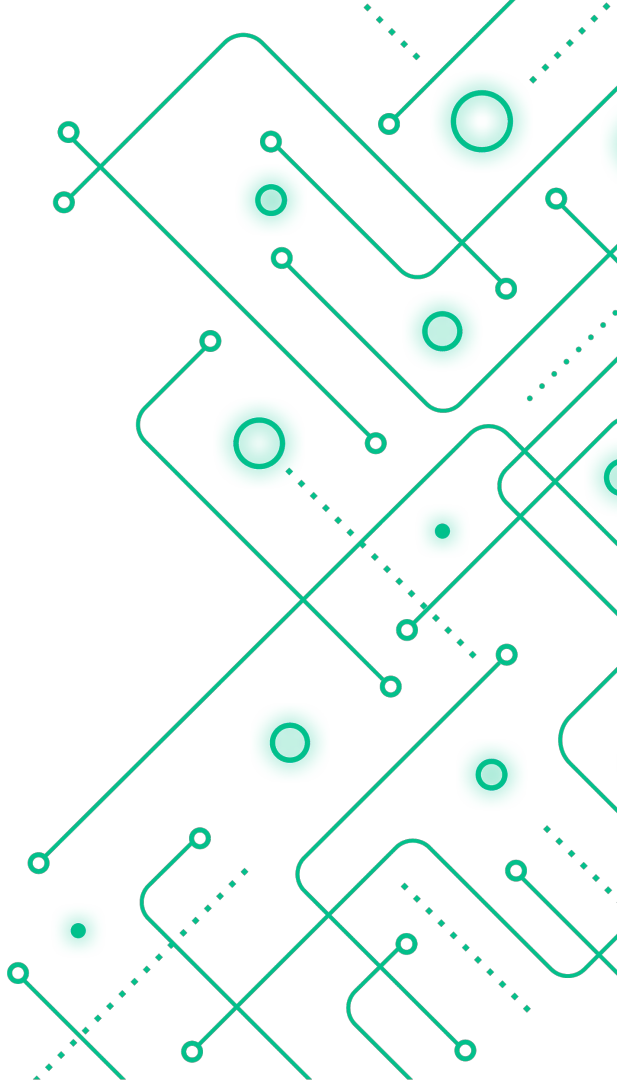

TerminusDB
Building a Native Revision Control *Graph* DB from Scratch

Outline

- Motivation
  - Graph
  - Distributed Revision Control
- Architecture
  - Bottom-up architecture
  - Top-down view



TerminusDB

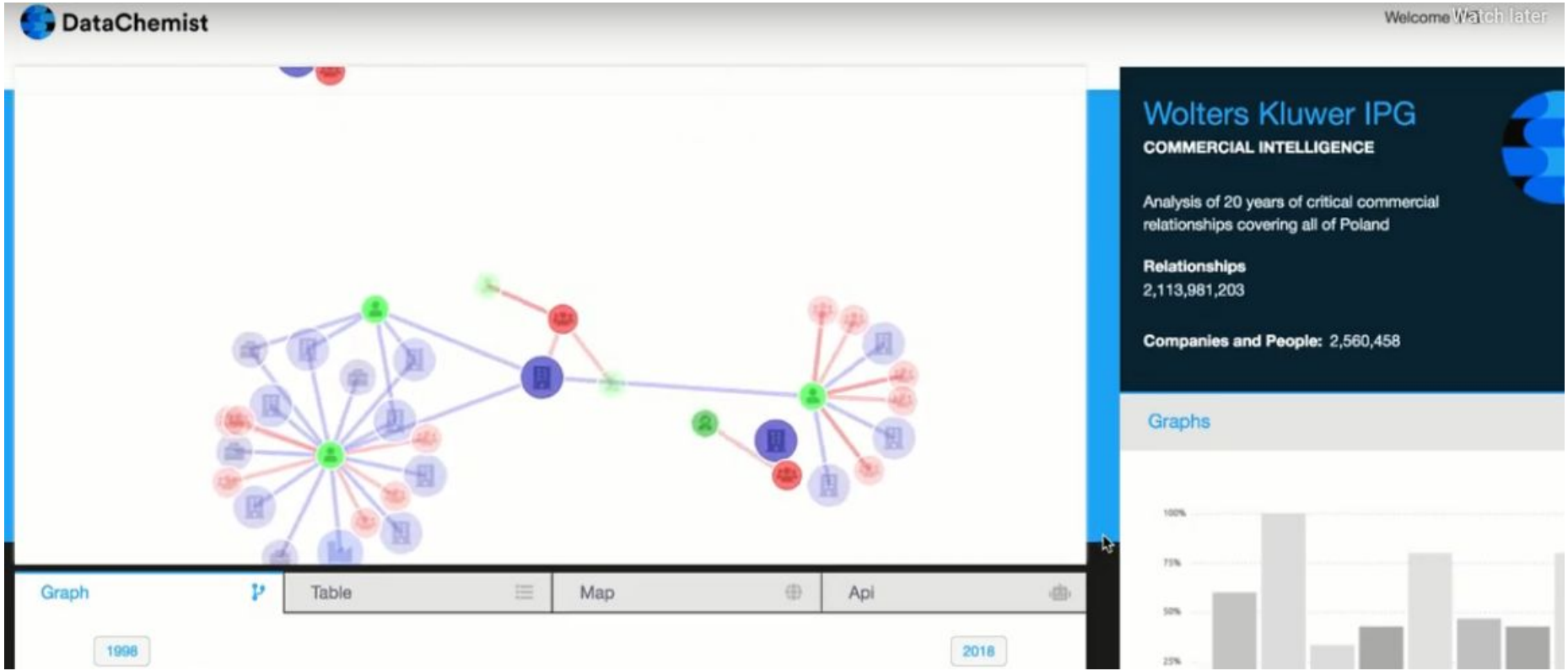
Why Graph?

Seshat: Global History Databank

Academic collaboration seeking to create datasets describing every human society since 10,000BCE

Major publications in 2018-19 in Nature and PNAS on evolution of social complexity and religion based on a high quality RDF DB.

