Enabling Data Science for the Majority

DataSpread zenvisage GESTALT
OrpheusDB Datamaran Populace

Aditya Parameswaran
Assistant Professor
University of Illinois

http://data-people.cs.illinois.edu

Many many contributors!

DataSpread zenvisage GESTALT

OrpheusDB Datamaran Populace

- **PIs**: Kevin Chang, Amol Deshpande, Karrie Karahalios, Aaron Elmore, Sam Madden (Spanning Illinois, Chicago, MIT, UMD)
- **PhD Students**: Mangesh Bendre, Akash Das Sarma, Yihan Gao, Silu Huang, Doris Lee, Stephen Macke, Sajjadur Rahman, Tarique Siddiqui, Tana Wattanawaroon, Doris Xin, Liqi Xu
- **MS Students**: Ayush Jain, Vipul Venkataraman, Chao Wang, Ed Xue, Paul Zhou, …
- **Many Undergrads!**

Adobe TOYOTA SIEBEL ENERGY INSTITUTE NIH NIH Big Data to Knowledge (BD2K) NSF

It was the year 2013 …

Many of us (the database community) were doing the exact same thing!

The "99%" of Data Analytics Needs

So far, focused on the data analytics needs of the 1%

- Companies w/ **massive data**, **resources** & **know-how**

Ignoring the 99%:

- scientists
- small business owners
- statistical analysts
- journalists
- consultants, …

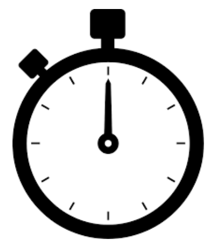
Our research has been focused on easing the burden of data analytics for the 99%

So what were their frustrations?

What about the Needs of the 99%?

The bottleneck is not one of **scale…**

but is actually the **"humans-in-the-loop"**



Human Time

Cognitive Load

Analysis Skills

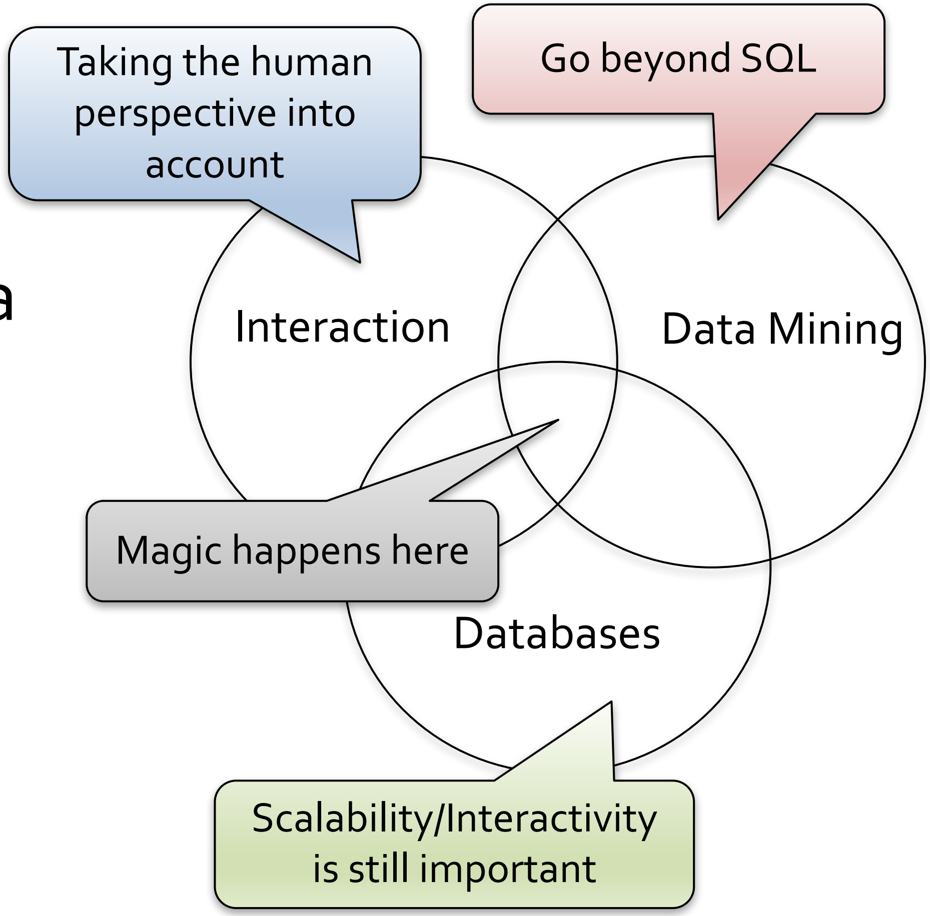
From "Big data and and its Technical Challenges", CACM 2014

For big data to fully reach its potential, we need to consider scale not just for the system but also from the perspective of humans. We have to make sure that the end points—humans—can properly "absorb" the results of the analysis and not get lost in a sea of data.

Need of the hour:
Human-In-the-Loop Data Analytics Tools

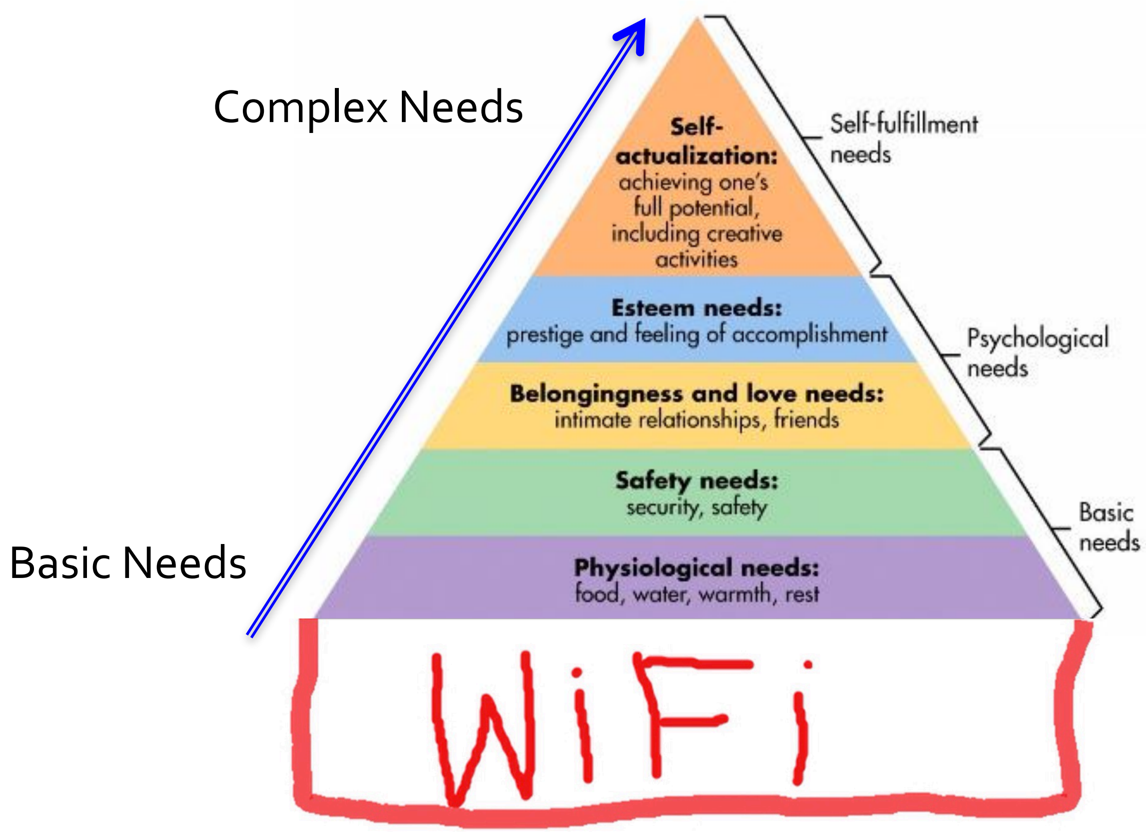
HILDA tools:

- treat both humans and data as **first-class citizens**
- reduce human **labor**
- minimize **complexity**

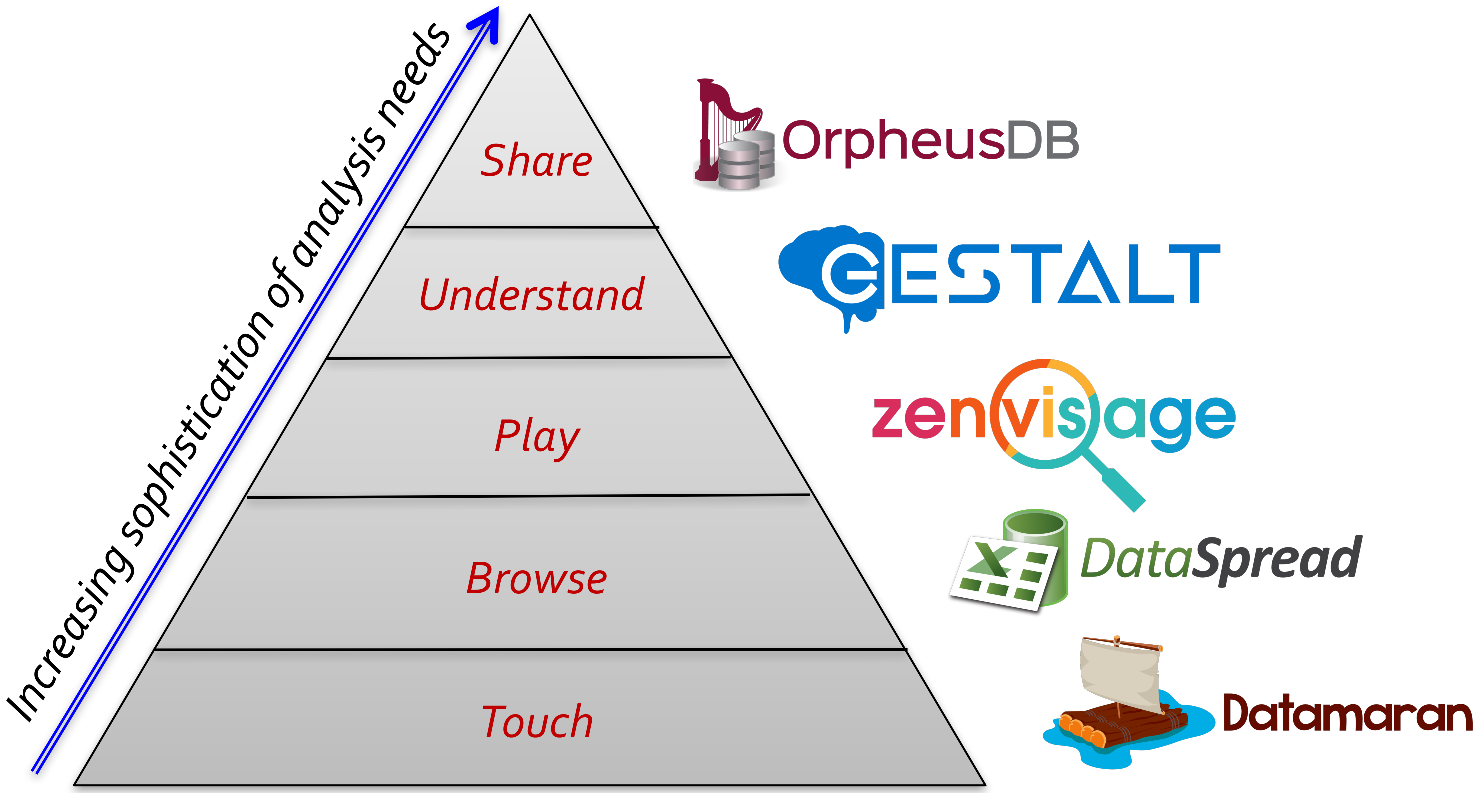


A Maslow's Hierarchy for HILDA

Background: Maslow developed a theory for what motivates individuals in 1943; highly influential



A Maslow's Hierarchy for HILDA



Browse & Explore: DataSpread

DataSpread is a **spreadsheet-database hybrid**:

Goal: Marrying the flexibility and ease of use of spreadsheets with the scalability and power of databases

Enables the “99%” with large datasets but limited prog. skills to open, touch, and examine their datasets

http://dataspread.github.io

[VLDB’15,VLDB’15,ICDE’16]

Play and View:

Zenvisage is **effortless visual exploration tool.**

Goal: "fast-forward" to visual patterns, trends, without having analyst step through each one individually

Enables individuals to play with, and extract insights from large datasets at a fraction of the time.

http://zenvisage.github.io

[VLDB'17, CIDR'17, VLDB'16,VLDB'15, VLDB'14 x 2]

Collaborate and Share: OrpheusDB

OrpheusDB is a tool for **managing dataset versions** with a database

Goal: building a versioned database system to reduce the burden of recording datasets in various stages of analysis

Enables individuals to collaborate on data analysis, and share, keep track of, and retrieve dataset versions.

http://orpheus-db.github.io

(also part of datahub: a collab. analysis system w/ MIT & UMD)

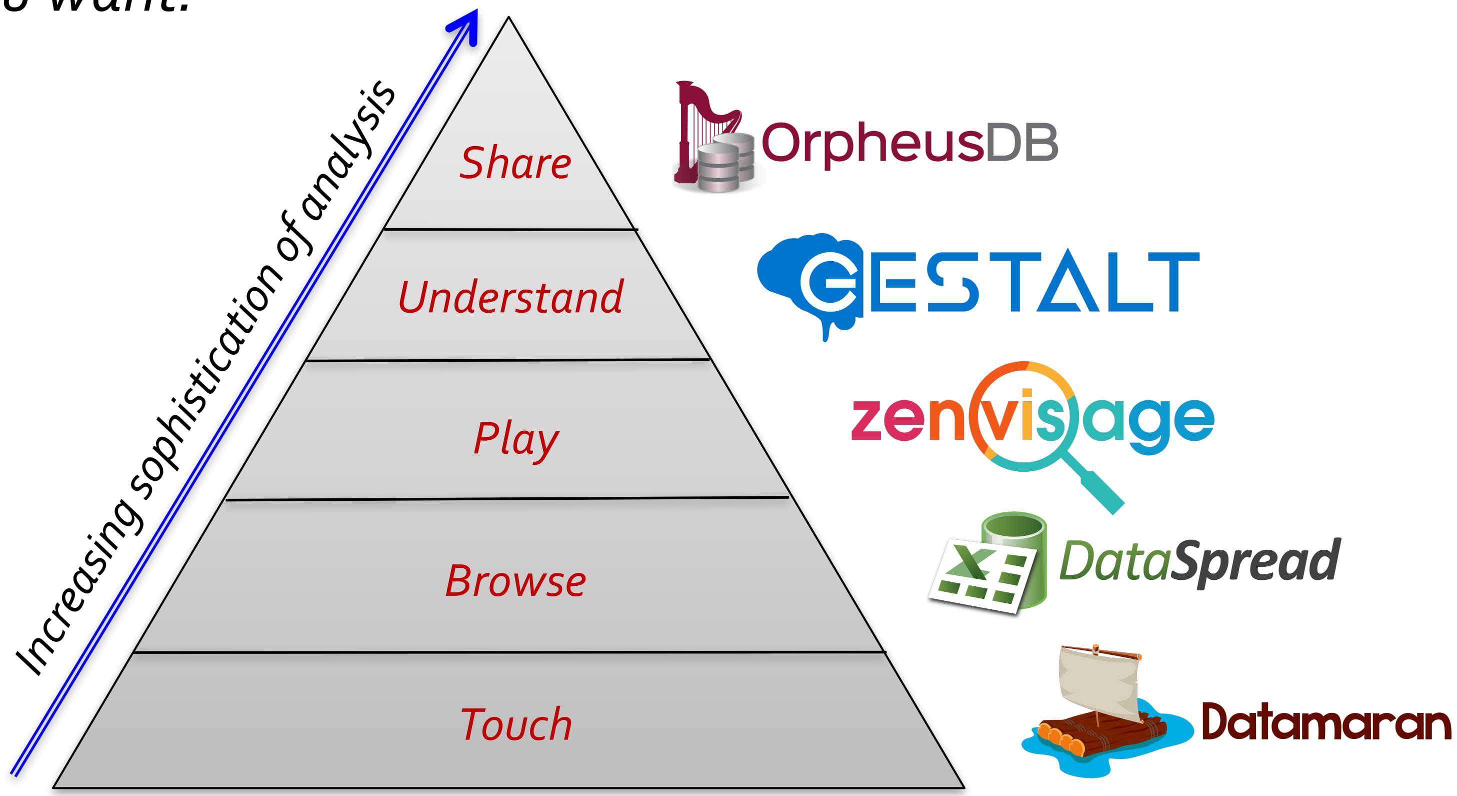
[VLDB’17, SIGMOD’17, VLDB’16,VLDB’15 x 2, TAPP’15, CIDR’15]

This talk

About 10 minutes per system:

overview + architecture + one key technical challenge

***Common theme**: if you torture databases enough, you can get them to do what you want!*



DataSpread

Motivation

Most of the people doing ad-hoc data manipulation and analysis use spreadsheets, e.g., Excel

Why?

- *Easy to use: direct manipulation*
- *Built-in visualization capabilities*
- *Flexible: schema-free*

But Spreadsheets are Terrible!

- *Slow*
  - single change ➔ wait minutes on a 10,000 x 10 spreadsheet
  - can't even open a spreadsheet with >1M cells
  - speed by itself can prevent analysis
- *Tedious + not Powerful*
  - filters via copy-paste
  - only FK joins via VLOOKUPs; others impossible
  - even simple operations are cumbersome
- *Brittle*
  - sharing excel sheets around, no collab/recovery
  - using spreadsheets for collaboration is painful and error-prone

Let's turn to Databases

Databases are:

- ~~*Slow*~~ Scalable
- ~~*Tedious + not Powerful*~~ Powerful and expressive (SQL)
- ~~*Brittle*~~ Collaboration, recovery, succinct

So why not use databases?

Well, for the same reason why spreadsheets are so useful:

- ~~*Easy to use*~~ Not easy to use
- ~~*Built-in visualization*~~ No built-in visualization
- ~~*Flexible*~~ Not flexible

Combining the benefits of spreadsheets and databases

DataSpread

Spreadsheet as a frontend interface

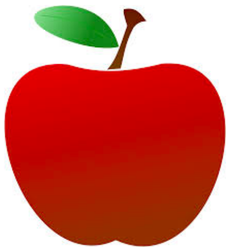
Databases as a backend engine

Result: retain the benefits of both!

But it's not that simple…

Different Ideologies

Feature	Databases	Spreadsheets
Data Model	Schema-first	Dynamic/No Schema
Addressing	Tuples with PK	Cells, using Row/Col
Presentation	Set-oriented, no such notion	Notion of current window, order
Modifications	Must equal queries	Can be done at any granularity
Computation	Query at a time	Value at a time



Due to this, the integration is not trivial…

First Problem: Representation

Q: how do we represent spreadsheet data?

	A	B	C	D	E	F
1	snp	chromoso	position	minor	major	
2	rs1208247	1	740857	T	C	
3	rs3094315	1	752566	G	A	
4	rs3131972	1	752721	A	G	
5	rs3115860	1	753406	C	A	
6	rs3131969	1	754182	A	G	
7	rs1048488	1	760912	G	A	
8	rs3115850	1	761147	A	G	
9	rs2286139	1	761732	C	T	
10	rs1256203	1	768448	A	G	
11	rs1212481	1	776546	G	A	
12	rs2980319	1	777122	A	T	
13	rs4040617	1	779322	G	A	
14	rs2980300	1	785989	A	G	
15	rs1124077	1	798959	A	G	
16	rs4970383	1	838555	A	C	
17	rs4475691	1	846808	A	G	
18	rs2860985	1	851190	A	G	
19	rs1806509	1	853954	C	A	
20	rs7537756	1	854250	G	A	
21	rs1330298	1	861808	A	G	
22	rs4040604	1	863124	C	A	
23	rs2340587	1	864938	G	A	
24	rs2857669	1	870645	G	A	
25	rs1110052	1	873558	C	A	
26	rs7523549	1	879317	A	G	
27	rs3748592	1	880238	A	G	
28	rs3748593	1	880390	A	C	
29	rs2272756	1	882033	A	G	
30	rs2340582	1	882803	A	G	
31	rs4246503	1	884815	A	G	
32	rs3748594	1	886384	A	G	
33	rs3748595	1	887560	A	C	
34	rs3748597	1	888659	T	C	
35	rs1330310	1	891945	A	G	
36	rs1330301	1	894573	G	A	
37	rs2870521	1	900505	C	G	
38	rs3935066	1	900730	G	A	
39	rs6696281	1	903104	A	G	
40	rs2839128	1	904165	A	G	
41	rs2869570	1	904355	A	G	
42	rs2856232	1	904628	A	G	

	A	B	C	D	E	F	G	H
1	bob							
2								
3		sally						steven
4				james				
5							jennifer	
6			charles					
7					dan			
8								
9						alice		
10								
11								
12					rick			
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								
34								
35								
36								
37								
38								
39								
40								
41								
42								

Dense spreadsheets: represent as tables
(Row #, Col1 val, Col2 val, ...)

Sparse spreadsheets: represent as triples
(Row #, Column #, Value)

First Problem: Representation

Q: how do we represent spreadsheet data?

	A	B	C	D	E	F	G	H	I	J	K	L
1	snp	chromoso	position	minor	major					snp	chromoso	position
2	rs1208247	1	740857	T	C		dasas			rs1208247	1	740857
3	rs3094315	1	752566	G	A					rs3094315	1	752566
4	rs3131972	1	752721	A	G					rs3131972	1	752721
5	rs3115860	1	753406	C	A					rs3115860	1	753406
6	rs3131969	1	754182	A	G					rs3131969	1	754182
7	rs1048488	1	760912	G	A					rs1048488	1	760912
8	rs3115850	1	761147	A	G					rs3115850	1	761147
9	rs2286139	1	761732	C	T					rs2286139	1	761732
10	rs1256203	1	768448	A	G					rs1256203	1	768448
11	rs1212481	1	776546	G	A					rs1212481	1	776546
12	rs2980319	1	777122	A	T					rs2980319	1	777122
13	rs4040617	1	779322	G	A					rs4040617	1	779322
14	rs2980300	1	785989	A	G					rs2980300	1	785989
15	rs1124077	1	798959	A	G					rs1124077	1	798959
16	rs4970383	1	838555	A	C					rs4970383	1	838555
17	rs4475691	1	846808	A	G					rs4475691	1	846808
18	rs2860985	1	851190	A	G					rs2860985	1	851190
19	rs1806509	1	853954	C	A					rs1806509	1	853954
20	rs7537756	1	854250	G	A					rs7537756	1	854250
21	rs1330298	1	861808	A	G					rs1330298	1	861808
22	rs4040604	1	863124	C	A					rs4040604	1	863124
23	rs2340587	1	864938	G	A					rs2340587	1	864938
24	rs2857669	1	870645	G	A					rs2857669	1	870645
25	rs1110052	1	873558	C	A					rs1110052	1	873558
26	rs7523549	1	879317	A	G					rs7523549	1	879317
27	rs3748592	1	880238	A	G					rs3748592	1	880238
28	rs3748593	1	880390	A	C					rs3748593	1	880390
29	rs2272756	1	882033	A	G					rs2272756	1	882033
30	rs2340582	1	882803	A	G					rs2340582	1	882803
31	rs4246503	1	884815	A	G					rs4246503	1	884815
32	rs3748594	1	886384	A	G					rs3748594	1	886384
33	rs3748595	1	887560	A	C					rs3748595	1	887560
34	rs3748597	1	888659	T	C					rs3748597	1	888659
35	rs1330310	1	891945	A	G					rs1330310	1	891945
36	rs1330301	1	894573	G	A					rs1330301	1	894573
37	rs2870521	1	900505	C	G					rs2870521	1	900505
38	rs3935066	1	900730	G	A					rs3935066	1	900730
39	rs6696281	1	903104	A	G					rs6696281	1	903104
40	rs2839128	1	904165	A	G					rs2839128	1	904165
41	rs2869570	1	904355	A	G					rs2869570	1	904355
42	rs2856232	1	904628	A	G					rs2856232	1	904628

Can we do even better than the two extremes? **Yes!**

Carve out
dense areas ➔ store as tables,
sparse areas ➔ store as triples

First Problem: Representation

However, even if we only use "tables", carving out the ideal # partitions (min. storage, modif., access)

