UNIVERSITÄT LEIPZIG

LIFE Leipziger Forschungszentrum für Zivilisationserkrankungen

Datomic
A Functional Database

Alexander Kiel
alexanderkiel@gmx.net

Universität Leipzig

Motivation

- Why a talk about a database?
- Databases important aspect of most real-world software projects
- Is there a functional database?

Datomic

- Commercial product of Clojure founder Rich Hickey and Relevance, Inc.
- Started 2010, first public version 2012
- Active development: 4 new versions/month
- Free Edition suitable for small projects

Database Landscape

Relational Databases

- ACID
- Schema
- Queries
- Joins
- not scalable

Datomic

- ACID
- no Schema
- Queries
- Joins
- read scalable

Key-Value Stores

- no TX
- no Schema
- no Queries
- no Joins
- scalable

Problems to Solve

- Missing concept of Time
- In-Place Updates
- Monolitic Architecture

Missing Concept of Time

- Databases offer only NOW
- Data will be modified w/o keeping track
- No way to go back in time
- No snapshots

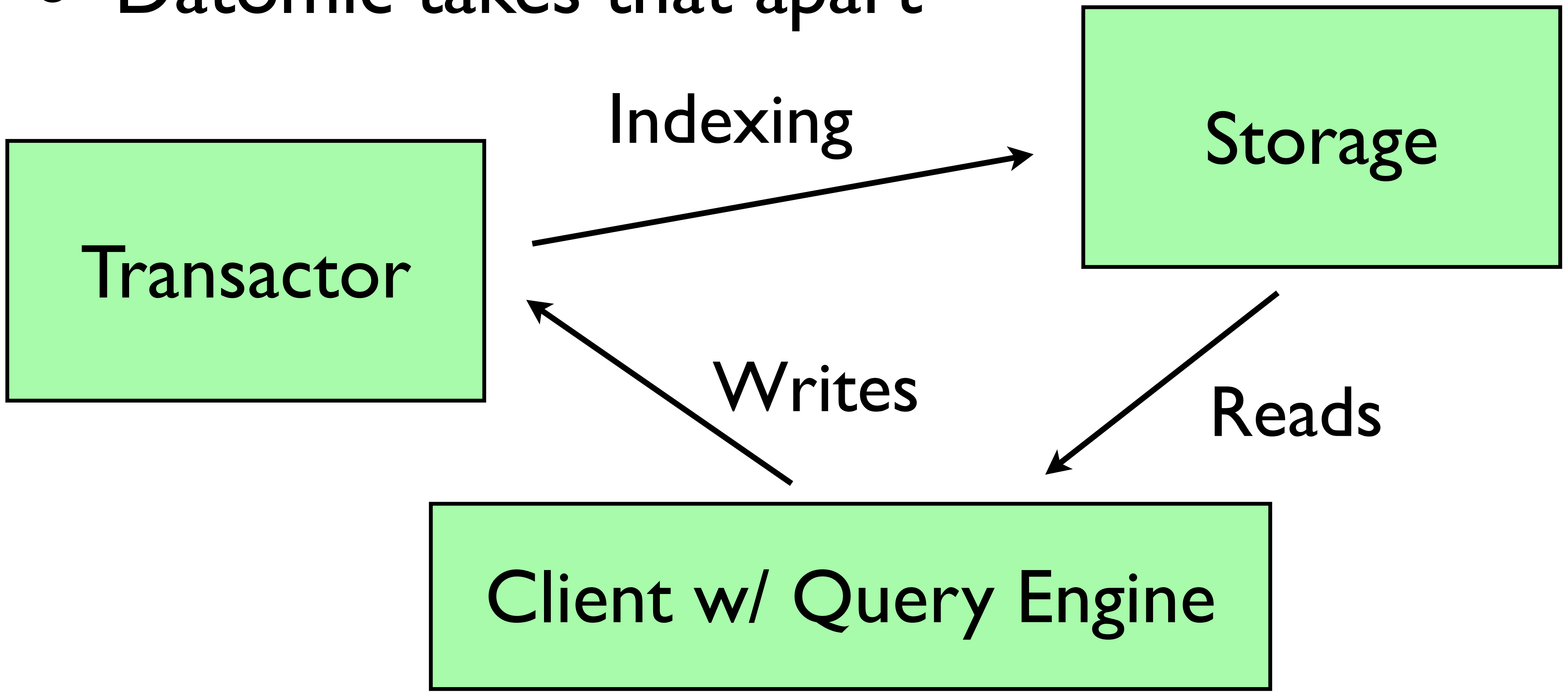
In-Place Updates

ID	Col A	Col B
1	foo	42
2	bar	23
3	baz	87

- Mutable Datastructure
- Need to coordinate reads and writes
- Motivation was: scarce resources

