DEMAND DRIVEN ARCHITECTURE

KOVAS BOGUTA & DAVID NOLEN, QCON NEW YORK 2015

CLIENTS

- More and more clients everyday
    - Internet of Things
- How to serve N clients with 1 service?

REST SERVICE

- Define logical “resources”
- Client requests a resource
  - But client actually needs to present joined resources
- Bloat resource? Multiple requests?
  - Incidental complexity? Mobile client latency?

SUPPLY DRIVEN ARCHITECTURE

- "You can have anything you want as long as it looks exactly like this"
- Engineering team issues
  - Front end team must request changes
  - N front end teams attacking service team

Our REST Service

THE PROBLEM

- Cannot predict what clients (especially future ones) will actually need

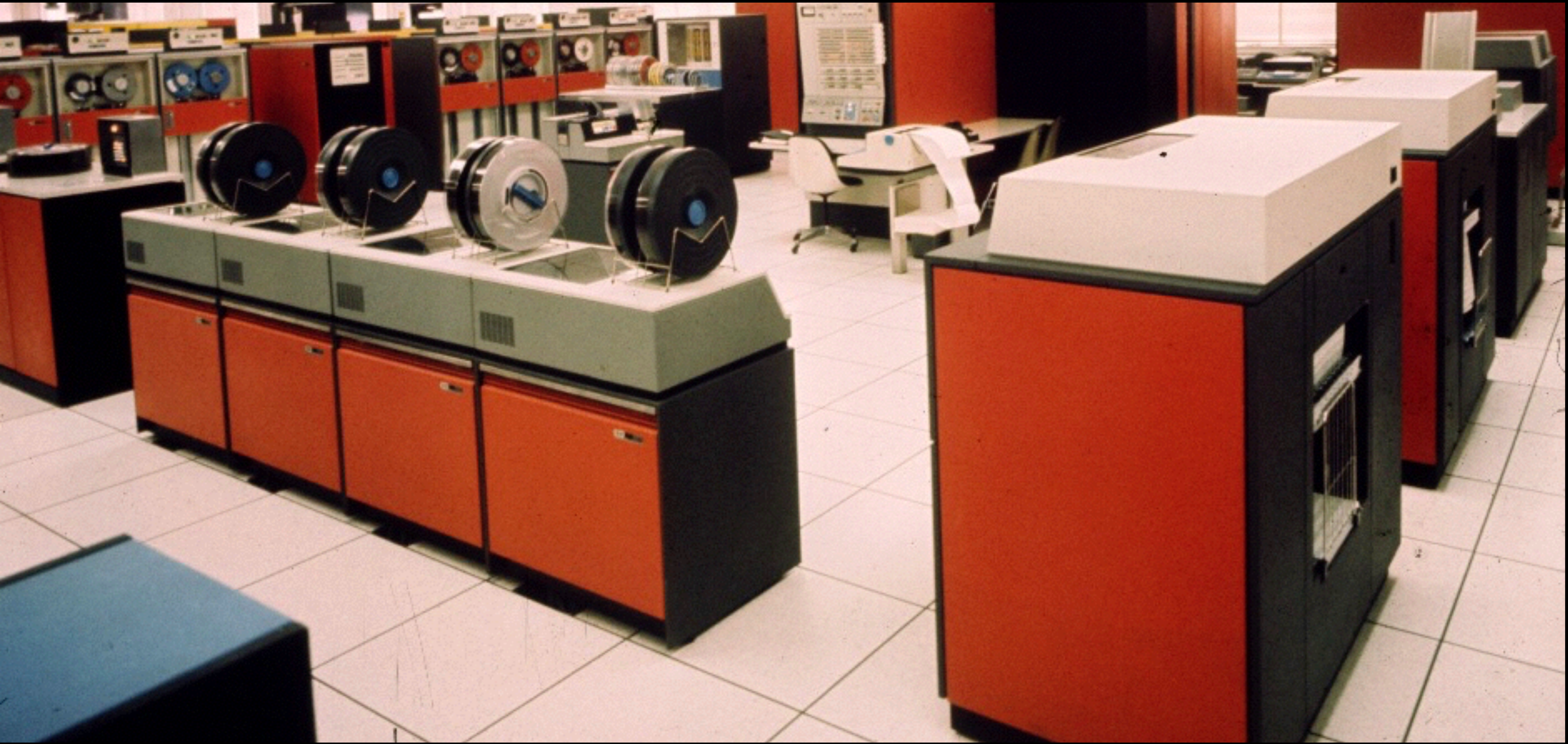
- Cannot put all clients into lockstep with a specific version of the API (Synchronization is expensive)

- Client applications are distributed systems (salad days are over)

Author Jun 2, 2015, 6:59 PM
database design
hierarchical databases

make sure clear not recommending storing data as tree

“This has all happened before”



RDBMS

- Client specifies exactly what they need
- Multiple clients not locked into same canned results
- Batching (latency considerations)

