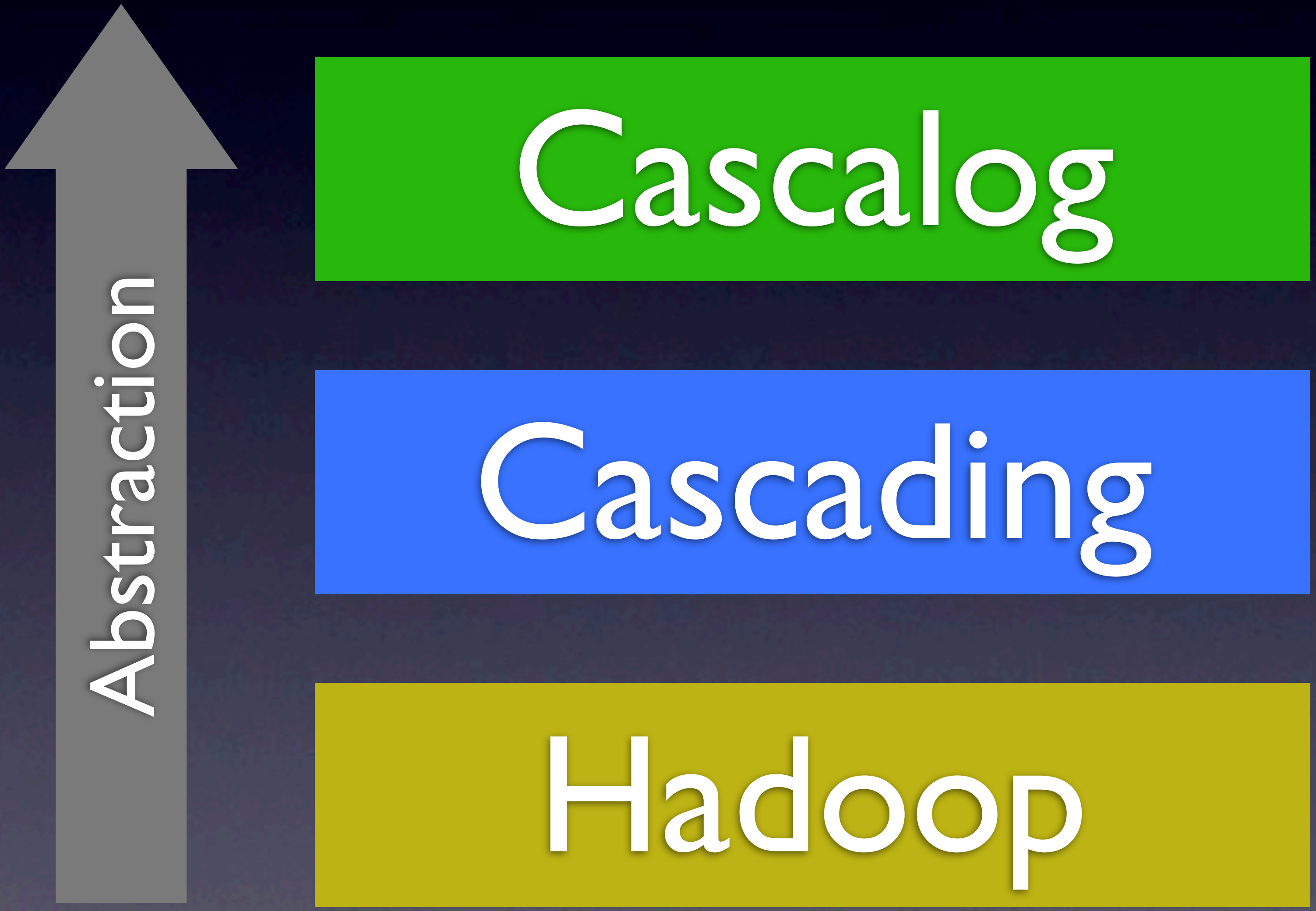
Cascalog

Nathan Marz
Twitter
@nathanmarz

Let’s do some analysis

Tweets during the Tunisian revolution

What is Cascalog?



Variables and logic

Tuples, data workflows

Key/value pairs,
simple aggregation

What is Hadoop MapReduce?

