Datomic

A Day of Datomic

cognitect

@stuarthalloway

agenda

what is Datomic?

information model

transaction model

query model

time model

operational model

what is Datomic?

a complete rethink of databases

agile

robust

powerful

time-aware

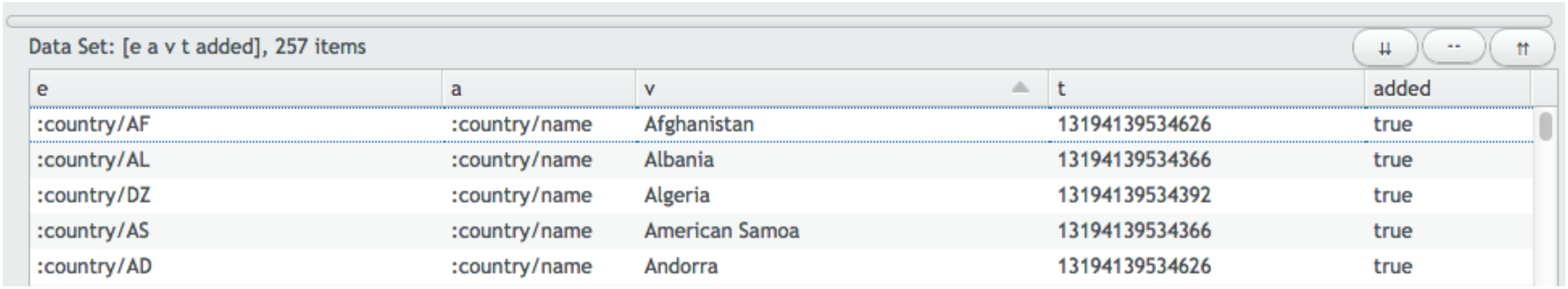
cloud

agile: universal schema

entity / attribute / value / tx / op

granular, attribute-level schema definition

- easy to model
- easy to migrate



Data Set: [e a v t added], 257 items

e	a	v	t	added
:country/AF	:country/name	Afghanistan	13194139534626	true
:country/AL	:country/name	Albania	13194139534366	true
:country/DZ	:country/name	Algeria	13194139534392	true
:country/AS	:country/name	American Samoa	13194139534366	true
:country/AD	:country/name	Andorra	13194139534626	true

update in place

sharing	difficult
distribution	difficult
concurrent access	difficult
access pattern	eager
caching	difficult
examples	Java and .NET collections relational databases NoSQL databases

NOT ROBUST

persistent data structures

sharing	trivial
distribution	easy
concurrent access	trivial
access pattern	eager or lazy
caching	easy
examples	Clojure, F# collections Datomic database

ROBUST

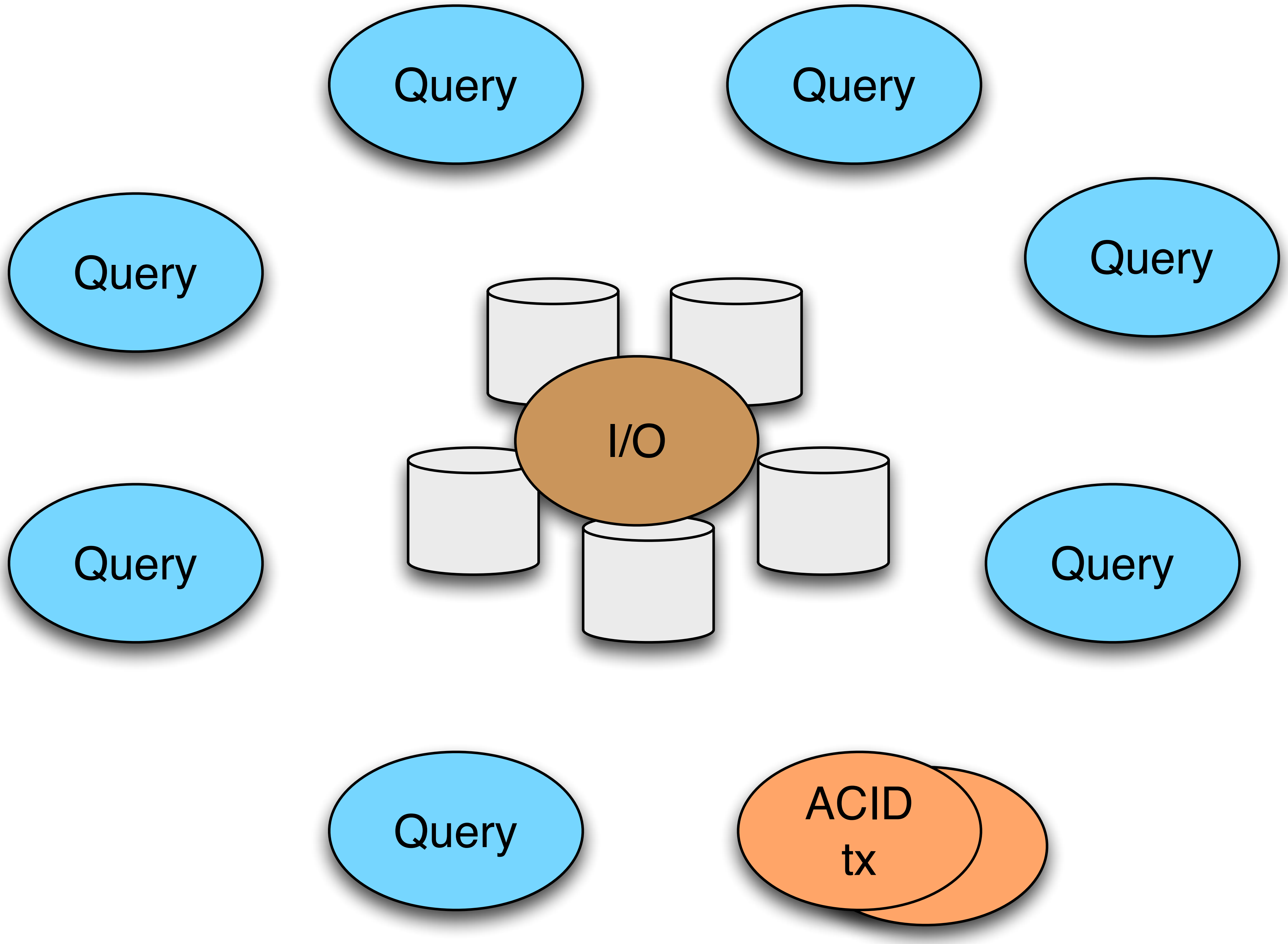
powerful

- universal schema supports many access styles
  - row
  - column, key-value, document, and graph
- declarative
  - datalog query
  - pull

time aware

db view	semantics	supports
(default)	current state	what is the current situation?
.asOf	state at point in past	how were things in the past?
.since	state since point in past	how have things changed?
tx report	before / after / change view of tx	automated event response
.with	state with proposed additions	what would happen if we did X?
.history	timeless view of all history	anything!

a database of little services



lab: running the tools

install Datomic, start dev transactor	http://docs.datomic.com/getting-started.html
install mbrainz dataset to datomic:dev://localhost:4334/mbrainz	https://github.com/Datomic/mbrainz-sample
explore mbrainz from console	http://docs.datomic.com/console.html
explore mbrainz from REST service	http://docs.datomic.com/rest.html
install examples	https://github.com/datomic/day-of-datomic
install your favorite REPL (links show Cursive)	https://www.jetbrains.com/idea/download/ + https://cursiveclojure.com/userguide/
play through CRUD example	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/crud.clj

information model

notation

datoms

databases

entities

schema

notation: edn

```
{ :firstName "John"
  :lastName "Smith"
  :age 25
  :address {
    :streetAddress "21 2nd Street"
    :city "New York"
    :state "NY"
    :postalCode "10021" }
  :phoneNumber
    [ {:type "name" :number "212 555-1234"}
      {:type "fax" :number "646 555-4567" } ] }
```

http://edn-format.org

type	examples
string	`"foo"`
character	`\f`
integer	`42, 42N`
floating point	`3.14, 3.14M`
boolean	`true`
nil	`nil`
symbol	`foo, +`
keyword	`:foo, ::foo`

type	properties	examples
list	sequential	`(1 2 3)`
vector	sequential and random access	`[1 2 3]`
map	associative	`{:a 100 :b 90}`
set	membership	`#{:a :b}`

generic extensbility

`#name edn-form`

name describes interpretation of following element

recursively defined

all data can be literal

built-in tags

```
#inst "rfc-3339-format"
```

tagged element is a string in RFC-3339 format

```
#uuid "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
```

tagged element is a canonical UUID string

datoms

granular, atomic facts

immutable

5-tuple: entity / attribute / value / transaction / op

example datoms

e	a	v	tx	op
jane	likes	broccoli	1008	true
jane	likes	pizza	1008	true
jane	likes	pizza	1148	false

datom API

```java
public interface Datom {
    Object e();
    Object a();
    Object v();
    Object tx();
    boolean added();

    // positional
    Object get(int index);
}
```

database

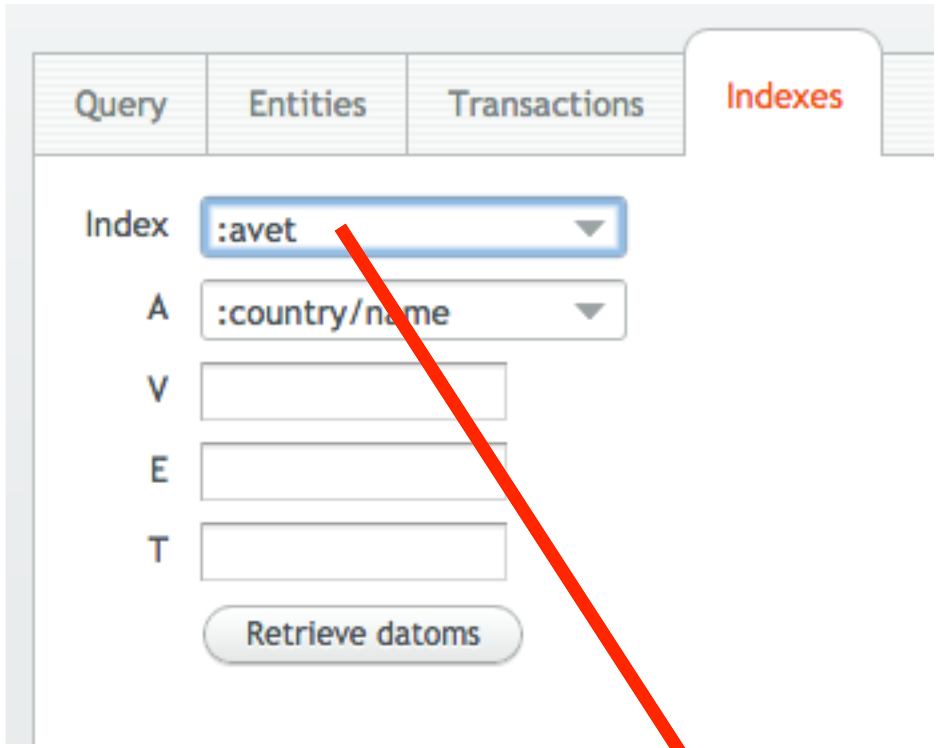
a set of datoms (universal relation)

efficient storage

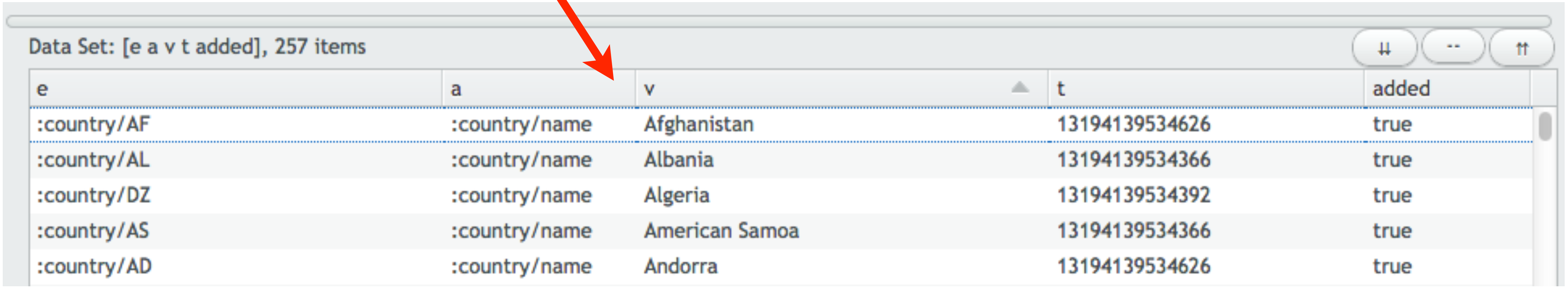
many sort orders

accumulate-only (not append-only)

