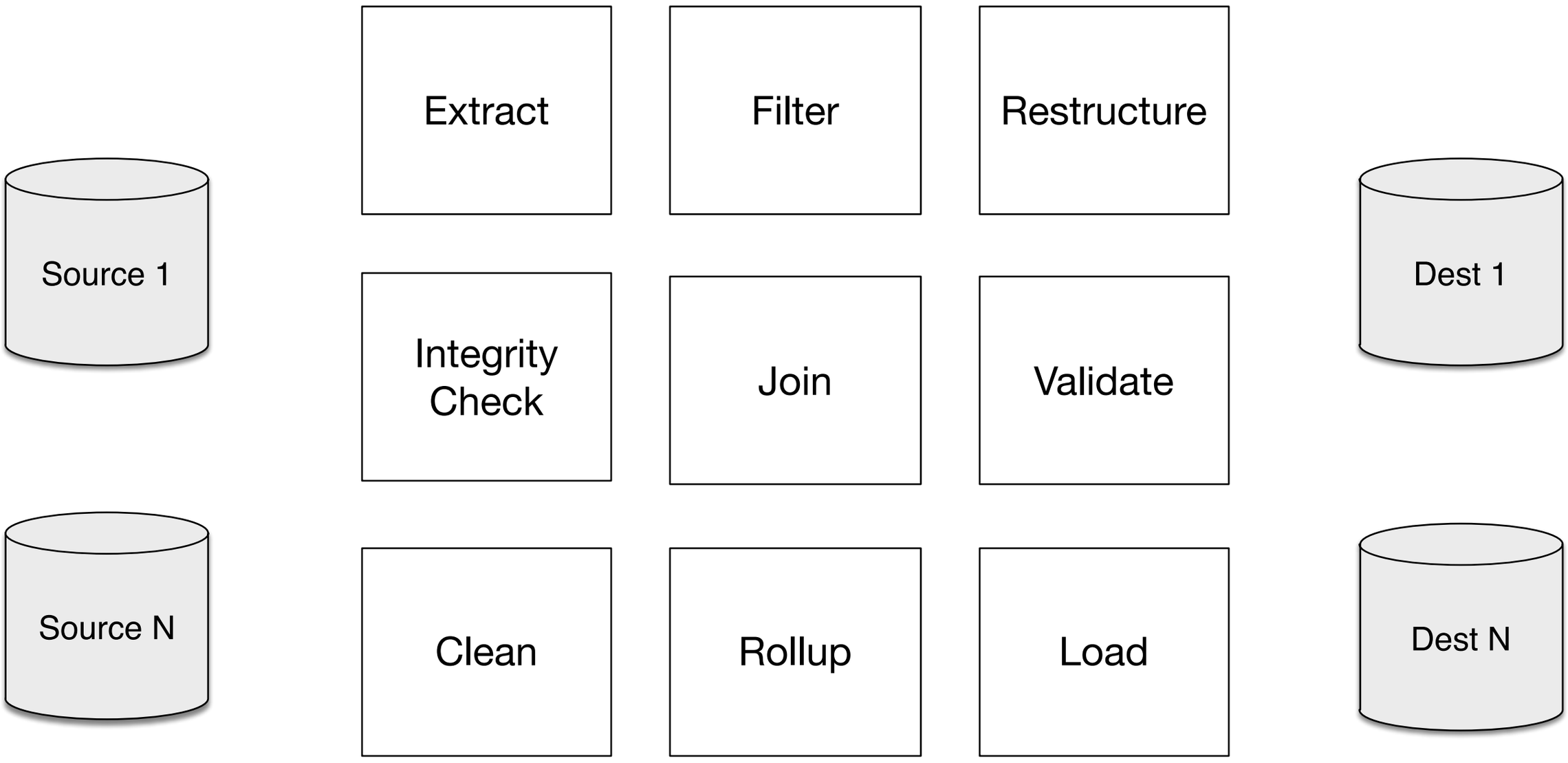
ETL with Clojure and Datomic

@stuarthalloway

Datomic

extract, transform, load

or maybe


Extract
Filter
Restructure
Source 1
Integrity Check
Join
Validate
Dest 1
Source N
Clean
Rollup
Load
Dest N