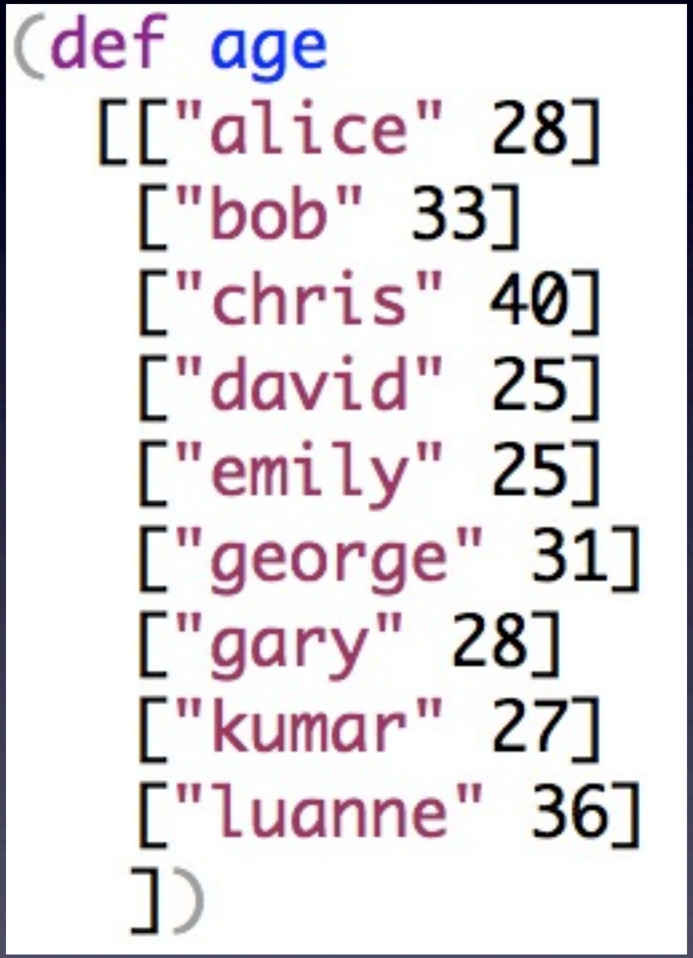
- High latency batch processing
- Massive scale (petabytes)
- Fault-tolerant

Why Cascalog?

Abstraction

Composition

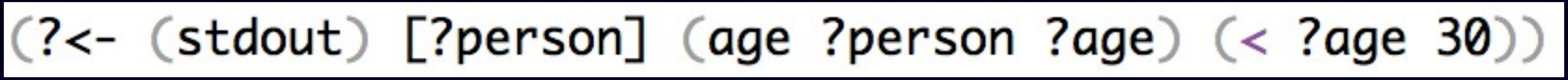
Cascalog basics



```
(def age
  [["alice" 28]
   ["bob" 33]
   ["chris" 40]
   ["david" 25]
   ["emily" 25]
   ["george" 31]
   ["gary" 28]
   ["kumar" 27]
   ["luanne" 36]
   ])
```

The “age” dataset

Cascalog basics



```
(?<- (stdout) [?person] (age ?person ?age) (< ?age 30))
```

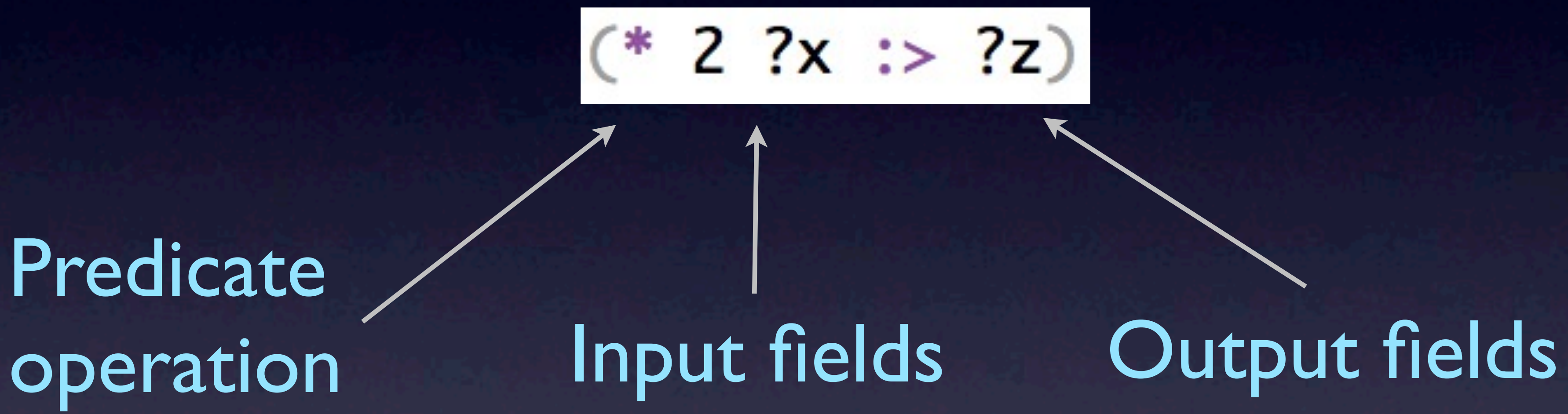
Where to emit results

"Predicates": constrain the output variables

Output variables

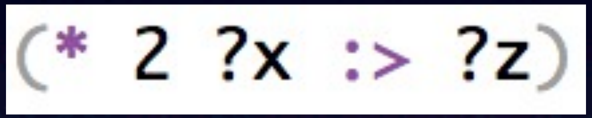
Define and execute a query

Predicates



```
(* 2 ?x :> ?z)
```

Predicates

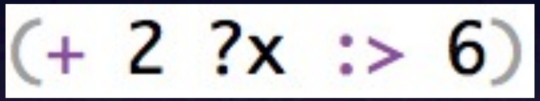


```
(* 2 ?x :> ?z)
```

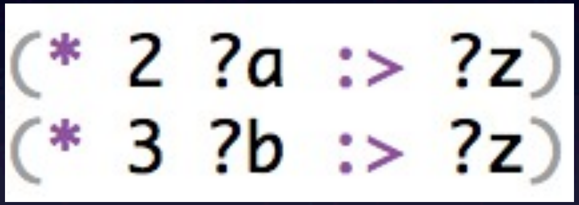
Fields can be constants or variables

Variables are prefixed with ? or !

