DataHub: Collaborative Data Science and Dataset Version Management at Scale

Aditya Parameswaran

U Illinois
Deep, Dark Secrets of Data Science

Many ad hoc data management systems (e.g., Dropbox) being used

Much of the data is unstructured so typically can’t use DBs

The process of data science itself is quite ad hoc and exploratory

Scientists/researchers/analysts are pretty much on their own

![Cartoon](https://xkcd.com/)

**Pro-tip:** Never look in someone else’s documents folder.

Courtesy: XKCD
How bad could dataset management get?
The Investigator Team

Aaron Elmore
Aditya Parameswaran
Amol Deshpande
Sam Madden

Chicago
Illinois
Maryland
MIT

Amit Chavan
Anant Bhardwaj
Shouvik Bhattacharerjee
A True (Horror) Story of Dataset Management

Before
What did we learn?

We use about 100TB of data across 20-30 researchers.

We spend a LOT of money on this.

Everything is organized around shared folders, and everyone has access.

Our dataset management scheme is so simple, it’s great!
What did we learn?

So how do users work on datasets?

They typically make a private copy.

But wouldn’t that mean lots of redundant versions and duplication?

Yes. That’s why our storage is 100TB.

I: Massive redundancy in stored datasets

ARE YOU SERIOUS?
What did we learn?

Do you have datasets being analyzed by multiple users simultaneously?

Sure, but we have no way of knowing or resolving modifications.

But wouldn’t that mean you cannot combine work across users?

True. The users will need to discuss.

I: True collaboration is near impossible!

ARE YOU SERIOUS?
What did we learn?

Do you get rid of redundant datasets, given that you have space issues?

All the time!

What if the user had left, and if the dataset is crucial for reproducibility?

We cross our fingers!

III: Unknown dependencies between datasets

ARE YOU SERIOUS?
What did we learn?

Is there any way users can search for specific dataset versions of interest?

Not really. They talk to me.

What if you leave?

Let’s pray for the group’s sake that that doesn’t happen!

IV: No organization or management of dataset versions.
What did we learn?

The four

1. Massive redundancy in stored datasets
2. Truly collaborative data science is impossible
3. Unknown dependencies between dataset versions
4. No efficient organization or management of datasets
Happens all the time...

Every collaborative data science project ends up in dataset version management hell

Surely, there must be a better way?

1. Massive redundancy in stored datasets
2. Truly collaborative data science is impossible
3. Unknown dependencies between dataset versions
4. No efficient organization or management of datasets
Have we seen this before?

Analogous to management of source code before source code version control!

How about:
DataHub: a “GitHub for data”

Solving the “AYS” problems

1. Massive redundancy in stored datasets
2. Truly collaborative data science is impossible
3. Unknown dependencies between versions
4. No efficient organization or management

Compact storage
“Branching” allowed
Explicit and implicit
Rich retrieval methods
What about alternatives?

Many issues with directly using GitHub or SC-VC:

- Cannot handle large datasets or large # of versions
- Querying and retrieval functionality is primitive
- Datasets have regular repeating structure

Many issues with temporal databases: similar issues, plus one major one:

- Only supports a linear chain of versions
The Vision for DataHub

The

for collaborative data science and dataset version management

satisfying all your dataset book-keeping needs.
The Vision for DataHub

Basics:
• Efficient maintenance and management of dataset versions

DataHub will also have:
• A rich query language encompassing data and versions
• In-built essential data science functionality such as ingestion, and integration, plus API hooks to external apps (MATLAB, R, …)
Ingest (Import)

