UNIVERSITY of WASHINGTON eScience Institute ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

SQLSHARE Results from a Multi-Year SQL-as-a-Service Experiment

SHRAINIK JAIN

DB DAY 2015

S Jain, D Moritz, B Howe, et al. SQLShare: Results from a Multi-Year SQL-as-a-Service Experiment, **SIGMOD16**.

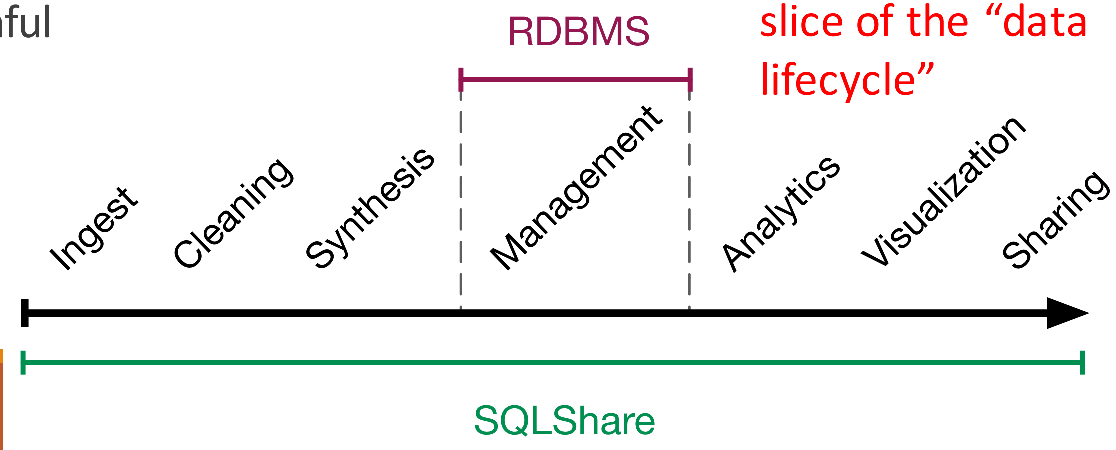
Part 1: SQLShare, DBaaS for Scientists

Databases are great, but....

They are underused in science workloads (physical, life and social sciences)

- Why?
  - Is there a fundamental mismatch in Data Model?
  - "Scientists can't write SQL"?
  - Other barriers to adoption?
    - Pre-engineered Schemas: Don't usually exist, given the ad-hoc nature of research.
    - Clean data a pre-condition: But real data is messy!
    - A database can make it harder rather than easier to share results.
    - Provenance: Tracking the history of operations is painful
    - Low data lifetime.

Databases only support a small slice of the "data lifecycle"



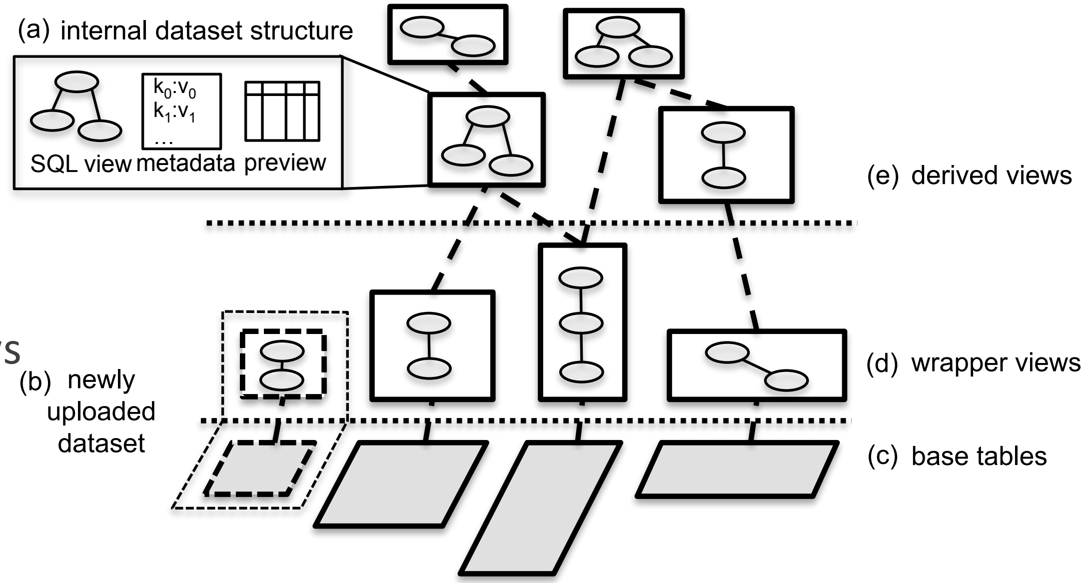
SQLShare: Making databases easier to use

Relaxed Schemas

- Infer data types automatically.
- Tolerate errors. For example:
  - use NULLs for the case of variable number of columns per row.
  - No column names
  - …

Provenance and Sharing

- Views as first class citizens
- Checkpoint operations as derived datasets or views



