Bolt-on Versioning for Relational Databases

Motivation:

Collaborative data science is ubiquitous

- Many users, many versions of same dataset at various stages of analyses
- Status quo:
  - Stored in a file system
  - Relationships between versions unknown
- **Can we build a versioned data store?**
  - Support efficient access, retrieval, and modification of versions

Motivation: Starting Points

- Git/SVN is inefficient and unsuitable
  - Ordered semantics
  - No data manipulation API
  - No efficient multi-versioning queries
  - Poor support for massive files

- Relational databases are great!
  - Pros: efficient, scalable
  - However, no support for versions

Our approach

- Leverage existing DBMS to support branched versioning
- PostgreSQL + Versioning commands

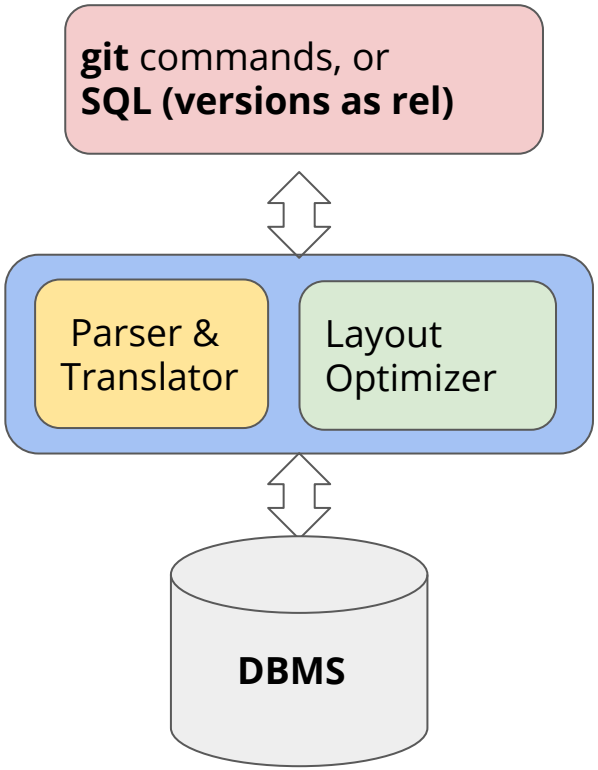
Get all the benefits of relational databases for free!

Framework

User Interface Layer

“Versioning” Layer
(translation/bookkeeping)

Unmodified Postgres Backend
(not aware of versions)



OrpheusDB

User Interface Layer: What is currently supported?

- Typical Workflow 1:
  - Checkout a specific version (or multiple versions -- as a merge) as a relation
  - Perform updates via SQL commands
  - Commit back as a new version
- Typical Workflow 2:
  - Checkout a specific version (or versions) as a relation
  - Export relation as a csv
  - Perform updates in favorite programming language
  - Import back as a database relation
  - Commit back as a new version

Edits happen by checking out versions: you don’t need to clone the entire repository (not desirable for large dataset collections)

User Interface Layer: What is currently supported?

Git commands: clone, commit, merge, diff

Export-based commands: dump, load, ls (list contents of a version)

SQL-based commands: SQL on a checked out version, or SQL across one or more versions directly (mix + match with versioning)

Underlying Storage Model: A High Level View

- Data Table + Index Table

rid	badgeID	age	gender	salary
r1	**0001**	**25**	**F**	**6500**
r2	0002	30	F	7500
r3	0003	28	M	7000
r4	0004	40	M	9000
r5	0005	35	F	6500
r6	**0001**	**25**	**F**	**7500**
r7	0006	32	M	7000

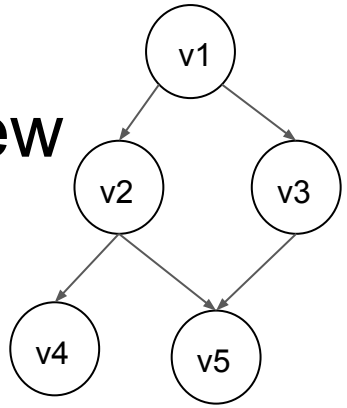
Data Table

vid	rlist
v1	{r1,r2,r3}
v2	{r1,r2,r3,r4}
v3	{r3,r5}
v4	{r2,r3,r4,r6,r7}
v5	{r1,r2,r4,r5}

Index Table

Underlying Storage Model: A High Level View

- Track versioning information in metadata table



Version Graph

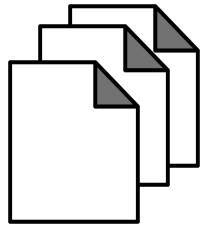
vid	num_of_ records	parent	children	create_time	commit_time	commit_msg
v1	3	{}	{r1,r2,r3}	...	...	...
v2	4	{v1}	{r1,r2,r3,r4}	...	...	...
v3	2	{v1}	{r3,r5}	...	...	...
v4	5	{v2}	{r2,r3,r4,r6, r7}	...	...	...
v5	4	{v2,v3}	{r1,r2,r4,r5}	...	...	...

Version Metadata Table

OrpheusDB

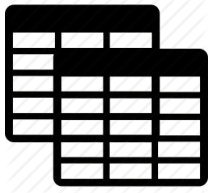
Supported Commands:

User Workspace



dump

load



clone

commit

Backend Storage Layout

rid	badgeID	age	gender	salary
r1	**0001**	**25**	F	**6500**
r2	0002	30	F	7500
r3	0003	28	M	7000
r4	0004	40	M	9000
r5	0005	35	F	6500
r6	**0001**	**25**	F	**7500**
r7	0006	32	M	7000

Data Table

vid	rlist
v1	{r1,r2,r3}
v2	{r1,r2,r3,r4}
v3	{r3,r5}
v4	{r2,r3,r4,r6, r7}
v5	{r1,r2,r4,r5}

Index Table

merge
diff
ls
sql

OrpheusDB

Takeaways

- Current prototype can support general version control of structured datasets
- Lightweight and convenient
  - Can operate on any existing DBMS
- Easy to use
  - Similar semantics as Git
  - Plus easy access to SQL

Future Plans

- Open-source release in a month
- Improvements
  - Partitioning for efficiency improvements
  - Complex SQL support for cross-version operations

OrpheusDB

Commands: Usage

- **clone -v [version id] -t [table name] -f [file name] -ignore**
  - Clone version(s) into a table or a file
- **commit -t [table name] -m [message]**
  - Commit a modified table with commit message
- **load -f [file path] -t [table name] -n [new table schema] -ignore**
  - Load records from file to database a (new or existing) table
- **dump -t [table name] -f [file path]**
  - Export a table into a file

More Commands

- **merge**
  - Merge multiple tables/versions into a new table/version
- **diff**
  - Show changes between tables/versions
- **ls**
  - Show tuples of a particular table / version
  - Show meta information of a particular version
- **sql**
  - Standard SQL commands (SELECT, UPDATE, INSERT …)