Reactive Datalog for Datomic

by Nikolas Göbel

“We want to make reactive systems that don't poll. And we want those systems to get a consistent view of the world.”

– *Rich Hickey, “Deconstructing the Database”*

Nikolas Göbel

niko@clockworks.io

in collaboration with:

Frank McSherry (ETH)

David Bach (Clockworks)

Malte Sandstede (Clockworks)

ETH zürich

Systems Group

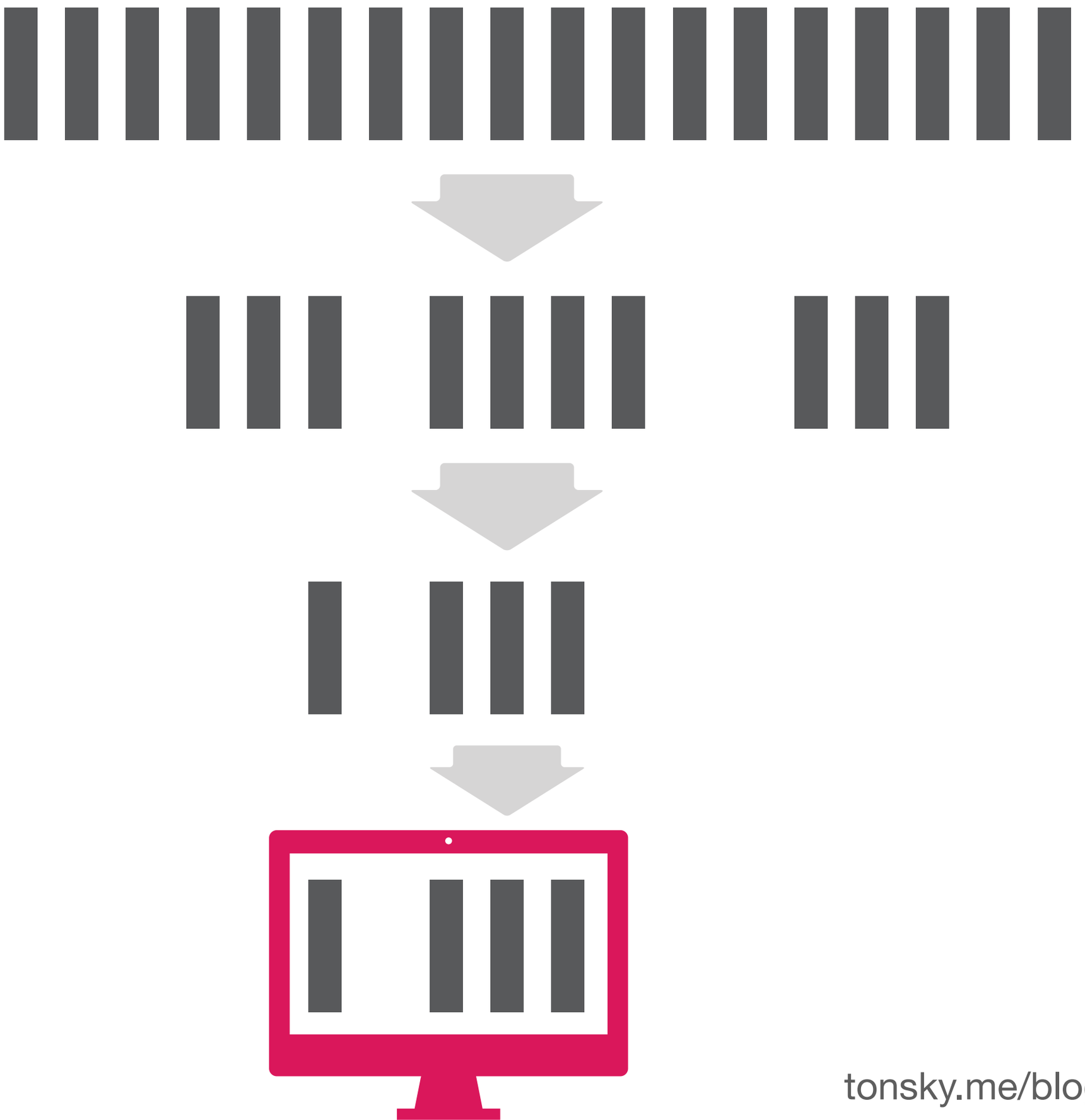
clockworks

Reactive Systems?

- live-updating web applications
- alerting and real-time dashboards
- stream processors
- rule engines

Reactive Systems?

