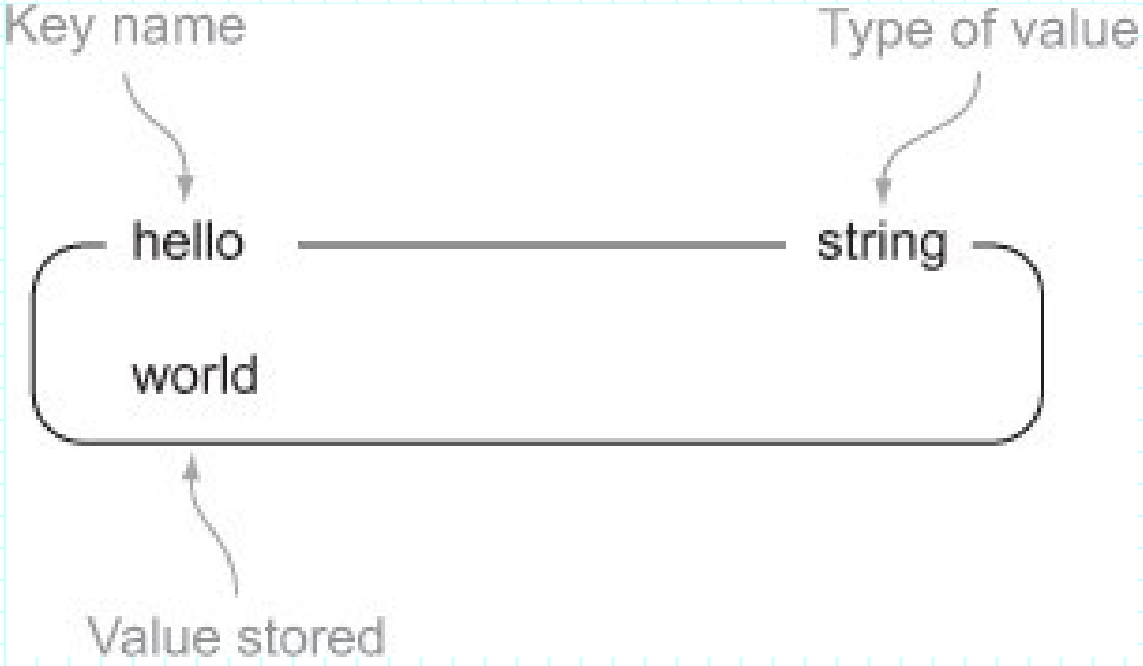
Database Management Systems

- Key-Value Stores
- Graph Databases

Doug Shook

Redis

- Key-Value pairs
  - Limited data types
- In-memory database
  - What about persistence?



- "Redis in Action", Josiah Carlson, 2013

Data Types

- String
- List
- Set
- Hash
- Zset

Strings

set hello world
get hello
del hello
get hello

Lists

```
rpush list-key item
rpush list-key item2
rpush list-key item

lrange list-key 0 -1

lindex list-key 1

lpop list-key

lrange list-key 0 -1
```

Sets

```
sadd set-key item
sadd set-key item2
sadd set-key item3
sadd set-key item

smembers set-key

sismember set-key item4
sismember set-key item

srem set-key item2
srem set-key item2

smembers set-key
```

Hashes

```
hset hash-key sub-key1 value1
hset hash-key sub-key2 value2
hset hash-key sub-key1 value1

hgetall hash-key1

hdel hash-key sub-key2
hdel hash-key sub-key2
hget hash-key sub-key1

hgetall hash-key
```

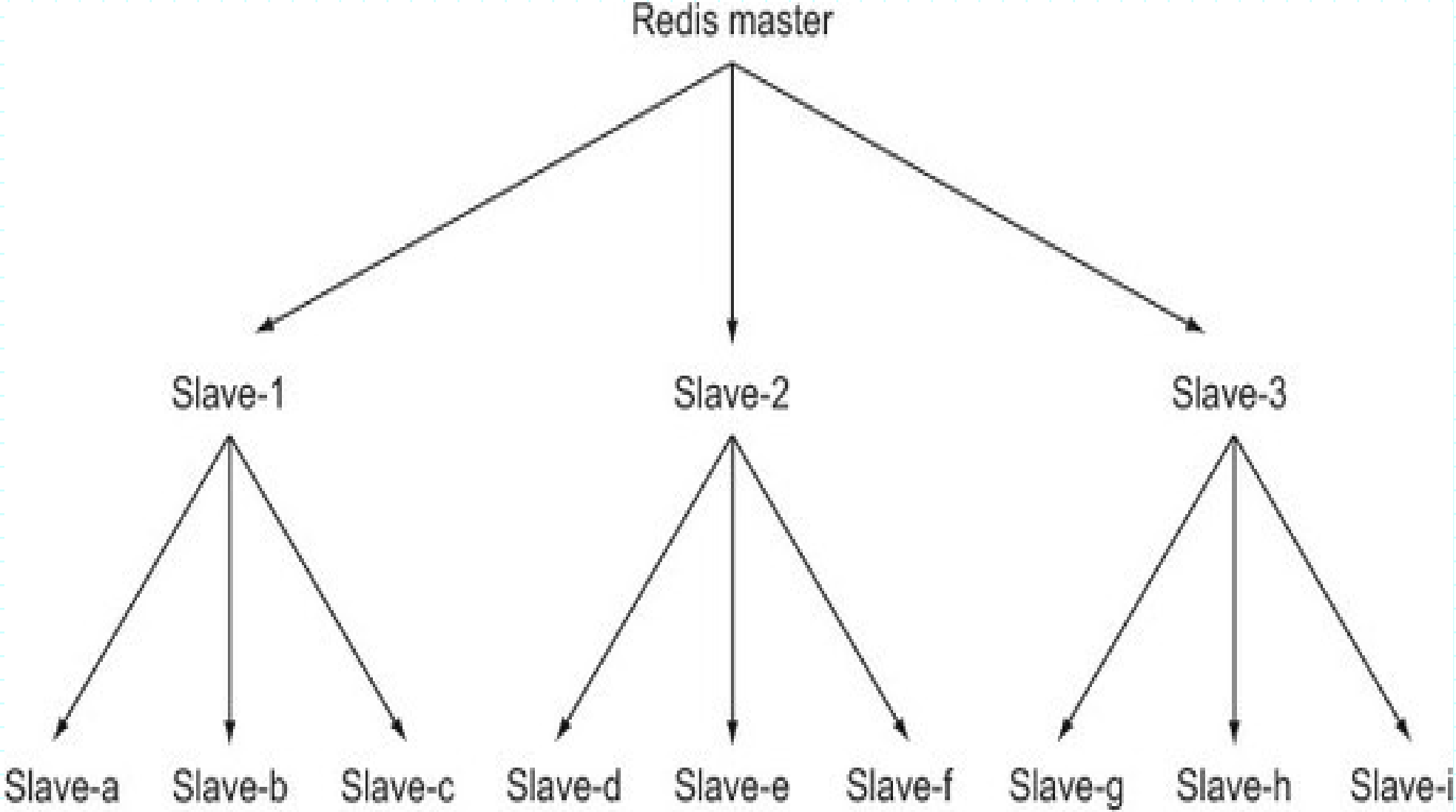
Zsets

```
zadd zset-key 728 member1
zadd zset-key 982 member0
zadd zset-key 982 member0

zrange zset-key 0 -1 withscores

zrangebyscore zset-key 0 800 withscores

zrem zset-key member1
zrem zset-key member1

zrange zset-key 0 -1 withscores
```

