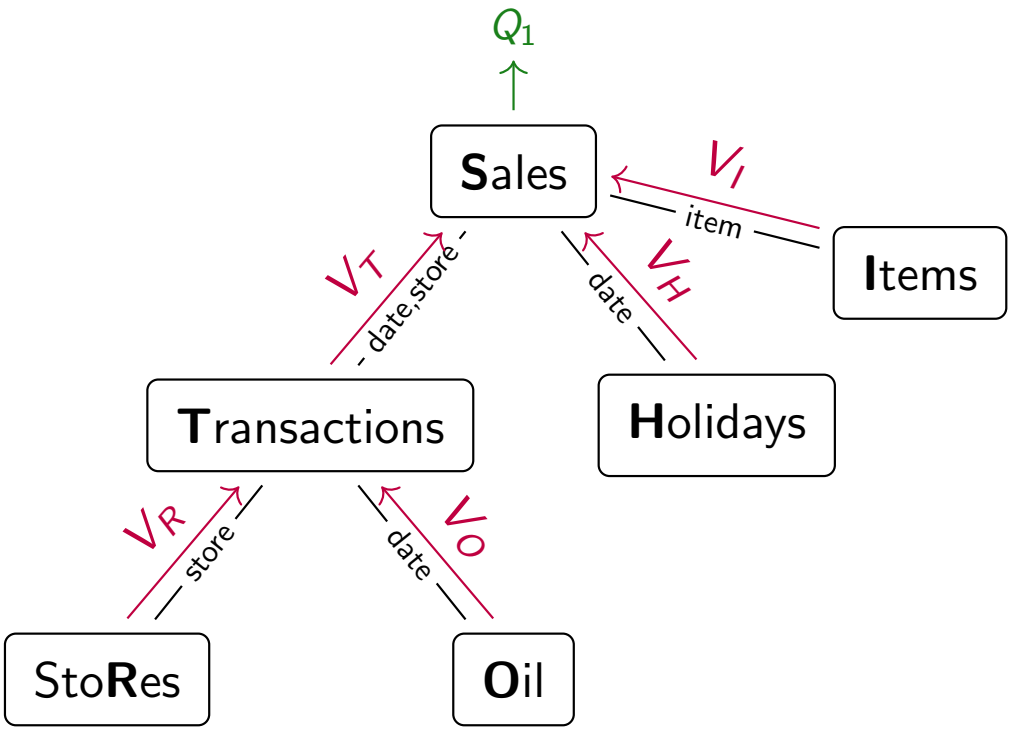
Factorized Query Evaluation: Example

Q_1: SELECT SUM ($\texttt{units} * c_T * c_H * c_I$) FROM D

Q_2: SELECT color, SUM ($\texttt{units} \cdot g(\texttt{item})$) FROM D GROUP BY color

Q_3: SELECT store, SUM ($g(\texttt{item}) \cdot h(\texttt{date})$) FROM D GROUP BY store

... where $D = S \bowtie V_T \bowtie V_I \bowtie V_H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

Aggregate Pushdown: Break down query into views [Yannakakis'81]

Similar to **Variable Elimination** and **Message Passing** [Pearl'82]

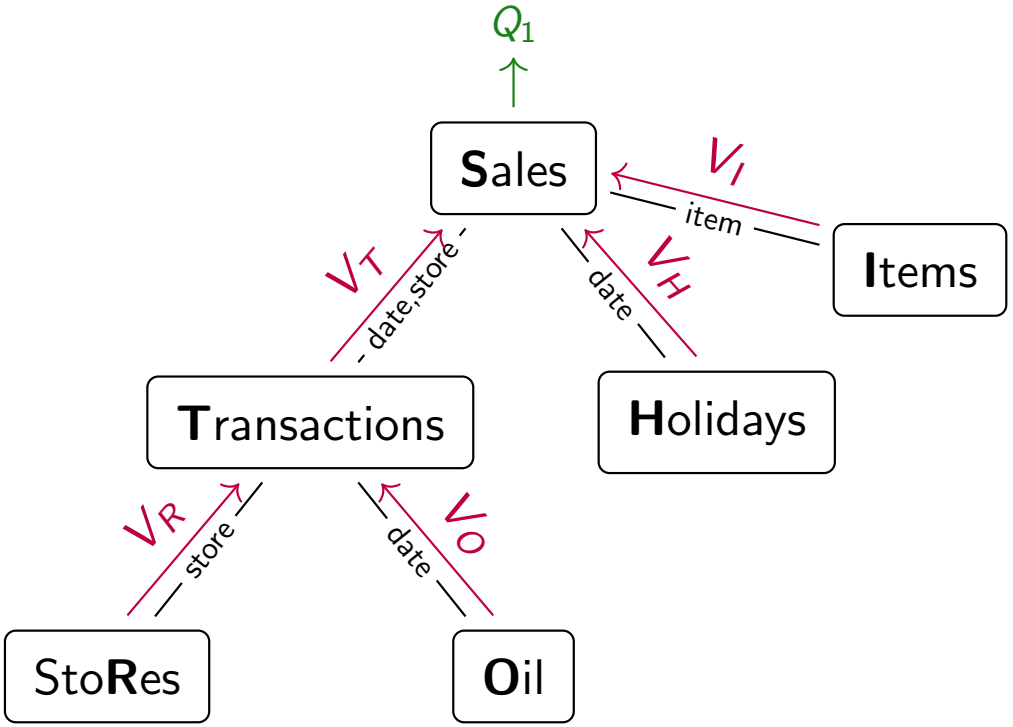
Factorized Query Evaluation: Example

Q_1: `SELECT SUM (units) FROM D`

Q_2: `SELECT color, SUM (units` $\cdot\, g($`item`$)$`) FROM D GROUP BY color`

Q_3: `SELECT store, SUM (`$g($`item`$) \cdot h($`date`$)$`) FROM D GROUP BY store`

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : `store,SUM(1) AS` c_R
`FROM R`

V_O : `date,SUM(1) AS` c_O
`FROM O`

V_T : `store,date,SUM(`$c_R * c_O$`) AS` c_T
`FROM T` $\bowtie$ V_R $\bowtie$ V_O

V_H : `date,SUM(1) AS` c_H
`FROM H`

V_I : `item,SUM(1) AS` c_I
`FROM I`

What if we add Q_2?

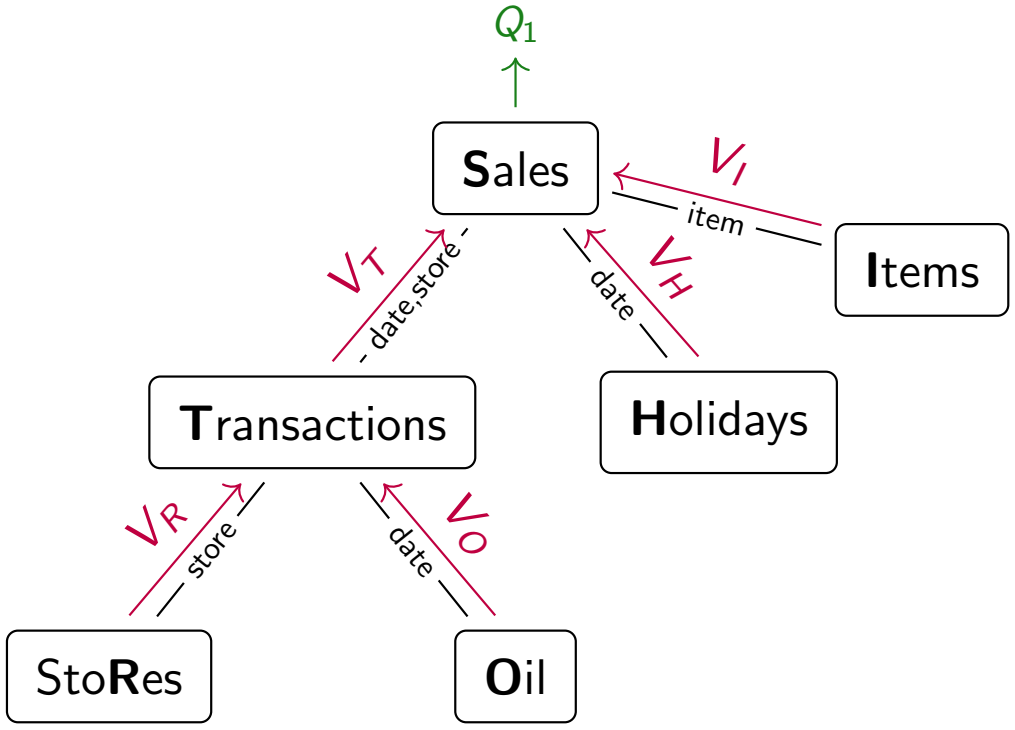
Factorized Query Evaluation: Example

Q_1: SELECT SUM (units) FROM D

Q_2: SELECT color, SUM (units · g(item)) FROM D GROUP BY color

Q_3: SELECT store, SUM (g(item) · h(date)) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

