

Transactions in shardman

Distributed transactions:

- ▶ Distributed atomicity
- ▶ Distributed isolation
- ▶ Profit! (distributed)

Transactions in shardman

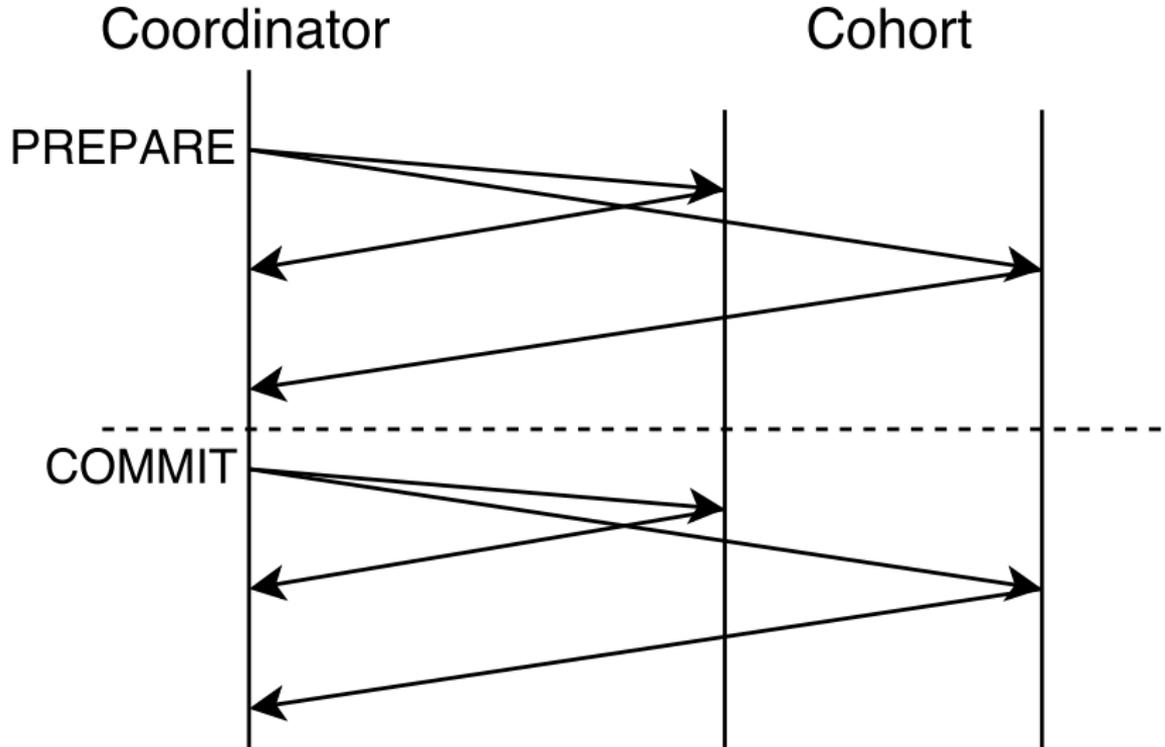
All reliable distributed systems are alike
each unreliable is unreliable in its own way.
Kyle Kingsbury and Leo Tolstoy.

Transactions in shardman

Distributed transactions:

- ▶ Atomicity: 2PC
- ▶ Isolation: Clock-SI

Transactions in shardman: 2PC

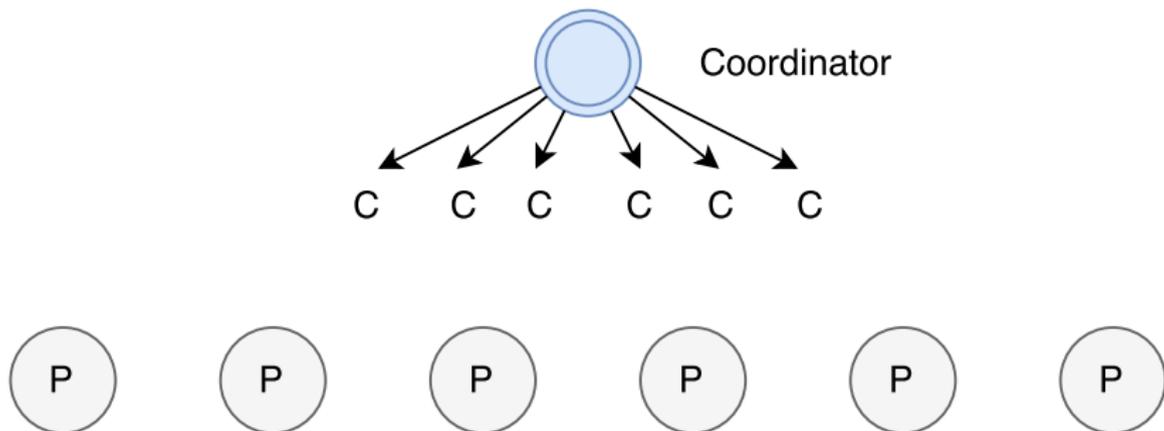


Transactions in shardman: 2PC

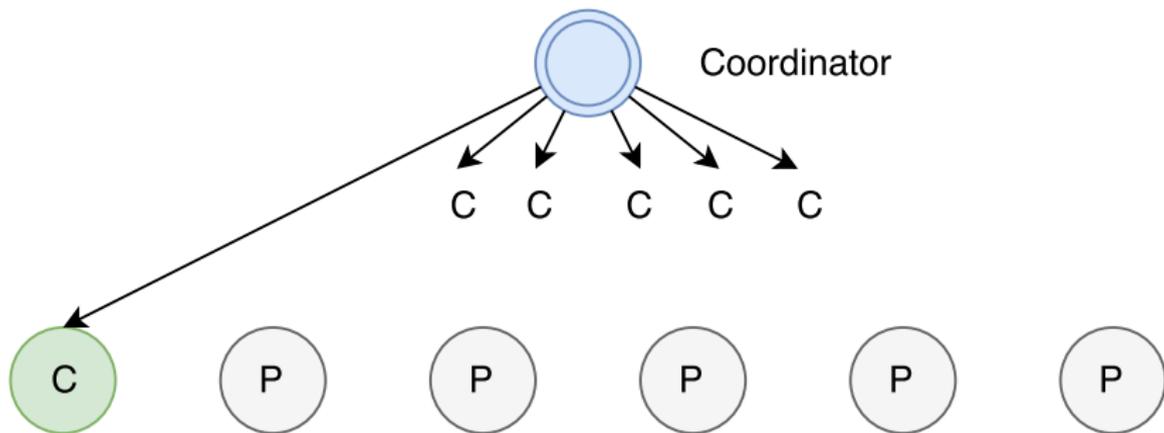
Two-phase commit is the anti-availability protocol.