Monolitic Architecture

- One central Database Server which does Transactions/Storage/Queries
- Datomic takes that apart



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

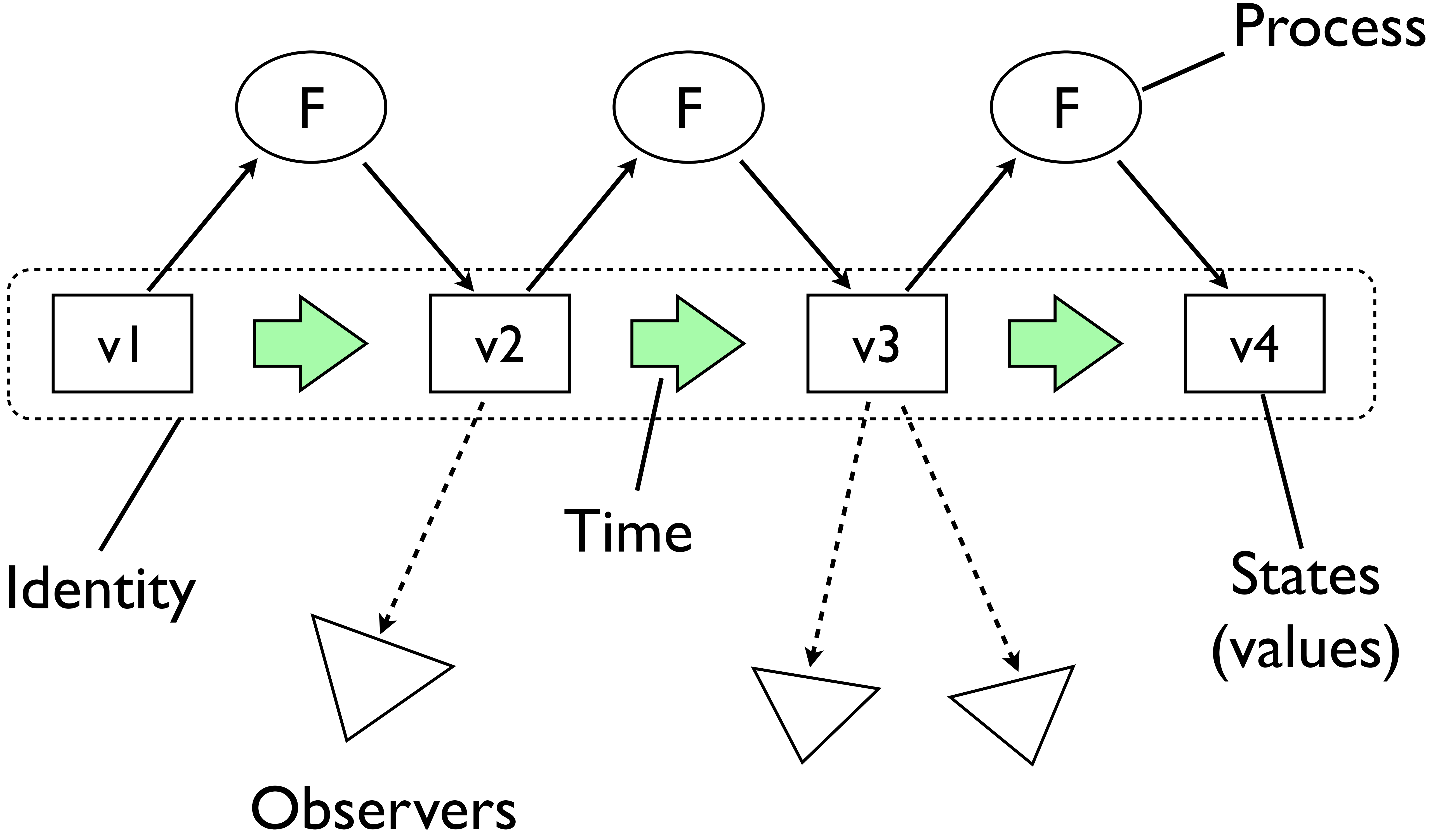
State

Value of an identity at a moment in time

Time

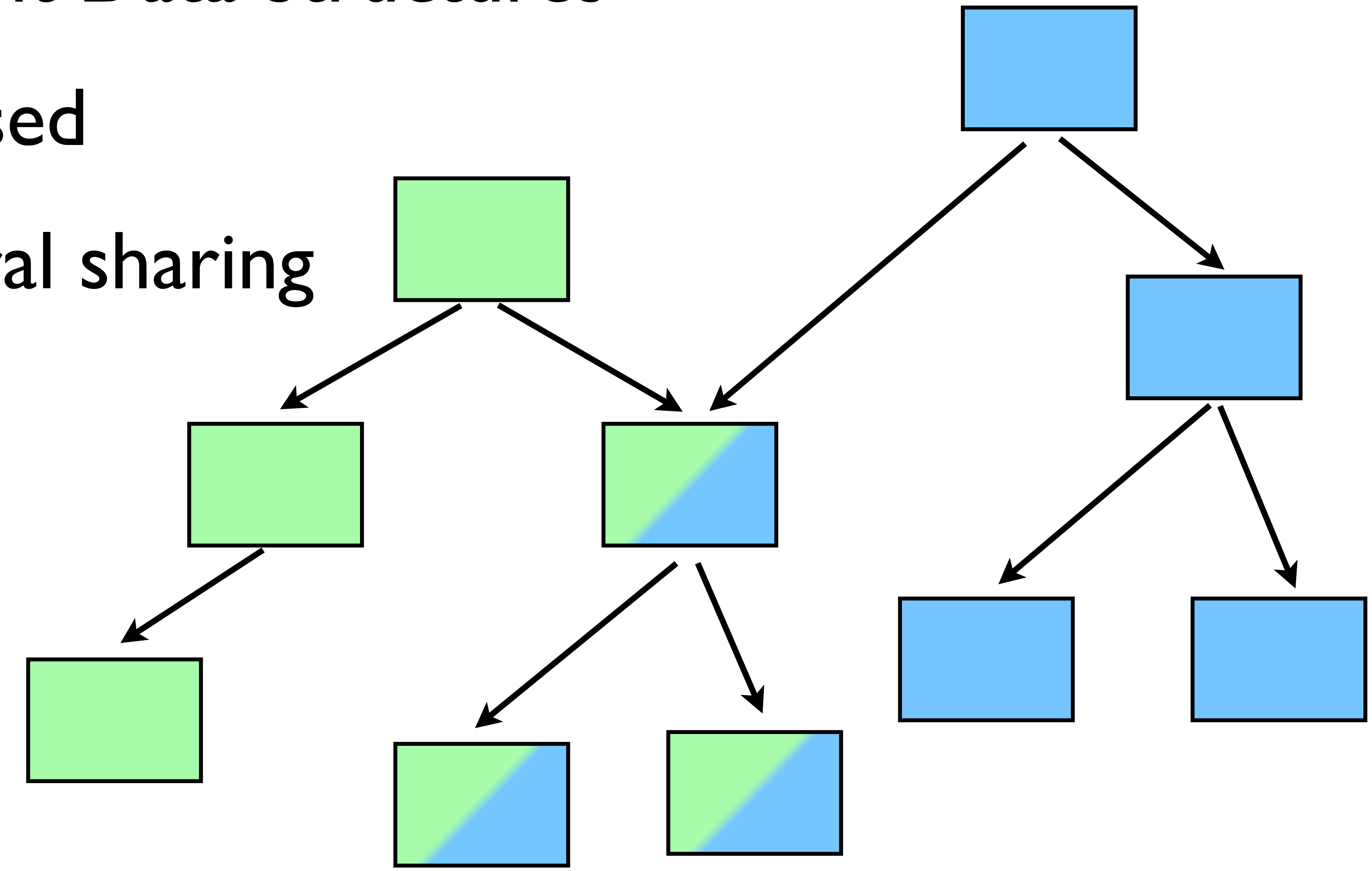
Relative before/after ordering of causal values

