Designing Data Intensive Applications

Capítulo 2:
Modelos de Dados e Linguagens de Consulta

Carmem Hara

Modelos de Dados nas Aplicações

- Cada nível esconde as complexidades dos níveis inferiores
- Necessidade de mapeamento entre níveis

Modelos de Dados - Histórico

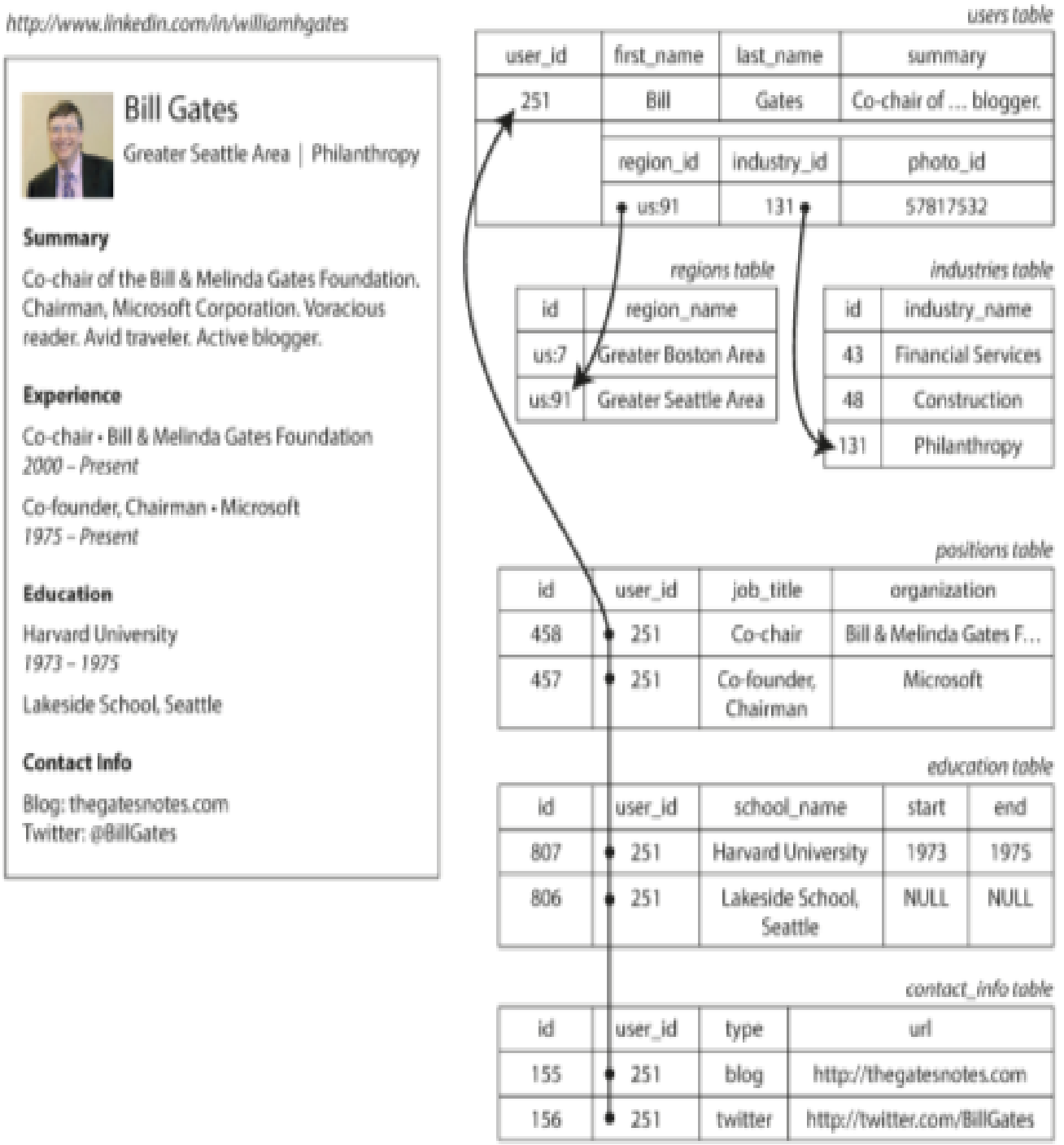
- 1970: surgimento do modelo relacional
- 1985: domínio do modelo relacional no mercado
- Final de 1980-1990: surgimento dos SGBDs orientados a objetos
- 2000: surgimento do XML
- 2010: noSQL
  - Escalabilidade para dar suporte ao Volume
  - Flexibilidade para dar suporte à Variedade
  - Velocidade de processamento