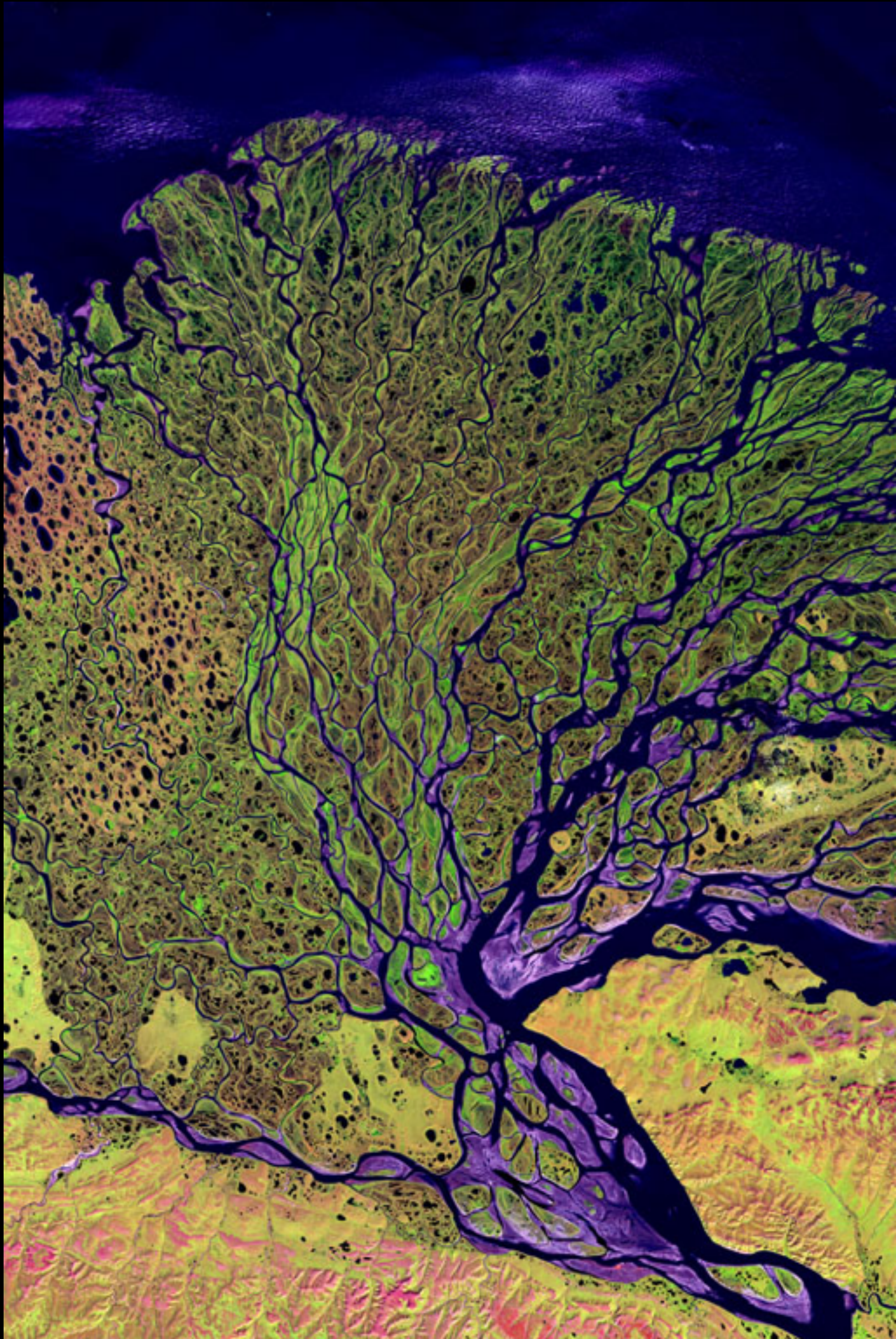
? ? ?

Author Jun 2, 2015, 9:10 PM
known unknowns
open questions

we’re not just going to expose SQL but what are we going to do?

- Can an endpoint provide a restrictive yet expressive query language?
- Can an endpoint evolve with clients?
- Can an endpoint serve multiple tiers of demand from a client?

DEMAND DRIVEN



THE BIG IDEA

- Represent client demand as data
    - Client describes demand, service fulfills
- Variation captured in data, on the client
- Contract between client and service