Epochal Time Model



Value Implementation

- Persistent Data Structures
- Tree based
- Structural sharing

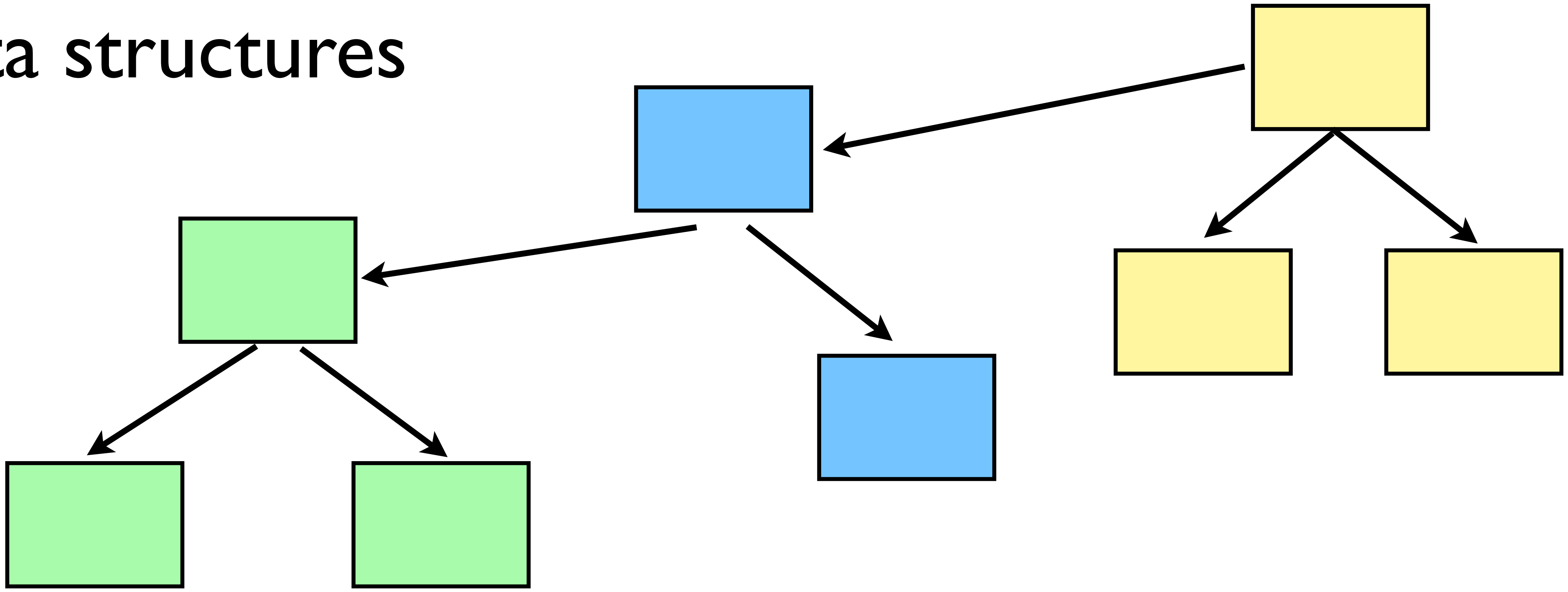


Database State

- The database is an ever expanding value
- An accretion of facts
- The past doesn't change
- Each process needs new space.
- Not place-oriented

Accretion

- Value has to reference all past values
- Latest value contains always the whole history
- Nothing is GC'ed unlike typical persistent data structures



Information Model

- Data stored in form of facts
- Fact: atomic information item
- Entity/Attribute/Value/Transaction (Time)