Persistence

- Snapshots
- Append only-log files

Replication



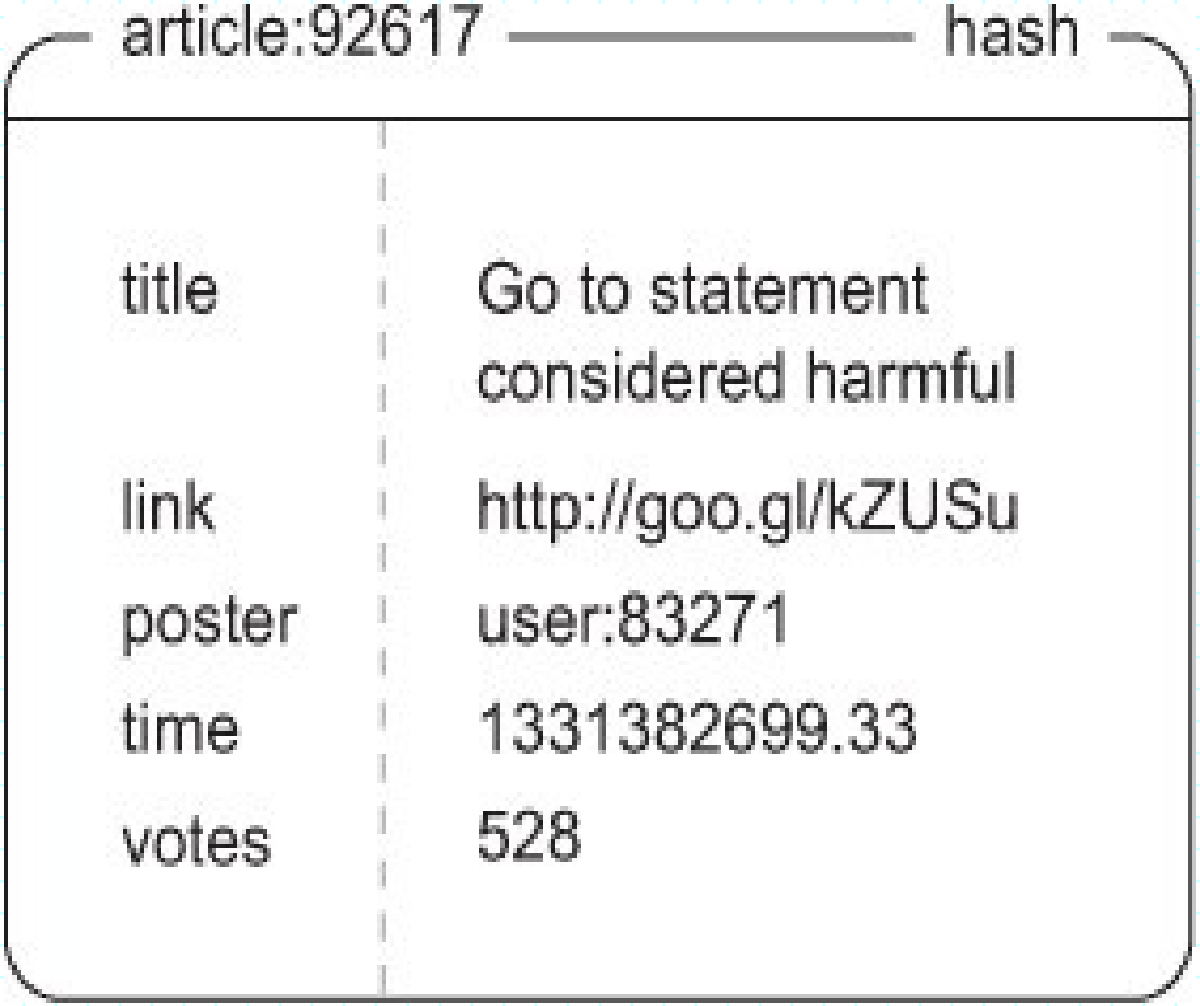
Suitable Use Cases

- Storing Session Information
- User Profiles
- Shopping Cart Data

Example Use Case

Voting occurs after you have clicked through to read a question and any existing answers.

Example Use Case



Example Use Case

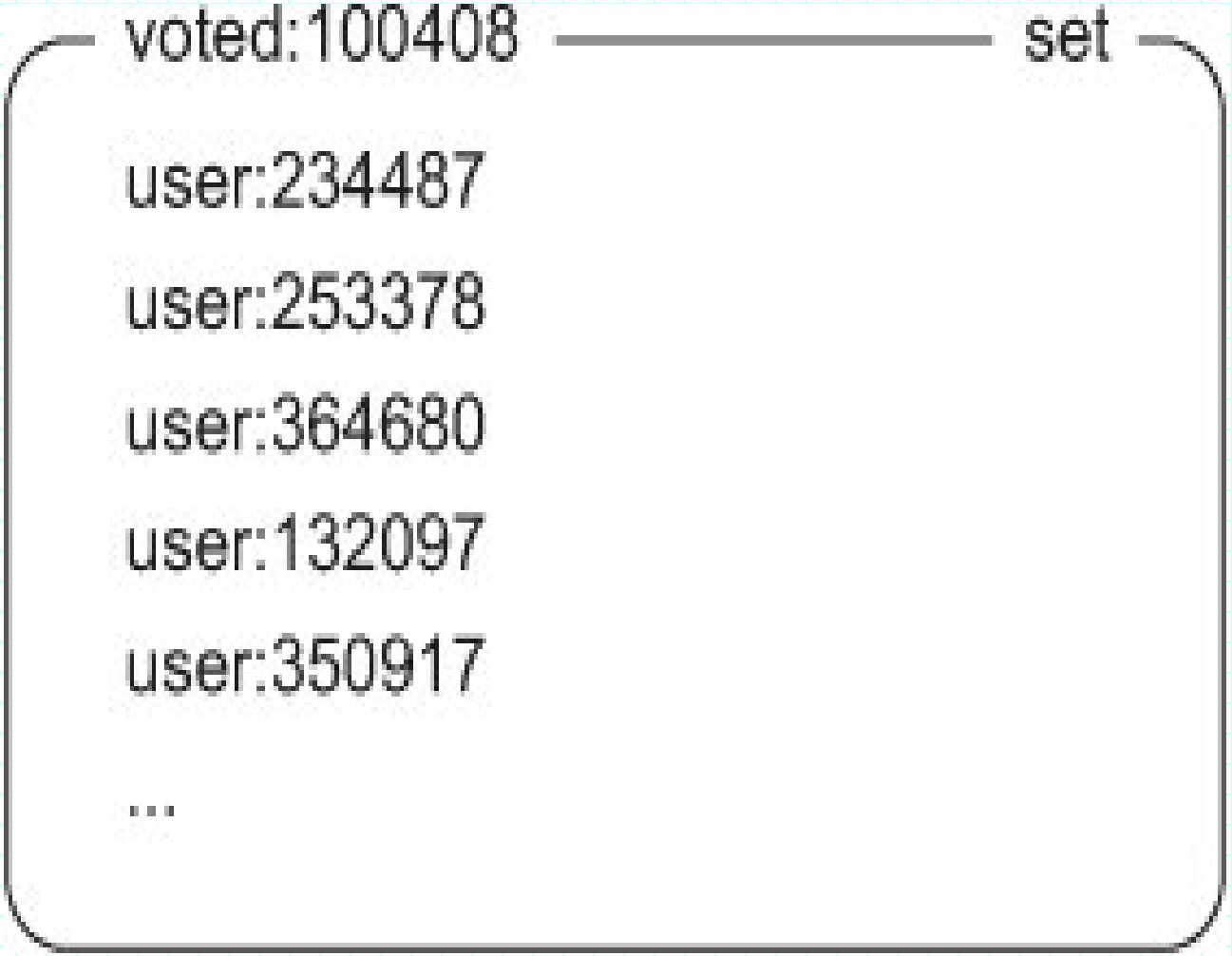
time:	zset
article:100408	1332065417.47
article:100635	1332075503.49
article:100716	1332082035.26