SQLSHARE

3) Share the results

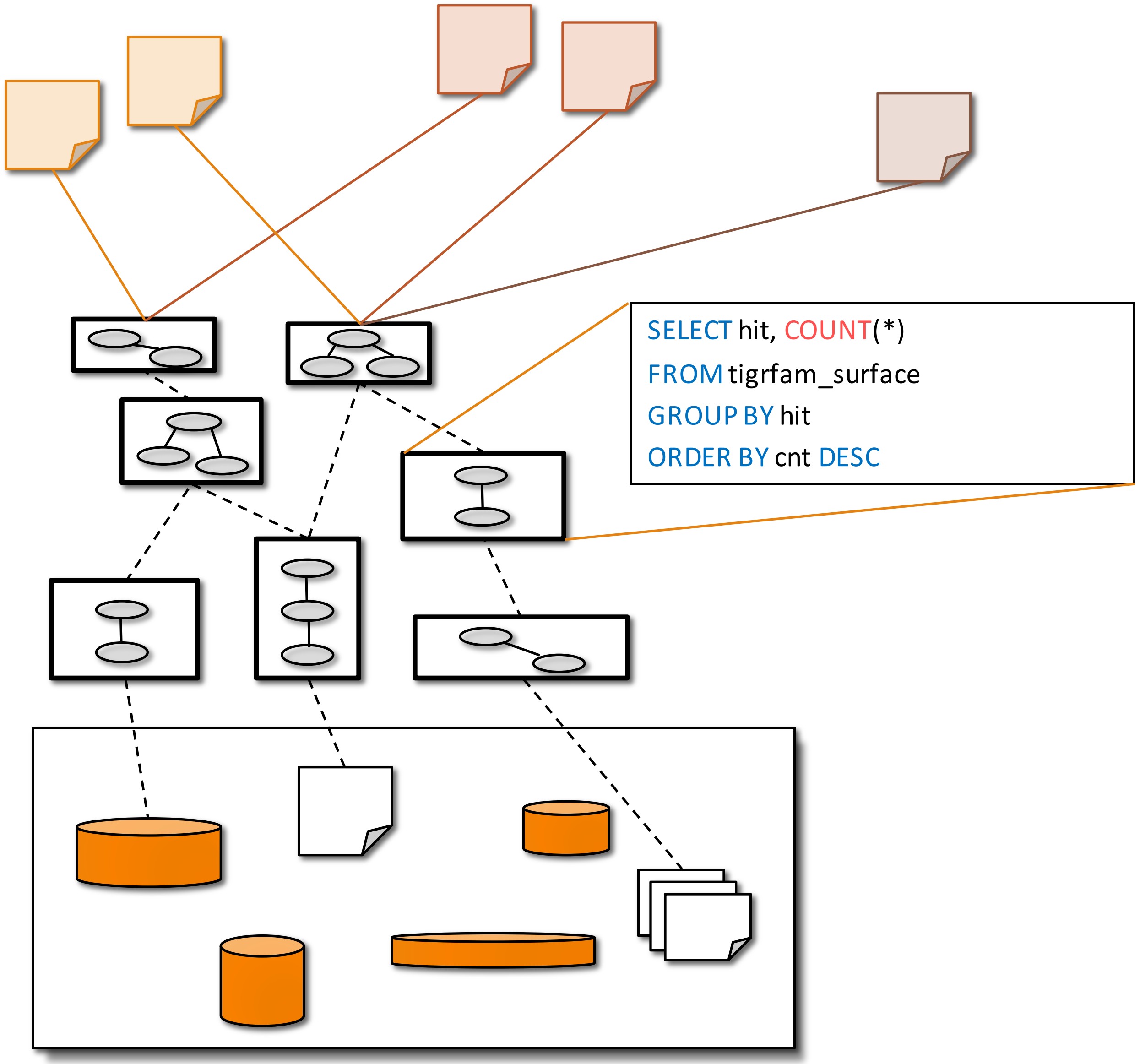
Make them public, tag them, share with specific colleagues – anyone with access can query

2) Write Queries

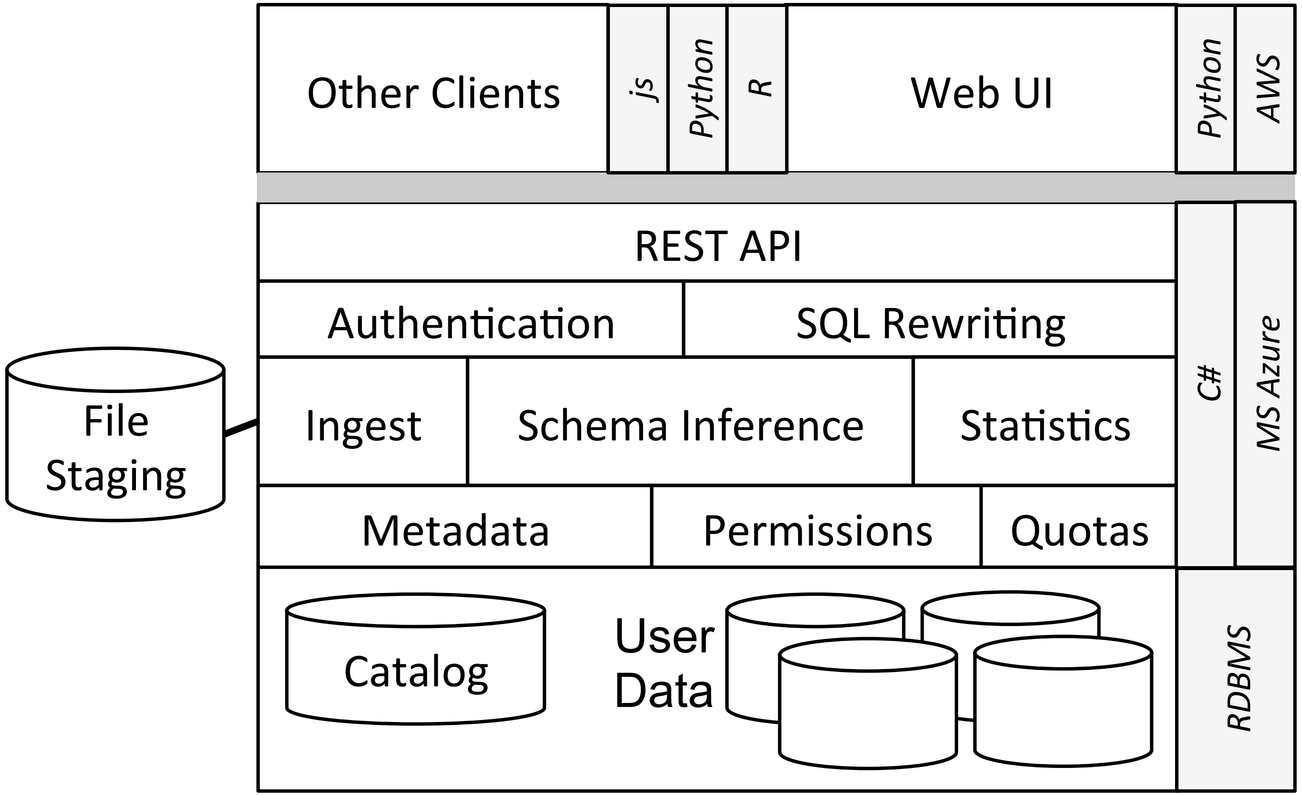
Right in your browser, writing views on top of views on top of views ...

