A Little Graph Theory for the Busy Developer

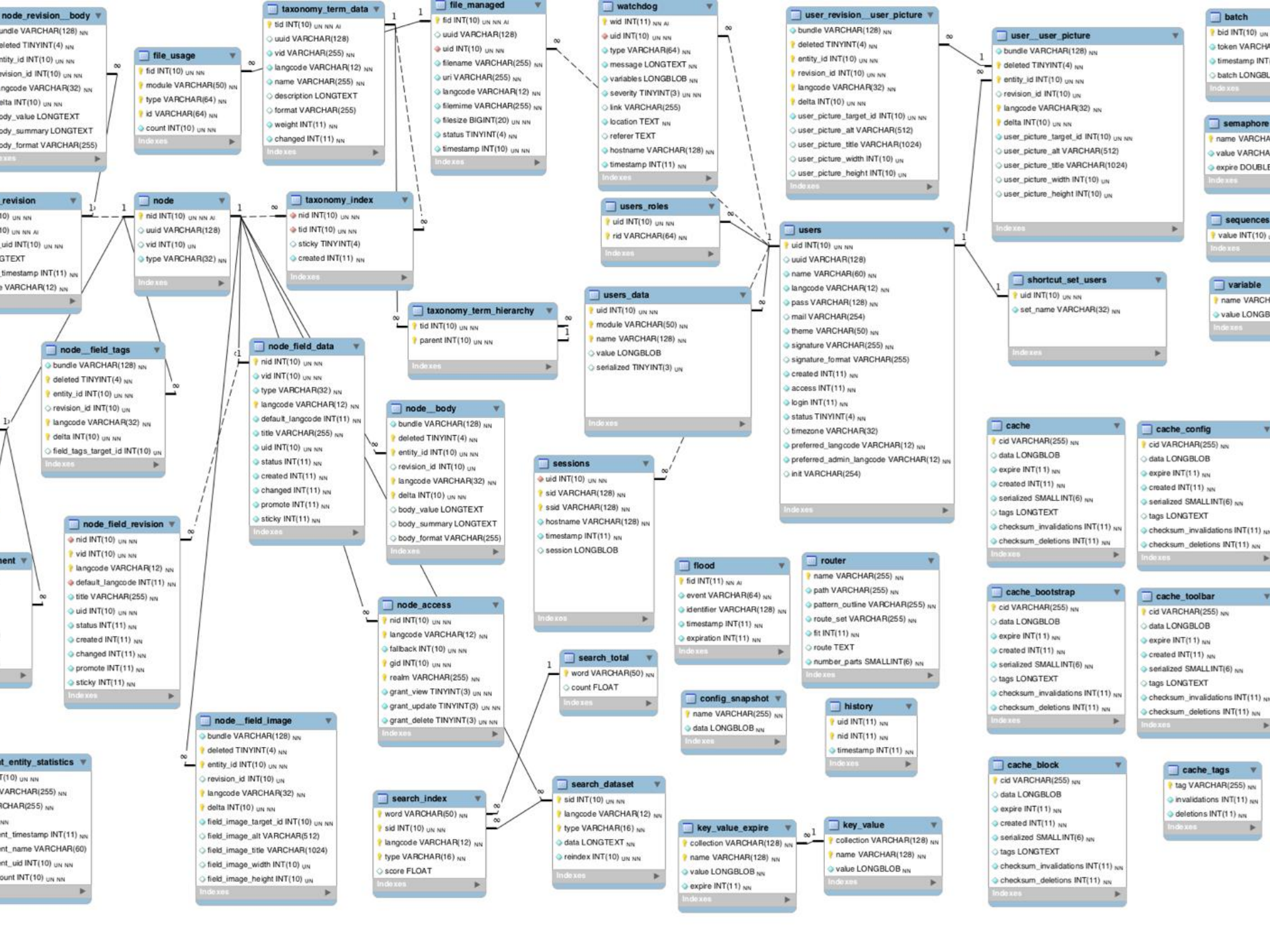
Dr. Jim Webber

Chief Scientist, Neo Technology

@jimwebber

Roadmap

- Imprisoned data models
  - Why most NoSQL stores and RDBMS are clumsy for connected data
- Labeled Property Graph model
- Graph theory for predictive analytics
  - South East London and World War I
- Graph matching for real-time insight
  - Beer, nappies and Xbox
- Q&A


node_revision__body
undle VARCHAR(128) NN
eleted TINYINT(4) NN
ntity_id INT(10) UN NN
evision_id INT(10) UN NN
ngcode VARCHAR(32) NN
elta INT(10) UN NN
ody_value LONGTEXT
ody_summary LONGTEXT
ody_format VARCHAR(255)
taxonomy_term_data
tid INT(10) UN NN AI
uuid VARCHAR(128)
vid VARCHAR(255) NN
langcode VARCHAR(12) NN
name VARCHAR(255) NN
description LONGTEXT
format VARCHAR(255)
weight INT(11) NN
changed INT(11) NN
Indexes
file_managed
fid INT(10) UN NN AI
uuid VARCHAR(128)
uid INT(10) UN NN
filename VARCHAR(255) NN
uri VARCHAR(255) NN
langcode VARCHAR(12) NN
filemime VARCHAR(255) NN
filesize BIGINT(20) UN NN
status TINYINT(4) NN
timestamp INT(10) UN NN
Indexes
watchdog
wid INT(11) NN AI
uid INT(10) UN NN
type VARCHAR(64) NN
message LONGTEXT NN
variables LONGBLOB NN
severity TINYINT(3) UN NN
link VARCHAR(255)
location TEXT NN
referer TEXT
hostname VARCHAR(128) NN
timestamp INT(11) NN
Indexes
user_revision__user_picture
bundle VARCHAR(128) NN
deleted TINYINT(4) NN
entity_id INT(10) UN NN
revision_id INT(10) UN NN
langcode VARCHAR(32) NN
delta INT(10) UN NN
user_picture_target_id INT(10) UN NN
user_picture_alt VARCHAR(512)
user_picture_title VARCHAR(1024)
user_picture_width INT(10) UN
user_picture_height INT(10) UN
Indexes
user__user_picture
bundle VARCHAR(128) NN
deleted TINYINT(4) NN
entity_id INT(10) UN NN
revision_id INT(10) UN
langcode VARCHAR(32) NN
delta INT(10) UN NN
user_picture_target_id INT(10) UN NN
user_picture_alt VARCHAR(512)
user_picture_title VARCHAR(1024)
user_picture_width INT(10) UN
user_picture_height INT(10) UN
Indexes
batch
bid INT(10) UN
token VARCHA
timestamp INT(
batch LONGBL
Indexes
semaphore
name VARCHA
value VARCHA
expire DOUBLE
Indexes
sequences
value INT(10)
Indexes
variable
name VARCH
value LONGB
Indexes
file_usage
fid INT(10) UN NN
module VARCHAR(50) NN
type VARCHAR(64) NN
id VARCHAR(64) NN
count INT(10) UN NN
Indexes
node
nid INT(10) UN NN AI
uuid VARCHAR(128)
vid INT(10) UN
type VARCHAR(32) NN
Indexes
taxonomy_index
nid INT(10) UN NN
tid INT(10) UN NN
sticky TINYINT(4)
created INT(11) NN
Indexes
users_roles
uid INT(10) UN NN
rid VARCHAR(64) NN
Indexes
users
uid INT(10) UN NN
uuid VARCHAR(128)
name VARCHAR(60) NN
langcode VARCHAR(12) NN
pass VARCHAR(128) NN
mail VARCHAR(254)
theme VARCHAR(50) NN
signature VARCHAR(255) NN
signature_format VARCHAR(255)
created INT(11) NN
access INT(11) NN
login INT(11) NN
status TINYINT(4) NN
timezone VARCHAR(32)
preferred_langcode VARCHAR(12) NN
preferred_admin_langcode VARCHAR(12) NN
init VARCHAR(254)
Indexes
shortcut_set_users
uid INT(10) UN NN
set_name VARCHAR(32) NN
Indexes
taxonomy_term_hierarchy
tid INT(10) UN NN
parent INT(10) UN NN
Indexes
users_data
uid INT(10) UN NN
module VARCHAR(50) NN
name VARCHAR(128) NN
value LONGBLOB
serialized TINYINT(3) UN
node__field_tags
bundle VARCHAR(128) NN
deleted TINYINT(4) NN
entity_id INT(10) UN NN
revision_id INT(10) UN
langcode VARCHAR(32) NN
delta INT(10) UN NN
field_tags_target_id INT(10) UN
Indexes
node_field_data
nid INT(10) UN NN
vid INT(10) UN NN
type VARCHAR(32) NN
langcode VARCHAR(12) NN
default_langcode INT(11) NN
title VARCHAR(255) NN
uid INT(10) UN NN
status INT(11) NN
created INT(11) NN
changed INT(11) NN
promote INT(11) NN
sticky INT(11) NN
Indexes
node__body
bundle VARCHAR(128) NN
deleted TINYINT(4) NN
entity_id INT(10) UN NN
revision_id INT(10) UN
langcode VARCHAR(32) NN
delta INT(10) UN NN
body_value LONGTEXT
body_summary LONGTEXT
body_format VARCHAR(255)
Indexes
sessions
uid INT(10) UN NN
sid VARCHAR(128) NN
ssid VARCHAR(128) NN
hostname VARCHAR(128) NN
timestamp INT(11) NN
session LONGBLOB
Indexes
cache
cid VARCHAR(255) NN
data LONGBLOB
expire INT(11) NN
created INT(11) NN
serialized SMALLINT(6) NN
tags LONGTEXT
checksum_invalidations INT(11) NN
checksum_deletions INT(11) NN
Indexes
cache_config
cid VARCHAR(255) NN
data LONGBLOB
expire INT(11) NN
created INT(11) NN
serialized SMALLINT(6) NN
tags LONGTEXT
checksum_invalidations INT(11) NN
checksum_deletions INT(11) NN
Indexes
node_field_revision
nid INT(10) UN NN
vid INT(10) UN NN
langcode VARCHAR(12) NN
default_langcode INT(11) NN
title VARCHAR(255) NN
uid INT(10) UN NN
status INT(11) NN
created INT(11) NN
changed INT(11) NN
promote INT(11) NN
sticky INT(11) NN
Indexes
flood
fid INT(11) NN AI
event VARCHAR(64) NN
identifier VARCHAR(128) NN
timestamp INT(11) NN
expiration INT(11) NN
Indexes
router
name VARCHAR(255) NN
path VARCHAR(255) NN
pattern_outline VARCHAR(255) NN
route_set VARCHAR(255) NN
fit INT(11) NN
route TEXT
number_parts SMALLINT(6) NN
Indexes
cache_bootstrap
cid VARCHAR(255) NN
data LONGBLOB
expire INT(11) NN
created INT(11) NN
serialized SMALLINT(6) NN
tags LONGTEXT
checksum_invalidations INT(11) NN
checksum_deletions INT(11) NN
Indexes
cache_toolbar
cid VARCHAR(255) NN
data LONGBLOB
expire INT(11) NN
created INT(11) NN
serialized SMALLINT(6) NN
tags LONGTEXT
checksum_invalidations INT(11) NN
checksum_deletions INT(11) NN
Indexes
node_access
nid INT(10) UN NN
langcode VARCHAR(12) NN
fallback INT(10) UN NN
gid INT(10) UN NN
realm VARCHAR(255) NN
grant_view TINYINT(3) UN NN
grant_update TINYINT(3) UN NN
grant_delete TINYINT(3) UN NN
Indexes
search_total
word VARCHAR(50) NN
count FLOAT
Indexes
config_snapshot
name VARCHAR(255) NN
data LONGBLOB NN
Indexes
history
uid INT(11) NN
nid INT(11) NN
timestamp INT(11) NN
Indexes
node__field_image
bundle VARCHAR(128) NN
deleted TINYINT(4) NN
entity_id INT(10) UN NN
revision_id INT(10) UN
langcode VARCHAR(32) NN
delta INT(10) UN NN
field_image_target_id INT(10) UN NN
field_image_alt VARCHAR(512)
field_image_title VARCHAR(1024)
field_image_width INT(10) UN
field_image_height INT(10) UN
Indexes
search_index
word VARCHAR(50) NN
sid INT(10) UN NN
langcode VARCHAR(12) NN
type VARCHAR(16) NN
score FLOAT
Indexes
search_dataset
sid INT(10) UN NN
langcode VARCHAR(12) NN
type VARCHAR(16) NN
data LONGTEXT NN
reindex INT(10) UN NN
Indexes
key_value_expire
collection VARCHAR(128) NN
name VARCHAR(128) NN
value LONGBLOB NN
expire INT(11) NN
Indexes
key_value
collection VARCHAR(128) NN
name VARCHAR(128) NN
value LONGBLOB NN
Indexes
cache_block
cid VARCHAR(255) NN
data LONGBLOB
expire INT(11) NN
created INT(11) NN
serialized SMALLINT(6) NN
tags LONGTEXT
checksum_invalidations INT(11) NN
checksum_deletions INT(11) NN
Indexes
cache_tags
tag VARCHAR(255) NN
invalidations INT(11) NN
deletions INT(11) NN
Indexes

