



The Database as a Value

Rich Hickey

What is Datomic?

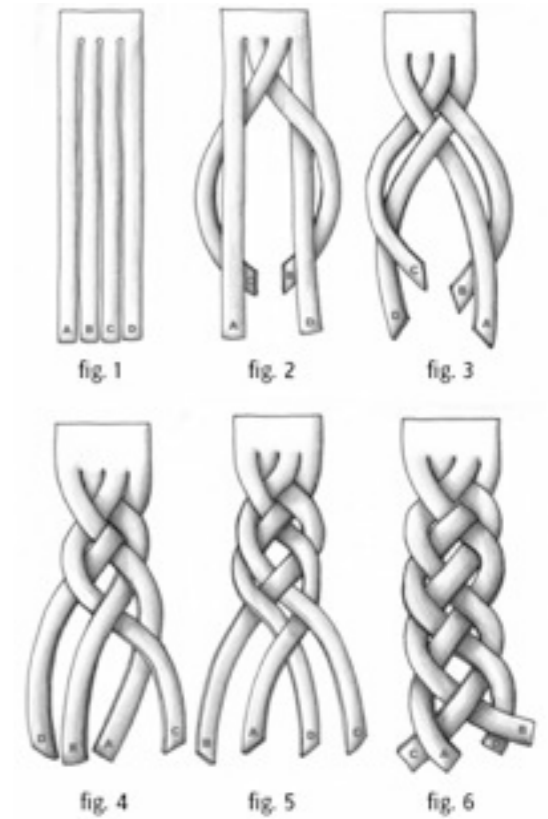
- A new database
- Bringing data power *into* the application
- A sound model of information, with **time**
- Enabled by architectural and capacity advances

Complexity

- Out of the Tar Pit

Moseley and Marks (2006)

- Complexity caused by state and control
- Close the loop - **process**



Information

- Inform
 - ‘to convey knowledge via facts’
 - ‘give shape to (the mind)’
- Information
 - the facts

DB Complexity

- Stateful, inextricably
- Same query, different results
 - no basis
- Over there
- 'Update' poorly defined
 - Places

Basis

- Calculation and decision making:
may involve multiple components
may visit a component more than once
- Broken by simultaneous change

Update

- What does update mean?
- Does the new replace the old?
- Granularity? new ____ replace the old ____
- Visibility?

Manifestations

- Wrong programs
- Scaling problems
- Round-trip fears
- Fear of overloading server
- Coupling, e.g. questions with reporting

The Choices

- Coordination
 - how much, and where?
 - process requires it
 - perception shouldn't
- Immutability
 - sine qua non

Coming to Terms

Value

- An immutable magnitude, quantity, number.. or immutable composite thereof

Identity

- A putative entity we associate with a series of causally related values (states) over time