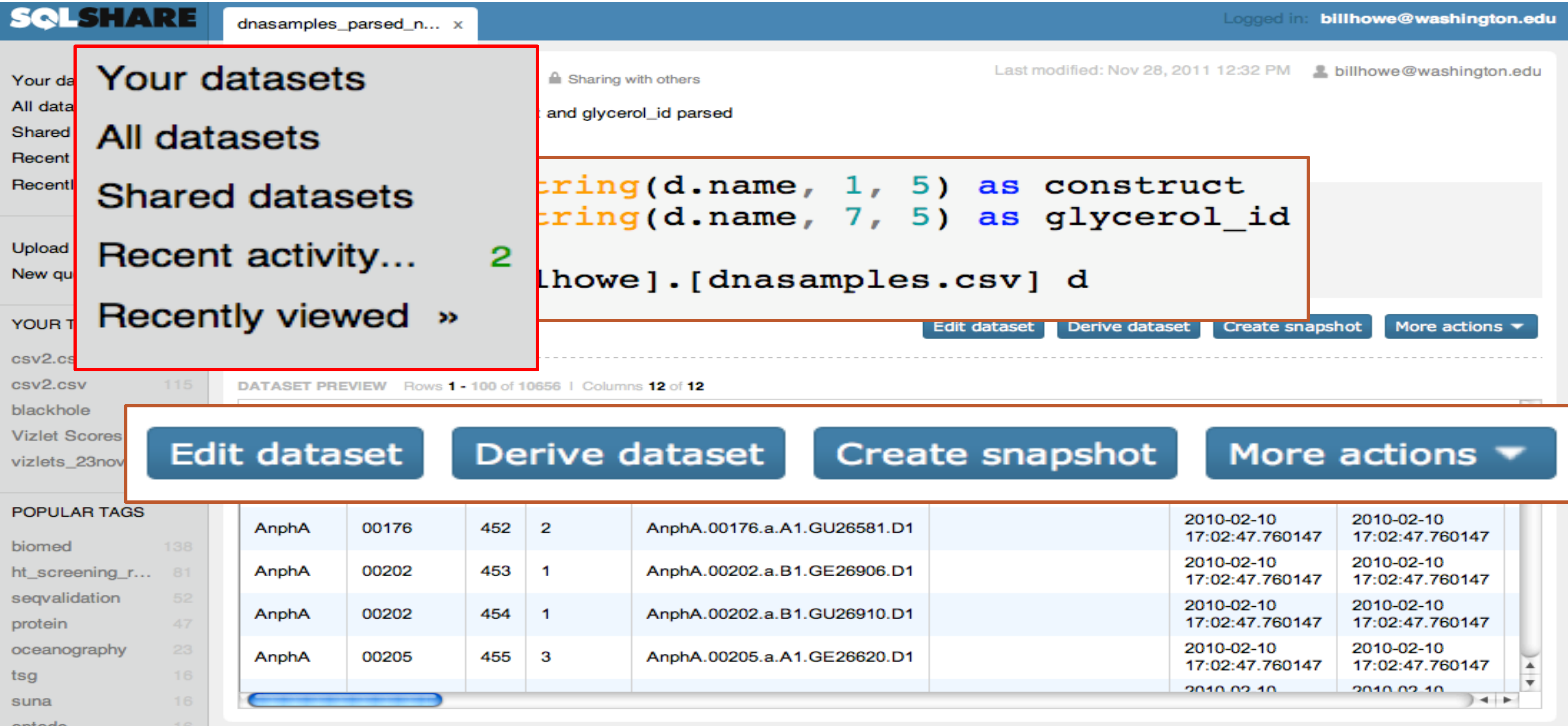
1) Upload data "as is"

Cloud-hosted, secure; no need to install or design a database; no pre-defined schema; schema inference; some itegration

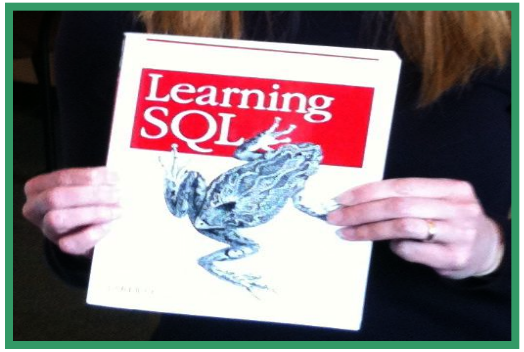


SQLShare


Other Clients
js
Python
R
Web UI
Python
AWS
REST API
Authentication
SQL Rewriting
File Staging
Ingest
Schema Inference
Statistics
C#
MS Azure
Metadata
Permissions
Quotas
Catalog
User Data
RDBMS