database sorts in console



Index :avet
A :country/name
V
E
T



Data Set: [e a v t added], 257 items

e	a	v	t	added
:country/AF	:country/name	Afghanistan	13194139534626	true
:country/AL	:country/name	Albania	13194139534366	true
:country/DZ	:country/name	Algeria	13194139534392	true
:country/AS	:country/name	American Samoa	13194139534366	true
:country/AD	:country/name	Andorra	13194139534626	true

entities

immutable

lazy

associative

inferred from datoms sharing a common **e**

point-in-time

bidirectional navigation

http://docs.datomic.com/entities.html

example entity

entity

```
{:db/id jane
 :likes "broccoli"}
```

↕

datoms

e	a	v	tx	op
jane	likes	broccoli	1008	true
jane	likes	pizza	1008	true
jane	likes	pizza	1148	false

entity API

```
interface Database
{
    Entity entity(Object entityId);
    // unrelated methods elided for brevity
}

interface Entity {
    Object get(Object key);
    Entity touch();
    Set keySet();
}
```

relation direction

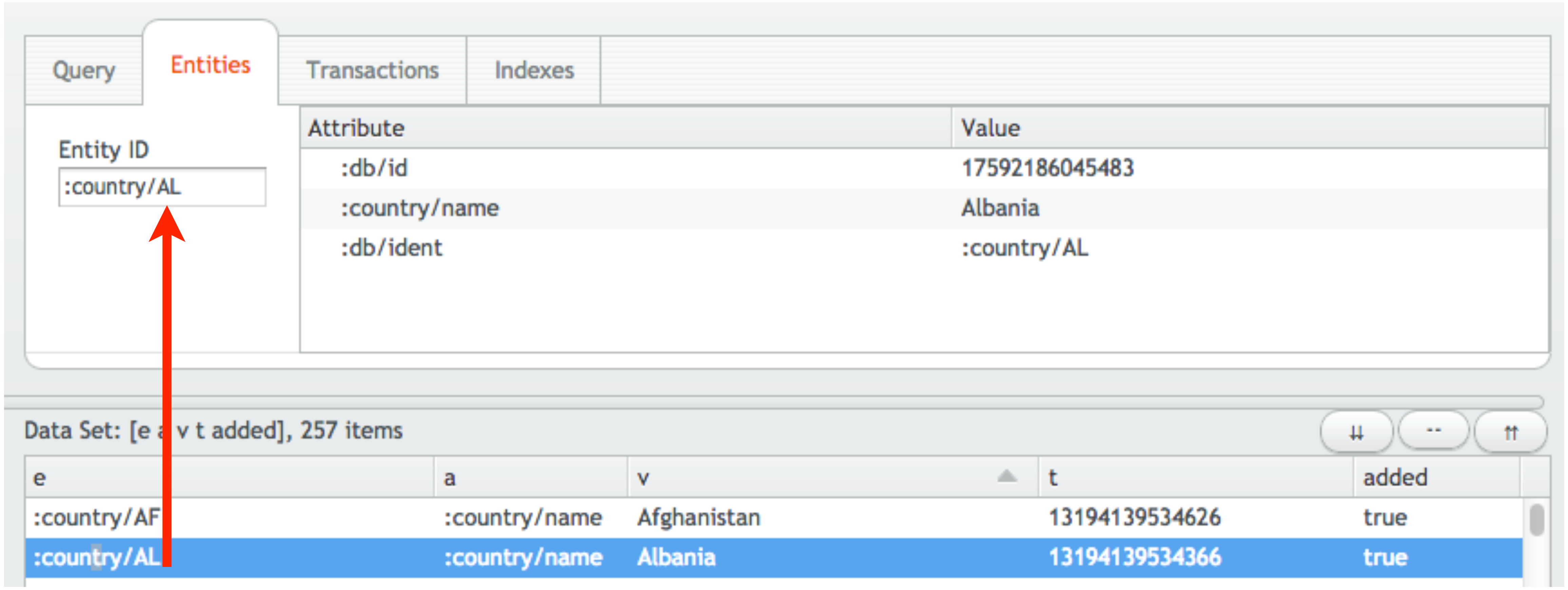
```
// things that Jane likes
jane.get("likes");
```

reversing direction

```
// people that like broccoli
broccoli.get("_likes");
```

underscore means
“nav backwards”

entities in console



Attribute	Value
:db/id	17592186045483
:country/name	Albania
:db/ident	:country/AL

Data Set: [e a v t added], 257 items

e	a	v	t	added
:country/AF	:country/name	Afghanistan	13194139534626	true
:country/AL	:country/name	Albania	13194139534366	true

click any e

schema

schemas *add power*

schema is plain data

schema elements installed via transactions

make history-compatible changes at any time

http://docs.datomic.com/schema.html

common schema attributes

attribute	type	use
db/ident	keyword	programmatic name
db/valueType	ref	attribute type
db/cardinality	ref	one- or many- valued?
db/index	ref	creates AVET
db/unique	ref	unique, “upsert”
db/isComponent	ref	ownership

stories

attribute	type	cardinality
story/title	string	1
story/url	string	1
story/slug	string	1
news/comments	ref	many

schema is plain old data

attribute	type	card
story/title	string	1
story/url	string	1
story/slug	string	1
news/comments	ref	many

```
{:db/id #db/id[:db.part/db]
 :db/ident :story/url
 :db/valueType :db.type/string
 :db/cardinality :db.cardinality/one
 :db.install/_attribute :db.part/db}
```

users

attribute	type	cardinality
user/firstName	string	1
user/lastName	string	1
user/email*	string	1
user/upVotes	ref	many

*unique

comments

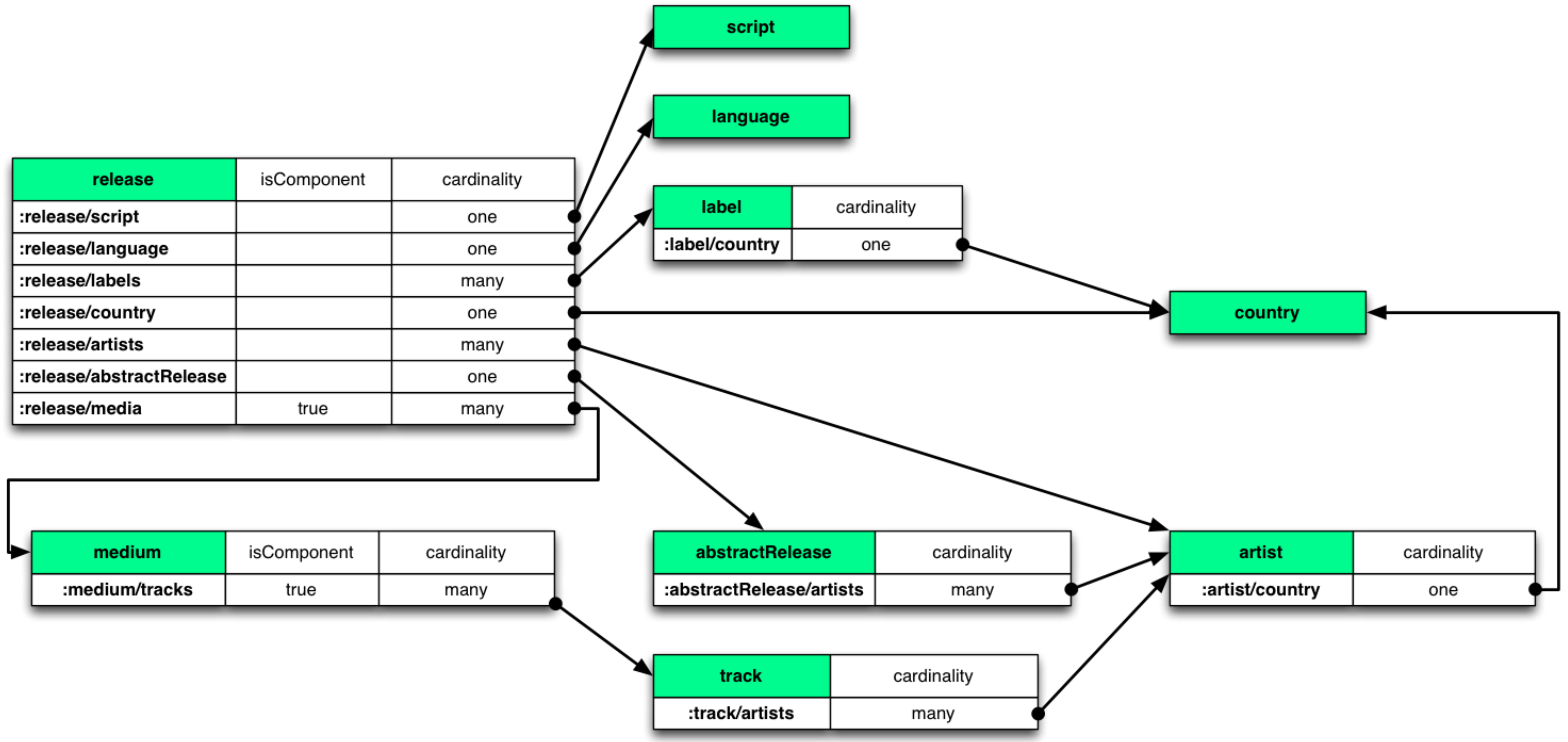
attribute	**type**	**cardinality**
comment/body	string	1
comment/author	ref	1
news/comments	ref	many

“types” do not dictate attrs

attribute	type	cardinality
story/title	string	1
story/url	string	1
story/slug	string	1
news/comments	ref	many

attribute	type	cardinality
comment/body	string	1
comment/author	ref	1
news/comments	ref	many

example: mbrainz



release	isComponent	cardinality
:release/script		one
:release/language		one
:release/labels		many
:release/country		one
:release/artists		many
:release/abstractRelease		one
:release/media	true	many

