UNIVERSITY of WASHINGTON

Overview: Relational Abstractions

Bill Howe, PhD
Director of Research,
Scalable Data Analytics
University of Washington
eScience Institute

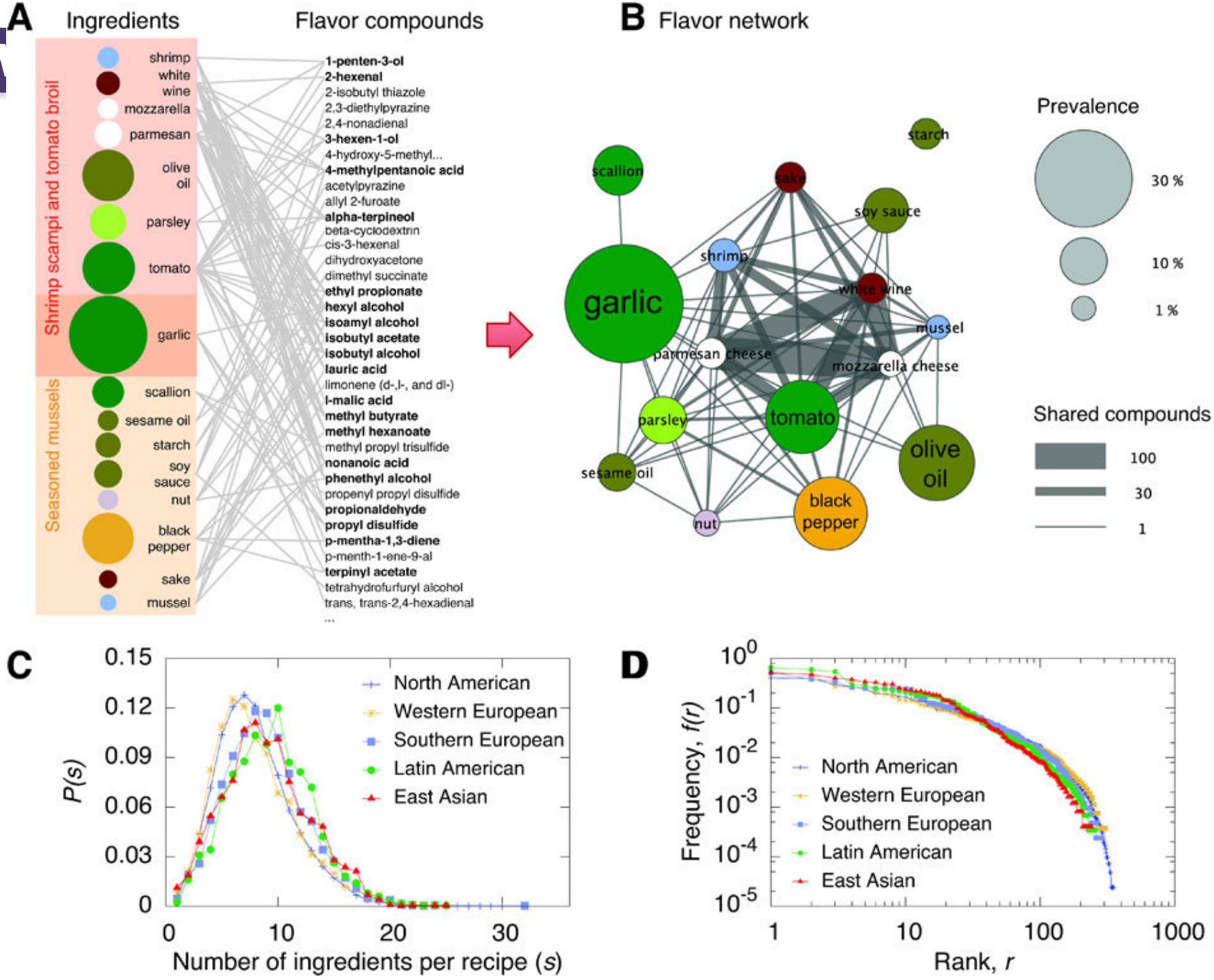
Flavor network and the principles of food pairing

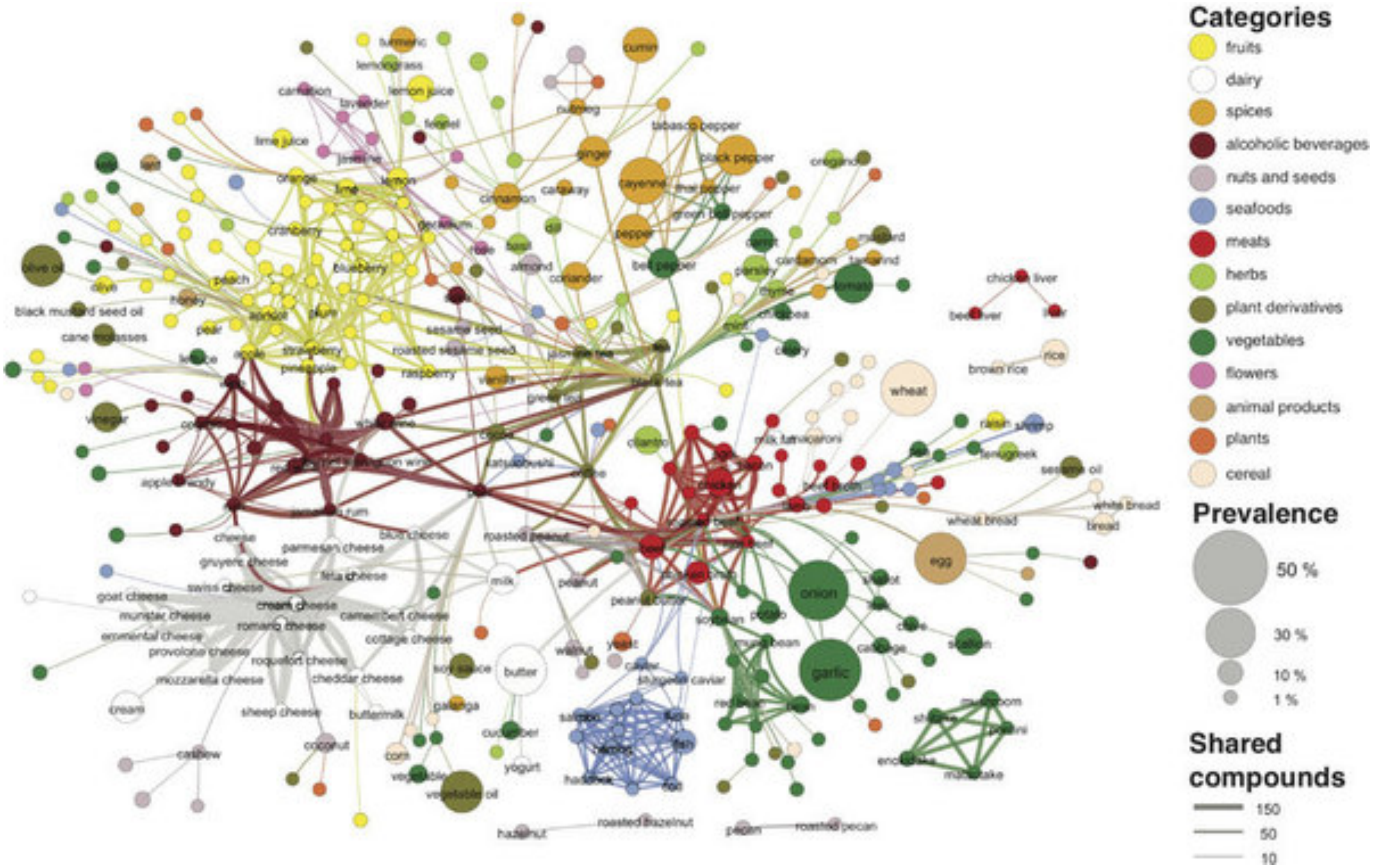
Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow & Albert-László Barabási