PRINCIPLES

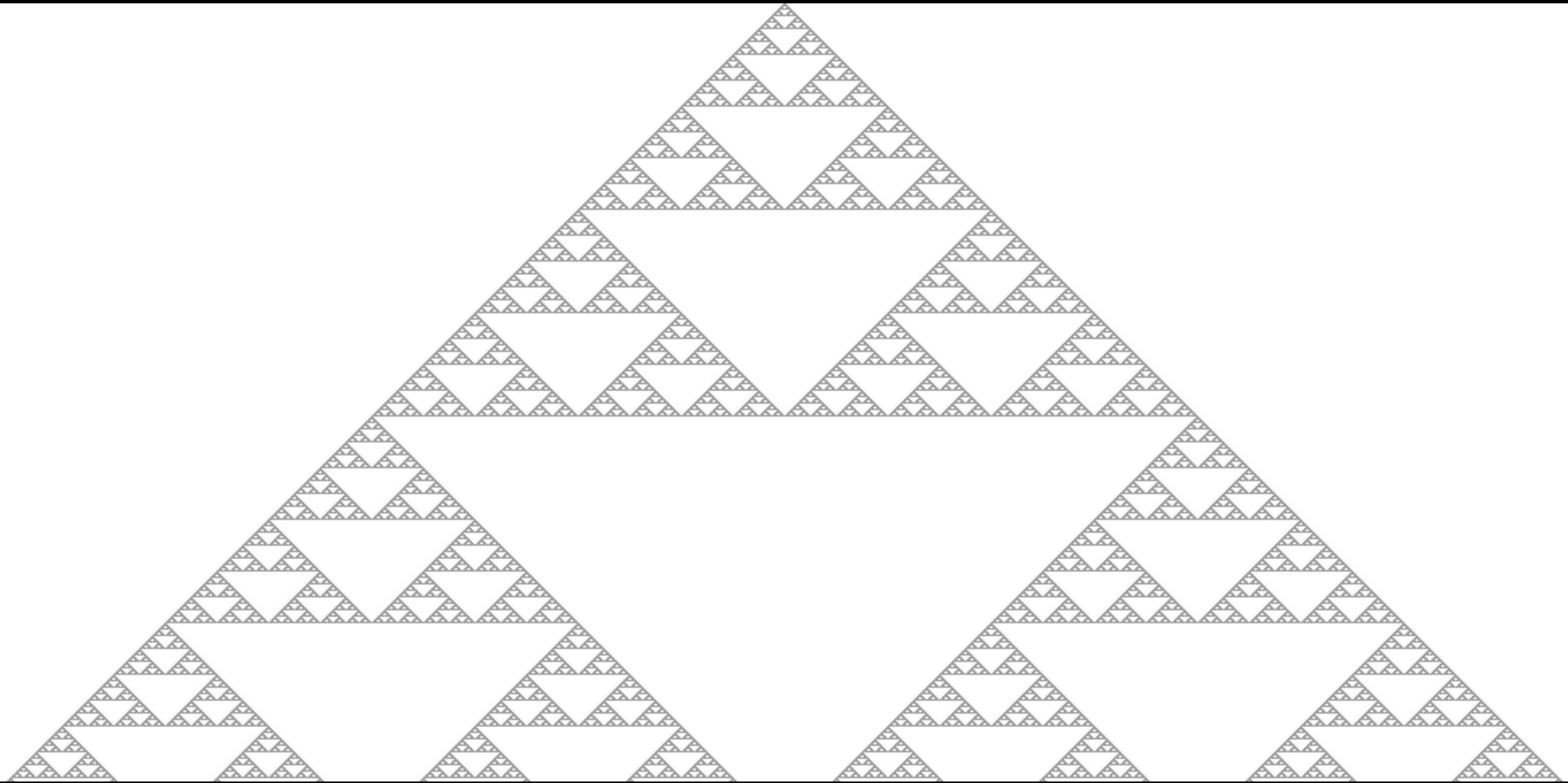
Demand

Author Jun 2, 2015, 7:17 PM
We don’t want to decide what is a resource and what isn’t

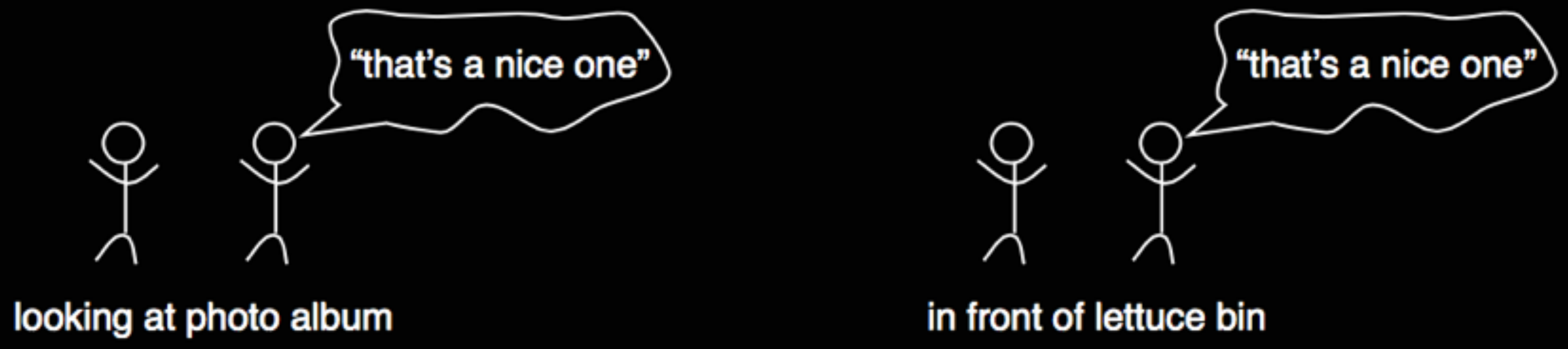
The payload includes a mixture of thing which are and aren’t resources

