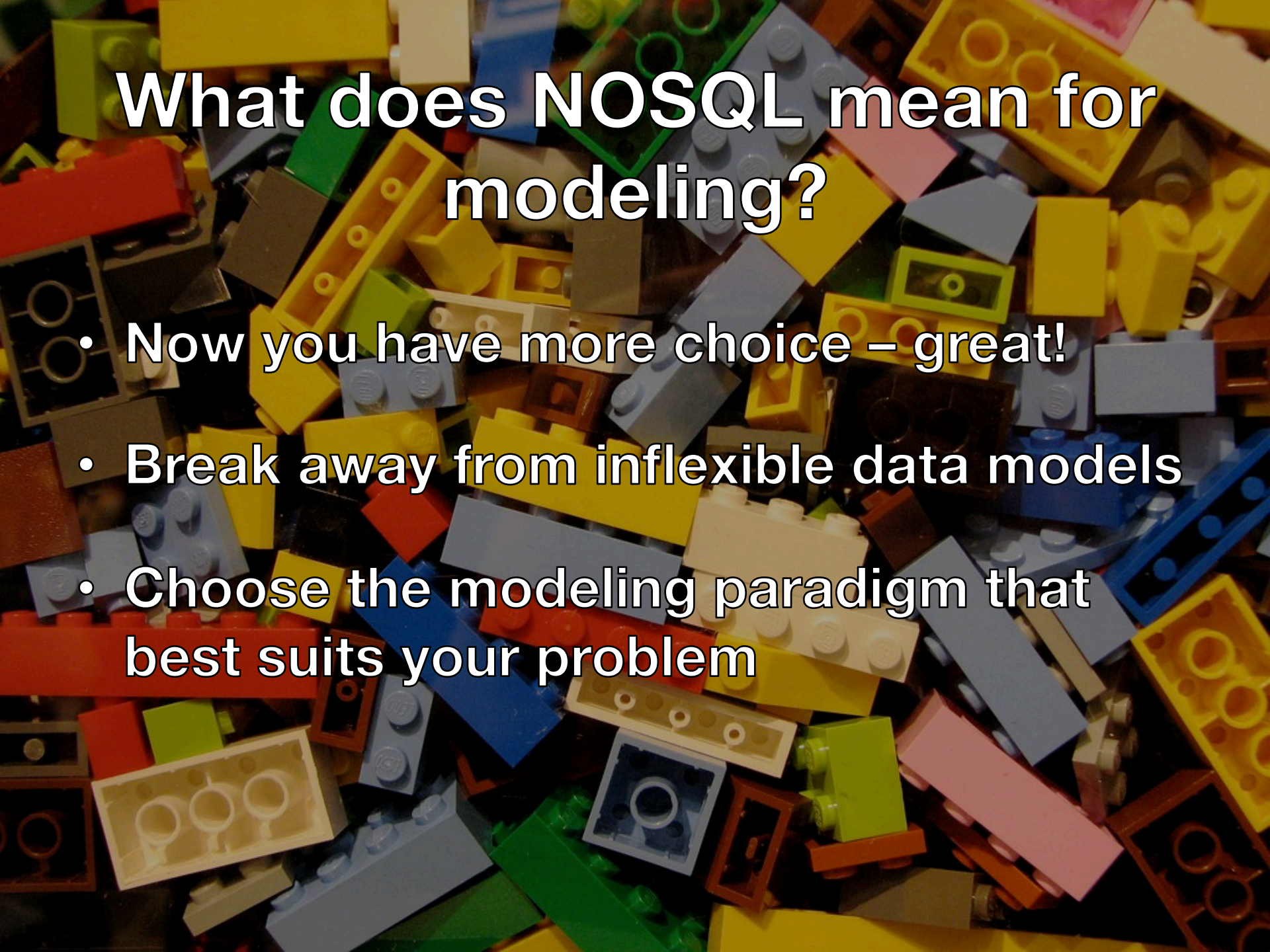
no:sql(br)/v2

Modeling with Graphs

alistair.jones@neotechnology.com
@apcj


What is modeling?
• Where does this data belong?
• How should I query and update it?



What does NOSQL mean for modeling?

- Now you have more choice – great!
- Break away from inflexible data models
- Choose the modeling paradigm that best suits your problem

Simple

Model Complexity

Complex

Key-Value

K ⟶ V
K ⟶ V
K ⟶ V
K ⟶ V
K ⟶ V
K ⟶ V
K ⟶ V
K ⟶ V

Keys

Values

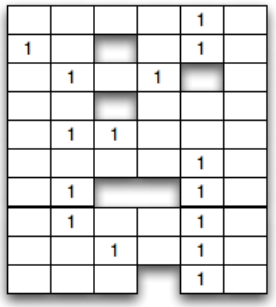
Document

K ⟶ { k: v, k: v }
K ⟶ { k: v, k: v }
K ⟶ { k: v, k: v }
K ⟶ { k: v, k: v }
K ⟶ { k: v, k: v }

Keys

Documents
(Structured Values)

BigTable



Rows

Columns

(but no joins)

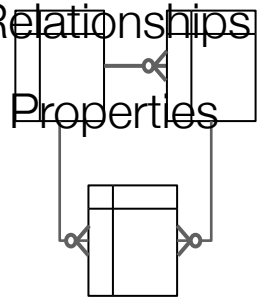
Graph

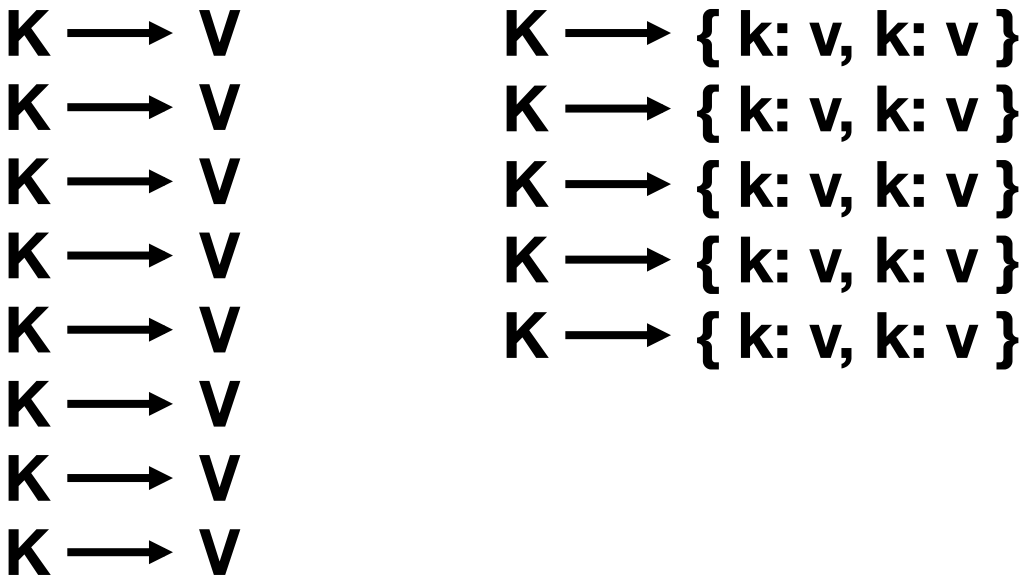
Nodes

Relationships

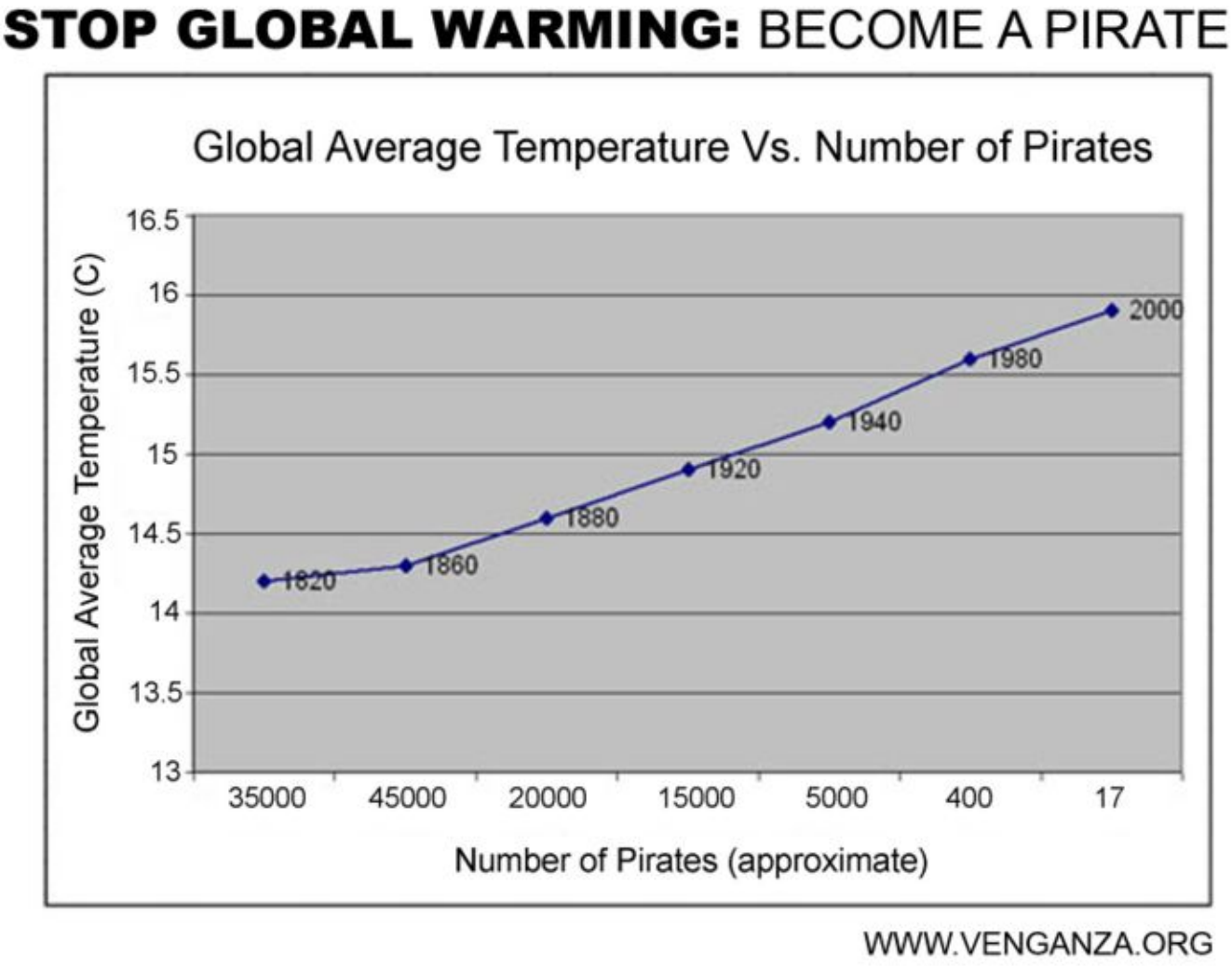
Properties

Relational

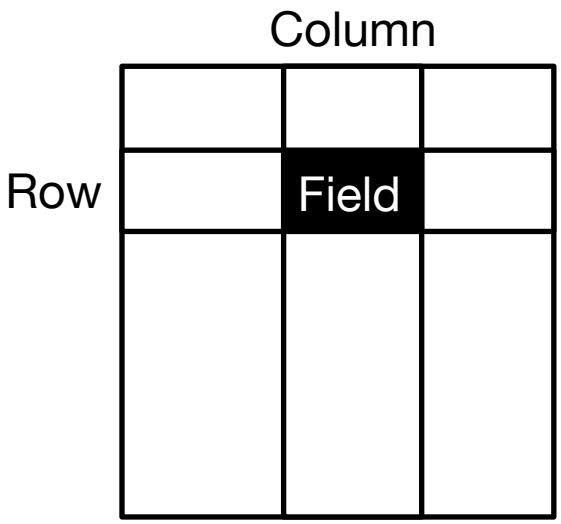



K → V
K → V
K → V
K → V
K → V
K → V
K → V
K → V
K → { k: v, k: v }
K → { k: v, k: v }
K → { k: v, k: v }
K → { k: v, k: v }
K → { k: v, k: v }

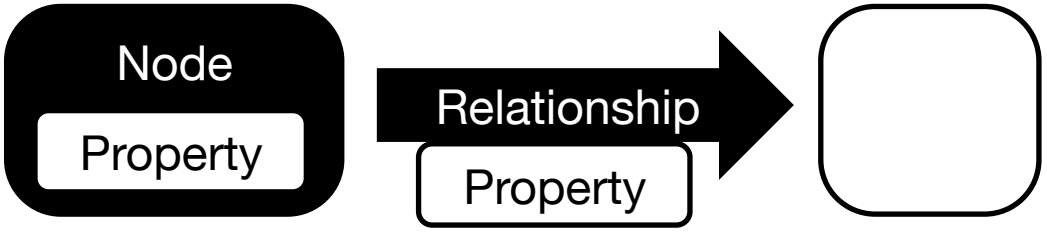
What's a graph?



