

Accidental Complexity

Simplifying CRDTs for Local-First Software

Matt Wonlaw, Feb 2023

Self Intro

who is this person?

- Passionate about removing unneeded complexity
- Prior Work @ Meta
 - Anti Abuse (simplify integration with abuse protection systems)
 - Privacy (private by default)
 - Security (secure by default)
- Leaving Work
 - Resurrect a hobby project? 🤔

Resurrecting strut.io

(this site)

- Began as: hobby project of a poor recent grad
- My thinking then:
 - User devices have compute and storage.
 - Why am I paying for server!? 💰
- Developed as a static & local web app in ~2012
- Early 2000s local vs 2010s+ local vs 2022 local

Modern Local First 🐰 🍻










All the complexity

- Many platforms (Android, iOS, Desktop, Web, Server)
- Sync state between devices
- Respond to changes immediately
- Persistence
- Permissions
- Offline Editing
- Realtime Collab

Seeking a Solution

SQLite???

-  All platforms
-  Syncing state between devices
-  Responding immediately to state changes
-  Persistence
-  Permissions
-  Offline editing
-  Realtime collaboration

Sync & Realtime Collab

- Two sides of the same problem
- Realtime collar:
 - Write and interact locally without waiting for the network
- Sync:
 - Merge offline edits between devices
- Both are a “merging of edits” problem
 - Differ in time scale