(We should probably examine and respond to HATEOS)

Composition



Interpretation

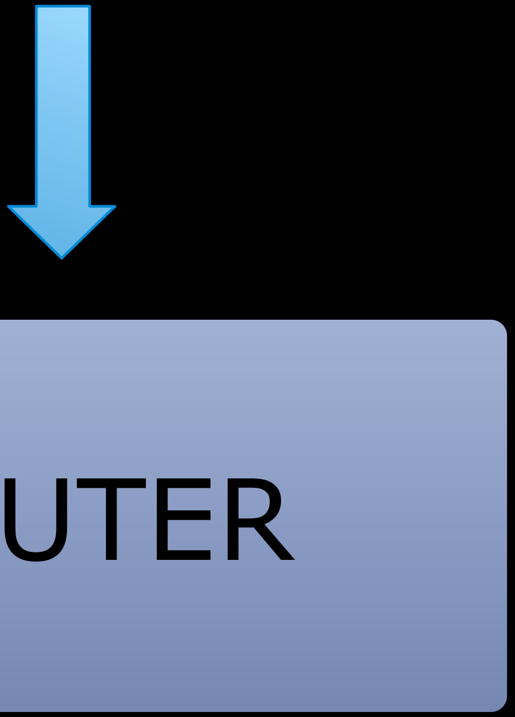


DEMO

```
[{:app/contacts
  [:person/first-name]}]
```

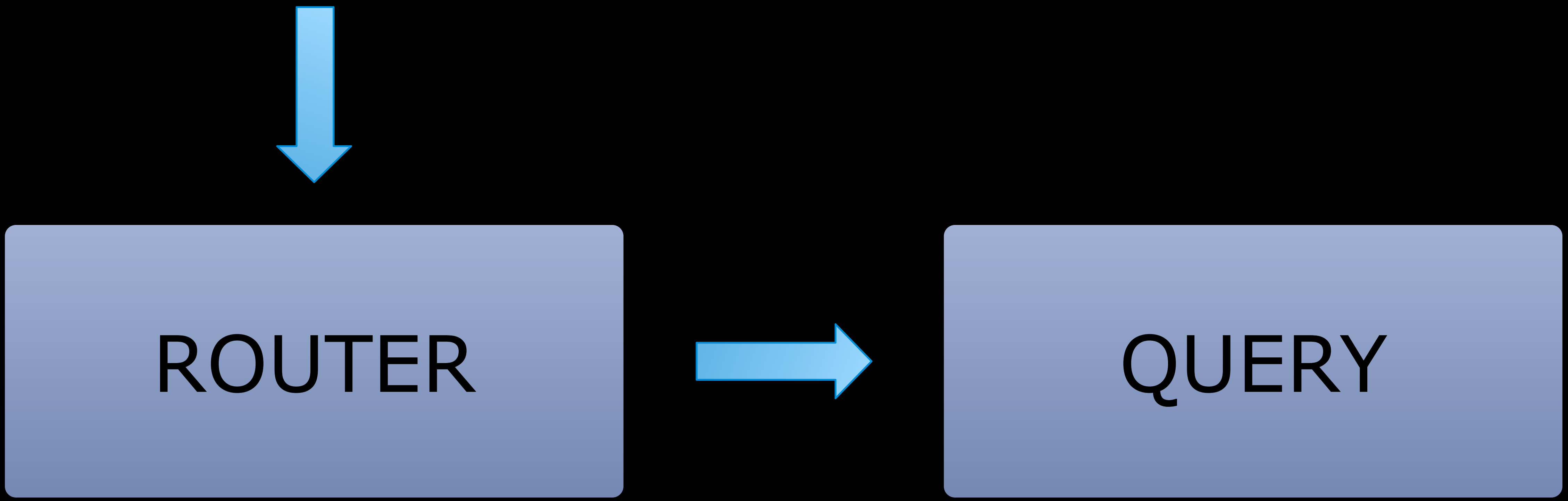
```
[{:app/contacts
  [:person/first-name]}]
```

```
[{:app/contacts
  [:person/first-name]}]
```

