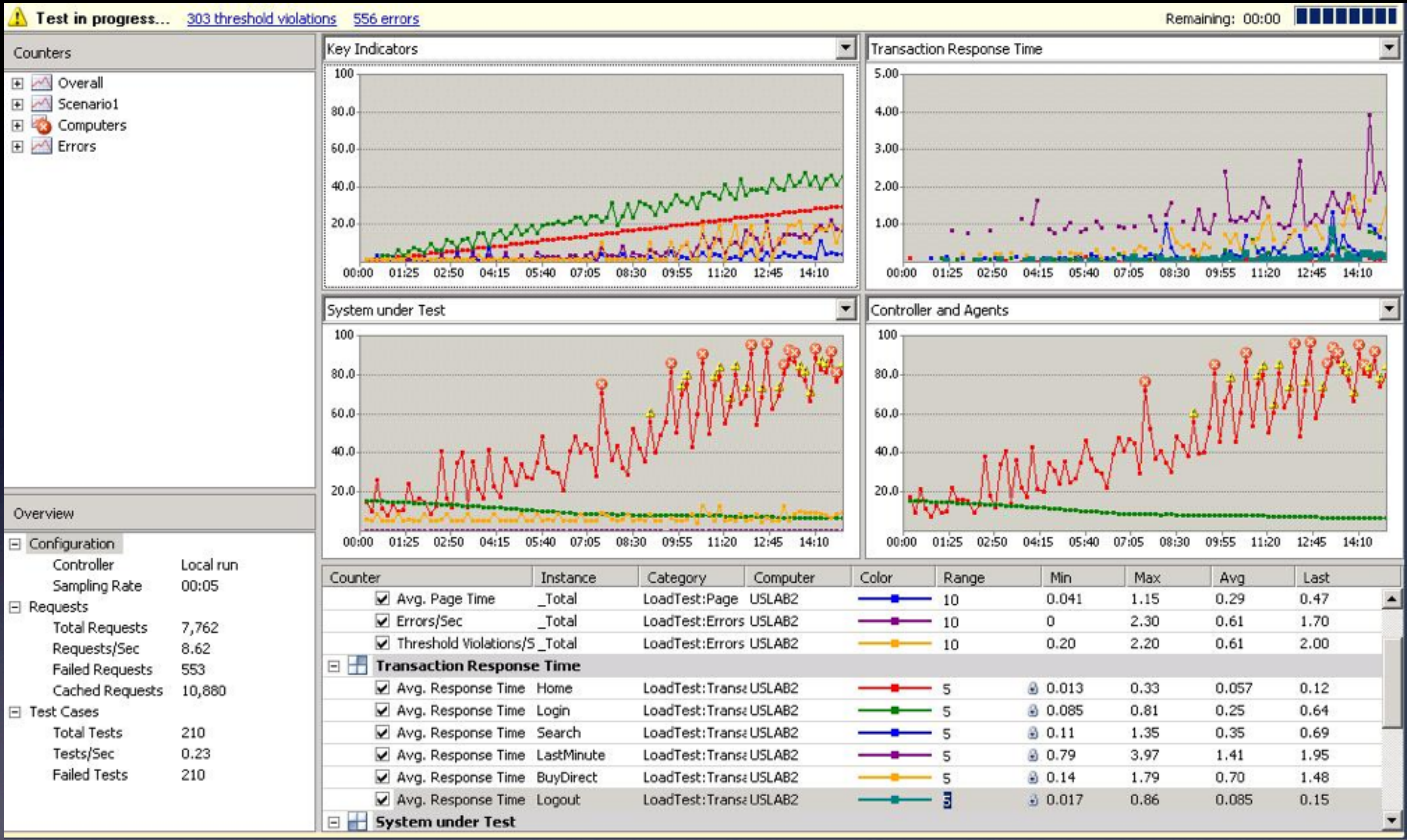
Simulation Testing with Datomic

Julian Gamble
@juliansgamble
fp-syd Aug 2013

The Business Problem

Types of Testing

- Unit Testing
- User Acceptance Testing
- Performance Testing



⚠ **Test in progress...** 303 threshold violations 556 errors

Remaining: 00:00

Counters

- ⊞ Overall
- ⊞ Scenario1
- ⊞ Computers
- ⊞ Errors

Key Indicators

Transaction Response Time

System under Test

Controller and Agents

Overview

⊟ Configuration	
Controller	Local run
Sampling Rate	00:05
⊟ Requests	
Total Requests	7,762
Requests/Sec	8.62
Failed Requests	553
Cached Requests	10,880
⊟ Test Cases	
Total Tests	210
Tests/Sec	0.23
Failed Tests	210

Counter	Instance	Category	Computer	Color	Range	Min	Max	Avg	Last
☑ Avg. Page Time	_Total	LoadTest:Page	USLAB2	🟦	10	0.041	1.15	0.29	0.47
☑ Errors/Sec	_Total	LoadTest:Errors	USLAB2	🟪	10	0	2.30	0.61	1.70
☑ Threshold Violations/S	_Total	LoadTest:Errors	USLAB2	🟧	10	0.20	2.20	0.61	2.00
⊟ **Transaction Response Time**									
☑ Avg. Response Time	Home	LoadTest:Transa	USLAB2	🟥	5	0.013	0.33	0.057	0.12
☑ Avg. Response Time	Login	LoadTest:Transa	USLAB2	🟩	5	0.085	0.81	0.25	0.64
☑ Avg. Response Time	Search	LoadTest:Transa	USLAB2	🟦	5	0.11	1.35	0.35	0.69
☑ Avg. Response Time	LastMinute	LoadTest:Transa	USLAB2	🟪	5	0.79	3.97	1.41	1.95
☑ Avg. Response Time	BuyDirect	LoadTest:Transa	USLAB2	🟧	5	0.14	1.79	0.70	1.48
☑ Avg. Response Time	Logout	LoadTest:Transa	USLAB2	■	5	0.017	0.86	0.085	0.15
⊟ **System under Test**									

