The Database as a Value

Rich Hickey

What is Datomic?

- A functional database
- A sound model of information, with time
- Provides database as a value to applications
- Bring declarative programming to applications
- Focus on reducing complexity

DB Complexity

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

Manifestations

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

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

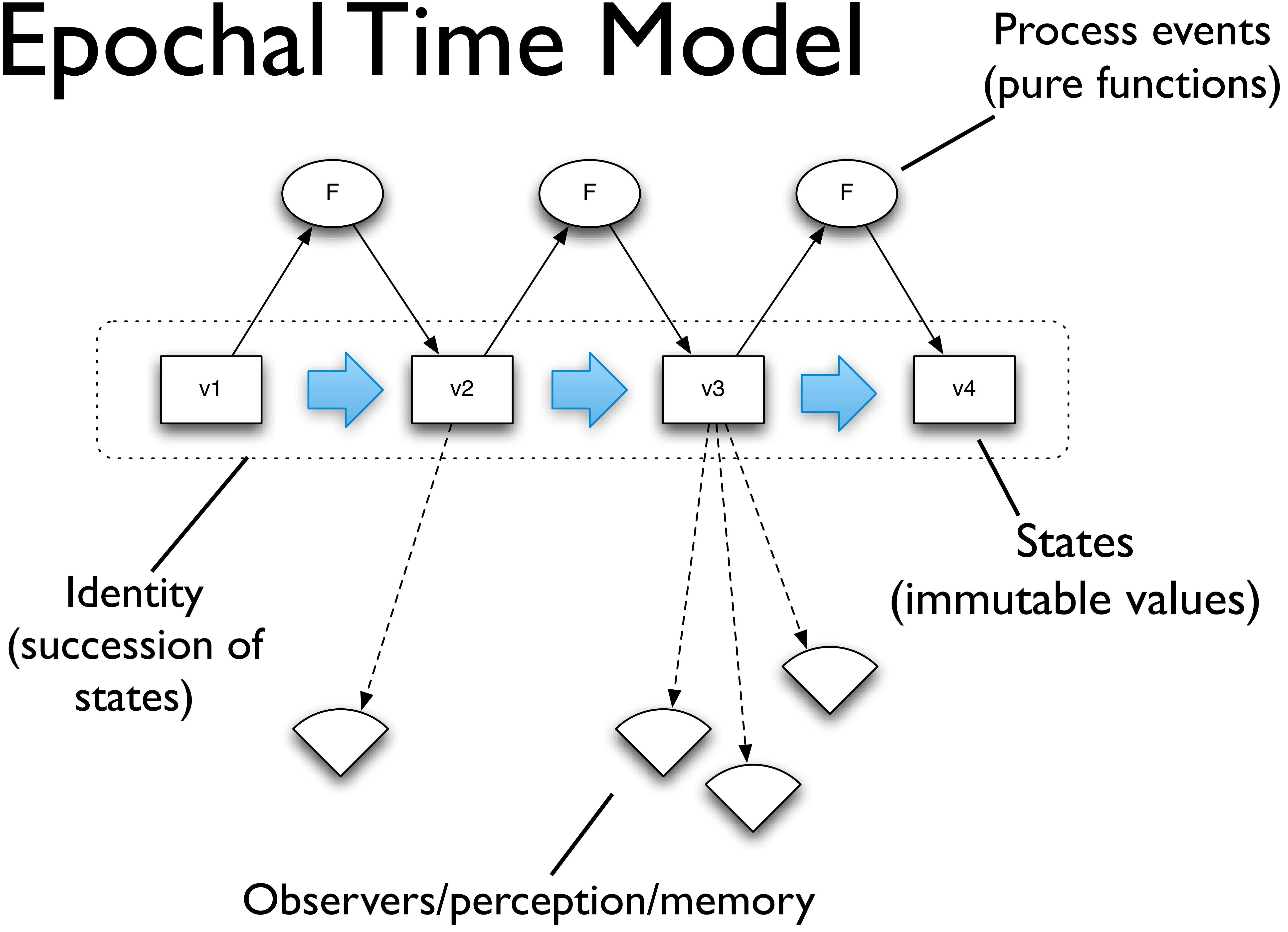
State

- Value of an identity at a moment in time

Time

- Relative before/after ordering of causal values

Epochal Time Model

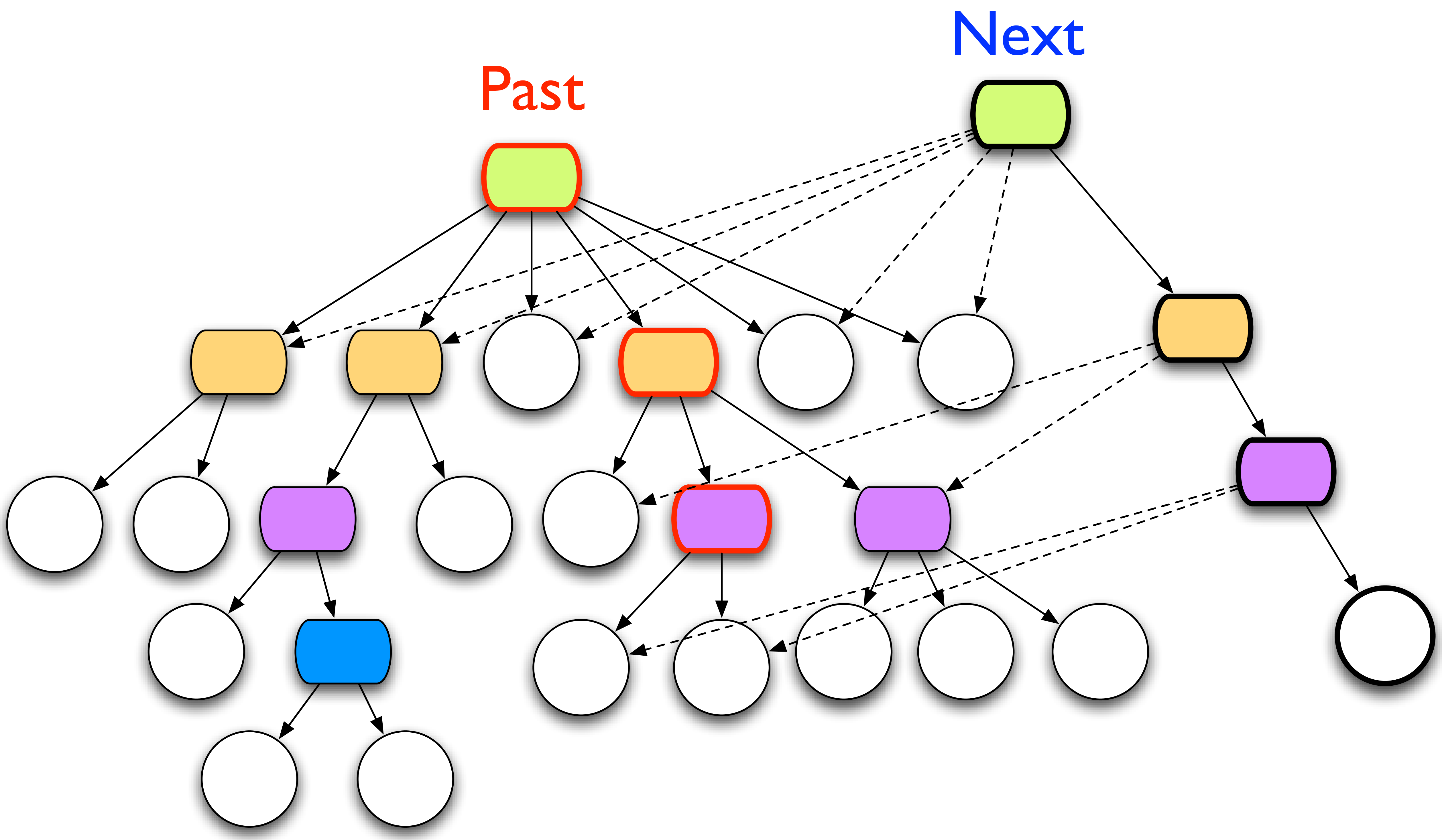


Process events (pure functions)
F
F
F
v1
v2
v3
v4
Identity (succession of states)
States (immutable values)
Observers/perception/memory

