Efficient, Effective Interactive Visualizations

eugene wu

Columbia University

This presentation contains some animation.

See the animation in the powerpoint file at:

https://cudbg.github.io/sigmod19tutorial//files/wu.pptx

More Great Tutorials/Workshops!

Evaluating Interactive Data Systems: Workloads, Metrics, & Guidelines.

SIGMOD18. Jiang, Rahman, Nandi

Overview of Data Exploration Techniques.

SIGMOD15. Idreos, Papaemmanouil, Chaudhuri

HILDA: Human in the Loop Data Analysis

SIGMOD Workshop. http://hilda.io

DSIA: Data Systems for Interactive Analysis

VIS Workshop. https://www.interactive-analysis.org

Connecting Visualization and Data Management Research

Dagstuhl Workshop. Chang, Fekete, Freire, Scheidegger.

Road Map

Background

Mechanisms

A Relational Story

Visualization is Ubiquitous

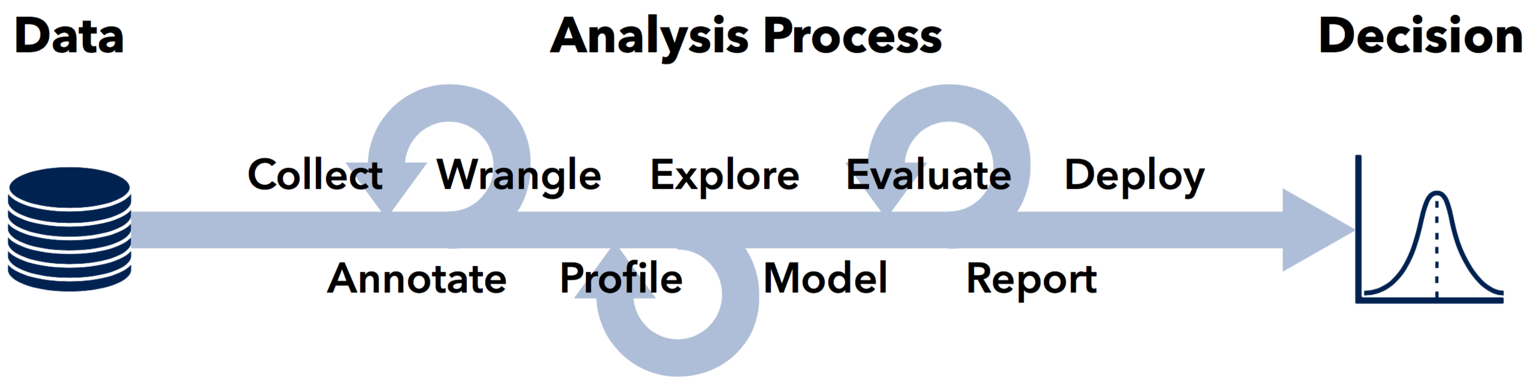
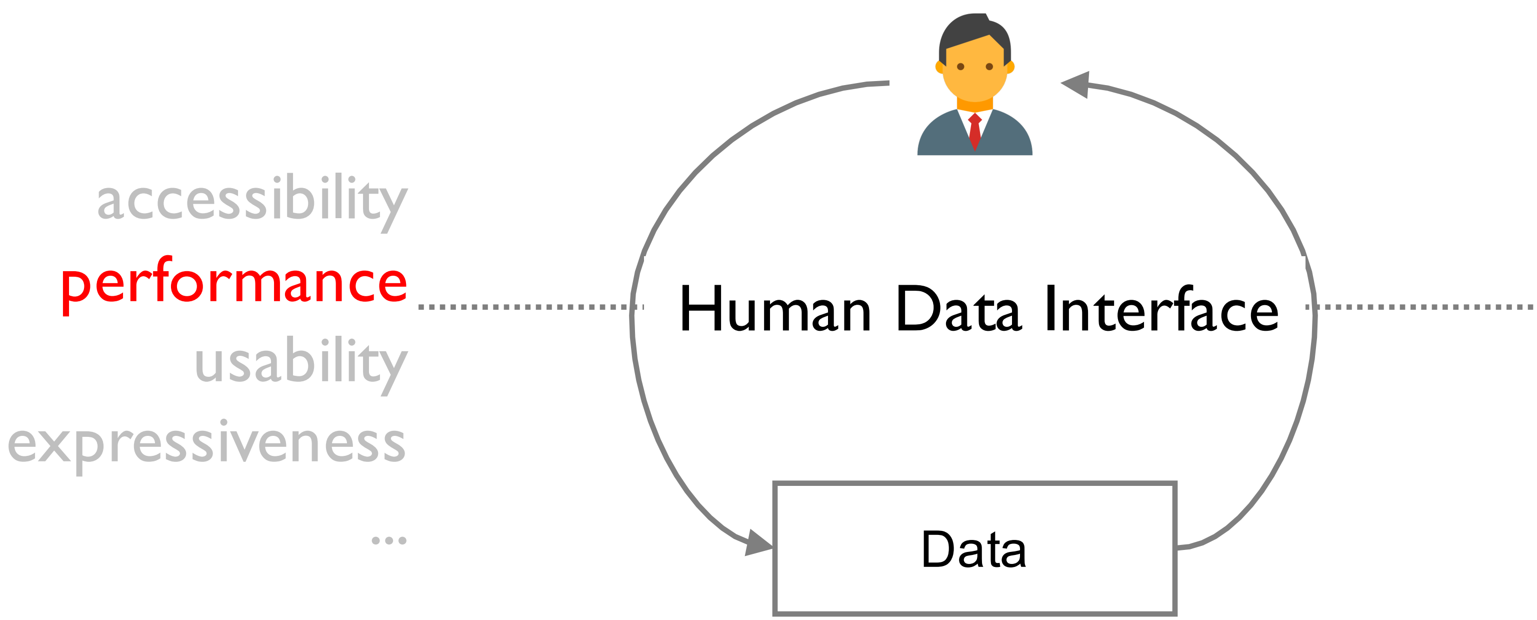
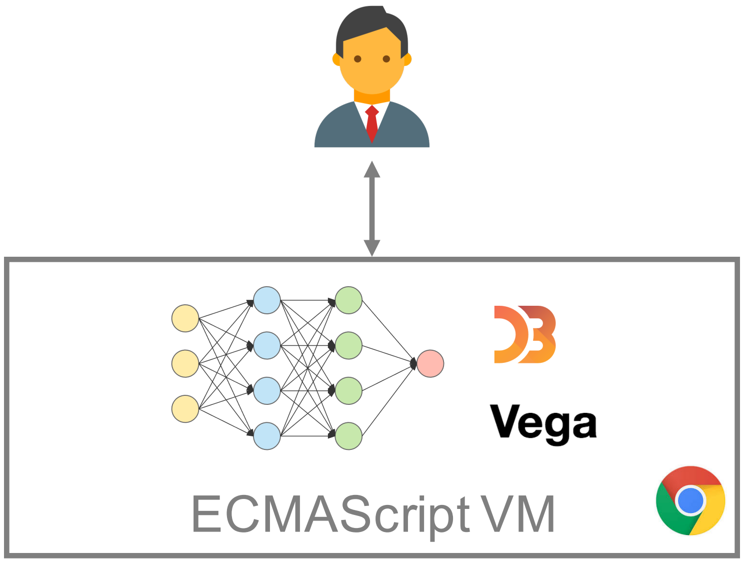


Image: Visualization is Not Enough - Heer EuroVIS19



Remove computing bottleneck

Tutorial: focus on performance



SVD architecture

SQL,Vis,DB

Client-side vis libraries very robust

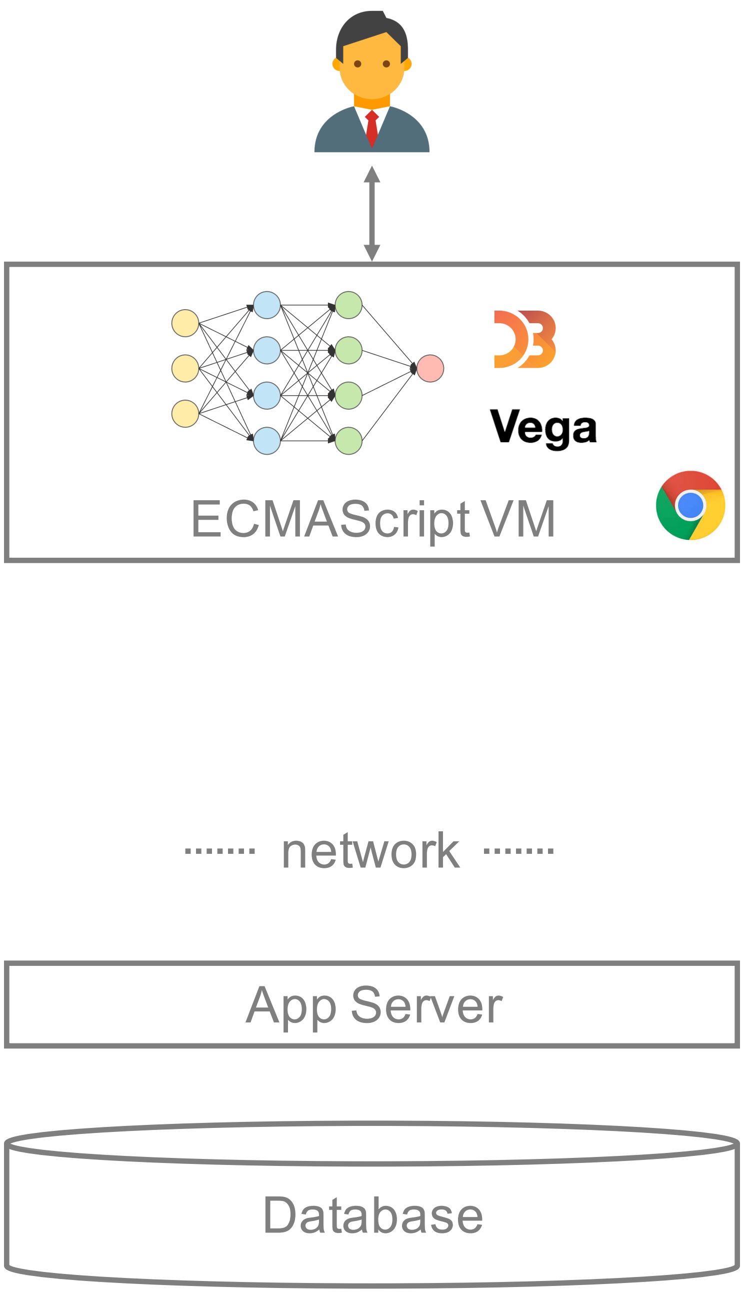
Borrows data flow, event processing, incremental updates from databases.

Outputs scene graph

Great for small datasets

SVD architecture

SQL,Vis,DB



Client-side vis libraries very robust

Borrows data flow, event processing, incremental updates from databases.

Outputs scene graph

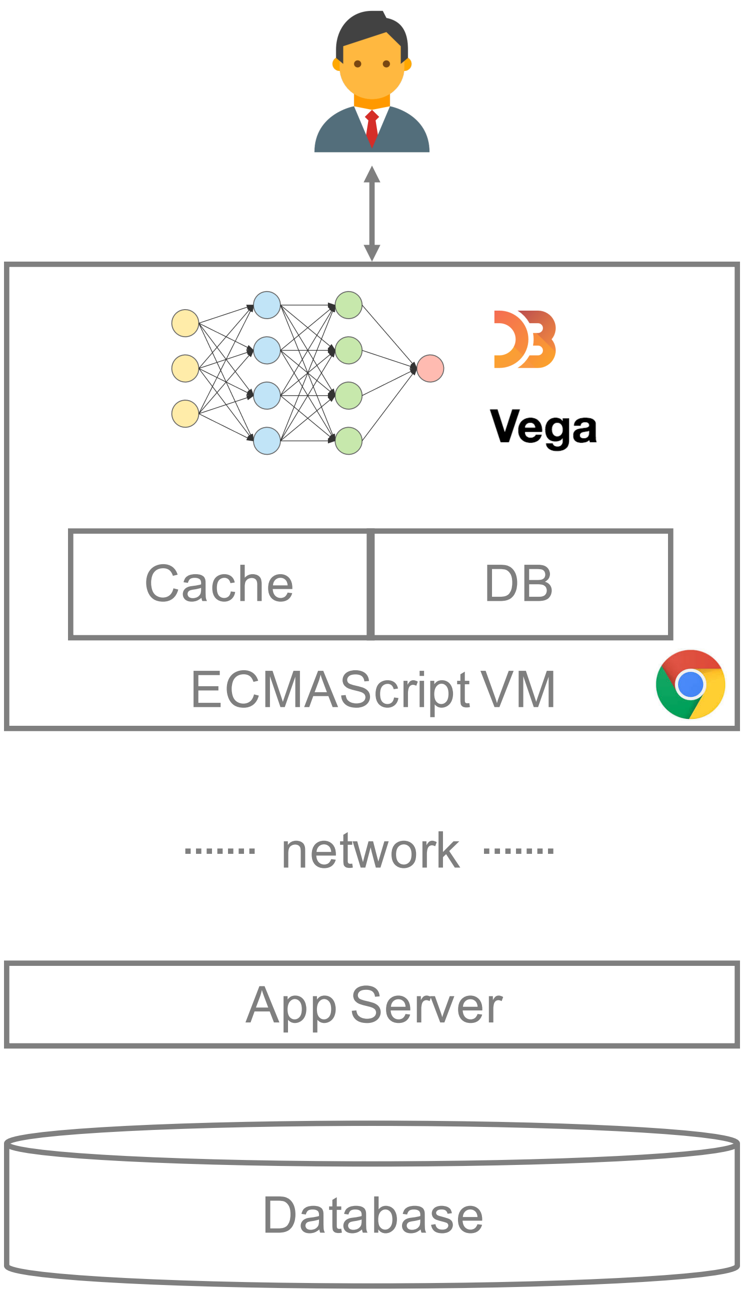
Great for small datasets

Bigger dataset? Much messier!

Communication overheads

DBs slower

Queries slower



SVD architecture

SQL, Vis, DB

Client-side vis libraries very robust

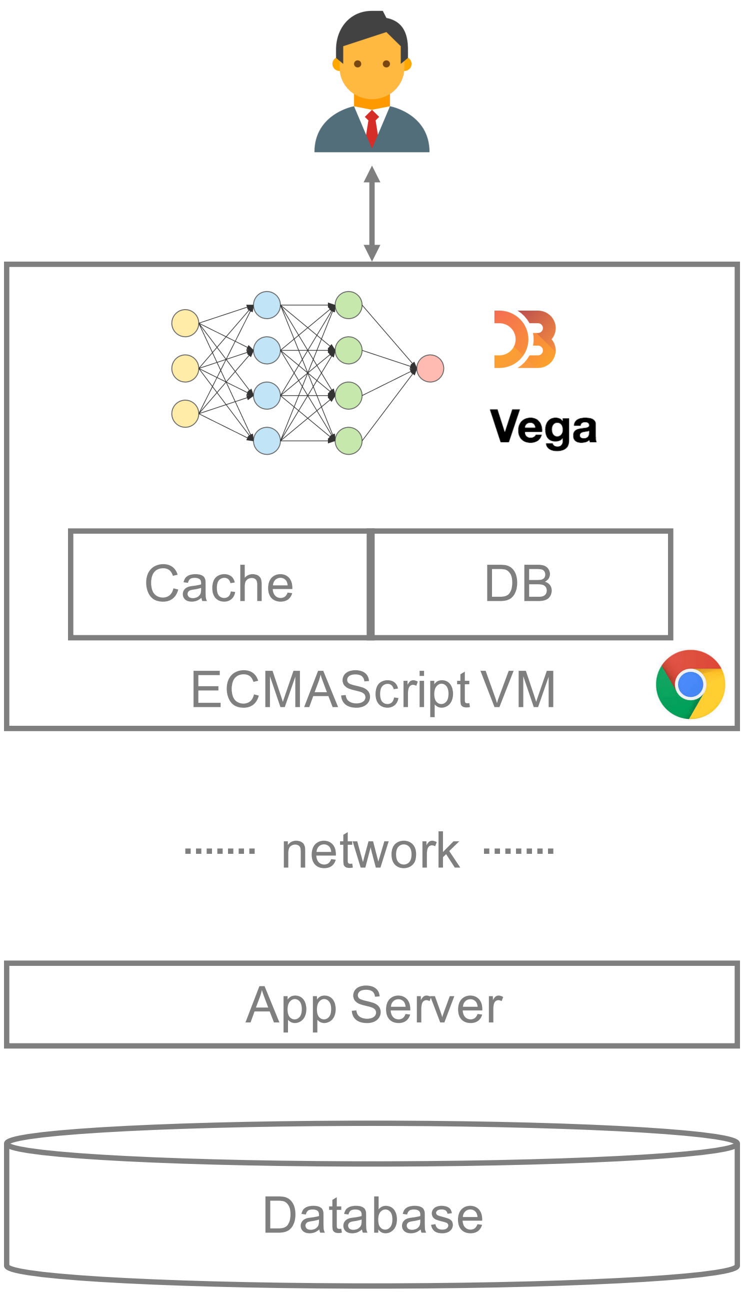
- Borrows data flow, event processing, incremental updates from databases.
- Outputs scene graph
- Great for small datasets

Bigger dataset? Much messier!