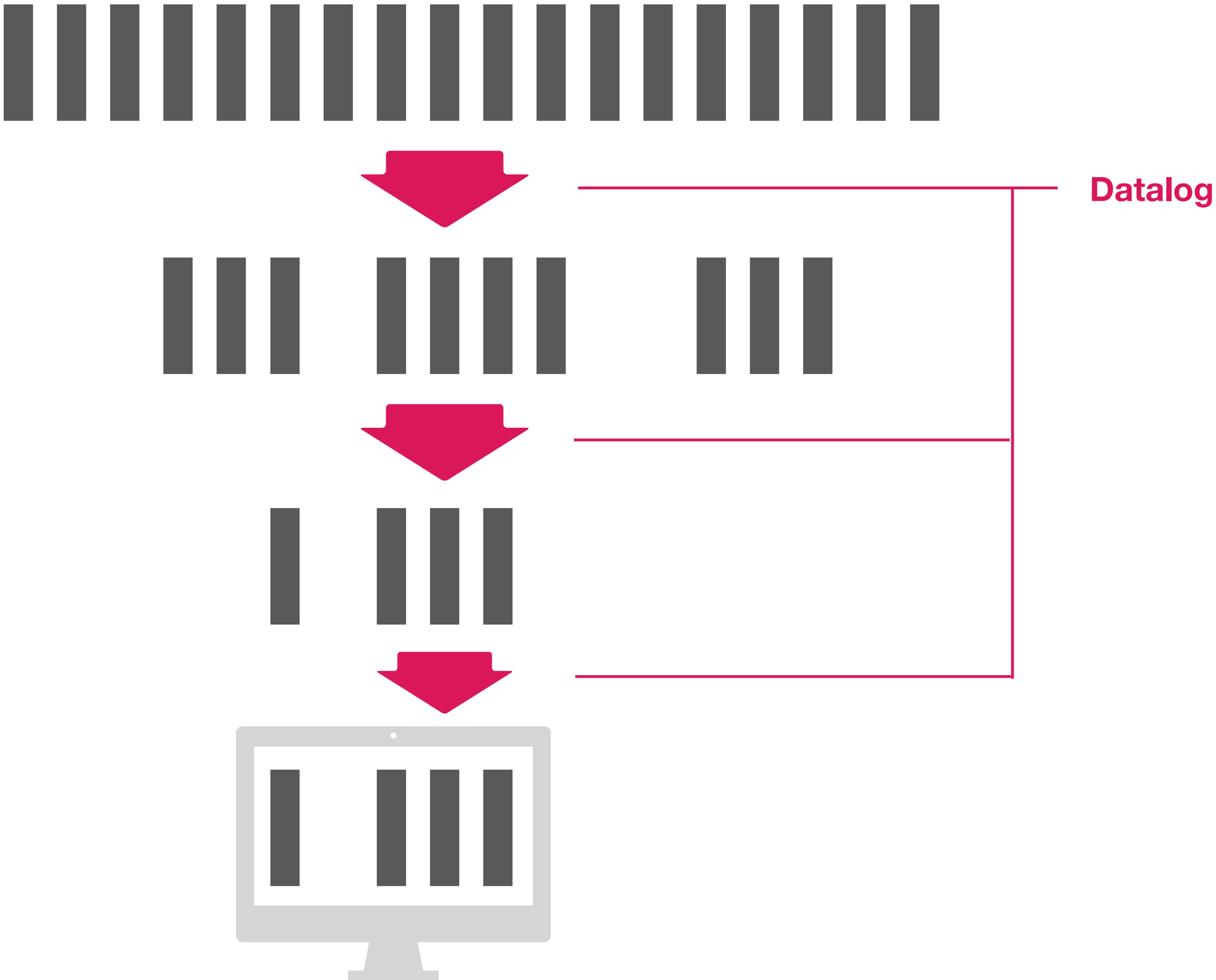
The arrival of data coordinates code, not the other way around.