Persistência Poliglota

Incompatibilidade Relacional-Objeto

- Utilização de linguagens de programação orientadas a objeto
- Armazenamento em um SGBD Relacional
  - normalização
- Mapeamento:
  - ferramentas ORM (*Object-relational mapping*)
  - Hibernate, ActiveRecord

Relacionamentos 1:n

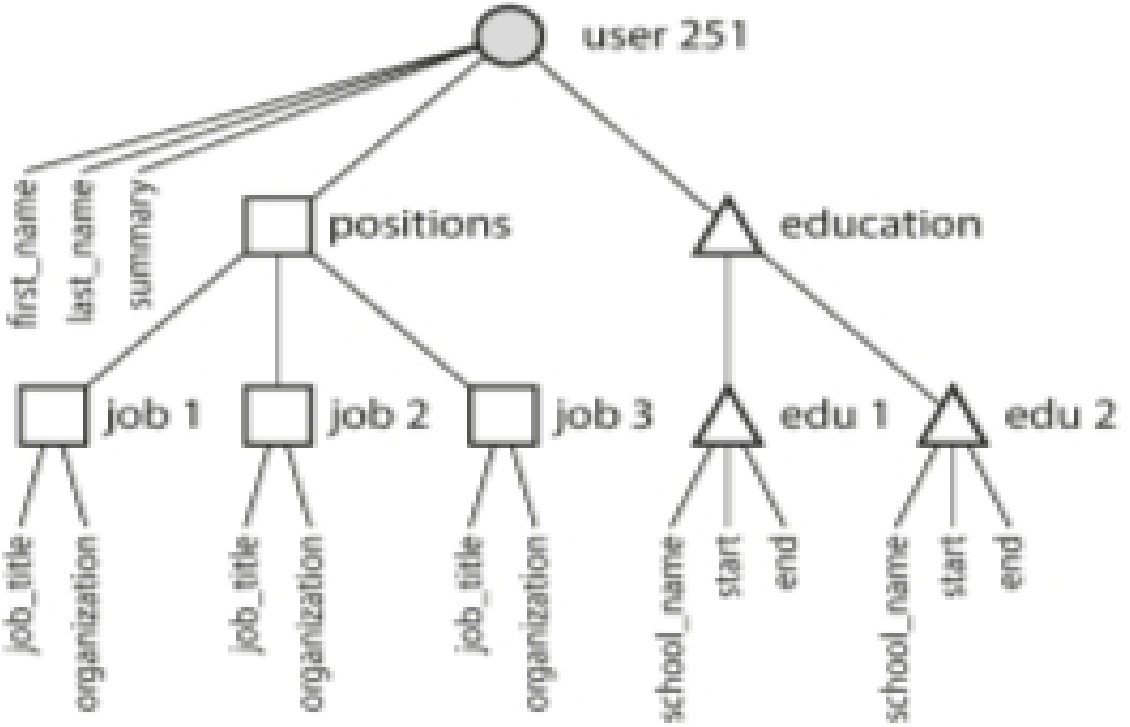


Representação em JSON

```
{“user_id”: 251,
 “first_name”: “Bill”,
 “last_name”: “Gates”,
 “summary”: “Co-chair of the Bill&Melinda...Active Blogger”,
 “region_id”: “us:91”,
 “industry_id”: 131,
 “photo_url”: “/p/7/000/253/05b/308dde.jpg”,
 “positions”: [
  {“job_title”: “Co-chair”,
   “organization”: “Bill&Melinda Gates Foundation”},
   “job_title”: “Co-founder, Chairman”,
   “organization”:“microsoft”}],
 “education”:[
  {“school_name”: “Harvard”,  “start”: 1973, “end: 1975},
  {“school_name”: “Lakeside”, “start”: null, “end”: null}],
 “contact_info”: {
  “blog”: “http://thegatesnotes.com”,
  “twitter”: “http://twitter.com/BillGates”}
}
```