- Communication overheads
- DBs slower
- Queries slower
- Caches everywhere
- Smart caches (DBs) everywhere

SVD architecture

SQL,Vis,DB



Latency is cumulative

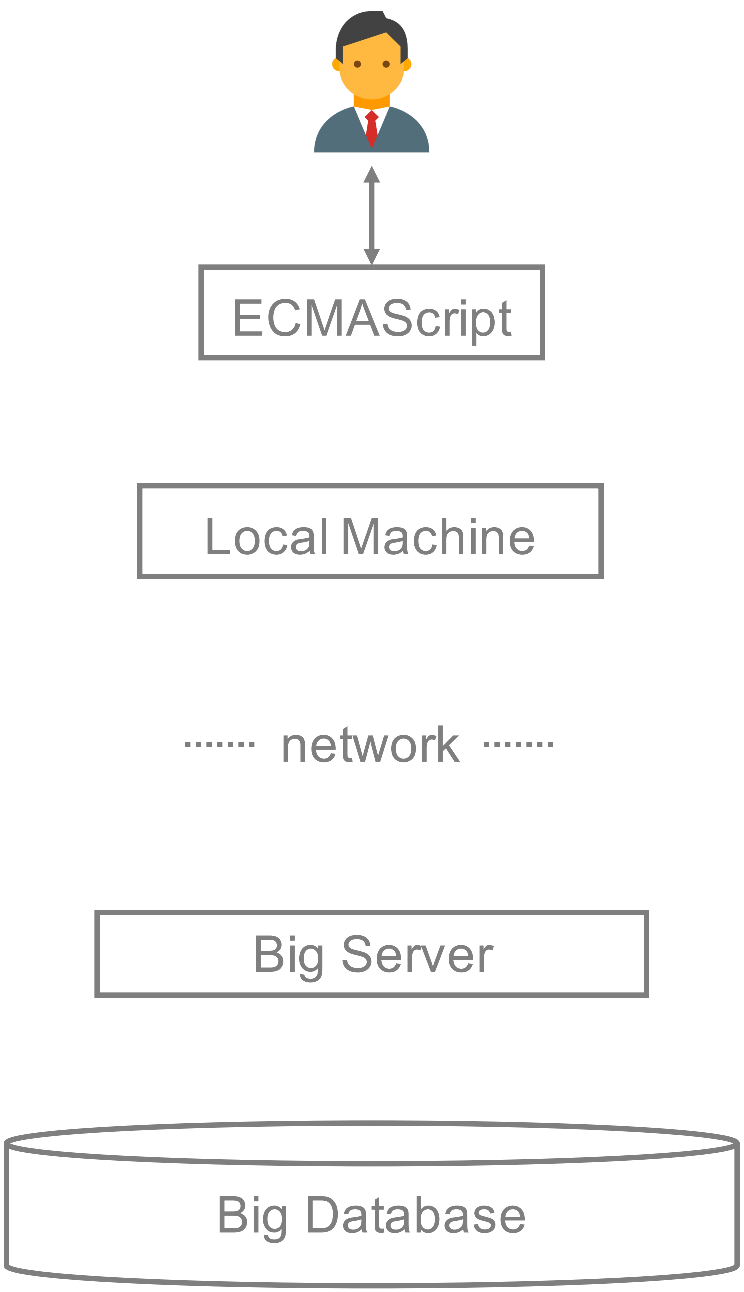
Separate components/code lines

Manual implementation

Manual optimization

Akin to writing queries
before Rel Alg.

Vis Memory Hierarchy

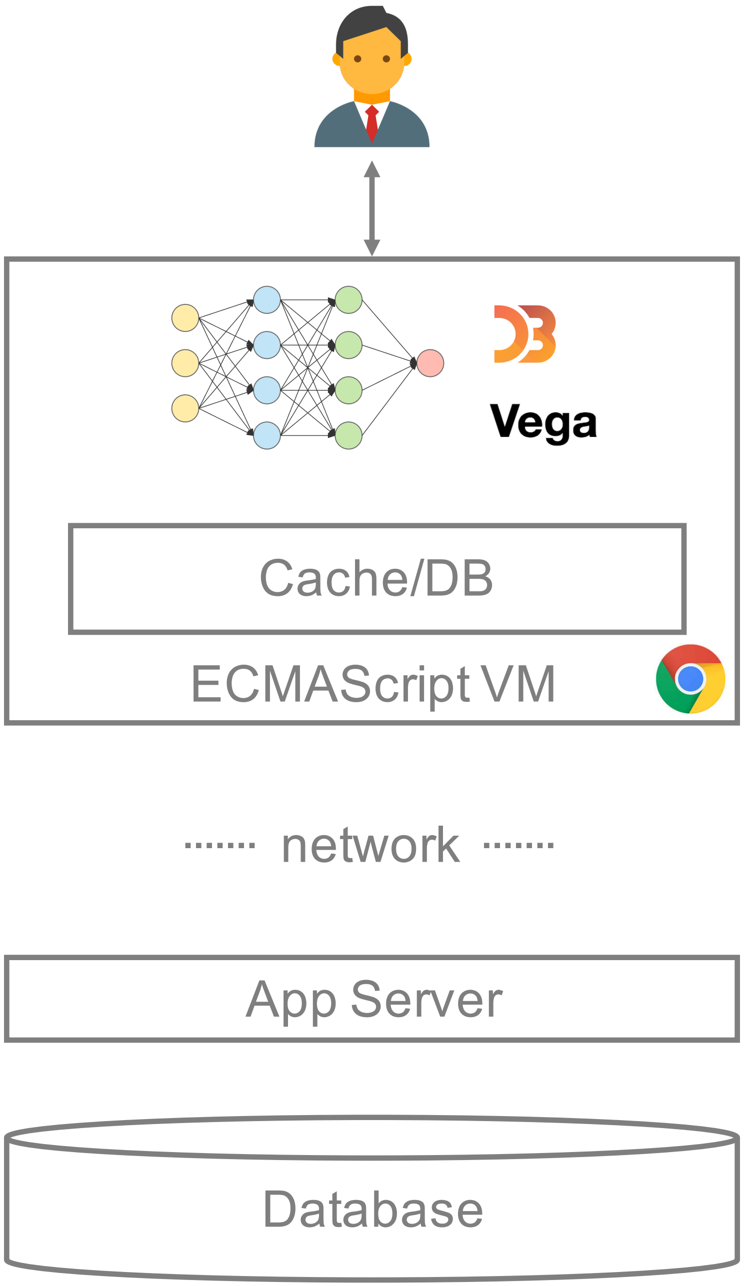


Absolute latency expectations

Complex analysis workloads

Data dense

Analysis (usually) already known



The Bottom Line

Simple programming API for interactive applications over large-scale/in-DB data is

unsolved

→

Massive opportunity

Mechanisms

that leverage vis semantics

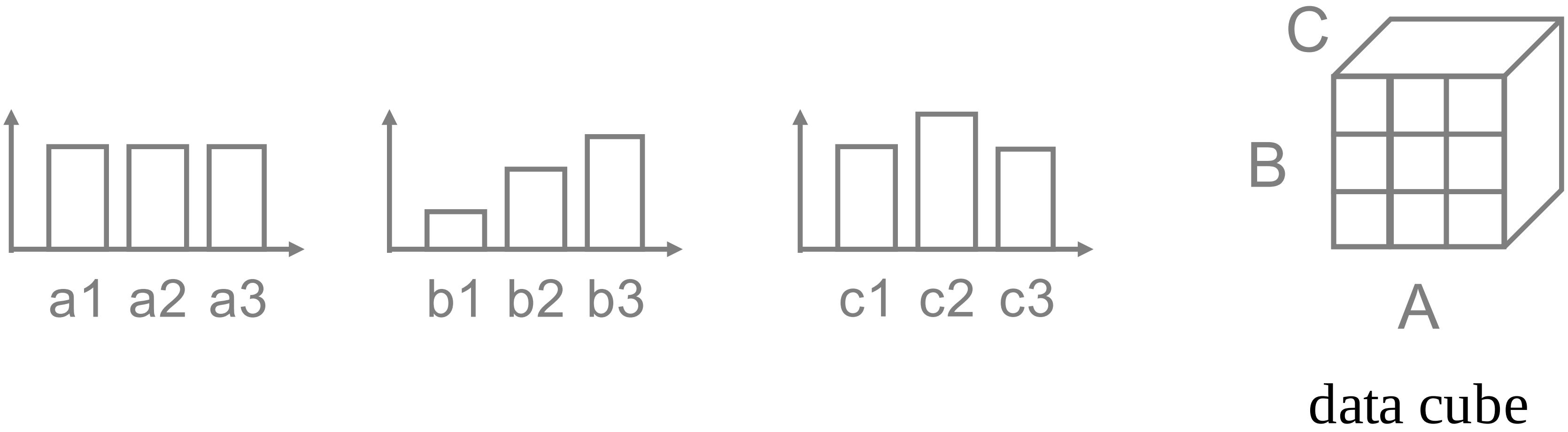
Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

Cubes: Immens

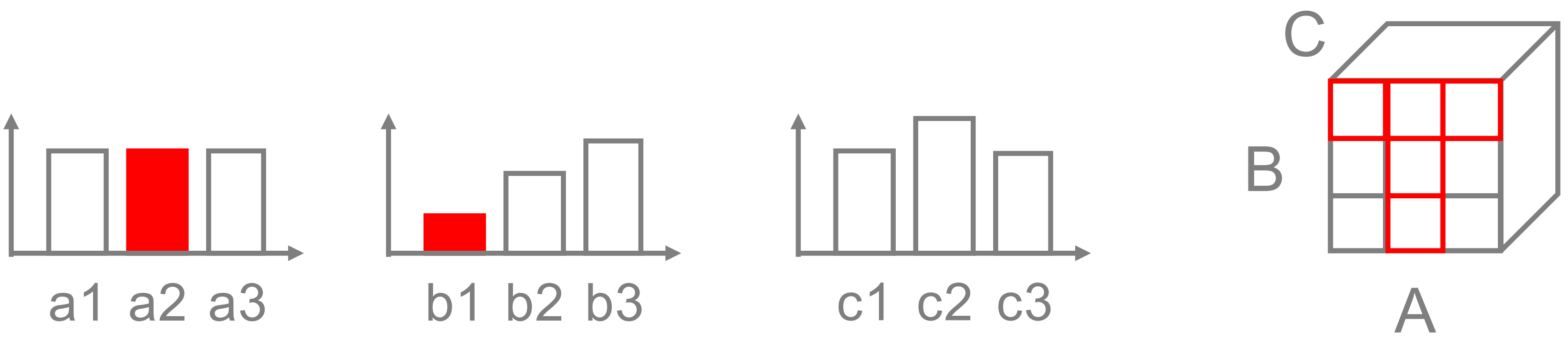


Cross filtering

interact with data in A, filter data in B

Build data cube

Cubes: Immens



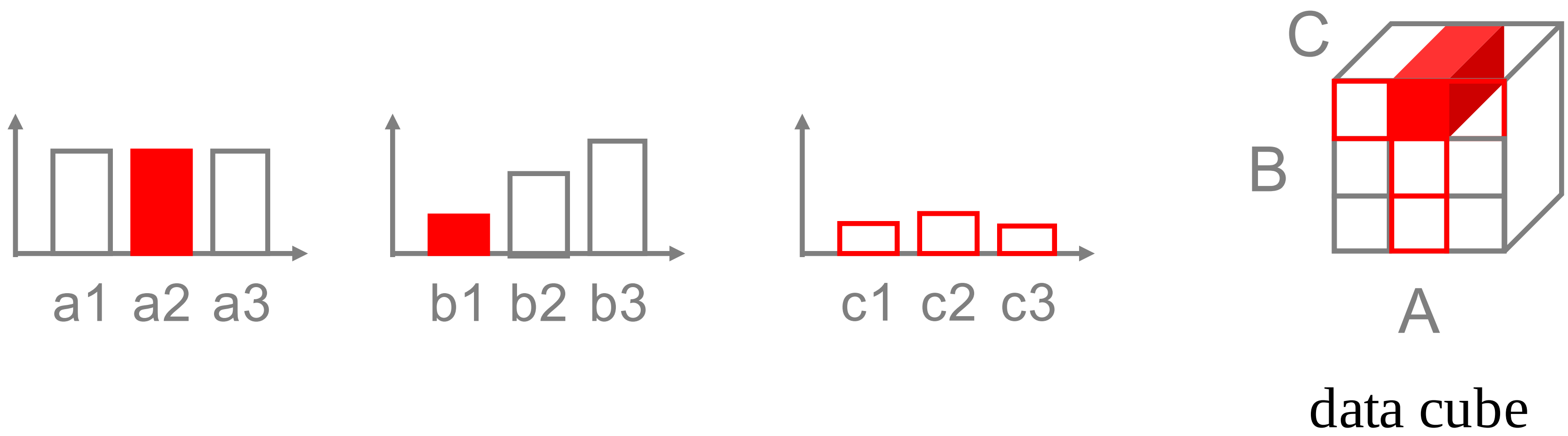
data cube

Cross filtering

interact with data in A, filter data in B

Build data cube

Cubes: Immens



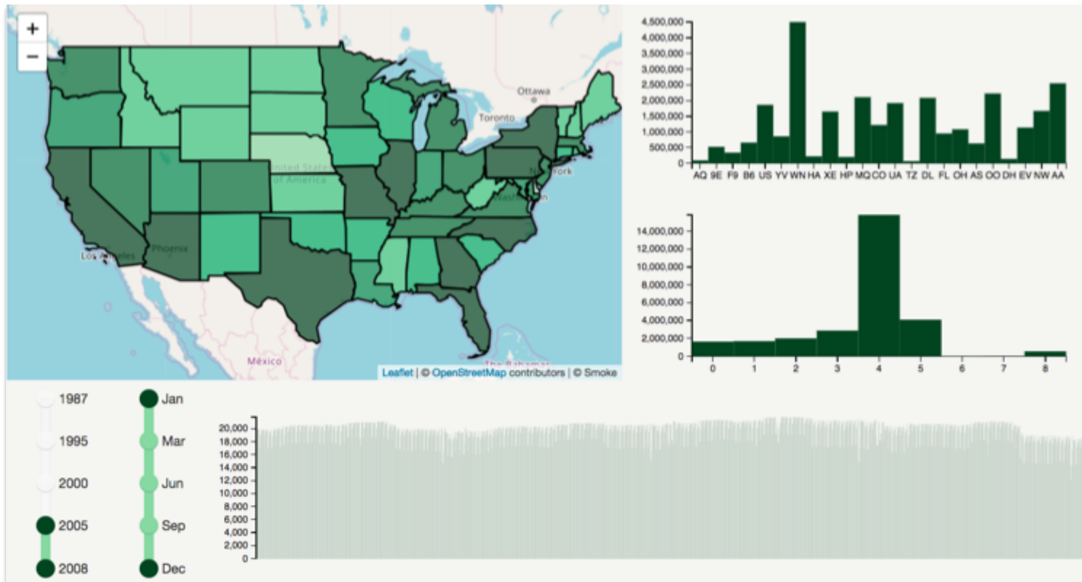
Cross filtering

interact with data in A, filter data in B

Build data cube

Cubes: Immens

States



Airlines

Departure Delay

Years

Day

Months

6D CUBE

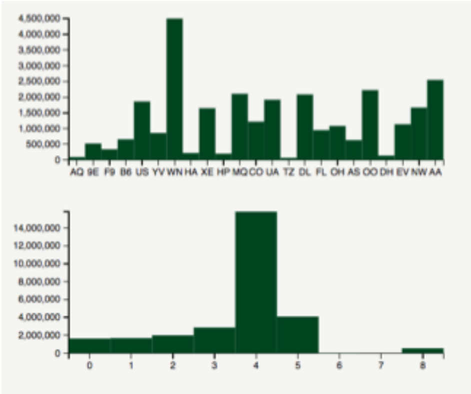
50 * 21 * 12 * 23 * 9 * 7000 > 18B ints ~ 144GB

Idea: User doesn't express all combinations

Cubes: Immens

`SELECT airline, COUNT`

`SELECT delay, COUNT`



Airlines

Departure Delay

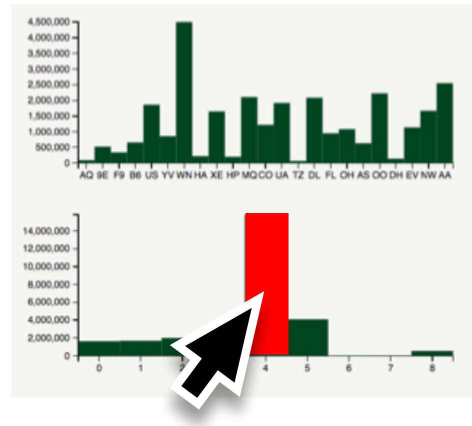
Cubes: Immens

1D

```
SELECT airline, COUNT
        WHERE delay = 4
```

1D

```
SELECT delay, COUNT
```



Airlines

Departure Delay

Interaction is 2D:

CUBE on airline, delay

Build cube for *pairs* of plots

K 1D plots, N values per attr

Naïve:	N^K	~18,000,000,000
Immens:	N^2K^2	809,765

Cubes

Takeaway

Scale to dimensionality of vis interactions

data is high dim

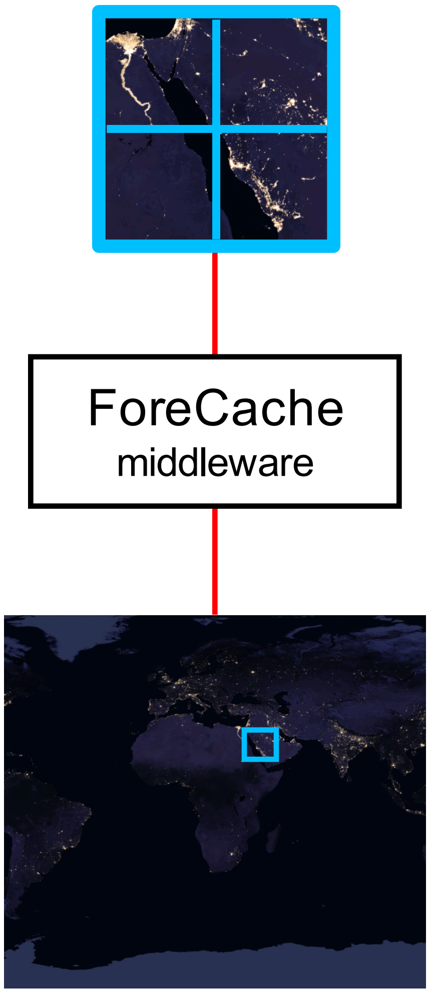
interactions can be low dim

Refs: nanocubes (Lins. TVCG13); hashedcubes (Pahins. InfoVIS16); Falcon (Moritz. CHI19)

Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

Network: ForeCache



NASA World View

Local viewport

Prefetcher

Massive tiled dataset

Middleware

easier to deploy

middleware has more resources

Middleware sees *inputs*

Client requests (pan/zoom)

Data content

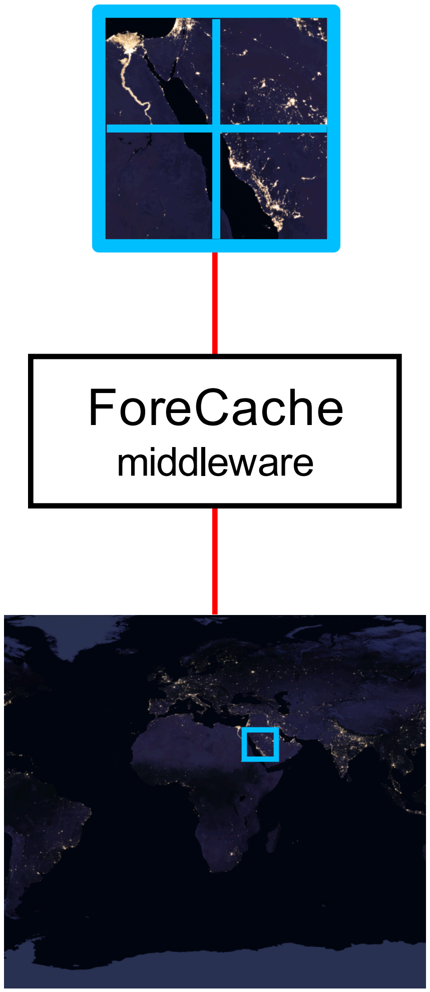
Middleware does *actions*

make (good) predictions

pre-fetch tiles/requests

Dynamic Prefetching of Data Tiles for Interactive Visualization. Battle et al.

