Leipziger Forschungszentrum
für Zivilisationserkrankungen

UNIVERSITÄT LEIPZIG

Datomic
Fast at Scale?

Alexander Kiel
alexanderkiel@gmx.net
@alexander_kiel

Universität Leipzig

Motivation

Can a functional database used in projects beyond toy examples?

LIFE Study

- Cohort Study in Leipzig
- 10.000 Adults
- 10.000 Childs with Parents
- 70 Assessments per Participant
- Central data management and storage

Central Research DB

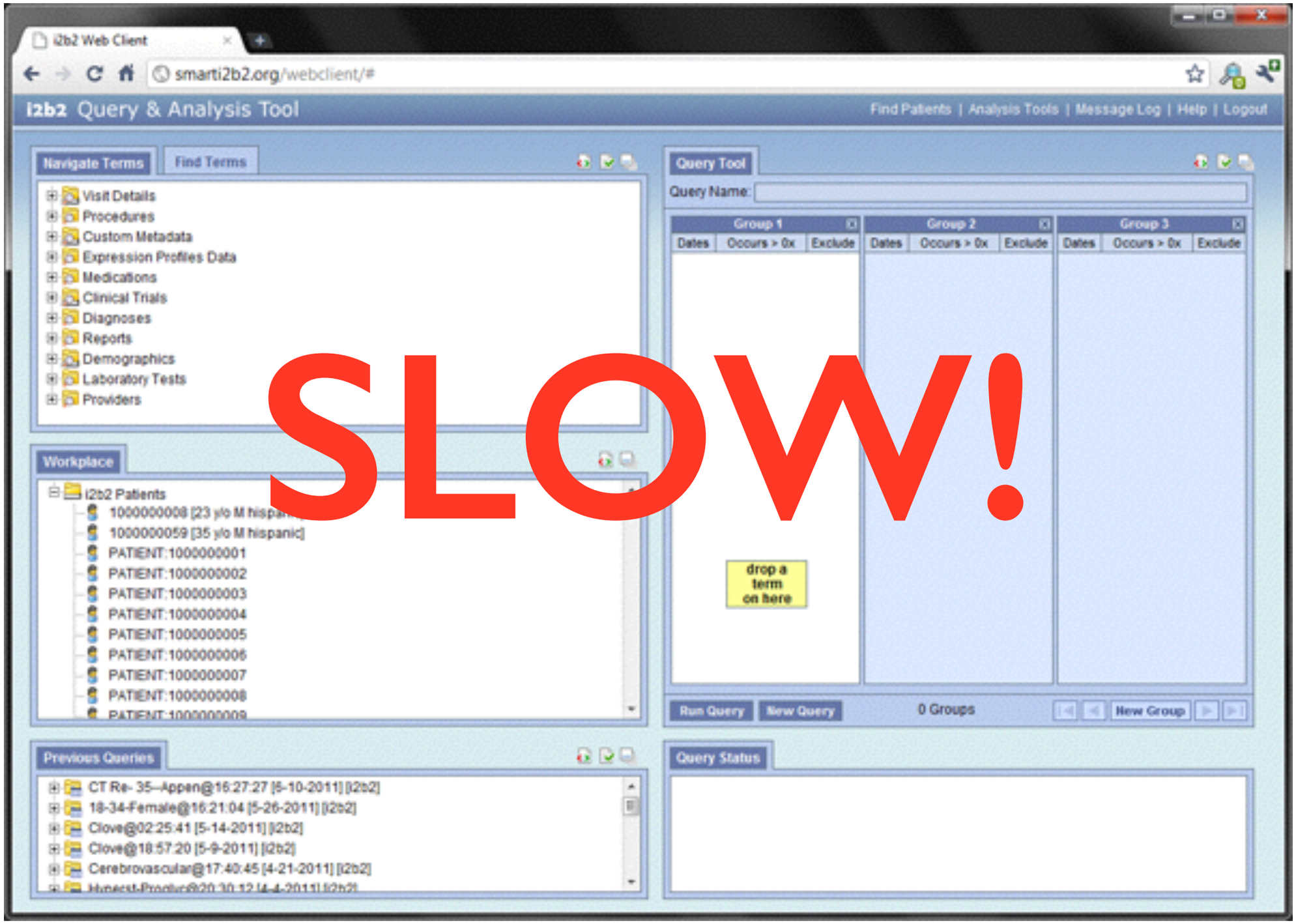
- Currently relational Oracle DB
- 5 GB Data in 600 Tables
- 2 Million Rows / 40.000 Spalten
- 36 Million Non-Null Data Points

The Problem

How many participants do we have for a particular research question?

