Kyle in Absentia

Jepsen test of Datomic, and Unusual Intra-Transaction Semantics

Adrian Cockcroft (Nubank - advisory role)

HPTS.ws – September 2024

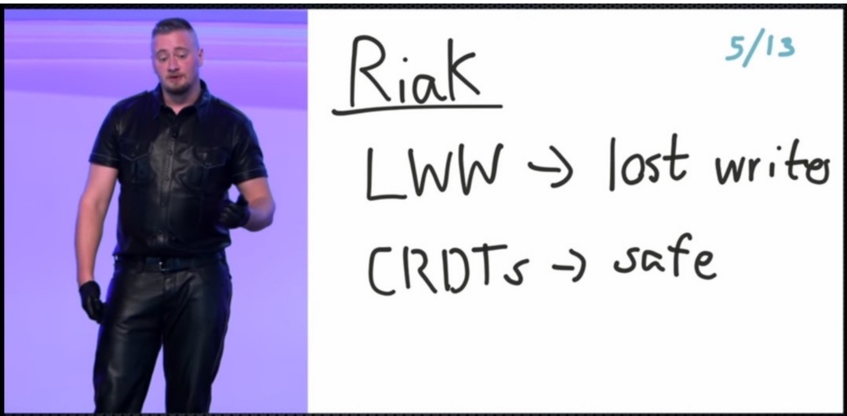
These slides will be shared

Previously...

Screen shots from Jepsen 9: A Fsyncing Feeling - GOTO Chicago 2018 - https://www.youtube.com/watch?v=tRc0O9VgzB0

Every → ← thing

is ↗ ↖ fine!