Affiliations | Contributions | Corresponding authors

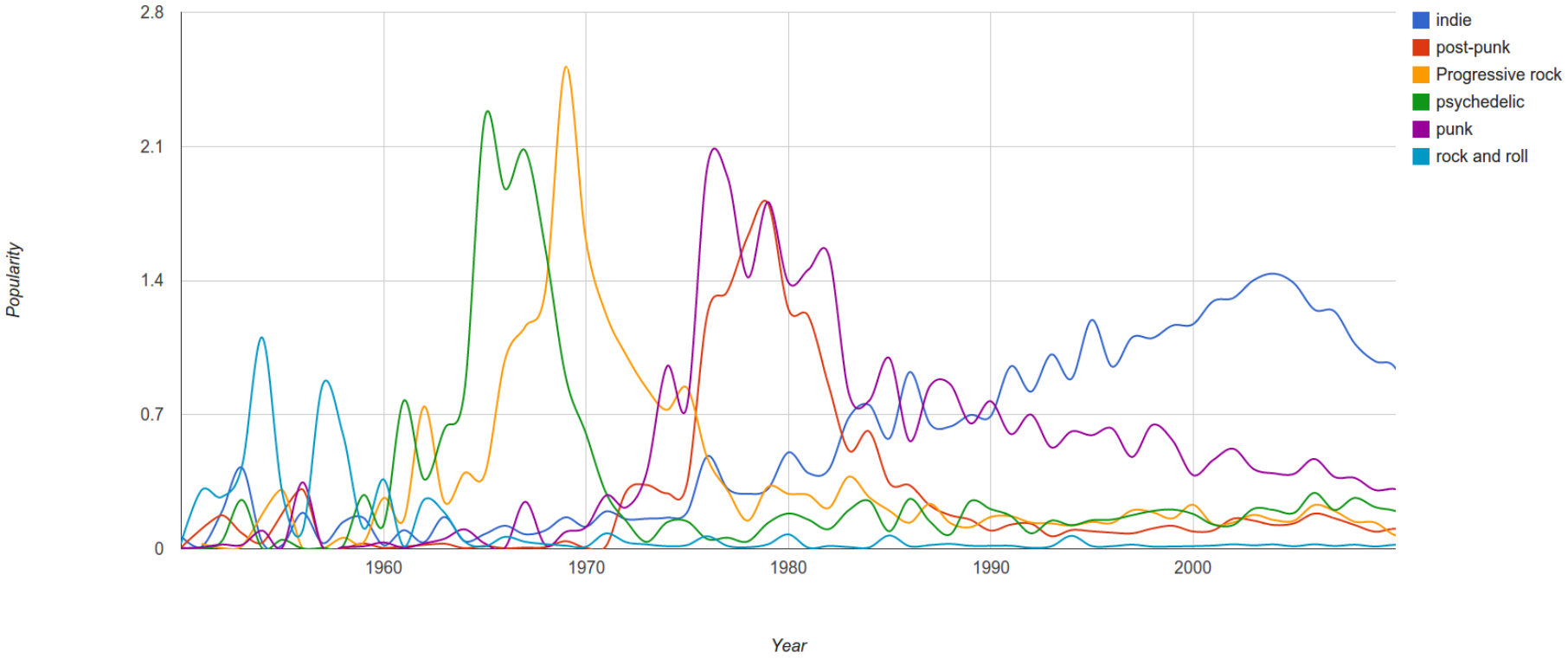
Scientific Reports **1**, Article number: 196 | doi:10.1038/srep00196

Received 18 October 2011 | Accepted 24 November 2011 | Published 15 December 2011


A
Ingredients
Flavor compounds
Shrimp scampi and tomato broil
Seasoned mussels
shrimp
white wine
mozzarella
parmesan
olive oil
parsley
tomato
garlic
scallion
sesame oil
starch
soy sauce
nut
black pepper
sake
mussel
1-penten-3-ol
2-hexenal
2-isobutyl thiazole
2,3-diethylpyrazine
2,4-nonadienal
3-hexen-1-ol
4-hydroxy-5-methyl...
4-methylpentanoic acid
acetylpyrazine
allyl 2-furoate
alpha-terpineol
beta-cyclodextrin
cis-3-hexenal
dihydroxyacetone
dimethyl succinate
ethyl propionate
hexyl alcohol
isoamyl alcohol
isobutyl acetate
isobutyl alcohol
lauric acid
limonene (d-,l-, and dl-)
l-malic acid
methyl butyrate
methyl hexanoate
methyl propyl trisulfide
nonanoic acid
phenethyl alcohol
propenyl propyl disulfide
propionaldehyde
propyl disulfide
p-mentha-1,3-diene
p-menth-1-ene-9-al
terpinyl acetate
tetrahydrofurfuryl alcohol
trans, trans-2,4-hexadienal
...
B Flavor network
Prevalence
30 %
10 %
1 %
Shared compounds
100
30
1
C
P(s)
0.15
0.12
0.09
0.06
0.03
0
0
10
20
30
Number of ingredients per recipe (s)
North American
Western European
Southern European
Latin American
East Asian
D
Frequency, f(r)
Rank, r
1
10
100
1000
North American
Western European
Southern European
Latin American
East Asian


Categories
fruits
dairy
spices
alcoholic beverages
nuts and seeds
seafoods
meats
herbs
plant derivatives
vegetables
flowers
animal products
plants
cereal
Prevalence
50 %
30 %
10 %
1 %
Shared compounds
150
50
10
wheat
egg
onion
garlic
cumin

Last.FM



Since we have a massive amount of user tag data available we can easily correlate tags and years and measure “popularity” of a genre by counting the number of artists formed in a specific year.