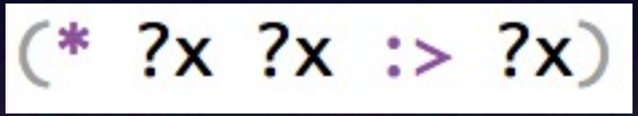
Predicates



```
(+ 2 ?x :> 6)
```



```
(* 2 ?a :> ?z)
(* 3 ?b :> ?z)
```

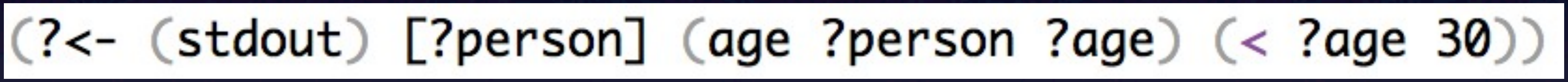


```
(* ?x ?x :> ?x)
```

Predicates

- Functions
- Filters
- Aggregators
- Generators: finite sources of tuples

Example #1

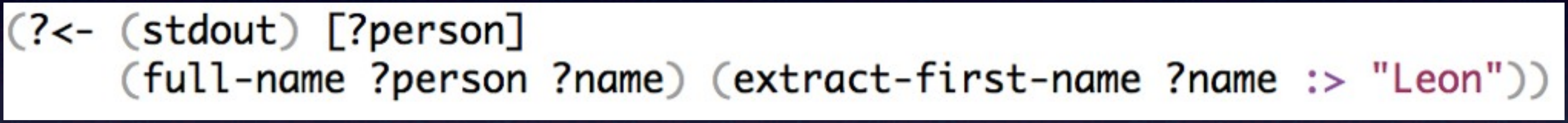


```
(?<- (stdout) [?person] (age ?person ?age) (< ?age 30))
```

Generator

Filter

Example #2

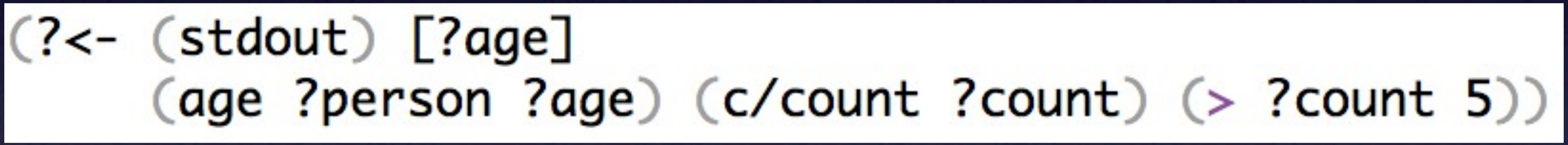


```
(?<- (stdout) [?person]
     (full-name ?person ?name) (extract-first-name ?name :> "Leon"))
```

Generator

Function

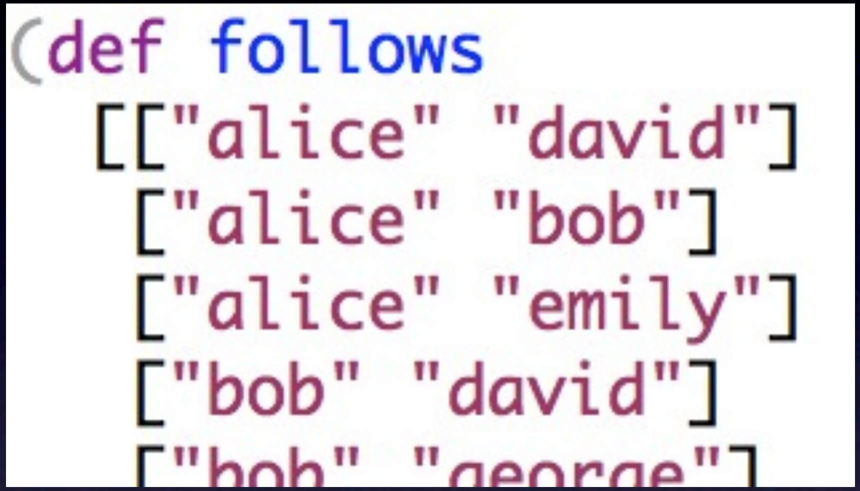
Example #3



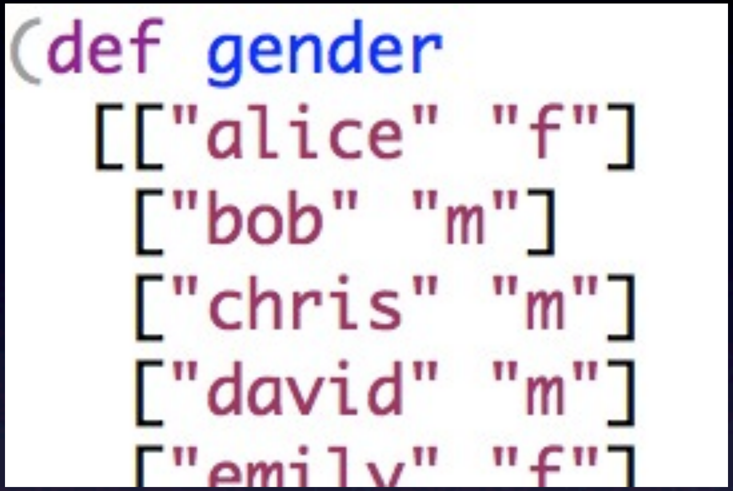
```
(?<- (stdout) [?age]
     (age ?person ?age) (c/count ?count) (> ?count 5))
```