A time-ordered ZSET of articles

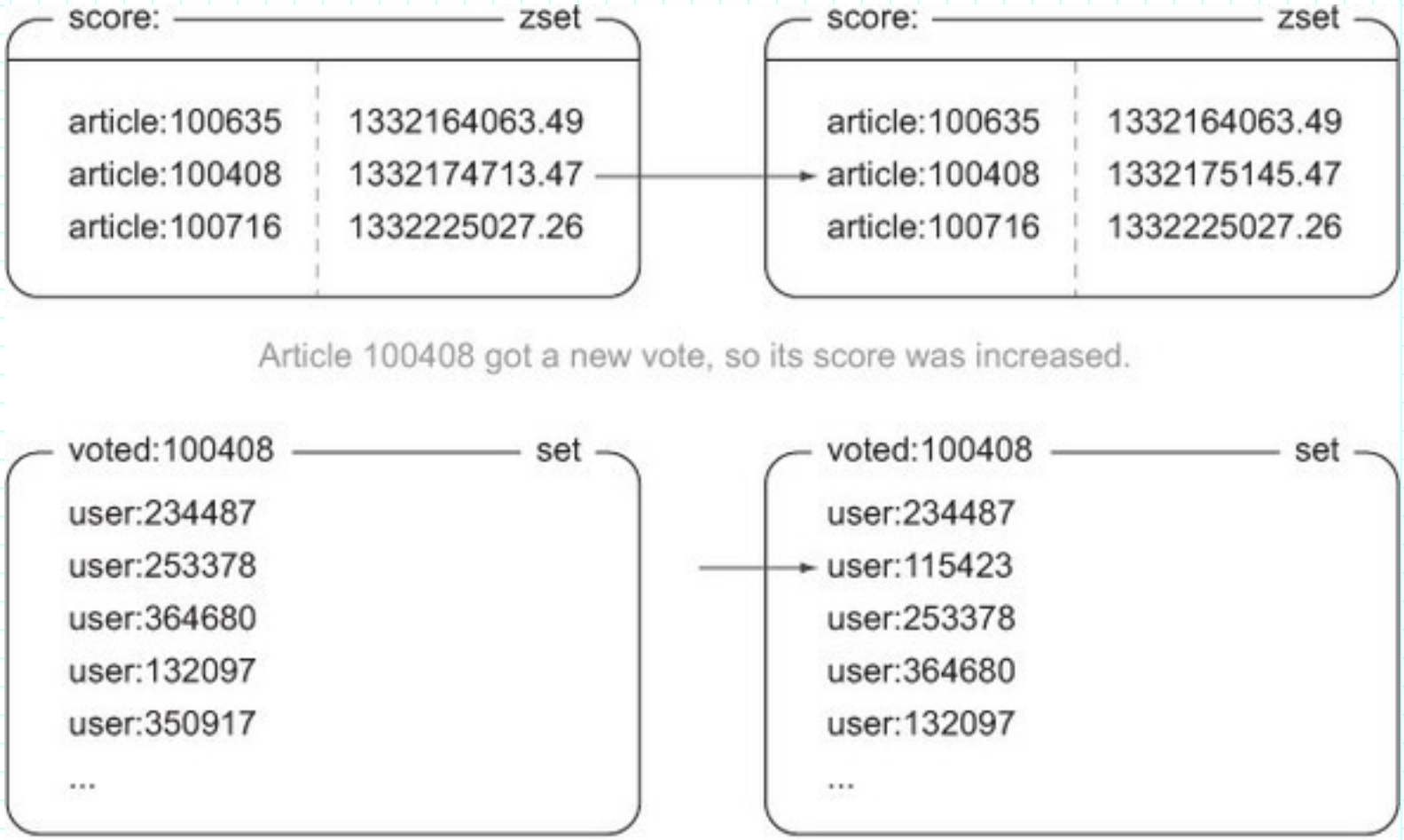
score:	zset
article:100635	1332164063.49
article:100408	1332174713.47
article:100716	1332225027.26