THE TWITTER POLITICAL INDEX

OCTOBER 23, 2012: OBAMA +18 LEARN MORE

Tweet Follow @gov

BARACK OBAMA

44 +2 SINCE YESTERDAY

Follow @BarackObama 10M+ followers

MITT ROMNEY

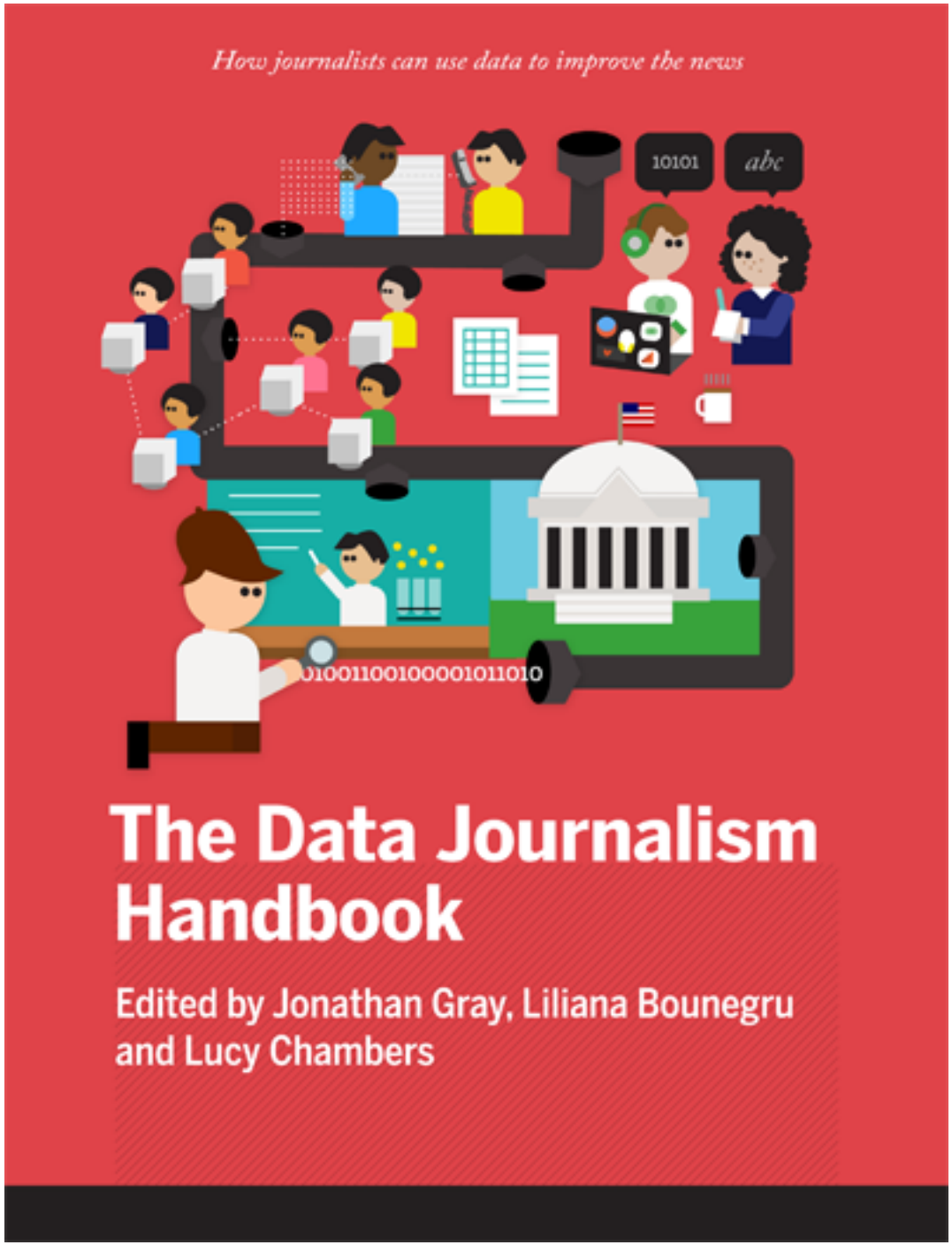
26 +0 SINCE YESTERDAY

Follow @MittRomney 1.5M followers

The Million Song Dataset Challenge: Part I

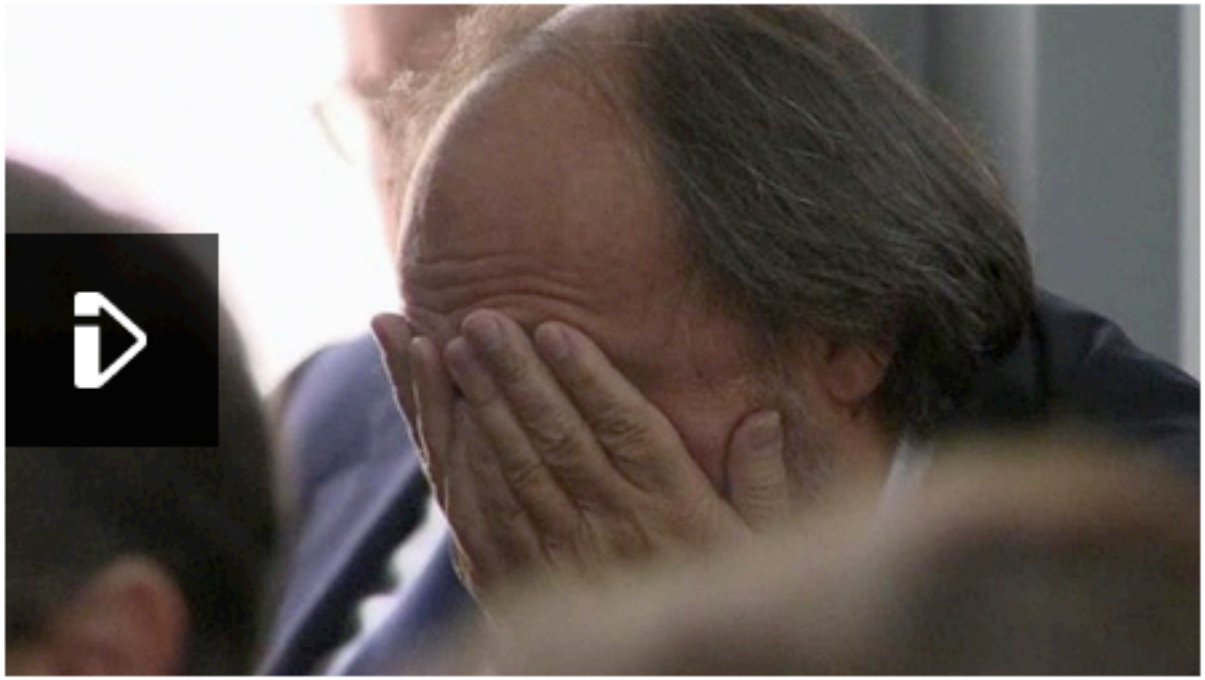
"people who listened to X also listened to Y"

UNIVERSITY of



L'Aquila quake: Italy scientists guilty of manslaughter

COMMENTS (1022)



The BBC's Alan Johnston in Rome says the prosecution argued that the scientists were "just too reassuring"

Six Italian scientists and an ex-government official have been sentenced to six years in prison over the 2009 deadly earthquake in L'Aquila.

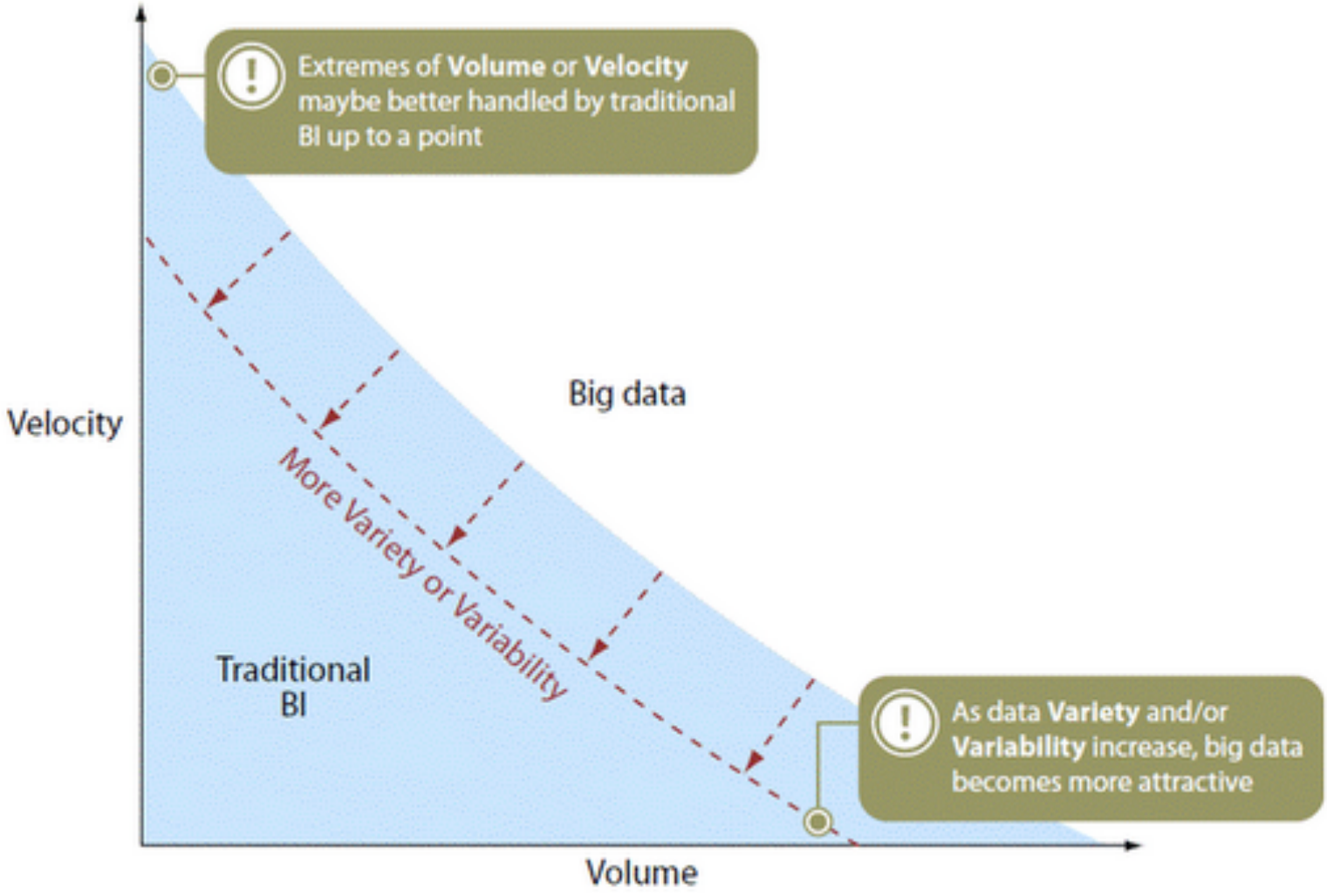
Related Stories

Roadmap

- Bonus slides from last lecture
- Relational Algebra and query evaluation
- In-Database Analytics
- Demo

UNIVE

Volume — exceeds physical limits of vertical scalability	**Velocity** — decision window small compared to data change rate
Variety — many different formats makes integration expensive	**Variability** — many options or variable interpretations confound analysis



Relational Database History

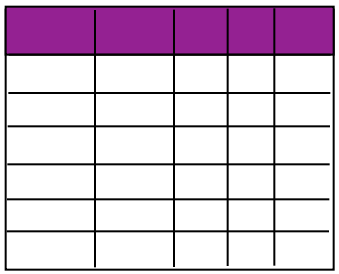
Pre-Relational: if your data changed, your application broke.

Early RDBMS were buggy and slow (and often reviled), but required only 5% of the application code.

"Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed." *-- Codd 1979*

Key Ideas: Programs that manipulate tabular data exhibit an algebraic structure allowing reasoning and manipulation independently of physical data representation

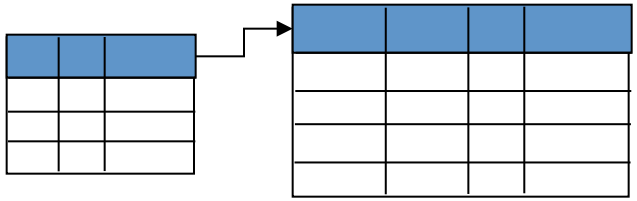
Key Idea: Data Independence



views

```
SELECT *
  FROM my_sequences
```

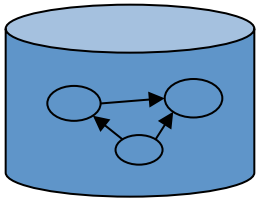
logical data independence



relations

```
SELECT seq
  FROM ncbi_sequences
 WHERE seq = 'GATTACGATATTA';
```

physical data independence



files and pointers

```
f = fopen('table_file');
fseek(10030440);
while (True) {
  fread(&buf, 1, 8192, f);
  if (buf == GATTACGATATTA) {
```

Key Idea: Indexes

- Databases are especially, but not exclusively, effective at "Needle in Haystack" problems:
  - **Extracting small results from big datasets**
  - **Transparently provide "old style" scalability**
  - **Your query will always* finish, regardless of dataset size.**