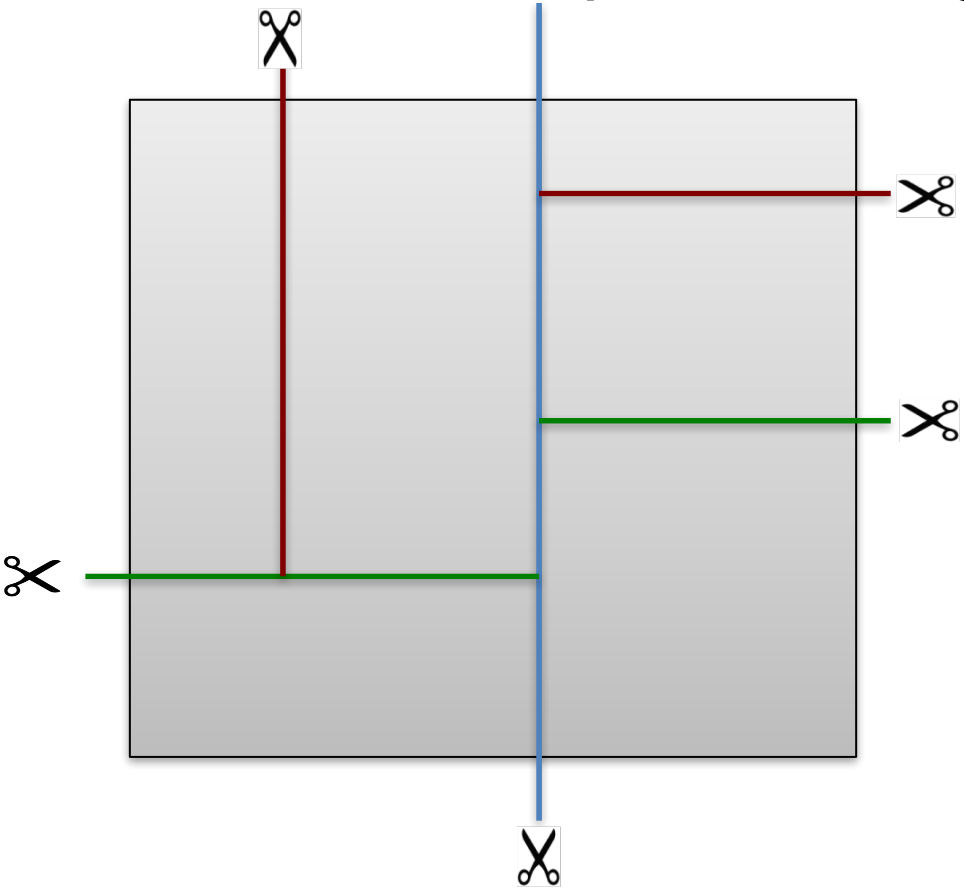
is **NP-Hard**

➔*Reduction from min. edge-length partition of rectilinear polygons*

Thankfully, we have a way out…

Solution: Constrain the Problem

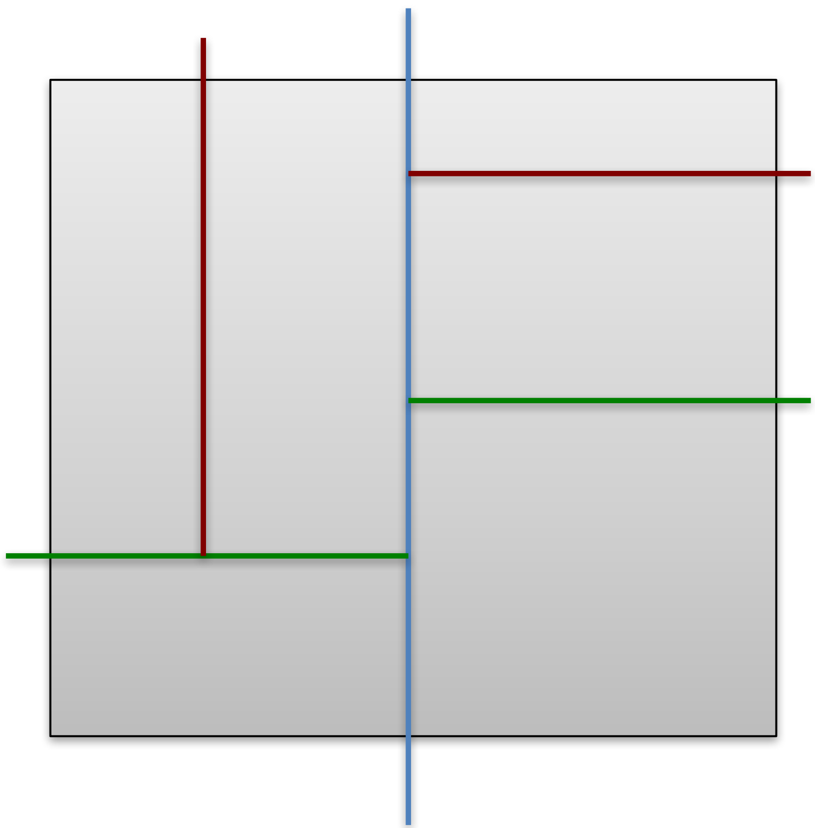
A new class of partitionings: **recursive decomp.**



	A	B	C	D	E	F	G	H	I
1	*	*		*	*	*	*	*	*
2	*	*		*	*	*	*	*	*
3	*	*							
4	*	*						*	*
5								*	*
6	*	*	*	*	*	*		*	*
7	*	*	*	*	*	*		*	*

A very natural class of partitionings!

Solution: Constrain the Problem



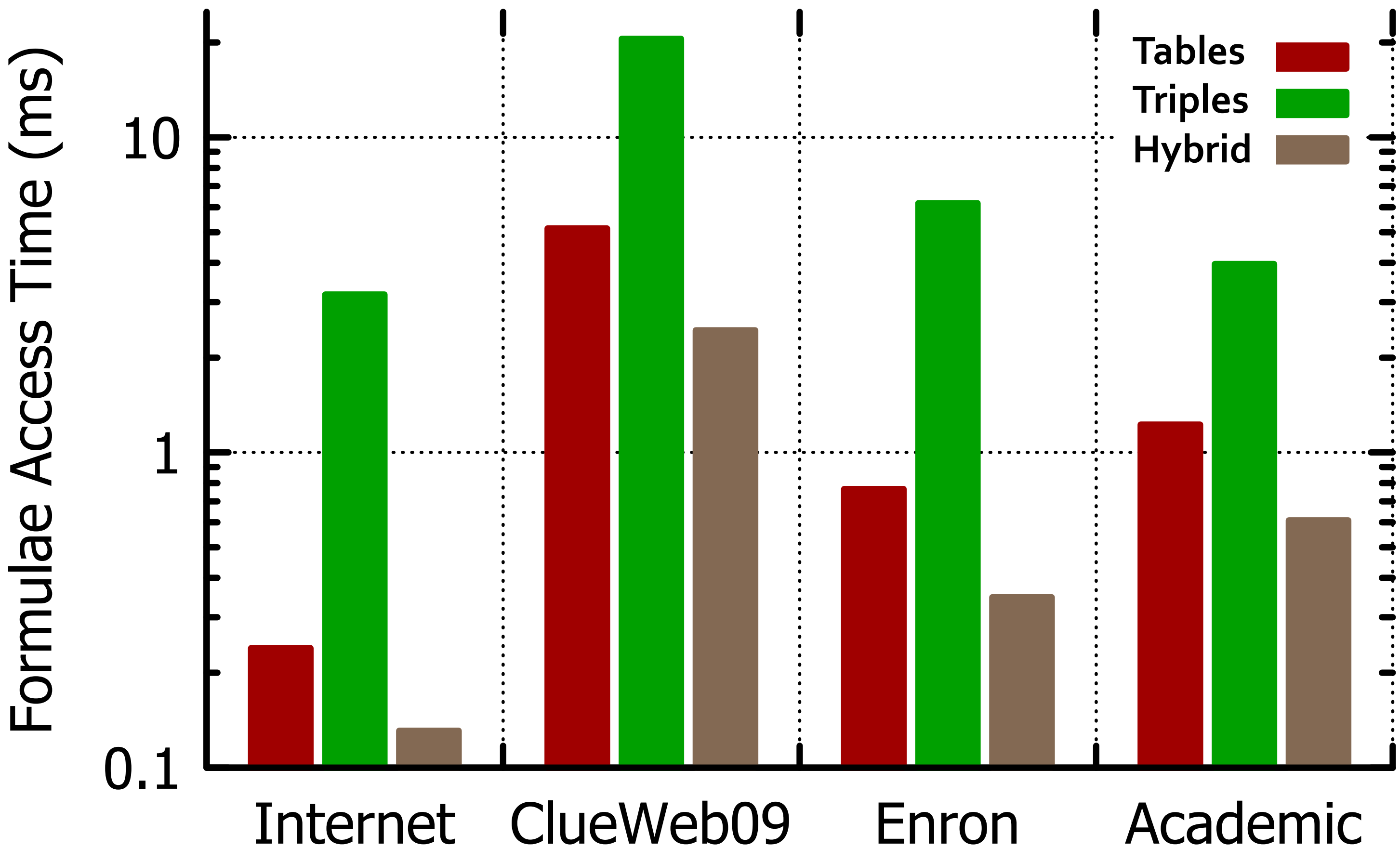
The optimal recursive decomp. partitioning can be found in PTIME using DP