label	cardinality
:label/country	one

medium	isComponent	cardinality
:medium/tracks	true	many

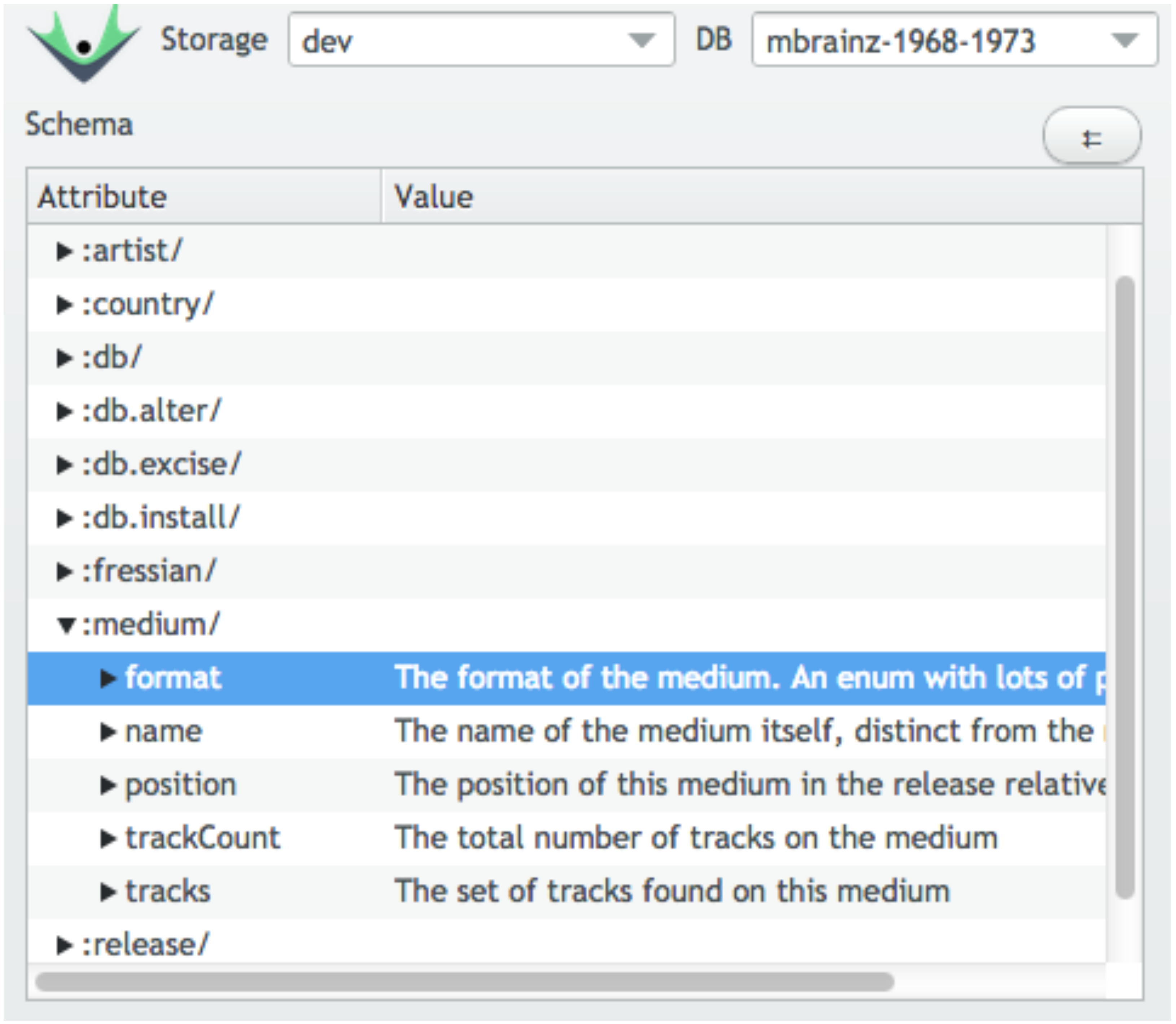
abstractRelease	cardinality
:abstractRelease/artists	many

artist	cardinality
:artist/country	one

track	cardinality
:track/artists	many

https://github.com/Datomic/mbrainz-sample

schema in console



Storage: dev
DB: mbrainz-1968-1973

Schema

Attribute	Value
▶ :artist/	
▶ :country/	
▶ :db/	
▶ :db.alter/	
▶ :db.excise/	
▶ :db.install/	
▶ :fressian/	
▼ :medium/	
▶ format	The format of the medium. An enum with lots of p
▶ name	The name of the medium itself, distinct from the
▶ position	The position of this medium in the release relative
▶ trackCount	The total number of tracks on the medium
▶ tracks	The set of tracks found on this medium
▶ :release/	

lab: designing a schema

pick a **small** domain you know well

draw an entity/relationship diagram

convert to edn data

transact into a database

view schema in console

https://github.com/Datomic/day-of-datomic/tree/master/resources/day-of-datomic

transaction model

ACID

assertion and retraction

entity maps

ids and partitions

uniqueness

transaction functions

http://docs.datomic.com/transactions.html

ACID

atomic	transaction is a set of datoms transaction entirely in single write
consistent	all processes see same global ordering of transactions
isolated	single writer system (nobody to be isolated *from*)
durable	always flush through to durable storage before reporting transaction complete

http://docs.datomic.com/acid.html

assertion and retraction

```
[:db/add entity-id attribute value]
```

```
[:db/retract entity-id attribute value]
```

entity maps

concise form for multiple assertions about an entity

equivalent to corresponding list of asserts

can nest arbitrarily

http://docs.datomic.com/transactions.html#map-forms

lists vs. entity maps

```
[:db/add 42 :likes "pizza"]
[:db/add 42 :firstName "John"]
[:db/add 42 :lastName "Doe"]
```

assertion list

↕

```
{:db/id 42
 :likes "pizza"
 :firstName "John"
 :lastName "Doe"}
```

entity map

cross reference

```
[{:db/id #db/id[:db.part/user -1]
  :person/name "Bob"
  :person/spouse #db/id[:db.part/user -2]}

 {:db/id #db/id[:db.part/user -2]
  :person/name "Alice"
  :person/spouse #db/id[:db.part/user -1]}]
```

nesting

```
[{:db/id order-id
  :order/lineItems [{:lineItem/product chocolate
                     :lineItem/quantity 1}
                    {:lineItem/product whisky
                     :lineItem/quantity 2}]}]
```

partitions

http://docs.datomic.com/schema.html#partitions

partitions

- group related entities in a partition
  - coarser granularity than e.g. tables
- partition is a hint to indexing
  - group these things together
  - can help locality
  - does not affect semantics

partition locality

part	e	a	v	tx	added
0	0x0000000000000000	10	:db.part/db	0xc0000000000	TRUE
0	0x0000000000000000	11	0	0xc0000000036	TRUE
0	0x0000000000000000	11	3	0xc0000000036	TRUE
0	0x0000000000000000	11	4	0xc0000000036	TRUE
3	0x00000c0000000036	50	Wed Dec 31 19:00:00 E…	0xc0000000036	TRUE
3	0x00000c0000000038	50	Wed Dec 31 19:00:00 E…	0xc0000000038	TRUE
3	0x00000c000000003f	50	Wed Dec 31 19:00:00 E…	0xc000000003f	TRUE
3	0x00000c00000003e8	50	Mon Oct 13 18:52:59 E…	0xc00000003e8	TRUE
4	0x00001000000003eb	10	:country/BF	0xc00000003ea	TRUE
4	0x00001000000003eb	84	Burkina Faso	0xc00000003ea	TRUE
4	0x00001000000003ec	10	:country/JE	0xc00000003ea	TRUE
4	0x00001000000003ec	84	Jersey	0xc00000003ea	TRUE

https://github.com/Datomic/day-of-datomic/blob/master/tutorial/partition_locality.clj

built-in partitions

partition	usage
:db.part/db	schema entities
:db.part/tx	transaction entities
:db.part/user	user entities

creating partitions

```
[{:db.install/_partition :db.part/db,
  :db/id #db/id[:db.part/db],
  :db/ident :inventory}
 {:db.install/_partition :db.part/db,
  :db/id #db/id[:db.part/db],
  :db/ident :customers}]
```

identity

http://docs.datomic.com/identity.html

identity

requirement	model with	value types
db-relative opaque id	entity id	opaque (long)
external id	:db.unique/identity attribute	string, uuid, uri
global opaque id	:db.unique/identity squuid	uuid
programmatic name	:db/ident	keyword

squuids

semi-sequential UUIDs

do not fragment indexes

```
public class Peer;
    public static UUID squuid();
    public static long squuidTimeMillis(UUID squuid);
    // other methods elided for brevity
}
```

transaction functions

http://docs.datomic.com/database-functions.html#transaction-functions

add and retract

```
[[:db/add john :likes pizza]
 [:db/retract john :likes iceCream]]
```

what about update?

```
[[:db/add john :likes pizza]
 [:db/retract john :likes iceCream]
 [:db/add john :balance 110?]]
```

atomic increment

```
[[:db/add john :likes pizza]
 [:db/retract john :likes iceCream]
 [:inc john :account 10]]
```

transaction fns

subset of data fns

run inside transactions

have access to in-tx value of database

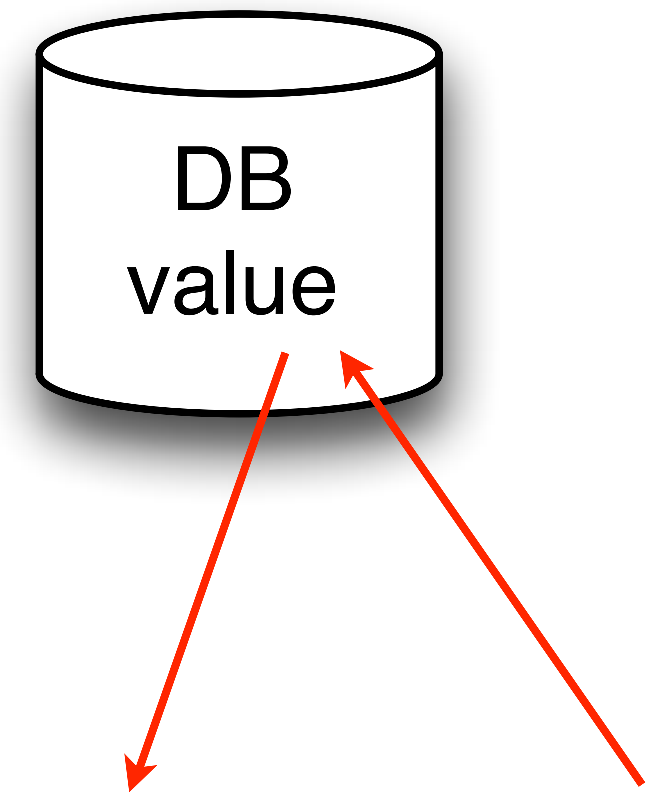
- as first argument

tx function expansion

```
[[:db/add john :likes pizza]
 [:db/retract john :likes iceCream]
 [:inc john :balance 10]]
```

```
[[:db/add john :likes pizza]
 [:db/retract john :likes iceCream]
 [:db/add john :balance 110]]
```

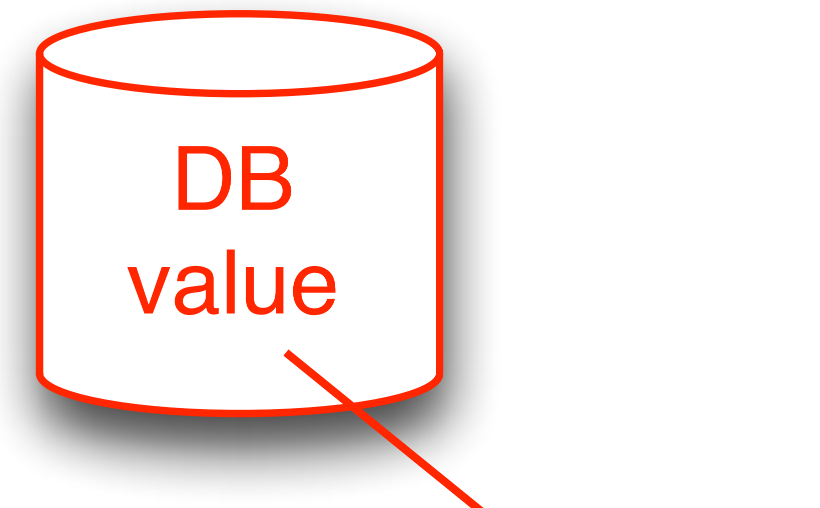
lookup the function



```
[:inc john :balance 10]
```

```
inc = db.entity("inc").get("db/fn");
```

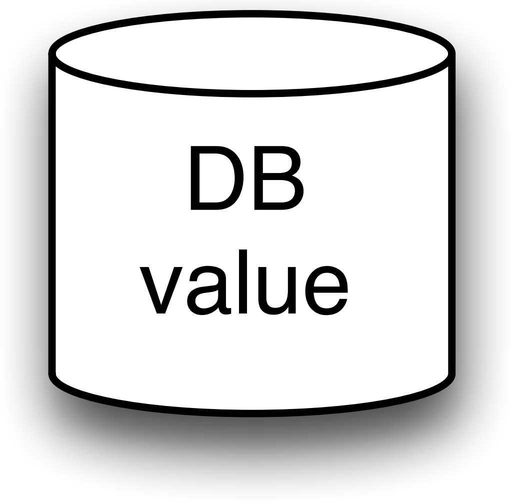
pass in current db



```
[:inc john :account 10]
```

```
inc.invoke(db, ...);
```