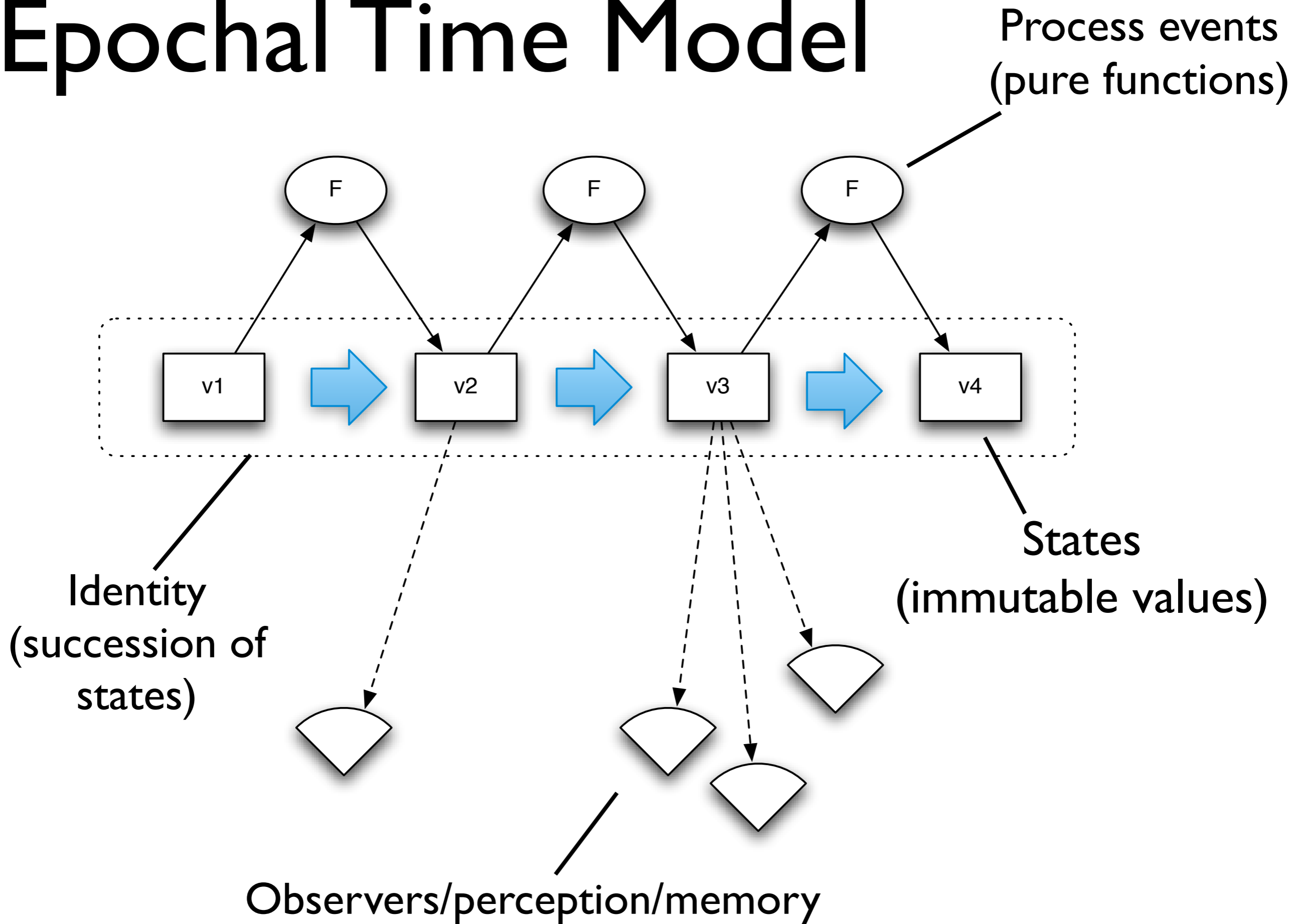
State

- Value of an identity at a moment in time

Time

- Relative before/after ordering of causal values

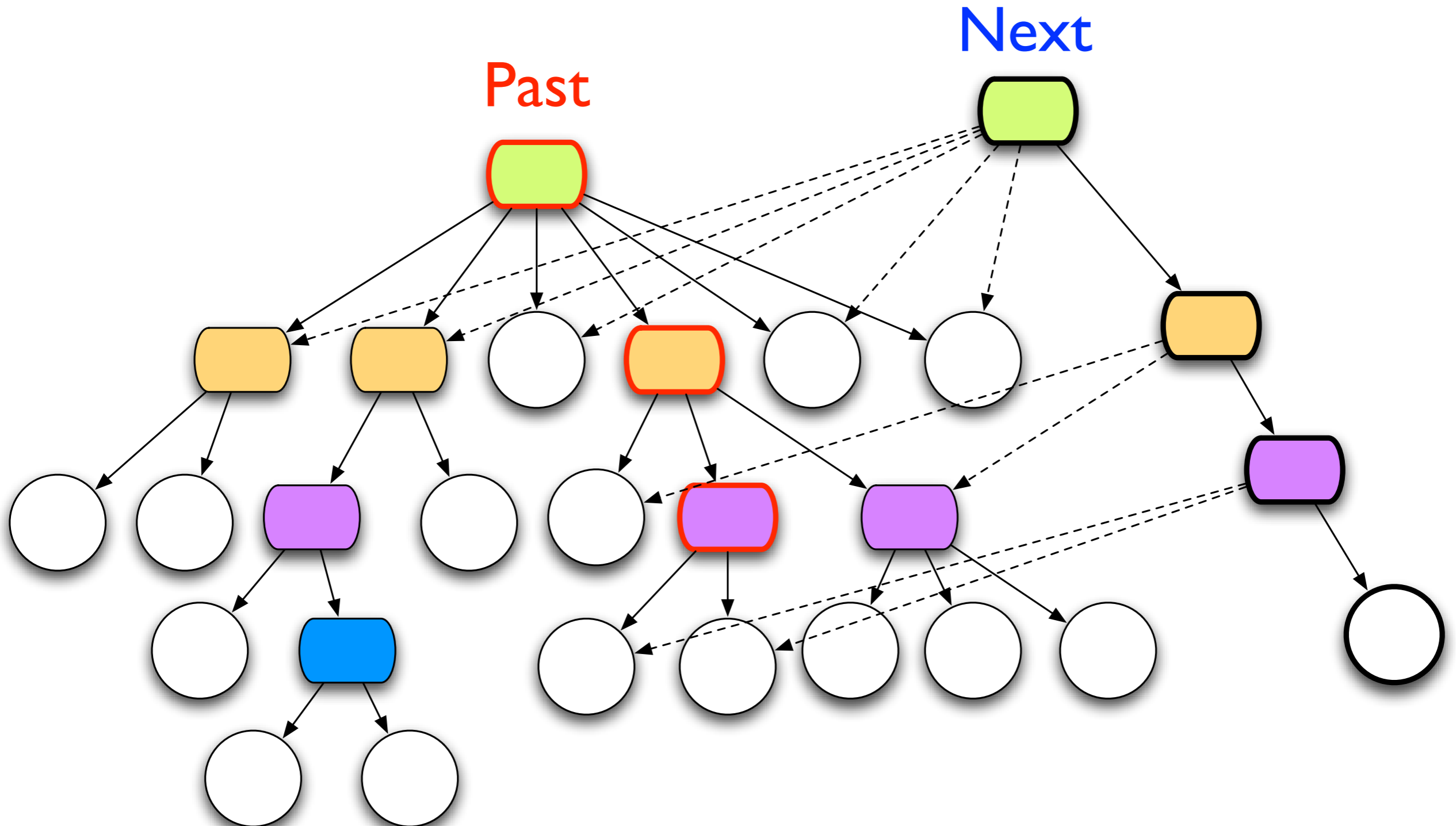
Epochal Time Model



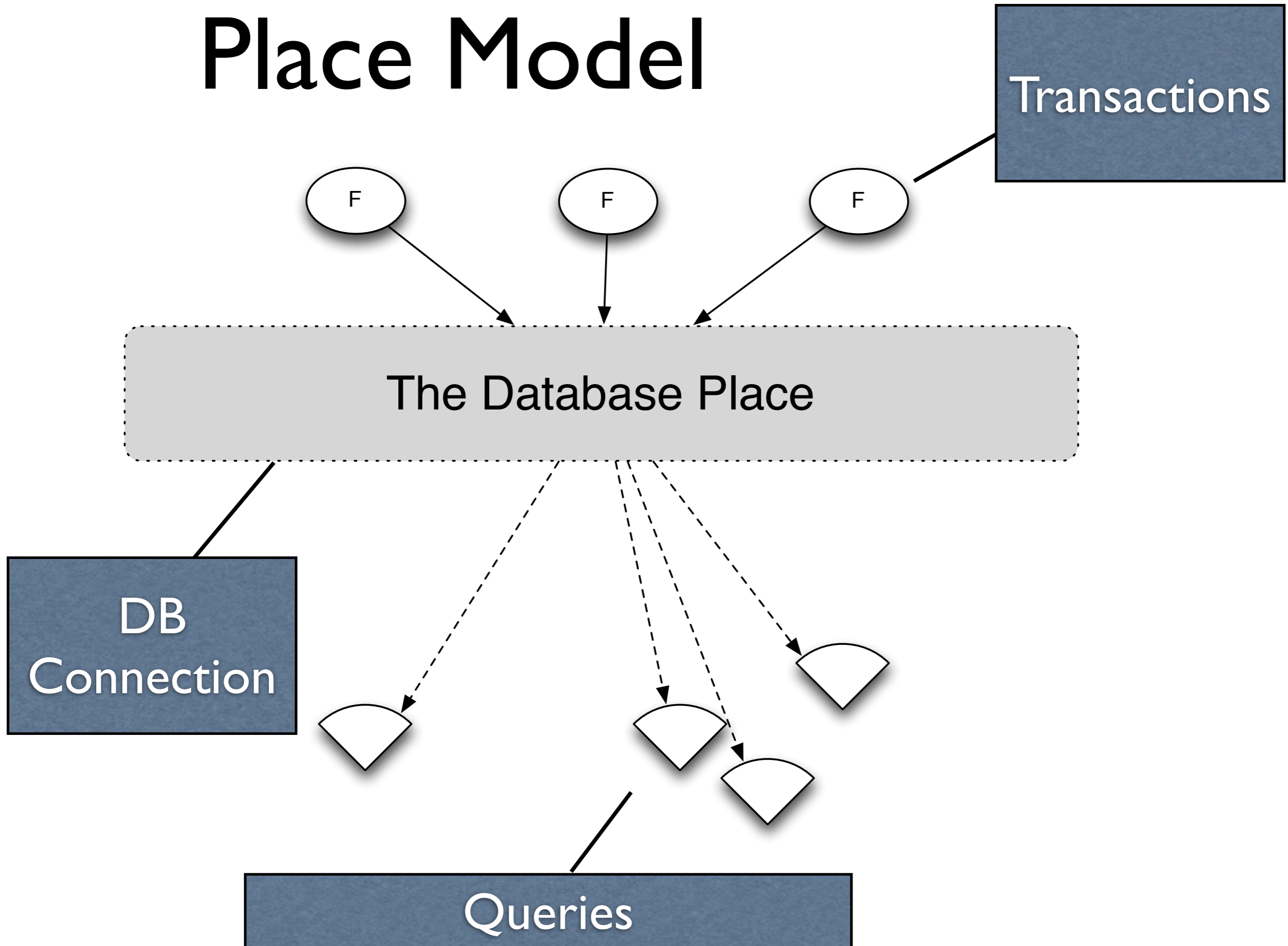
Implementing Values

- Persistent data structures
- Trees
- Structural sharing

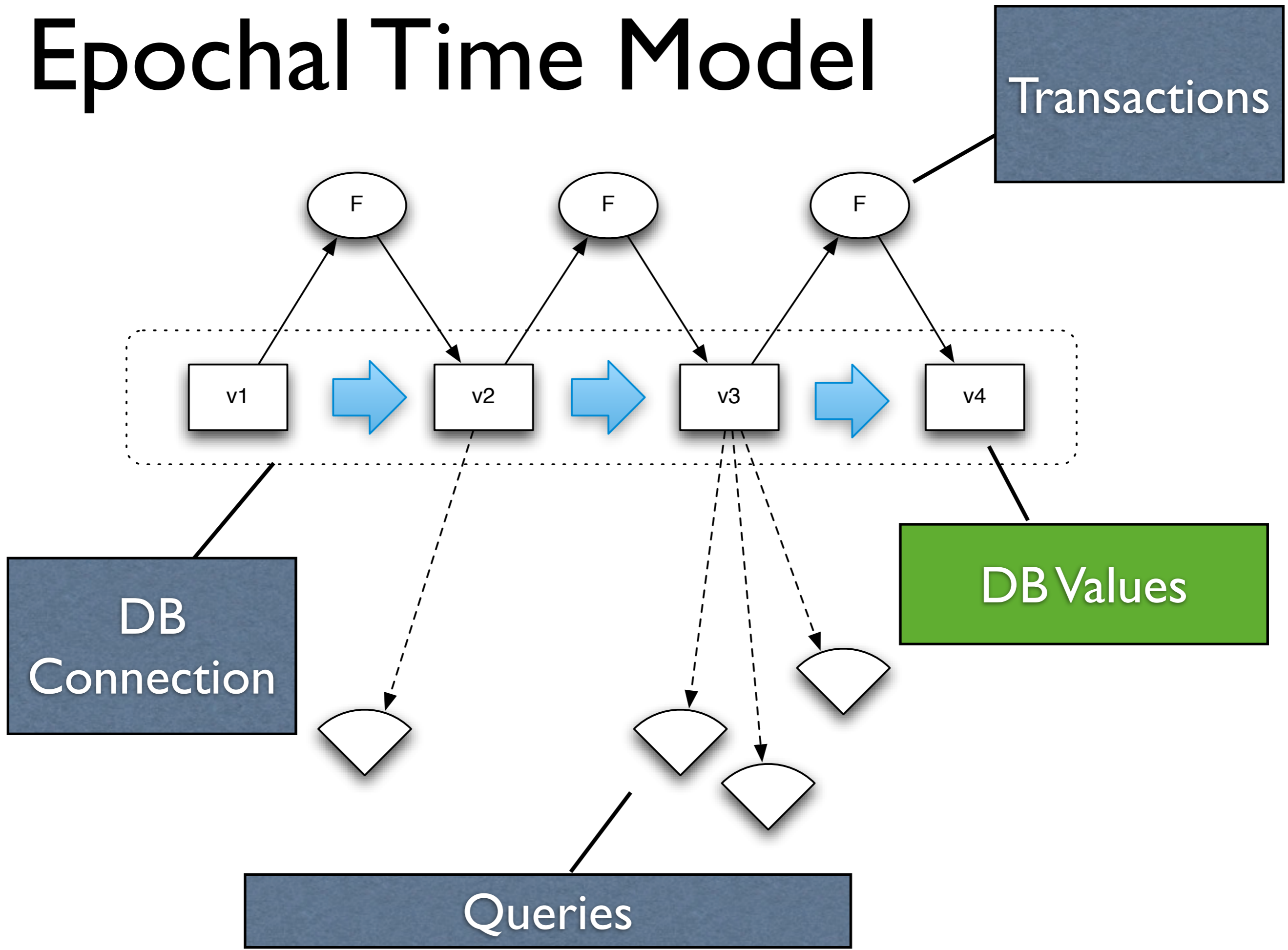
Structural Sharing



Place Model



Epochal Time Model