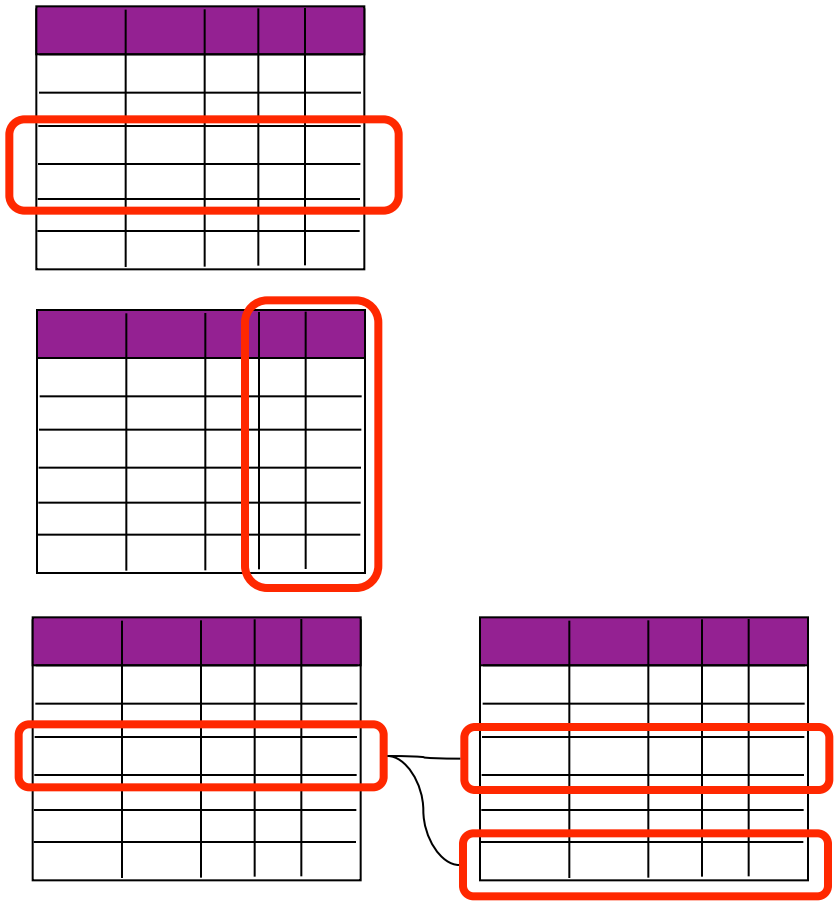
  - **Indexes are easily built and automatically used when appropriate**

```
CREATE INDEX seq_idx ON sequence(seq);

SELECT seq
  FROM sequence
 WHERE seq = 'GATTACGATATTA';
```

*almost

Key Idea: An *Algebra of Tables*



select

project

join

Other operators: aggregate, union, difference, cross product

Key Idea: Algebraic Optimization

N = ((z*2)+((z*3)+0))/1

Algebraic Laws:

1. (+) identity: x+0 = x
2. (/) identity: x/1 = x
3. (*) distributes: (n*x+n*y) = n*(x+y)
4. (*) commutes: x*y = y*x

Apply rules 1, 3, 4, 2:

N = (2+3)*z

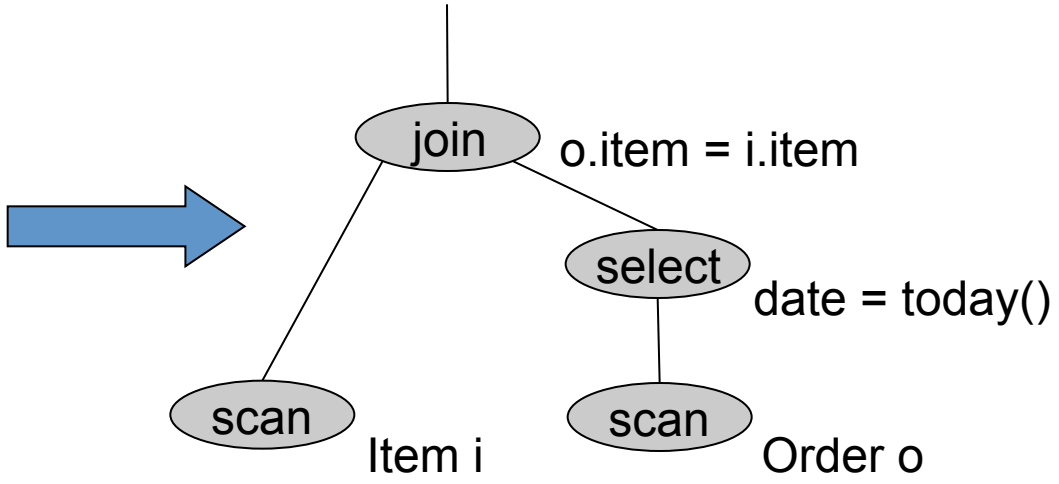
two operations instead of five, no division operator

Same idea works with the Relational Algebra!

Key Idea: Declarative Languages

Find all orders from today, along with the items ordered

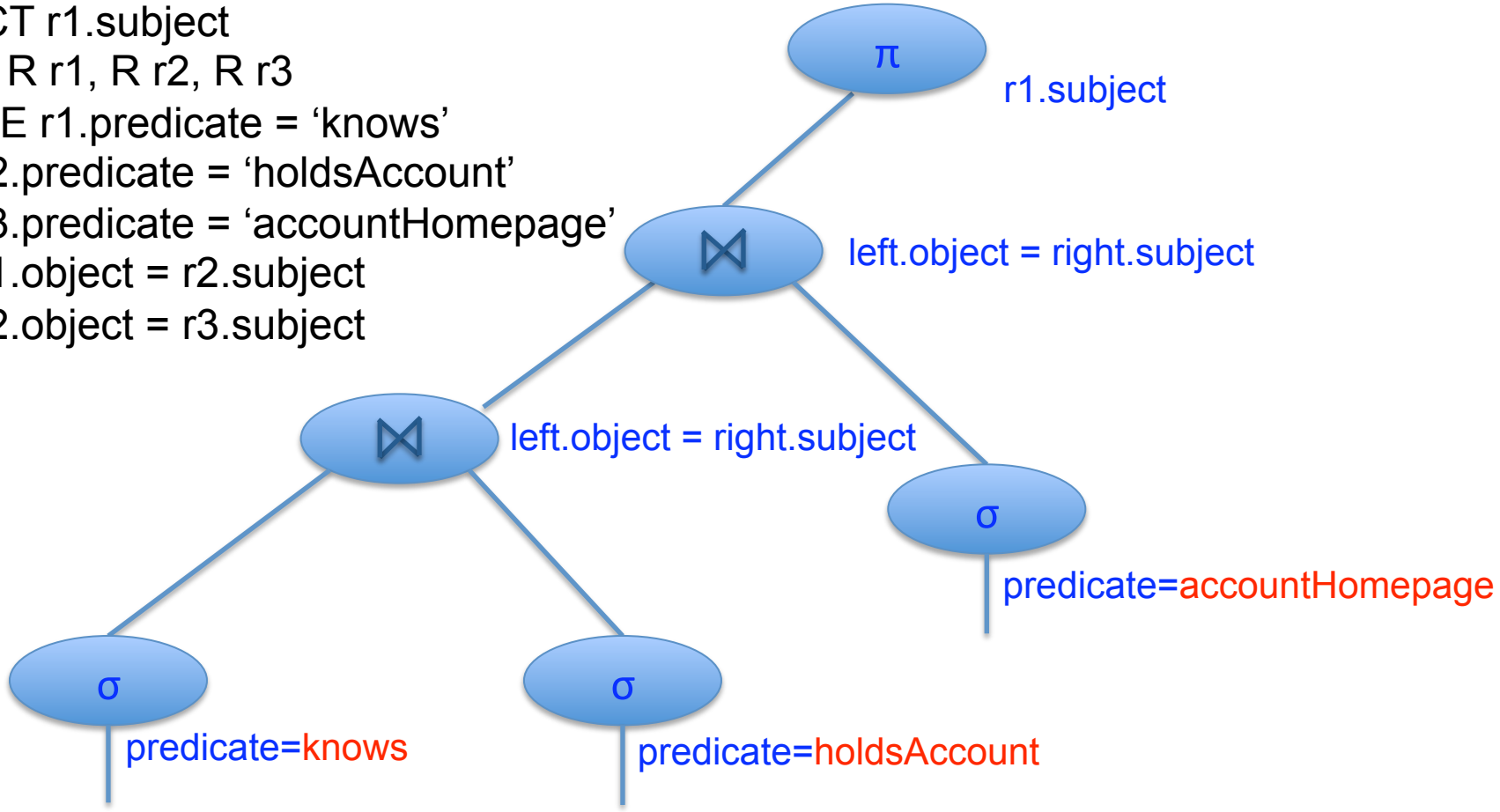
```
SELECT *
  FROM Order o, Item i
 WHERE o.item = i.item
   AND o.date = today()
```



Example Query in Relational Algebra

R(subject, predicate, object)

```
SELECT r1.subject
FROM R r1, R r2, R r3
WHERE r1.predicate = 'knows'
AND r2.predicate = 'holdsAccount'
AND r3.predicate = 'accountHomepage'
AND r1.object = r2.subject
AND r2.object = r3.subject
```