http://www.flickr.com/photos/crazyneighborlady/355232758/


http://gallery.nen.gov.uk/image82582-.html

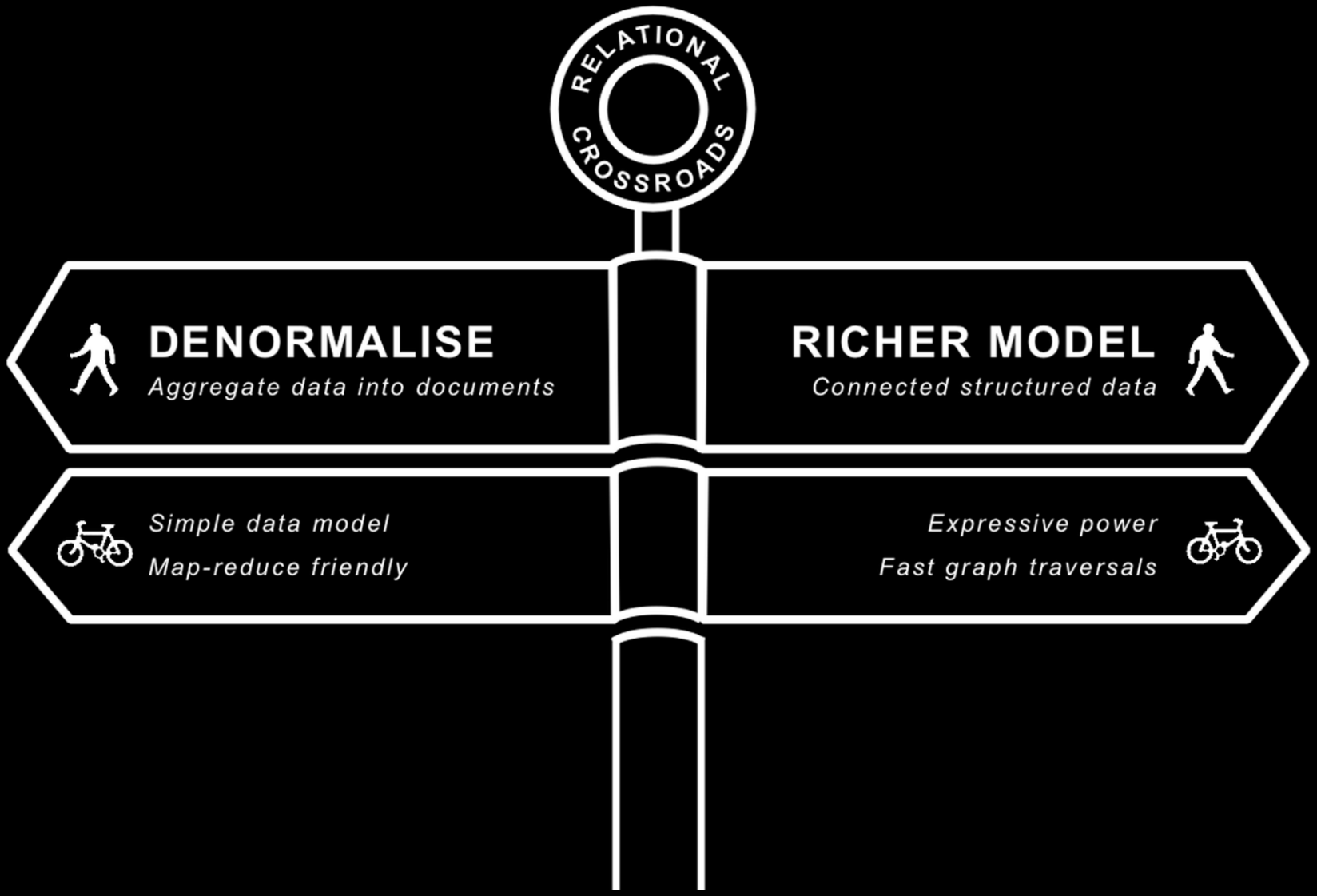
Aggregate-Oriented Data

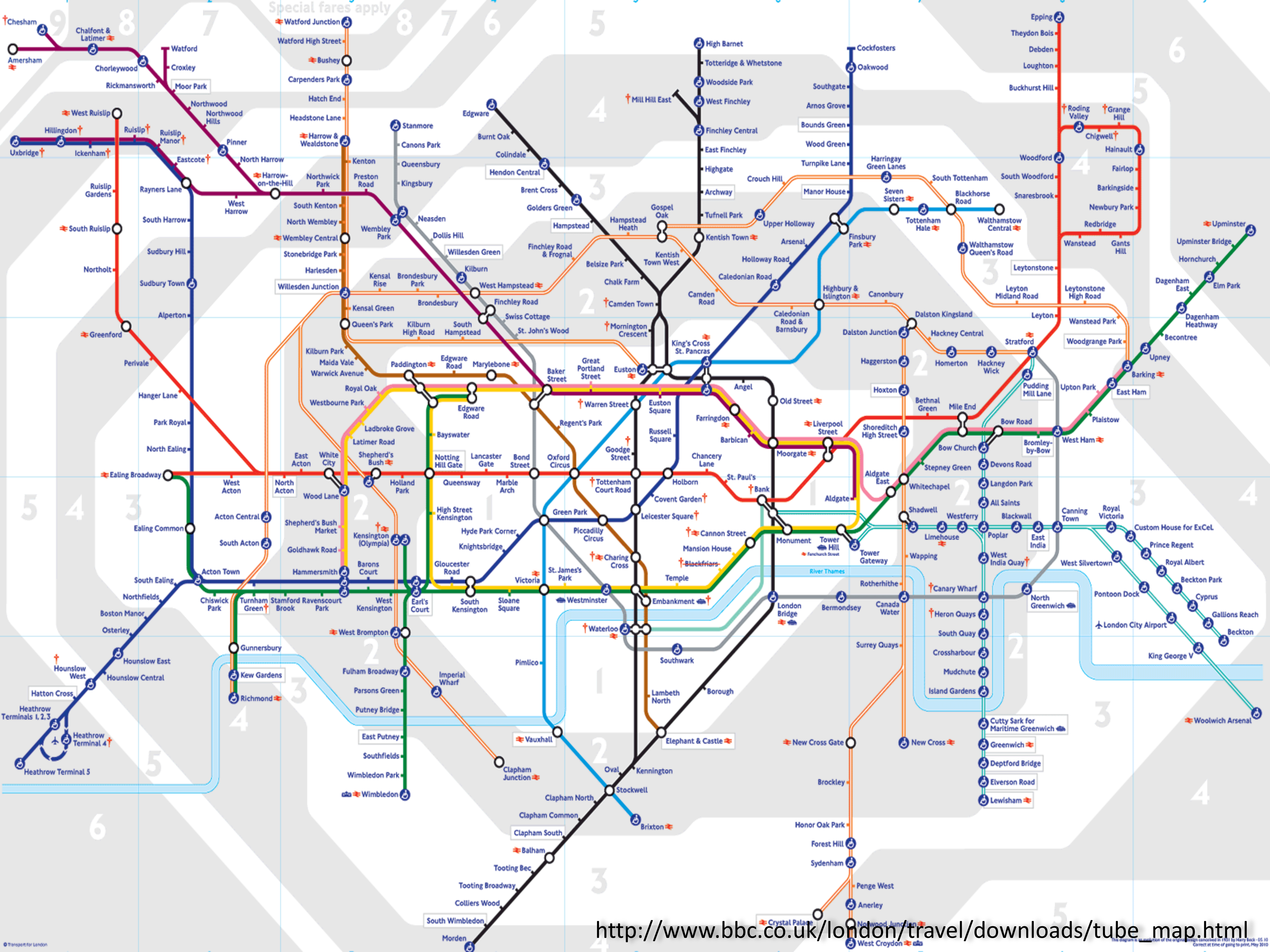
http://martinfowler.com/bliki/AggregateOrientedDatabase.html

"There is a significant downside - the whole approach works really well when data access is aligned with the aggregates, but what if you want to look at the data in a different way? Order entry naturally stores orders as aggregates, but analyzing product sales cuts across the aggregate structure. The advantage of not using an aggregate structure in the database is that it allows you to slice and dice your data different ways for different audiences.

This is why aggregate-oriented stores talk so much about map-reduce."

$$complexity = f(size, connectedness, uniformity)$$


RELATIONAL
CROSSROADS
DENORMALISE
Aggregate data into documents
RICHER MODEL
Connected structured data
Simple data model
Map-reduce friendly
Expressive power
Fast graph traversals



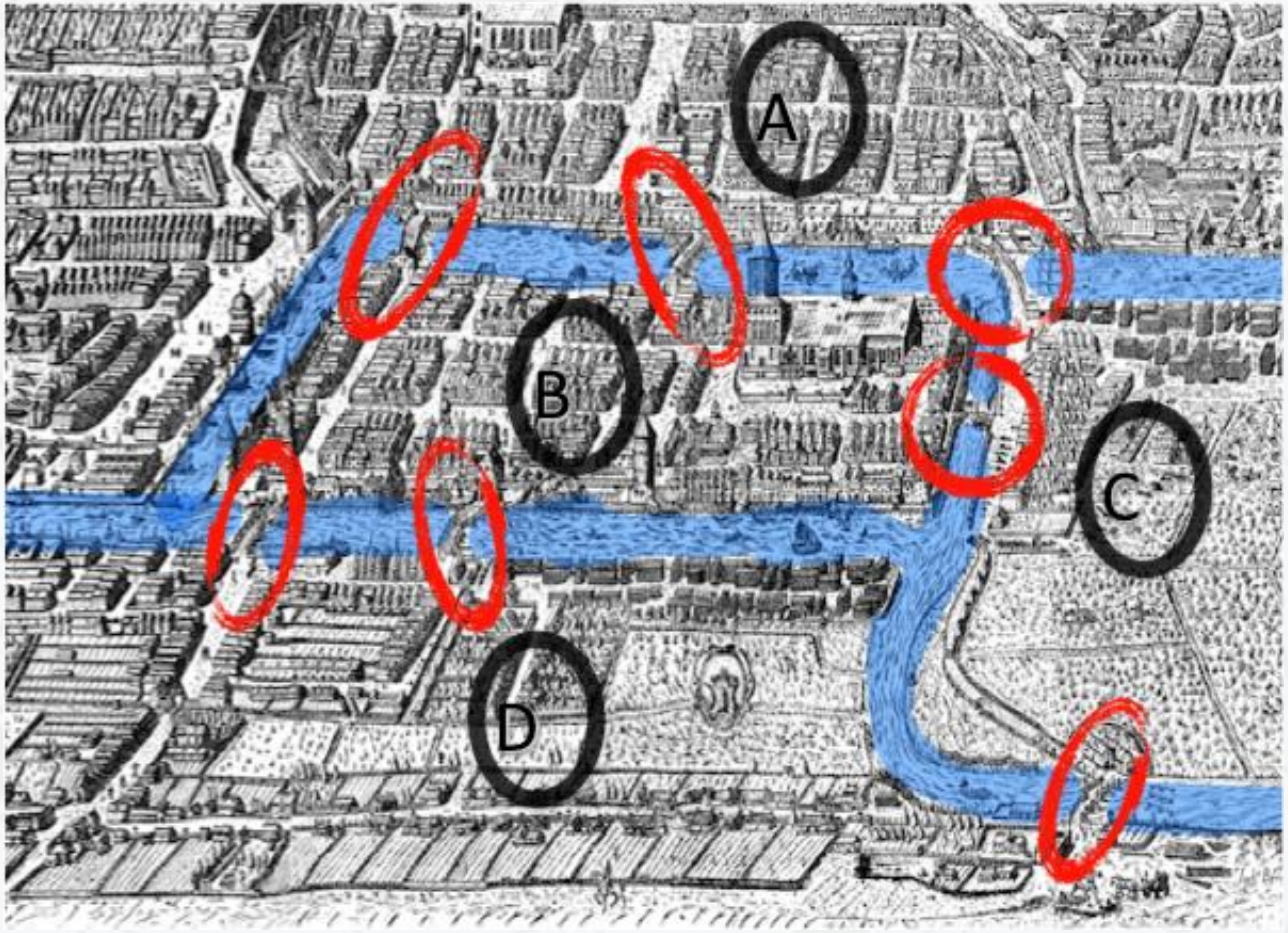
http://www.bbc.co.uk/london/travel/downloads/tube_map.html

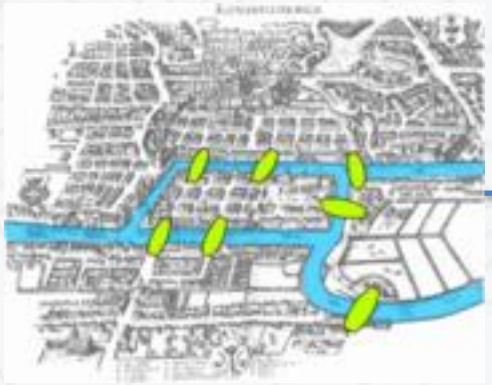
Meet Leonhard Euler

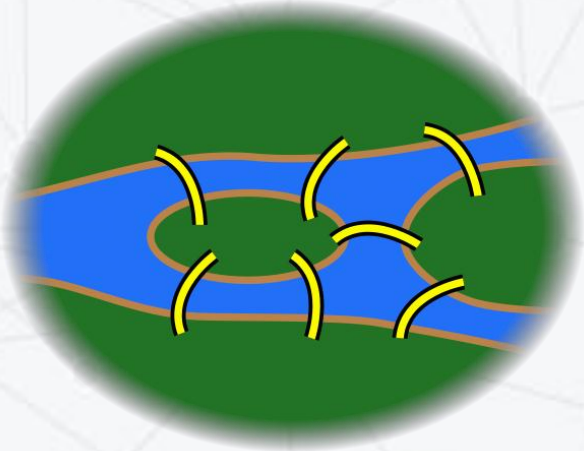
- Swiss mathematician
- Inventor of Graph Theory (1736)

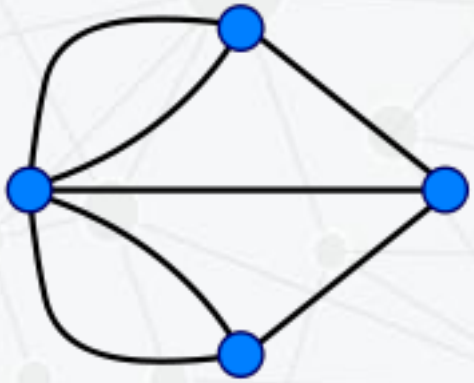
http://en.wikipedia.org/wiki/File:Leonhard_Euler_2.jpg

Königsberg (Prussia) - 1736





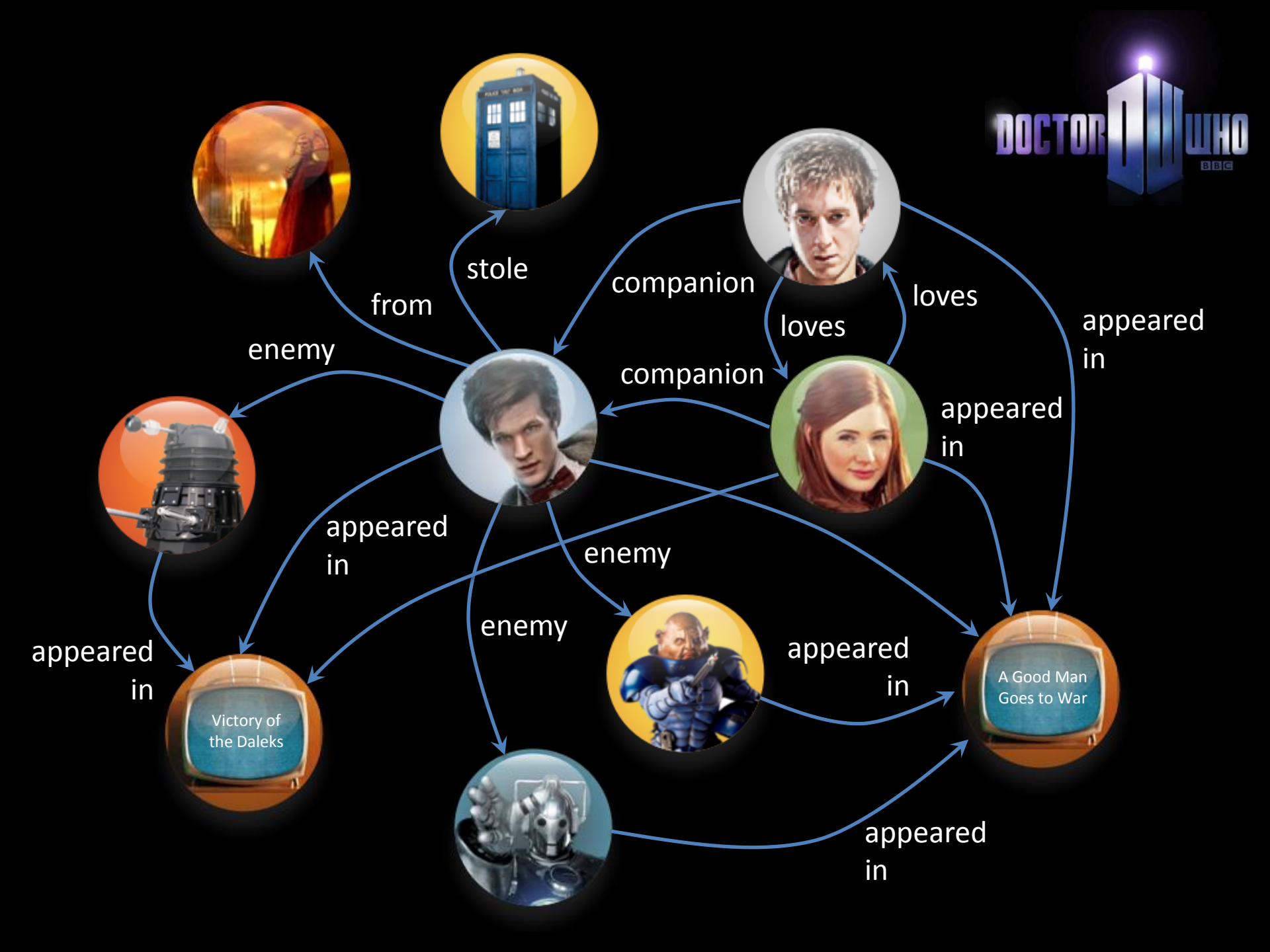


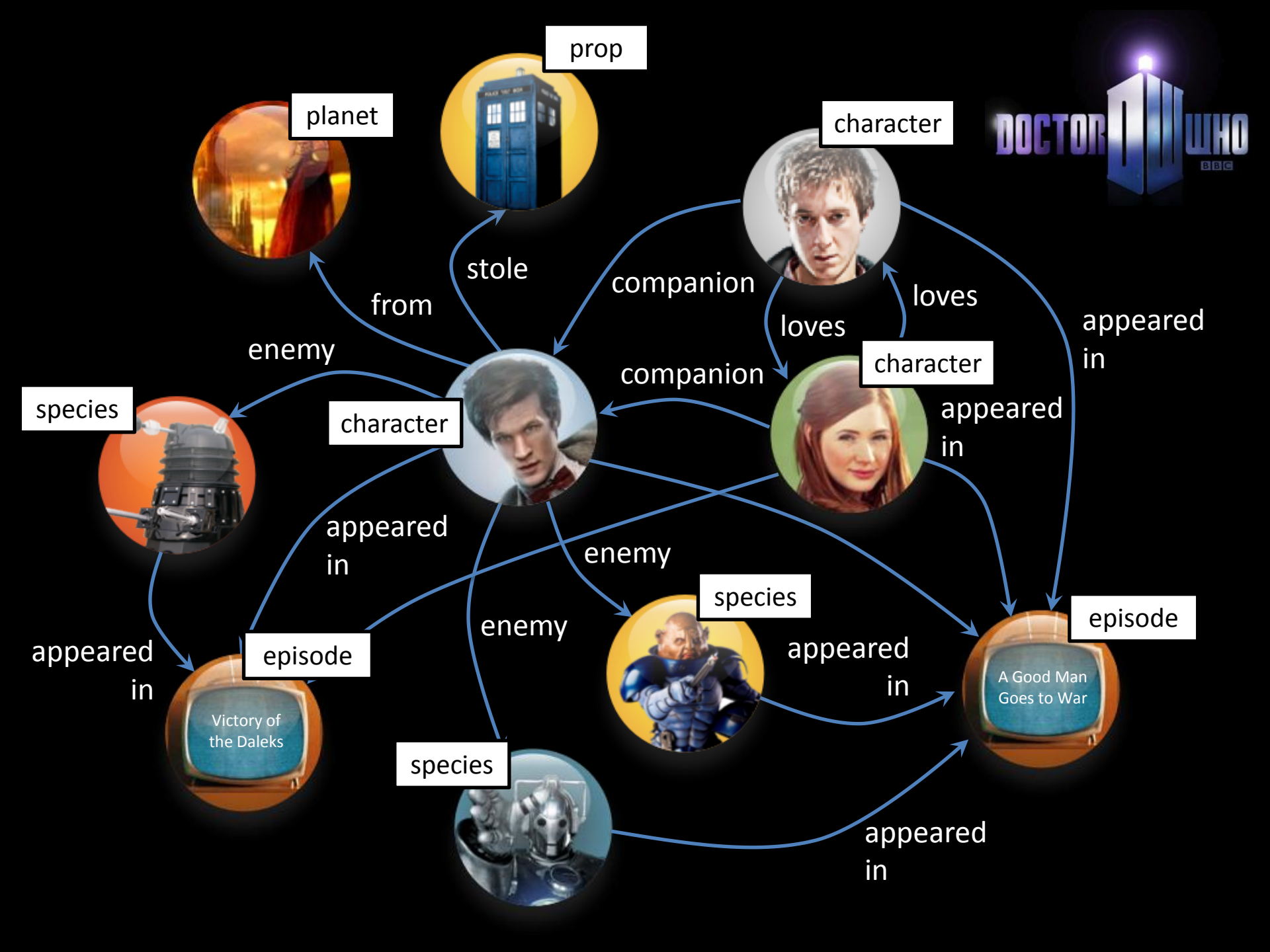


http://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

Labeled Property graph model

- Nodes with optional properties and optional labels
- Named, directed relationships with optional properties
  - Relationships have exactly one start and end node
  - Which may be the same node