Database State

- The database as an expanding **value**
- An accretion of **facts**
- The past doesn't change - immutable
- Process requires new space
- Fundamental move away from **places**

Accretion

- Root per transaction doesn't work
 - Crossing processes and time
 - Can't convey/find/maintain roots
 - Can't do global GC
- Instead, latest values include past as well
 - The past is sub-range
- Important for information model

Facts

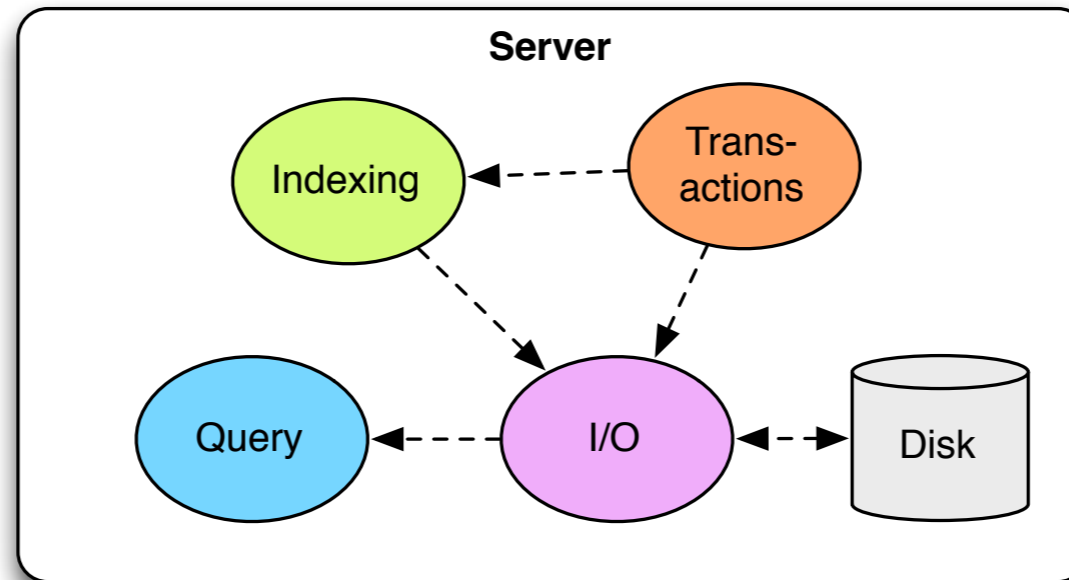
- **Fact** - ‘an event or thing known to have happened or existed’
 - From: factum - ‘something done’
 - Must include time
- Remove structure (a la RDF)
- Atomic **Datom**
 - Entity/Attribute/Value/Transaction

Process

- Reified
- Primitive representation of novelty
 - Assertions and retractions of **facts**
 - **Minimal**
- Other transformations expand into those