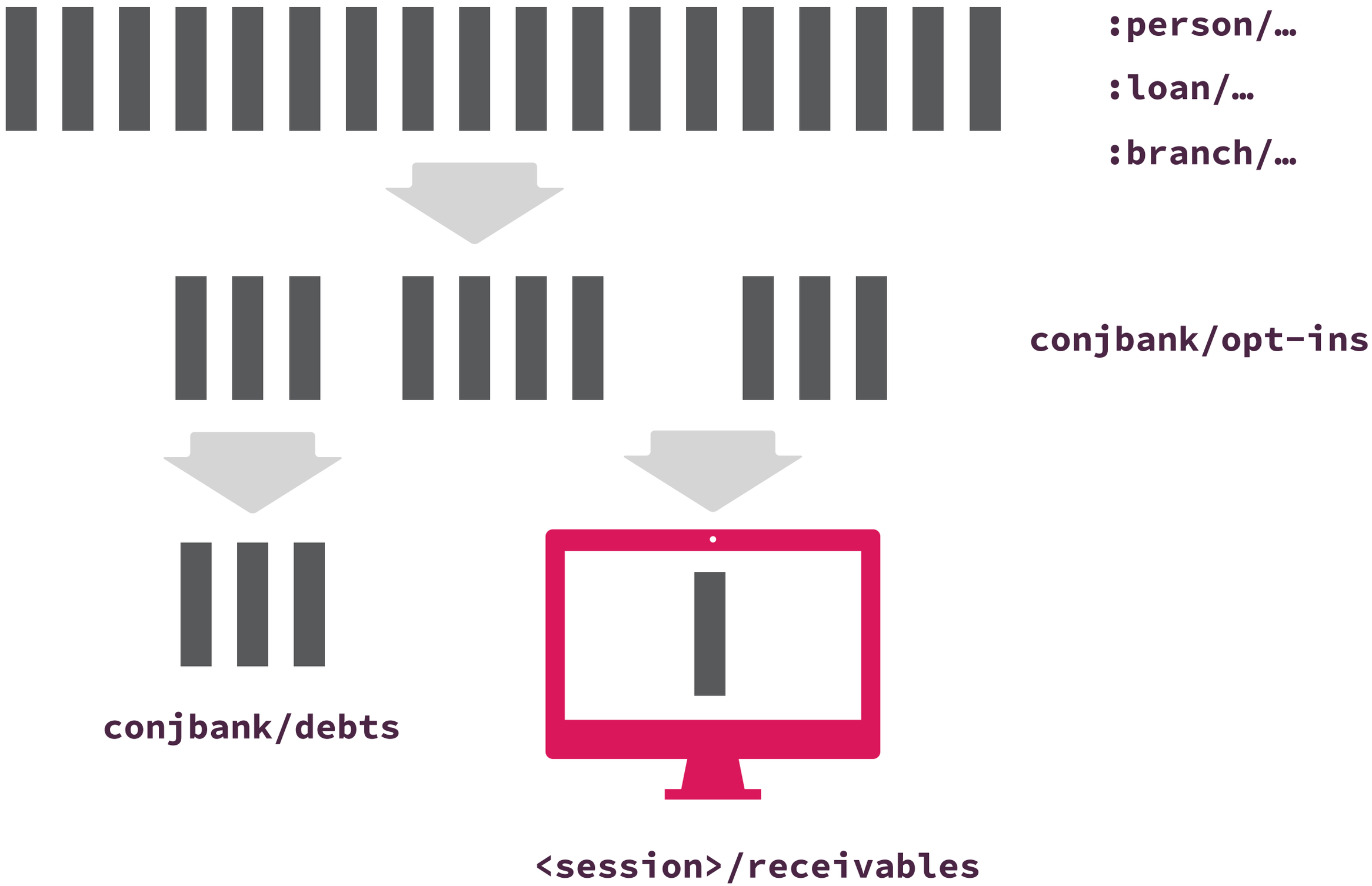


tonsky.me/blog/the-web-after-tomorrow

Data Coordinates Code

- “run a query” => “subscribe to a query”
- re-run to incorporate changes => changes propagate

