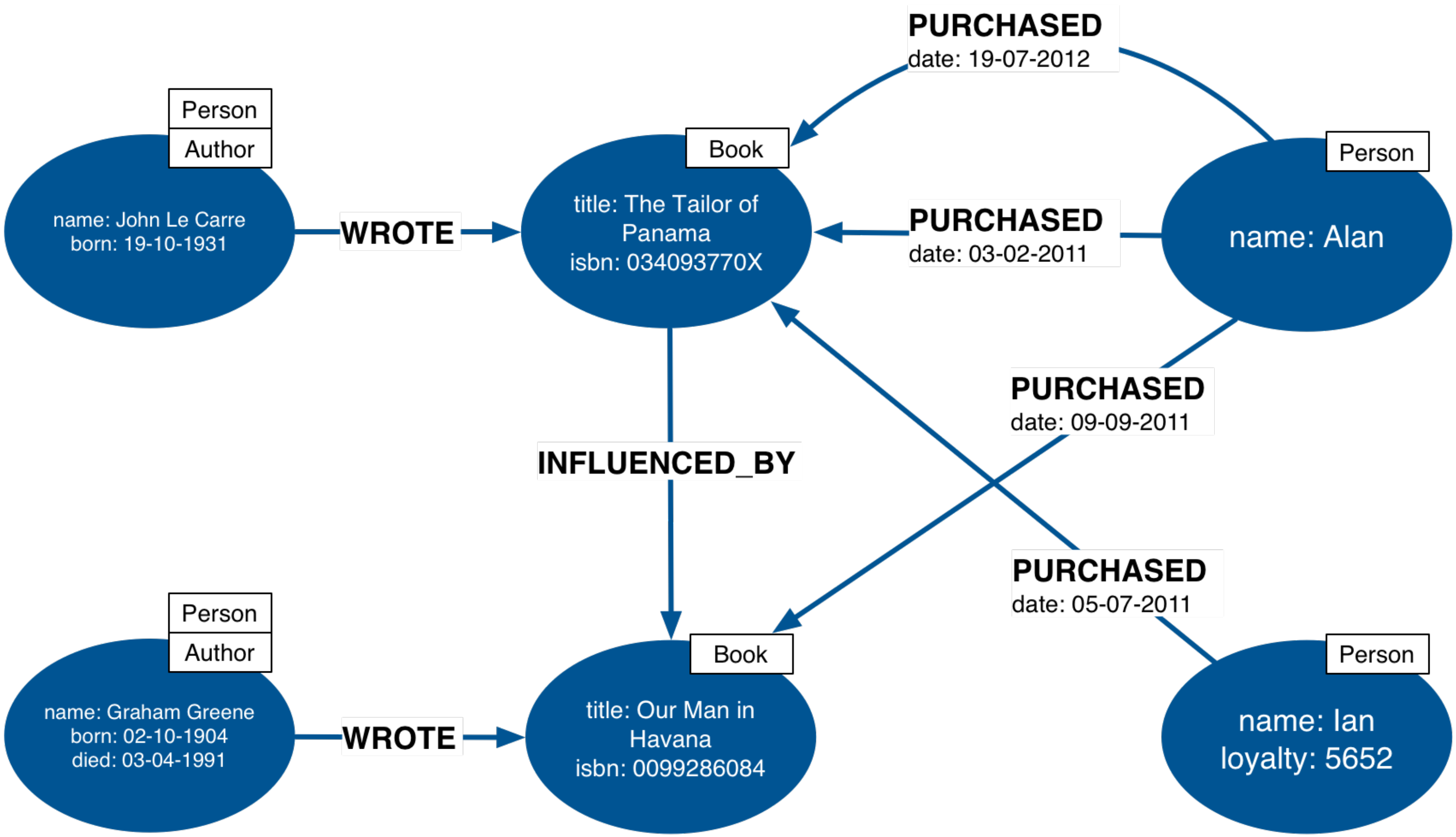
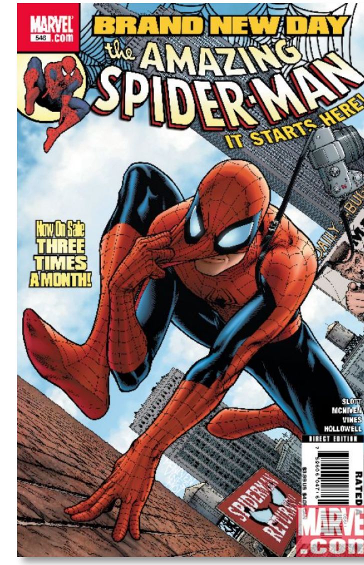
New Opportunities for ConnectedData

@ianSrobinson
ian@neotechnology.com

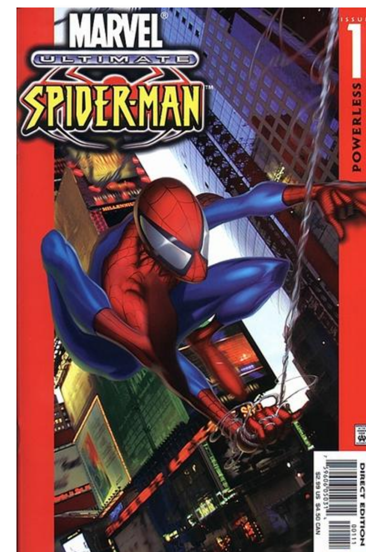
Neo4j Graph Database



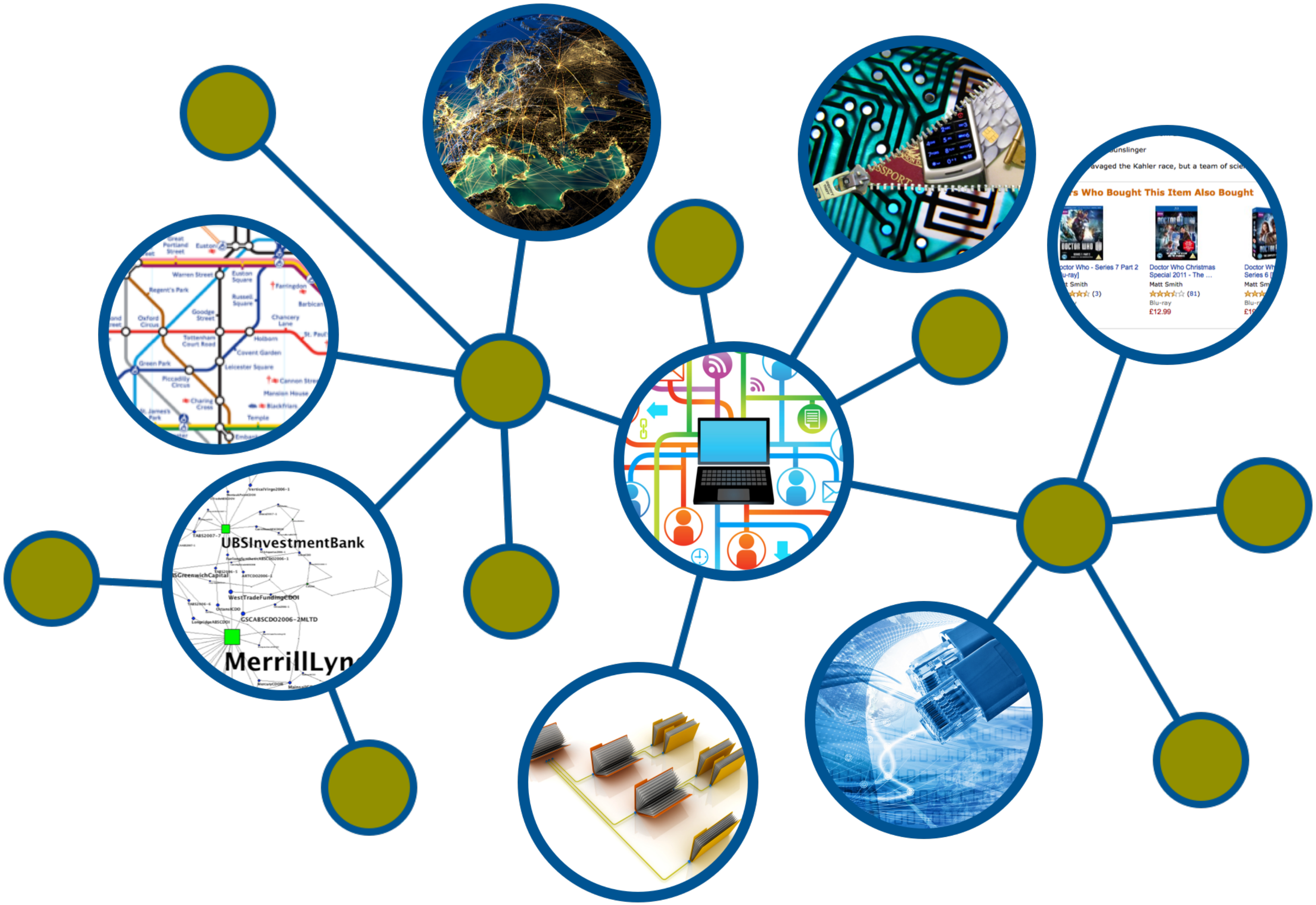

AMAZING
FANTASY
12¢
15 AUG.
THOUGH THE WORLD MAY MOCK PETER PARKER, THE TIMID TEEN-AGER...
INTRODUCING SPIDER MAN
...IT WILL SOON MARVEL AT THE AWESOME MIGHT OF... SPIDER-MAN!
ALSO IN THIS ISSUE! AN IMPORTANT MESSAGE TO YOU, FROM THE EDITOR--ABOUT THE NEW AMAZING!