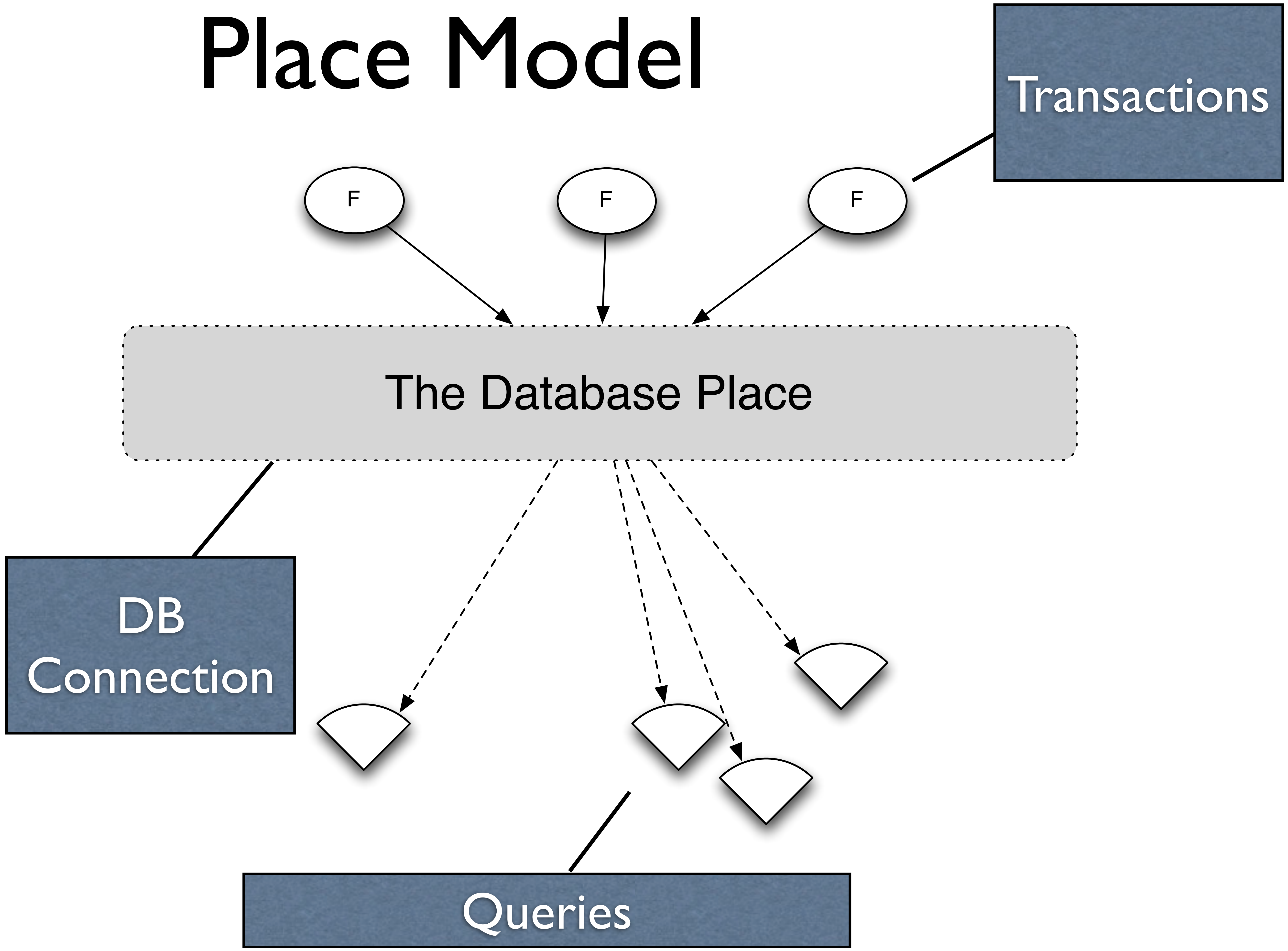
Implementing Values

- Persistent data structures
- Trees
- Structural sharing

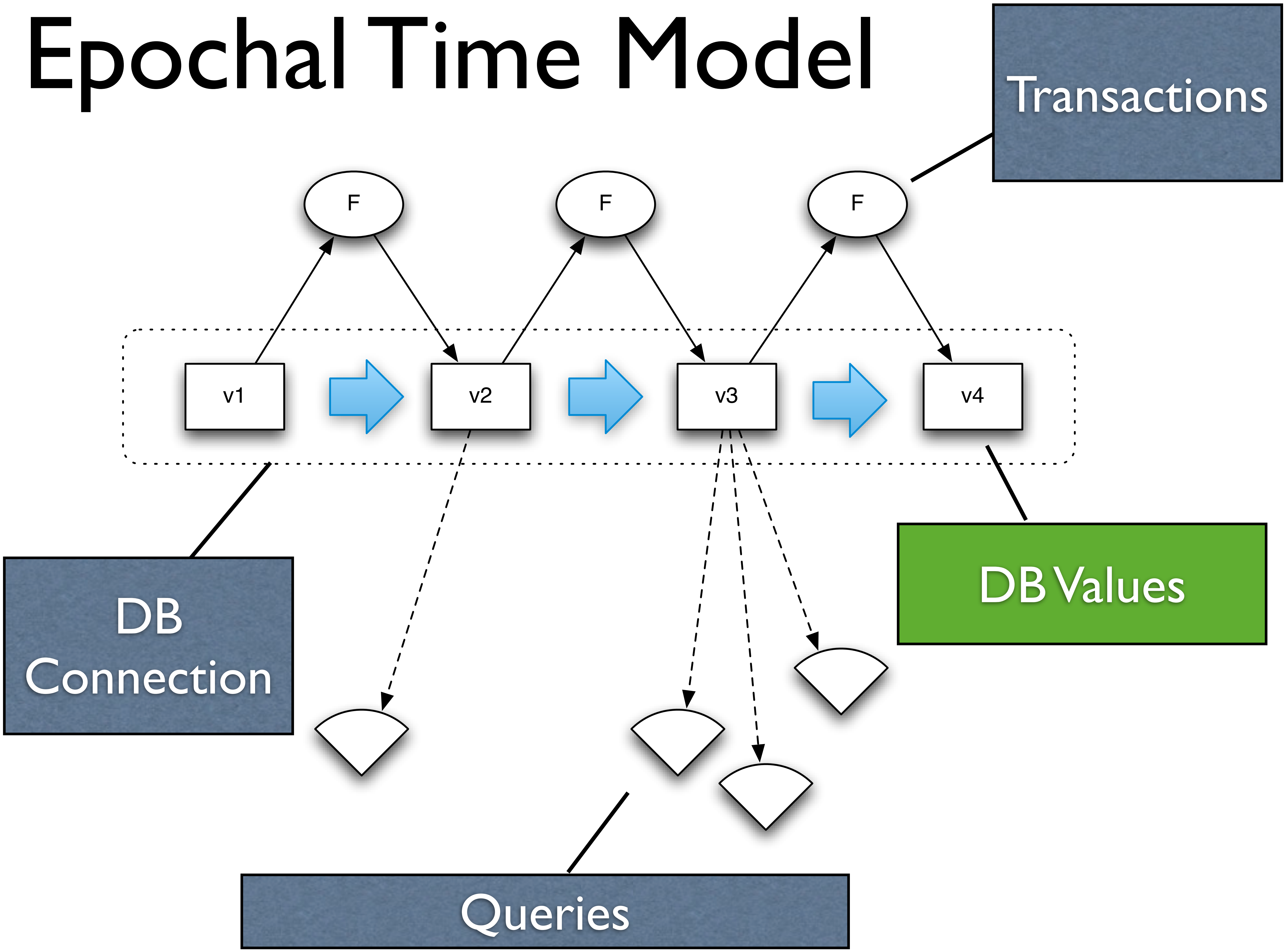
Structural Sharing



Place Model


Transactions
F
F
F
The Database Place
DB Connection
Queries

Epochal Time Model



2 Notions of DB

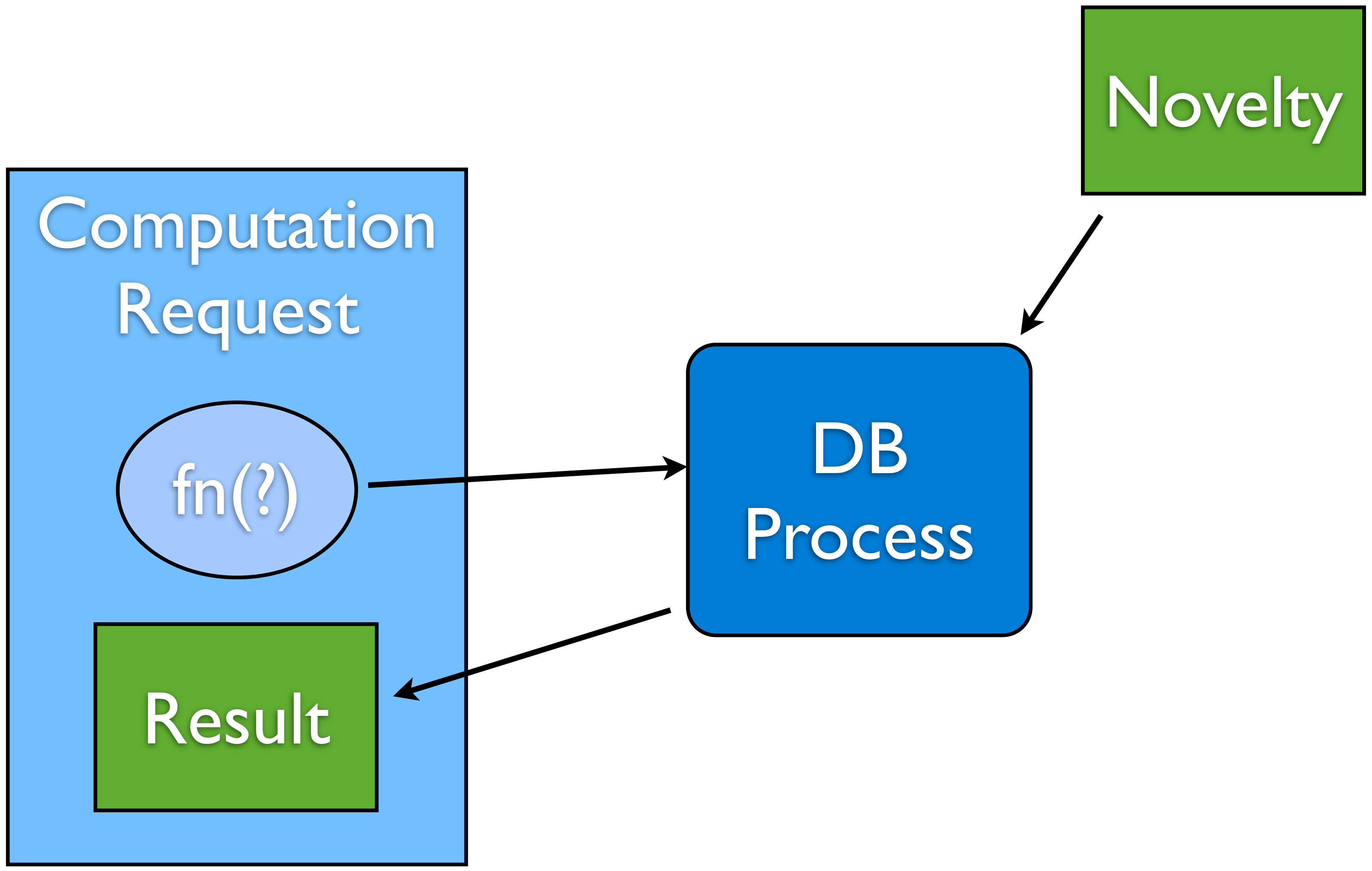
2 Notions of DB

- Database system
    - facilitates the process of creating, sharing, growing db values
    - a machine
    - has identity

