



Datomic

an overview

Ryan Mulligan

Datomic

- Rich Hickey
- Clojure
- Immutable data structures

Datomic

- Database for time-based facts
- Database as a value (immutable)
- Highly-scalable reads
- Simple schemas
- Moves querying from server to application
- Keeps declarative queries, joins, and ACID ("NoNoSQL")
- Closed source



Datoms

Datoms (facts)

entity / attribute / value / transaction (txn.)

ryan / favorite programming language / Java / as of txn. 1

ryan / favorite programming language / Ruby / as of txn. 2

Datoms are immutable

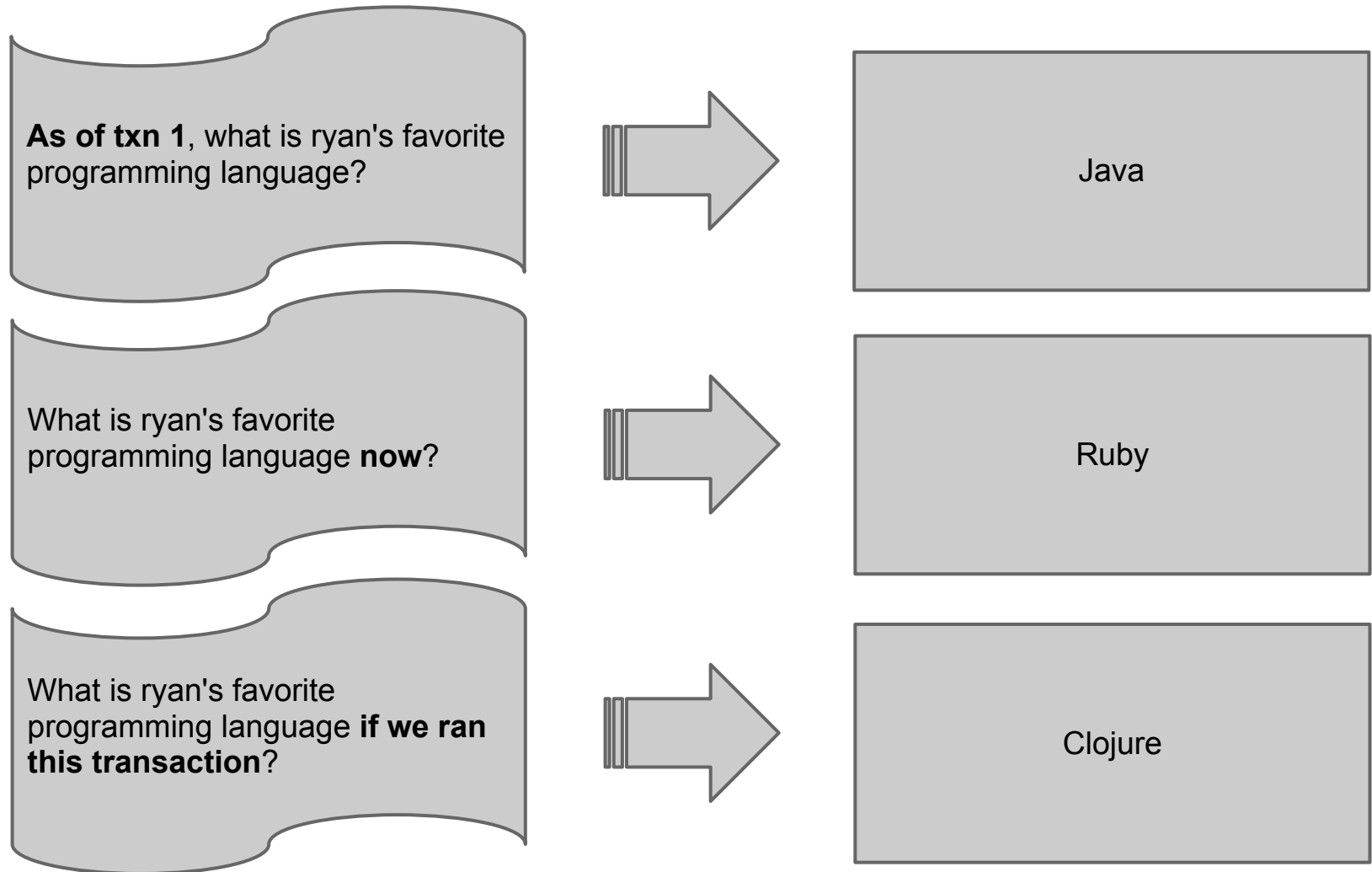
entity / attribute / value / transaction (txn.)