http://sqlshare.escience.washington.edu



Example: Computing the overlaps of two sets of blast results

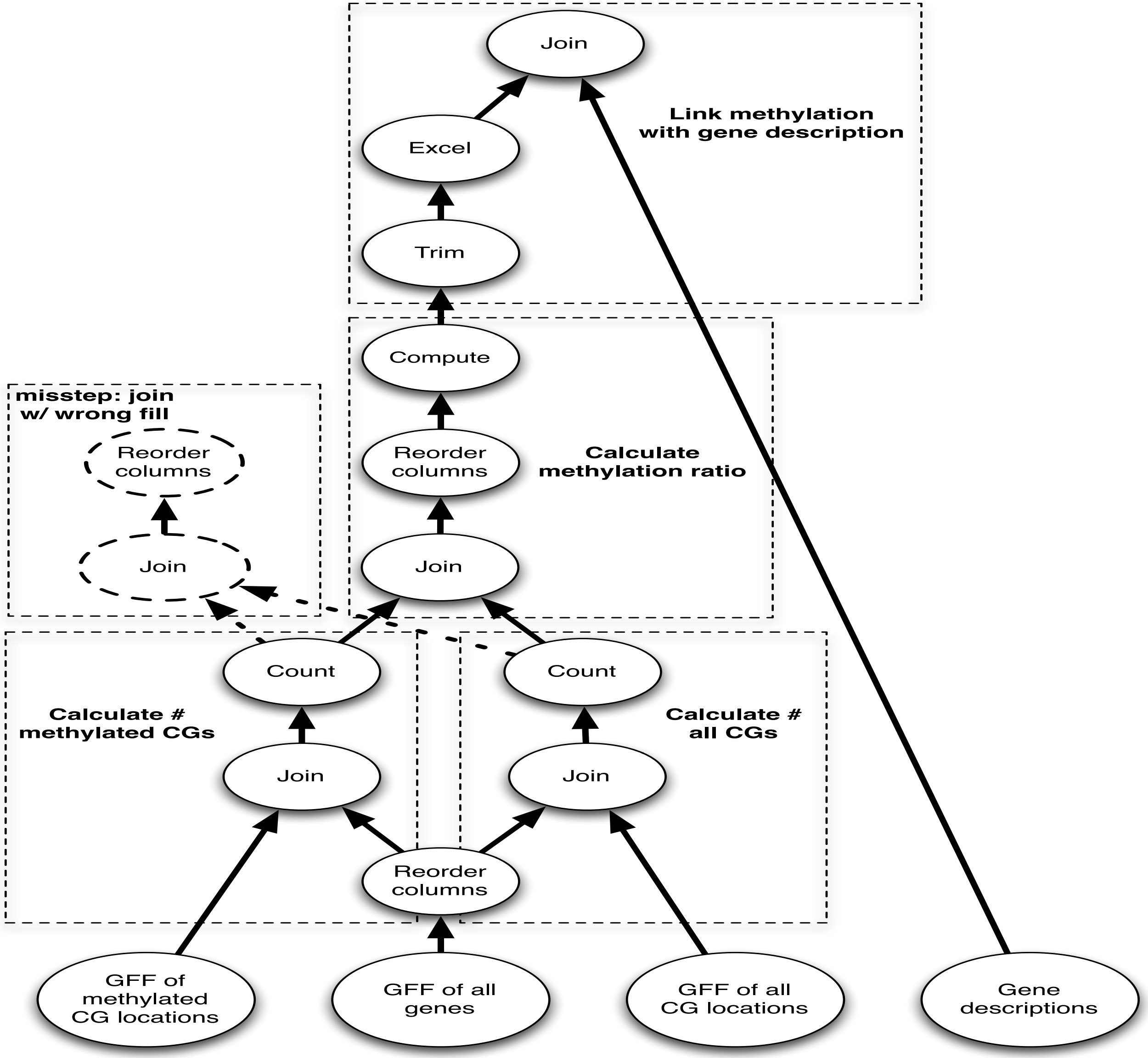
```
SELECT x.strain, x.chr, x.region as snp_region,  x.start_bp as snp_start_bp
    , x.end_bp as snp_end_bp,  w.start_bp as nc_start_bp, w.end_bp as nc_end_bp
    , w.category as nc_category
    , CASE WHEN (x.start_bp >= w.start_bp AND x.end_bp <= w.end_bp)
   THEN x.end_bp - x.start_bp + 1
   WHEN (x.start_bp <= w.start_bp AND w.start_bp <= x.end_bp)
   THEN x.end_bp - w.start_bp + 1
   WHEN (x.start_bp <= w.end_bp AND w.end_bp <= x.end_bp)
   THEN w.end_bp - x.start_bp + 1
 END  AS len_overlap

 FROM [koesterj@washington.edu].[hotspots_deserts.tab]  x
 INNER JOIN [koesterj@washington.edu].[table_noncoding_positions.tab]  w
 ON x.chr = w.chr
 WHERE (x.start_bp >= w.start_bp AND x.end_bp <= w.end_bp)
 OR (x.start_bp <= w.start_bp AND w.start_bp <= x.end_bp)
 OR (x.start_bp <= w.end_bp  AND w.end_bp  <= x.end_bp)
 ORDER BY x.strain, x.chr ASC, x.start_bp ASC
```

We see thousands of queries written by non-programmers

Howe, et al., CISE 2013

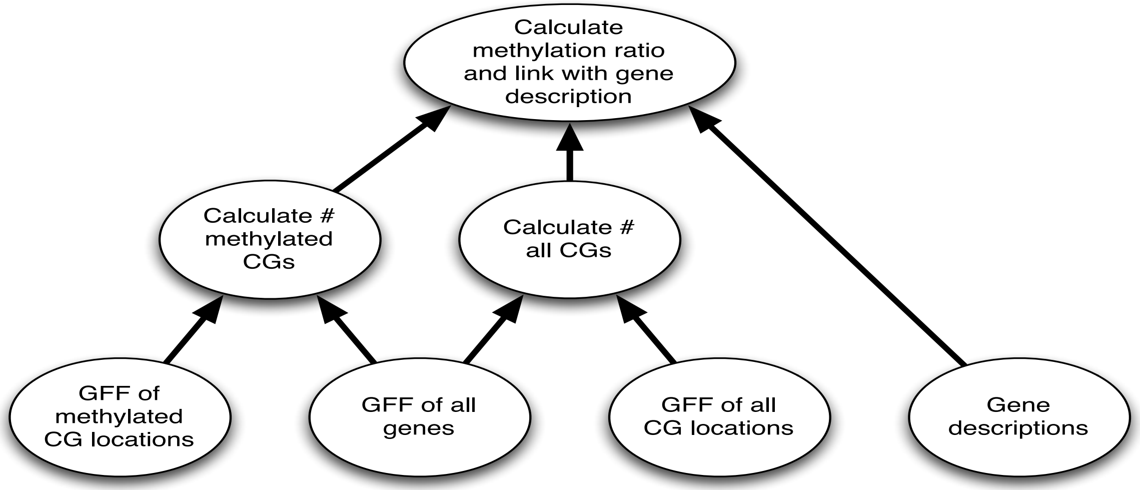
```r
table <- read.csv(‘‘table.csv’’)
# define 3 min time intervals
breaks <- seq(
          min(table$time),
          max(table$time),
           by=3)
# bin the table according to the breaks
b <- cut(table$time, breaks=breaks)

# calculate the mean of each variable
b.time <- tapply(table$time, b, mean))
b.Fluo <- tapply(table$Fluo, b, mean))
b.Temp <- tapply(table$Temp, b, mean))
b.Oxyg <- tapply(table$Oxyg, b, mean))
b.Nitr <- tapply(table$Nitr, b, mean))
b.Lat  <- tapply(table$Lat, b, mean))
b.Lon  <- tapply(table$Lon, b, mean))

binned.table <- data.frame(
          cbind(b.time, b.Fluo, b.Temp,
                b.Oxyg, b.Nitr,
                b.Lat, b.Lon))
write.csv(binned.table,‘‘binned.csv’’)
```

```sql
WITH data AS
  (SELECT * FROM [table.csv]),
     -- compute the minimum timestamp
     bounds AS
  (SELECT min(time) AS mintime FROM data),
     -- assign each timestamp a bin
     binned AS
  (SELECT bounds.mintime +
          floor((data.time - bounds.mintime)/3.0) *
          3.0 as binid
     FROM data, bounds)
-- compute the average of each bin
SELECT binid
  , avg(Fluo) as Fluo
  , avg(Temp) as Temp
  , avg(Oxyg) as Oxyg
  , avg(Nitr) as Nitr
  , avg(Lon) as Lon
  , avg(Lat) as Lat
  , avg(time) as time
FROM binned
GROUP BY binid
ORDER BY binid asc
```