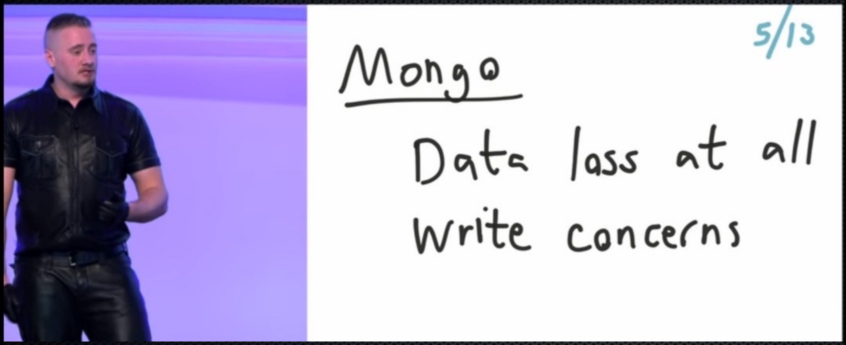


5/13

Riak

LWW → lost write

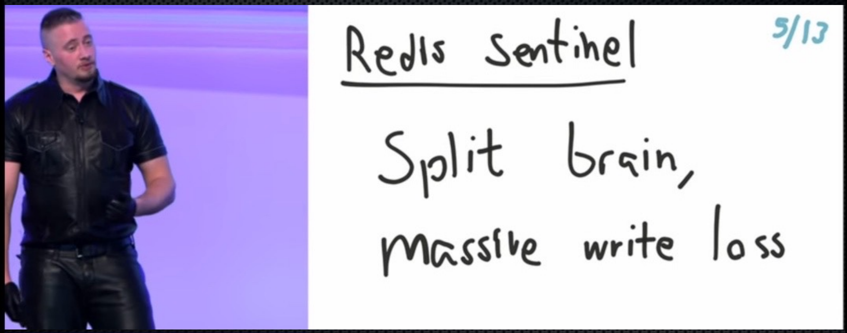
CRDTs → safe



5/13

Mongo

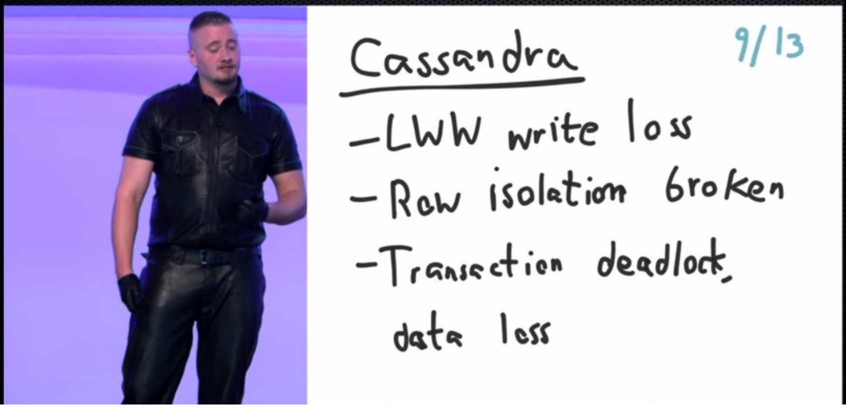
Data loss at all write concerns



5/13

Redis Sentinel

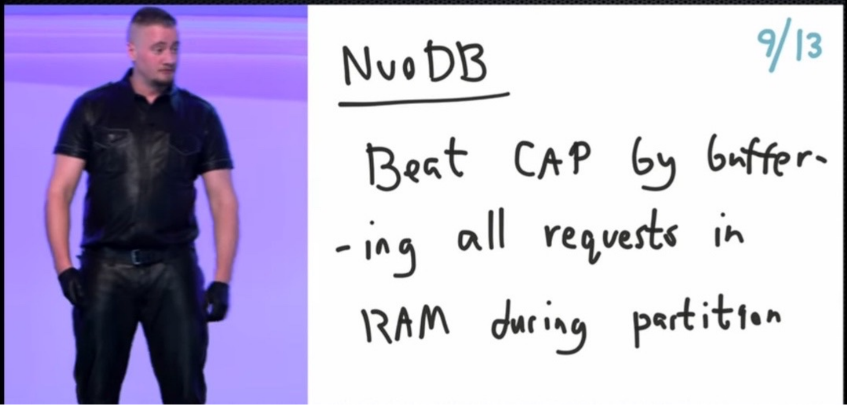
Split brain, massive write loss



9/13

Cassandra

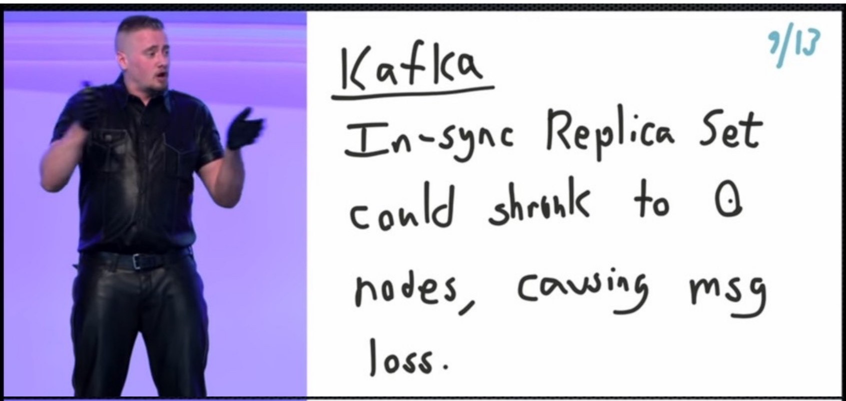
- LWW write loss
- Row isolation broken
- Transaction deadlock, data loss



9/13

NuoDB

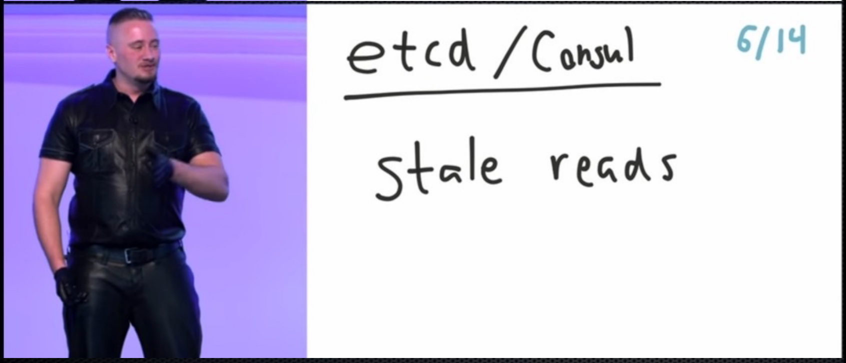
Beat CAP by buffering all requests in RAM during partition



Kafka

9/13

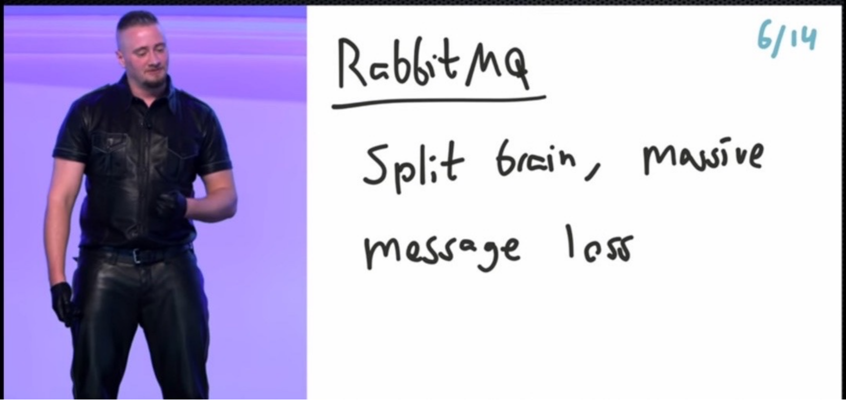
In-sync Replica Set could shrunk to 0 nodes, causing msg loss.



etcd / Consul

6/14

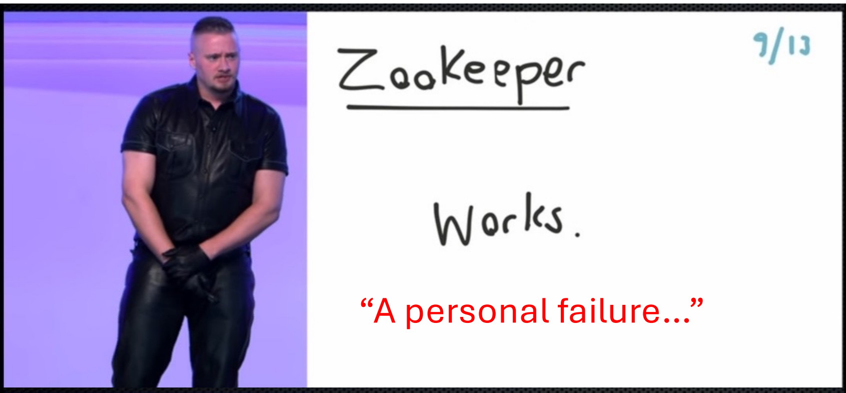
stale reads



RabbitMQ

6/14

Split brain, massive message loss

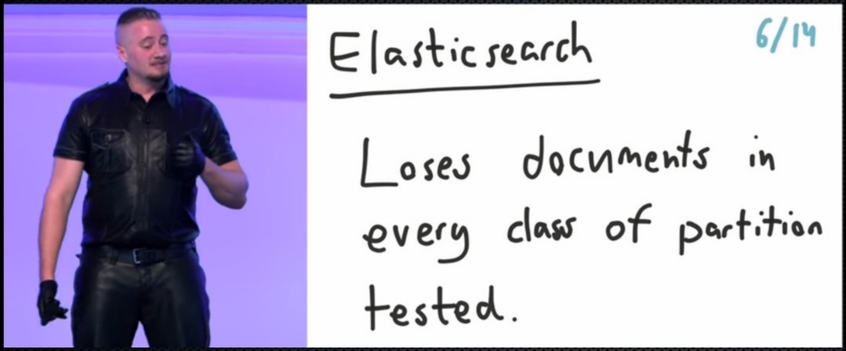


Zookeeper

9/13

Works.