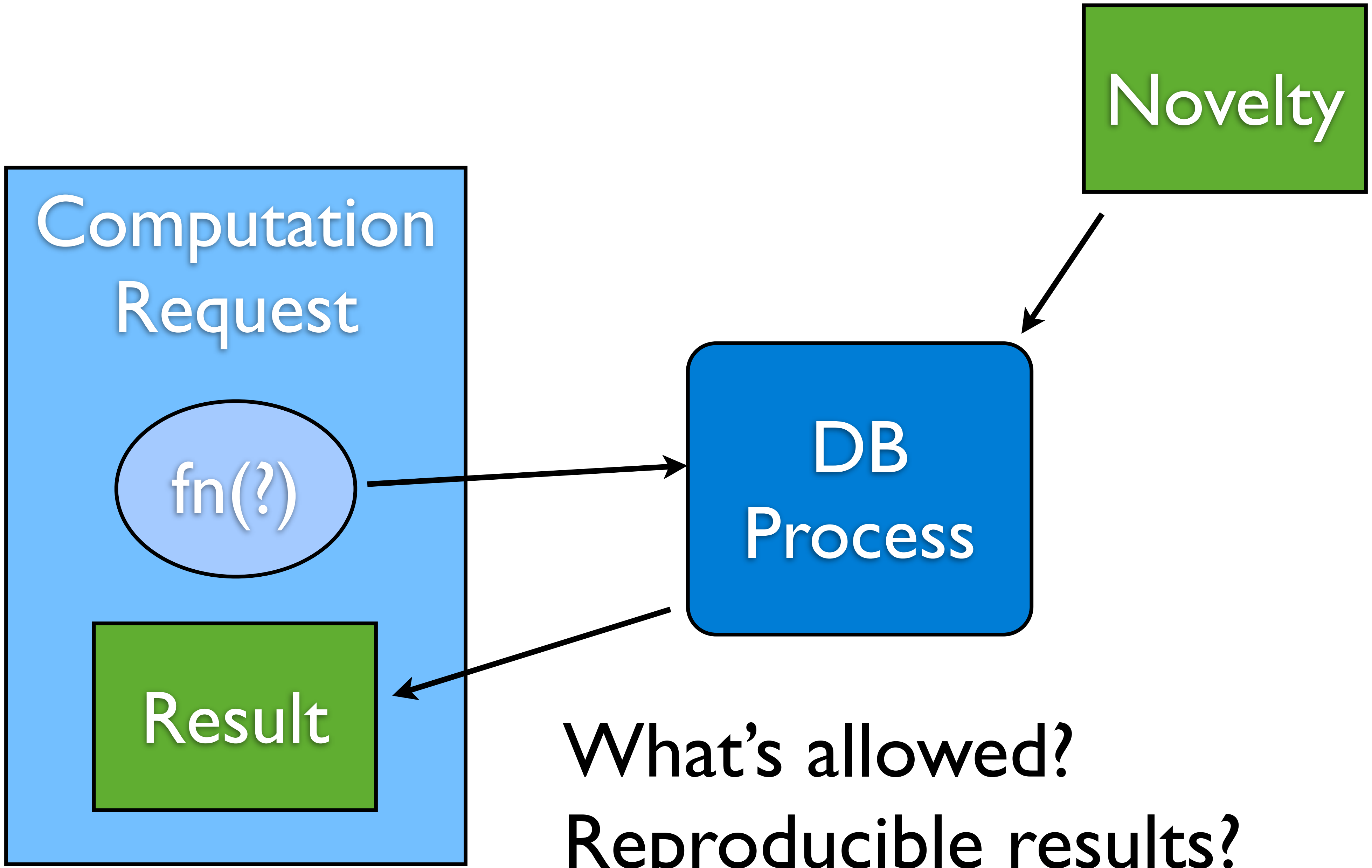
2 Notions of DB

- Database system
    - facilitates the process of creating, sharing, growing db values
    - a machine
    - has identity
- Database values
    - the things with which we compute