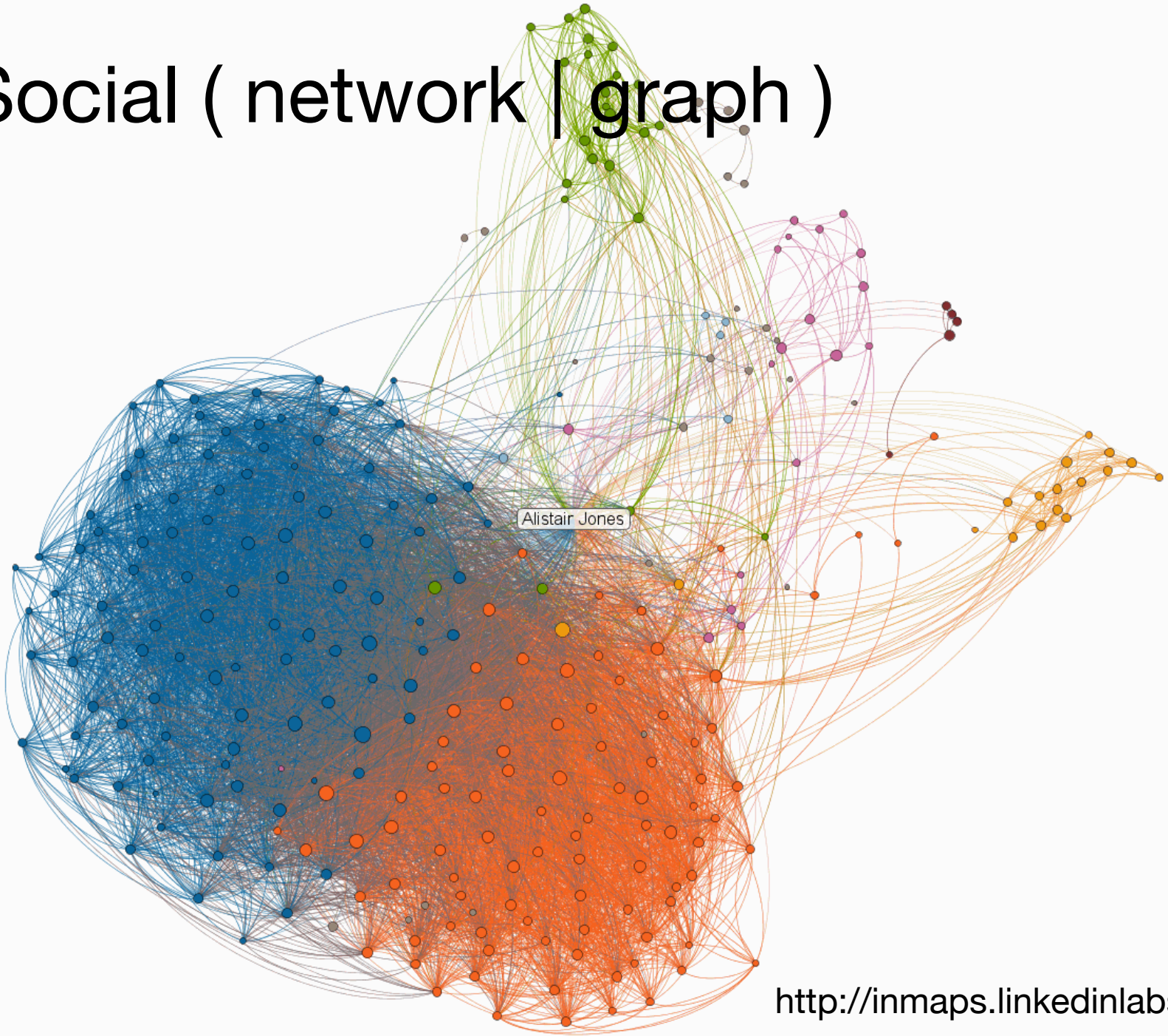
Relational



Graph

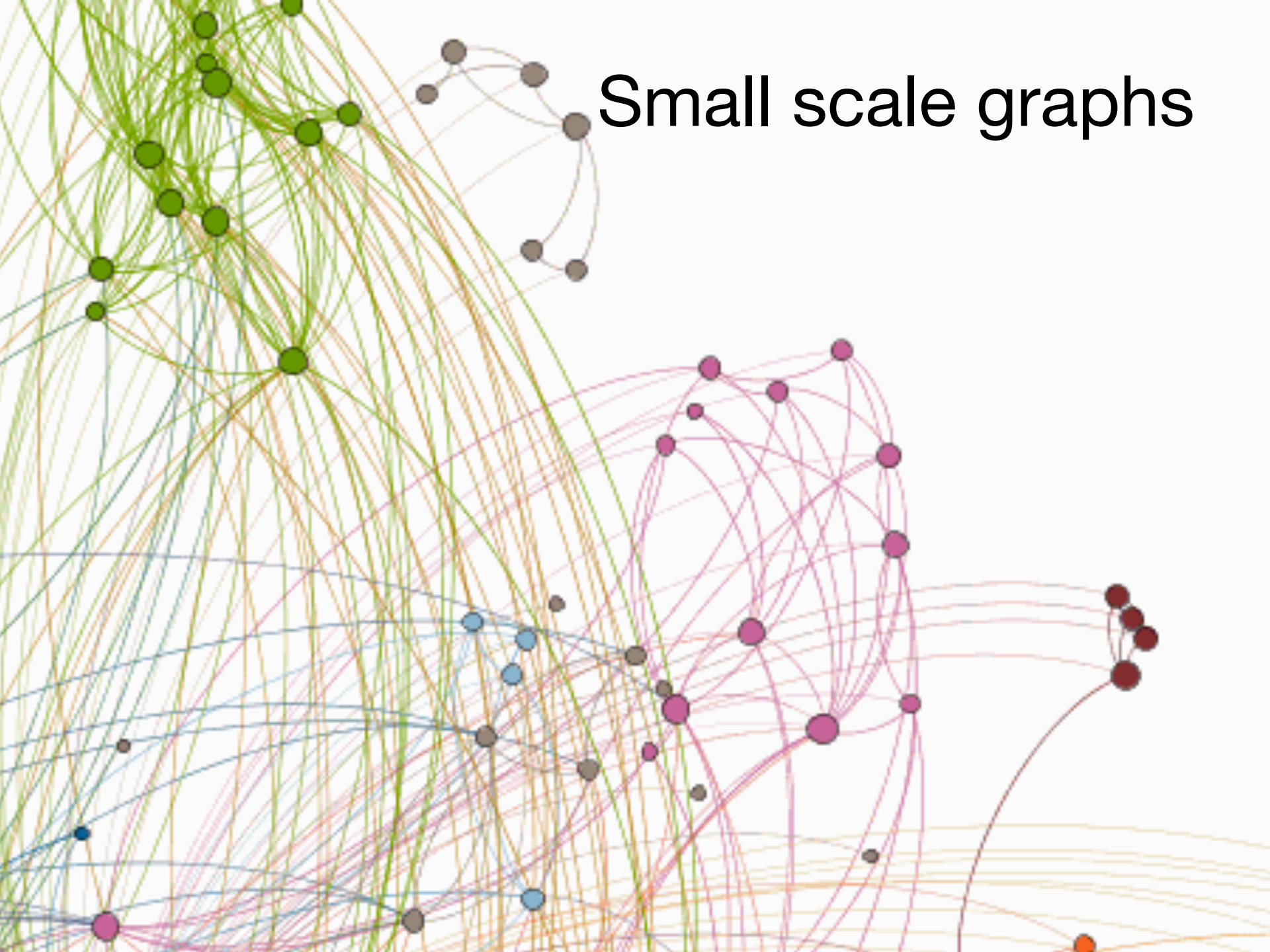


Social (network | graph)



http://inmaps.linkedinlabs.com/

Small scale graphs




Graph Databases
Are really good for
Highly connected data

Why choose a graph database?

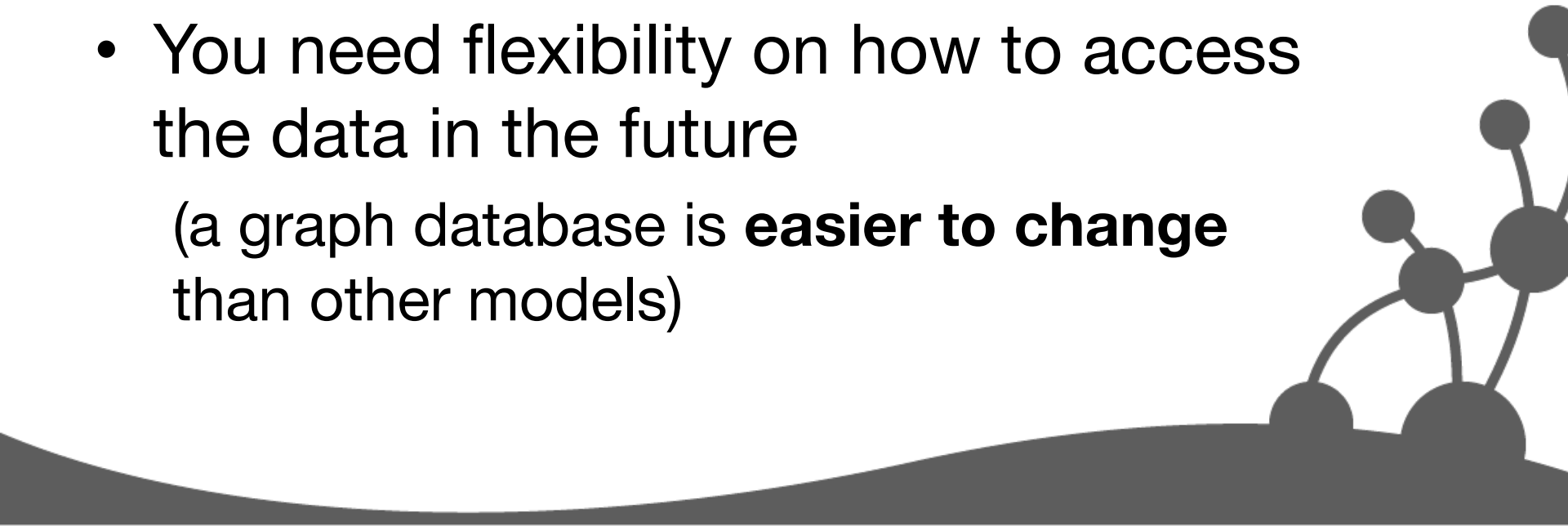
- You need to access data through multi-step traversals