Implementation

Deconstruction



- Process

- Transactions

- Indexing

- O

- Perception/Reaction

- Query

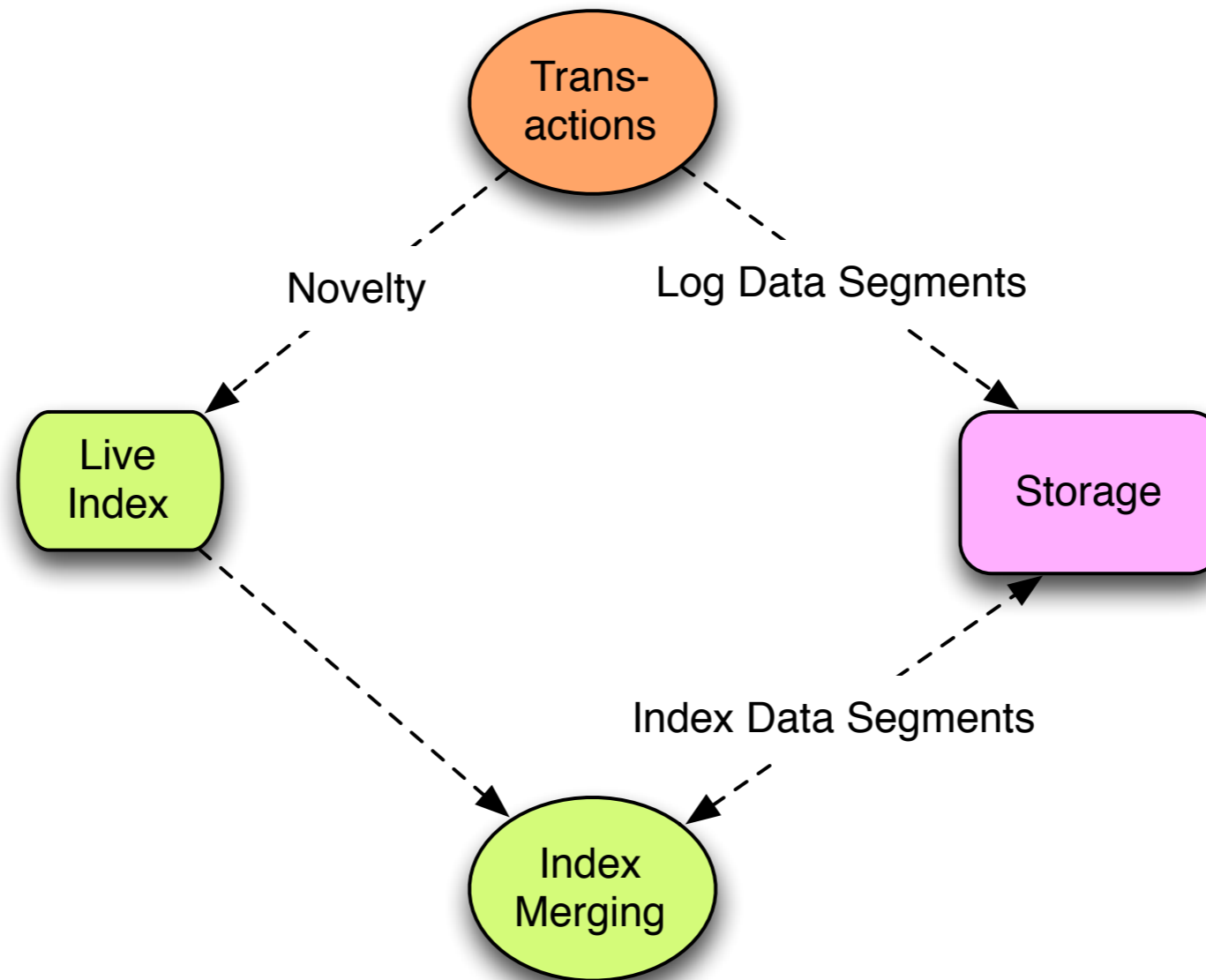
- Indexes

- I

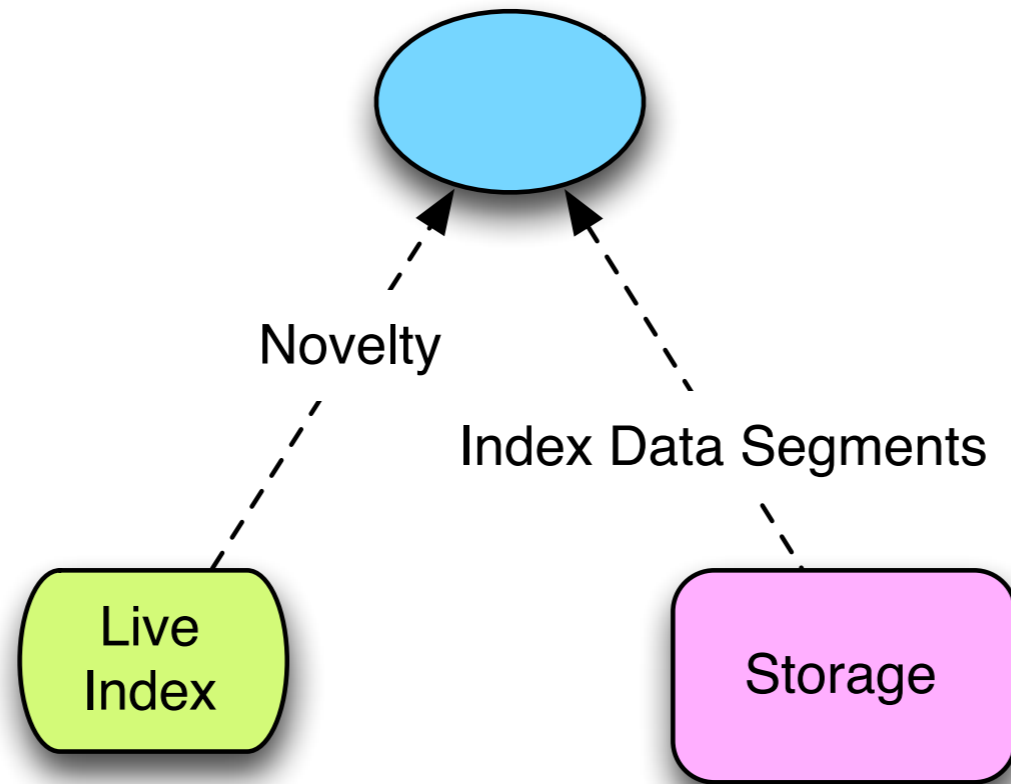
State

- Must be organized to support query
- Sorted set of facts
- Maintaining sort live in storage - bad
 - BigTable - mem + storage merge
 - occasional merge into storage
 - persistent trees