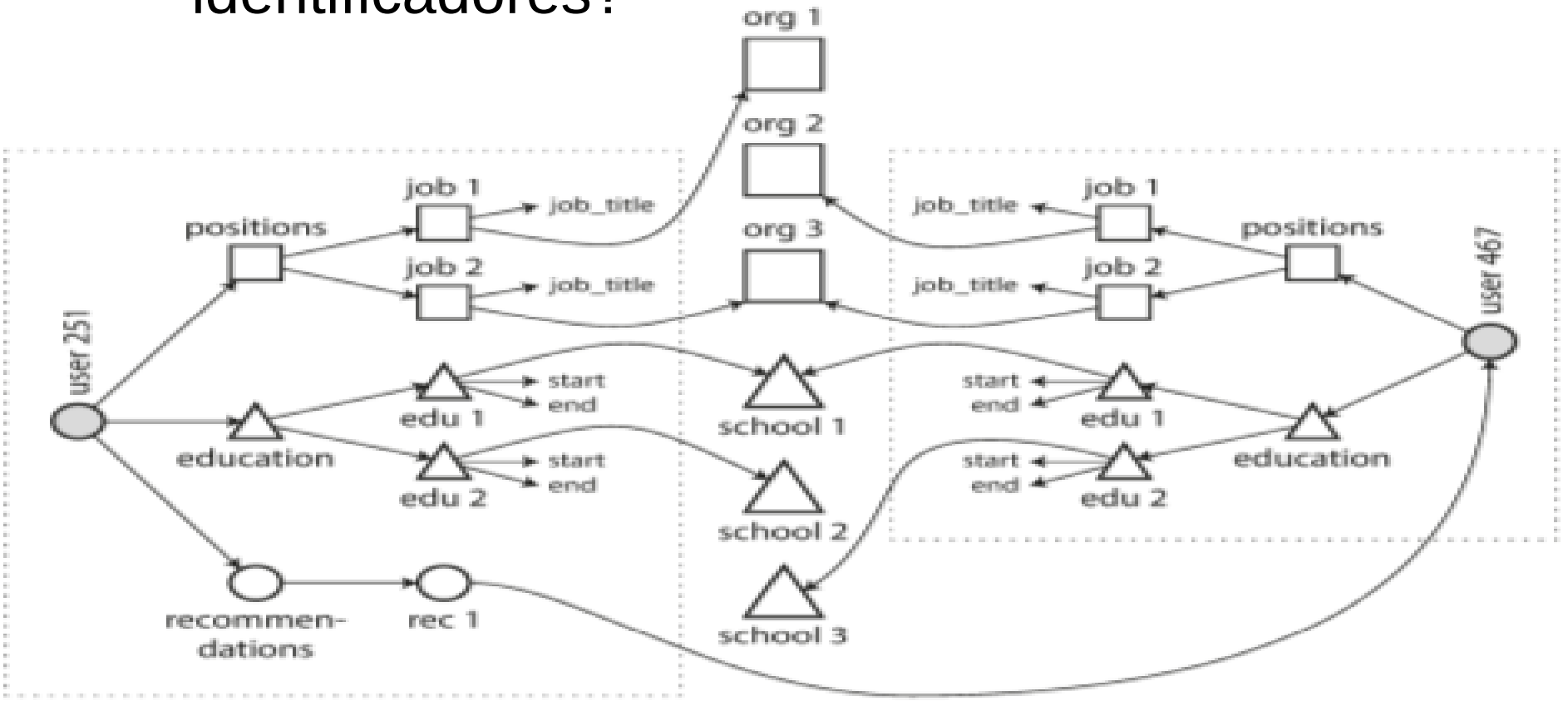
JSON

- Adota um modelo orientado a documento
  - MongoDB, RethinkDB, CouchDB, Espresso
- Representação tem localidade
  - Representação explícita de relacionamentos 1:n