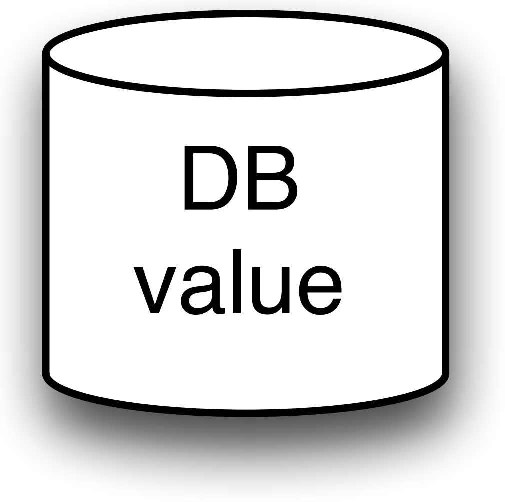
pass in args



```
[:inc john :account 10]
```

```
inc.invoke(db, john, :account, 10)
```

data out



```
[:inc john :account 10]
```

```
inc.invoke(db, john, :account, 10)
```

$$\downarrow$$

```
[[:db/add john :account 110]]
```

inc

```
public static Object inc(Object db, Object e, Object amount)
{
    // lookup entity
    // calculate new balance
    // create assertion
    // return list containing assertion
}
```

inc

```
public static Object inc(Object db, Object e, Object a, Object amount) {
    Entity ent = ((Database)db).entity(e);
    Long balance = (Long) ent.get(a) + (Long) amount;
    List updated = list("db/add", e, a, balance);
    return list(updated);
}
```

lab: adding some data

create assertions and retractions	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/crud.clj
create entity maps	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/component_attributes.clj
modify an existing entity	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/crud.clj
review your data at the console	
bonus: create a constructor function	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/data_functions.clj

query model

datalog

pull

entities

raw indexes

why datalog?

equivalent to relational model + recursion

better fit than prolog for query

- no clause order dependency
- guaranteed termination

pattern-matching style easy to learn

example database

entity	attribute	value
42	:email	jdoe@example.com
43	:email	jane@example.com
42	:orders	107
42	:orders	141

data pattern

Constrains the results returned,
binds variables

```
[?customer :email ?email]
```

data pattern

Constrains the results returned,
binds variables

```
[?customer :email ?email]
```

entity attribute value

data pattern

Constrains the results returned,
binds variables

$$\begin{tikzcd}\text{constant} \arrow[d] \\ \texttt{:email}\end{tikzcd}$$

```
[?customer :email ?email]
```

data pattern

Constrains the results returned, binds variables

$$\overset{\text{variable}}{\downarrow}\qquad\qquad\qquad\qquad\overset{\text{variable}}{\downarrow}$$

```
[?customer :email ?email]
```

example database

entity	attribute	value
42	:email	jdoe@example.com
43	:email	jane@example.com
42	:orders	107
42	:orders	141

```
[?customer :email ?email]
```

constants anywhere

"Find a particular customer's email"

```
[42 :email ?email]
```

entity	attribute	value
42	:email	jdoe@example.com
43	:email	jane@example.com
42	:orders	107
42	:orders	141

```
[42 :email ?email]
```

variables anywhere

“What attributes does customer 42 have?

```
[42 ?attribute]
```

entity	attribute	value
42	:email	jdoe@example.com
43	:email	jane@example.com
42	:orders	107
42	:orders	141

```
[42 ?attribute]
```

variables anywhere

“What attributes and values does customer 42 have?

```
[42 ?attribute ?value]
```

entity	attribute	value
42	:email	jdoe@example.com
43	:email	jane@example.com
42	:orders	107
42	:orders	141

```
[42 ?attribute ?value]
```

where clause

data
pattern

```
[:find ?customer
 :where [?customer :email]]
```

find clause

variable to return

```
[:find ?customer
 :where [?customer :email]]
```

implicit join

"Find all the customers who have placed orders."

```
[:find ?customer
 :where [?customer :email]
        [?customer :orders]]
```

API

```
import static datomic.Peer.q;


q("[:find ?customer
    :where [?customer :id]
           [?customer :orders]]",
   db);
```

q

```
import static datomic.Peer.q;

q("[:find ?customer
    :where [?customer :id]
           [?customer :orders]]",
  db);
```

query

```
import static datomic.Peer.q;

q("[:find ?customer
    :where [?customer :id]
           [?customer :orders]]",
   db);
```

input(s)

```
import static datomic.Peer.q;

q("[:find ?customer
    :where [?customer :id]
           [?customer :orders]]",
   db);
```

in clause

Names inputs so you can refer to them elsewhere in the query

```
:in $database ?email
```

parameterized query

"Find a customer by email."

```
q([:find ?customer
   :in $database ?email
   :where [$database ?customer :email ?email]],
   db,
   "jdoe@example.com");
```

first input

“Find a customer by email.”

```
q([:find ?customer
   :in $database ?email
   :where [$database ?customer :email ?email]],
   db,
   "jdoe@example.com");
```

second input

“Find a customer by email.”

```
q([:find ?customer
   :in $database ?email
   :where [$database ?customer :email ?email]],
   db,
   "jdoe@example.com");
```

verbose?

“Find a customer by email.”

```
q([:find ?customer
   :in $database ?email
   :where [$database ?customer :email ?email]],
  db,
  "jdoe@example.com");
```

shortest name possible

“Find a customer by email.”

```
q([:find ?customer
   :in $ ?email
   :where [$ ?customer :email ?email]],
   db,
   "jdoe@example.com");
```

elide $ in where

"Find a customer by email."

```
q([:find ?customer
   :in $ ?email
   :where [ ?customer :email ?email]],
   db,
   "jdoe@example.com");
```

no need to
specify $

predicates

Functional constraints that can appear in a :where clause

```
[(< 50 ?price)]
```

adding a predicate

“Find the expensive items”

```
[:find ?item
 :where [?item :item/price ?price]
        [(< 50 ?price)]]
```

functions

*Take bound variables as inputs
and bind variables with output*

```
[(shipping ?zip ?weight) ?cost]
```

function args

```
[(shipping ?zip ?weight) ?cost]
```

bound inputs

function returns

```
[(shipping ?zip ?weight) ?cost]
```

$\uparrow$

bind return
values

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

navigate from customer to zip

```
[: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)]]
```

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

get product facts
needed *during query*

```
[: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)]]
```

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

```
[: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)]]
```

call web service to bind shipCost

BYO functions

Functions can be plain JVM code.

```
public class Shipping {
  public static BigDecimal
  estimate(String zip1, int pounds);
}
```

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

```
[: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)]]
```

constrain price

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

return customer, product pairs

```
[: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)]]
```

calling a function

"Find me the customer/product combinations where the shipping cost dominates the product cost."

```
[: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)]]
```

entities

maplike, point-in-time view of datoms sharing a common **e**

```
{:db/id 42
 :likes "pizza"
 :firstName "John"
 :lastName "Doe"}
```

entity

datoms

```
[42 :likes "pizza"]
[42 :firstName "John"]
[42 :lastName "Doe"]
```

entities

transformation is purely mechanical

special key for e

```
{:db/id 42
 :likes "pizza"
 :firstName "John"
 :lastName "Doe"}
```

```
[42 :likes "pizza"]
[42 :firstName "John"]
[42 :lastName "Doe"]
```

one database, many indexes

structure	**attribute**
k/v	AVET
row	EAVT
column	AEVT
document	EAVT, partitions, components
graph	VAET

lab: query

explore mbrainz data with Query API	https://github.com/Datomic/mbrainz-sample/wiki/Queries
explore mbrainz data with Pull API	http://docs.datomic.com/pull.html
query your own data	http://docs.datomic.com/query.html
pull your own data	http://docs.datomic.com/pull.html
navigate your data with the Entity API	http://docs.datomic.com/entities.html

time model

reified transactions

t & basis-t

log

filters

sync

excision

reified transactions

entities like any other entity in the system

associated with every datom

increasing entity ids over time

associated with increasing counter **t**

have a :db/txInstant

have any other attributes you specify

have their own index (the log)

basis

```
(def basis-t (d/basis-t db))
=> 130223
```

basis is most recent t, tx

```
(def basis-tx (d/t->tx basis-t))
=> 13194139663535
```

```
(d/pull db '[*] basis-tx)
=> {:db/id 13194139663535,
    :db/txInstant #inst "2014-10-13T22:58:03.832-00:00"}
```

facts about basis tx

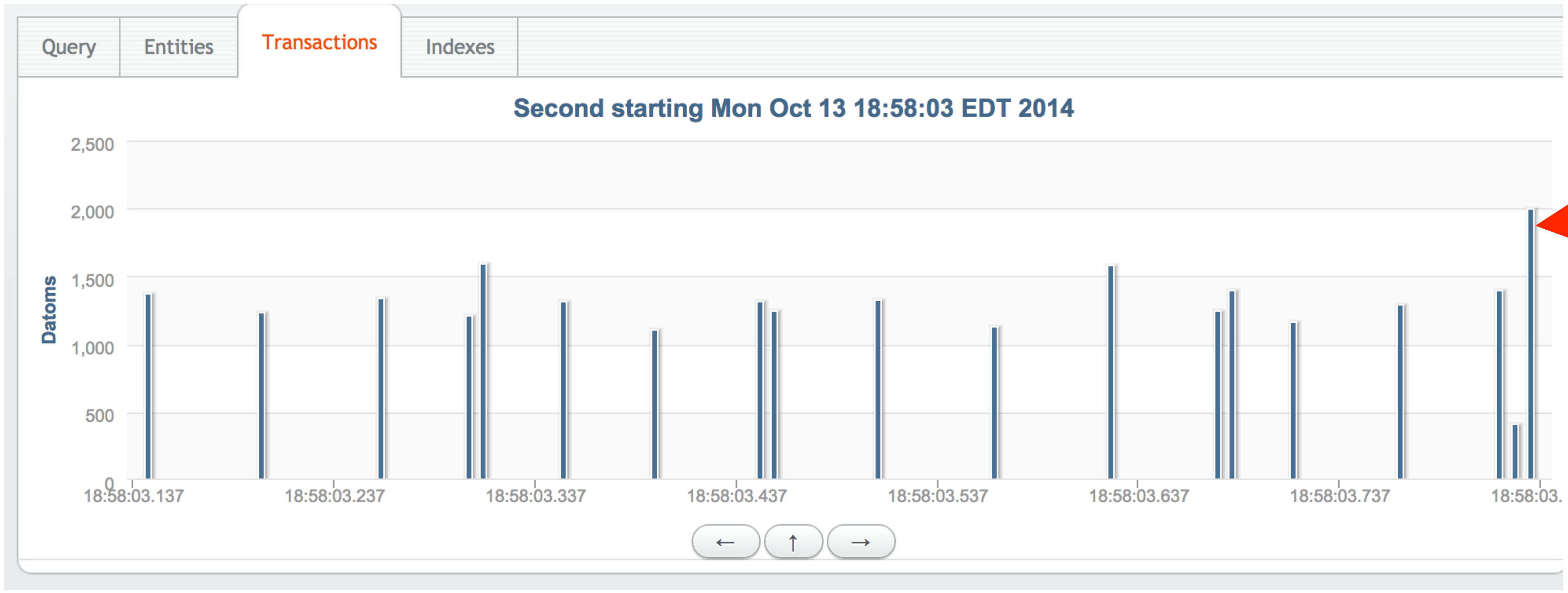
a *time point* is any of: t, tx, instant

https://github.com/Datomic/day-of-datomic/blob/master/tutorial/basis_t_and_log.clj

log

```
(def log (d/log conn))
(-> (d/tx-range log basis-tx (inc basis-tx))
    seq first :data count)
```

contents of txes



basis tx

http://docs.datomic.com/log.html

transaction attributes

tx partition

references current tx

```
[{:db/id #db/id [:db.part/user]
  :story/title "codeq"
  :story/url "http://blog.datomic.com/2012/10/codeq.html"}
 {:db/id (d/tempid :db.part/tx)
  :publish/at (java.util.Date.)}]
```

add your own attributes

```
[{:db/id [:item/id "DLC-042"]
  :item/count 250}
 {:db/id #db/id [:db.part/tx]
  :db/txInstant #inst "2013-02"}]
```

override txInstant (for imports)

https://github.com/Datomic/day-of-datomic/blob/master/tutorial/filter.clj

https://github.com/Datomic/day-of-datomic/blob/master/tutorial/filters.clj

filters

filter	semantics	supports
(default)	current state	what is the current situation?
.asOf	state at point in past	how were things in the past?
.since	state since point in past	how have things changed?
.history	timeless view of all history	anything!