ryan / favorite programming language / Java / **as of txn. 1**

ryan / favorite programming language / Ruby / **as of txn. 1**

invalid

Datoms are not forgotten





Database as a value

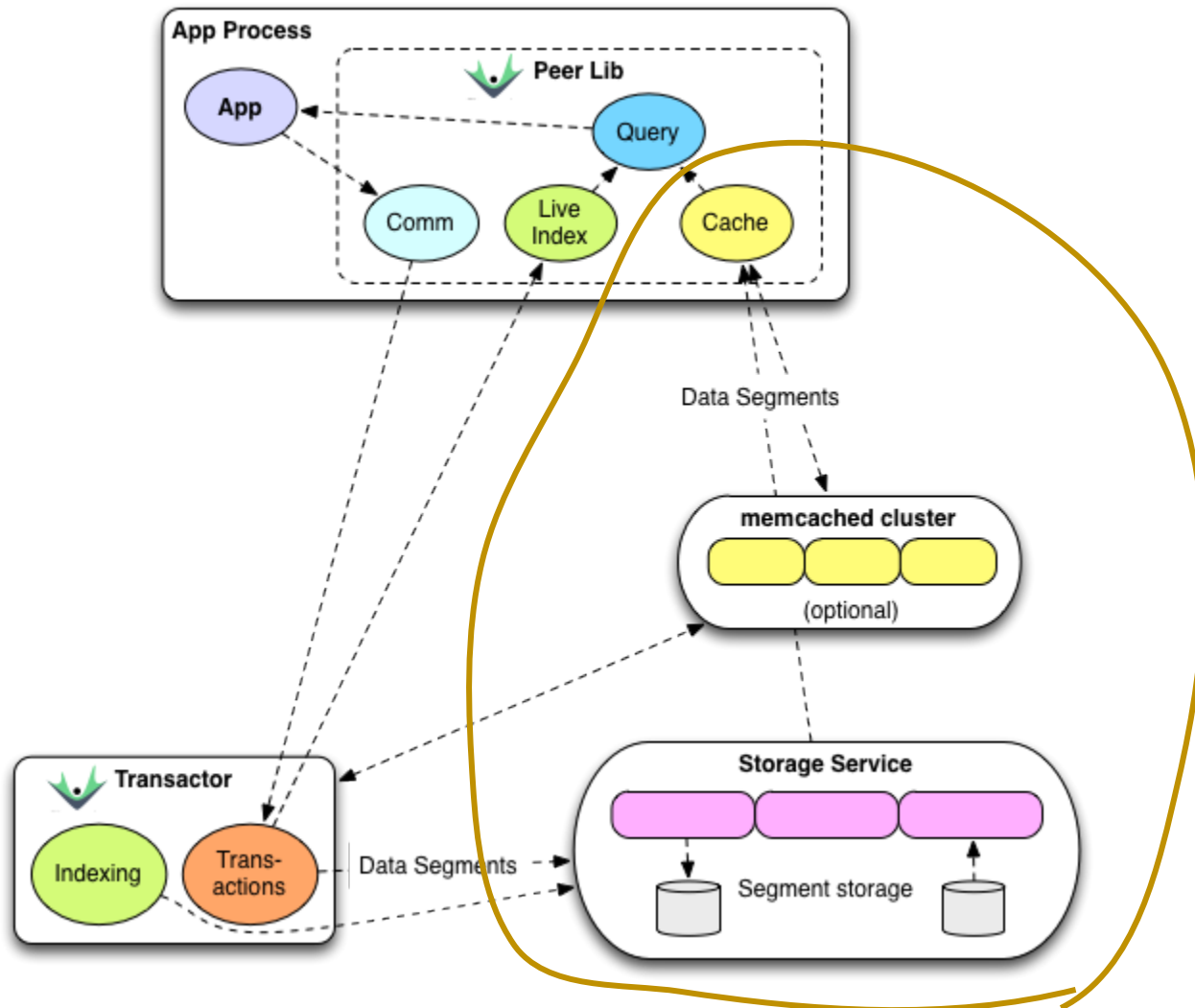
Database is a value

- query is a function of txn number
- safe to communicate the db value to others
- last night?
- don't need to do all your read queries in one go

