Towards a Unified Query Language for Provenance and Versioning

Amit Chavan

Amol Deshpande

Silu Huang

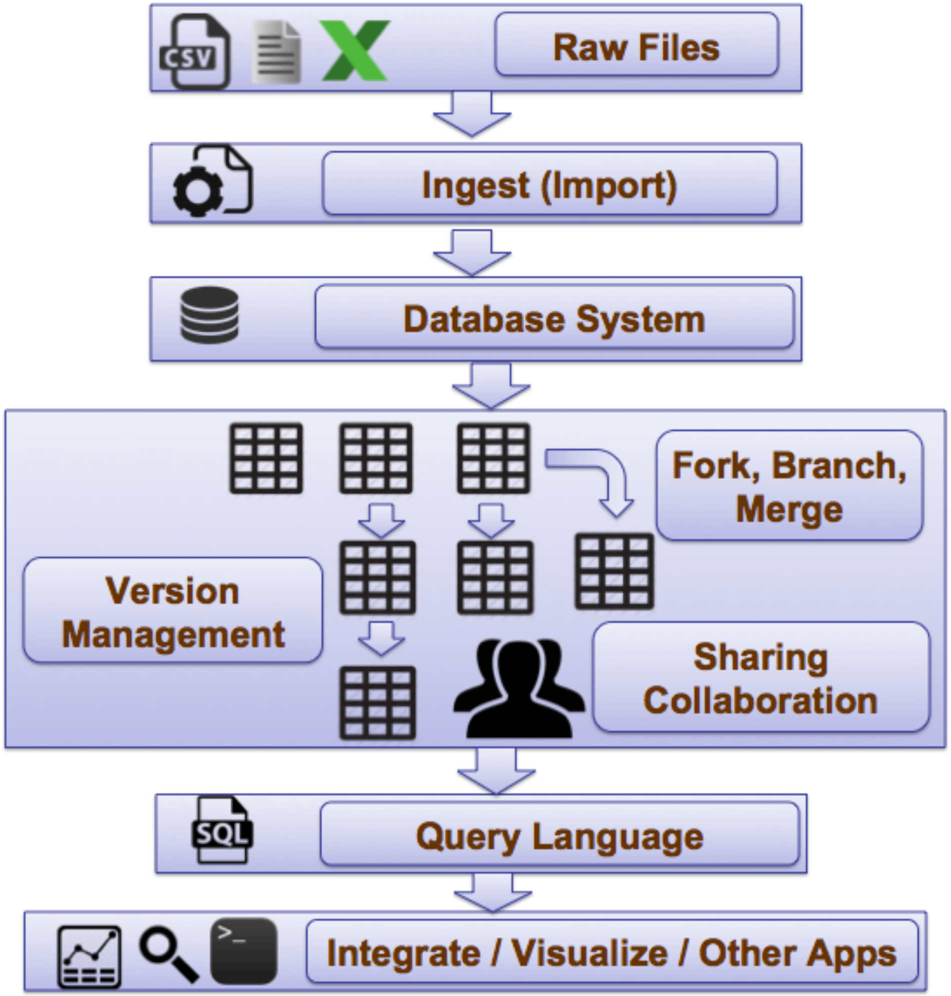
Aditya Parameswaran

Aaron Elmore

Samuel Madden

DATAHUB: A COLLABORATIVE HOSTED DATA SCIENCE PLATFORM

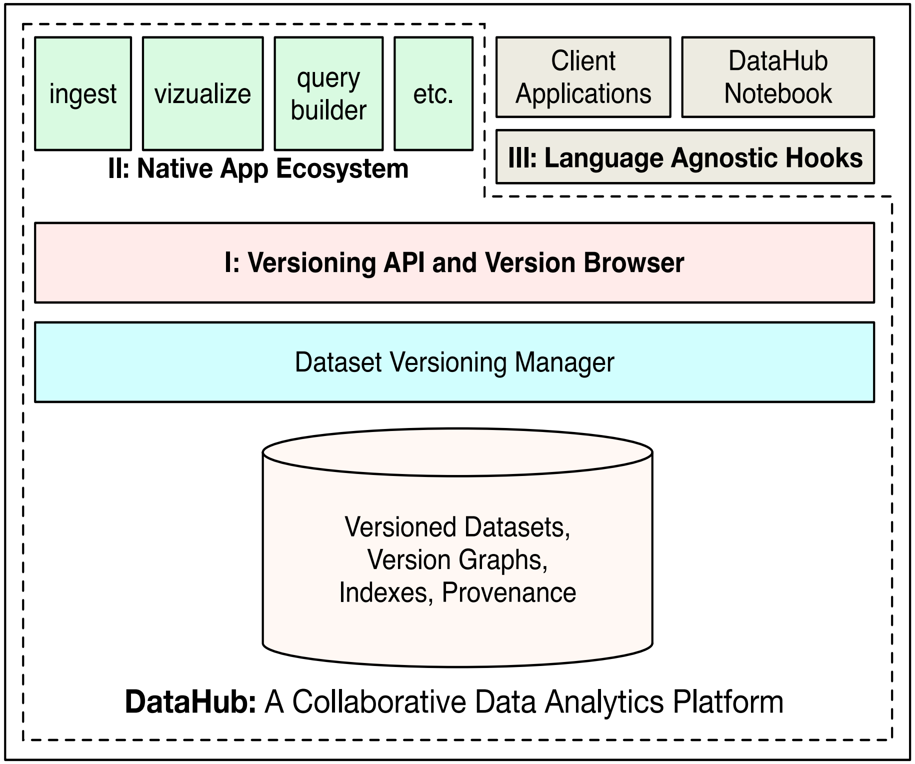
The one-stop solution for collaborative data science and dataset version management



http://data-hub.org

DATAHUB: A COLLABORATIVE HOSTED DATA SCIENCE PLATFORM

- a dataset management system – import, search, query, analyze a large number of (public) datasets
- a dataset version control system – branch, update, merge, transform large structured or unstructured datasets
- an app ecosystem and hooks for external applications (Matlab, R, iPython Notebook, etc)


ingest
vizualize
query builder
etc.