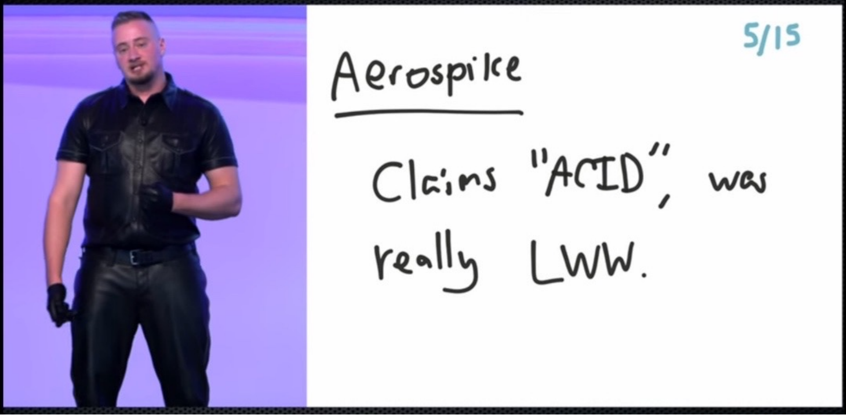
“A personal failure...”



Elasticsearch

6/14

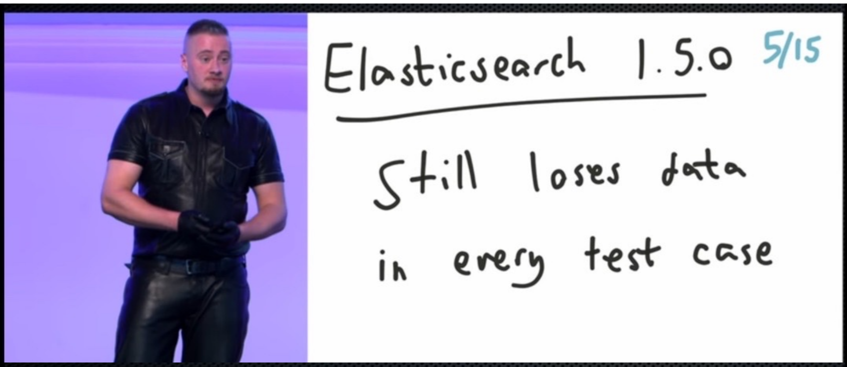
Loses documents in every class of partition tested.



Aerospike

5/15

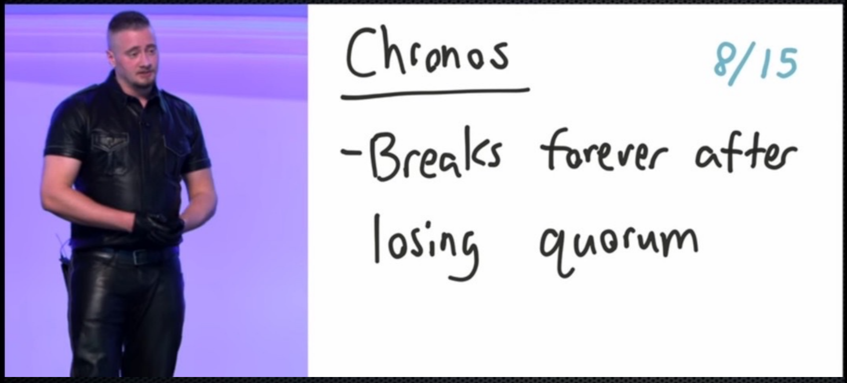
Claims "ACID", was really LWW.