Sprawling collection of datasets covering 1000s of variables, mostly human entered and error prone, with wide variety of formats and schemata

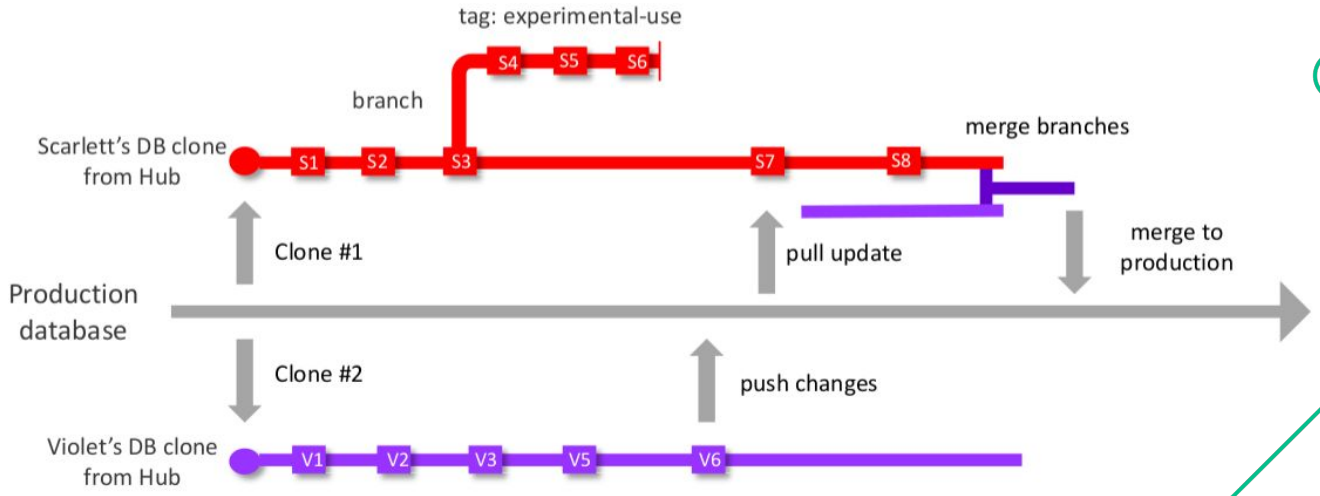

DataChemist
Welcome
Watch later
Wolters Kluwer IPG
COMMERCIAL INTELLIGENCE
Analysis of 20 years of critical commercial relationships covering all of Poland
Relationships
2,113,981,203
Companies and People: 2,560,458
Graphs
Graph
Table
Map
Api
1998
2018

```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

Why Distributed Revision Control?