https://github.com/Datomic/day-of-datomic/blob/master/tutorial/filters.clj

raw data

e	a	v	tx	added
0x00000c00000003e9	:db/txInstant	Mon Dec 31 19:00:00	0xc00000003e9	TRUE
0x00001000000003ea	:item/id	DLC-042	0xc00000003e9	TRUE
0x00001000000003ea	:item/description	Dilitihium Crystals	0xc00000003e9	TRUE
0x00001000000003ea	:item/count	100	0xc00000003e9	TRUE
0x00000c00000003eb	:db/txInstant	Thu Jan 31 19:00:00	0xc00000003eb	TRUE
0x00001000000003ea	:item/count	100	0xc00000003eb	FALSE
0x00001000000003ea	:item/count	250	0xc00000003eb	TRUE
0x00000c00000003ec	:db/txInstant	Thu Feb 27 19:00:00	0xc00000003ec	TRUE
0x00001000000003ea	:item/count	250	0xc00000003ec	FALSE
0x00001000000003ea	:item/count	50	0xc00000003ec	TRUE
0x00000c00000003ed	:db/txInstant	Mon Mar 31 20:00:00	0xc00000003ed	TRUE
0x00000c00000003ed	:tx/error	TRUE	0xc00000003ed	TRUE
0x00001000000003ea	:item/count	50	0xc00000003ed	FALSE
0x00001000000003ea	:item/count	9999	0xc00000003ed	TRUE
0x00000c00000003ee	:db/txInstant	Wed May 14 20:00:00	0xc00000003ee	TRUE
0x00001000000003ea	:item/count	9999	0xc00000003ee	FALSE
0x00001000000003ea	:item/count	100	0xc00000003ee	TRUE

default filter

e	a	v	tx	added
0x00000c00000003e9	:db/txInstant	Mon Dec 31 19:00:00	0xc00000003e9	TRUE
0x00001000000003ea	:item/id	DLC-042	0xc00000003e9	TRUE
0x00001000000003ea	:item/description	Dilitihium Crystals	0xc00000003e9	TRUE
0x00001000000003ea	:item/count	100	0xc00000003e9	TRUE
0x00000c00000003eb	:db/txInstant	Thu Jan 31 19:00:00	0xc00000003eb	TRUE
0x00001000000003ea	:item/count	100	0xc00000003eb	FALSE
0x00001000000003ea	:item/count	250	0xc00000003eb	TRUE
0x00000c00000003ec	:db/txInstant	Thu Feb 27 19:00:00	0xc00000003ec	TRUE
0x00001000000003ea	:item/count	250	0xc00000003ec	FALSE
0x00001000000003ea	:item/count	50	0xc00000003ec	TRUE
0x00000c00000003ed	:db/txInstant	Mon Mar 31 20:00:00	0xc00000003ed	TRUE
0x00000c00000003ed	:tx/error	TRUE	0xc00000003ed	TRUE
0x00001000000003ea	:item/count	50	0xc00000003ed	FALSE
0x00001000000003ea	:item/count	9999	0xc00000003ed	TRUE
0x00000c00000003ee	:db/txInstant	Wed May 14 20:00:00	0xc00000003ee	TRUE
0x00001000000003ea	:item/count	9999	0xc00000003ee	FALSE
0x00001000000003ea	:item/count	100	0xc00000003ee	TRUE

as-of filter

e	a	v	tx	added
0x00000c00000003e9	:db/txInstant	Mon Dec 31 19:00:00	0xc00000003e9	TRUE
0x00001000000003ea	:item/id	DLC-042	0xc00000003e9	TRUE
0x00001000000003ea	:item/description	Dilitihium Crystals	0xc00000003e9	TRUE
0x00001000000003ea	:item/count	100	0xc00000003e9	TRUE
0x00000c00000003eb	:db/txInstant	Thu Jan 31 19:00:00	0xc00000003eb	TRUE
0x00001000000003ea	:item/count	100	0xc00000003eb	FALSE
0x00001000000003ea	:item/count	250	0xc00000003eb	TRUE
0x00000c00000003ec	:db/txInstant	Thu Feb 27 19:00:00	0xc00000003ec	TRUE
0x00001000000003ea	:item/count	250	0xc00000003ec	FALSE
0x00001000000003ea	:item/count	50	0xc00000003ec	TRUE
0x00000c00000003ed	:db/txInstant	Mon Mar 31 20:00:00	0xc00000003ed	TRUE
0x00000c00000003ed	:tx/error	TRUE	0xc00000003ed	TRUE
0x00001000000003ea	:item/count	50	0xc00000003ed	FALSE
0x00001000000003ea	:item/count	9999	0xc00000003ed	TRUE
0x00000c00000003ee	:db/txInstant	Wed May 14 20:00:00	0xc00000003ee	TRUE
0x00001000000003ea	:item/count	9999	0xc00000003ee	FALSE
0x00001000000003ea	:item/count	100	0xc00000003ee	TRUE

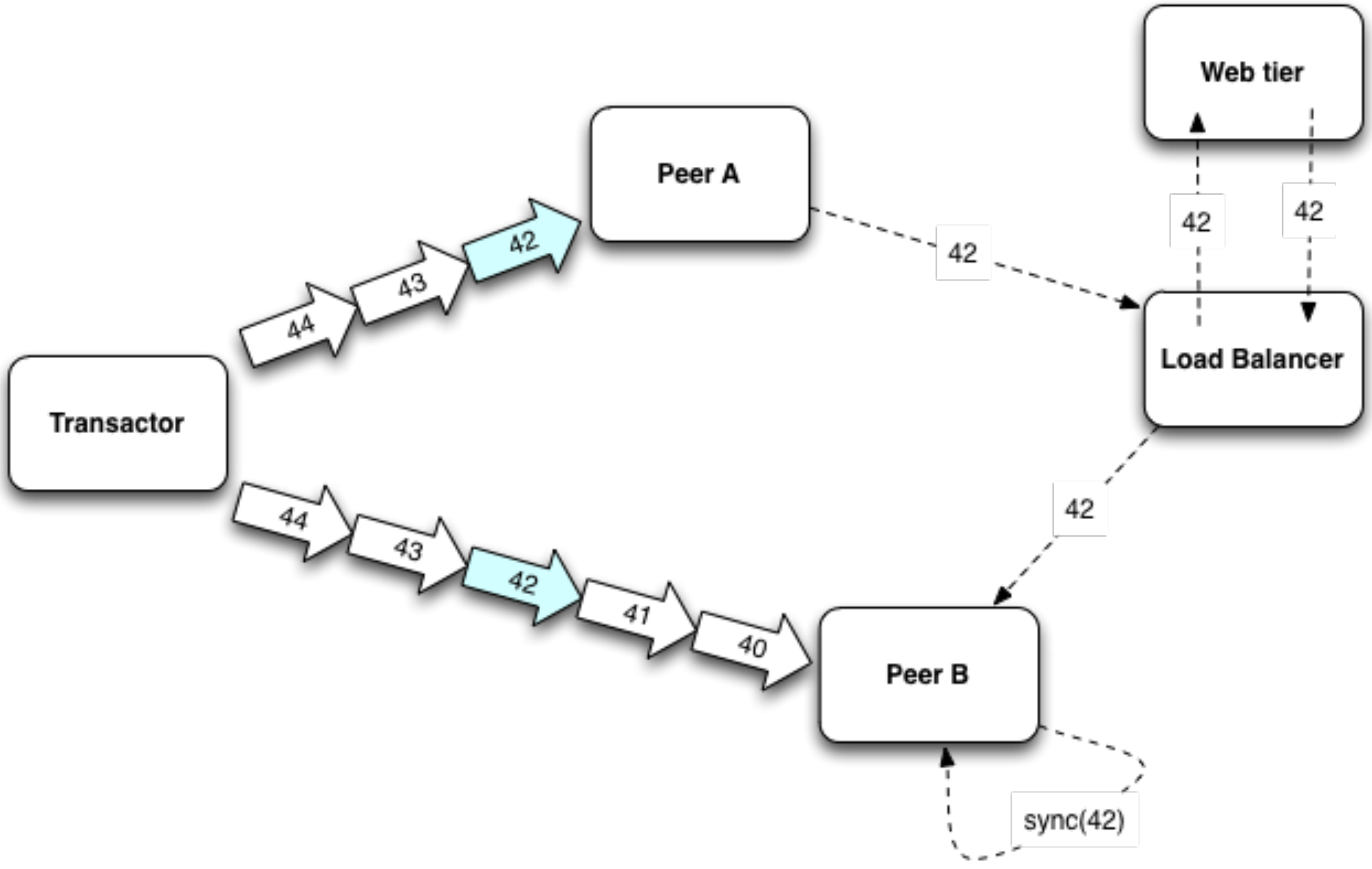
since filter

e	a	v	tx	added
0x00000c00000003e9	:db/txInstant	Mon Dec 31 19:00:00	0xc00000003e9	TRUE
0x00001000000003ea	:item/id	DLC-042	0xc00000003e9	TRUE
0x00001000000003ea	:item/description	Dilitihium Crystals	0xc00000003e9	TRUE
0x00001000000003ea	:item/count	100	0xc00000003e9	TRUE
0x00000c00000003eb	:db/txInstant	Thu Jan 31 19:00:00	0xc00000003eb	TRUE
0x00001000000003ea	:item/count	100	0xc00000003eb	FALSE
0x00001000000003ea	:item/count	250	0xc00000003eb	TRUE
0x00000c00000003ec	:db/txInstant	Thu Feb 27 19:00:00	0xc00000003ec	TRUE
0x00001000000003ea	:item/count	250	0xc00000003ec	FALSE
0x00001000000003ea	:item/count	50	0xc00000003ec	TRUE
0x00000c00000003ed	:db/txInstant	Mon Mar 31 20:00:00	0xc00000003ed	TRUE
0x00000c00000003ed	:tx/error	TRUE	0xc00000003ed	TRUE
0x00001000000003ea	:item/count	50	0xc00000003ed	FALSE
0x00001000000003ea	:item/count	9999	0xc00000003ed	TRUE
0x00000c00000003ee	:db/txInstant	Wed May 14 20:00:00	0xc00000003ee	TRUE
0x00001000000003ea	:item/count	9999	0xc00000003ee	FALSE
0x00001000000003ea	:item/count	100	0xc00000003ee	TRUE

history = raw

e	**a**	**v**	**tx**	**added**
0x00000c00000003e9	:db/txInstant	Mon Dec 31 19:00:00	0xc00000003e9	TRUE
0x00001000000003ea	:item/id	DLC-042	0xc00000003e9	TRUE
0x00001000000003ea	:item/description	Dilitihium Crystals	0xc00000003e9	TRUE
0x00001000000003ea	:item/count	100	0xc00000003e9	TRUE
0x00000c00000003eb	:db/txInstant	Thu Jan 31 19:00:00	0xc00000003eb	TRUE
0x00001000000003ea	:item/count	100	0xc00000003eb	FALSE
0x00001000000003ea	:item/count	250	0xc00000003eb	TRUE
0x00000c00000003ec	:db/txInstant	Thu Feb 27 19:00:00	0xc00000003ec	TRUE
0x00001000000003ea	:item/count	250	0xc00000003ec	FALSE
0x00001000000003ea	:item/count	50	0xc00000003ec	TRUE
0x00000c00000003ed	:db/txInstant	Mon Mar 31 20:00:00	0xc00000003ed	TRUE
0x00000c00000003ed	:tx/error	TRUE	0xc00000003ed	TRUE
0x00001000000003ea	:item/count	50	0xc00000003ed	FALSE
0x00001000000003ea	:item/count	9999	0xc00000003ed	TRUE
0x00000c00000003ee	:db/txInstant	Wed May 14 20:00:00	0xc00000003ee	TRUE
0x00001000000003ea	:item/count	9999	0xc00000003ee	FALSE
0x00001000000003ea	:item/count	100	0xc00000003ee	TRUE

sync



http://blog.datomic.com/2013/06/sync.html excision: permanently, unrecoverably, lose data

don’t do this
unless you must

http://docs.datomic.com/excision.html

lab: navigating time

explore your data with filters in console	
explore your data with transaction view in console	
query against an as-of db	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/filters.clj
use the log to recover a recent transaction	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/basis_t_and_log.clj
annotate a transaction	https://github.com/Datomic/day-of-datomic/blob/master/tutorial/filters.clj