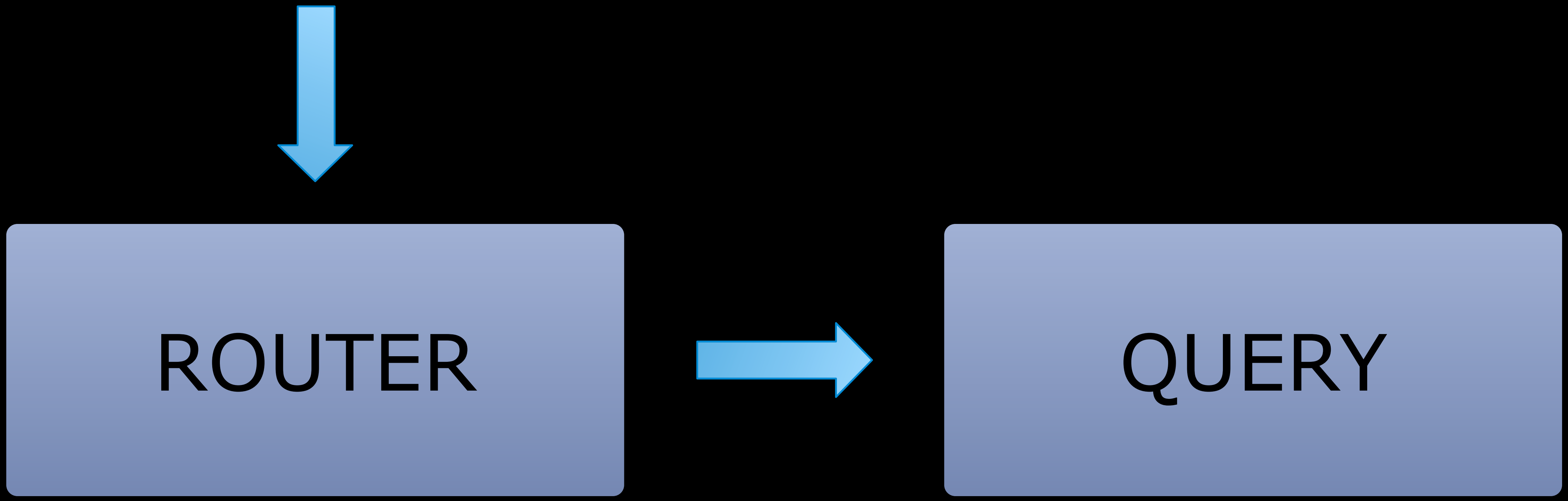


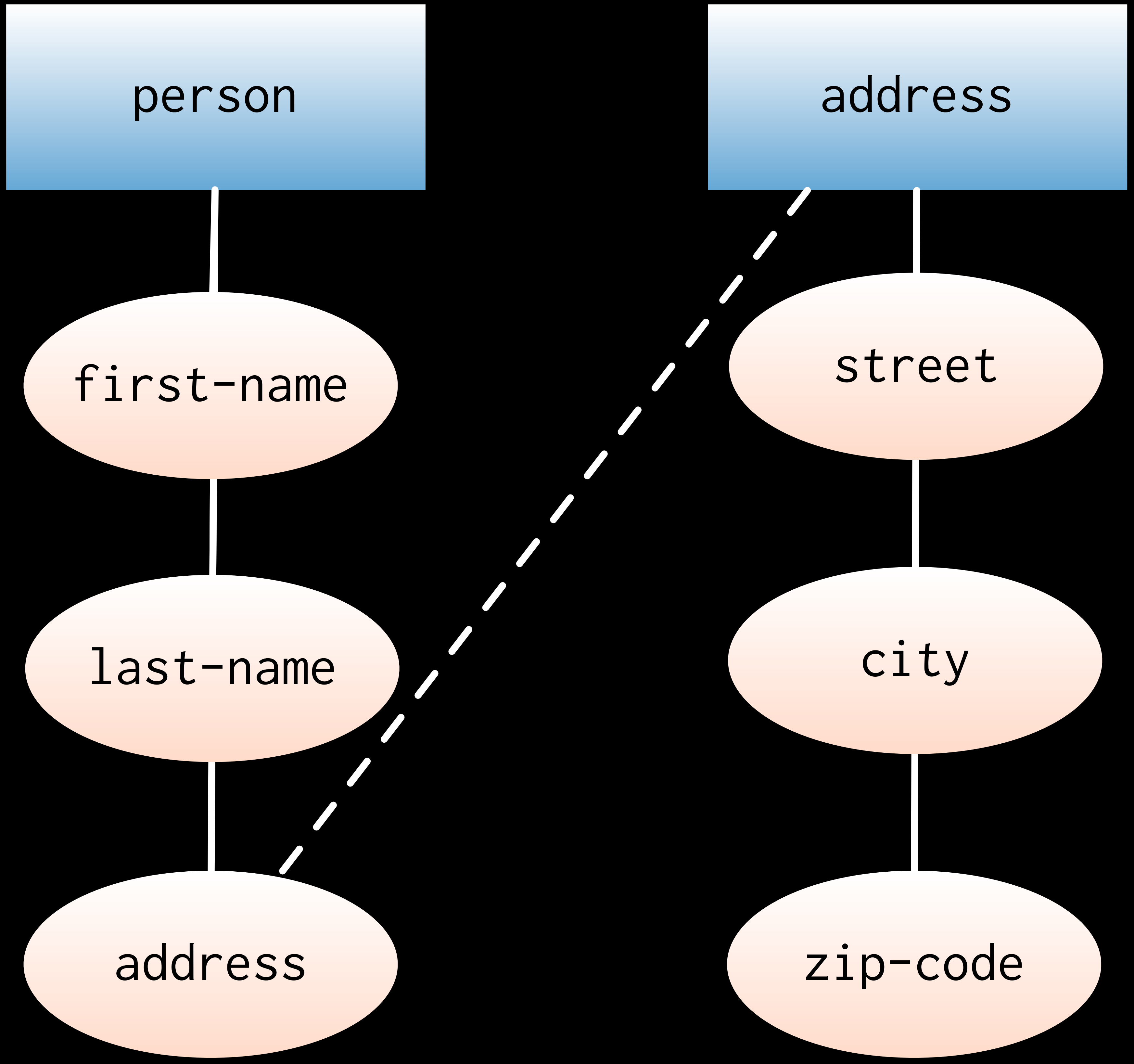
ROUTER

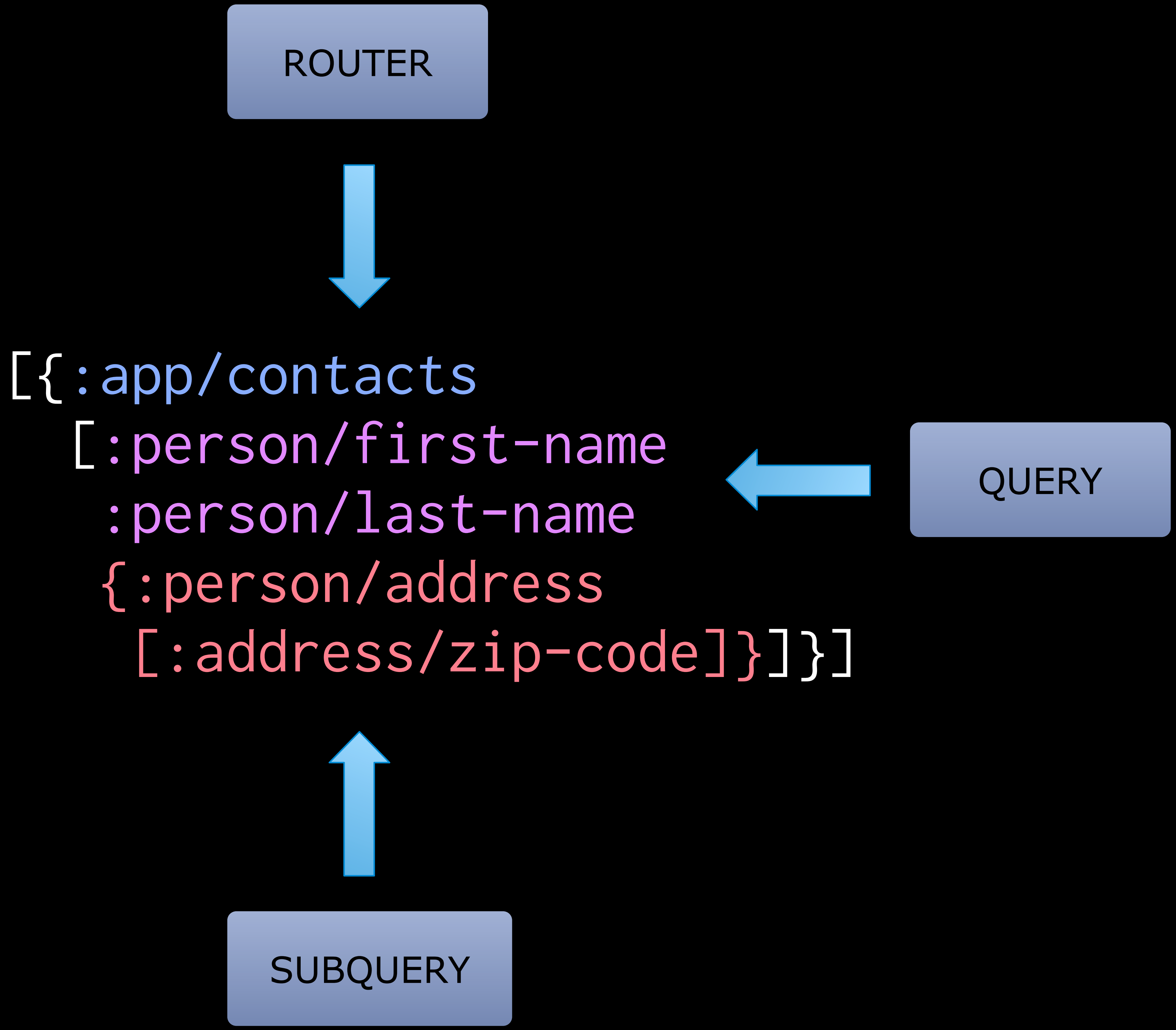
```
[{:app/contacts
  [:person/first-name]}]
```



```
[{:app/contacts
  [:person/first-name
   :person/last-name
   :person/address]}]
```




person
address
first-name
street
last-name
city
address
zip-code



ROUTER

```
[{:app/contacts
  [:person/first-name
   :person/last-name
   {:person/address
    [:address/zip-code]}]}]
```