II: Native App Ecosystem
Client Applications
DataHub Notebook
III: Language Agnostic Hooks
I: Versioning API and Version Browser
Dataset Versioning Manager
Versioned Datasets, Version Graphs, Indexes, Provenance
DataHub: A Collaborative Data Analytics Platform

DataHub Architecture

CHALLENGES IN DATASET VERSION MANAGEMENT

Collaborative data science projects end up in dataset version management hell

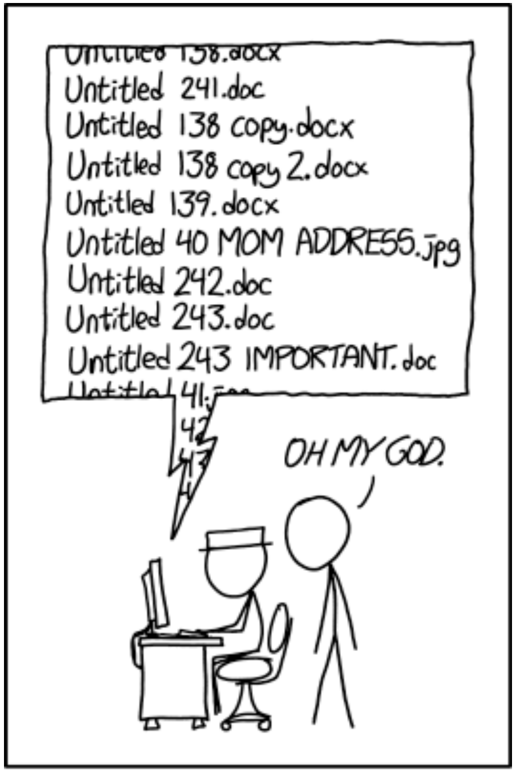
- Many private copies of the datasets ➔ Massive redundancy

- No easy way to keep track of dependencies between datasets

- Manual intervention needed for resolving conflicts

- No efficient organization or management of datasets

- No way to analyze/compare/query versions



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

Courtesy: XKCD

WHAT ABOUT GIT/SVN/… ?