Sharing: Use $V_R, V_O, V_T,$ and V_H to compute Q_1 and Q_2

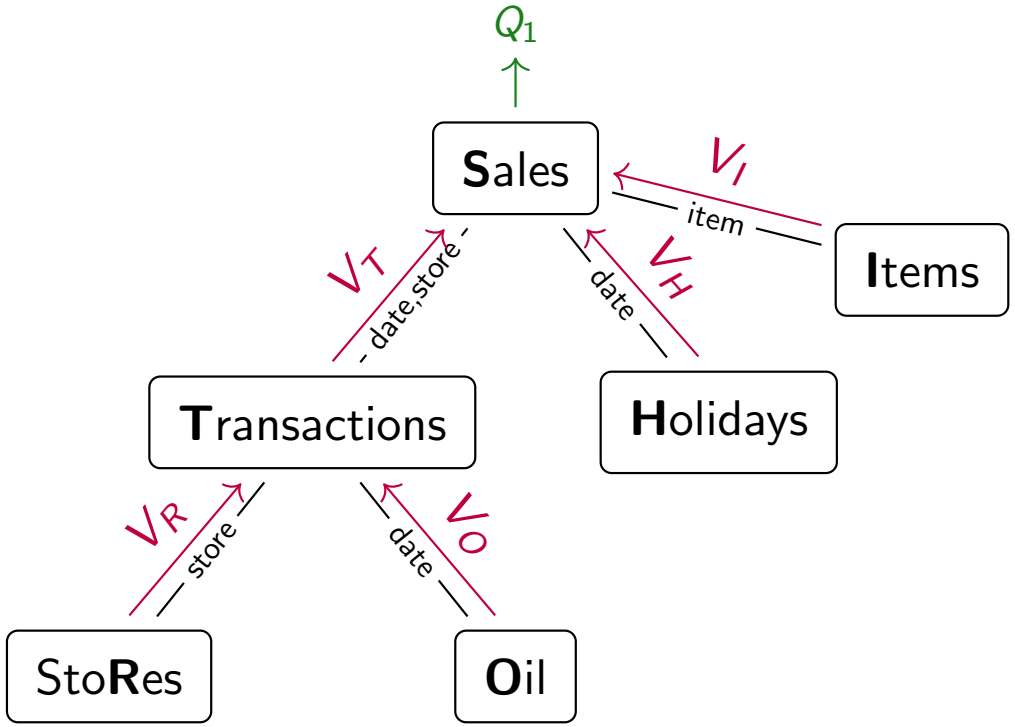
Factorized Query Evaluation: Example

Q_1: SELECT SUM (units) FROM D

Q_2: SELECT color, SUM (units · g(item)) FROM D GROUP BY color

Q_3: SELECT store, SUM (g(item) · h(date)) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

V_I' : item,color,SUM(1) AS c_I'
FROM I

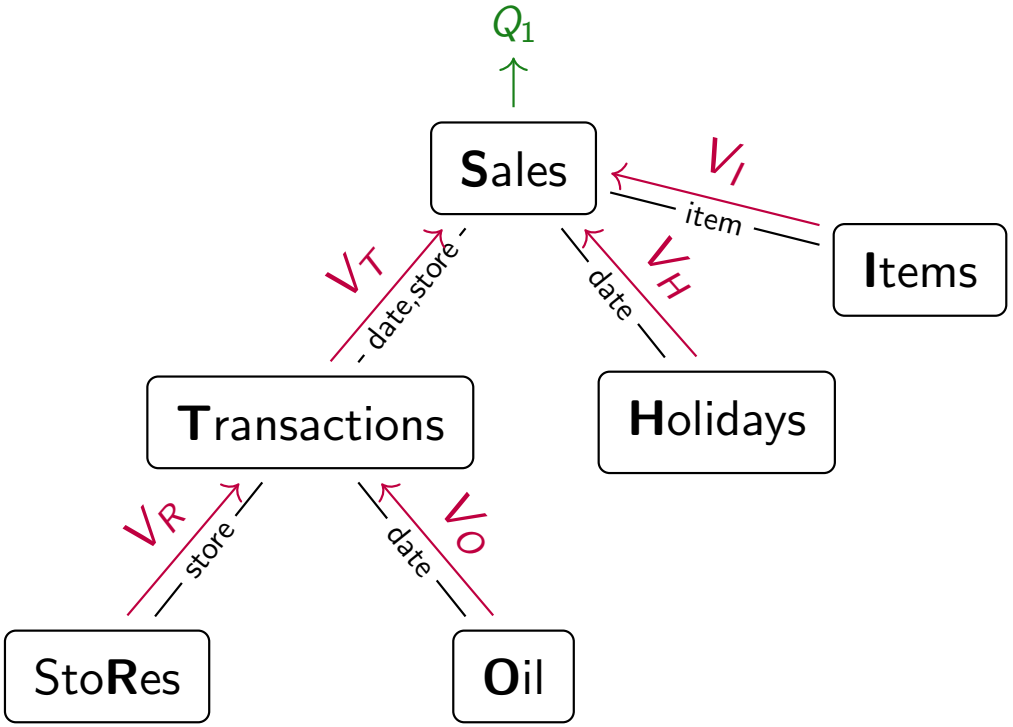
Factorized Query Evaluation: Example

Q_1: SELECT SUM (units) FROM D

Q_2: SELECT color, SUM (units · g(item)) FROM D GROUP BY color

Q_3: SELECT store, SUM (g(item) · h(date)) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

V_I' : item,color,SUM(1) AS c_I'
FROM I

Problem: Extra group-by attribute color increases the size of V_I'!

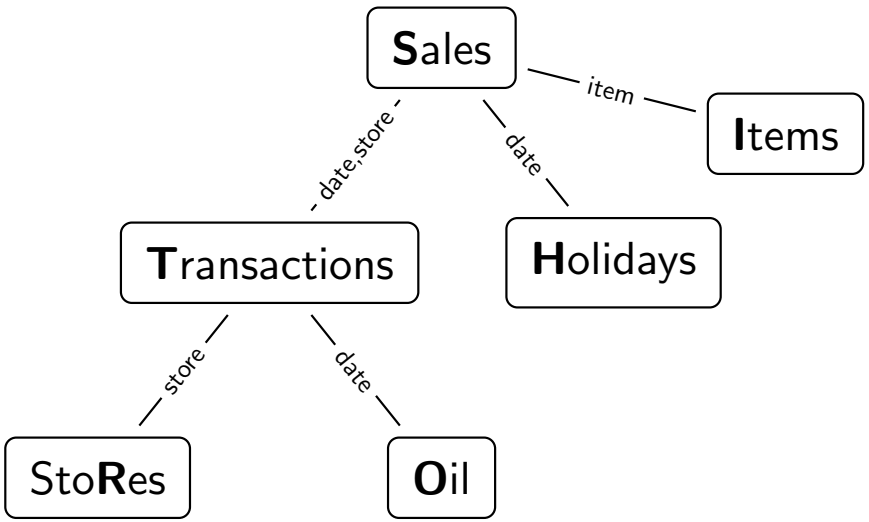
How To Compute Multiple Aggregate Queries?

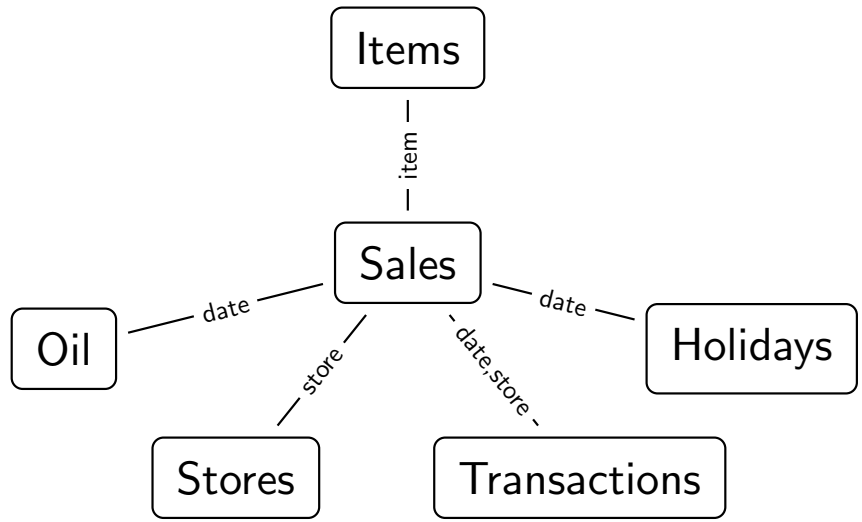
The Challenge:

Compute each aggregate so that the views are small

The current approach:

Use best join tree for each query





How To Compute Multiple Aggregate Queries?