TerminusDB

Distributed Revision Control

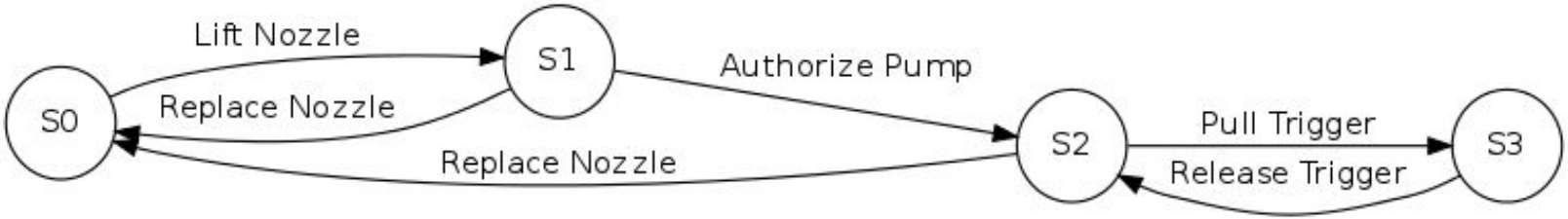


TerminusDB

Recap of store

TerminusDB-store is our Rust library which implements our delta-encoded append-only succinct datastructures.

We represent data as graphs (triples)



S0 — Lift Nozzle → S1
S1 — Replace Nozzle → S0
S1 — Authorize Pump → S2
S2 — Replace Nozzle → S0
S2 — Pull Trigger → S3
S3 — Release Trigger → S2

We represent graphs with succinct data-structures

More details in: "Succinct Data Structures and Delta Encoding for Modern Databases"

String	Offset	Remainder
Pearl Jam	0	Pearl Jam
Pink Floyd	1	ink Floyd
Pixies	2	xies
The Beatles	0	The Beatles
The Who	4	Who

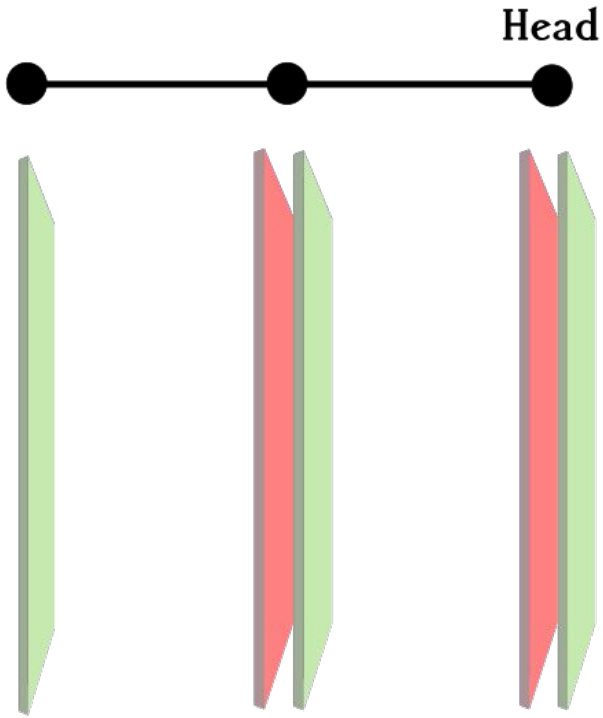
Table 1: *Plain Front Coding Dictionary*