Network: ForeCache



NASA World View

Local viewport

Prefetcher

Massive tiled dataset

Middleware

easier to deploy

middleware has more resources

Middleware sees *inputs*

Client requests (pan/zoom)

Data content

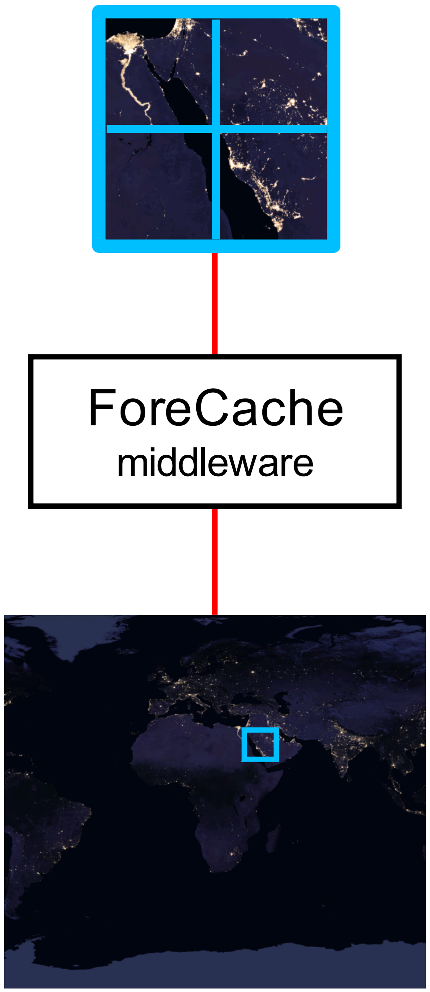
Middleware does *actions*

make (good) predictions

pre-fetch tiles/requests

Dynamic Prefetching of Data Tiles for Interactive Visualization. Battle et al.

Network: ForeCache



NASA World View

Local viewport

Prefetcher

Massive tiled dataset

Predict 1 of k actions

P(action | state)

Actions:

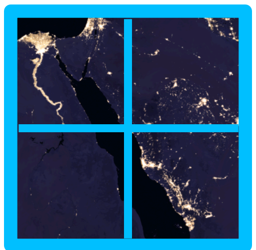
- Pan ← ↑ → ↓ ↖ ↗ ↘ ↙
- Zoom

Extend State using:

- Behavior features
  - Overview, Zoom, Details on Demand
- Map tile content features

Dynamic Prefetching of Data Tiles for Interactive Visualization. Battle et al.

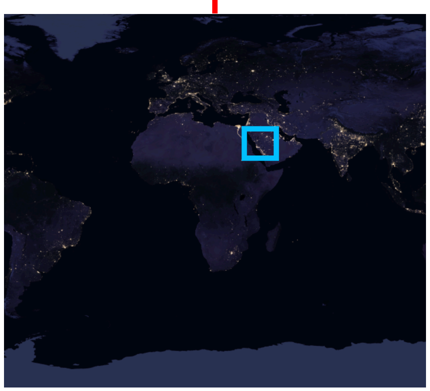
Network: Other Examples



Local viewport

ForeCache
middleware

Prefetcher



NASA World View

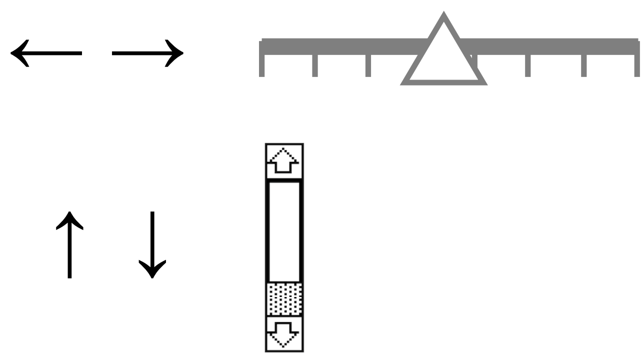
Massive tiled dataset

Predict 1 of k actions

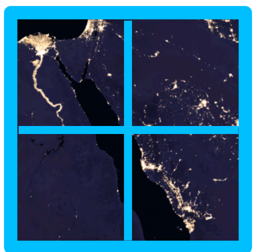
P(action | state)

Restrict Actions

Semantic windows, ATLAS, SCOUT



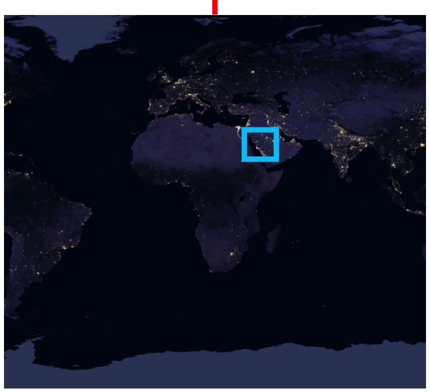
Network: Other Examples



Local viewport

ForeCache
middleware

Prefetcher



NASA World View

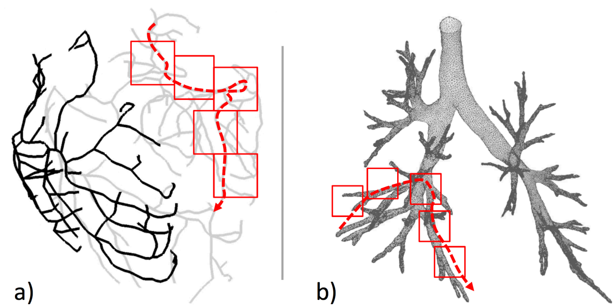
Massive tiled dataset

Predict 1 of k actions

P(action | state)

Apply Domain Structure

SCOUT: graph edges instead of arbitrary 3D navigation. Prefetch from disk



DICE, Query Steering: OLAP/Markov assumptions

Network

Takeaway

Pre-fetch relies on prediction

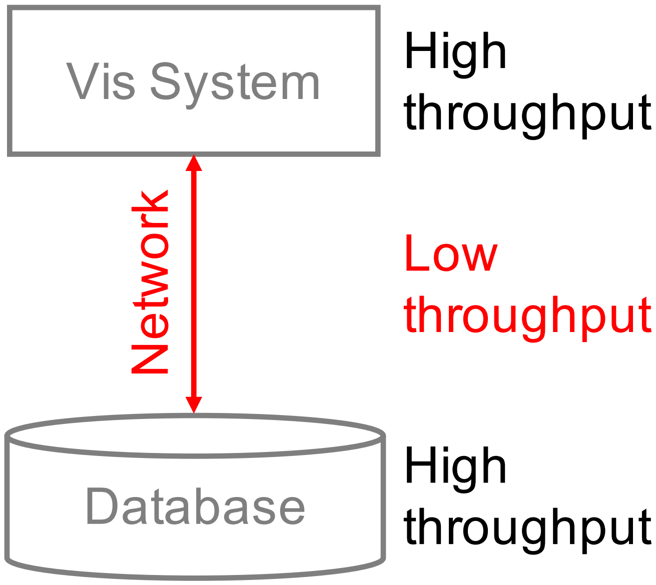
improve predictor w/
interaction + content

Refs: Falcon (Moritz. CHI19); Momentum/Hotspot (Doshi. Thesis); ATLAS (Chan. VAST08); DICE (Kamat. ICDE14); Semantic Windows (Kalinin. SIGMOD14); Scout (Tauheed. VLDB12)

Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

Network: M4



Prefetching trades bandwidth for latency

What if bandwidth limited?

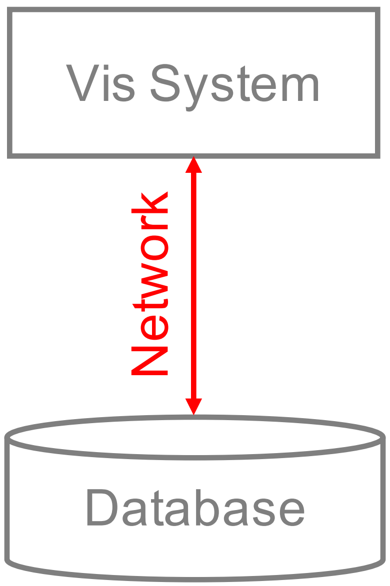
Prefetch makes things *worse*

M4: A Visualization-Oriented Time Series Data Aggregation. Jugel et al.

Network: M4

Low
Resolution

```
res = Q(database)      // on DB
vis = render(res)      // on client
```



High
throughput

Network is bottleneck
Query results can be massive

Low
throughput

Idea
vis resolution << result size

High
throughput

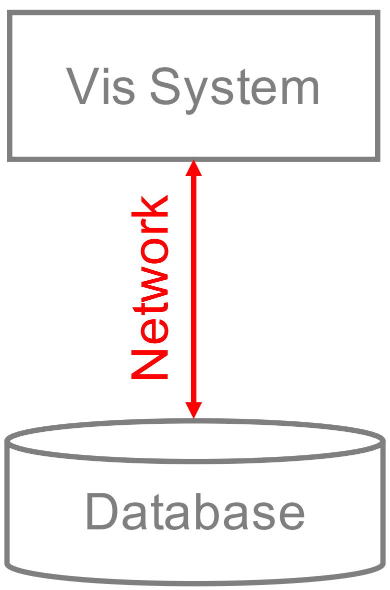
kills the network

M4: A Visualization-Oriented Time Series Data Aggregation. Jugel et al.

Network: M4

Low Resolution

```
res' = M4_Q(database)   // on DB
vis  = render(res')     // on client
```



High throughput

Low throughput

High throughput

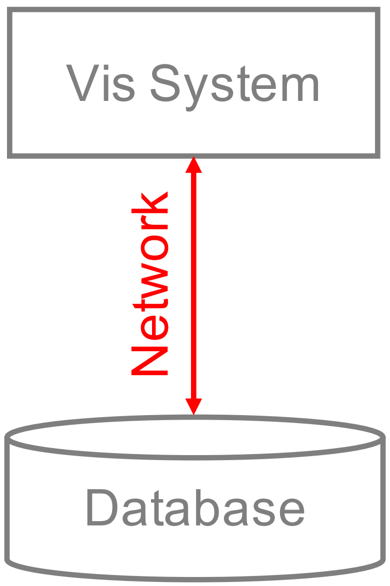
Network is bottleneck
Query results can be massive

Idea
vis resolution ~ M4(results)

M4: A Visualization-Oriented Time Series Data Aggregation. Jugel et al.

Network: M4

Low
Resolution



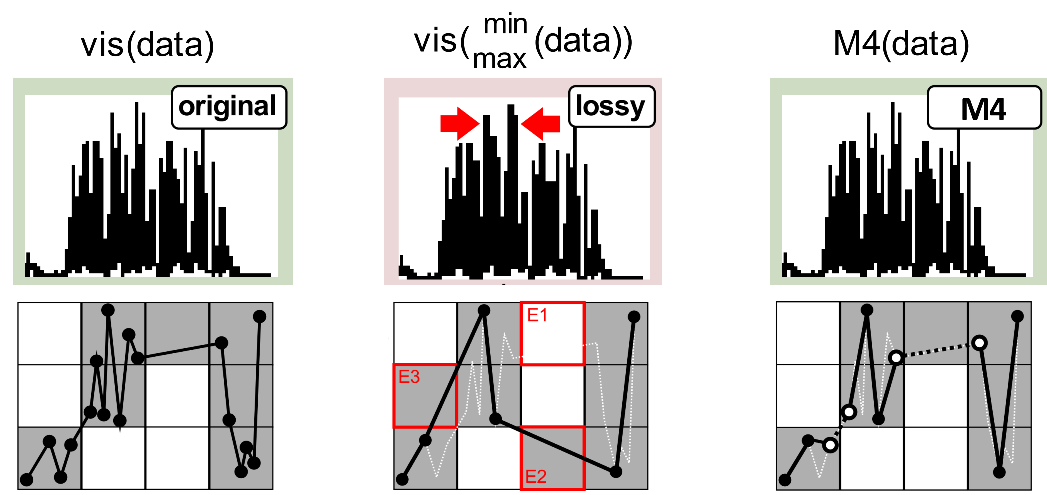
High
throughput

Low
throughput

High
throughput

Each pixel has *many* data values.
Aggregate by pixel? How?

Account for rasterization & antialiasing



M4: A Visualization-Oriented Time Series Data Aggregation. Jugel et al.

Network

Takeaway

Rendering Push-down

open rendering black-box &
push into database

Overview of Mechanisms

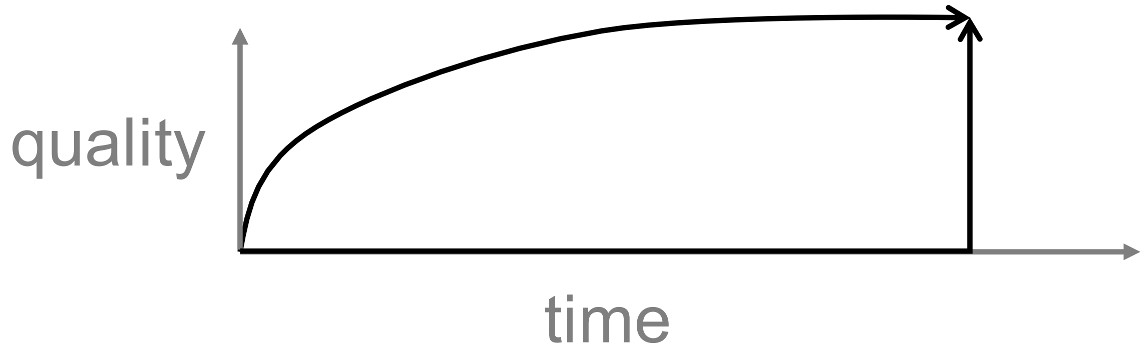
Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Progressive Vis and Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

Progressive Visualization

Visualization *quality* improves over time

Many forms of Progressive, such as...

- Queries
- Rendering
- Interaction granularity
- Data sampling
- Result Encoding
- Interaction Design



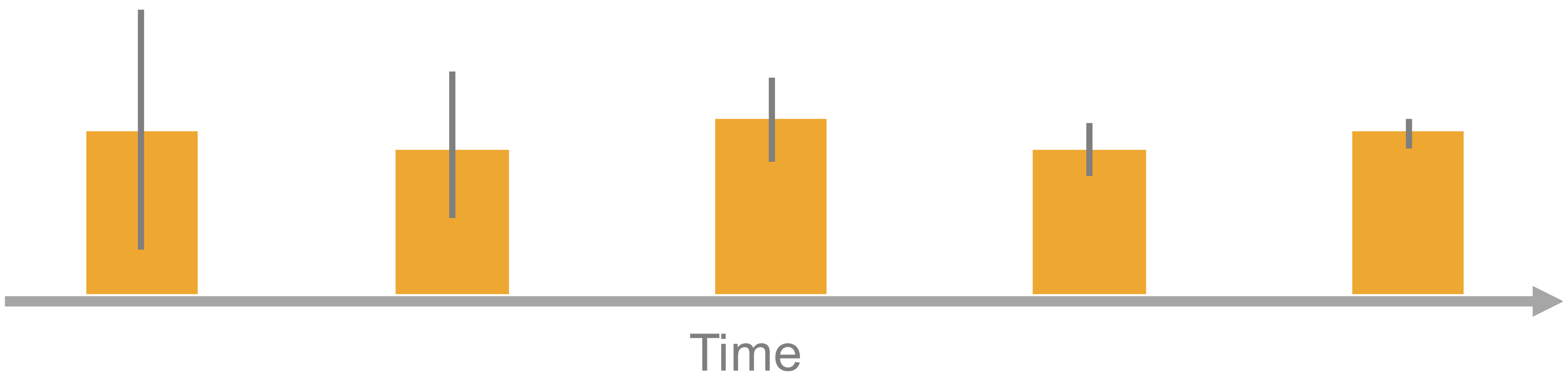
Progressive Queries

Quality ~ error bounds for Q(D)

Online Aggregation (Hellerstein SIGMOD97)

Wander Join (Li SIGMOD16)

Iterative sampling w/ perceptual models (P-funk Abali15)



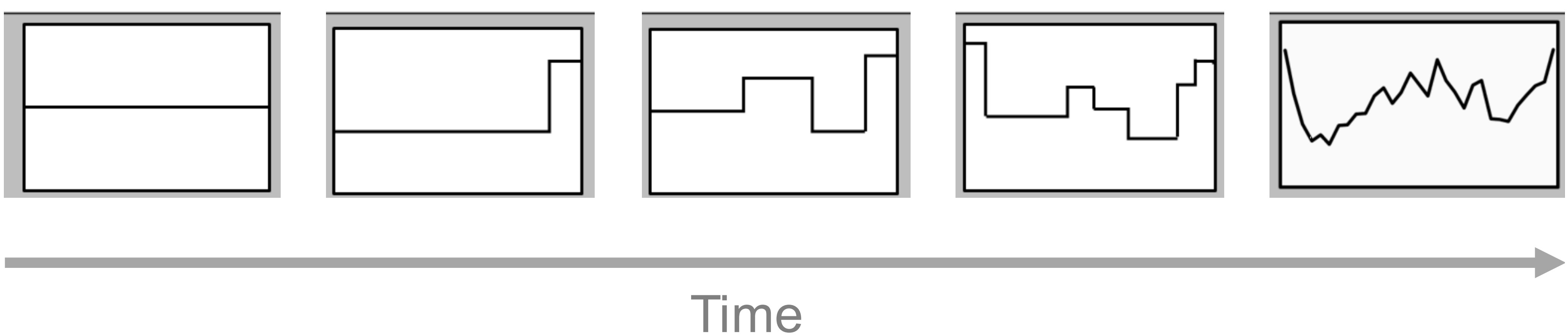
Progressive Communication

Quality ~ E[difference from full resolution]

Progressive image encoding (JPEG)

Speak summary, then fill in details (CiceroDB Trummer19)

Incrementally sample at higher resolutions (IncVisage Rahman17)

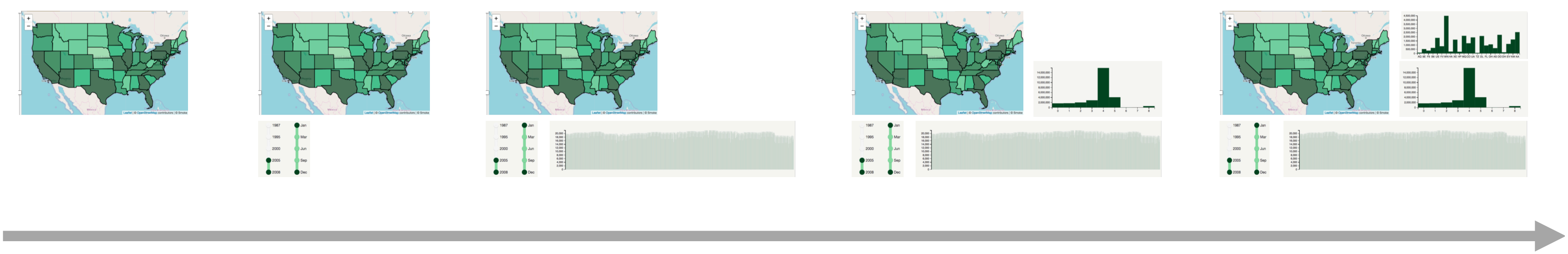


Progressive Loading

Quality ~ How much of the vis is shown

Number of charts loaded in dashboard

Webpage loading “above the fold” (Polaris netravali16)