The Business Problem

Types of Testing

- Unit Testing
- User Acceptance Testing
- Performance Testing
- Simulation Testing

The Business Problem

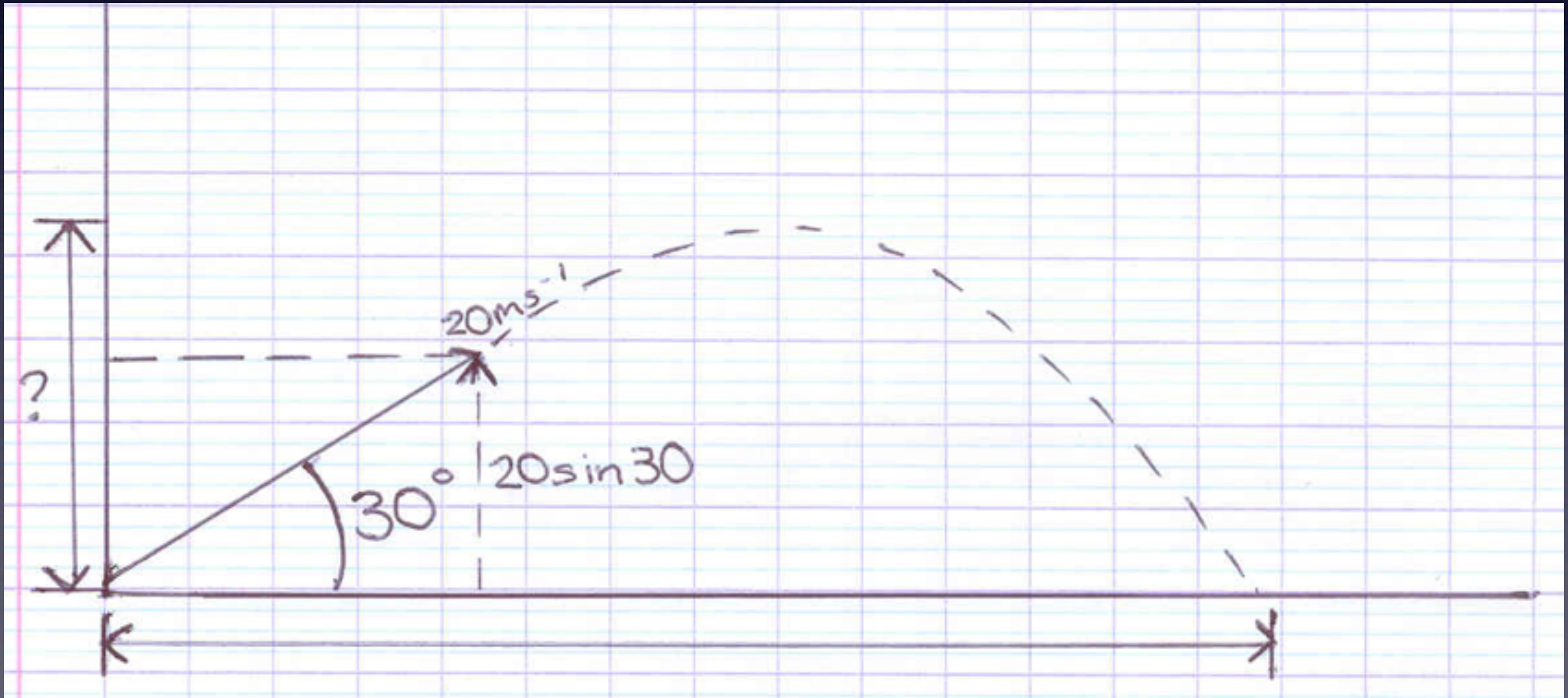
Generations of Simulations

- High School

The Business Problem

Generations of Simulations

- High School



The Business Problem

Generations of Simulations

- High School
- Stock Analysts

The Business Problem

Generations of Simulations

- High School
- Stock Analysts

The Business Problem

Generations of Simulations

- High School
- Stock Analysts
- Analytics driven audit

The Business Problem

Generations of Simulations

- High School
- Stock Analysts
- Analytics driven audit

The Business Problem

Generations of Simulations

- High School
- Stock Analysts
- Analytics driven audit
- Business scenarios

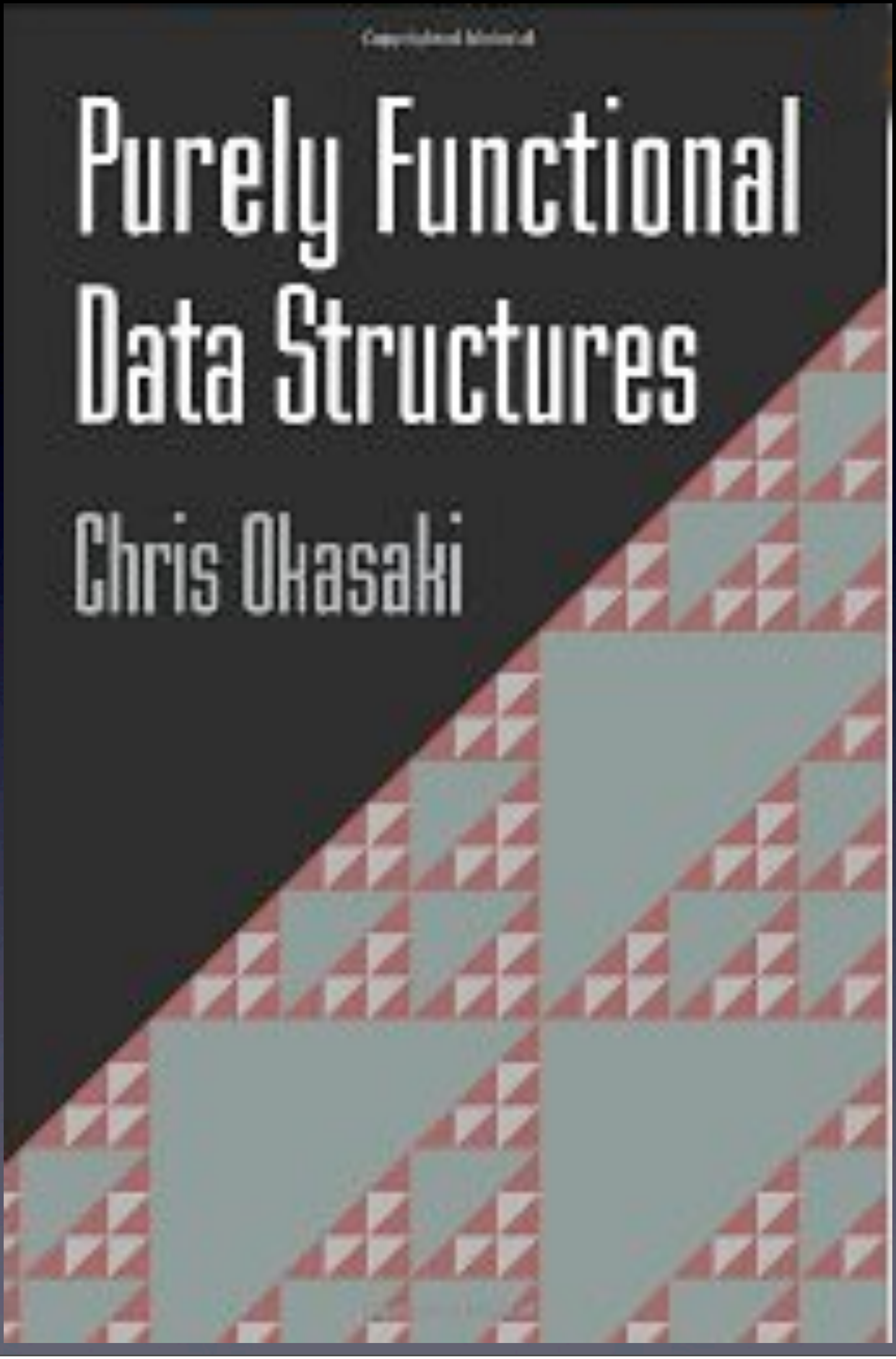
The Business Problem

Generations of Simulations

- High School
- Stock Analysts
- Analytics driven audit
- Business scenarios
- The next level