MARVEL
BRAND NEW DAY
the AMAZING SPIDER-MAN
IT STARTS HERE!
Now On Sale THREE TIMES A MONTH!
MARVEL.com


MARVEL Comics
1ST ALL-NEW COLLECTOR'S ITEM ISSUE!
SPIDER-MAN
THE LEGEND OF THE ARACHKNIGHT
TORMENT
PART ONE OF FIVE

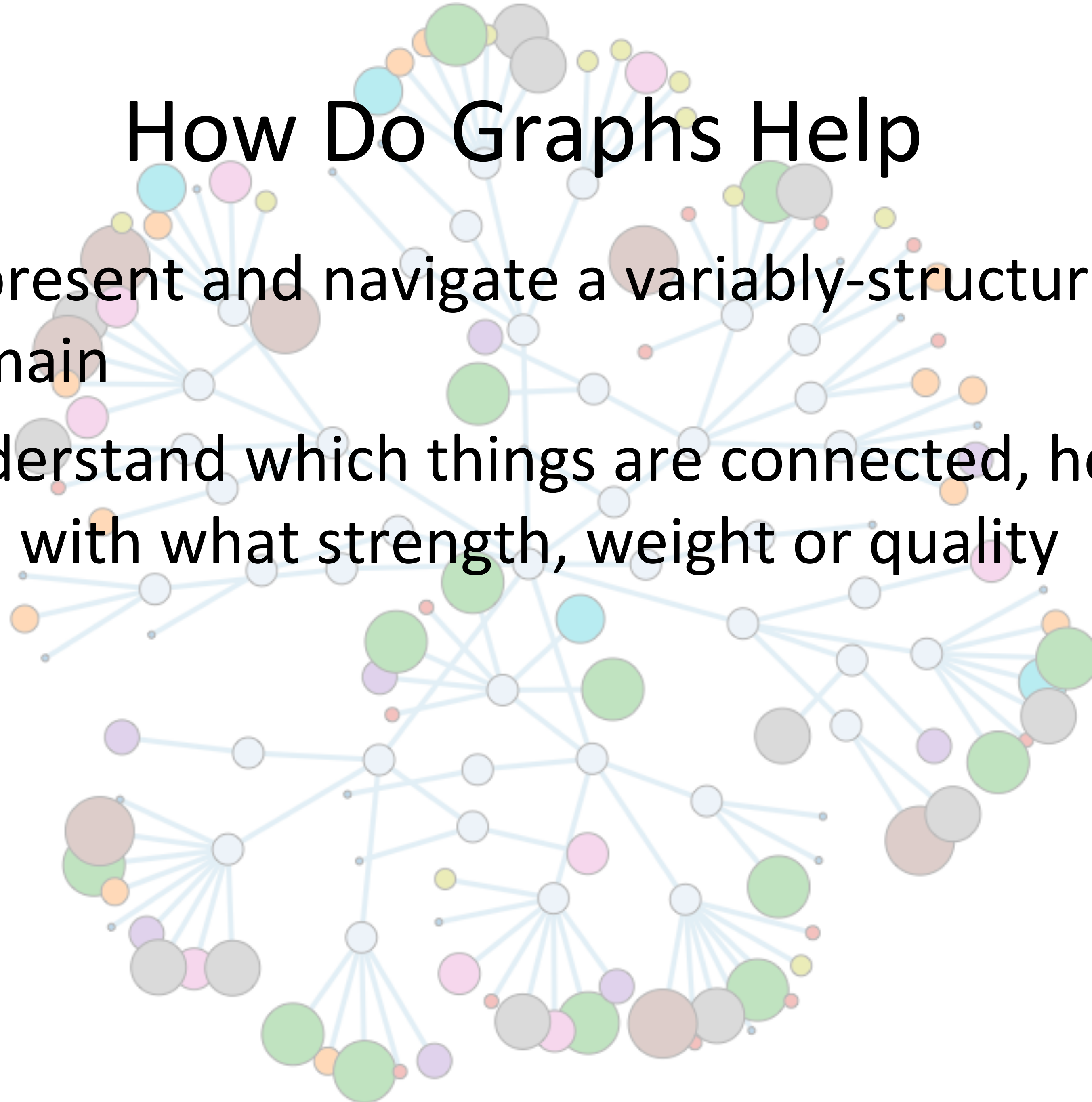

MARVEL
ULTIMATE
SPIDER-MAN
ISSUE 1
POWERLESS

complexity = f(size, variable structure, connectedness)



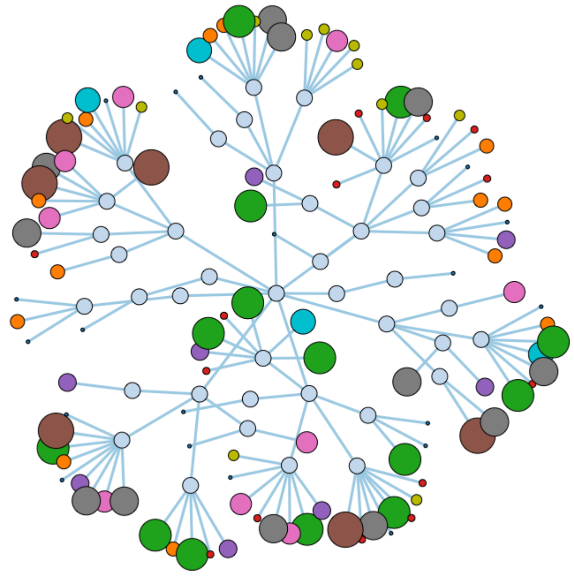
How Do Graphs Help

- Represent and navigate a variably-structured domain
- Understand which things are connected, how, and with what strength, weight or quality



Variable Structure

- Relationships provide structure
- Importantly, they are defined with regard to node *instances*, not *classes* of nodes