Time

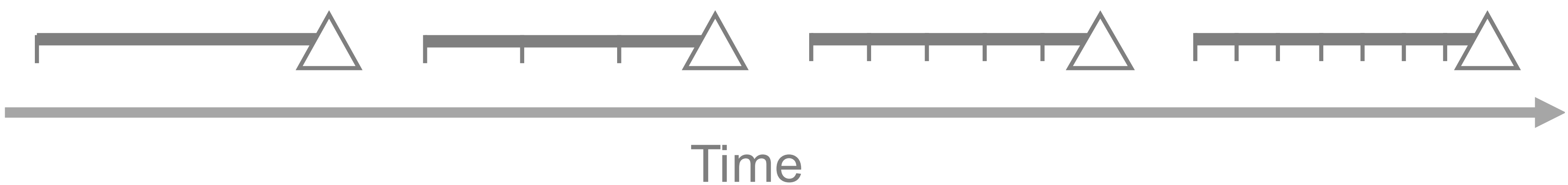
Progressive Interactions

Quality ~ Interaction granularity

Request throttling

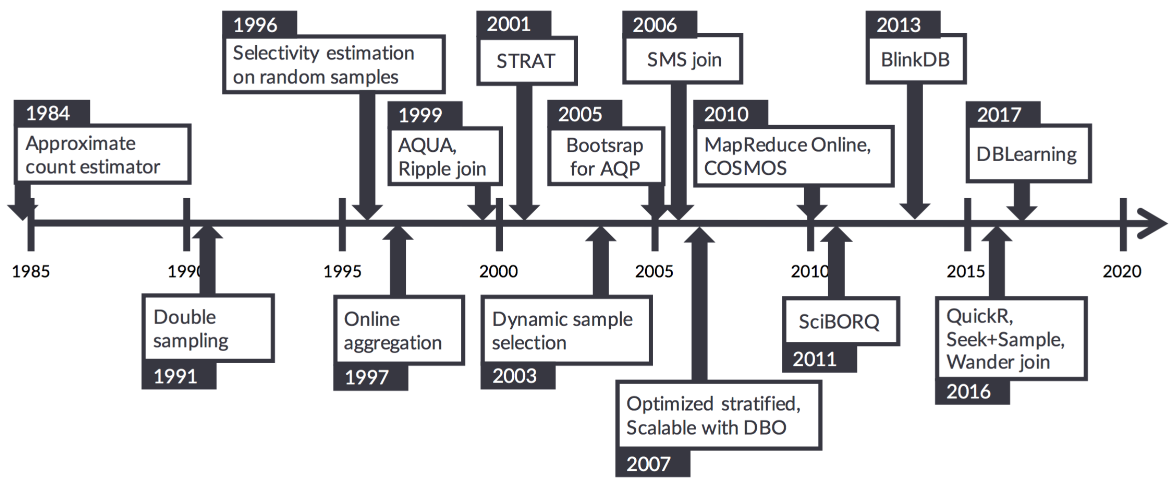
Higher brushing resolution over time (Falcon Moritz19)

Number of ticks in slider increase over time



Approximate Query Processing

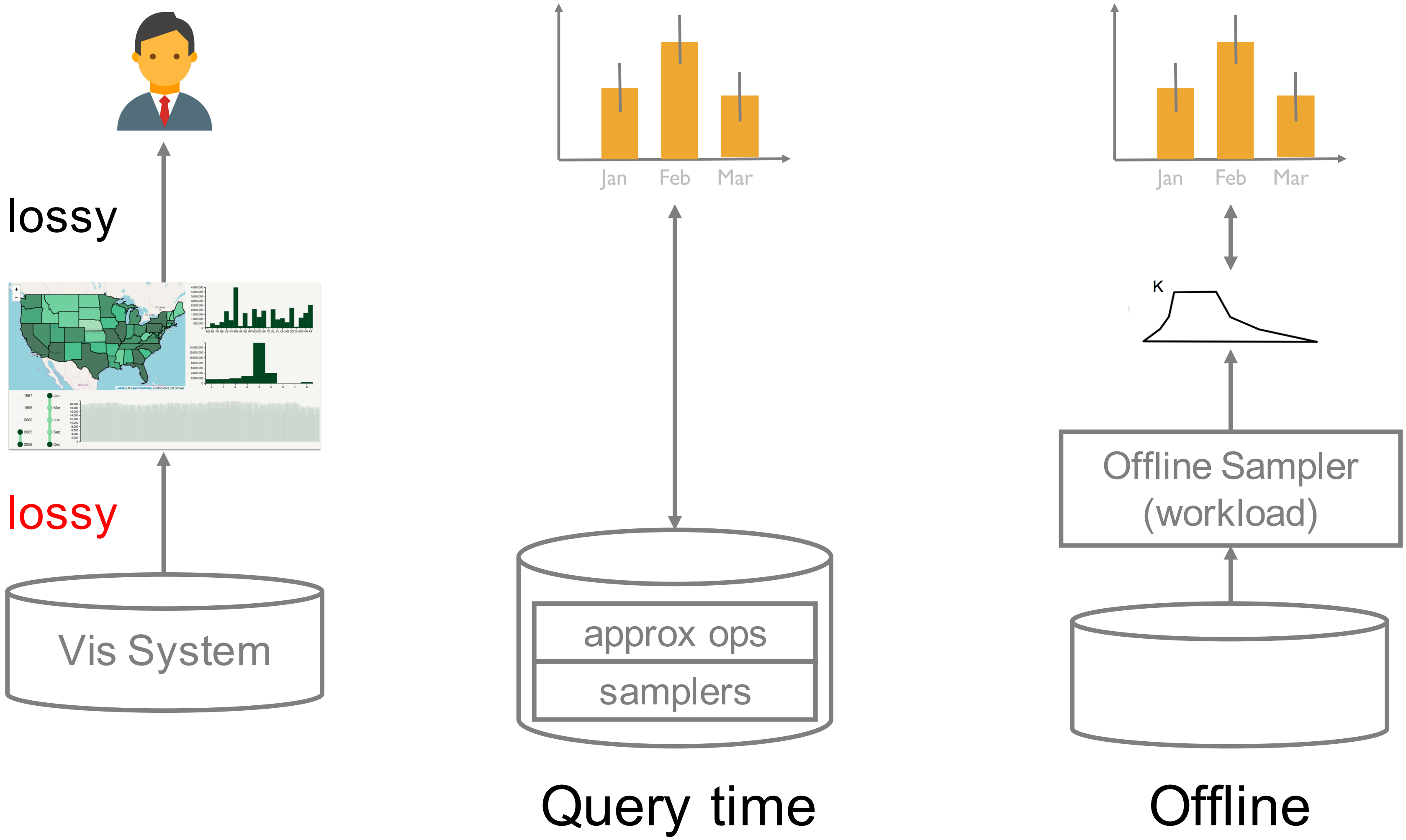
Progressive query processing and AQP are long-standing problems in databases



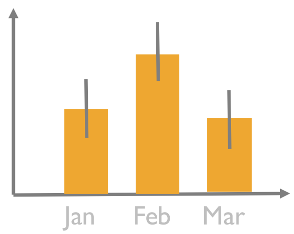
Adapting AQP to visualization also depends on visualization semantics

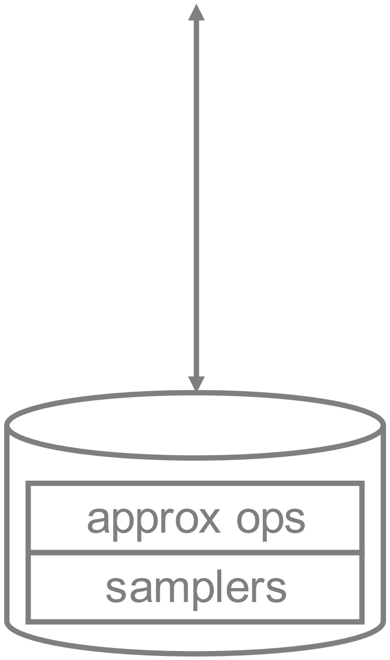
from https://yongjoopark.com/resources/verdictdb_sigmod18.pdf

Approximate Query Processing



Query Time Approximation





Query time

Offline:
Do nothing

When running Q:
Choose sample operators
Draw samples to answer Q

Query Time: WanderJoin

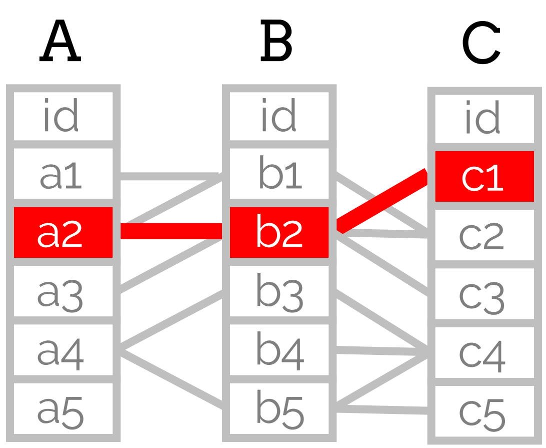
$$A \bowtie B \bowtie C$$

WanderJoin: leverage join indexes

edges represent join matches

Sample from A. Then path from A to C (a2-b2-c1) is a join sample

random walk is non-uniform independent sample



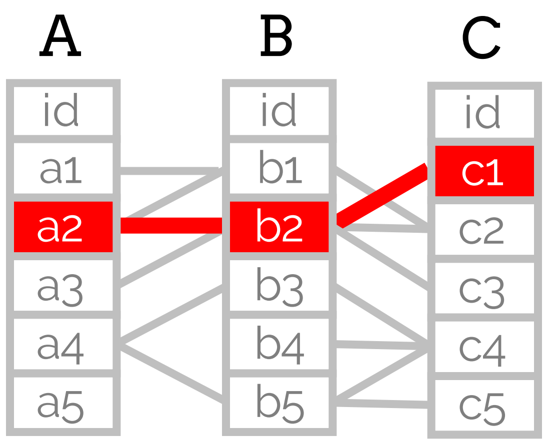
Wander Join: Online Aggregation via Random Walks – Li et al SIGMOD16

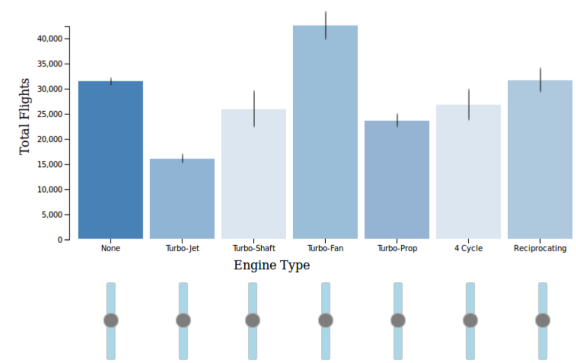
Query Time: WanderJoin

$$A \bowtie B \bowtie C$$

Vis is dominated by filtering and group-bys (filter by group)
Adapt WJ by biasing random walk via importance sampling for..

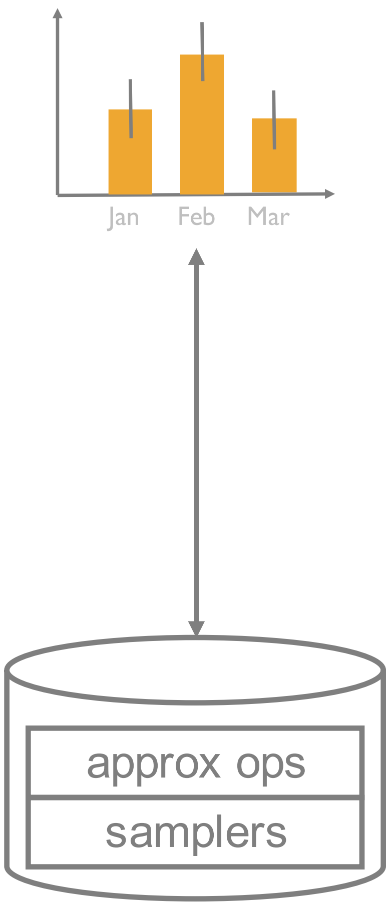
- Filters and dynamic selections
- User preferences





Selective Wander Join: Fast Progressive Visualizations for Data Joins – Procopio et al

Query Time Approximation



Query time

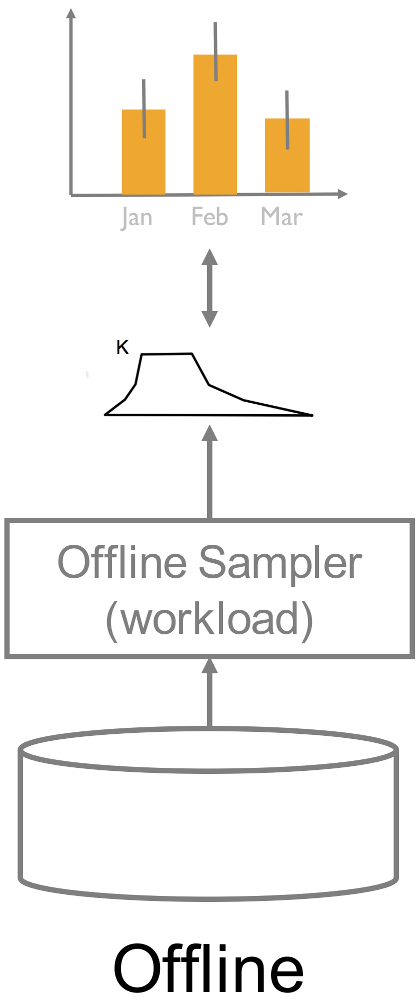
Offline:
Do nothing

When running Q:
Choose sample operators
Draw samples to answer Q

Sampling is expensive
WanderJoin uses join indexes.
Could use indexing time to build other data structs?
Can take long time for bounds to be small

Approximate Query Processing: No Silver Bullet – Chaudhuri et al

Offline



Offline:

Precompute samples given workload W

Typically stratify on columns groups in W

When running Q:

Pick precomputed samples

Use CLT/Hoeffding/bootstrap for err bounds

Approximate Query Processing: No Silver Bullet – Chaudhuri et al

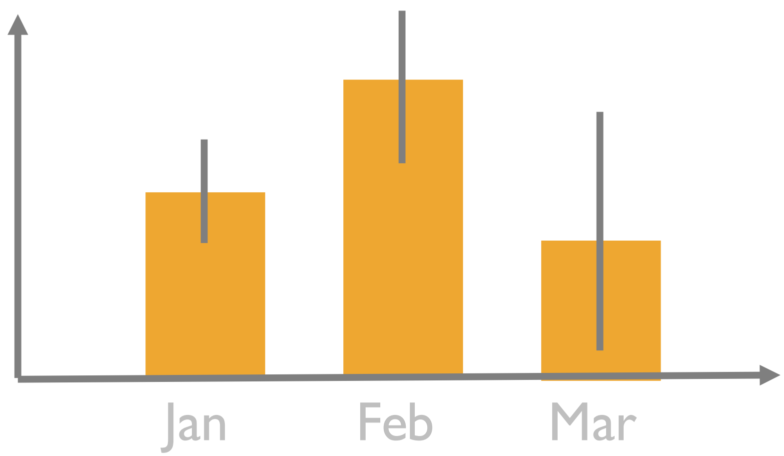
Offline: Sample+Seek

Adapts AQP towards visualization needs in 2 ways.

Challenges with confidence intervals (CIs)

- CI ~ $\underbrace{\text{std(n samples)}}_{\text{data dependent} \rightarrow \text{sensitive to outliers}}$ / sqrt(n)

- CIs unintuitive



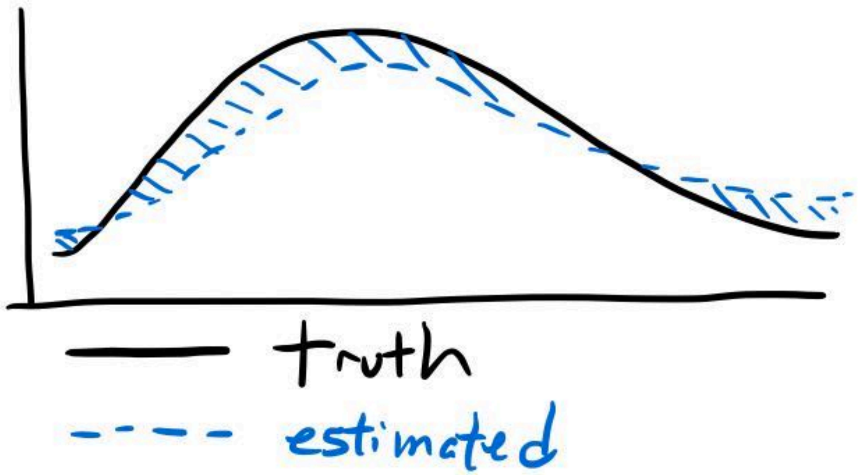
90th CIs

Useful?
¯_(ツ)_/¯

Offline: Sample+Seek

Proposes *Distributional Guarantee*

- Result modeled as normalized distribution
- Offline pre-computation will bound L_2 distance $\leq \varepsilon$
- Closer to understandable semantics



$|\text{truth} - \text{estimated}|_2 \leq \varepsilon$

Offline: Sample+Seek

Don't stratify by column groups

- Column groups may be different later on

Compute measure-biased samples for aggregated attrs

- Need to know aggregated measures up front

```
  SELECT a1, a2, ..., SUM(v1), COUNT(v2)
    FROM ...
GROUP BY a1, a2, ...
```

Offline: Sample+Seek

Measure-biased samples for `SUM(val)`: proportional to **`val`**

100 rows

`val`	p_{naive}	p_{s+s}
1	1/100	1/199
..	..	..
1	1/100	1/199
100	**1/100**	**100/199**

If we sample 1 val (**`100`**), what is our estimate?