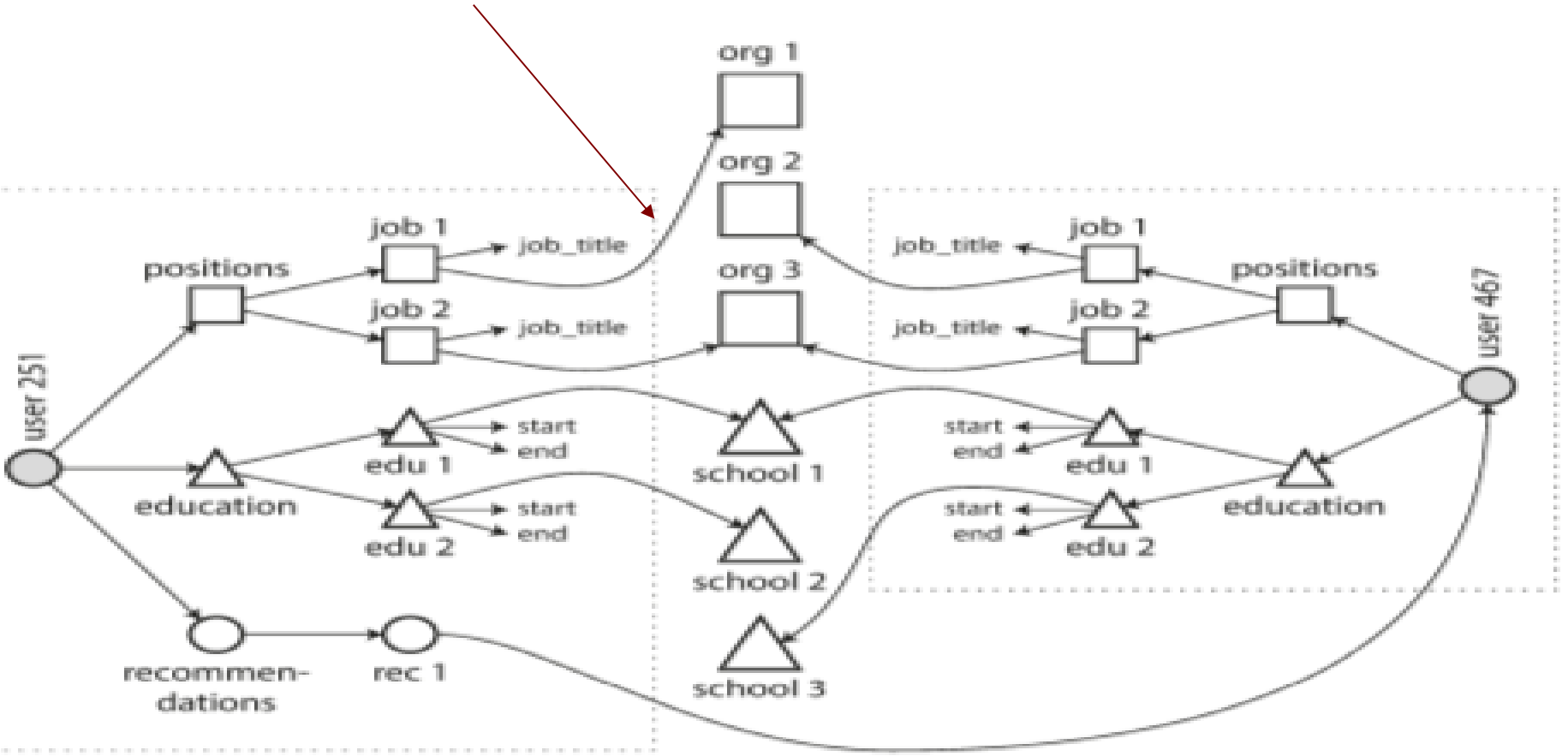
Relacionamentos N:1 e N:M

- Utilização de identificadores evita duplicação
  - `region_id, industry_id`
  - `organization` e `school_name` deveriam ser identificadores?