A score-ordered ZSET of articles

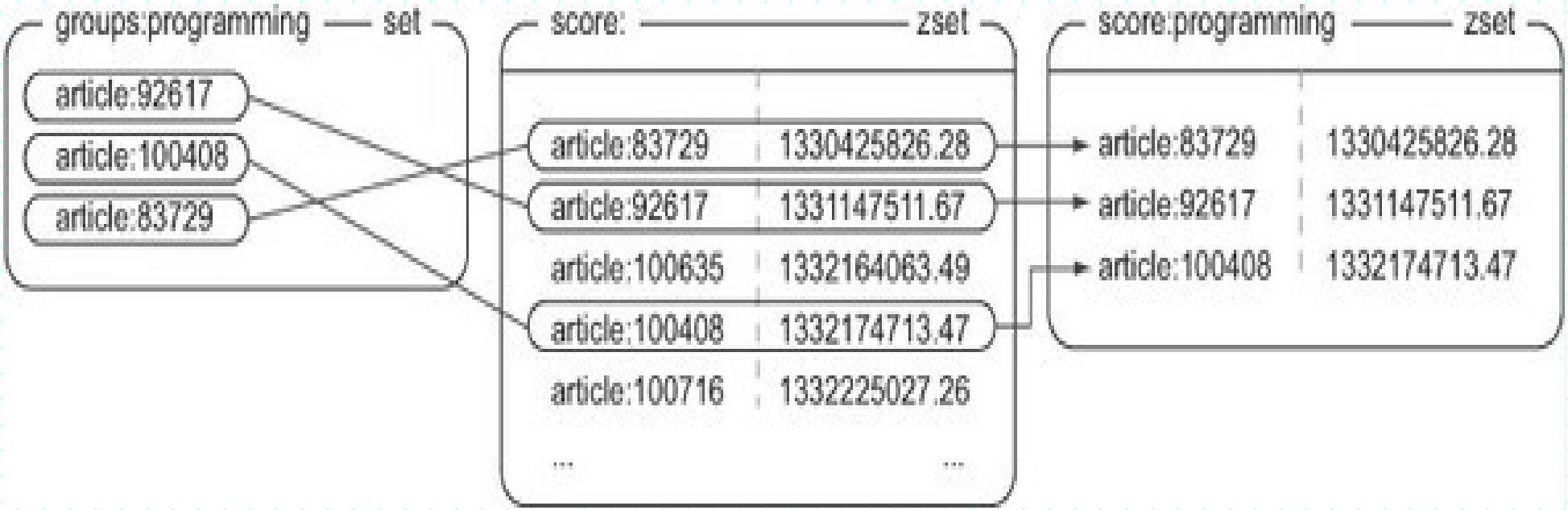
Example Use Case



Example Use Case



Example Use Case



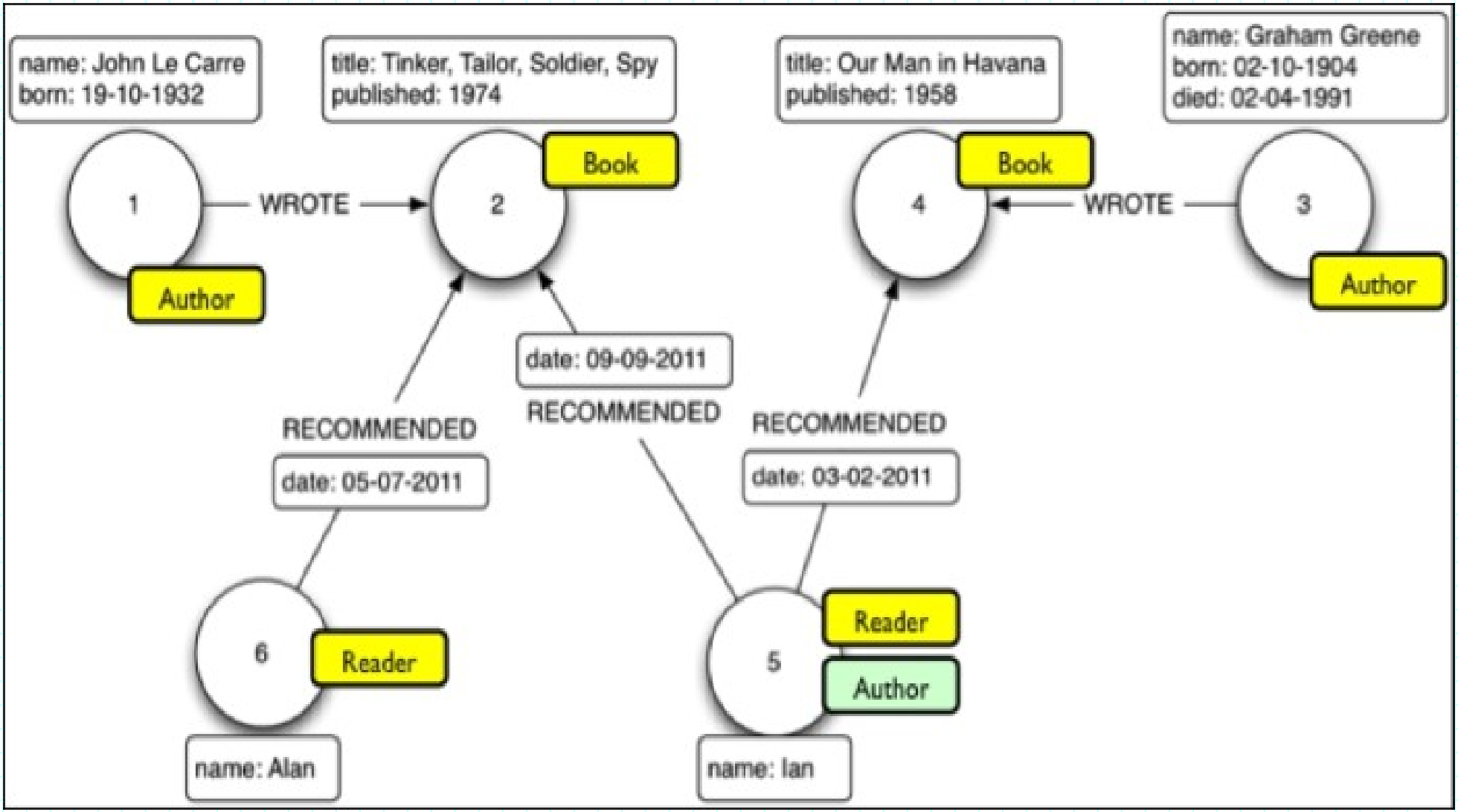
When Not to Use

- Relationships between data sets
- Multioperation transactions
- Query by data
- Operation by sets

When Not to Use

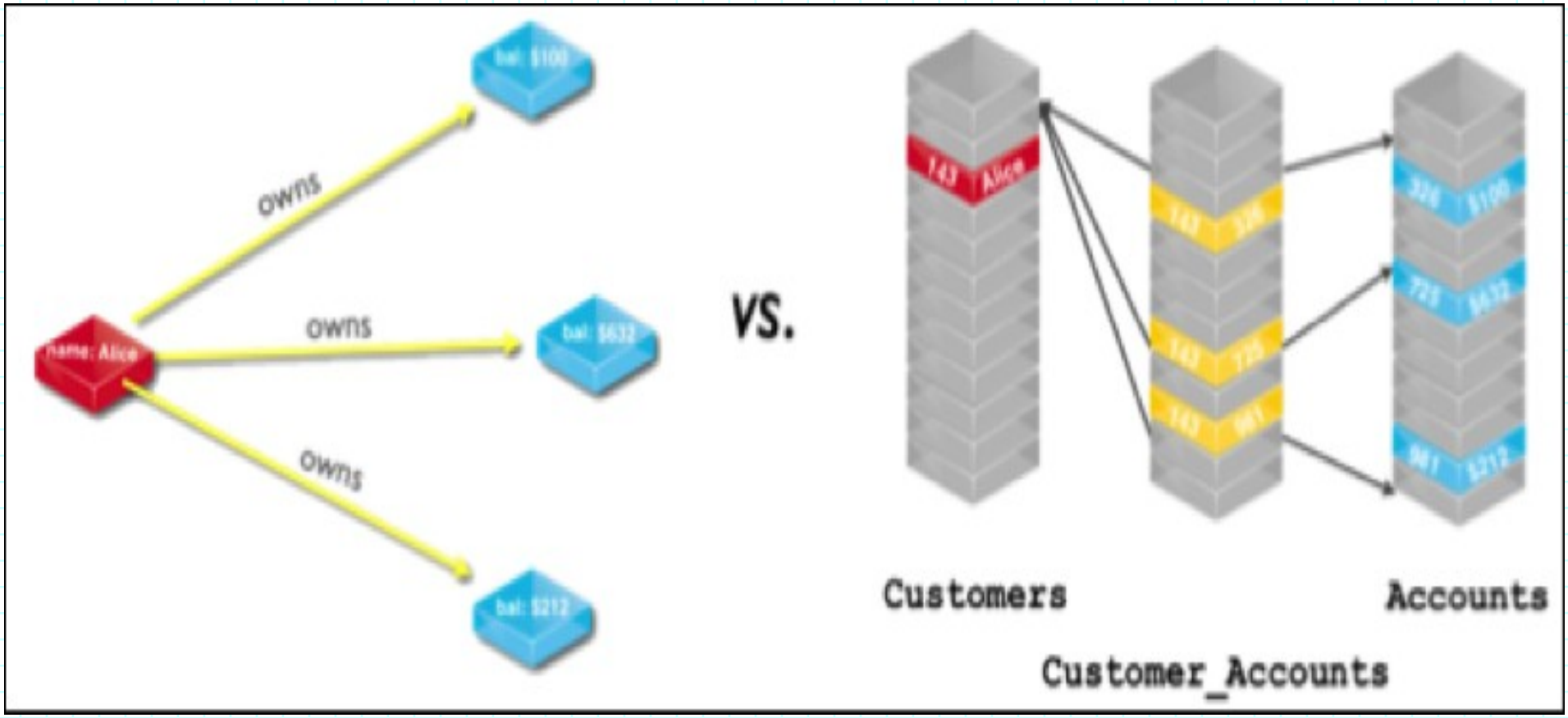
- Relationships between data sets
- Multioperation transactions
- Query by data
- Operation by sets