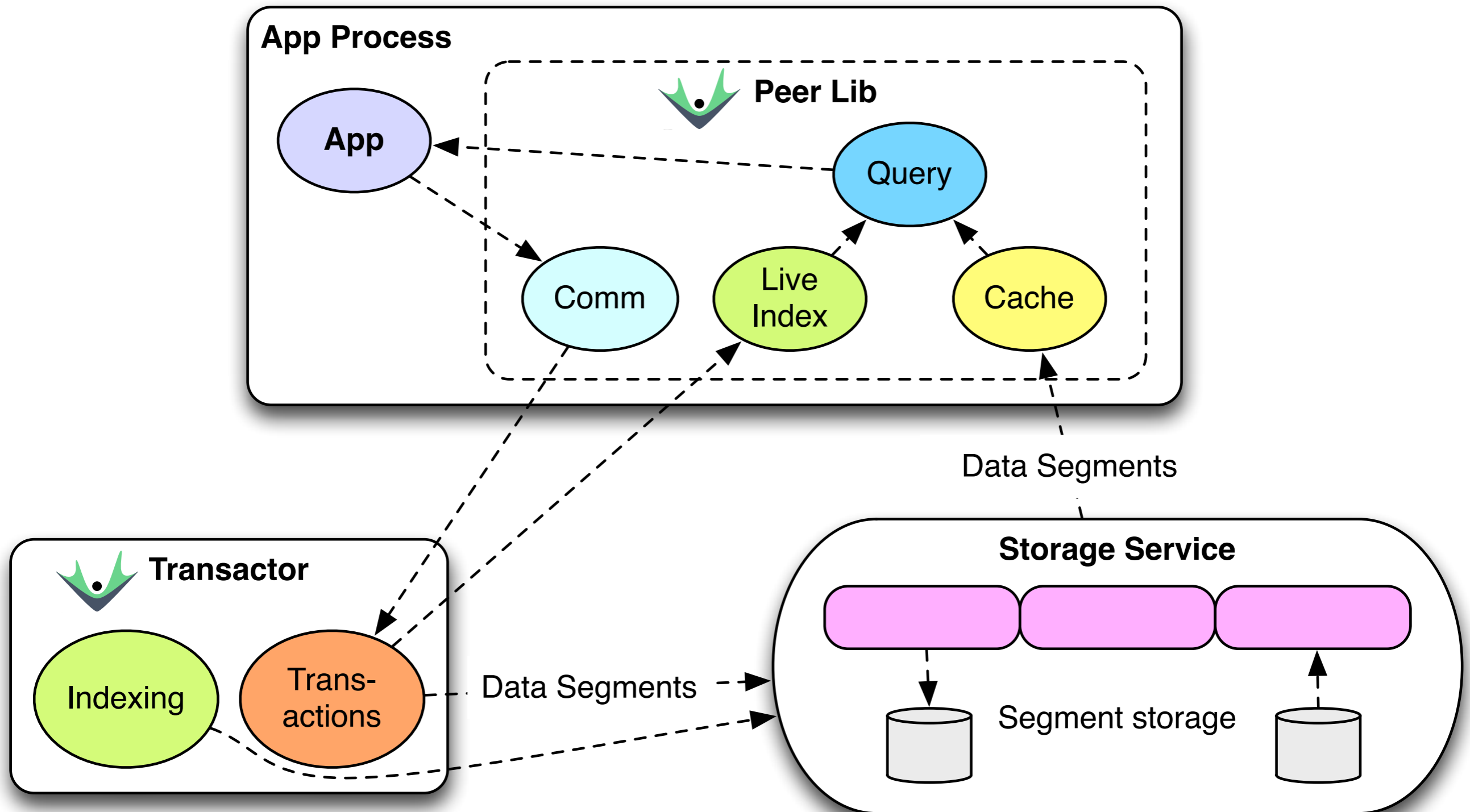
Transactions and Indexing



Perception



Datomic Architecture



Memory Index

- Persistent sorted set
- Large internal nodes
- Pluggable comparators
- 2 sorts always maintained
 - EAVT, AEVT
- plus AVET, VAET