We represent graphs with succinct data-structures

More details in: "Succinct Data Structures and Delta Encoding for Modern Databases"

Triples	Encoding	Description
(1, 2, 3)	1 2 3	Subject Ids
(1, 2, 4)	1 1 0 1	Encoded Subject Bit Sequence
(2, 3, 5)	2 3 4 5	Predicate Vector
(2, 4, 6)	1 0 1 1 1	Encoded Predicate Bit Sequence
(3, 5, 7)	3 4 5 6 7	Object Vector

Table 2: *Succinct Graph Representation*



+(x,y,z)
+(a,b,c)
+(1,2,3)

+(e,f,g)
-(x,y,z)
-(1,2,3)

+(9,8,7)
-(a,b,c)

Labels

Labels are a file which points at a layer, giving us named access to a head. They can be updated in a safe manner - file locking and update to an already finalised layer.

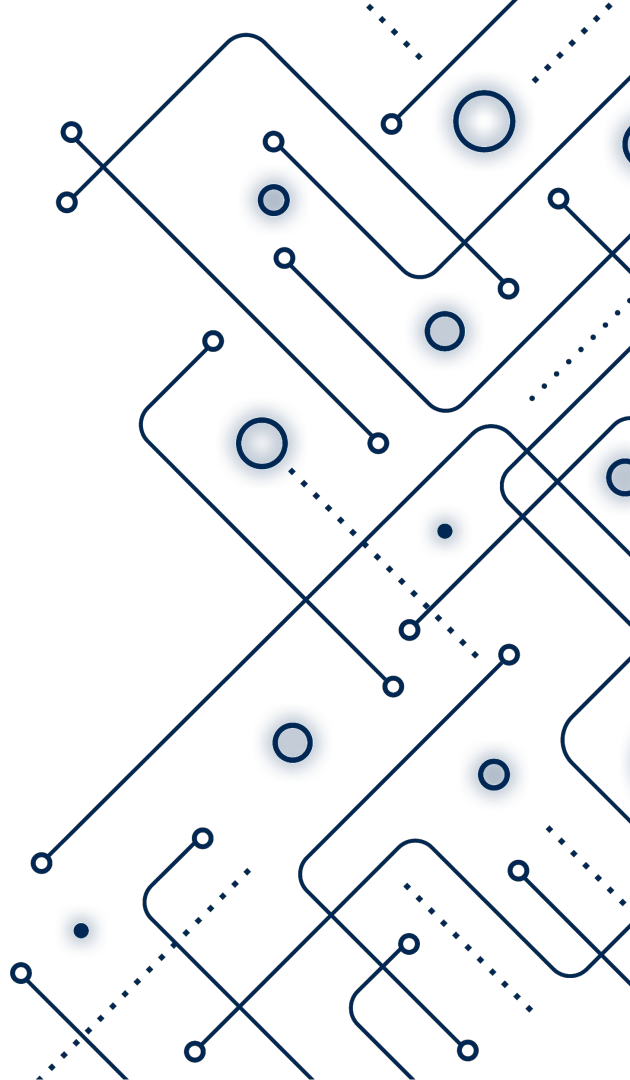
TerminusDB

Why is this not enough?