The Challenge:

Compute each aggregate so that the views are small

The current approach:

Use best join tree for each query

In practice:

Tradeoff between redundant computation and view size

How To Compute Multiple Aggregate Queries?

The Challenge:

Compute each aggregate so that the views are small

The current approach:

Use best join tree for each query

In practice:

Tradeoff between redundant computation and view size

Our approach:

1. Fix join tree
2. Choose root where to output the query

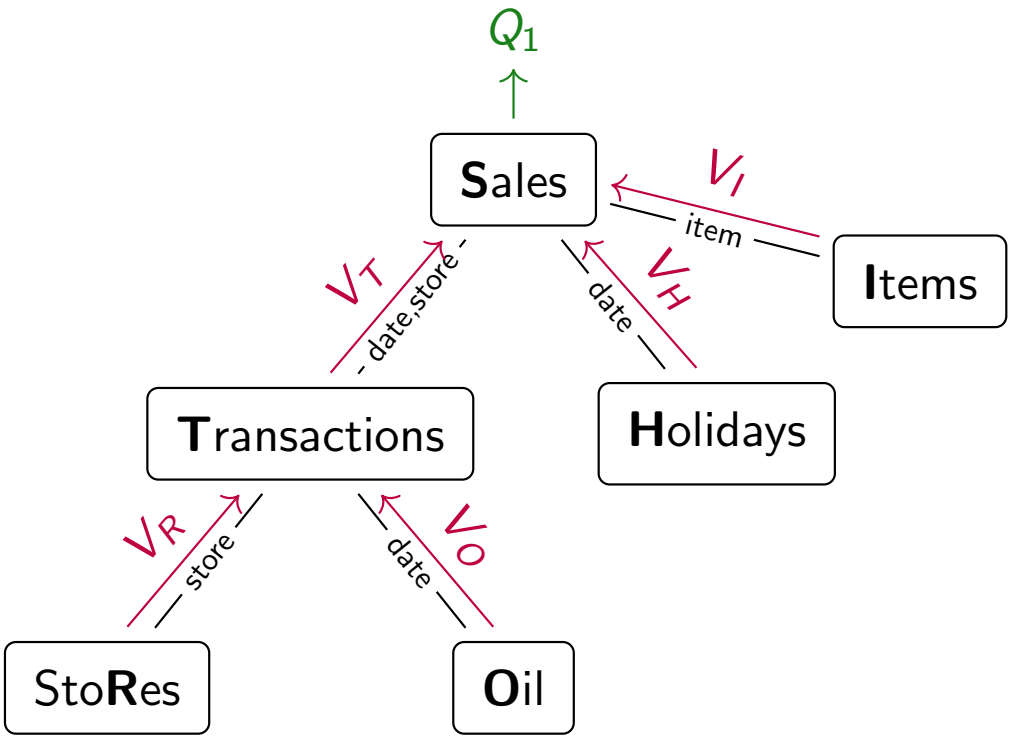
Factorized Query Evaluation: Multiple Aggregates

Q_1: SELECT SUM (units) FROM D

Q_2: SELECT color, SUM (units · g(item)) FROM D GROUP BY color

Q_3: SELECT store, SUM (g(item) · h(date)) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

V'_I : item,color,SUM(1) AS c'_I
FROM I

Our Solution: Choose Items as **root** for Q_2.

Compute V_S instead of V'_I $\Rightarrow$ no extra group-by attribute

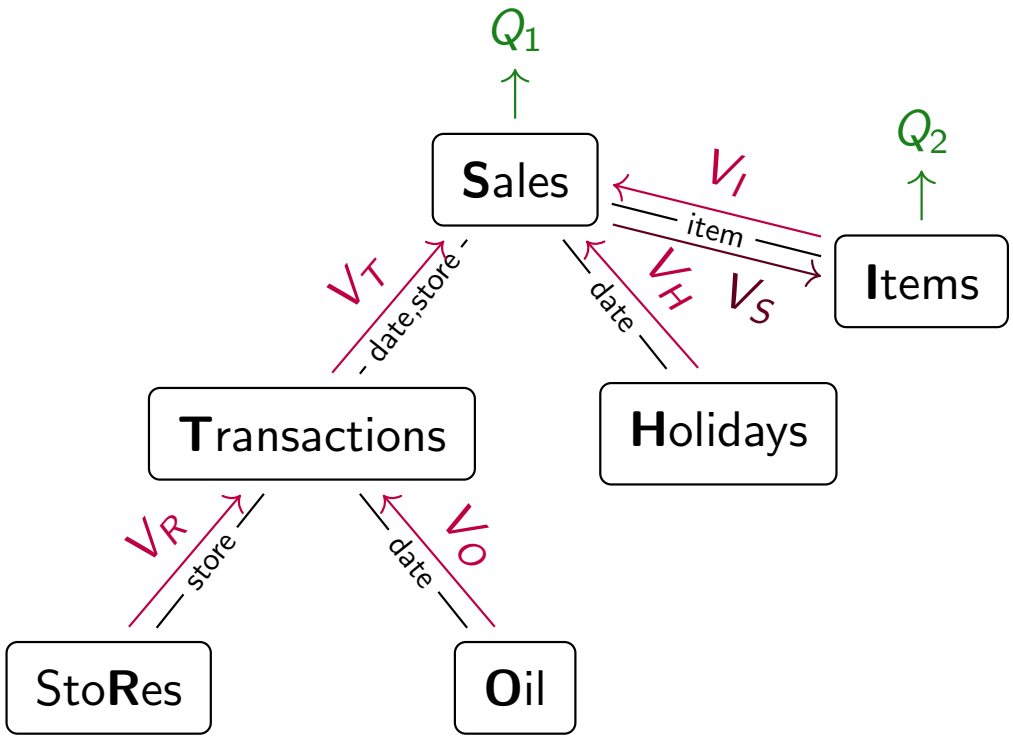
Factorized Query Evaluation: Multiple Aggregates

Q_1: `SELECT SUM (units)` `FROM D`

Q_2: `SELECT color, SUM (units` $\cdot\, g($`item`$)$`)` `FROM D GROUP BY color`

Q_3: `SELECT store, SUM (`$g($`item`$) \cdot h($`date`$)$`)` `FROM D GROUP BY store`

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : `store,SUM(1) AS` c_R
`FROM R`

V_O : `date,SUM(1) AS` c_O
`FROM O`

V_T : `store,date,SUM(`$c_R * c_O$`) AS` c_T
`FROM T` $\bowtie$ V_R $\bowtie$ V_O

V_H : `date,SUM(1) AS` c_H
`FROM H`

V_I : `item,SUM(1) AS` c_I
`FROM I`

V_S : `item,SUM(units*`$c_T * c_H$`) AS` c_S
`FROM S` $\bowtie$ V_T $\bowtie$ V_H

Our Solution: Choose Items as **root** for Q_2.

Compute V_S instead of V'_I $\Rightarrow$ no extra group-by attribute

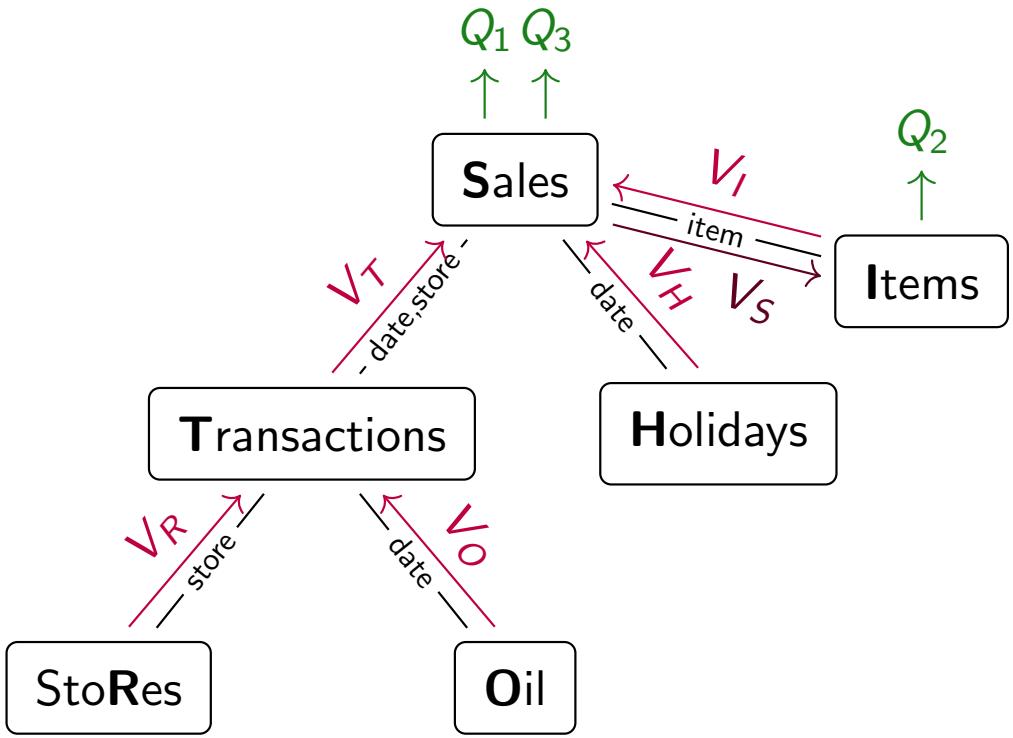
Factorized Query Evaluation: Multiple Aggregates