DB as Process

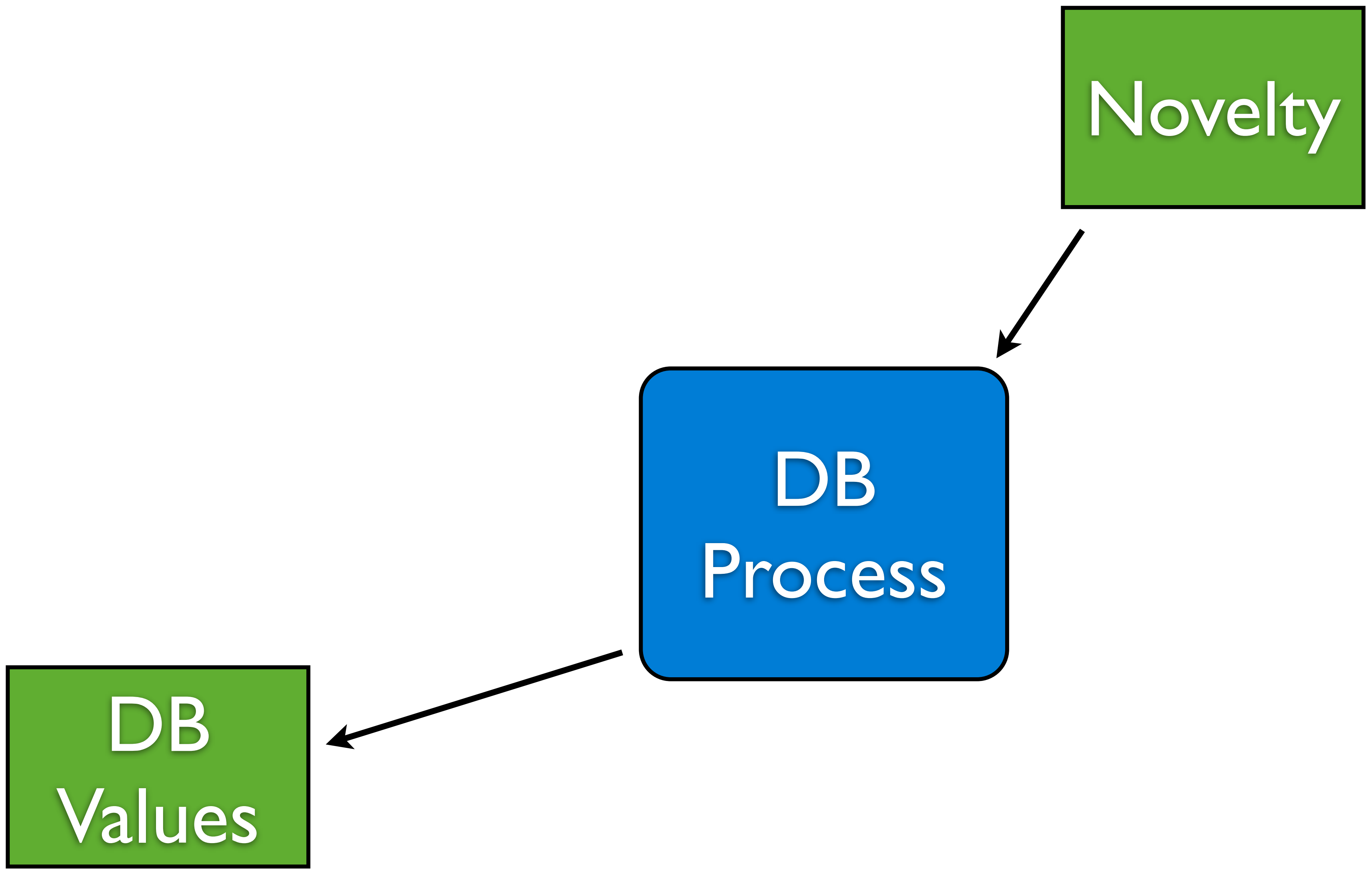


DB as Process

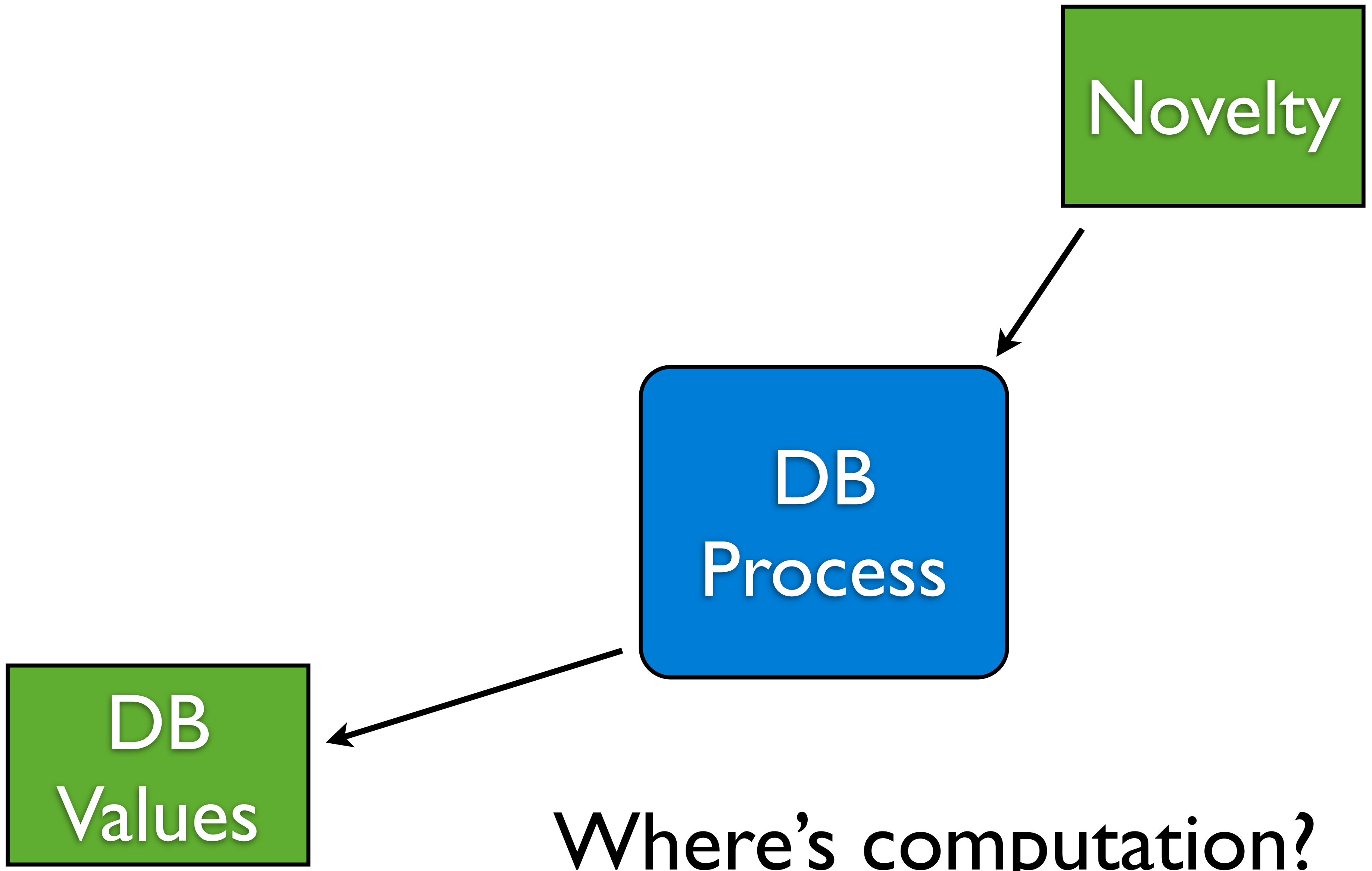


What's allowed?
Reproducible results?
How to use more than one db?

Functional DB Process

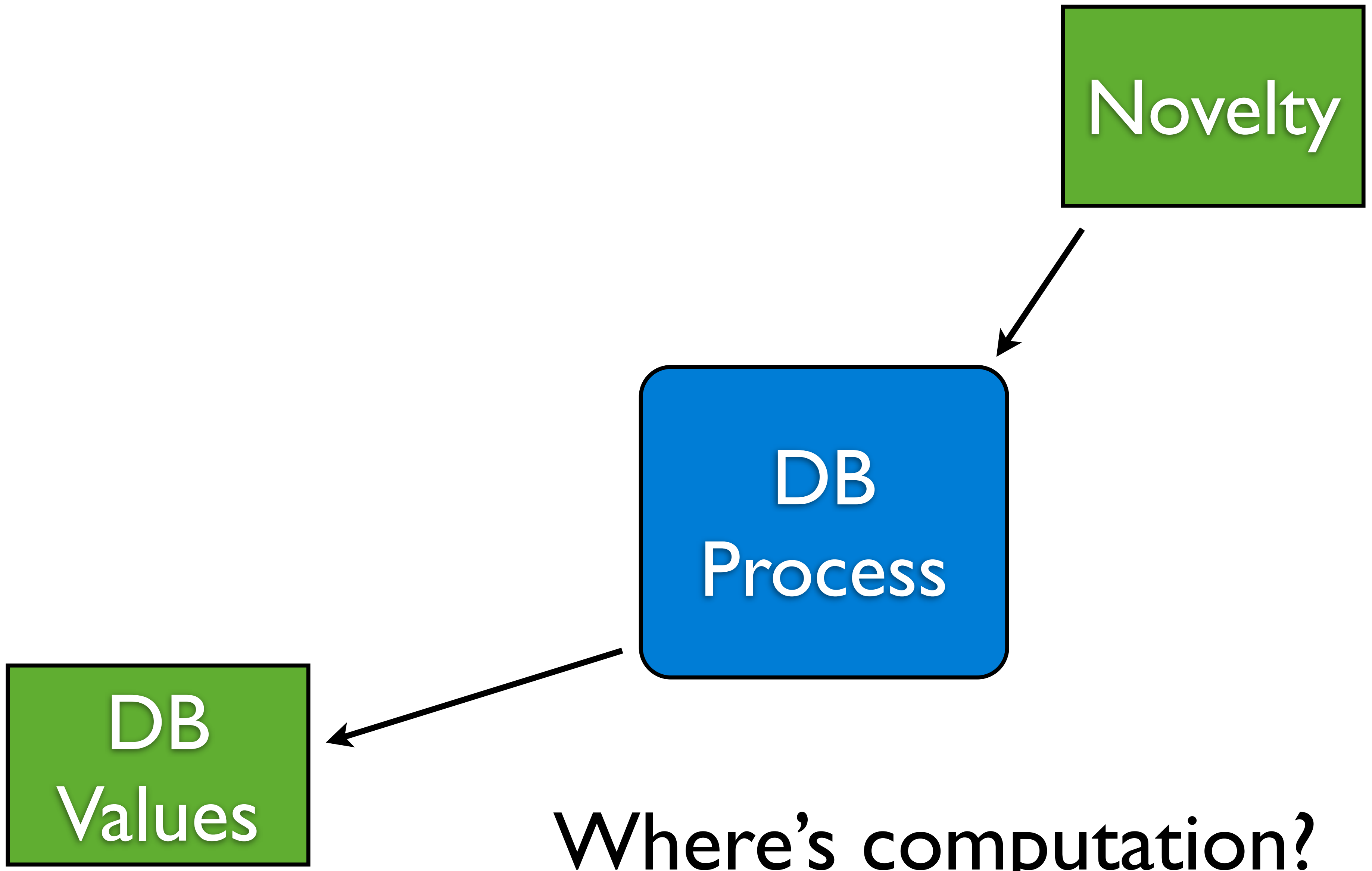


Functional DB Process



Where's computation?