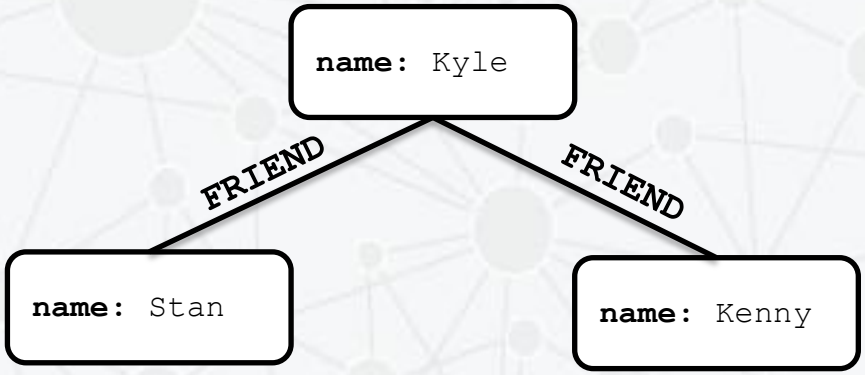

DOCTOR WHO
BBC
stole
from
companion
loves
loves
appeared in
enemy
companion
appeared in
appeared in
enemy
enemy
appeared in
appeared in
Victory of the Daleks
A Good Man Goes to War
appeared in


prop
planet
character
DOCTOR WHO
BBC
stole
from
companion
loves
loves
appeared
in
enemy
species
character
companion
character
appeared
in
appeared
in
enemy
species
episode
enemy
appeared
in
episode
appeared
in
Victory of
the Daleks
A Good Man
Goes to War
species
appeared
in

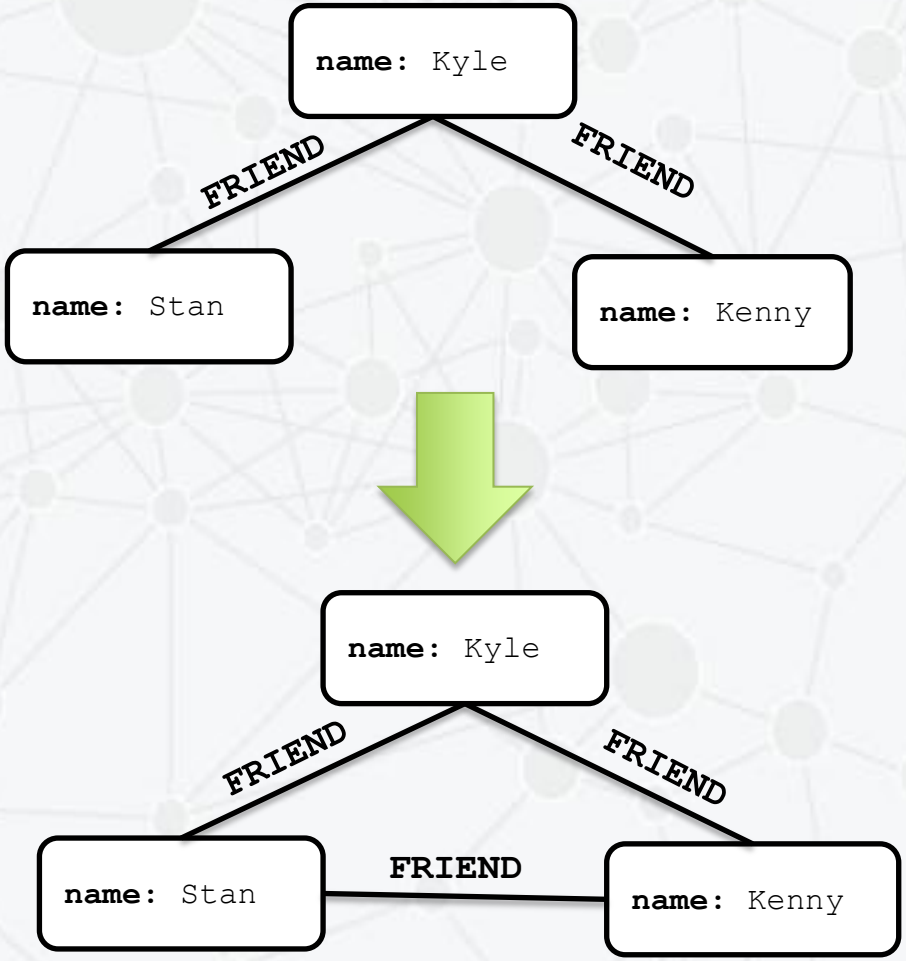
http://blogs.adobe.com/digitalmarketing/analytics/predictive-analytics/predictive-analytics-and-the-digital-marketer/


DULWICH SUPERMARKET & OFF LICENCE
ENGLISH , TURKISH , GREEK , MEDITERRANEAN FOOD
18
DELICATESSEN & ORGANICS
TEL : 020 8299 2214
FRESHLY CUT SANDWICHES · BILTONG
BILTONG
BOEREWORS