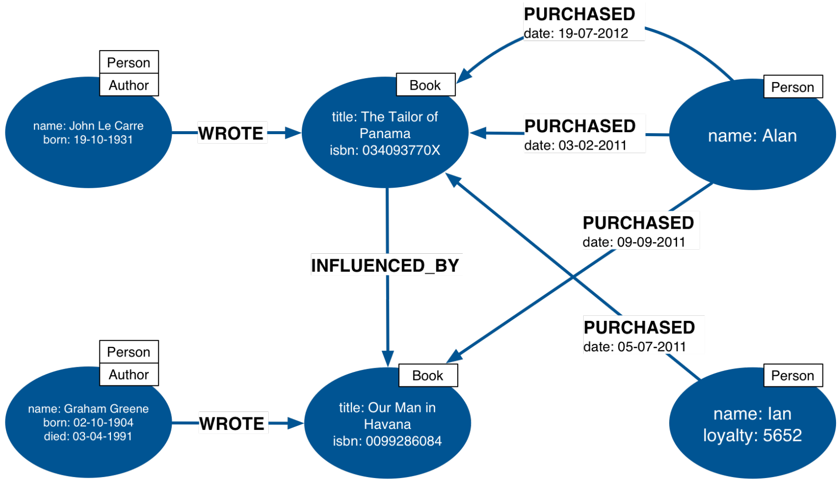
Connectedness

Relationship Names

- Semantics first-class element in data model

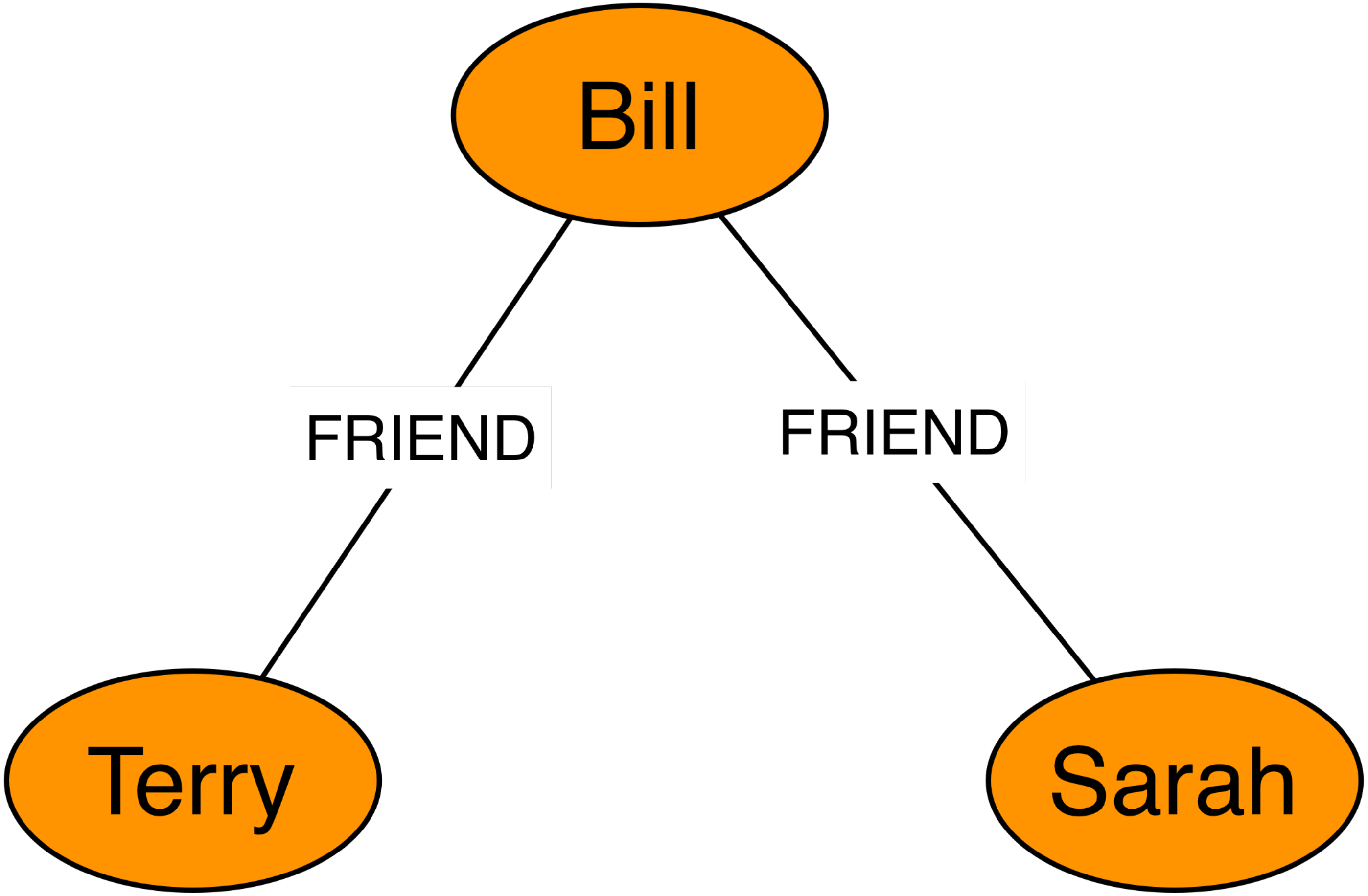
Relationship Properties

- Describe weight, strength or quality of a relationship

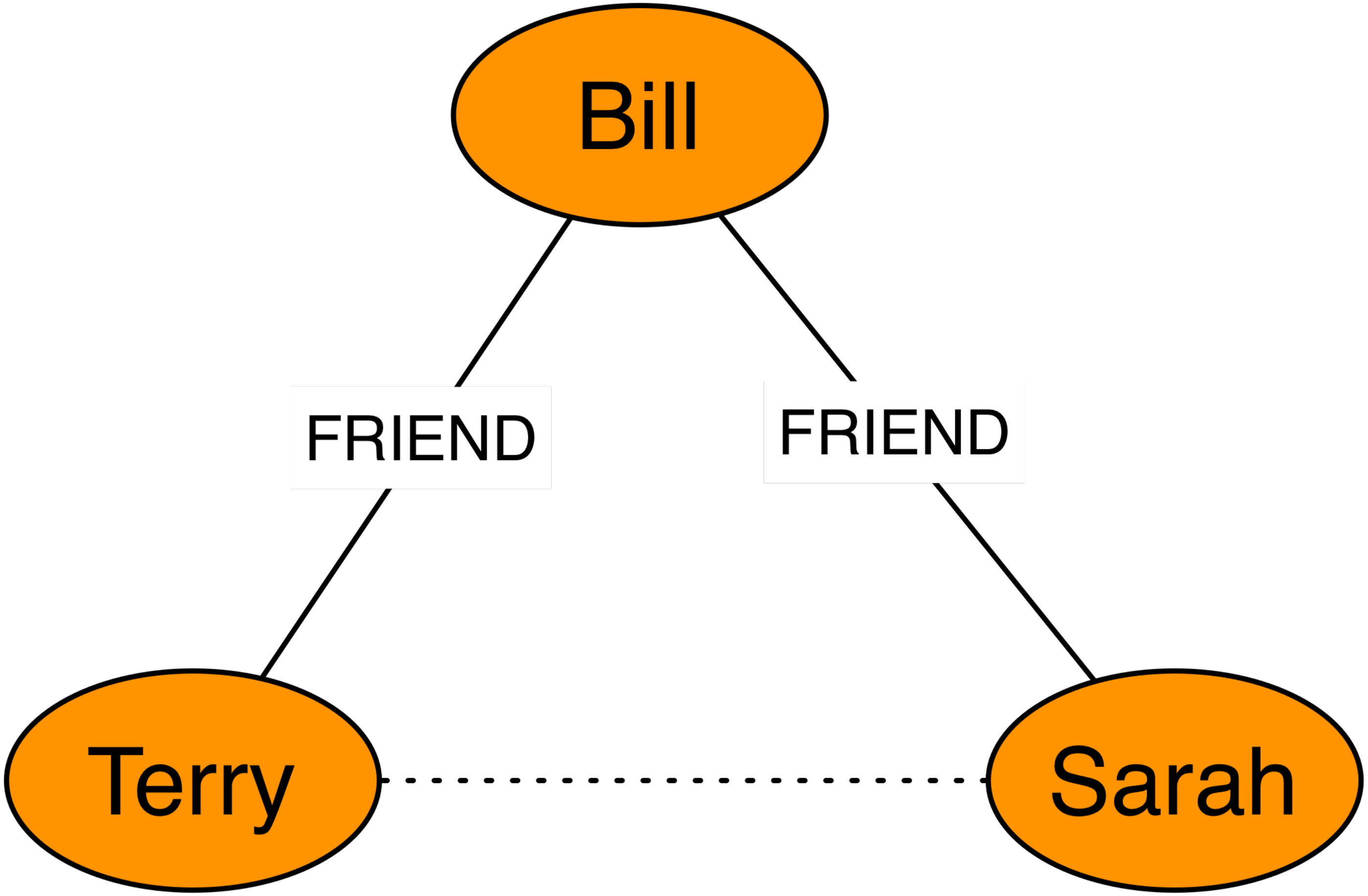


Making Connections

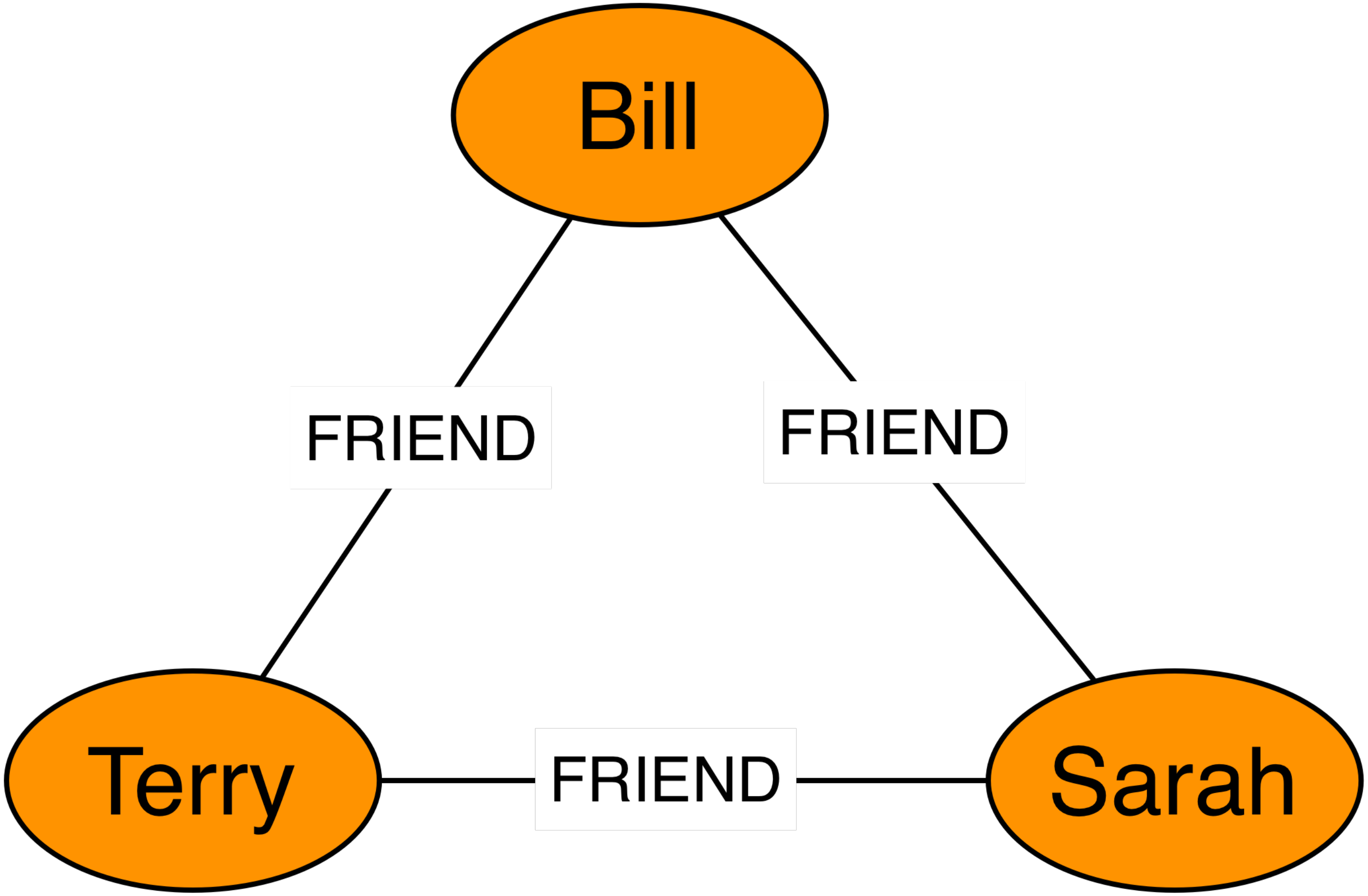
Triadic Closure – Closing Triangles



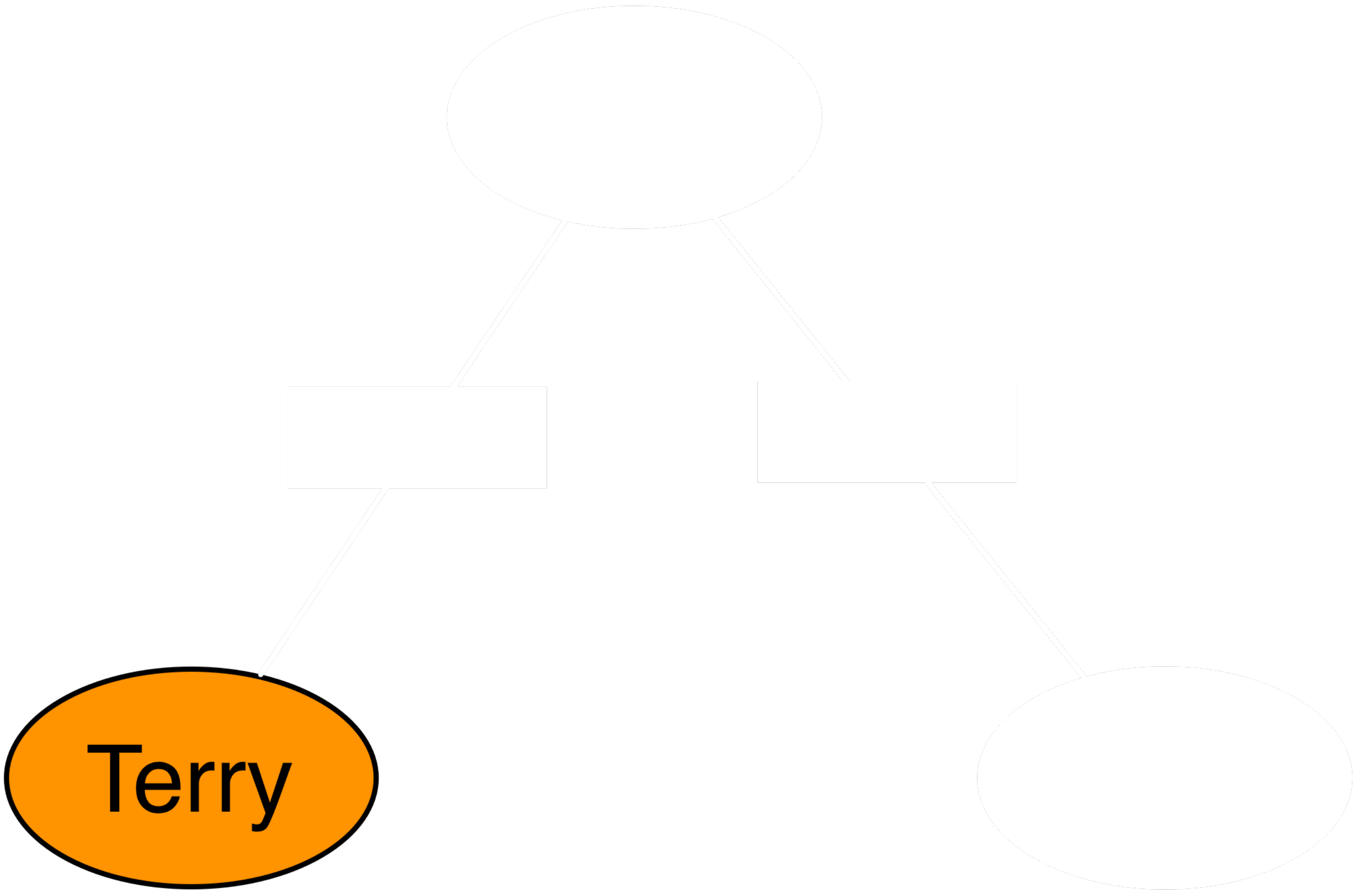
Triadic Closure – Closing Triangles