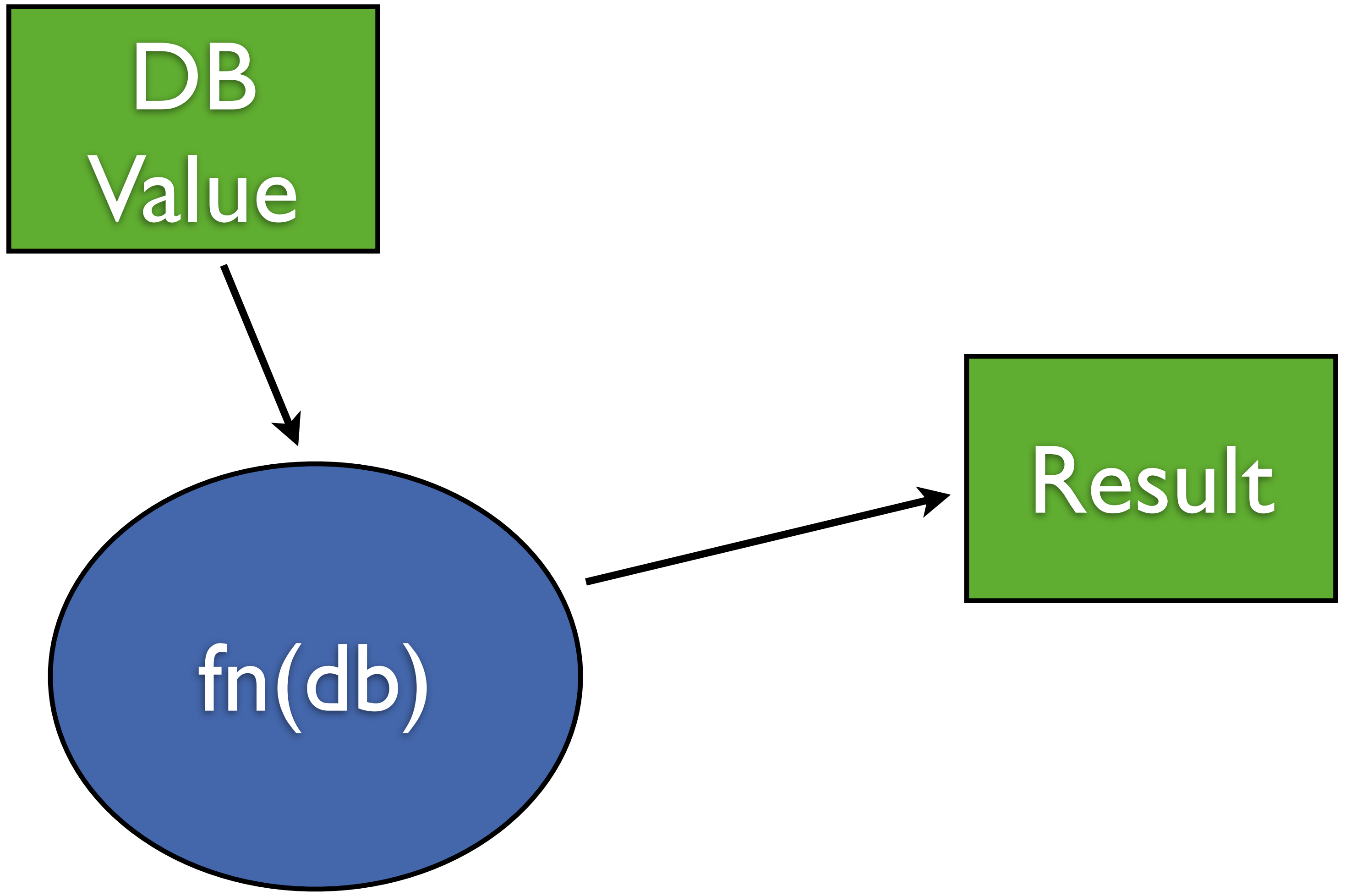
Functional DB Process



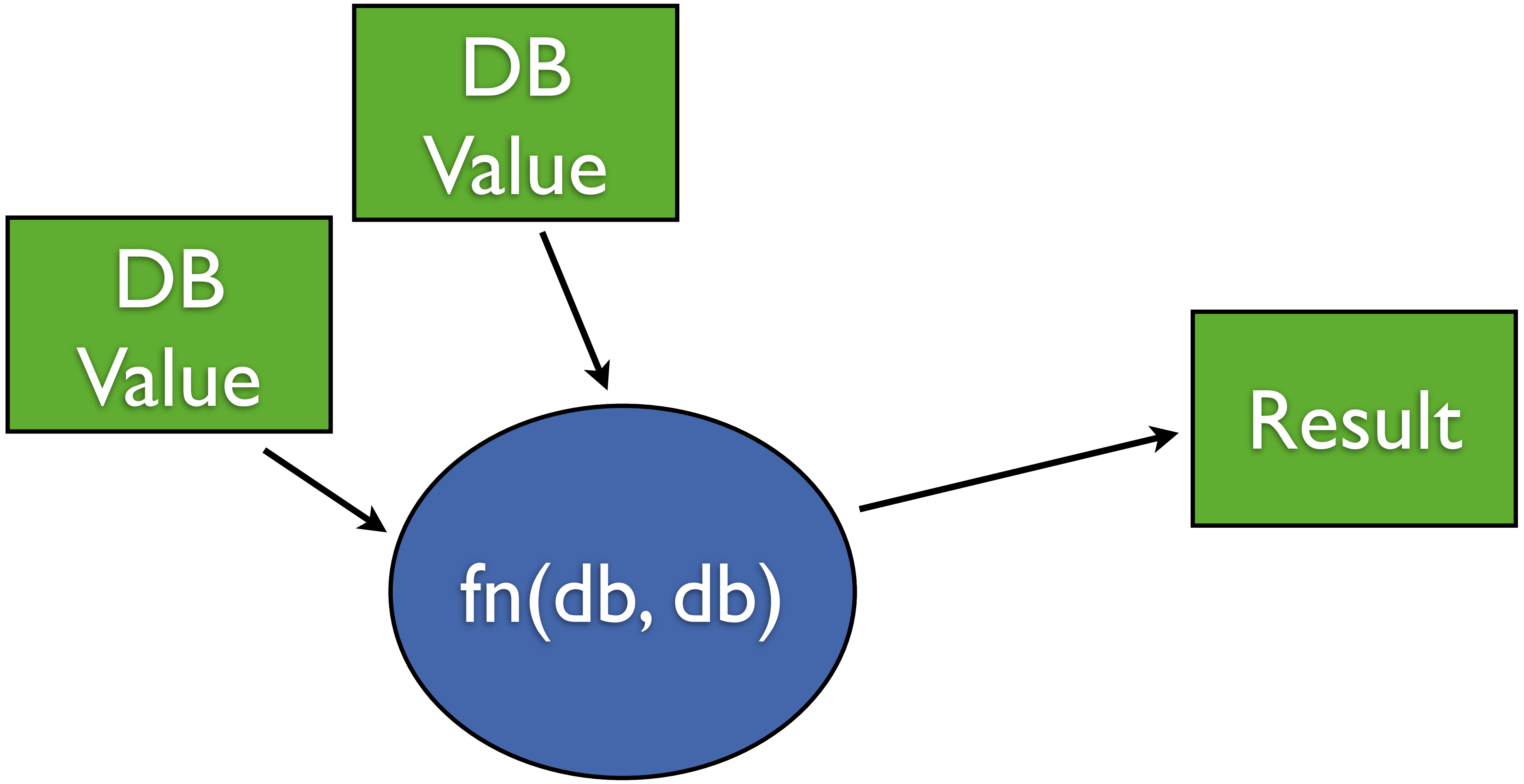
Where's computation?
Separate from process!

Functional DB Computation

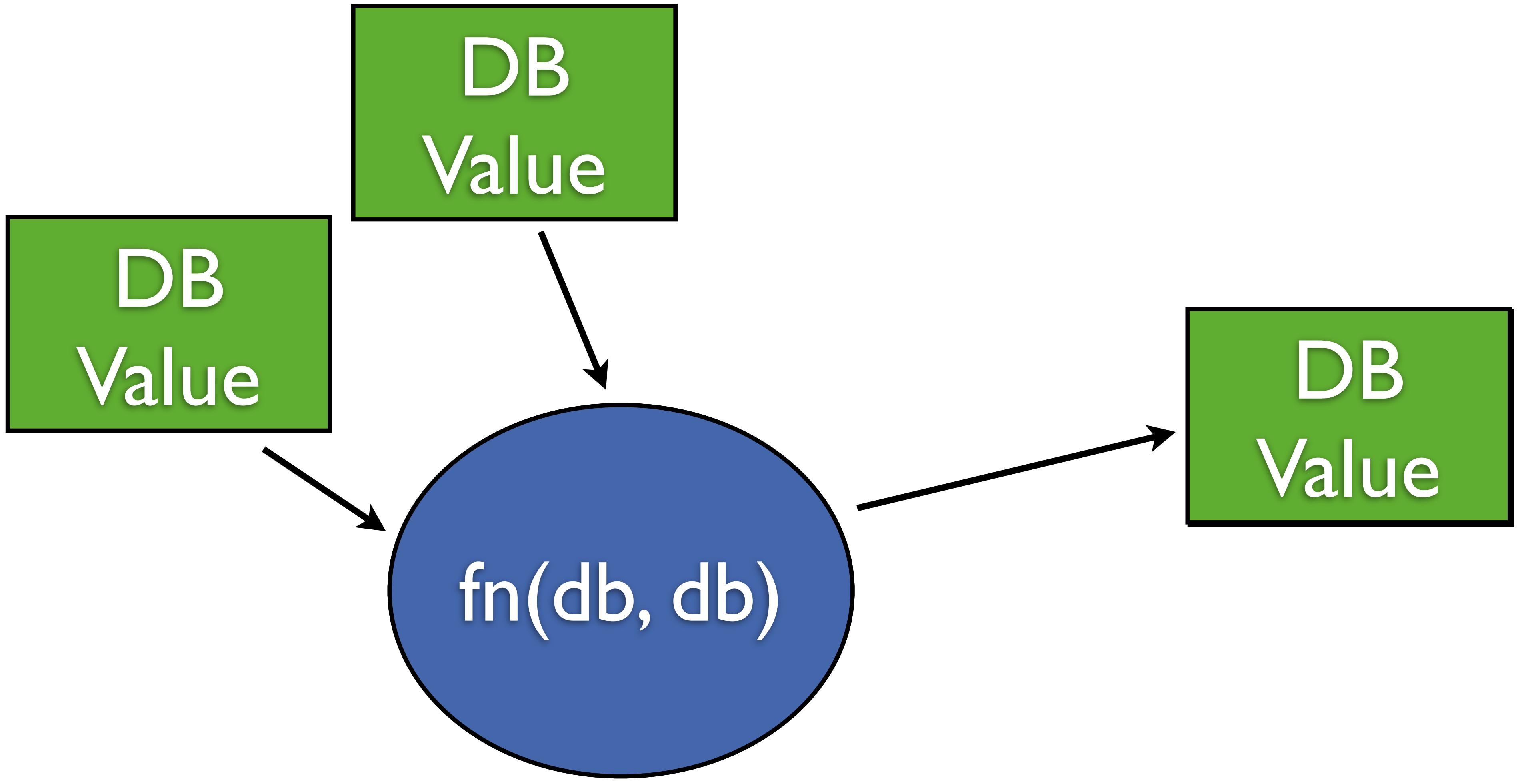
Functional DB Computation



Functional DB Computation



Functional DB Computation



Value Propositions

- Just data
  - language-independent
  - aggregate, compose
- Persistent data structures
  - alias freedom
  - efficient incremental ‘change’

One Structure, Many Functions

- Datalog queries
- Other query langs
- Direct index access
  - seek + scan
- Entity navigation

Speculation

- What-if scenarios
  - Just drop to backtrack
- Datomic's "with"

  dbval tx-data -> dbval
- Try before you buy/transact
- Tree propagation

Time Travel

- Accretive values contain all history
- Query as-of and/or since a point in time
- Query across time

Testing

- Flowing connections around, ugh

  ambient connection pool no different
- Reproducibility
- Values can easily be fabricated/generated

Stable Bases

```
//Peer
Database db = connection.db().asOf(1000);
Peer.q(aQuery, db);

//Client
GET /data/mem/test/1000/datoms?index=aevt
```

basis

- Same query, same results
- db permalinks!
  - communicable, recoverable
- Multiple conversations about same value

Datomic Datalog

- dbs are arguments to query, not implicit

```
q(query, db1, db2, otherInputs ...);

{:find [?customer ?product]
 :where [[?customer :shipAddress ?addr]
         [?addr :zip ?zip]
         [?product :product/weight ?weight]
         [?product :product/price ?price]
         [(Shipping/estimate ?zip ?weight) ?shipCost]
         [(<= ?price ?shipCost)]]}
```

