Immutable data architecture with Datomic, Spark and Kafka

Konrad Scorciapino @konr

Mauro Lopes @maurolopes23

nu bank

Share our experience

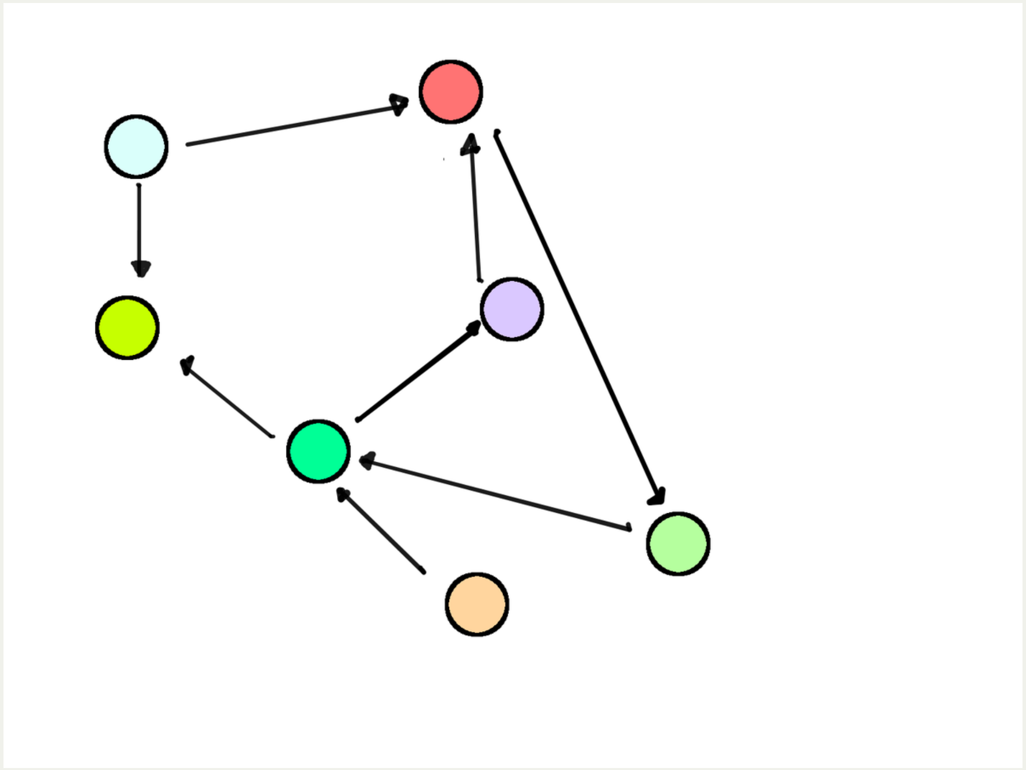
- Data architecture
- Supporting machine learning
- Data access

Overview

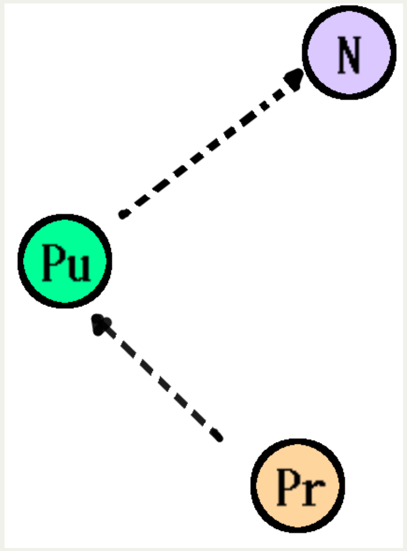
0. Background
1. Scoring
2. Training
3. Analysing