Elasticsearch 1.5.0

5/15

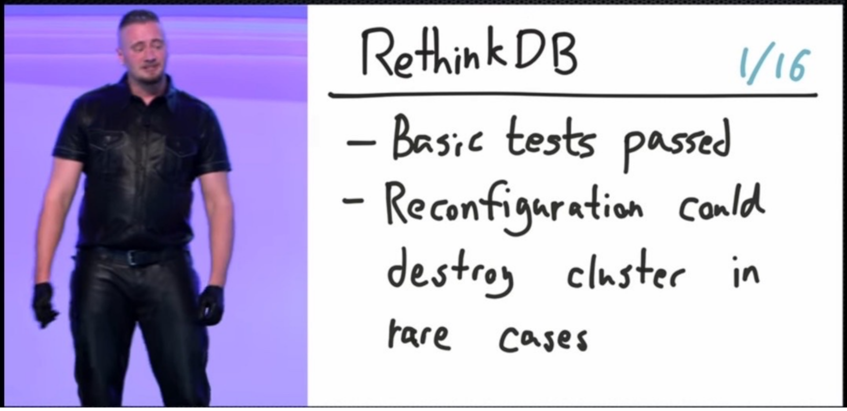
Still loses data in every test case



Chronos

8/15

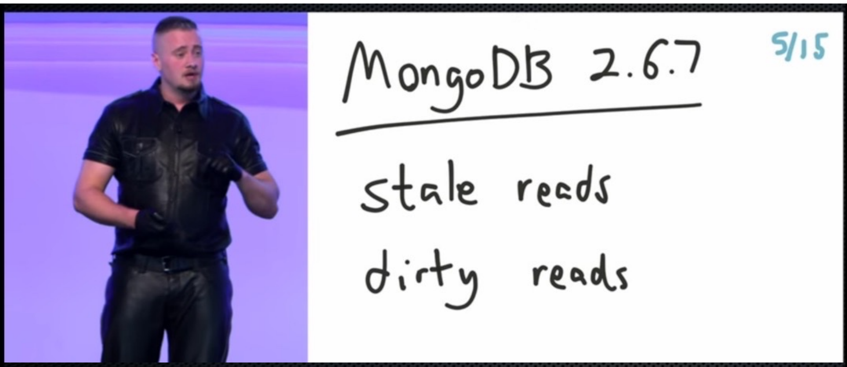
- Breaks forever after losing quorum



RethinkDB

1/16

- Basic tests passed
- Reconfiguration could destroy cluster in rare cases

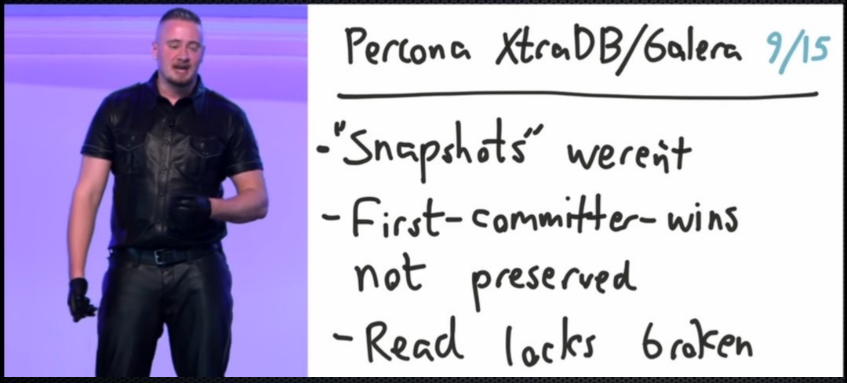


MongoDB 2.6.7

5/15

stale reads

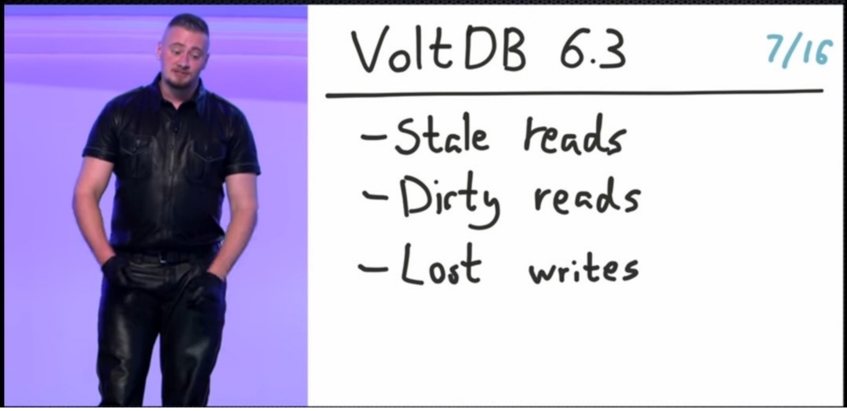
dirty reads



Percona XtraDB/Galera

9/15

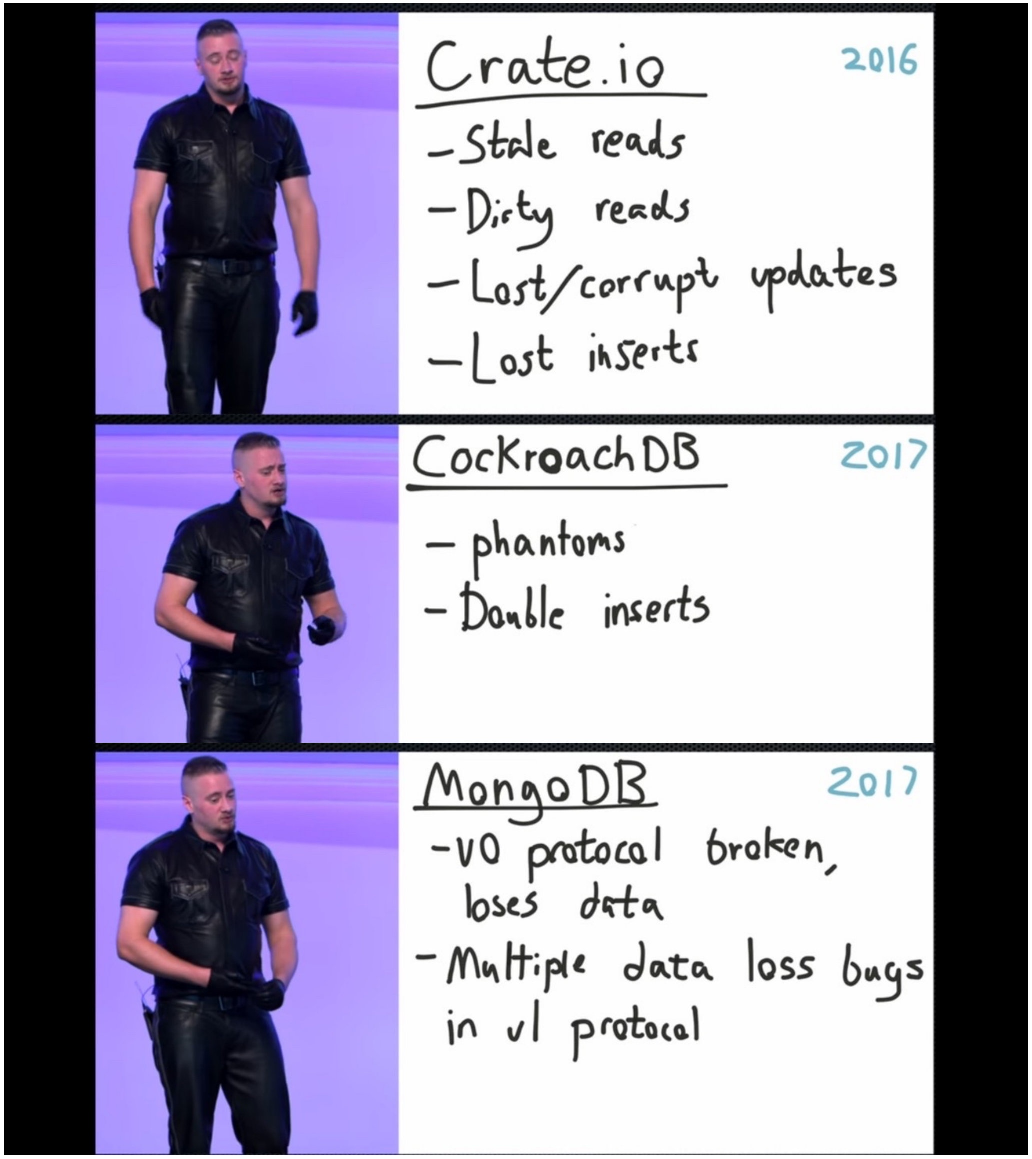
- "Snapshots" weren't
- First-committer-wins not preserved
- Read locks broken