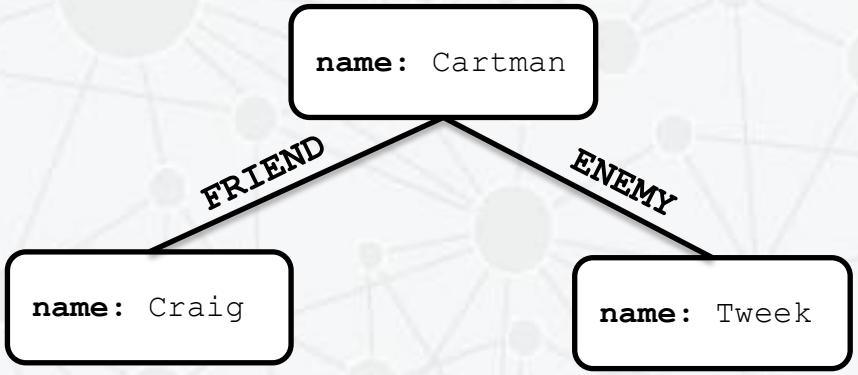
Triadic Closure



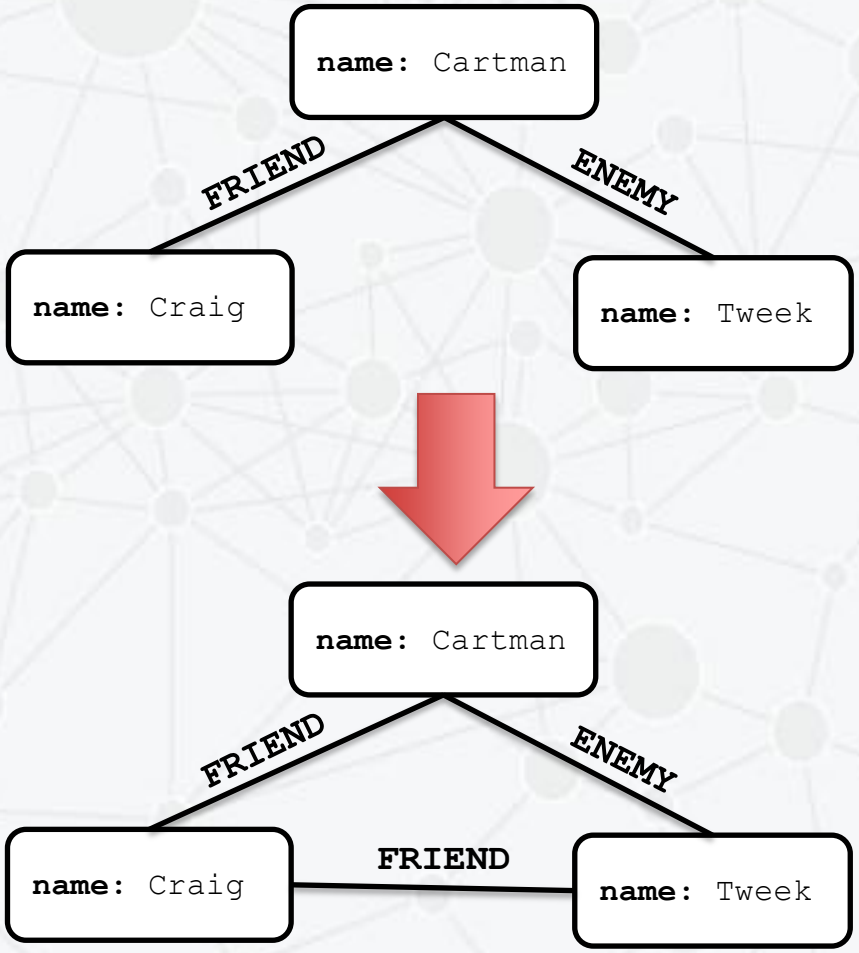
Triadic Closure



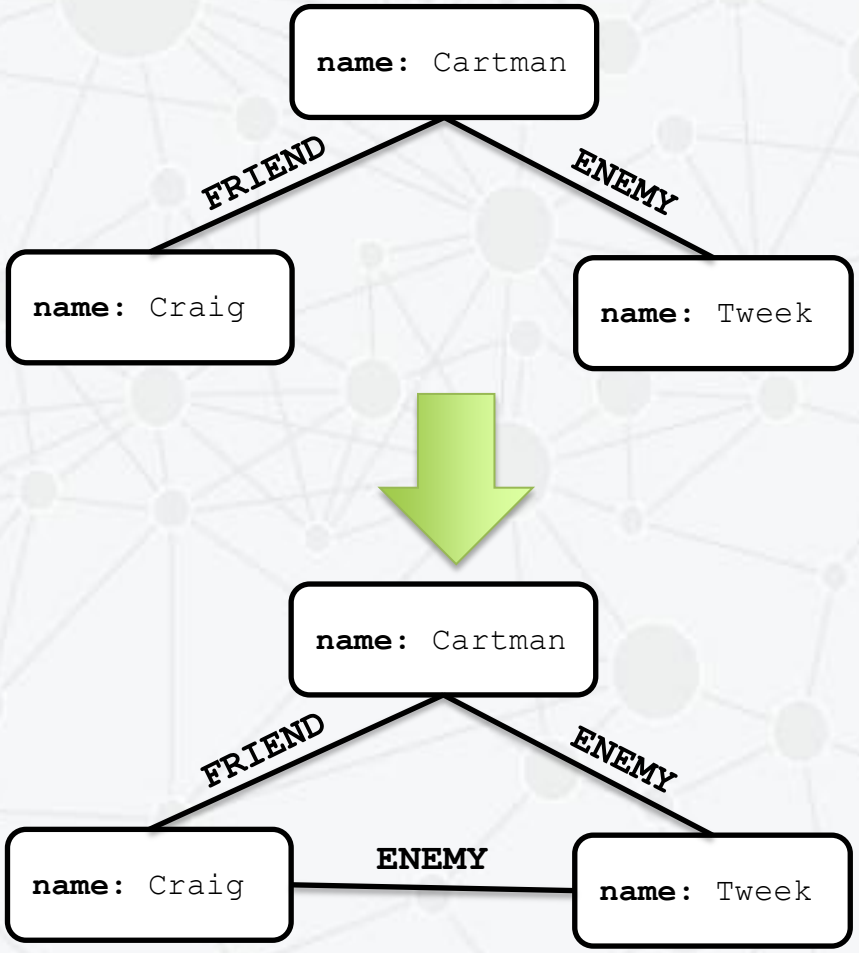
Structural Balance



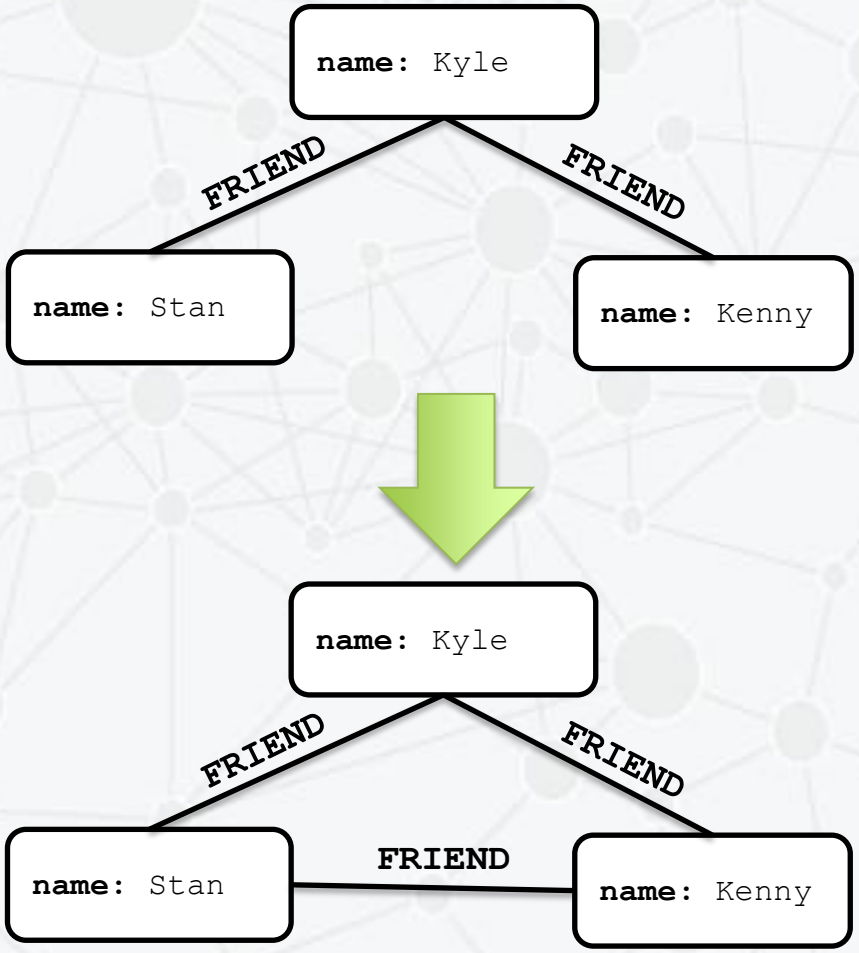
Structural Balance



Structural Balance



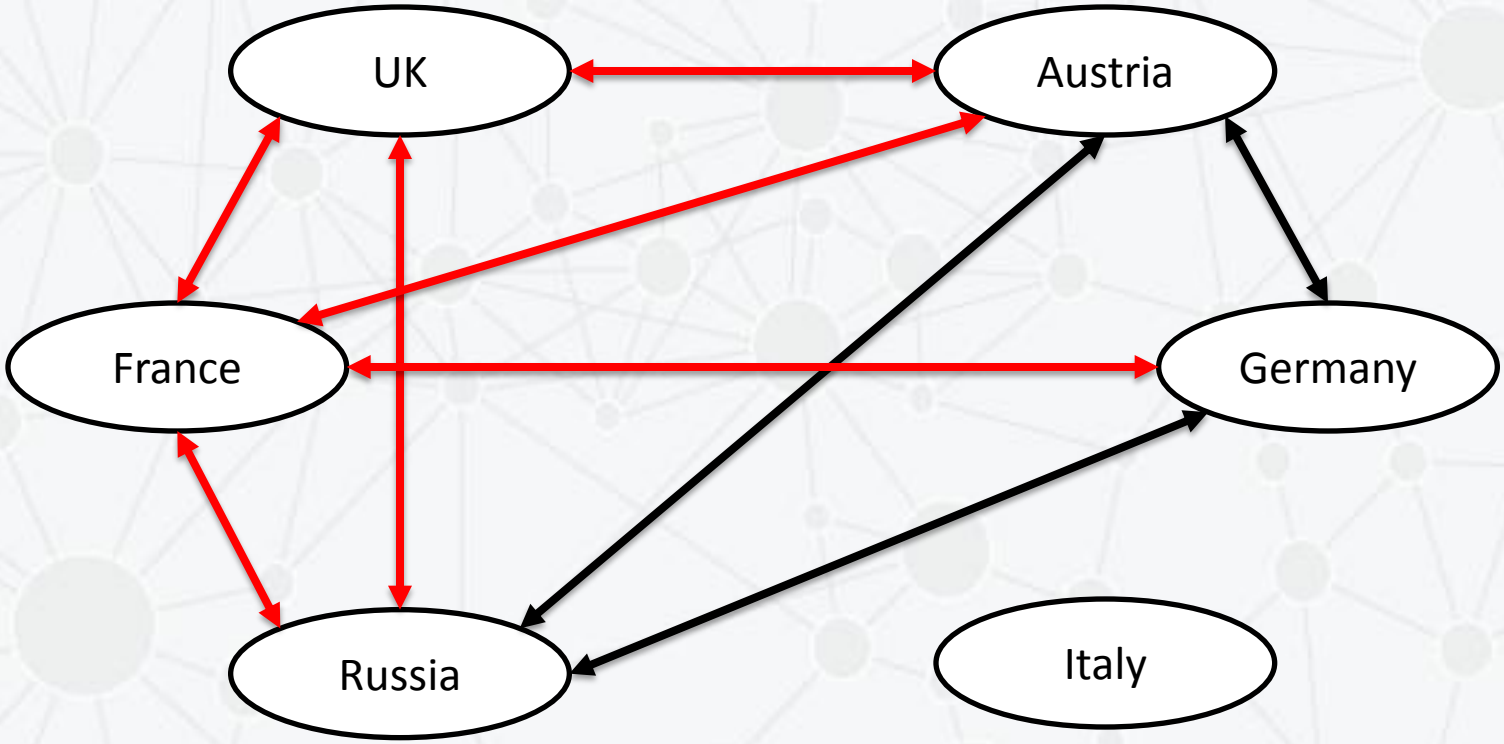
Structural Balance



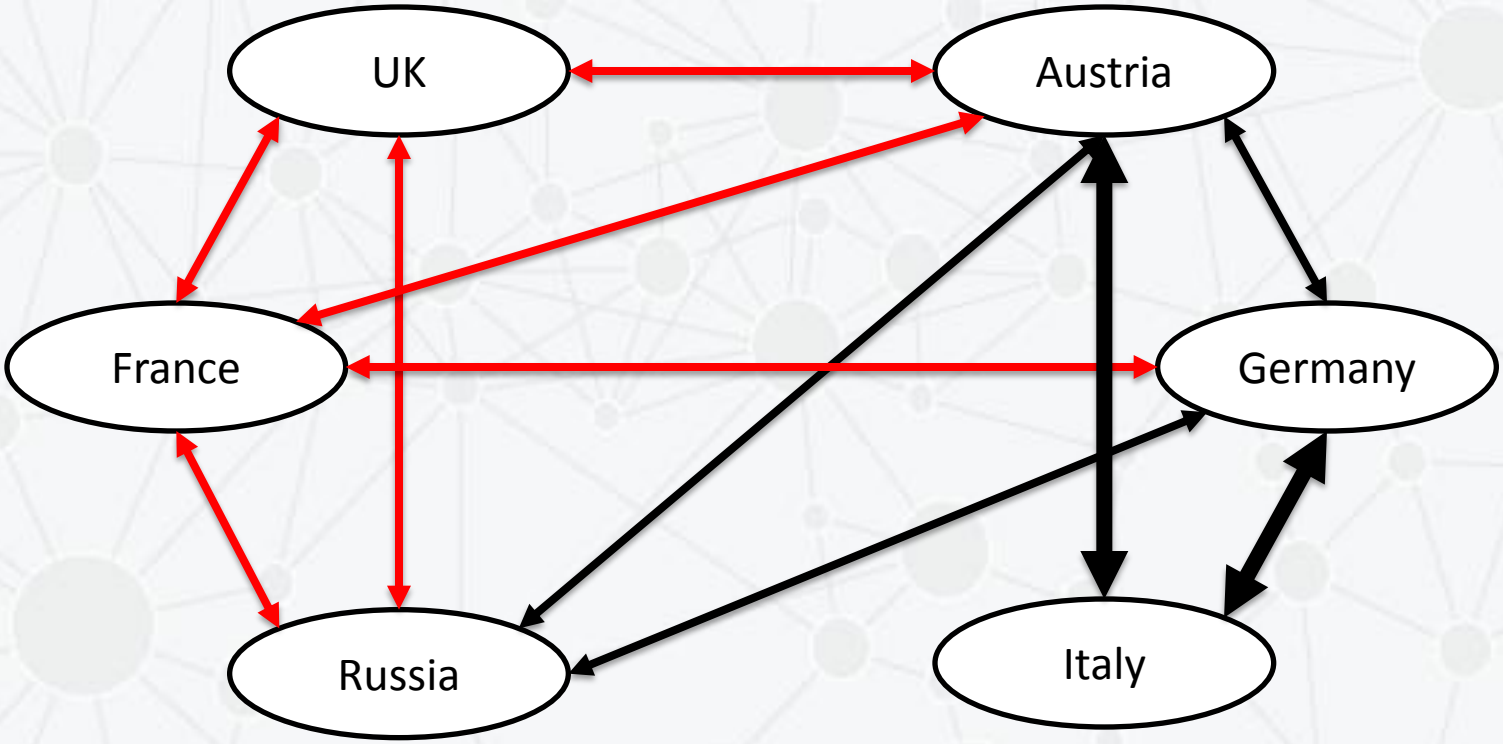
Structural Balance is a *key* predictive technique

And it’s domain-agnostic

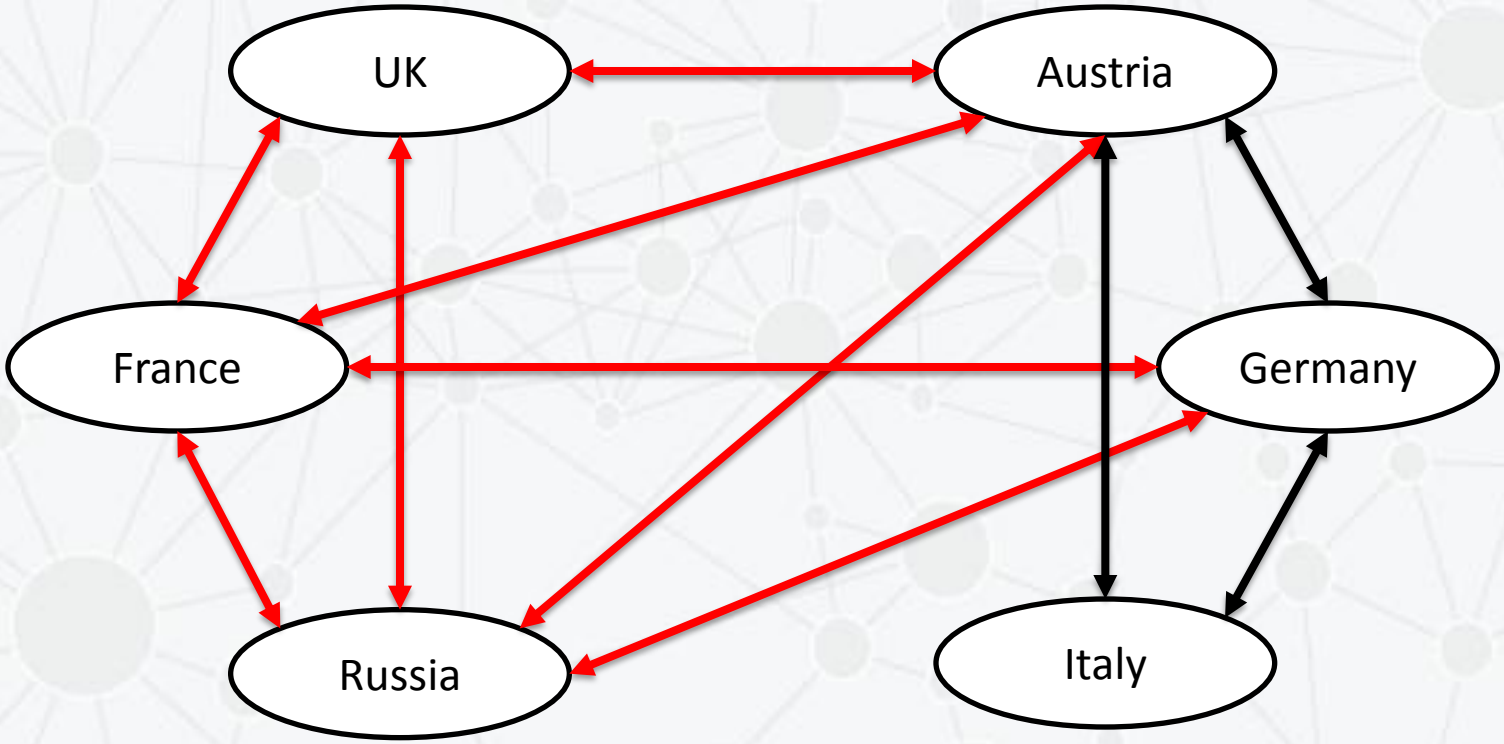
Allies and Enemies



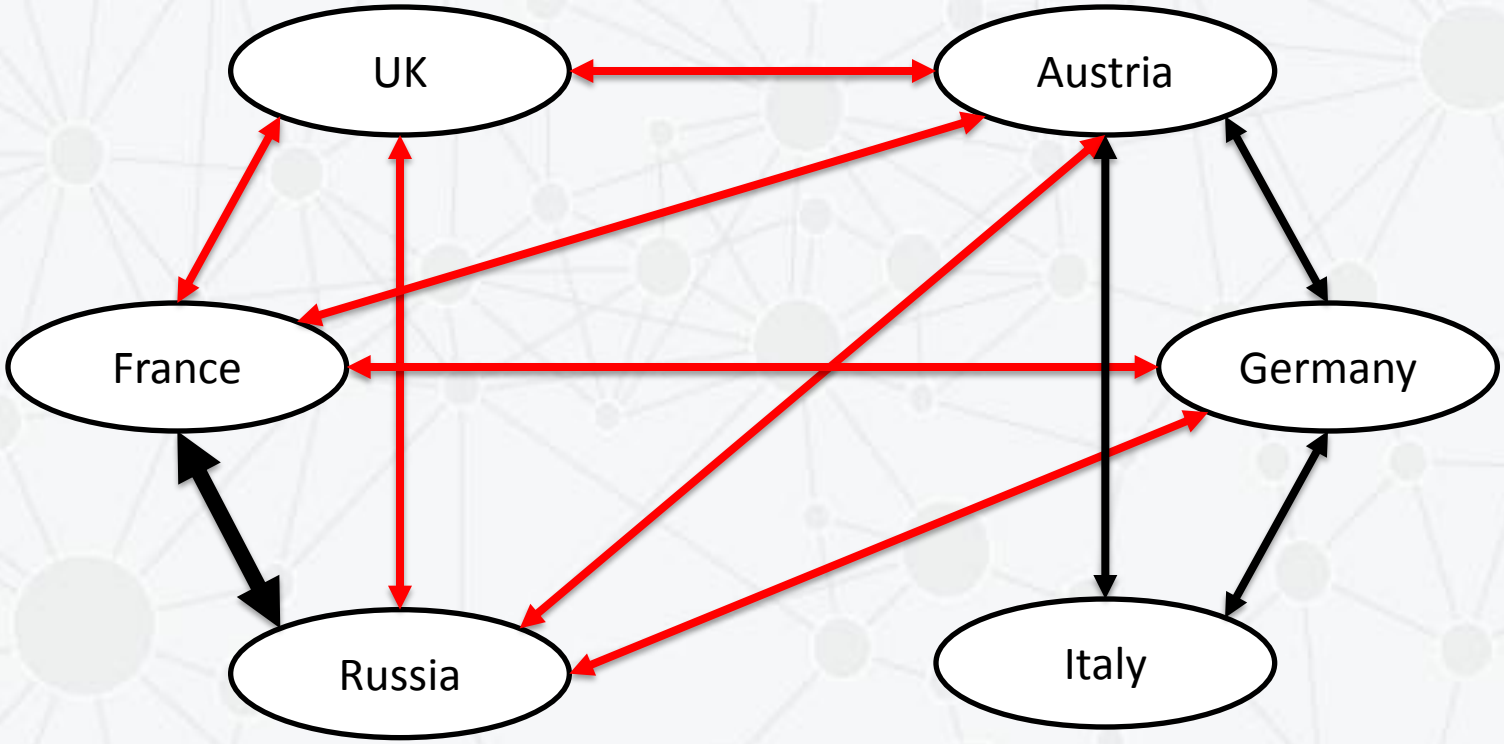
Allies and Enemies



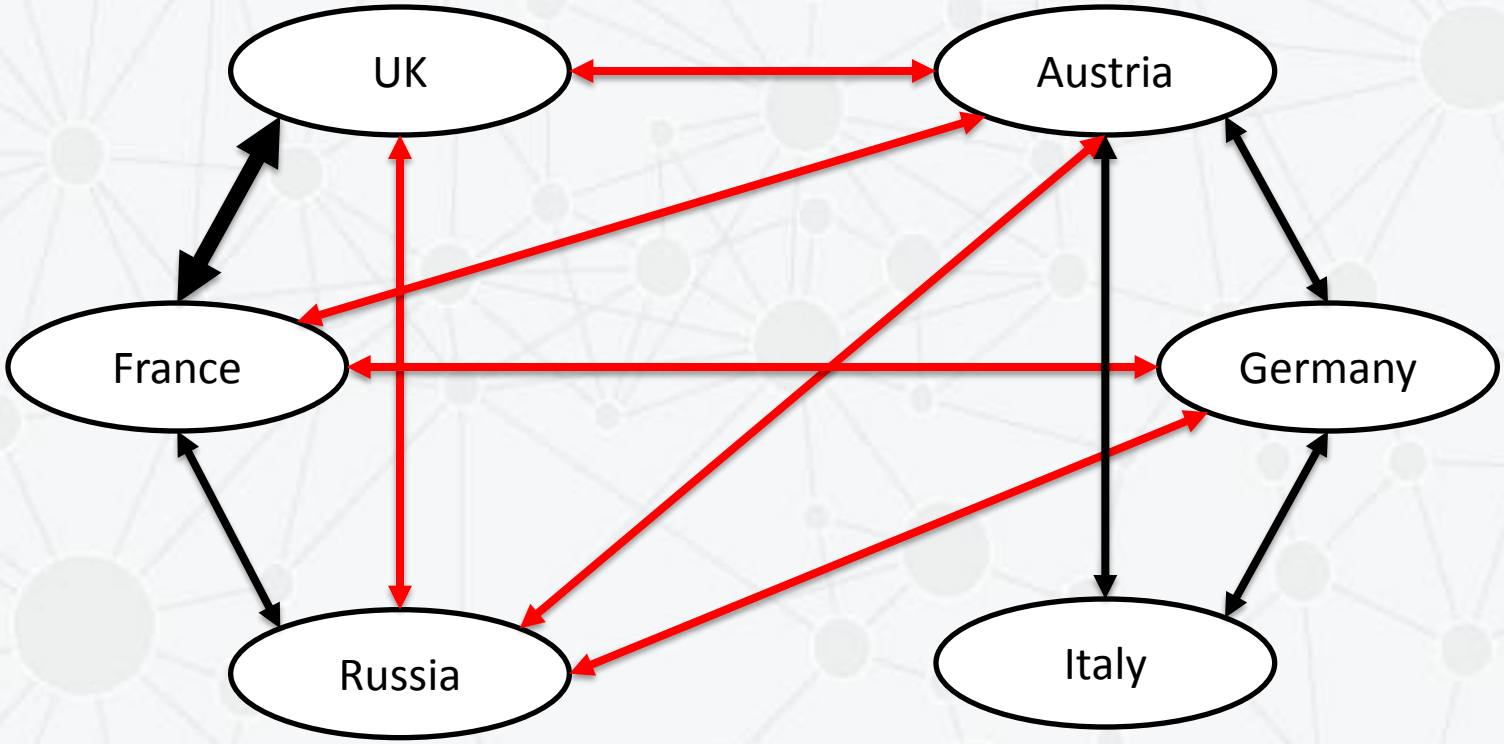
Allies and Enemies



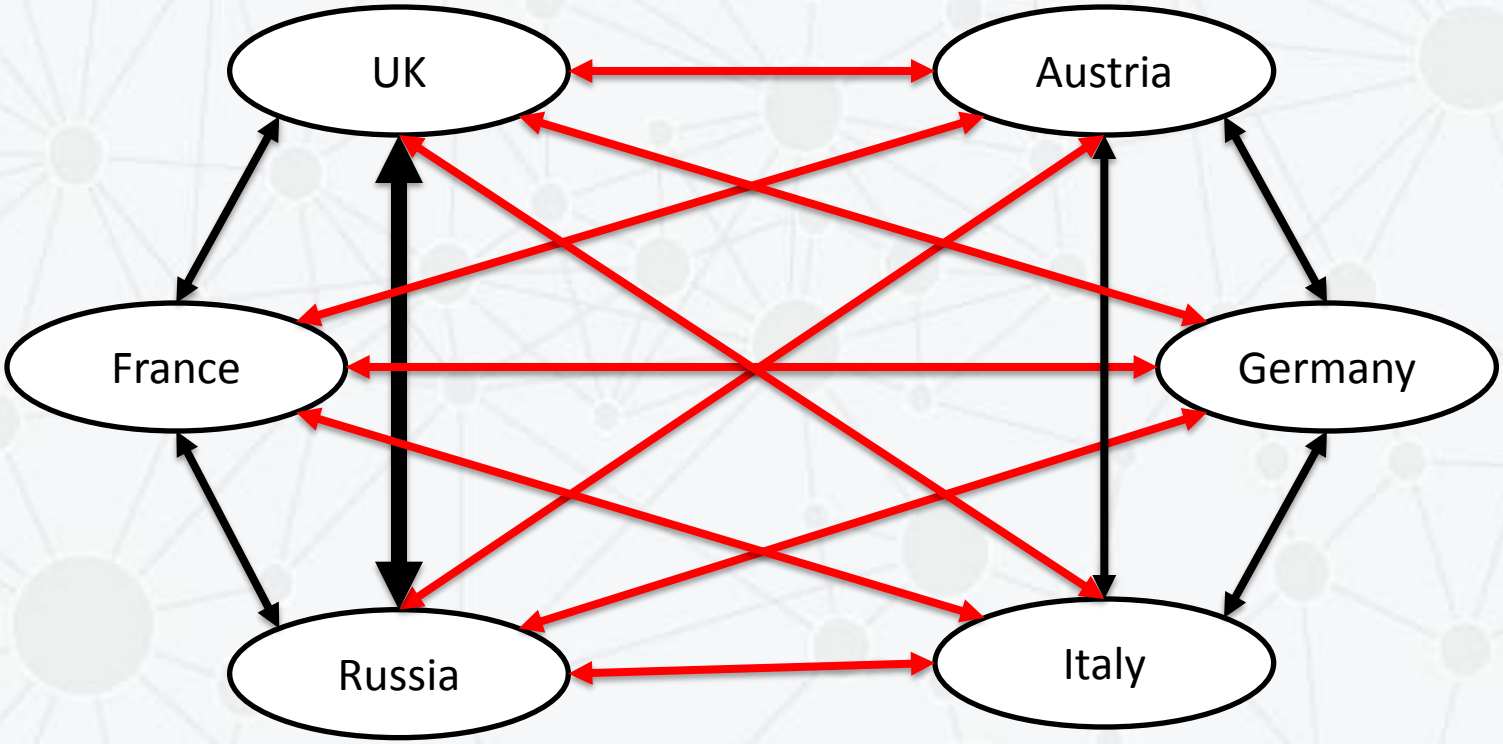
Allies and Enemies



Allies and Enemies

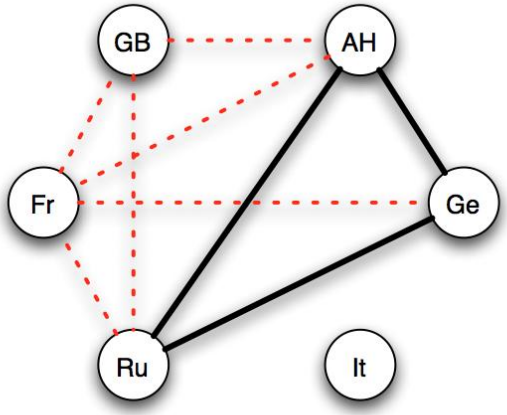


Allies and Enemies

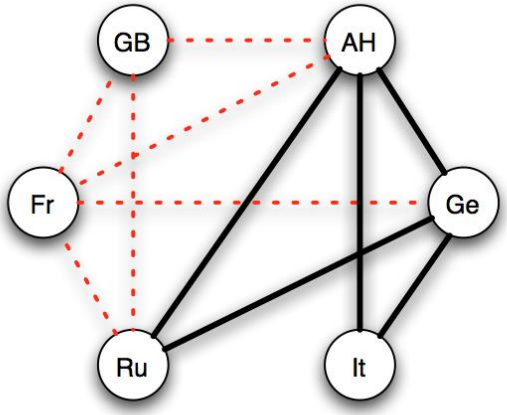


Predicting WWI

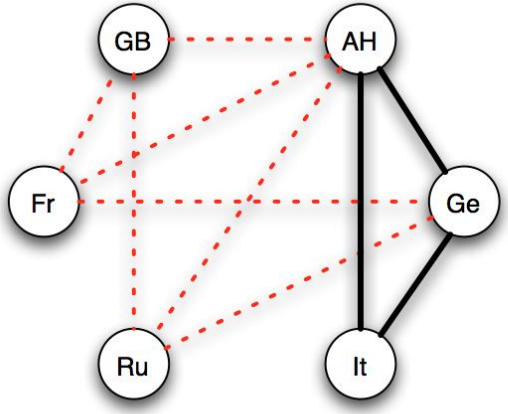
[Easley and Kleinberg]