100/(1/100) = 10,000

100/(100/199) = 199

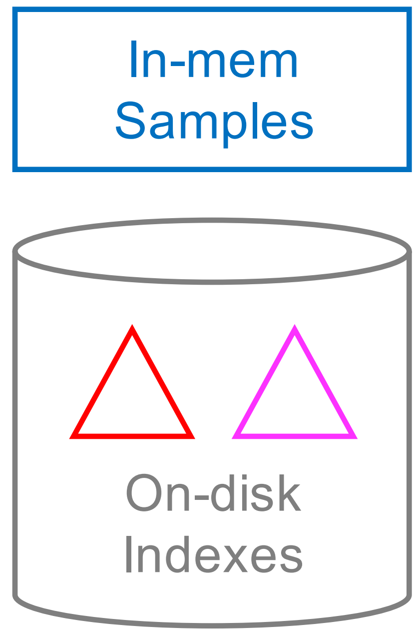
Addresses err bound's *Data Dependency*

Offline: Sample+Seek

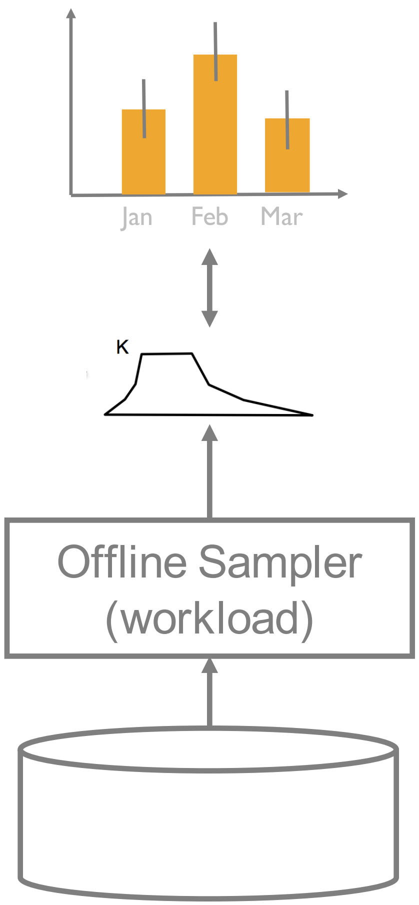
Sample probability based on value

```
  Q = SELECT  a, SUM(b)
      WHERE   c=1 ...
      GROUPBY b
```

```
result = Q(in-memory sample)
if enough samples in mem:
  return result
if very low selectivity:
  lookup rows directly
else:
  use measure-augmented index to
  draw sample biased by b
```



Offline



Offline

Offline:
Precompute samples given workload W
Stratify on columns groups in W

When running Q:
Pick precomputed samples
Use CLT/Hoeffding/bootstrap for err bounds

Hard to guarantee bounds are small if Q uses unseen col group
Measure-biased sampling helps, works if *aggregation function* is over sampled attributes!

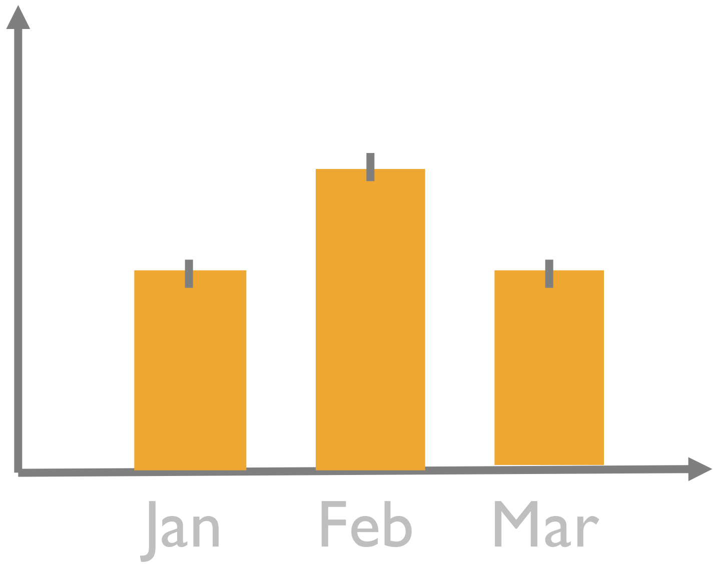
Approximate Query Processing: No Silver Bullet – Chaudhuri et al

What is *Quality?*

SELECT avg(sales)
GROUP BY month
ERROR 0.1 **CONF** 99%

Confidence interval per record

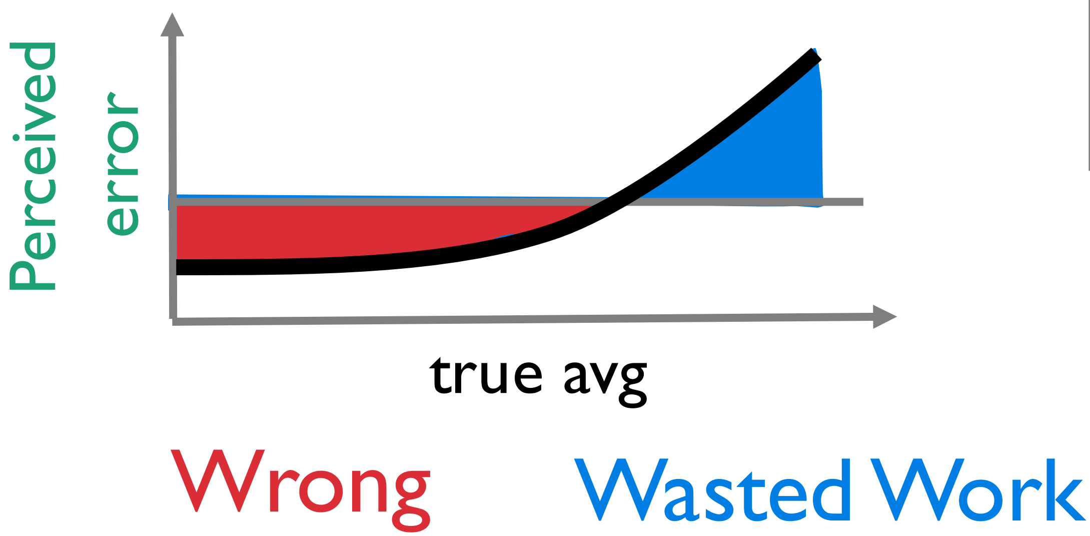
Error 0.1 Conf 99%

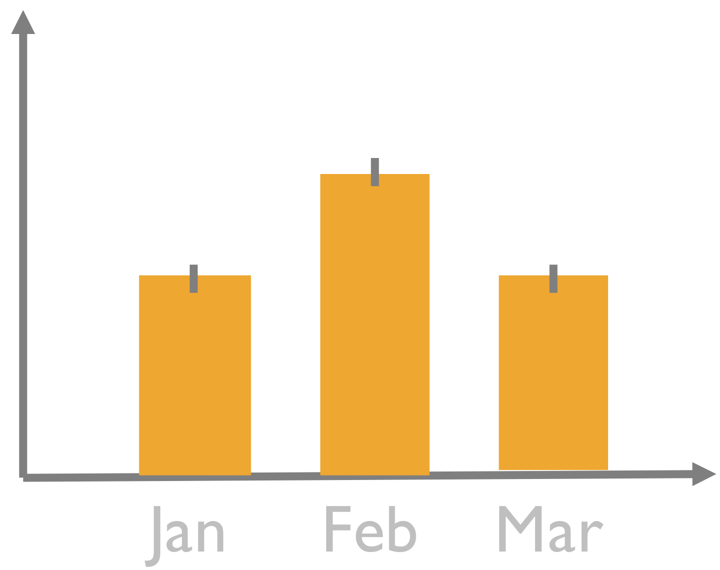


Stepping back, a bigger question is what quality should mean! Tricky even for classic error specifications in AQP

What is *Quality?*

SELECT avg(sales)
GROUP BY month
ERROR 0.1 **CONF** 99%





A single error for a query may not be sufficient. Perceptual research says that the error bounds DEPENDS on the result values.

PFunk-H (Alabi HILDA16); At-a-Glance (Ryan InfoVIS18)

What is *Quality?*

```
SELECT      avg(sales)
GROUP BY    month
ERROR 0.1 CONF 99%
```

Confidence interval per record

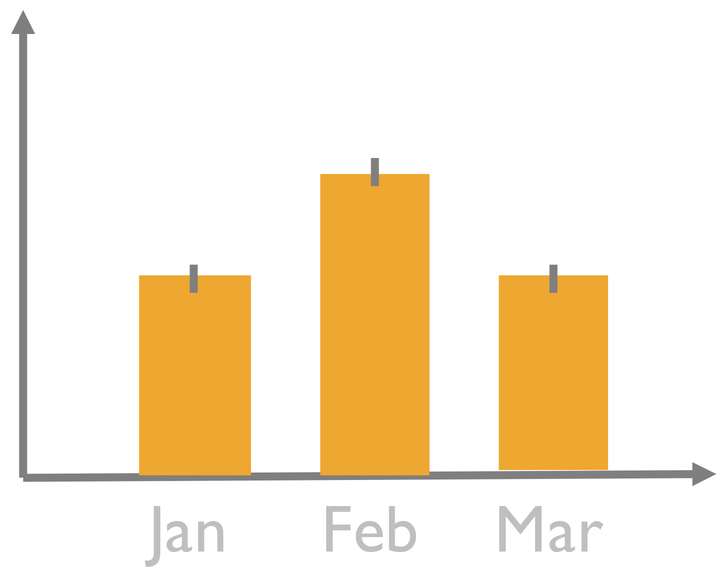
Error 0.1 Conf 99%

Pairwise statistical test

Pairwise CI don't overlap too much

Distributional guarantee

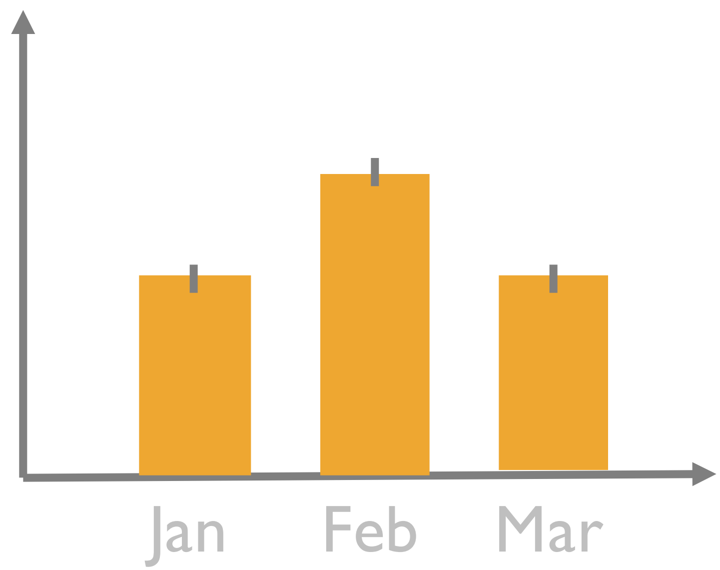
E[distance from true distribution]



CI isn't even the only notion of quality to begin with!

What is *Quality?*

```
SELECT      avg(sales)
GROUP BY    month
ERROR 0.1 CONF 99%
```



Q: Who decides Quality?

A: Perception Science

The Human Side (a sample of works)

At a Glance: Approximate Entropy as a Measure of Line Chart Visualization Complexity

Ryan et al. InfoVIS19

The Human User in Progressive Visual Analytics

Micallef et al. EuroVIS (Dagstuhl report)

What Users Don't Expect about Exploratory Data Analysis on Approximate Query Processing Systems

Moritz et al. HILDA17

Why Evaluating Uncertainty Visualization is Error Prone

Jessica Hullman BELIV16

Approx + Progressive

Takeaway

Perception Push-down

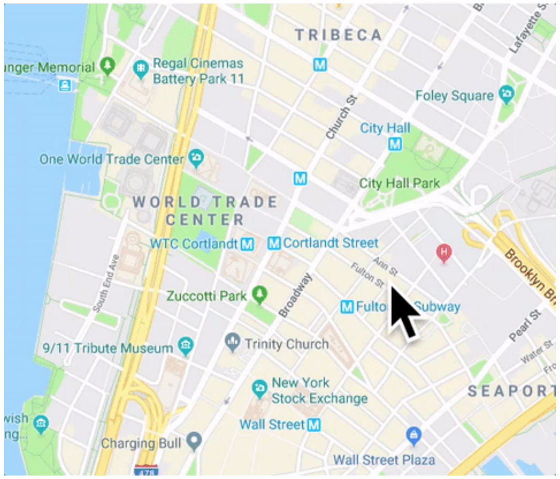
Model perceptual inaccuracy &
push into database

Dagstuhl report on progressive visualization: http://drops.dagstuhl.de/opus/volltexte/2019/10346/

Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

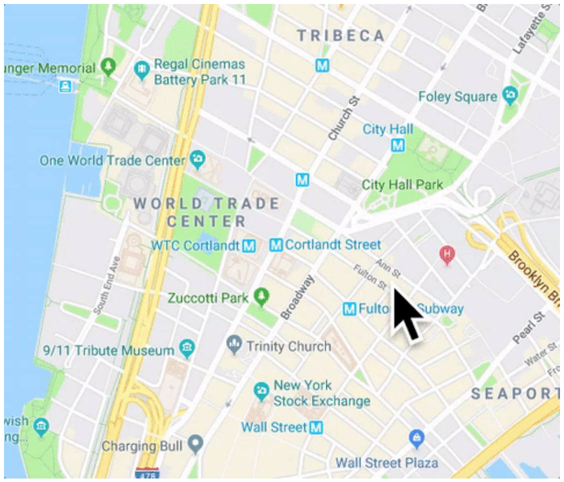
Tasks: Kyrix



Kyrix: visualization as map

Layers render rows (map tiles, pins)
User sees through viewport
Interaction = change bounding boxes
→Pan viewport to see more data
→Zoom/click to switch layers

Tasks: Kyrix



Kyrix: visualization as map

Layers render rows (map tiles, pins)

User sees through viewport

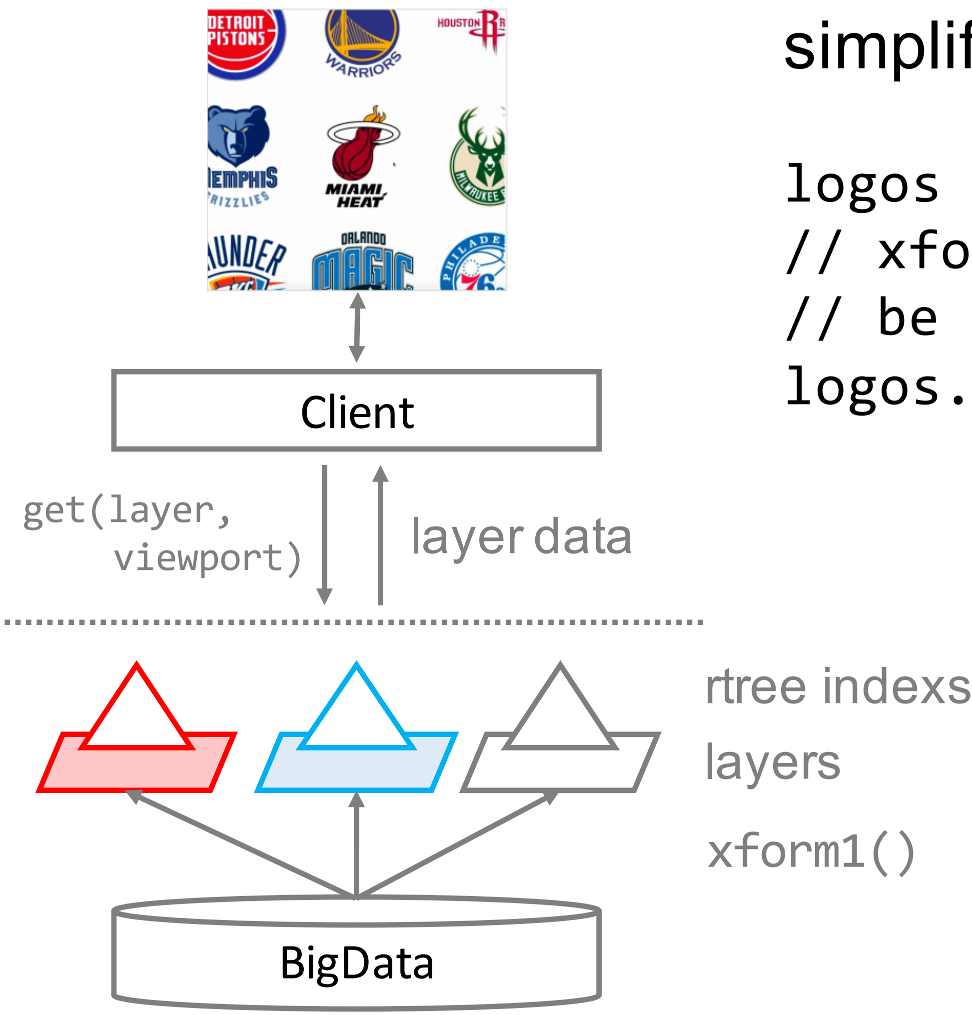
Interaction = change bounding boxes

→Pan viewport to see more data

→Zoom/click to switch layers

Kyrix: Interactive Pan/Zoom Visualizations at Scale – Tao EuroVis19

Tasks: Kyrix



simplified pseudocode

```
logos = vis.canvas(w, h)
// xform1 computes logo positions
// be rendered in layer
logos.newlayer(xform1, render1)
```

Easy Developer API

Tasks

Takeaway

Know the Task

Task-based Programming API

Leverage richer semantics

An End-to-end Relational Story

Overview of Mechanisms

Semantics	Idea
Interaction	Scale cube dimensionality to interactions
Interaction	Network Pre-fetch as prediction
Render	Aggregate results to reduce network cost Push rendering logic into query processing
Perception	Approximation Push perceptual inaccuracies into query processing
Task	Know the Task Ensure Vis API provides rich optimization hints

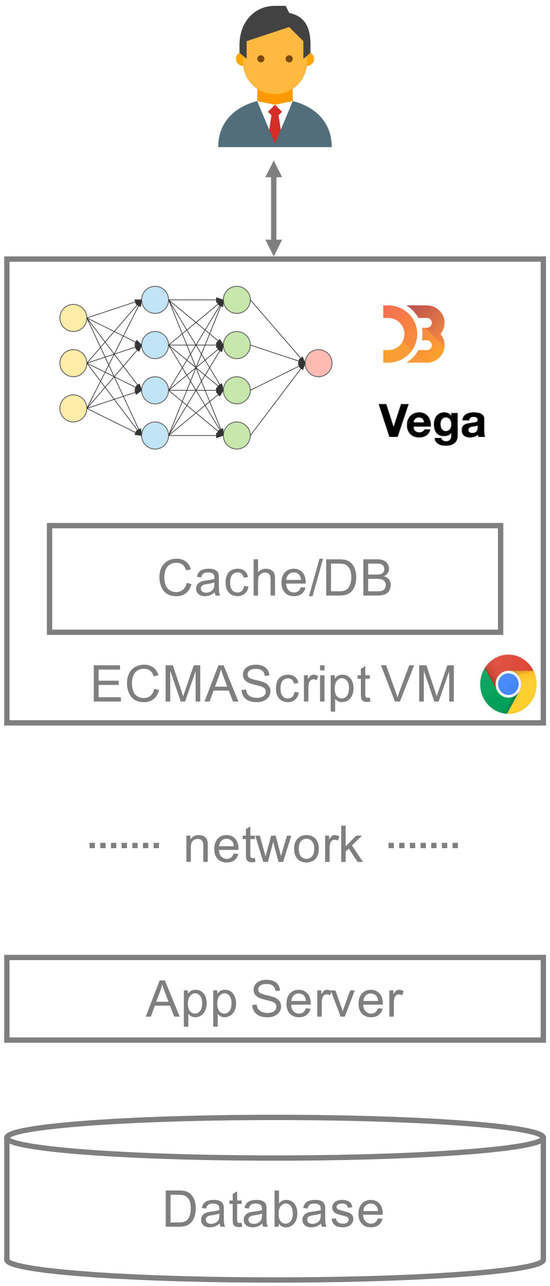
Many Disparate Optimizations

How to choose?