VoltDB 6.3

7/16

- Stale reads
- Dirty reads
- Lost writes



Crate.io 2016

- Stale reads
- Dirty reads
- Lost/corrupt updates
- Lost inserts

CockroachDB 2017

- phantoms
- Double inserts

MongoDB 2017

- v0 protocol broken, loses data
- Multiple data loss bugs in v1 protocol

Aerospike	2015-05-04	3.5.4
	2018-03-07	3.99.0.3
Cassandra	2013-09-24	2.0.0
Chronos	2015-08-10	2.4.0
CockroachDB	2017-02-16	beta-20160829
Crate	2016-06-28	0.54.9
Datomic	2024-05-15	1.0.7075
Dgraph	2018-08-23	1.0.2
	2020-04-30	1.1.1
Elasticsearch	2014-06-15	1.1.0
	2015-04-27	1.5.0
etcd	2014-06-09	0.4.1
	2020-01-30	3.4.3
FaunaDB	2019-03-05	2.5.4
Hazelcast	2017-10-06	3.8.3
jetcd	2024-08-08	0.8.2
Kafka	2013-09-24	0.8 beta
MariaDB Galera	2015-09-01	10.0
MongoDB	2013-05-18	2.4.3
	2015-04-20	2.6.7
	2017-02-07	3.4.0-rc3
	2018-10-23	3.6.4
	2020-05-15	4.2.6
MySQL	2023-12-19	8.0.34
NuoDB	2013-09-23	1.2
Percona XtraDB Cluster	2015-09-04	5.6.25
PostgreSQL	2020-06-12	12.3
RabbitMQ	2014-06-06	3.3.0
Radix DLT	2022-02-05	1.0-beta.35.1
RavenDB	2024-01-31	6.0.2
Redis	2013-05-18	2.6.13
	2013-12-10	WAIT
Redis-Raft	2020-06-23	1b3fbf6
Redpanda	2022-04-29	21.10.1
RethinkDB	2016-01-04	2.1.5
	2016-01-22	2.2.3
Riak	2013-05-19	1.2.1
Scylla	2020-12-23	4.2-rc3
Tendermint	2017-09-05	0.10.2
TiDB	2019-06-12	2.1.7
VoltDB	2016-07-12	6.3
YugaByte DB	2019-03-26	1.1.9
	2019-09-05	1.3.1
Zookeeper	2013-09-23	3.4.5

Jepsen test of Datomic Pro 1.0.7075

- Nubank depends on Datomic backed by AWS DynamoDB to run almost all our workloads for over 105 million customers

- 2020 Nubank aquired Cognitect, authors of Clojure and Datomic

- Jepsen.io collaboration with Nubank, initiated and observed by Adrian Cockcroft, Kyle worked closely with Dan Aguiar, Guilherme Baptista, Stu Halloway, Keith Harper, and Chris Redinger

Full Jepsen Report - https://jepsen.io/blog/2024-05-15-datomic-pro-1.0.7075
Systems Distributed 2024 Talk by Kyle including Datomic https://www.youtube.com/watch?v=ecZp6cWhDjg

What is Datomic?

- *Datomic is a temporal Entity-Attribute-Value OLTP database which supports non-interactive transactions on top of pluggable storage engines.*

- *It offers a variety of query mechanisms across thick and thin clients, including Datalog, graph traversal, and an ODM-style API.*

- *At any instant in time, the state of the database is represented by a set of* [entity, attribute, value] *(EAV) triples, known as datoms.*

Jepsen test of Datomic Pro 1.0.7075

We found that Datomic's inter-transaction safety properties appeared stronger than claimed.

Datomic Pro appeared to offer Strong Session Serializable isolation, and Strong Serializable for histories restricted to update transactions.

However, Datomic defines unusual intra-transaction semantics.

While consistent with Datomic's documentation, this could cause invariants preserved by individual transaction functions to be broken when those same functions are applied within a single transaction.

Datomic broke Kyle!

https://jepsen.io/blog/2024-05-15-datomic-pro-1.0.7075

How do Datomic Transactions Behave?

- *Most OLTP databases offer interactive transactions: one begins a transaction, submits an operation, receives results from that operation, submits another, and so on before finally committing.*