Steven Roberts

SQL as a lab notebook:
http://bit.ly/16Xj2JP



Galaxy

Popular service for Bioinformatics Workflows

SQLSHARE

Did it work?

SQLShare used for real life work:

- Multiple labs used it and liked it.
- And wrote increasingly complex queries.
- And wrote queries on dirty data.
  - And even used SQL to cleanup data!

SQLShare Queries were diverse.

Attracted new kinds of ad-hoc queries

- That were written to replace files and scripts

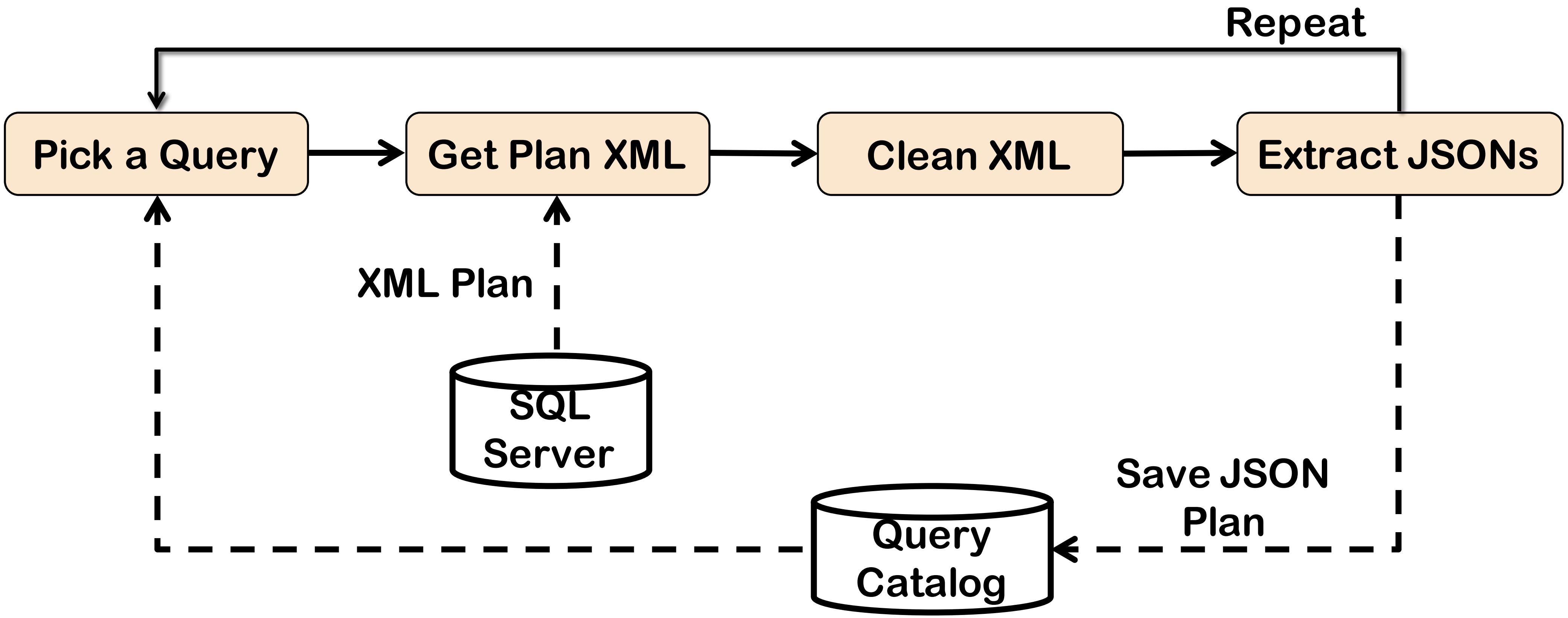
SQLShare Attracted High-Churn Work

- Varying data lifecycle.
- Quick insights from dirty data

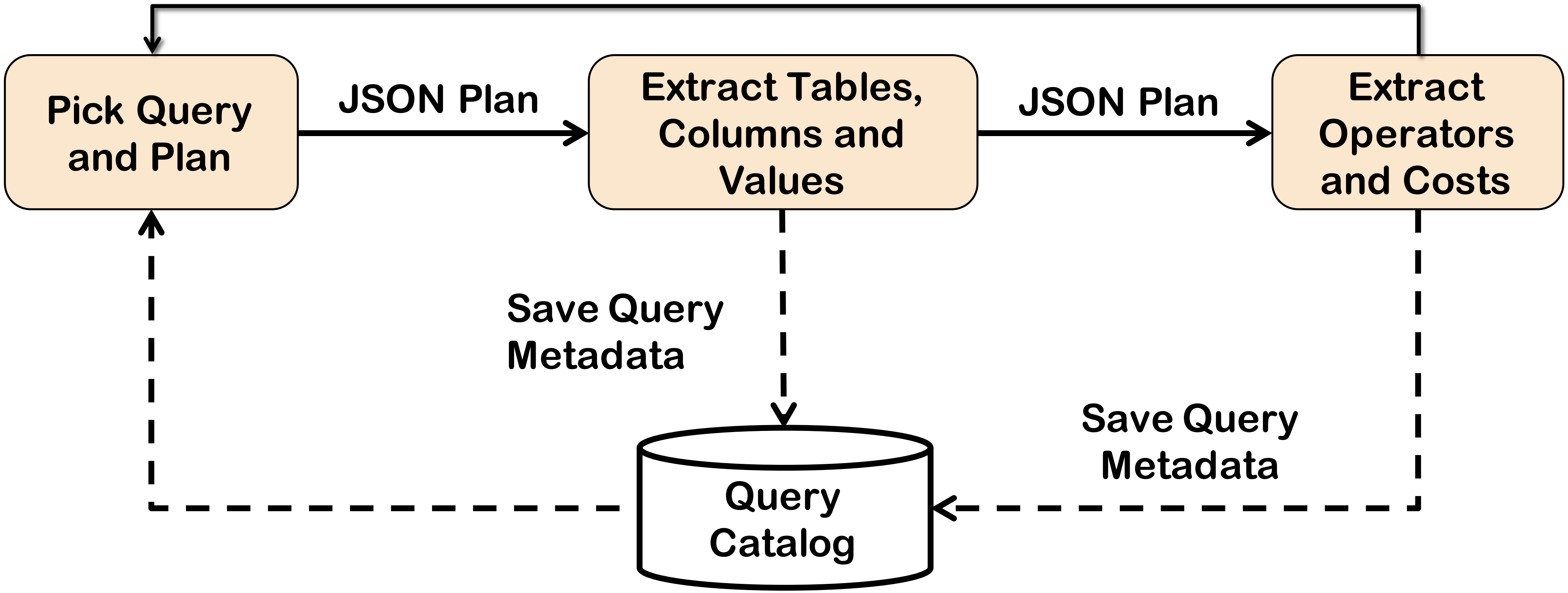
Part 2: SQLShare Workload Analysis

… Or reasons why this is potential benchmark dataset.

Analyzing the corpus: Methodology



Analyzing the corpus: Methodology



SQLShare query corpus

Workload Metadata

Measure	Value
Users	591
Tables	3891
Columns	73070
Views	7958