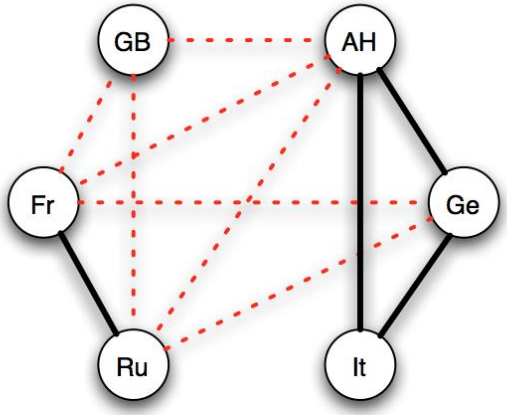
(a) *Three Emperors' League 1872–81*



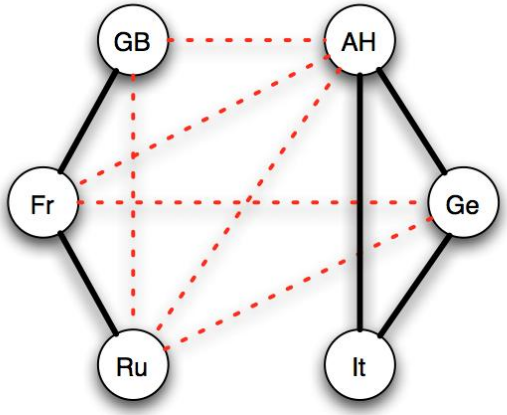
(b) *Triple Alliance 1882*



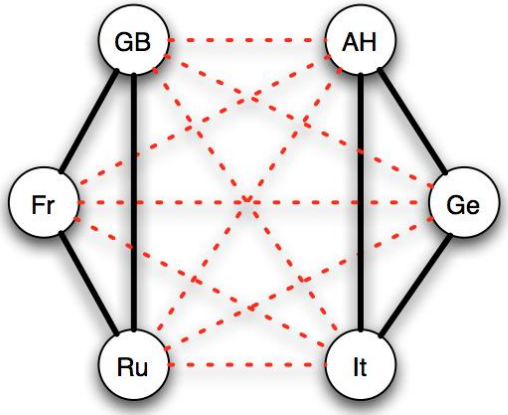
(c) *German-Russian Lapse 1890*



(d) *French-Russian Alliance 1891–94*



(e) *Entente Cordiale 1904*



(f) *British Russian Alliance 1907*

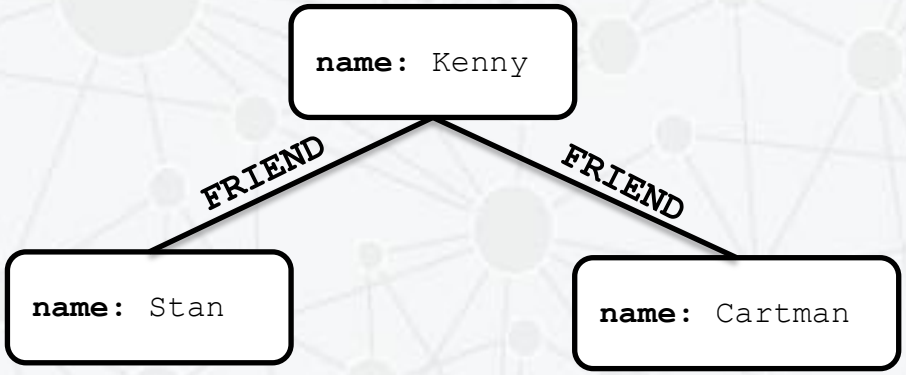
Strong Triadic Closure

It if a node has strong relationships to two neighbours, then these neighbours must have at least a weak relationship between them.

[Wikipedia]

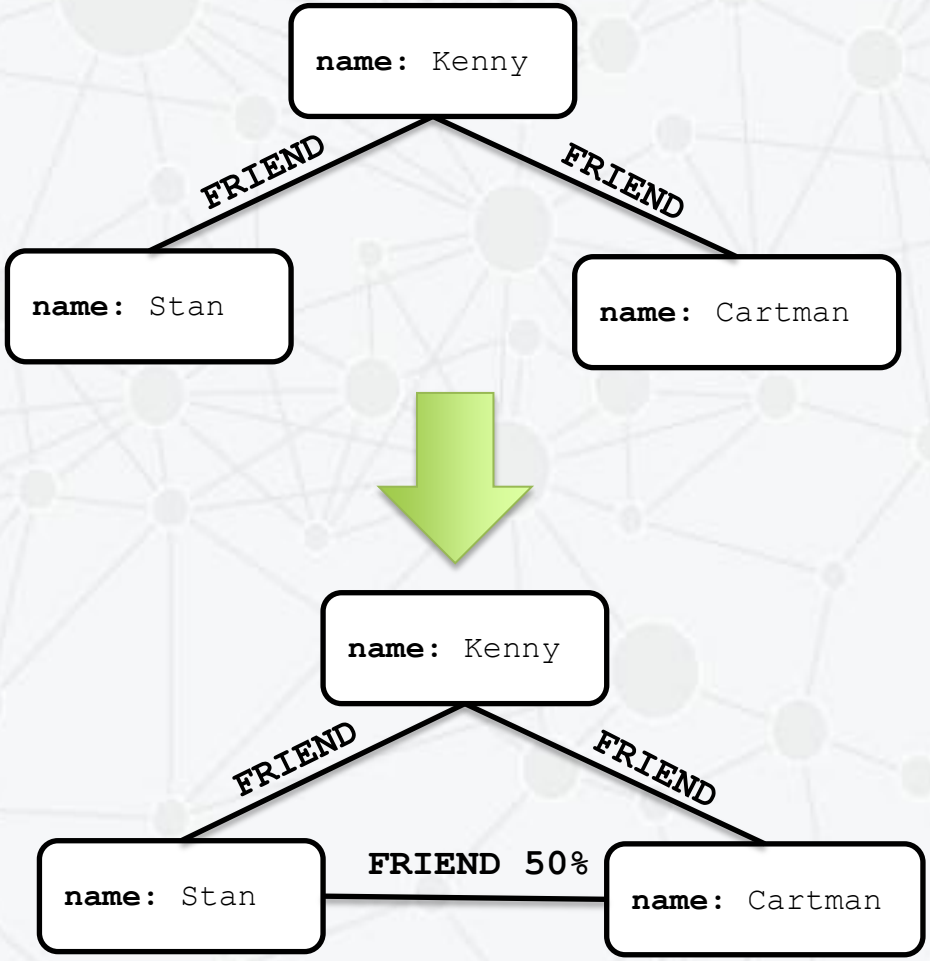
Triadic Closure

(weak relationship)



Triadic Closure

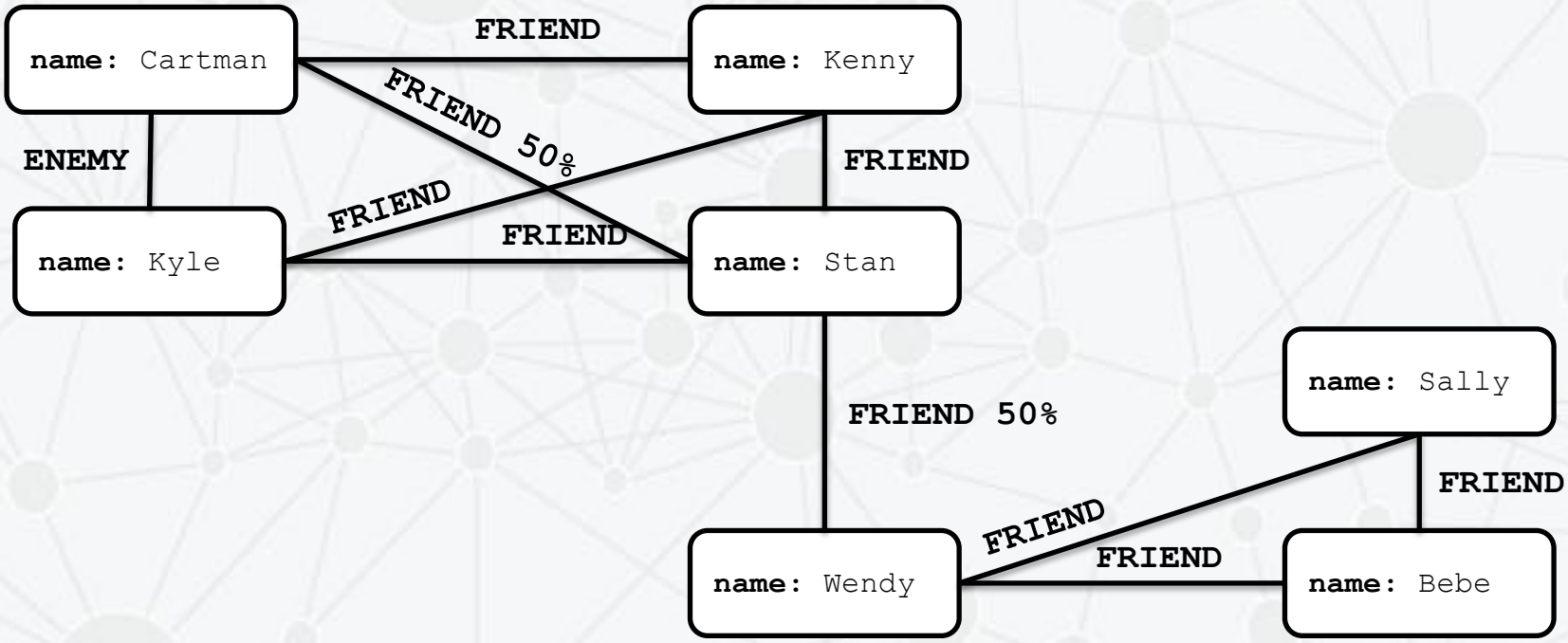
(weak relationship)



Weak relationships

- Relationships can have “strength” as well as intent
  - Think: weighting on a relationship in a property graph
- Weak links play another super-important structural role in graph theory
  - They bridge neighbourhoods

Local Bridges

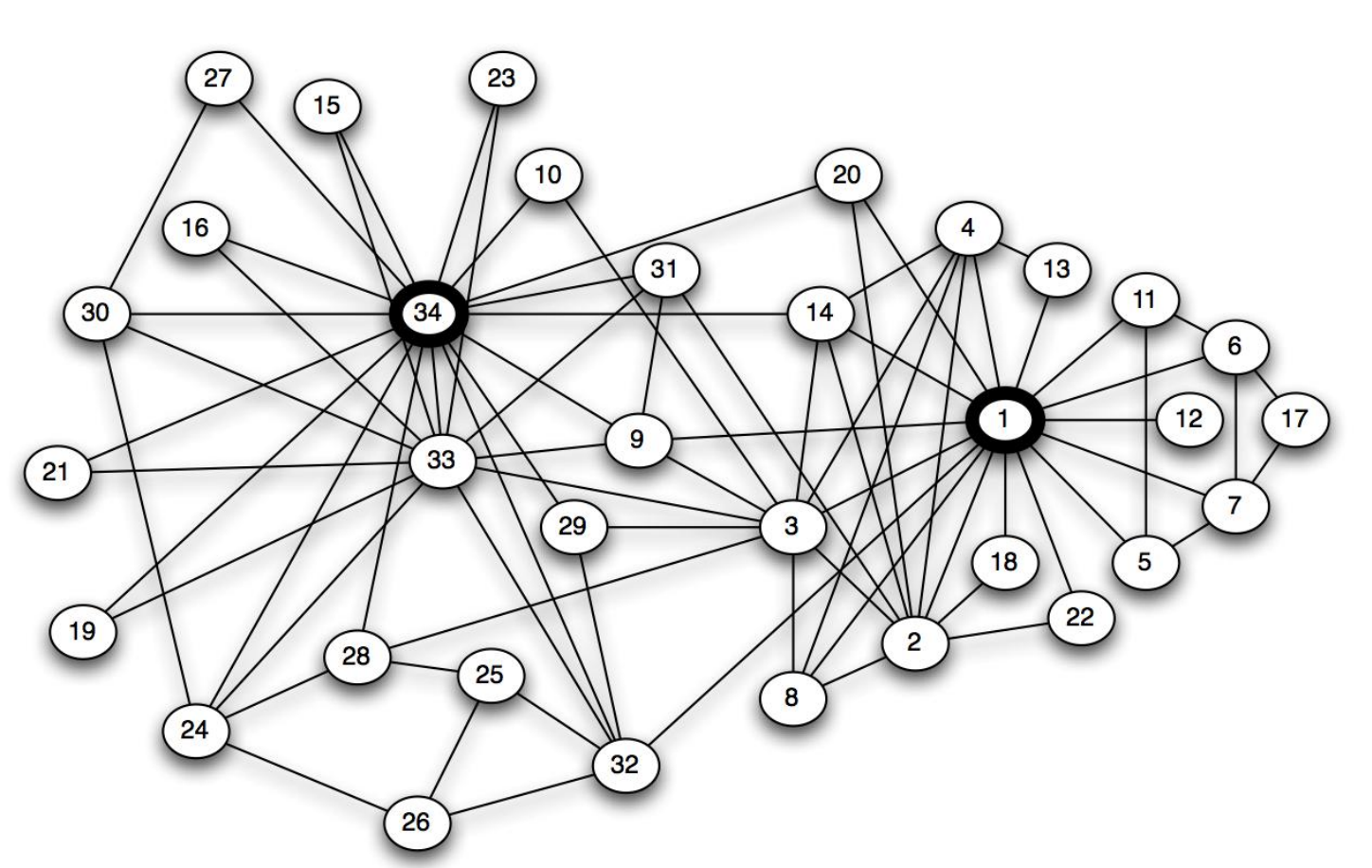


Local Bridge Property

*"If a node **A** in a network satisfies the Strong Triadic Closure Property and is involved in at least two strong relationships, then any local bridge it is involved in must be a weak relationship."*

[Easley and Kleinberg]