  (find data in a **network** or **hierarchy**)

  or

- You need flexibility on how to access the data in the future

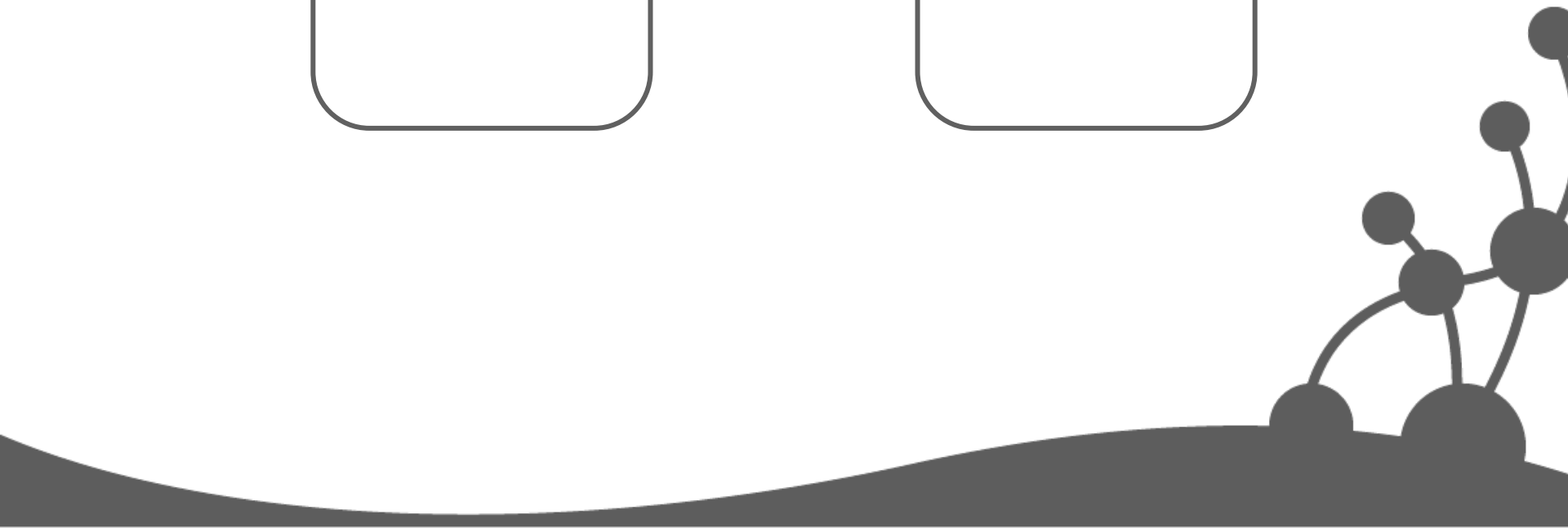
  (a graph database is **easier to change** than other models)



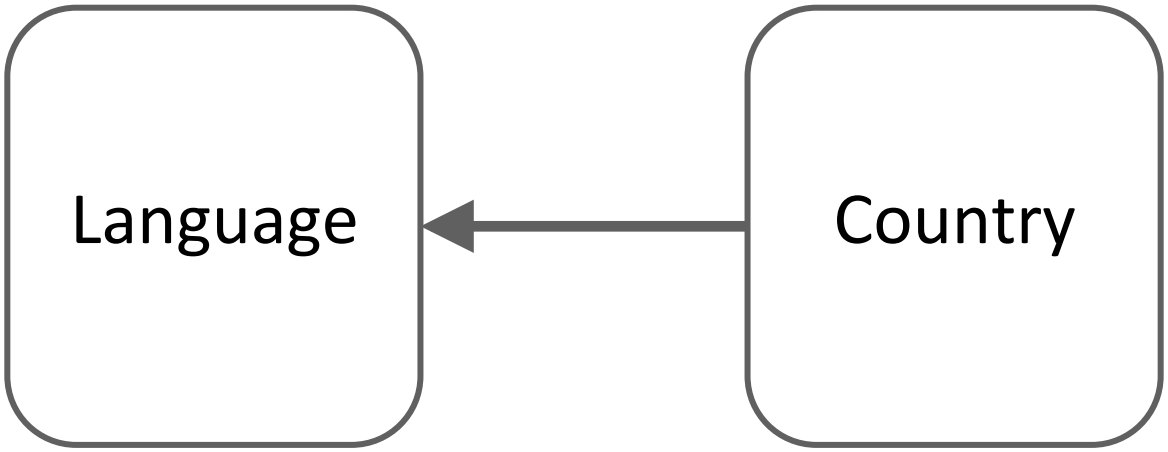
What language do they speak here?

Language

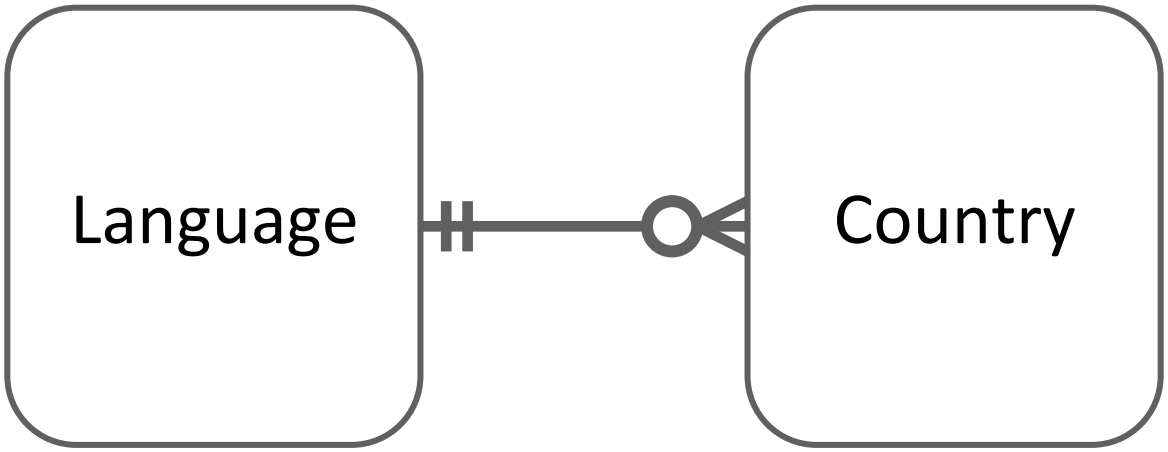
Country



What language do they speak here?



What language do they speak here?