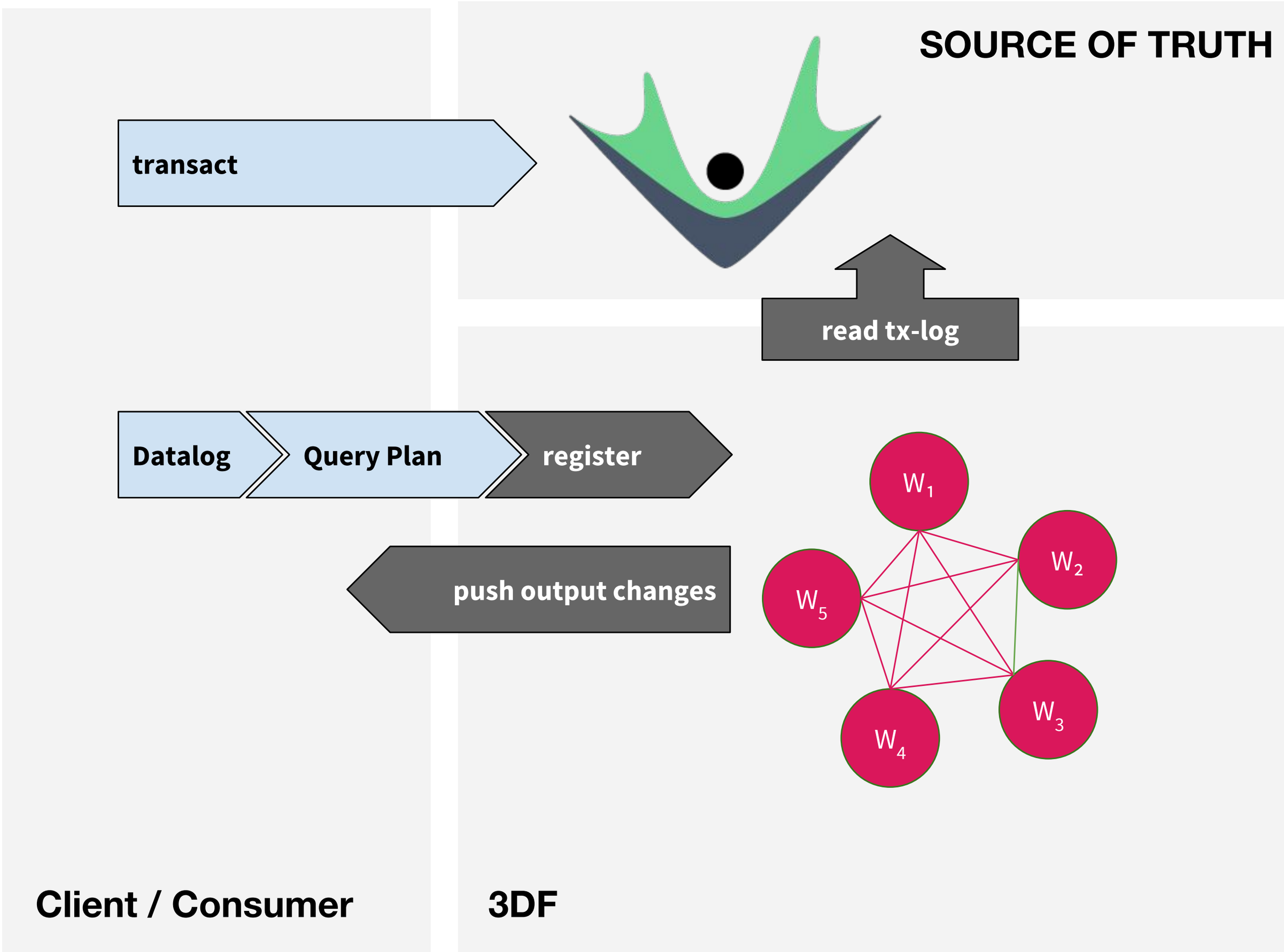

Datalog


:person/…
:loan/…
:branch/…
conjbank/opt-ins
conjbank/debts
<session>/receivables

3DF

- reactive Datalog evaluation on a distributed, data-parallel stream processor
- feeds from durable data sources (Datomic!)
- propagates only changes

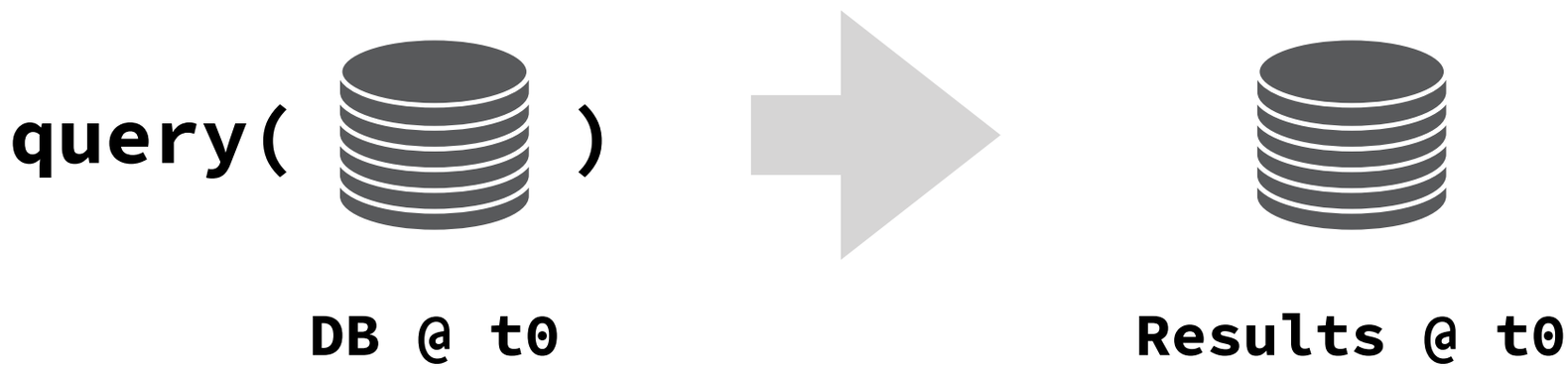
github.com/comnik/clj-3df github.com/comnik/declarative-dataflow


SOURCE OF TRUTH
transact
read tx-log
Datalog
Query Plan
register
push output changes
W1
W2
W3
W4
W5
Client / Consumer
3DF