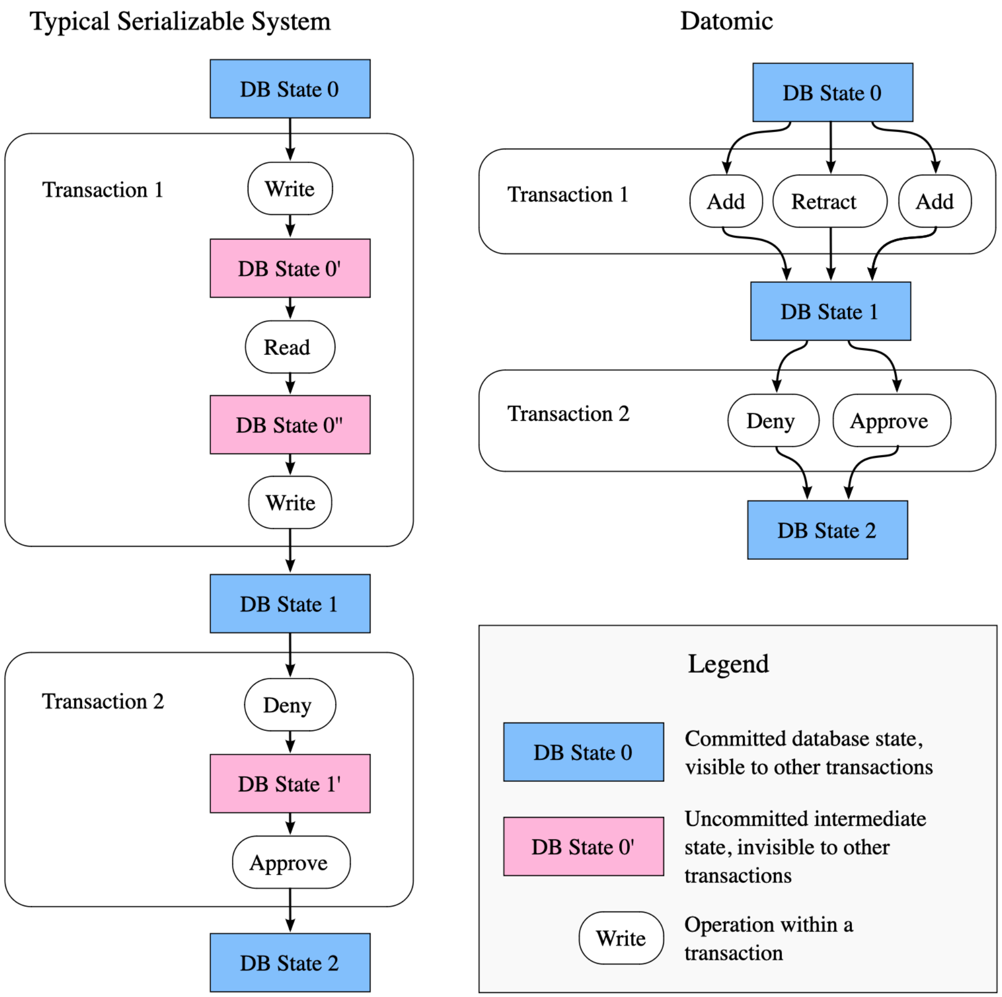
- *Datomic does something rather different. It enforces a strict separation between read and write paths. There are no interactive transactions.*

How do Datomic Transactions Work?

- *Instead of offering arbitrary return values from transactions, every call to* transact *returns the database state just before the transaction, the database state the transaction produced, and the set of datoms the transaction expanded to.*

- *Datomic offers a view of an alternate universe: one where database snapshots are cheap, efficient, and can be passed from node to node with just a timestamp.*

(From this point of view, other databases feel impoverished. What do you mean, Postgres can't give you the state of the entire database a transaction observed?)