- Developer tells the system — Hard for dev. API?
- Special case the system — Limits Flexibility

Do they compose?

Need to model **application semantics** How?



Data Visualization Management System

Want to express *visualization*, *interaction*, *tasks*, *perception* all together

Vis and interaction as queries

Apply relational ideas end-to-end
- to interactions
- to consistency
- to design

Data Visualization Management System

Want to express visualization, interaction, tasks, perception all together

Vis and interaction as queries

Apply relational ideas end-to-end

- to interactions
- to consistency
- to design

Roadmap

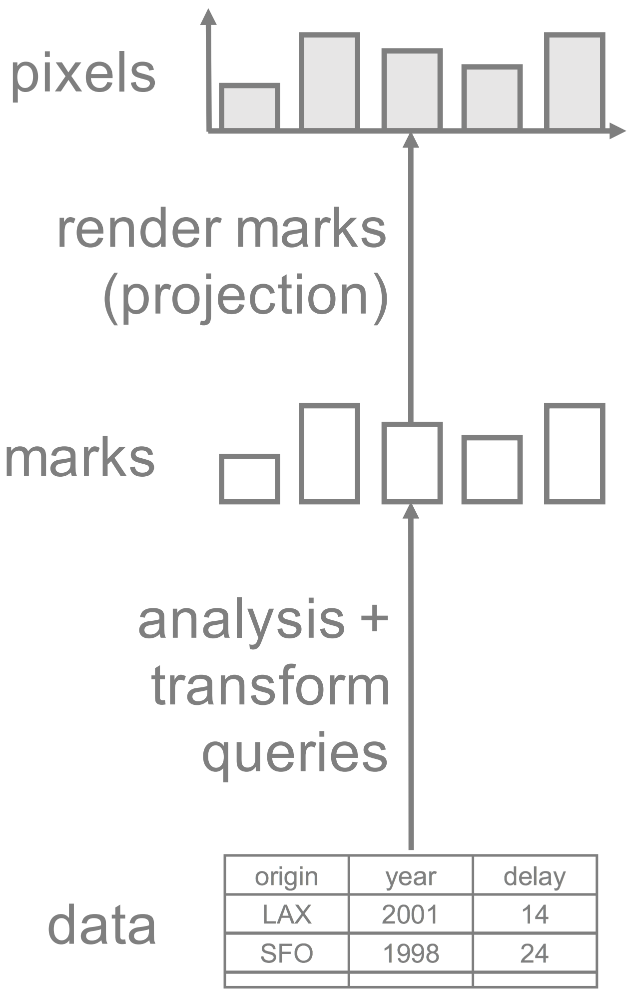
Overview of Relational Perspective

Optimizations within relational framework

Interactions as logical expressions

- Single-view interaction
- Multi-view interactions

A Relational Perspective

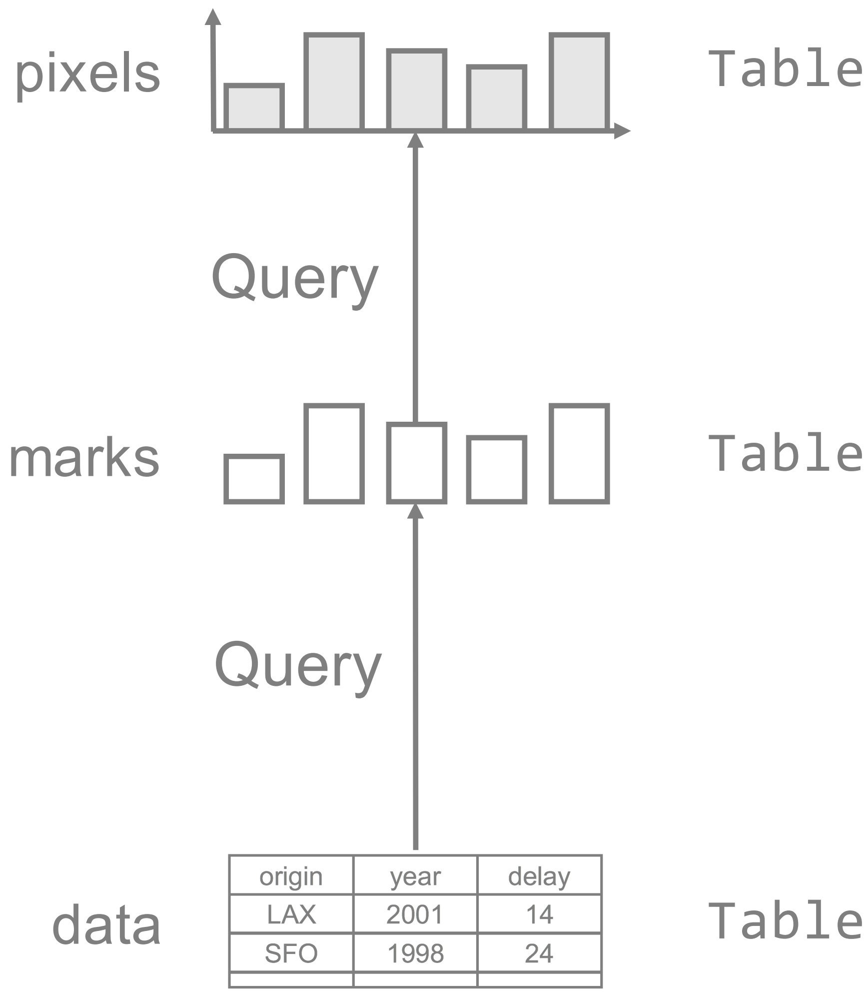



```
pixels(x, y, rgba)

marks(type, x, w, h, color)

flights(origin, year, delay,...)
```

A Relational Perspective

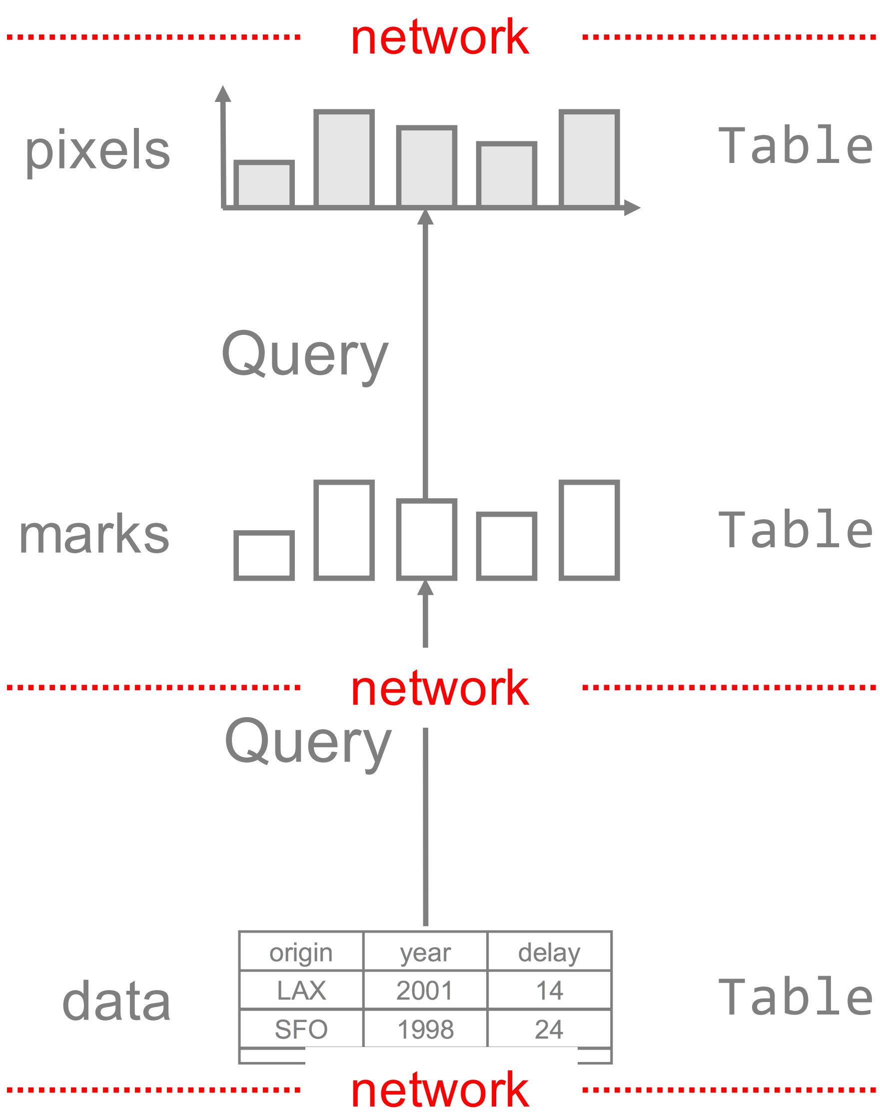


Logical model:

pipeline as a big query

input data, marks, pixels as tables.

A Relational Perspective

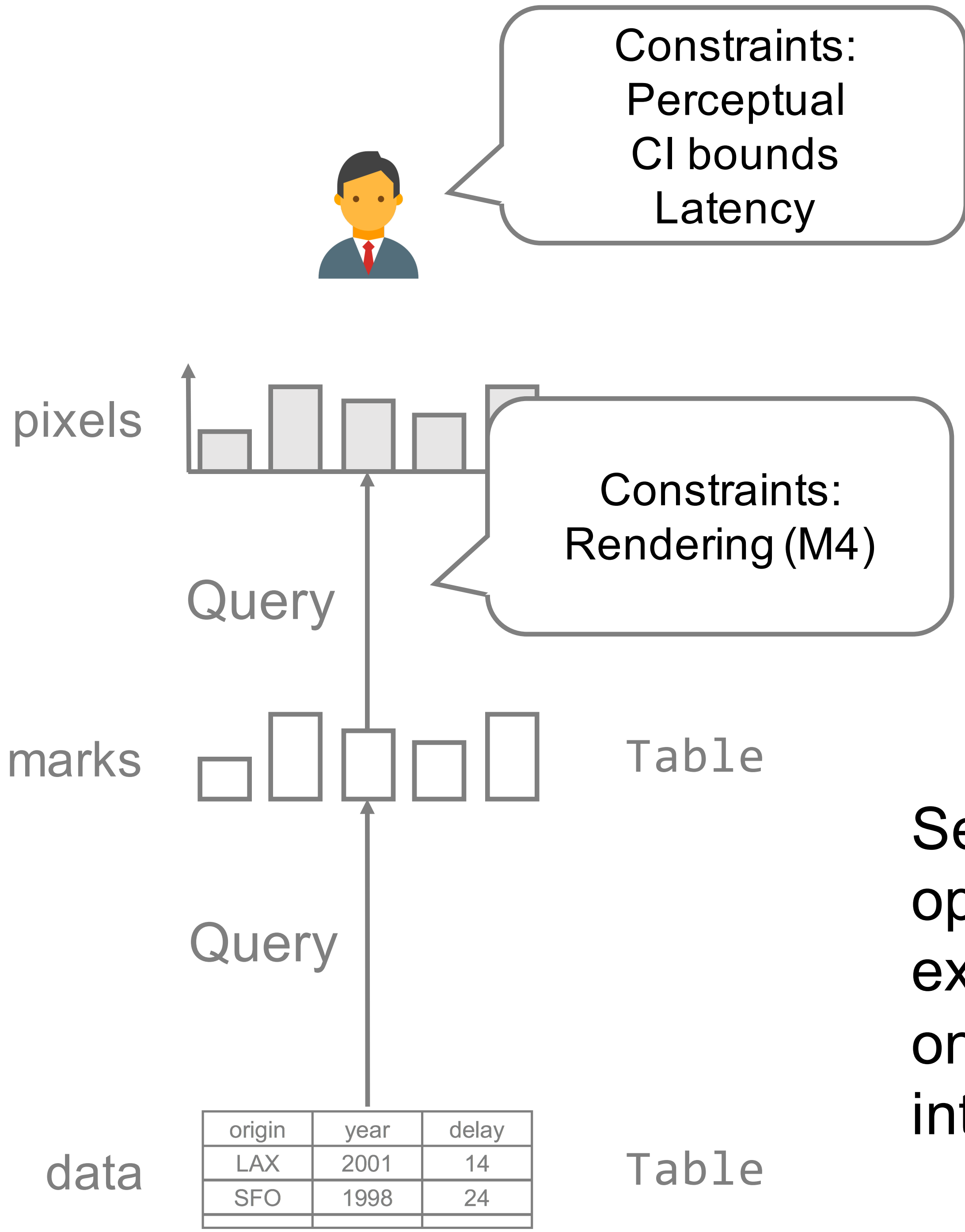


Render on server & send images

Cutting the query plan using the network already expresses many app architectures

Ship query results to client

Ship dataset to client



Several earlier push-down optimizations can now be expressed as constraints on the query output or intermediate results

Example: Single View Interactions

- Using Kyrix as an example

origin	year	delay
LAX	2001	14
SFO	1998	24

```
SELECT x, y, img
  FROM NBA_icons
 WHERE x >= ? AND
       x  < ? AND
       y >= ? AND
       y  < ?
```

At its core, Kyrix can be expressed as parameterized filter queries!

Sufficient to infer RTree indexes and caching

```
SELECT x, y, img
  FROM NBA_icons, viewport vp
 WHERE x >= vp.minx AND
       x <  vp.maxx AND
       y >= vp.miny AND
       y <  vp.maxy
```

Q_1

We can remove the “?”s by “relational-izing” the current viewport. This gives us freedom to redefine vp as a view

origin	year	delay
LAX	2001	14
SFO	1998	24

viewport(minx, maxx, miny, maxy)

user interaction as data

Q_1

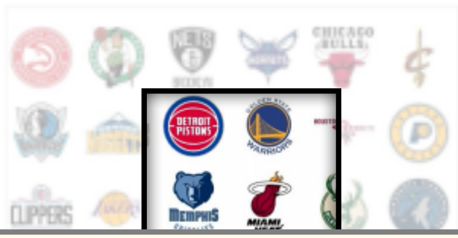
```
SELECT x, y, img
  FROM NBA_icons, last_vp vp
 WHERE x >= vp.minx AND
       x <  vp.maxx AND
       y >= vp.miny AND
       y <  vp.maxy

last_vp = SELECT *
            FROM viewport
        ORDER BY tstamp DESC
           LIMIT 1
```

origin	year	delay
LAX	2001	14
SFO	1998	24

viewport(minx, maxx, miny, maxy, **t**)

user interaction as data



```
SELECT x, y, img
  FROM NBA_icons, last_vp vp
 WHERE x >= vp.minx AND
```

Manipulating viewport view definition enables historical replay, undo, ... for free!

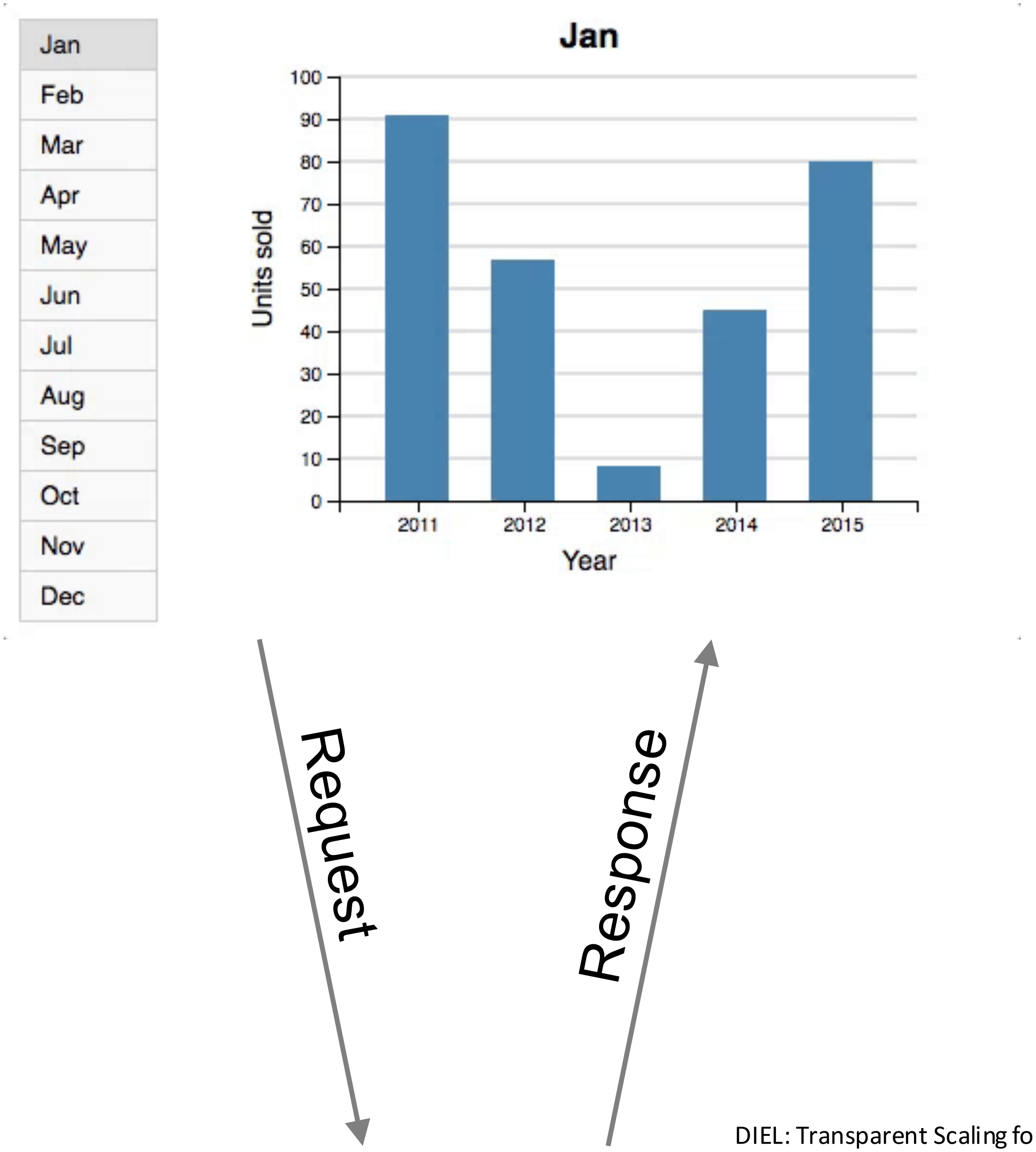
origin	year	delay
LAX	2001	14
SFO	1998	24