Applying functional programming

- Purely functional data structures



Copyrighted Material

Purely Functional Data Structures

Chris Okasaki

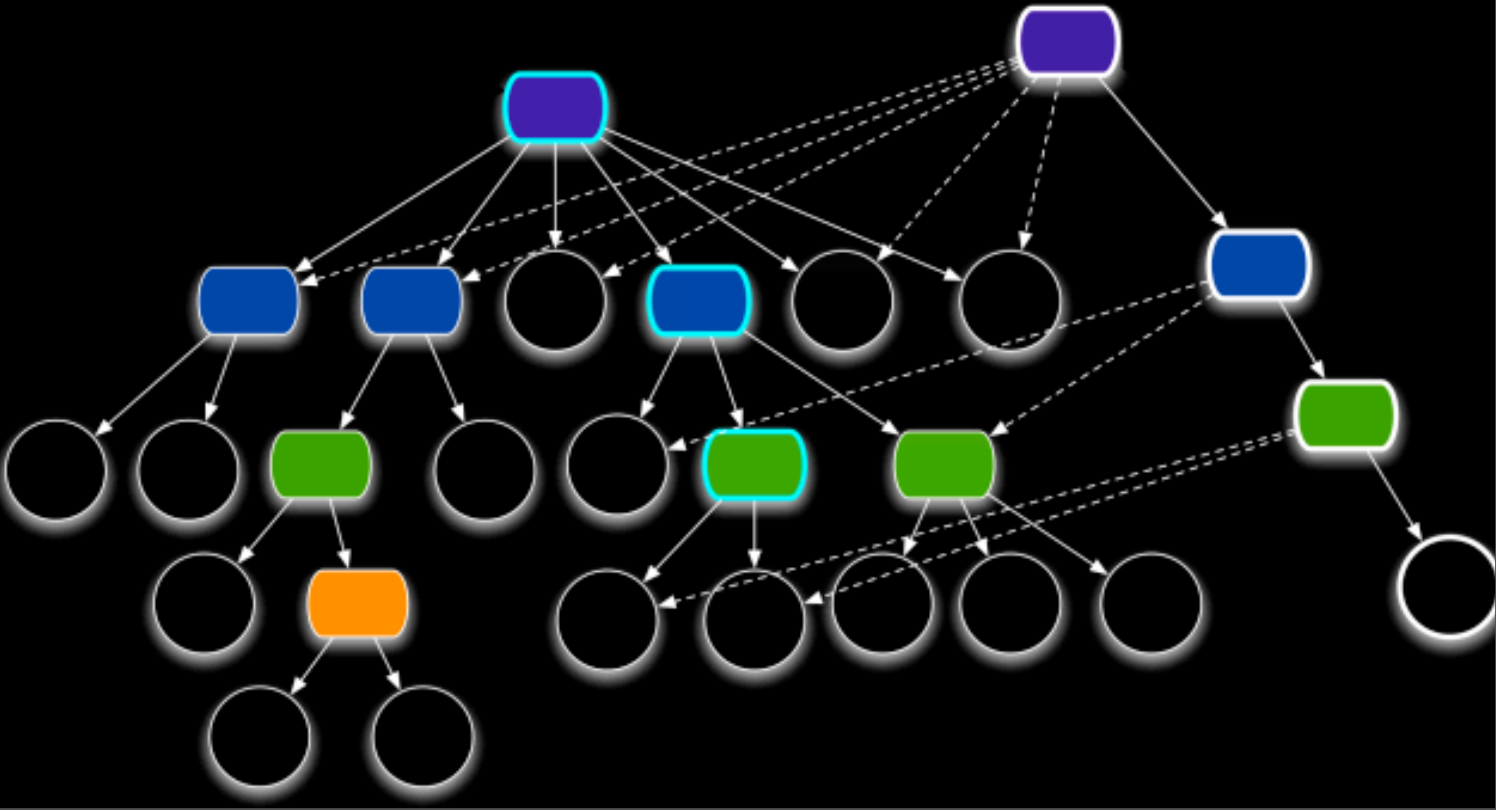
Persistent Data Structures

- Immutable, and old version of the collection is still available after 'changes'
- Collection maintains its performance guarantees for most operations
- Therefore new versions are not full copies
- Hash map and vector both based upon array mapped hash tries (Bagwell)
- Sorted map is red-black tree

Structural sharing

- Key to efficient ‘copies’ and therefore persistence
- Everything is final so no chance of interference