Background

Microservices



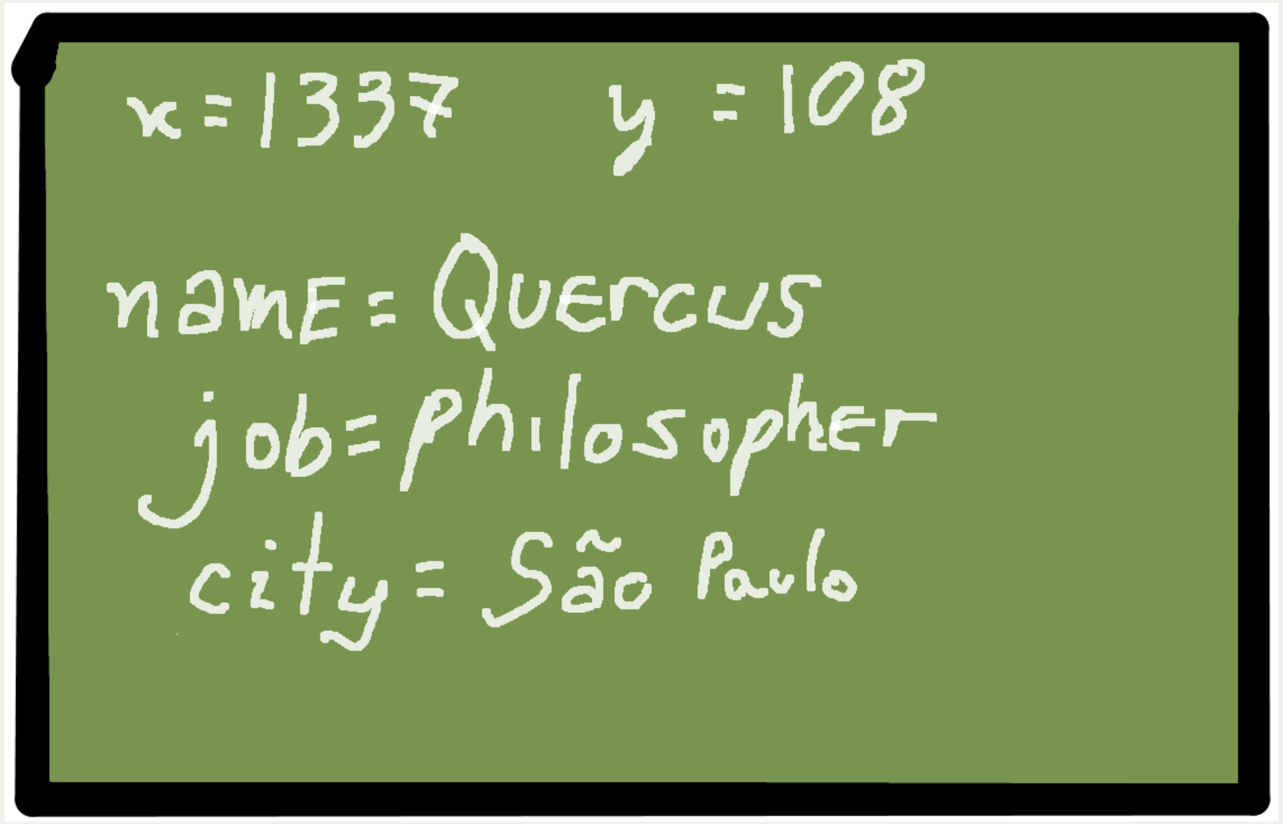
An interaction



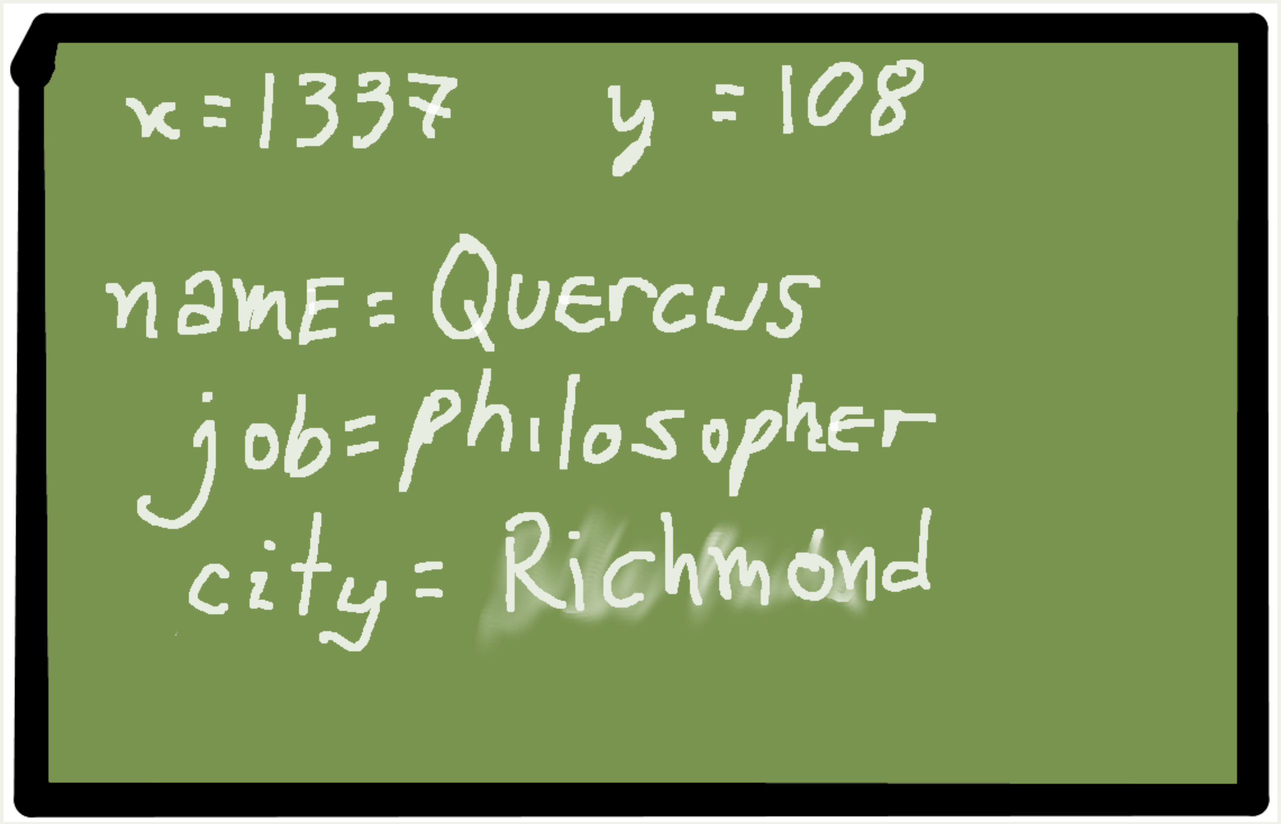
Database

- Not SQL
- Not NoSQL
- Datomic

Board



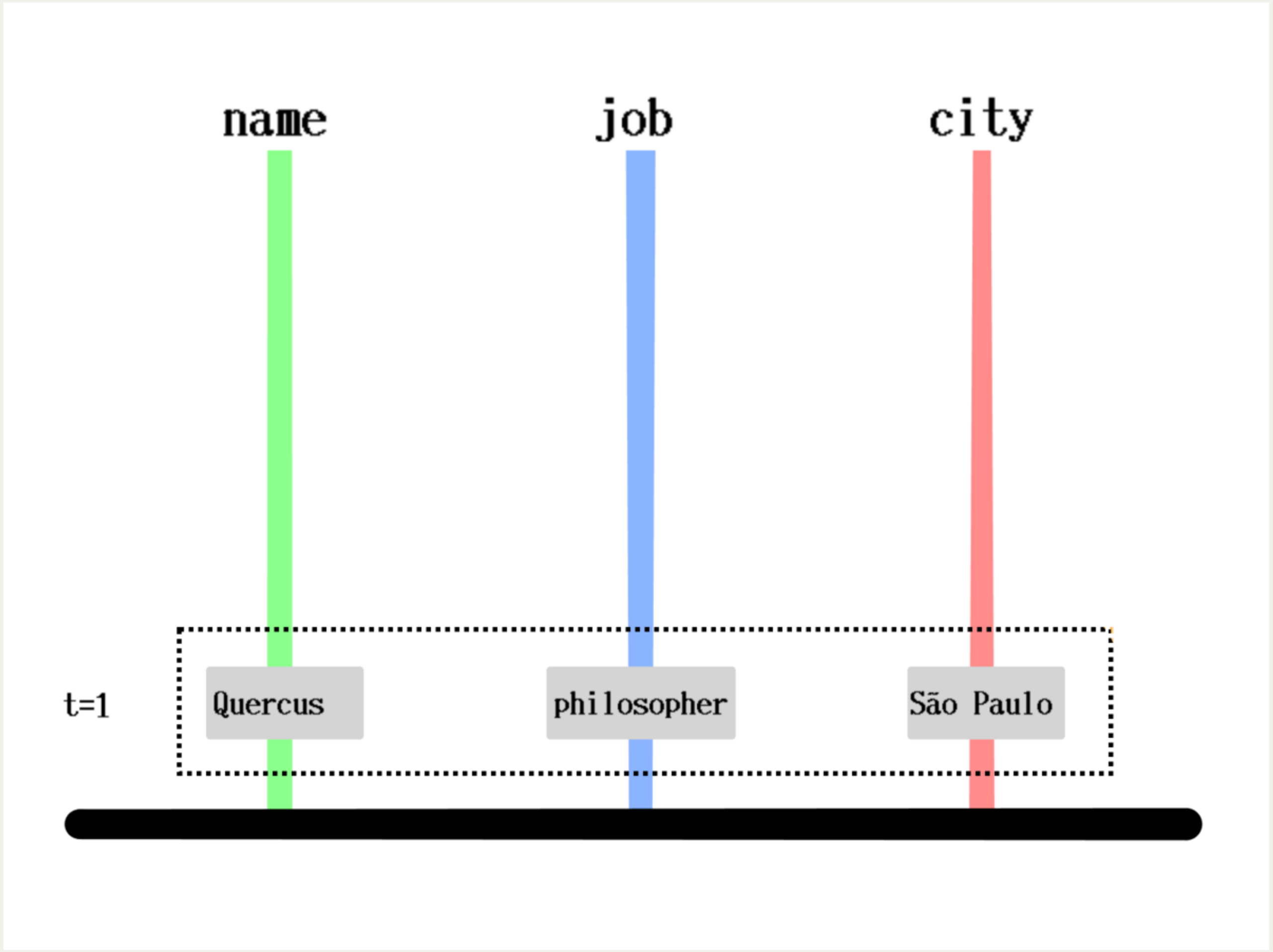
Board



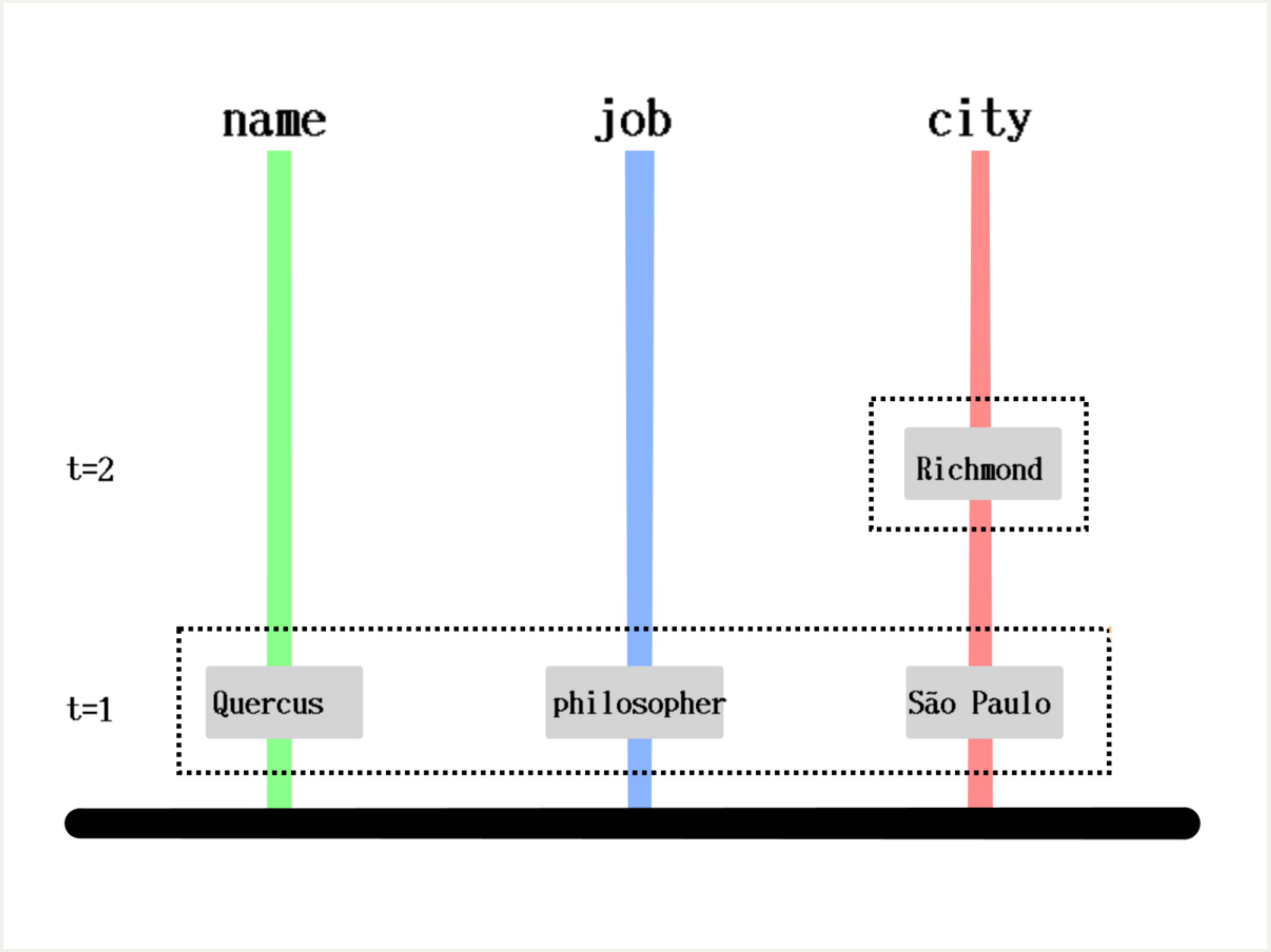
x = 1337 y = 108

name = Quercus
job = philosopher
city = Richmond

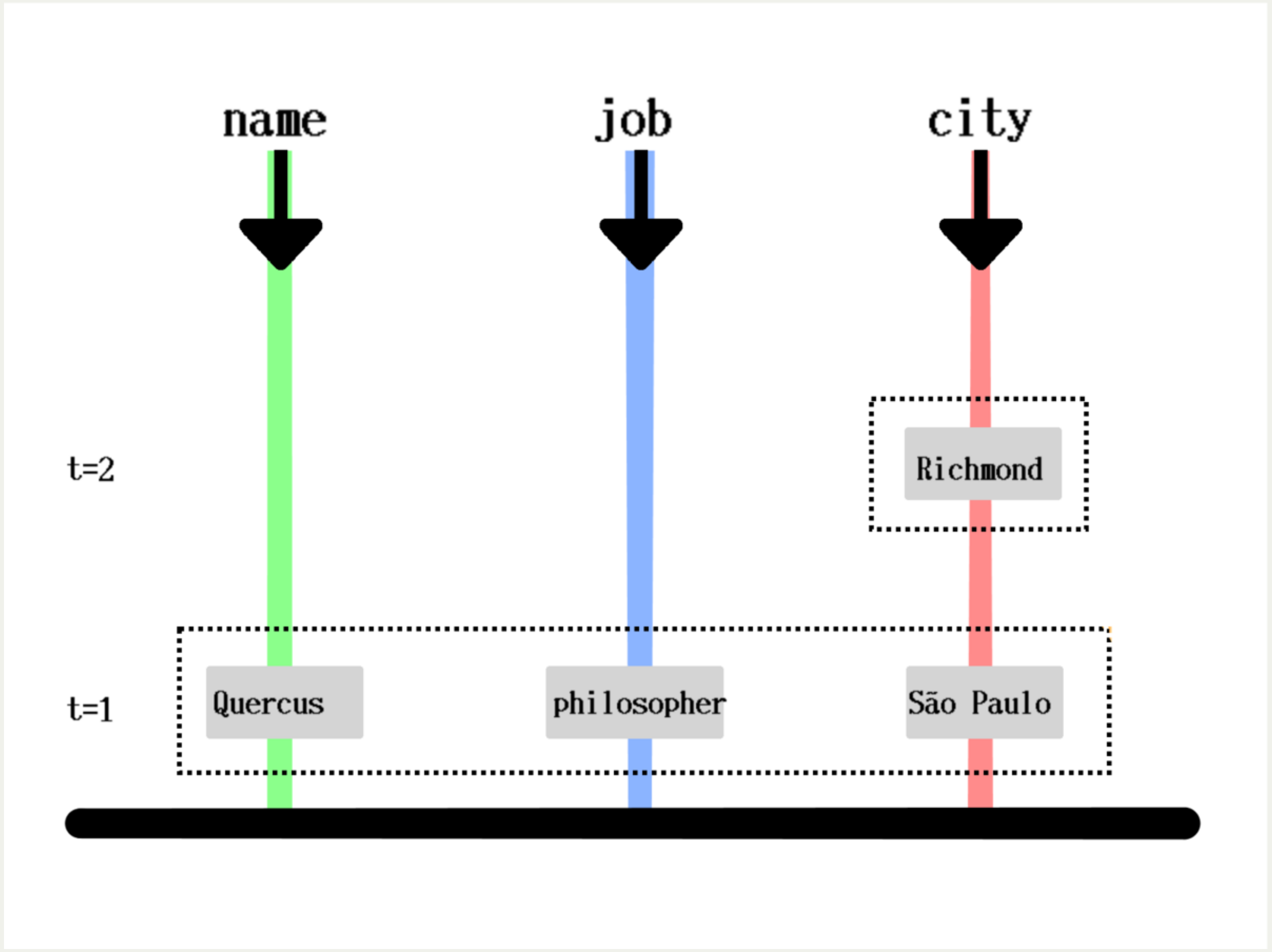
Piles



Pile



Pile



How is it stored?

```
[[1000 :person/name "Quercus"       1]
 [1000 :person/job  "Philosopher"   1]
 [1000 :person/city "São Paulo"     1]
 [1000 :person/city "Richmond"      2]]
```

How is it queried?

```
{:find [?job]
 :in   [?name]
 :where [[?p :person/name ?name]
         [?p :person/job ?job]]}
```

How is it queried?

```
{:find [?name]
 :in   [?job]
 :where [[?p :person/name ?name]
         [?p :person/job ?job]]}
```

How is it queried?

```
{:find [?name ?job]
 :in  []
 :where [[?p :person/name ?name]
         [?p :person/job ?job]]}
```

Scoring

(approach)

