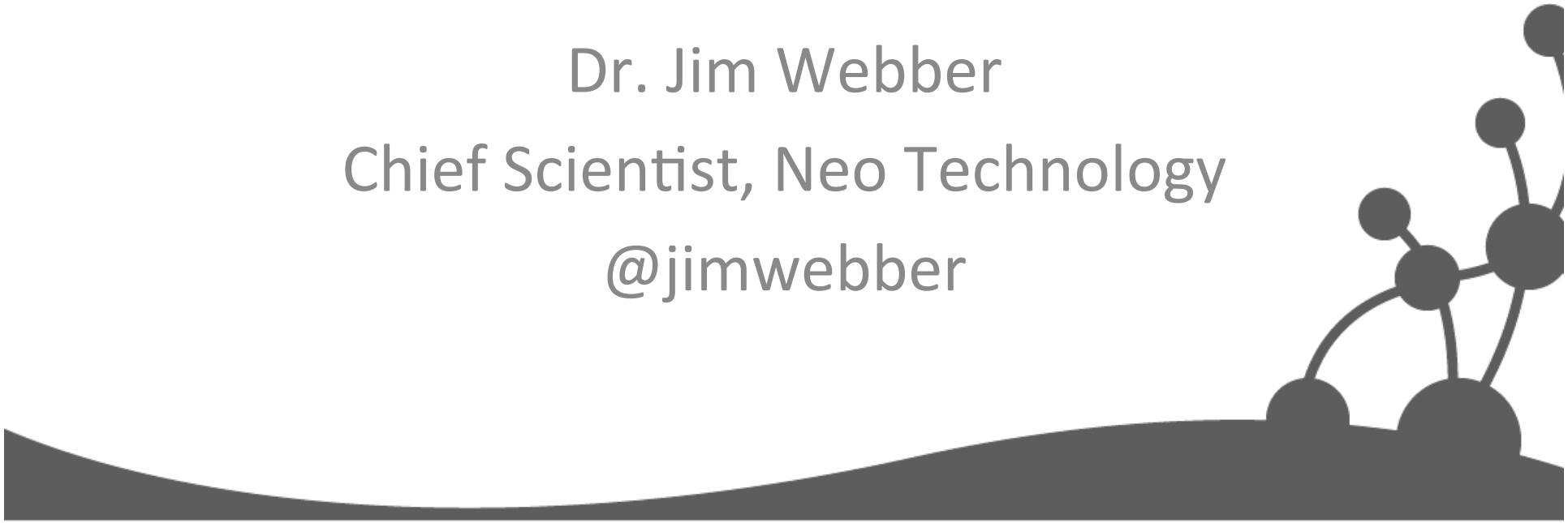
A Little Graph Theory for the Busy Developer

Dr. Jim Webber

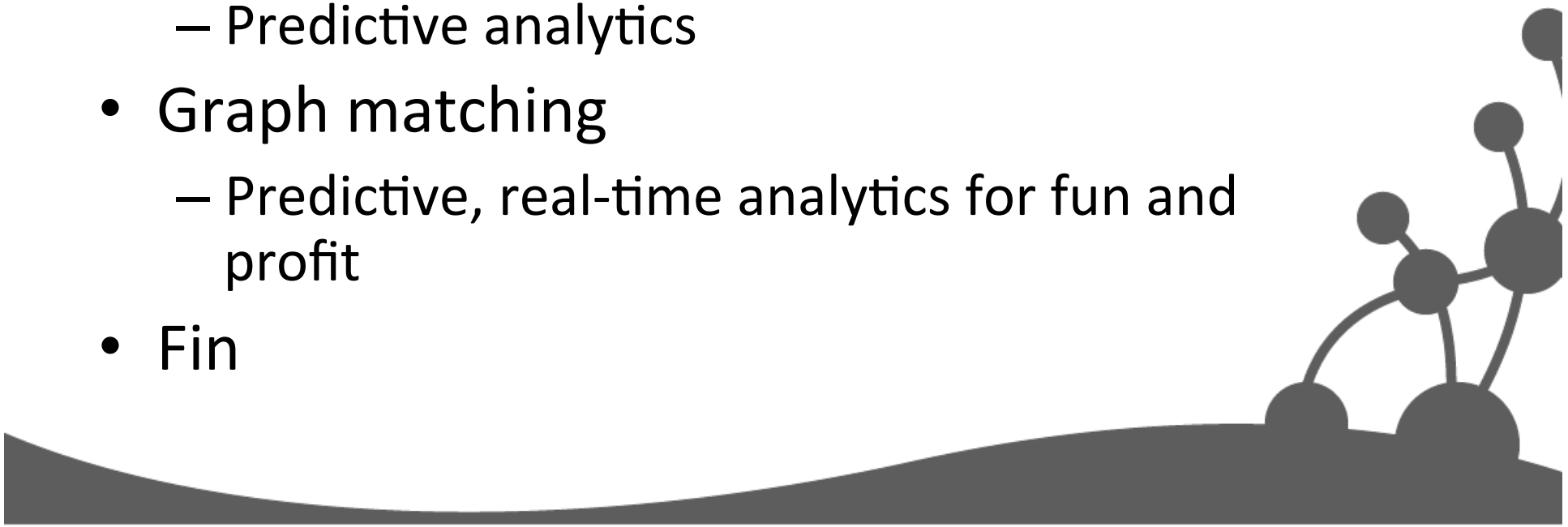
Chief Scientist, Neo Technology

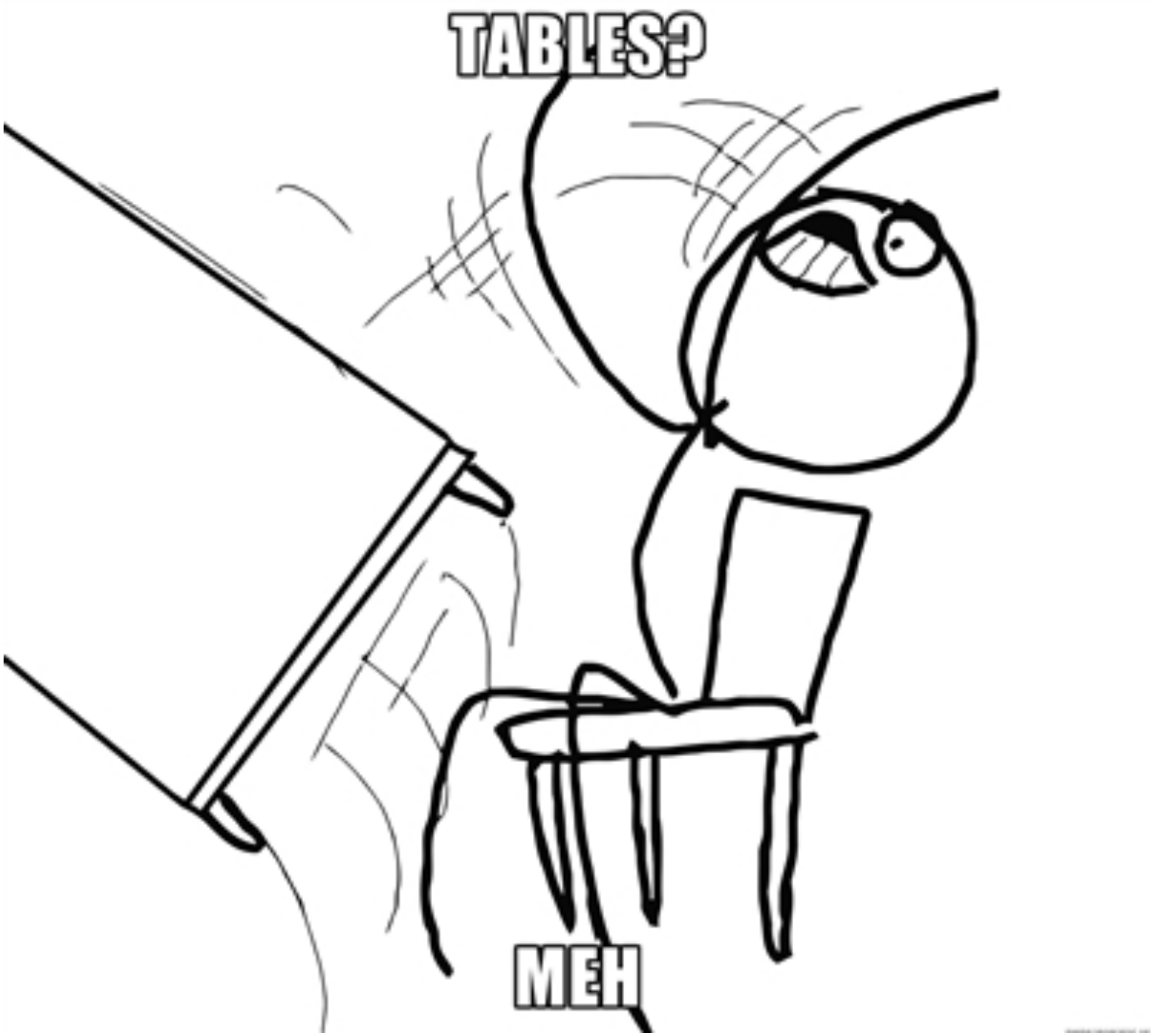
@jimwebber



Roadmap

- Imprisoned data
- Graph models
- Graph theory
  - Local properties, global behaviours
  - Predictive analytics
- Graph matching
  - Predictive, real-time analytics for fun and profit
- Fin




TABLES?
MEH

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


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

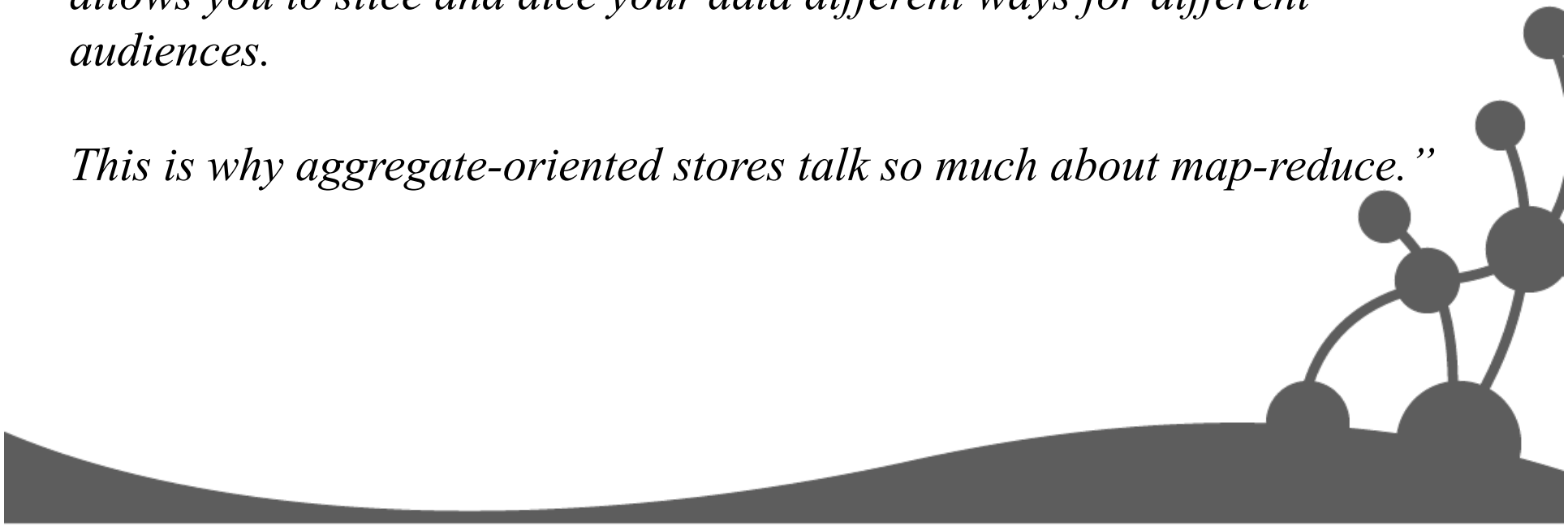

http://www.xtranormal.com/watch/6995033/mongo-db-is-web-scale

Aggregate-Oriented Data

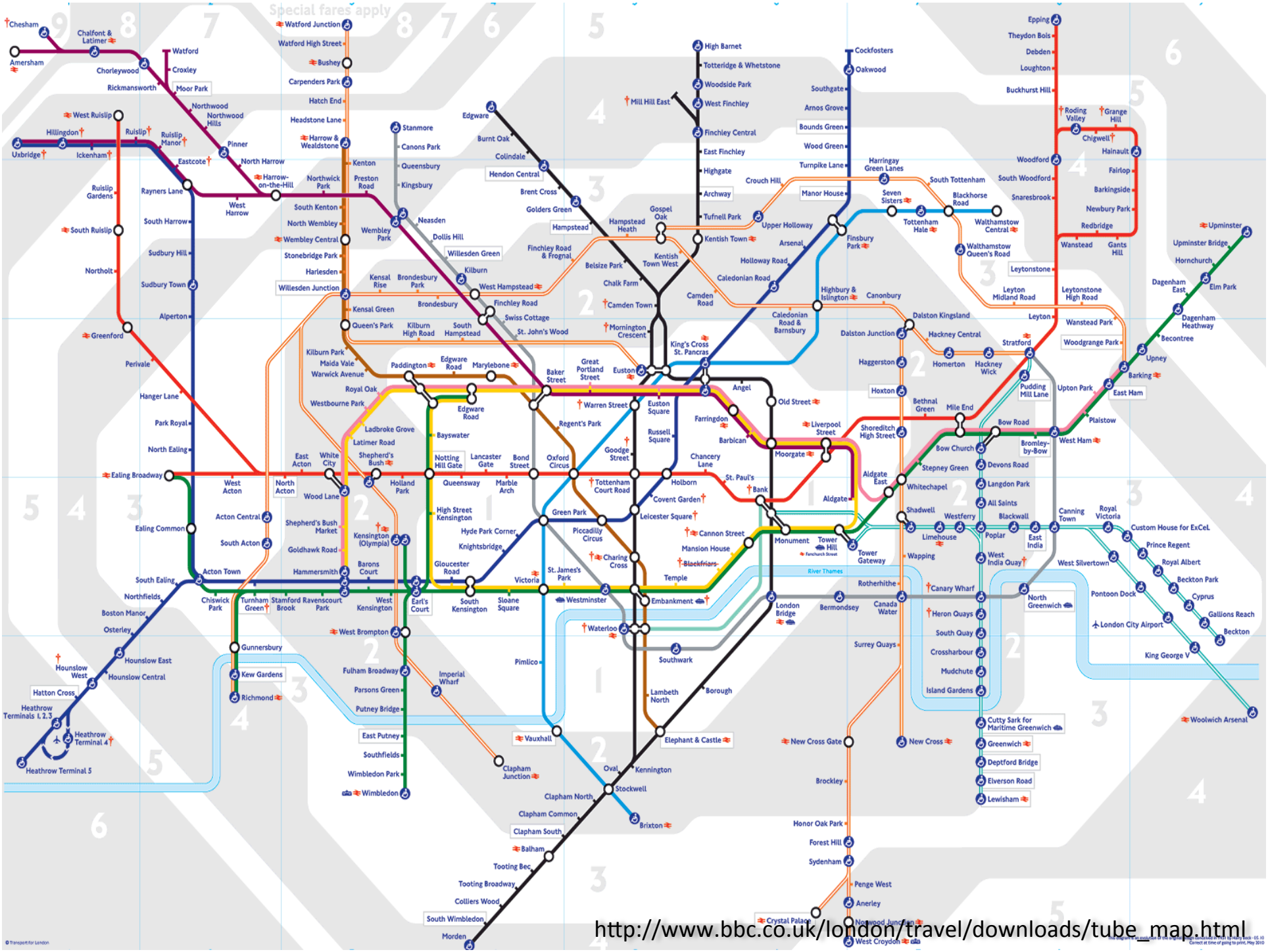
http://martinfowler.com/bliki/AggregateOrientedDatabase.html

"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."