Tables

Language

language_code

language_name

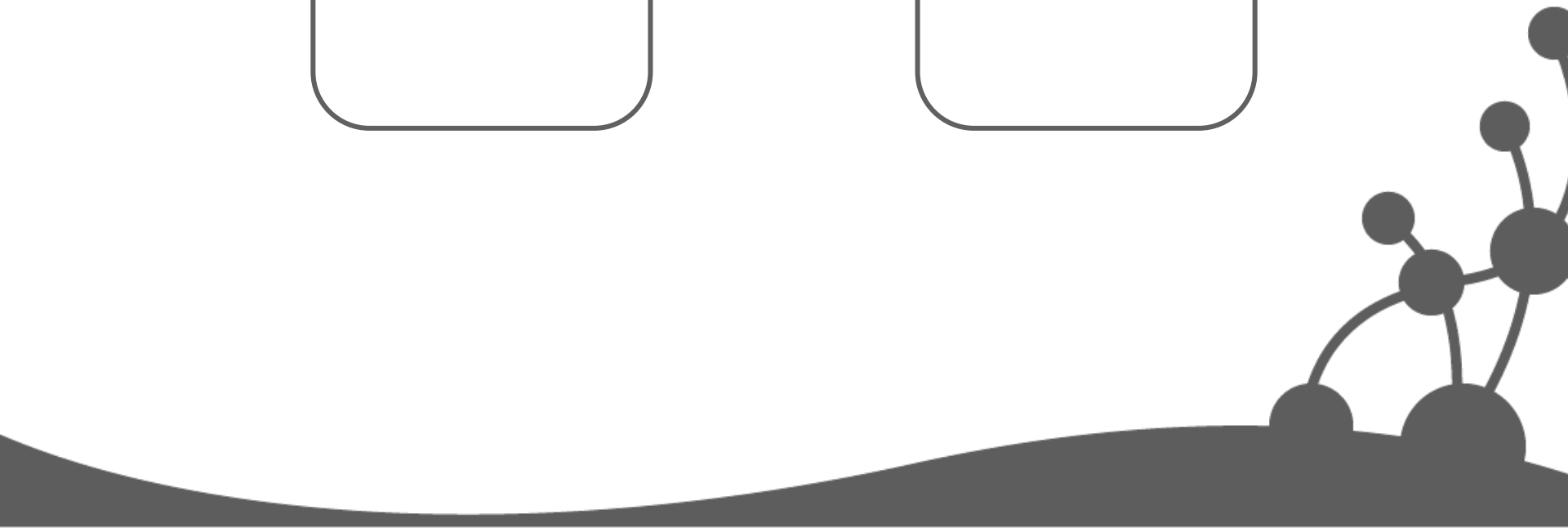
word_count

Country

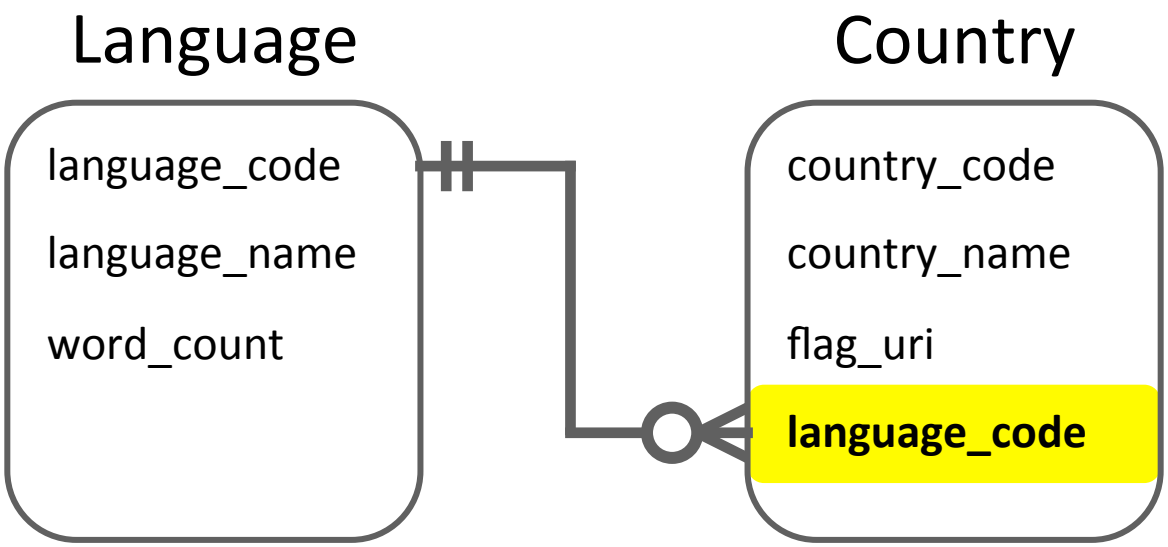
country_code

country_name

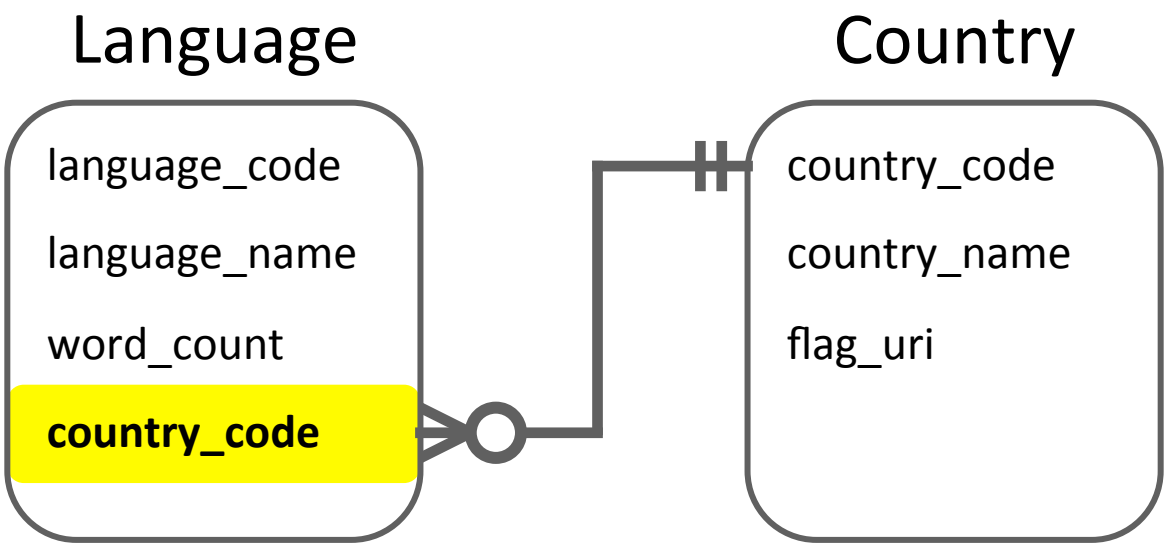
flag_uri



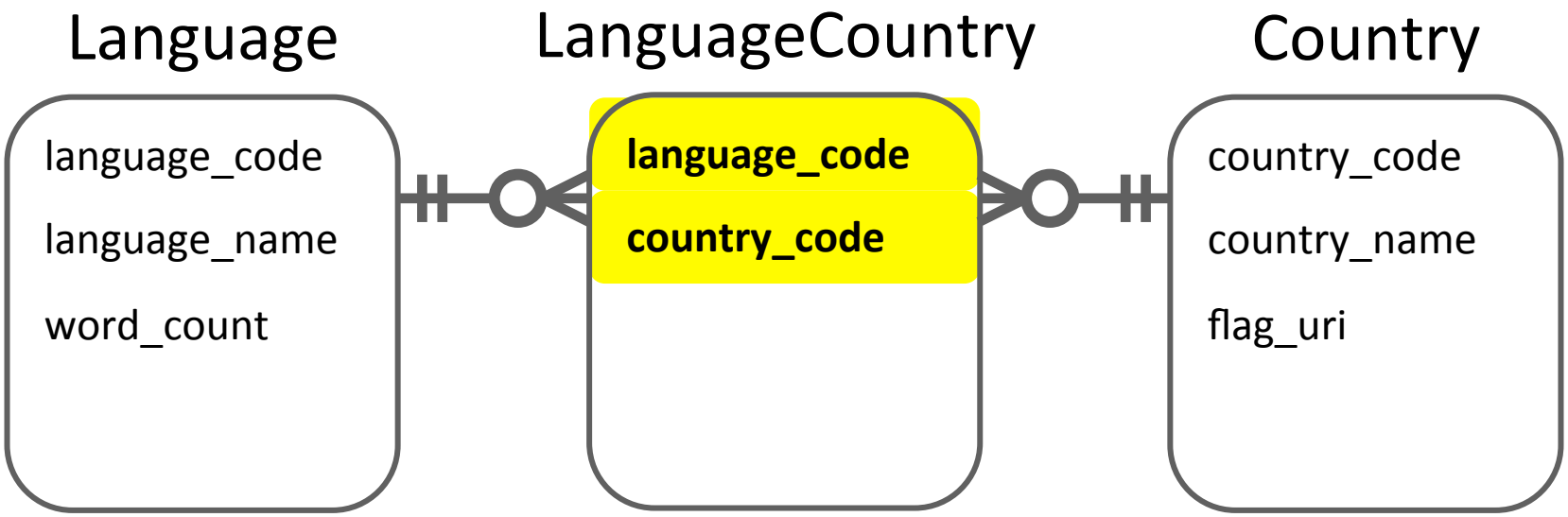
Need to model the relationship



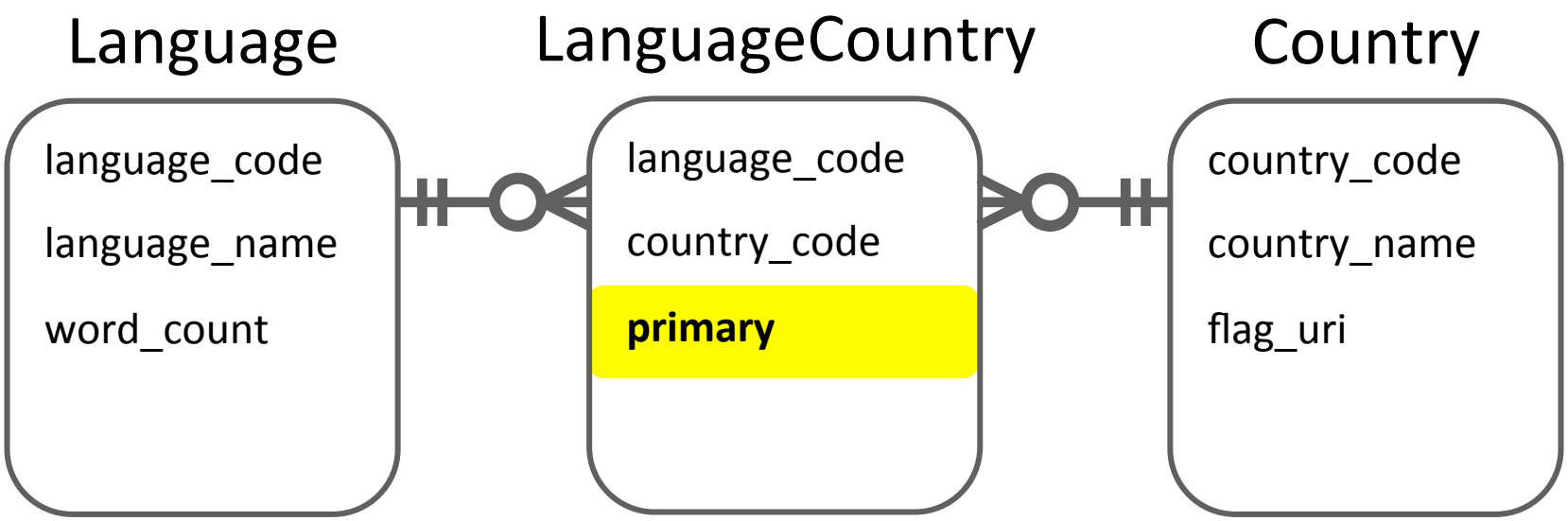
What if the cardinality changes?



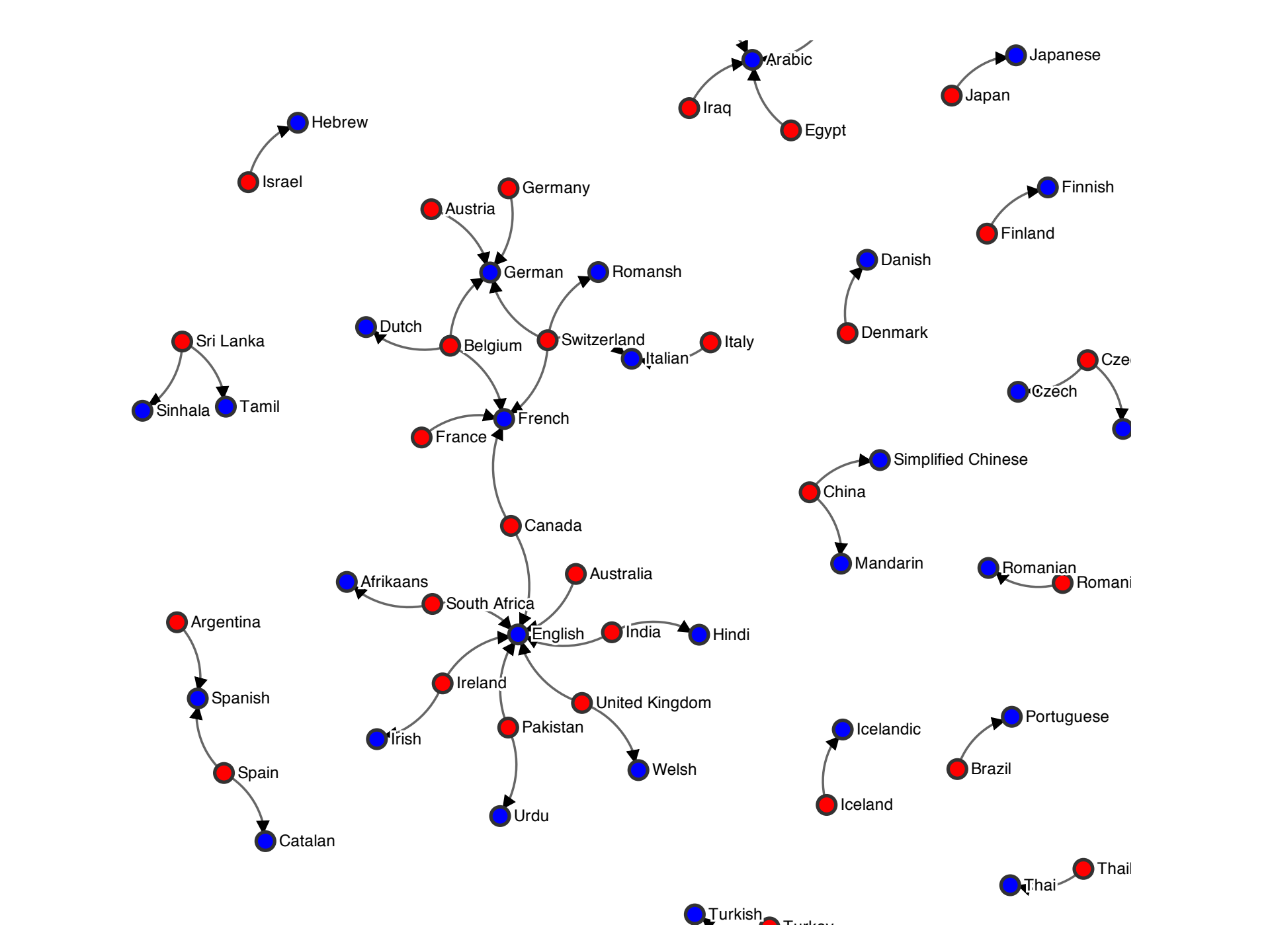
Or we go many-to-many?