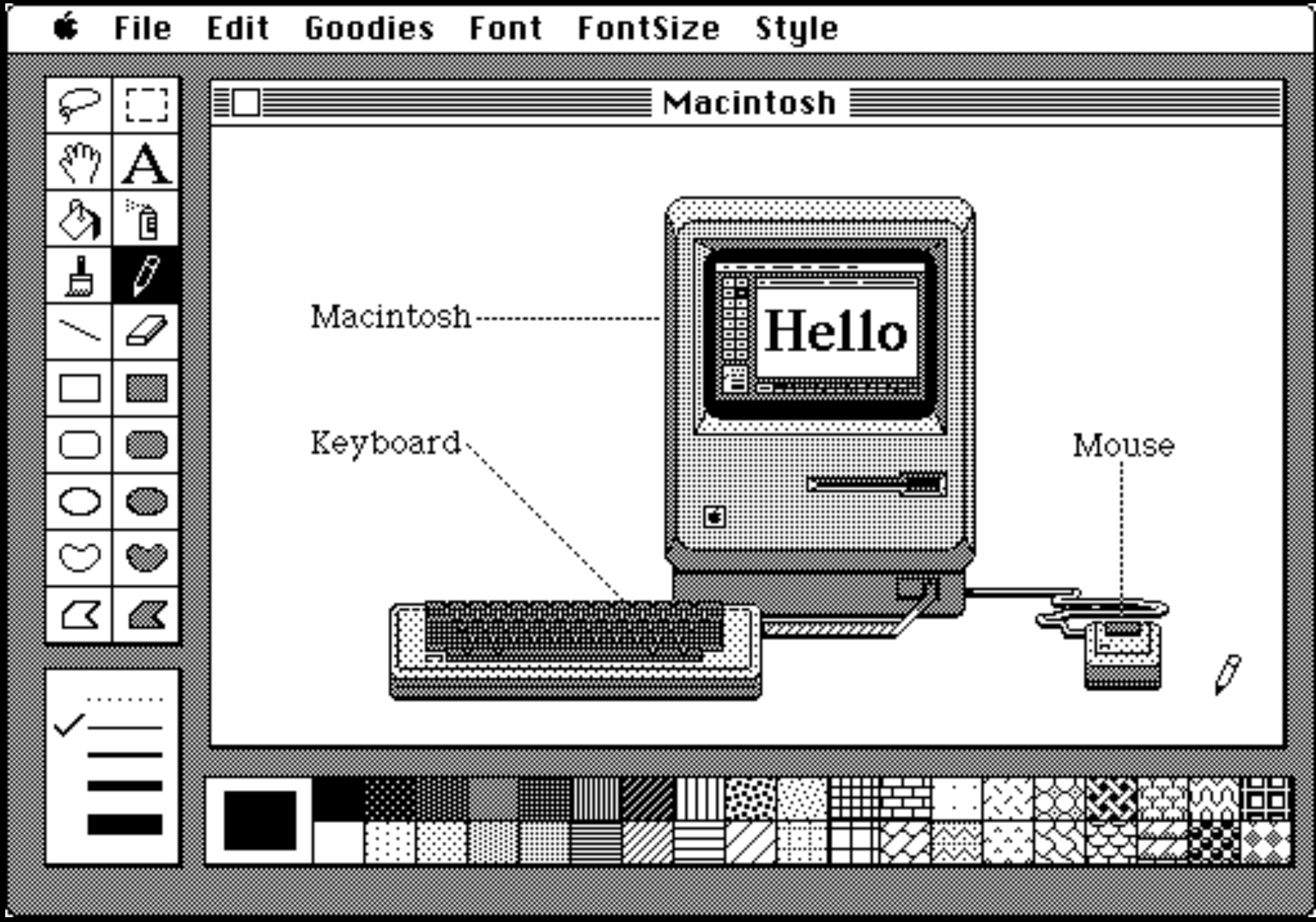
QUERY

SUBQUERY

SUMMARY

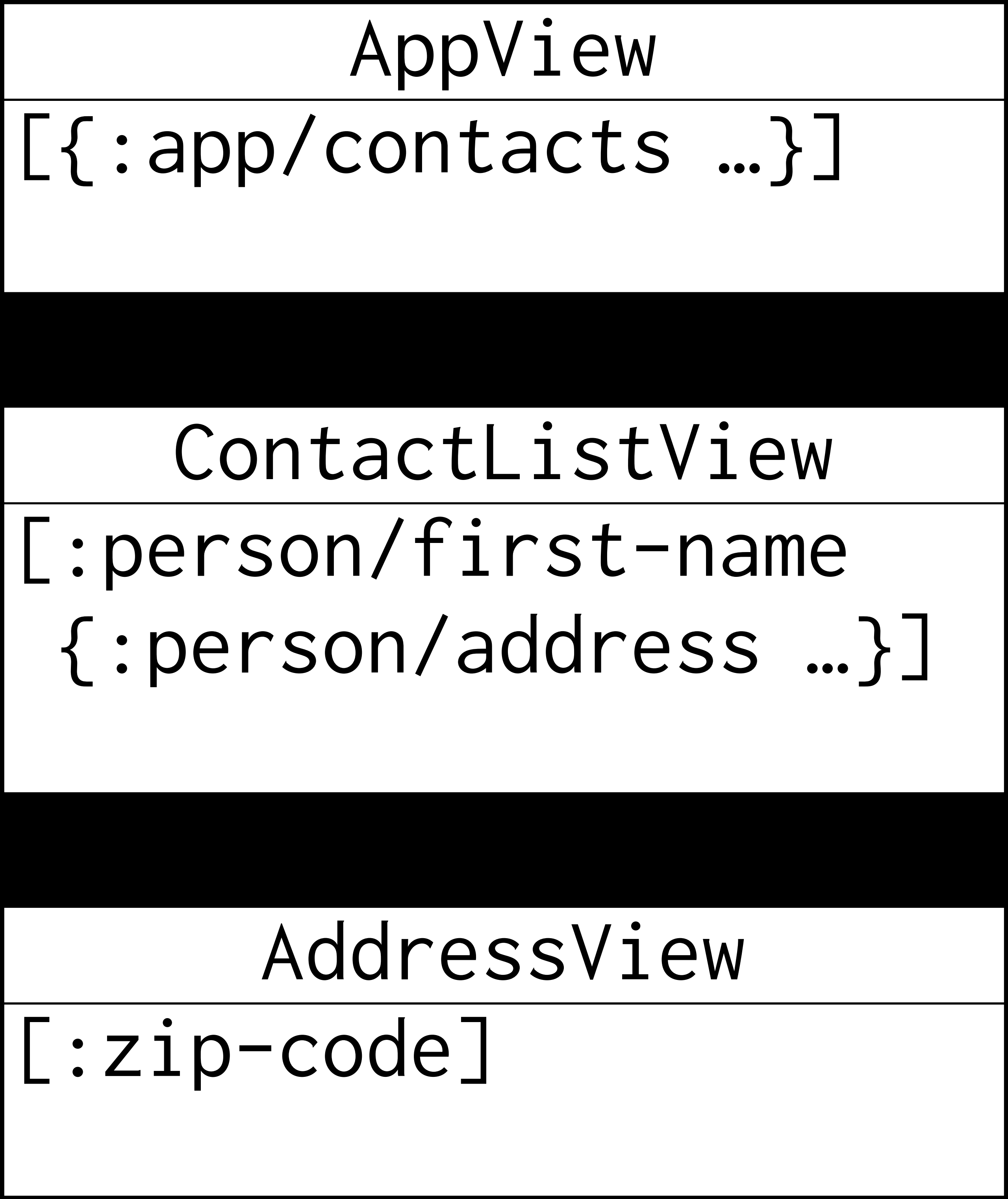
- Client specifies exactly what it needs
- Batched
- Details of demand are easily modified
- Decoupled from service implementation

UX IMPLICATIONS



USER INTERFACES

- User interfaces are trees
- Graphical clients that talk to traditional REST endpoints general involve error prone reshaping code
- Demand Driven Architecture can dramatically simplify rich clients



AppView

```
[{:app/contacts …}]
```

ContactListView

```
[:person/first-name
 {:person/address …}]
```

AddressView

```
[:zip-code]
```

DEMO

```
[{:app/contacts
 [:person/first-name
  {:person/address [:zip-code]}]}]
```

CAVEATS

- Doesn’t mean no backend
- Security
- Routing
- Caching

LEVELING UP

DATOMIC

- Queries work out of the box (pull syntax)
- Caching (peers)
- Evolvable schema (not migrations)
- Query arbitrary points in time (without logs)
- Client can trivially receive change sets (transaction report)

RELAY/GRAPHQL

- FaceBook software layer over React
- Monolithic application architecture
- Relay/GraphQL deliver demand driven queries for React user interfaces

JSONGRAPH/FALCOR

- NetFlix eliminated 90% of their networking code
- Can now serve many different kinds of clients
- Unlike Facebook microservices based design
    - Still, same benefits - phones, tablets, browsers, and set-top boxes can get exactly what they need

RECAP

- The pace of client innovation is only accelerating
- Demand driven architecture guides us toward evolvable systems
- Can remove incidental complexity from client and server by meeting on simple data

REFERENCES

- Relay/GraphQL (FaceBook)
- JSONGraph/Falkor (NetFlix)
- Datomic

QUESTIONS?