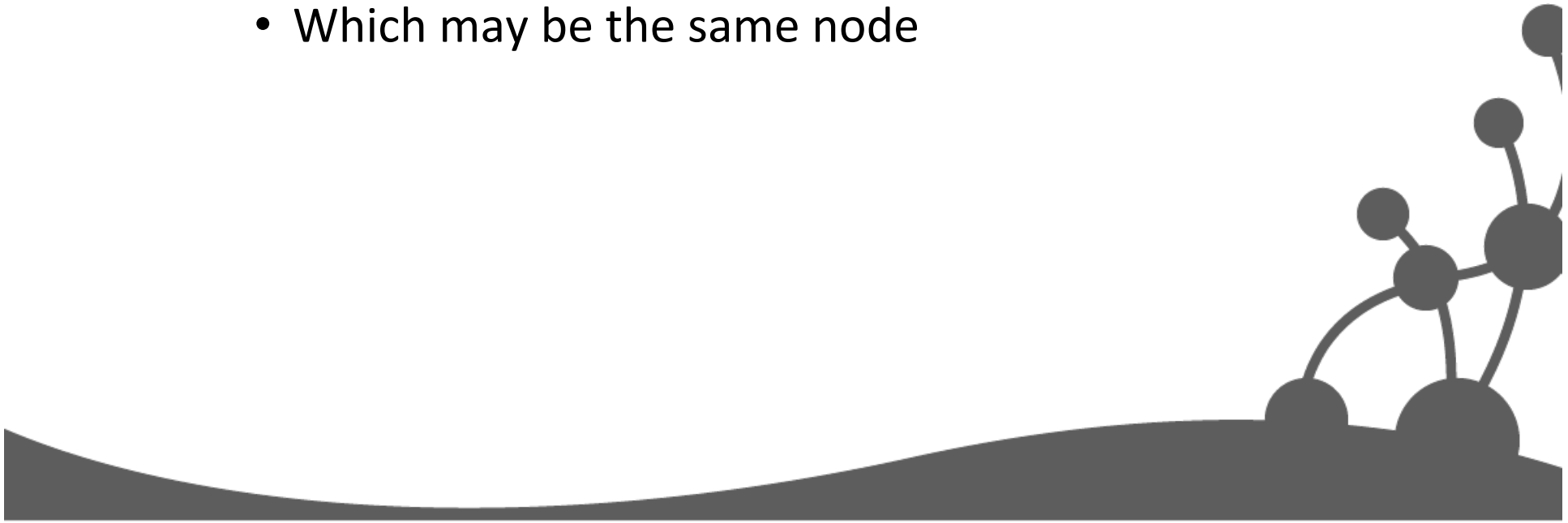
$$complexity = f(size, connectedness, uniformity)$$



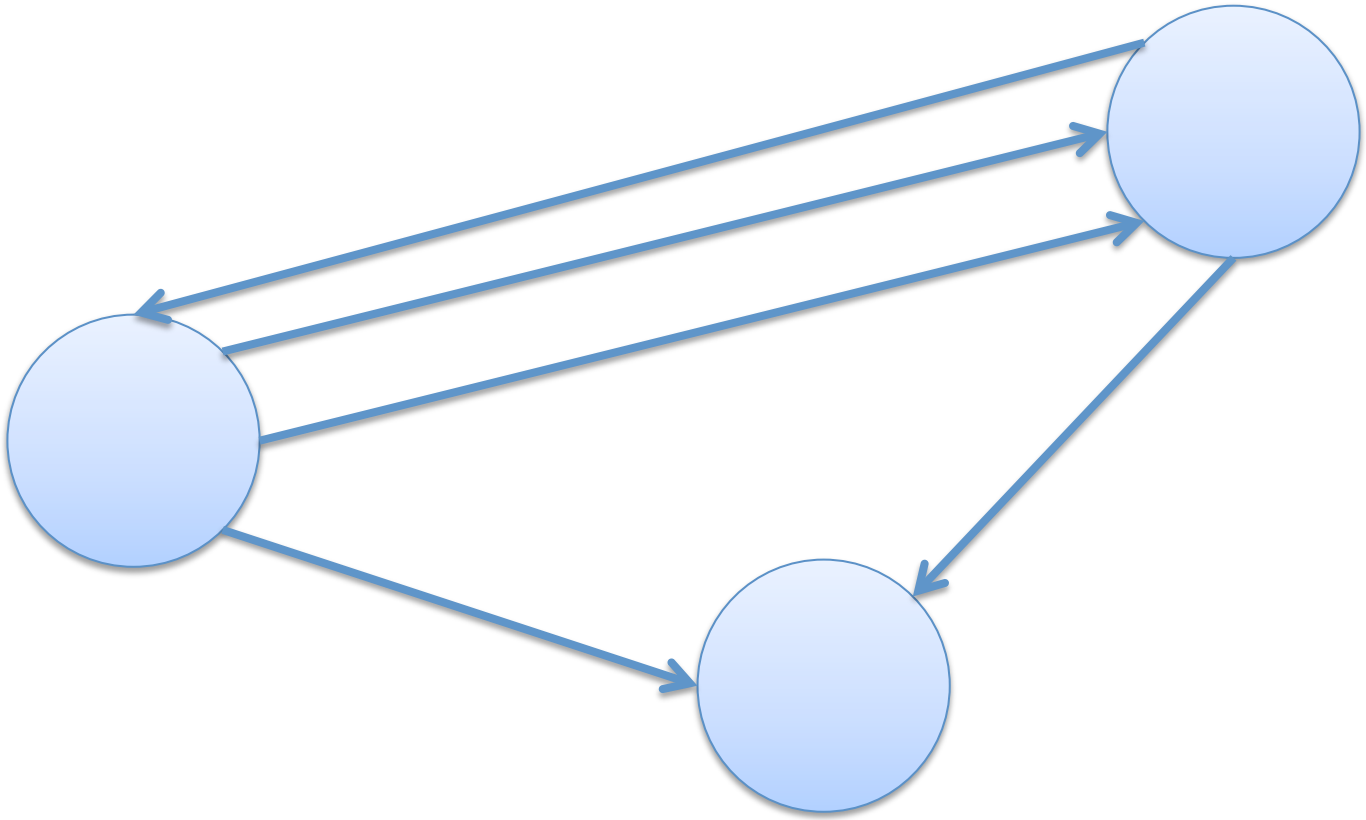
http://www.bbc.co.uk/london/travel/downloads/tube_map.html

Property graphs

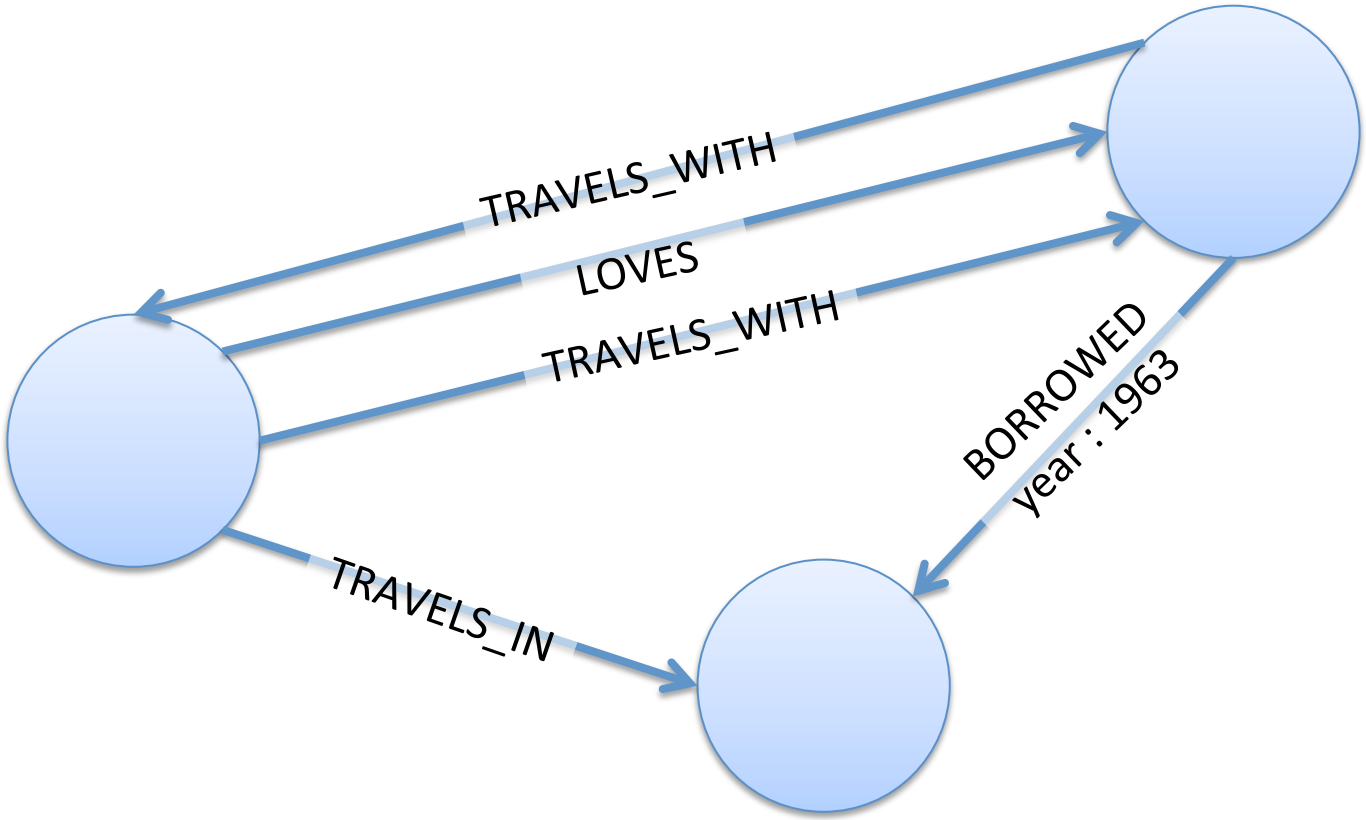
- Property graph model:
  - Nodes with properties
  - Named, directed relationships with properties
  - Relationships have exactly one start and end node
    - Which may be the same node