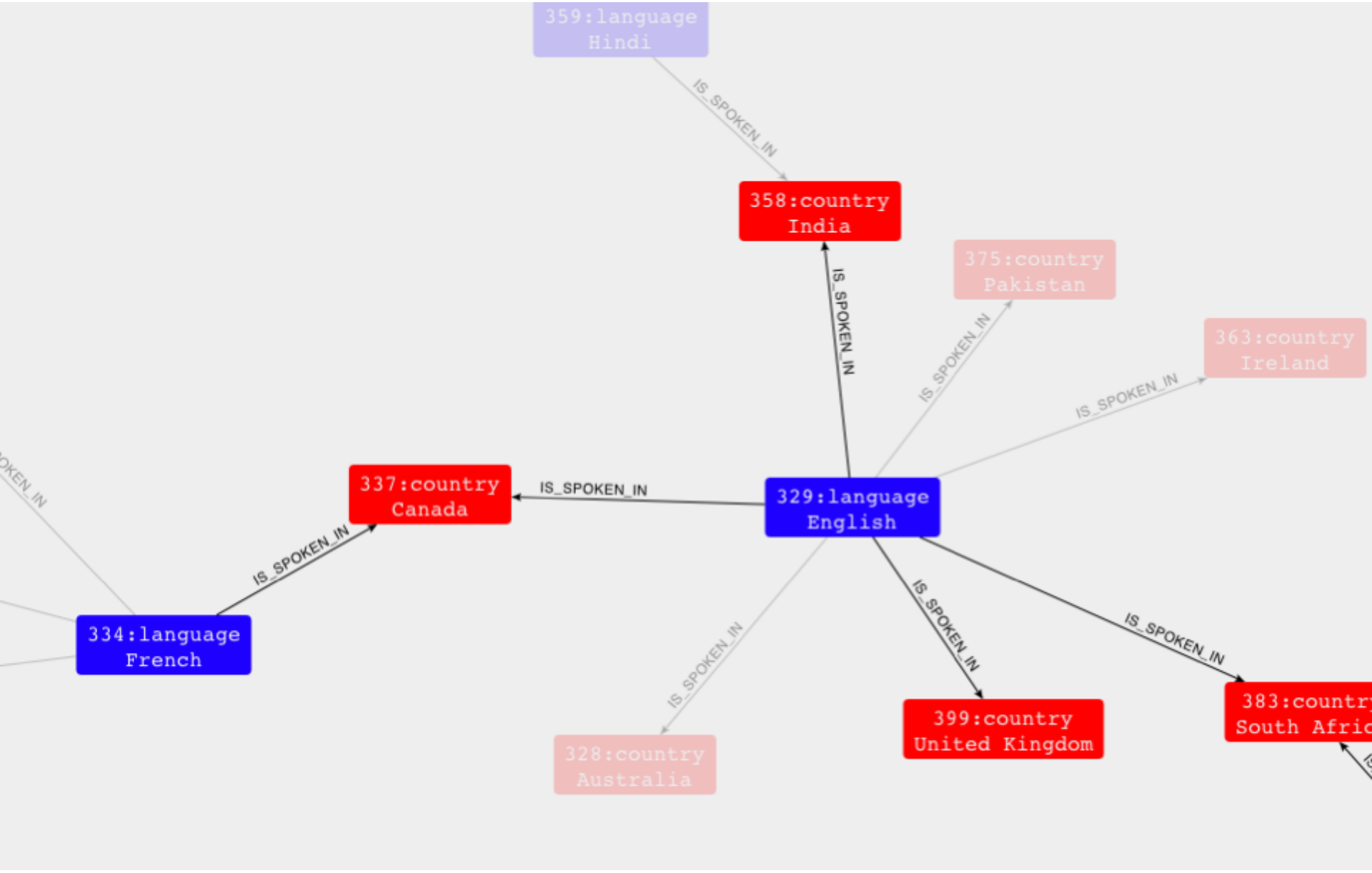


Or we want to qualify the relationship?

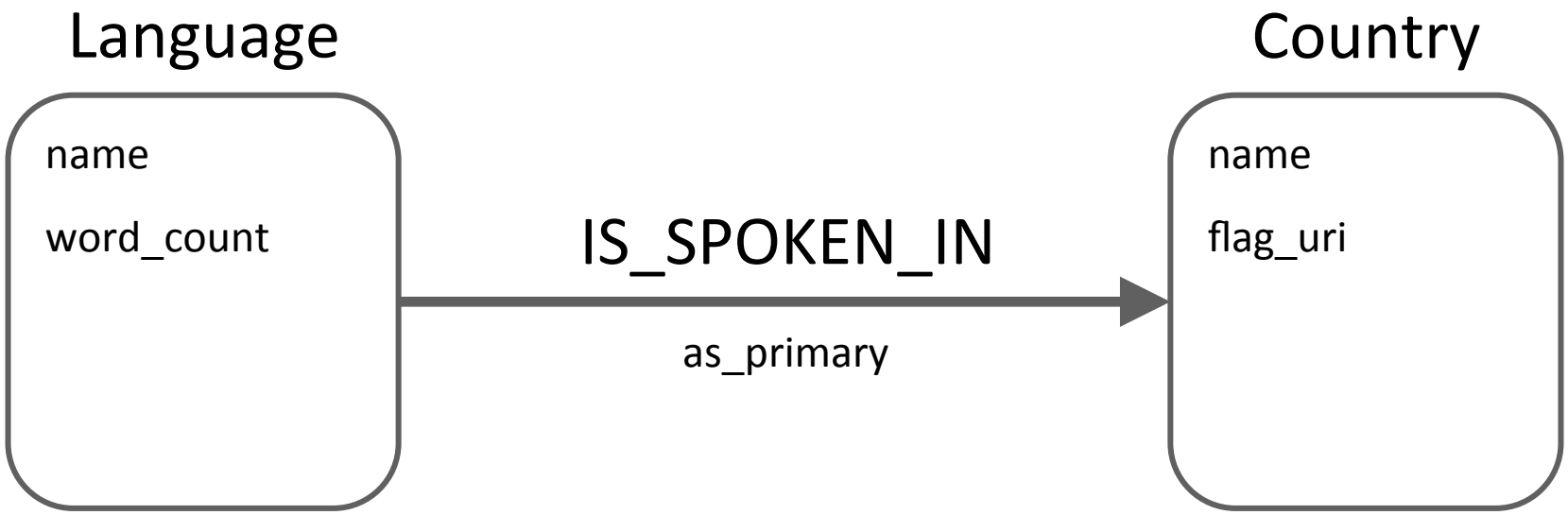


Explicit Relationship

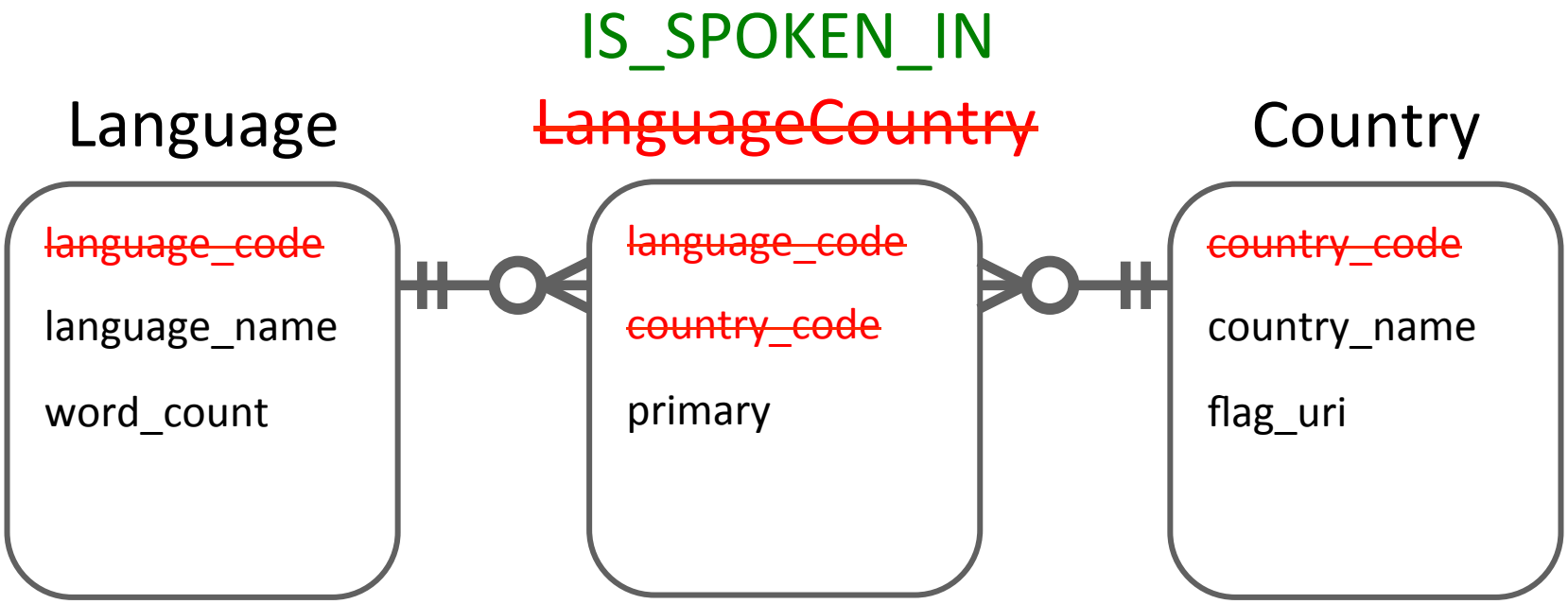

Arabic
Iraq
Egypt
Japanese
Japan
Hebrew
Israel
Germany
Austria
German
Romansh
Finnish
Finland
Danish
Denmark
Dutch
Belgium
Switzerland
Italian
Italy
Sri Lanka
Sinhala
Tamil
French
France
Czech
Simplified Chinese
China
Mandarin
Canada
Romanian
Romani
Afrikaans
South Africa
Australia
Argentina
English
India
Hindi
Spanish
Ireland
United Kingdom
Portuguese
Brazil
Icelandic
Iceland
Pakistan
Irish
Welsh
Spain
Urdu
Catalan
Thai
Turkish


359:language
Hindi
IS_SPOKEN_IN
358:country
India
375:country
Pakistan
363:country
Ireland
337:country
Canada
329:language
English
334:language
French
328:country
Australia
399:country
United Kingdom
383:country
South Afric

Relationship Properties



What's different?



- Implementation of maintaining relationships is left up to the database
- Surrogate keys disappear or are unnecessary (with the exception of node id)
- Relationships get an explicit name

Object (graph | relational) mapping

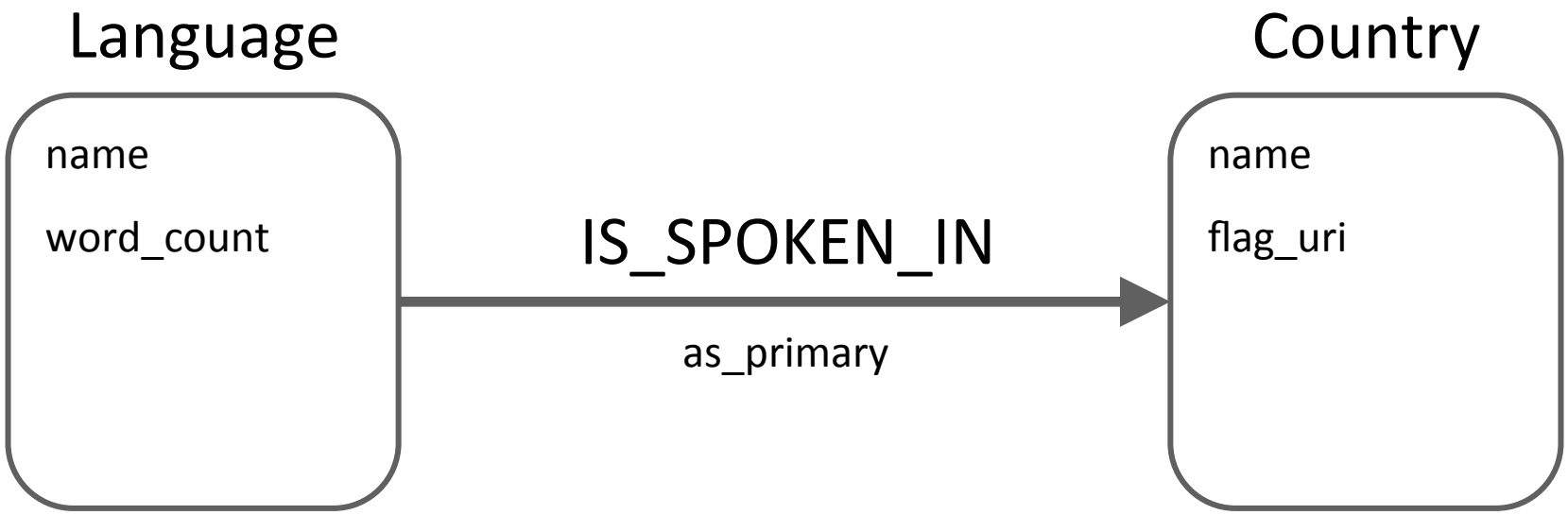
Relational		Graph
Java Domain Objects / POJOs	*Your Application Code*	Java Domain Objects / POJOs
Hibernate	*Object Mapping Library*	Spring Data Neo4J
SQL	*Query Language*	Cypher
Tables Rows Columns Foreign Keys	*Core Database Concepts*	Nodes Relationships Properties