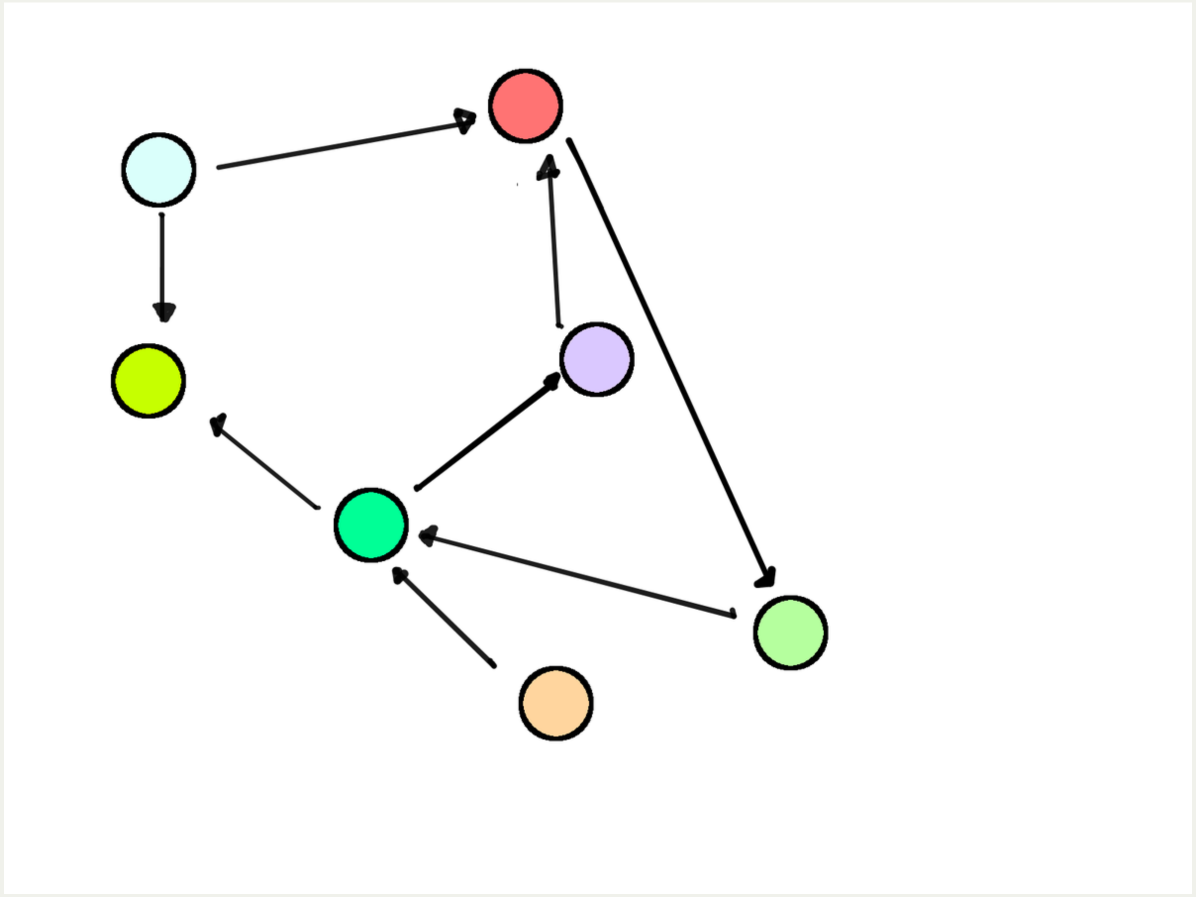
Model training vs Scoring

$$train(x_0, x_1, ..., x_m) -> Blackbox$$
$$Blackbox(x) -> Score$$

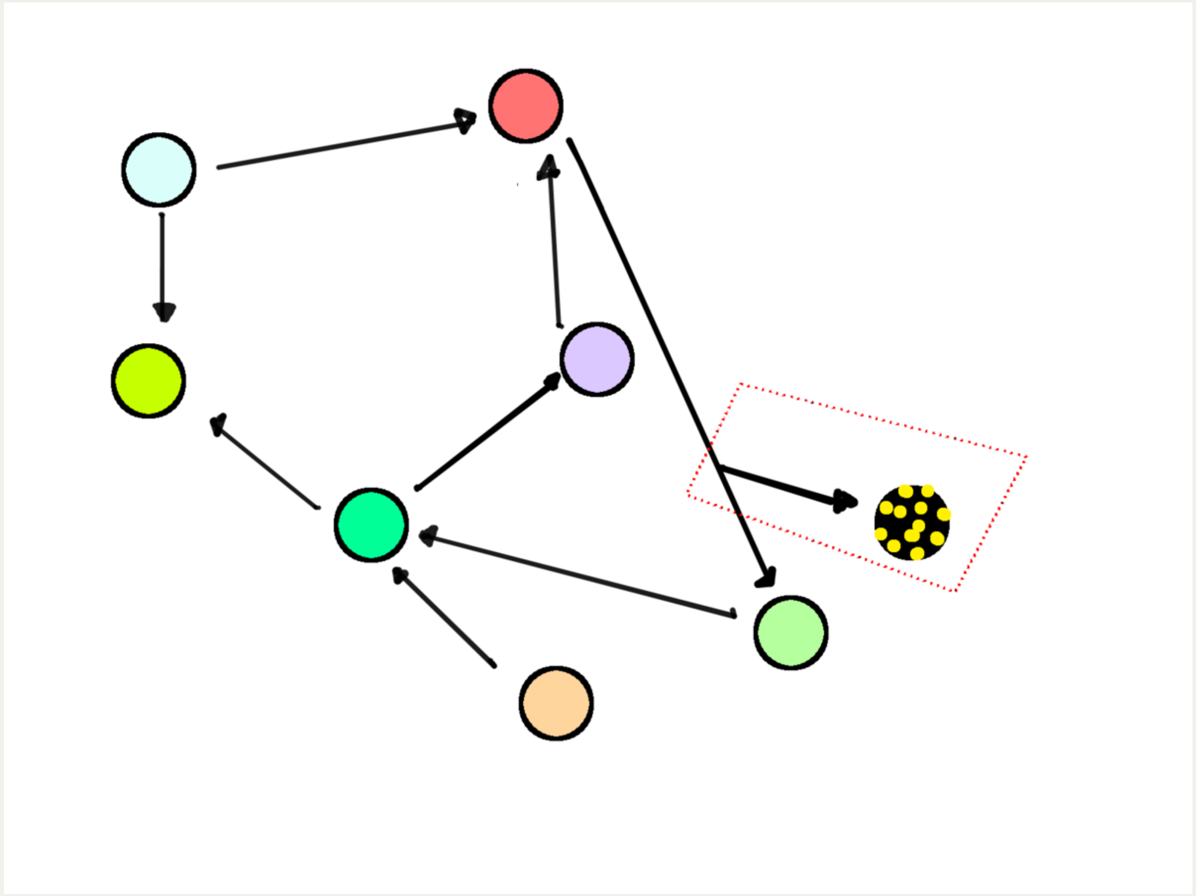
Scoring

$$Blackbox(x) -> Score$$

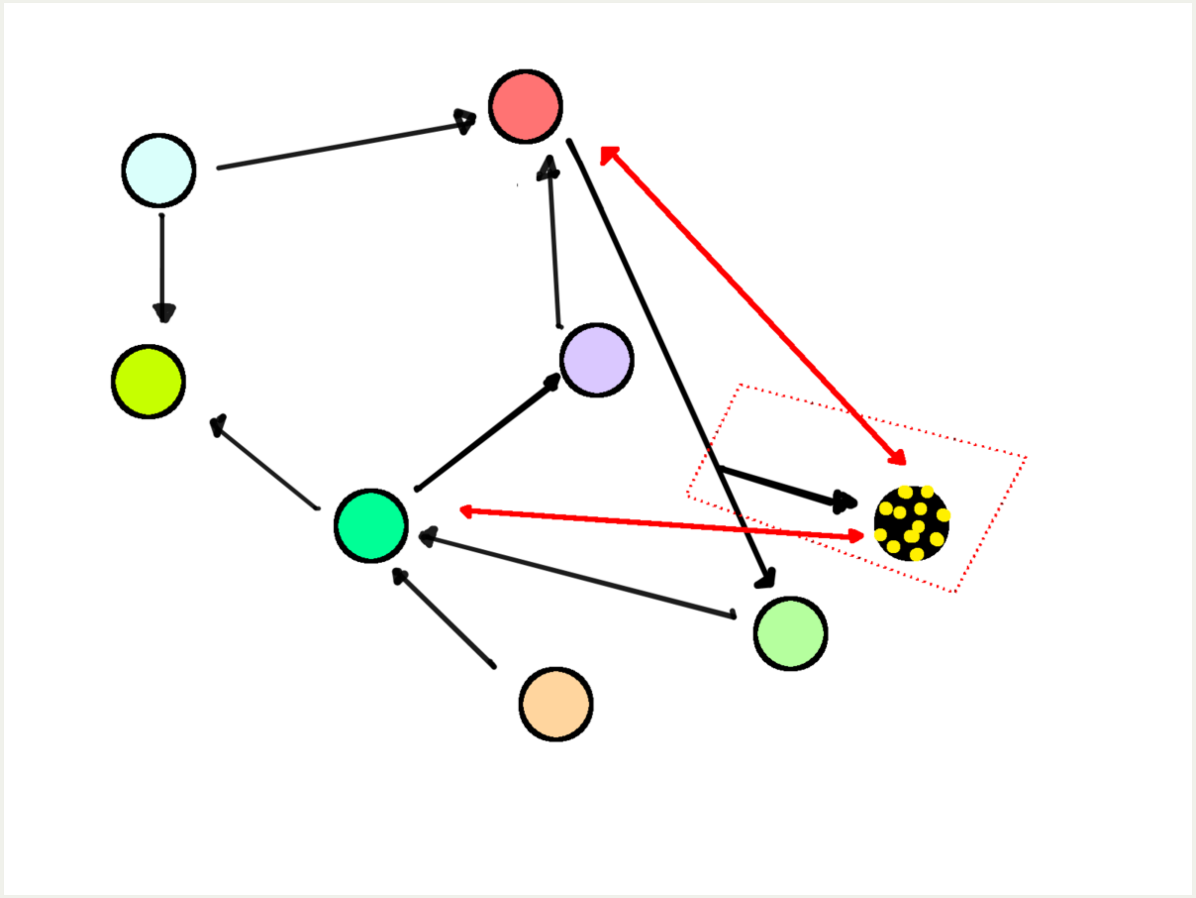
How do we get x?



Entity from a queue



Enriched entity

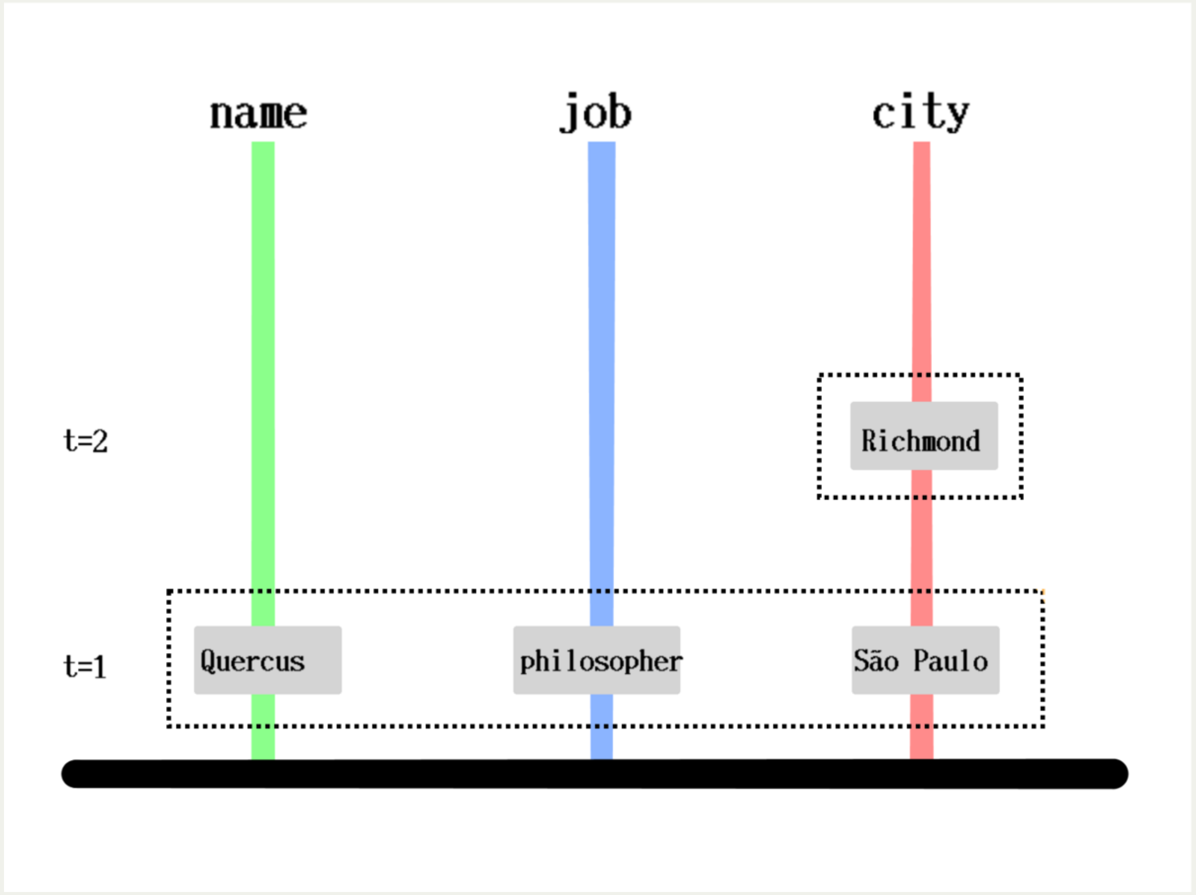


Downsides

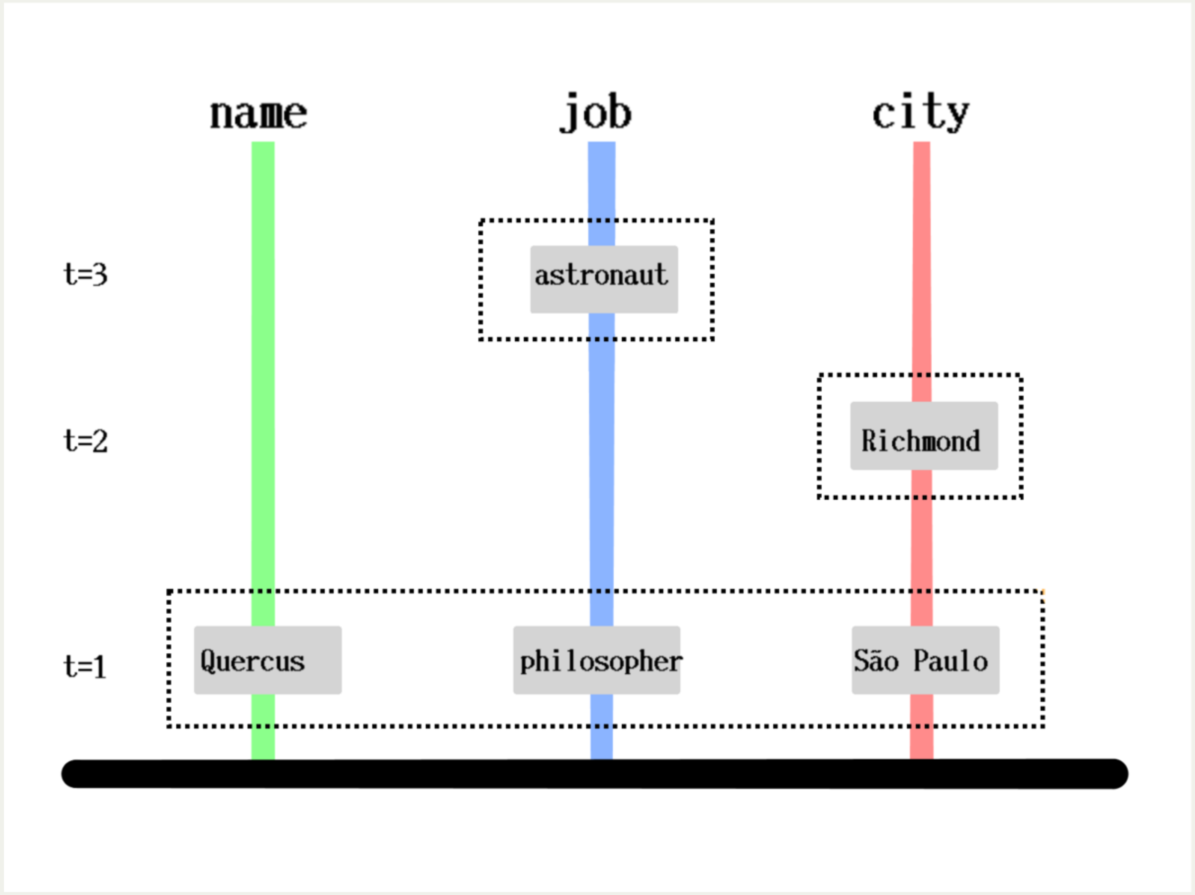
- Affects production
- Complexity on services
- Traceability (Data, PII)