P. Helland. ACM Queue, Vol. 14, Issue 2, March-April 2016.

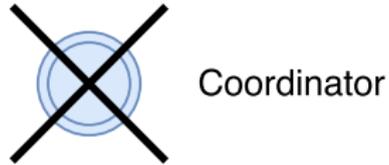
Transactions in shardman: 2PC



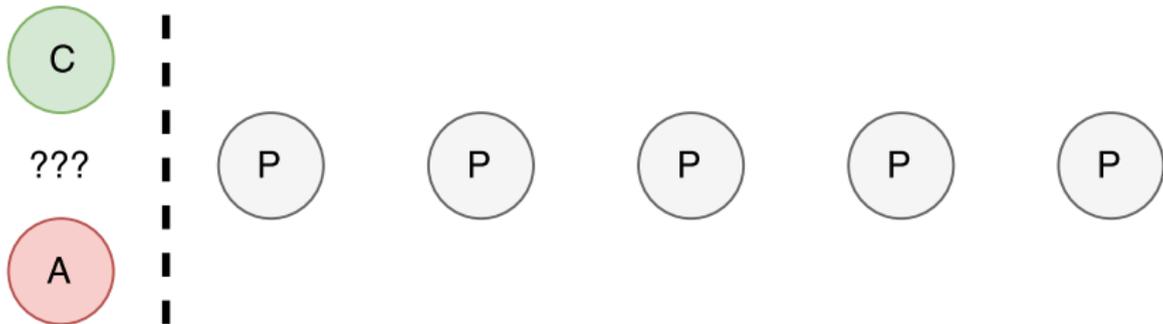
Transactions in shardman: 2PC



Transactions in shardman: 2PC



Transactions in shardman: 2PC

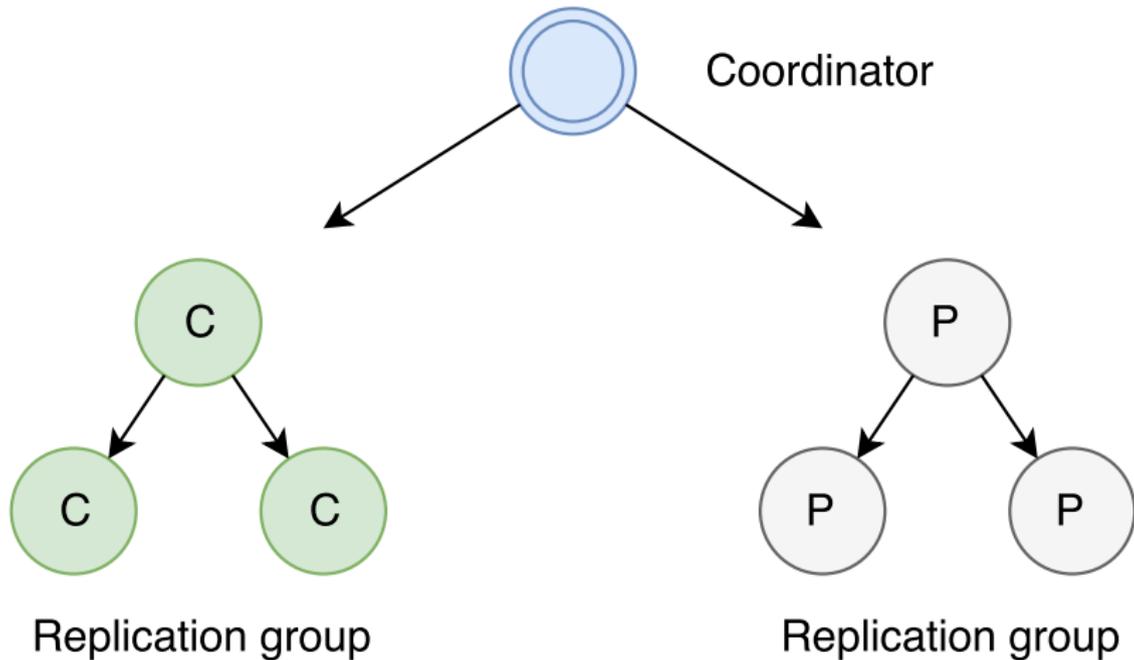


Transactions in shardman: 2PC

So what we can do about it?

- ▶ Make 2PC fail-recovery tolerant: X3PC, Paxos Commit
- ▶ Back-up partitions!