viewport(minx, maxx, miny, maxy, **t**)

user interaction as data

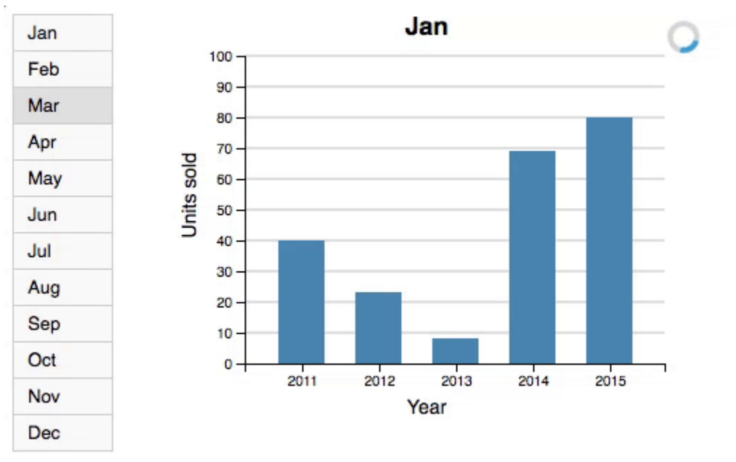
Another Benefit of Relationalizing

This slide contains animation. See powerpoint slides to see animation

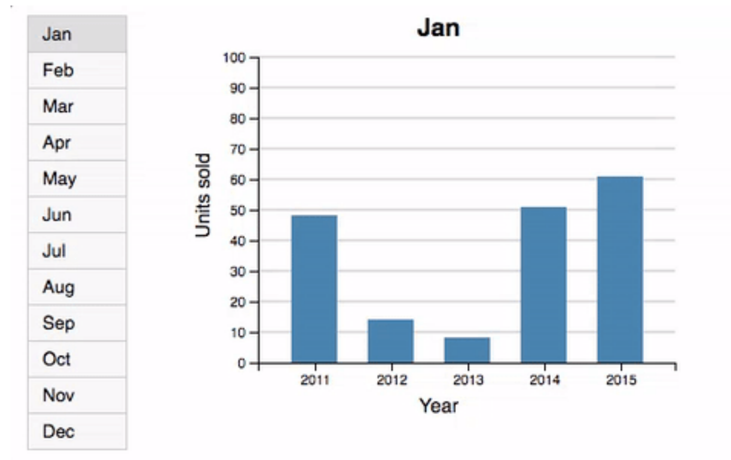


But what if requests take time?

This slide contains animation. See powerpoint slides to see animation



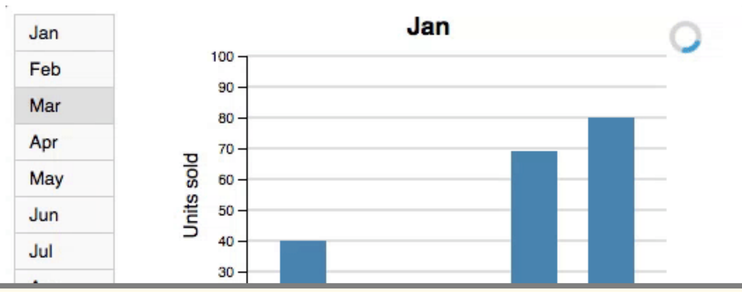
No CC

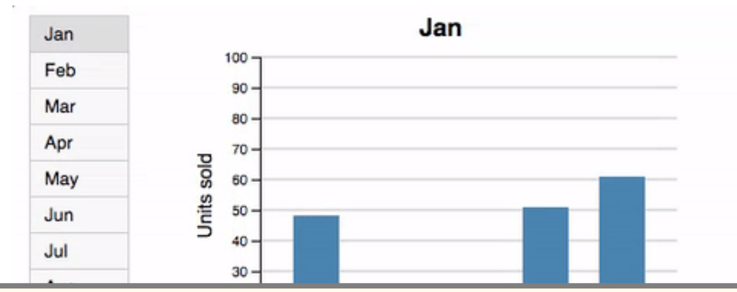


Serial Order

Historical Small Multiples

All of these are sensible choices for a designer when dealing with latencies.



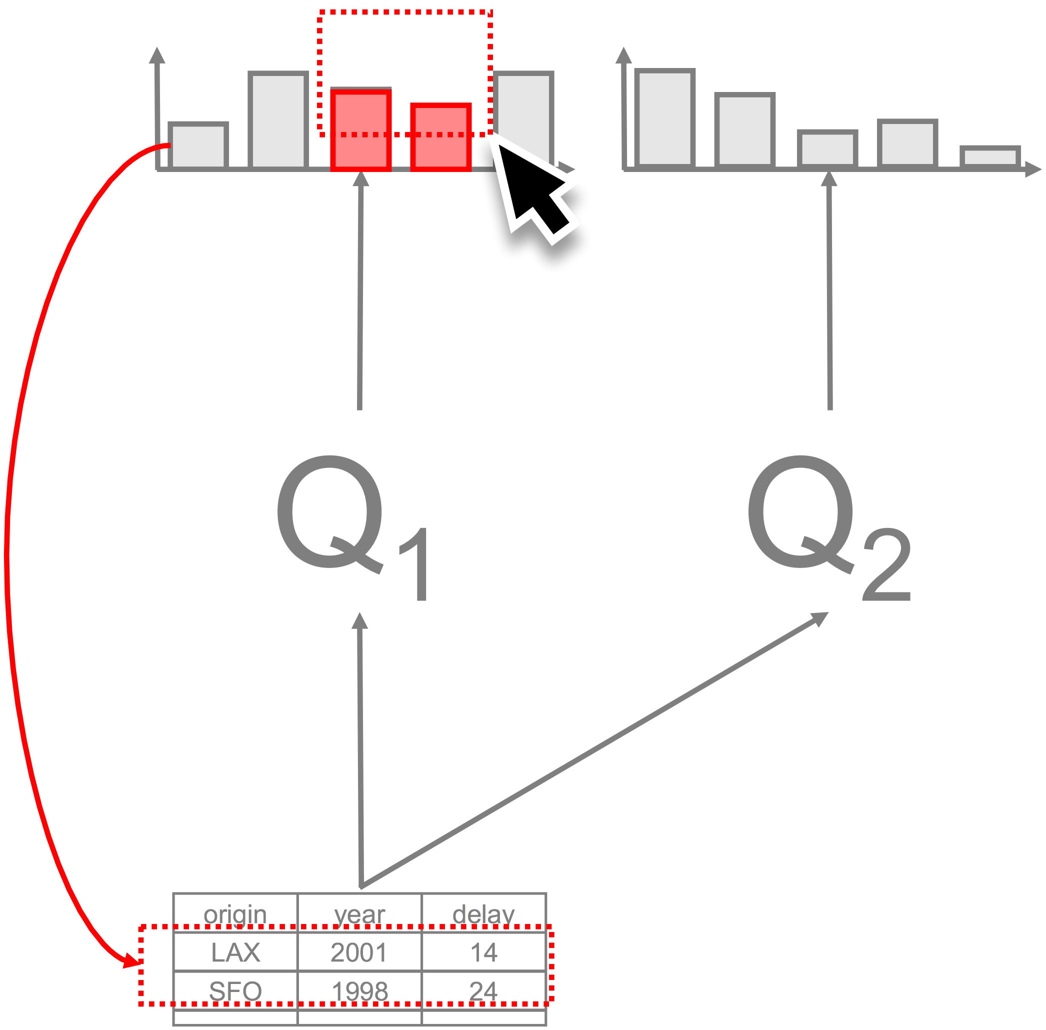


Picking concurrency control
for interaction design is simple by
relationalizing user inputs & responses

designer when dealing
with latencies.

Historical Small Multiples

Example: Multi-view Interactions

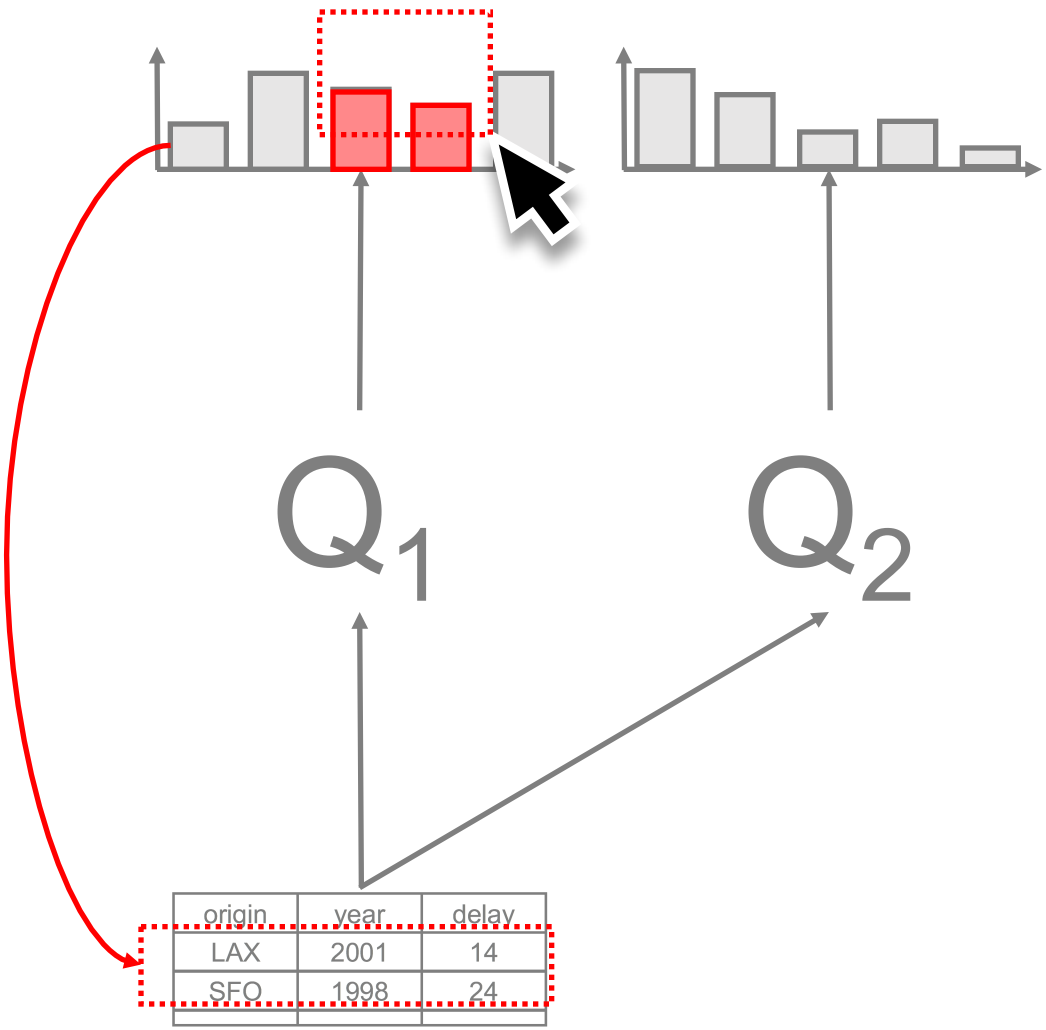



```
Q2 = SELECT x, f(y)
     GROUP BY x
```

User selects bars in Q1's chart...

Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas19

Example: Multi-view Interactions

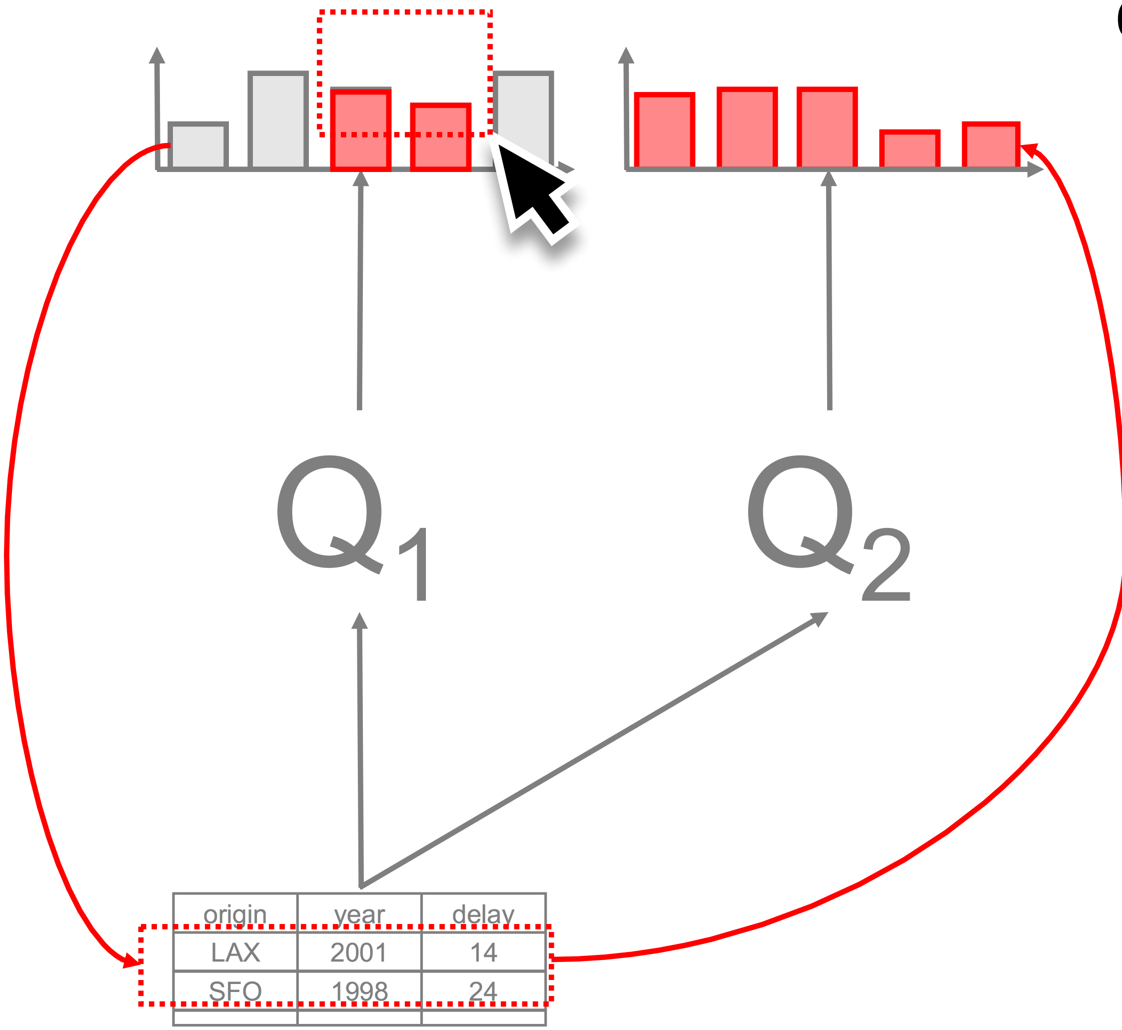



```
Q2 = SELECT x, f(y)
     WHERE id∈
     GROUP BY x
```

Update Q2's chart by adding the WHERE clause to Q2.

Many apps do this by manipulating SQL string literals to construct the query!

Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19



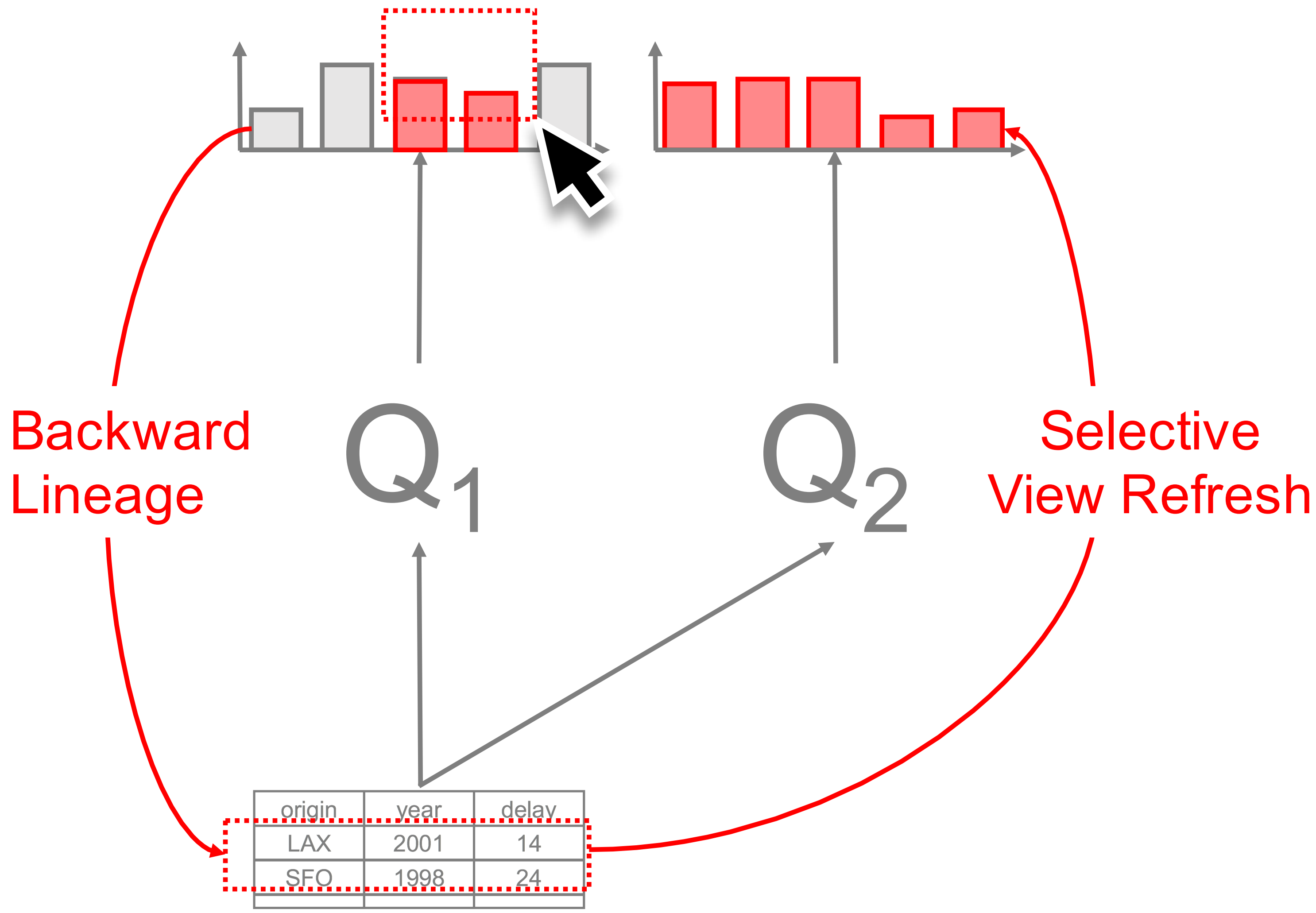

```
Q2 = SELECT x, f(y)
     WHERE id∈
     GROUP BY x
```

Syntactic manipulation rather than logical spec 👎

Can do better?