Relacionamentos N:1 e N:M

- Modelo relacional: chaves estrangeiras
- Modelo de documentos: referência de documento – similar ao modelo de redes



Modelos de Dados

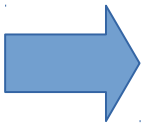
- JSON ~ Modelo Hierárquico
  - Bom para representar relacionamentos 1:n
  - Sem suporte para junções
- Modelo de Redes
  - Registros com múltiplos pais
  - Ligações em forma de apontadores
- Modelo Relacional
  - Ligações baseadas em valores
  - Maior nível de abstração

Comparação

	Relacional	Documento
Localidade de dados baseado em relacionamento 1:N	Não	Sim
Suporte a junções	Sim	Não
Suporte para relacionamentos N:M	Sim	Não
Flexibilidade de Esquema	Verificação na escrita ~ checagem de tipo estática	Verificação na leitura ~ checagem de tipo dinâmica Bom para dados heterogêneos

Localidade de Relacionamentos 1:N

- Existem propostas para extensão do modelo relacional
  - Oracle: *Multiple-table index cluster tables*
- Google Spanner
  - Linhas aninhadas dentro da tabela “pai”
- Big Table
  - Conceito de família de colunas



Modelo híbrido relacional + documento

(similar a o que ocorreu com o modelo XML)

Linguagens de Consulta

Declarativa

- Relacional: SQL

```
select *
from animals
where family=”sharks”
```

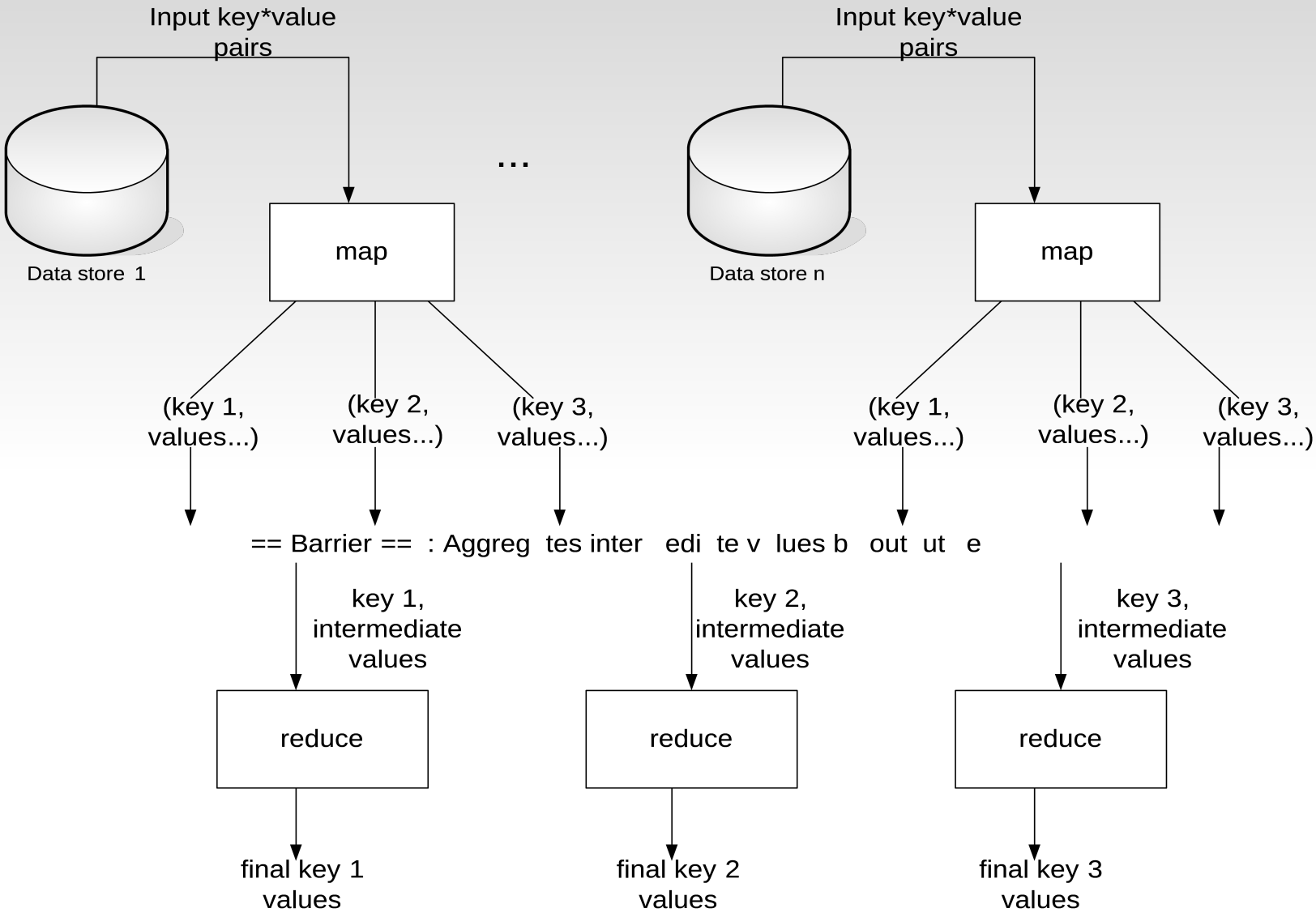
- Mais fácil de
  - Otimizar
  - Paralelizar