A second approach:
data directly from the database

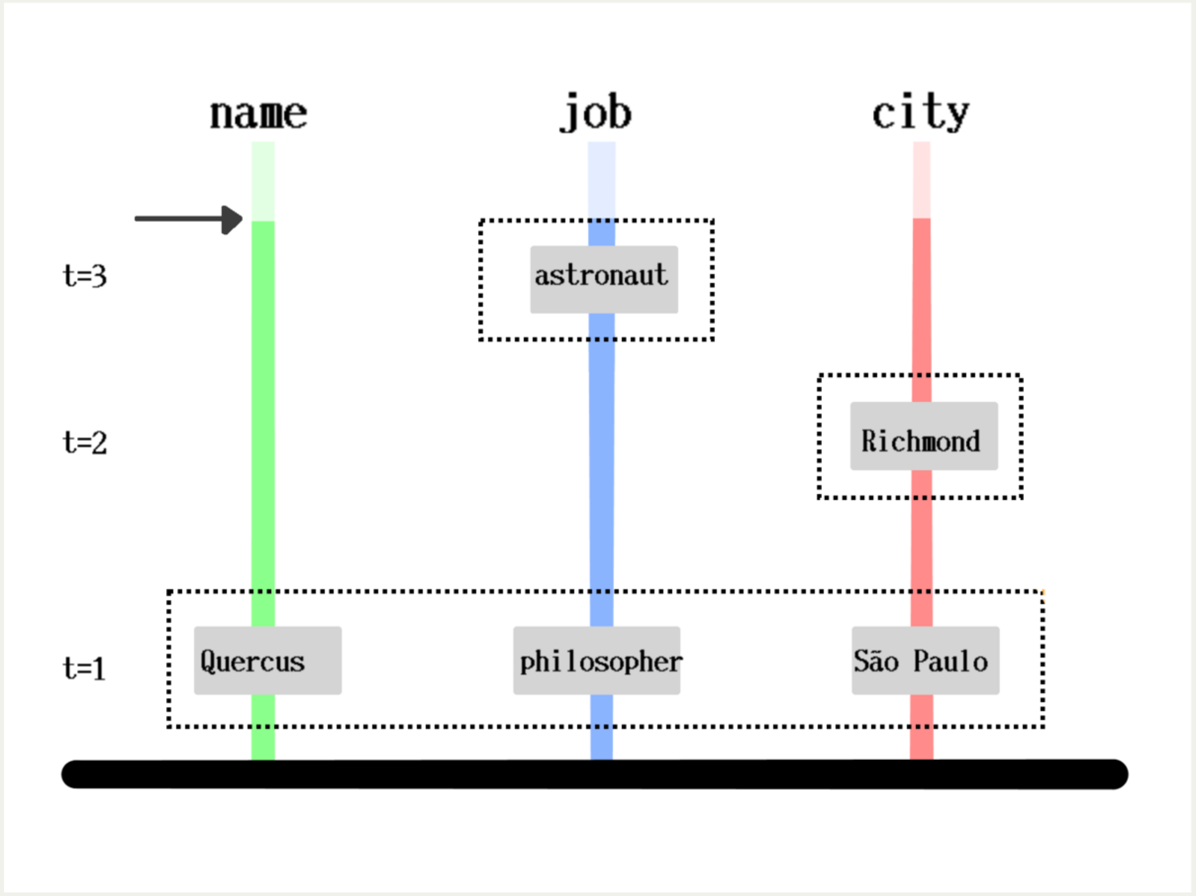
Cursors



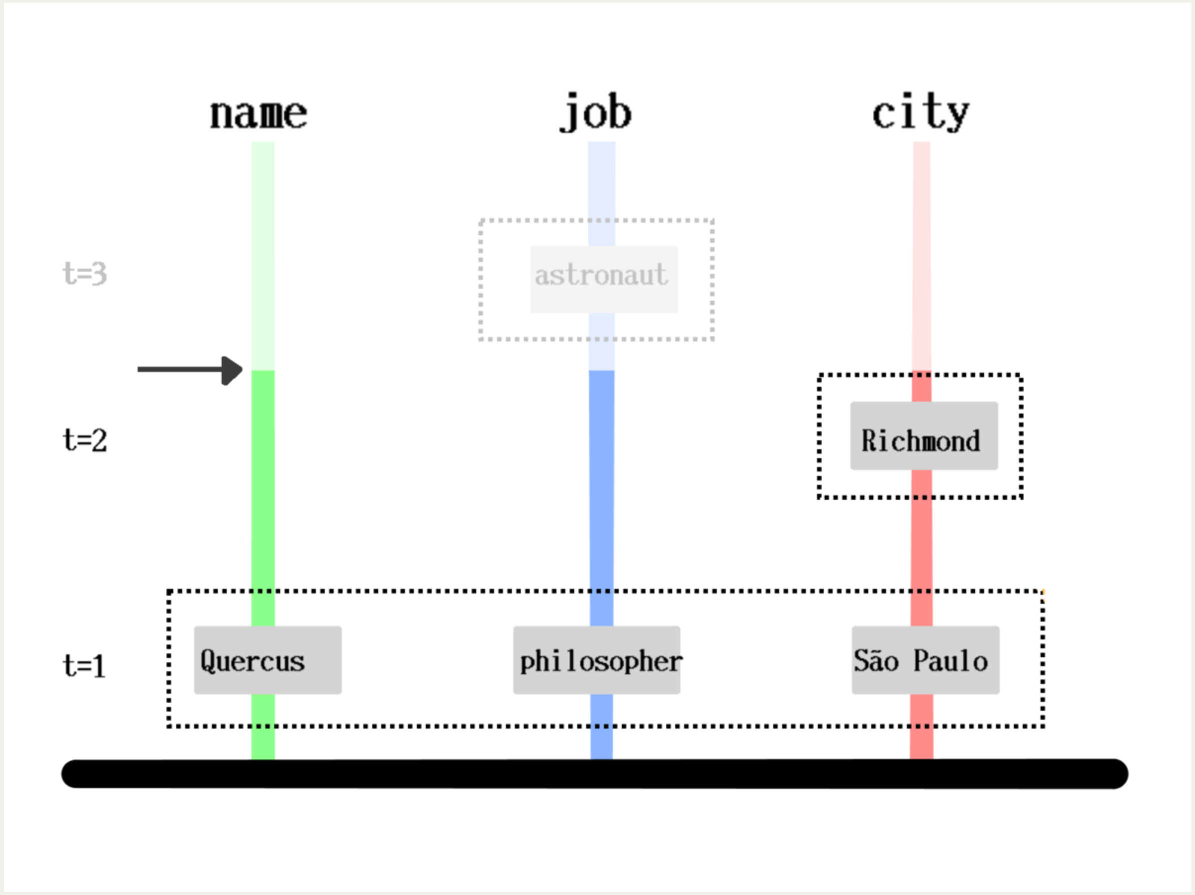
Cursor


name
job
city
t=3
t=2
t=1
astronaut
Richmond
Quercus
philosopher
São Paulo

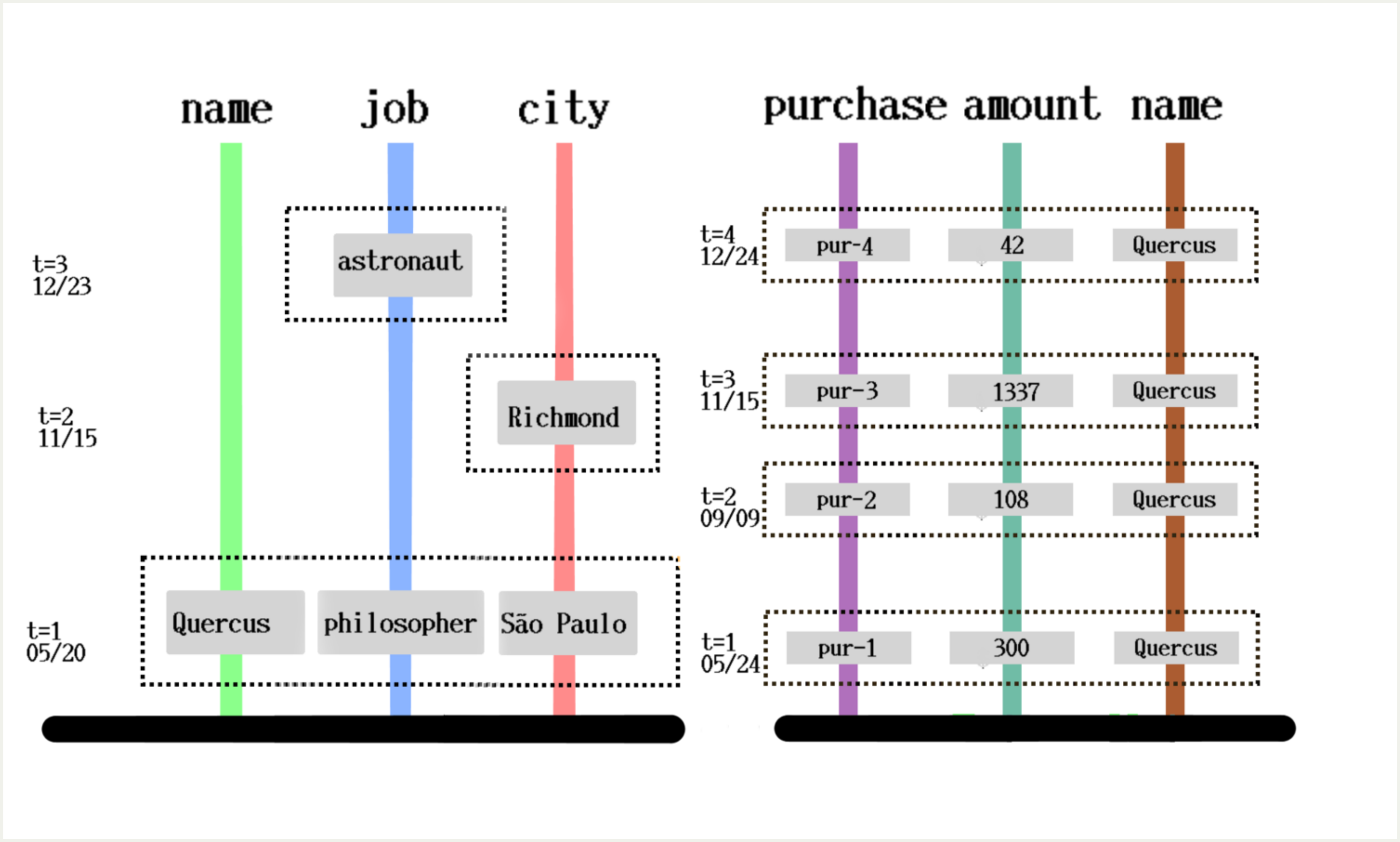
Cursor


name
job
city
t=3
t=2
t=1
astronaut
Richmond
Quercus
philosopher
São Paulo

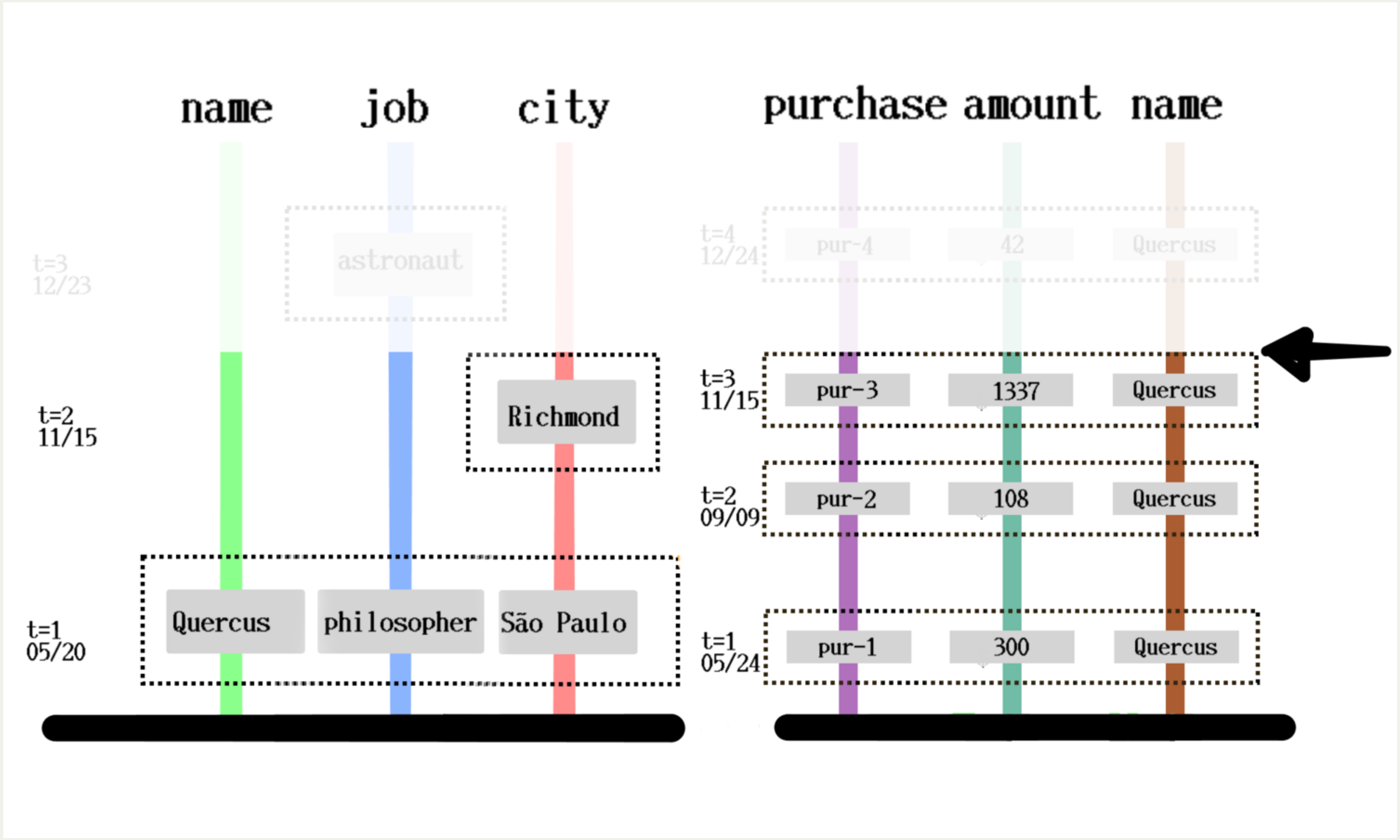
Entity from cursor and id



Multiple DBs



Multiple DBs



Scoring

(services)

