A Little Graph Theory for the Busy Developer

Dr. Jim Webber
Chief Scientist, Neo Technology
@jimwebber
Roadmap

• Imprisoned data
• Labeled Property Graph Model
  – And some cultural similarities
• Graph theory
  – South East London
  – World War I
• Graph matching
  – Beer, diapers and Xbox
• End
“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 = \( f(size, connectedness, uniformity) \)
DENORMALISE
Aggregate data into documents

RICHER MODEL
Connected structured data

Simple data model
Map-reduce friendly

Expressive power
Fast graph traversals
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
stole from love of enemy
A Good Man Goes to War
appeared in

loves companion
appeared in

loves companion
appeared in

appears in enemy

appears in enemy

appears in companion

appears in companion

appears in enemy

appears in enemy

appears in

appears in

appears in

Victory of the Daleks
Meet Leonhard Euler

• Swiss mathematician
• Inventor of Graph Theory (1736)
Königsberg (Prussia) - 1736
Triadic Closure

name: Kyle

name: Stan

name: Kenny
Triadic Closure

name: Kyle

name: Stan

name: Kenny

name: Kyle

name: Stan

name: Kenny
Structural Balance

name: Cartman

FRIEND

name: Craig

ENEMY

name: Tweek
Structural Balance

name: Cartman

name: Craig  name: Tweek

FRIEND  ENEMY

name: Cartman

name: Craig  name: Tweek

FRIEND  ENEMY
Structural Balance

- name: Cartman
  - FRIEND: Craig
  - ENEMY: Tweek
  - FRIEND: ENEMY
    - FRIEND: Cartman
    - ENEMY: Tweek
    - ENEMY: Craig
Structural Balance is a *key* predictive technique

And it’s domain-agnostic
Allies and Enemies

- UK
- Austria
- France
- Germany
- Russia
- Italy
Allies and Enemies

- UK
- Austria
- France
- Russia
- Germany
- Italy

Connections:
- UK to Austria
- Austria to Germany
- Germany to Italy
- Italy to Russia
- Russia to France
- France to UK
Allies and Enemies

- UK
- Austria
- France
- Germany
- Russia
- Italy
Allies and Enemies

UK

Austria

France

Russia

Germany

Italy
Allies and Enemies

- UK
- Austria
- France
- Germany
- Russia
- Italy

Connections:
- UK to Austria
- UK to France
- UK to Russia
- Austria to France
- Austria to Germany
- France to Russia
- France to Italy
- Russia to Italy
- Germany to Italy
Predicting WWI

[By 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)

name: Kenny

name: Stan

name: Cartman
Triadic Closure
(weak relationship)

name: Kenny

name: Stan
name: Cartman

name: Kenny

name: Stan
name: Cartman

FRIEND 50%
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

- **name:** Cartman
  - ENEMY
  - FRIEND 50%

- **name:** Kyle
  - FRIEND
  - FRIEND

- **name:** Stan
  - FRIEND

- **name:** Kenny
  - FRIEND

- **name:** Sally
  - FRIEND 50%

- **name:** Wendy
  - FRIEND

- **name:** Bebe
  - FRIEND
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]
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 Club(s) (predicted by Graph Theory)
University Karate Clubs
(what actually happened!)
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...
Firstname: *
Surname: *
DoB: 1996 > x
> 1972

Category: game console

Category: nappies

Category: beer

BOUGHT

BOUGHT

BOUGHT

!BOUGHT
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))
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>daddy</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Node[15] {name: &quot;Rory Williams&quot;, dob: 19880121}</strong></td>
<td></td>
</tr>
</tbody>
</table>

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 (1.x) query

START me=node:person(name = 'Jim'),
    location=node:location(location='New York'),
    cuisine=node:cuisine(cuisine='Sushi')

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

RETURN restaurant
Cypher query is trivial!

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

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

RETURN restaurant
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...
Free O’Reilly book!
(if you come talk to me or attend tomorrow’s Neo4j meetup)

Or visit http://graphdatabases.com for the eBook version
NEO4J

ALL THE THINGS!

Thanks for listening
@jimwebber