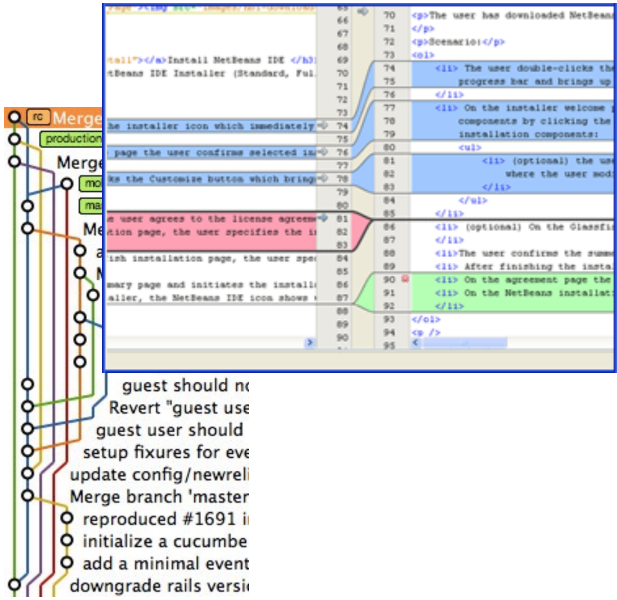
Analogous to management of source code before source code version control!

Many issues with directly using GitHub etc..

- Cannot handle large datasets or large # of versions (VLDB 2015)
- Datasets have regular repeating structure
- Querying and retrieval functionality is primitive

Focus of this work

Temporal databases only support a linear chain of versions



```
tall"></a>Install NetBeans IDE </h3
tBeans IDE Installer (Standard, Ful
he installer icon which immediately
 page the user confirms selected in
ks the Customize button which bring
e user agrees to the license agreem
tion page, the user specifies the i
ish installation page, the user spec
mary page and initiates the install
aller, the NetBeans IDE icon shows
```

```
<p>The user has downloaded NetBeans
</p>
<p>Scenario:</p>
<ol>
    <li> The user double-clicks the
        progress bar and brings up
    </li>
    <li> On the installer welcome
        components by clicking the
        installation components:
        <ul>
            <li> (optional) the us
                where the user mod
            </li>
        </ul>
    </li>
    <li> (optional) On the Glassfi
    </li>
    <li>The user confirms the summ
    <li> After finishing the insta
    <li> On the agreement page the
    <li> On the NetBeans installat
    </li>
</ol>
<p />
```

guest should nc
Revert "guest use
guest user should
setup fixures for eve
update config/newreli
Merge branch 'master
reproduced #1691 i
initialize a cucumbe
add a minimal event
downgrade rails versi

NEED A RICH LANGUAGE FOR QUERYING AND RETRIEVAL

Querying in traditional VCS largely revolves around single version and metadata retrieval

No way to specify queries like:

- identify all versions derived from version *A* that satisfy property *P*
- identify all predecessor versions of version *A* that differ from it by a large number of records
- rank a set of versions according to a scoring function
- find the version where the result of an aggregate query is above a threshold
- find parent records of all records in version *A* that satisfy certain property

GOALS

To fully realize the DataHub vision, need a language that can:

- support all existing VCS API
- allow working with both versions and data seamlessly
- navigate the ad-hoc derivation graph of versions
- allow declarative querying of the data to the extent possible

Why a new language?

- Temporal query languages (e.g., TQuel) only work with a linear history of versions
- SQL is ill-suited to traversing a graph structure, and has a cumbersome aggregate syntax
- Several languages for workflow systems, but often quite specific to the platform

HELLO VQUEL

```
retrieve “Hello World”
```

Generalization of Quel – a tuple calculus-based language developed for INGRES

Chosen primarily because of cleaner syntax

VQuel combines:

- full-fledged relational features and powerful aggregate constructs from Quel
- syntactic features from GEM, SQL, and path-based query languages
- iterator-based access to both versions and data items

NOTATION & DATA MODEL

“version”: immutable and consists of one or more datasets (files, relations) that are semantically grouped together

New versions created through the application of transformation programs or updates to one or more existing versions.