Imperativa

- Hierárquico, Rede

```
function getSharks(){
  var sharks = []
  for (i=0; i<animals.length; i++){
    if (animals[i].family == 'sharks'
      sharks.push(animals[i]);
  }
  return sharks;
}
```

Map Reduce


Input key*value pairs
Input key*value pairs
...
Data store 1
Data store n
map
map
(key 1, values...)
(key 2, values...)
(key 3, values...)
(key 1, values...)
(key 2, values...)
(key 3, values...)
== Barrier == : Aggreg tes inter edi te v lues b out ut e
key 1, intermediate values
key 2, intermediate values
key 3, intermediate values
reduce
reduce
reduce
final key 1 values
final key 2 values
final key 3 values

Map Reduce: MongoDB

Número de observações de tubarão por mês

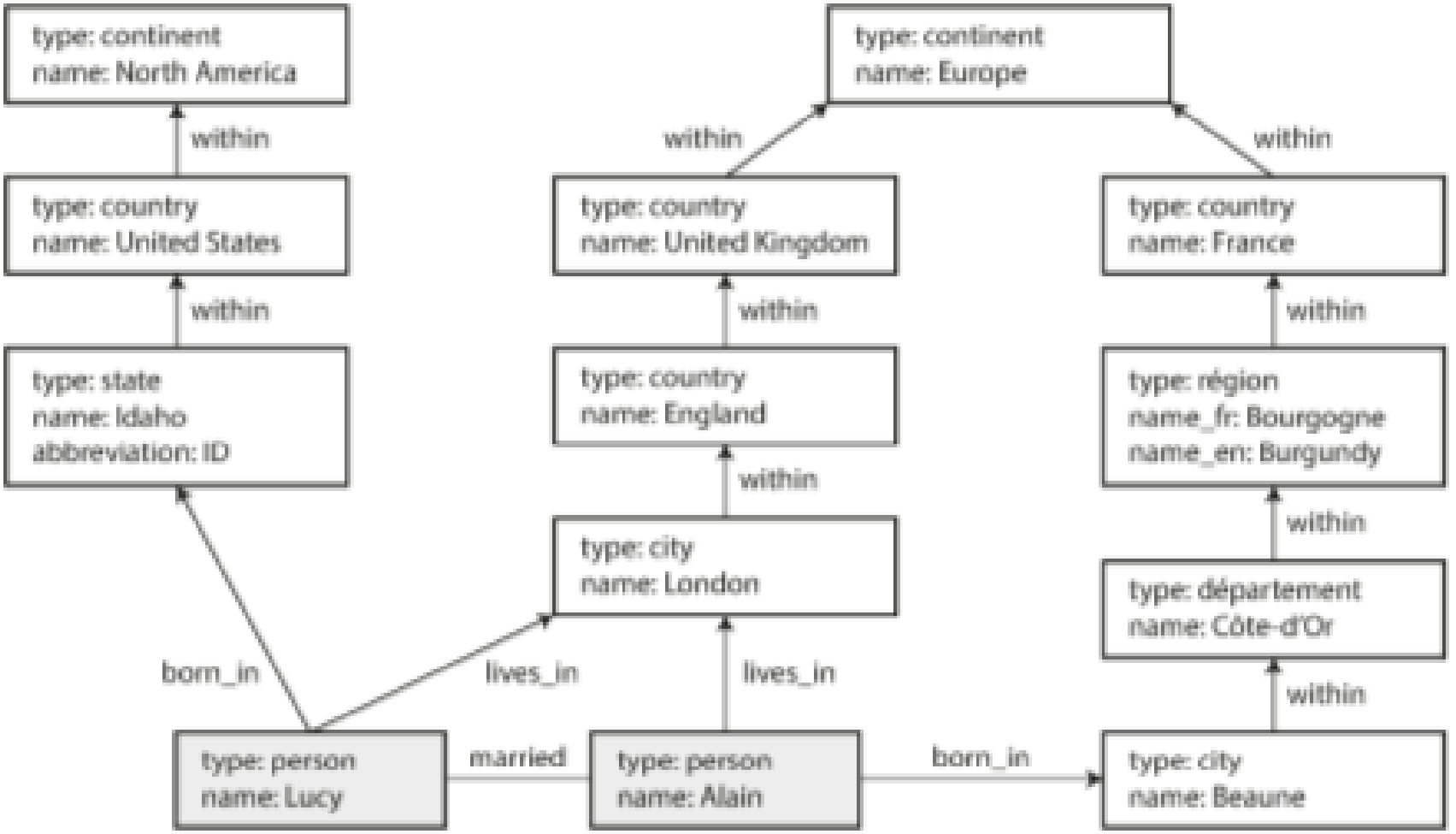
```
db.observations.mapreduce{
  function map(){
    var year = this.observationTS.getFullYear();
    var month = this.observatonTS.getMonth()+1;
    emit( year + “-” + month, this.numAnimals);
  },
  function reduce( key, values ){
    return Array.sum( values );
  },
  {
    query: {family: “Sharks” },
    out: “monthlySharkReport”
  }
}
```