University Karate Club

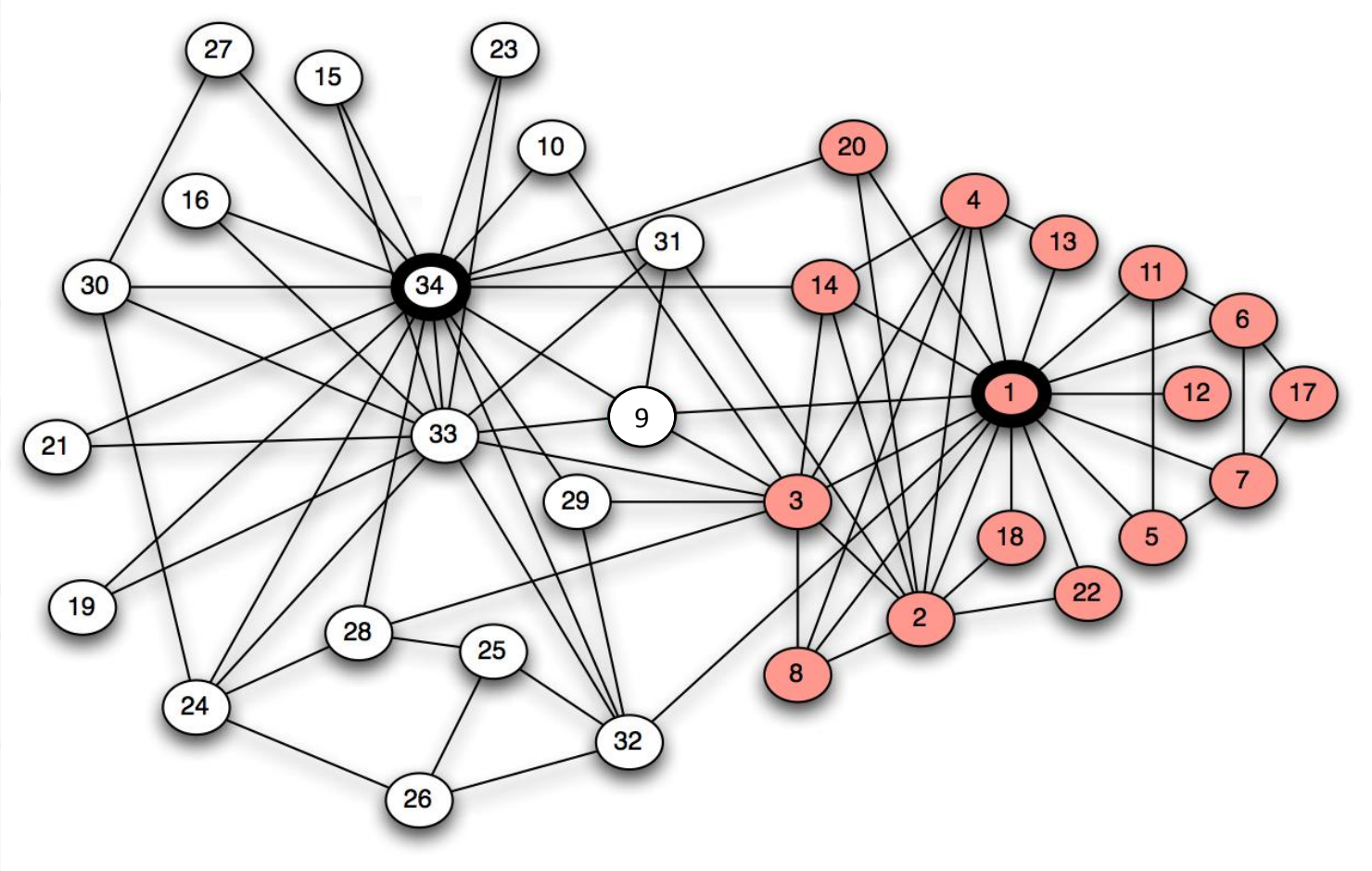


Graph Partitioning

- (NP) Hard problem
  - Recursively remove the spanning links between dense regions
  - Or recursively merge nodes into ever larger “subgraph” nodes
  - Choose your algorithm carefully – some are better than others for a given domain
- Can use to (almost exactly) predict the break up of the karate club!

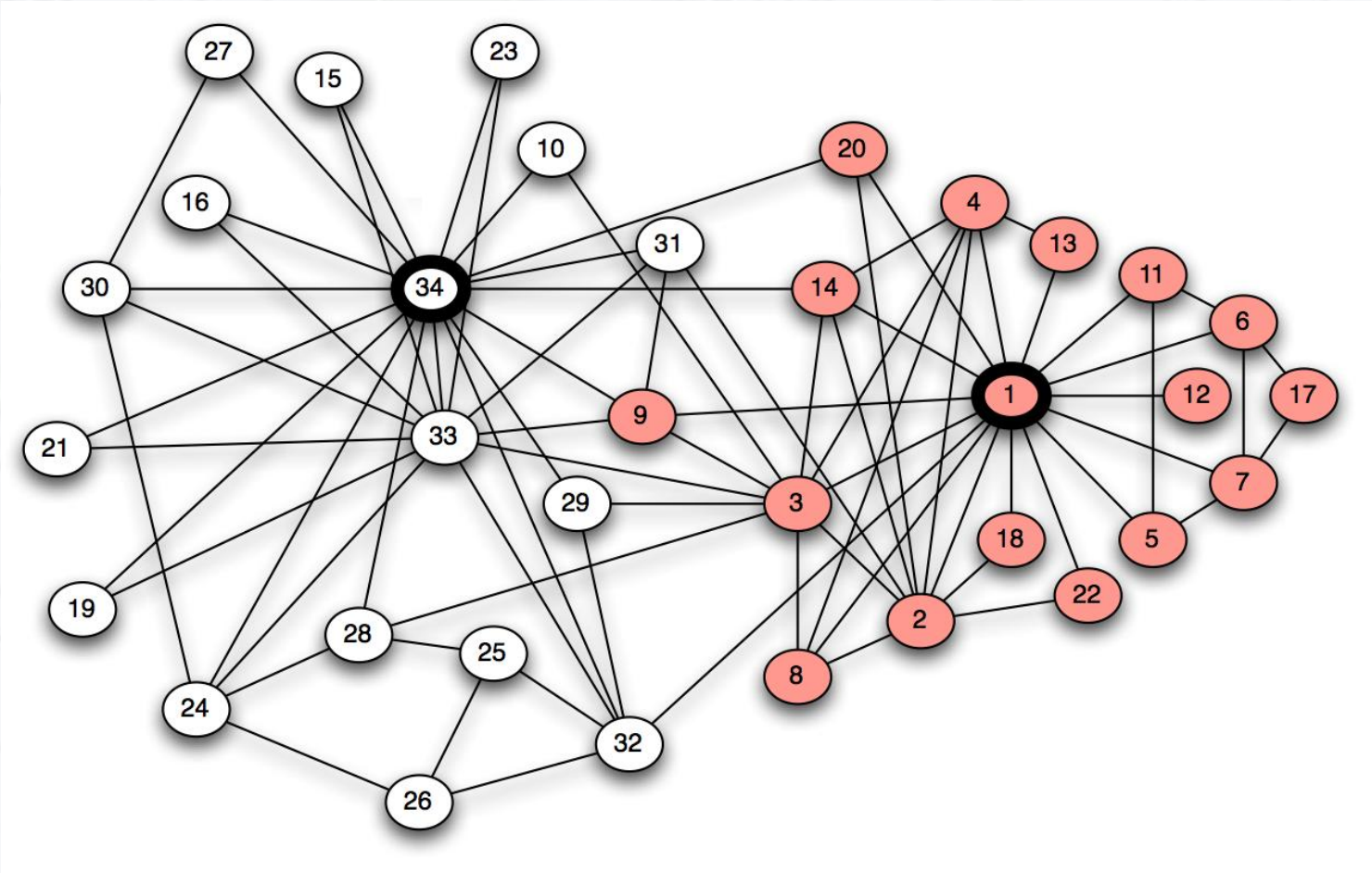
University Karate Clubs

(predicted by Graph Theory)



University Karate Clubs

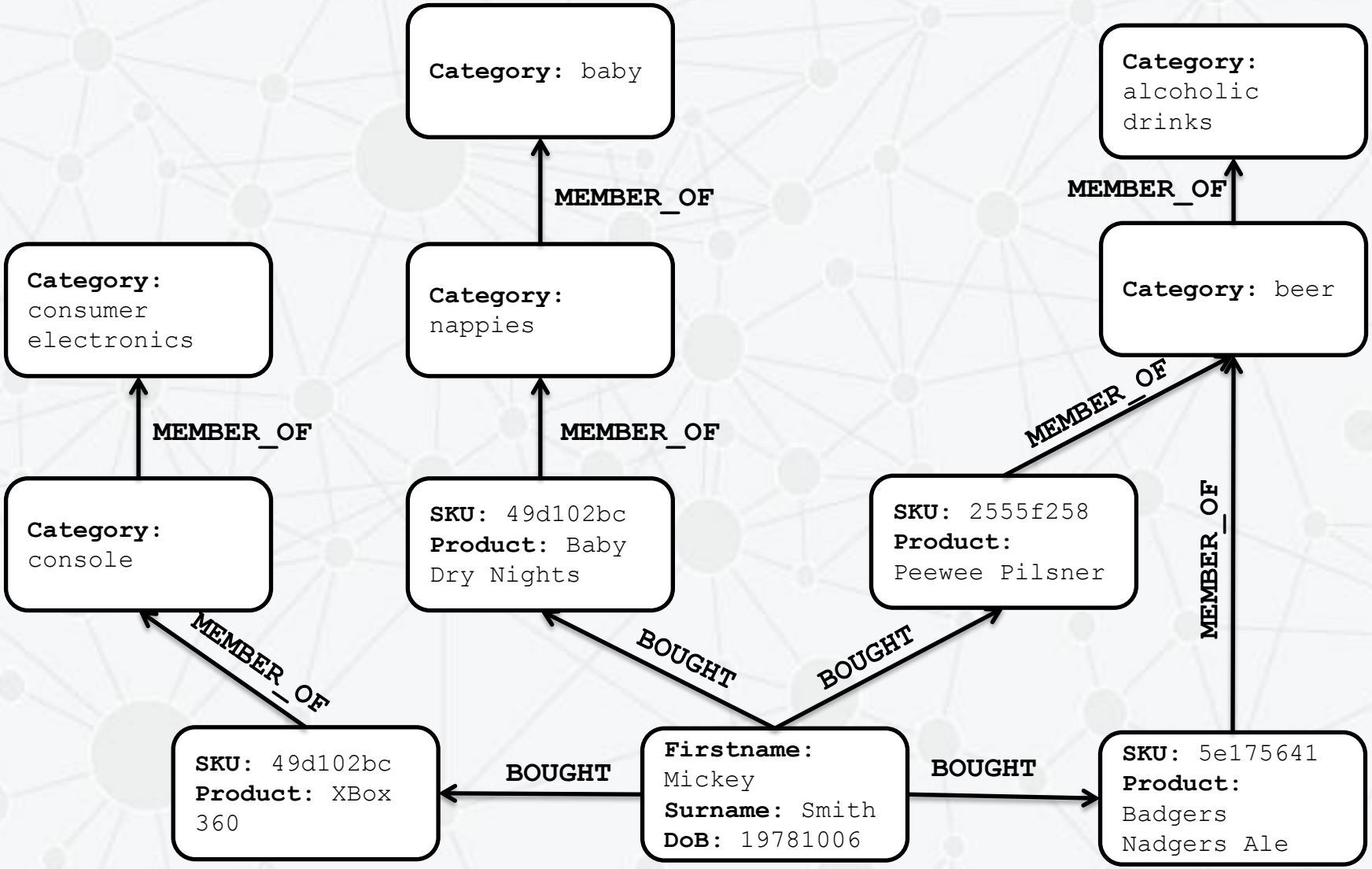
(what actually happened!)




john
crane
Ltd

Cypher

- Declarative graph pattern matching language
  - "SQL for graphs"
  - A humane tool pioneered by a tamed SQL DBA
- A pattern graph matching language
  - Find me stuff like…

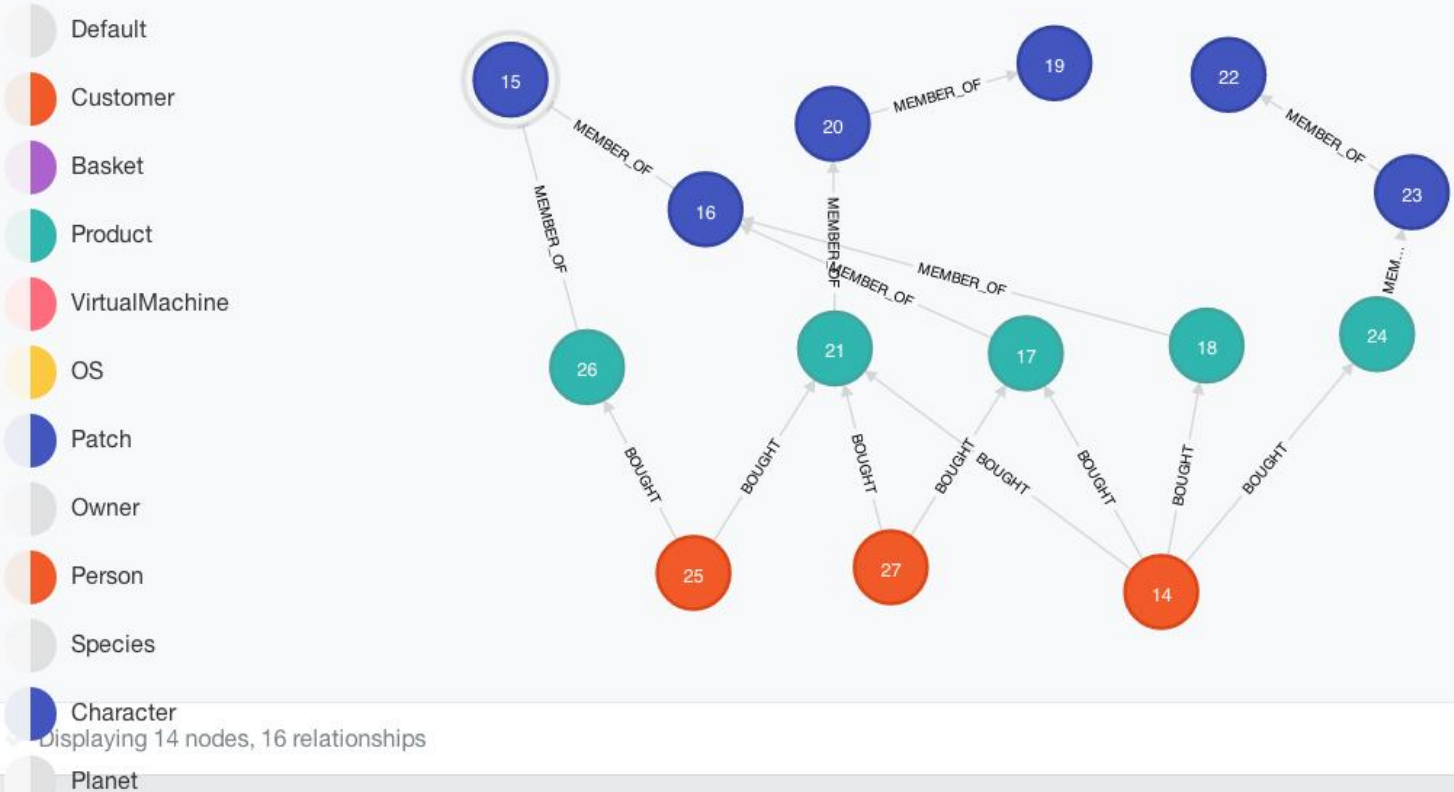

Category: baby
MEMBER_OF
Category: alcoholic drinks
MEMBER_OF
Category: consumer electronics
Category: nappies
Category: beer
MEMBER_OF
MEMBER_OF
MEMBER_OF
Category: console
SKU: 49d102bc Product: Baby Dry Nights
SKU: 2555f258 Product: Peewee Pilsner
MEMBER_OF
BOUGHT
BOUGHT
MEMBER_OF
SKU: 49d102bc Product: XBox 360
BOUGHT
Firstname: Mickey Surname: Smith DoB: 19781006
BOUGHT
SKU: 5e175641 Product: Badgers Nadgers Ale