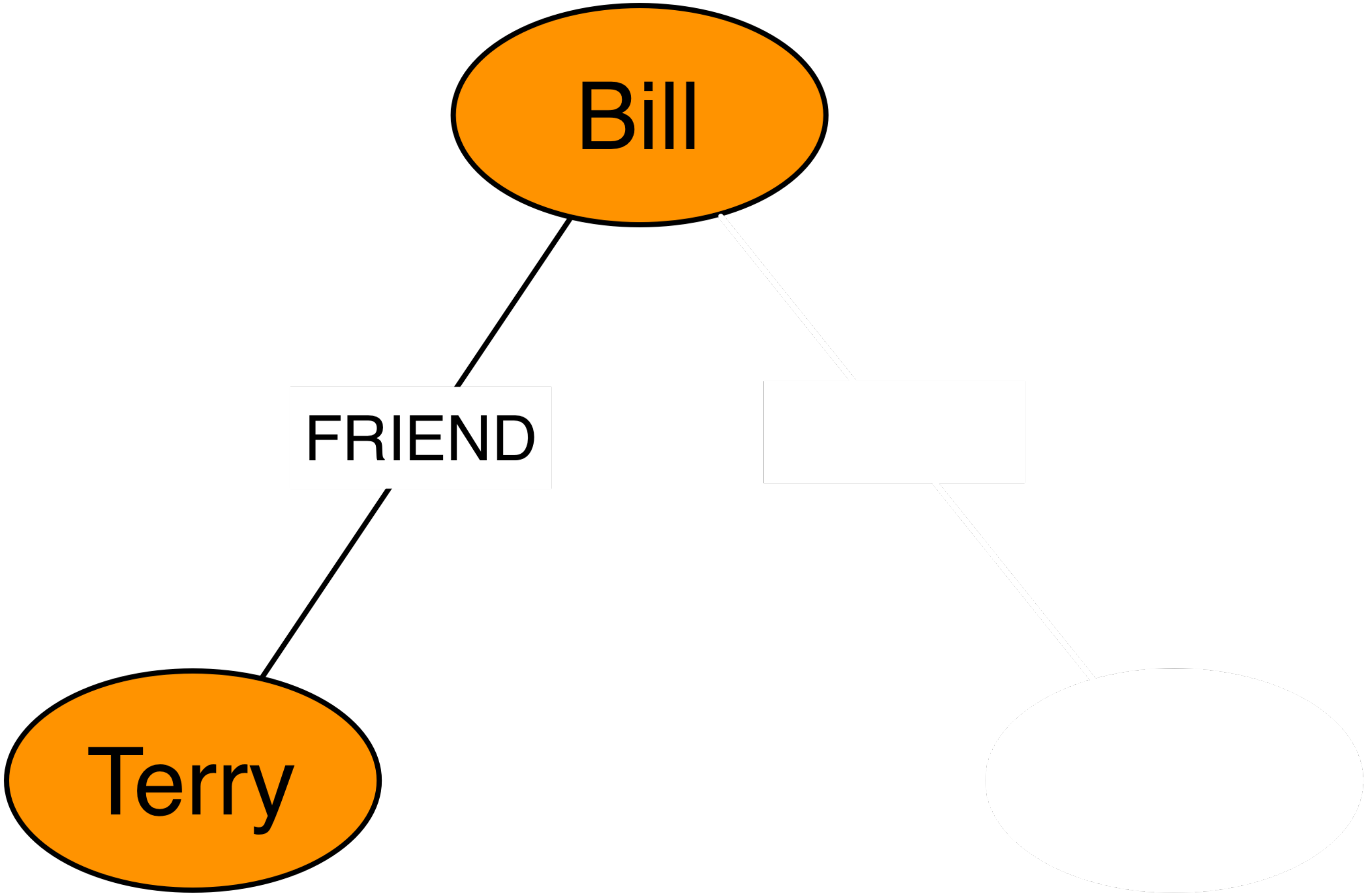
Triadic Closure – Closing Triangles



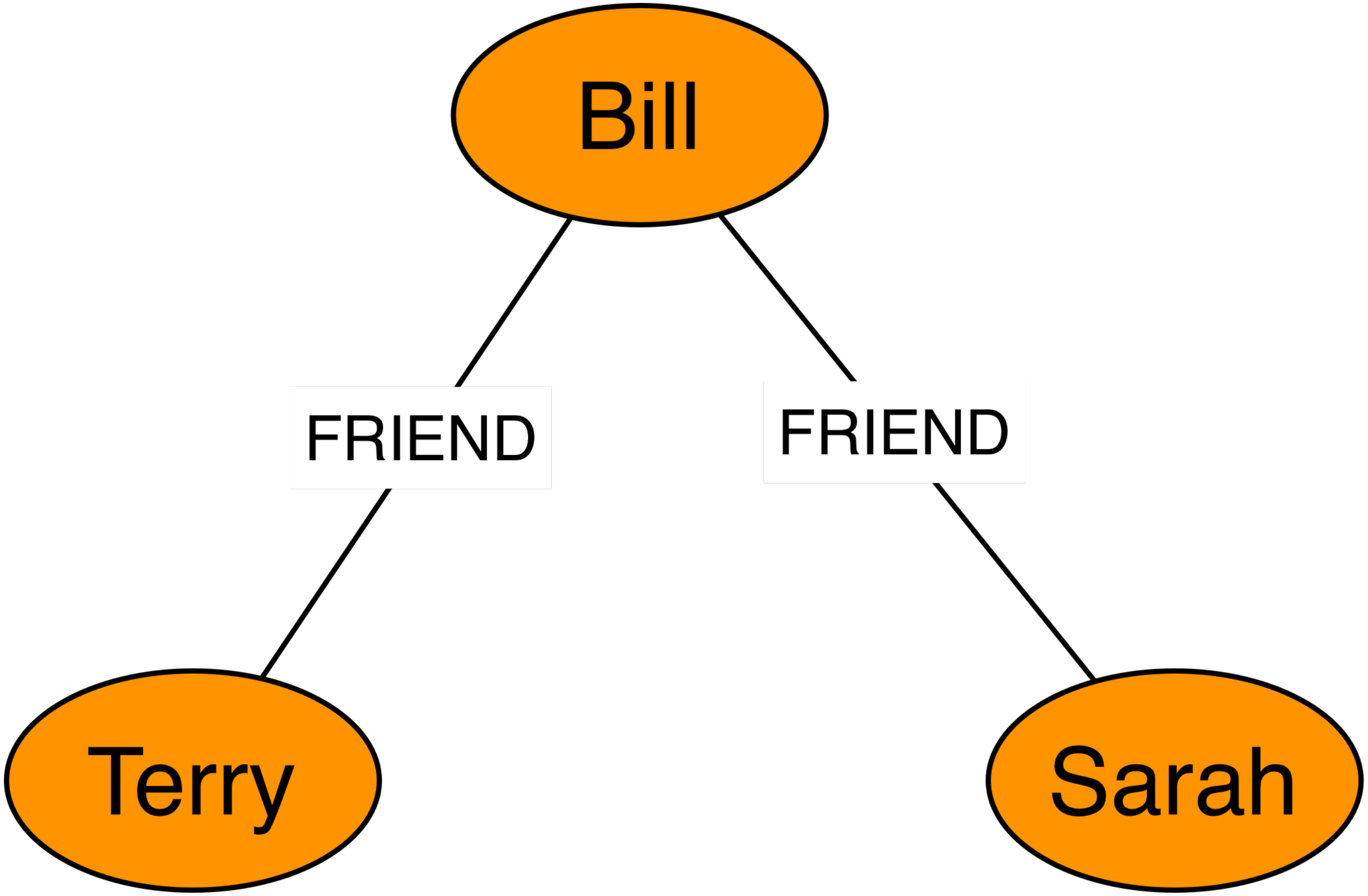
Recommending New Connections



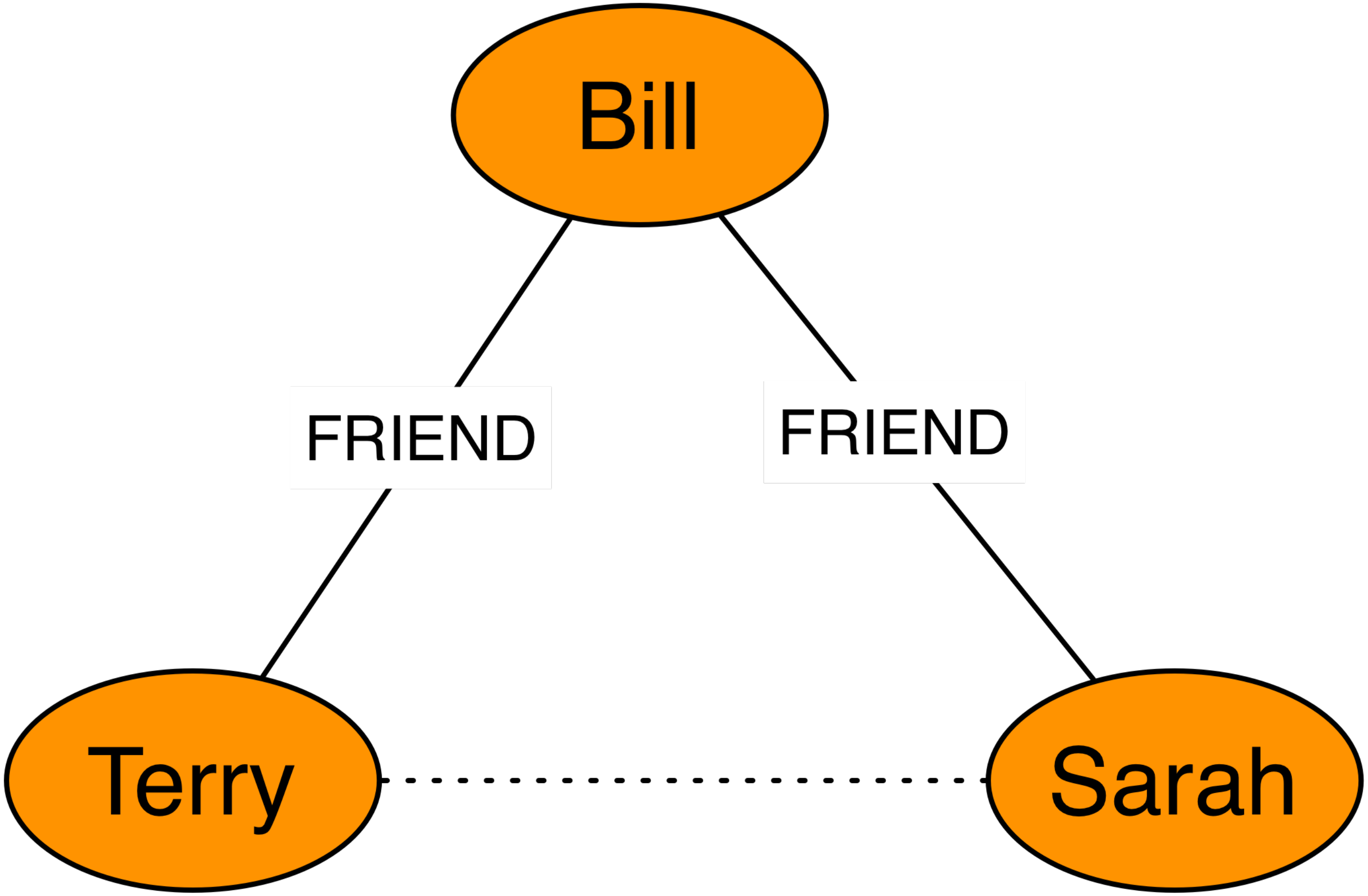
Immediate Friendships



Means and Motive



Recommendation



Recommend New Connections

```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

Find Terry

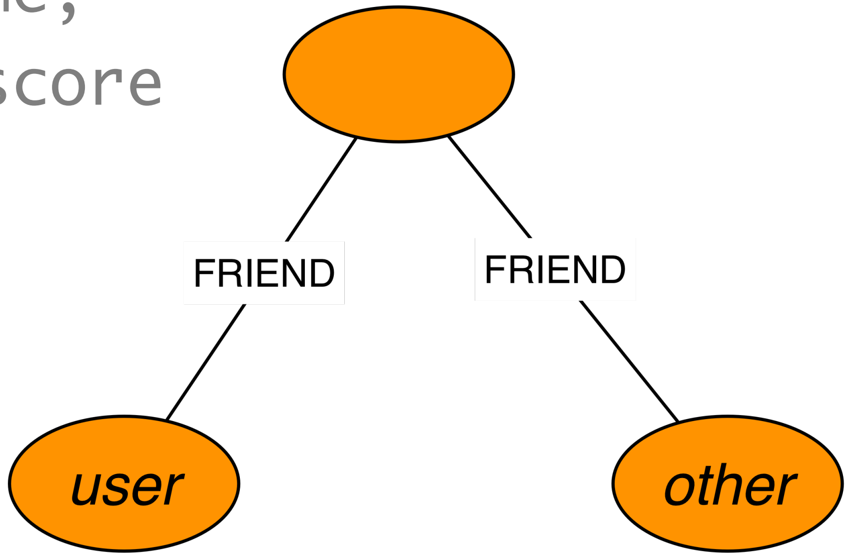
```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