Entity	Attribute	Value	Time
Sally	likes	pizza	02.05.2013

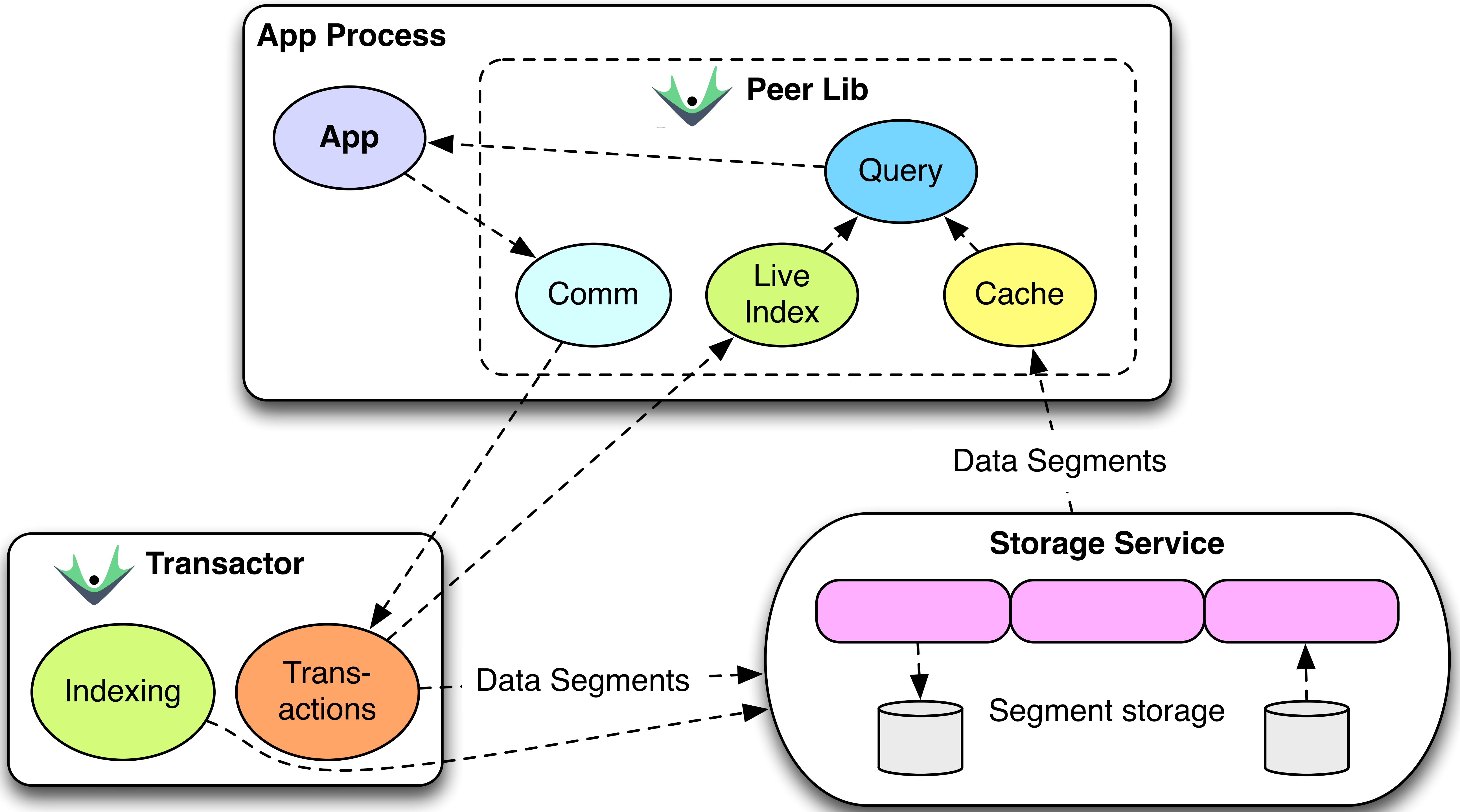
Process/Transactions

- Primitive representation of novelty
- Assertions and retractions of facts
- Reified transactions
  - Facts reference transactions
  - Transactions are entities
  - Fully queryable

Indexing

- Query engines need sorted facts
- Sorted sets of facts are kept in memory
- Periodic merges into storage
- Every client (peer) has its own memory index