exact steps don’t matter

how you decompose the steps does

functional pipeline

decouples steps

this is both semantic and operational

facilitates

parallelization

durability

checkpoint and restart

feedback

obvious advantages

power of the JVM

interactive, dynamic development

data-oriented functional programming

spec

obvious advantages

power of the JVM

interactive, dynamic development

data-oriented functional programming

spec

systemic generality

generality

all domains use the same general-purpose data structures and functions

systemic

in libs, in apps, in config, on wires, at rest

information

Paradigm	Examples	Approach	Correctness	Reach
Enterprise OO	Java C# C++	encapsulated specificity	types	libs, apps
Agile Scripting	Ruby Python JavaScript	encapsulated specificity	tests	libs, apps
	Clojure	systemic generality	functional, specs	systemic

obvious advantages

power of the JVM

interactive, dynamic development

data-oriented functional programming

spec

spec

integrated language discipline for *a la carte* specificity

without sacrificing generality

dynamic leverage

anytime

anywhere

up to you

correct / agile / robust

	Example Tests	Types	Spec
expressive	very	varying	very
powerful	stakeholder correctness	type correctness	stakeholder correctness
integrated	rare	compile-time, must flow	dynamic
specification	no	static	yes
testing	manual	rare	generative
agility	expensive	fragility	dynamic
reach	expensive	libs, apps	systemic

input entity specs

```
(s/def ::artist-ent
  (s/keys :req-un [::gid ::sortname ::name]
          :opt-un [::type ::gender ::country]))
```

```
;; ::begin_date_year ::begin_date_month ::begin_date_day
;; ::end_date_year ::end_date_month ::end_date_day
```

qualified key specs
could check these too

conformance check

```
(defn conform!
  "Like s/conform, but throws an error with s/explain-data on failure."
  ([spec x]
     (conform! spec x ""))
  ([spec x msg]
     (let [conformed (s/conform spec x)]
       (if (= ::s/invalid conformed)
         (throw (ex-info (str "Failed to conform " spec ", see ex-data")
                         {:data (s/explain-data spec x)
                          :value x}))
         conformed))))
```

less obvious advantages

transducers

strong namespaces

reified transactions

universal schema

less obvious advantages

transducers

strong namespaces

reified transactions

universal schema

transducers

decouple transformation from input or output sources

ideal for functional pipelines

make incidental complexity more evident

removal of inputs and outputs leaves fewer places to hide

make commonality more evident

de-structure task abstractions

http://clojure.org/reference/transducers

emergent simplicity

different details

```
(input-data->tx-data
 [this type]
 (case type
       :schema cat
       :enums enums->tx-data
       :super-enums super-enums->tx-data
       :artists (map #(transform-entity this % artist-attrs))
       :areleases (map #(transform-entity this % arelease-attrs))
       :releases (map #(transform-entity this % release-attrs))
       :labels (map #(transform-entity this % label-attrs))
       :media (map
               (fn [ent]
                 (assoc (transform-entity this ent medium-attrs)
                   :medium/tracks
                   (transform-entity this ent track-attrs))))
       :releases-artists (map #(transform-entity this % release-artist-attrs))
       :areleases-artists (map #(transform-entity this % arelease-artist-attrs))))
```