Equivalent logical expressions; different costs

$\sigma_{p=knows}(R) \bowtie_{o=s} (\sigma_{p=holdsAccount}(R) \bowtie_{o=s} \sigma_{p=accountHomepage}(R))$

right associative

$(\sigma_{p=knows}(R) \bowtie_{o=s} \sigma_{p=holdsAccount}(R)) \bowtie_{o=s} \sigma_{p=accountHomepage}(R)$

left associative

$\sigma_{p1=knows\ \&\ p2=holdsAccount\ \&\ p3=accountHomepage\ \&\ r1.o=r2.s\ \&\ r2.o=r3.s}\ (R \times R \times R)$

cross product

Same logical expression, different physical algorithms

A = select(p=knows)
B = select(p=holdsAccount)
C = select(p=accountWebpage)

hA = hash(A)
AB = probe hA with B

hC = hash(C)
ABC = probe hC with AB

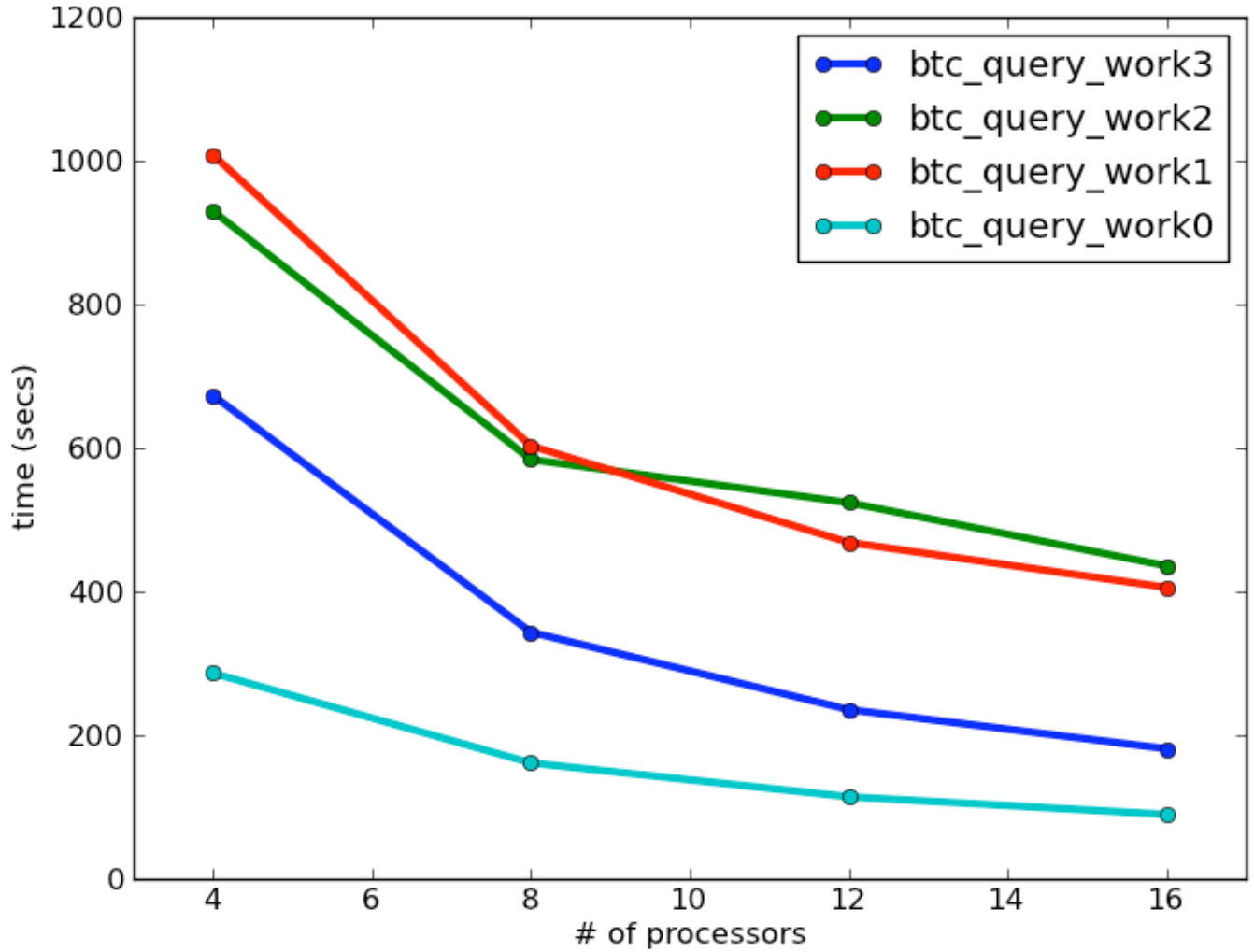
A = select(p=knows)
B = select(p=holdsAccount)
C = select(p=accountWebpage)

hB = hash(B)
AB = probe hB with A

hC = hash(C)
ABC = probe hC with AB

Which is faster?

Algebraic Optimization Matters

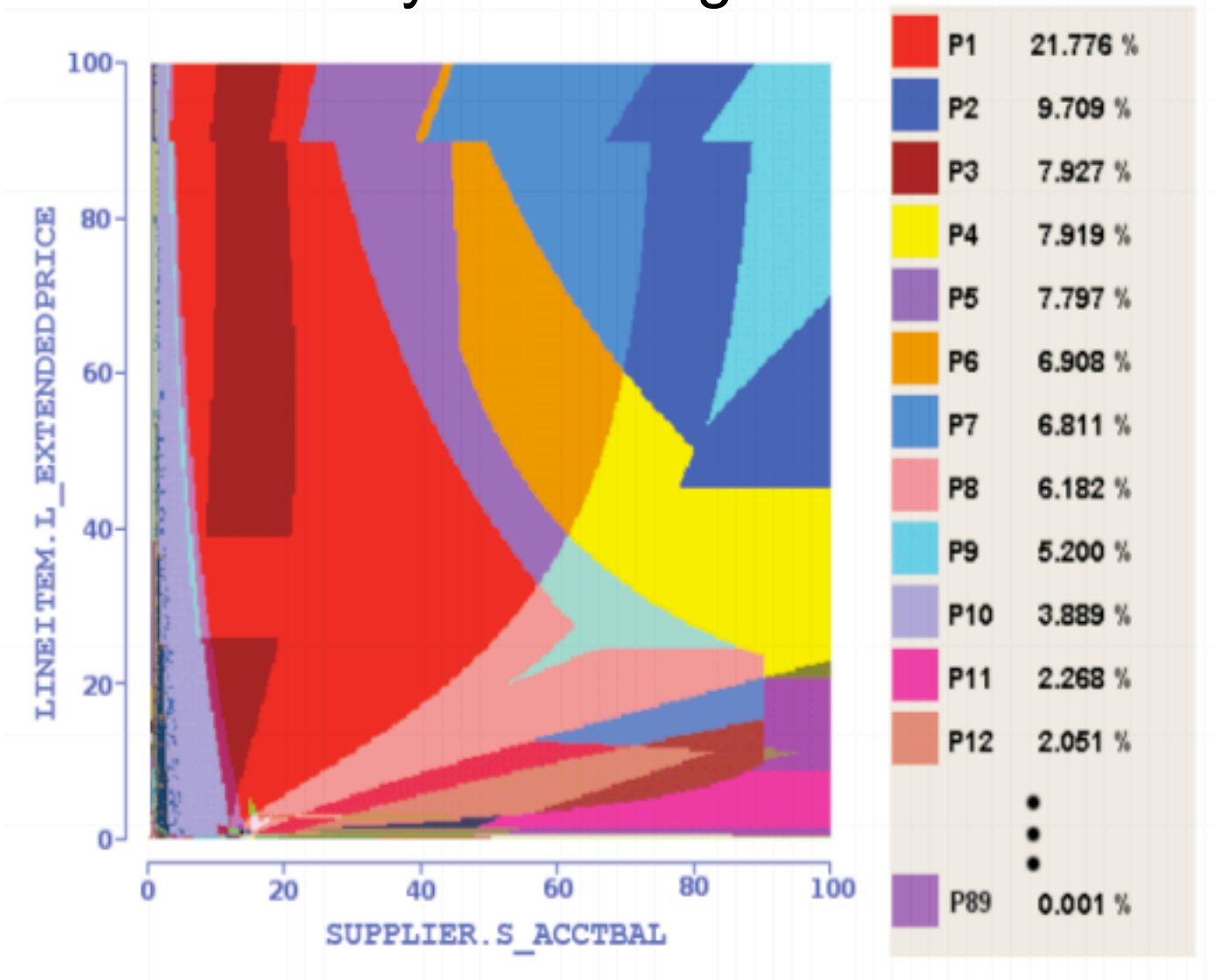


BTC 2010 Dataset

3B quads
623 GB processed

Haritsa, VLDB 2010

Picasso Query Plan Diagrams



IN-DATABASE ANALYTICS

"There is no point in bringing data ... into the data warehouse environment without integrating it. If the data arrives at the data warehouse in an unintegrated state, it cannot be used to support a corporate view of data. And a corporate view of data is one of the essences of the architected environment."