Neo4j data/graph.db

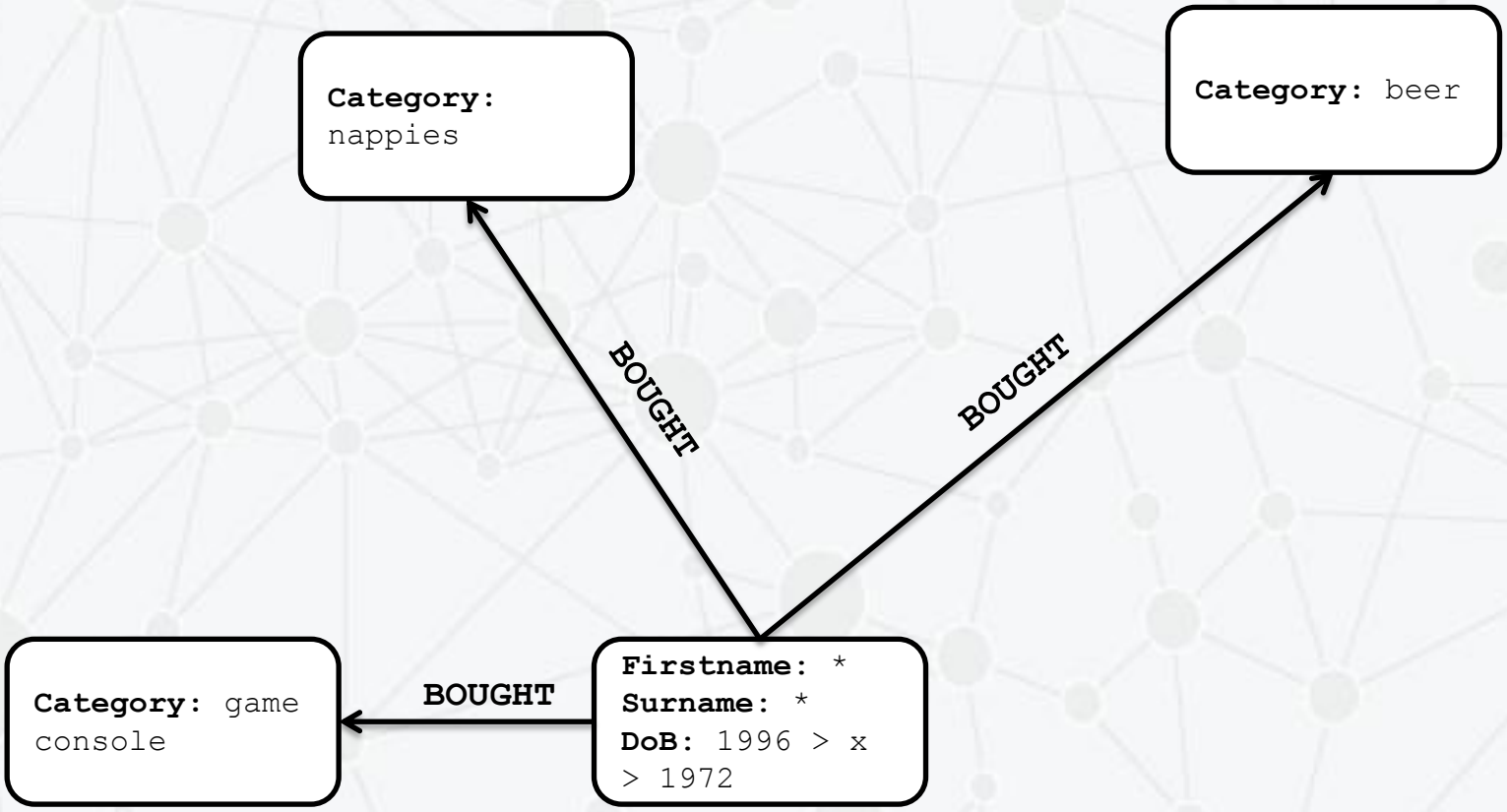
localhost:7474/browser/

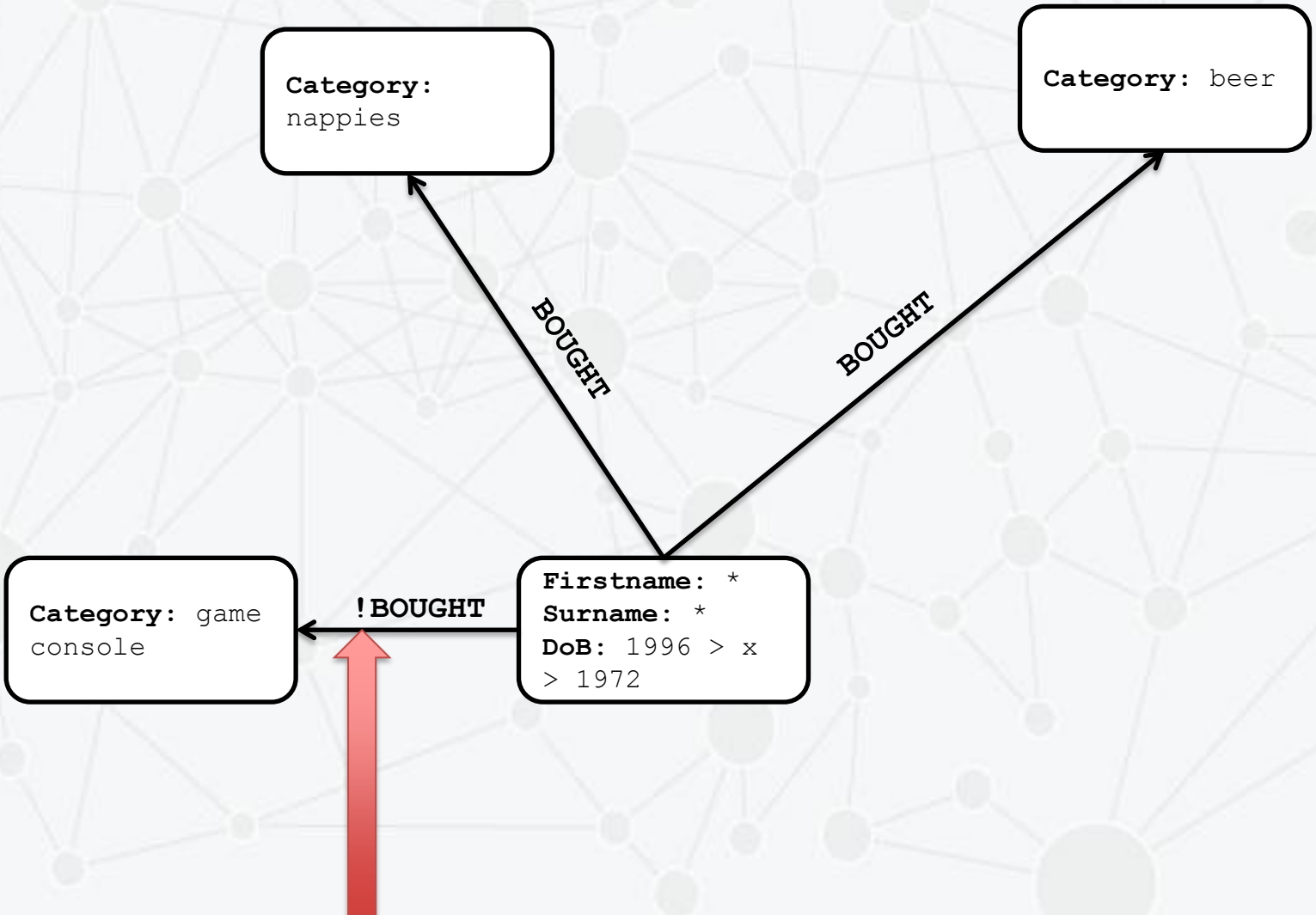
```
1 CREATE (daddy1:Person { name: 'Mickey Smith', dob: 19781006 })
2
3 CREATE (alcohol:Category { category : 'alcoholic drinks'})
4 CREATE (beer:Category { category : 'beer'})
5 CREATE beer-[:MEMBER_OF]->alcohol
6
```

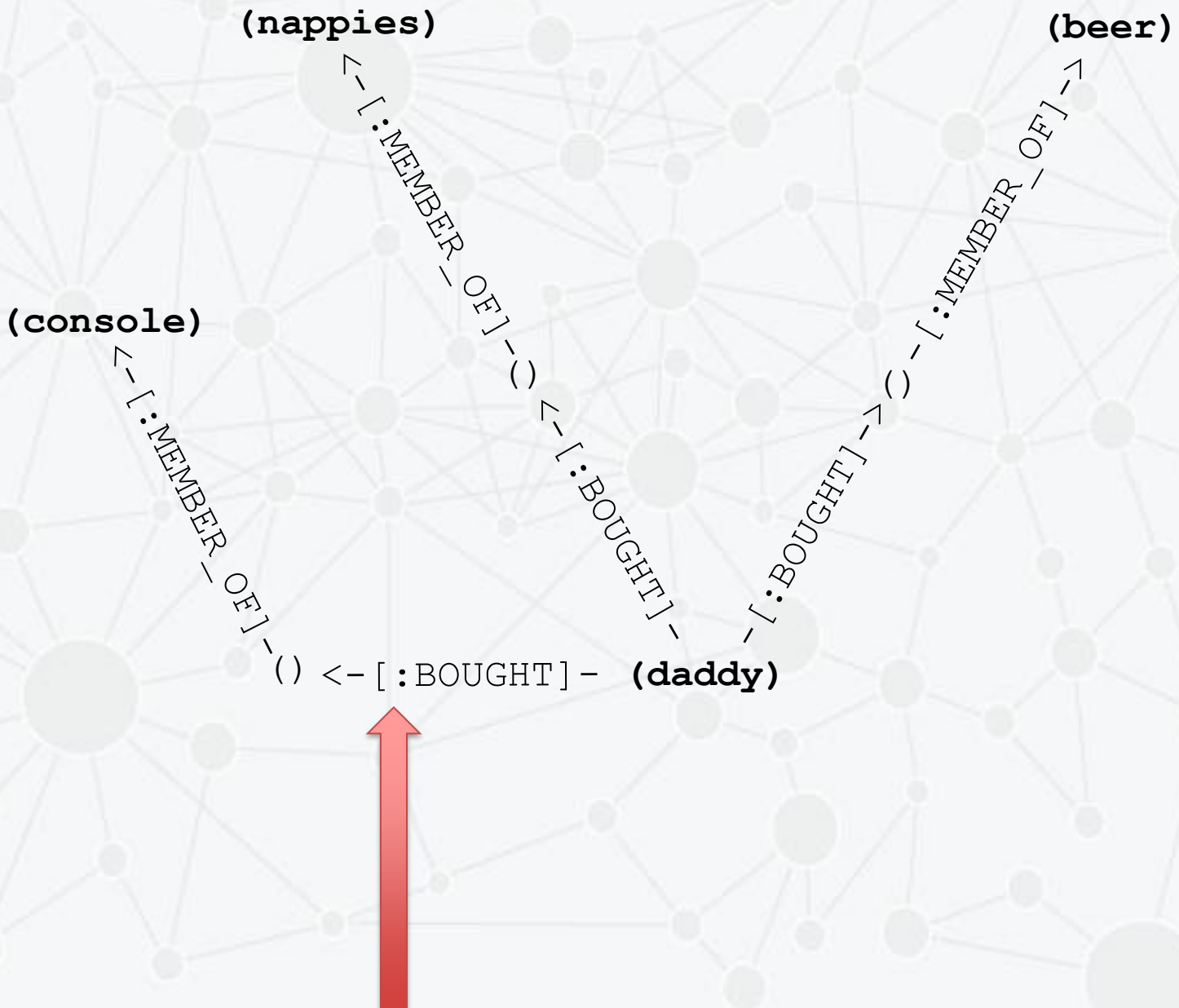
CYPHER MATCH n RETURN n



Displaying 14 nodes, 16 relationships


Category: nappies
Category: beer
BOUGHT
BOUGHT
Category: game console
BOUGHT
Firstname: *
Surname: *
DoB: 1996 > x > 1972


Category: nappies
Category: beer
BOUGHT
BOUGHT
Category: game console
!BOUGHT
Firstname: *
Surname: *
DoB: 1996 > x > 1972


(nappies)
(beer)
(console)
-[:MEMBER_OF]->
-[:MEMBER_OF]->
-[:MEMBER_OF]->
()
-[:BOUGHT]->
()
-[:BOUGHT]->
-[:MEMBER_OF]->
() <-[:BOUGHT]- (daddy)

Flatten the graph

```
(d)-[:BOUGHT]->()-[:MEMBER_OF]->(n)
(d)-[:BOUGHT]->()-[:MEMBER_OF]->(b)


(d)-[:BOUGHT]->()-[:MEMBER_OF]->(c)
```

Include any labels

```
(d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(n:Category)
(d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(b:Category)
```

```
(d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(c:Category)
```

Add a MATCH clause

```
MATCH (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(n:Category),
      (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(b:Category)
```

Constrain the Pattern

```
MATCH (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(n:Category),
      (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(b:Category),
      (c:Category)

WHERE NOT((d)-[:BOUGHT]->()-[:MEMBER_OF]->(c))
```

Add property constraints

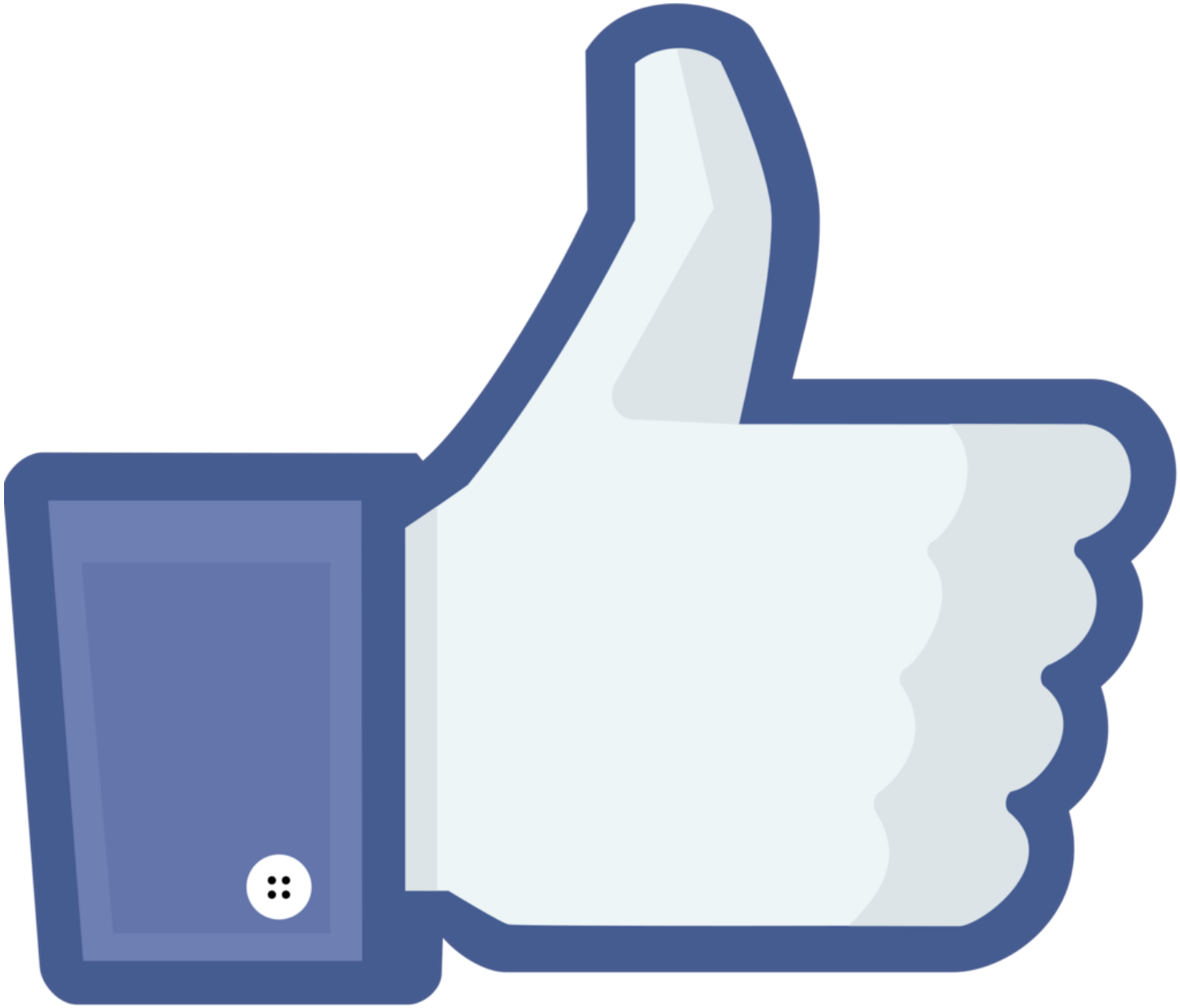
```
MATCH (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(n:Category),
      (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(b:Category),
      (c:Category)

WHERE n.category = "nappies" AND
      b.category = "beer" AND
      c.category = "console" AND
      NOT((d)-[:BOUGHT]->()-[:MEMBER_OF]->(c))
```

Profit!

```
MATCH (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(n:Category),
      (d:Person)-[:BOUGHT]->()-[:MEMBER_OF]->(b:Category),
      (c:Category)

WHERE n.category = "nappies" AND
      b.category = "beer" AND
      c.category = "console" AND
      NOT((d)-[:BOUGHT]->()-[:MEMBER_OF]->(c))

RETURN DISTINCT d AS daddy
```

Results

```
==> +-----------------------------------------------+
==> | daddy                                         |
==> +-----------------------------------------------+
==> | Node[15]{name:"Rory Williams",dob:19880121} |
==> +-----------------------------------------------+
==> 1 row
==> 0 ms
==>
neo4j-sh (0)$
```

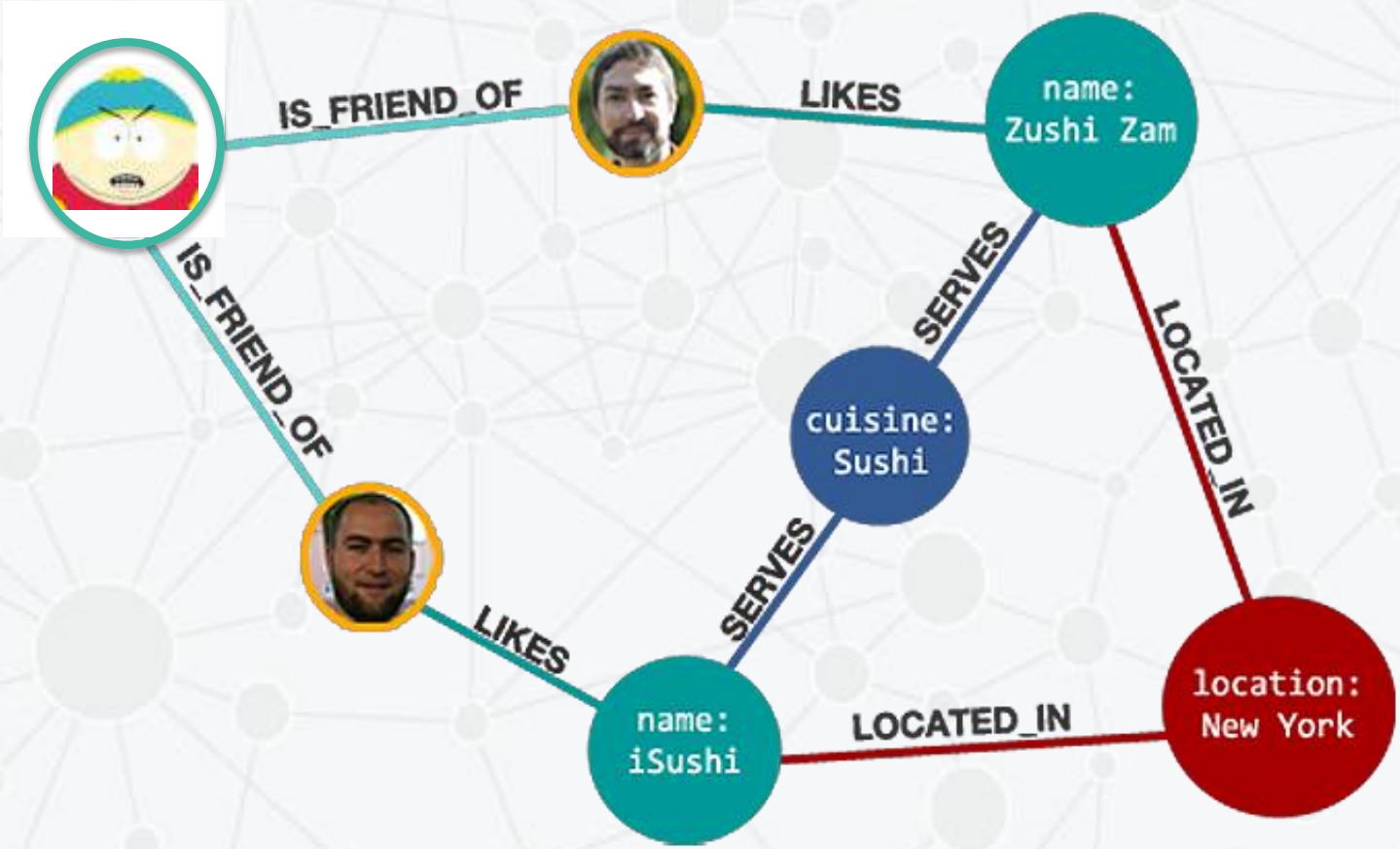


Facebook Graph Search

Which sushi restaurants in NYC do my friends like?