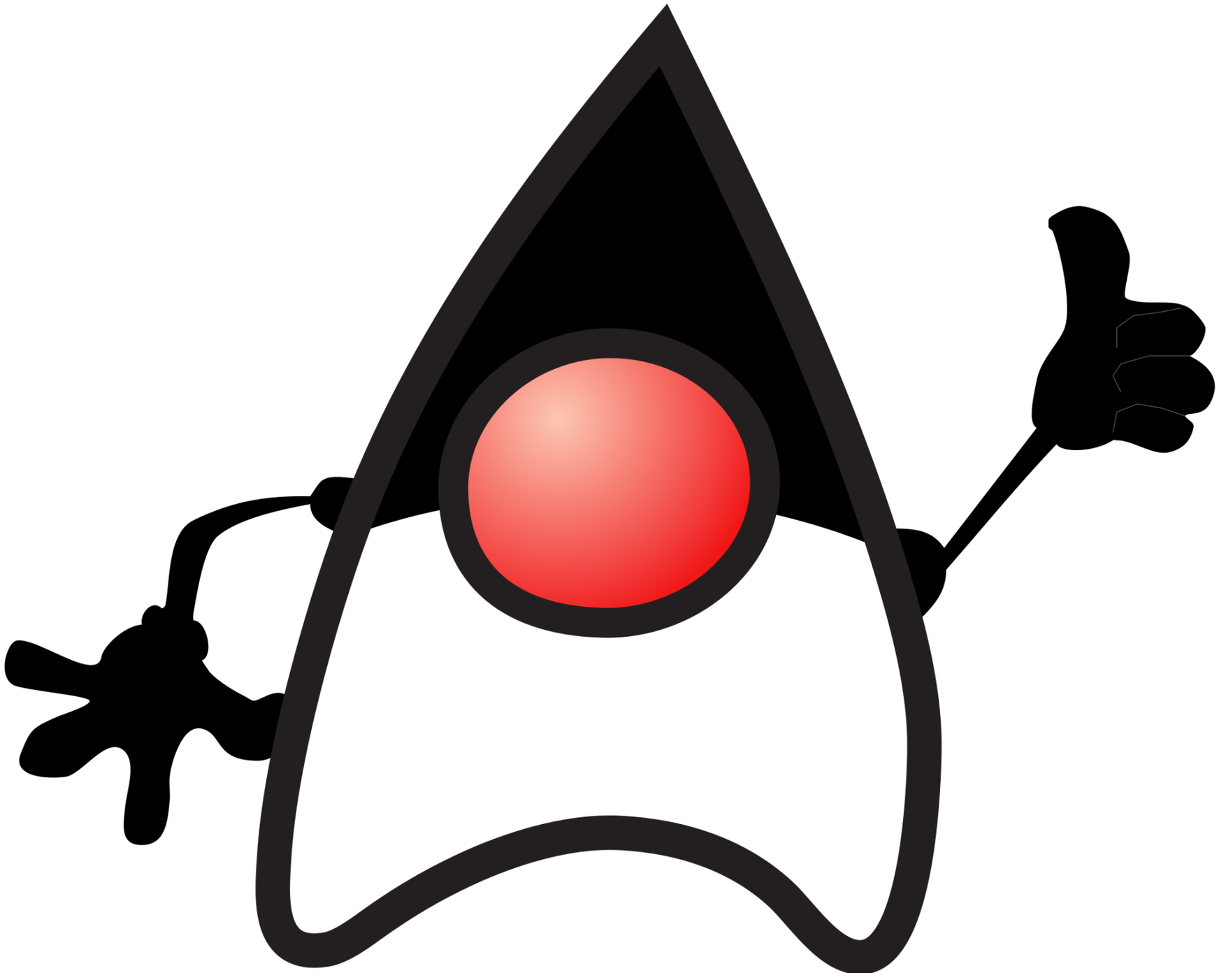
one “shape”:
transducing

this reusable
helper emerged

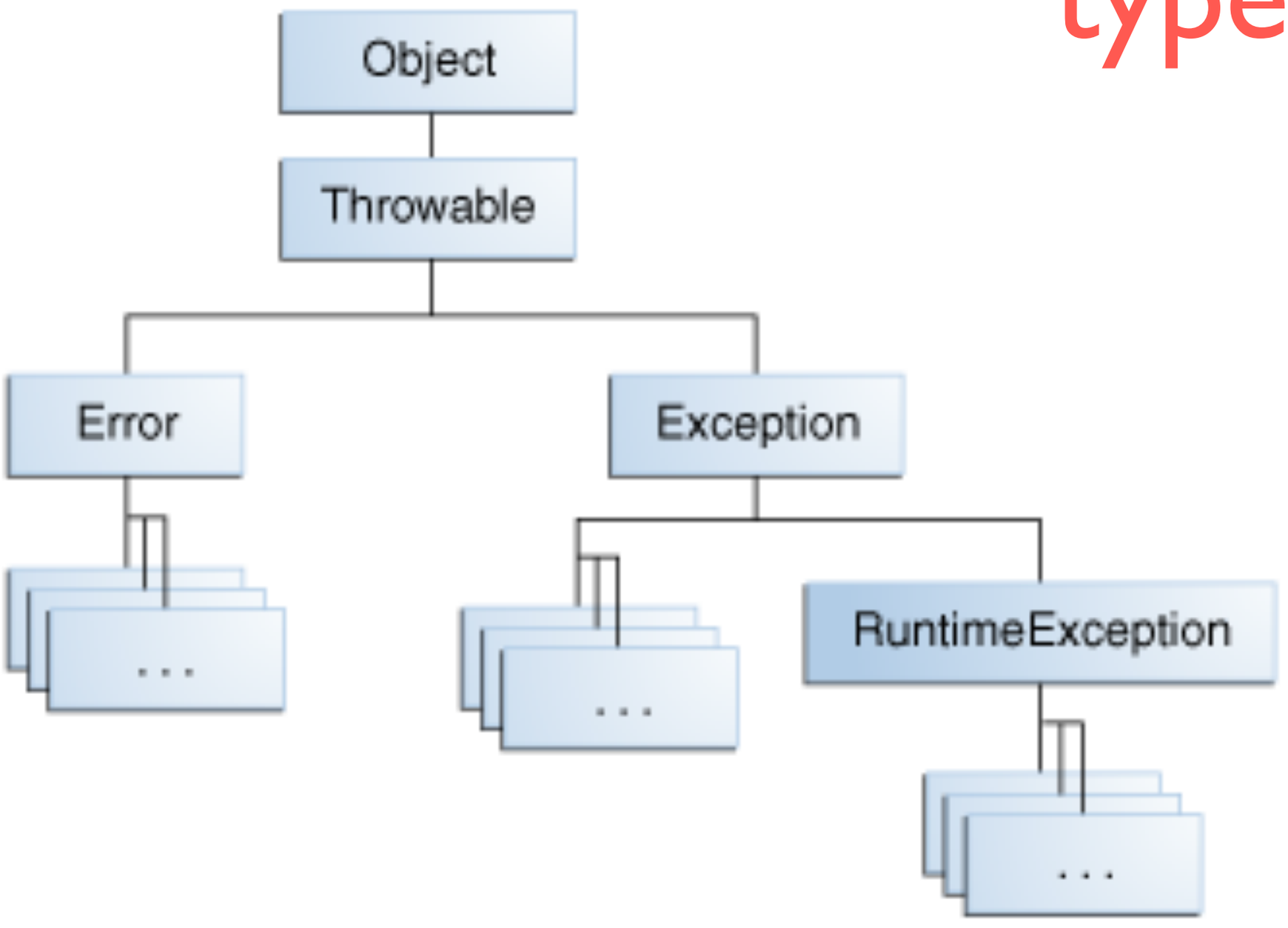
halt-when

```
(comp
 (map #(<!! (client/transact conn {:tx-data %})))
 (halt-when client/error?))
```

stop import on error
return error as information



type bias



https://docs.oracle.com/javase/tutorial/figures/essential/exceptions-throwable.gif

By United States Senate Historical Office - http://bioguide.congress.gov/scripts/biodisplay.pl?index=G000321, Public Domain, https://commons.wikimedia.org/w/index.php?curid=17706778

place bias

code	meaning
1xx	information
2xx	success
3xx	redirection
4xx	client error
5xx	server error

decision bias

category	Hall and Oates song
unavailable	Out of Touch
interrupted	It Doesn't Matter Anymore
incorrect	You'll Never Learn
forbidden	I Can't Go For That
unsupported	Your Imagination
not-found	She's Gone
conflict	Give It Up
fault	Falling
busy	Wait For Me