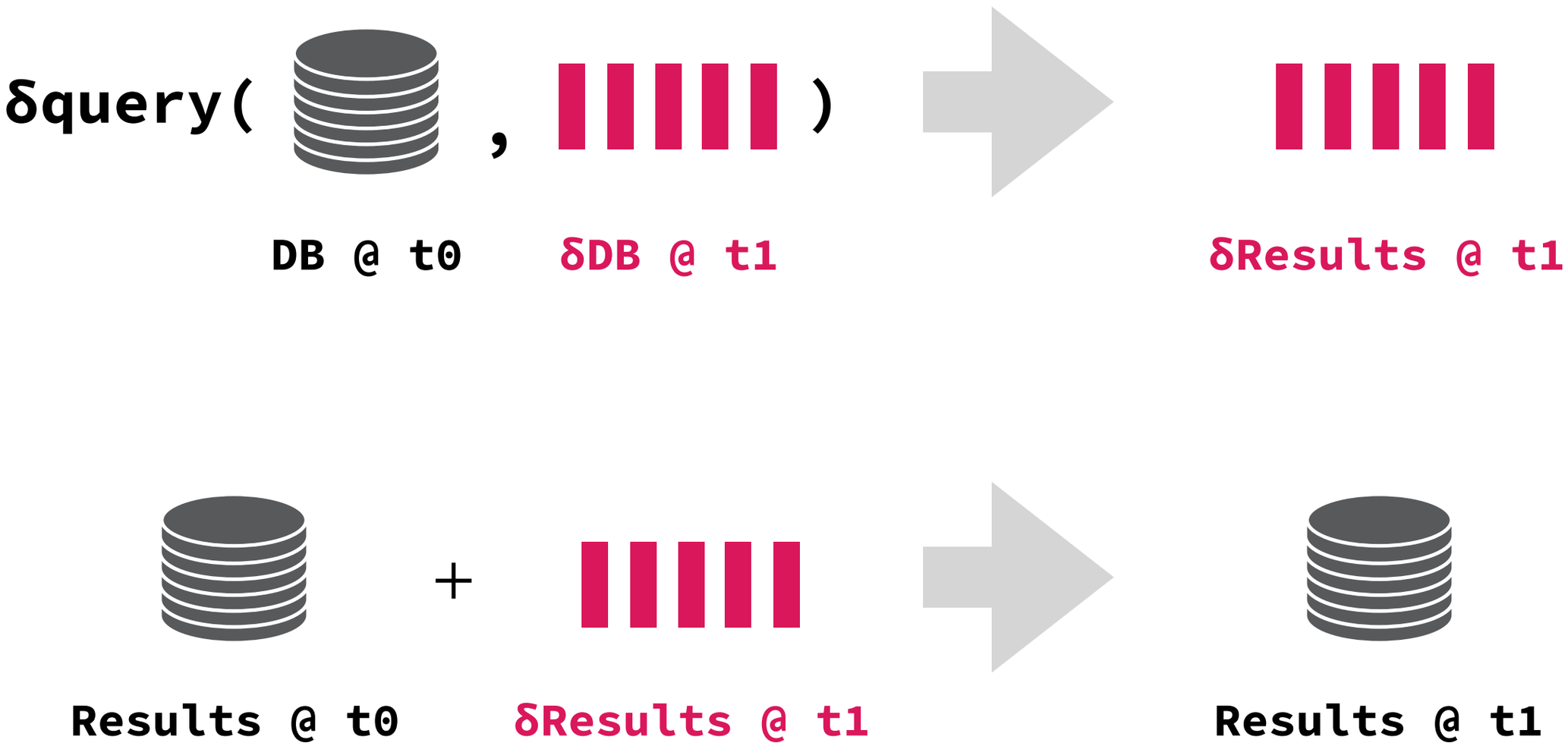
Requirements

- data coordinates code: work like a stream processor
- transparent: notion of change should not leak into Datalog
- reactive iteration w/ arbitrary retractions (recursion!)

Computing w/ Collections



Incremental Computation

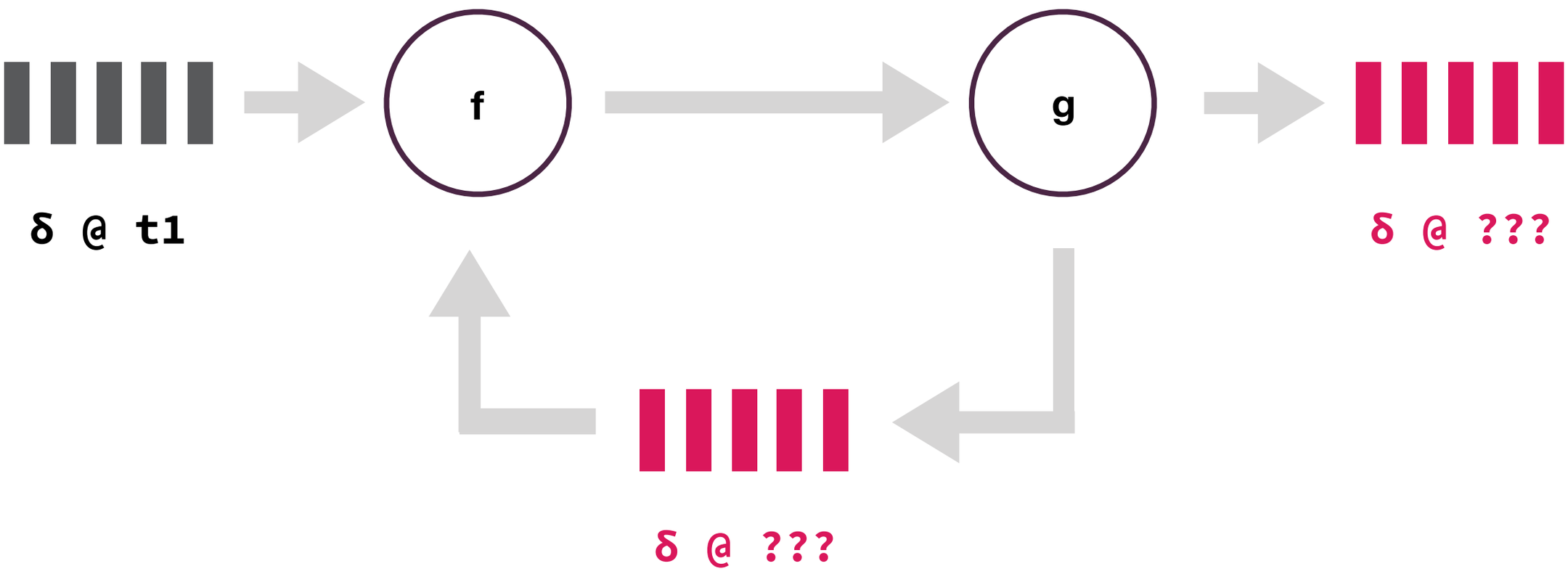


Naive δquery

```
[:find ?device
 :where
 [?device :settings/speed ?target]
 [?device :device/speed ?speed]
 [(< ?speed ?target)]]
```

```
[:find ?device
 :in $ ?tx
 :where
  [$ ?device :settings/speed ?target]
  [?tx ?device :device/speed ?speed]
  [(< ?speed ?target)]]
```