Find Terry's Friends' Friends

```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

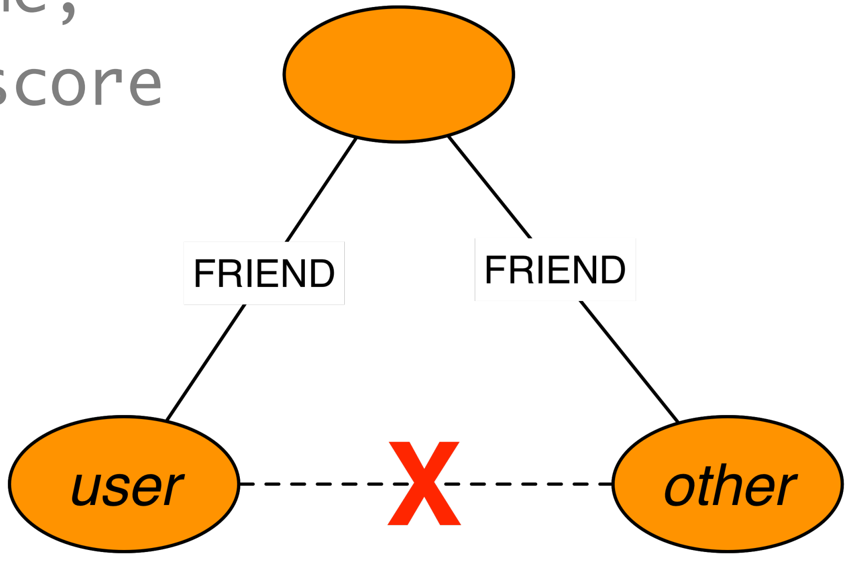
Find Terry's Friends' Friends

```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```



…Who Terry Doesn't Know

```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```



Count Matches Per Person

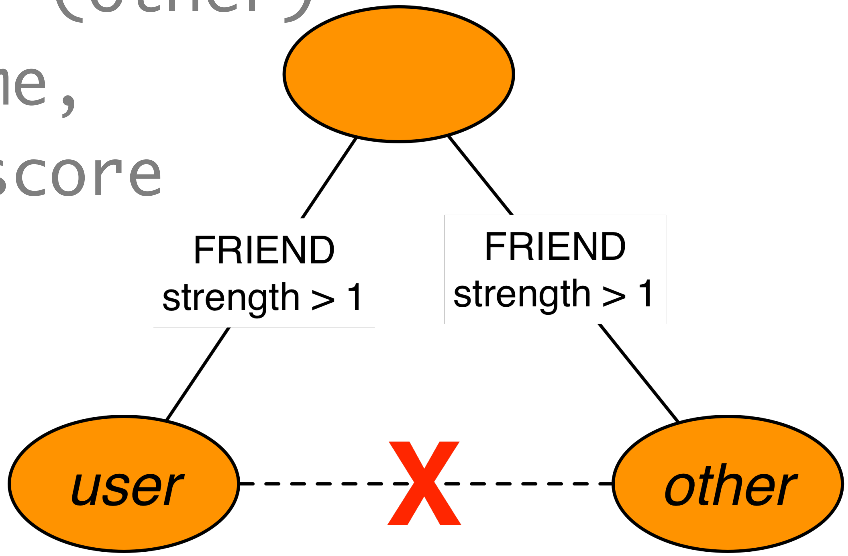
```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

Return The Results

```
MATCH (user:User{name:'Terry'})
      -[:FRIEND*2]-
      (other:User)
WHERE NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

Taking Account of Friendship Strength