Transactions in shardman: 2PC



Transactions in shardman: 2PC

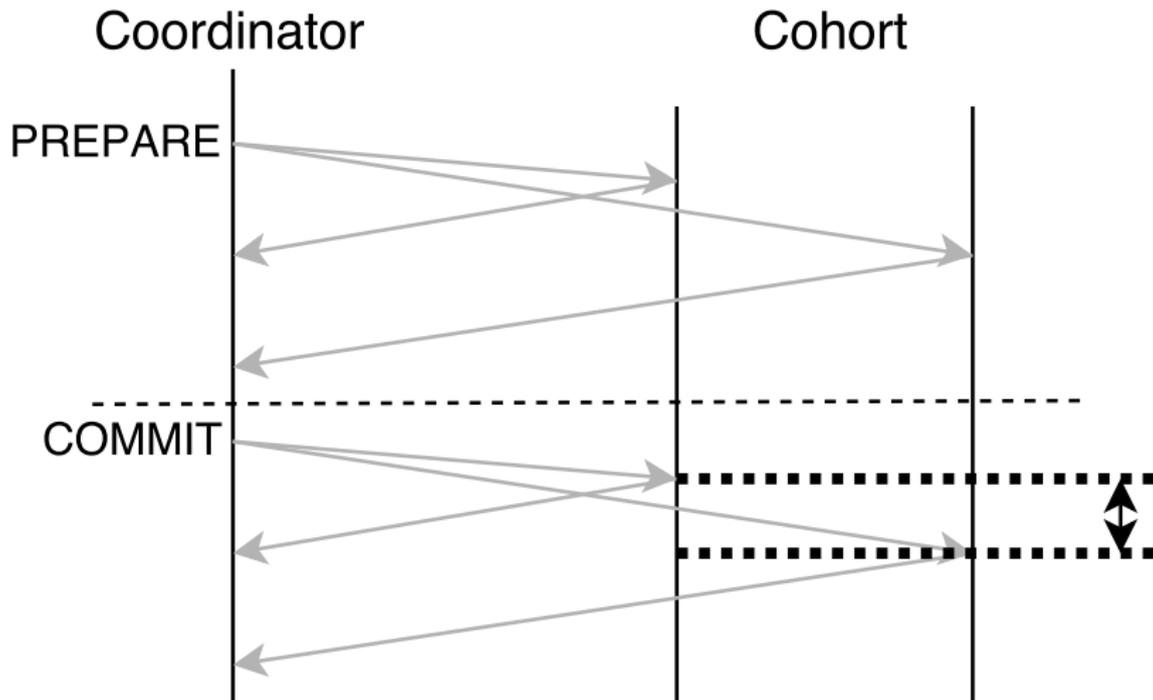
Spanner mitigates this by having each member be a Paxos group, thus ensuring each 2PC “member” is highly available even if some of its Paxos participants are down.

Eric Brewer.

Transactions in shardman: isolation

Profit? Not yet!

Transactions in shardman: isolation



```
postgres_fdw.use_twophase = on
BEGIN;
UPDATE holders SET horns -= 1 WHERE holders.id = $id1;
UPDATE holders SET horns += 1 WHERE holders.id = $id2;
COMMIT;
SELECT sum(horns_count) FROM holders;
-> 1
-> 2
-> 0
-> -2
```

MVCC in two sentences:

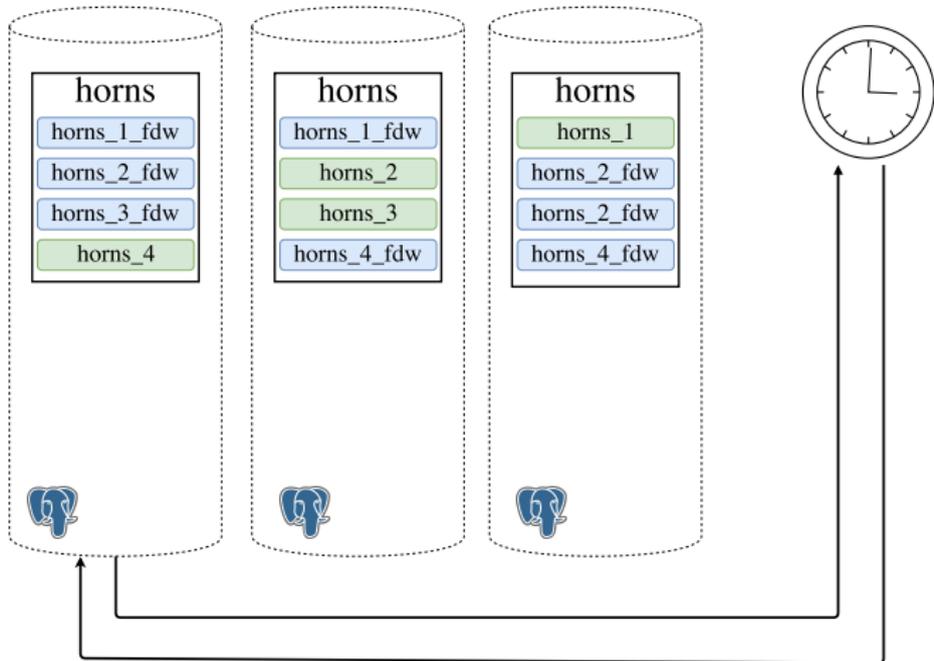
- ▶ UPDATE/DELETE create new tuple version, without in-place override
- ▶ Each tx gets current database version at start (xid, csn,timestamp) and able to see only appropriate versions.

```

                acc1
ver 10: {1, 0}
ver 20: {1, 2}
ver 30: {1, 4}
----- snapshot = 34 -----
ver 40: {1, 2}
  