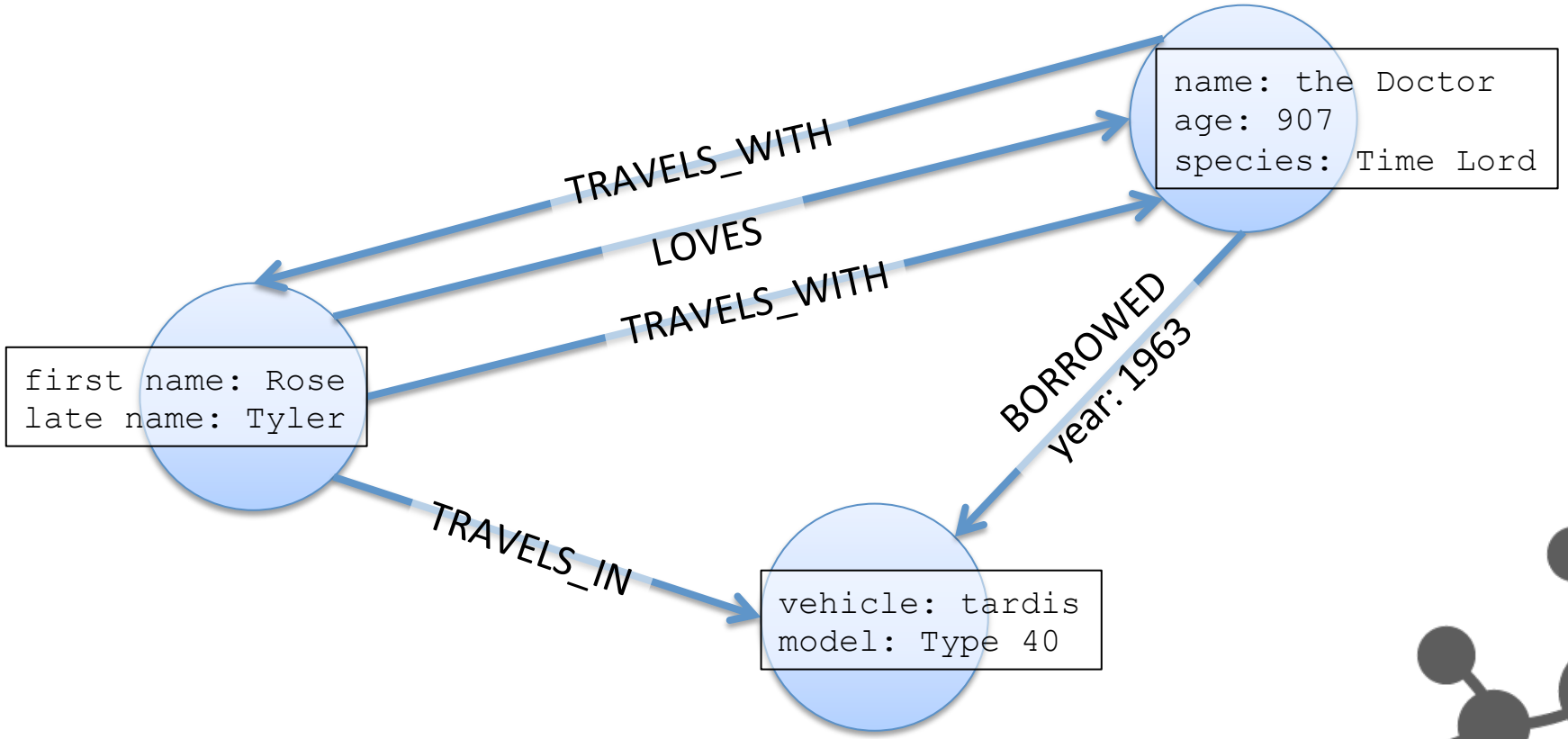
Property Graph Model



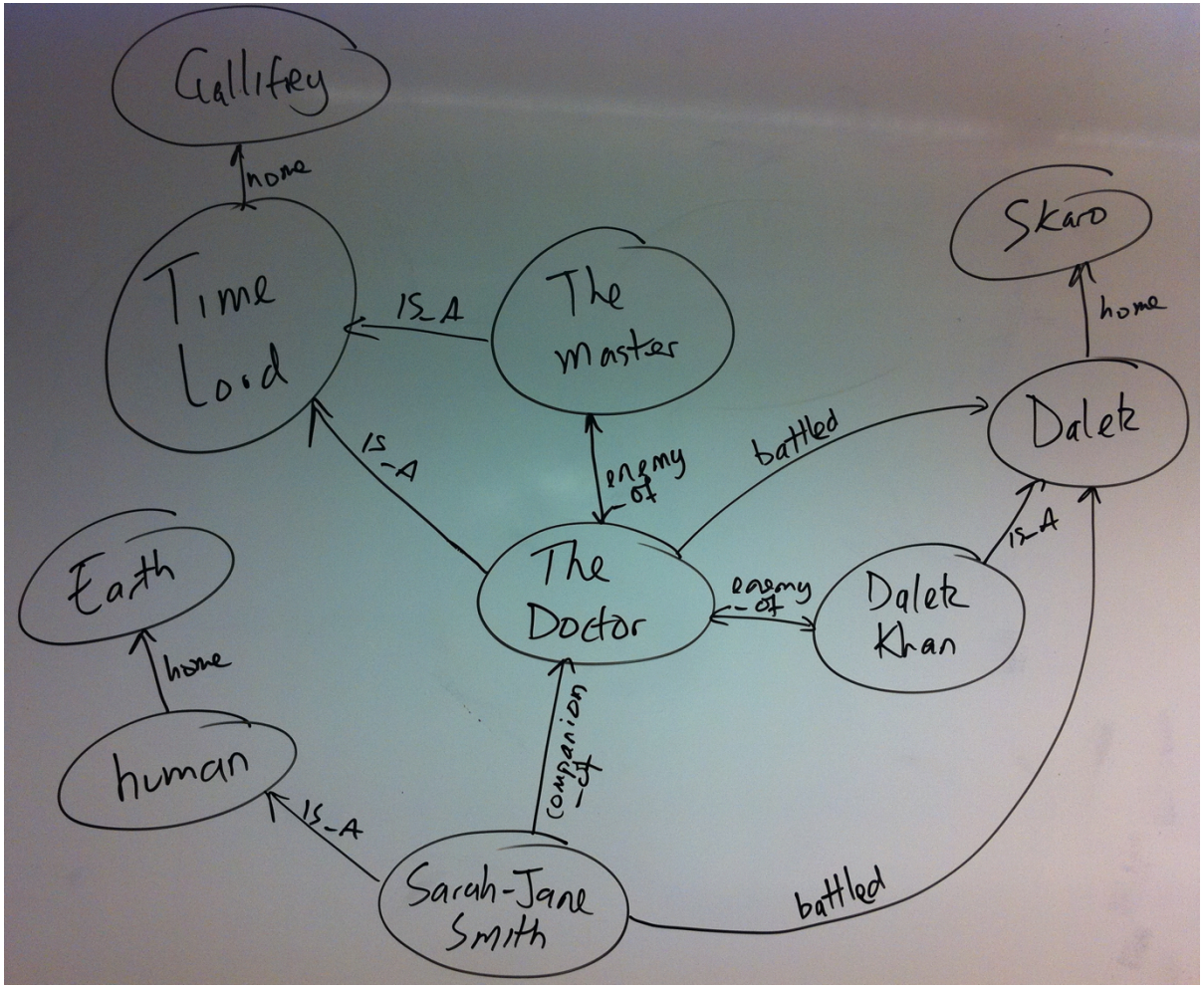
Property Graph Model



Property Graph Model



Property graphs are *very* whiteboard-friendly

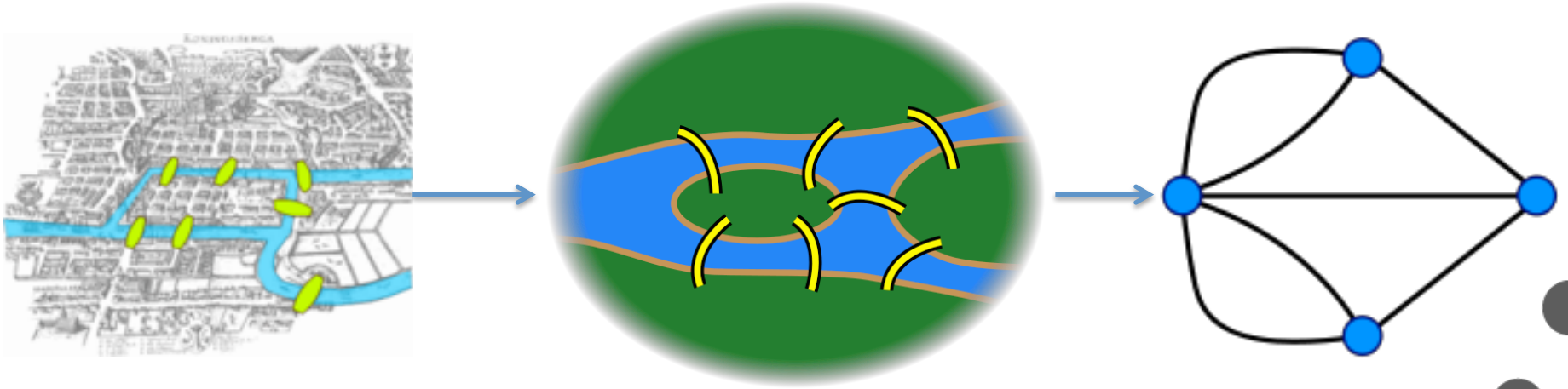


http://blogs.adobe.com/digitalmarketing/analytics/predictive-analytics/predictive-analytics-and-the-digital-marketer/

Meet Leonhard Euler

- Swiss mathematician
- Inventor of Graph Theory (1736)

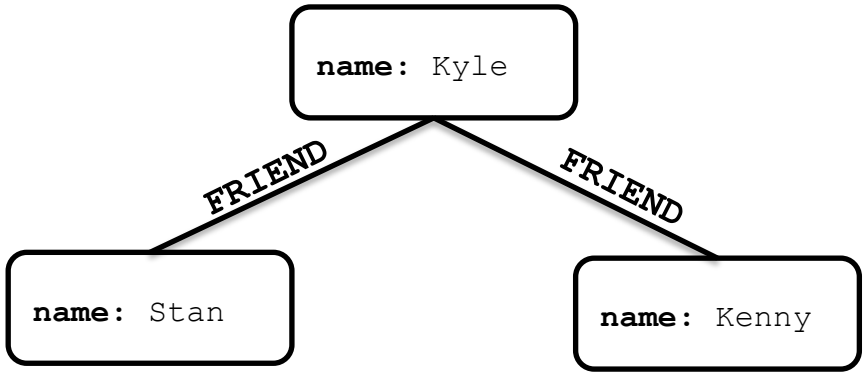
http://en.wikipedia.org/wiki/File:Leonhard_Euler_2.jpg



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

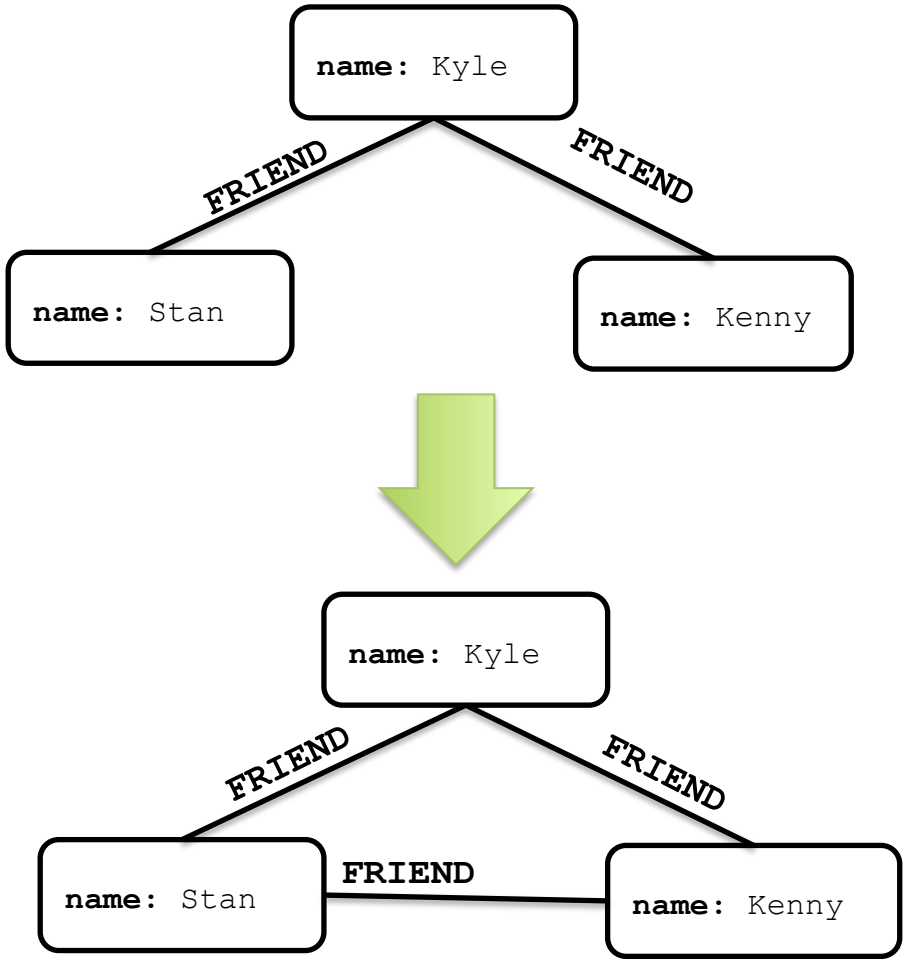

DULWICH SUPERMARKET & OFF LICENCE
ENGLISH , TURKISH , GREEK , MEDITERRANEAN FOOD
18
DELICATESSEN & ORGANICS
TEL : 020 8299 2214
FRESHLY CUT SANDWICHES - BILTONG
BILTONG
BOEREWORS

Triadic Closure

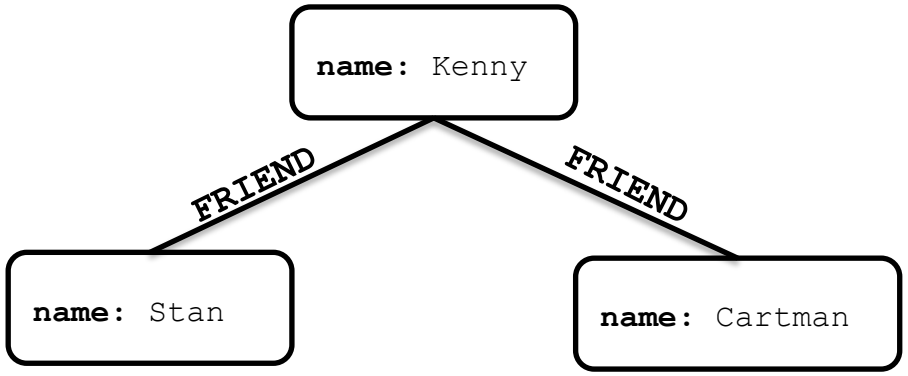


Triadic Closure

(strong relationship)

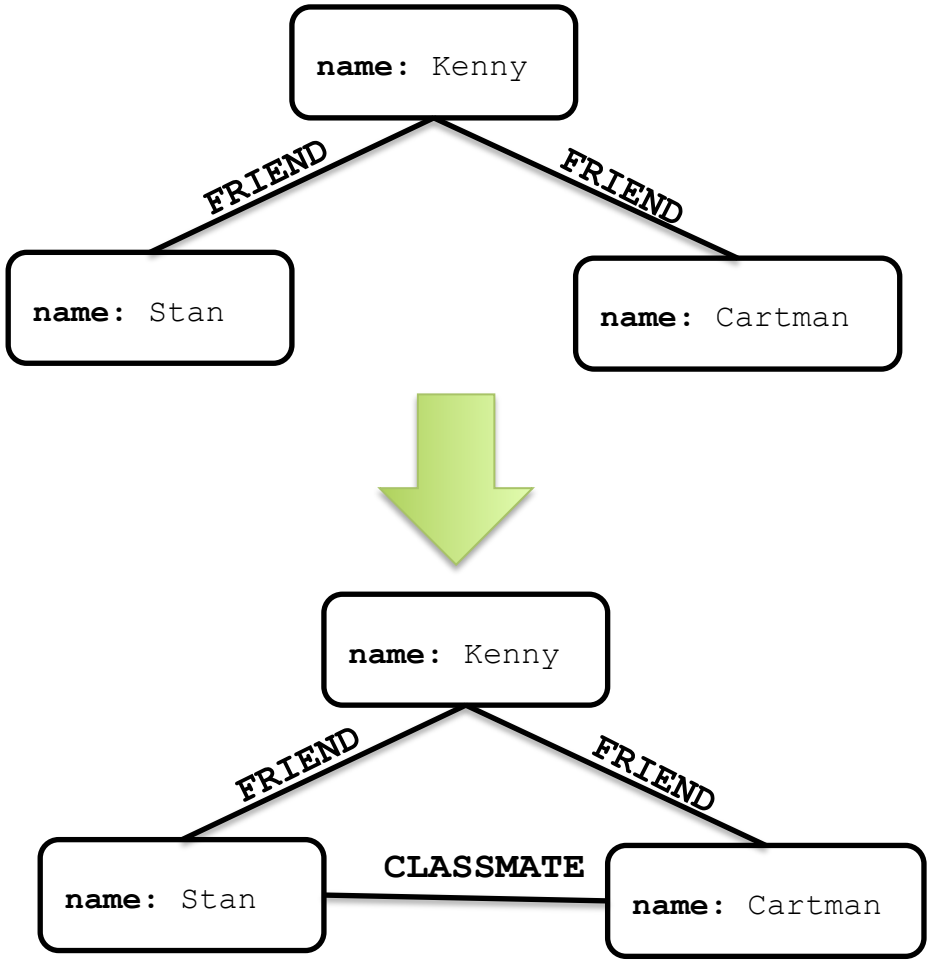


Triadic Closure

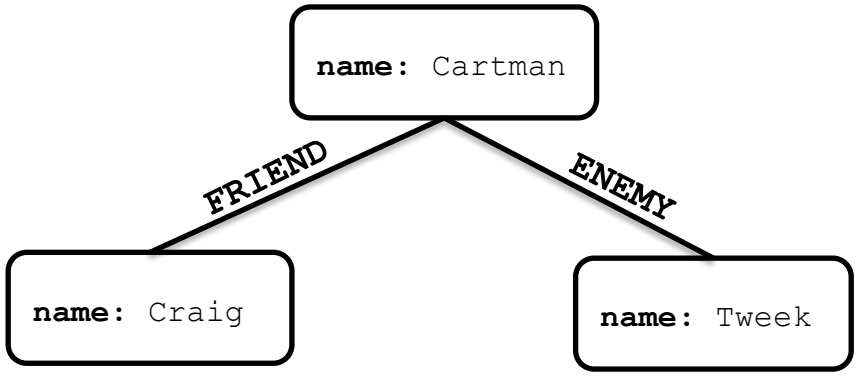


Triadic Closure

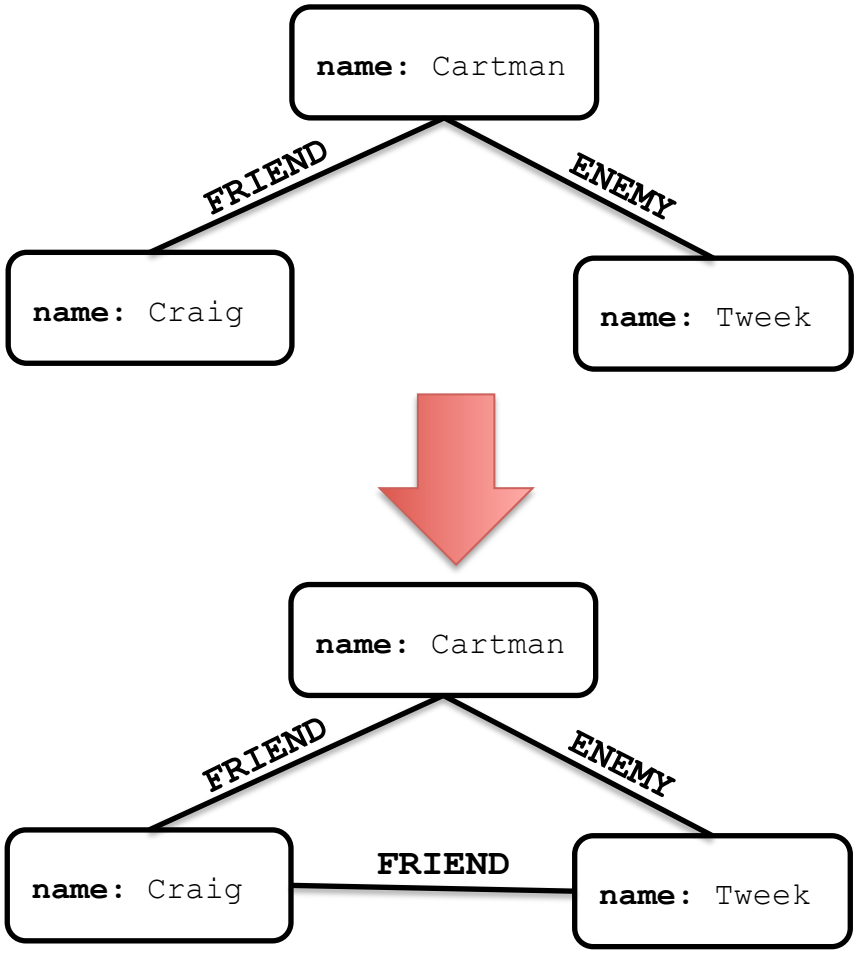
(weak relationship)