Inmon, 2005, "Building the Data Warehouse"

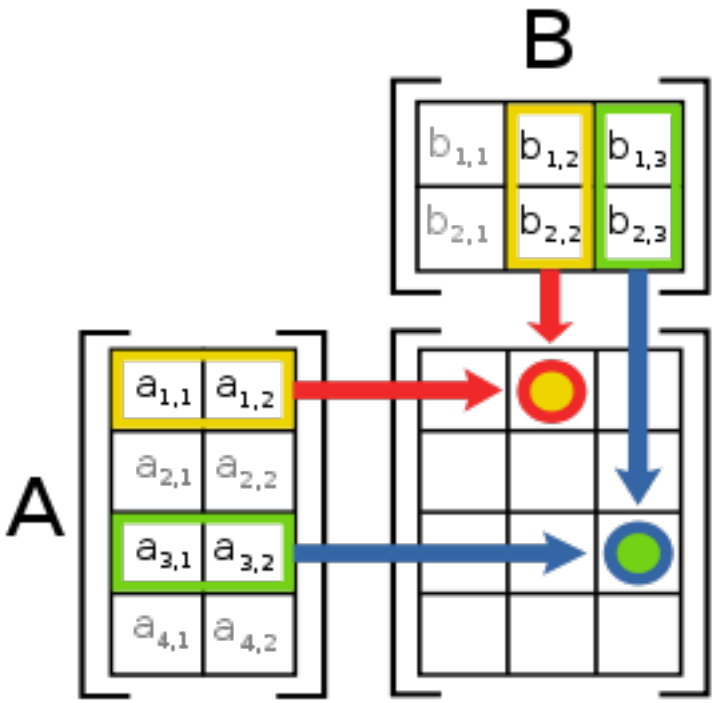
Not a good fit for analytics!

Matrix Addition, vector representation

A(row_number, row float[])
B(row_number, row float[])

```
SELECT A.row_number, A.vector + B.vector
FROM A, B
WHERE A.row_number = B.row_number;
```

Matrix Multiplication



Matrix multiplication, vector representation

A(row_number, row float[])

SELECT 1, array_accum(row_number, vector*v) FROM A;

$$\vec{x} \cdot \vec{y} = \sum_i x_i y_i$$

Matrix Transpose, vector representation

A(row_number, row float[])

```
SELECT S.col_number, array_accum(A.row_number, A.vector[S.col_number])
FROM A, generate_series(1,3) AS S(col_number)
GROUP BY S.col_number;
```

generate_series is a *table function*

Matrix Multiplication, sparse representation

A(row_number, column_number, value)
B(row_number, column_number, value)

```
SELECT A.row_number, B.column_number, SUM(A.value * B.value)
FROM A, B
WHERE A.column_number = B.row_number
GROUP BY A.row_number, B.column_number
```

Aside: User-defined functions and types

- Scalar functions

```
SELECT myfunc(r.a, r.b)…
```

- Aggregate functions

```
SELECT concat(r.s) …
```

- Table functions

```
SELECT … FROM tablefunc(a,b)
```

Experiment design

```
CREATE VIEW design AS
SELECT a.trial_id,
floor (100 * random()) AS row_id
FROM generate_series(1,10000) AS a (trial_id),
generate_series(1,3) AS b (subsample_id)
```

```
CREATE VIEW trials AS
SELECT d.trial_id, AVG(a.values) AS avg_value
FROM design d, T
WHERE d.row_id = T.row_id
GROUP BY d.trial_id
```

src: Cohen et al., VLDB 2009

Prior to the implementation of this functionality within the DBMS, one Greenplum customer was accustomed to calculating the OLS by exporting data and importing the data into R for calculation, a process that took several hours to complete. They reported signicant performance improvement when they moved to running the regression within the DBMS. Most of the benet derived from running the analysis in parallel close to the data with minimal data movement.

src: Cohen et al., VLDB 2009

Assignment

- Matrix Addition for Sparse Representation in SQL
  - Assume the following schema:

  A(row_number, column_number, value)

  B(row_number, column_number, value)

- Bonus: tf-idf in a series of SQL statements
  - Assume the following schema:

  frequency(term, docid, count)

UNIVERSITY of WASHINGTON

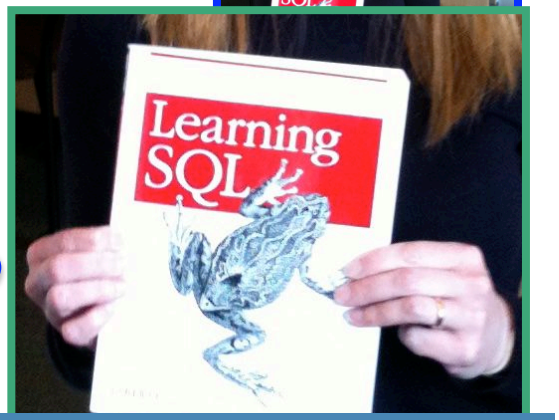
~~Query~~ Database-as-a-Service for the 99%

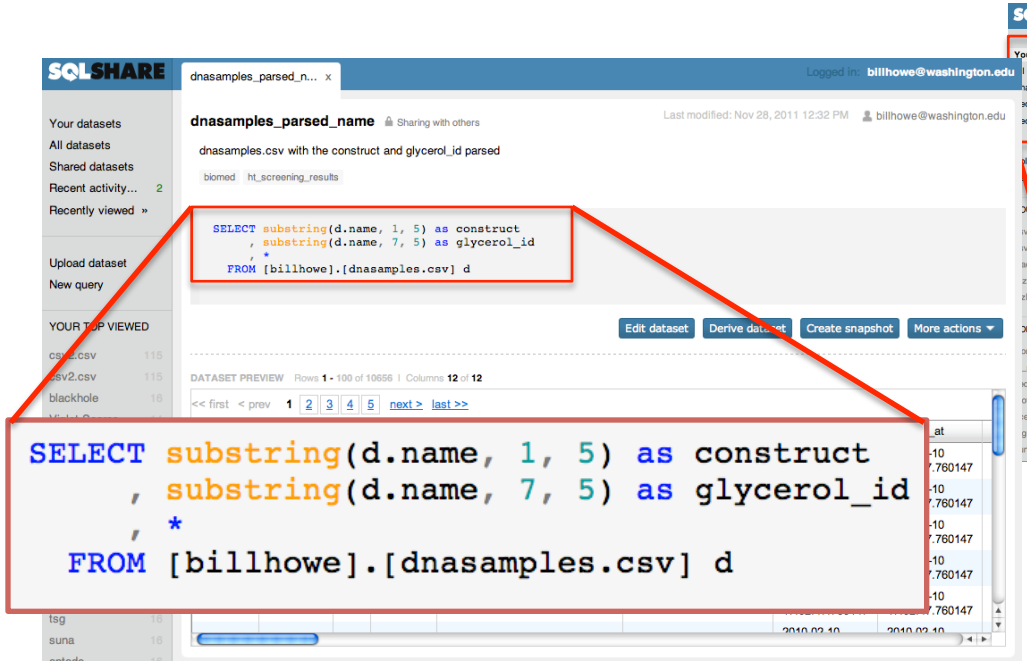
Approach: Strip down databases to bare essentials

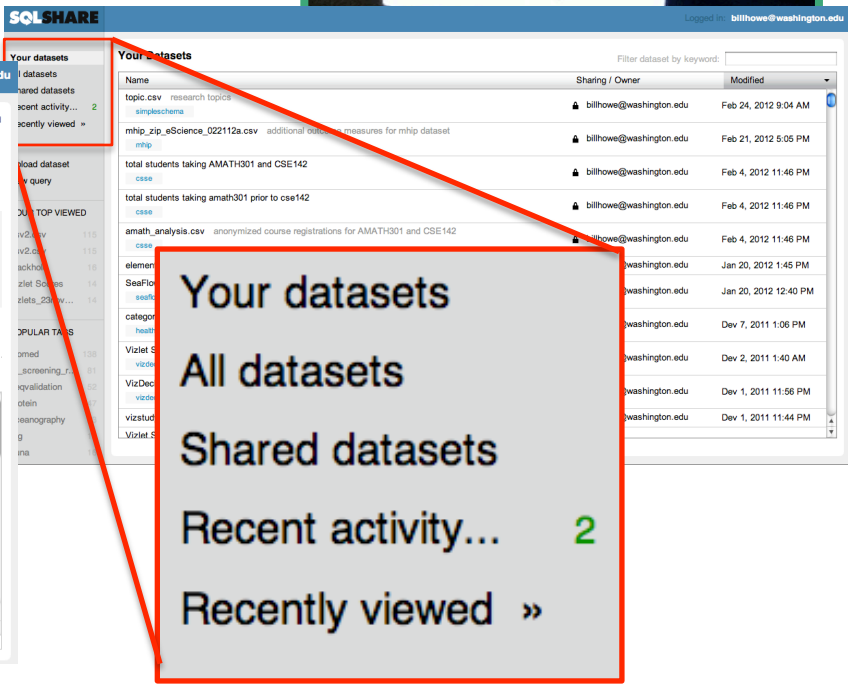
- Upload -> Query -> Share
- Try to eliminate installation, configuration,
  schema design, data loading, tuning, app-building

easy, thanks to the cloud

harder







Getting Data In

- Upload data through a browser to a cloud database (so no need to manage a local system)
- No need to "design a database" before you use your data – just upload and get to work
- Write SQL queries, with some automated help and some guided help from experts
- Build on your own results
  - Output of one query can be the input to another
  - Encourages sharing and reuse
  - Avoids everyone on the team running the same data processing steps over and over
  - Provides provenance – "how did you get this result?"