"Von wie vielen Probanden haben wir eine Soziodemographie und ein Großes Blutbild in der Erwachsenenstudie?"

i2b2


i2b2 Web Client
smarti2b2.org/webclient/#
i2b2 Query & Analysis Tool
Find Patients | Analysis Tools | Message Log | Help | Logout
Navigate Terms
Find Terms
Visit Details
Procedures
Custom Metadata
Expression Profiles Data
Medications
Clinical Trials
Diagnoses
Reports
Demographics
Laboratory Tests
Providers
Query Tool
Query Name:
Group 1
Group 2
Group 3
Dates
Occurs > 0x
Exclude
drop a term on here
Run Query
New Query
0 Groups
New Group
Workplace
i2b2 Patients
1000000059 [35 y/o M hispanic]
PATIENT:1000000001
PATIENT:1000000002
PATIENT:1000000003
PATIENT:1000000004
PATIENT:1000000005
PATIENT:1000000006
PATIENT:1000000007
PATIENT:1000000008
Previous Queries
CT Re- 35--Appen@16:27:27 [5-10-2011] [i2b2]
18-34-Female@16:21:04 [5-26-2011] [i2b2]
Clove@02:25:41 [5-14-2011] [i2b2]
Clove@18:57:20 [5-9-2011] [i2b2]
Cerebrovascular@17:40:45 [4-21-2011] [i2b2]
Query Status
SLOW!

Functional Database

dictionary.com:

> A database which uses a functional language as its query language. Databases would seem [...] **inappropriate** [...] since, a purely functional language would have to **return a new copy** of the **entire database** every time (part of) it was updated.

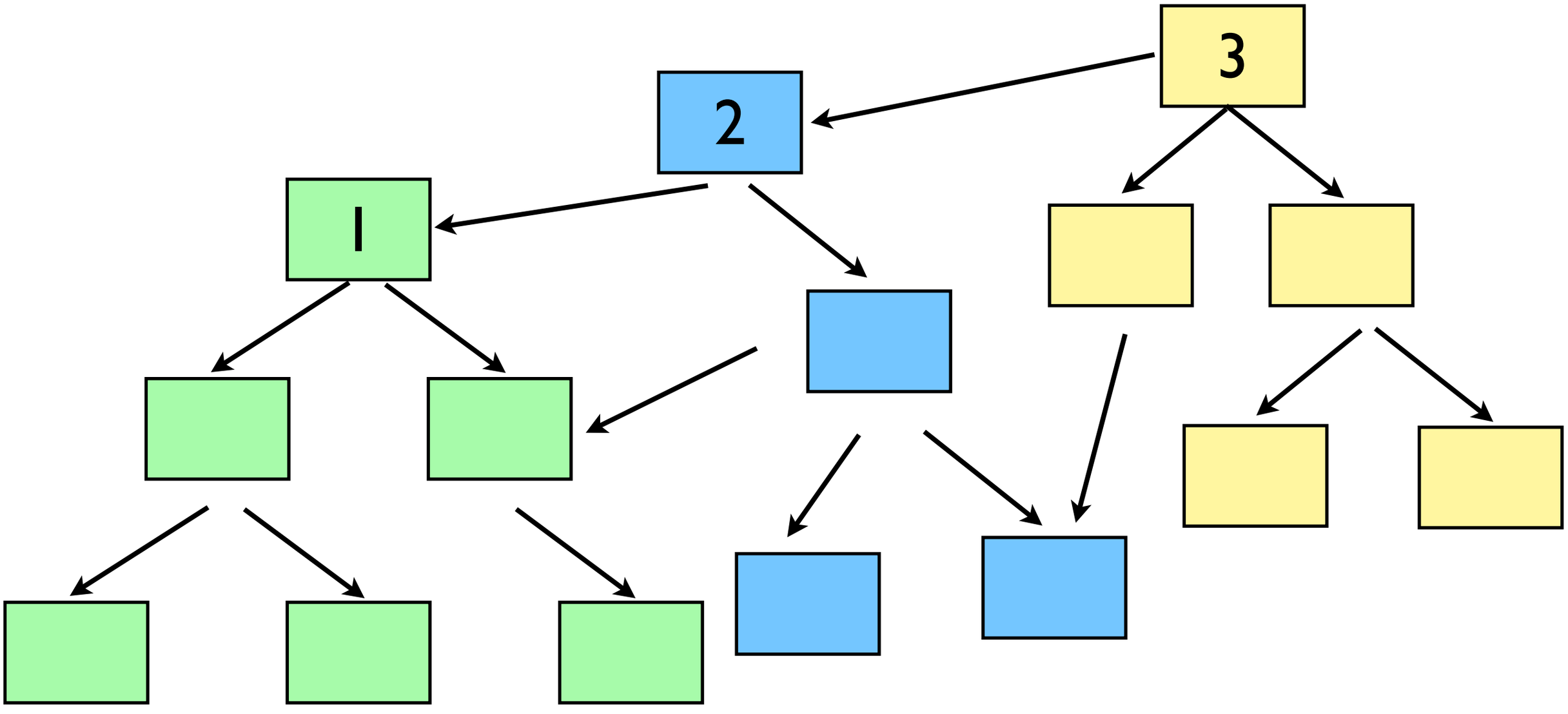
But thats exactly what Datomic does!