Non-trivial Views	4535
Queries	25052

Query Metadata

Feature	Average Value
Length	217.32
Runtime	3175.38 s
# of operators	18.12
# of Distinct Operators	2.71
# of Tables accessed	2.31
# of Columns accessed	16.22
# of Queries per Table	12

Results from SQLShare Workload analysis

SQLShare Queries are *Complex*

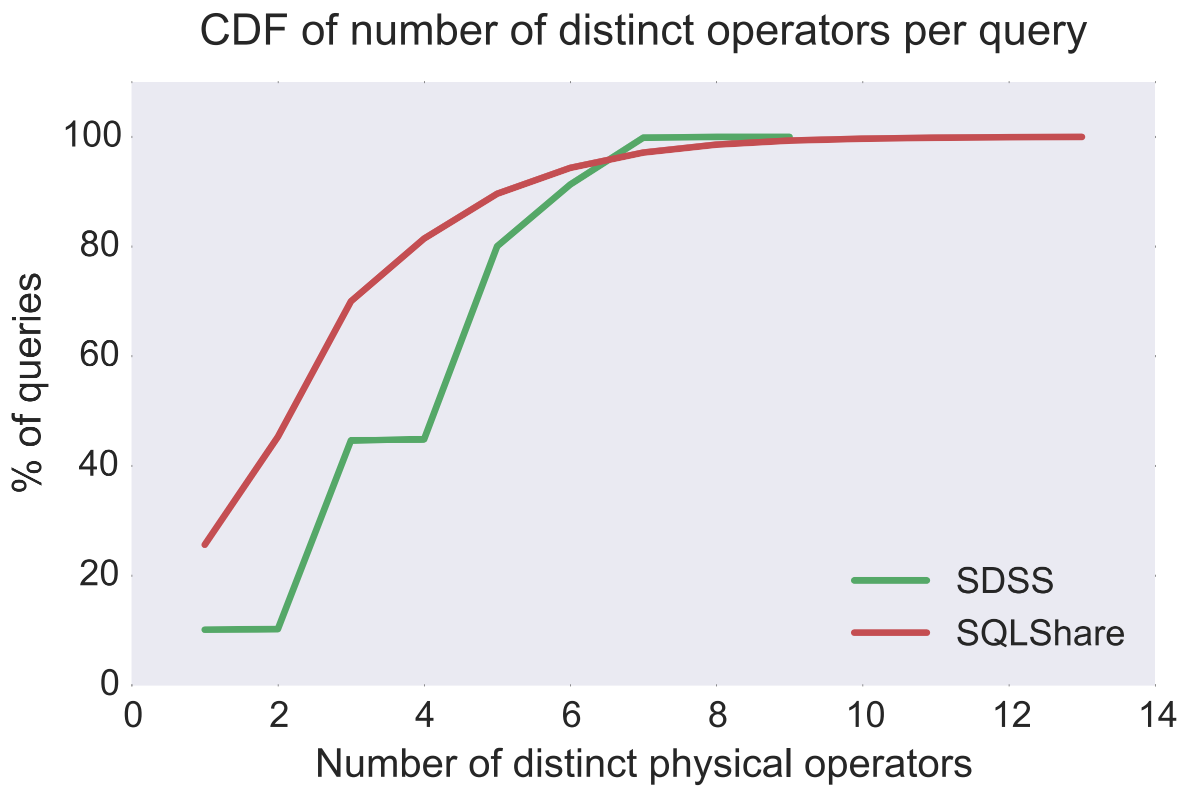
SQLShare Queries are *Diverse*

Views Afford Controlled *Data Sharing*

Dataset Permanence Varies by User: Ad hoc data analytics often deals with *low lifetime datasets*

SQLShare Attracts High-Churn Work

Results: SQLShare Queries are Complex



Lots of simple queries, but huge complexity in the tail!

Results: SQLShare Queries are Diverse

Diversity:

- Percentage of ‘unique’ queries.

How do we define Uniqueness?

- Naïve measure:
  - ASCII string uniqueness
- If two queries reference different sets of attributes?
- Query plan template uniqueness?

