Graphs vs Fraud!

Dr. Jim Webber
Chief Scientist, Neo4j
@jimwebber

Overview

- First-party Fraud
- Whiplash for Cash
- Online Payment and Identity
- Master Data Management
- Provenance
- Governance

“First-party Fraud”

First-Party Fraud

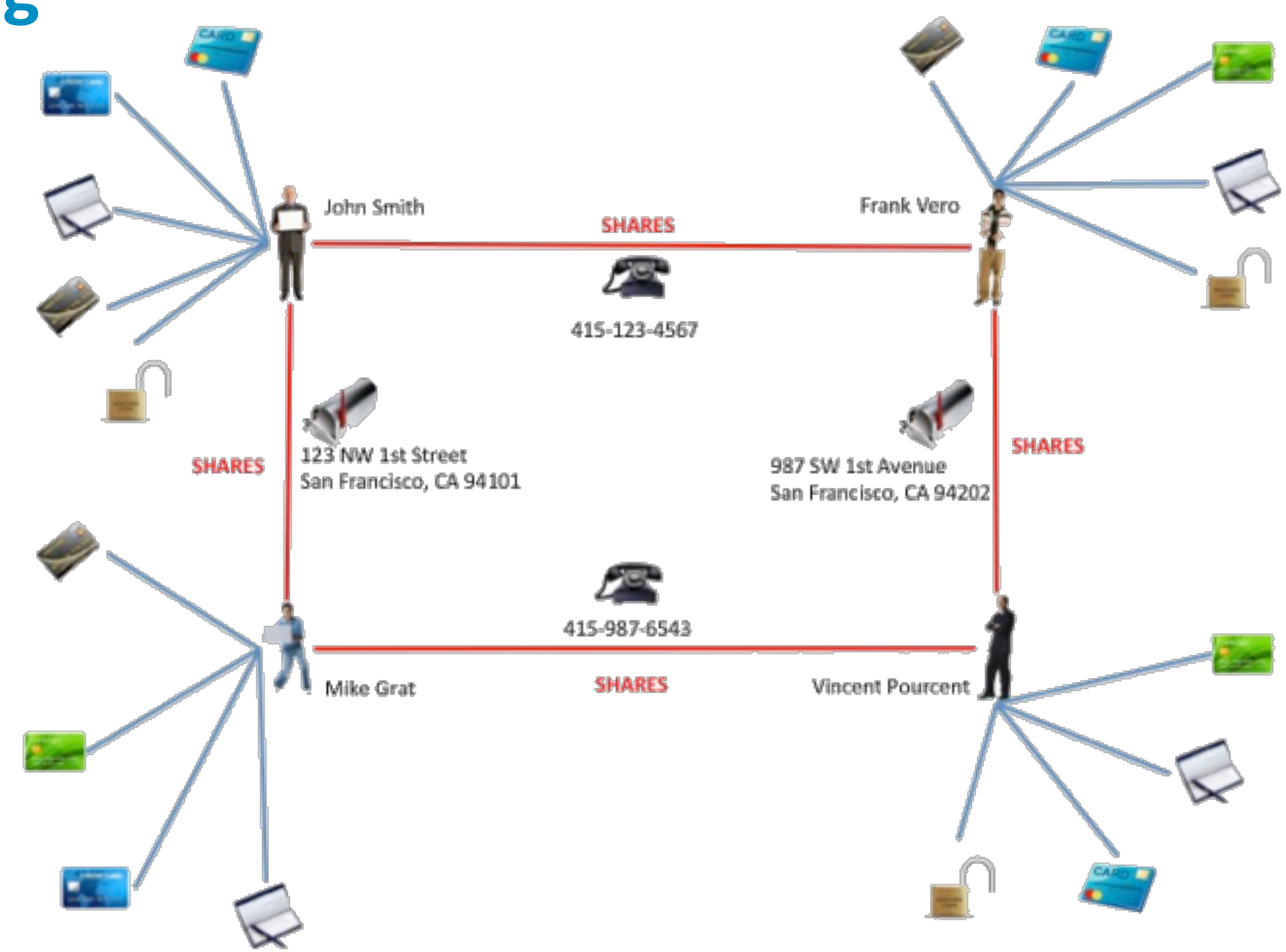
- Fraudster’s aim: apply for lines of credit, act normally, extend credit, then…run off with it
- Fabricate a network of synthetic IDs, aggregate smaller lines of credit into substantial value
- Often a hidden problem since only banks are hit
  - Whereas third-party fraud involves customers whose identities are stolen
  - More on that later…

So what?