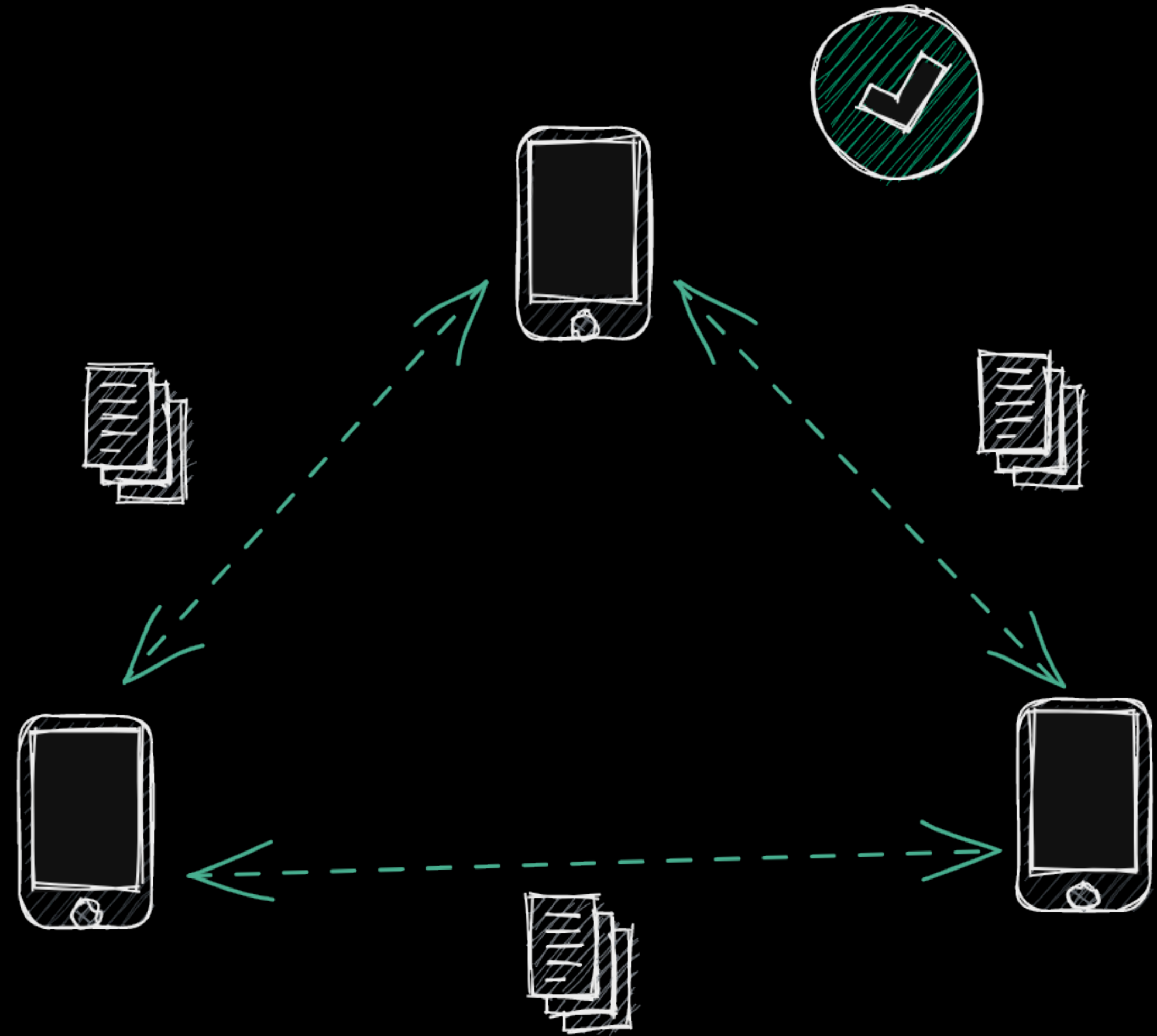
Merging Edits

Need:

- A way for each node to have its own copy of state and the ability to merge those copies deterministically.

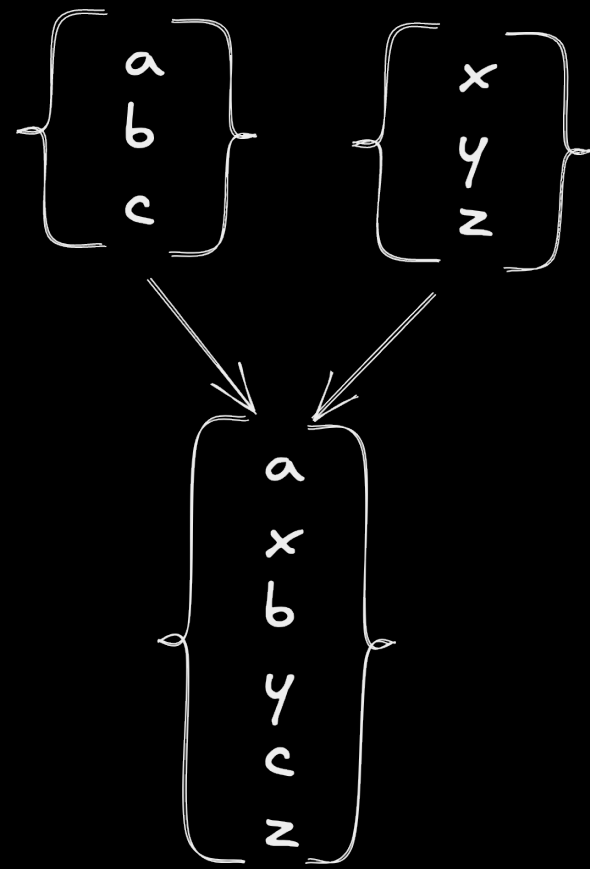
Solutions:

- Totally ordered event log
- Conflict Free Replicated Data Types (CRDTs)