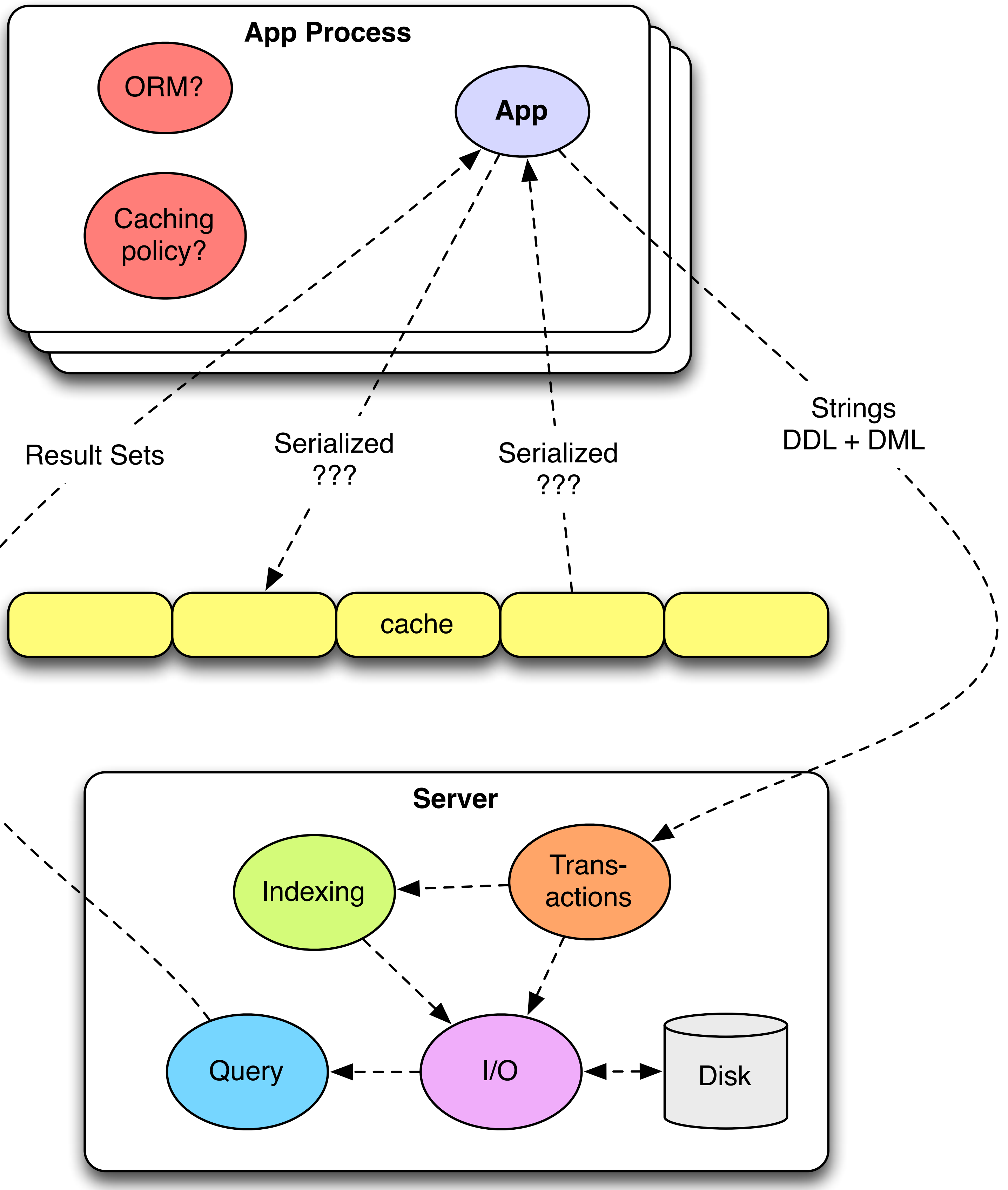
DB Values

- Time travel and more
  - `db.asOf` - past, `db.since` - windowed
  - `db.with(tx)` - speculative
  - `db.filter(pred)` - slice
- mock with datom-shaped data:

```
[[:fred :likes "Pizza"]
 [:sally :likes "Ice cream"]]
```

Implementation

Traditional Database



The Choices

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

Approach

- Move to information model
- Split process and perception
- Immutable basis in storage
- Novelty in memory