Information Model

- Data stored in form of facts
- Fact: atomic information item
- Entity/Attribute/Value/Transaction (Time)

Entity	Attribute	Value	Time
Sally	likes	pizza	02.05.2013

Transition DB -> DB



Indexes - EAVT

E	A	V	Tx	Op
41	:release/name	“Abbey Road”	1100	assert
42	:release/name	“Magical Mystery Tour”	1007	assert
42	:release/year	1967	1007	assert
42	:release/artistCredit	“The Beatles”	1007	assert
43	:release/name	“Let It Be”	1234	assert

Indexes - AEVT

A	E	V	Tx	Op
:release/artistCredit	42	“The Beatles”	1007	assert
:release/name	41	“Abbey Road”	1007	assert
:release/name	42	“Magical Mystery Tour”	1007	assert
:release/name	43	“Let It Be”	1234	assert
:release/year	42	1967	1007	assert

Indexes - AVET

A	V	E	Tx	Op
:release/name	“Abbey Road”	41	1199	assert
:release/name	“Let It Be”	43	1234	assert
:release/name	“Let It Be”	55	2367	assert
:release/name	“Magical Mystery Tour”	42	1007	assert
:release/year	1967	42	1007	assert
:release/year	1984	55	2367	assert

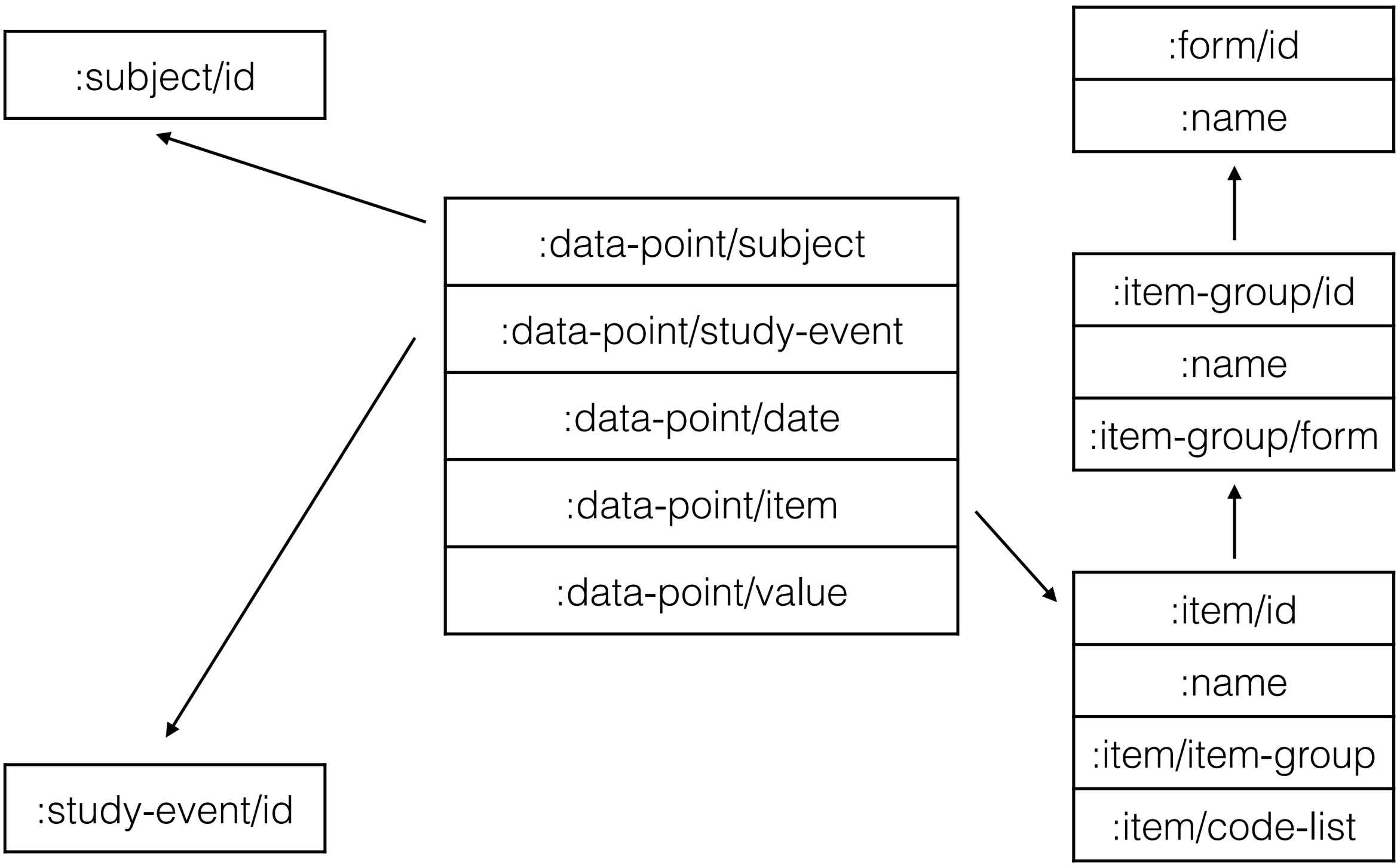
Indexes - VAET

Release

V	A	E	Tx	Op
100	:release/artists	41	1007	assert
100	:release/artists	42	1007	assert
100	:release/artists	43	1007	assert
200	:release/artists	55	2367	assert

Artist

Schema Research DB



Demo

Lens Query Language

```
[[[:form "T00001" ["A1_HAUPT01"]]]
 [[:form "T00505" ["A1_HAUPT01"]]]]
```

Lens Query Language

conjunction

```
[[<atom> <atom> <atom>]
 [<atom> <atom> <atom>]]
```

disjunction

atom:

```
[<type> <id> [<study-event-id> …]]
```

types:

```
:form
:item-group
:item