- Thread safe
- Iteration safe

Path copying



Next for persistent data structures?

Datomic

- “Database as a Value”
- Rich Hickey’s answer to the no-SQL world
- Append only database - extends concept of persistent data structures to the database scale
- Assumption that you can ‘think’ in key-values instead of table-rows

Business Problem

“I can’t reproduce your bug - I don’t know what the database state was at that point in time.”

[Database with a rewind button]

Implementation

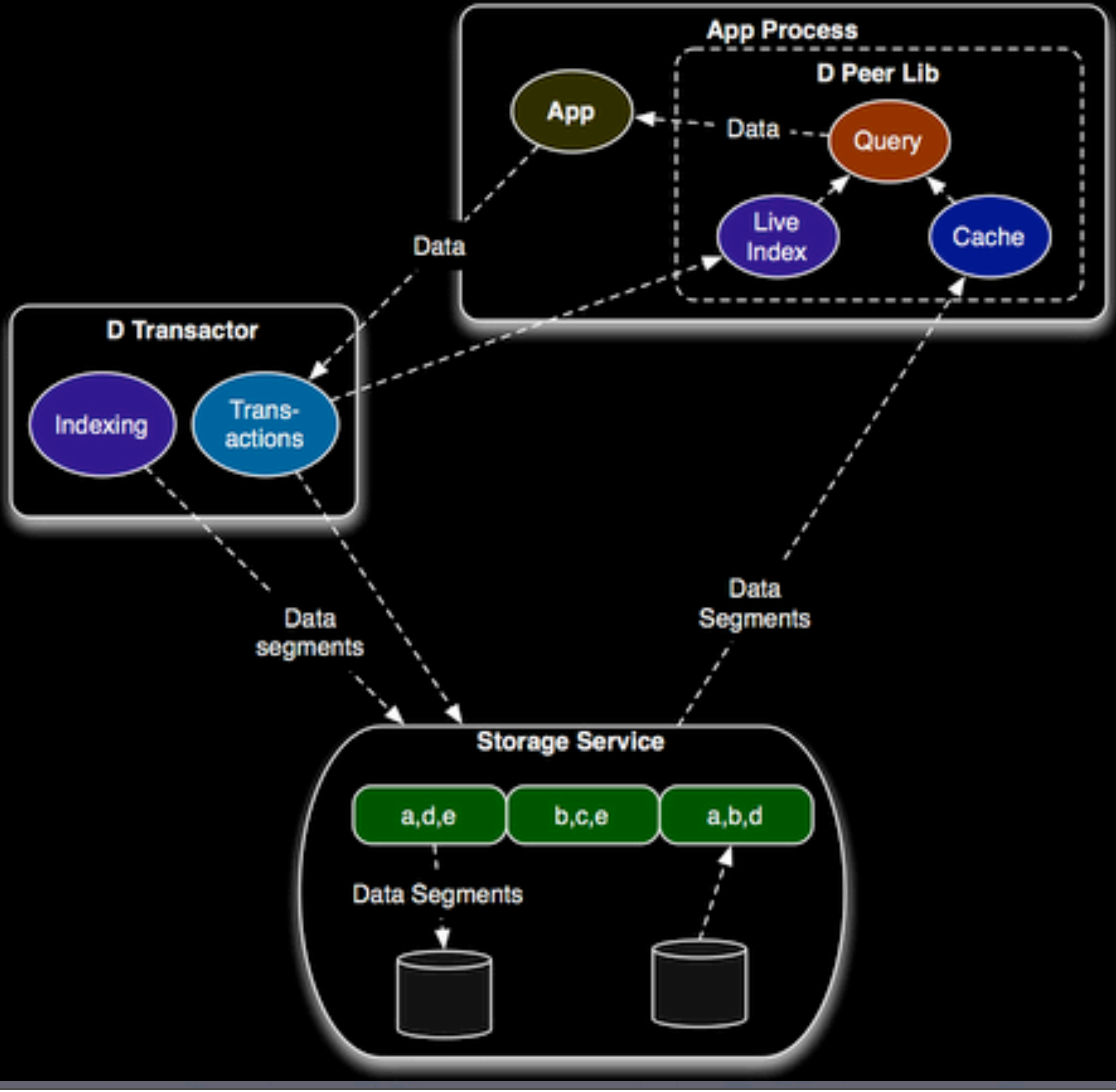
- Locally it runs on Hypersonic in Java
- It is designed to run on a variety of databases including Amazon's dynamo DB