MongoDB: *aggregation pipeline*

Número de observações de tubarão por mês

```
db.observations.aggregate([
   { $match: {family: “Sharks”} },
   { $group: {
      _id: {
            Year:  {$year: “$observationTS” },
            Month: {$month: “$observationTS” }
           },
           TotalAnimals: {$sum: “$numAnimals”}
   } }
])
```

Modelos de Grafos – Grafos de Propriedades



Grafos de Propriedades

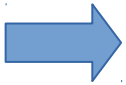
- Cada vértice contém:
  - Identificador único
  - Conjunto de arestas de saída
  - Conjunto de arestas de entrada
  - Conjunto de propriedades (pares chave-valor)
- Cada aresta contém:
  - Identificador único
  - Vértice de origem
  - Vértice de destino
  - Rótulo que identifica o tipo de relacionamento
  - Conjunto de propriedades (pares chave-valor)

Armazenamento em um SGBDR

```
CREATE TABLE vertice (
  id_vertice integer PRIMARY KEY,
  propriedades json
);

CREATE TABLE arestas (
  id_aresta integer PRIMARY KEY,
  origem integer REFERENCES vertice,
  destino integer REFERENCES vertice,
  rotulo text,
  proriedades json
);

CREATE INDEX vertices_destino ON arestas (destino);
CREATE INDEX vertices_origem  ON arestas (origem);
```



Problema: consultas recursivas para percurso no grafo

Linguagem Cypher do Neo4j

Base:

```
CREATE
(Namerica: Location {name: 'North America', type: 'continent'}),
(USA: Location      {name: 'Unites States', type: 'country'}),
(Idaho: Location    {name: 'Idaho',         type: 'state'}),
(Lucy: Person       {name: 'Lucy'}),
(Idaho -[:WITHIN]-> (USA) -[:WITHIN]-> (Namerica),
(Lucy  -[:BORN_IN]-> (Idaho)
```

Consulta:

```
MATH
(person)-[:BORN_IN]-> ()
      -[:WITHIN*0..]-> (Namerica:Location {name: 'North America')
RETURN person.name
```

RDF - SPARQL

- RDF: modelo baseado em triplas
- SPARQL

```
SELECT ?personName WHERE {
 ?person :name ?personName .
 ?person
    :bornIn / :within* / :name “North   America” .
}
```

Datalog

Base: conjunto de fatos

```
name(namerica, 'North America')
Type(namerica, continent)

name(usa, 'United States')
type(usa, country)
within(usa, namerica)

name(idaho, 'Idaho')
type(idaho, state)
Within(idaho, usa)

name(lucy, 'Lucy')
born_in(lucy, idaho)
```