Legend

DB State 0 — Committed database state, visible to other transactions

DB State 0' — Uncommitted intermediate state, invisible to other transactions

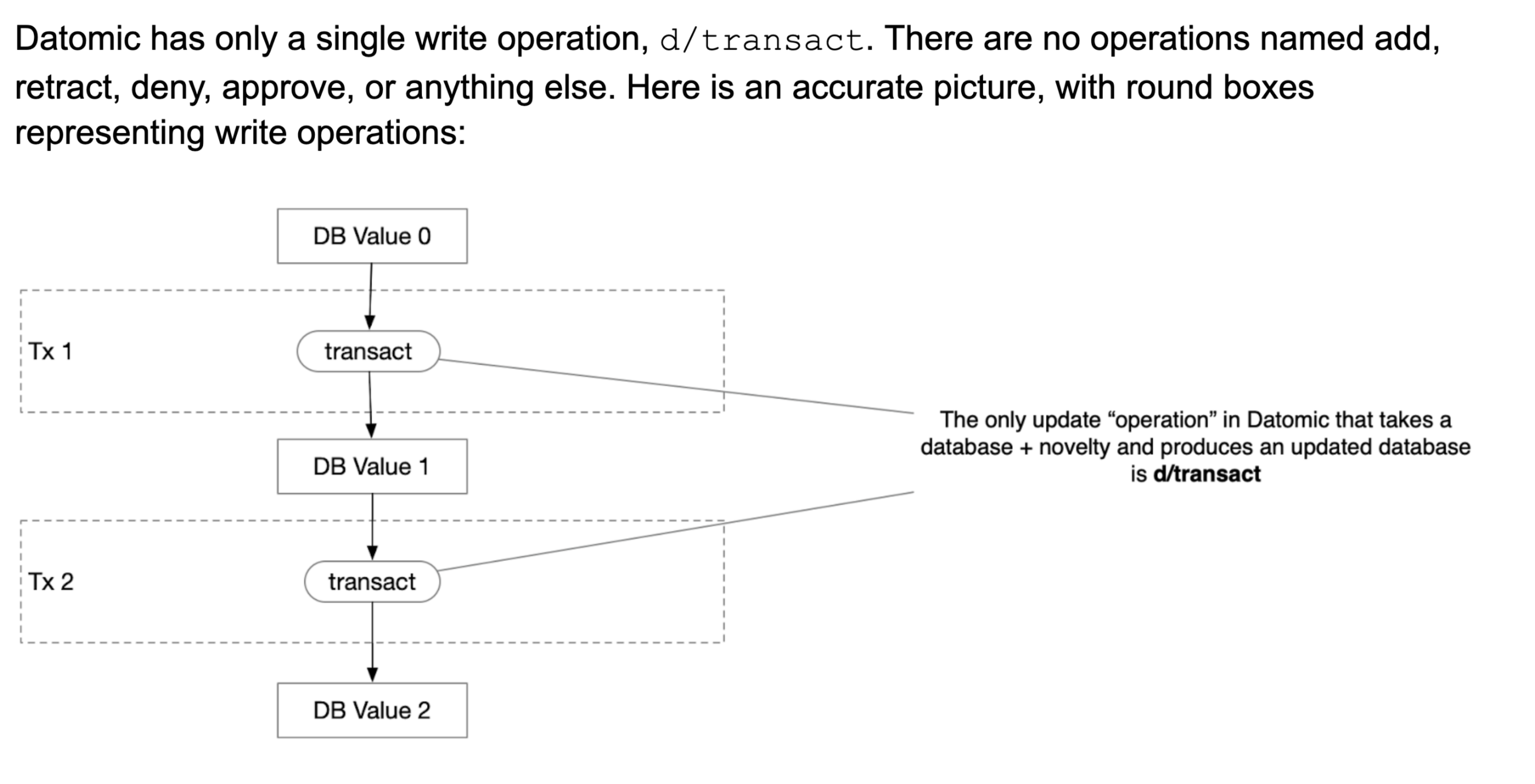
Write — Operation within a transaction

Jepsen Report Diagram

However – this diagram for Datomic is incorrect…

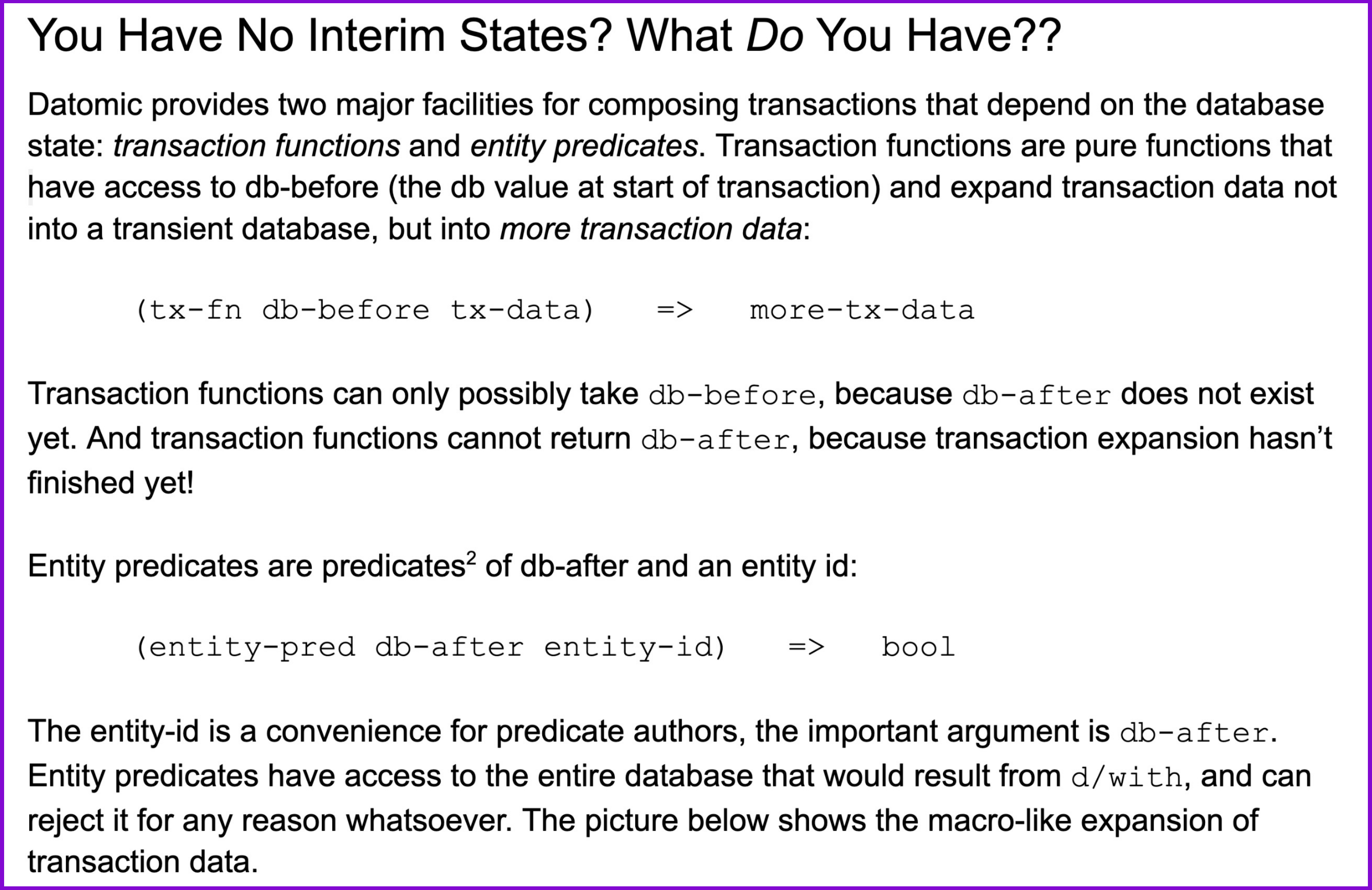
Datomic does not have any operations called Add, Retract etc.

Diagram from Nubank Datomic team clarifying the issue



Datomic has only a single write operation, `d/transact`. There are no operations named add, retract, deny, approve, or anything else. Here is an accurate picture, with round boxes representing write operations:

The only update “operation” in Datomic that takes a database + novelty and produces an updated database is **d/transact**



You Have No Interim States? What *Do* You Have??