less obvious advantages

transducers

strong namespaces

reified transactions

universal schema

strong namespaces

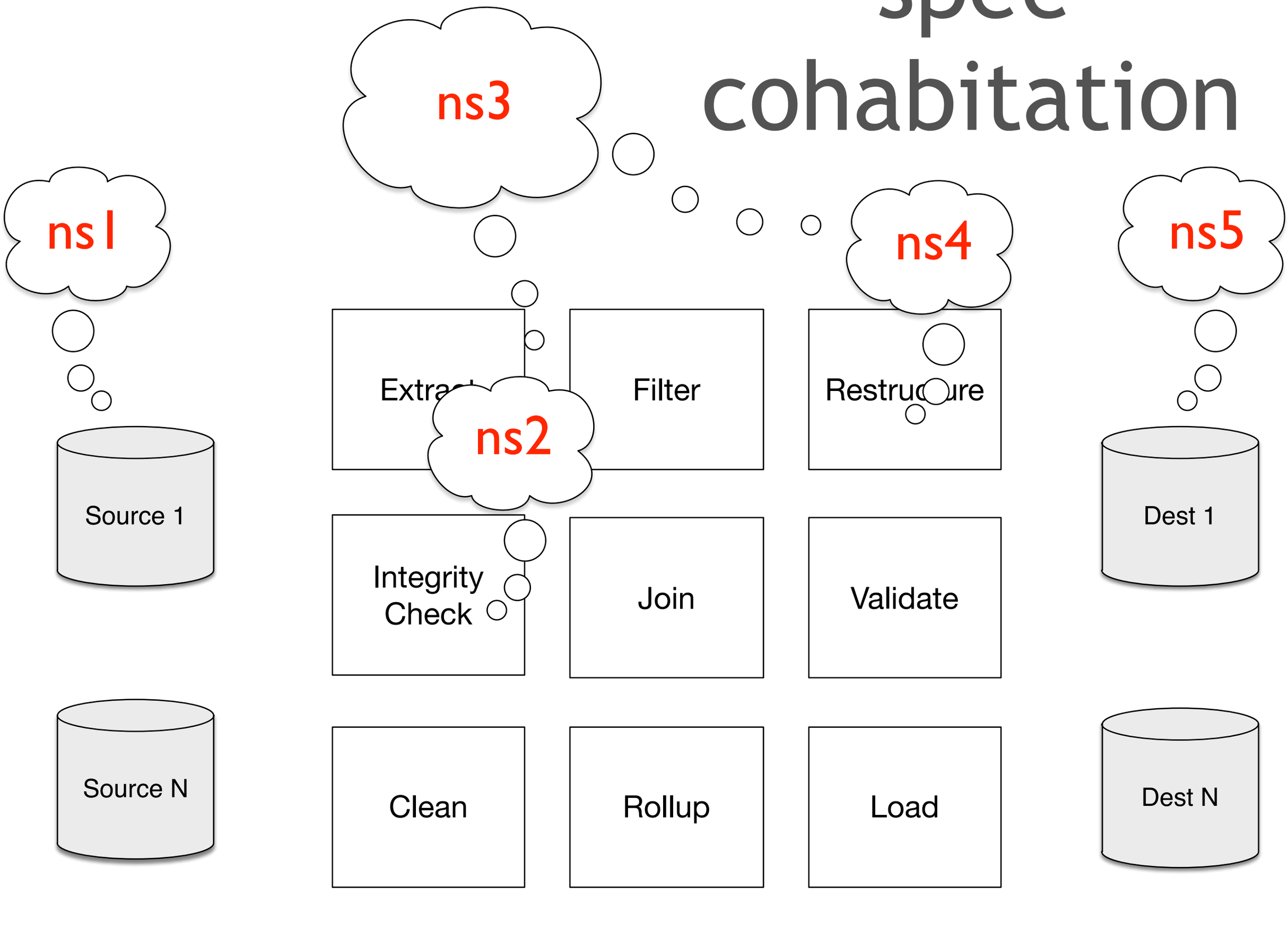
systemic

- at the bottom, in names themselves

connected to spec

literally easy to use

spec cohabitation



less obvious advantages

transducers

strong namespaces

reified transactions

universal schema

reified transactions

entities like any other entity in the system

associated with every datom

have a `:db/txInstant`

have any other attributes you specify

have their own index (the log)

transaction attributes

identifies transaction entity

```
[{:db/id "datomic.tx"
  :db/txInstant #inst "2013-02"
  :mbrainz.initial-import/batch-id “artists-1"}
 ...]
```

establishes provenance
tracks progress
makes import restartable
supports parallel and pipelined txes