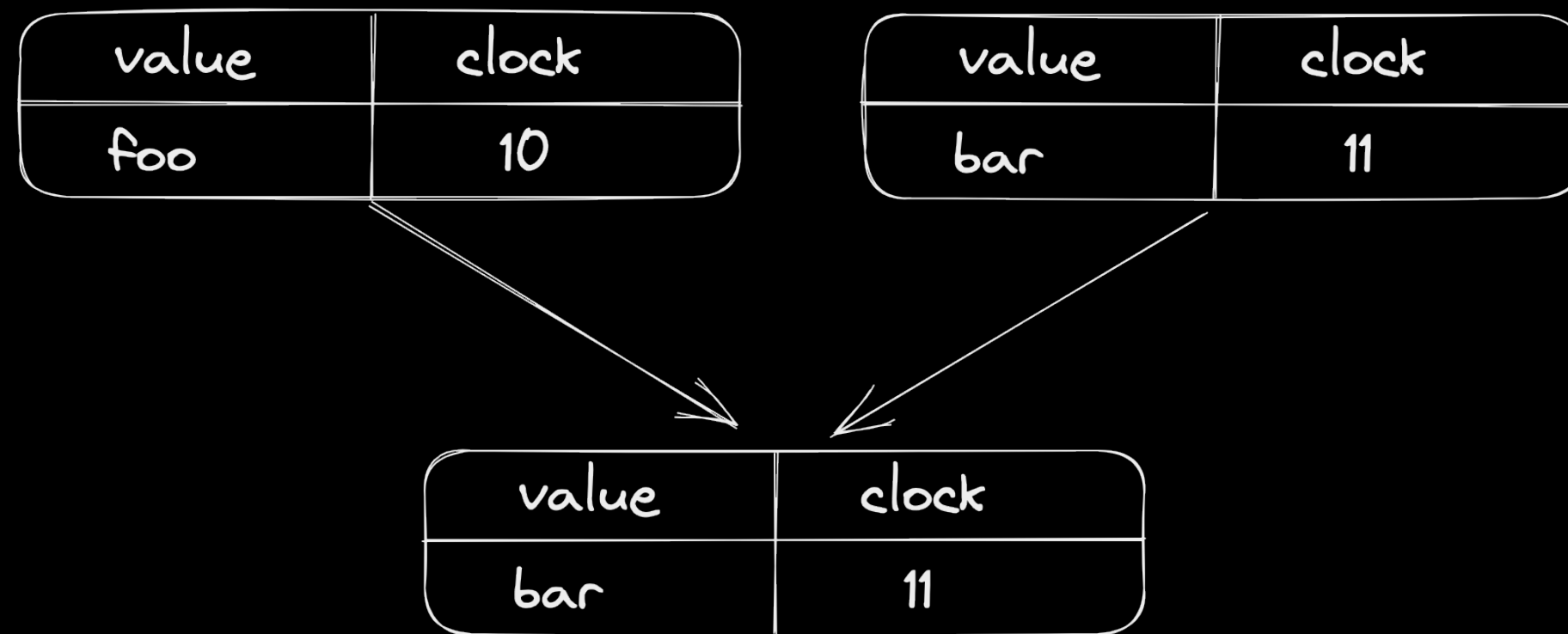


Some CRDT Types

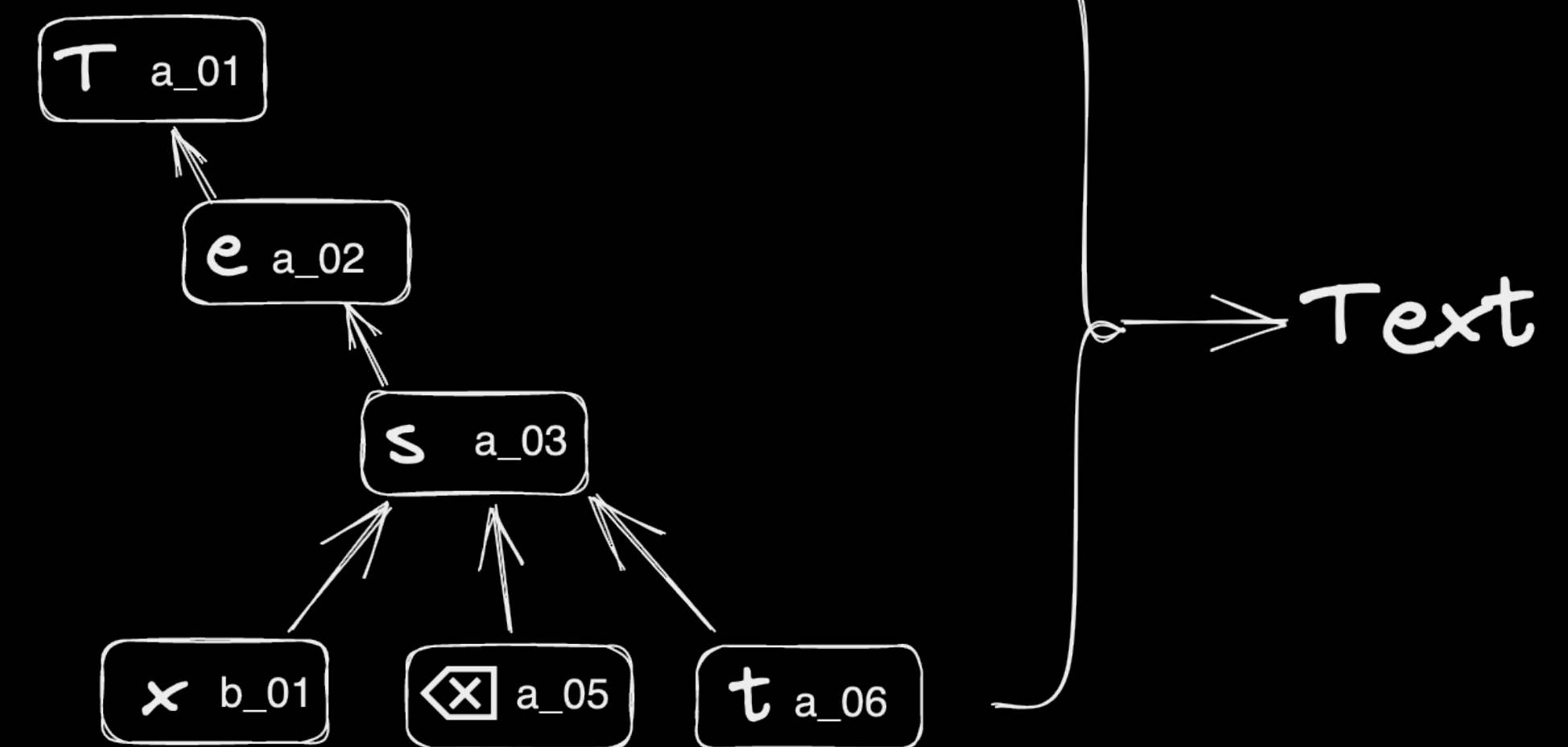
Grow Only Set



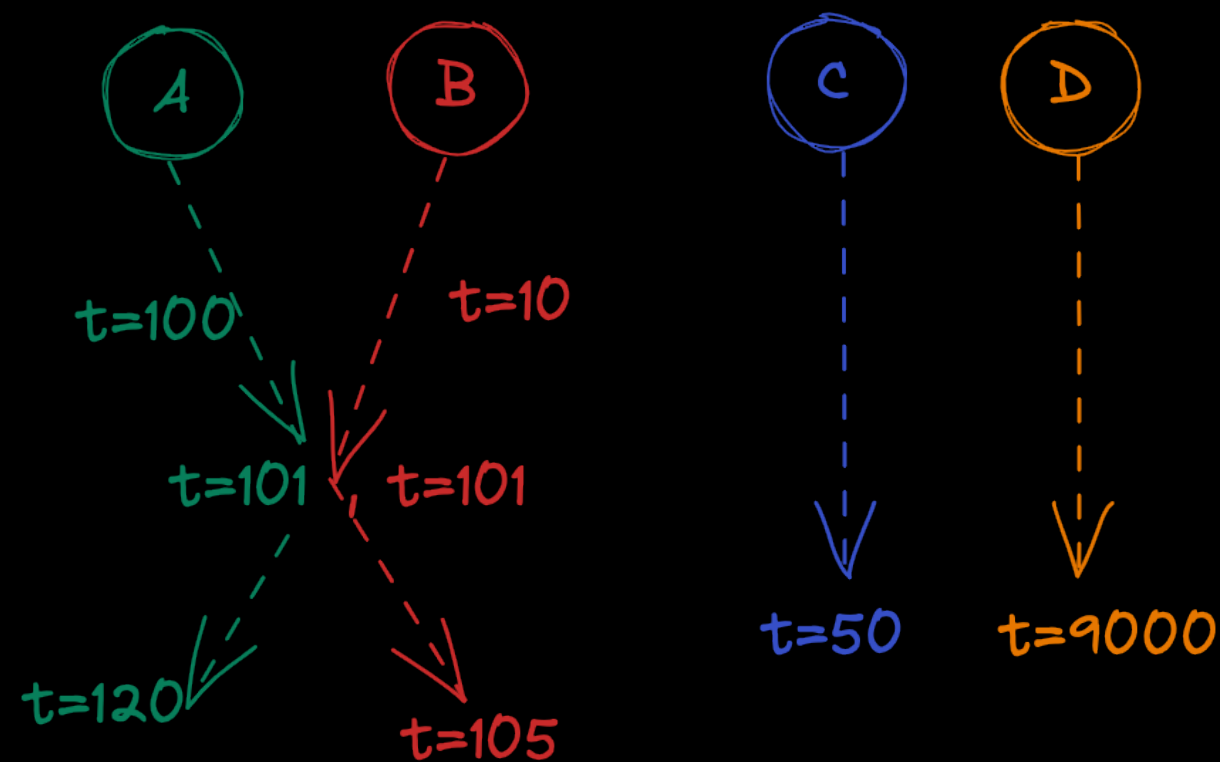
Last Write Wins (LWW) Registers



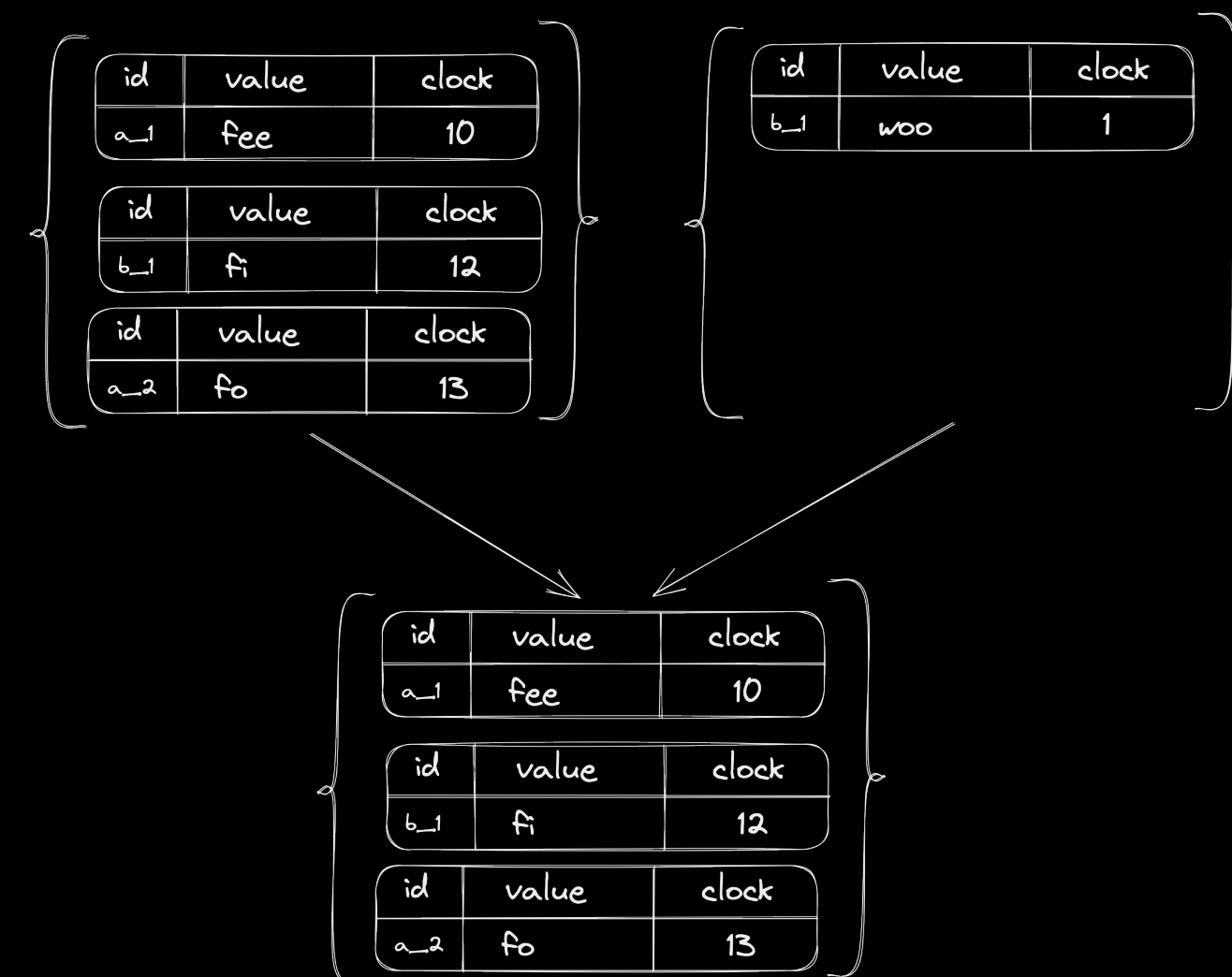
Causal Tree



Time, Causal Consistency, & Happens Before



G-Set of LWW Registers



Counter, MV Register,
OR Set
Delta State Variants,
Compose into more complex types

Removing Complexity

Taming CRDTs

- Relational DBs made transactions, relations, indexing, and constraints declarative
- Can we not extend them to do the same for CRDTs?

CRDTs in SQLite

- Conflict Free Replicated Relations^[1]
- Base Concepts:
 - Tables as G-Set or OR-Sets^[2]
 - Rows as maps of CRDTs (lww, counter, rag)^[3]
 - User defined ordering via convergent fractional indices^[4]
 - Recursive relations to model causal tress, RGA, WOOT, Logoot^[5], event logs, etc.

[1] - <https://hal.inria.fr/hal-02983557/document>

[2] - <https://vlcn.io/blog/gentle-intro-to-crdts.html#a-simple-crdt>

[3] - <https://vlcn.io/pages/lww-vs-dag/> [4] - <https://www.youtube.com/watch?v=BghFgK6VJIE>

[5] - <https://hal.inria.fr/inria-00629503/document>

Upgrade Tables to CRRs

```
-- define table
CREATE TABLE "slide" (
  "id" PRIMARY KEY,
  "deck_id",
  "order",
  "created",
  "modified"
);

-- upgrade to crr
SELECT
  crsql_as_crr('slide');
```

See strut.io schema: <https://github.com/tantaman/Strut/blob/main/app-server-shared/src/schemas/strut#L80-L111>

- Collaborative tables are upgraded to `crrs`
- Local only state (like undo stack or selection state) are not `crrs` and thus not replicated

Add a Distributed Fractional Index

```
-- add fractional index  
SELECT  
  crsql_fract_as_ordered('slide', 'order', 'deck_id');
```

UI Integration

```
const todos = useQuery<Todo>(  
  ctx,  
  `SELECT * FROM "todo" WHERE "complete" IS ?`,  
  [filter],  
);  
  
todos.map(t => <Todo todo={t} />);
```

Merge Databases

```
-- pull all changes from db a
SELECT * FROM crsql_changes
  WHERE db_version > ?
  AND site_id IS NOT ?

-- merge changes into db b
INSERT INTO crsql_changes
  VALUES (?, ?, ?, ?, ?, ?, ?)
```

CR-SQLite

[GitHub.com/vlcen-io/cr-sqlite](https://github.com/vlcen-io/cr-sqlite)
vlcen.io

- ✓ All platforms
- ✓ Syncing state between devices
- ✓ Responding immediately to state changes
- ✓ Persistence
- 🚧 Permissions
- ✓ Offline editing
- ✓ Realtime collaboration



Future



- Custom SQL syntax for CRDTs in LibSQL^[1]
- Postgres integration (for traditional client-server setups)
- Row level security & partial DB sync
- Peer to peer event sourcing
- Text CRDTs

[1] - <https://github.com/libsql/libsql>