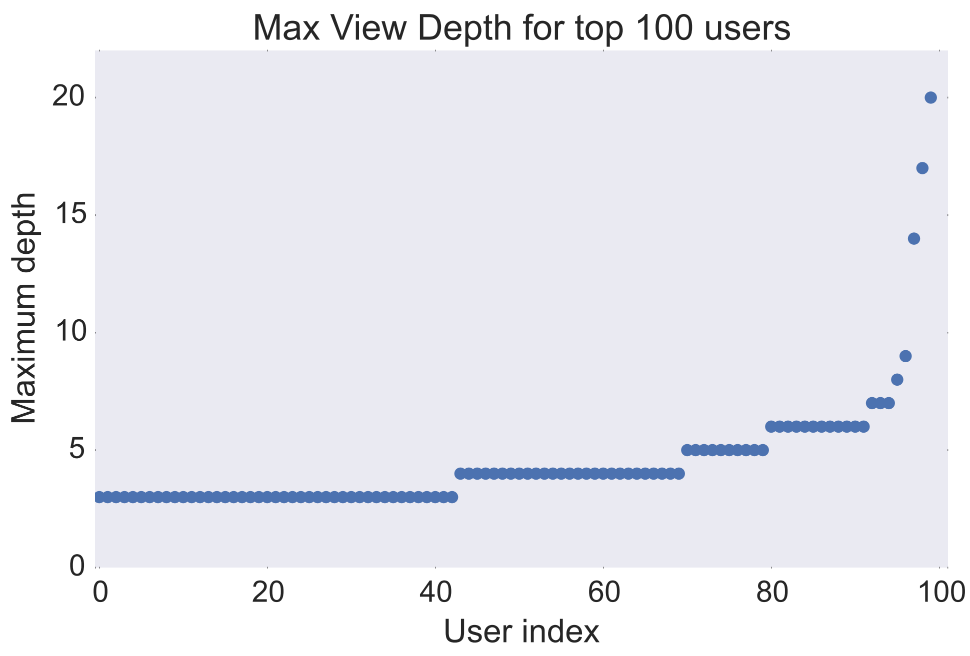
Results: SQLShare Queries are Diverse

Diversity Metric	SDSS	SQLShare
Total Queries	7M	25052
String distinct queries	200K	24096
Column distinct queries	467	10928
Distinct query template	686	15199

SQLShare queries are more diverse and have 63% distinct query templates. In contrast, SDSS has only 0.3% distinct query templates

Results: Views Afford Controlled Data Sharing & Provenance

- The view-centric data model also affords collaboration: users can share the derived dataset (and its provenance)
- About 56% of the datasets in the system are derived from other datasets using views.
- 37% of the datasets in SQLShare are public.
- 10% of the queries logged in the system access datasets that the query author does not own.



Results: Dataset Permanence Varies by User

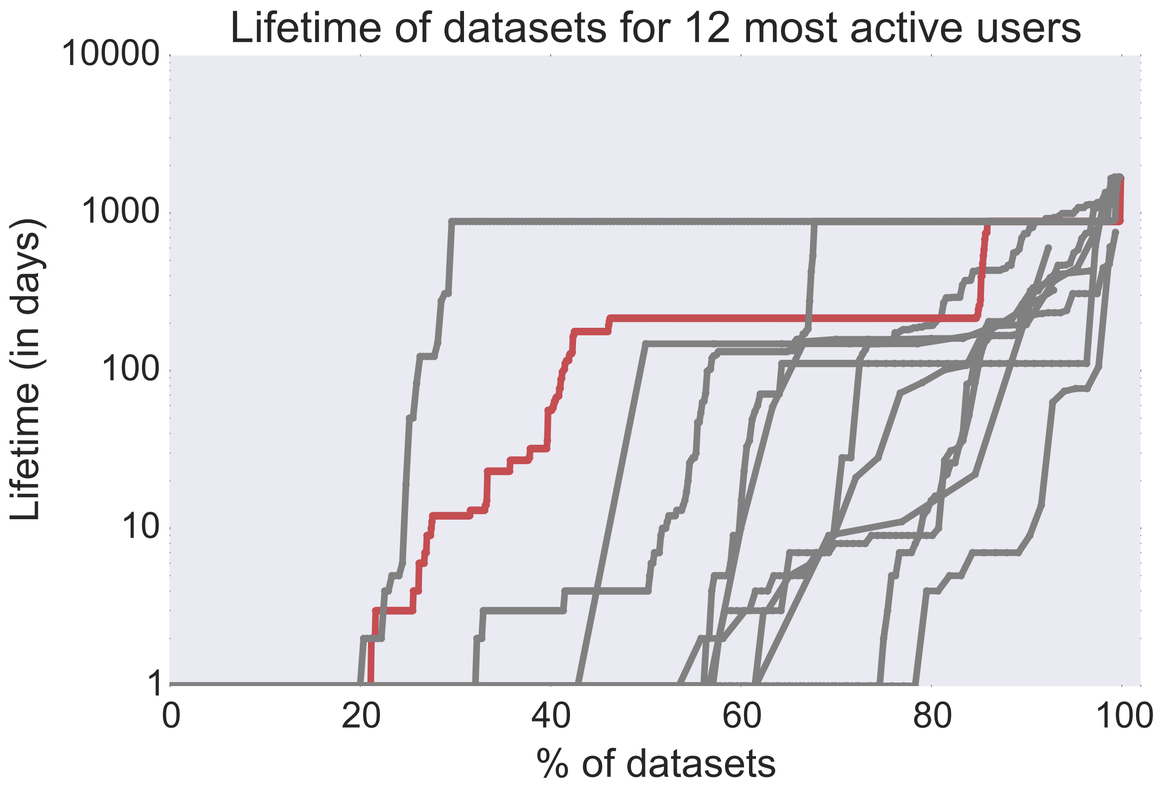
Many users are operating in short-duration analysis loops, where they upload some data, write a few queries, and then move on to another task.