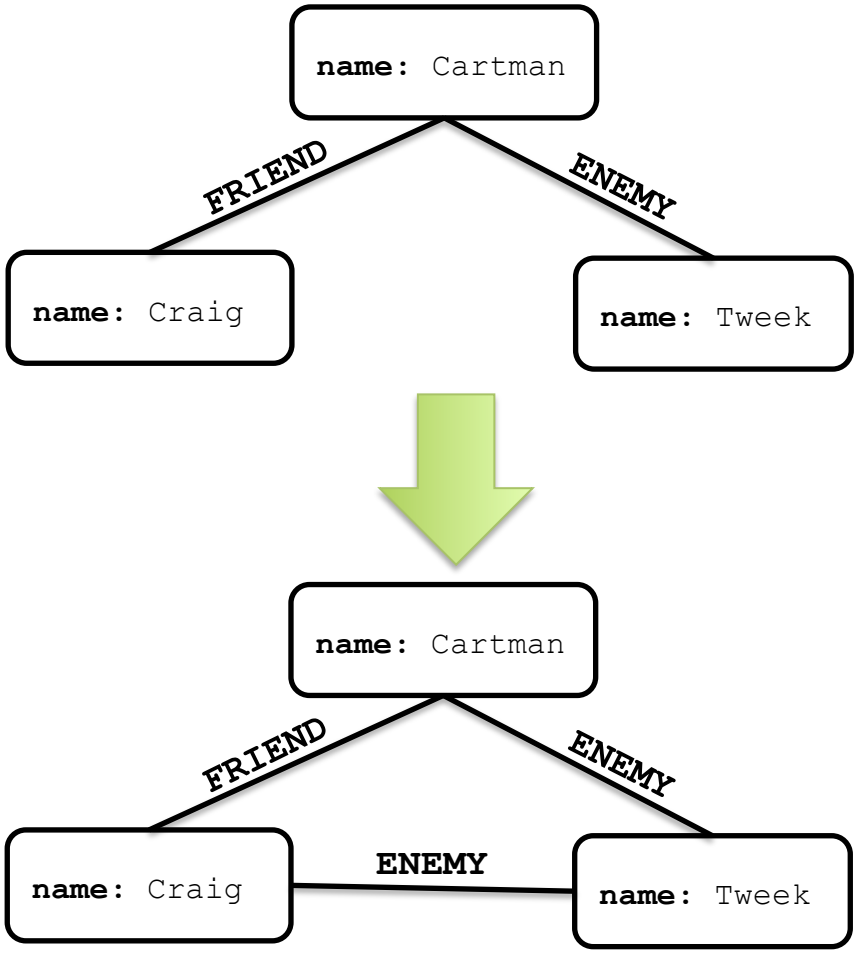
Structural Balance



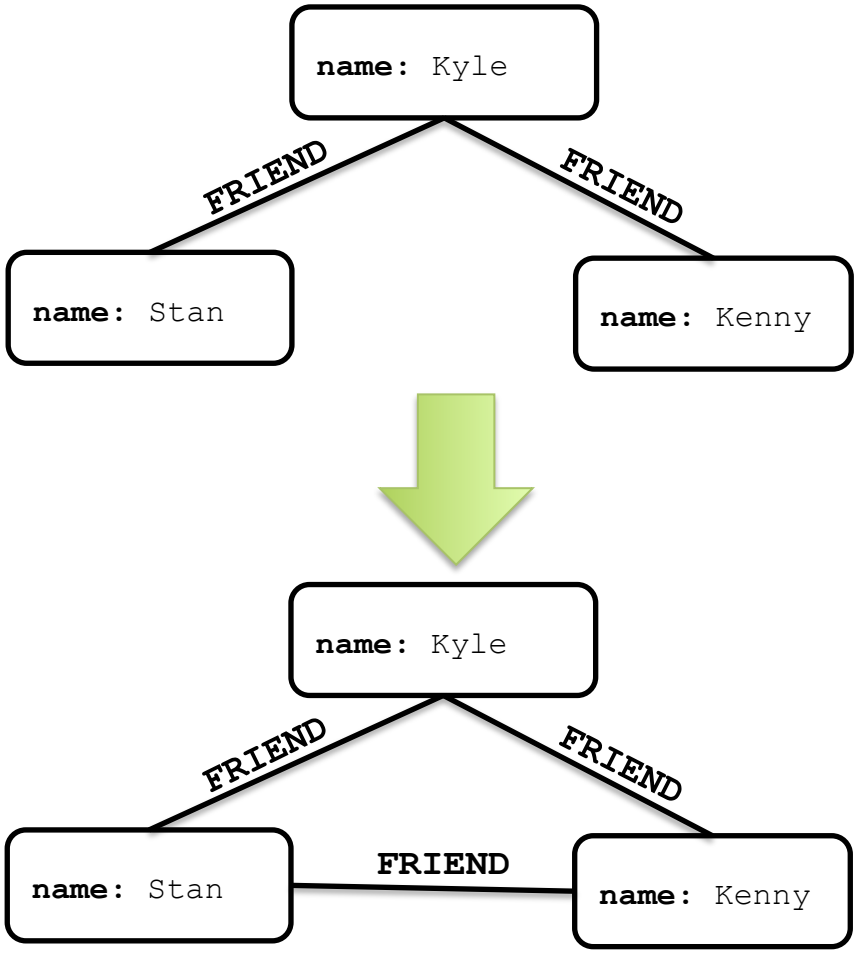
Structural Balance



Structural Balance

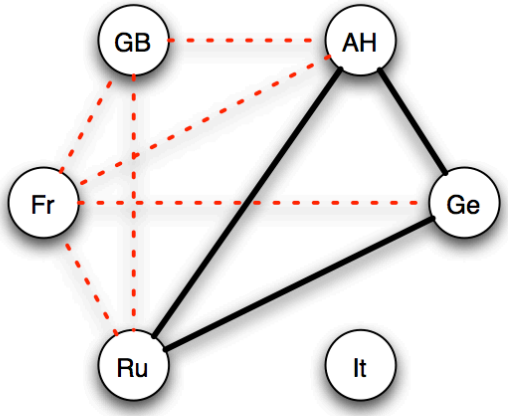


Structural Balance

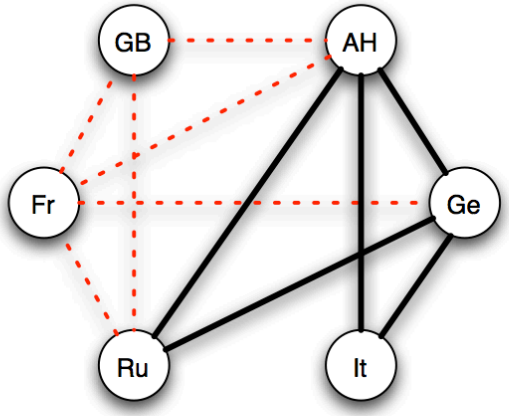


WWI caused by balanced closures?

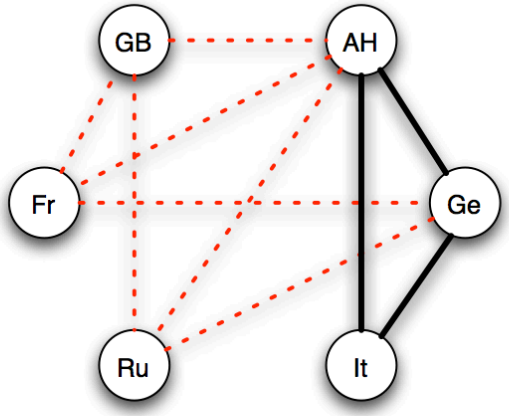
[Easley and Kleinberg]