Neo4J Data Model – Data Constructs



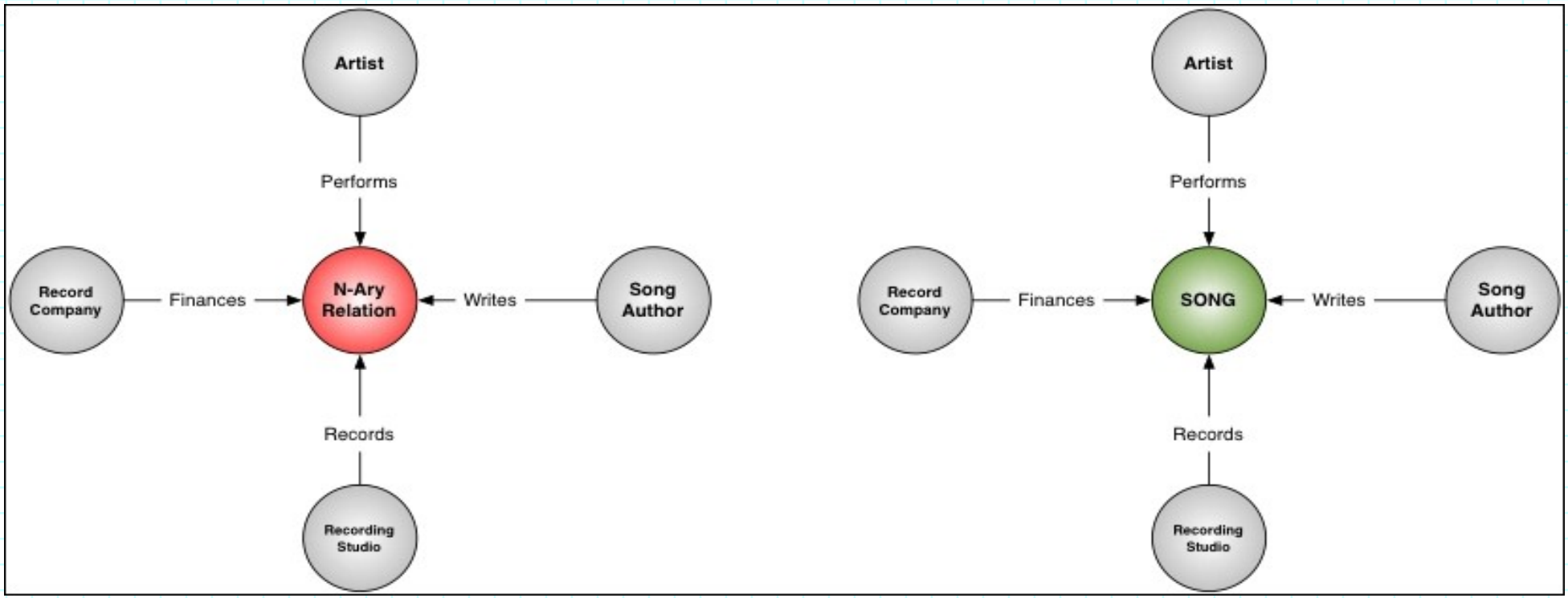
"Learning Neo4J", Rik Van Bruggen, 2014

Neo4J Data Model



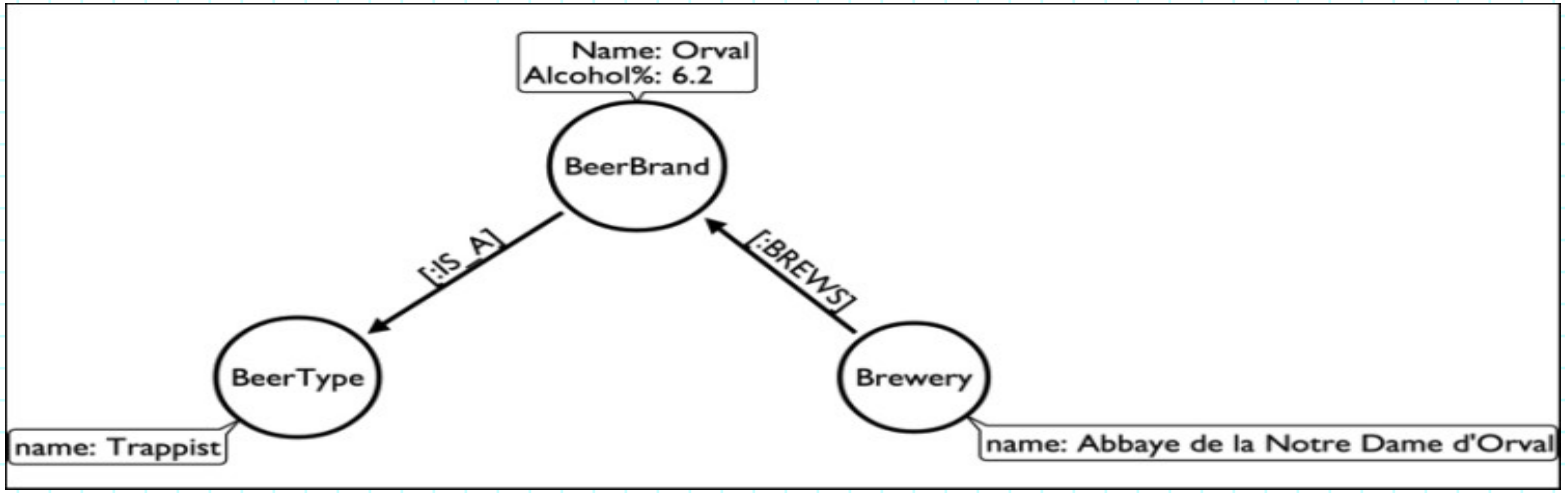
Neo4J Data Model – Best Practices

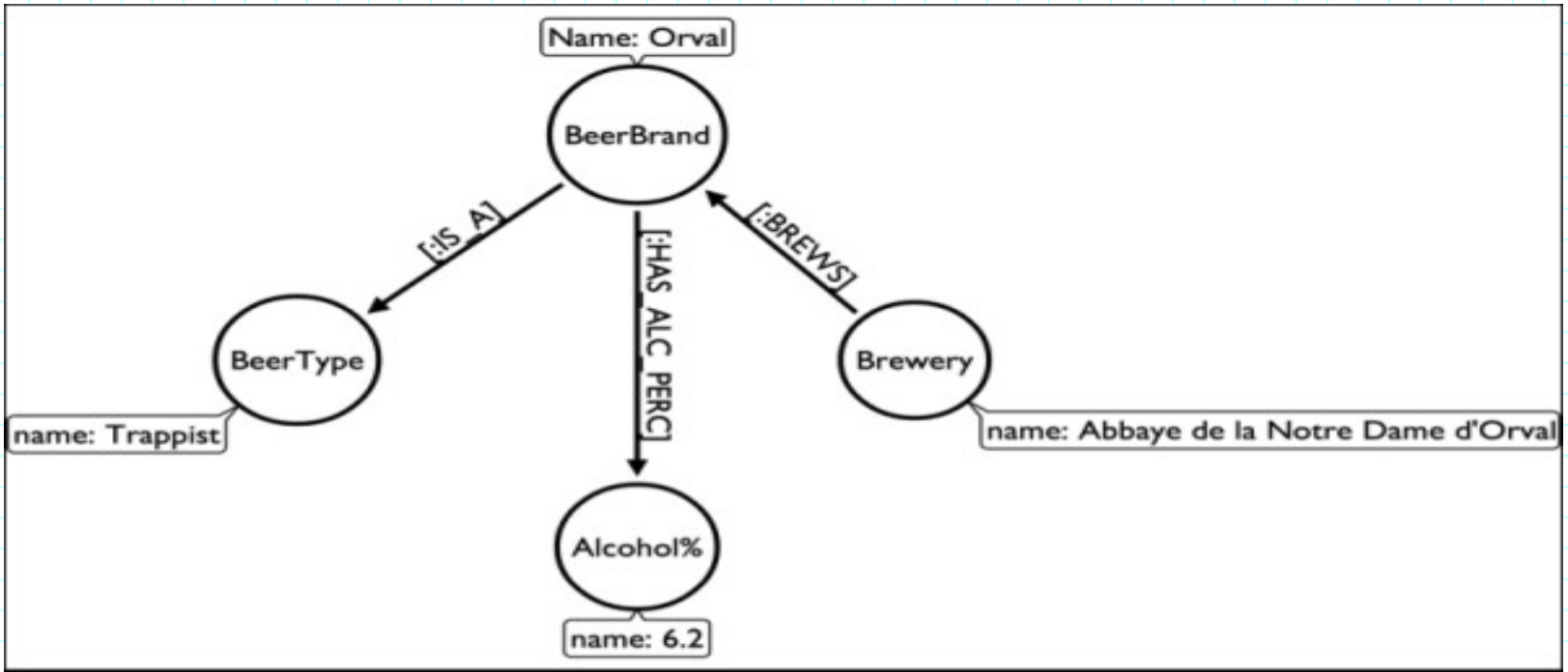
- Design for query-ability
- Align relationships with use cases
- Look for n-ary relationships