:code-list-item
```

Atom Query Execution

Datomic Datalog Query

```
(q `[:find (distinct ?s)
     :in $ ?i
     :where [?e :data-point/item ?i]
            [?e :data-point/subject ?s]]
db item-id)
```

Atom Query Execution

Datomic Datalog Query

```
(q '[:find (distinct ?s)
     :in $ ?i
 :where [?e :data-point/item ?i]
        [?e :data-point/subject ?s]]
db item-id)
```

Database Reference and Arguments

Atom Query Execution

Datomic Datalog Query

```
(q `[:find (distinct ?s)
     :in $ ?i
     :where [?e :data-point/item ?i]
            [?e :data-point/subject ?s]]
db item-id)
```

Unification of Clauses

Atom Query Execution

Datomic Datalog Query

```
(q '[:find (distinct ?s)
     :in $ ?i
 :where [?e :data-point/item ?i]
        [?e :data-point/subject ?s]]
db item-id)
```

Entity

Atom Query Execution

Datomic Datalog Query

```
(q `[:find (distinct ?s)
     :in $ ?i
     :where [?e :data-point/item ?i]
            [?e :data-point/subject ?s]]
db item-id)
```

Attribute

Atom Query Execution

Datomic Datalog Query

```
(q `[:find (distinct ?s)
     :in $ ?i
     :where [?e :data-point/item ?i]
            [?e :data-point/subject ?s]]
db item-id)
```

Value

Atom Query Execution

```
[?e :data-point/item ?i]
```

1

Reverse Index

V	A	E
100	:data-point/item	23
100	:data-point/item	42
100	:data-point/item	96

Atom Query Execution

Reverse Index

```
[?e :data-point/item ?i]
```

1

V	A	E
100	:data-point/item	23
100	:data-point/item	42
100	:data-point/item	96

```
[?e :data-point/subject ?s]
```