A service where

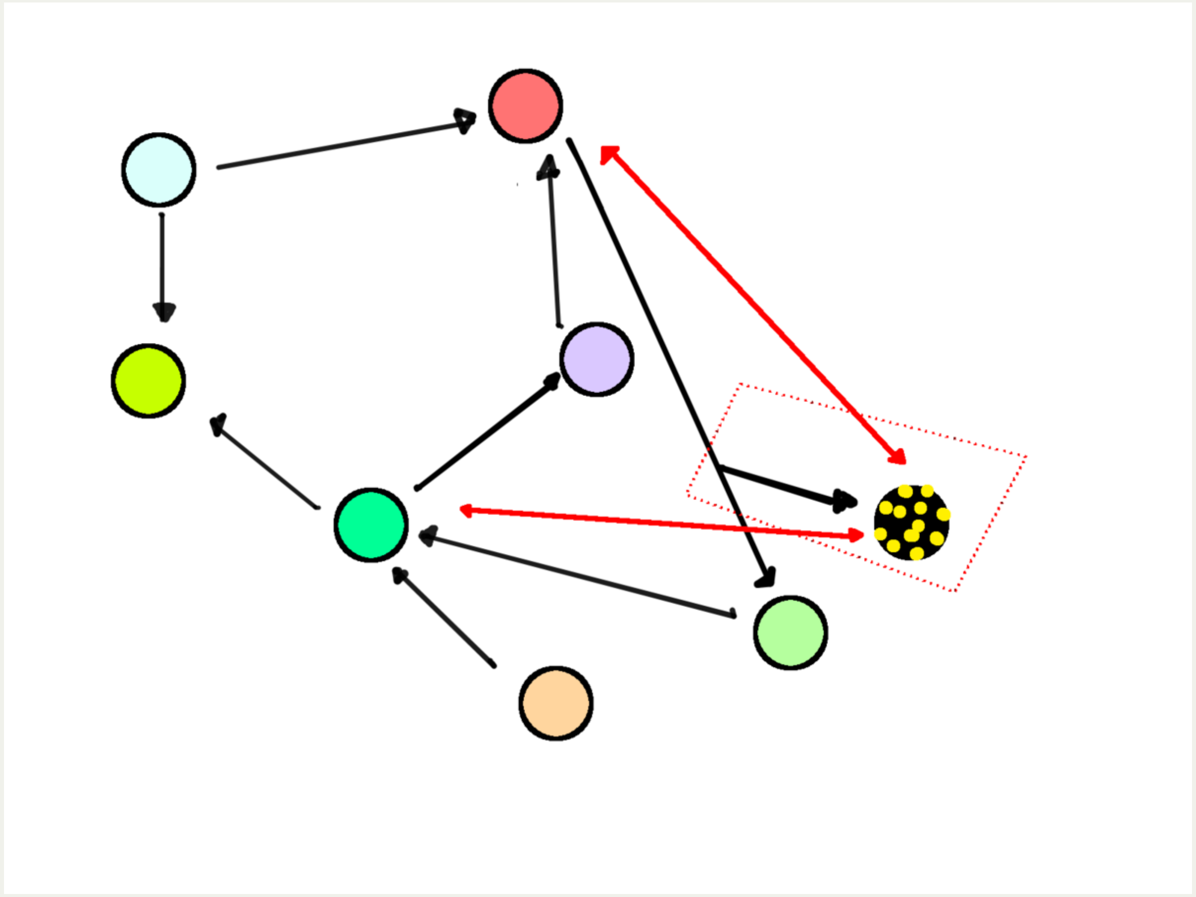
- All the data is avaliable
- All services become one
- All the chaos intertwines