Version-level provenance is captured in the “version graph”

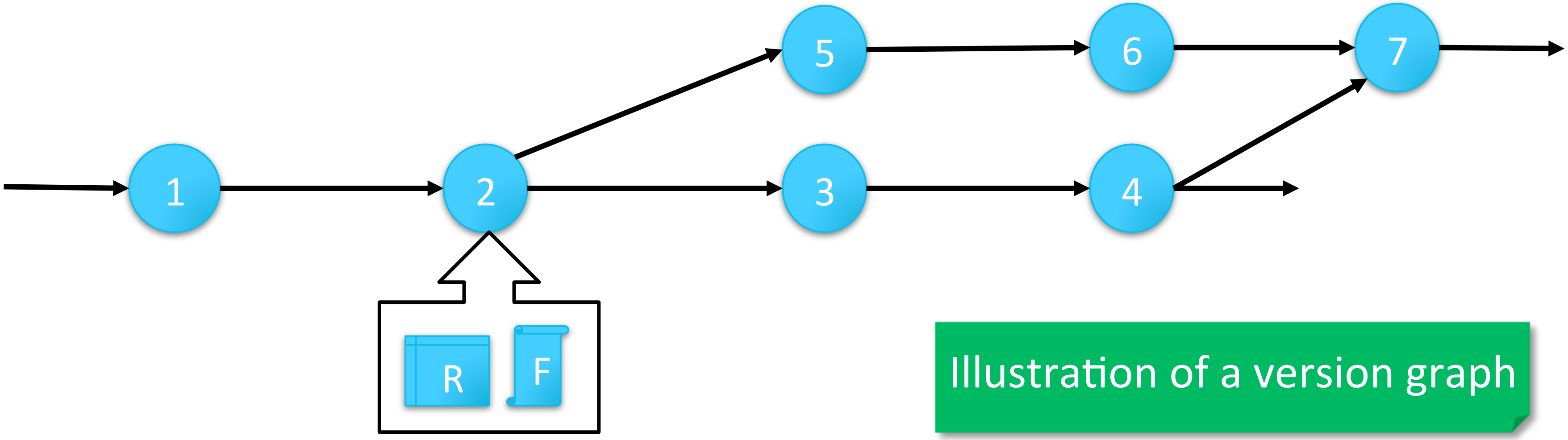
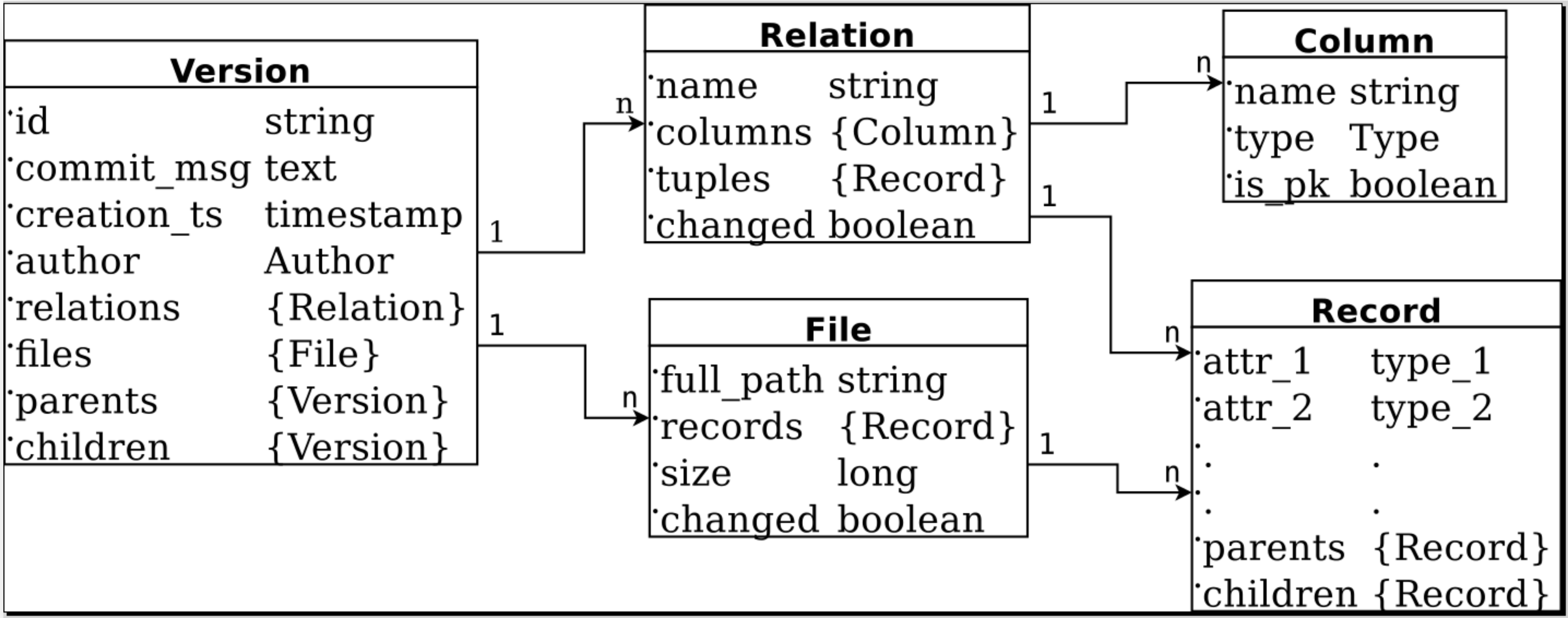