Datalog - consulta

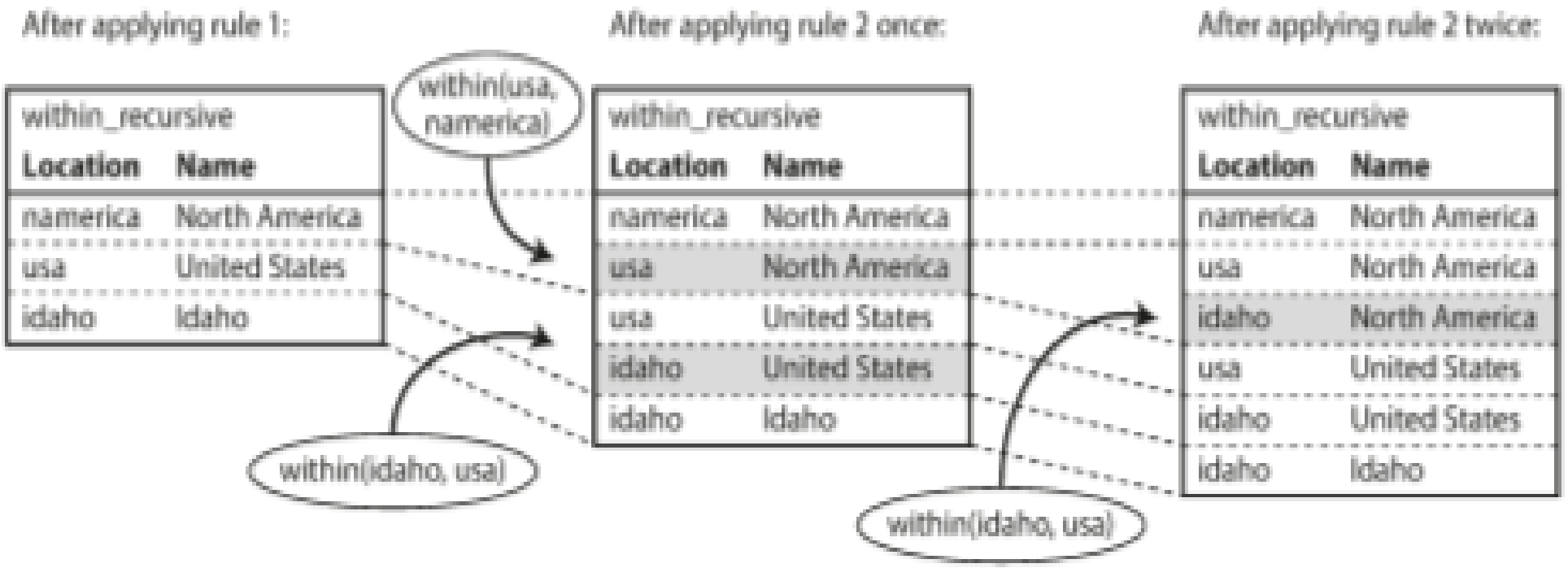
Regras:

```
within_recursive(X,Y):- name(X,Y)
within_recursive(X,Y):- within(X,Z),
                        within_recursive(Z,Y)
Born_places(Name,Place) :-
                name(Person,Name),
                born_in(Person, L),
                within_recursive(L, Place)
```

Consulta:

```
?- Born_places( Name, 'North America' )
```

Avaliação do predicado `within_recursive`



Conclusão

- Modelo relacional
  - representação de relacionamentos N:M baseado em valores
  - Maior nível de abstração
- Modelos noSQL
  - Documentos:
    - Contém dados auto-contidos
    - Relacionamentos entre documentos são raro
  - Grafos
    - Representa relacionamentos arbitrários entre dados e documentos

Perguntas

1) Considere um sistema de compras (ou leilões online). Apresente possíveis falhas de *confiabilidade, escalabilidade e de manutenção* que podem ocorrer e uma possível solução para cada uma delas.
2) Considere os modelos relacional, de documento e de grafos. Para cada um, apresente uma aplicação para a qual o modelo é apropriado para o armazenamento dos seus dados. Justifique por que o modelo é apropriado.