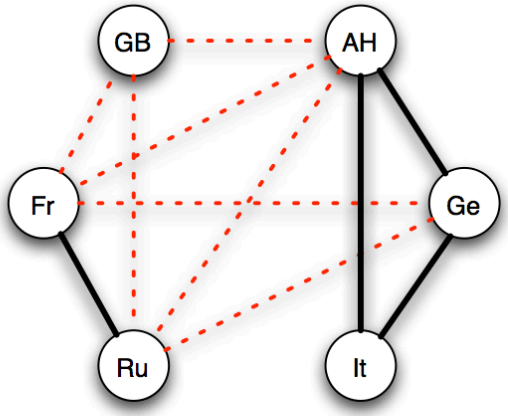
(a) *Three Emperors' League 1872–81*



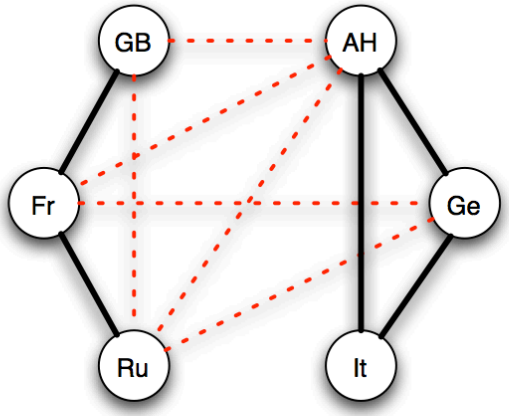
(b) *Triple Alliance 1882*



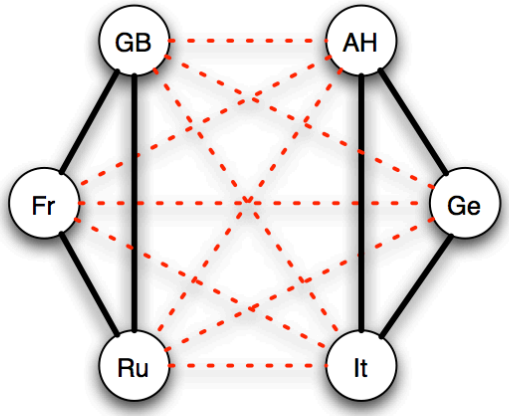
(c) *German-Russian Lapse 1890*



(d) *French-Russian Alliance 1891–94*

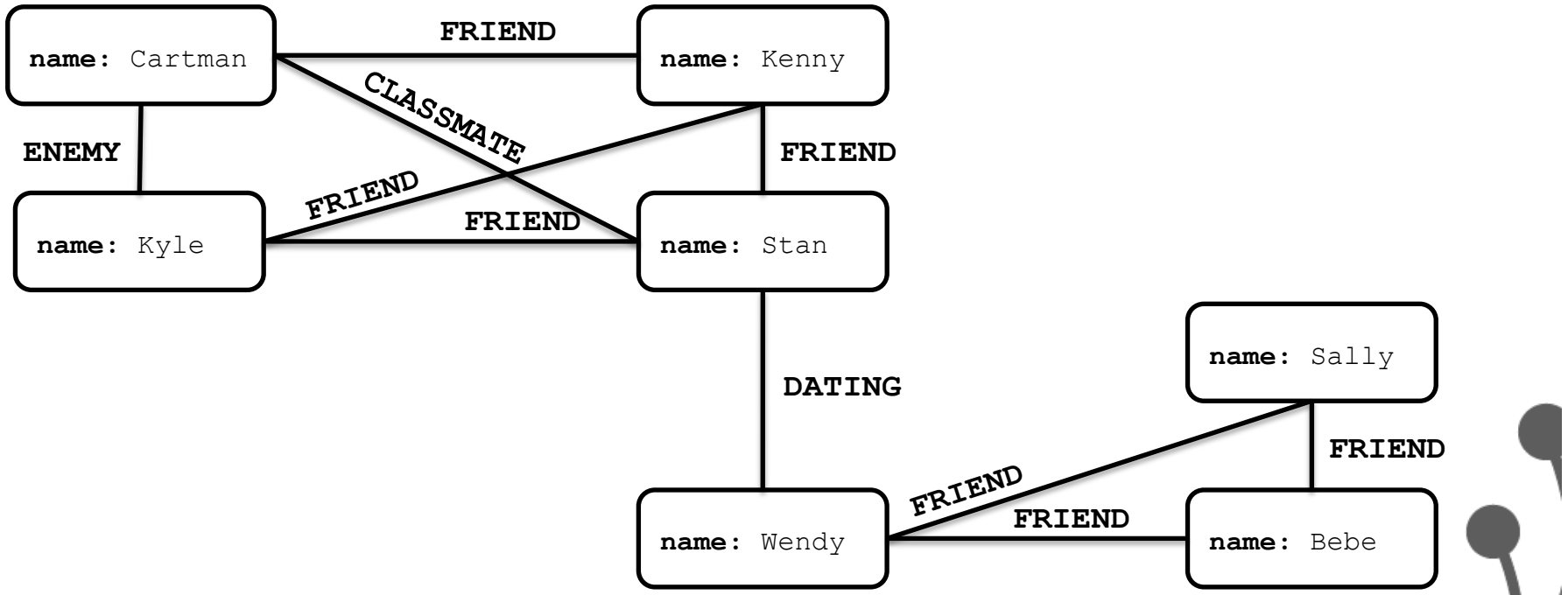


(e) *Entente Cordiale 1904*



(f) *British Russian Alliance 1907*

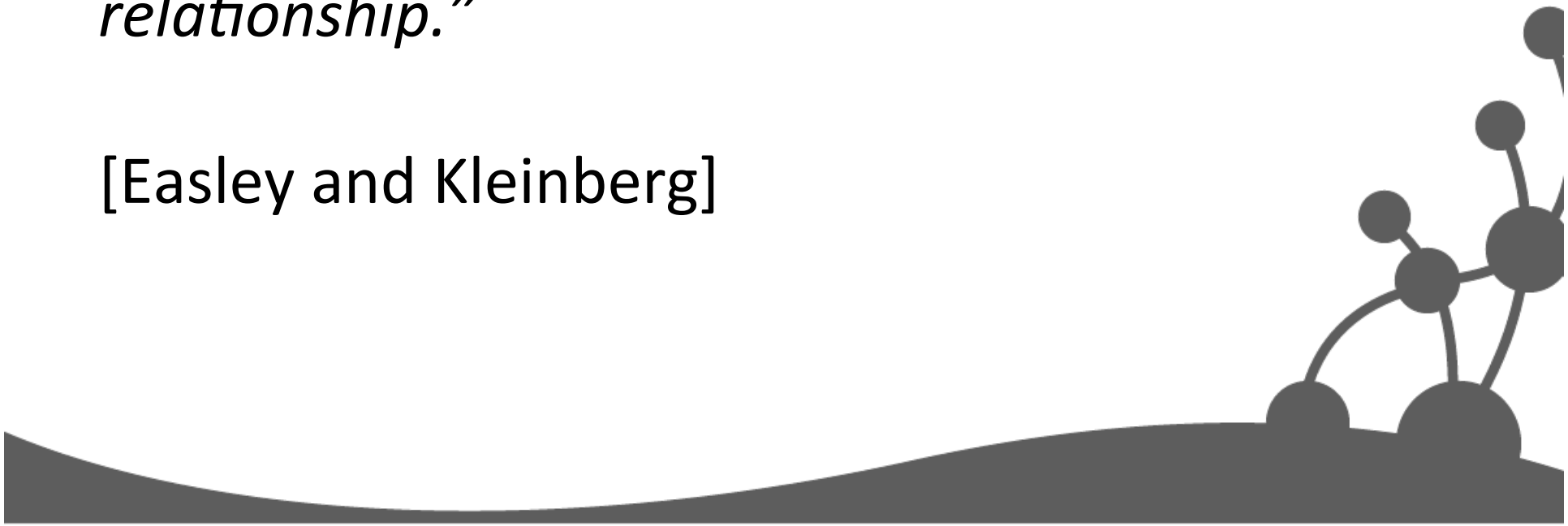
Local Bridges



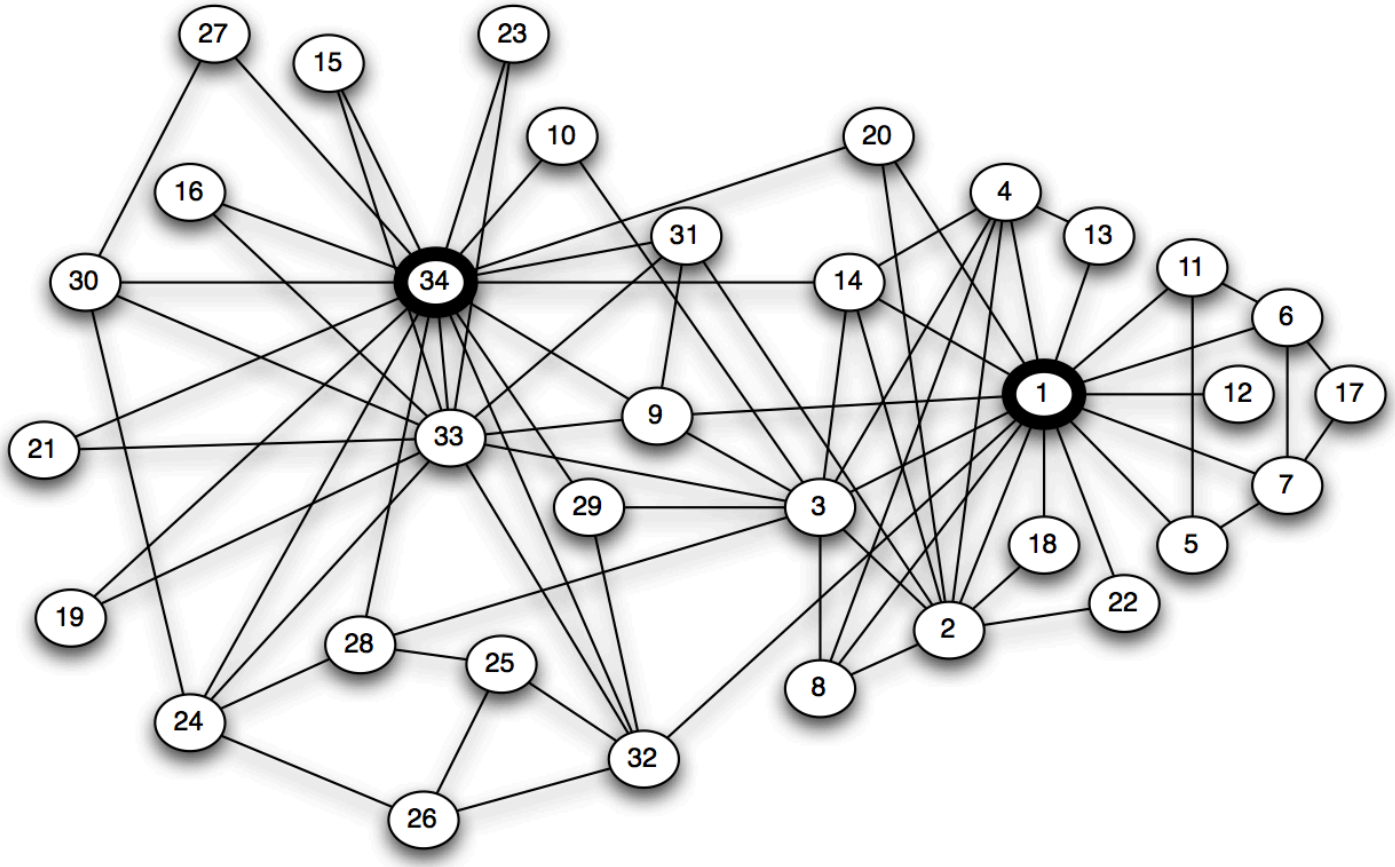
Local Bridge Property

*"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."*

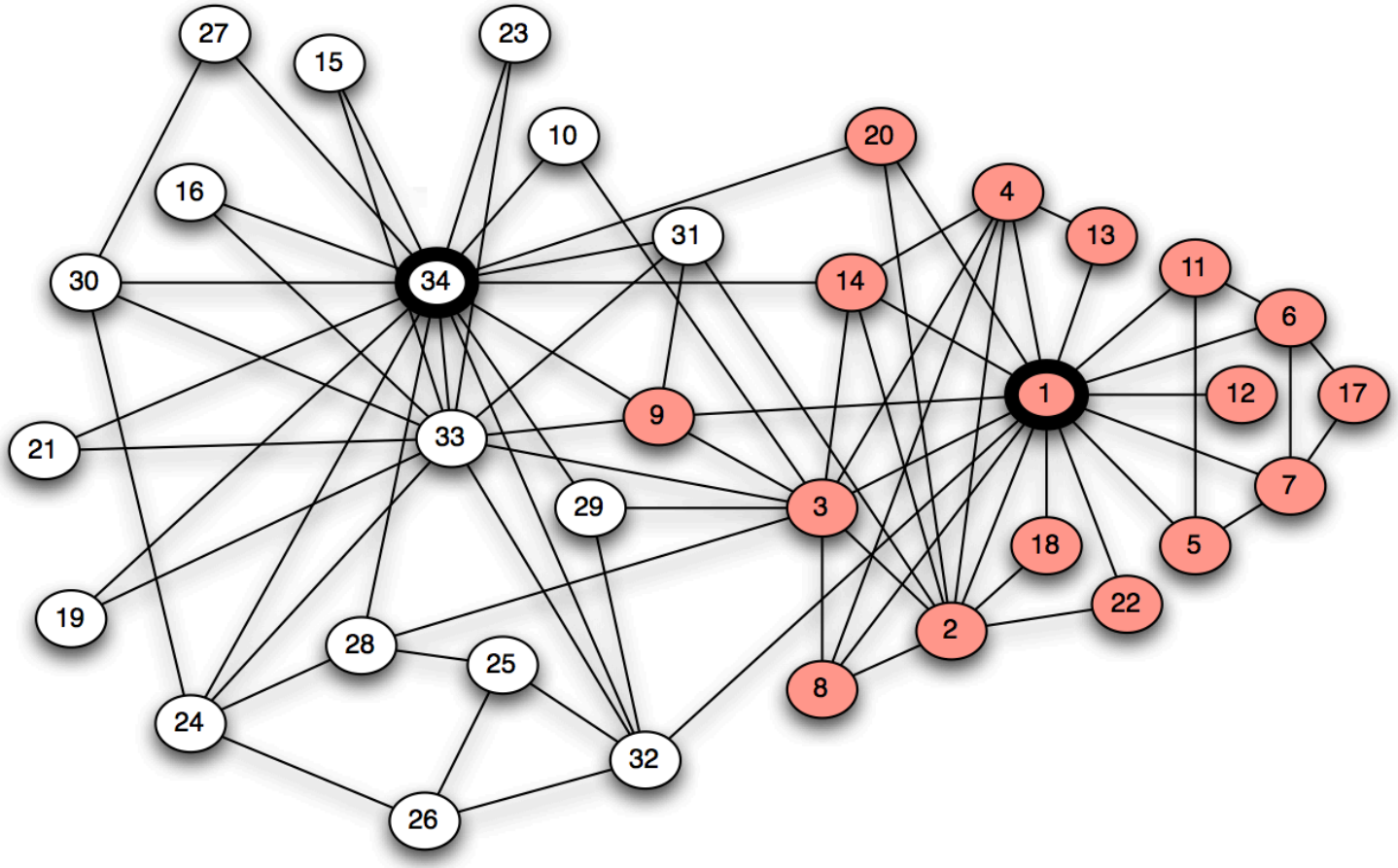
[Easley and Kleinberg]



University Karate Club

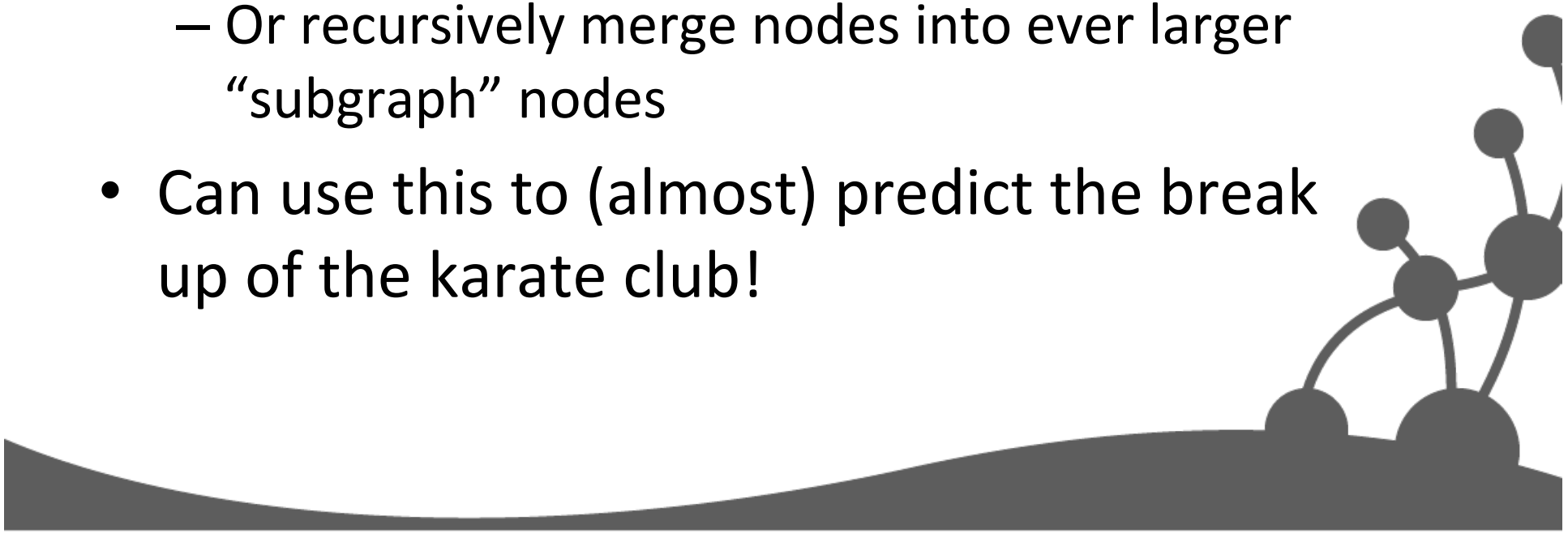


University Karate Clubs



Graph Partitioning

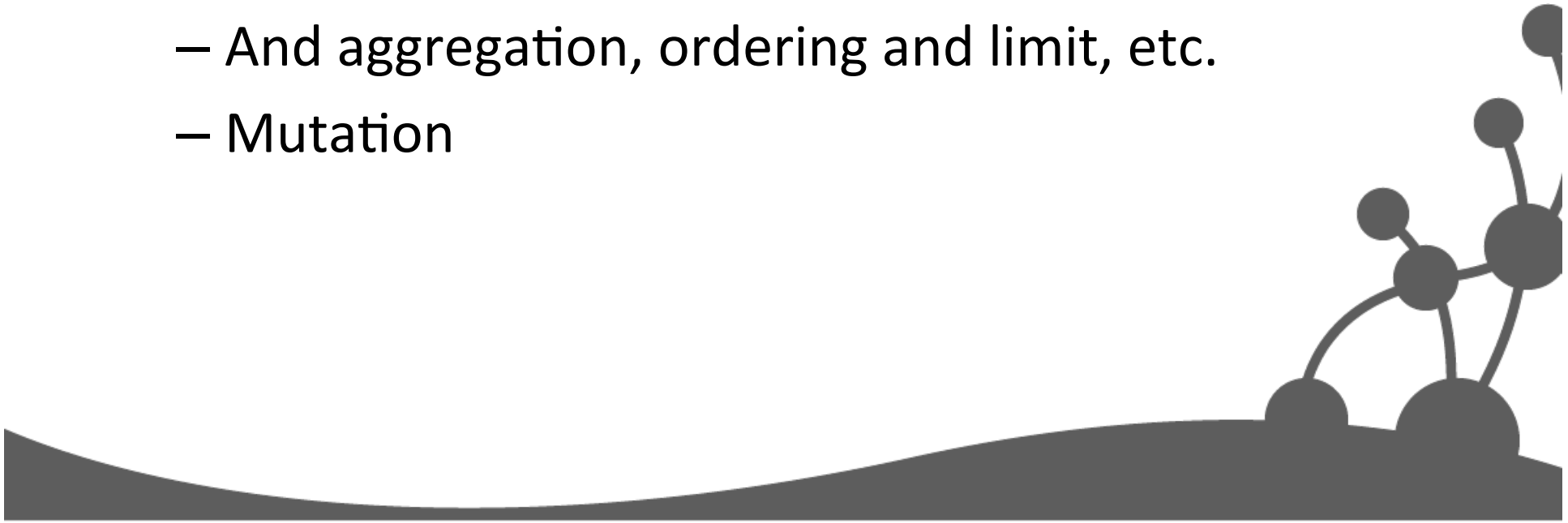
- (NP) Hard problem, so not great for massive graphs
  - Recursively remove the spanning links between dense regions
  - Or recursively merge nodes into ever larger "subgraph" nodes
- Can use this to (almost) predict the break up of the karate club!




john
crane
Ltd

Cypher

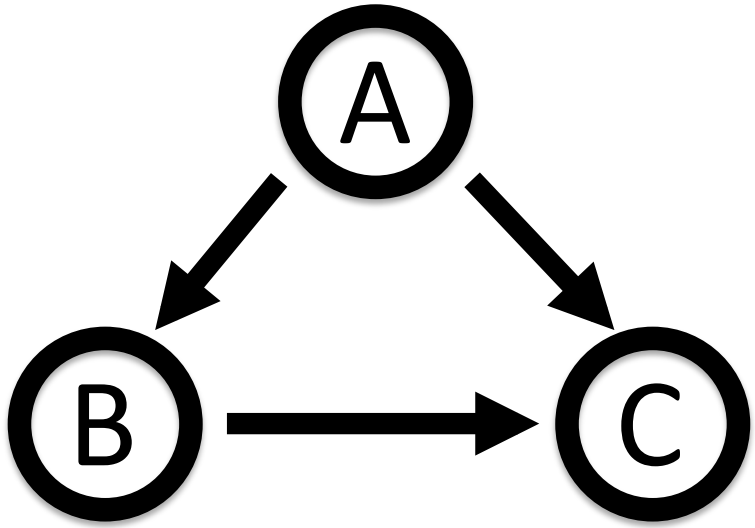
- Declarative graph pattern matching language
  - "SQL for graphs"
  - Columnar results
- Supports graph matching queries
  - And aggregation, ordering and limit, etc.
  - Mutation



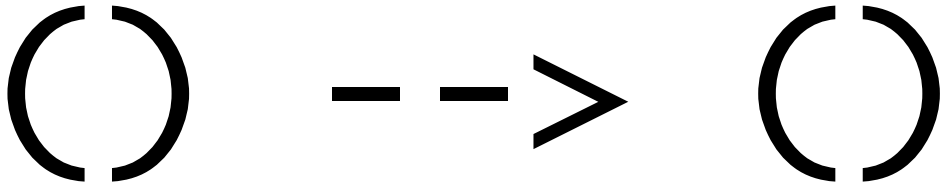
Cypher is Declarative

- Imperative
  - follow relationship
  - breadth-first vs depth-first
  - explicit algorithm
- Declarative
  - specify starting point
  - specify desired outcome
  - algorithm adaptable
  - based on query

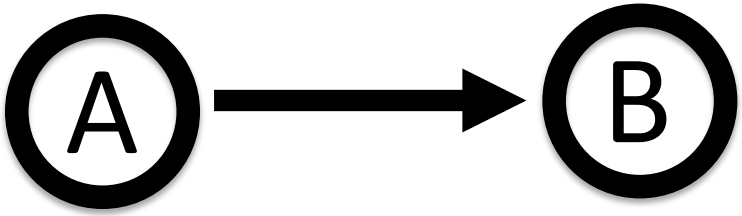
Cypher is a pattern matching language



Un-named Nodes & Rels

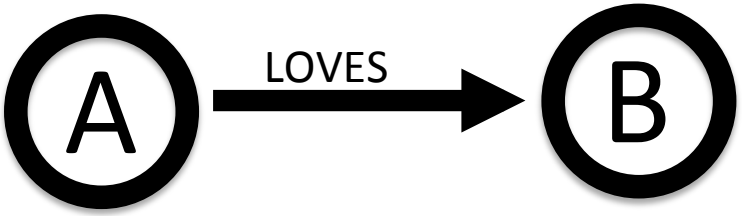


Un-named Relationship



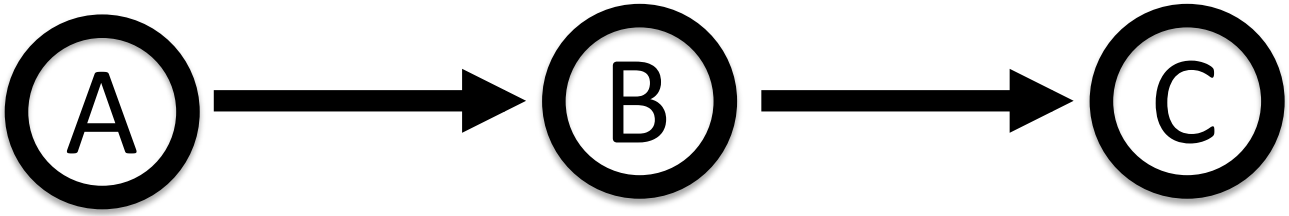
(A) --> (B)