Generator   Aggregator   Filter

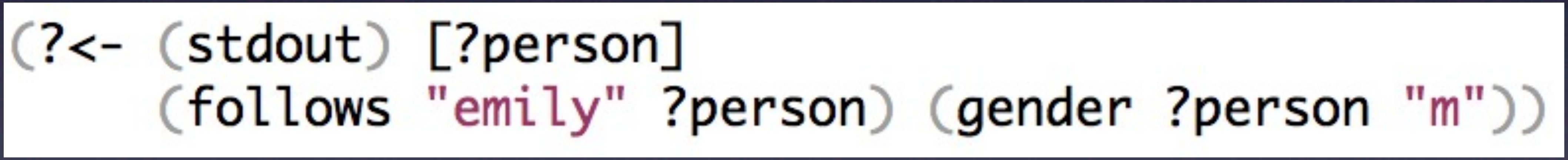
Join example



```
(def follows
  [["alice" "david"]
   ["alice" "bob"]
   ["alice" "emily"]
   ["bob" "david"]
   ["bob" "george"]
```



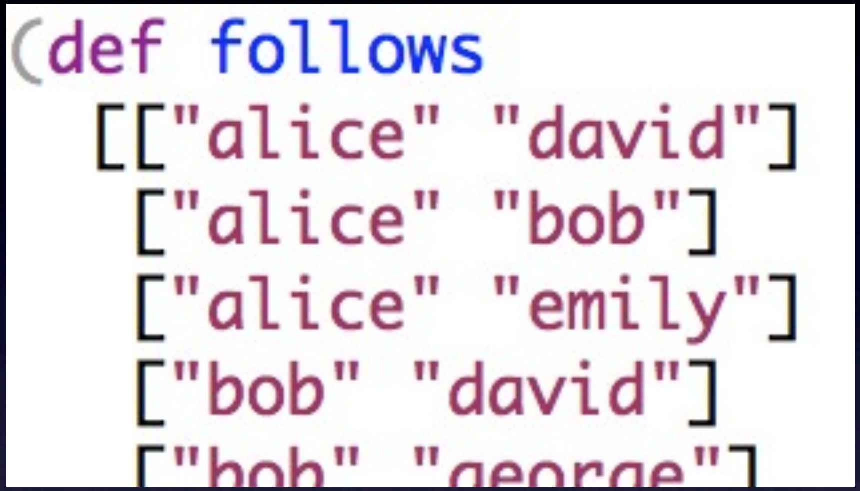
```
(def gender
  [["alice" "f"]
   ["bob" "m"]
   ["chris" "m"]
   ["david" "m"]
   ["emily" "f"]
```



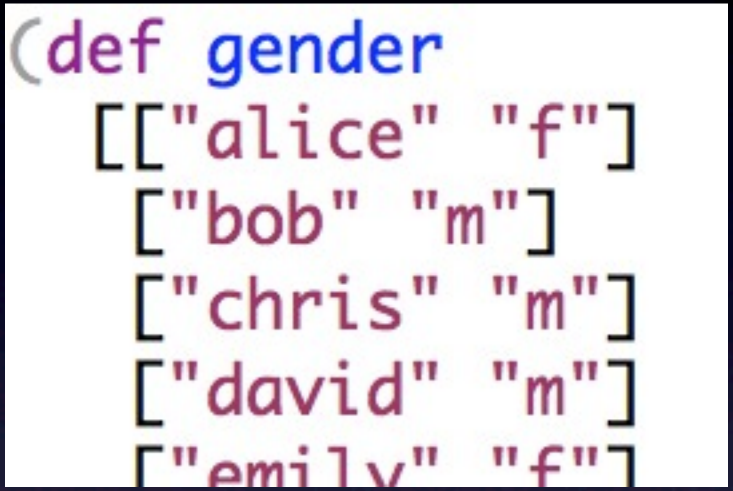
```
(?<- (stdout) [?person]
     (follows "emily" ?person) (gender ?person "m"))
```

Triggers a join

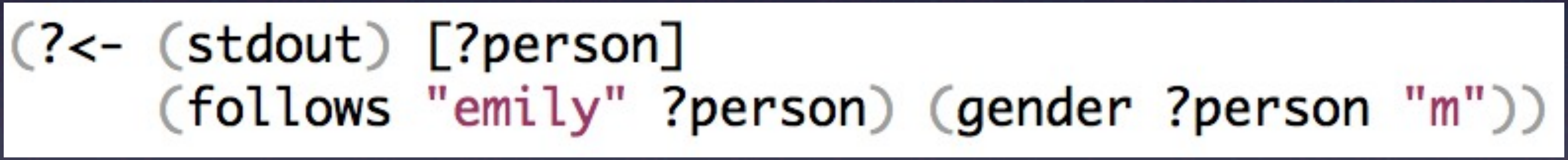
Join example



```
(def follows
  [["alice" "david"]
   ["alice" "bob"]
   ["alice" "emily"]
   ["bob" "david"]
   ["bob" "george"]
```