MORDOR

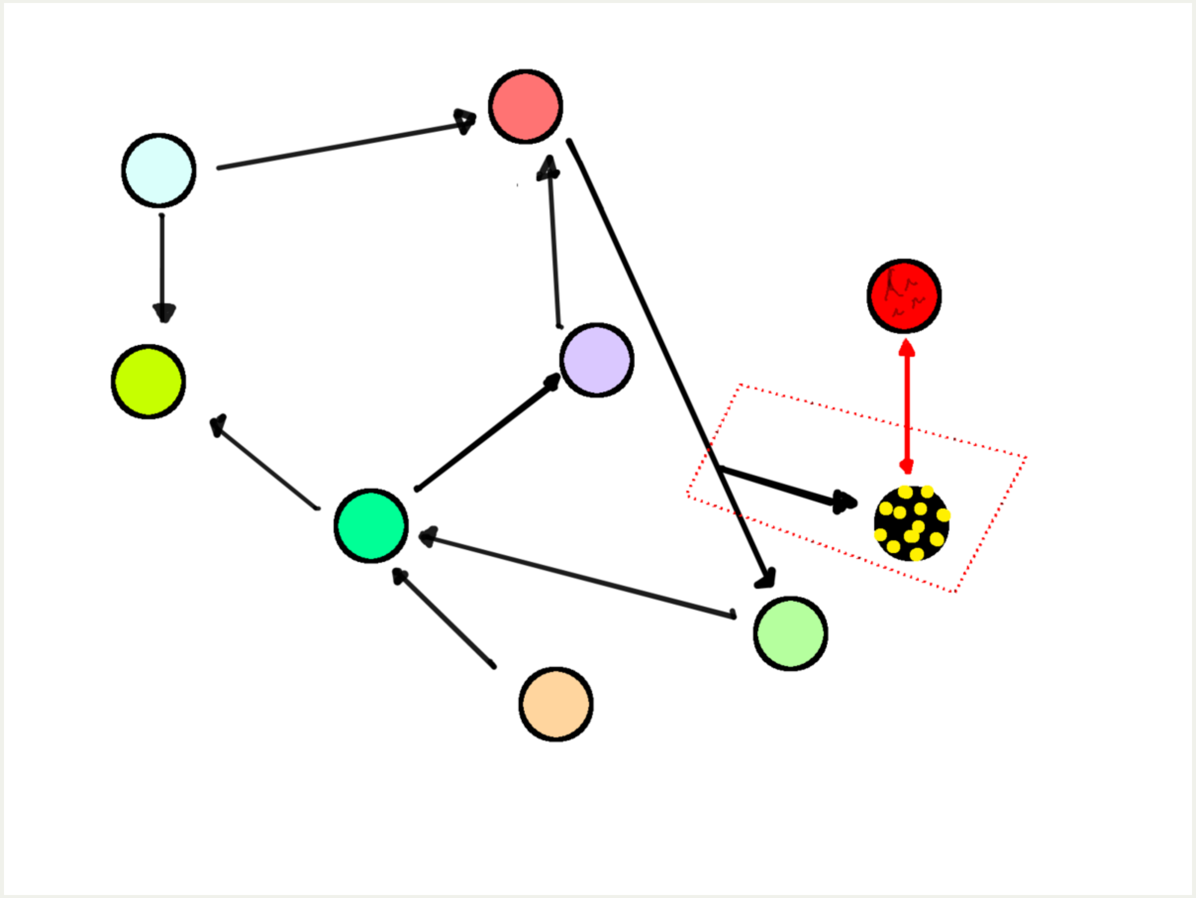
One service to query them all

- Read-only querying service
- No interference

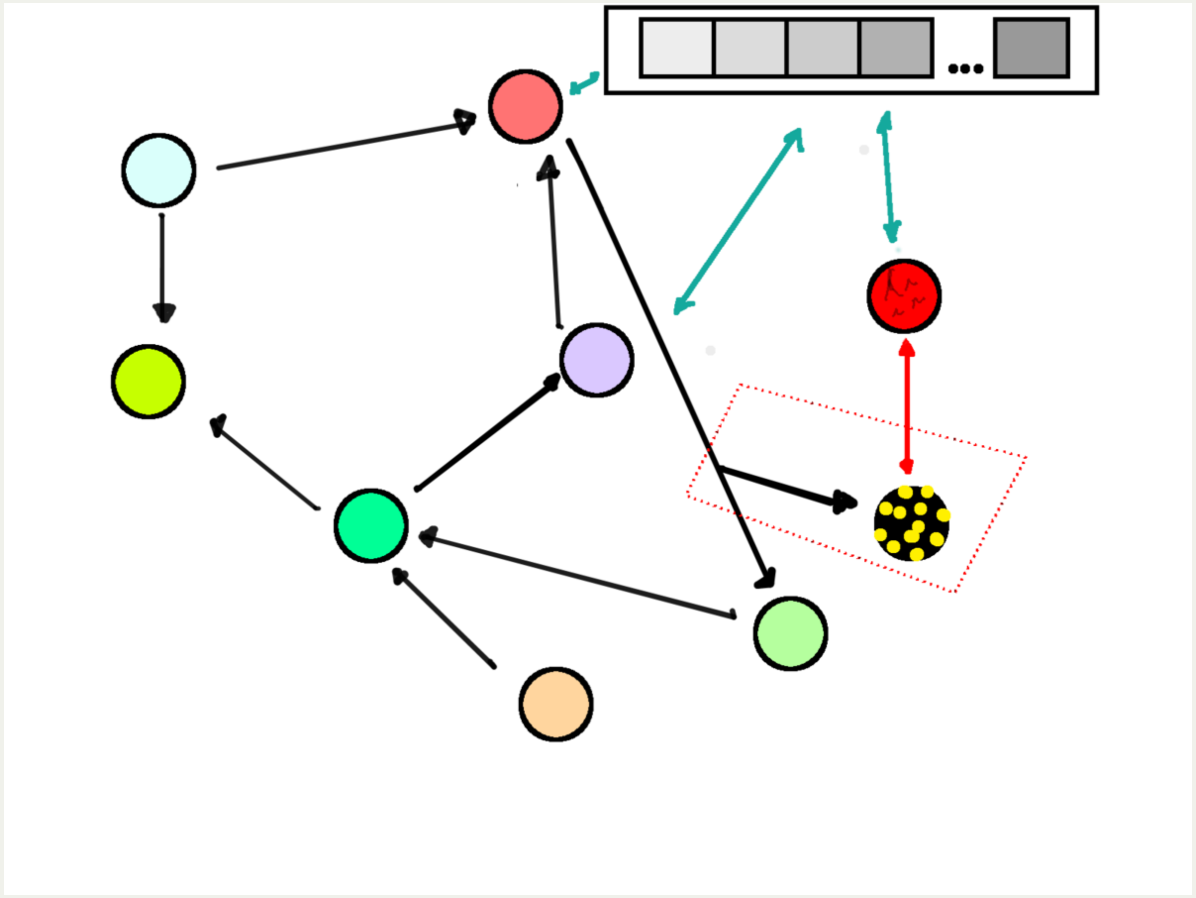
No interference



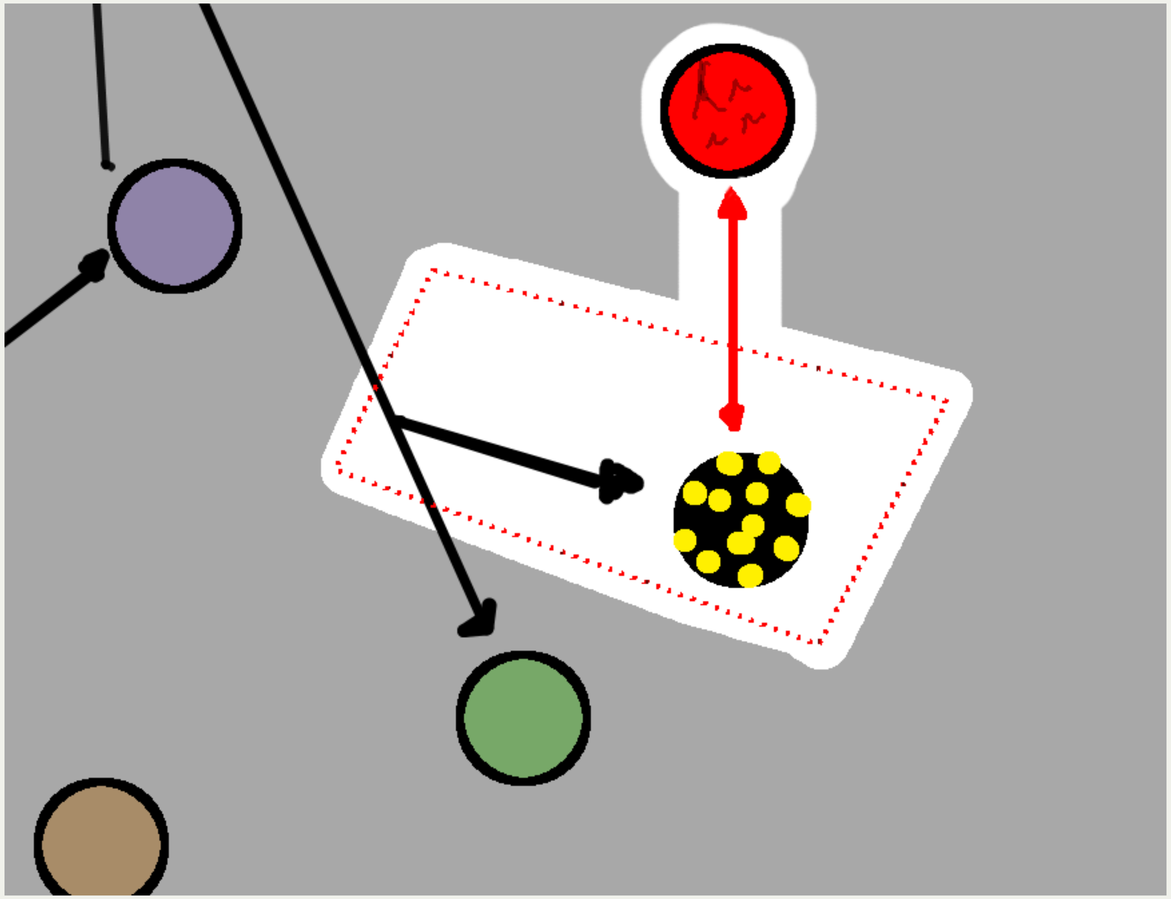
No interference



No interference



Using it



Sample message

```json
{
  "purchase": {
    "id": "pur-108",
    "amount": "133700",
    "merchant": "Quercus Bookshop",
    "purchase-date": "2014-11-15",
    ...
  },
  ...
  "timestamp": "2014-11-15T13:37:42.108Z"
}
```

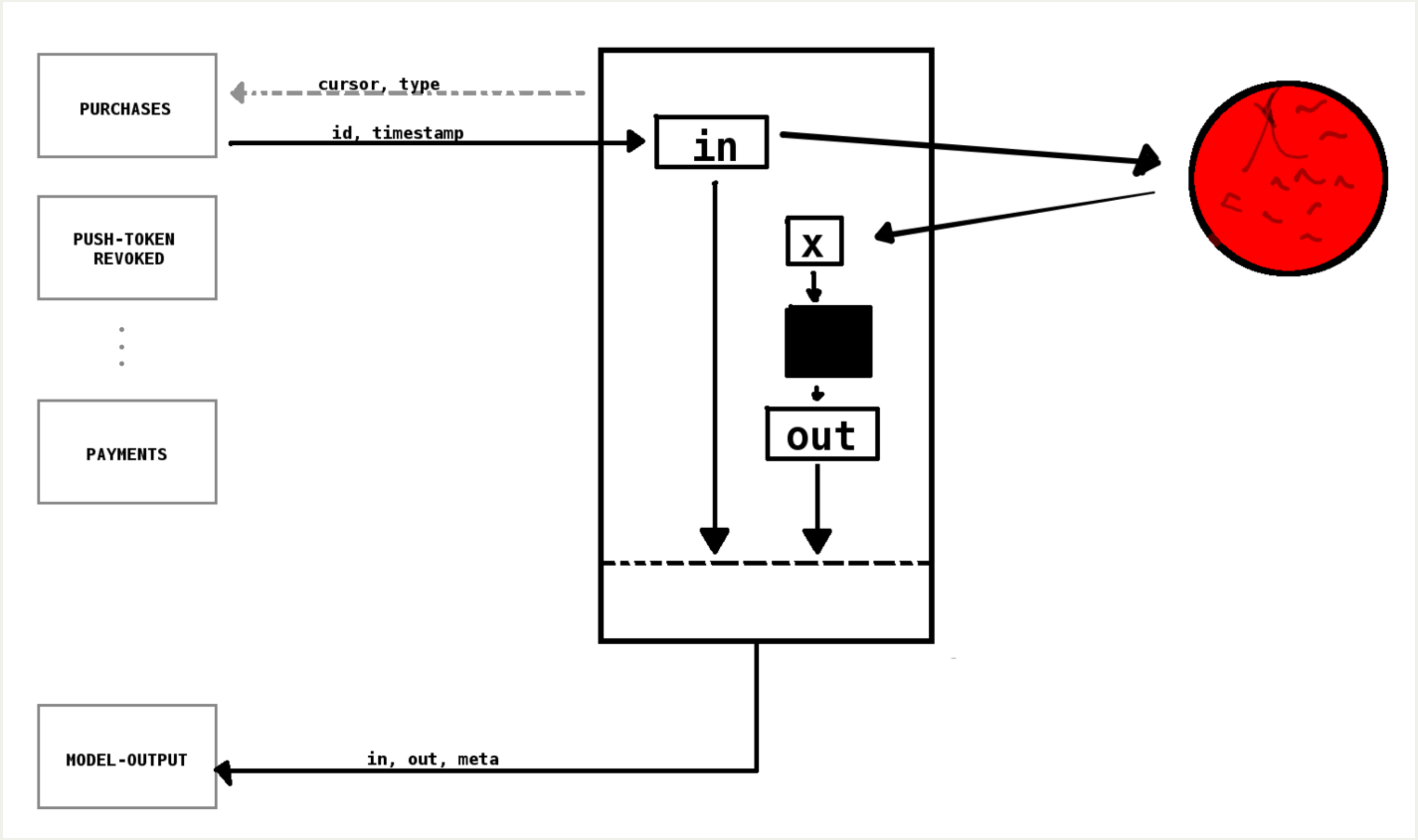
Sample message

```
{
  "purchase": {
    "id": "pur-108",
    "amount": "133700",
    "merchant": "Quercus Bookshop",
    "purchase-date": "2014-11-15",
    ...
  },
  ...
  "timestamp": "2014-11-15T13:37:42.108Z"
}
```

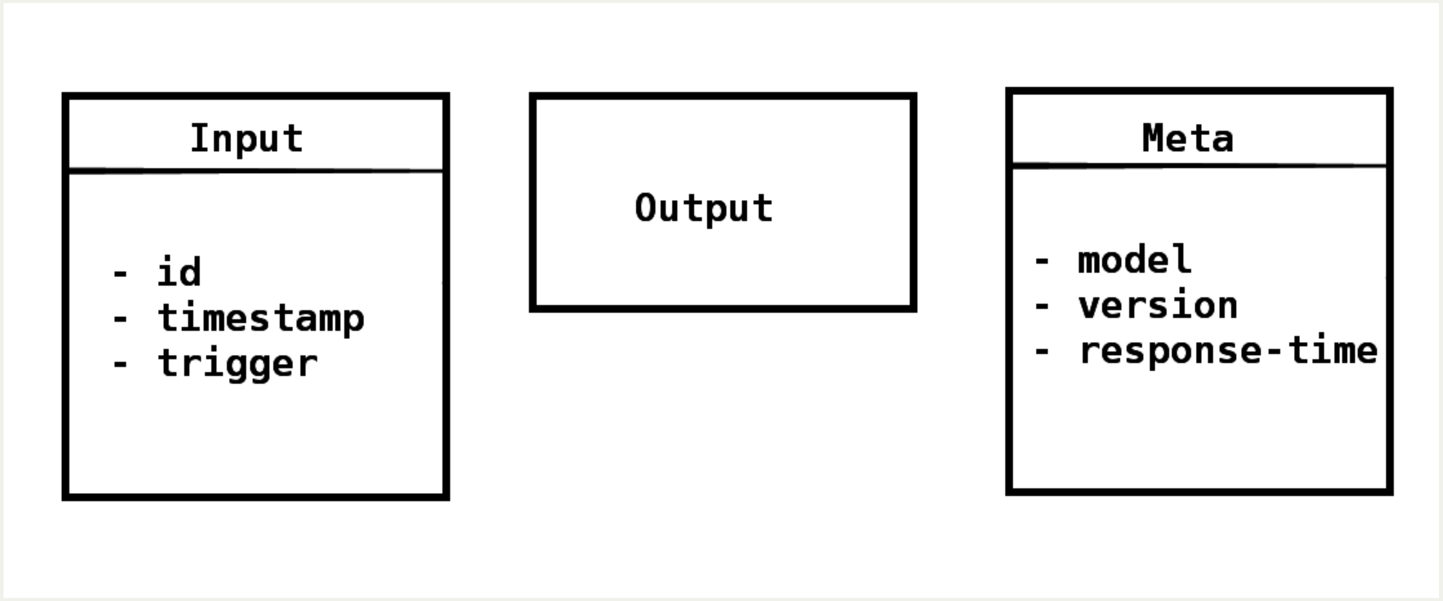
Sample query

```
{:as-of #inst "2014-11-15T13:37:42Z"
 :args ["pur-8"]
 :query
   {:find  [?name (max ?amount)]]
    :in    [?purchase-id]
    :where [[$pur ?purchase  :purchase/id ?purchase-id]
            [$pur ?purchase  :purchase/customer-id ?customer-id]

            [$cus ?customer  :customer/id ?customer-id]
            [$cus ?customer  :customer/name ?name]

            [$pur ?purchase2 :purchase/customer-id ?customer-id]
            [$pur ?purchase2 :purchase/amount ?amount]]}}
```

Model service



Output


Input
- id
- timestamp
- trigger
Output
Meta
- model
- version
- response-time

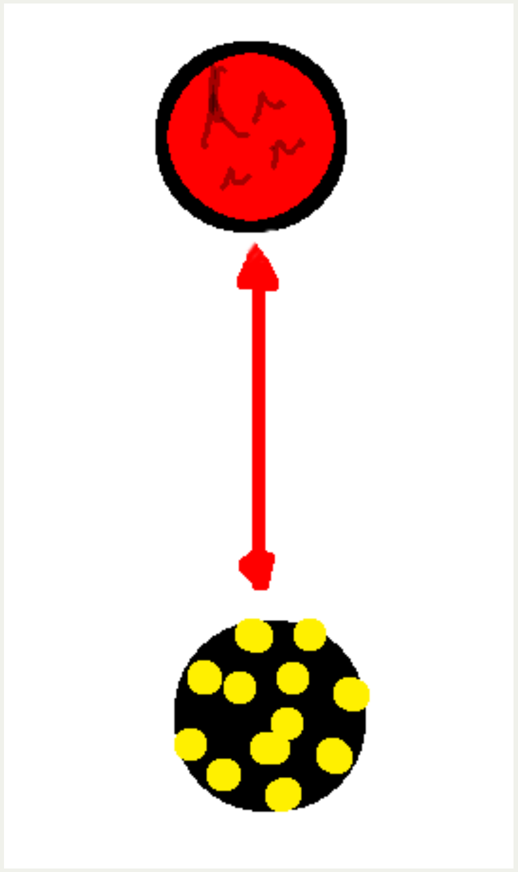
Training

Goal

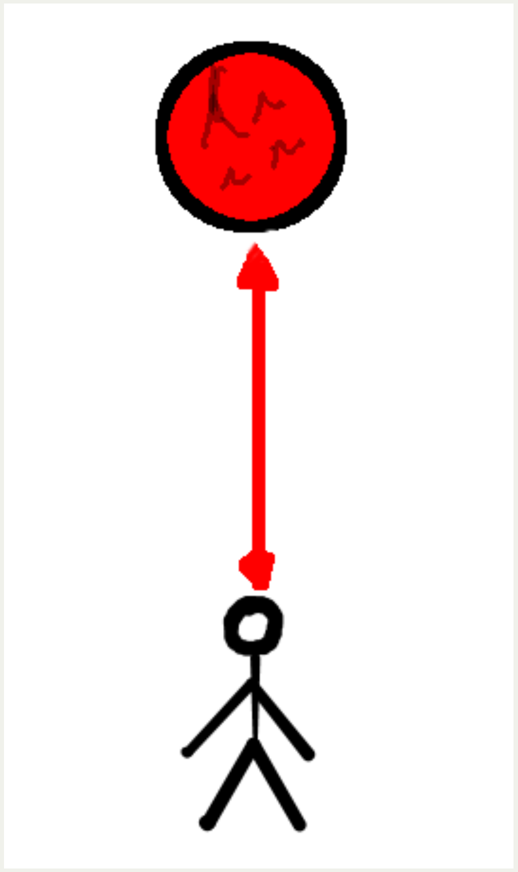
$$train(x_0, x_1, ..., x_m) -> Blackbox$$