  (in a sense it is a transaction layer over existing DB key-value implementations)
- the shell language is Beanshell

Transactions

- single writer model (the transactor)
- transactions are synchronous between the transactor and the storage (for atomicity)
- transactions can be asynchronous between your app and the transactor (ie you are returned computational futures - *Future*<*Map*>)
- transactions between app and transactor can also be synchronous

The Datomic Architecture



Back to the Problem

What does Datomic have to do with Simulation Testing?

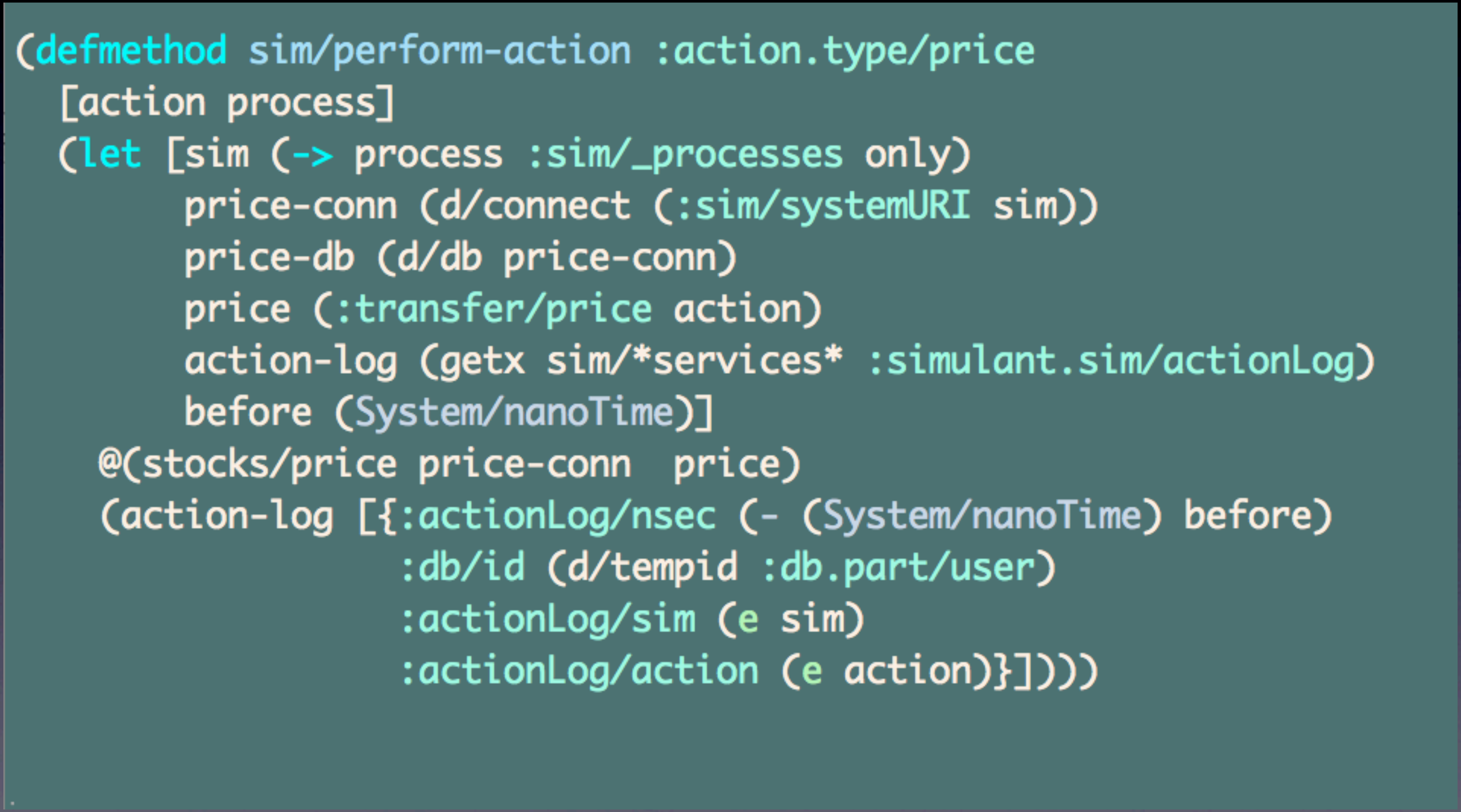
Simulant

- rigorous, scalable, and reproducible approach to simulation testing
- Assumptions - you have a testable model
- Phases of testing model - modeling, test data generation, simulation and validation