Reactive Iteration?



Differential Dataflow

“a data-parallel programming framework designed to quickly respond to arbitrary changes in input”

github.com/frankmcsherry/differential-dataflow

frankmcsherry.org

github.com/frankmcsherry/timely-dataflow

Persistent Collections

:edge

(a b)	+1	t0
(c d)	+1	t0
(c d)	-1	t1
(c e)	+1	t1

(#{[a b] [c d]} t0)

(#{[a b] [c e]} t1)

Partially-ordered Times

```
(bfs :edge a)
```

(a b) | +1 | t0		
(c d) | +1 | t0		
(b c) | +1 | t1	(a c) | +1 | t1.1	(a d) | +1 | t1.2
		(c d) | -1 | t2

Partially-ordered Times

```
(bfs :edge a)
```

(a b)	+1	(t0 0)
(c d)	+1	(t0 0)
(b c)	+1	**(t1 0)**
(c d)	-1	(t2 0)

(a c)	+1	**(t1 1)**
(a d)	+1	**(t1 2)**

Partially-ordered Times

```
(bfs :edge a)
```

(a b) | +1 | (t0 0)		
(c d) | +1 | (t0 0)		
(b c) | +1 | (t1 0)	(a c) | +1 | (t1 1)	**(a d) | +1 | (t1 2)**
(c d) | -1 | (t2 0)		

Partially-ordered Times

```
(bfs :edge a)
```

(a b)	+1	(t0 0)
(c d)	+1	(t0 0)
(b c)	+1	(t1 0)
(c d)	-1	(t2 0)

(a c)	+1	(t1 1)
(a d)	+1	(t1 2)

Partially-ordered Times

```
(bfs :edge a)
```

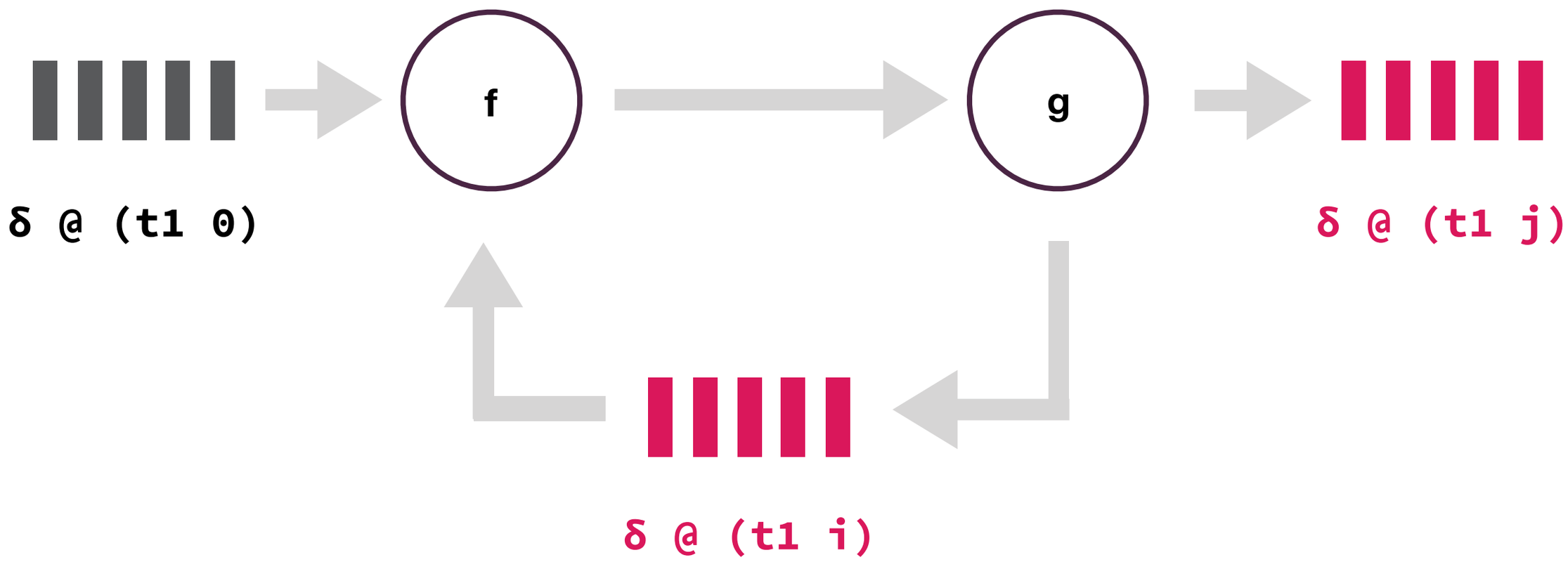
(a b) | +1 | (t0 0)		
(c d) | +1 | (t0 0)		
(b c) | +1 | (t1 0)	(a c) | +1 | (t1 1)	(a d) | +1 | (t1 2)
(c d) | -1 | (t2 0)	-	