Spring Data Neo4j

```
@NodeEntity
public class Language {

  @Indexed
  String languageCode;

  String name;

  int wordCount;

  @RelatedTo(type="SPOKEN_IN",
      direction = OUTGOING)
  Set<Country> countriesSpokenIn;

}
```

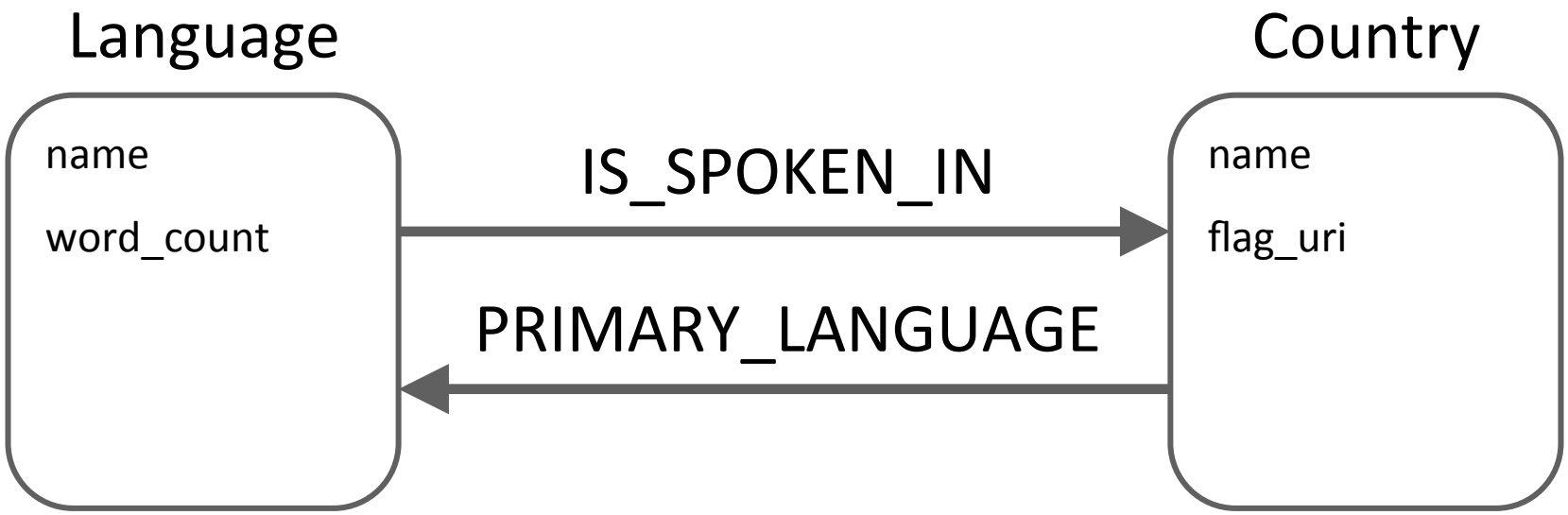
```
@NodeEntity
public class Country {

  @Indexed
  String countryCode;

  String name;

  String flagUri;

  @RelatedTo(type="PRIMARY_LANGUAGE",
      direction = OUTGOING)
  Language primaryLanguage;

}
```

Relationship specialisation

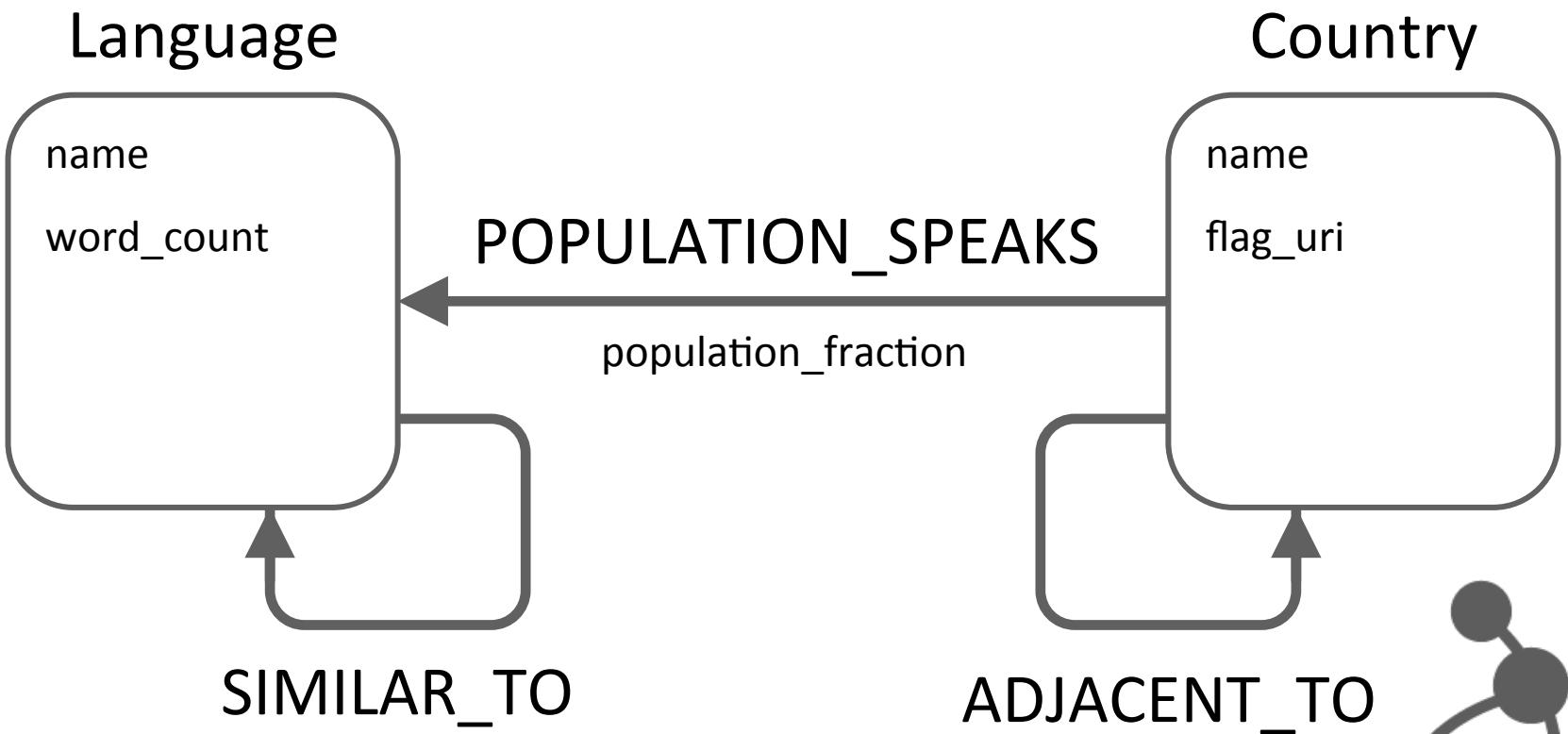


Weighted relationships

Bidirectional relationships



Keep on adding relationships



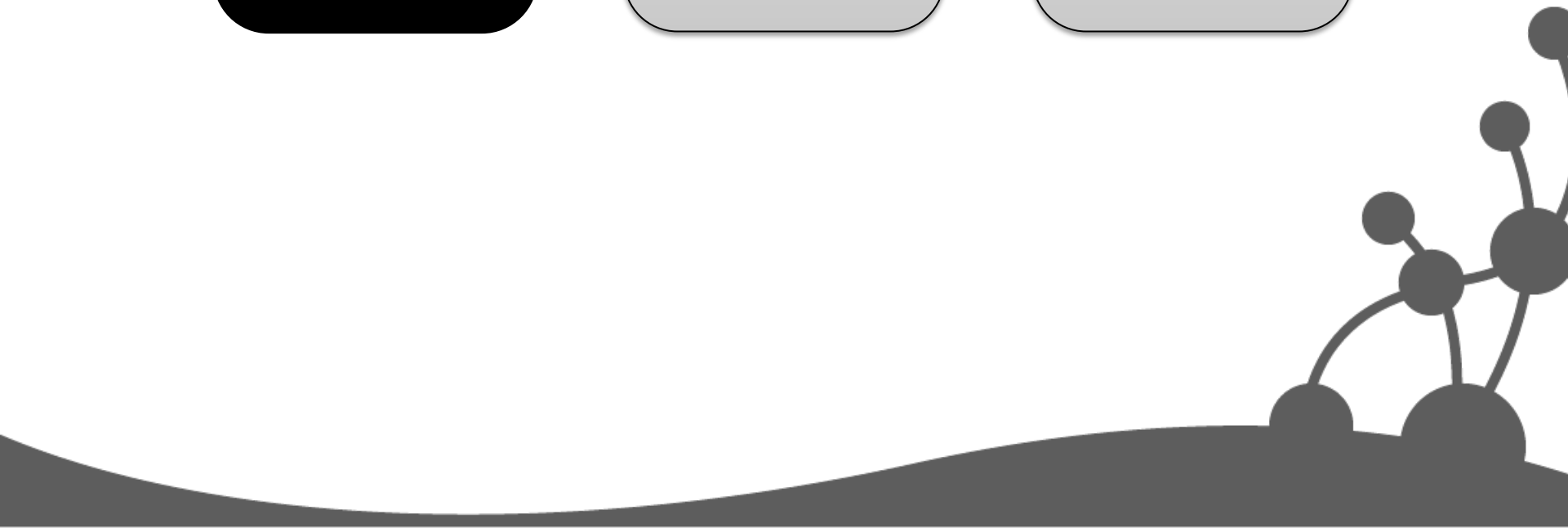
Graph Modeling Principles

Graph Modeling Principles

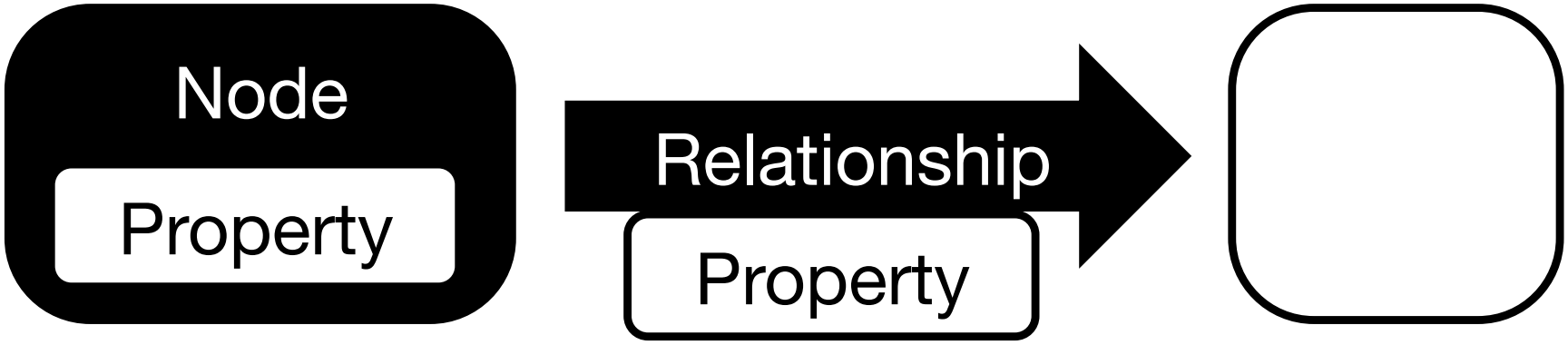
Embrace the Paradigm

2

3



Use the building blocks



Anti-pattern: rich properties

name: “Canada”