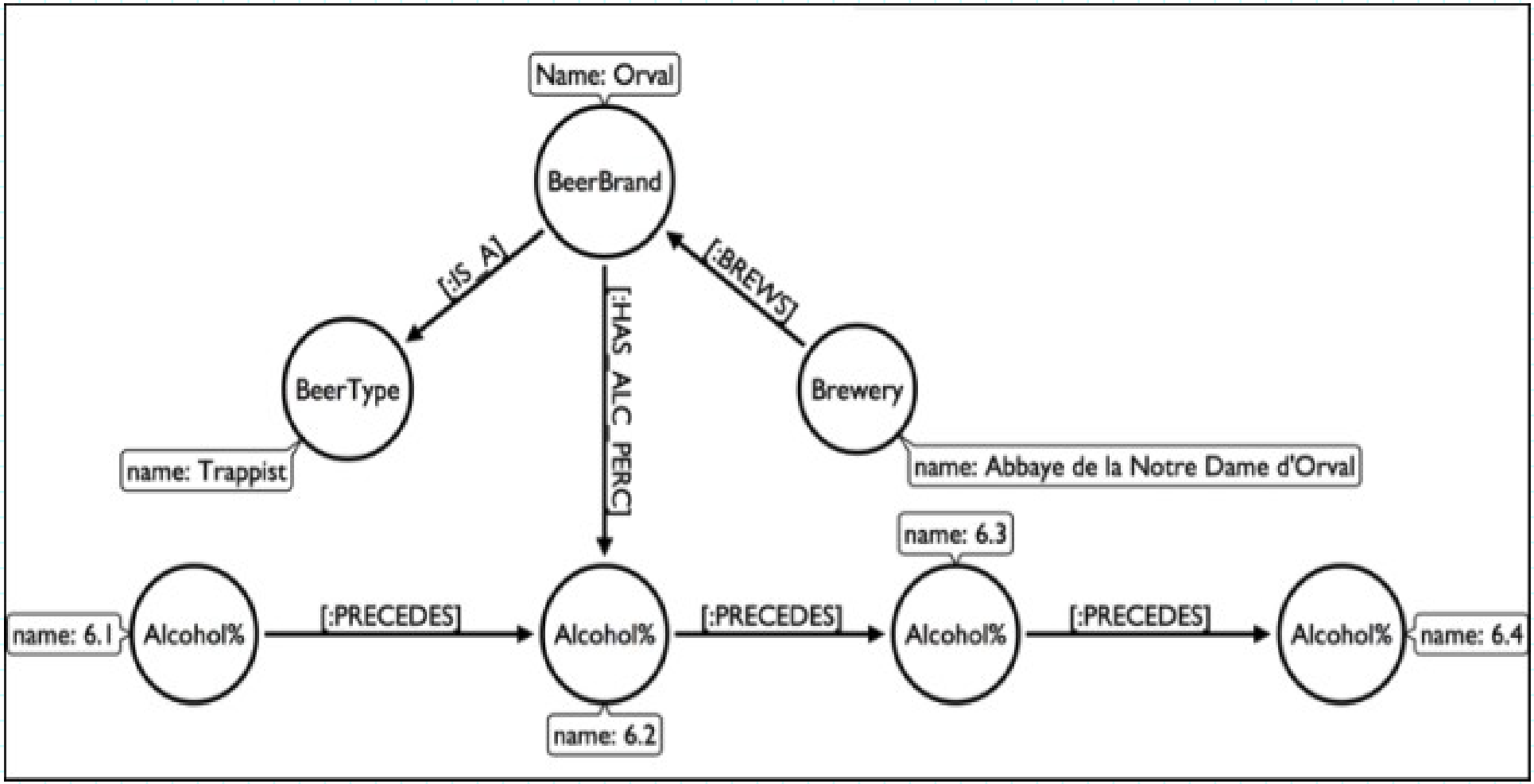
Neo4J Data Model – Best Practices

- Granulate Nodes

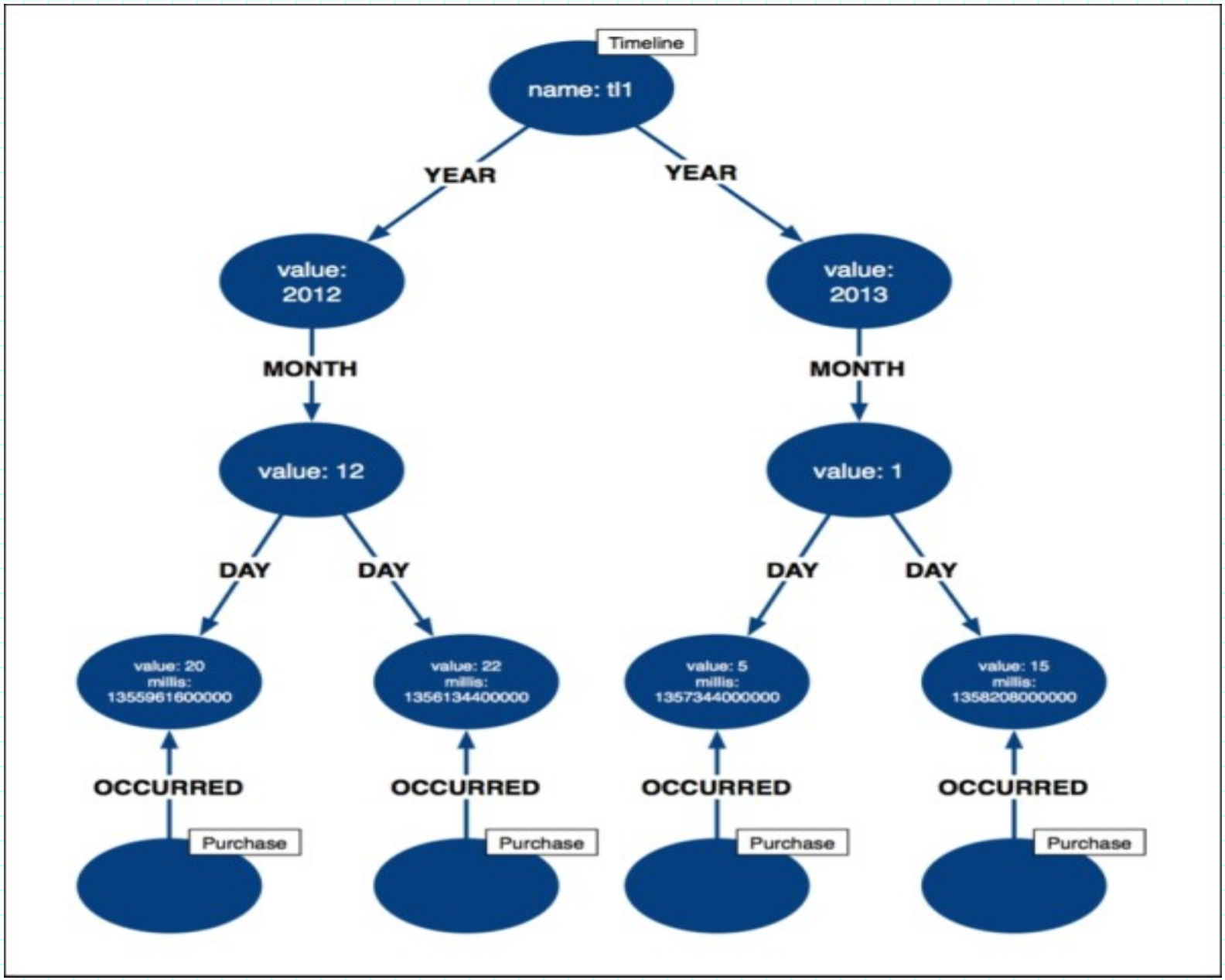




Neo4J Data Model – Indexes



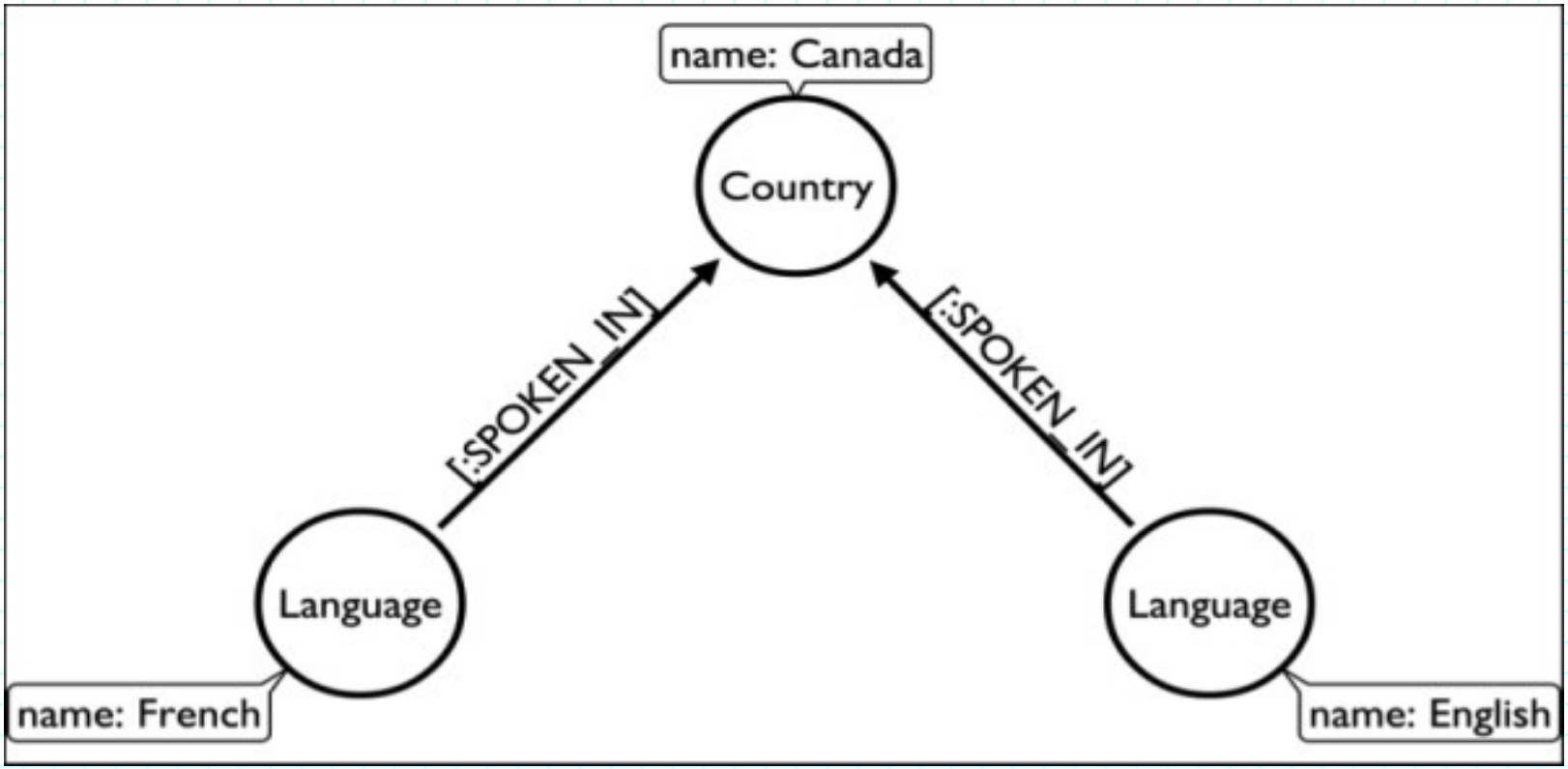