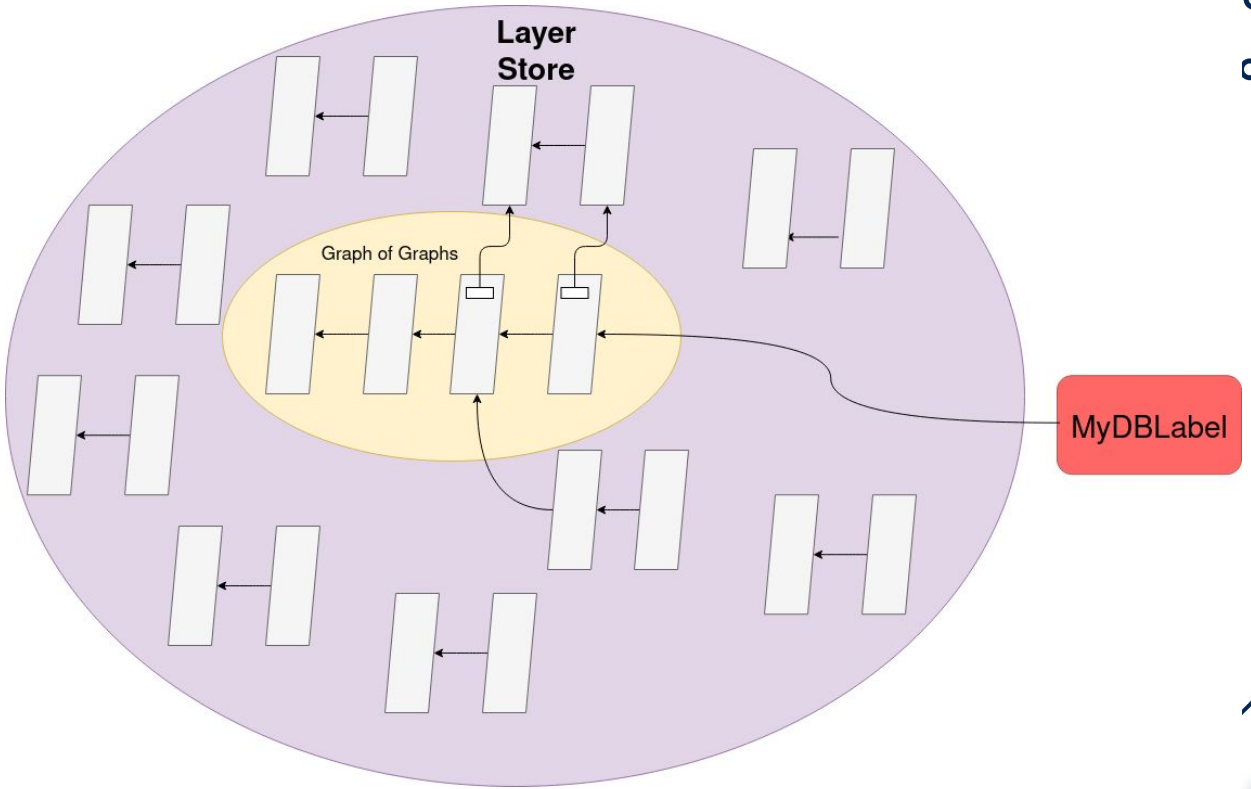
- We need to orchestrate changes to multiple graphs! How do we move them all in concert? (for instance a schema and an instance)
- Commit history - time stamps, authors, messages - how do we keep track of all these things?
- Branches (tags, versions) - all need to point at collections of graphs.

Generic solution:

- Point at a layer from another layer.
- Build up a hierarchy of layers
- Move the label of the top layer when everything is done



TerminusDB



TerminusDB

Welcome to the Commit Graph!