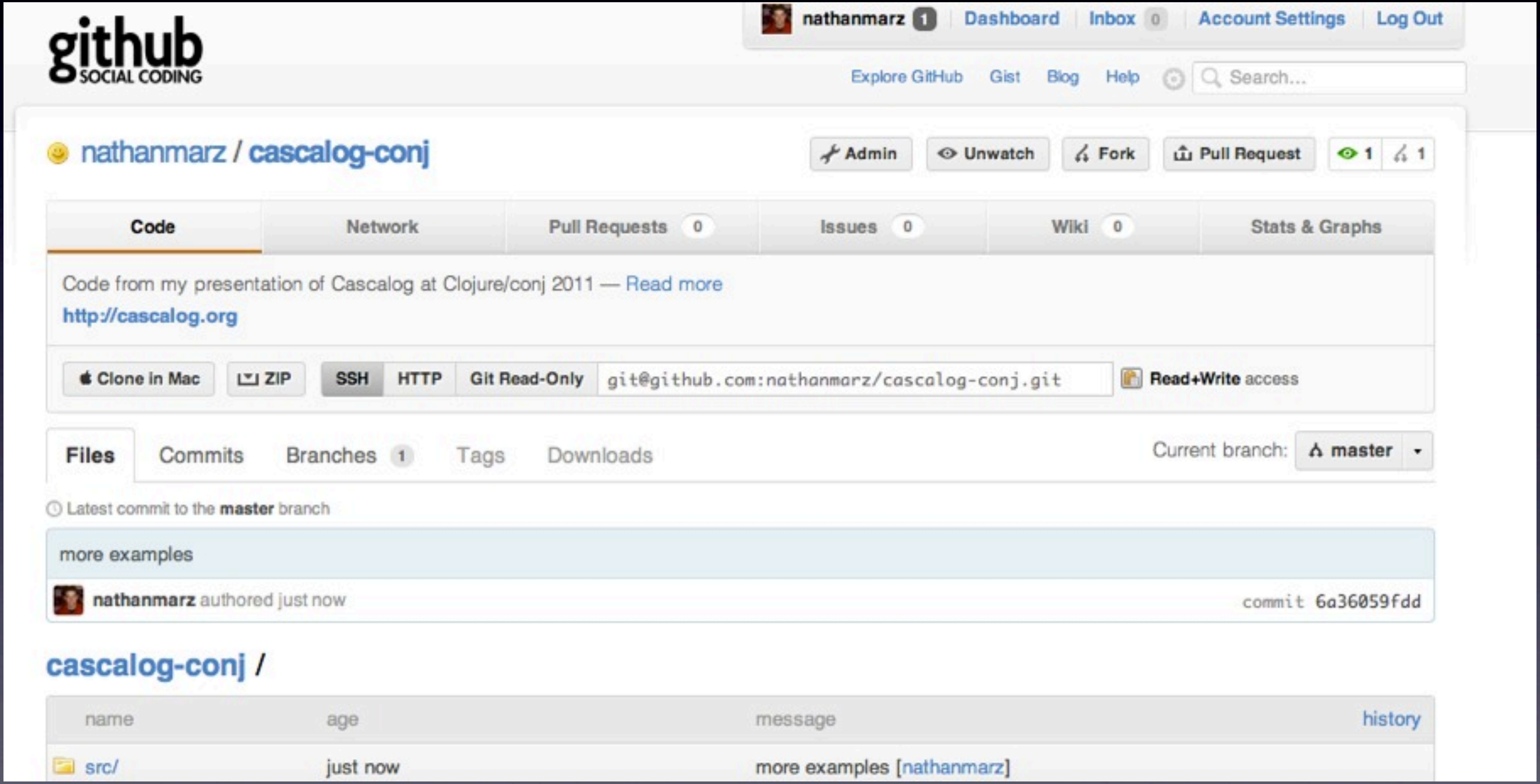
```
(def gender
  [["alice" "f"]
   ["bob" "m"]
   ["chris" "m"]
   ["david" "m"]
   ["emily" "f"]
```



```
(?<- (stdout) [?person]
     (follows "emily" ?person) (gender ?person "m"))
```

Joins are implicit

Demo code


github SOCIAL CODING
nathanmarz 1 Dashboard Inbox 0 Account Settings Log Out
Explore GitHub Gist Blog Help Search...
Admin Unwatch Fork Pull Request 1 1
Code Network Pull Requests 0 Issues 0 Wiki 0 Stats & Graphs
Clone in Mac ZIP SSH HTTP Git Read-Only
Files Commits Branches 1 Tags Downloads
Current branch: master

nathanmarz / cascalog-conj

Code from my presentation of Cascalog at Clojure/conj 2011 — Read more
http://cascalog.org