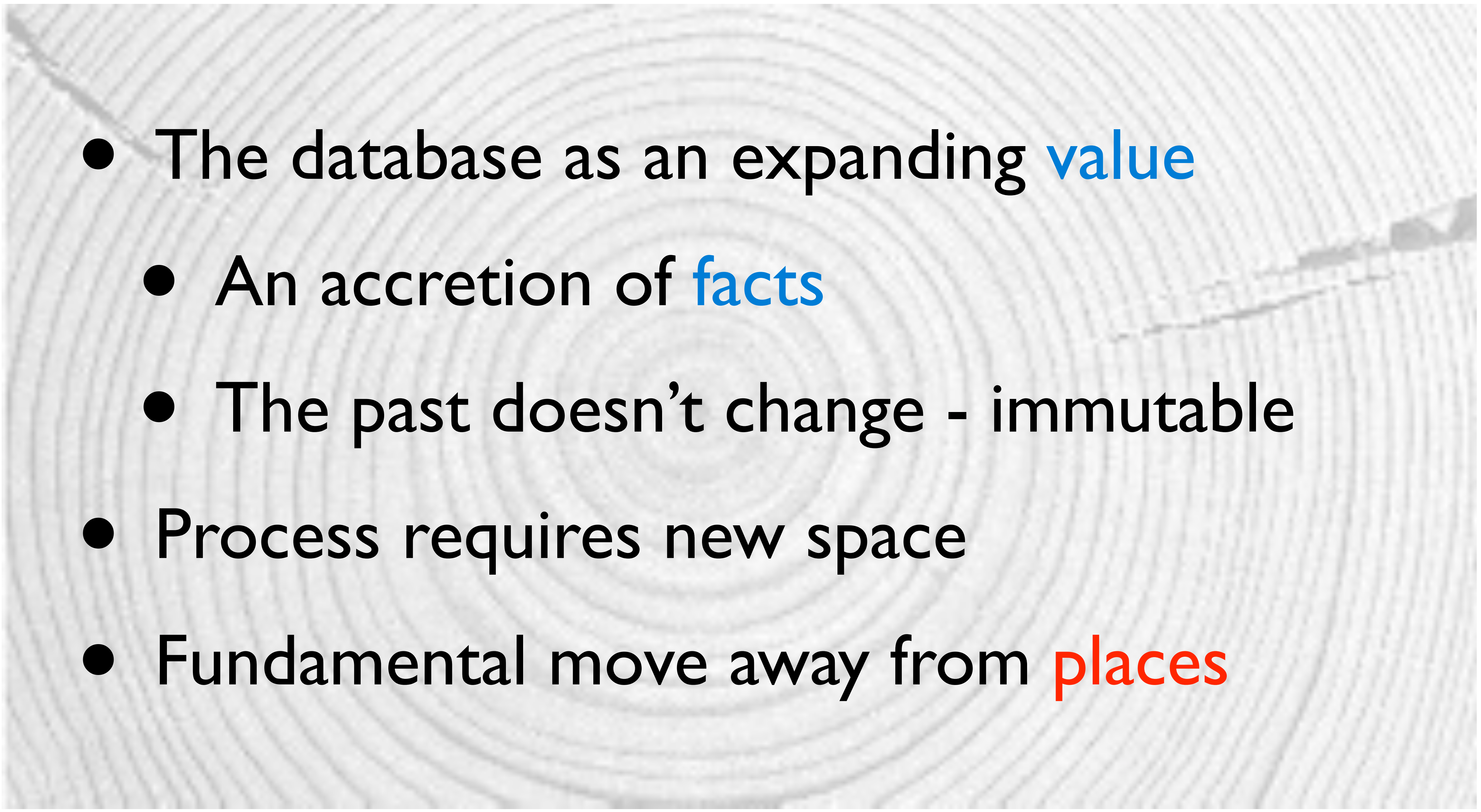
Information

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

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

Database State

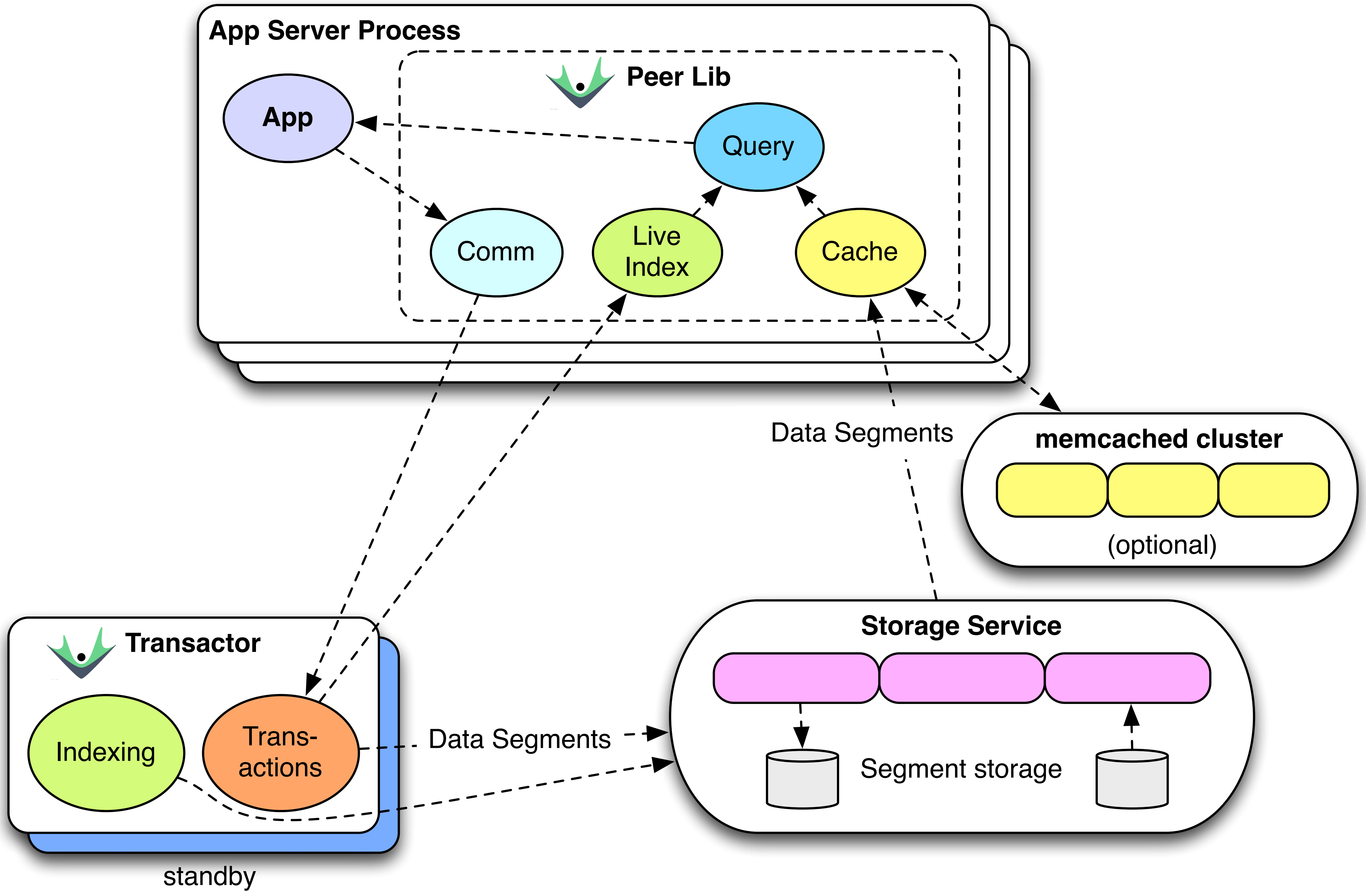


- The database as an expanding value
  - An accretion of facts
  - The past doesn't change - immutable
- Process requires new space
- Fundamental move away from places

Accretion

- Root per transaction doesn’t work
- Latest values include past as well
  - The past is sub-range
- Important for information model

Datomic Architecture

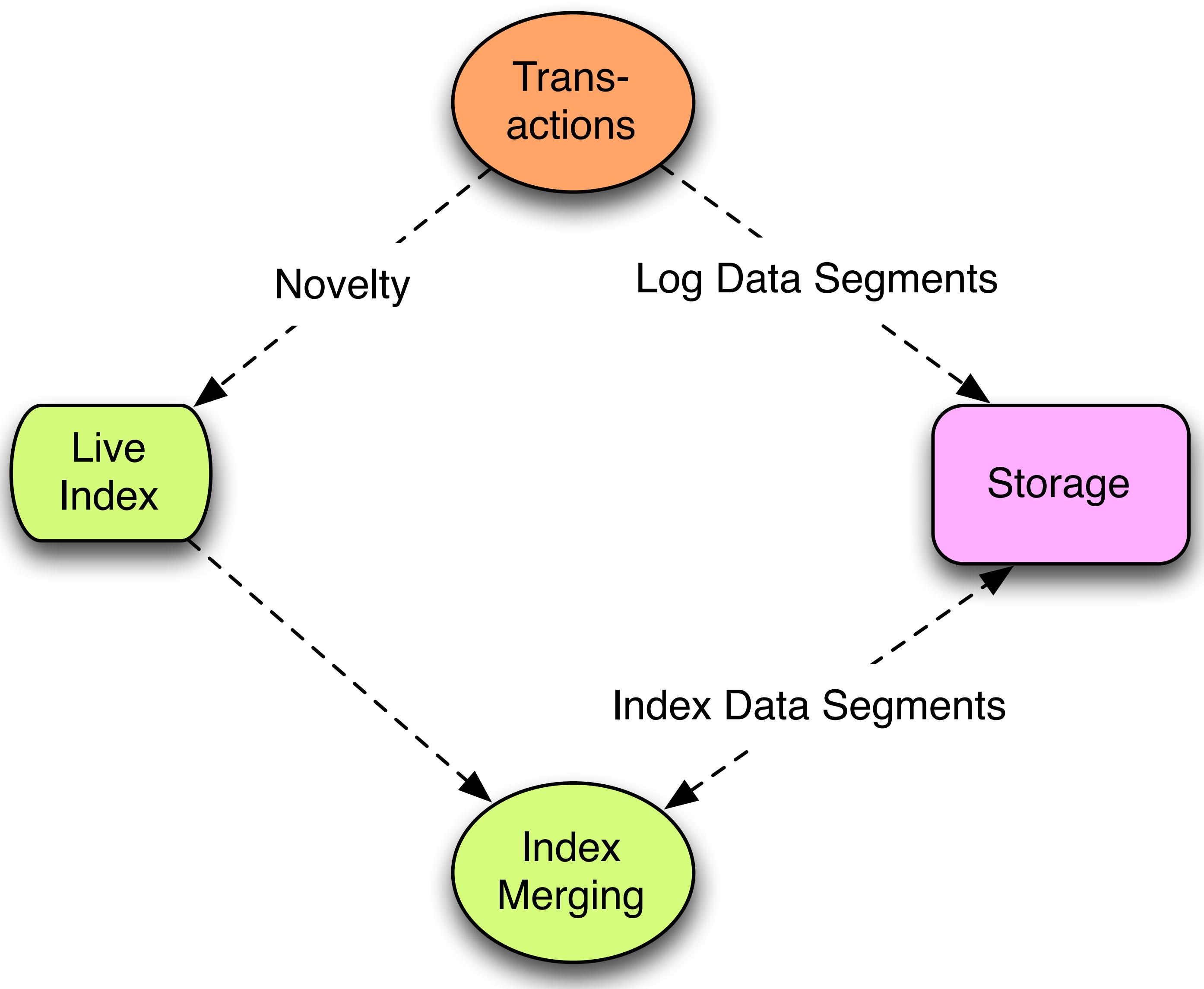


Indexing

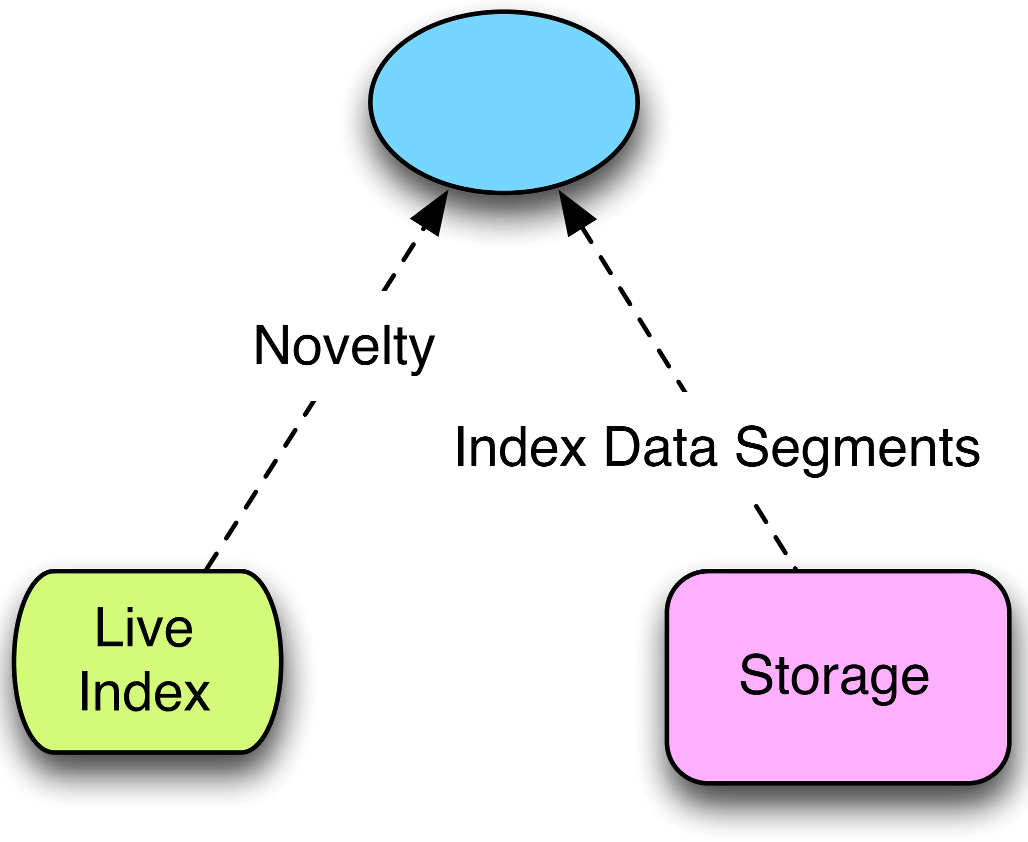
- Maintaining sort live in storage - bad
- BigTable et al:
  - Accumulate novelty in memory
  - Current view: mem + storage merge
  - Occasional integrate mem into storage