- $10’s **billions** lost by US banks **every year**
- **25%** of the total consumer credit write-offs in the USA
- Around **20%** of unsecured bad debt in EU and USA is misclassified
  - In reality it is first-party fraud

These are enormous numbers

Fraud Ring


John Smith
SHARES
Frank Vero
415-123-4567
123 NW 1st Street
San Francisco, CA 94101
SHARES
987 SW 1st Avenue
San Francisco, CA 94202
SHARES
415-987-6543
Mike Grat
SHARES
Vincent Pourcent

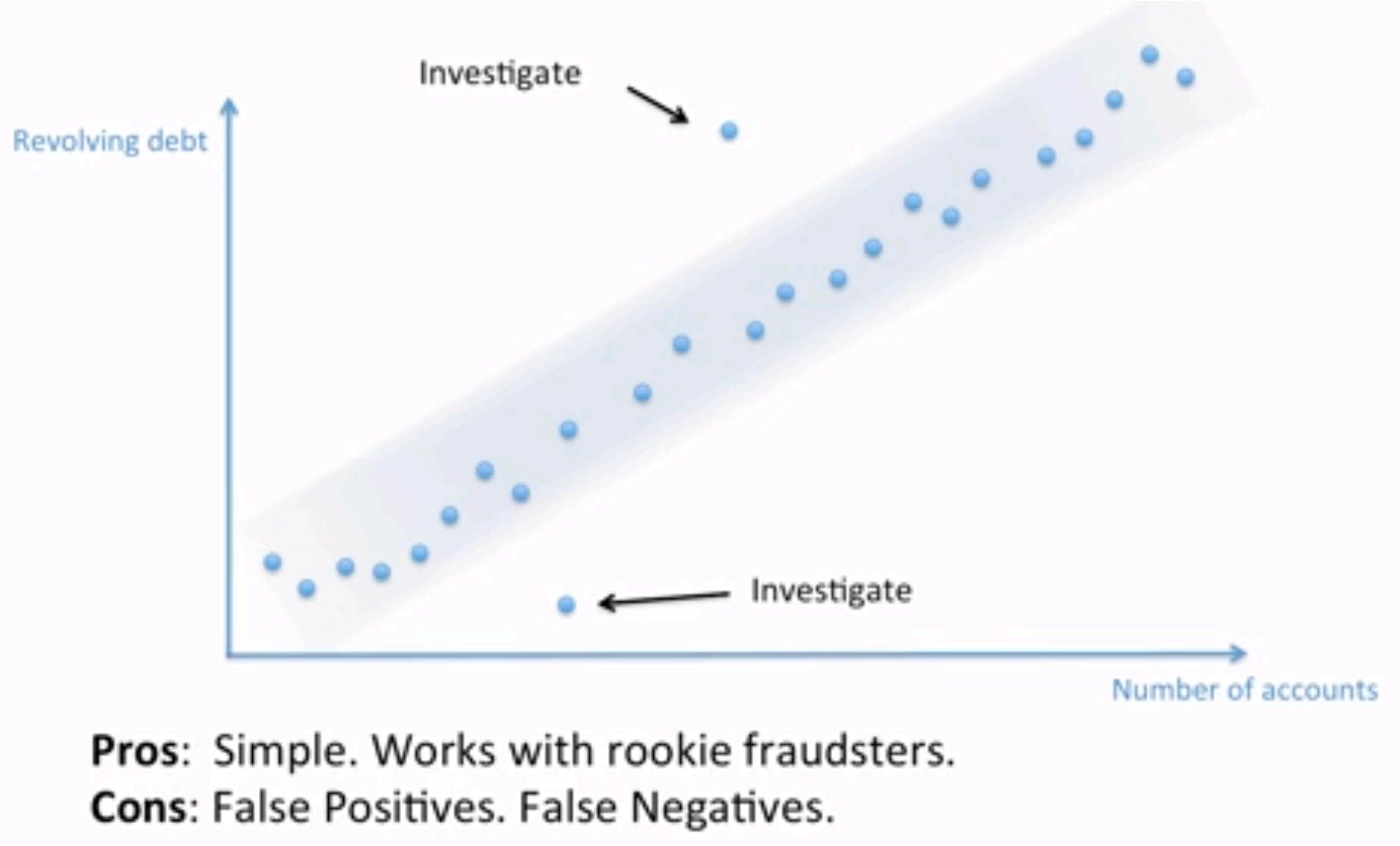
Then the fraud happens…

- Revolving doors strategy
  - Money moves from account to account to provide legitimate transaction history
- Banks duly increase credit lines
  - Observed responsible credit behaviour
- Fraudsters max out all lines of credit and then **bust out**

… and the Bank loses

- Collections process ensues
  - Real addresses are visited
  - Fraudsters deny all knowledge of synthetic IDs
  - Bank writes off debt
- Two fraudsters can easily rack up $80k
- Well organised crime rings can rack up many times that

Discrete Analysis Fails to predict…



Pros: Simple. Works with rookie fraudsters.