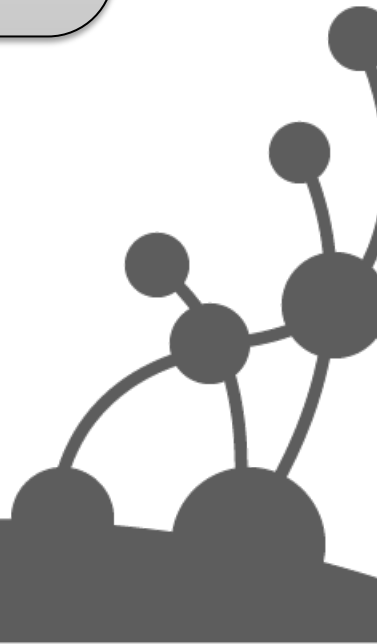
languages_spoken: “[‘English’, ‘French’]”

Graph Modeling Principles

Embrace the Paradigm

Nodes for Identity

3



Node represents multiple concepts

Country

name

flag_uri

language_name

number_of_words

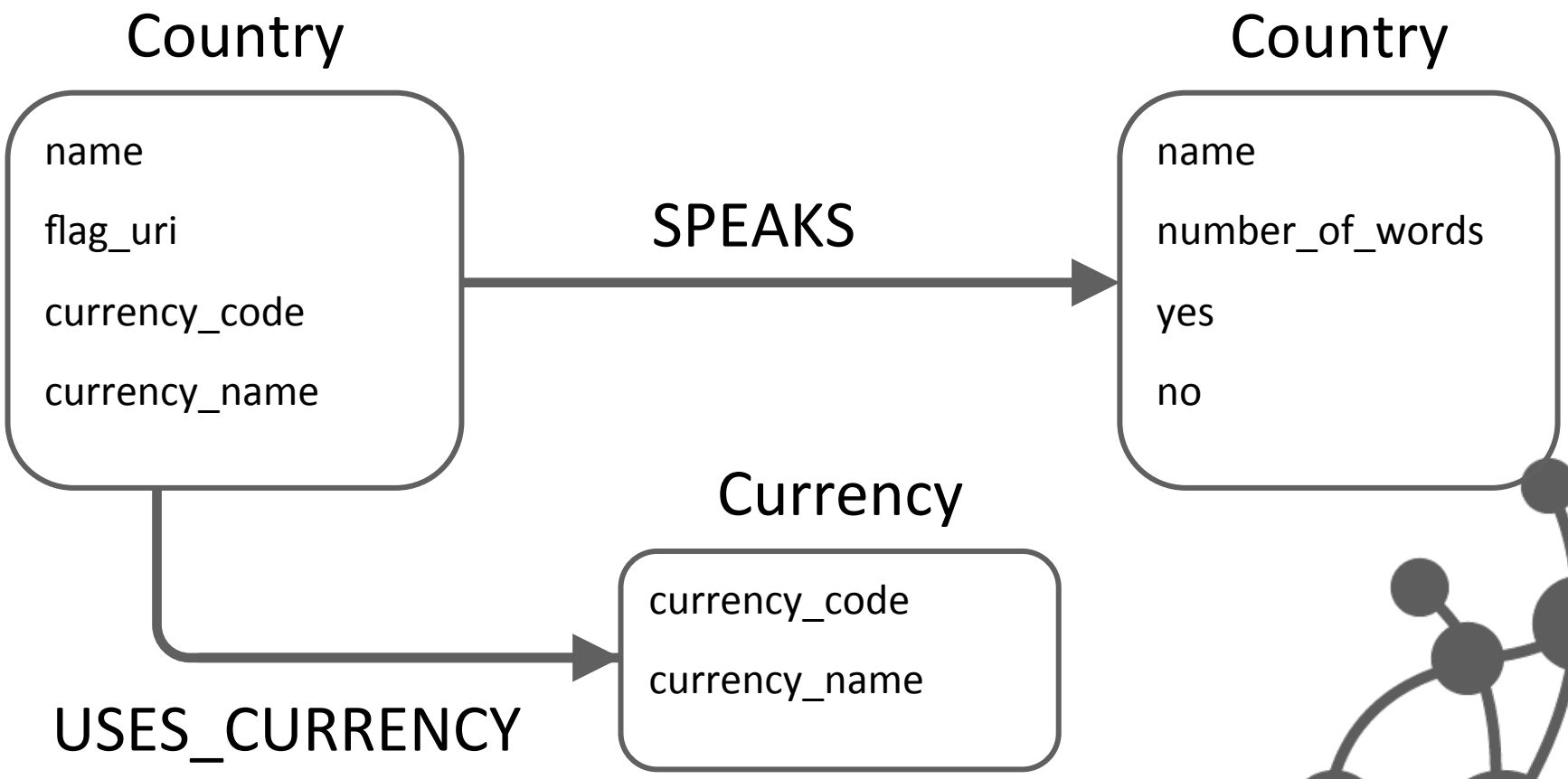
yes_in_langauge

no_in_language

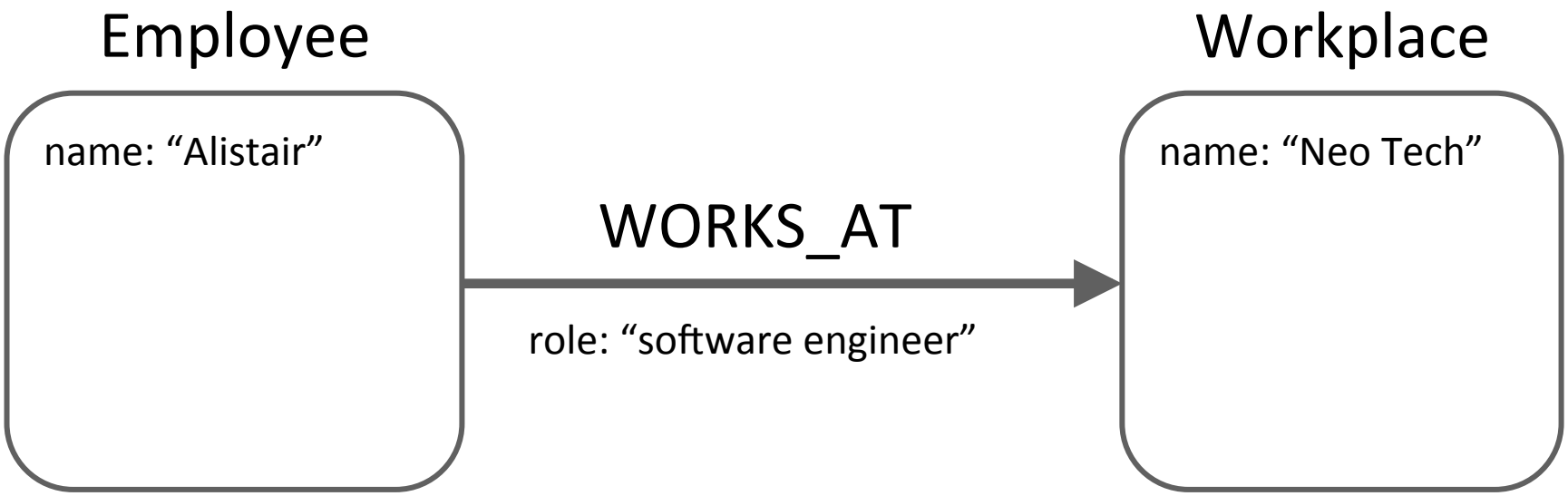
currency_code

currency_name

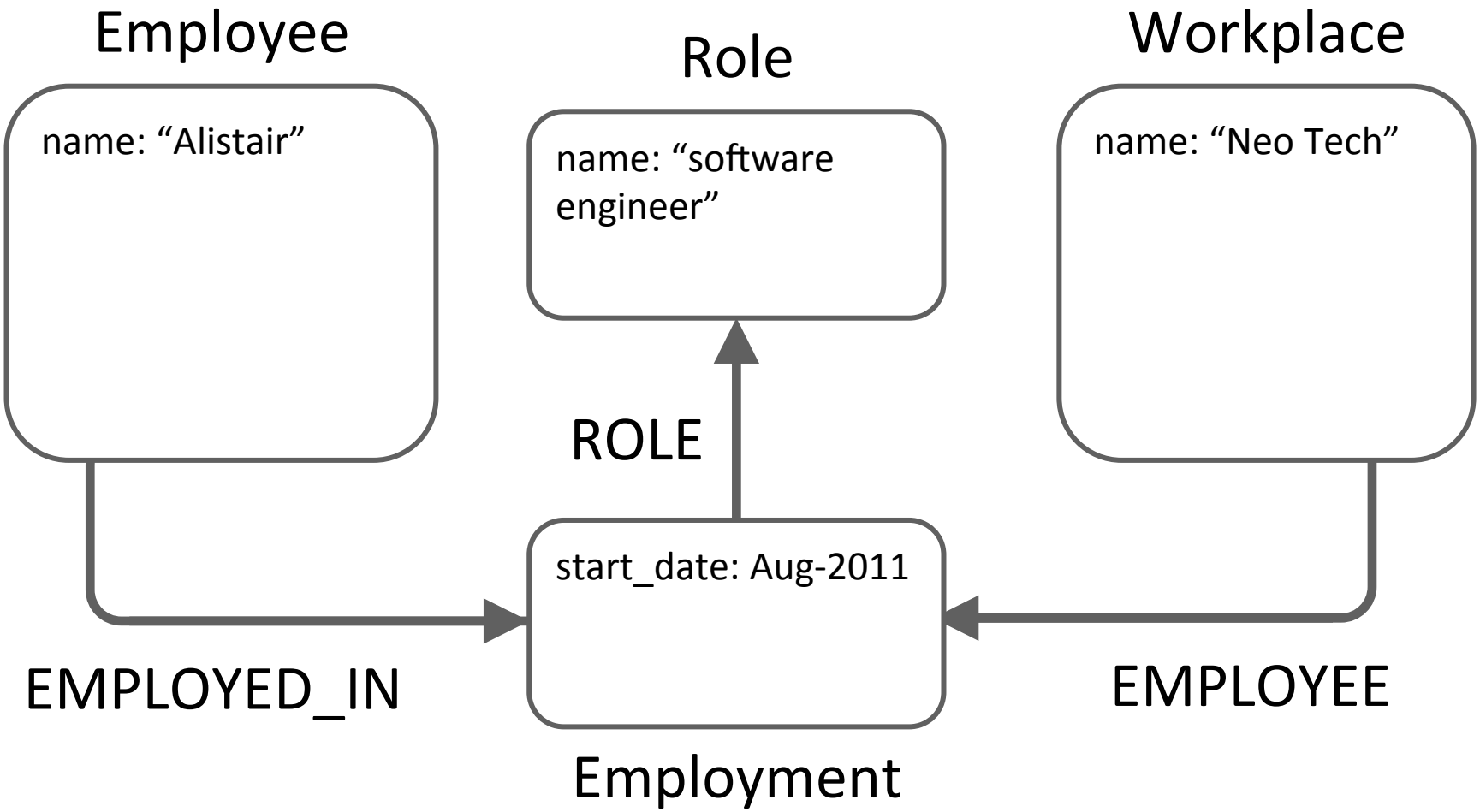
Break out separate concepts



Property represents entity



(Reify | Nodify) connecting entities

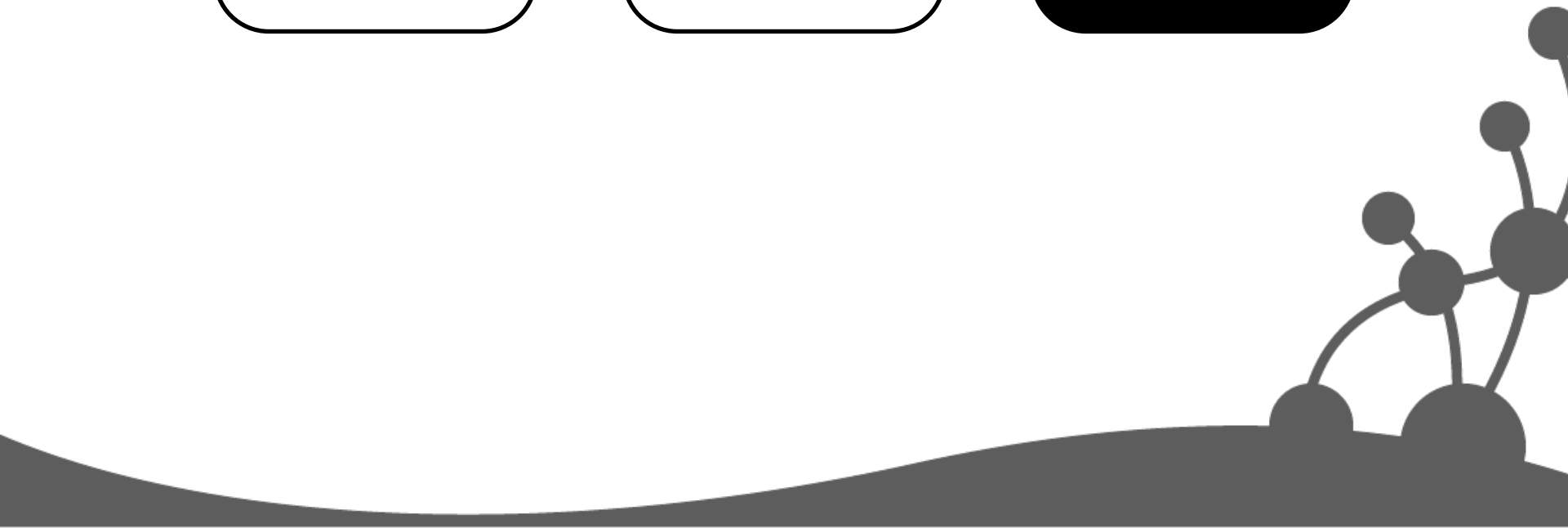


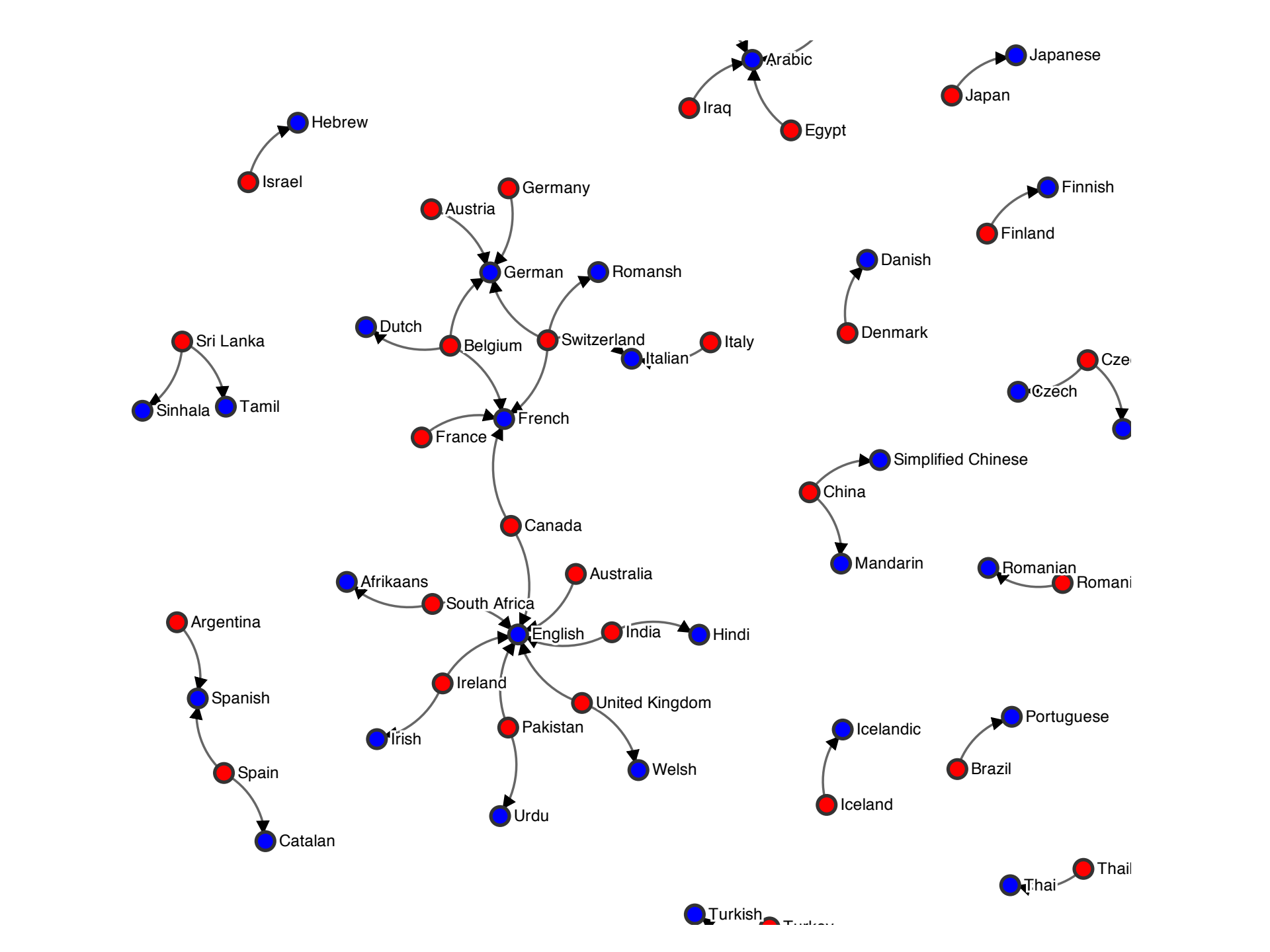
Graph Modeling Principles

Embrace the Paradigm

Nodes for Identity

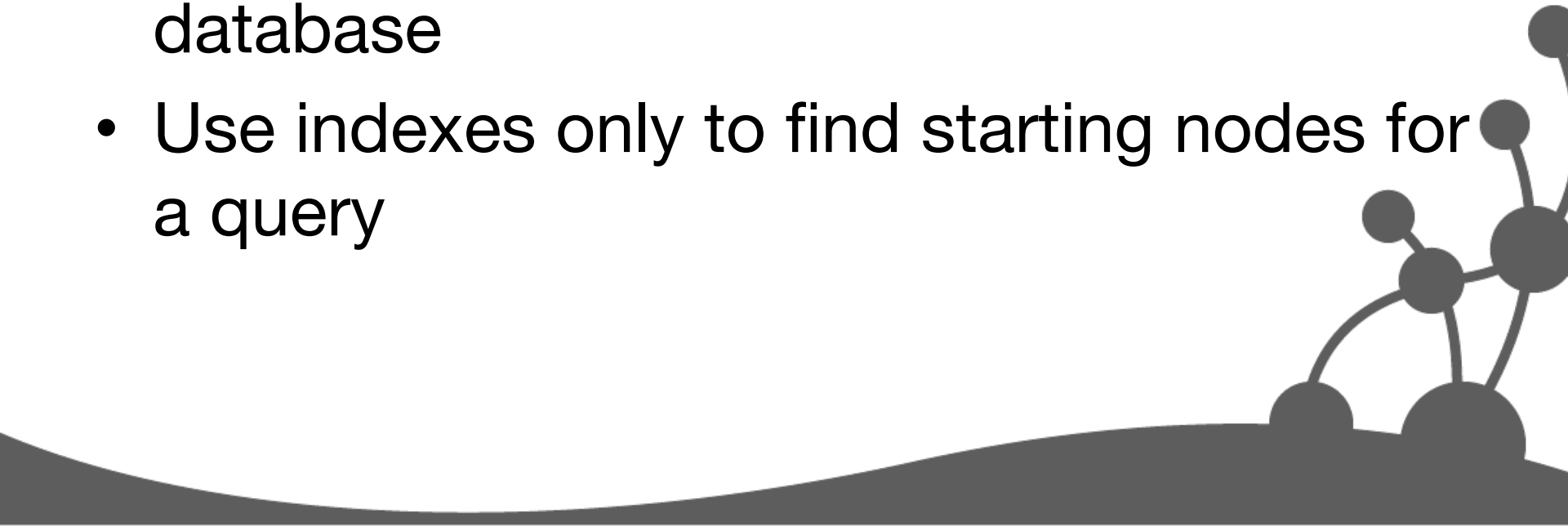
Relation-ships for Access




Arabic
Iraq
Egypt
Japanese
Japan
Hebrew
Israel
Germany
Austria
German
Romansh