The first level of the hierarchy in our plan of graph domination

TerminusDB

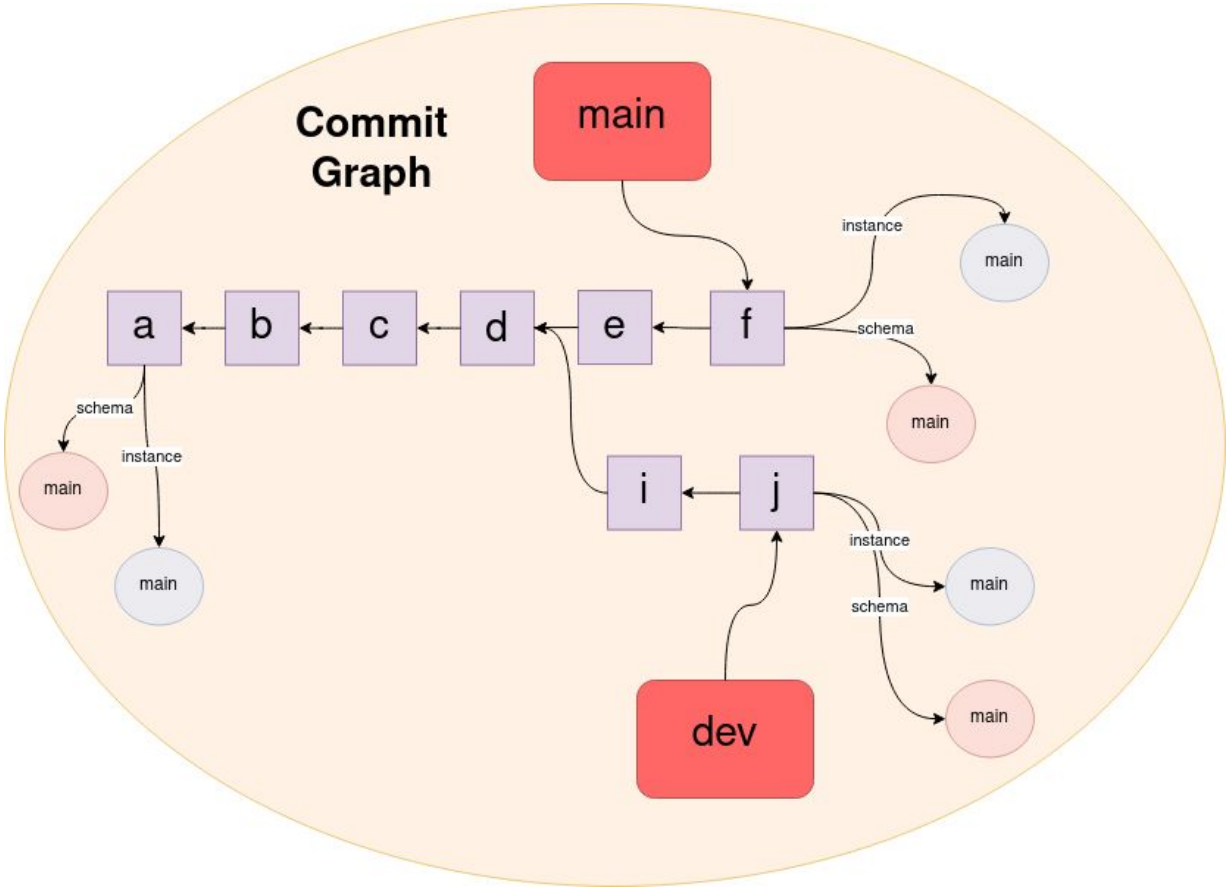
The Commit Graph

- Why a commit graph?
- We want to have commits - collections of graphs with a known state satisfying some nice properties
- We want the history of these graph collections
- We want some meta-data

TerminusDB

What's in a commit graph

- Commit objects, with their author, message and time-stamp
- Named graph objects which are pointed to by commit objects {instance,schema,inference}
- Layer references, which are pointed to by named graph objects
- Branches, which refer to a commit object
  - Eventually tags can go here too
- Prefix definitions which help us name objects in our graph


Commit Graph
main
a
b
c
d
e
f
i
j
dev
instance
schema
main
schema
instance
main
main
instance
schema
main
main
main

TerminusDB

Commit graph is a validated graph

- It has a schema (two actually) of its own
- It checks the instance data on commit
- We could relax this later but it's nice for development

The Commit Graph

- This is enough to do: branch, merge, rebase, tagging, reset, revert
- We do these operations with WOQL queries on the commit graph!