Architecture



Schema

- Consists of attribute definitions

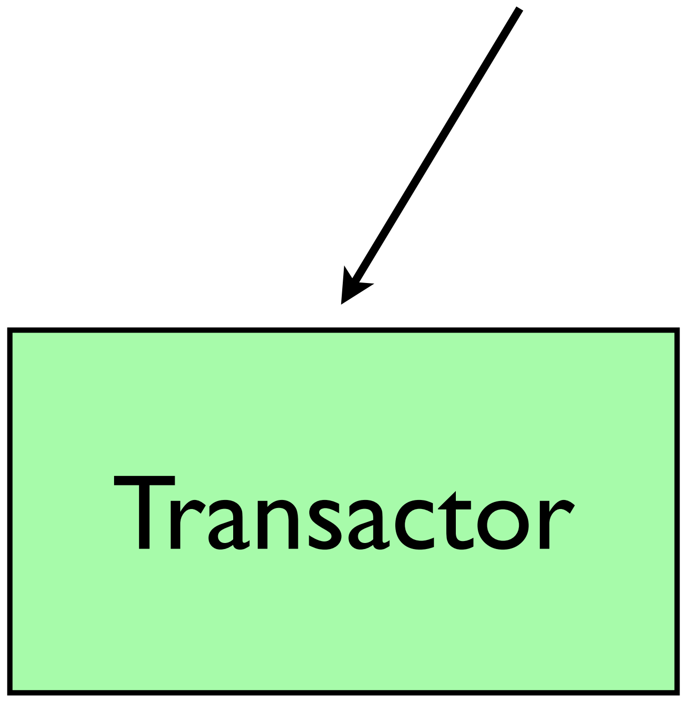
```
{:db/ident          :person/name,
 :db/valueType      :db.type/string,
 :db/cardinality :db.cardinality/one,
 :db/doc            "A person's name"}
```

- Other attributes of attributes:
  - :db/unique
  - :db/index
  - :db/fulltext

Transactions

- Lists of assertions and retractions

```
[[:db/add entity-id attr value]
 [:db/retract entity-id attr value]...]
```



- Expands transactions
- Serializes all transactions
- Creates new DB state after each transaction
- Sends novelity to all peers

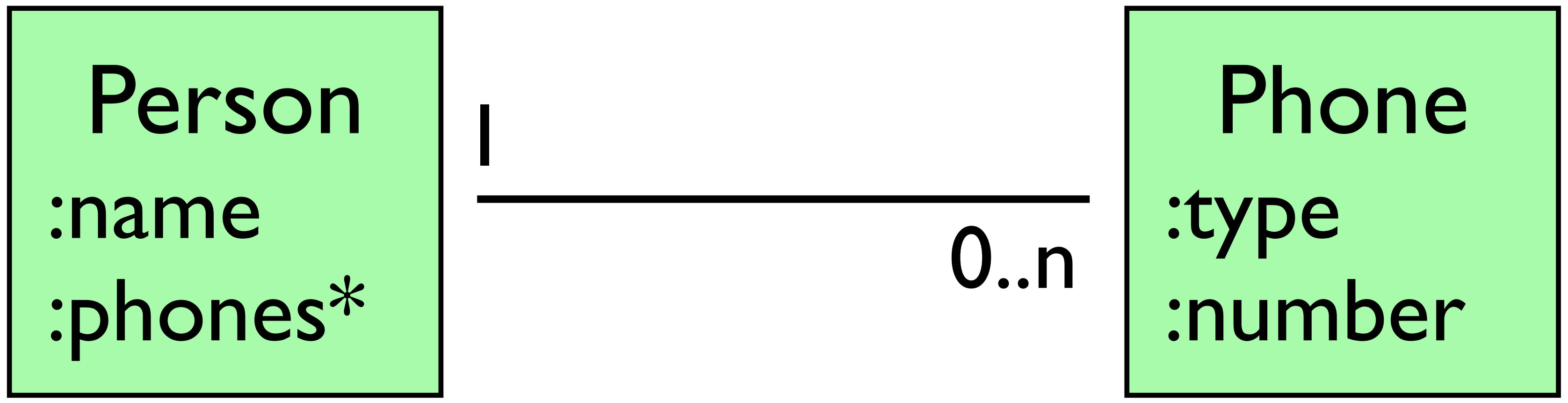
Query Language

- Datalog as default query language
- Queries take a database value as input

```
[:find variables... :where clauses...]

(q '[:find ?p
      :where [?p :name "Sally"]]
 db-value)
```

Query - Joins



- All persons which have a work phone

```
 [:find ?person
  :where [?person :phones ?phone]
         [?phone :type "work"]]
```

```
SELECT DISTINCT persion.id FROM person
JOIN phone ON phone.person_fk = person.id
WHERE phone.type = "work"
```

Thanks