Select from a list of English descriptions

Edit a Query

```
SELECT interpro_description, count(DISTINCT gene_id) as num_genes
FROM [koesterj@washington.edu].[809_interpro2GO0005515_sort.csv]
GROUP BY interpro_description
order by num_genes DESC
```

Your datasets
All datasets
Shared datasets
Recent activity... 0
Recently viewed »

Upload dataset
New query

YOUR TOP VIEWED
exon.feature.o... 38
Lengths_hotsp... 28
noncoding_po... 25
hotspot_desert... 17
HD_NC_overla... 16

POPULAR TAGS
protein 49
oceanography 23
fluorimeter 18
suna 18

GO0005515_domains Only you can view this

Last modified: Apr 21, 2011 2:34 PM koesterj@washington.edu

How many genes have which domain

Click here to add a tag

```
SELECT interpro_descripti
FROM [koesterj@washington
GROUP BY interpro_descrip
order by num_genes DESC
```

DATASET PREVIEW Rows 1 - 100 of 148

<< first < prev 1 2 3 4 5 nex

interpro_description
NULL
WD40/YVTN repeat-like-containing do
WD40 repeat-like-containing domain
Zinc finger RING/FYVE/PHD-type
WD40 repeat subgroup
Tetratricopeptide-like helical
WD40 repeat

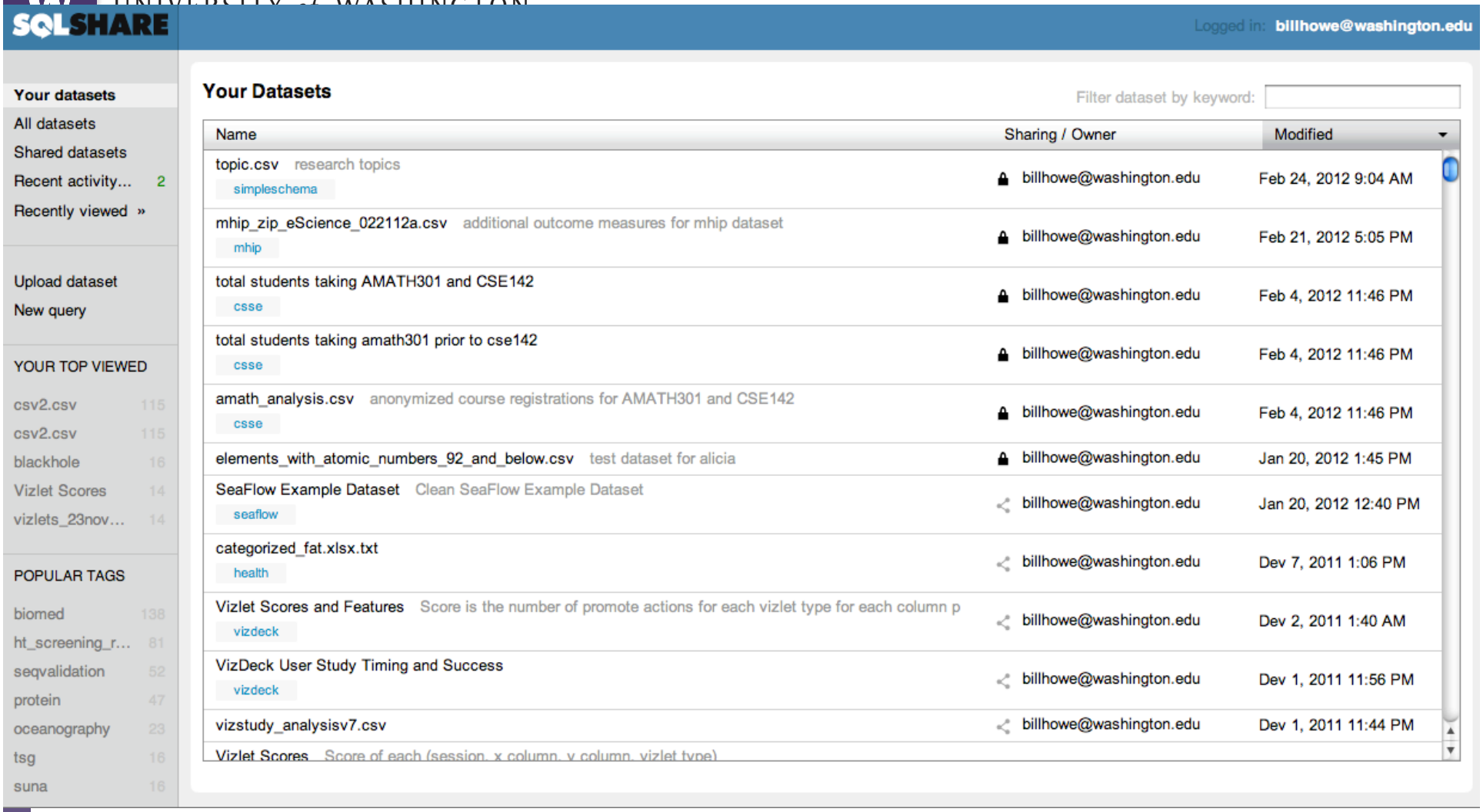
Save the results, share them with others

DATASET PREVIEW Rows 1 - 100 of 148 | Columns 2 of 2

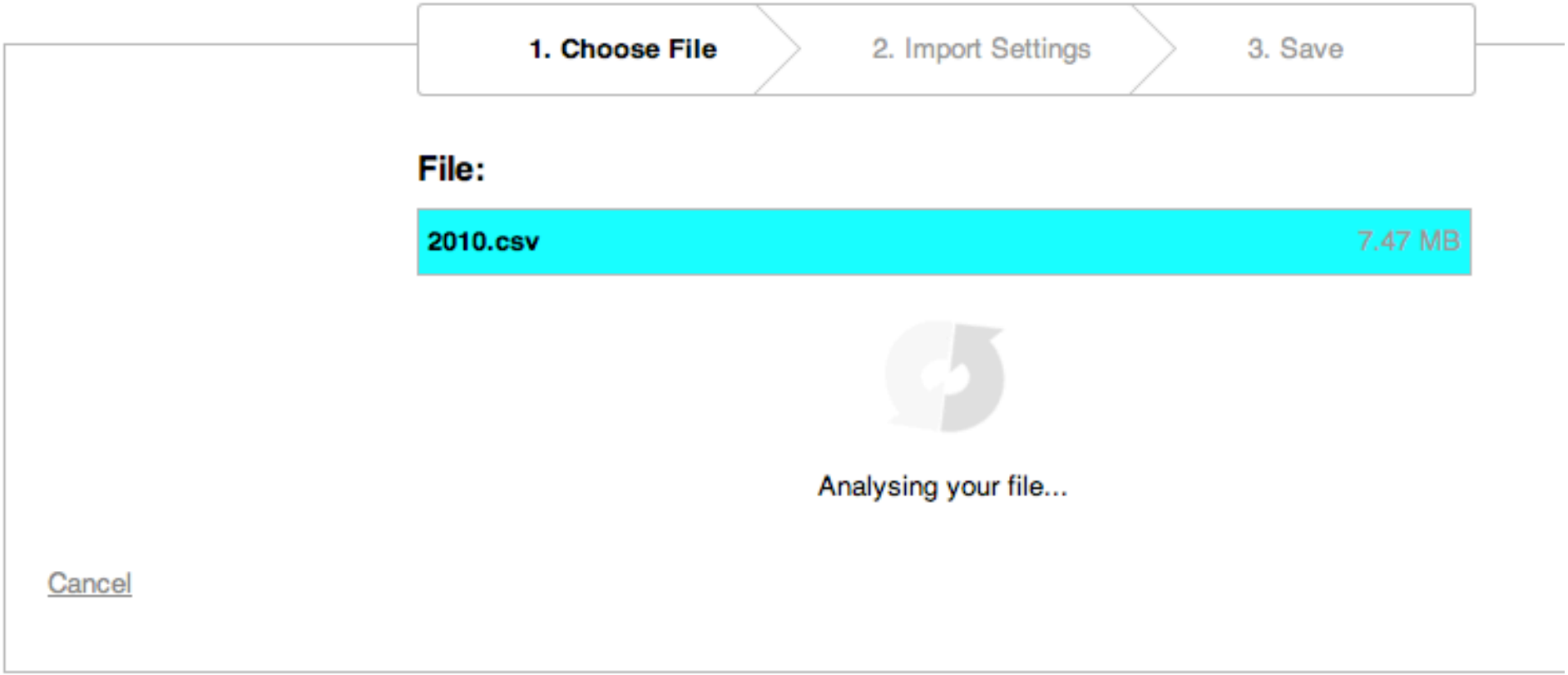
<< first < prev 1 2 3 4 5 next > last >>

interpro_description	num_genes
NULL	112
WD40/YVTN repeat-like-containing domain	18
WD40 repeat-like-containing domain	14
Zinc finger RING/FYVE/PHD-type	13
WD40 repeat subgroup	13
Tetratricopeptide-like helical	13
WD40 repeat	12
Tetratricopeptide repeat	11
Tetratricopeptide repeat-containing	9
WD40-repeat-containing domain	9

UNIVERSITY of WASHINGTON



http://sqlshare.escience.washington.edu



http://sqlshare.escience.washington.edu

Upload Dataset

1. Choose File | **2. Import Settings** | 3. Save

Dataset was imported with the following settings:

You can change the parser options if your data was not properly imported.