- Schema includes git file hashes at time of execution

Schema Assumptions

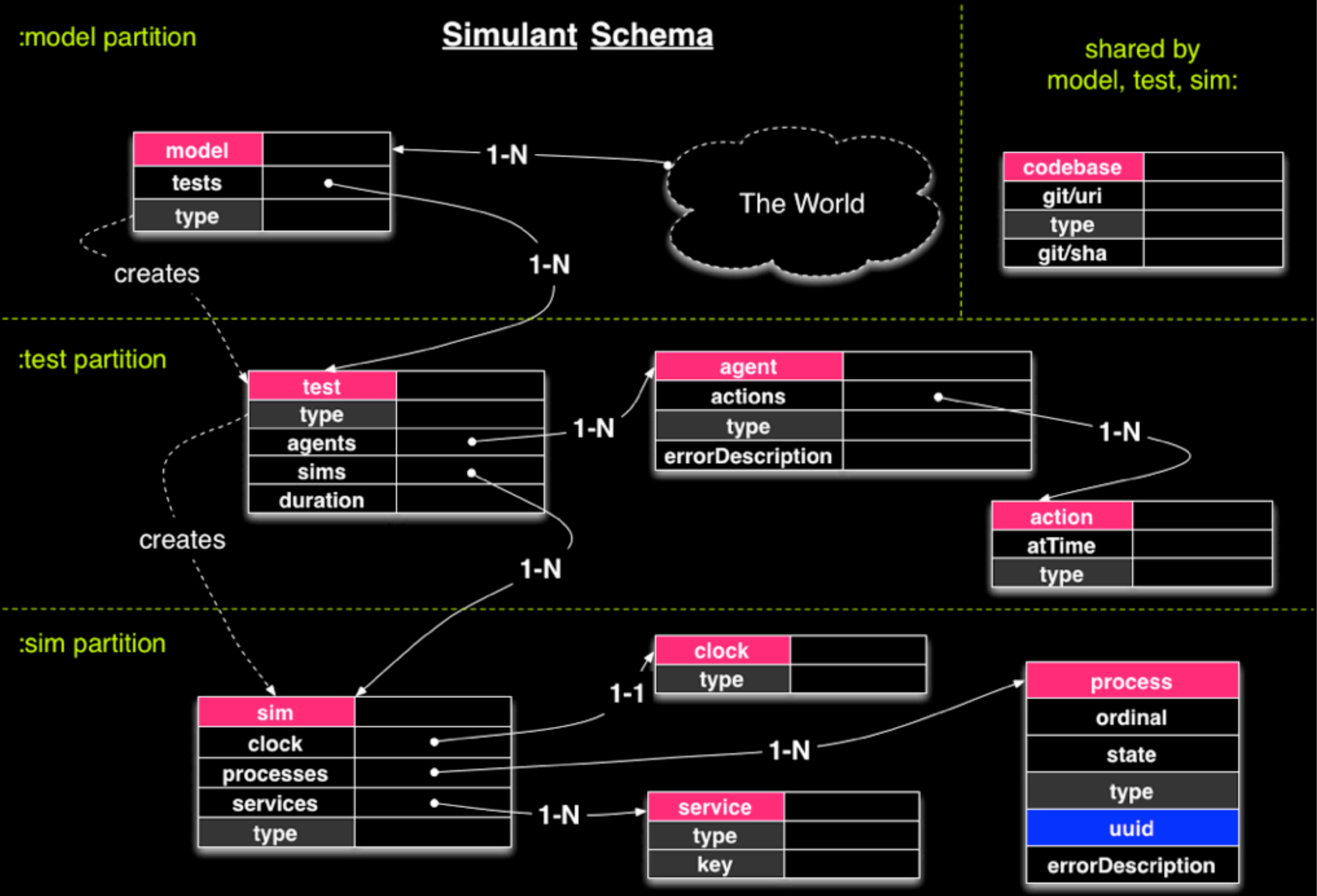
- You’re going to model *agents* and *actions* in the simulant schema - and the rest in your own schema

Schema Assumptions



```
(defmethod sim/perform-action :action.type/price
  [action process]
  (let [sim (-> process :sim/_processes only)
        price-conn (d/connect (:sim/systemURI sim))
        price-db (d/db price-conn)
        price (:transfer/price action)
        action-log (getx sim/*services* :simulant.sim/actionLog)
        before (System/nanoTime)]
    @(stocks/price price-conn  price)
    (action-log [{:actionLog/nsec (- (System/nanoTime) before)
                  :db/id (d/tempid :db.part/user)
                  :actionLog/sim (e sim)
                  :actionLog/action (e action)}])))
```