➔ Still **quadratic** in # rows, columns ☹

➔ Merge rows/columns with identical signatures

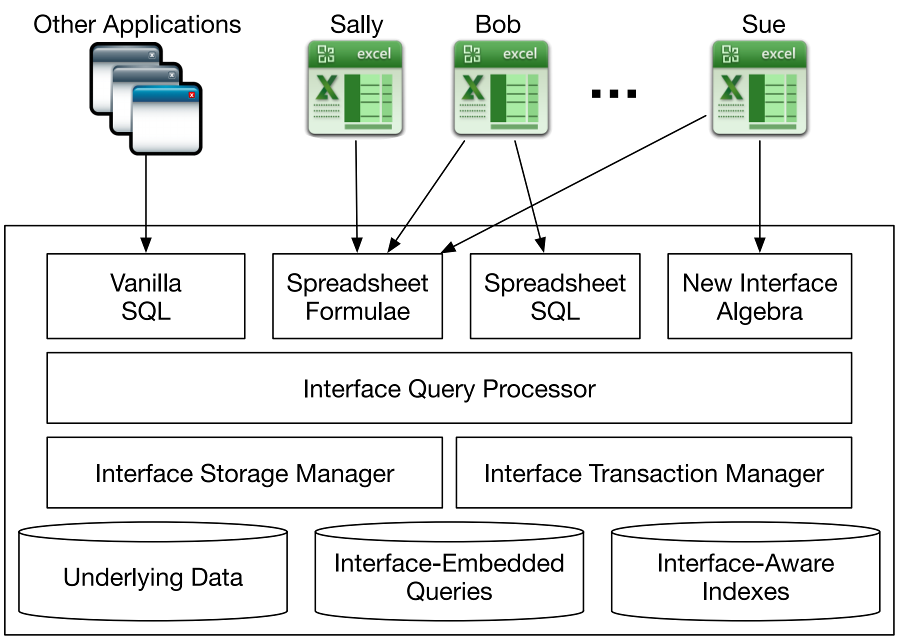
~ the time for a single scan

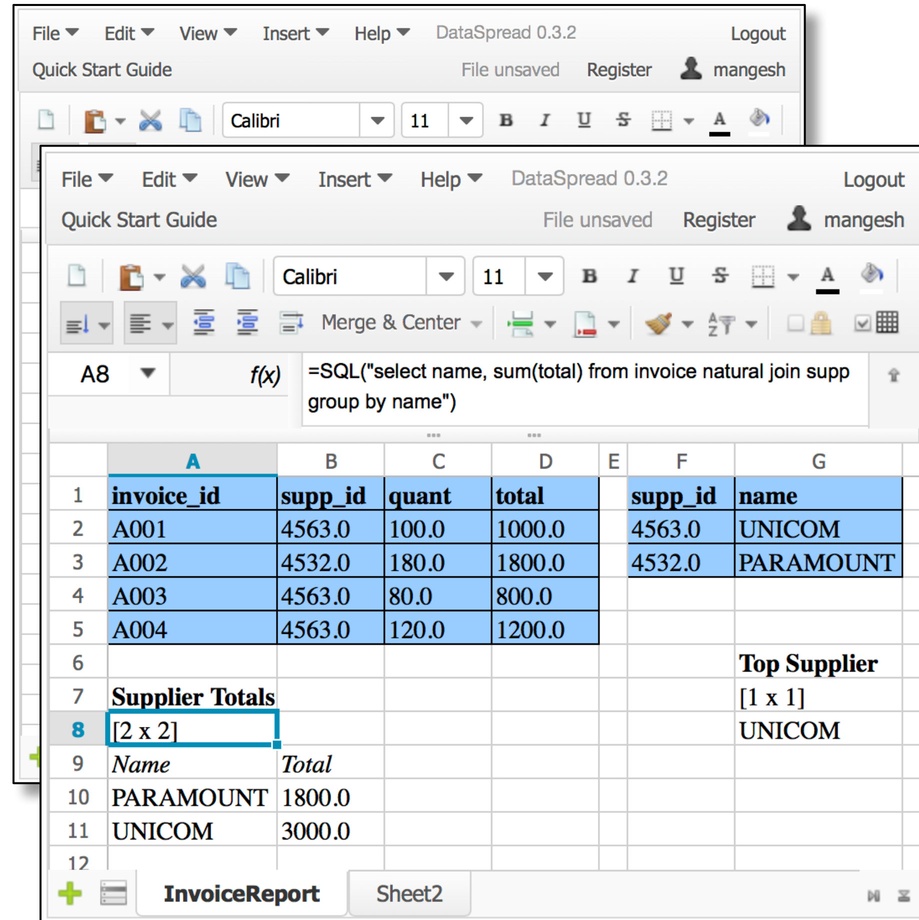
One Sample Result



Up to 30% reduction in storage, 40% reduction in eval time

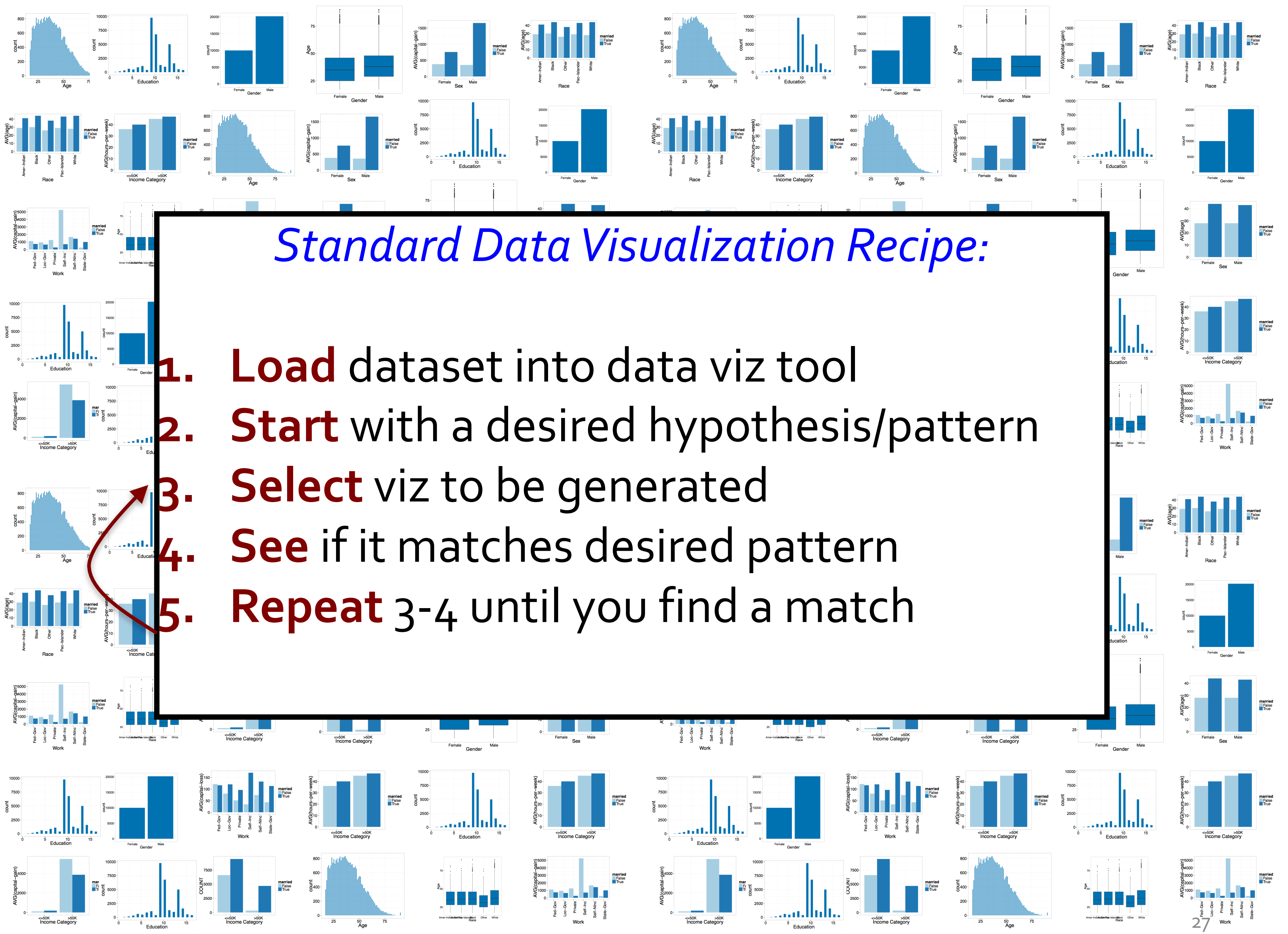
Initial Progress and Architecture





Hopefully bring spreadsheets to the big data age!

zenvisage



Standard Data Visualization Recipe:

1. **Load** dataset into data viz tool
2. **Start** with a desired hypothesis/pattern
3. **Select** viz to be generated
4. **See** if it matches desired pattern
5. **Repeat** 3-4 until you find a match

Laborious and Time-consuming!

Key Issue:

Visualizations can be generated by varying

- data subsets
- visualized attributes

Too many visualizations to look at to find desired visual patterns!

Broadly Applicable

TURN

Carnegie Mellon University
Scott Institute
for Energy Innovation

knoweng

THE DARK
ENERGY SURVEY

- find keywords with similar CTRs to a specific one
- find solvents with desired properties
- find aspects on which two sets of genes differ
- find supernovae with specific patterns

Common theme: **manual labor** for finding desired patterns to test hypotheses, derive insights

Key Insight : **Automation**

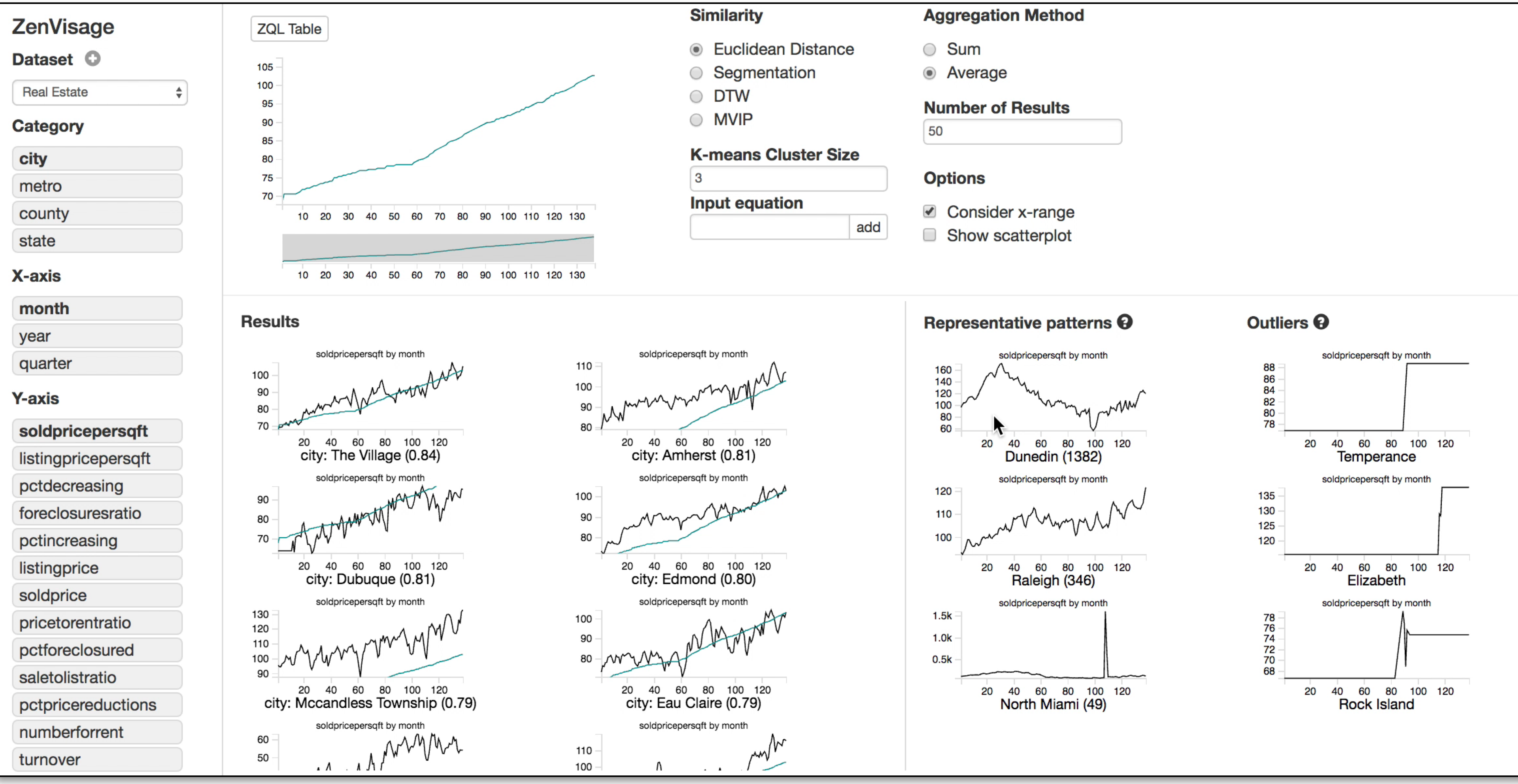
We can automate that!

Desiderata for automation:

- Expressive – specify what you want
- Interactive – interact with results, cater to non-programmers
- Scalable – get interesting results quickly

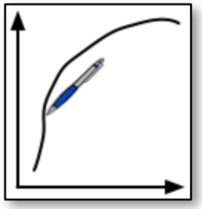
Enter Zenvisage:
(zen + envisage: to effortlessly visualize)

Overview

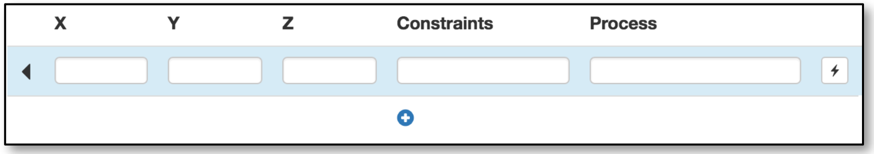

ZenVisage
Dataset
Real Estate
Category
city
metro
county
state
X-axis
month
year
quarter
Y-axis
soldpricepersqft
listingpricepersqft
pctdecreasing
foreclosuresratio
pctincreasing
listingprice
soldprice
pricetorentratio
pctforeclosured
saletolistratio
pctpricereductions
numberforrent
turnover
ZQL Table
Similarity
Euclidean Distance
Segmentation
DTW
MVIP
K-means Cluster Size
3
Input equation
add
Aggregation Method
Sum
Average
Number of Results
50
Options
Consider x-range
Show scatterplot
Results
city: The Village (0.84)
city: Amherst (0.81)
city: Dubuque (0.81)
city: Edmond (0.80)
city: Mccandless Township (0.79)
city: Eau Claire (0.79)
Representative patterns
Dunedin (1382)
Raleigh (346)
North Miami (49)
Outliers
Temperance
Elizabeth
Rock Island
soldpricepersqft by month

Zenvisage: Two Modes

- **First Mode**: Interactions, drawing, drag-and-drop
  - Simple needs
  - Starting point / context