```

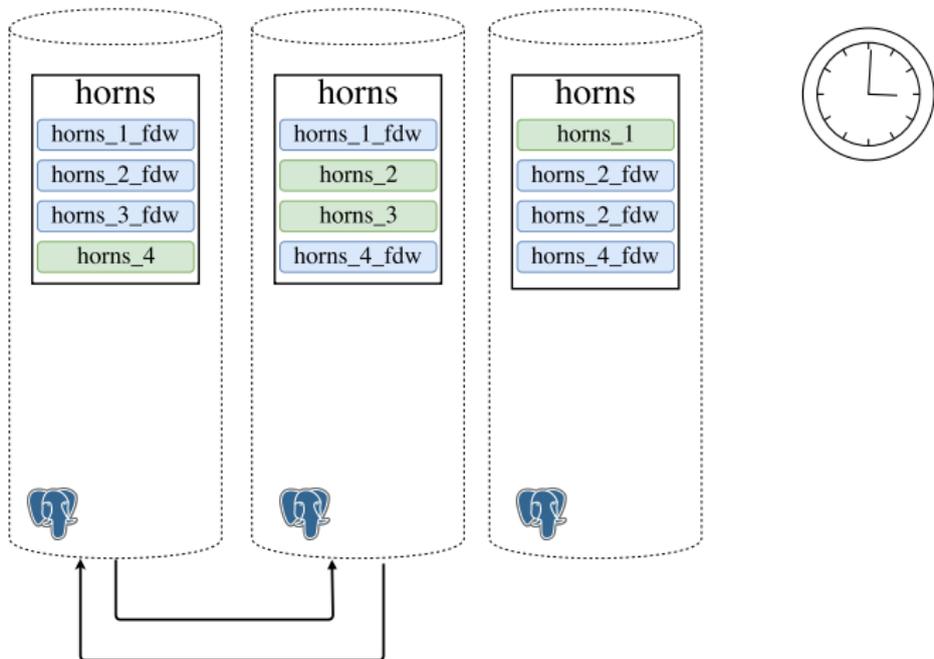
Transactions in shardman: isolation

BEGIN



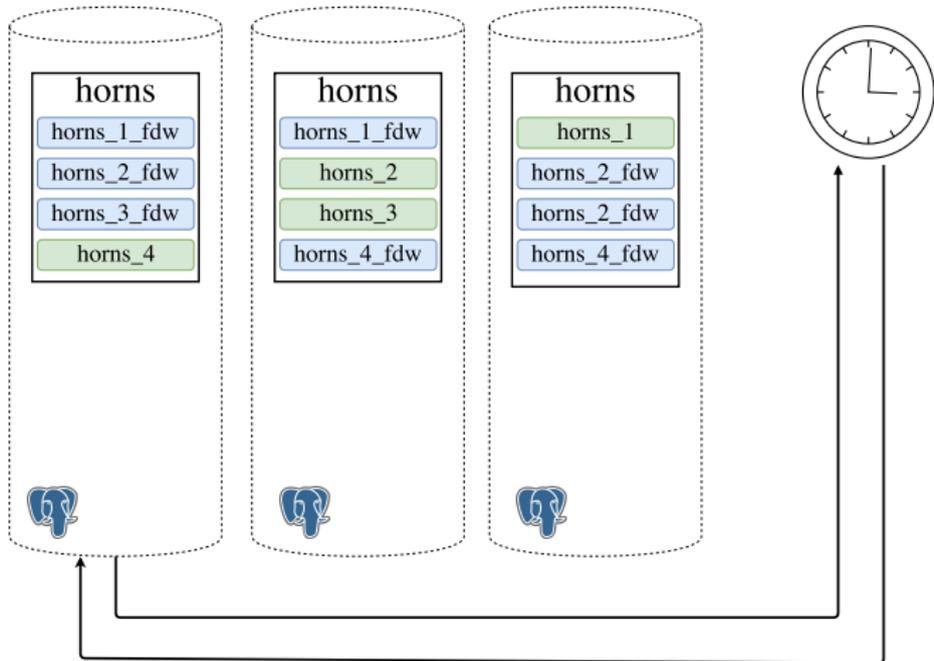
Transactions in shardman: isolation

Do some serious stuff



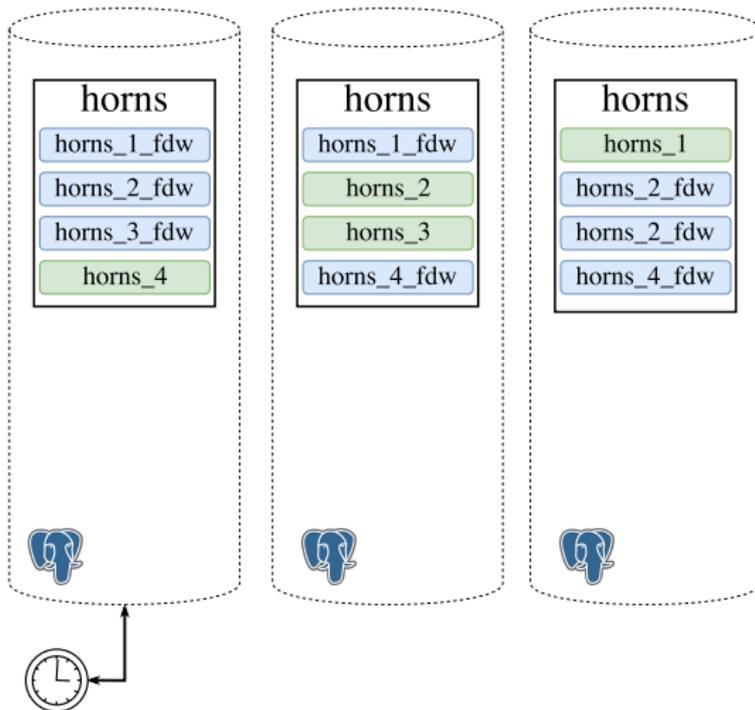
Transactions in shardman: isolation

COMMIT



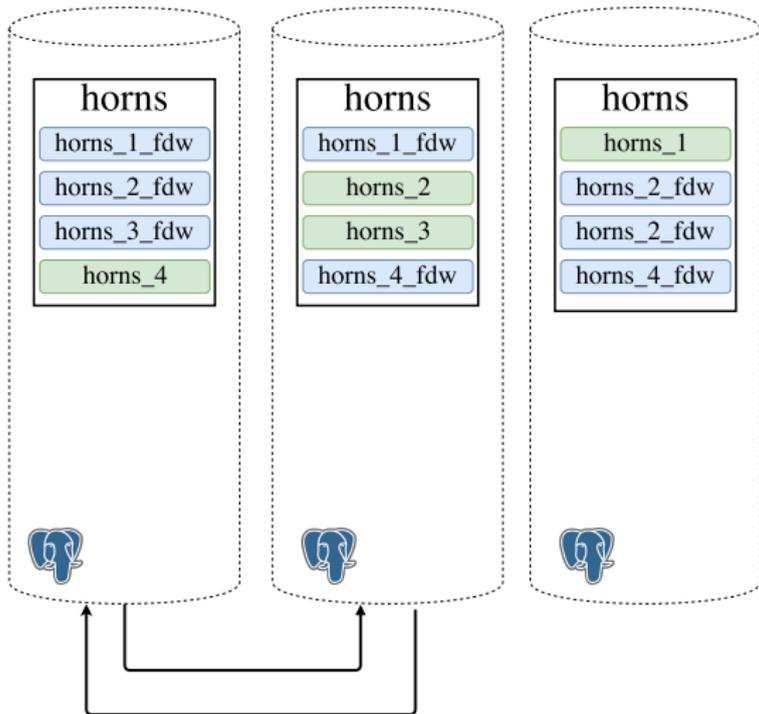
Transactions in shardman: isolation

BEGIN



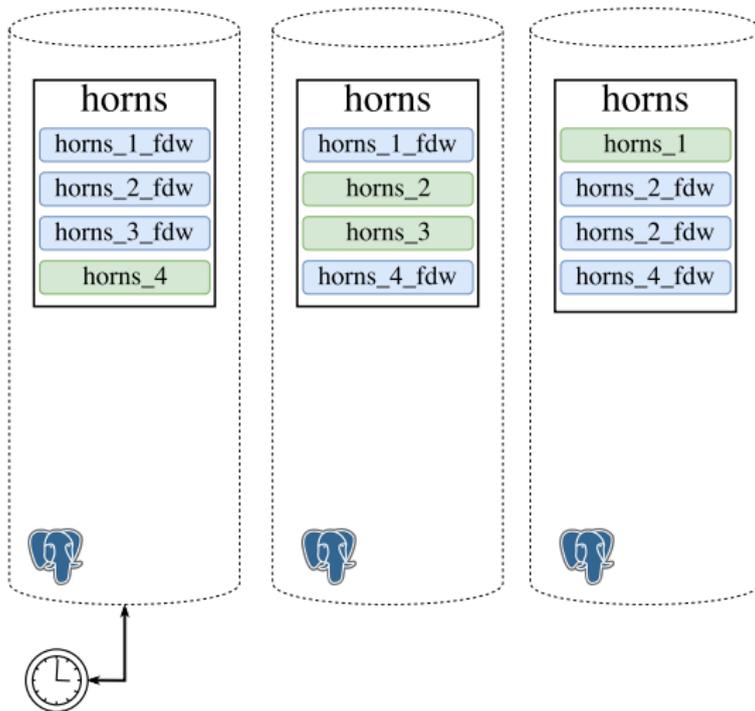
Transactions in shardman: isolation

Do some serious web scale stuff



Transactions in shardman: isolation

COMMIT



Clock-SI slightly changes visibility rules:

version = timestamp

- ▶ Visibility': Waits if tuple came from future. (Do not allow time-travel paradoxes!)
- ▶ Visibility'': Waits if tuple already prepared(P) but not yet committed(C).
- ▶ Commit': Receives local versions from partitions on Prepare and Commits with maximal version.

Transactions in shardman

And that's it! Thank you.