operational model

caching

indexing

bring your own storage

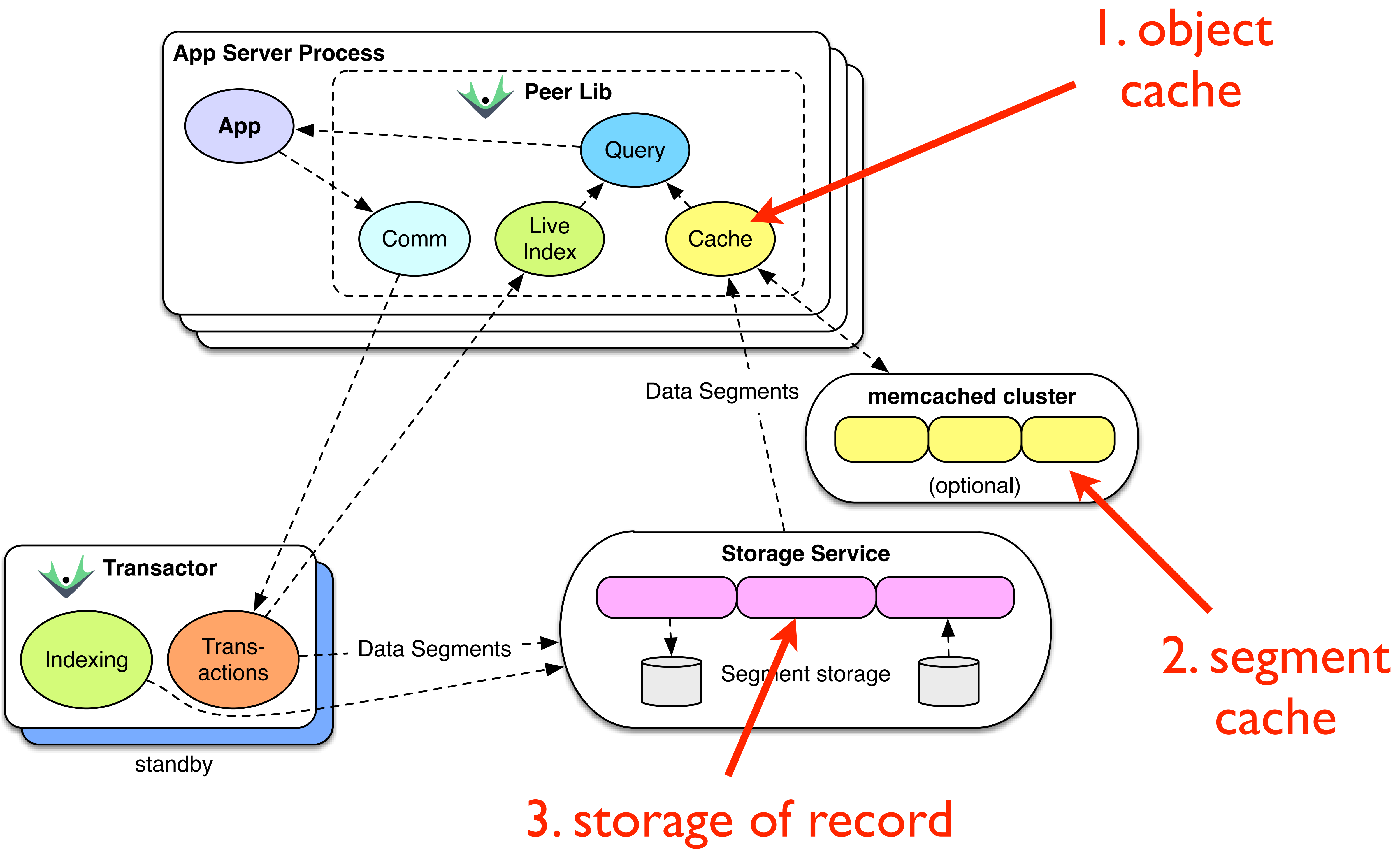
transactors

peers

monitoring

capacity planning

caching


1. object cache
App Server Process
Peer Lib
App
Query
Comm
Live Index
Cache
Data Segments
memcached cluster
(optional)
2. segment cache
Transactor
Indexing
Trans-actions
Data Segments
Storage Service
Segment storage
standby
3. storage of record

understanding caching

single knob: how much?

segments (1000s of datoms), not individual facts

segments are immutable, never expire

segment names can cohabit with other caches

object cache is *in your process!*

  will beat any RPC-based database when cache hits

http://docs.datomic.com/caching.html

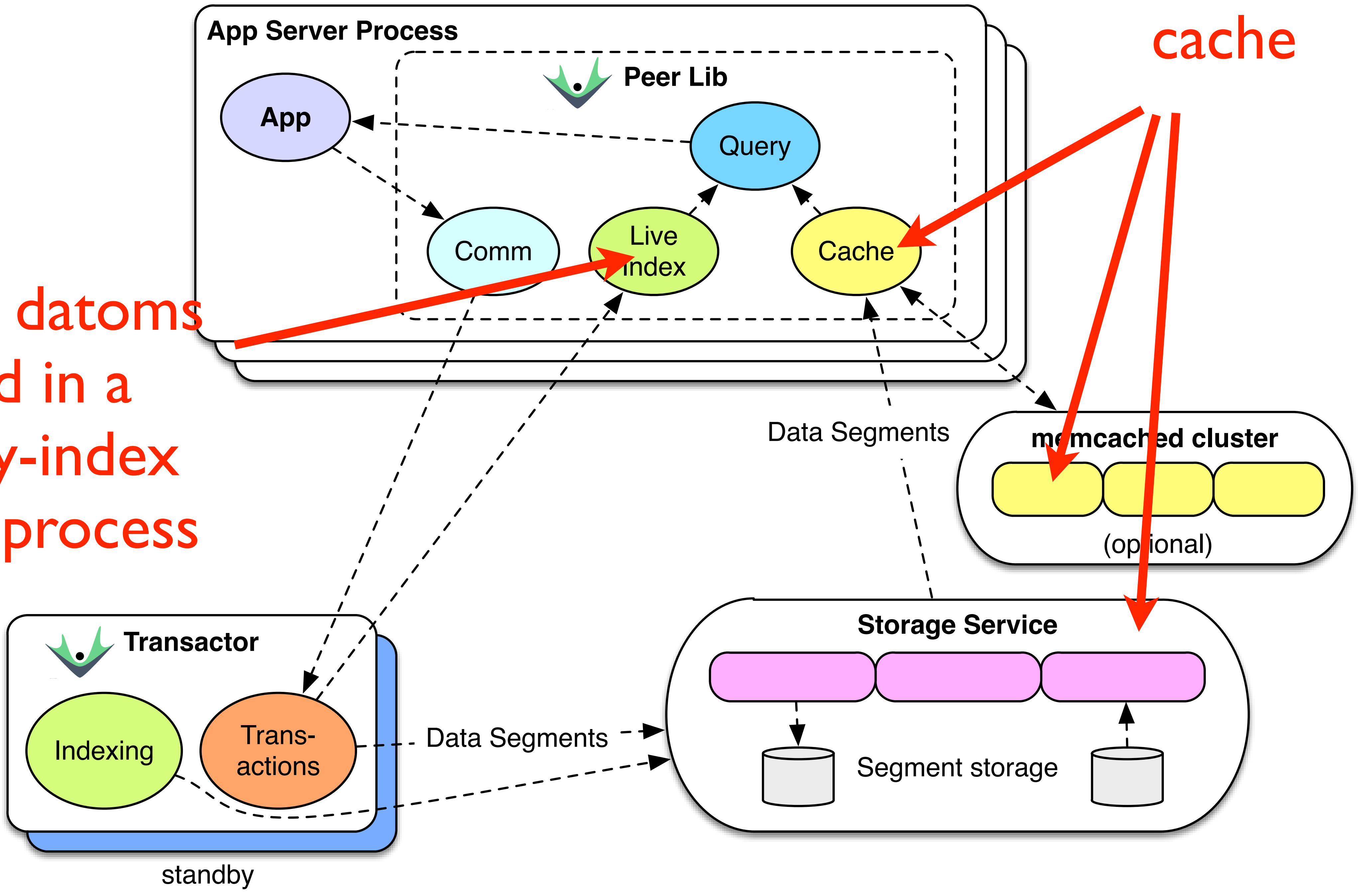
reading data

source	format	capacity	latency
peer object cache	Java objects	~ 10 GB / peer	sub-microsecond
memcached	zipped fressian	arbitrary	~ 1 msec
storage	zipped fressian	arbitrary	~ 10 msec

indexing

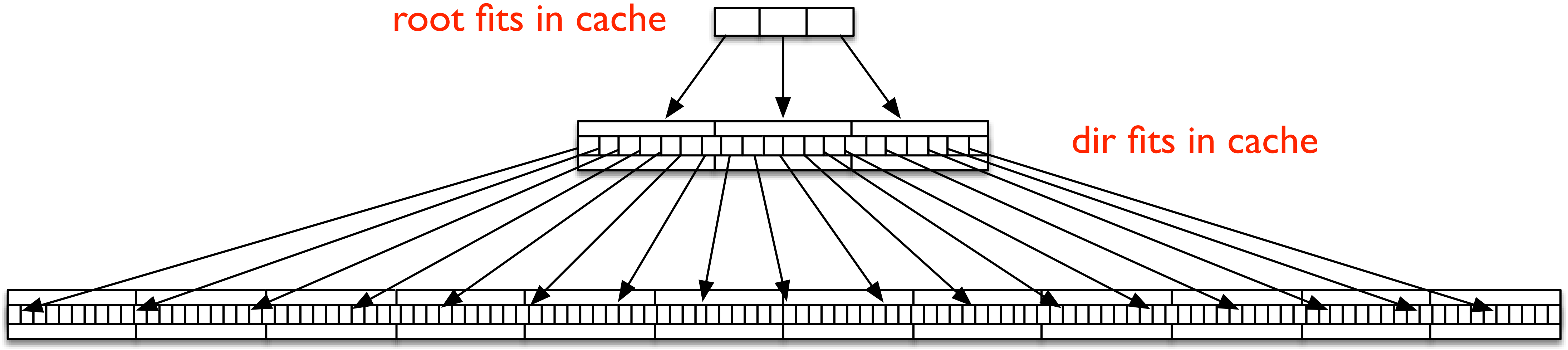
2. older datoms in tiered, distributed cache