Partially-ordered Times

```
(bfs :edge a)
```

(a b) | +1 | (t0 0)		
(c d) | +1 | (t0 0)		
(b c) | +1 | (t1 0)	(a c) | +1 | (t1 1)	(a d) | +1 | (t1 2)
(c d) | -1 | (t2 0)	-	(a d) | -1 | (t2 2)

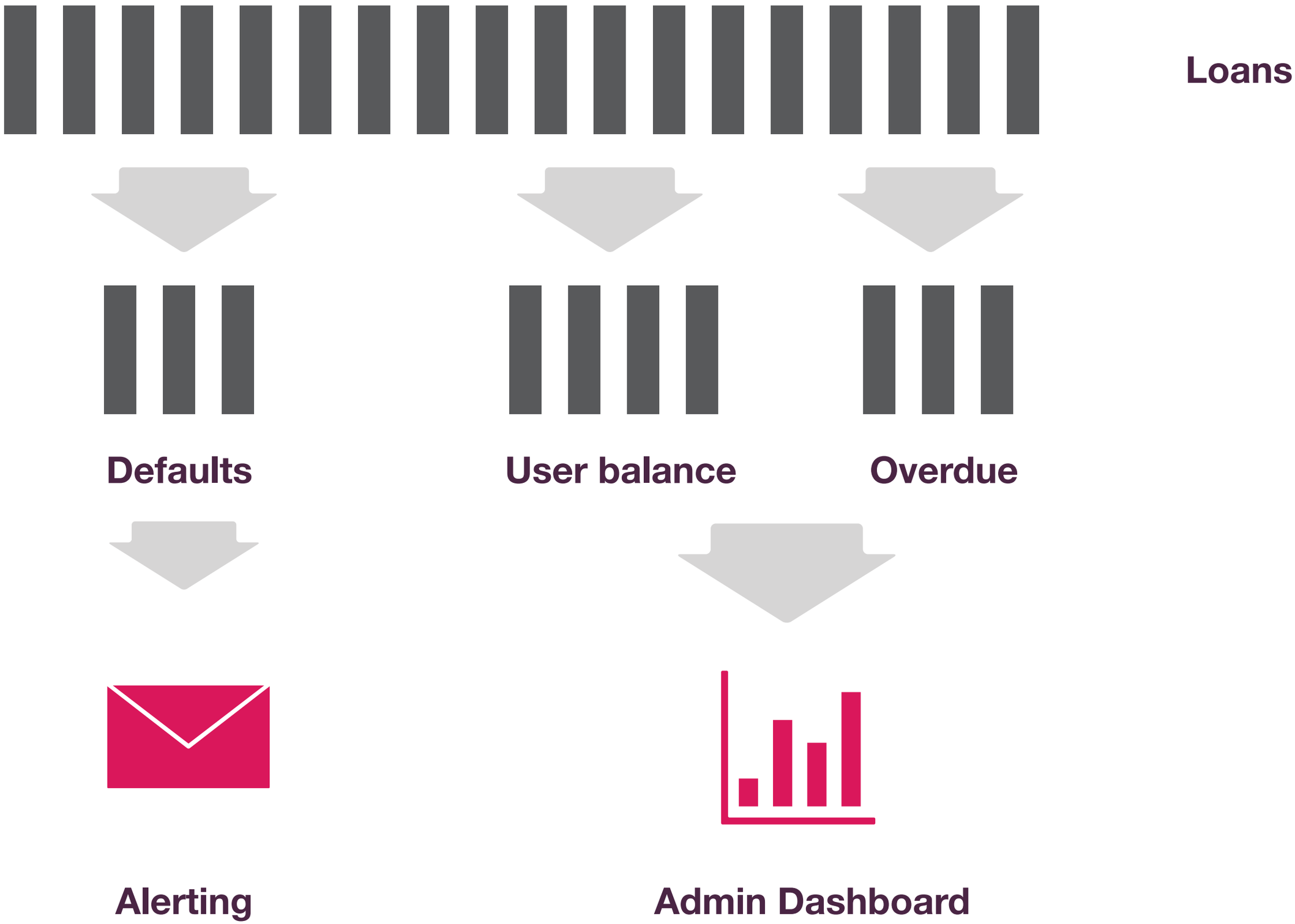
Reactive Iteration!