Q_1: SELECT SUM (units) FROM D

Q_2: SELECT color, SUM (units $\cdot$ g(item)) FROM D GROUP BY color

Q_3: SELECT store, SUM (g(item) $\cdot$ h(date)) FROM D GROUP BY store

... where $D = S \bowtie T \bowtie R \bowtie I \bowtie O \bowtie H$



V_R : store,SUM(1) AS c_R
FROM R

V_O : date,SUM(1) AS c_O
FROM O

V_T : store,date,SUM($c_R * c_O$) AS c_T
FROM T $\bowtie$ V_R $\bowtie$ V_O

V_H : date,SUM(1) AS c_H
FROM H

V_I : item,SUM(1) AS c_I
FROM I

V_S : item,SUM(units$* c_T * c_H$) AS c_S
FROM S $\bowtie$ V_T $\bowtie$ V_H

Sharing: Q_3 reuses views defined for Q_1

How To Achieve This Performance Improvement?

Idea 1: Turn Data Intensive ML Computation into DB Problem

Idea 2: Exploit Problem Structure to Lower Complexity

Idea 3: **Use DB Optimizations to Lower the Constant Factors**

Engineering Tools Of A Database Researcher

1. Sharing low-level data access
    - Share data access across views
2. Specialisation for workload and data
    - Generate code specific to the query batch and dataset
    - Improve cache locality for hot data path
3. Parallelisation:
    - Task and domain multicore parallelism

[DEEM'18, SIGMOD'19, CGO'20]

Structure-Aware versus Structure-Agnostic Learning

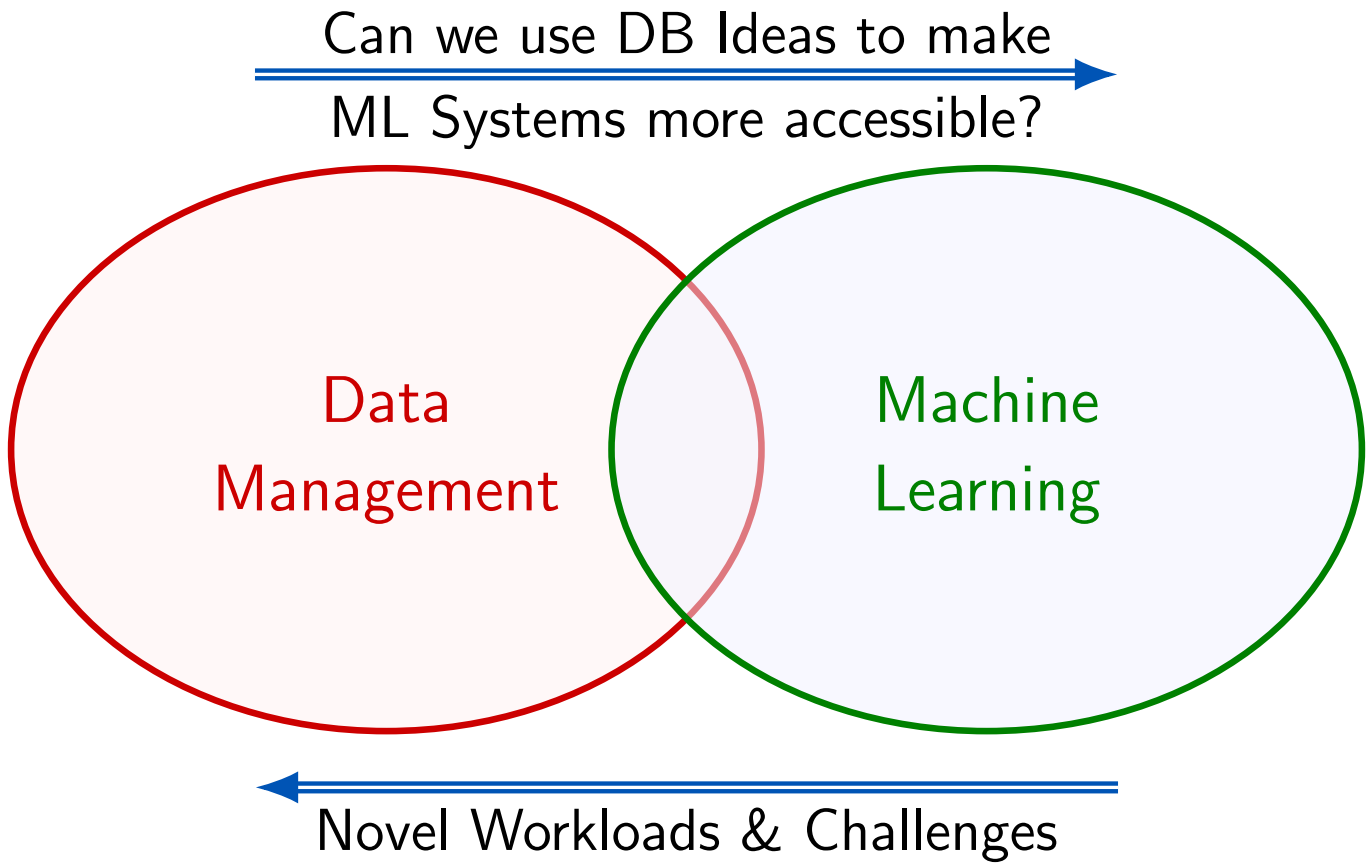
Train linear regression model to predict *inventory* given all features

	PostgreSQL+TensorFlow		Structure Aware	
	Time	**Size**	**Time**	**Size**
Database	–	2.1 GB	–	2.1 GB
Join	152.06 secs	23 GB	–	–
Export	351.76 secs	23 GB	–	–
Shuffling	5,488.73 secs	23 GB	–	–
Query batch	–	–	6.08 secs	37 KB
Grad Descent	7,249.58 secs	–	0.05 secs	–
Total time	13,242.13 secs		6.13 secs	

Intel i7-4770, 3.4GHz, 32GB, 8 cores [SIGMOD'19]

TensorFlow: 1 epoch with 100K tuple batch **Size:** File Size in CSV

Research Conjecture

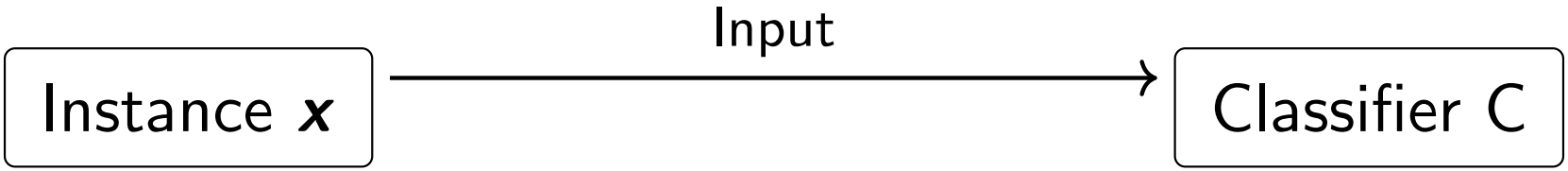


This talk:

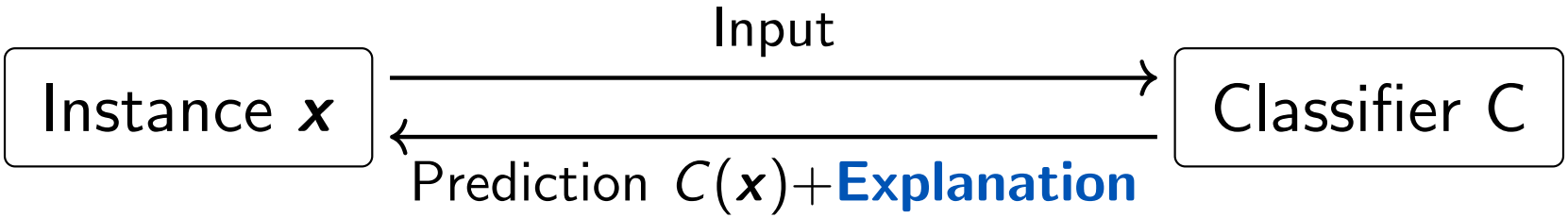
Part 1: Learning over Relational Databases