- **Second Mode**: the Zenvisage Query Language (ZQL)
  - Sophisticated needs
  - Multiple steps

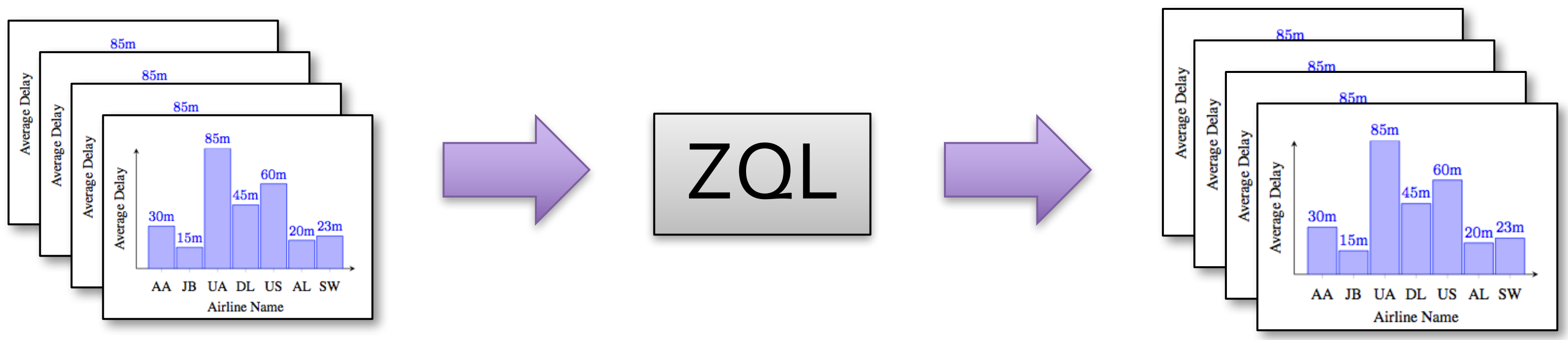


Can switch back and forth, as user needs evolve

Both modes developed after many discussions with potential users

ZQL: High Level Overview

ZQL is a viz exploration language



- Captures four key operations on viz collections

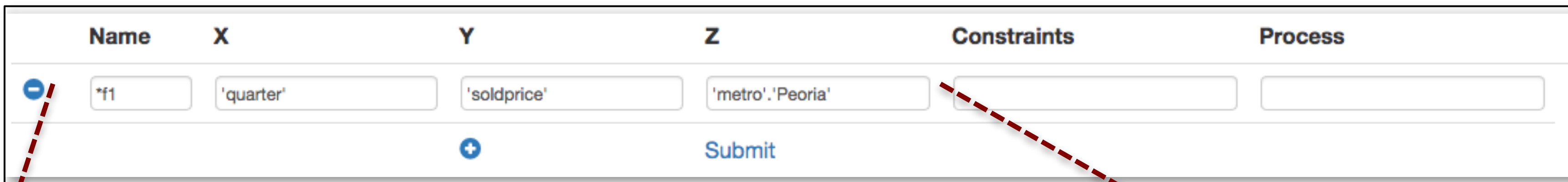
Compose *Filter* *Compare* *Sort*

- Incorporates data mining primitives

Powerful; formally demonstrated "completeness"

ZQL: A Bird's Eye View

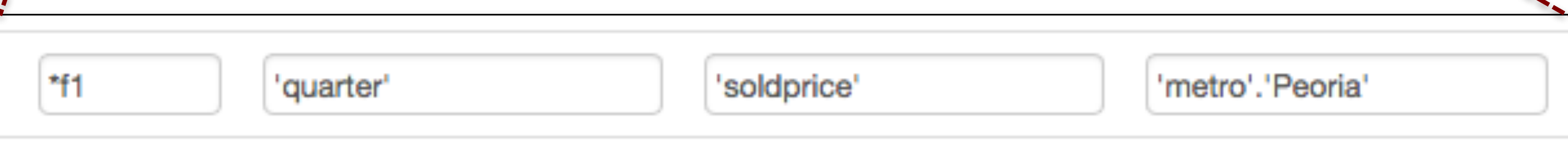
Name X Y Z Constraints Process



Output spec and identifiers

Composition of visualizations, often using values from previous steps

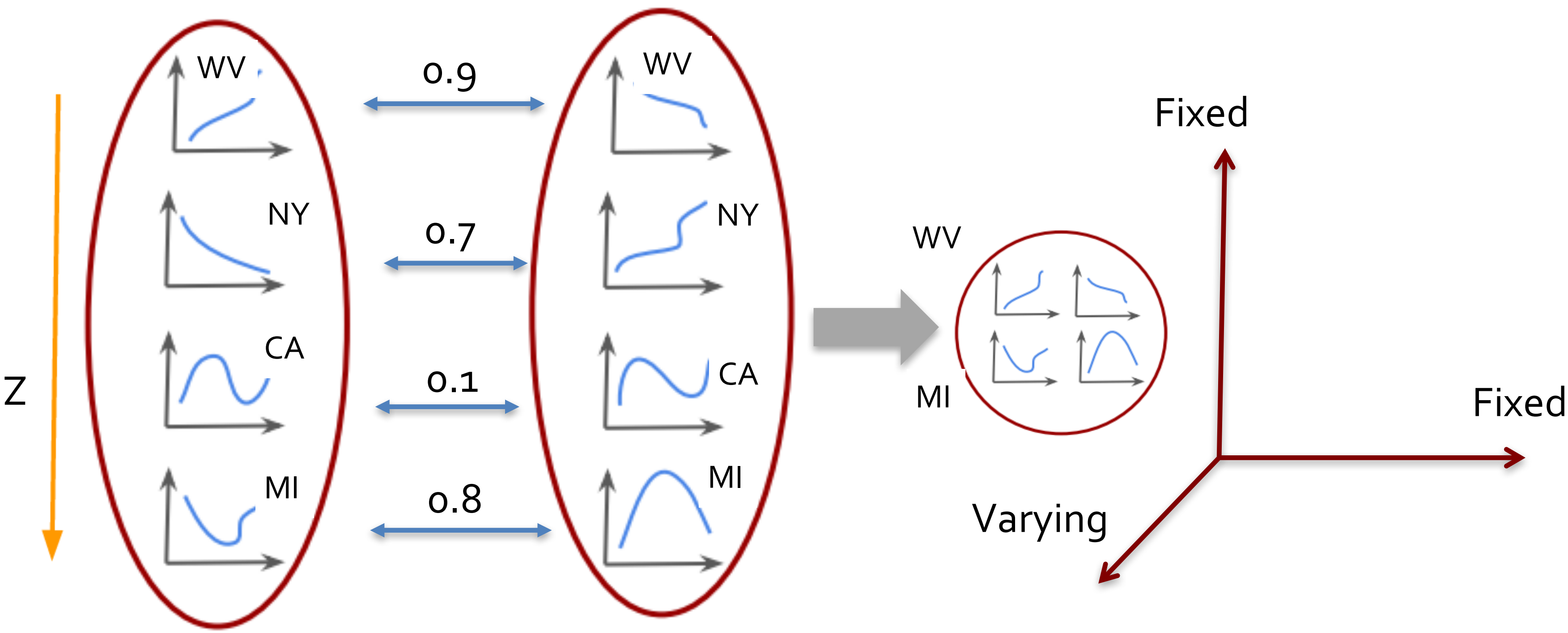
Sorting, comparing, and filtering visualizations



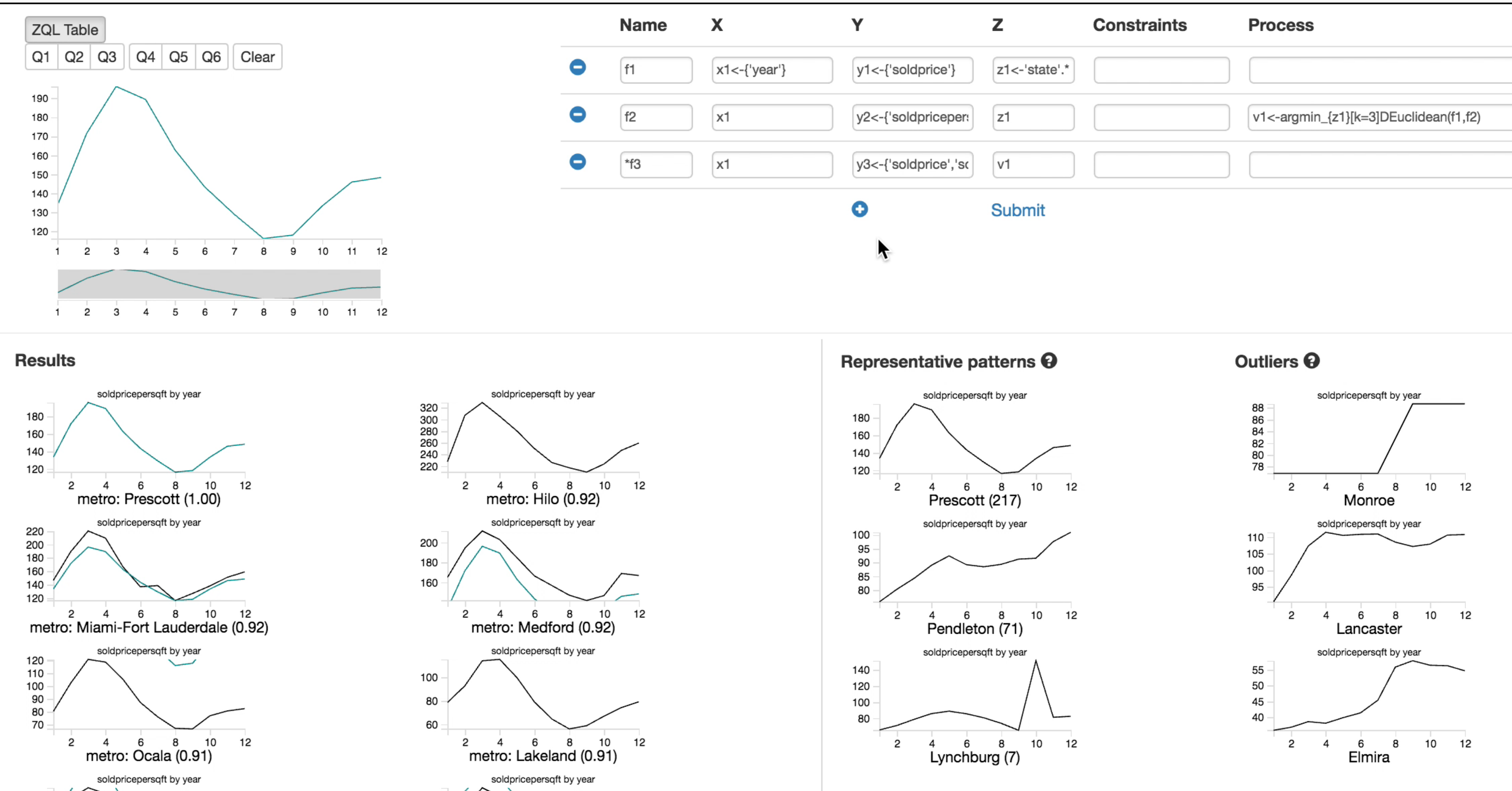
Example 1: Comparisons

Find the states where the *soldprice* trend is most similar to (or most different from) the *soldpricepersqft* trend.