Finnish
Finland
Danish
Denmark
Dutch
Belgium
Switzerland
Italian
Italy
Sri Lanka
Sinhala
Tamil
French
France
Czech
Simplified Chinese
China
Mandarin
Canada
Romanian
Romani
Afrikaans
South Africa
Australia
Argentina
English
India
Hindi
Spanish
Ireland
United Kingdom
Portuguese
Brazil
Icelandic
Iceland
Pakistan
Irish
Welsh
Spain
Urdu
Catalan
Thai
Turkish

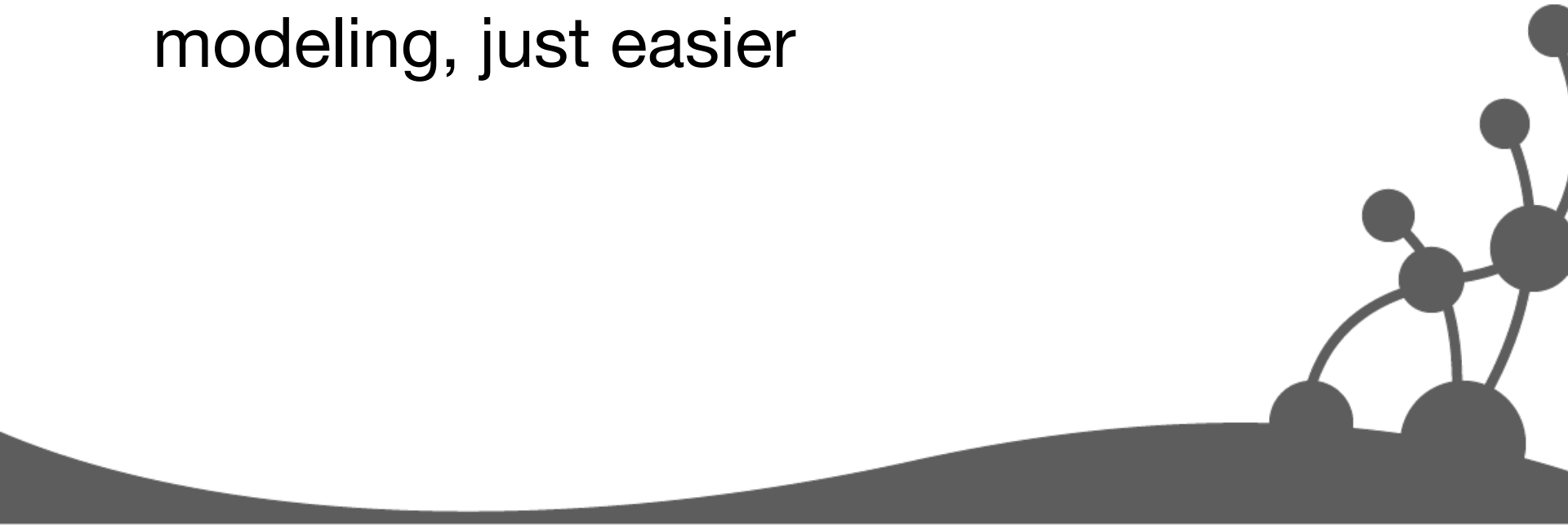
Relationships for querying

- Relationships should the primary means to access nodes in the database
- Traversing relationships is cheap – that's the whole design goal of a graph database
- Use indexes only to find starting nodes for a query



Reminders

- Is your data connected?
- What's the right database for your application?
- Modeling with graphs similar to relational modeling, just easier



alistair.jones@neotechnology.com
@apcj