Database System

Version Management

Fork, Branch, Merge

Sharing, Collaboration

Query Language

Integrate / Visualize / Other Apps
DataHub Architecture

Data: Versioned Datasets

Metadata: Version Graphs, Indexes, Provenance

DataHub: A Collaborative Dataset Management Platform

Support for Data Science

Versioning API

Versioning QL

Client Applications

INGEST
INTEGRATE
OTHER

Dataset Versioning Manager

Client Applications
Data Model and Basic API

Flexible “Schema-later” Data Model

Groups of records with different schemas in same table

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>(Berkeley, 2003, Hellerstein)</td>
</tr>
<tr>
<td>Amol</td>
<td>(Berkeley, 2004, Hellerstein)</td>
</tr>
<tr>
<td>Aaron</td>
<td>(UCSB, 2014, El Abbadi and Agrawal)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>School</th>
<th>Year</th>
<th>Advisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam</td>
<td>Berkeley</td>
<td>2003</td>
<td>Hellerstein</td>
</tr>
<tr>
<td>Amol</td>
<td>Berkeley</td>
<td>2004</td>
<td>Hellerstein</td>
</tr>
<tr>
<td>Aaron</td>
<td>UCSB</td>
<td>2014</td>
<td>El Abbadi and Agrawal</td>
</tr>
</tbody>
</table>

Standard git commands: branch, commit, fork, merge, rollback, checkout
Storing and Retrieving Versions

Simplest Strawman Approach:

Store: For every version, store “delta” from previous DAG version
Retrieve: Start from version pointer, walk up to root

Q: Why store delta from the previous version?
Q: Why not materialize some versions completely?
Q: What kind of indexes should we use?
Branching and Merging

More questions than answers!

• Q: How do we allow users operate on servers and/or their local machines without missing updates?

• Q: What if the datasets are large? Can users work on samples?

• Q: How do we detect conflicts and allow users to merge conflicting branches with as little effort as possible?
Rich Query Language

Can combine versions and data!

```
SELECT * FROM R[V1], R[V4] WHERE R[V1].ID = R[V4].ID

SELECT VNUM FROM VERSIONS(R) WHERE EXISTS
(SELECT * FROM R[VNUM] WHERE NAME='AARON')
```

Other examples:

- All versions that are vastly different in size from a given version.
- The first version where a certain tuple was introduced
- All tuples that were introduced in a given version and subsequently deleted

Still a work in progress!
Screenshots
App: Ingest by Example

Paste data below (or, select an example data)

Example Data:

<table>
<thead>
<tr>
<th>Crime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Example Input

```json
[ "Reported crime in 'Alabama',
  +4029.3
  +3900
  +3937
  +3974.9
  +4081.9"
, "Reported crime in 'Alaska',
  +3370.9
  +3615
  +3582
  +3373.9
  +2928.3" ]
```

Example Output

```json
[ ["Alabama",+4029.3, +3900, +3937, +3974.9, +4081.9],
  ["Alaska", +3370.9, +3615, +3582, +3373.9, +2928.3] ]
```

Data

Reported crime in 'Alabama',

- 2004, +4029.3
- 2005, +3900
- 2006, +3937
- 2007, +3974.9
- 2008, +4081.9

Reported crime in 'Alaska',

- 2004, +3370.9
- 2005, +3615
- 2006, +3582
- 2007, +3373.9
- 2008, +2928.3

Example from Data Wrangler Paper
App: Automatic Visualization

Facets

- agency
- comprate
- name
- ot_earnings
- pay_basis
- title
- total_earning

SELECT
Papers in the works..

- **Fundamentals:**
  - *Blobs:* Exploring the trade-off between storage and recreation/retrieval cost for blob stores
  - *Relational:* Exploring SQL-based versioning implementations and indexing

- **Add-on functionality:**
  - *Ingest:* Ingest by example
  - *Viz:* Automatically generating query visualizations
To Summarize

• Dataset management as of today is bad, bad, bad

• DataHub is “GitHub for data”; an essential prerequisite to collaborative data science
  • Tracking, managing, reasoning about, and retrieving versions
  • Fundamental building block for study of other problems

• DataHub has in-built data science functionality, plus hooks
  • Ingestion: ingest by example
  • Integration: search, and auto-integrate
  • Provenance: explicit and implicit
  • Visualization: manual and automatic

Lots of related work!

Integrated with versioned storage
To find out more and contribute...

datahub.csail.mit.edu

Aditya Parameswaran

data-people.cs.illinois.edu