Part 2: Explaining Prediction Outcomes

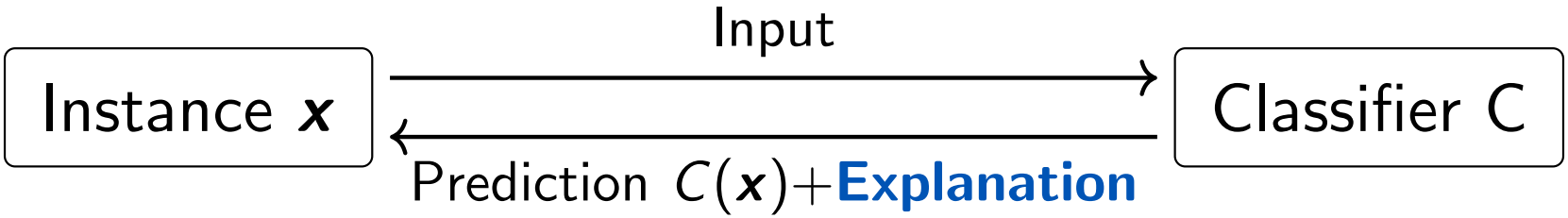
Counterfactual Explanations by Example



Counterfactual Explanations by Example



Counterfactual Explanations by Example

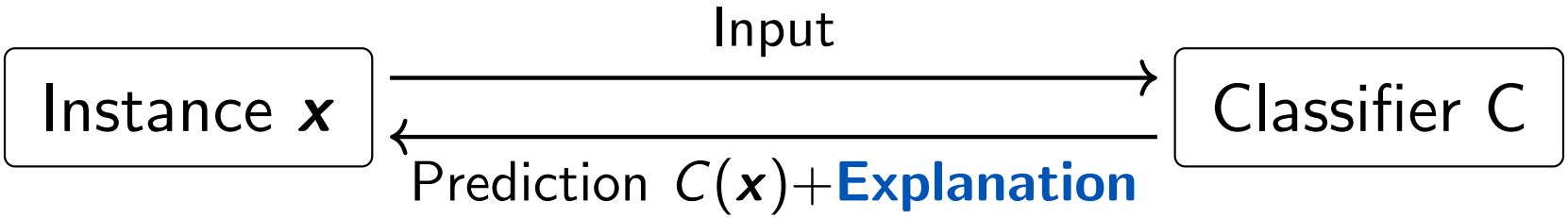


Factual

Age	Income	Debt	Accounts
28	800	200	5

C() = no loan

Counterfactual Explanations by Example



Factual

Age	Income	Debt	Accounts
28	800	200	5

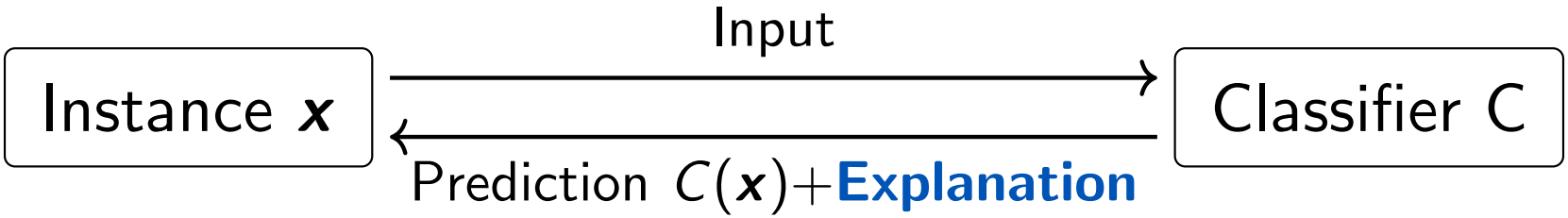
C() = no loan

Counterfactual

Age	Income	Debt	Accounts
28	1000	200	3

C() = loan

Counterfactual Explanations by Example



Factual

Age	Income	Debt	Accounts
28	800	200	5

C() = no loan

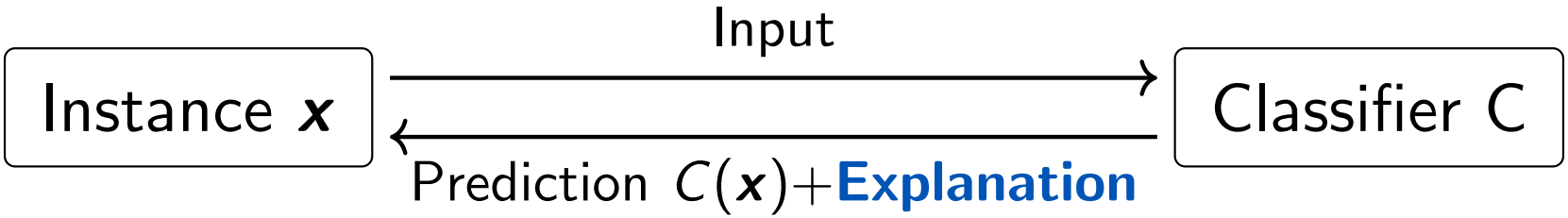
Counterfactual

Age	Income	Debt	Accounts
28	1000	200	3

C() = loan

Explanation: Your loan will be approved, if you increase income by \$200 and close two accounts.

Counterfactual Explanations by Example



Factual

Age	Income	Debt	Accounts
28	800	200	5

C( ) = no loan

Counterfactual

Age	Income	Debt	Accounts
28	1000	200	3

C( ) = loan

Explanation: Your loan will be approved, if you increase income by \$200 and close two accounts.

GDPR *right to explanation* compliant [Wachter et.al.'17]

Computing Counterfactual Explanations

Optimization Problem:

Find *nearest* counterfactual that $\arg\min_{\boldsymbol{x}_{cf}} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_{cf})$

Computing Counterfactual Explanations

Optimization Problem:

Find *nearest* counterfactual that

1. achieves the desired outcome

$$\arg\min_{\boldsymbol{x}_{cf}} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_{cf})$$
$$\text{s.t. } C(\boldsymbol{x}_{cf}) = \text{“good”}$$

Computing Counterfactual Explanations

Optimization Problem:

Find *nearest* counterfactual that

1. achieves the desired outcome
2. is realistic

$$\begin{aligned} &\arg\min_{\boldsymbol{x}_{cf}} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_{cf}) \\ &\text{s.t. } C(\boldsymbol{x}_{cf}) = \text{“good”} \\ &\quad \boldsymbol{x}_{cf} \text{ is plausible} \\ &\quad \boldsymbol{x}_{cf} \text{ is feasible} \end{aligned}$$

Computing Counterfactual Explanations

Optimization Problem:

Find *nearest* counterfactual that

1. achieves the desired outcome
2. is realistic

$$\begin{aligned} &\arg\min_{\boldsymbol{x}_{cf}} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_{cf}) \\ &\text{s.t. } C(\boldsymbol{x}_{cf}) = \text{“good”} \\ &\quad \boldsymbol{x}_{cf} \text{ is plausible} \\ &\quad \boldsymbol{x}_{cf} \text{ is feasible} \end{aligned}$$

Challenge:

- Ensuring explanations are realistic
- Huge search space

Computing Counterfactual Explanations

Optimization Problem:

Find *nearest* counterfactual that

1. achieves the desired outcome
2. is realistic

$$\begin{aligned} &\arg\min_{\boldsymbol{x}_{cf}} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_{cf}) \\ &\text{s.t. } C(\boldsymbol{x}_{cf}) = \text{“good”} \\ &\quad \boldsymbol{x}_{cf} \text{ is plausible} \\ &\quad \boldsymbol{x}_{cf} \text{ is feasible} \end{aligned}$$

Challenge:

- Ensuring explanations are realistic
- Huge search space

GeCo: Quality Counterfactual Explanations in Real Time

Under Revision. arXiv abs.2101.01292. 2020.