Storage

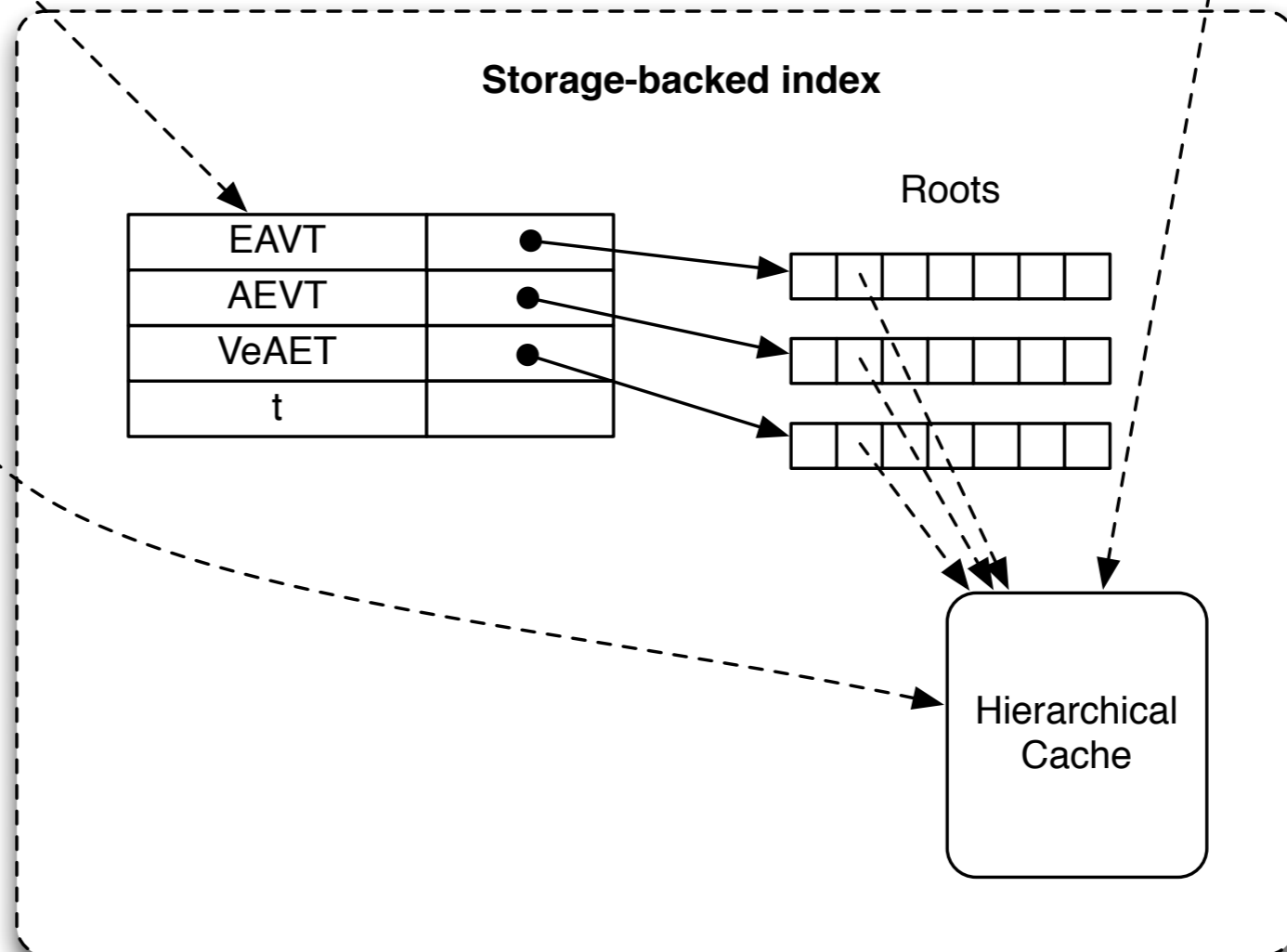
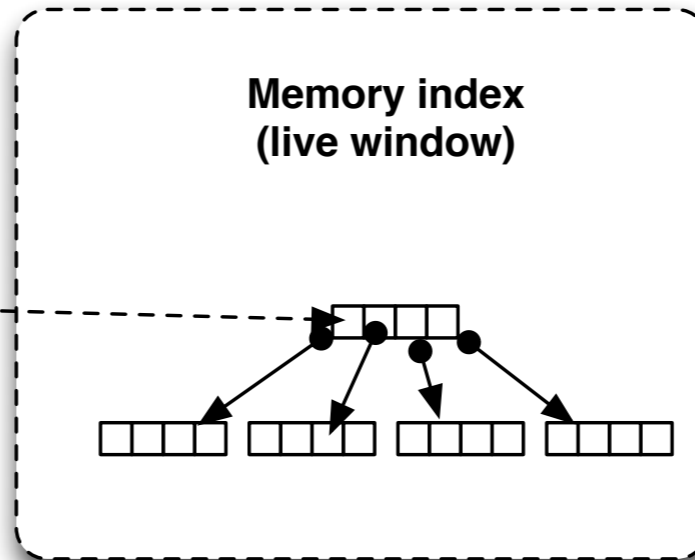
- Log of tx asserts/retracts (in tree)
- Various covering indexes (trees)
- Storage requirements
 - Data segment values (K->V)
 - atoms (consistent read)
 - pods (conditional put)

What's in a DB Value?

Identity

db atom	
db value	
live	---
index	
history	
nextT	
asOfT	
sinceT	
Lucene index	
live Lucene	