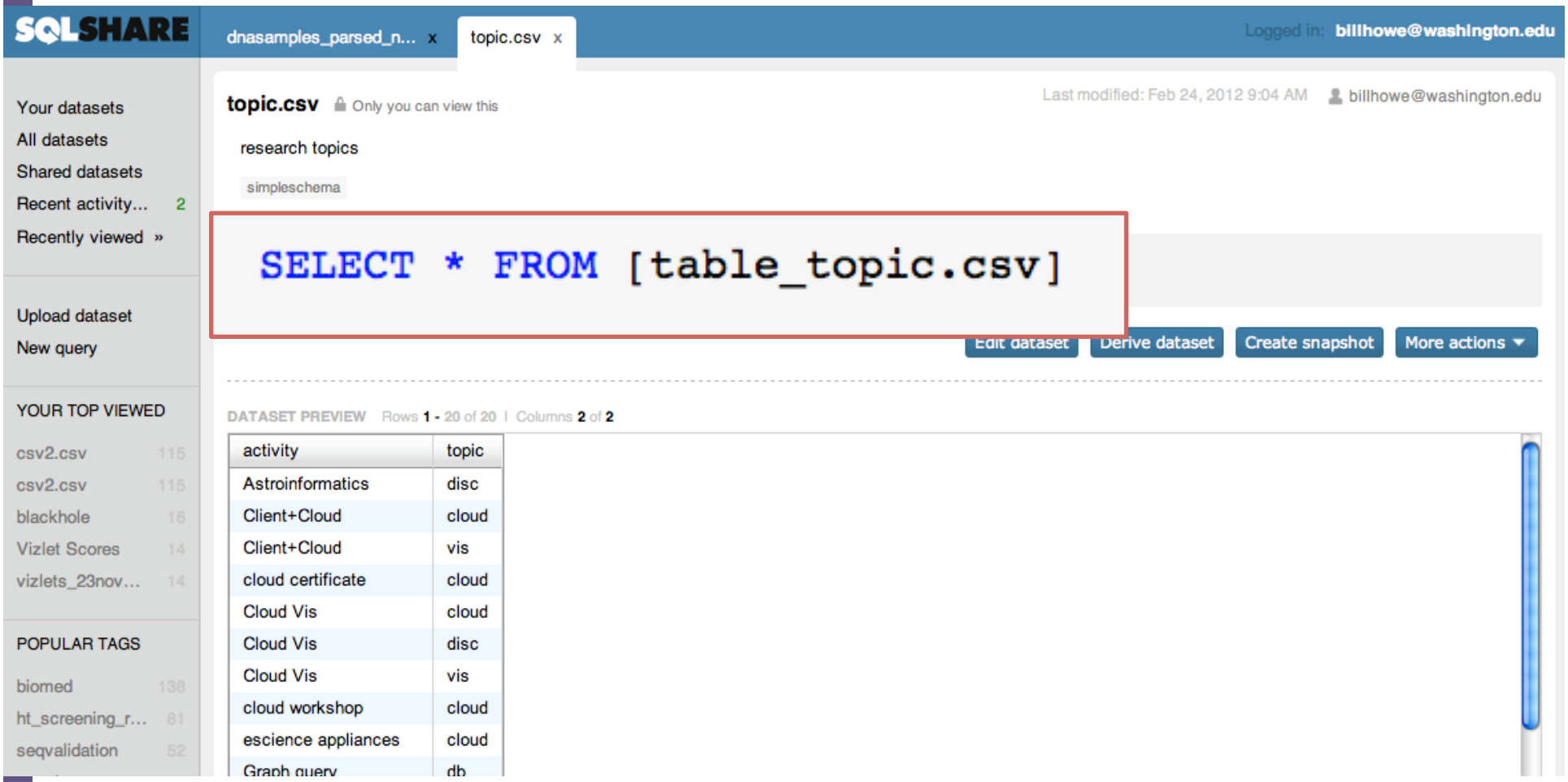
☑ Contains column header | Values are separated by ,

DATASET PREVIEW (Imported table with **3 columns**)

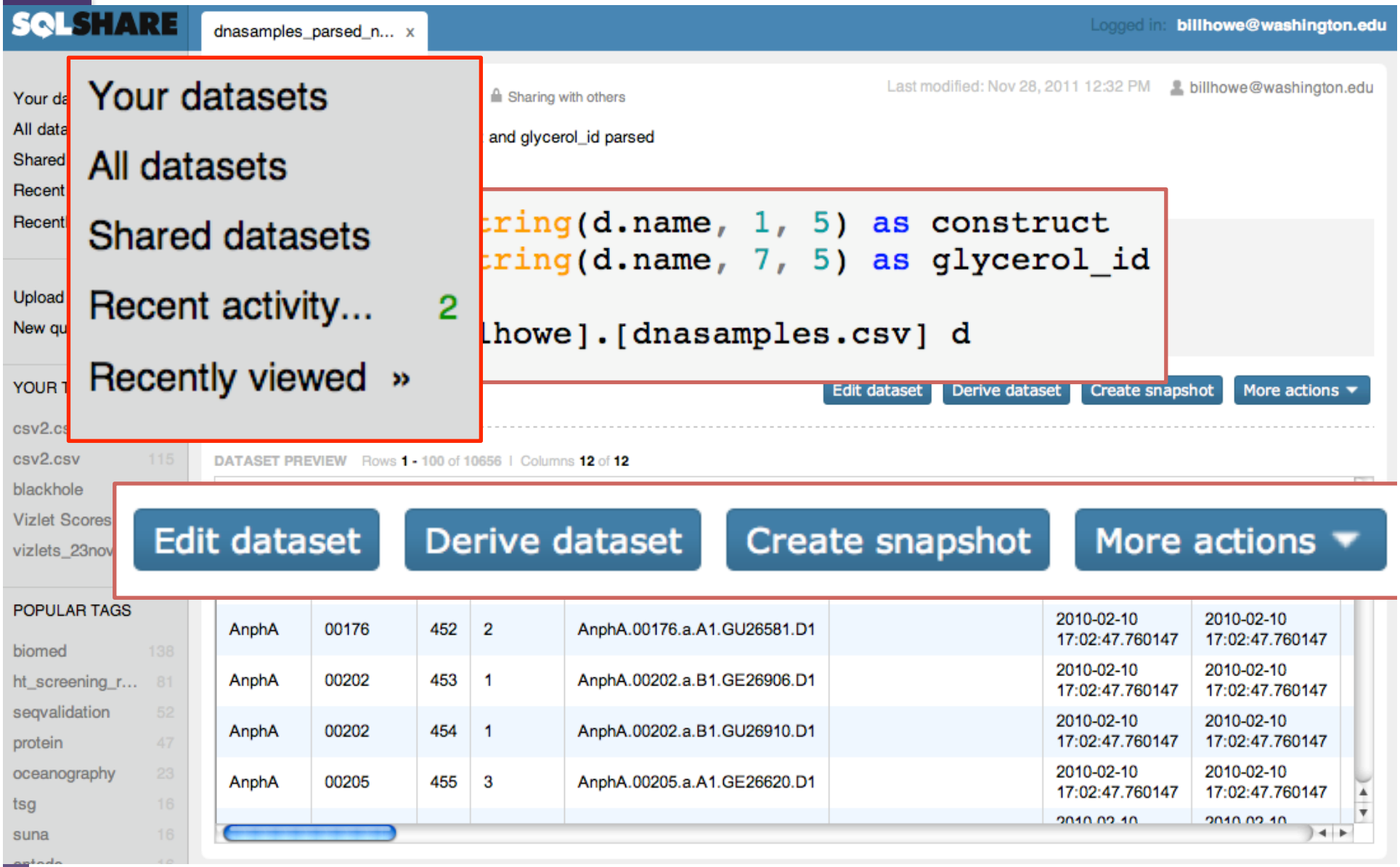
activity	thrust	time in past 12 months
SQLShare Engineering	long-tail	1
SQLShare research	long-tail	1
Client+Cloud	long-tail	1
HaLoop	scalable analytics	1.5
Cloud Vis	scalable analytics	1

Cancel Back Next

http://sqlshare.escience.washington.edu



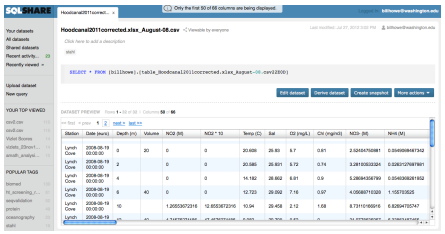
http://sqlshare.escience.washington.edu



http://sqlshare.escience.washington.edu

UNIVERSITY of WASHINGTON

amazon web services™ | EC2



VizDeck

"Flagship" SQLShare App (Python) on EC2

Spreadsheet Crawler

Excel Addin

Python Client

R Client

ASP.NET

SQLShare REST API

OAuth2

WCF

Windows Azure™

Microsoft® SQL Azure™

BACKUP SLIDES

Term frequency, inverse document frequency (tf-idf)

a(row_id, row_v)
row_v is a document vector

```
SELECT a1.row_id AS document_i
, a2.row_id AS document_j,
(a1.row_v * a2.row_v) /
((a1.row_v * a1.row_v) * (a2.row_v * a2.row_v)) AS theta

FROM a AS a1, a AS a2
WHERE a1.row_id > a2.row_id
```