ASCII Art Patterns

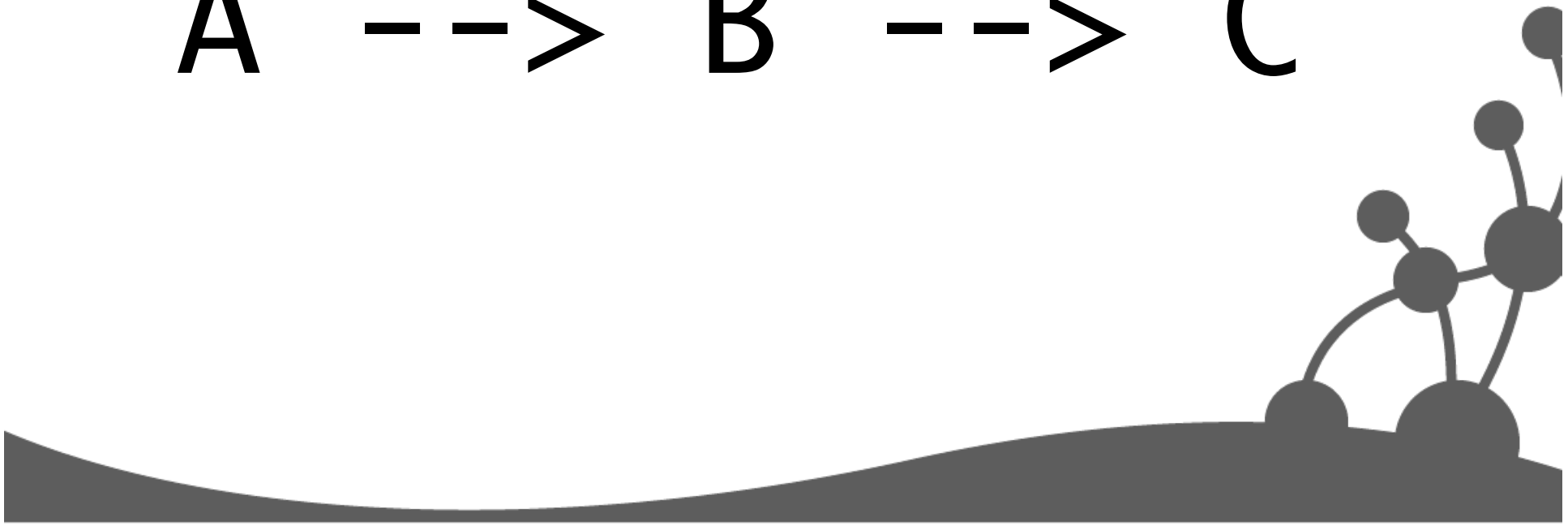



```
A -[:LOVES]-> B
```

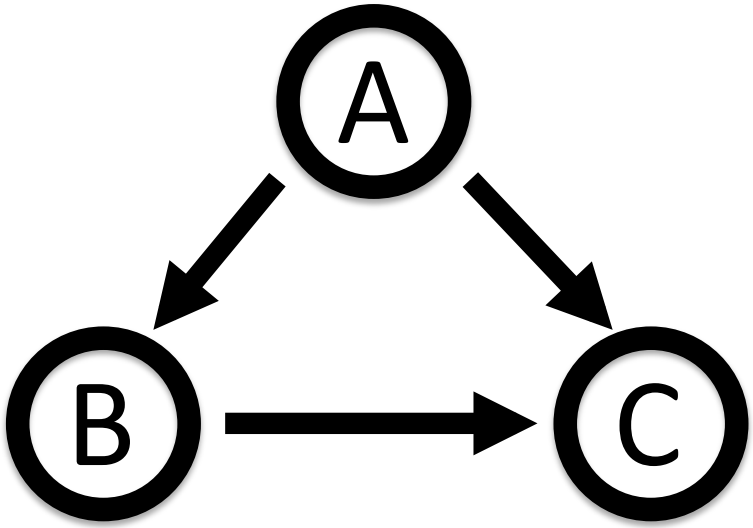
ASCII Art Patterns




```
A --> B --> C
```

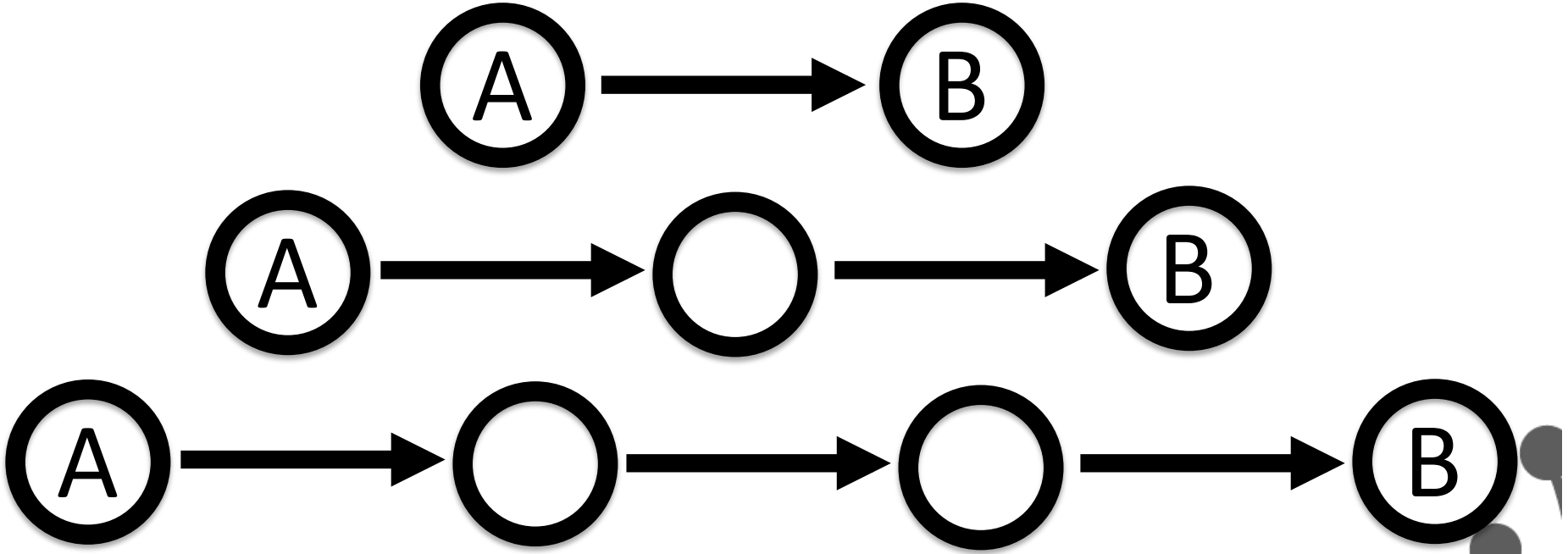


ASCII Art Patterns




```
A --> B --> C, A --> C
A --> B --> C <-- A
```

Variable Length Paths

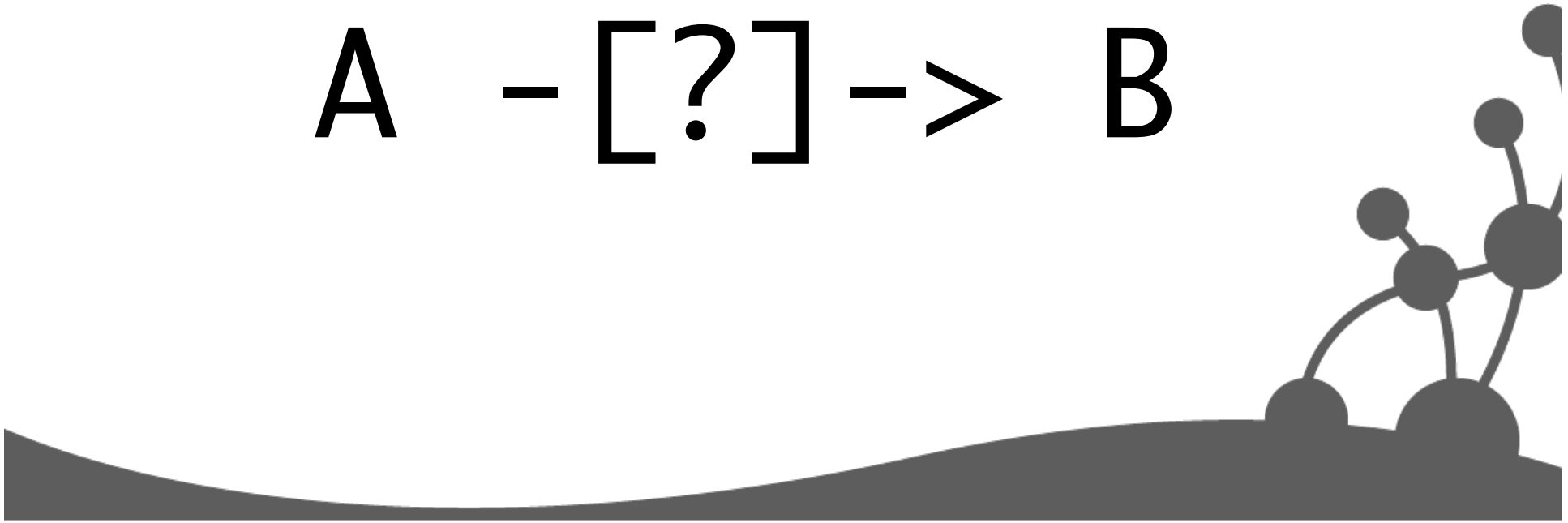



```
A -[*]-> B
```

Design Decisions

- Optional relationships


```
A -[?]-> B
```

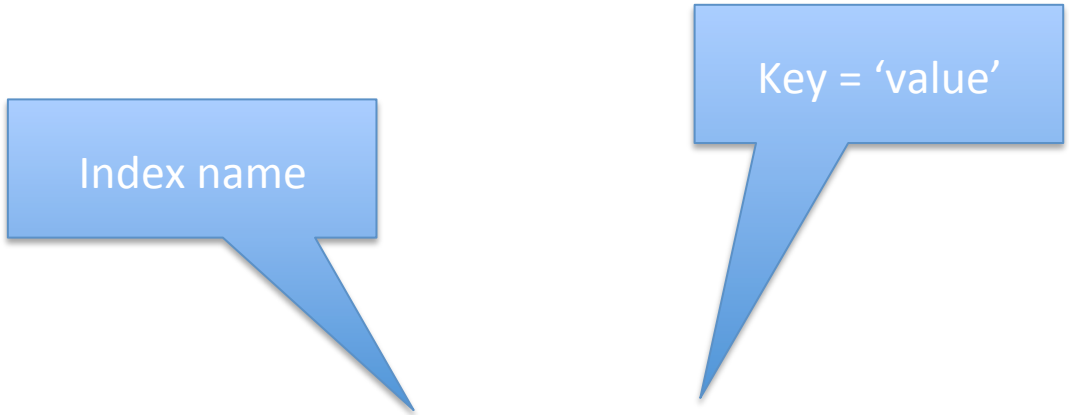


SQL Familiarity

- select
- from
- where
- group by
- order by

- start
- match
- where
- return

Using Indexes

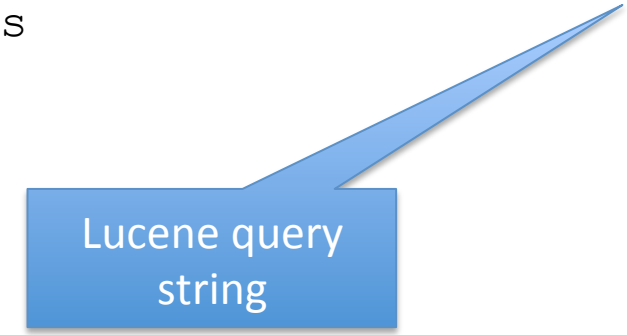


- Lookup:

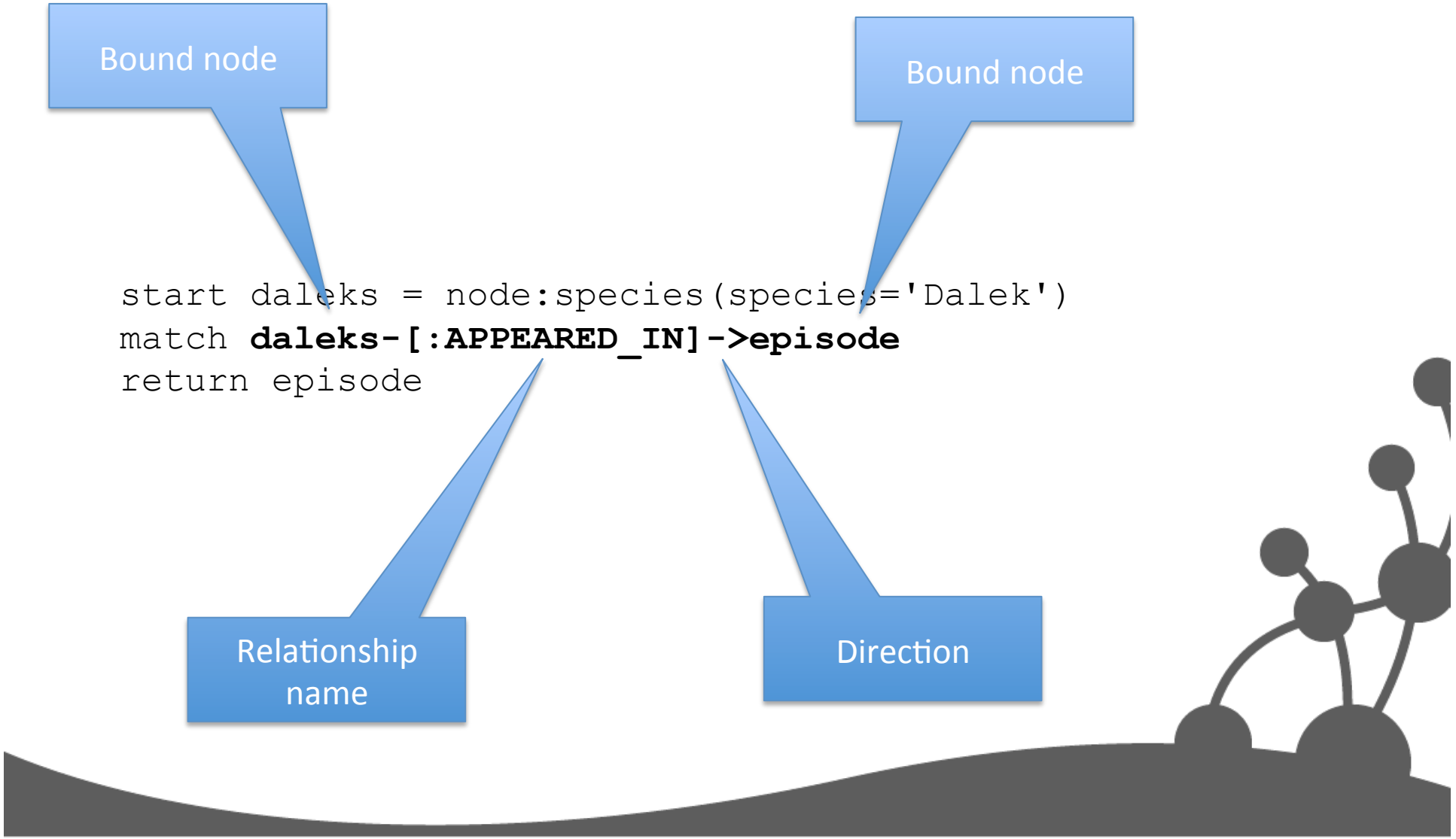
```
start daleks = node:species(species='Dalek')
return daleks
```

- Query:

```
start monsters = node:species('species:S*n')
return monsters
```



Match (ASCII art for graphs)