Datomic provides two major facilities for composing transactions that depend on the database state: *transaction functions* and *entity predicates*. Transaction functions are pure functions that have access to db-before (the db value at start of transaction) and expand transaction data not into a transient database, but into *more transaction data*:

```
        (tx-fn db-before tx-data)    =>    more-tx-data
```

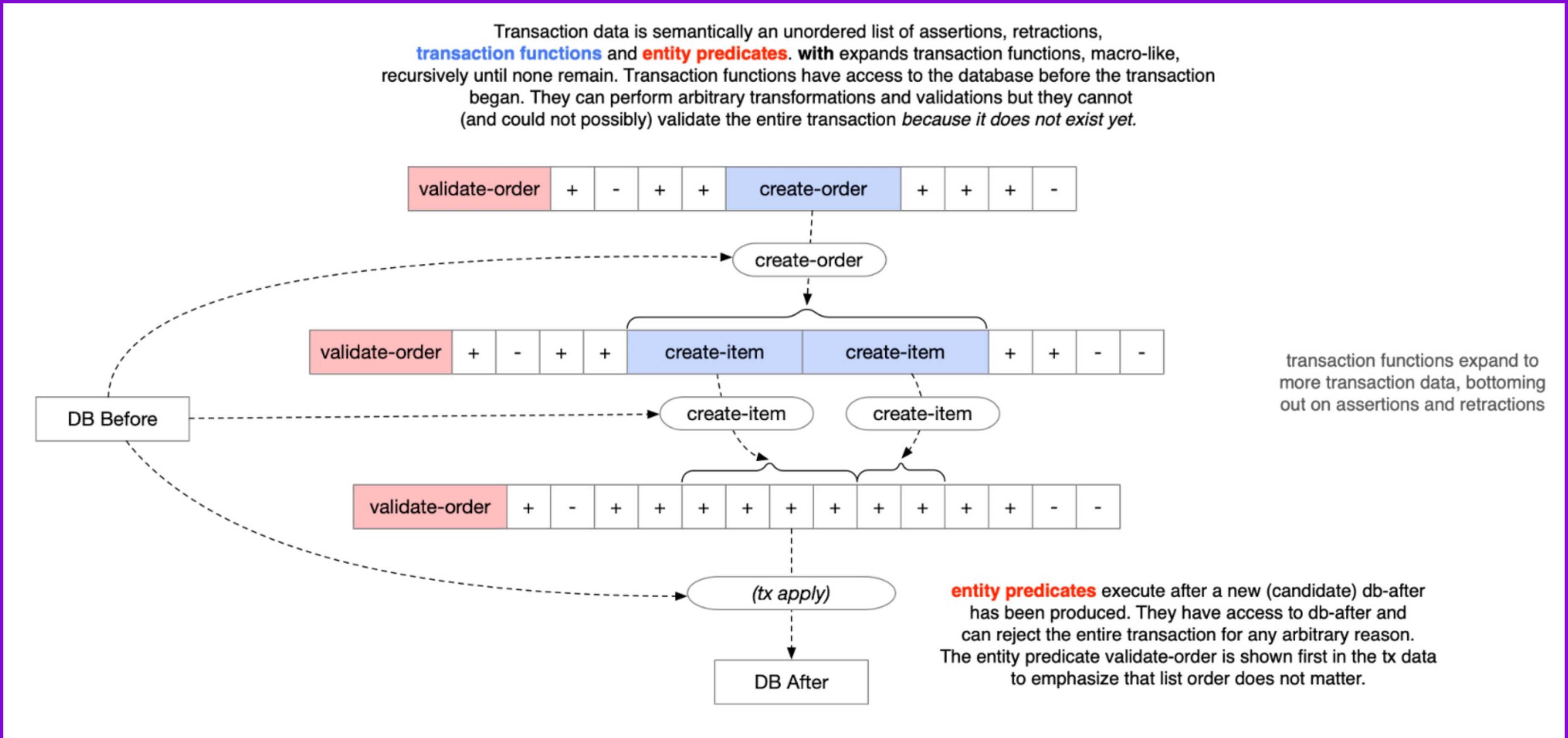
Transaction functions can only possibly take `db-before`, because `db-after` does not exist yet. And transaction functions cannot return `db-after`, because transaction expansion hasn't finished yet!

Entity predicates are predicates[2] of db-after and an entity id:

```
        (entity-pred db-after entity-id)    =>    bool
```

The entity-id is a convenience for predicate authors, the important argument is `db-after`. Entity predicates have access to the entire database that would result from `d/with`, and can reject it for any reason whatsoever. The picture below shows the macro-like expansion of transaction data.

Diagram from Nubank Datomic team showing how it works



Transaction data is semantically an unordered list of assertions, retractions, transaction functions and entity predicates. **with** expands transaction functions, macro-like, recursively until none remain. Transaction functions have access to the database before the transaction began. They can perform arbitrary transformations and validations but they cannot (and could not possibly) validate the entire transaction *because it does not exist yet.*

transaction functions expand to more transaction data, bottoming out on assertions and retractions

entity predicates execute after a new (candidate) db-after has been produced. They have access to db-after and can reject the entire transaction for any arbitrary reason. The entity predicate validate-order is shown first in the tx data to emphasize that list order does not matter.

Revisions to Datomic Documentation

Following our collaboration, Datomic has made extensive revisions to their documentation.

First, we worked together to rewrite Datomic's transaction safety documentation. It now reflects the stronger safety properties we believe Datomic actually offers: Serializability globally, monotonicity on each peer, and Strict Serializability when restricted to writes, or reads which use sync. Datomic also removed the "single-writer" argument from their safety documentation.

Datomic's docs now include a comprehensive explanation of transaction syntax and semantics. It covers the structure of transaction requests, the rules for expanding map forms and transaction functions, and the process of applying a transaction. Expanded documentation for transaction functions explains Datomic's various mechanisms for ensuring consistency, how to create and invoke functions, and the behavior of built-in functions. The transaction function documentation no longer says they can be used to "atomically analyze and transform database values", nor does it claim transaction functions can "ensure atomic read-modify-write processing".

Datomic has also added documentation arguing for a difference between Datomic transactions and SQL-style "updating transactions." There is also a new tech note which discusses the differences between transaction functions and entity predicates when composing transactions.

Read the docs!
CAREFULLY