```
MATCH (user:User{name:'Terry'})
      -[rels:FRIEND*2]-
      (other:User)
WHERE ALL(r IN rels WHERE r.strength > 1)
AND NOT (user)-[:FRIEND]-(other)
RETURN other.name AS name,
       COUNT(other) AS score
ORDER BY score DESC
```

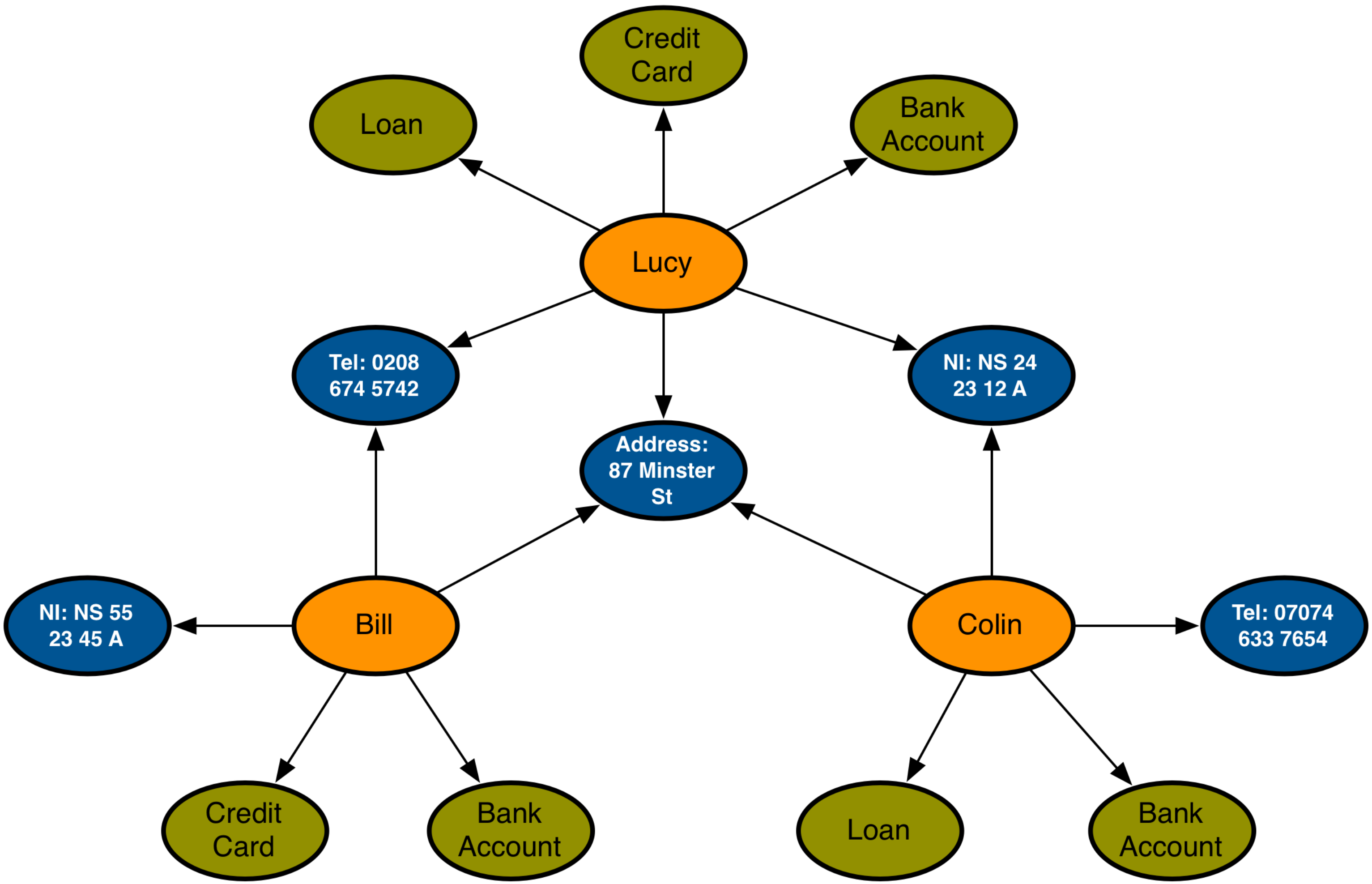


Nowhere To Hide

First-Party Fraud

- Fraudsters apply for credit
  - No intention of repaying
- Appear normal until they “burst out”
  - Clear out accounts
- Fraud ring
  - Share bits of identity (NI, address, telephone)
  - Coordinated “burst out”

Fraud Ring


Credit Card
Loan
Bank Account
Lucy
Tel: 0208 674 5742
Address: 87 Minster St
NI: NS 24 23 12 A
NI: NS 55 23 45 A
Bill
Colin
Tel: 07074 633 7654
Credit Card
Bank Account
Loan
Bank Account

Query

- Create new applicant
- Connect applicant to identity info
  - Reuse existing identify info where possible

Then

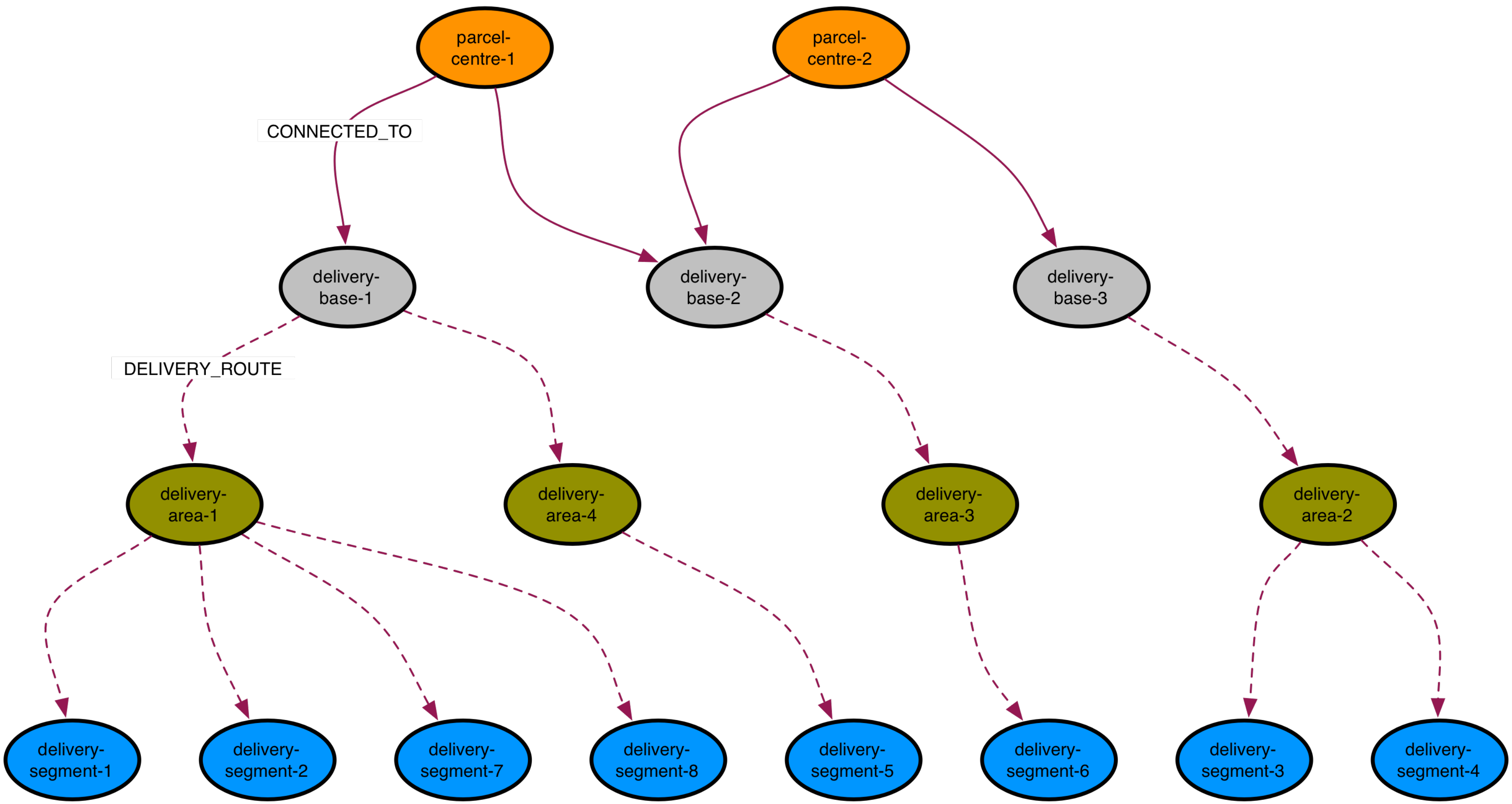
- Select applicant's identity info
- Crawl surrounding graph
  - Look for expansive clusters of account holders

Path Calculations

Problem

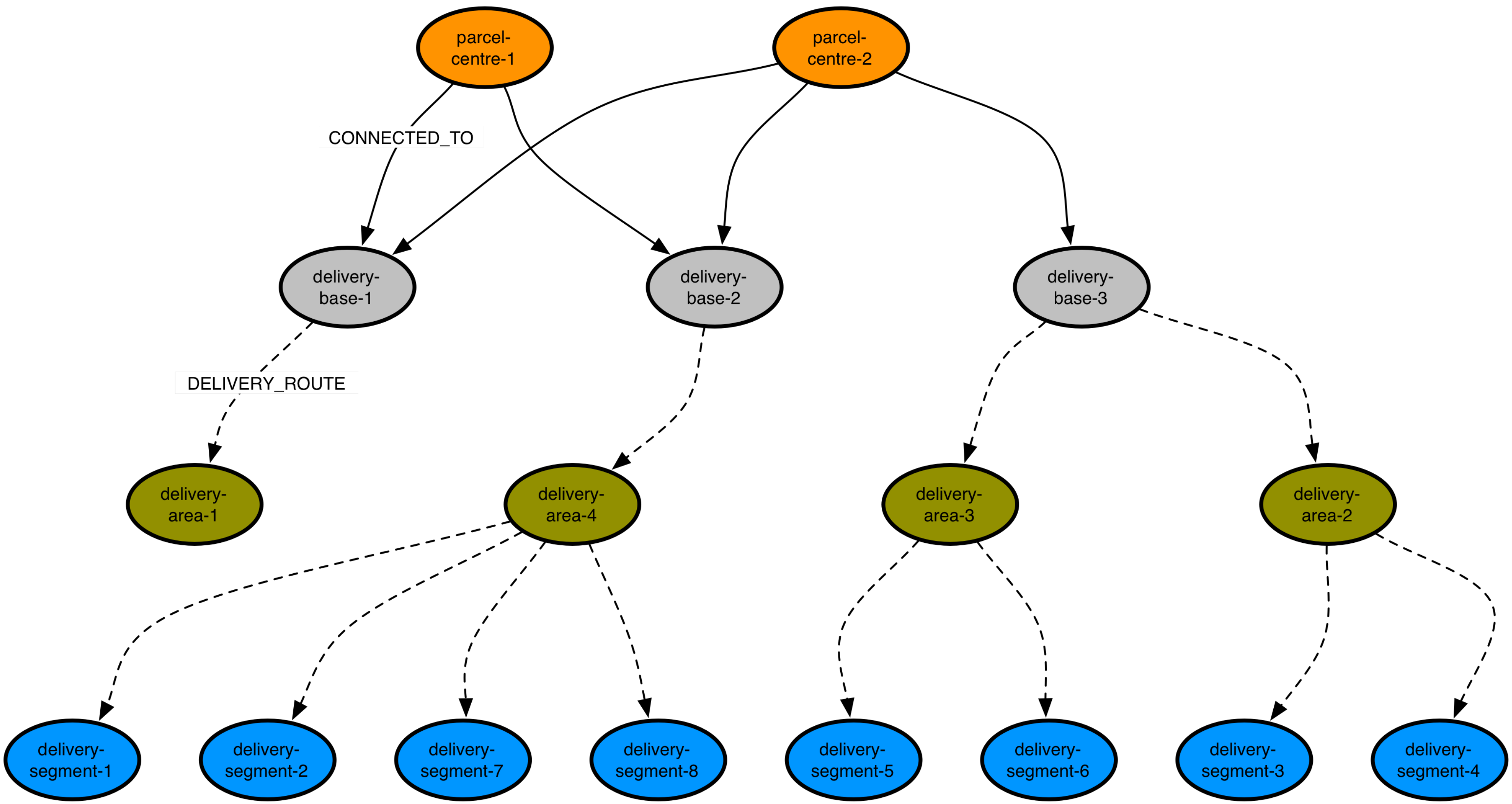
- Increase in parcel traffic
  - Amazon, eBay
  - Current infrastructure can't cope
- Calculate optimal route
  - Under 20ms
  - Routes vary over time
- Numbers:
  - 2000-3000 parcels per second
  - 25 national parcel centres, 2 million postcodes, 30 million address

Period 1



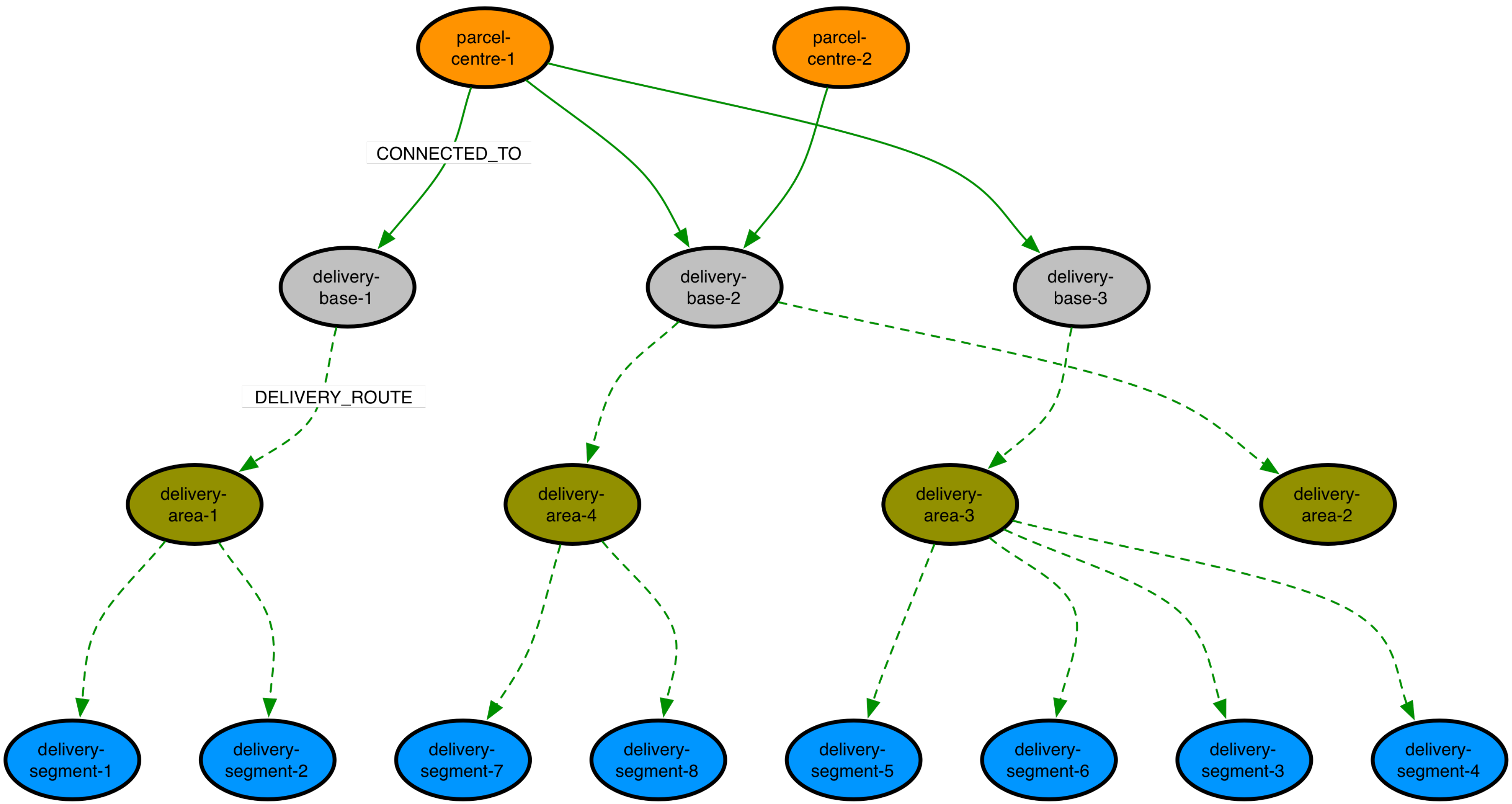
parcel-centre-1
parcel-centre-2
CONNECTED_TO
delivery-base-1
delivery-base-2
delivery-base-3
DELIVERY_ROUTE
delivery-area-1
delivery-area-4
delivery-area-3
delivery-area-2