2

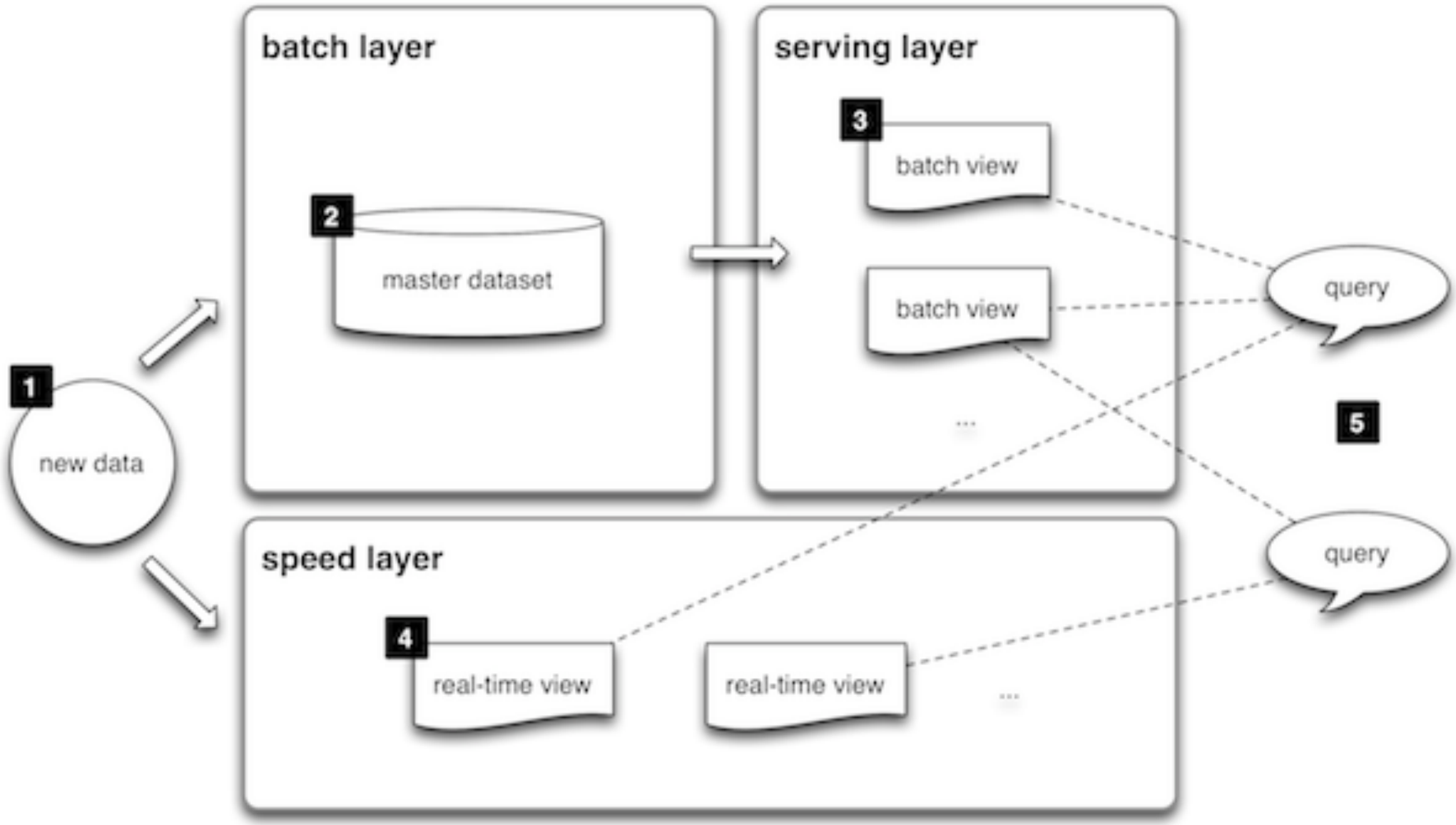
Attr. Index

A	E	V
:data-point/subject	23	1
:data-point/subject	42	1
:data-point/subject	96	2

→ subjects

Lambda Architecture

Nathan Marz



Batch-View Schema

:subject-stat/form
:subject-stat/subjects +

:subject-stat/item
:subject-stat/subjects +

:subject-stat/item-group
:subject-stat/subjects +

:subject-stat/code-list-item
:subject-stat/subjects +

+ = cardinality/many

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
    (:subject-stat/subjects))
```

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
(:subject-stat/subjects))
```

Simple Reference to Stat Entity

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
(:subject-stat/subjects))
```

entity function returns map of attributes

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
    (:subject-stat/subjects))
```

Batch-View Database

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
    (:subject-stat/subjects))
```

Lookup Ref

Batch-View Queries

No Datalog - just Clojure