1. recent datoms cached in a memory-index in every process



index trees

root fits in cache



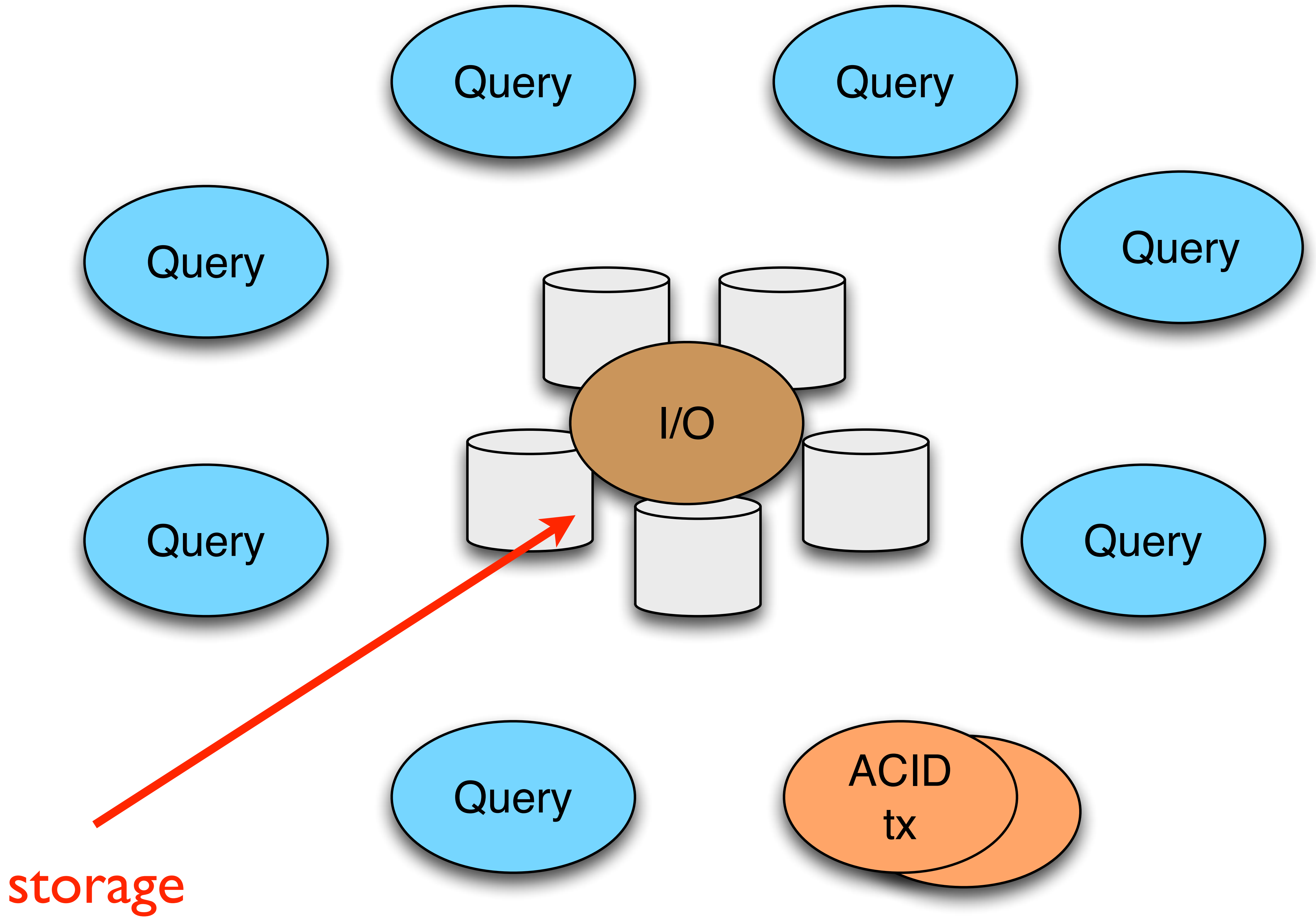
dir fits in cache

segments fit in cache for small dbs,
1-2 reads away for larger dbs

history is in separate trees, and is never in the way of present queries

understanding indexing

- indexes made as needed
  - background process (on transactor)
- indexing can fall too far behind
  - memory index exceeds predefined threshold
  - transactor throttles writes
  - only likely to see during import jobs


Query
Query
Query
Query
I/O
Query
Query
Query
ACID
tx
storage

storage options

dynamodb

sql (any JDBC)

cassandra

riak

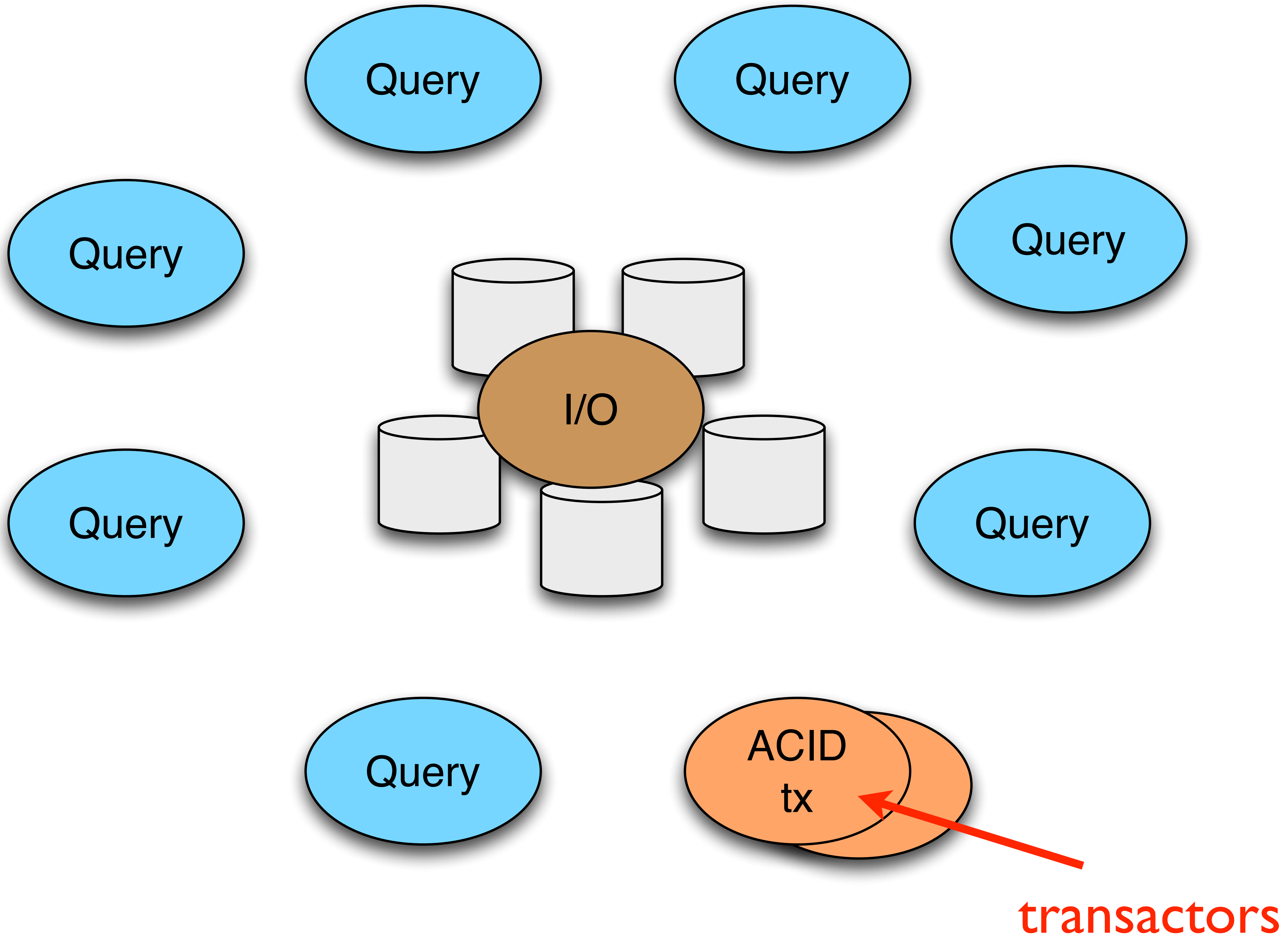
couchbase

infinispan

dev (local disk)

choosing storage

- more important
  - reliability
  - manageability
  - familiarity
- less important
  - datomic insulates storage from load
  - datomic tolerates storage latency


Query
Query
Query
Query
I/O
Query
Query
Query
ACID tx
transactors

understanding transactors

ACID (single writer thread!)