Neo4J Data Model – Indexes



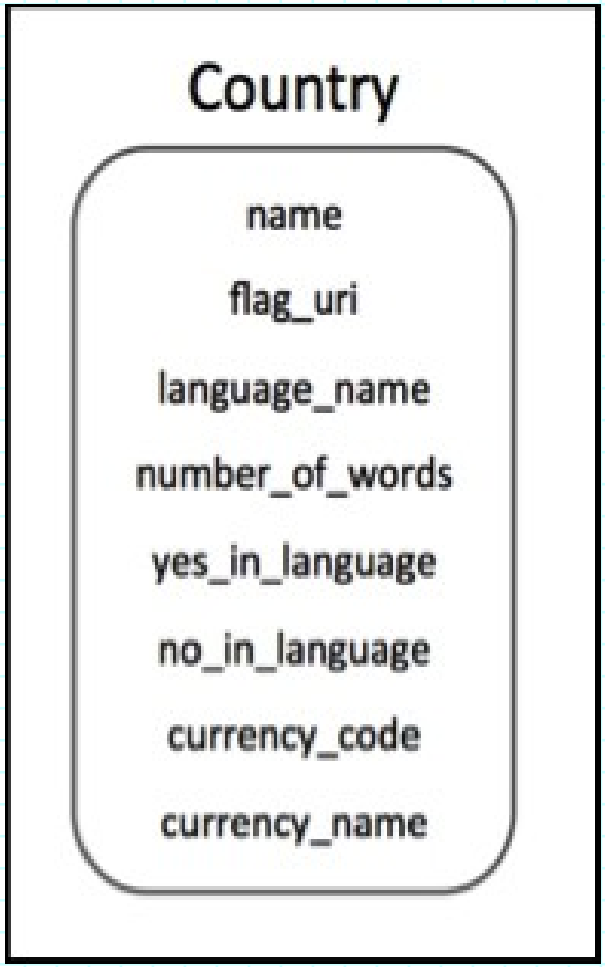
Modelling Pitfalls: Rich Properties

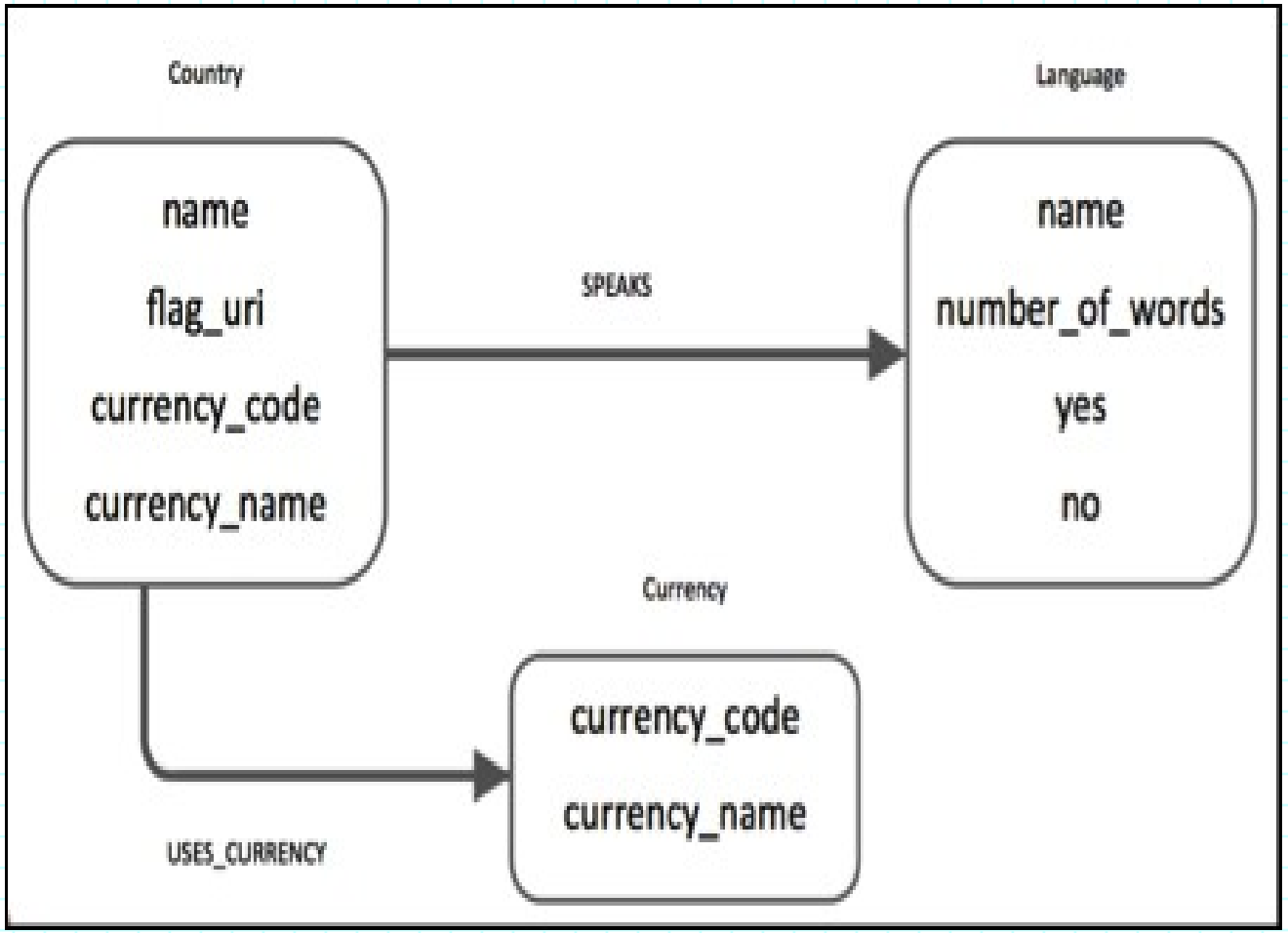
name: "Canada"

languages_spoken:
"['English', 'French']"