- Immense levels of meta-circularity!
  Level 32 lambda-cleric

TerminusDB

Meta-data Graph

- Why? Collaboration, that's why!
- Very simple contains repository objects
  - Remote objects
    - Has a URL
  - And local object(s?)
- Both types point to a commit graph

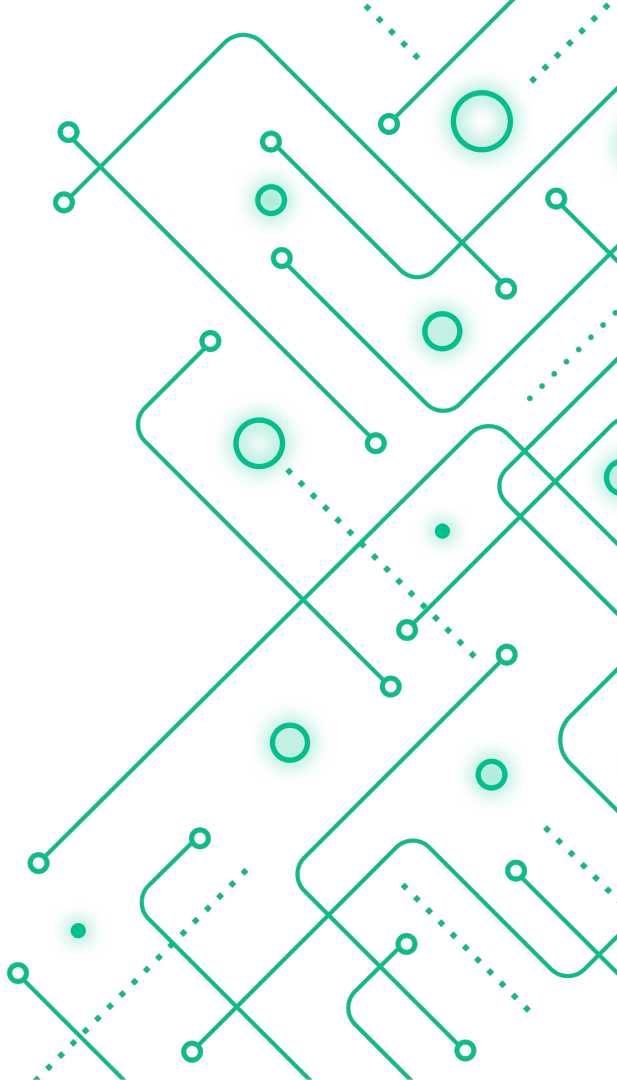
Meta-data Graph

- Gives us the operations between TerminusDB installations:
  - Clone
  - Push
  - Pull
- The operations are implemented by transmitting an entire commit graph and all associated layers.