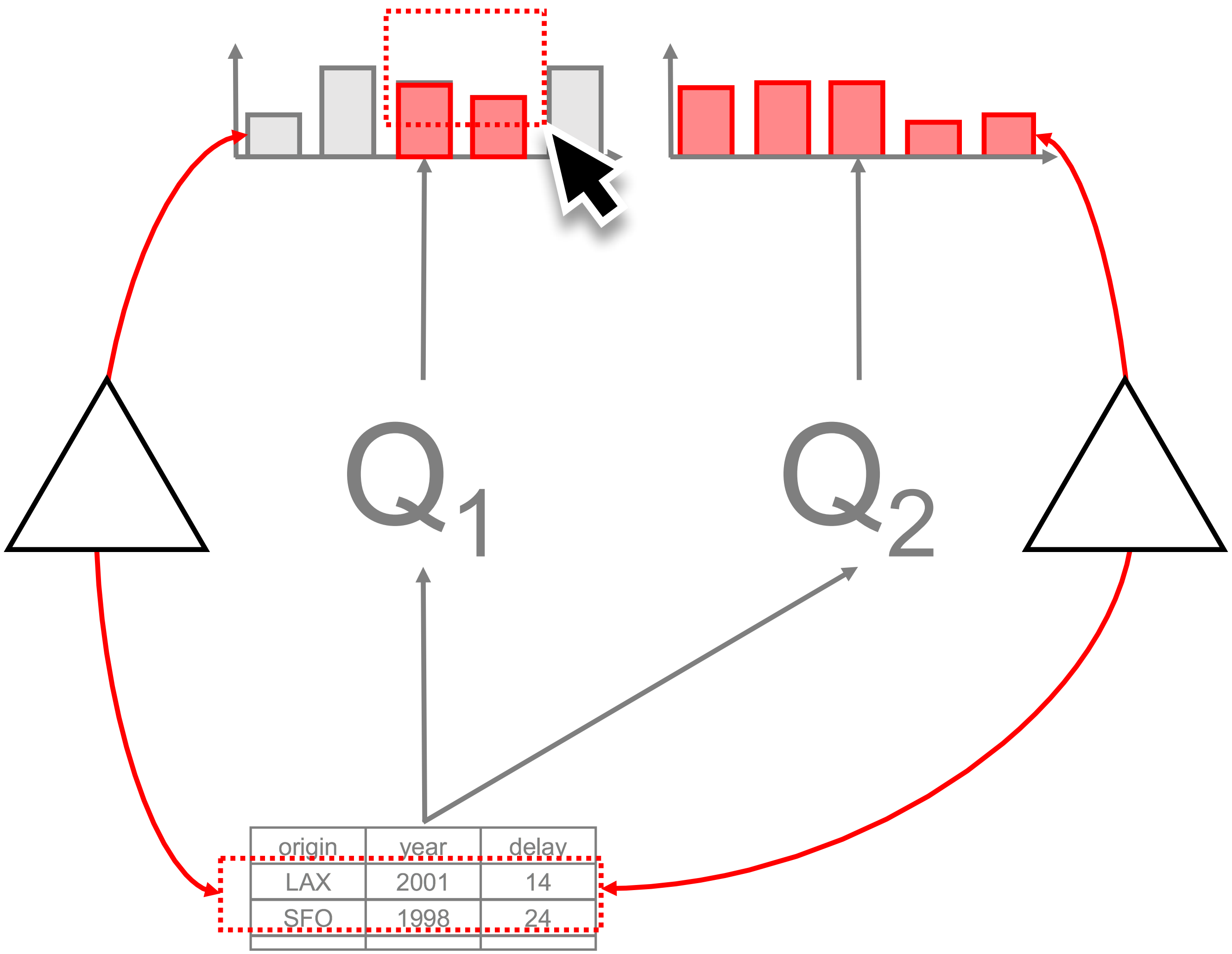
Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas19

Smoke: Interaction as Lineage



Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19

Smoke: Interaction as Lineage



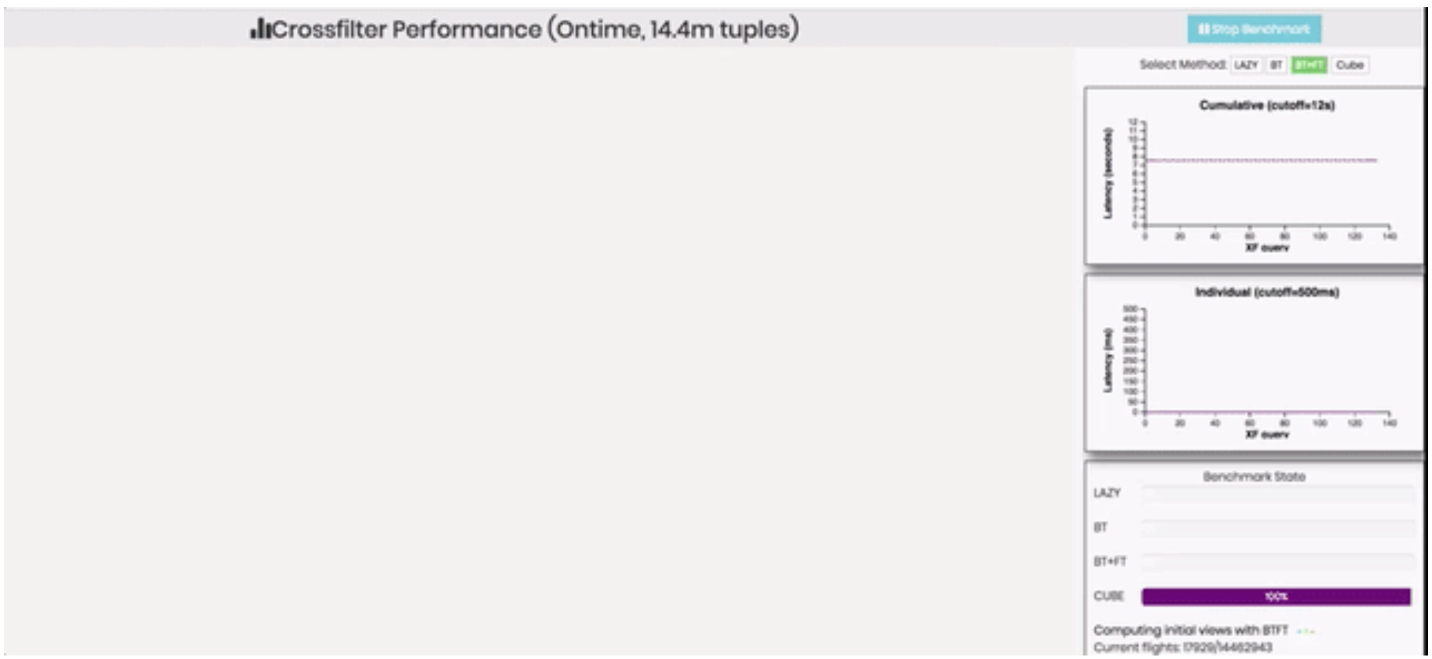
Builds lineage indexes while running Q_1 and Q_2 w/ low overhead

Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19

This slide contains animation. See powerpoint slides to see animation

Smoke: Interaction as Lineage

Lineage enables <100ms interactivity
Avoids data cube precomputation (mins or hrs)



Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19

Smoke: Interaction as Lineage

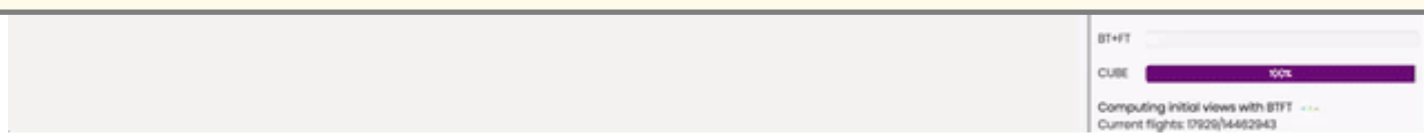
Lineage enables <100ms interactivity

Benefits

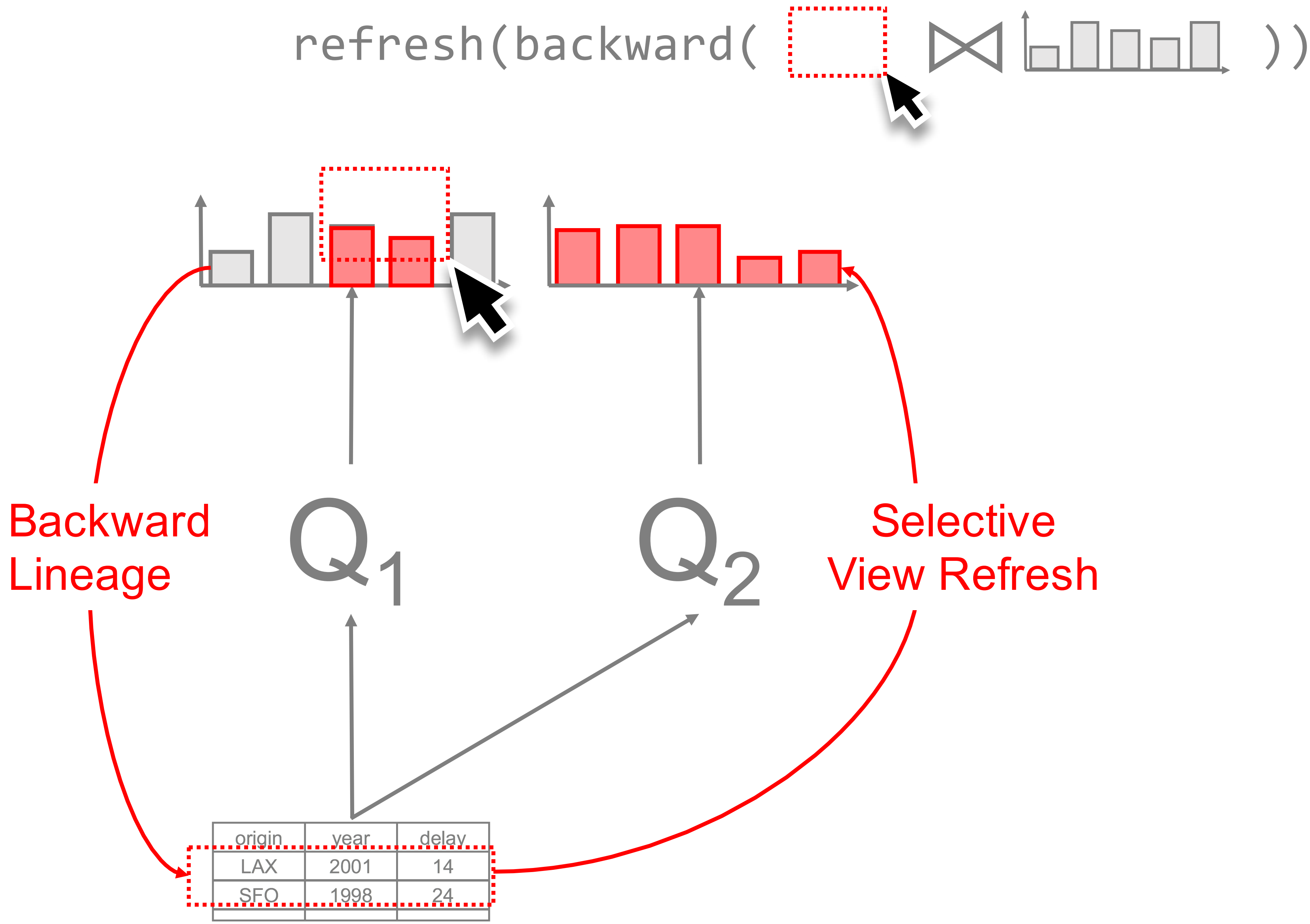
Any visualization expressible as lineage

(most coordinated visualizations)

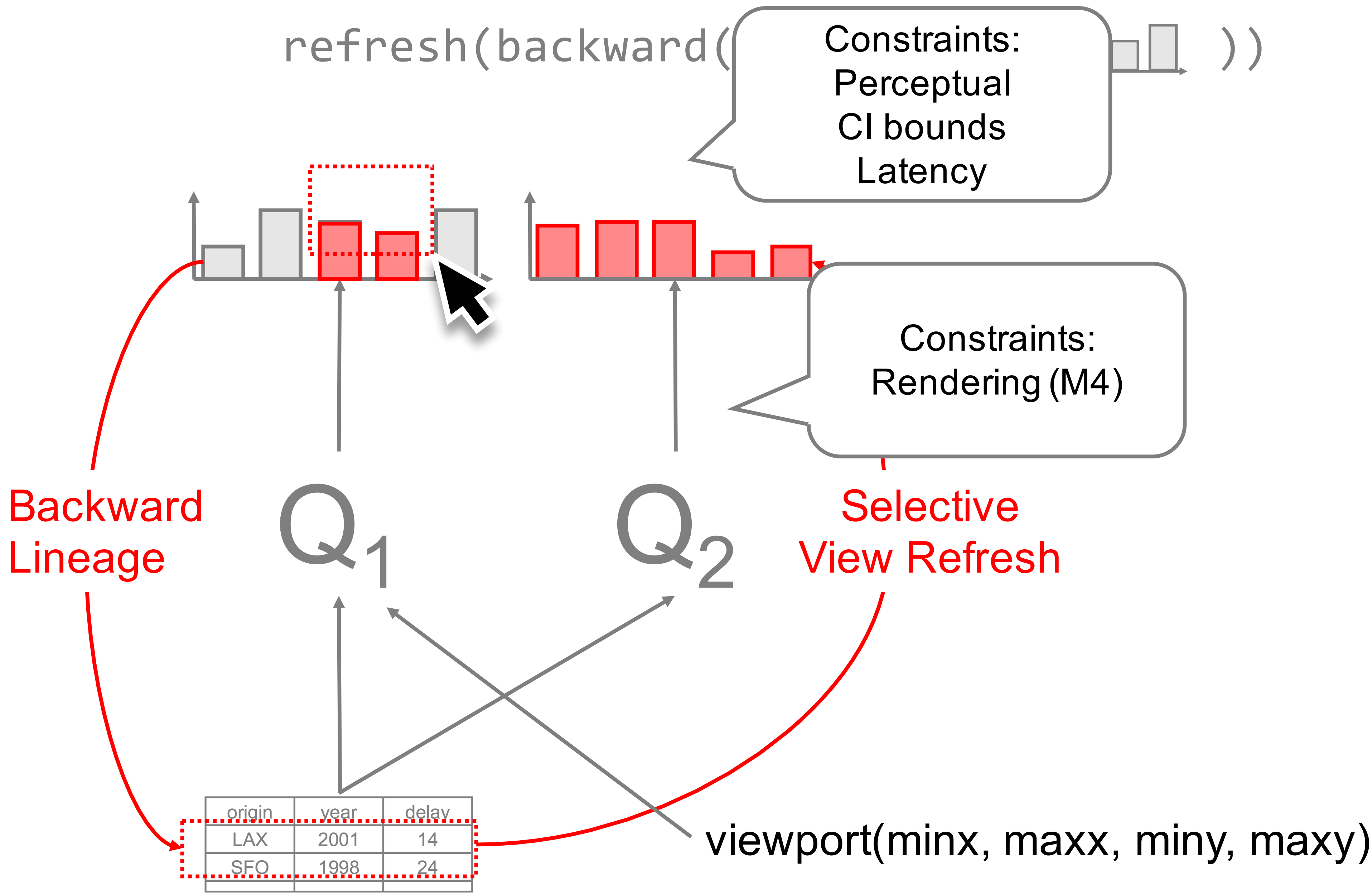
can be optimized *automatically*



Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19



Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas19



Smoke: Fine-grained Lineage at Interactive Speeds. Psallidas 19

Constraints

interaction(vis(database))

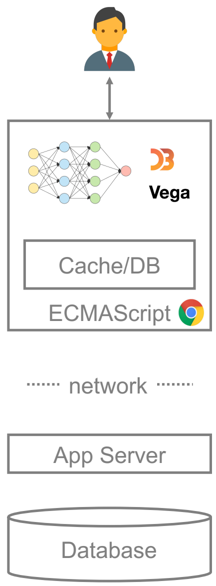
SQL(Lineage(SQL))

Stepping Back

Hacking entire SVD stack is hard
Programming API is important
Users often have existing data, analyses, designs

Wins are from moving up in semantics
Data flow mechanisms for execution
Higher level semantics for optimization

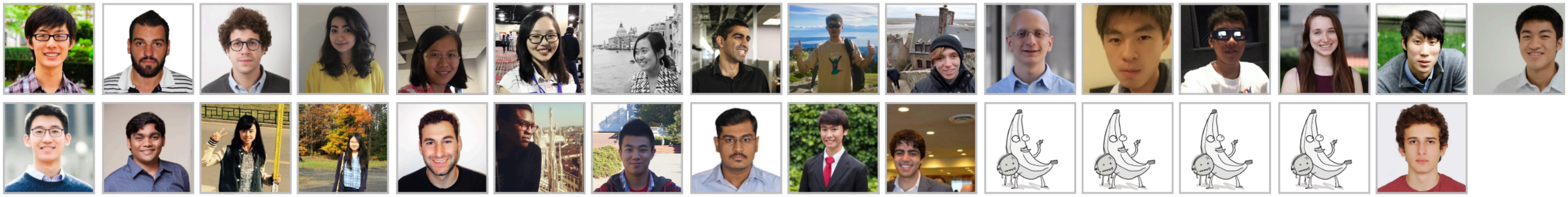
End-to-end relational approach needs to draw from...
Hierarchical models (for layout, subgraphs, etc)
Second order logic (changing group-by attrs)
Ordered relations (most vis is ordered)
...



Open Problem

Algebra to compose

data + interaction + design + task



DVMS

eugenewu.net

Google amazon NSF

Data Science Institute
COLUMBIA UNIVERSITY