➔ *Comparing a pair of y-axes for different "z"*



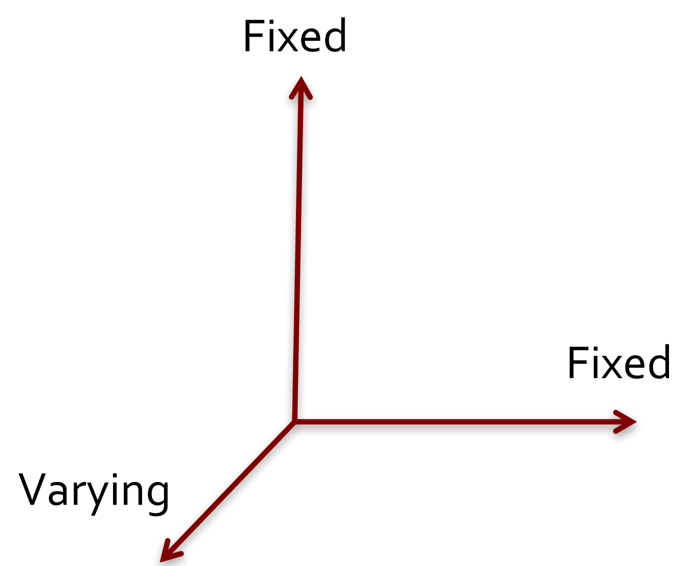
Example 1: Comparisons



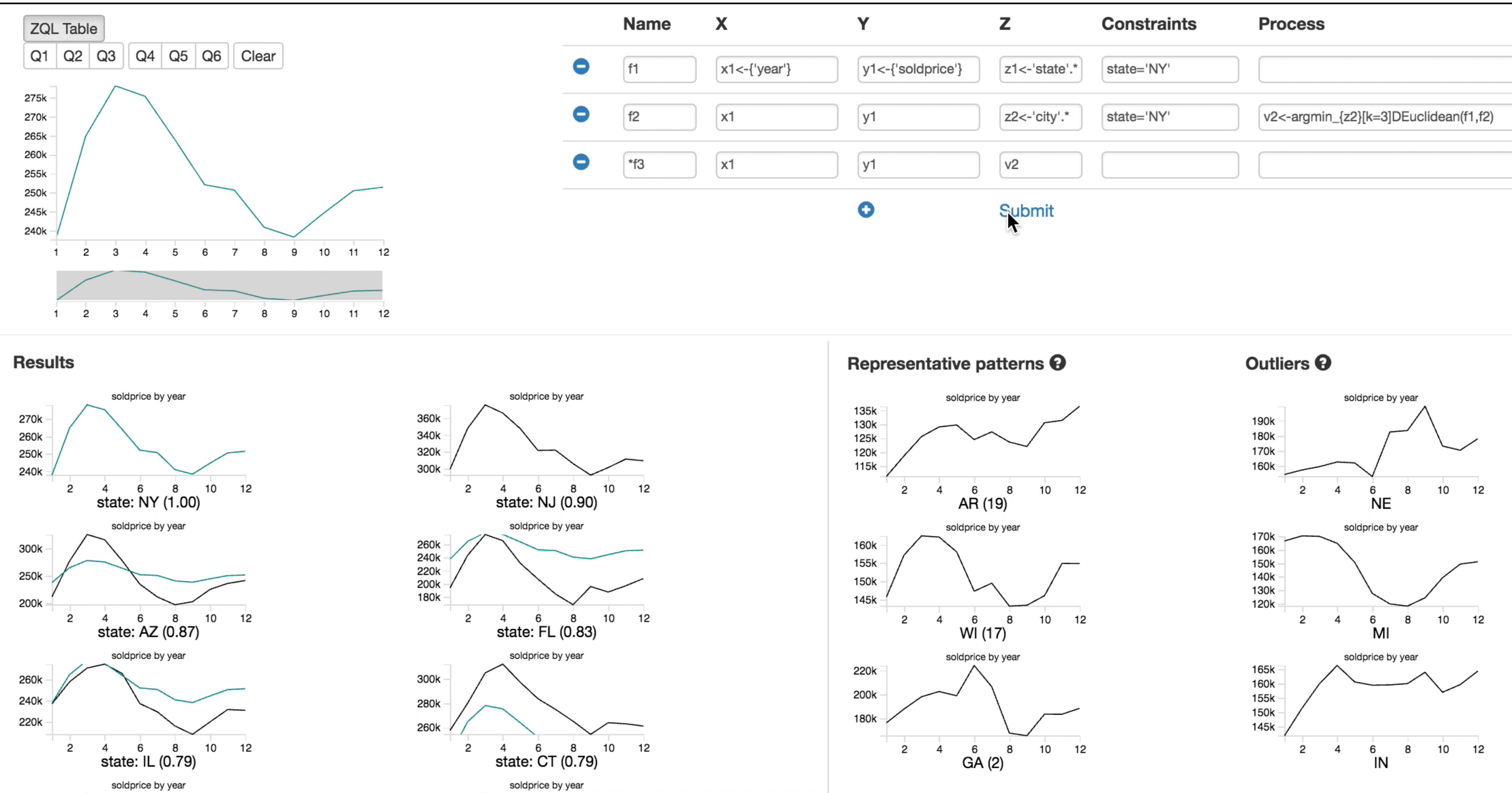
Example 2: Drill-downs

Find *cities in NY* where the trend for *soldprice* is most different from (or most similar to) the *overall NY trend.*

➔ *Comparing across different granularities of "z"*



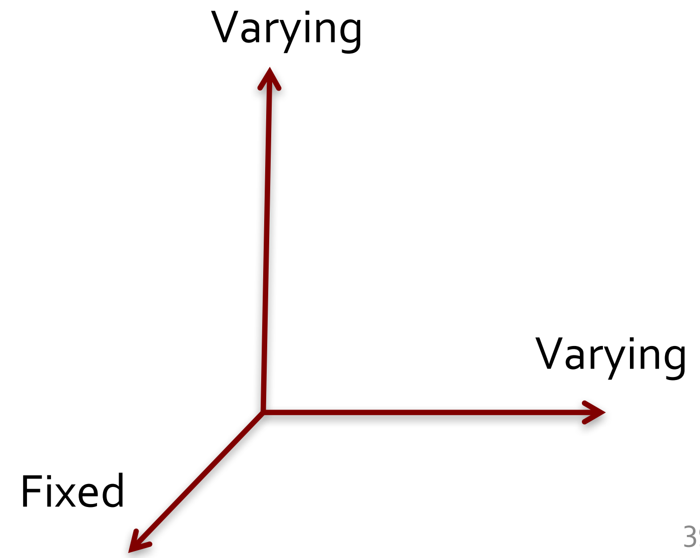
Example 2: Drill-downs



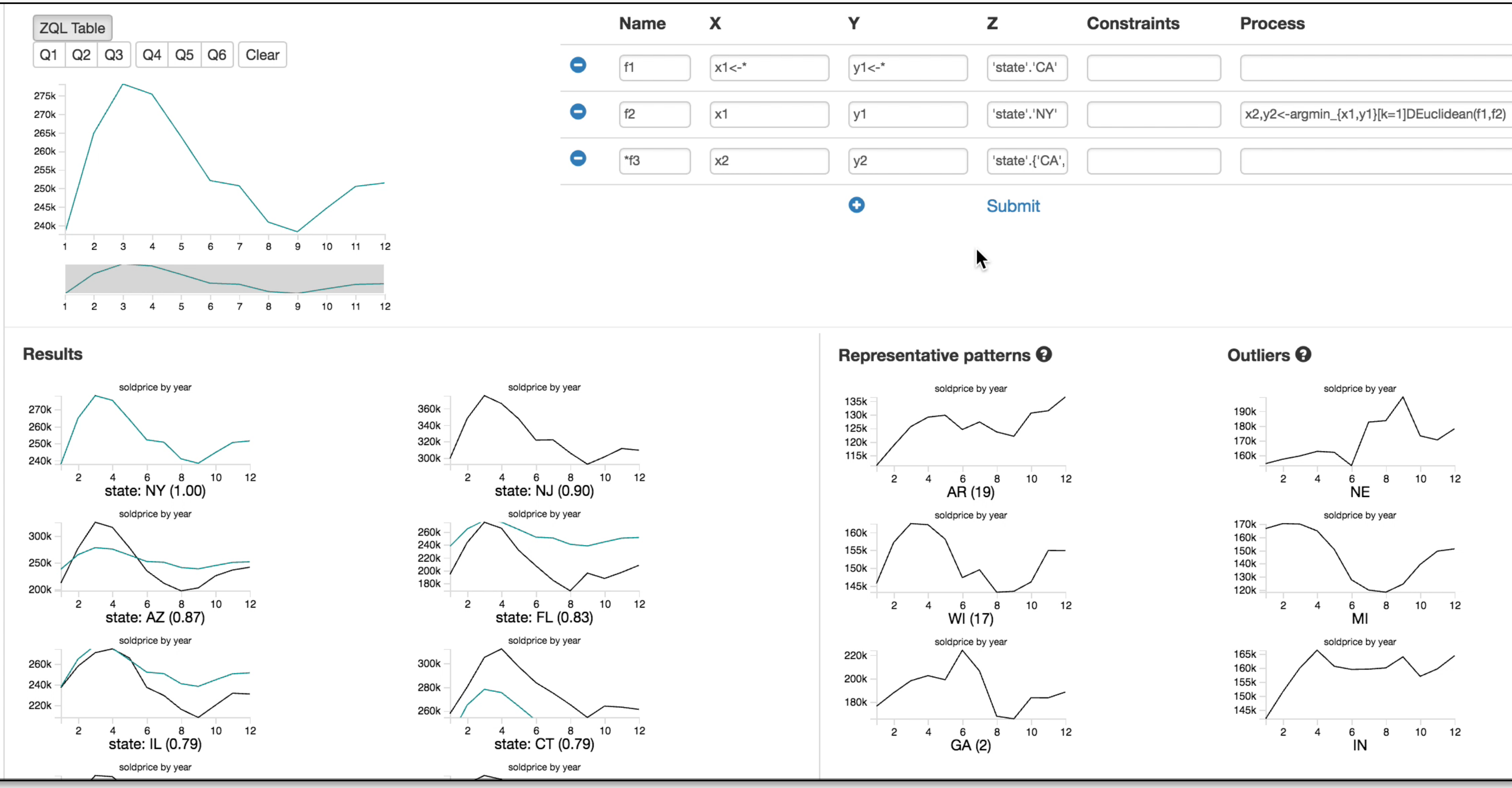
Example 3: Explanations/Diffs

Find visualizations on which the *states of CA* and *NY* are most different (or most similar).

➔ *Comparing across different "x", "y" for two "z"*



Example 3: Explanations/Diffs



ZQL Query Execution

Let's use a relational database as a backend

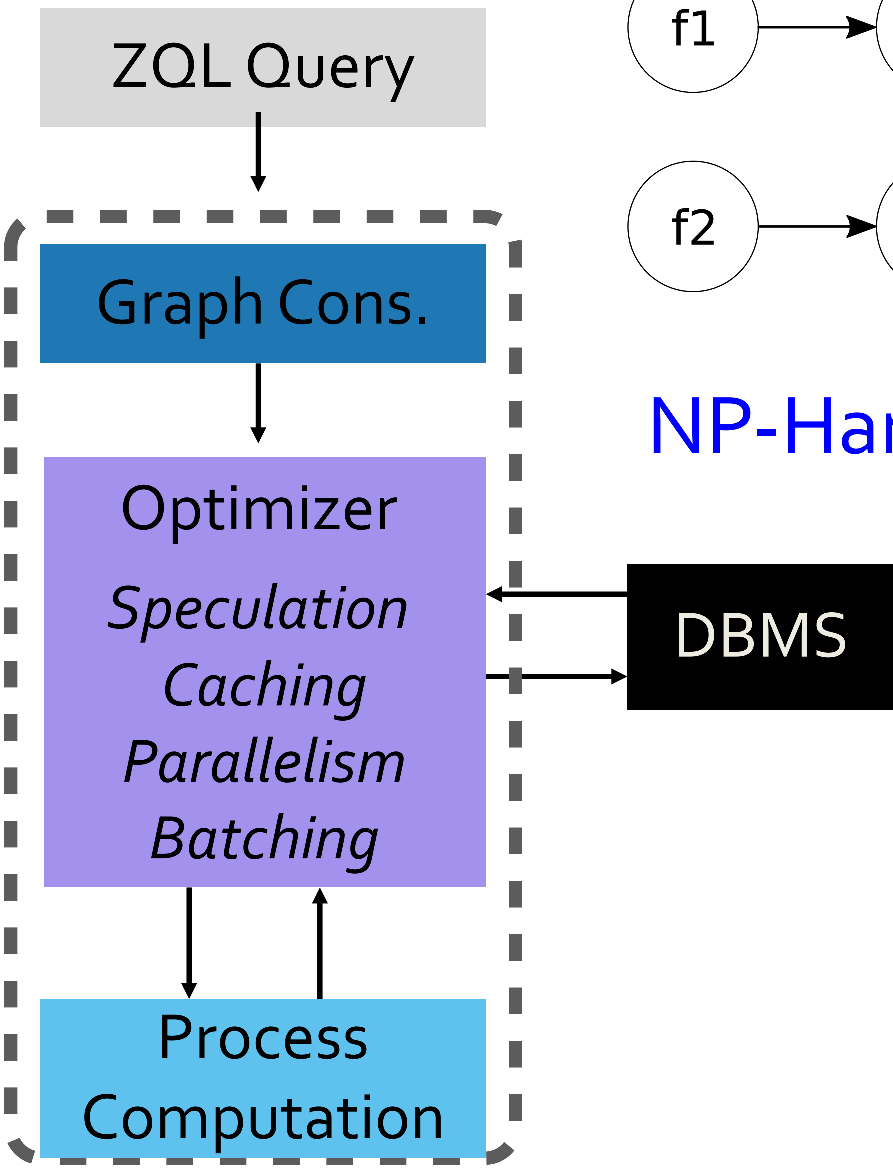
Naïve translation approach:

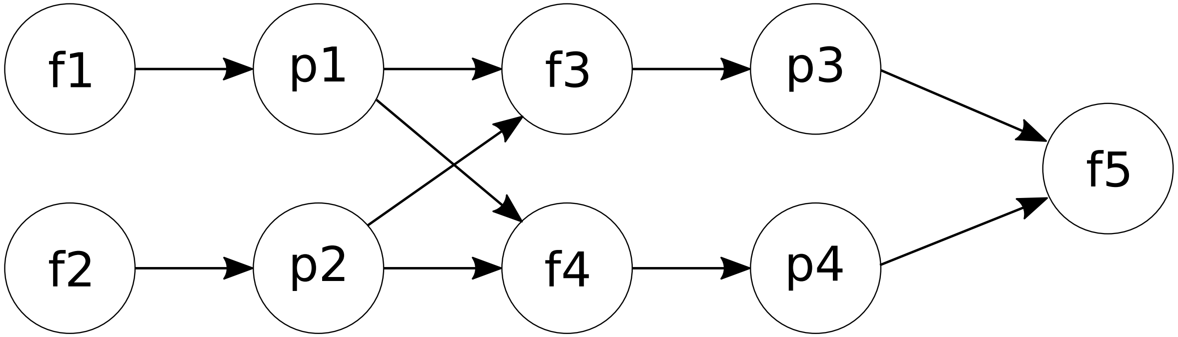
For each line of ZQL:
    Issue one SQL query for each combination of X, Y, Z;
    Apply further processing on result

Often 1000s of SQL queries issued per ZQL query!

➔ *wasteful, extremely high latency*

SmartFuse: Intelligent Query Optimizer





NP-Hard!

Sequential

↓(99.99%)

Grouped

↓(45%)

Parallel

↓(20%)

Speculation

↓(20%)

SmartFuse

User Study Takeaways (20 Participants)

Faster	**μ =115s**, σ =51.6 vs. μ =172.5s, σ =50.5
More accurate	**μ =96.3%**, σ =5.82 vs. μ =69.9%, σ =13.3

"***In Tableau, there is no pattern searching**. If I see some pattern in Tableau, such as a decreasing pattern, and I want to see if any other variable is decreasing in that month, I **have to go one by one** to find this trend. But here I can find this through the query table."*

*"you can just [edit] and draw to find out similar patterns. You'll **need to do a lot more through Matlab** to do the same thing."*

*"The obvious good thing is that you **can do complicated queries**, and you **don't have to write SQL** queries… I can imagine a non-cs student [doing] this."*

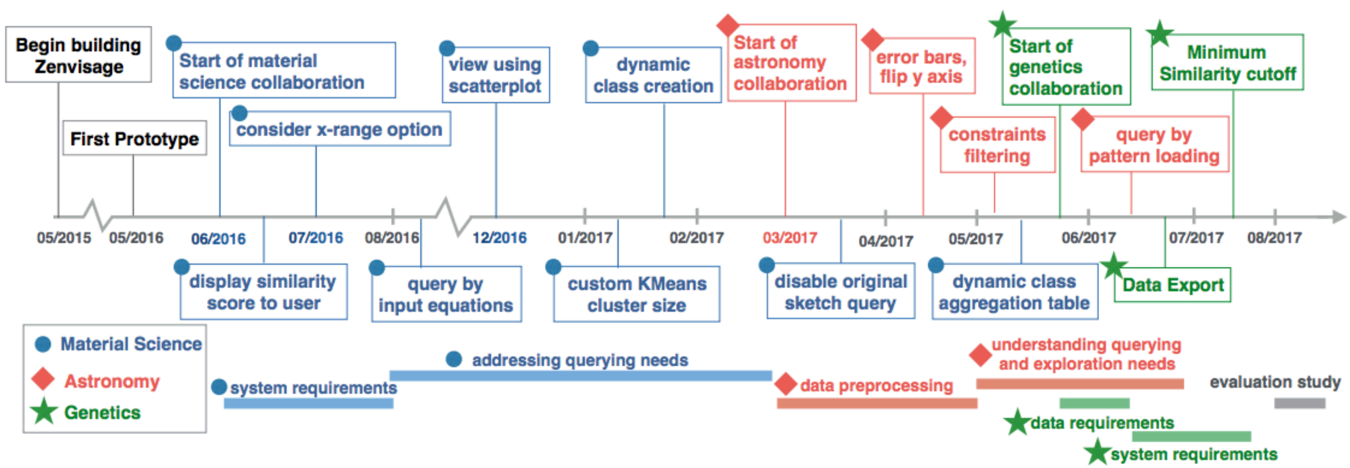
Real Usage Stories (1-year long dev)

Carnegie Mellon University
Scott Institute
for Energy Innovation

THE DARK
ENERGY SURVEY

knoweng

- Confirmed gene expression profiles in recent publication
- Unknown dip in an astro light curve was caused due to saturated image equipment
- Relationship between viscosity and lithium solvation energy is indep. of whether a solvent is a high or low V solvent



Effortless Visual Exploration of Large Datasets with zenvisage

Ingredients

- *Drag-and-drop and sketch based interactions*
  - to find specific patterns
- *Sophisticated visual exploration language, ZQL*
  - to ask more elaborate questions
- *Scalable visualization generation engine*
  - preprocess, batch and parallel eval. for interactive results
- *Rapid pattern matching algorithms*
  - sampling-based techniques

OrpheusDB

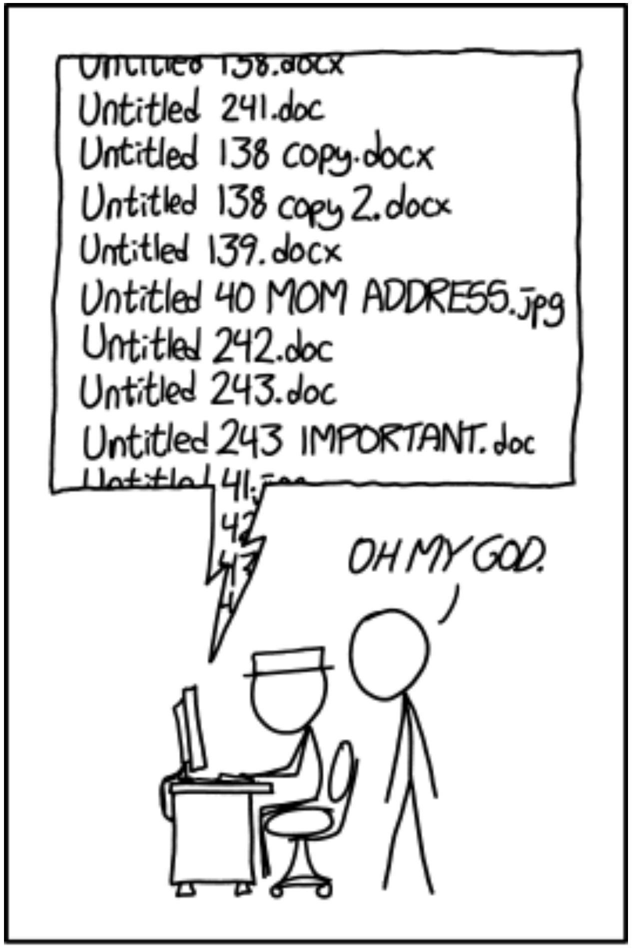
Motivation

Collaborative data science is ubiquitous

- Many users, many versions of the same dataset stored at many stages of analysis
- Status quo:
  - Stored in a file system, relationships unknown

Challenge: can we build a versioned data store?

  - Support efficient access, retrieval, querying, and modification of versions



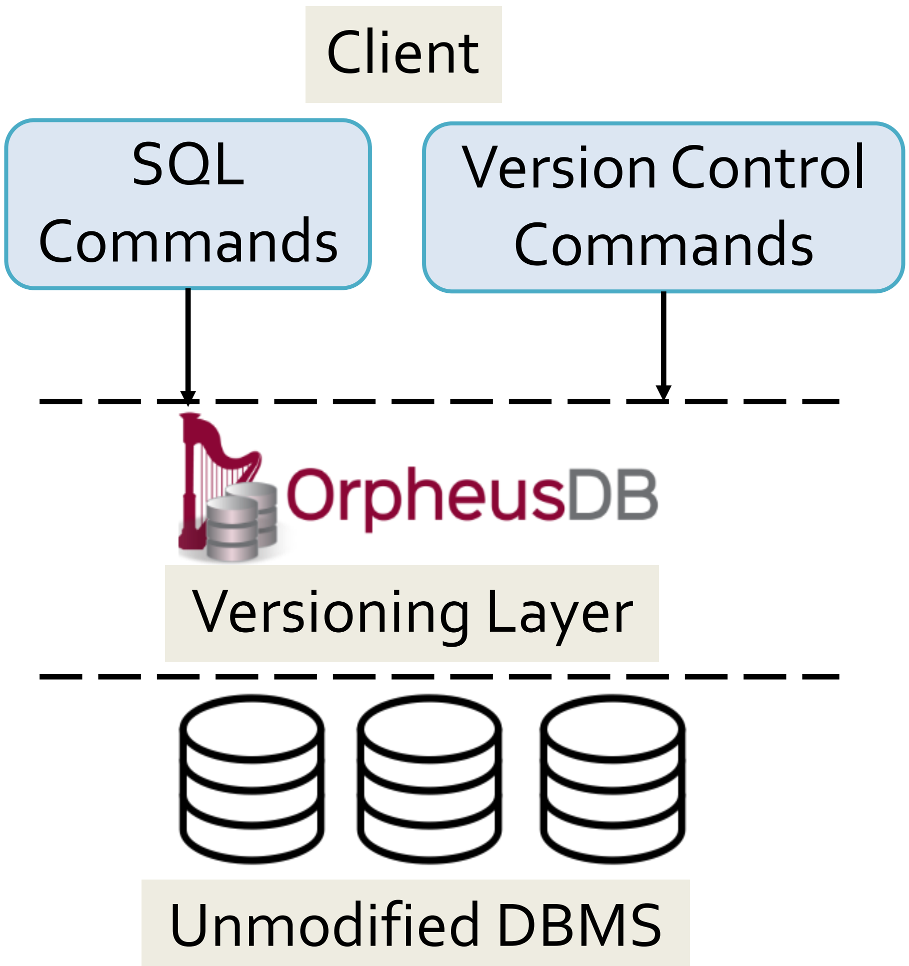
PROTIP: NEVER LOOK IN SOMEONE ELSE'S DOCUMENTS FOLDER.

Motivation: Starting Points

- **VCS**: Git/svn is inefficient and unsuitable
  - Ordered semantics
  - No data manipulation API
  - No efficient multi-version queries
  - Poor support for massive files

- **DBMS:** Relational databases don't support versioning, but are efficient and scalable

OrpheusDB: A Bolt-On Approach



- Retrieve the first version that contains this tuple
- Find versions where the average(salary) is greater than 1000
- Find all pairs of versions where over 100 new tuples were added
- Show the history of the tuple with record id 34.