delivery-segment-1
delivery-segment-2
delivery-segment-7
delivery-segment-8
delivery-segment-5
delivery-segment-6
delivery-segment-3
delivery-segment-4

Period 2



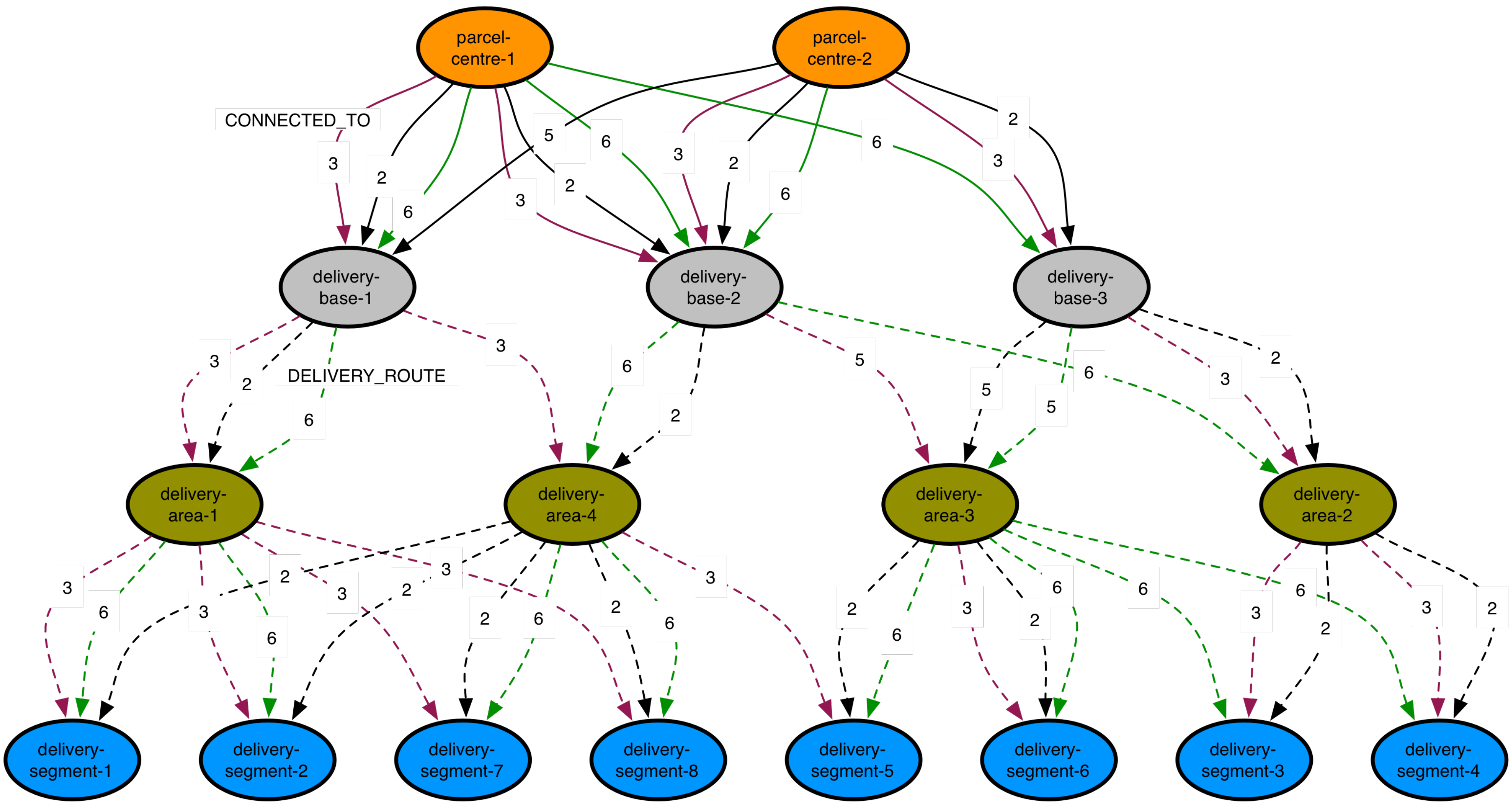
parcel-centre-1
parcel-centre-2
CONNECTED_TO
delivery-base-1
delivery-base-2
delivery-base-3
DELIVERY_ROUTE
delivery-area-1
delivery-area-4
delivery-area-3
delivery-area-2
delivery-segment-1
delivery-segment-2
delivery-segment-7
delivery-segment-8
delivery-segment-5
delivery-segment-6
delivery-segment-3
delivery-segment-4

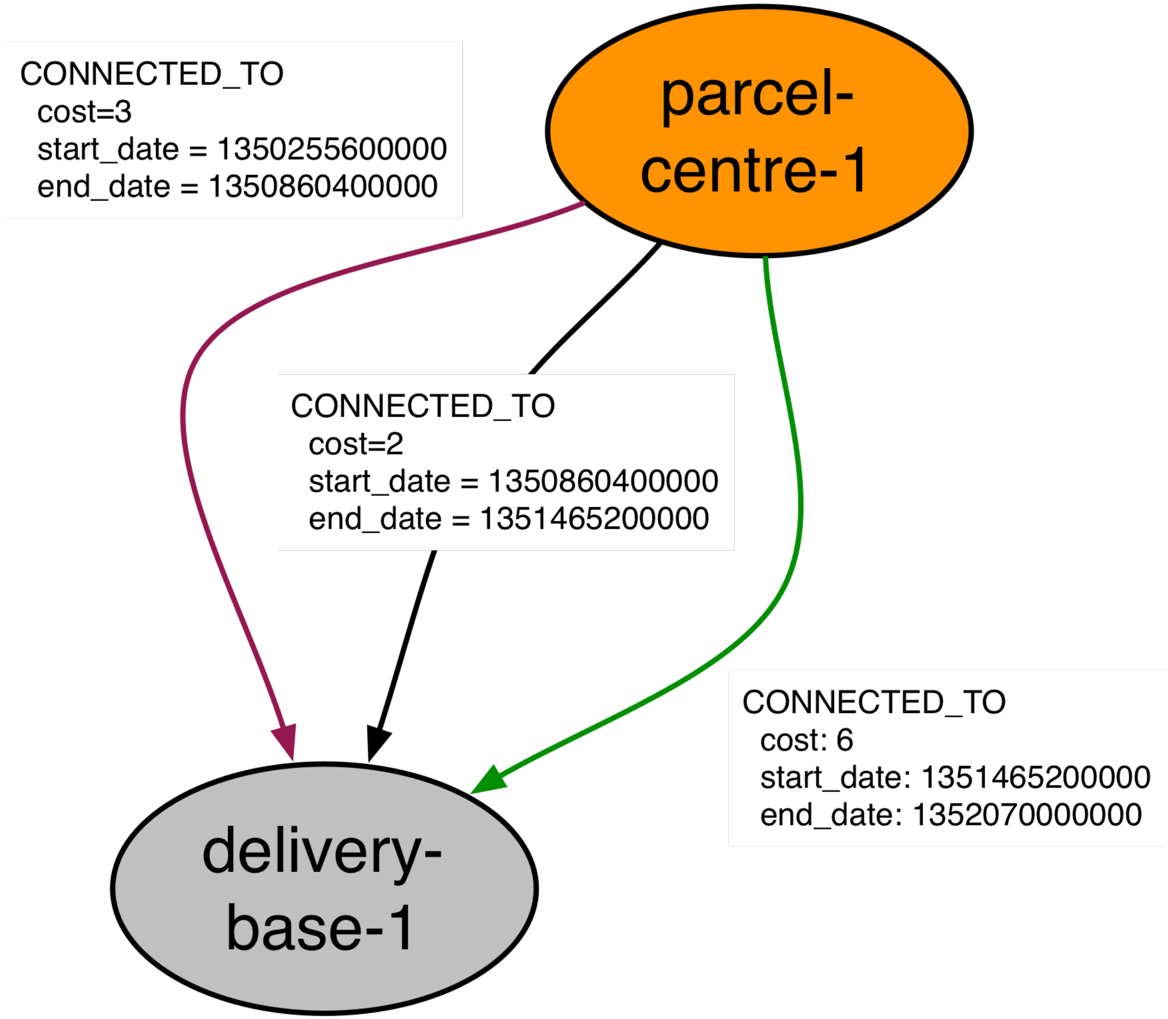
Period 3



parcel-centre-1
parcel-centre-2
CONNECTED_TO
delivery-base-1
delivery-base-2
delivery-base-3
DELIVERY_ROUTE
delivery-area-1
delivery-area-4
delivery-area-3
delivery-area-2
delivery-segment-1
delivery-segment-2
delivery-segment-7
delivery-segment-8
delivery-segment-5
delivery-segment-6
delivery-segment-3
delivery-segment-4

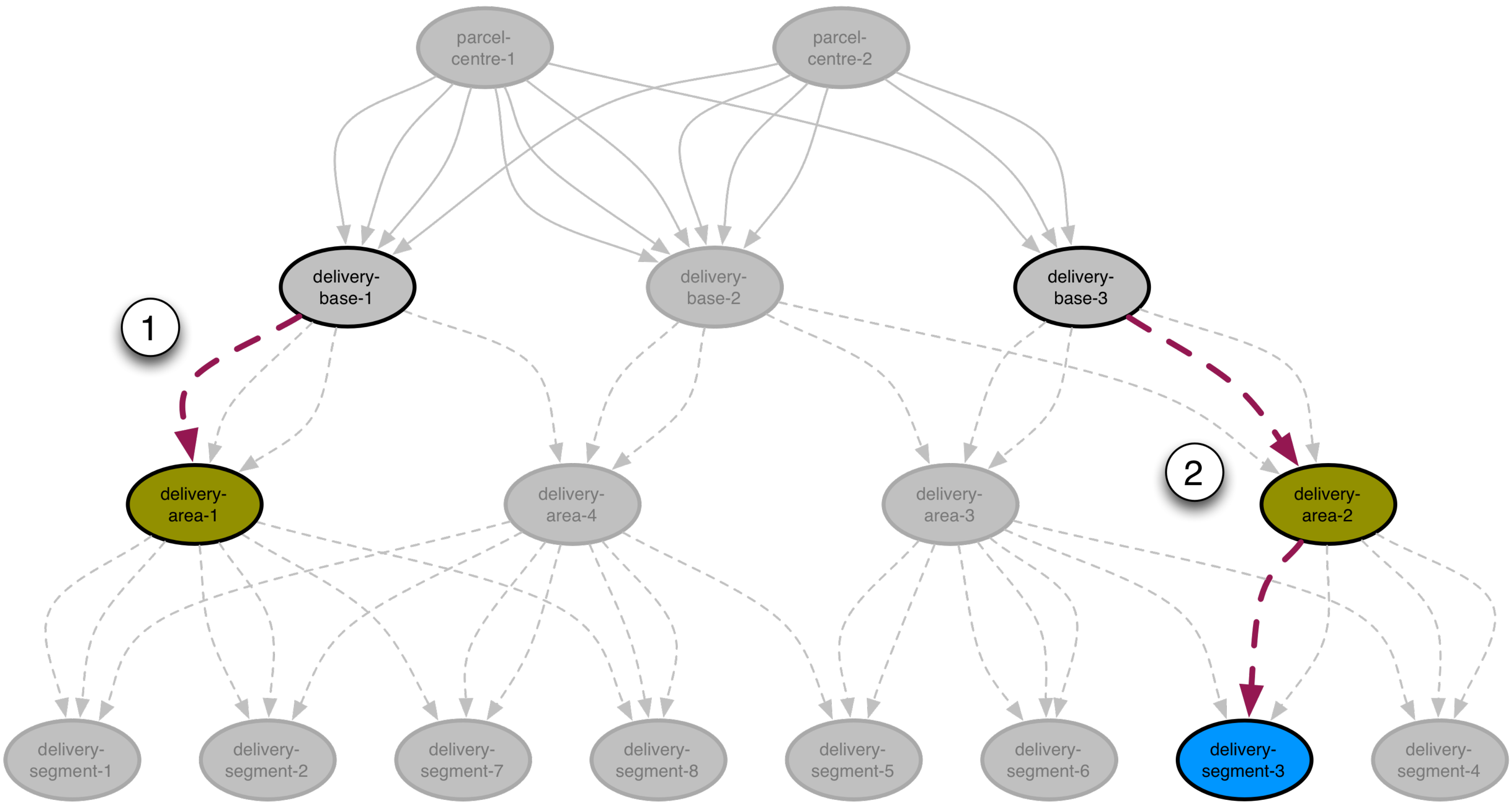
The Full Graph





parcel-
centre-1
CONNECTED_TO
cost=3
start_date = 1350255600000
end_date = 1350860400000
CONNECTED_TO
cost=2
start_date = 1350860400000
end_date = 1351465200000
CONNECTED_TO
cost: 6
start_date: 1351465200000
end_date: 1352070000000
delivery-
base-1

Steps 1 and 2



parcel-centre-1
parcel-centre-2
delivery-base-1
delivery-base-2
delivery-base-3
1
2
delivery-area-1
delivery-area-4
delivery-area-3
delivery-area-2
delivery-segment-1
delivery-segment-2
delivery-segment-7
delivery-segment-8
delivery-segment-5
delivery-segment-6
delivery-segment-3
delivery-segment-4

Find Start and End