See http://maxdemarzi.com/

Graph Structure



Cypher query is easy!

```
MATCH (me)
        -[:IS_FRIEND_OF]->()
        -[:LIKES]->(restaurant)
        -[:LOCATED_IN]->(city),
       (restaurant)-[:SERVES]->(cuisine)

WHERE me.name = 'Jim' AND
      city.location='New York' AND
      cuisine.cuisine='Sushi'

RETURN restaurant
```

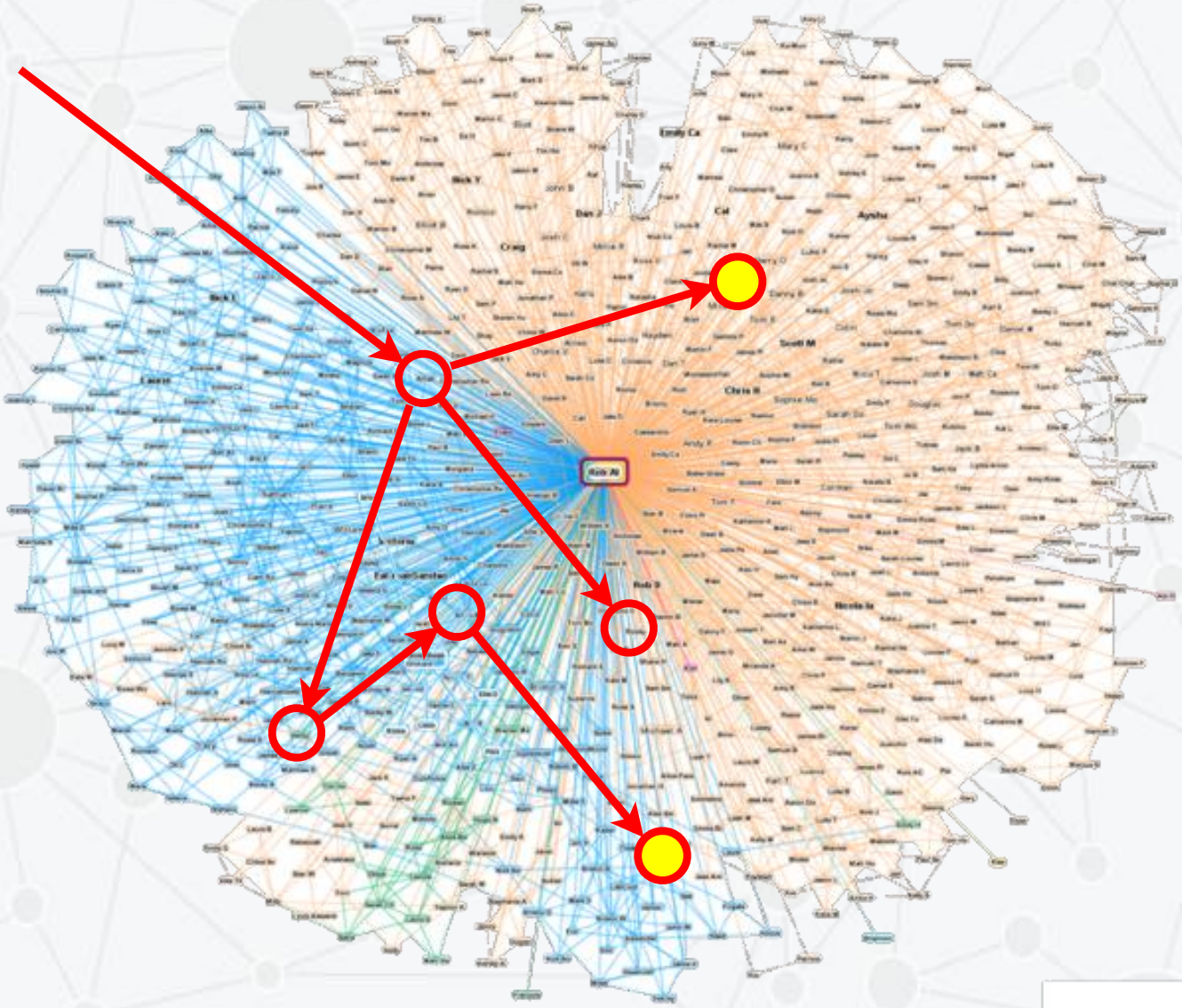
And richer with labels

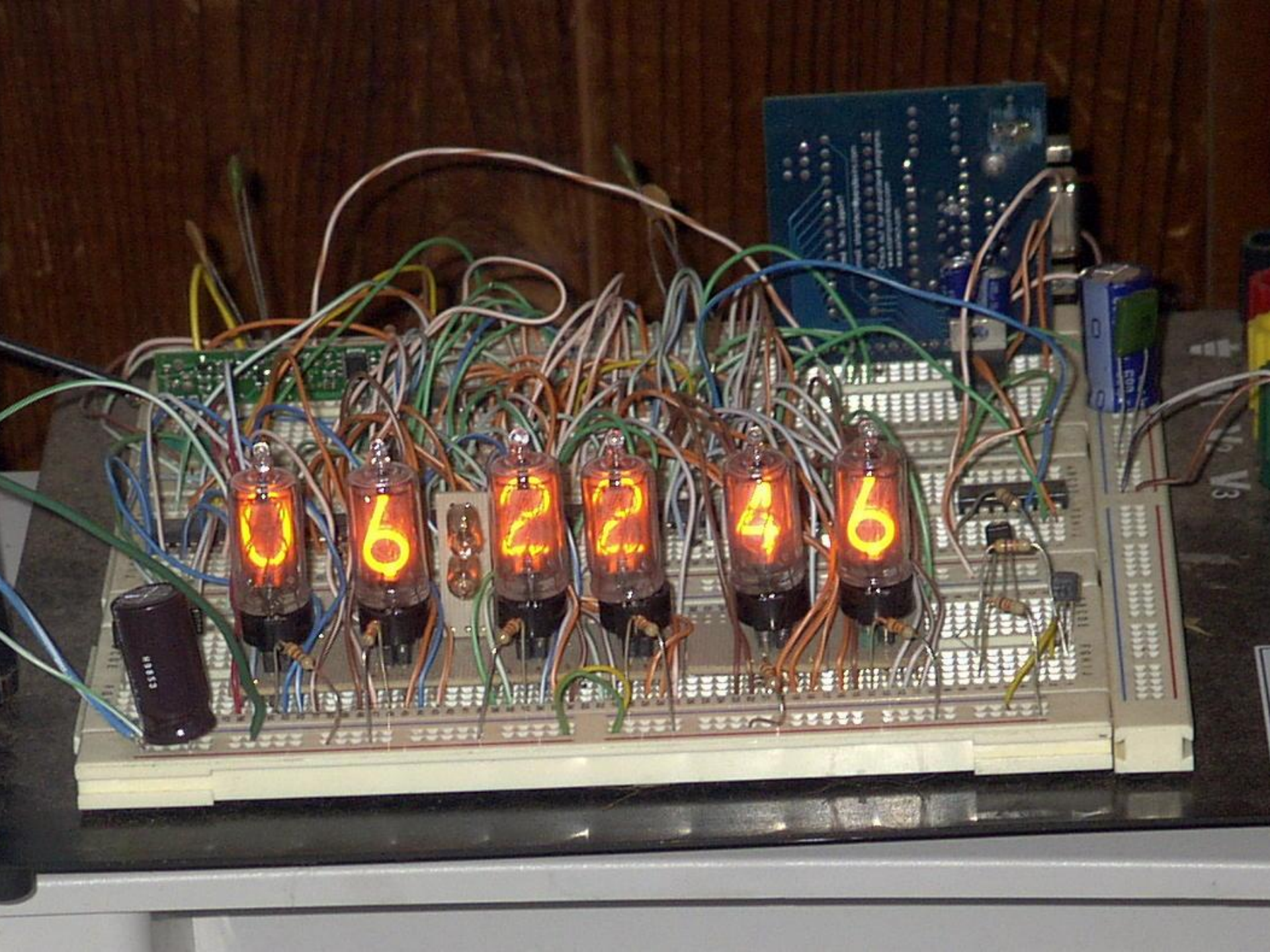
```
MATCH (me:Person)
        -[:IS_FRIEND_OF]->(:Person)
        -[:LIKES]->(restaurant:Restaurant)
        -[:LOCATED_IN]->(city:Place),
       (restaurant)-[:SERVES]->(cuisine:Cuisine)

WHERE me.name = 'Jim' AND
      city.location='New York' AND
      cuisine.cuisine='Sushi'

RETURN restaurant
```

And clearer with compact MATCH…

```
MATCH (:Person {name: 'Jim'})
  -[:IS_FRIEND_OF]->(:Person)
  -[:LIKES]->(restaurant:Restaurant)
  -[:LOCATED_IN]->(:Place {location: 'New York'}),
  (restaurant)-[:SERVES]->(:Cuisine {cuisine: 'Sushi'})

RETURN restaurant
```

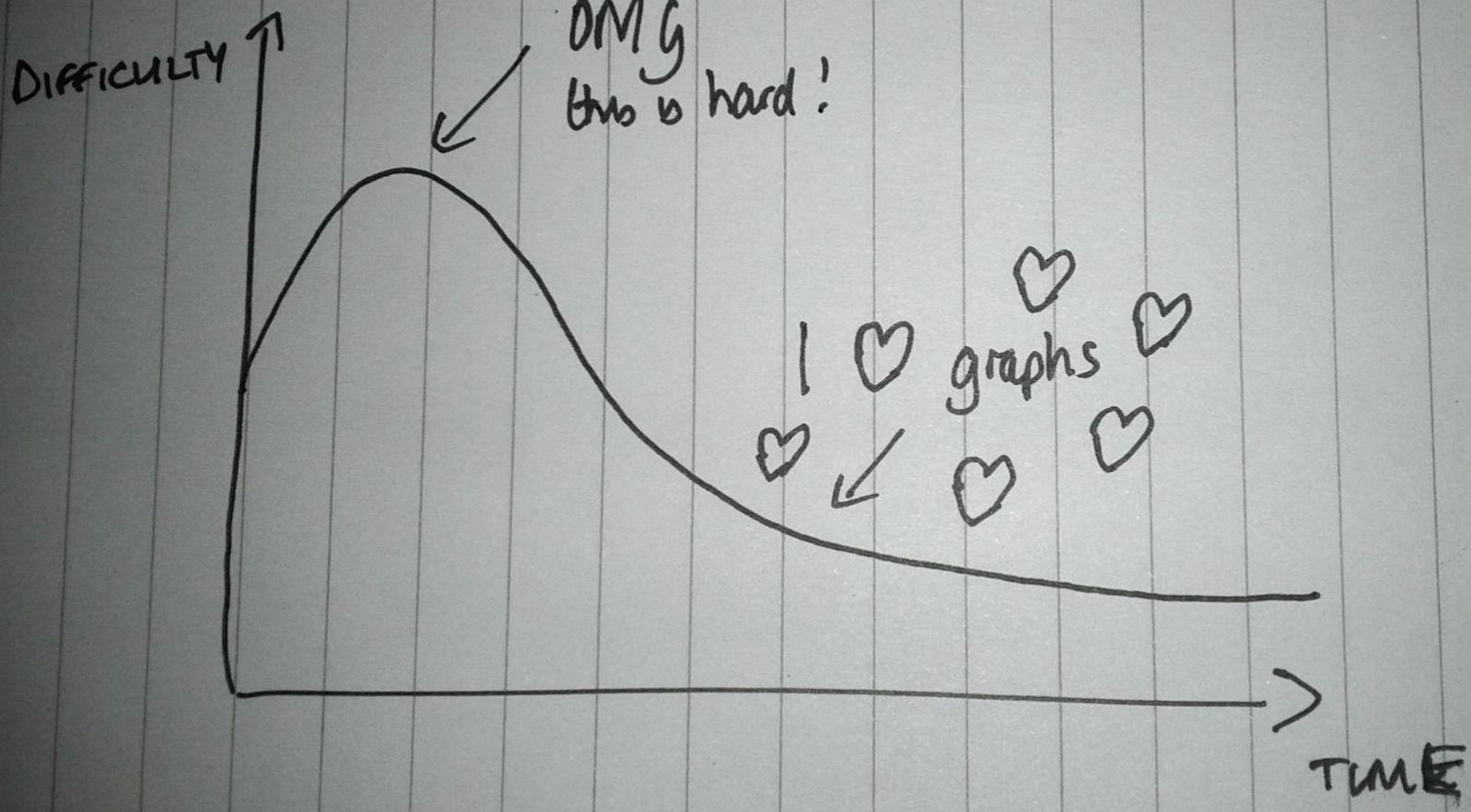
Search structure



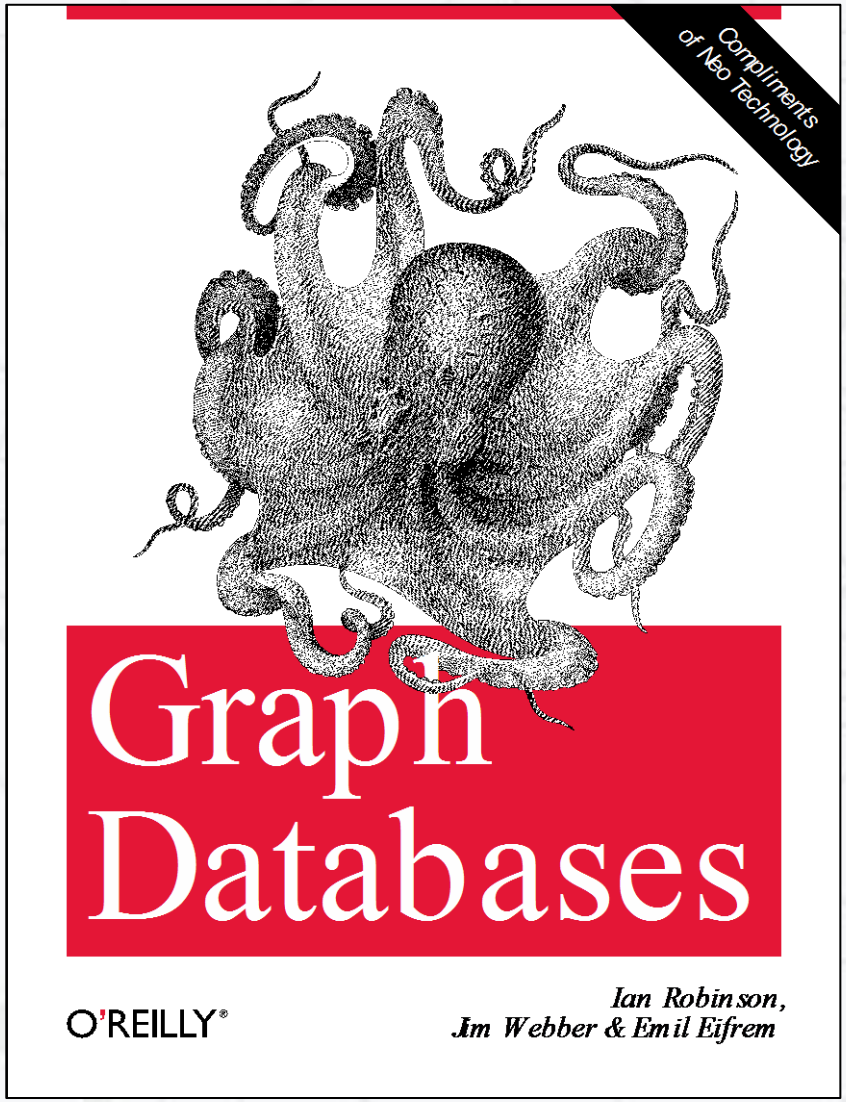

0 6 2 2 4 6

What's Neo4j good for?

- Data centre management
- Supply chain/provenance
- Recommendations
- Business intelligence
- Social computing
- MDM
- Web of things
- Time series/event data
- Product/engineering catalogue
- Web analytics, user journeys
- Scientific computing
- Spatial
- Geo/Seismic/Meteorological
- Bio/Pharma
- And many, many more…



credit: @markhneedham



Free O'Reilly book!

Free **Full** eBook version:
http://graphdatabases.com
for the eBook version

Don’t forget the Neo4j Stockholm Meetups!

Neo4j

Thanks for listening

@jimwebber