49

Representing Versions in a DB: Take 1

badgeID	age	gender	salary	vid
0001	25	F	6500	v_1
0001	25	F	7500	v_3
0001	25	F	7500	v_4
0002	30	F	7500	v_1
0002	30	F	7500	v_2
0002	30	F	7500	v_4
0003	28	M	7000	v_1
0003	28	M	7000	v_2
0003	28	M	7000	v_3
0003	28	M	7000	v_4
0004	40	M	9000	v_2
0004	40	M	9000	v_4
0005	35	F	6500	v_3
0005	35	F	6500	v_4
0006	32	M	7000	v_3
0006	32	M	7000	v_4

badgeID	age	gender	salary	vlist
0001	25	F	6500	$\{v_1\}$
0001	25	F	7500	$\{v_3, v_4\}$
0002	30	F	7500	$\{v_1, v_2, v_4\}$
0003	28	M	7000	$\{v_1, v_2, v_3, v_4\}$
0004	40	M	9000	$\{v_2, v_4\}$
0005	35	F	6500	$\{v_3, v_4\}$
0006	32	M	7000	$\{v_3, v_4\}$

Representing Versions in a DB: Take 2

badgeID	age	gender	salary	vlist
0001	25	F	6500	$\{v_1\}$
0001	25	F	7500	$\{v_3, v_4\}$
0002	30	F	7500	$\{v_1, v_2, v_4\}$
0003	28	M	7000	$\{v_1, v_2, v_3, v_4\}$
0004	40	M	9000	$\{v_2, v_4\}$
0005	35	F	6500	$\{v_3, v_4\}$
0006	32	M	7000	$\{v_3, v_4\}$

rid	badgeID	age	gender	salary
r_1	0001	25	F	6500
r_2	0002	30	F	7500
r_3	0003	28	M	7000
r_4	0004	40	M	9000
r_5	0001	25	F	7500
r_6	0005	35	F	6500
r_7	0006	32	M	7000

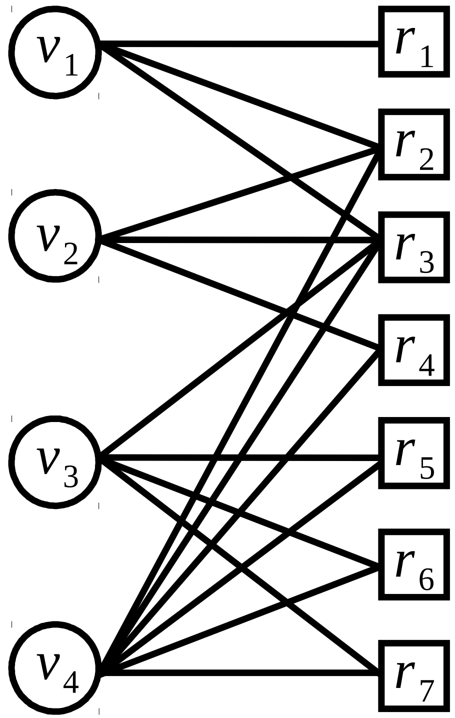
$\bowtie_\theta$

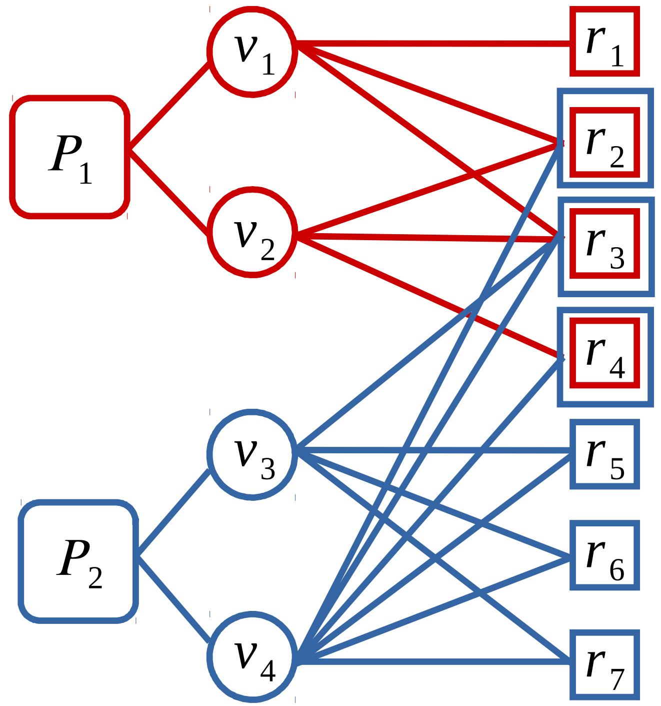
rid	vlist
r_1	$\{v_1\}$
r_2	$\{v_1, v_2, v_4\}$
r_3	$\{v_1, v_2, v_3, v_4\}$
r_4	$\{v_2, v_4\}$
r_5	$\{v_3, v_4\}$
r_6	$\{v_3, v_4\}$
r_7	$\{v_3, v_4\}$

vid	rlist
v_1	$\{r_1, r_2, r_3\}$
v_2	$\{r_2, r_3, r_4\}$
v_3	$\{r_3, r_5, r_6, r_7\}$
v_4	$\{r_2, r_3, r_4, r_5, r_6, r_7\}$

Representing Versions in a DB: Take 3

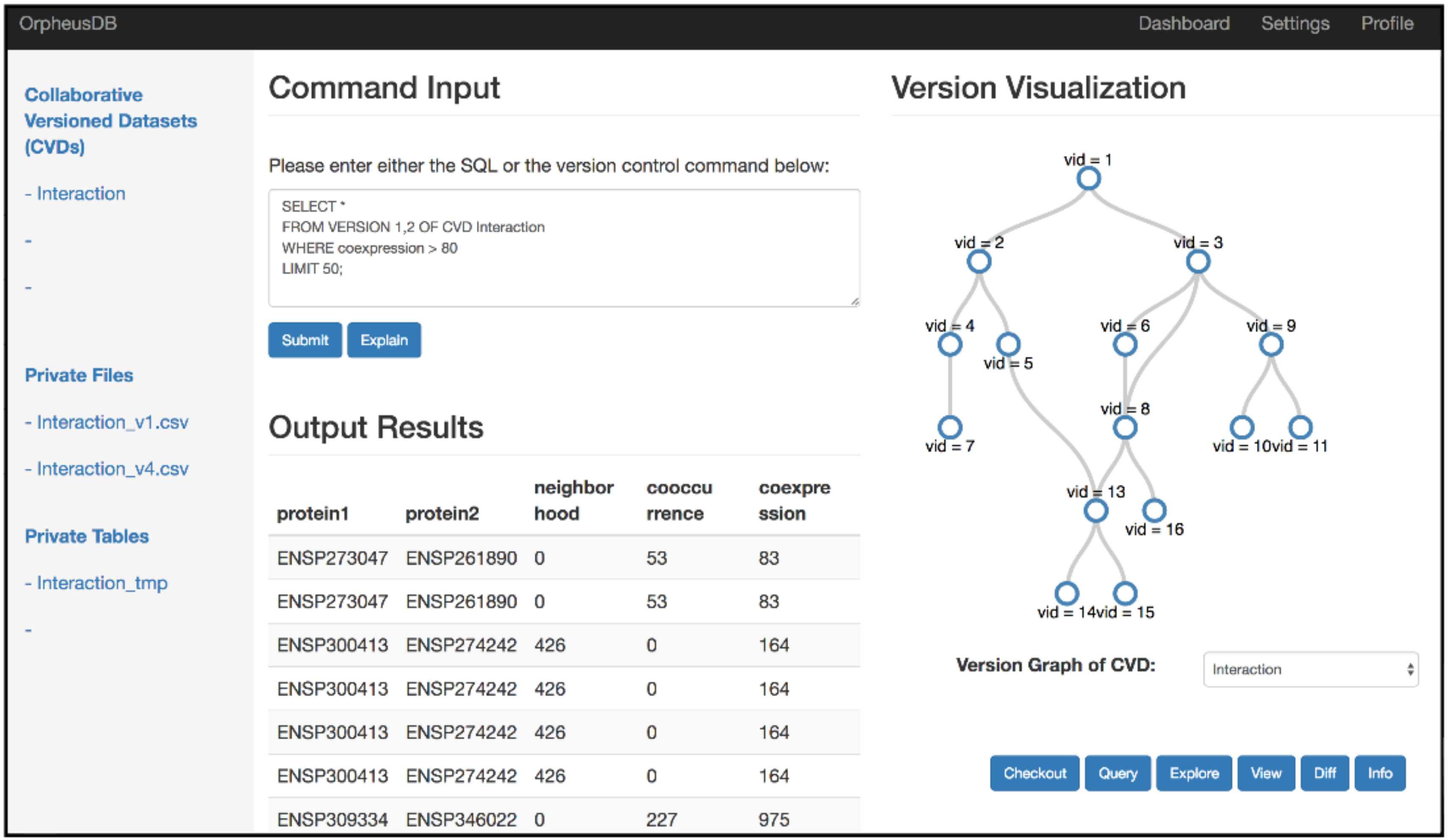
Still slow… Apply partitioning!





Optimally partitioning minimizing storage and retrieval: NP-Hard!

OrpheusDB


OrpheusDB
Dashboard
Settings
Profile
Collaborative Versioned Datasets (CVDs)
- Interaction
-
-
Private Files
- Interaction_v1.csv
- Interaction_v4.csv
Private Tables
- Interaction_tmp
-
Command Input
Please enter either the SQL or the version control command below:
SELECT *
FROM VERSION 1,2 OF CVD Interaction
WHERE coexpression > 80
LIMIT 50;
Submit
Explain
Output Results
protein1 protein2 neighborhood cooccurrence coexpression
ENSP273047 ENSP261890 0 53 83
ENSP273047 ENSP261890 0 53 83
ENSP300413 ENSP274242 426 0 164
ENSP300413 ENSP274242 426 0 164
ENSP300413 ENSP274242 426 0 164
ENSP300413 ENSP274242 426 0 164
ENSP309334 ENSP346022 0 227 975
Version Visualization
vid = 1
vid = 2
vid = 3
vid = 4
vid = 5
vid = 6
vid = 9
vid = 7
vid = 8
vid = 10
vid = 11
vid = 13
vid = 16
vid = 14
vid = 15
Version Graph of CVD:
Interaction
Checkout
Query
Explore
View
Diff
Info

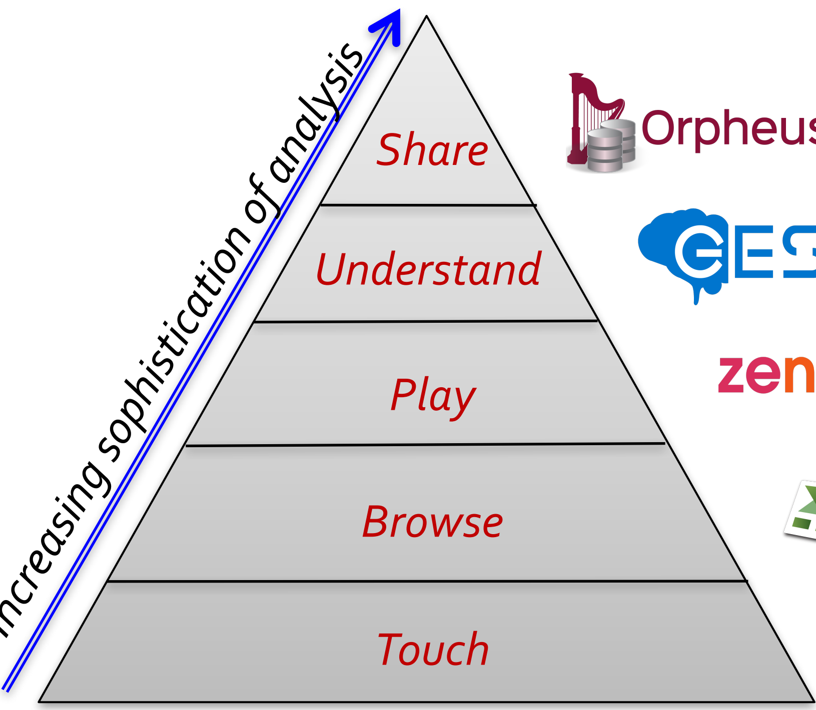
Some Takeaways…

1. Many underserved communities: *why only focus on the needs of the 1%?*

2. Working with consumers from the get go: *keeps you honest; avoid the non-problems*

3. The “Human-in-the-loop” is crucial: *the interfaces are as important as the algorithms*

Summary: Takeaways



OrpheusDB — orpheus-db.github.io

GESTALT — gestalt-ml.github.io

zenvisage — zenvisage.github.io

DataSpread — dataspread.github.io

Datamaran — datamaran.github.io

My website: http://data-people.cs.illinois.edu
Twitter: @adityagp