stream recent change to peers

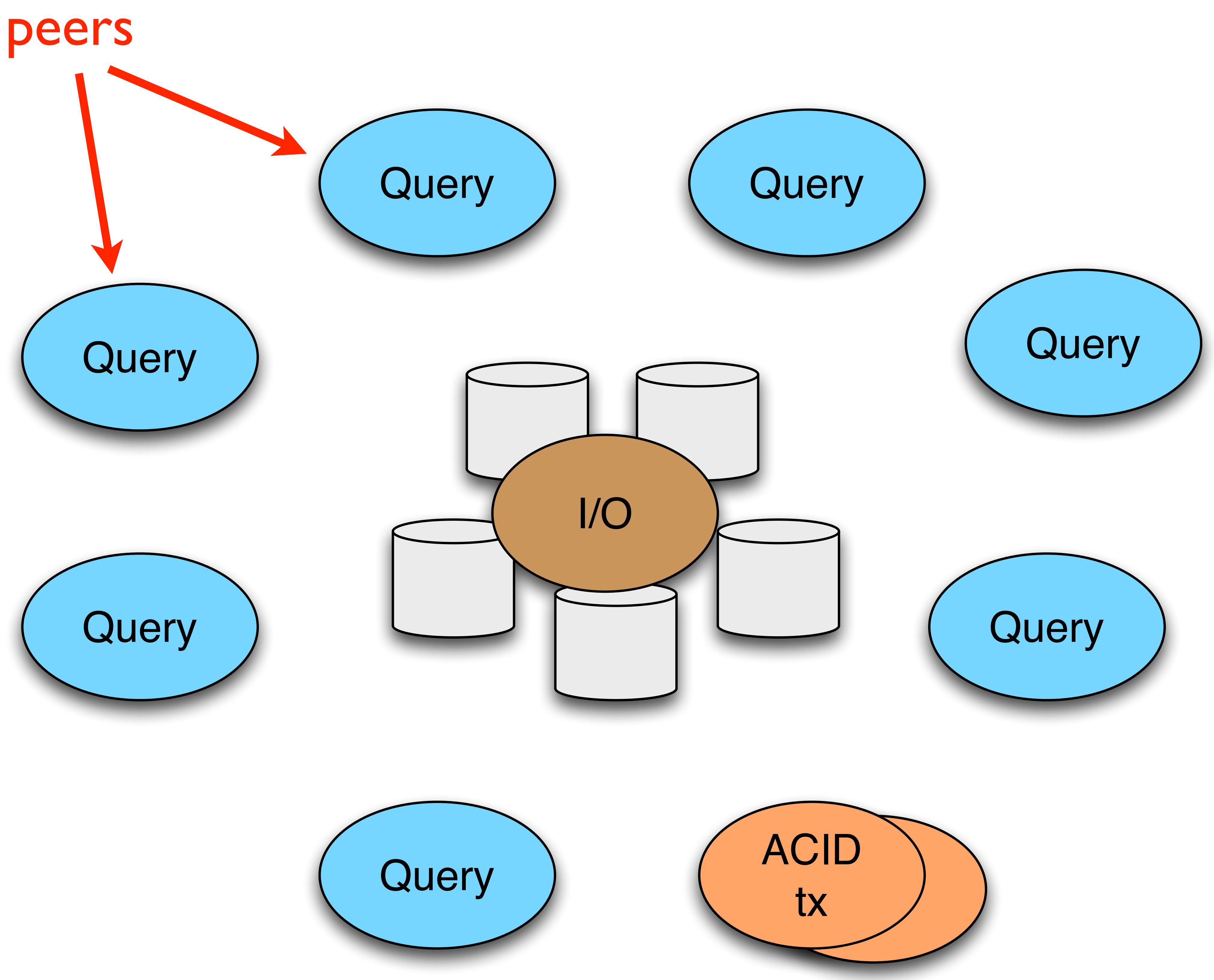
manage indexing

dead-simple HA configuration

- point two machines at one storage

dead-simple HA coordination

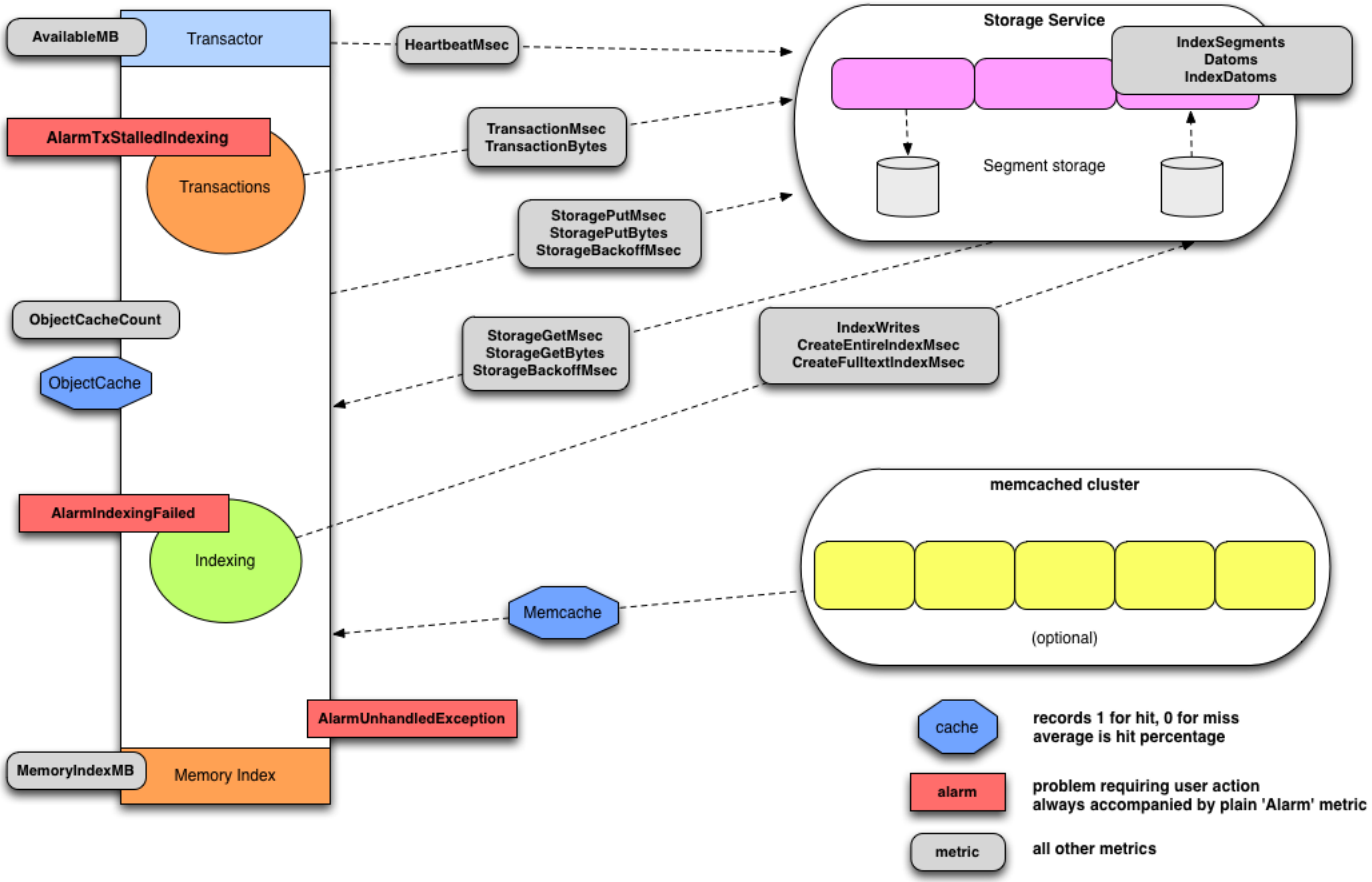
- conditional puts on storage

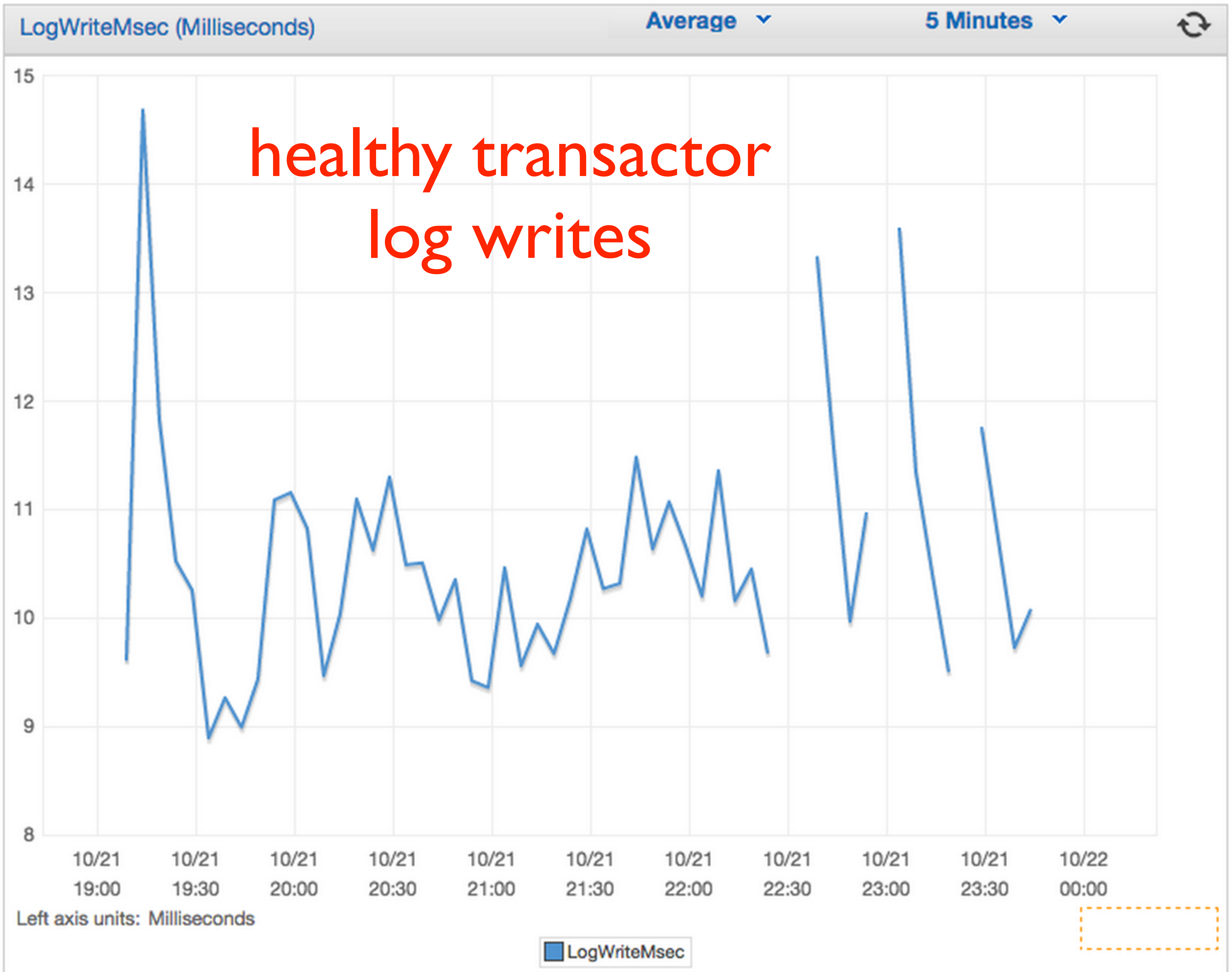

peers
Query
Query
Query
Query
I/O
Query
Query
Query
ACID
tx

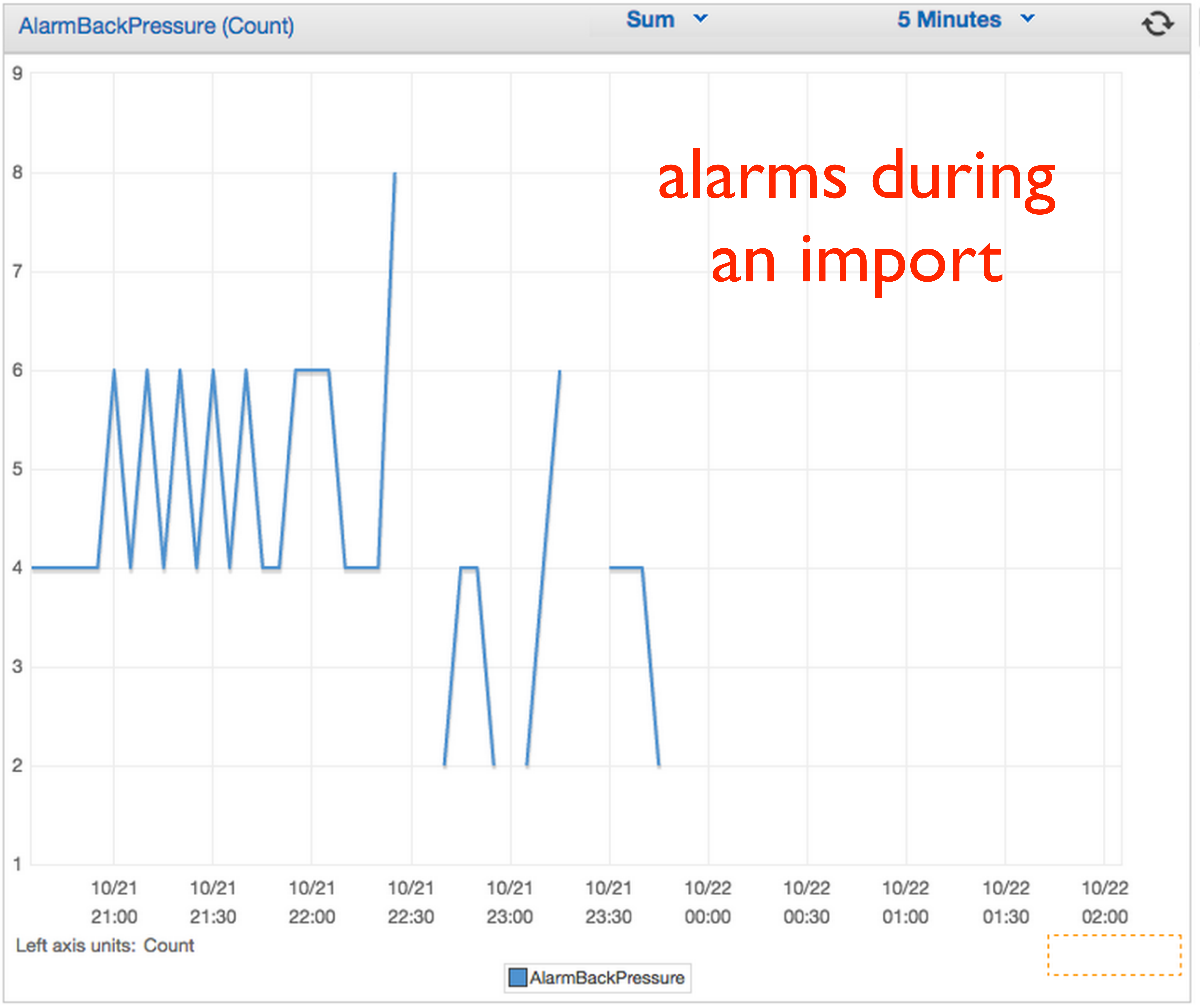
understanding peers

- embedded JVM lib
  - directly in application servers (most common)
  - introduce a service tier
- query load does not bother the transactor
- answer queries in-memory via automatic cache
  - `datomic.objectCacheMax` (default 50% RAM)

Datomic Monitoring




LogWriteMsec (Milliseconds)
Average
5 Minutes
15
14
13
12
11
10
9
8
10/21 19:00
10/21 19:30
10/21 20:00
10/21 20:30
10/21 21:00
10/21 21:30
10/21 22:00
10/21 22:30
10/21 23:00
10/21 23:30
10/22 00:00
Left axis units: Milliseconds
LogWriteMsec
healthy transactor log writes



alarms during an import

capacity planning

data size	up to 10 billion datoms
write volume	consistent with data size in the long run
read volume	elastically scale peers as needed
housekeeping	http://docs.datomic.com/capacity.html#garbage-collection
caching	test and monitor

Datomic is a poor fit for

write scale

media storage (unstructured docs, audio, video)

churn (e.g. hit counter)

big data, graphics, video, unstructured documents, log file analysis

Datomic is a good fit for

valuable information of record

dev and ops flexibility

history

read scale

transactional data, business records, medical records, financial records, scientific records, inventory, configuration, web apps, departmental databases, cloud apps

lab: assess Datomic

for your dataset, create a comparison matrix

which Datomic characteristics are

- beneficial?
- detrimental?
- irrelevant?

compared to ?

10 billion datoms
access pattern at read-time
ACID
as-of
attributes
background indexing
cardinality
cloudwatch monitoring
column access (AEVT)
component attributes
cross-db joins
datalog
data API
document access
edn

elastically scalable read
entity maps
excision
graph access (VAET)
high availability
history
joins
lazy entities
lookup refs
memory speeds
partitions
pluggable monitoring
pluggable storage
pull
query functions
query predicates

raw index access
reified transactions
row access (EAVT)
rules
since
sync
time points
transaction functions
transparent memcached
tx reports
unique ids
universal schema
value access (AVET)
upsert
with

thanks!

@stuarthalloway

https://github.com/stuarthalloway/presentations/wiki
http://www.linkedin.com/pub/stu-halloway/0/110/543/
https://twitter.com/stuarthalloway
mailto:stu@cognitect.com