Constraint Language for Plausibility and Feasibility

Correlated features:

```
GROUP education, income
GROUP city, state, country
```

Constraints on search space:

```
PLAF cf.gender = x.gender
PLAF cf.age >= x.age
PLAF IF cf.education > x.education
     THEN cf.age > x.age+4
```

Constraint Language for Plausibility and Feasibility

At Runtime:

Compute Feasible Sample Space for each Feature Group

Assume:

1. Dataset D
2. Group {`education,income`}
3. Constraint `cf.income <= 200K`

Feasible Sample Space:

```
SELECT education,income
FROM D
WHERE income <= 200K
```

Generating Counterfactuals

Custom Genetic Algorithm

- Metaheuristic based on process of natural selection

Benefits:

- Supports PLAF constraints
- No restrictions on classifier and data $\Rightarrow$ Black-Box Classifier
- Returns multiple explanations

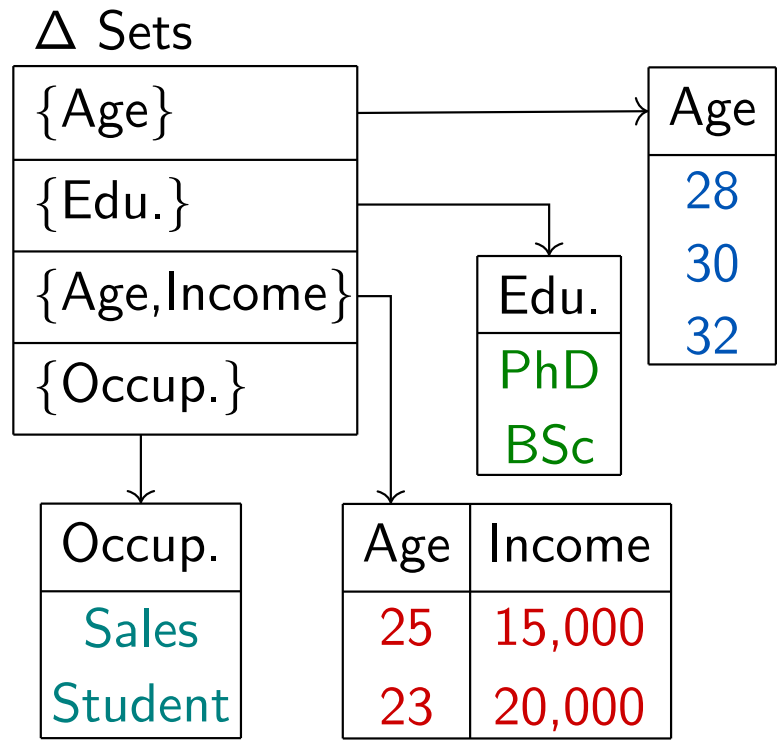
GeCo prioritizes counterfactuals that change only few features

Δ-Representation

Candidate counterfactuals for $\boldsymbol{x}$ = (22, HS, Service, 10,000)

Age	Edu.	Occup.	Income
28	HS	Service	10,000
30	HS	Service	10,000
32	HS	Service	10,000
22	PhD	Service	10,000
22	BSc	Service	10,000
25	HS	Service	15,000
23	HS	Service	20,000
22	HS	Sales	10,000
22	HS	Student	10,000

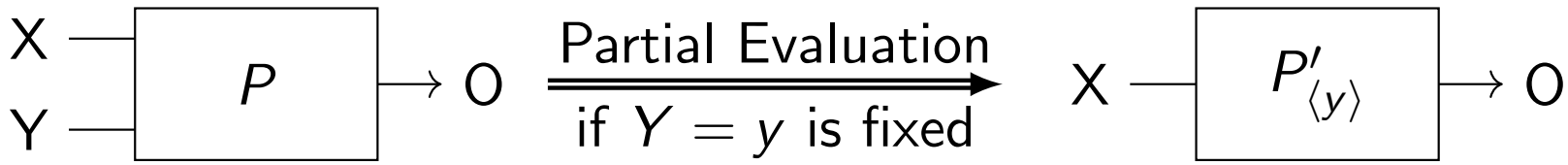
Naive Listing Representation



Compressed Δ-Representation

Specialization of Classifier via Partial Evaluation

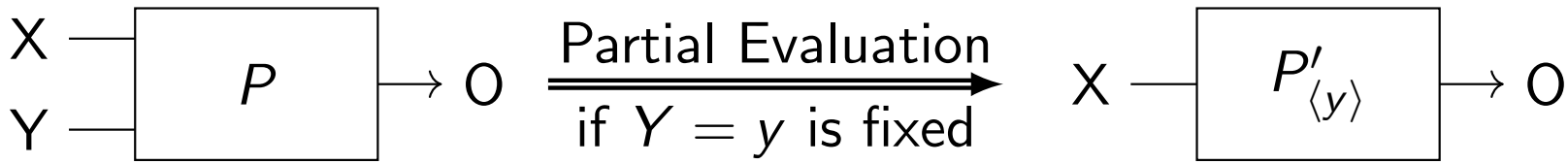
Partial Evaluation in a Nutshell:



Guarantee: $P(x, y) = P'_{\langle y \rangle}(x) \quad \forall x \in \mathrm{dom}(X)$

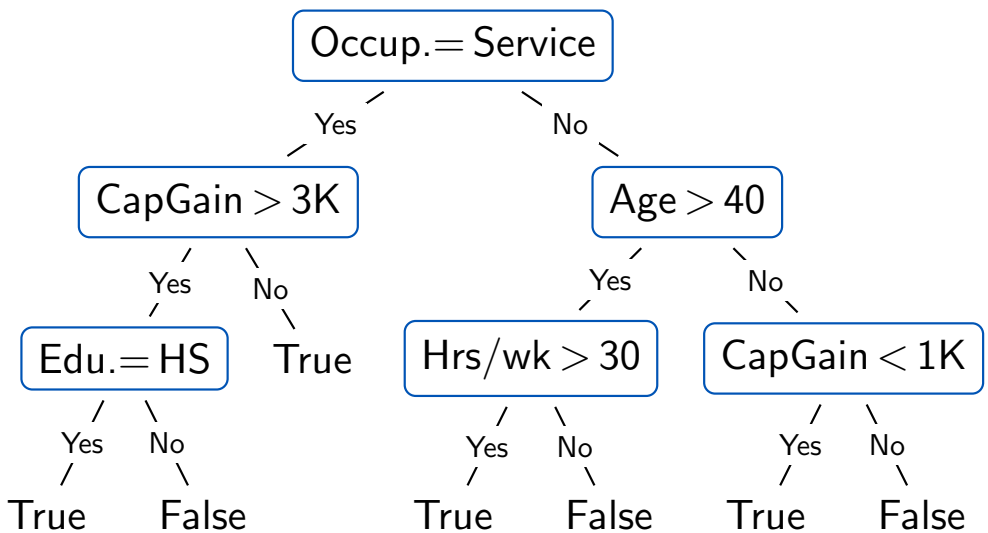
Specialization of Classifier via Partial Evaluation

Partial Evaluation in a Nutshell:



Guarantee: $P(x, y) = P'_{\langle y \rangle}(x) \quad \forall x \in \mathrm{dom}(X)$

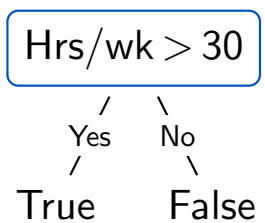
Decision Tree Example:



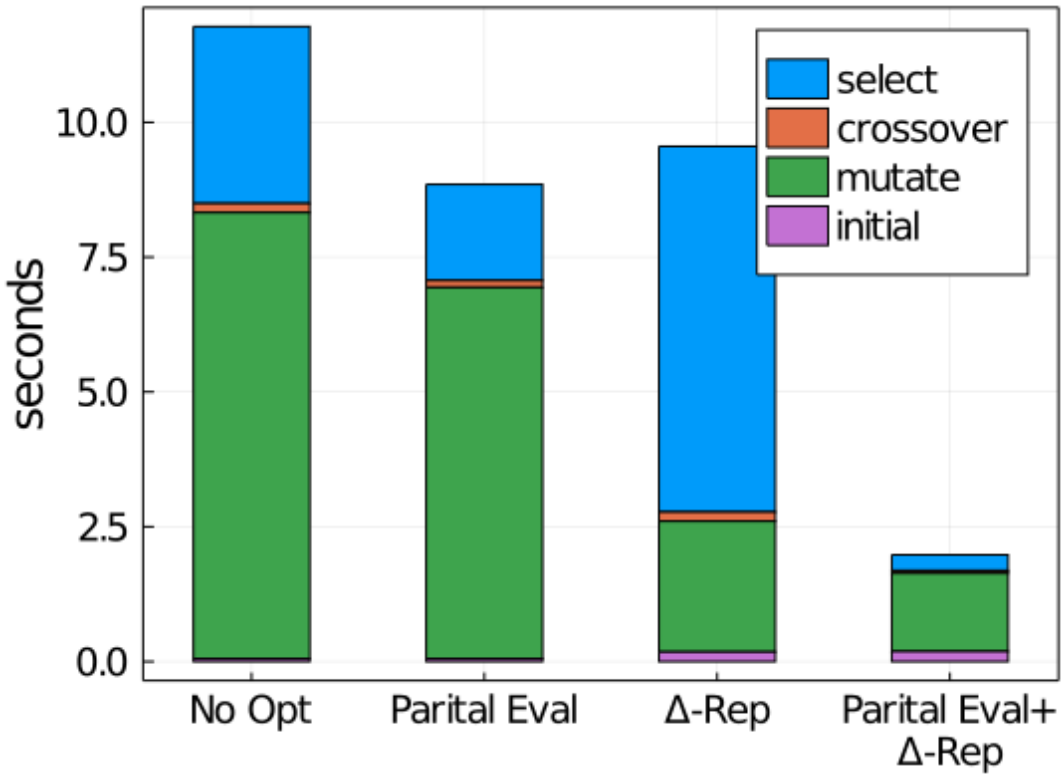
If we know:

Occup. = Blue Collar, Age = 43

Specialized Classifier:



Does it pay off?