git@github.com:nathanmarz/cascalog-conj.git Read+Write access

Latest commit to the master branch

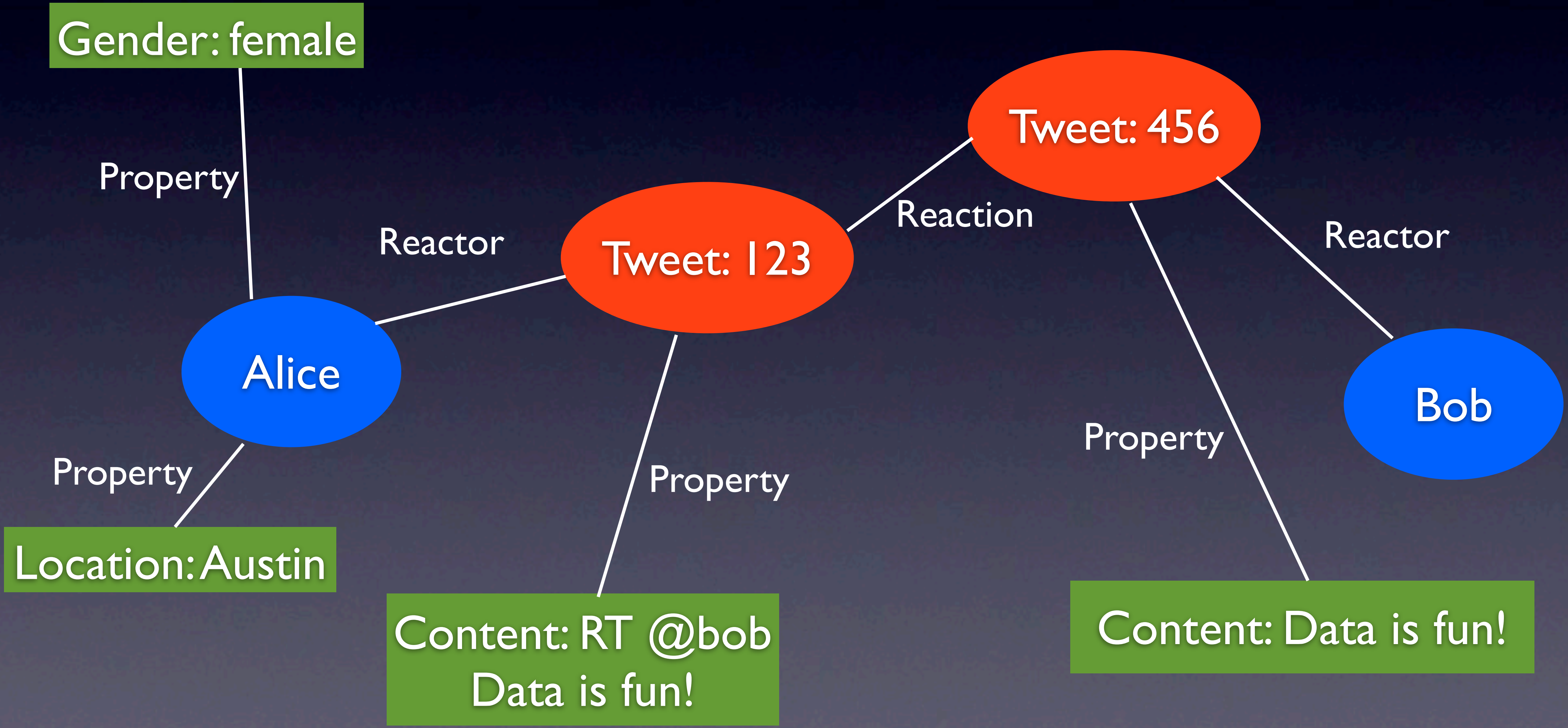
more examples

nathanmarz authored just now commit 6a36059fdd

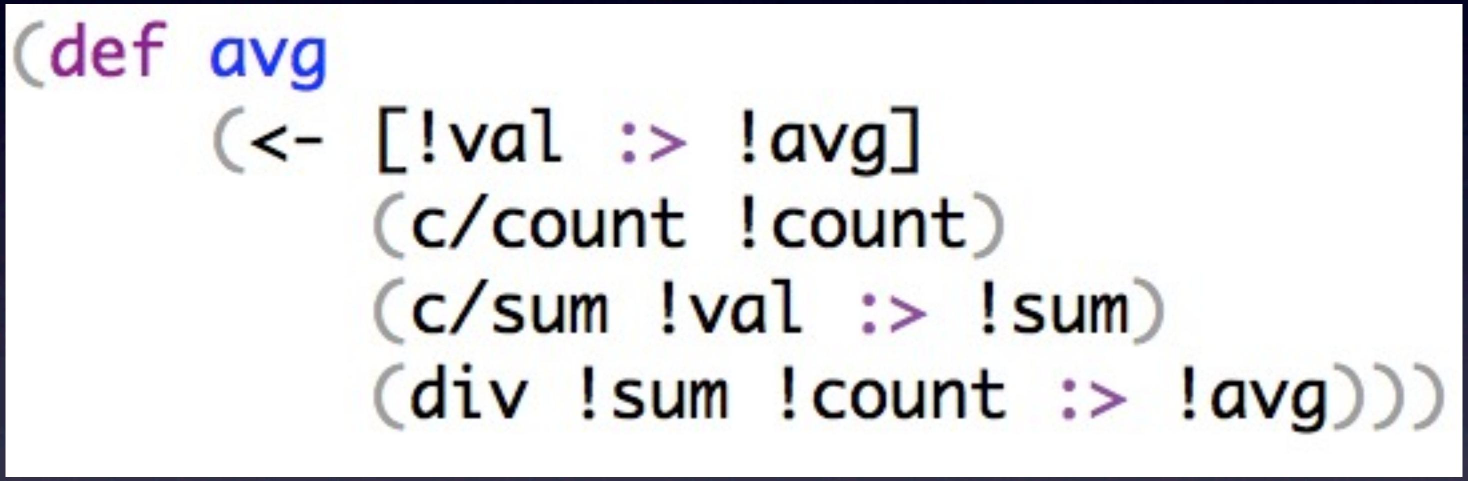
cascalog-conj /

name	age	message	history
src/	just now	more examples [nathanmarz]	

Data model


Gender: female
Property
Reactor
Tweet: 123