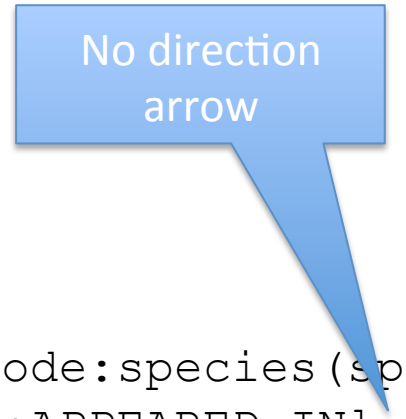


‘Circles and arrows’

Optional brackets

```
start daleks = node:species(species='Dalek')
match (daleks)-[:APPEARED_IN]->(episode)
return episode
```

Directionality




```
start daleks = node:species(species='Dalek')
match (daleks)-[:APPEARED_IN]-(episode)
return episode
```

Example Query

- The top 5 most frequently appearing companions:

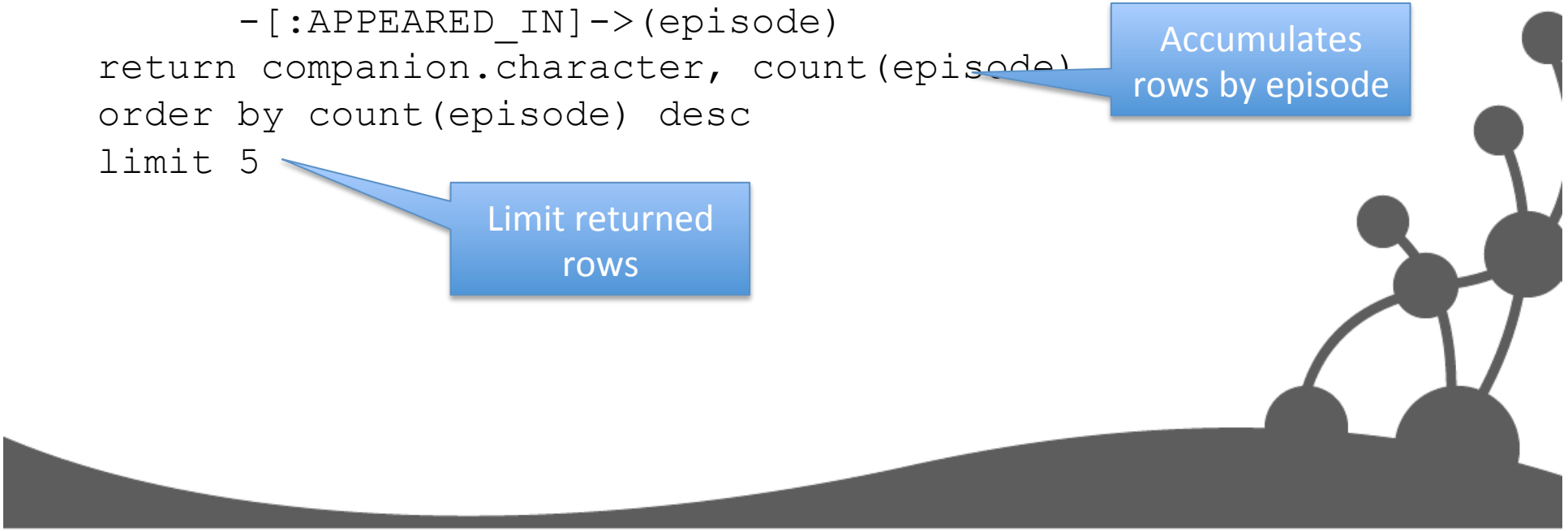
```
start doctor=node:characters(character = 'Doctor')
match (doctor)<-[:COMPANION_OF]-(companion)
      -[:APPEARED_IN]->(episode)
return companion.character, count(episode)
order by count(episode) desc
limit 5
```