$$Blackbox(x) -> Score$$

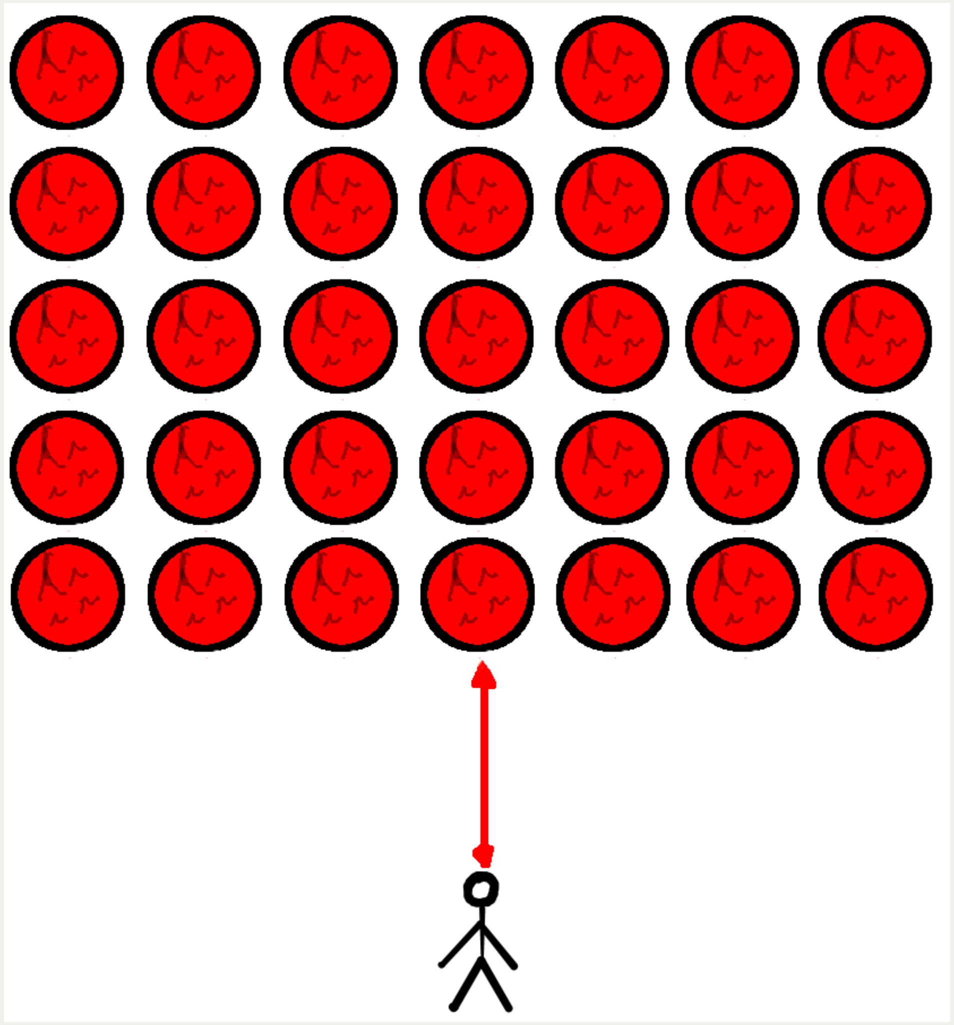
Scoring time



Training time



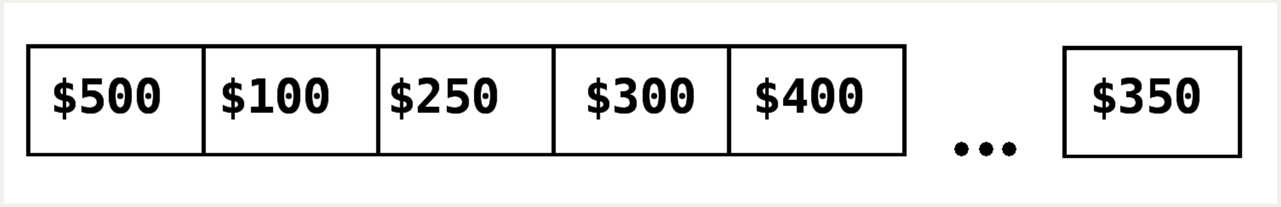
Training time



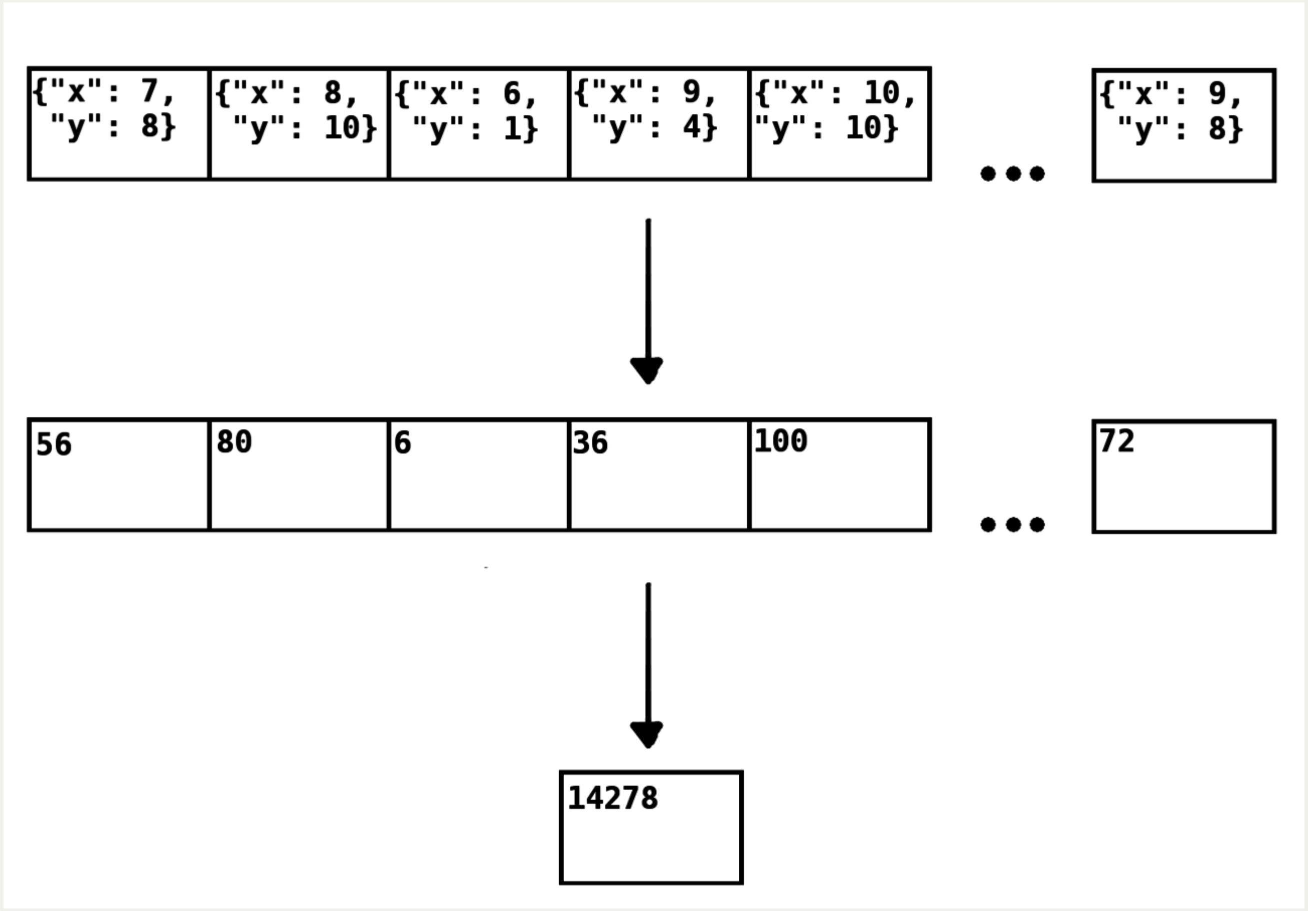
Apache Spark

- Large scale data processing
- High performance
- Easy to use
- Cluster

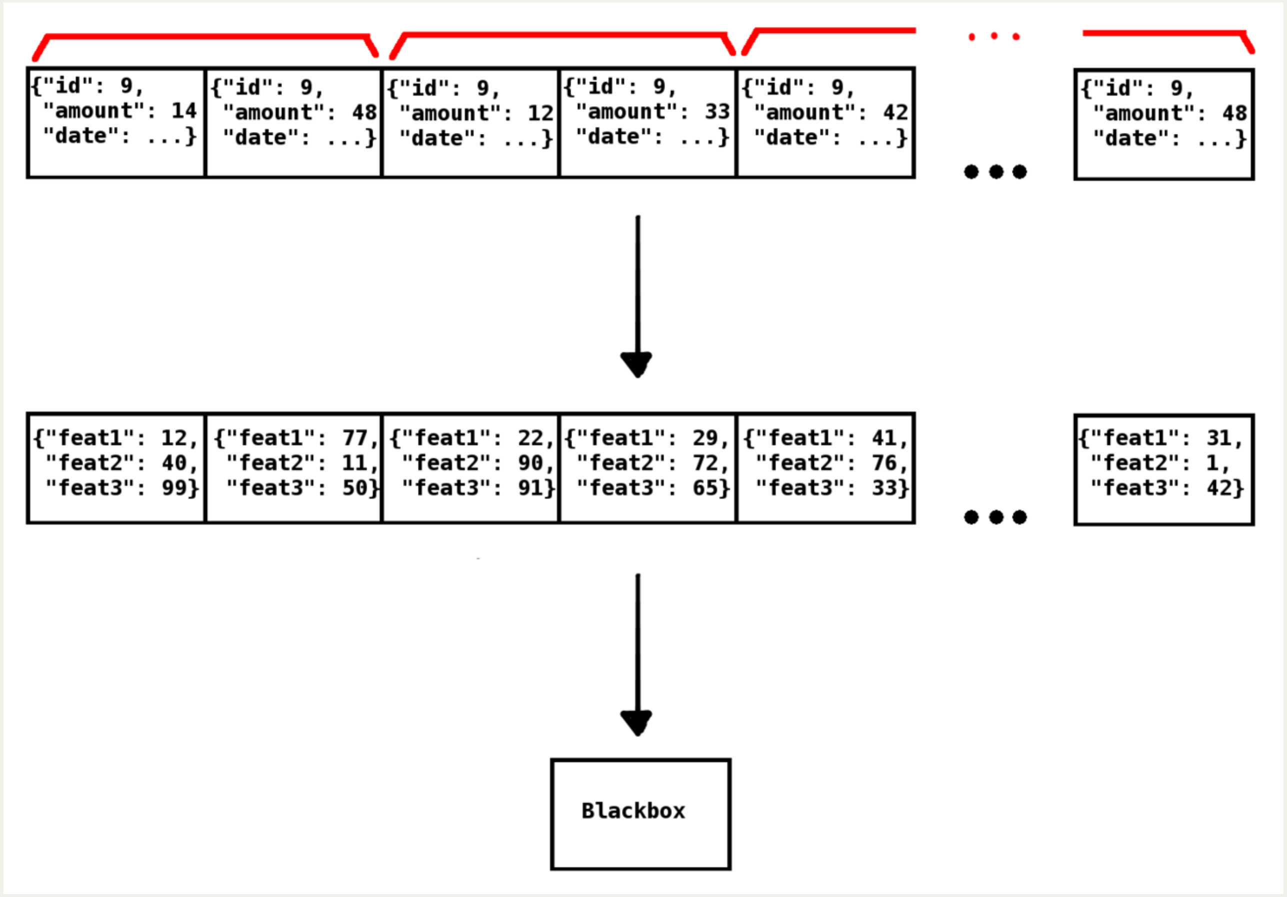
RDD



RDD

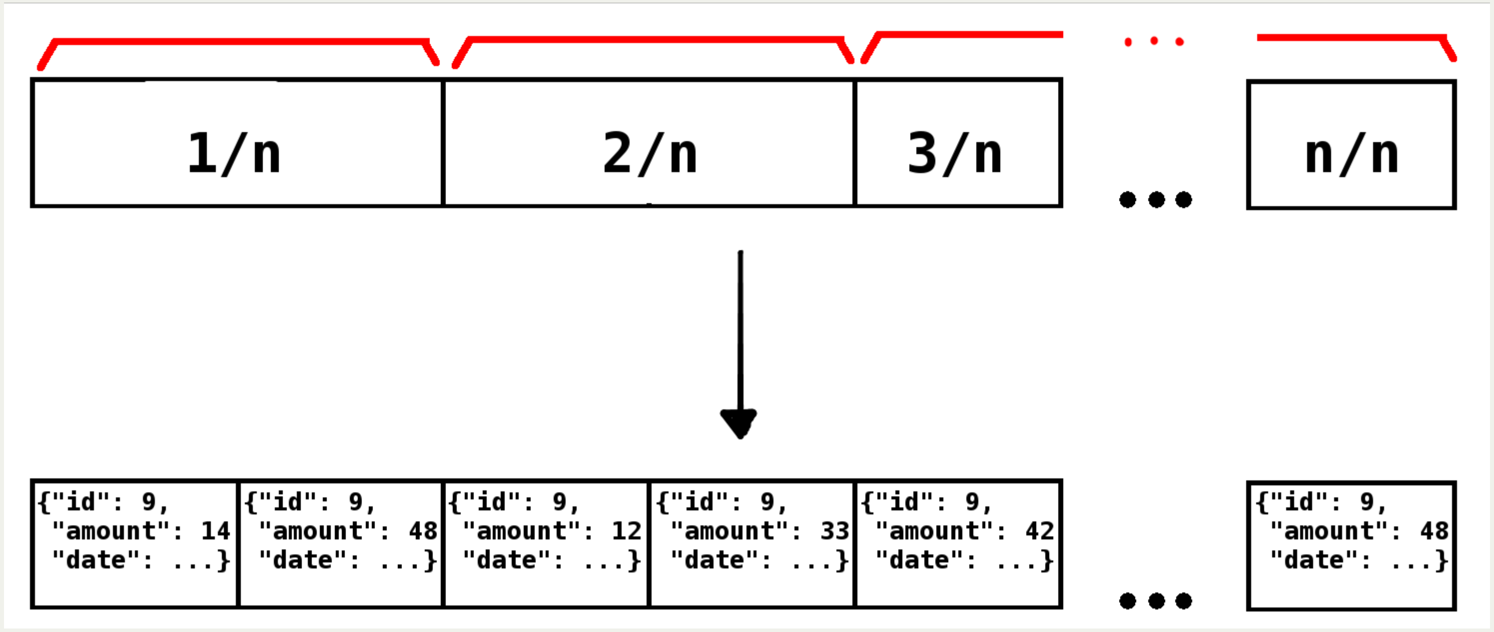


RDDs: our use case



{"id": 9, "amount": 14 "date": ...}
{"id": 9, "amount": 48 "date": ...}
{"id": 9, "amount": 12 "date": ...}
{"id": 9, "amount": 33 "date": ...}
{"id": 9, "amount": 42 "date": ...}
{"id": 9, "amount": 48 "date": ...}
{"feat1": 12, "feat2": 40, "feat3": 99}
{"feat1": 77, "feat2": 11, "feat3": 50}
{"feat1": 22, "feat2": 90, "feat3": 91}
{"feat1": 29, "feat2": 72, "feat3": 65}
{"feat1": 41, "feat2": 76, "feat3": 33}
{"feat1": 31, "feat2": 1, "feat3": 42}
Blackbox

RDDs: our use case



Sharding queries

- Regular query
- Base entity
- Filter using mod

Sharding queries

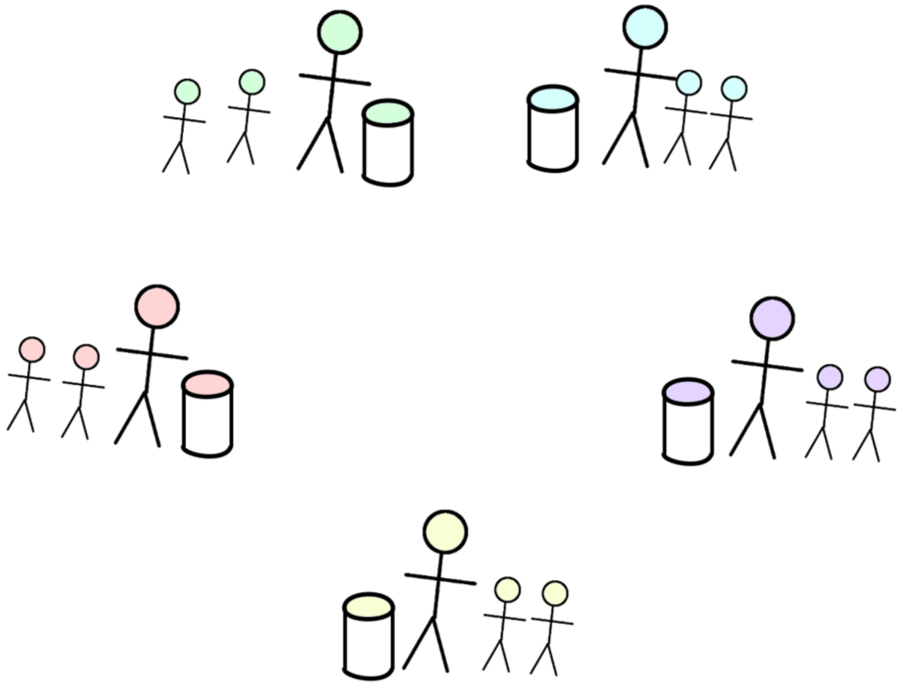
```
{:query
   {:find  [?purchase-amount ?bill-amount]]
    :where [[$pur ?purchase :purchase/id]
             [$pur ?purchase :purchase/bill-id ?bill-id]
             [$pur ?purchase :purchase/amount ?purchase-amount]

             [$bil ?bill :bill/id ?bill-id]
             [$bil ?bill :bill/amount ?bill-amount]]}}
```

Sharding queries

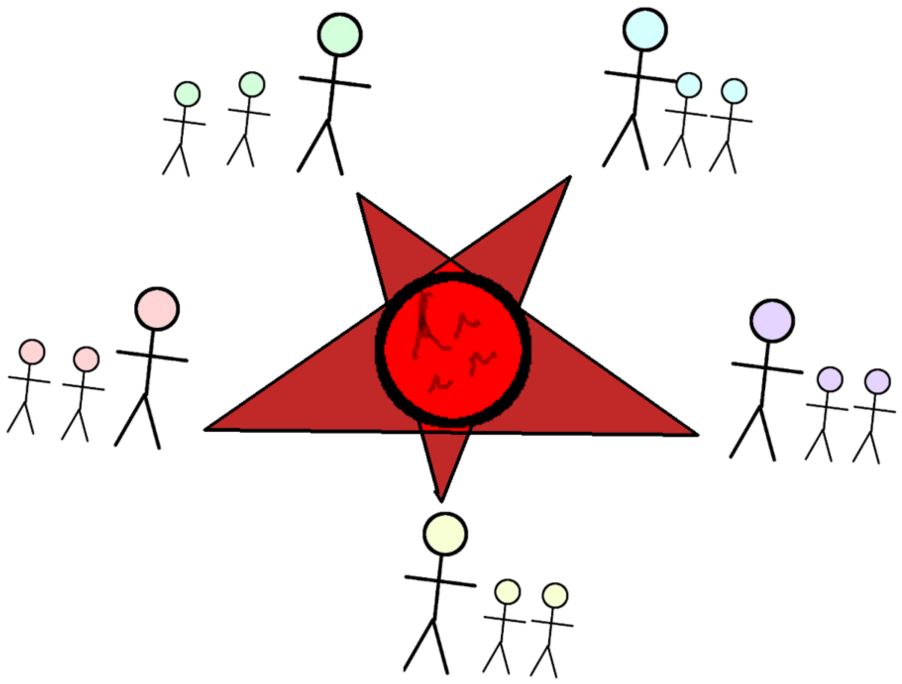
```
{:query
   {:find  [?purchase-amount ?bill-amount]]
    :in    [?shard-index ?shard-count]                    ;; <---------
    :where [[$pur ?purchase :purchase/id]
            [(mod ?purchase ?shard-count) ?shard-index] ;; <---------
            [$pur ?purchase :purchase/bill-id ?bill-id]
            [$pur ?purchase :purchase/amount ?purchase-amount]

            [$bil ?bill :bill/id ?bill-id]
            [$bil ?bill :bill/amount ?bill-amount]]}