Tweet: 456
Reaction
Reactor
Alice
Bob
Property
Property
Property
Location: Austin
Content: RT @bob Data is fun!
Content: Data is fun!

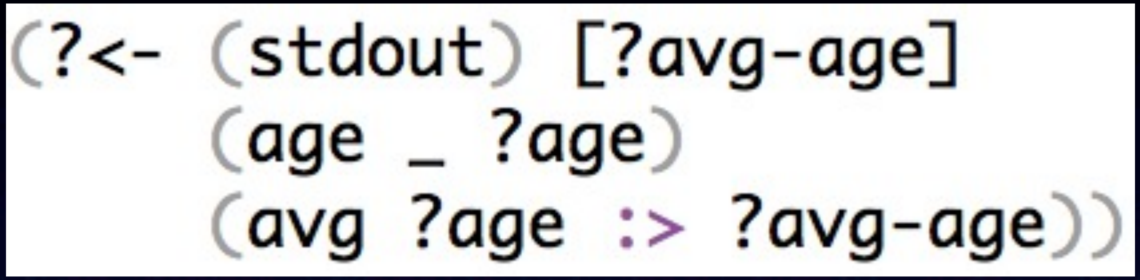
Composability



```
(def avg
    (<- [!val :> !avg]
        (c/count !count)
        (c/sum !val :> !sum)
        (div !sum !count :> !avg)))
```

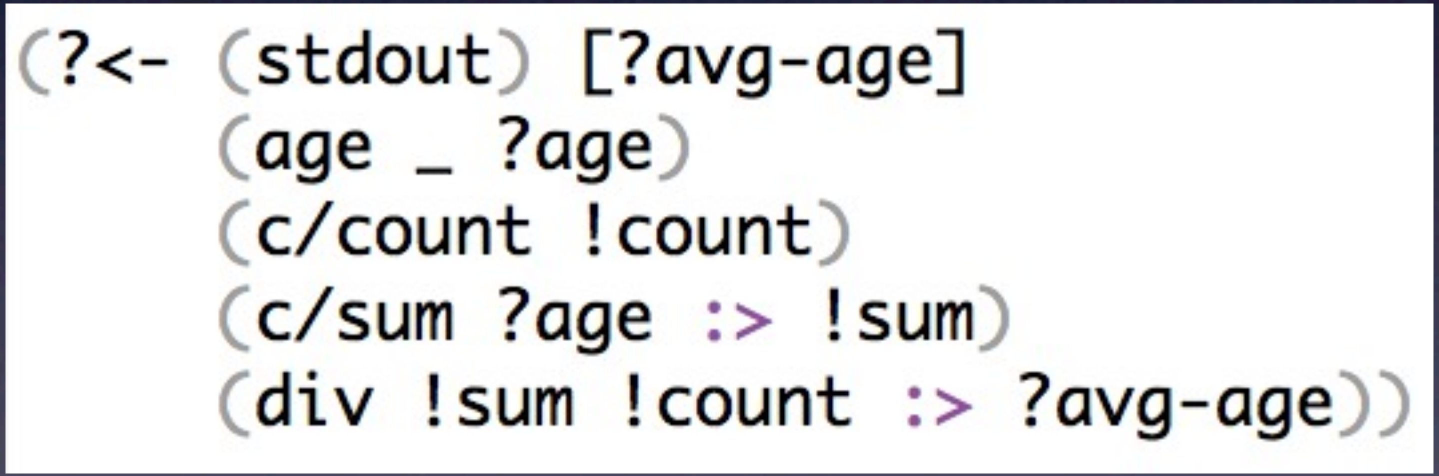
“Predicate macro”