less obvious advantages

transducers

strong namespaces

reified transactions

universal schema

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

universal schema benefits

logical schema ≈ physical schema

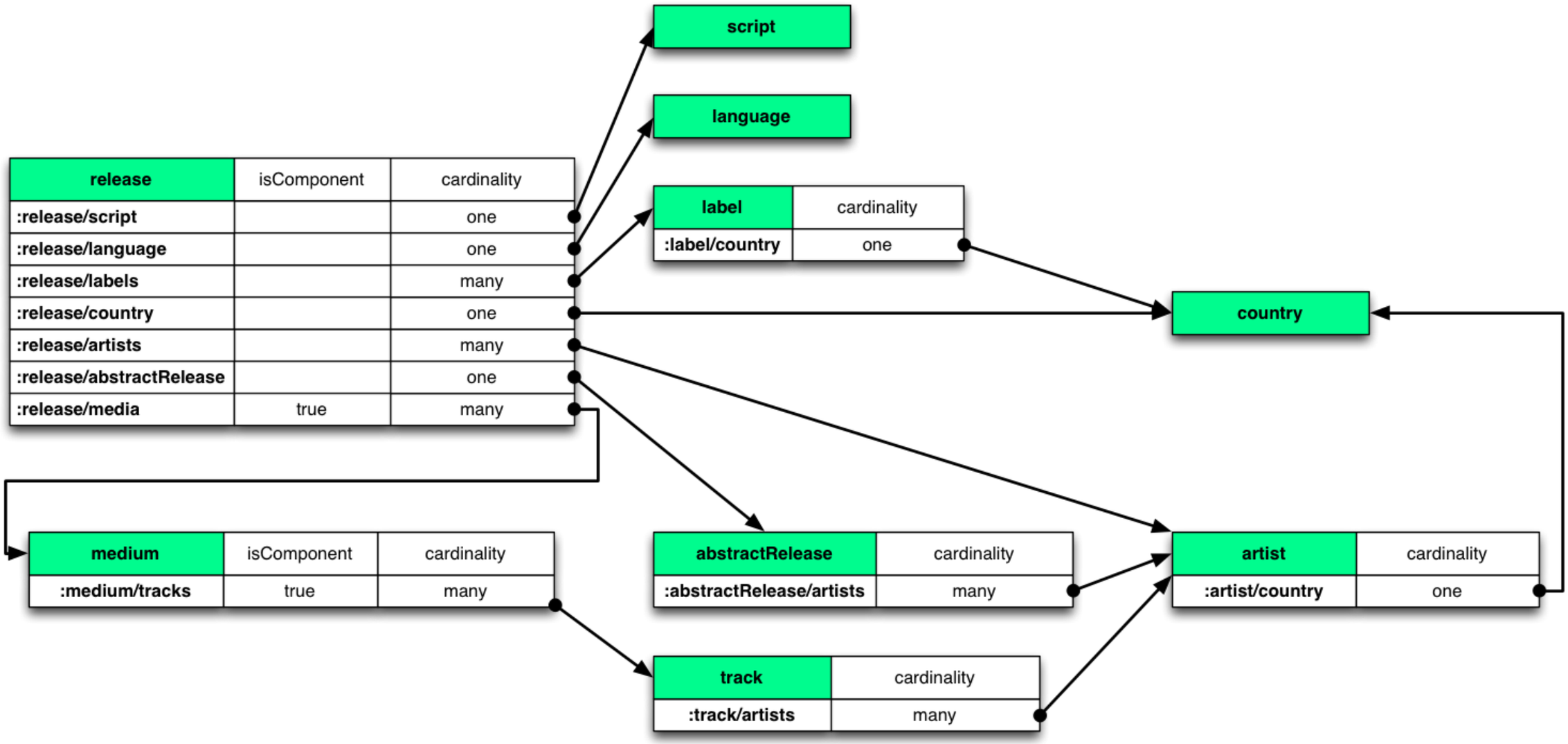
e.g. no join tables

store information, not query answers

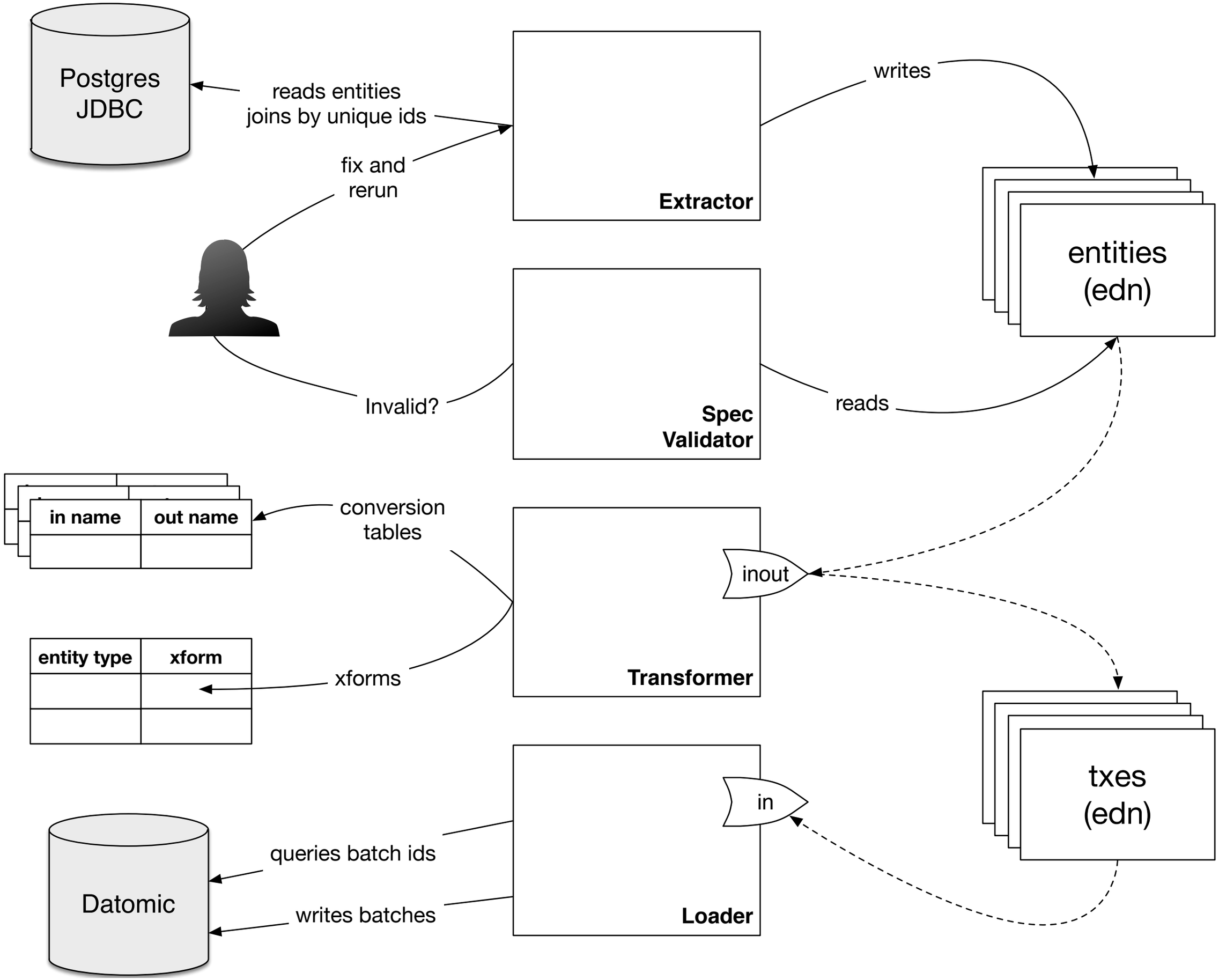
no denormalization

no “table per query pattern”

mbrainz



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


Postgres
JDBC
reads entities
joins by unique ids
fix and
rerun
Extractor
writes
entities
(edn)
Invalid?
Spec
Validator
reads
in name
out name
conversion
tables
inout
entity type
xform
xforms
Transformer
txes
(edn)
in
queries batch ids
Datomic
writes batches
Loader

lean and simple

task	LOC
general purpose ETL helpers	100
Datomic ETL helpers	50
data tables	100
specs	50
mbrainz transducers and wiring	150

ETL with Clojure and Datomic

@stuarthalloway

Datomic