```
MATCH (s:Location {name:{startLocation}}),
      (e:Location {name:{endLocation}})
```

Calculate Up Leg

```
MATCH upLeg = (s)<-[:DELIVERY_ROUTE*1..2]-(db1)
WHERE all(r in relationships(upLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Path From Start to a Delivery Base

```
MATCH upLeg = (s)<-[:DELIVERY_ROUTE*1..2]-(db1)
WHERE all(r in relationships(upLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

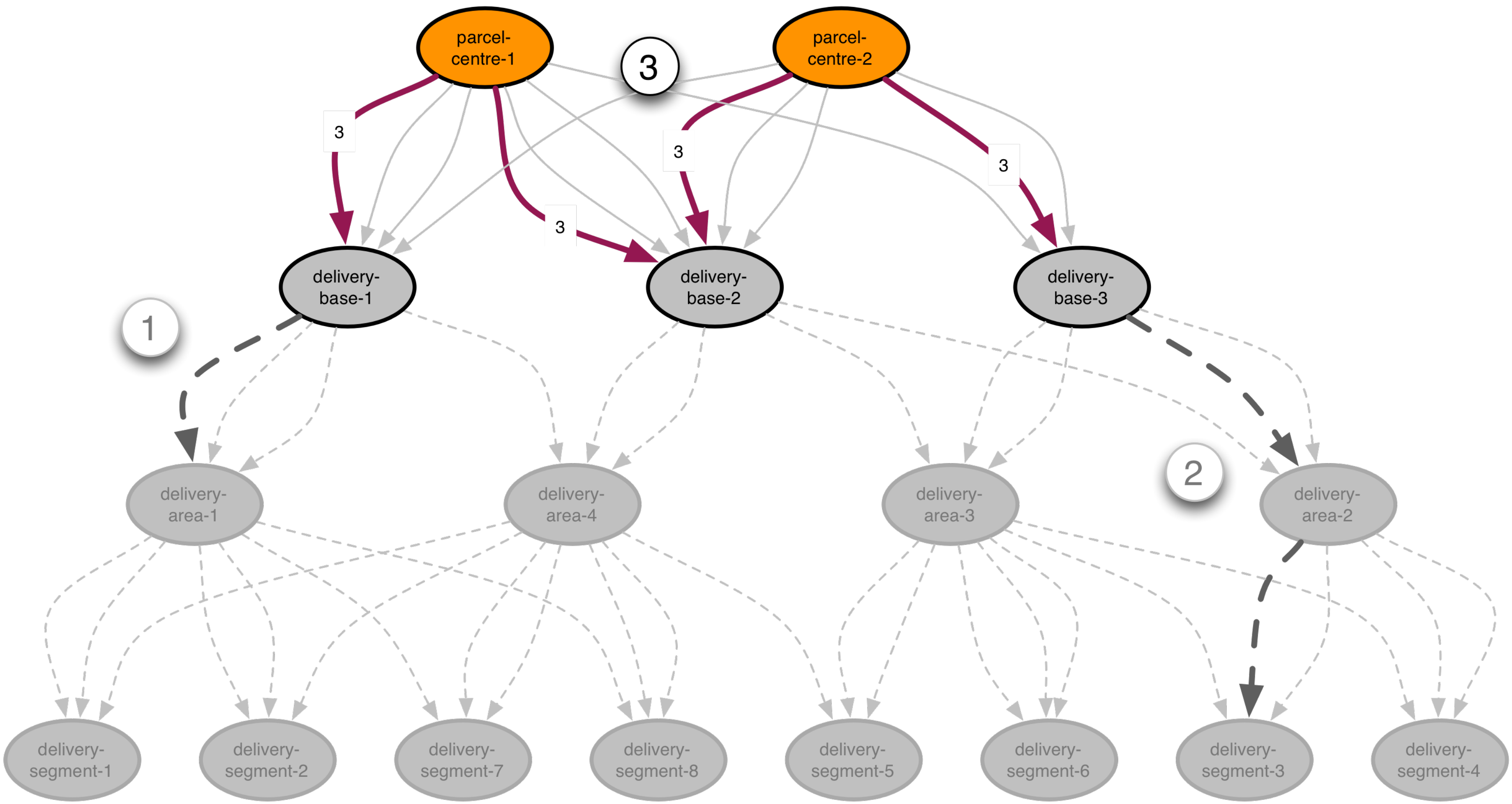
Filter Relationships by Period

```
MATCH upLeg = (s)<-[:DELIVERY_ROUTE*1..2]-(db1)
WHERE all(r in relationships(upLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Calculate Down **Path**

```
WITH  e, upLeg, db1
MATCH downLeg = (db2)-[:DELIVERY_ROUTE*1..2]->(e)
WHERE all(r in relationships(downLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Step 3



parcel-centre-1
parcel-centre-2
3
3
3
3
3
3
delivery-base-1
delivery-base-2
delivery-base-3
1
2
delivery-area-1
delivery-area-4
delivery-area-3
delivery-area-2
delivery-segment-1
delivery-segment-2
delivery-segment-7
delivery-segment-8
delivery-segment-5
delivery-segment-6
delivery-segment-3
delivery-segment-4

Find Routes Between Delivery Bases

```
WITH  db1, db2, upLeg, downLeg
MATCH topRoute =
      (db1)<-[:CONNECTED_TO]-()
       -[:CONNECTED_TO*1..3]-(db2)
WHERE all(r in relationships(topRoute)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Paths Between Delivery Bases

```
WITH  db1, db2, upLeg, downLeg
MATCH topRoute =
      (db1)<-[:CONNECTED_TO]-()
       -[:CONNECTED_TO*1..3]-(db2)
WHERE all(r in relationships(topRoute)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Filtered by Period

```
WITH  db1, db2, upLeg, downLeg
MATCH topRoute =
      (db1)<-[:CONNECTED_TO]-()
       -[:CONNECTED_TO*1..3]-(db2)
WHERE all(r in relationships(topRoute)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
```

Calculate Shortest Route Between Delivery Bases

```
WITH  upLeg, downLeg, topRoute,
      reduce (
        weight=0,
        r in relationships(topRoute) |
        weight+r.cost) AS score
      ORDER BY score ASC
      LIMIT 1
RETURN (nodes(upLeg) +
        tail(nodes(topRoute)) +
        tail(nodes(downLeg))) AS route
```

Calculate Shortest **Path** Between Delivery Bases

```
WITH  upLeg, downLeg, topRoute,
      reduce (
        weight=0,
        r in relationships(topRoute) |
        weight+r.cost) AS score
      ORDER BY score ASC
      LIMIT 1
RETURN (nodes(upLeg) +
        tail(nodes(topRoute)) +
        tail(nodes(downLeg))) AS route
```

Full Query

```
MATCH (s:Location {name:{startLocation}}),
      (e:Location {name:{endLocation}})
MATCH upLeg = (s)<-[:DELIVERY_ROUTE*1..2]-(db1)
WHERE all(r in relationships(upLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
WITH  e, upLeg, db1
MATCH downLeg = (db2)-[:DELIVERY_ROUTE*1..2]->(e)
WHERE all(r in relationships(downLeg)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
WITH  db1, db2, upLeg, downLeg
MATCH topRoute = (db1)<-[:CONNECTED_TO]-()-[:CONNECTED_TO*1..3]-(db2)
WHERE all(r in relationships(topRoute)
          WHERE r.start_date <= {intervalStart}
          AND r.end_date >= {intervalEnd})
WITH  upLeg, downLeg, topRoute,
      reduce(weight=0, r in relationships(topRoute) | weight+r.cost) AS score
      ORDER BY score ASC
      LIMIT 1
RETURN (nodes(upLeg) + tail(nodes(topRoute)) + tail(nodes(downLeg))) AS route
```

Online Training

http://www.neo4j.org/learn/online_course

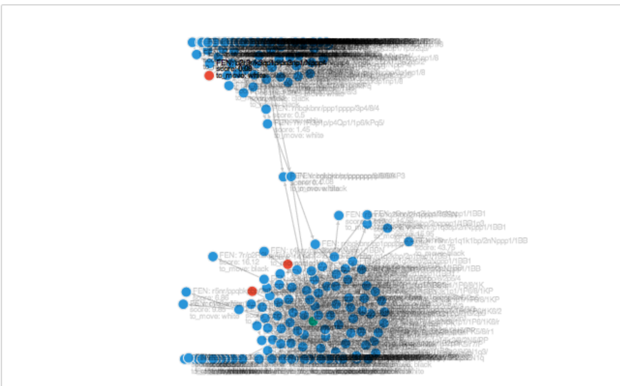
Online Training: Getting Started with Neo4j

Learn Neo4j at your own pace and time with our free online training course. Get introduced to graph databases, learn the core functionality of Neo4j, and practice Cypher with this engaging and interactive course.

Get started today »

Graph Gists

https://github.com/neo4j-contrib/graphgist/wiki



Here is the one of my games for analysis

I managed to squeek out a win, but it makes for an interesting game to look at because both of us made so many blunders.

Wes - Alvin, Fairfax Open

1. e4 c5 2. Nf3 Nc6 3. d4 cxd4 4. Nxd4 g6 5. f4 Bg7 6. e5 d6 7. Bb5 Qb6 8. Nc3 e6 9. Be3 dxe5 10. Nf5 Qc7 11. Nd6+ Ke7 12. Bc5 f6 13. Nxc8+ Kf7 14. Nd6+ Kf8 15. Nxb7+ Kf7 16. Nd6+ Ke7 17. Nf5+ Kf7 18. Nxg7 Kxg7 19. Qf3 Nge7 20. Bxe7 Nd4 21. Bxf6+ Kxf6 22. fxe5+ Kg7 23. Qf6+ Kg8 24. O-O-O Rf8 25. Qg5 Rf5 26. Qh4 Nxb5 27. Nxb5 Qxe5 28. Rd8+ Kg7 29. Rxh8 Kxh8 30. Qd8+ Kg7 31. Nc3 Qe3+ 32. Qd2 Qxd2+ 33. Kxd2 Rf2+ 34. Kd3 Rxg2 35. b4 Kf6 36. Rf1+ Ke7 37. Nb5 Rxh2 38. Nxa7 Rh3+ 39. Kc4 Ra3 40. Nc6+ Kd6 41. Na5 g5 42. Rh1 Ke5 43. Rxh7 Rxa2 44. Kb3 Ra1 45. Kb2 Re1 46. Rg7 Kf4 47. c4 g4 48. c5 Re2+ 49. Kc3 Re3+ 50. Kc4 Re1 51. c6 g3 52. Kc5 Rd1 53. c7 e5 54. c8=Q Rc1+ 55. Nc4 Kf3 56. Qg4+ Kf2 57. Qxg3+ Ke2 58. Qe3+ Kd1 59. Qd2# *

All duration times are in minutes and delay type definitons can be found here.

Included Airports

Name	City	Abbreviation
Hartsfield-Jackson Atlanta International Airport	Atlanta	ATL
O'Hare International Airport	Chicago	ORD
Los Angeles International Airport	Los Angeles	LAX
Dallas/Fort Worth International Airport	Dallas/Fort Worth	DFW

Proposed Model

graphdatabases.com