Combining the Δ-Representation with Partial Evaluation of the Classifier leads to 5× speedup.

Yelp Dataset, Random Forest Classifier

Experiments on UCI Adults Dataset

Classifier: Predict whether Income >50K

Comparing GeCo with MACE [Karimi et.al.'20]

- Generates counterfactuals via multiple runs of SMT-solver

Factual $\boldsymbol{x}$

Age	Educ.	Occup.	CapGain	CapLoss	Hrs/wk
49	School	Service	0.0	0.0	16

$C(\boldsymbol{x}) = \text{False}$

GeCo $\boldsymbol{x}_{cf}$

Age	Educ.	Occup.	CapGain	CapLoss	Hrs/wk
49	School	Service	4,687.0	0.0	16

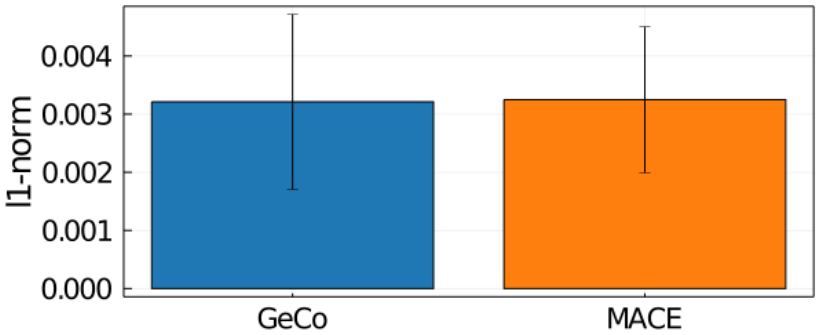
$C(\boldsymbol{x}_{cf}) = \text{True}$

MACE $\boldsymbol{x}_{cf}$

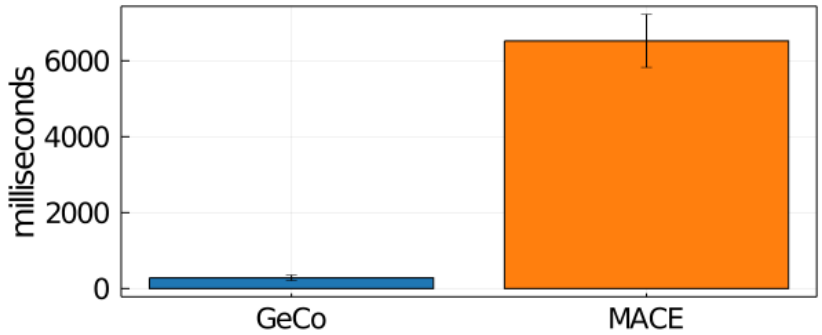
Age	Educ.	Occup.	CapGain	CapLoss	Hrs/wk
49	School	Service	4,826.0	19.9	16

$C(\boldsymbol{x}_{cf}) = \text{True}$

Distance



Runtime/Explanation



Experiments on UCI Adults Dataset

Classifier: Predict whether Income >50K

We compare GeCo with DiCE [Mahajan et.al.'20]

- Generates counterfactuals with deep VAE
- Very efficient $\Rightarrow$ 5$\times$ faster than GeCo

Factual

$\boldsymbol{x}$

Age	Educ.	Occup.	Hrs/wk	Race	Gender
49	School	Service	16	Other	Female

$C(\boldsymbol{x}) =$ False

GeCo

$\boldsymbol{x}_{cf}$

Age	Educ.	Occup.	Hrs/wk	Race	Gender
49	MSc	Service	16	Other	Female

$C(\boldsymbol{x}_{cf}) =$ True

DiCE

$\boldsymbol{x}_{cf}$

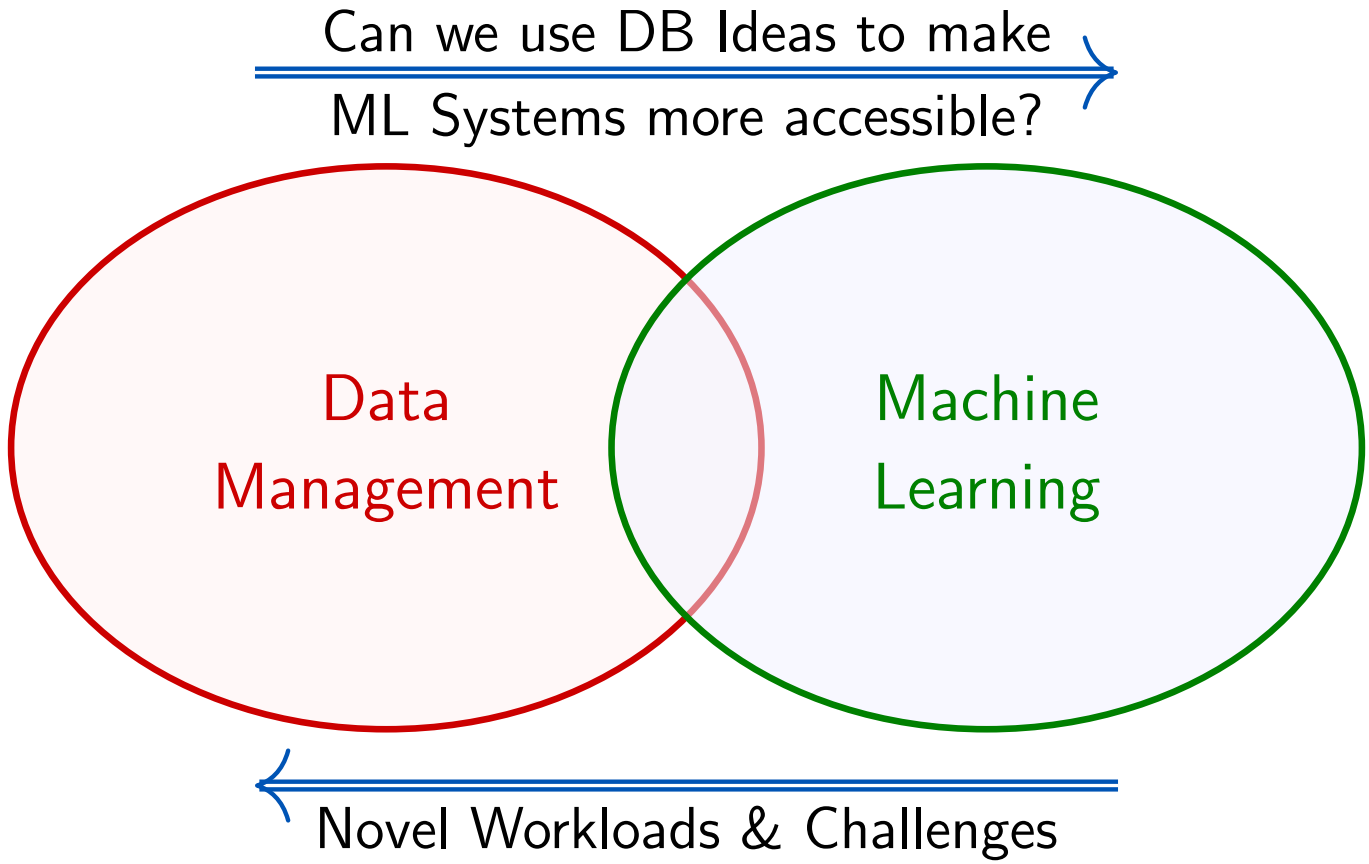
Age	Educ.	Occup.	Hrs/wk	Race	Gender
54	PhD	BlueCol	40	White	Male

$C(\boldsymbol{x}_{cf}) =$ True

Similar results for Google What-if Tool [Wexler et.al.'19]

GeCo provides quality counterfactual explanations in real time.

