Datomic Basics

Dog chases mud balls,
Lion bites thrower.

Database of Facts

- Datomic stores **facts** in a univeral schema.
- this schema is composed of **datoms**
- Datoms take the form "something *about* something is true/false *as of* some time"
- or, prefix "I know/believe..."
    - *I like that this suggests an empirical time basis*

Datom

Entity, attribute, value

```
[:dog :chases :mud-balls]
```

And a point in time

```
[:dog :chases :mud-balls :today]
```

And assertion or retraction

```
[:dog :chases :mudballs :today true]
```

Entities

```
{:dog/name      "mu"
 :chases        :mud-balls}
```

are projected from facts about them

```
[:dog :dog/name "mu"          12 true]
[:dog :chases   :mud-balls 20 true]
```

Develop an Intuition for this!

Entities ...

```
{:db/id     8
 :dog/name "mu"
 :chases    :mud-balls}
```

projected from:

```
[8 :dog/name "mu"]
[8 :chases    :mud-balls]
```

Entities ...

```
{:db/id     8
 :dog/name "mu"
 :chases    :mud-balls}
```

projected from:

```
[8 :dog/name "mu"]
[8 :chases   4]
[4 :db/ident :mud-balls]
```

Entities ...

```
{:db/id     8
 :dog/name "mu"
 :chases   #{:mud-balls :rabbits}}
```

projected from:

```
[8 :dog/name "mu"]
[8 :chases   4]
[8 :chases   7]
[4 :db/ident :mud-balls]
[7 :db/ident :rabbits]
```

Entities ...

```
{:db/id     4
 :db/ident :mud-balls
 :_chases   #{8}}
```

projected from:

```
[8 :dog/name "mu"]
[8 :chases    4]
[8 :chases    7]
[4 :db/ident :mud-balls]
[7 :db/ident :rabbits]
```

Time

```
[8 :dog/name "mu"         1 true]
[8 :chases   4            1 true]
[8 :chases   7            1 true]
[4 :db/ident :mud-balls   1 true]
[7 :db/ident :rabbits     1 true]
[8 :chases   7            2 false]
```

Look at time 1:

```
{:dog/name  "mu"
 :chases    #{:mud-balls :rabbits}}
```

Look at time 2:

```
{:dog/name  "mu"
 :chases    #{:mud-balls}}
```

```
[8 :dog/name "mu"          1 true]
[8 :chases   4             1 true]
[8 :chases   7             1 true]   ;; X
[4 :db/ident :mud-balls    1 true]
[7 :db/ident :rabbits      1 true]
[8 :chases   7             2 false]   ;; X
```

Retract cancels the assert, leaving us:

```
[8 :dog/name "mu"          1 true]
[8 :chases   7             1 true]
[4 :db/ident :mud-balls    1 true]
[7 :db/ident :rabbits      1 true]
```

Again, understanding entities<->datoms is crucial!

Assert something

What datom does this add to the database?

```
[:db/add 8 :chases :dreams]
```

reorder:

```
[8 :chases :dreams true]
```

Right?

Missing:

as of,

```
[8 :chases :dreams 3 true]
```

So far:

```
[8 :dog/name "mu"          1 true]
[8 :chases   4             1 true]
[8 :chases   7             1 true]   ;; X
[4 :db/ident :mud-balls    1 true]
[7 :db/ident :rabbits      1 true]
[8 :chases   7             2 false]   ;; X
[8 :chases   9             3 true]
[9 :db/ident :dreams       3 true]
```

What was the real assertion?

```
[:db/add 8 :chases 9]
[:db/add 9 :db/ident :dreams]
```

or:

```
[{:db/id "dreams"
  :db/ident :dreams}
 [:db/add 8 :chases "dreams"]]
```

entity<->facts
map<->list

Find the datoms!

```
[{:db/id             "bk"
  :person/name       "Ben"
  :person/age        who-told-you-ive-been-alive-forever
  :person/degree     :geography
  :person/children   [{:person/name "Sam"
                        :person/age 1}
                       {:person/name "Oliver"
                        :person/age 3}]}
 {:company/name "ThinkTopic"
  :company/employees ["bk"]}]
```

Ops and Architecture Detour!

Ditributed database:

- transactor writes
- peers read
- storage is independent of either
- new:
  - peer server
  - client

Obtaining

- Sign up for license key
- licenses:
  - no time out
  - previous: peer limited, unlimited new releases
  - current: unlimited peers, will have to pay eventually
- Yes, it's proprietary

Deploying

AWS:

- cloud formation
- dynamo storage
- tl;dr this is the happy path

Local/Remote modularity:

- roll your own
- e.g. docker compose
- voltron deployment model
- maybe a future discussion on this if it pans out

Query

Select:

- query via Datomic's datalog implementation
- low level constructs like `datoms` available

Project entites:

- eagerly to data with `pull`
- lazily as an entity map with `entity`

Back to our previous universe:

```
[8 :dog/name "mu"         1 true]
[8 :chases   4            1 true]
[8 :chases   7            1 true]   ;; X
[4 :db/ident :mud-balls   1 true]
[7 :db/ident :rabbits     1 true]
[8 :chases   7            2 false] ;; X
[8 :chases   9            3 true]
[9 :db/ident :dreams      3 true]
```

Ask a question:

```
[:find ?e
 :where
 [?e :chases :mud-balls]]
```

Get an answer:

```
#{[8]}
```

Schema

- Datomic enforces constraints with schema.
- Attributes are entities!

```
[8 100 "mu" 1 true]
```

implies:

```
{:db/id          100
 :db/ident       :dog/name
 :db/valueType   :db.type/string
 :db/cardinality :db.cardinality/one}
```

Optimizations

- Datomic stores all data in covering indexes
- Sorts are: `:eavt :aevt :avet :vaet`
- Indexes are shallow trees of segments
    - segments: clumps of datoms

To reason about index use, look for known to unknown variable relations:

```
[:find ?e
 :where
 [?e :chases :mud-balls]]
```

- we know `:mud-balls` (value position)
- we know `:chases` (attribute position)

For ref traversal, we use `:vaet`:

```
[:mud-balls :chases 9 1]
```

Let's go interactive!