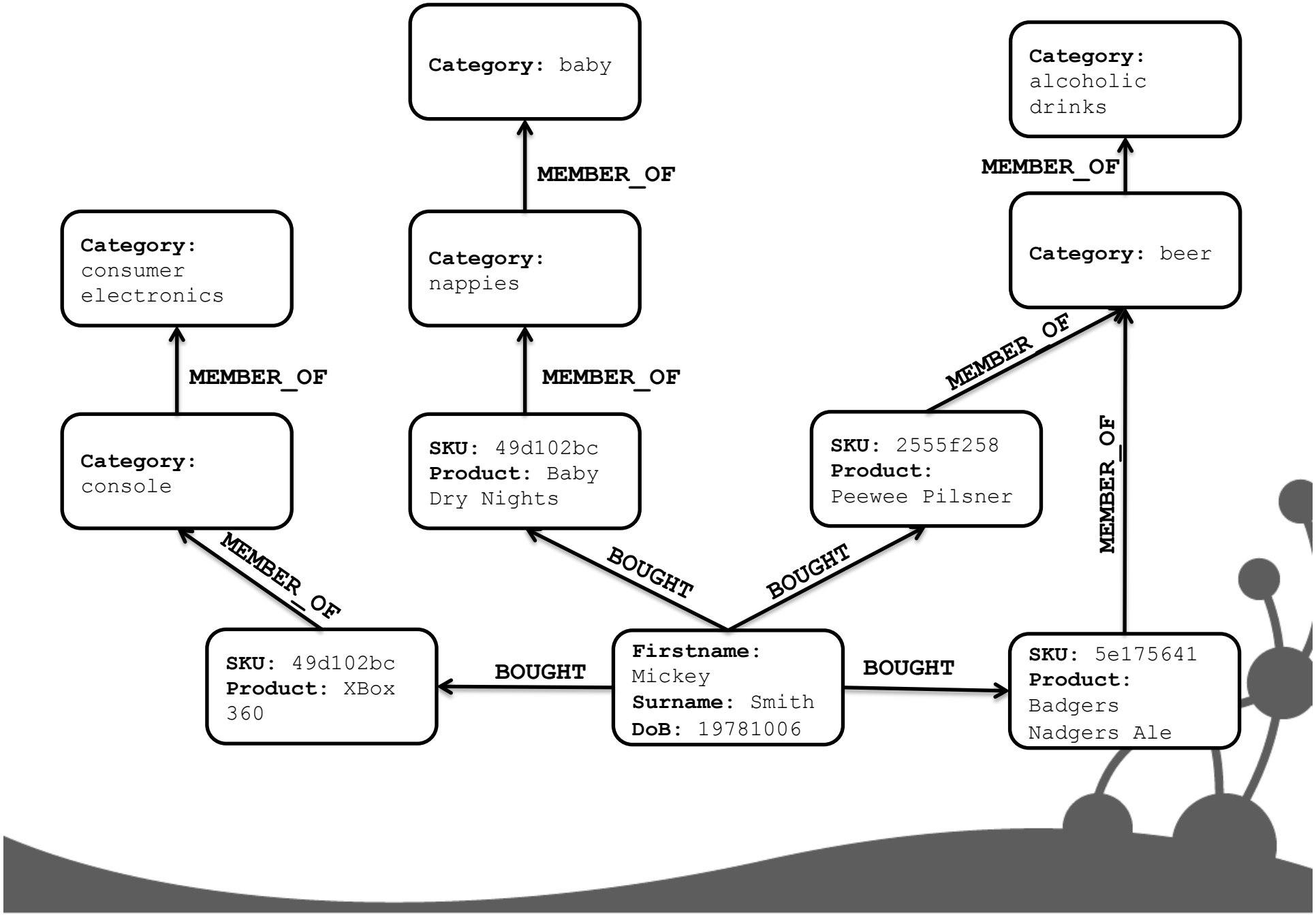
Start node from index

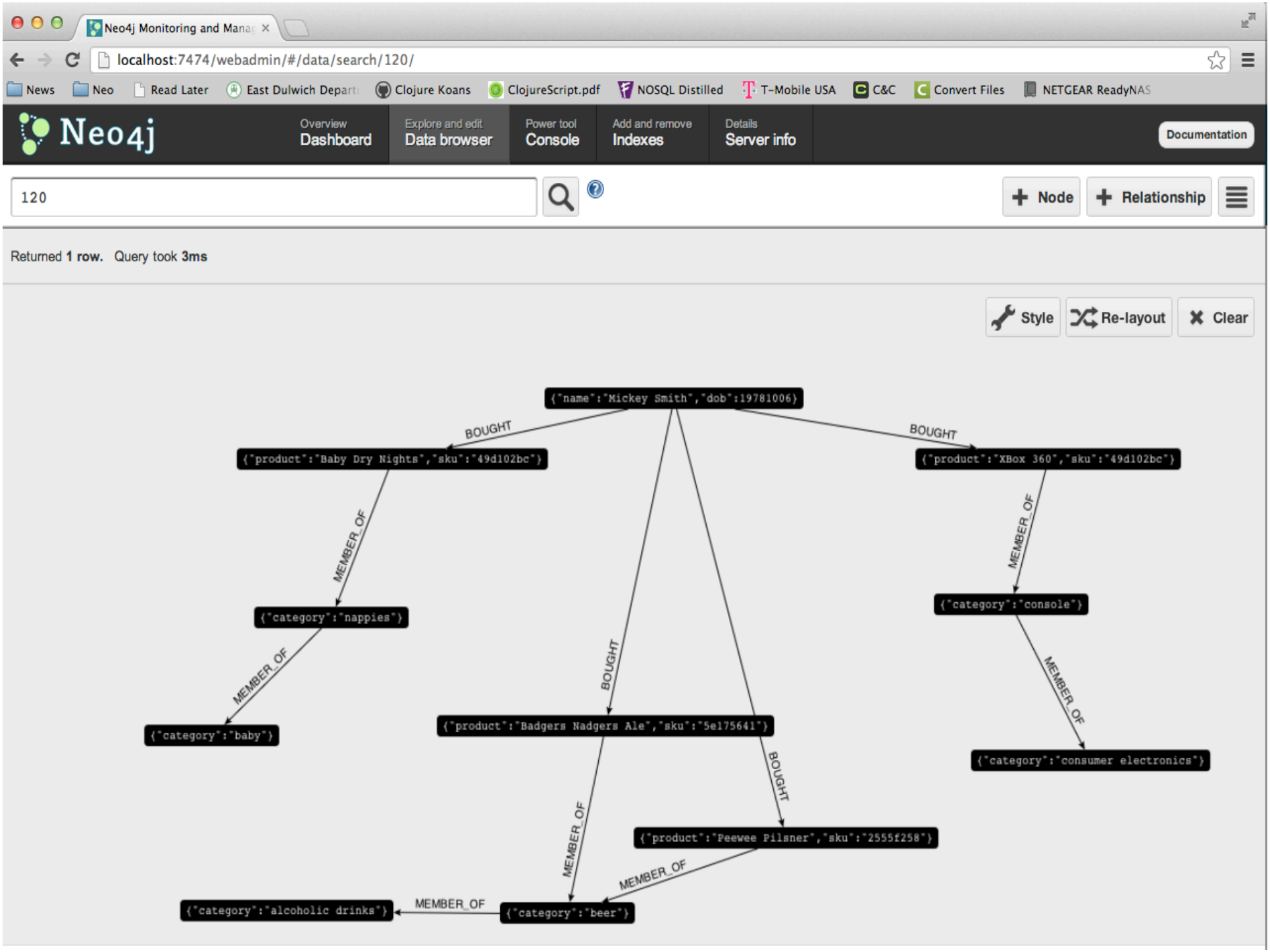
Subgraph pattern

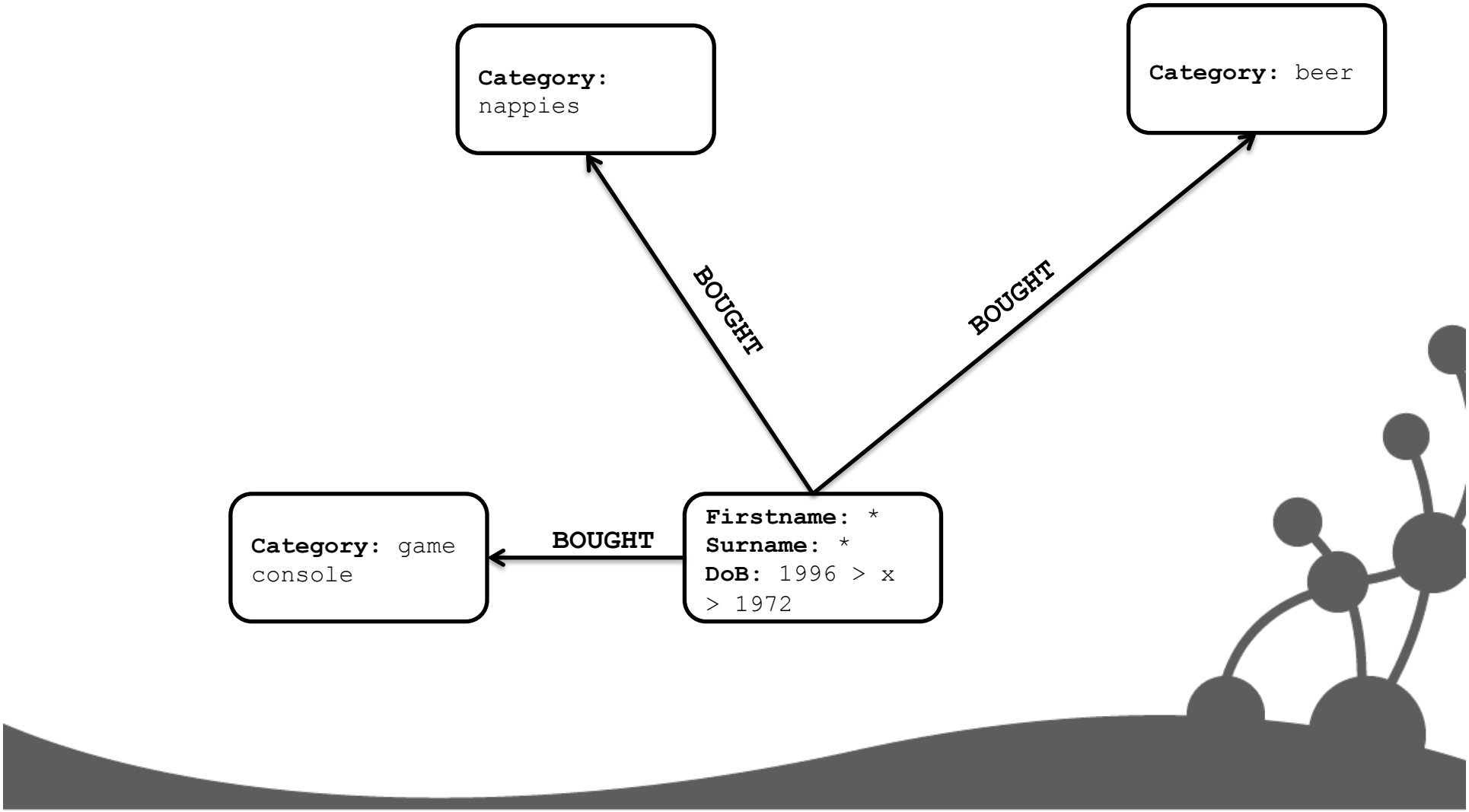
Accumulates rows by episode

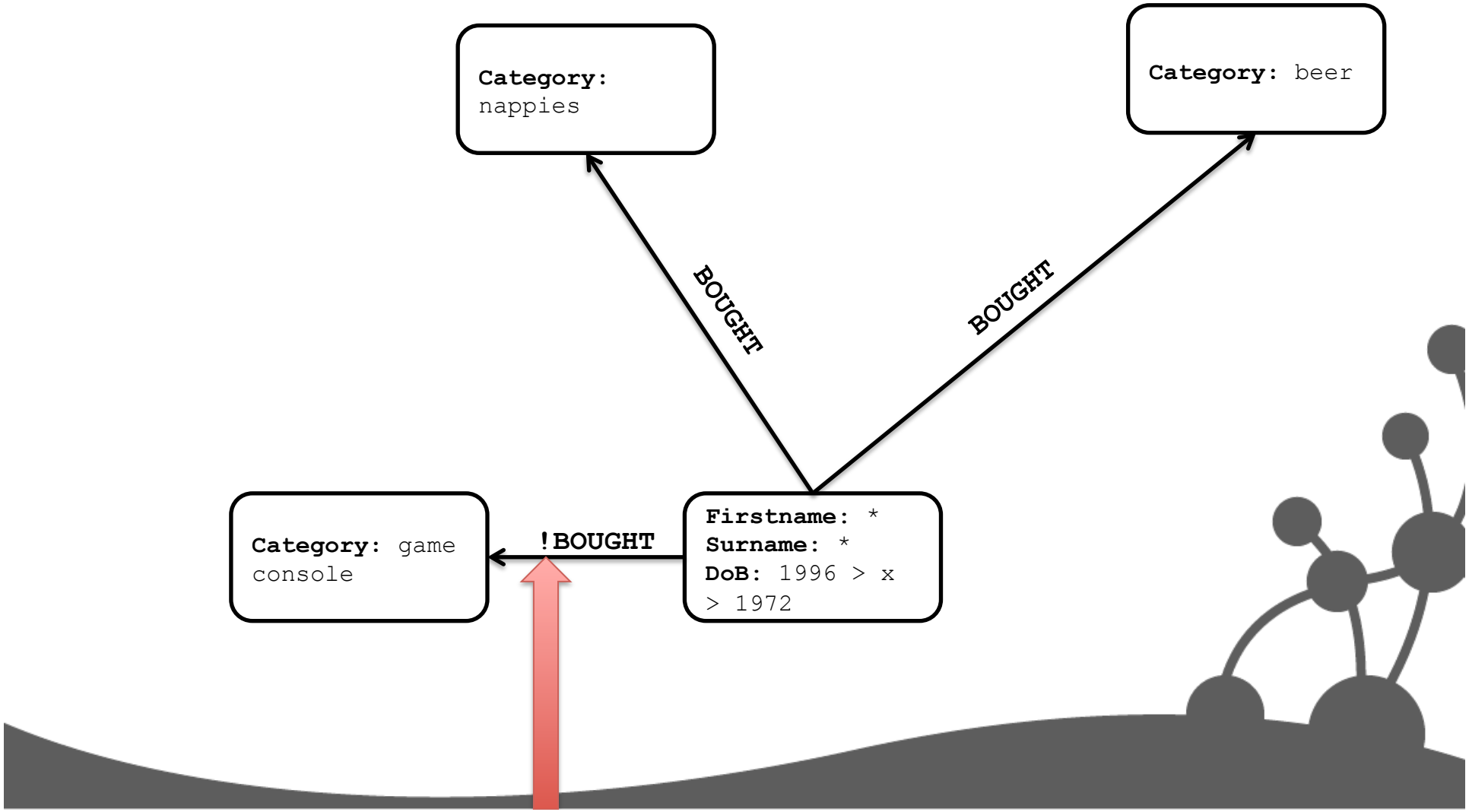
Limit returned rows

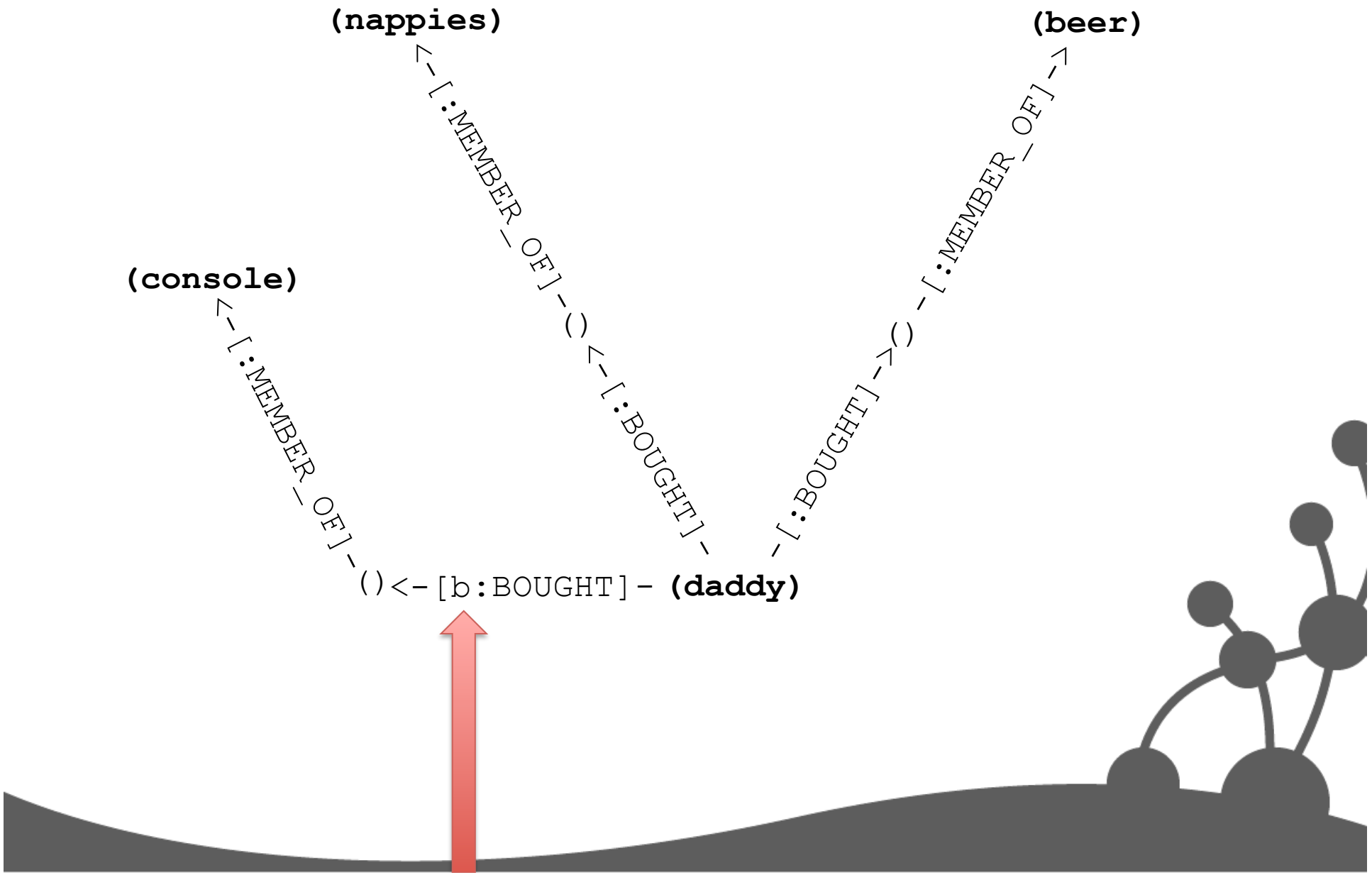



Category: baby
MEMBER_OF
Category: alcoholic drinks
MEMBER_OF
Category: consumer electronics
Category: nappies
Category: beer
MEMBER_OF
MEMBER_OF
MEMBER_OF
Category: console
SKU: 49d102bc Product: Baby Dry Nights
SKU: 2555f258 Product: Peewee Pilsner
MEMBER_OF
BOUGHT
BOUGHT
MEMBER_OF
SKU: 49d102bc Product: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


Neo4j Monitoring and Mana
localhost:7474/webadmin/#/data/search/120/
News
Neo
Read Later
East Dulwich Depart
Clojure Koans
ClojureScript.pdf
NOSQL Distilled
T-Mobile USA
C&C
Convert Files
NETGEAR ReadyNAS
Neo4j
Overview
Dashboard
Explore and edit
Data browser
Power tool
Console
Add and remove
Indexes
Details
Server info
Documentation
120
Node
Relationship
Returned 1 row. Query took 3ms
Style
Re-layout
Clear
{"name":"Mickey Smith","dob":19781006}
BOUGHT
BOUGHT
BOUGHT
{"product":"Baby Dry Nights","sku":"49d102bc"}
{"product":"XBox 360","sku":"49d102bc"}
{"product":"Badgers Nadgers Ale","sku":"5e175641"}
{"product":"Peewee Pilsner","sku":"2555f258"}
MEMBER_OF
MEMBER_OF
MEMBER_OF
MEMBER_OF
MEMBER_OF
MEMBER_OF
{"category":"nappies"}
{"category":"console"}
{"category":"baby"}
{"category":"consumer electronics"}
{"category":"alcoholic drinks"}
{"category":"beer"}


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]-
<-[:BOUGHT]-
-[:BOUGHT]->
-[:MEMBER_OF]->
<-[:MEMBER_OF]-
()<-[b:BOUGHT]- (daddy)

Flatten the graph

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

Wrap in a Cypher MATCH clause

```
MATCH (daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(nappies),
(daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(beer),
(daddy)-[b:BOUGHT]->()-[:MEMBER_OF]->(console)
```

Cypher WHERE clause

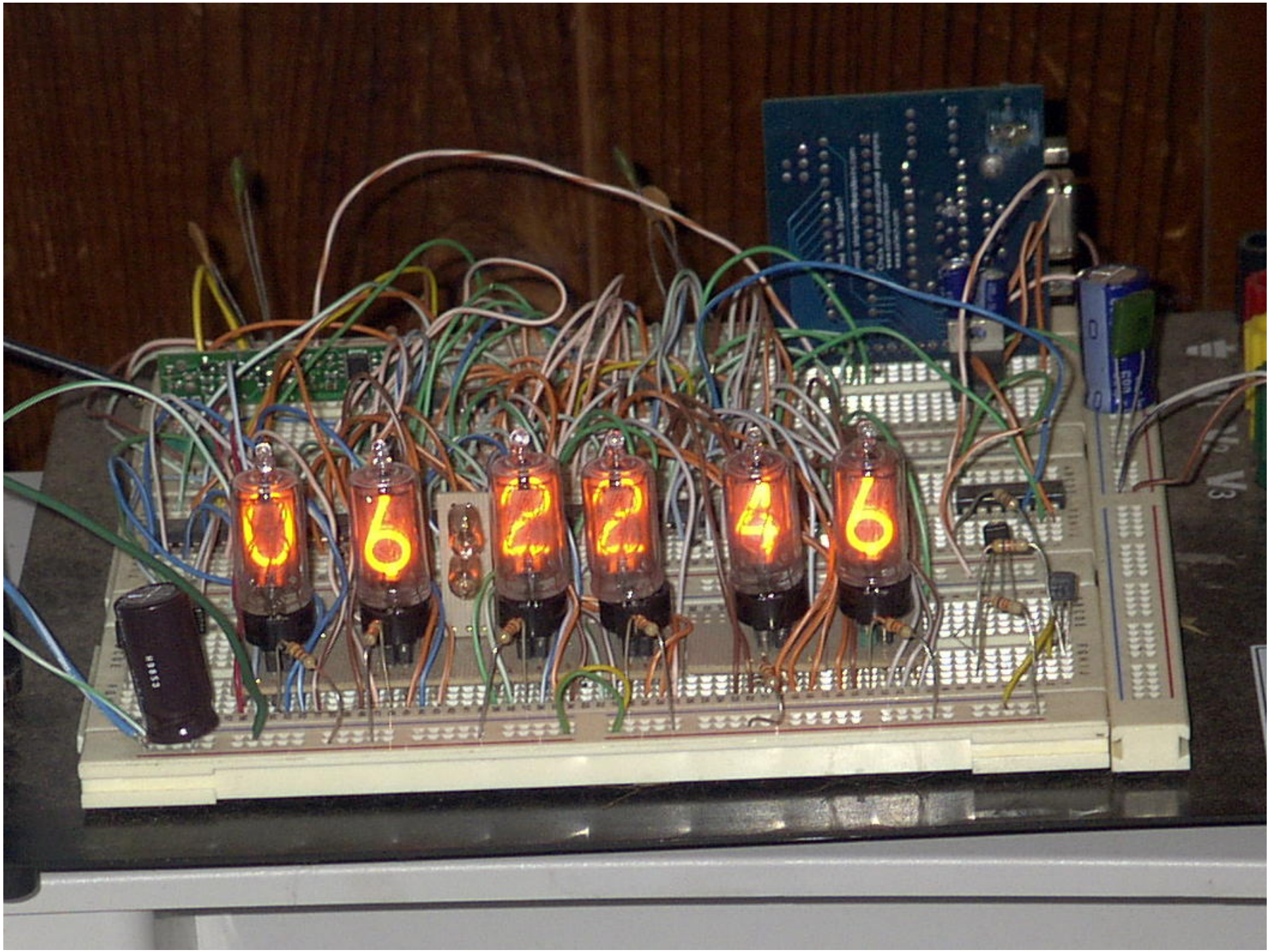
```
MATCH (daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(nappies),
(daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(beer),
(daddy)-[b:BOUGHT]->()-[:MEMBER_OF]->(console)
WHERE b is null
```

Full Cypher query

```
START beer=node:categories(category='beer'),
  nappies=de:categories(category='nappies'),
  xbox=node:products(product='xbox 360')

MATCH (daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(beer),
  (daddy)-[:BOUGHT]->()-[:MEMBER_OF]->(nappies),
  (daddy)-[b?:BOUGHT]->(xbox)

WHERE b is null

RETURN distinct daddy
```

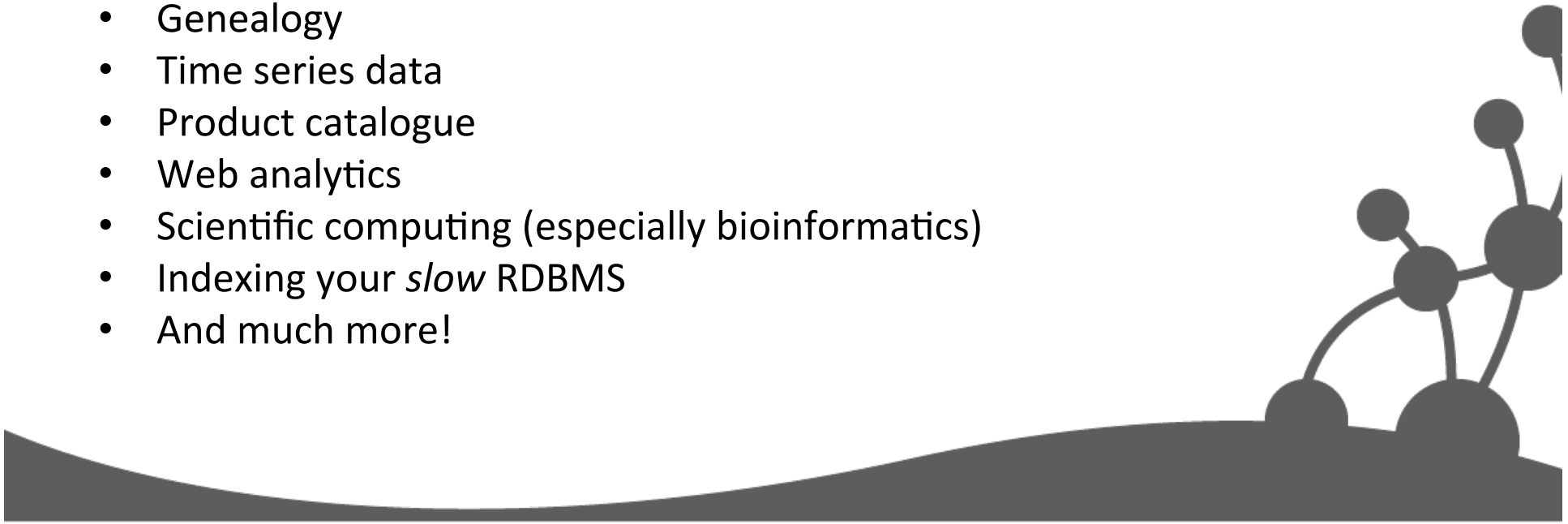
Results

```
==> +-------------------------------------------+
==> | daddy                                     |
==> +-------------------------------------------+
==> | Node[15]{name:"Rory Williams",dob:19880121} |
==> +-------------------------------------------+
==> 1 row
==> 6 ms
==>
neo4j-sh (0)$
```


0 6 2 2 4 6

What are graphs good for?

- Recommendations
- Business intelligence
- Social computing
- Geospatial
- MDM
- Systems management
- Web of things
- Genealogy
- Time series data
- Product catalogue
- Web analytics
- Scientific computing (especially bioinformatics)
- Indexing your *slow* RDBMS
- And much more!



Thanks for listening

Neo4j: **http://neo4j.org**

Neo Technology: **http://neotechnology.com**

Me: **@jimwebber**