Research Plan

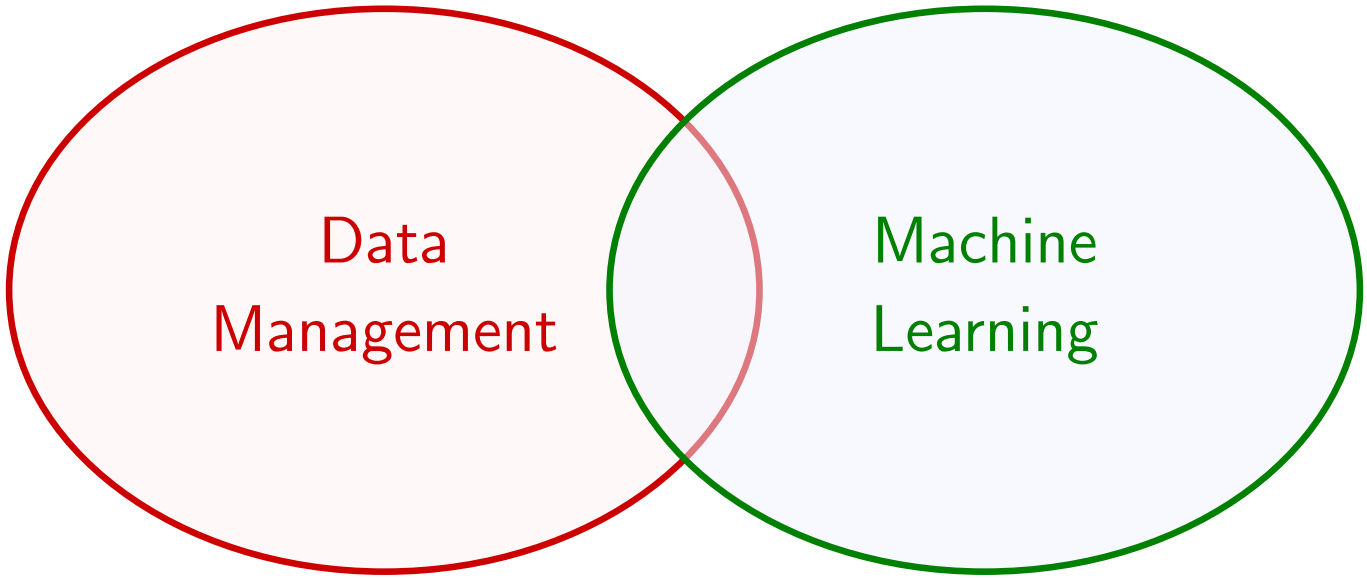


This talk:

Part 1: Learning over Relational Databases

Part 2: Explaining Prediction Outcomes

Future Research Plan

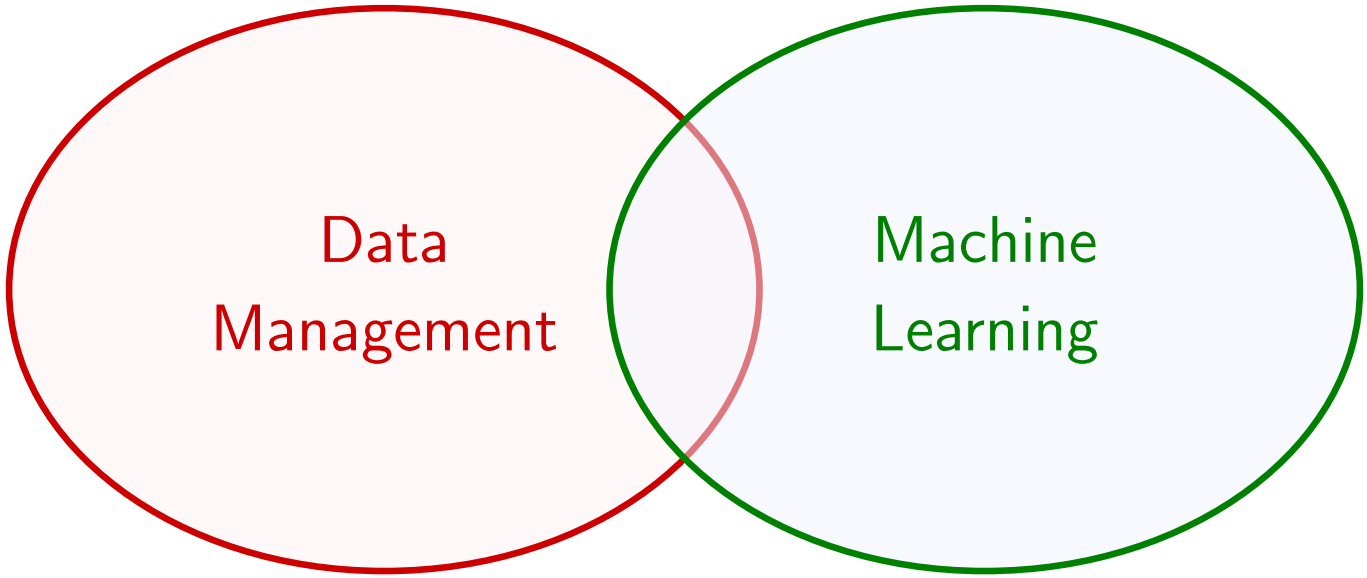


Long-Term Goal:

Data Management System for the Age of AI

- Natively support query processing and machine learning
- Grounded on a strong theoretical foundation

Future Research Plan



Interested in research at the interface of DB and ML?

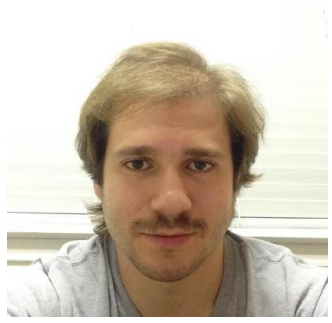
Get in Touch!

Email: schleich@cs.washington.edu
Twitter: @mjschleich

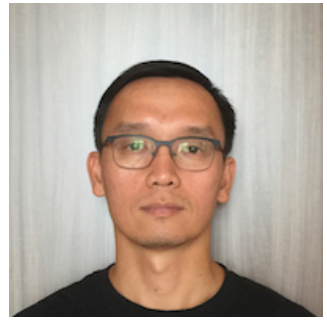
Acknowledgments

Dan O.
Zurich/Oxford

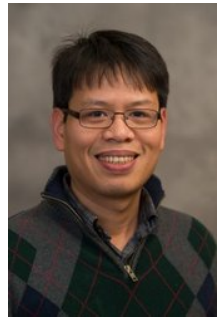
Dan S.
Washington



Mahmoud
RelationalAI



Hung
RelationalAI



Long
Michigan

Amir
Edinburgh

Guy
UCLA

Leo
UAI

Ben
CMU

Ryan
RelationalAI

Appendix

My Research on Structure-Aware Learning

Systems:

- A Layered Aggregate Engine for Analytics Workloads.
  ACM SIGMOD, pages 1642–1659, 2019.
- Rk-means: Fast Clustering for Relational Data.
  AISTATS, pages 2742-2752, 2020.
- Multi-layer optimizations for end-to-end data analytics.
  CGO, pages 145–157, 2020.
- AC/DC: In-Database Learning Thunderstruck.
  ACM SIGMOD DEEM Workshop, pages 1–10, 2018.
- Learning linear regression models over factorized joins.
  ACM SIGMOD, pages 3–18, 2016.
- Factorized databases.
  SIGMOD Records, 45(2):5–16, 2016.

My Research on Structure-Aware Learning

Theory:

- Functional Aggregate Queries with Additive Inequalities
  ACM TODS, 45(4), pages 1-41, 2020.

  Invited as Best of ACM PODS 2019.

- Learning Models over Relational Data using Sparse Tensors and Functional Dependencies.

  ACM TODS, 45(2), pages 1-66, 2020.

  Invited as Best of ACM PODS 2018.

My Research on Explainable AI

On Counterfactual Explanations:

- GeCo: Quality Counterfactual Explanations in Real Time
  Under Revision. arXiv abs.2101.01292. 2020.

On Feature Attribution Scores:

- On the Tractability of SHAP Explanations.
  AAAI, 2021. **Distinguished Paper Award**
- Causality-based Explanation of Classification Outcomes.
  ACM SIGMOD DEEM Workshop, 2020.

References - Part 1

- [NPRR'12]
  Hung Q. Ngo, Ely Porat, Christopher Re, and Atri Rudra. **Worst Case Optimal Join Algorithms**. ACM PODS, 2012

- [Veldhuizen'14]
  Todd L. Veldhuizen. **Leapfrog Triejoin: A Simple, Worst-Case Optimal Join Algorithm**. ICDT, 2014.

- [NRR'13]
  Hung Q. Ngo, Christopher Re, and Atri Rudra. **Skew Strikes Back: New Developments in the Theory of Join Algorithms**. SIGMOD Records, vol. 42 (4), 2013.

- [Yannakakis'81]
  M. Yannakakis. **Algorithms for acyclic database schemes.** PVLDB, pages 82–94, 1981.

- [Pearl'82]
  Judea Pearl. **Reverend Bayes on inference engines: A distributed hierarchical approach**. AAAI, 1982.

References - Part 2

- [Wachter et.al.'17]
  Sandra Wachter, Brent Mittelstadt, and Chris Russell. **Counterfactual explanations without opening the black box: Automated decisions and the GDPR.** Harv. JL & Tech., 31:841, 2017.

- [Mahajan et.al.'20]
  Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. **Model-agnostic counterfactual explanations for consequential decisions.** In AISTATS, 895–905, 2020.

- [Mahajan et.al.'20]
  Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2019. **Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers.** CausalML @ NeurIPS Workshop.

- [Wexler et.al.'19]
  James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. **The what-if tool: Interactive probing of machine learning models.** IEEE transactions on visualization and computer graphics 26, 1, 56–65, 2019.