- **This is a top-level graph with a label**

TerminusDB

System Graph

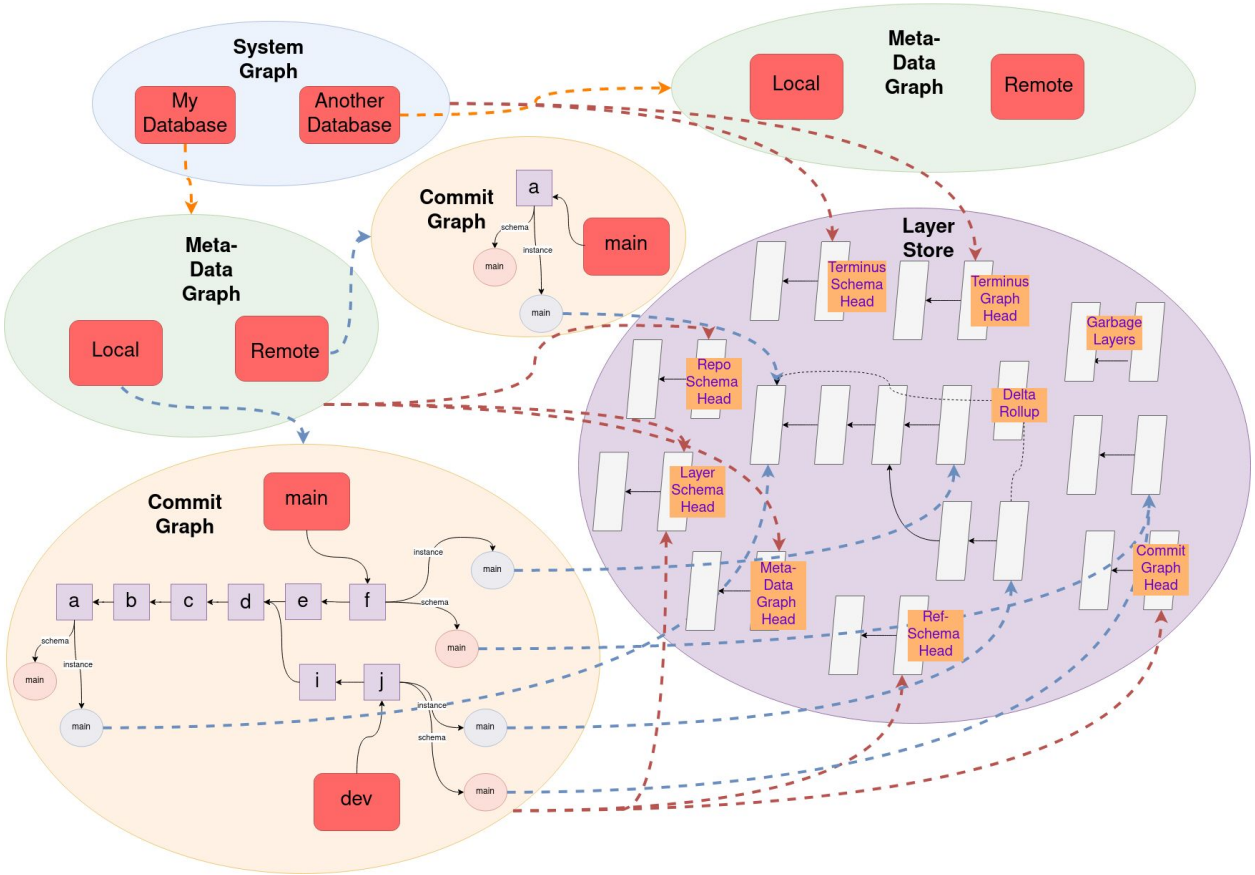
- Why? Lots of server system stuff!
- Authentication
- Authorization
- Capabilities
- Database names and who controls them!



TerminusDB

Why is the System Graph not the top level?

- Storing layer references in a graph and advancing a single label ensures transactionality
- We could have made it so that the system graph was the top level, this would allow multi-database transactions
- BUT! It also means contention between every database over advancing the system graph head.


System Graph
My Database
Another Database
Meta-Data Graph
Local
Remote
Commit Graph
a
schema
instance
main
main
main
Meta-Data Graph
Local
Remote
Layer Store
Terminus Schema Head
Terminus Graph Head
Garbage Layers
Repo Schema Head
Delta Rollup
Layer Schema Head
Meta-Data Graph Head
Ref-Schema Head
Commit Graph Head
Commit Graph
main
a
b
c
d
e
f
i
j
instance
schema
main
dev

TerminusDB

Give TerminusDB a try!

Terminusdb.com

https://github.com/terminusdb/terminusdb-quickstart