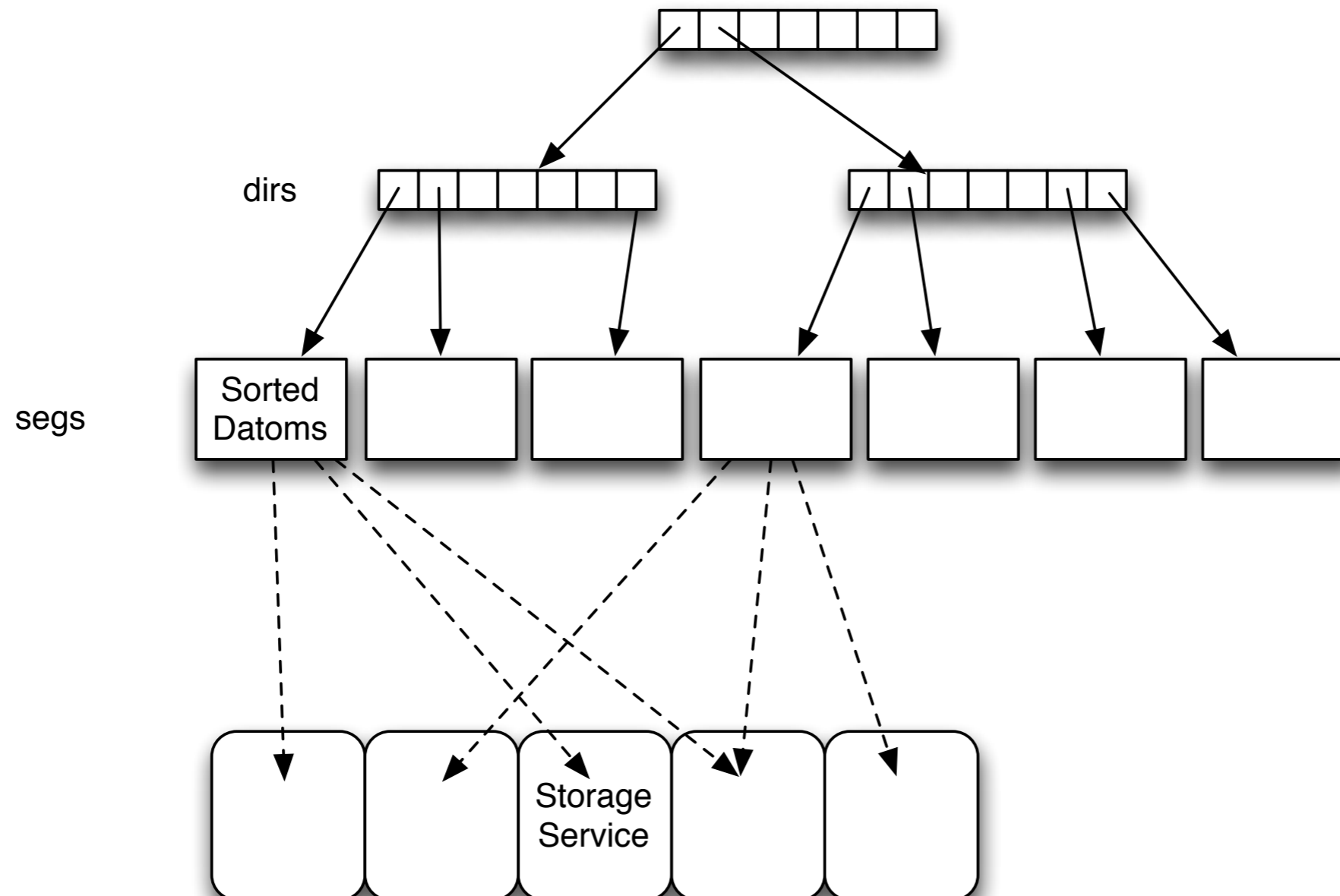
Value



Index Storage

T	EAVT	AEVT	VeAET	AVET	Lucene
42					

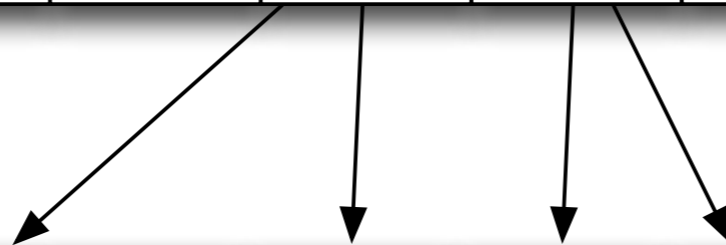
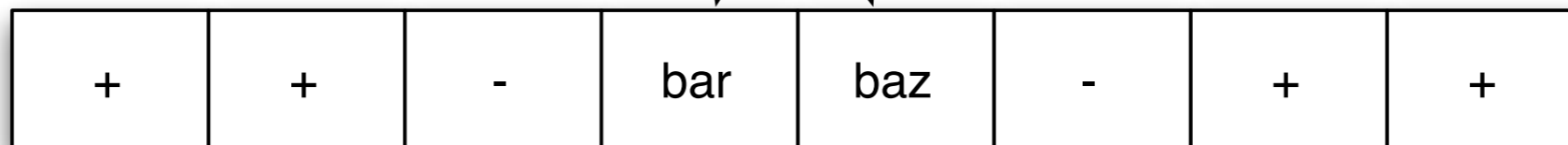
Index Root
of key->dir



Process

- Assert/retract can't express transformation
- Transaction function:
(f db & args) -> tx-data
- tx-data: assert|retract|(tx-fn args...)
- Expand/splice until all assert/retracts

Process Expansion



Transactor

- Accepts transactions
 - Expands, applies, logs, broadcasts
- Periodic indexing, in background
- Indexing creates garbage
 - Storage GC

Peers

- Peers directly access storage service
- Have own query engine
- Have live mem index and merging
- Two-tier cache
 - Segments (on/off heap)
 - Datoms w/object values (on heap)

DB Simplicity

- Epochal state
 - Coordination only for process
- Same query, same results
 - stable bases
- Transactions well defined
 - Functional accretion

Other Benefits

- Communicable, recoverable basis
- Freedom to relocate/scale storage, query
- Time travel - `db.asOf`, `db.since`, `db.asIf`
 - Queries comparing times
- Process events

The Database as a Value

- Dramatically less complex
- More powerful
- More scalable
- Better information model



Thanks for Listening!