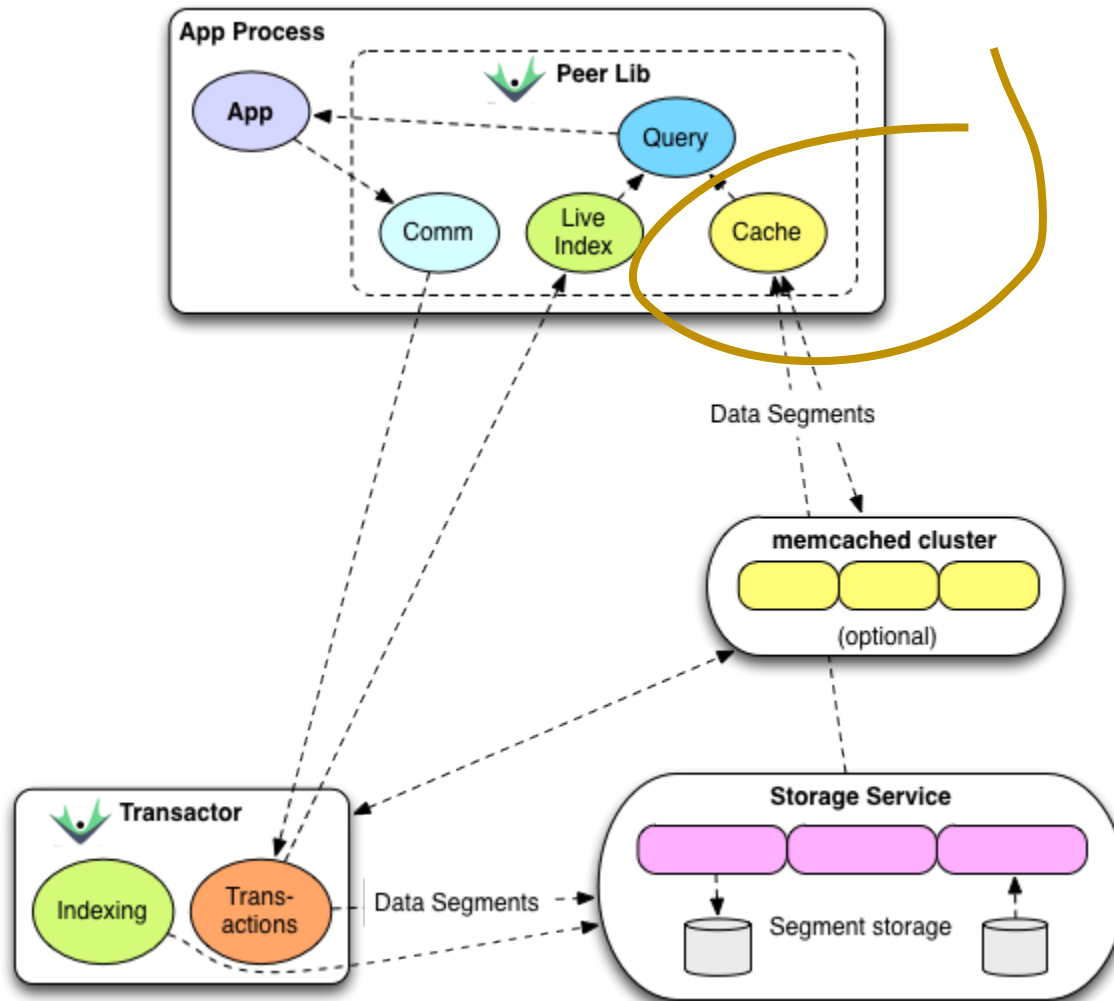


Performance

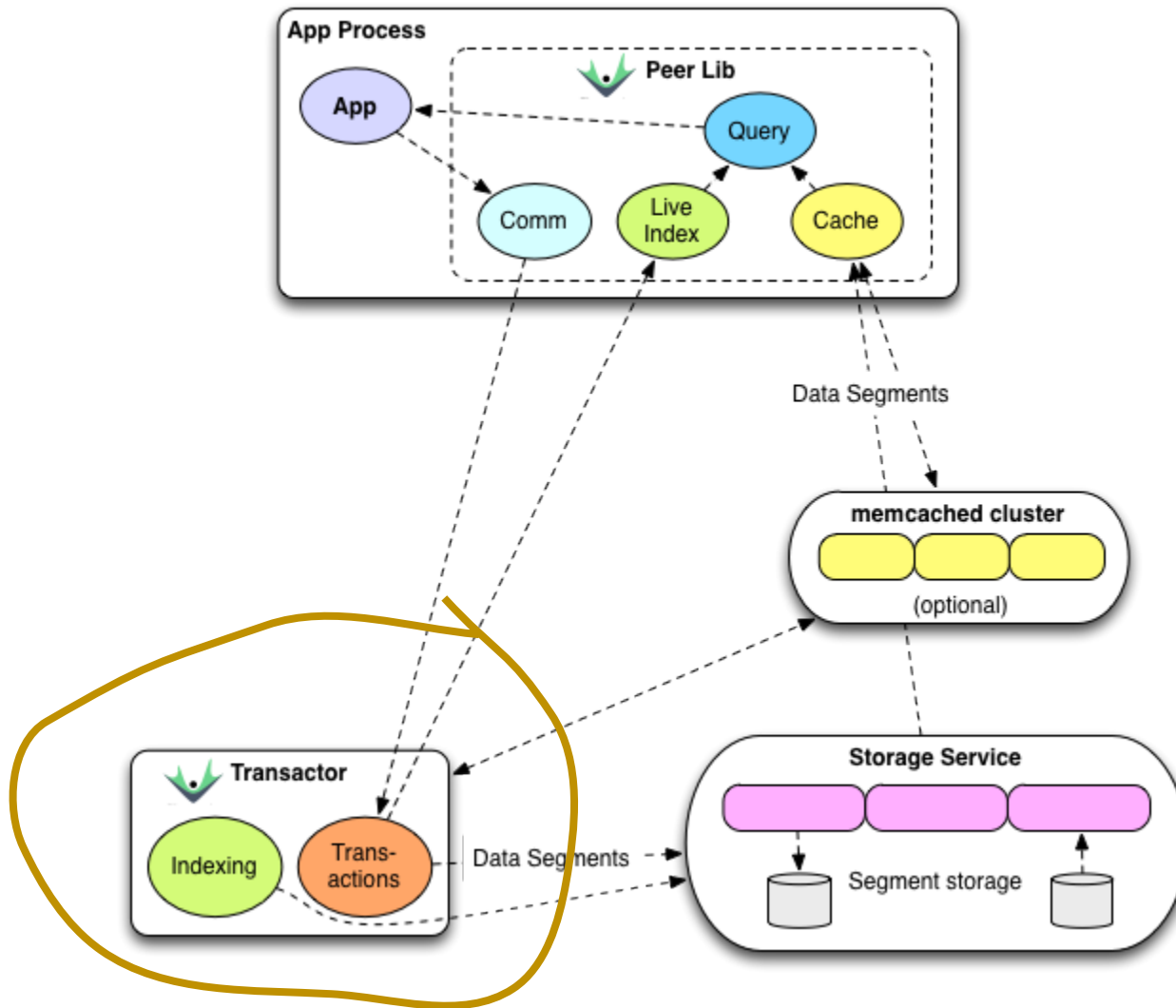
Immutability gives scalable reads



Immutability gives scalable reads



Only one machine for writes





Schema

Schema defines Datom attributes

- Name
 - :favorite_programming_language
- Type
 - :db.type/string
- Cardinality - one or many?
 - :db.cardinality/one