 :args [0 35]}                                            ;; <---------
```

Analysing

Data access



Data access



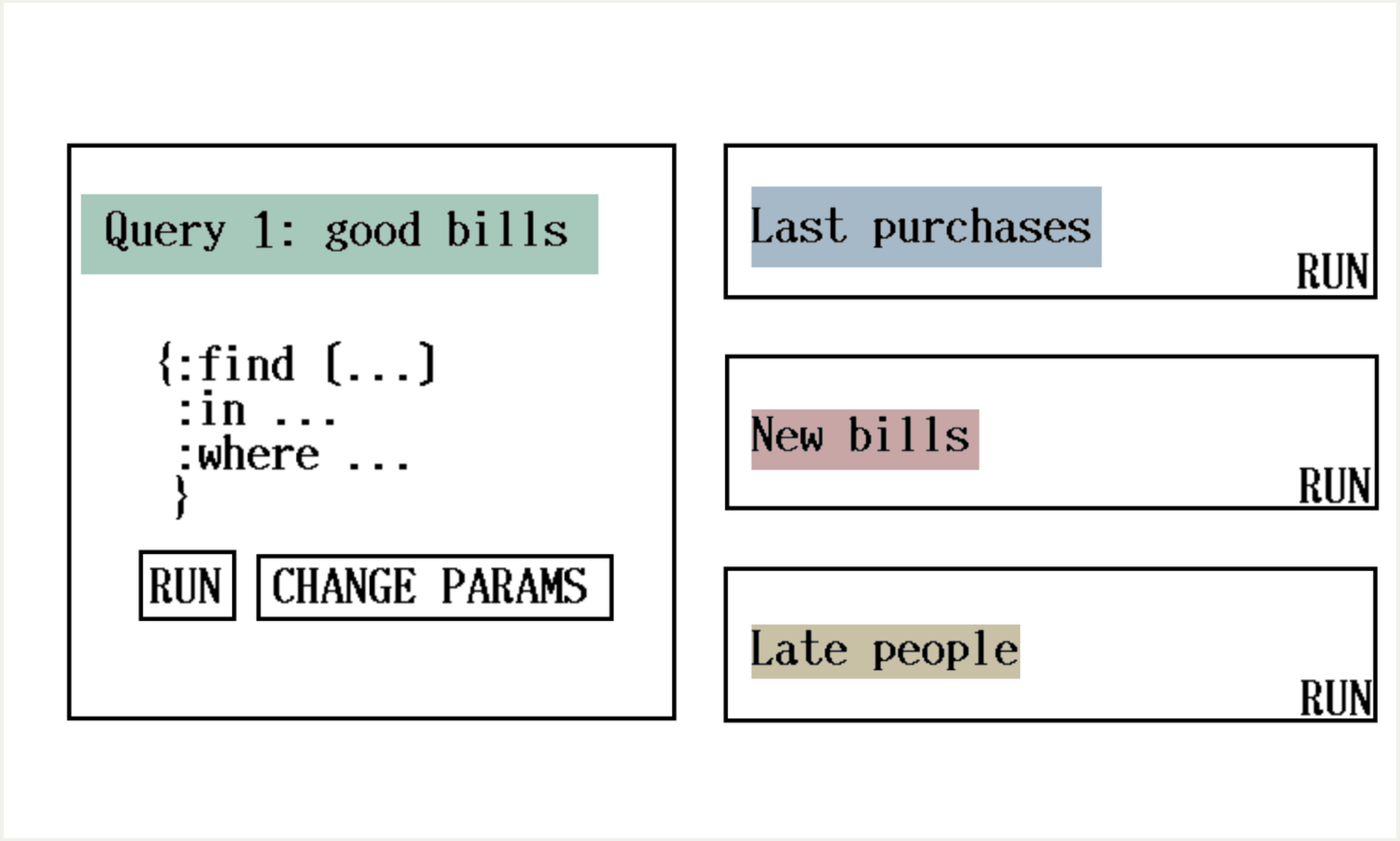
Learning curve

```
{:as-of #inst "2014-11-15T13:37:42Z"
 :args ["pur-8"]
 :query
   {:find  [?name (max ?amount)]]
    :in    [?purchase-id]
    :where [[$pur ?purchase  :purchase/id ?purchase-id]
            [$pur ?purchase  :purchase/customer-id ?customer-id]

            [$cus ?customer  :customer/id ?customer-id]
            [$cus ?customer  :customer/name ?name]

            [$pur ?purchase2 :purchase/customer-id ?customer-id]
            [$pur ?purchase2 :purchase/amount ?amount]]}}
```

Improving experience through saved queries

- Cross-functional teams
- Tech-savvy people create and save queries
- Other people reuse and learn
- Share data and procedures

UI



```
{:find [...]
 :in ...
 :where ...
 }
```

Usage statistics

- Used by ALL teams
- More than one million query executions
- 100s (usually 1000s) of queries by each team
- 618 saved queries

Stored procedures

- random_numbers()
- interest()
- late()

Testimonials

- *"<Our tooling> helps me be more assertive and back hypotheses with data from queries I write myself (...) I can get a snapshot of the data and then follow its evolution over a period of time"* - Business analyst
- *"<Our tooling> helps me find corner cases, trace back the origin of data, and figure out why it ended up that way (...)" - Software Engineer*
- *"I like how it's both a querying and analysing tool (...) having a timestamp from an analysis I made, I can choose to reproduce the results or redo it using up-to-date data just minutes before a meeting"* - Data analyst

Final remarks

What we have done

- Solution for scoring
- Solution for training
- Solution for data access

What we didn't have to

- Copy data around
- Duplicate functions/logic
- Create views

:)