Simulant Schema



:model partition

shared by
model, test, sim:

model	
tests	
type	

The World

1-N

1-N

creates

codebase	
git/uri	
type	
git/sha	

:test partition

test	
type	
agents	
sims	
duration	

1-N

agent	
actions	
type	
errorDescription	

1-N

action	
atTime	
type	

creates

1-N

:sim partition

sim	
clock	
processes	
services	
type	

1-1

clock	
type	

1-N

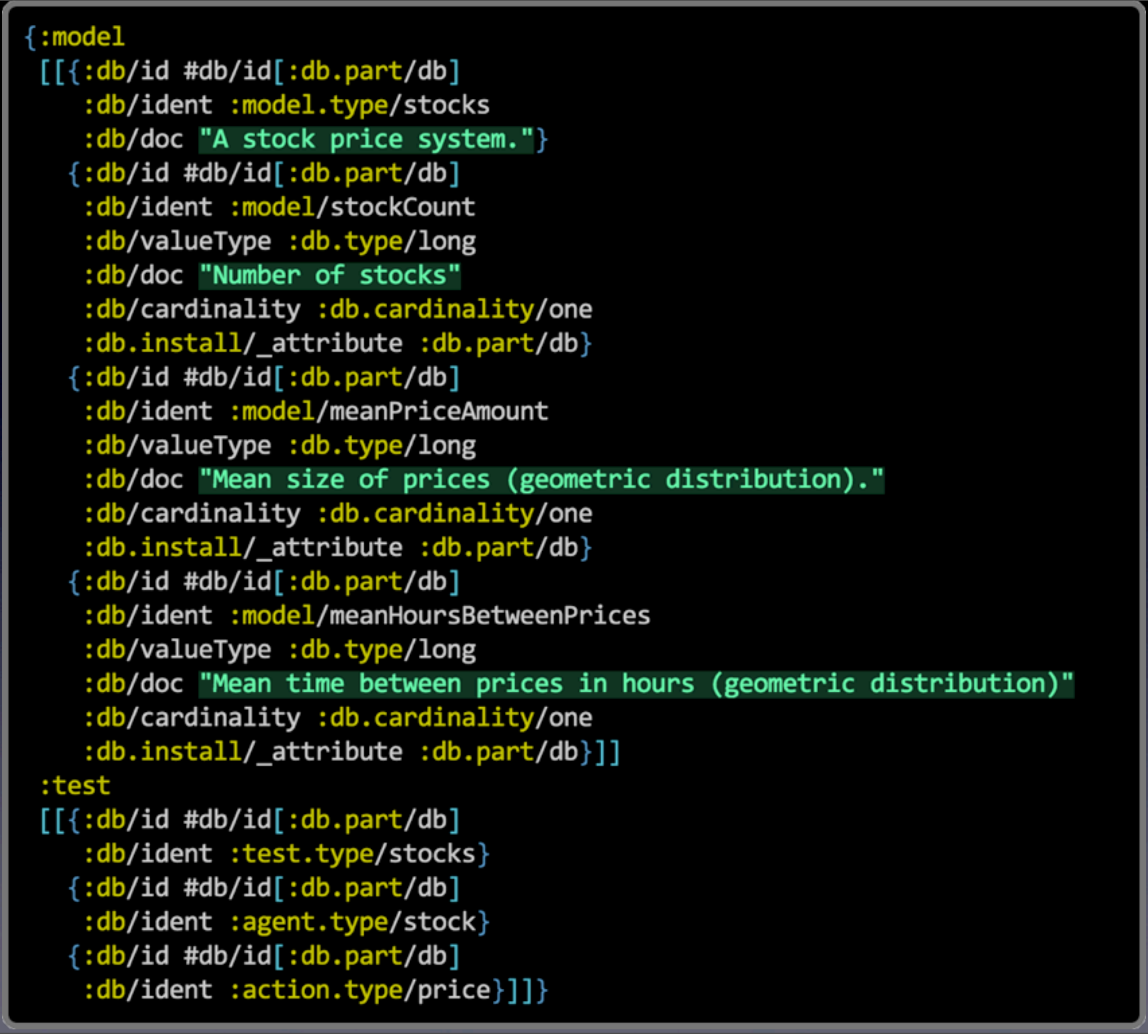
process
ordinal
state
type
uuid
errorDescription

1-N

service	
type	
key	

Process

- Create a datomic schema for your model



```
{:model
 [[{:db/id #db/id[:db.part/db]
    :db/ident :model.type/stocks
    :db/doc "A stock price system."}
   {:db/id #db/id[:db.part/db]
    :db/ident :model/stockCount
    :db/valueType :db.type/long
    :db/doc "Number of stocks"
    :db/cardinality :db.cardinality/one
    :db.install/_attribute :db.part/db}
   {:db/id #db/id[:db.part/db]
    :db/ident :model/meanPriceAmount
    :db/valueType :db.type/long
    :db/doc "Mean size of prices (geometric distribution)."
    :db/cardinality :db.cardinality/one
    :db.install/_attribute :db.part/db}
   {:db/id #db/id[:db.part/db]
    :db/ident :model/meanHoursBetweenPrices
    :db/valueType :db.type/long
    :db/doc "Mean time between prices in hours (geometric distribution)"
    :db/cardinality :db.cardinality/one
    :db.install/_attribute :db.part/db}]]
 :test
 [[{:db/id #db/id[:db.part/db]
    :db/ident :test.type/stocks}
   {:db/id #db/id[:db.part/db]
    :db/ident :agent.type/stock}
   {:db/id #db/id[:db.part/db]
    :db/ident :action.type/price}]]}
```