Illustration of a version graph

NOTATION & DATA MODEL

Queries written against a Conceptual Hierarchical Data Model



Version	
id	string
commit_msg	text
creation_ts	timestamp
author	Author
relations	{Relation}
files	{File}
parents	{Version}
children	{Version}

Relation	
name	string
columns	{Column}
tuples	{Record}
changed	boolean

File	
full_path	string
records	{Record}
size	long
changed	boolean

Column	
name	string
type	Type
is_pk	boolean

Record	
attr_1	type_1
attr_2	type_2
.	.
.	.
parents	{Record}
children	{Record}

ITERATORS AND PREDICATES

Example 1: What commits did Alice make after January 01, 2015?

V is an iterator over all the Versions

```
range of V is Version
retrieve V.all
where V.author.name = "Alice" and
      V.creation_ts >= "01/01/2015"
```

Predicates are used to restrict the results returned

NESTED ITERATION

Example 2: Show the history of the tuple with employee id “e01” from Employee relation.

R is an iterator over relations in a Version

E is an iterator over tuples in a Relation

```
range of V is Version
   range of R is V.Relations
      range of E is R.Tuples
         retrieve E.all, V.commit_id, V.creation_ts
         where E.employee_id = “e01” and
               R.name = “Employee”
sort by V.creation_ts
```

AGGREGATES

Example 3: Among a group of versions, find the version containing most tuples that satisfy a predicate. For instance, which version contains the most number of employees above age 50?

Aggregates can be used in both retrieve and where clauses

```
range of V is Version
   range of E is V.Relations(name = "Employee").Tuples
   retrieve into T (V.id as id,
                  count(E.id where E.age > 50) as c)
      retrieve T.id
      where T.c = max(T.c)
```

Evaluated once, used as a constant thereafter

Restricts the tuples being considered in the counting

“retrieve into” implicitly defines an iterator

VERSION GRAPH TRAVERSAL

Example 4: Find all versions within 2 commits of “v01” which have less than 100 employees.

```
range of V is Version(id = "v01")
range of N is V.N(2)
range of E is N.Relations(name = "Employee").Tuples

retrieve N.all
where count(E) < 100
```

N() returns the neighbors of a version in the version graph

AND MORE…

See paper for:

- Additional constructs for aggregates
- Partitioned aggregates – GROUP BY clause
- Joins across versions
- Additional constructs to traverse the version graph
- Querying fine grained provenance

THE ROAD AHEAD

Extensions

- Include user defined functions – e.g., custom “diff” functions for two versions
- Additional graph traversal operators

Engagement with users to refine the constructs

Implementation Challenges

Data is stored in a compressed fashion, to exploit overlaps between versions ➡ Need new query execution and optimization strategies

Version graph can become very large in a “dynamic update” environment ➡ Need scalable methods to handle the version graph

MORE ABOUT DATAHUB…

- Principles of Dataset Versioning: Exploring the Recreation/Storage Tradeoff.
  Souvik Bhattacherjee, Amit Chavan, Silu Huang, Amol Deshpande, and Aditya Parameswaran.
  41st International Conference on Very Large Data Bases (VLDB), 2015.

- Collaborative Data Analytics with Datahub (Demo).
  Anant Bhardwaj, Amol Deshpande, Aaron Elmore, David Karger, Sam Madden, Aditya Parameswaran, Harihar Subramanyam, Eugene Wu, and Rebecca Zhang.
  41st International Conference on Very Large Data Bases (VLDB), 2015.

- DataHub: Collaborative Data Science & Dataset Version Management at Scale.
  Anant Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J. Elmore, Samuel Madden, Aditya Parameswaran.
  Conference on Innovative Database Research (CIDR), 2015.

THANK YOU