entity / **attribute** / value / transaction (txn.)

Interesting schema type

- has all the types you expect
- but also, `:db.type/ref`

has many computers

```
{:db/id #db/id[:db.part/db]  
 :db/ident :computers  
 :db/valueType :db.type/ref  
 :db/cardinality :db.cardinality/many}
```


Optional schema attributes

- doc, unique, index
- fulltext!
- isComponent for refs
- noHistory

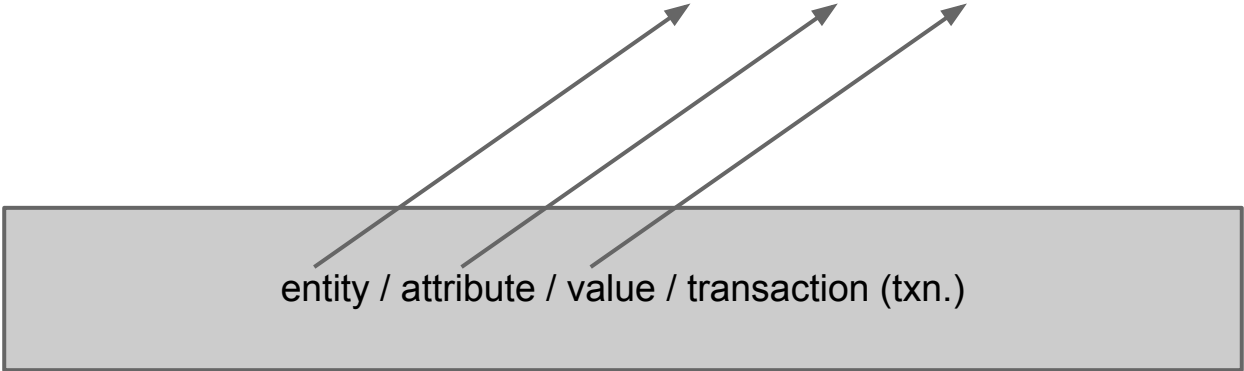


Query

Query with Datalog

```
[ :find ?e :where [?e :age 42]]
```

entity / attribute / value / transaction (txn.)



The diagram illustrates the mapping of a Datalog query to a transaction structure. A gray rectangular box at the bottom contains the text 'entity / attribute / value / transaction (txn.)'. Three arrows originate from this box and point upwards to the query components: the first arrow points to '?e' (representing the entity), the second arrow points to ':age' (representing the attribute), and the third arrow points to '42' (representing the value).

Joins are implicit

```
[ :find ?person ?ip :where [?person :computers ?c]  
                             [?c :ipaddress ?ip]]
```

Variables

```
[ :find ?e :in $ ?age :where [ ?e :age ?age ] ]
```

Querying notes

- happens on client
 - won't slow anyone else down
 - hierarchical data/nesting/recursion is not a problem



Conclusion

Conclusion

- A database for time-based facts
- Database as a value
- Highly-scalable reads
- Simple schemas
- Moves querying from server to application
- Keeps declarative queries, joins, and ACID ("NoNoSQL")



Questions?

Further study

- datomic.com
- <http://www.infoq.com/presentations/Datomic-Database-Value>
- Ruby examples
 - <https://github.com/cldwalker/datomic-client>
 - <https://github.com/crnixon/datomic-sinatra-wiki>

Slides at <https://github.com/ryantm/lvrug-presentations>