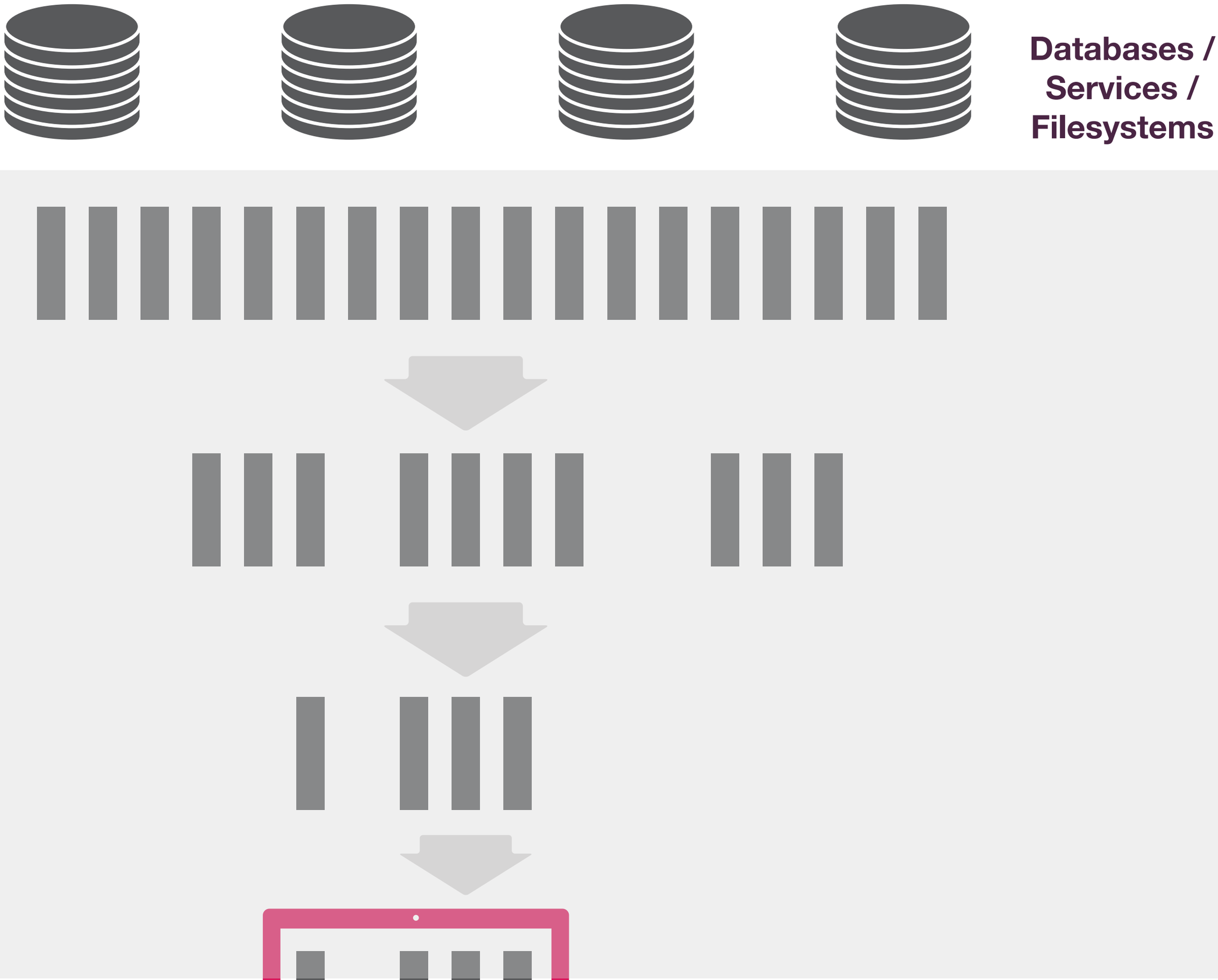
```
/// BFS
let nodes = roots.map(|x| (x, 0));
nodes.iterate(|inner| {
  let edges = edges.enter(&inner.scope());
  let nodes = nodes.enter(&inner.scope());

  inner.join_map(&edges, |_k,l,d| (*d, l+1))
       .concat(&nodes)
       .group(|_, s, t| t.push((*s[0].0, 1)))
})
```

Reactive Systems!

- ~~live-updating web applications~~
- alerting and real-time dashboards
- stream processors
- rule engines



clockworks.io/2018/11/25/reactive-datalog-with-vega.html


Databases /
Services /
Filesystems

Notes on Performance

- incremental computation pegs system load to rate of change, rather than to granularity ✖ # of users
- queries not limited to a single peer
- timestamps support nanosecond precision (no windows!)
- drop down to hand-built Differential Dataflow if needed

Future Work

- scalability and performance isolation for large # of queries
- bullet-proofing the drop-in Datomic integration
- Beta release

“We want to make reactive systems that don't poll. And we want those systems to get a consistent view of the world.”

– *Rich Hickey, “Deconstructing the Database”*

3DF lets you…

- …leverage Datalog to build reactive systems and distributed stream-processing jobs
- …scale queries beyond the limits of a single peer
- …continue using Datomic for everything else
- …integrate with hand-written Differential Dataflow

Interested? Let’s talk!

@NikolasGoebel | niko@clockworks.io

github.com/comnik/clj-3df

github.com/frankmcsherry/differential-dataflow