    Releases memory

Transactions and Indexing



Perception



Process

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

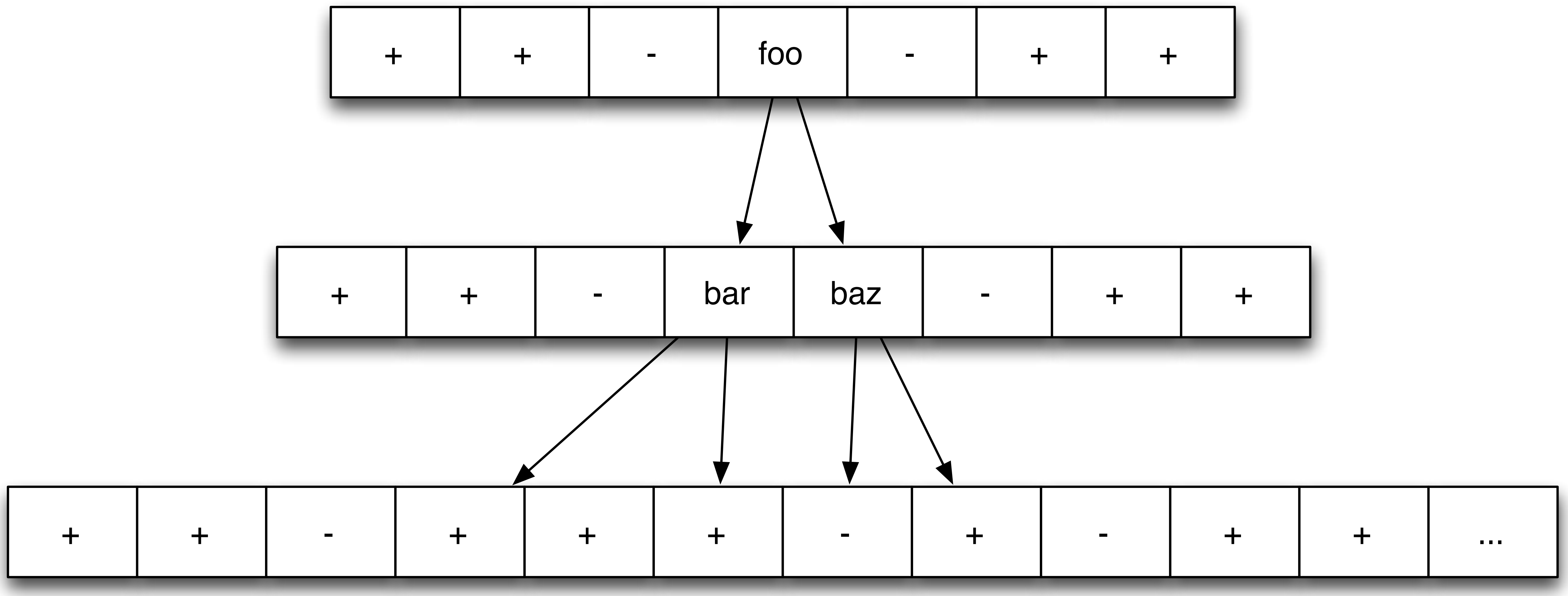
Process

- 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



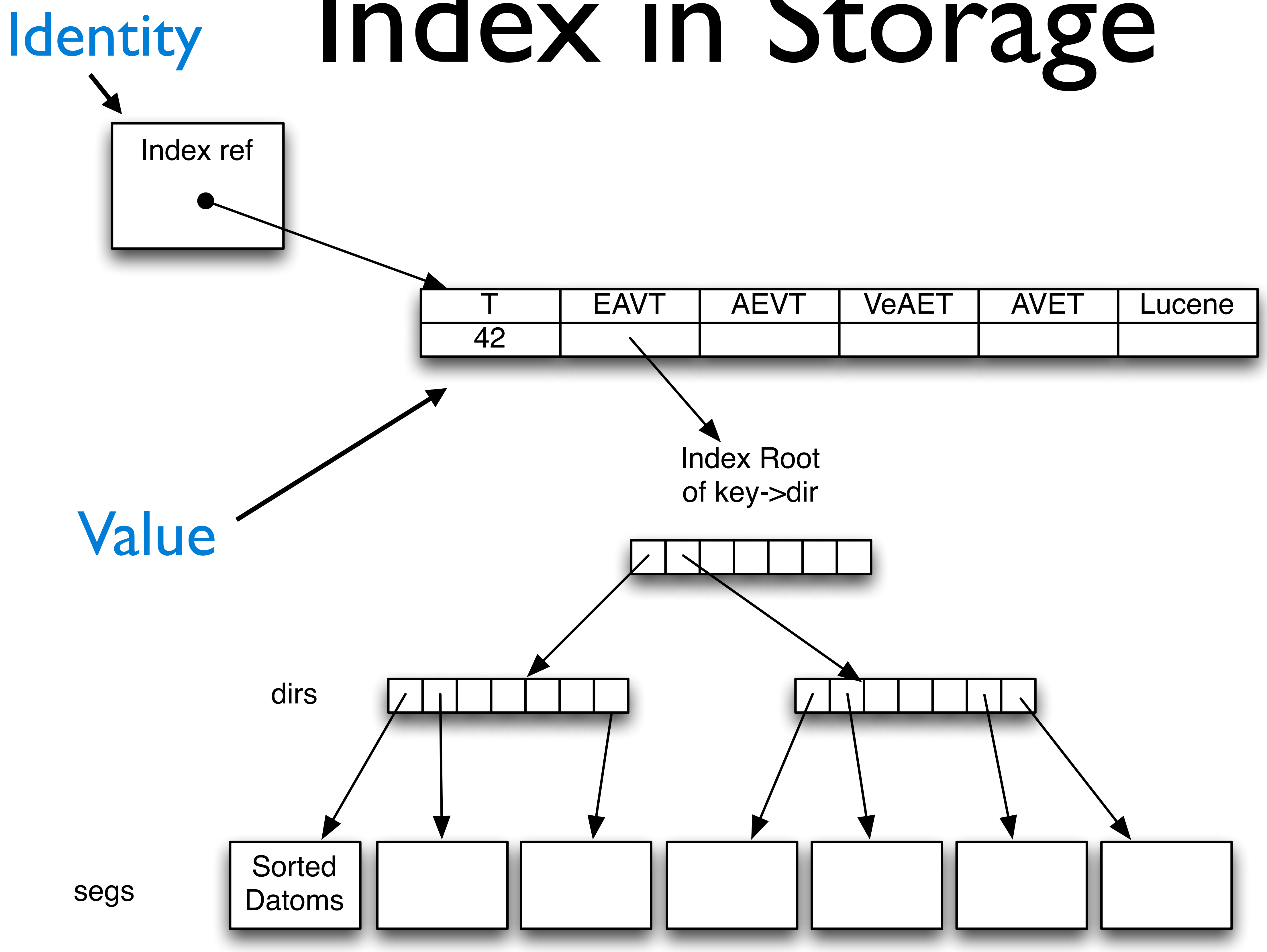
Memory Index

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

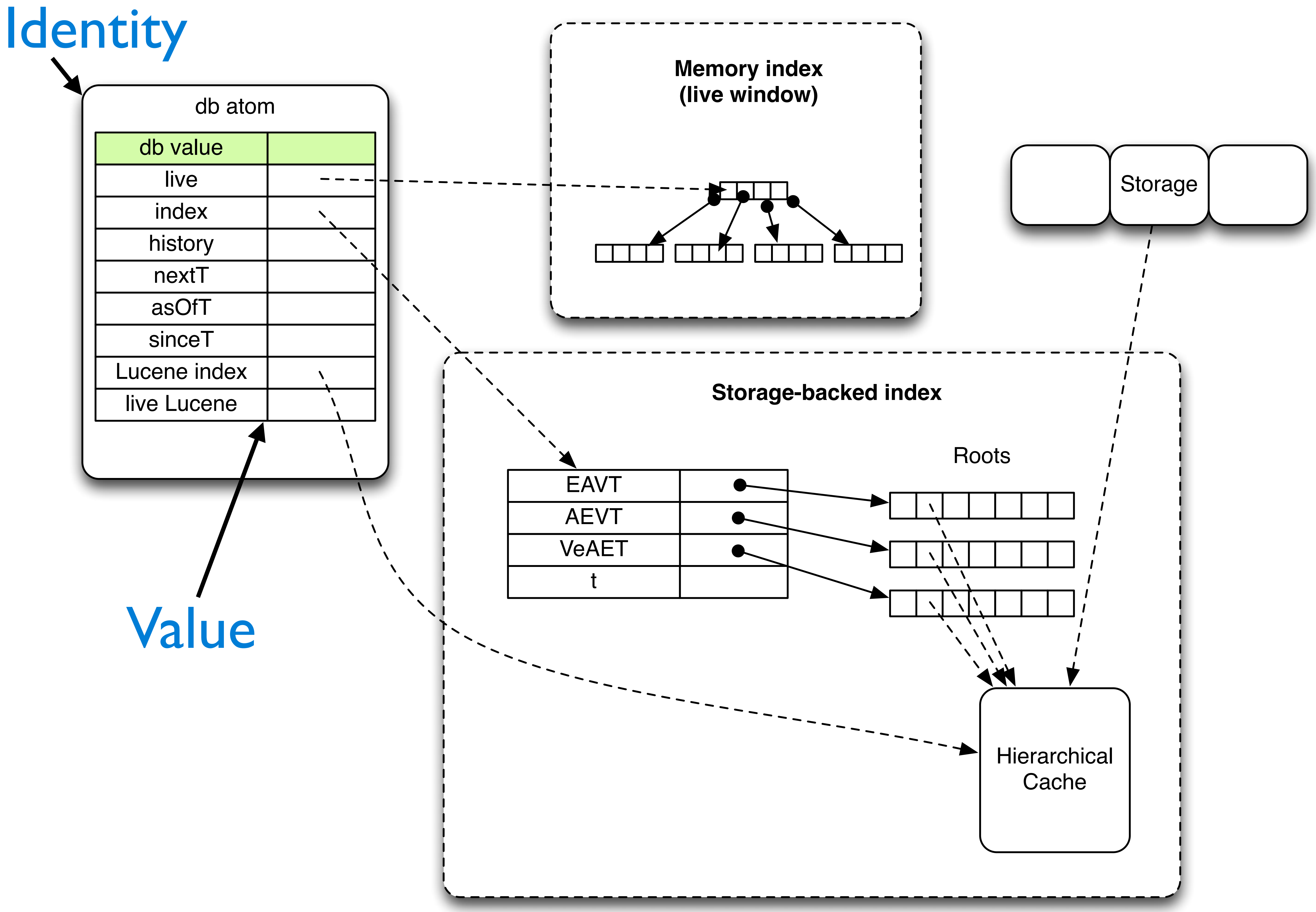
Storage

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

Index in Storage



What's in a DB Value?



Functional DB Benefits

- Epochal state
  - Coordination only for process
- Transactions well defined
  - Functional accretion
- Freedom to relocate/scale storage, query
- Extensive caching
- Process events

Thanks for Listening!