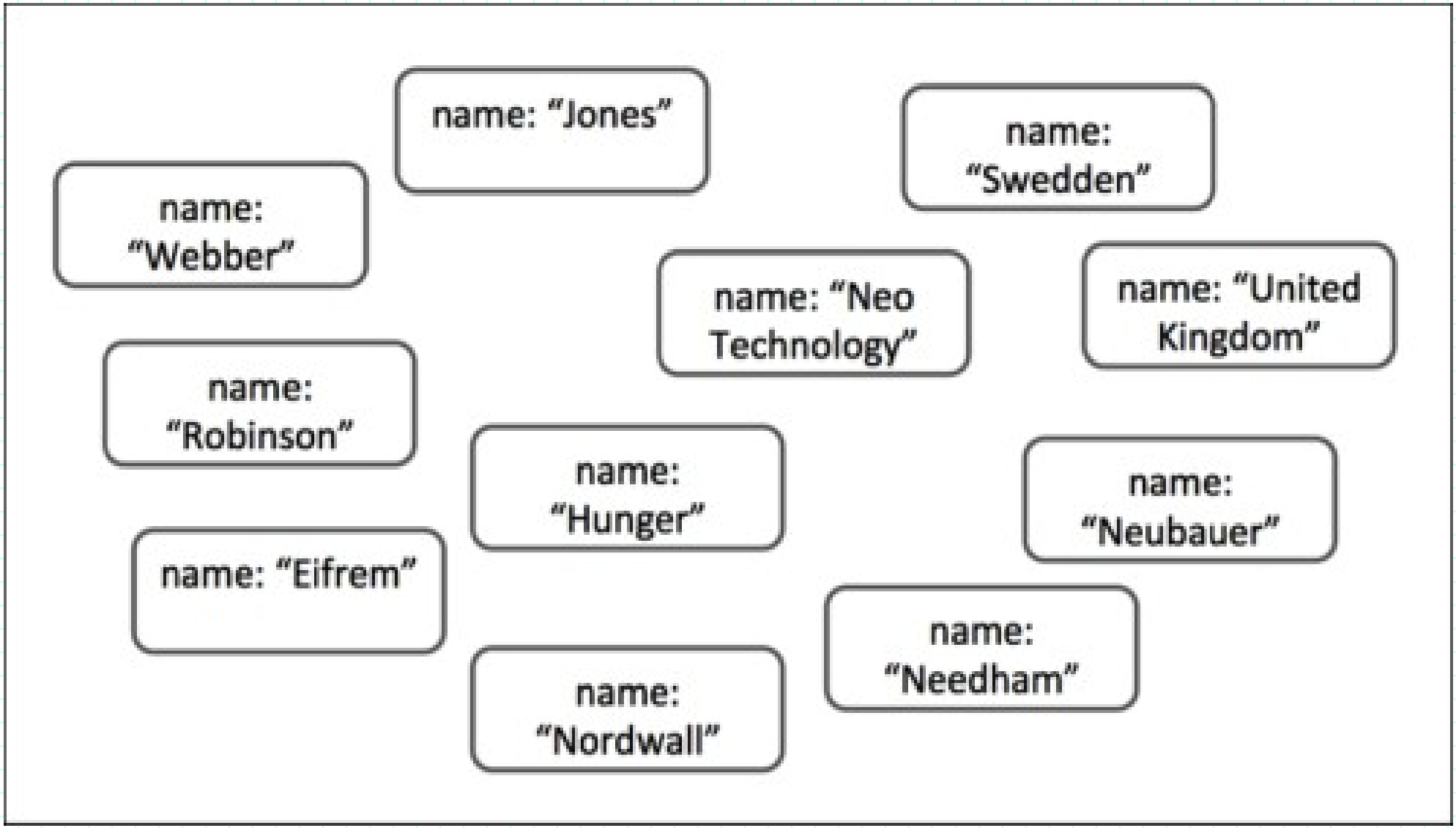


Modelling Pitfalls: Multiple Concepts

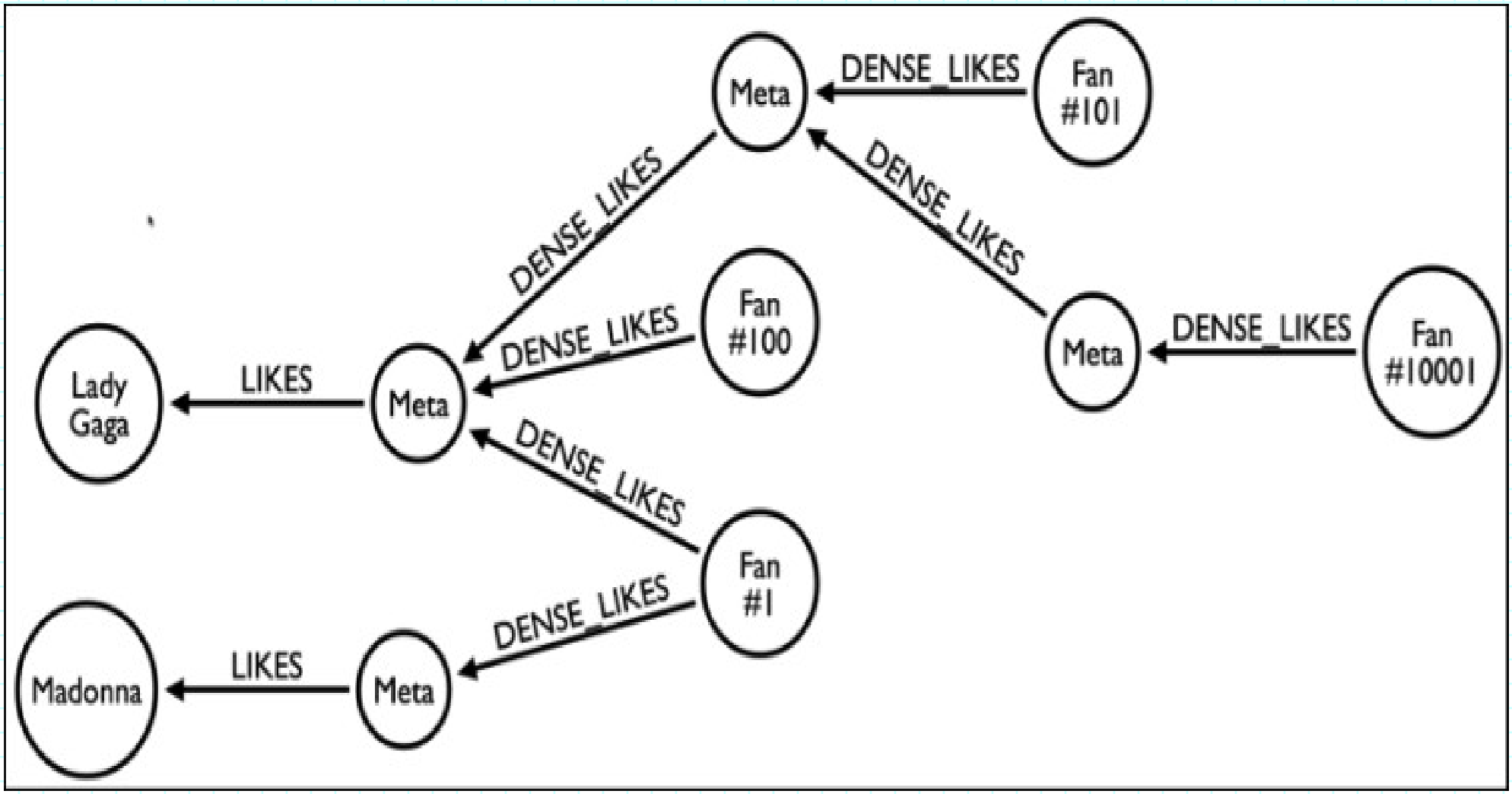




Modelling Pitfalls: Unconnected Graphs



Modelling Pitfalls: Dense Nodes



Cypher

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (LillyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)
```

Cypher

```
MATCH (tom {name: "Tom Hanks"}) RETURN tom

MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas

MATCH (people:Person) RETURN people.name LIMIT 10

MATCH (nineties:Movie) WHERE nineties.released >= 1990 AND nineties.released < 2000 RETURN nineties.title
```

Cypher

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies)
RETURN tom,tomHanksMovies

MATCH (cloudAtlas {title: "Cloud Atlas"})<-[:DIRECTED]-(directors) RETURN
directors.name

MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-
(coActors) RETURN coActors.name

MATCH (bacon:Person {name:"Kevin Bacon"})-[*1..4]-(hollywood)
RETURN DISTINCT hollywood

MATCH p=shortestPath(
  (bacon:Person {name:"Kevin Bacon"})-[*]-(meg:Person {name:"Meg Ryan"})
)
RETURN p
```

Other Features

- Consistency
- Transactions
- Availability
- Scaling

Use Cases

- Connected Data
- Routing, Dispatch, Location-based services
- Recommendation Engines