Composability



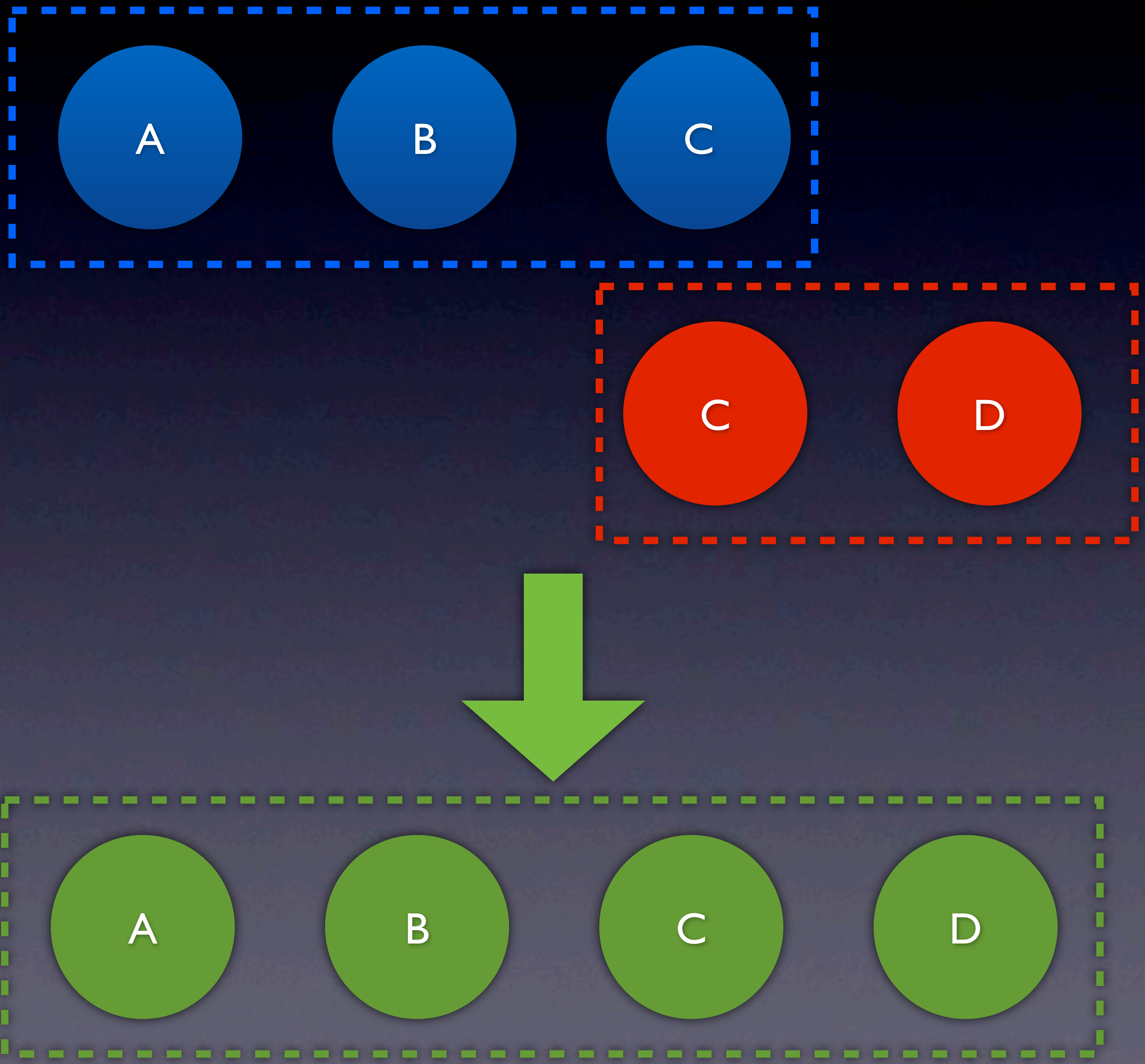
```
(?<- (stdout) [?avg-age]
     (age _ ?age)
     (avg ?age :> ?avg-age))
```

expands to



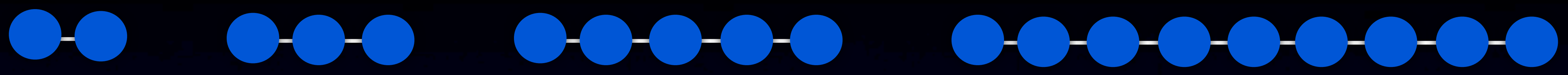
```
(?<- (stdout) [?avg-age]
     (age _ ?age)
     (c/count !count)
     (c/sum ?age :> !sum)
     (div !sum !count :> ?avg-age))
```

Attaching chains



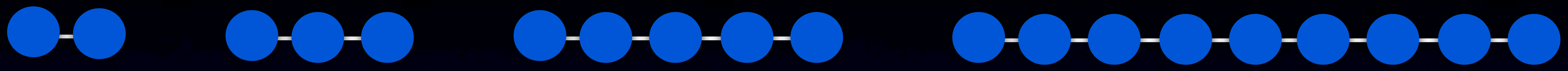
Chains

Chain



Chains

Chain

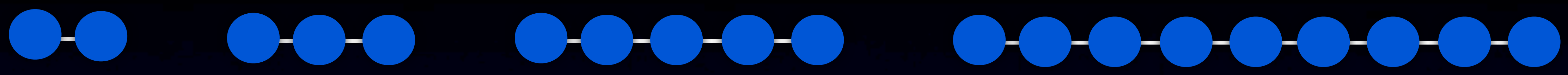


Reverse binary (length - 1)	0	1	1	

Chain of length 8

Chains

Chain



Reverse binary (length - 1)

0

1

Chain of length 3

Chains

Chain	2	3	5	9	17	33
Reverse binary (length - 1)	1	0	1	0	1	

Chain of length 22

Questions

http://github.com/nathanmarz/cascalog