Process

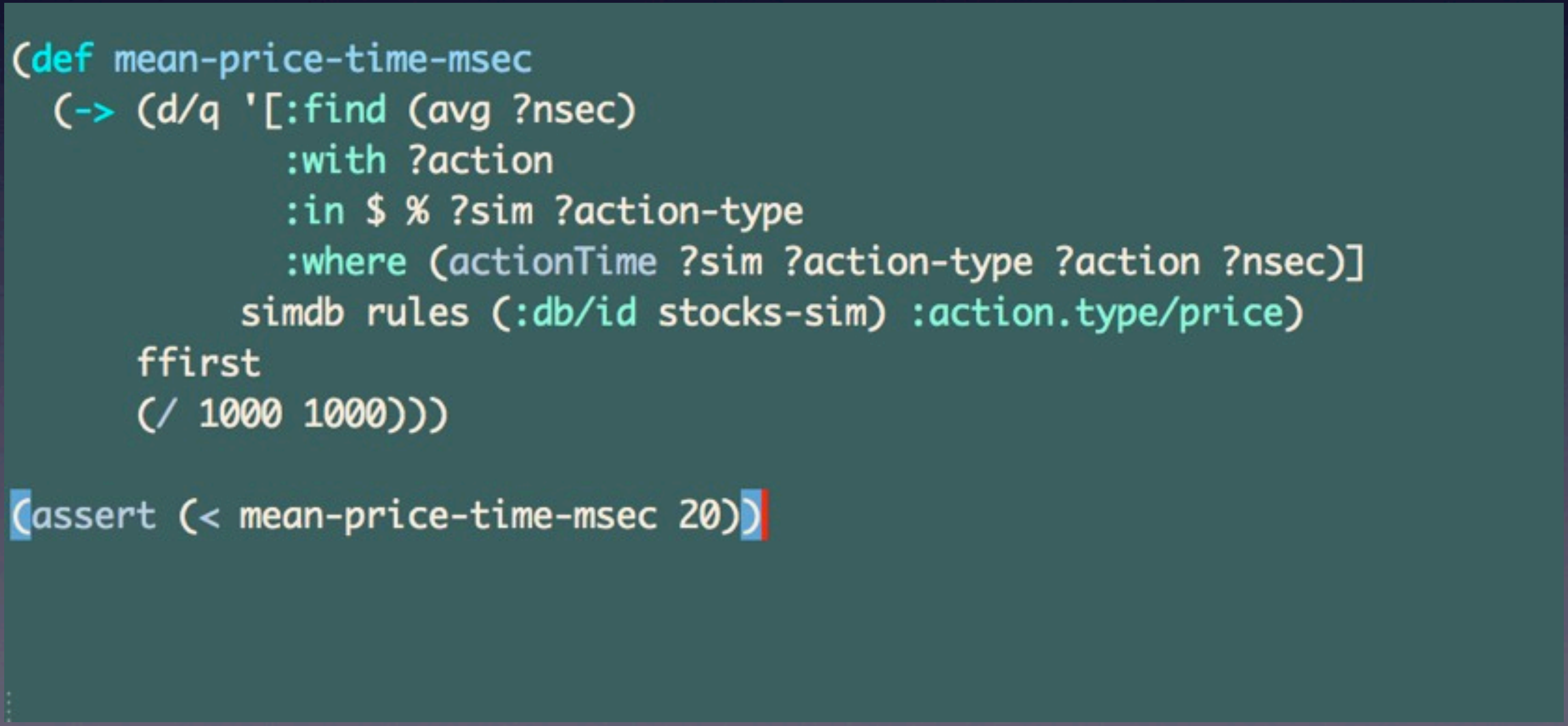
- Create a datomic schema for your model
- Set the model parameters
  - stocks/prices/portfolios/market assumptions
  - ants/food/world size

Process

- Create a datomic schema for your model
- Set the model parameters
  - stocks/prices/portfolios/market assumptions
  - ants/food/world size
- Run the simulation

Benefits

- Make statistical assertions about the system using the testing framework



```
(def mean-price-time-msec
  (-> (d/q '[:find (avg ?nsec)
             :with ?action
             :in $ % ?sim ?action-type
             :where (actionTime ?sim ?action-type ?action ?nsec)]
           simdb rules (:db/id stocks-sim) :action.type/price)
      ffirst
      (/ 1000 1000)))

(assert (< mean-price-time-msec 20))
```

Why is it better than a relational DB?

- You can go back to a point in time in your simulation - change the parameters - and ‘fork’ a different path (you *could* do this in a relational database with extra schema - but this is less work)
- point in time queries are very natural
- scalability

Applicability

- Assumption is you have a system sufficiently complex to require simulation (non-linear problem - multiple agents)
- Where you need a rich history of the state changes - very scalable - even across multiple versions of the software
- Strength is really in the statistics

Summary

- Simulation testing - testing your (non-trival) system against a model
- ‘Database as a value’ - a database with a rewind button
- Datomic is good for working with information rich scenarios - where state and time are linked
- Simulant - framework for statistical analysis

Questions