```
(-> (entity bv [:subject-stat/form 123])
    (:subject-stat/subjects))
```

Navigation of Maps by Keyword

Batch-View Queries

```
(entity bv [:subject-stat/form 123])
```

1

Reverse Index

V	A	E
123	:subject/stat-form	42

```
(:subject-stat/subjects)
```

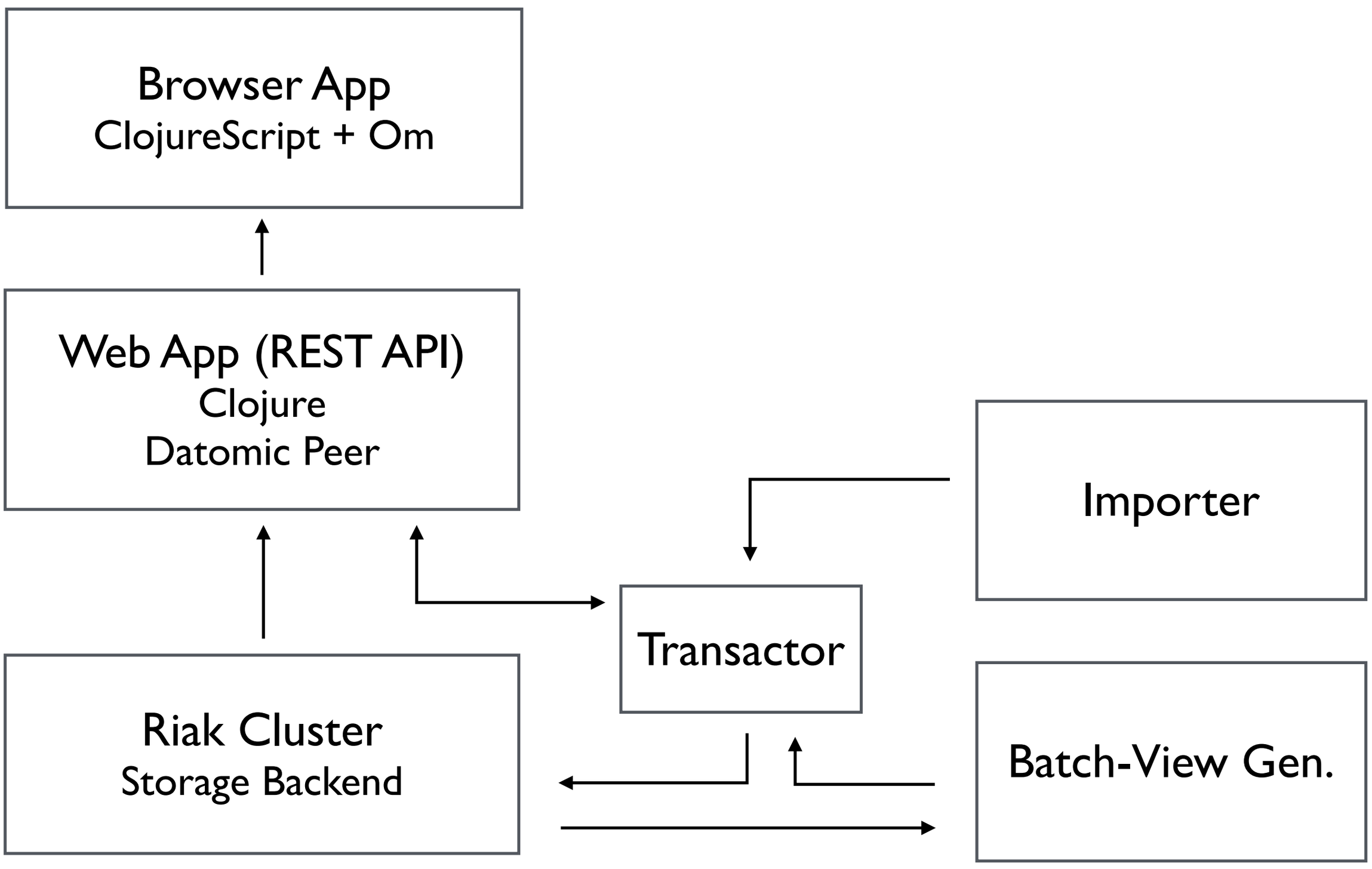
2

Attr. Index

A	E	V
:subject-stat/subjects	42	1
:subject-stat/subjects	42	2
:subject-stat/subjects	42	3

→ **subjects**

Lens Architecture



Thanks