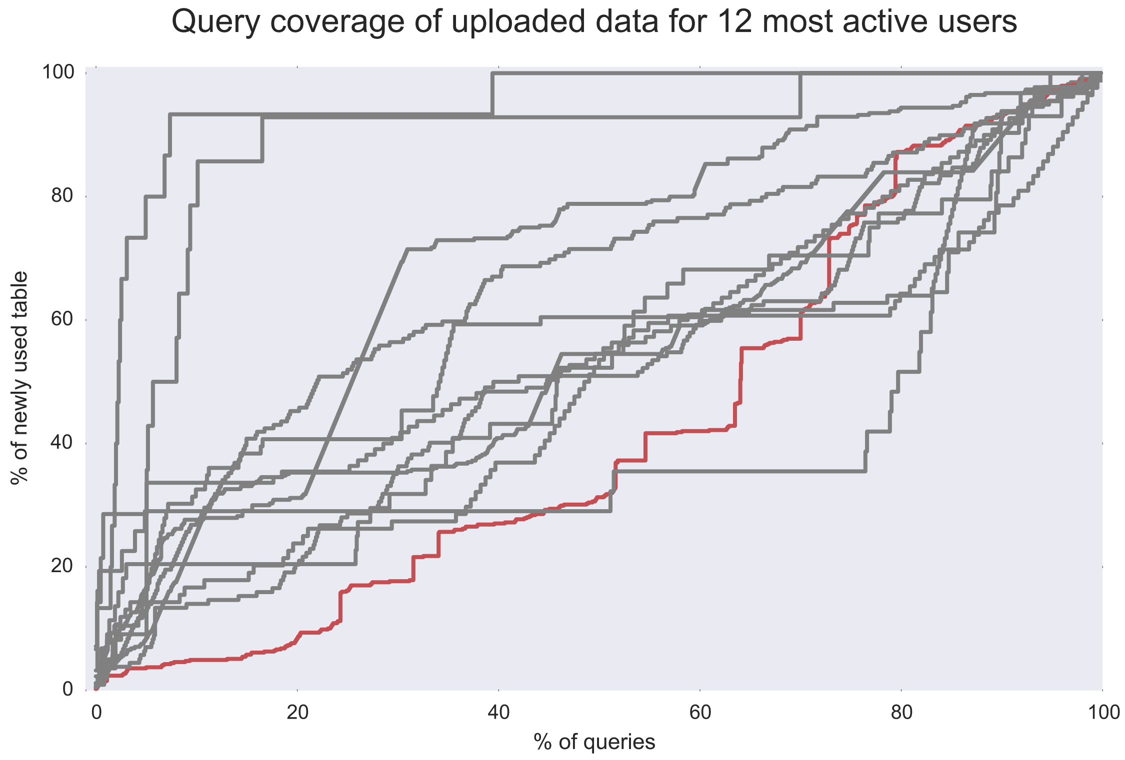
- How do we measure this?

Dataset Lifetime:

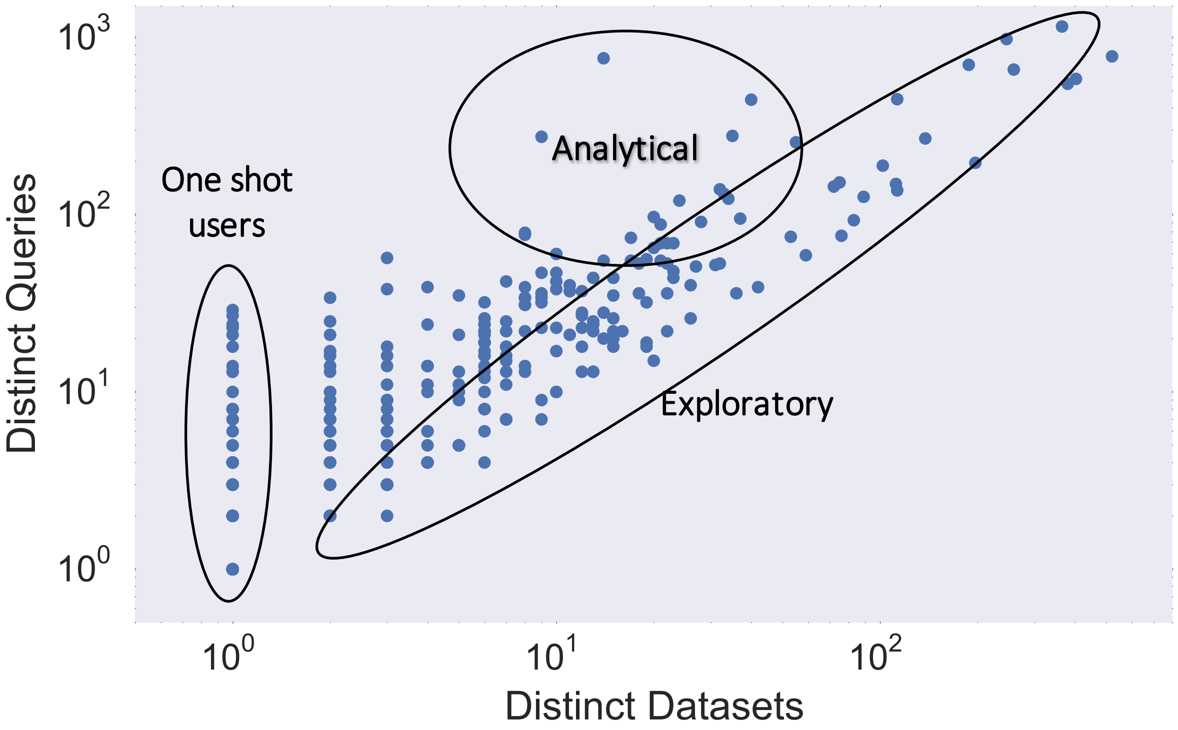
- *The difference in days between the first and the last time that dataset was accessed in a query*

Results: Dataset Permanence Varies by User





Results: SQLShare Attracts High-Churn Work



Towards a Benchmark Dataset

We're releasing the data, queries, and views as a research corpus

https://uwescience.github.io/sqlshare/data_release.html

Future work

Identify query idioms:

- Cleanup tasks.
- Binning.
- …

Publish the SQLShare dataset.

- A benchmark of ‘real’ queries.
  - Captures complex data & complex tasks.

Enable more real-life science operations:

- Write SQL to do more.
  - Matrix multiplication?
  - Automatic conversion from RA to SCIDB AFL.

Thanks!