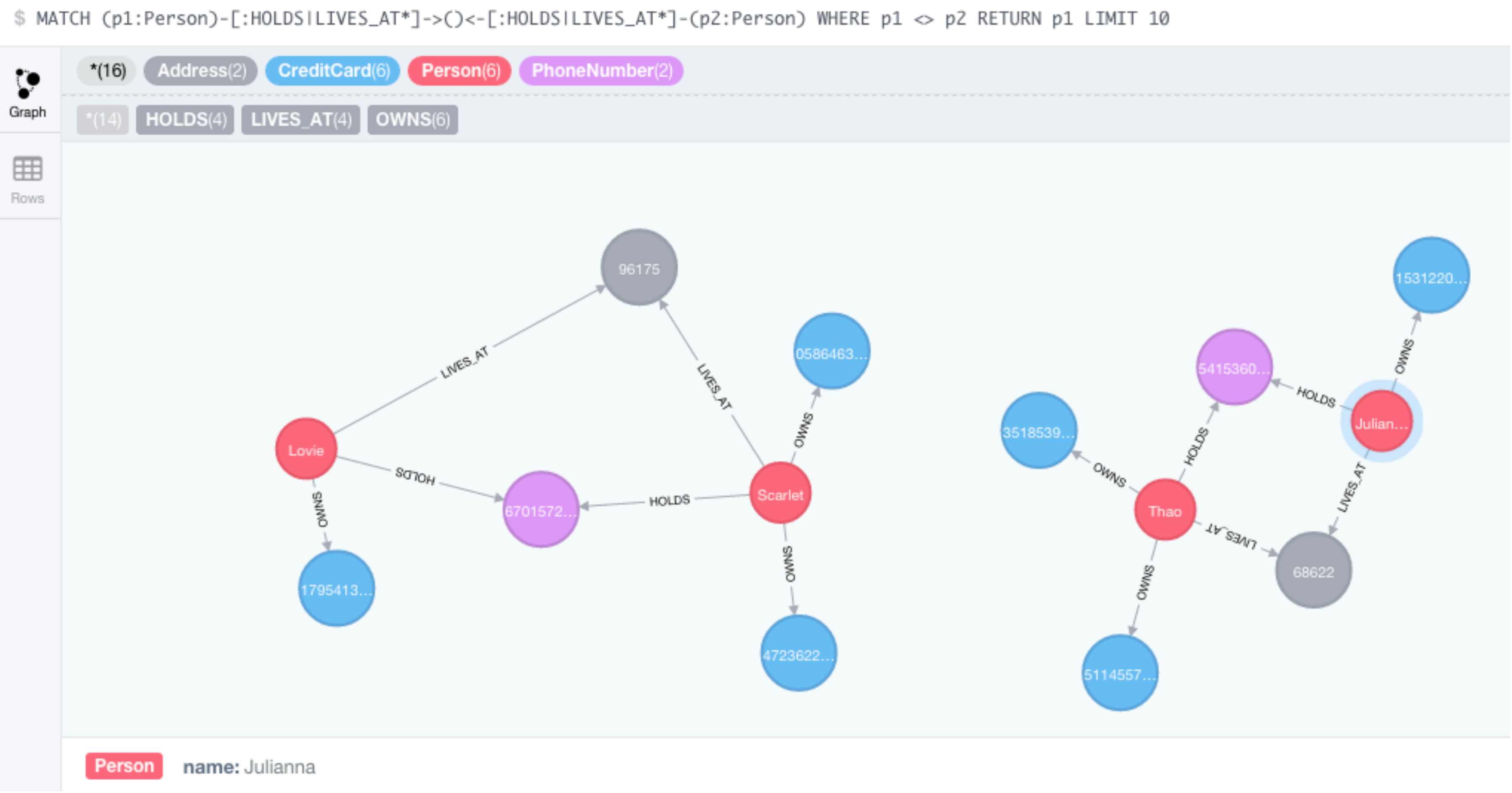
Cons: False Positives. False Negatives.

…and Makes it Hard to React

- When the bust out starts to happen, how do you know what to cancel?
- And how do you do it faster then the fraudster to limit your losses?

- A graph, that’s how!

Probably Non-Fraudulent Cohabiters



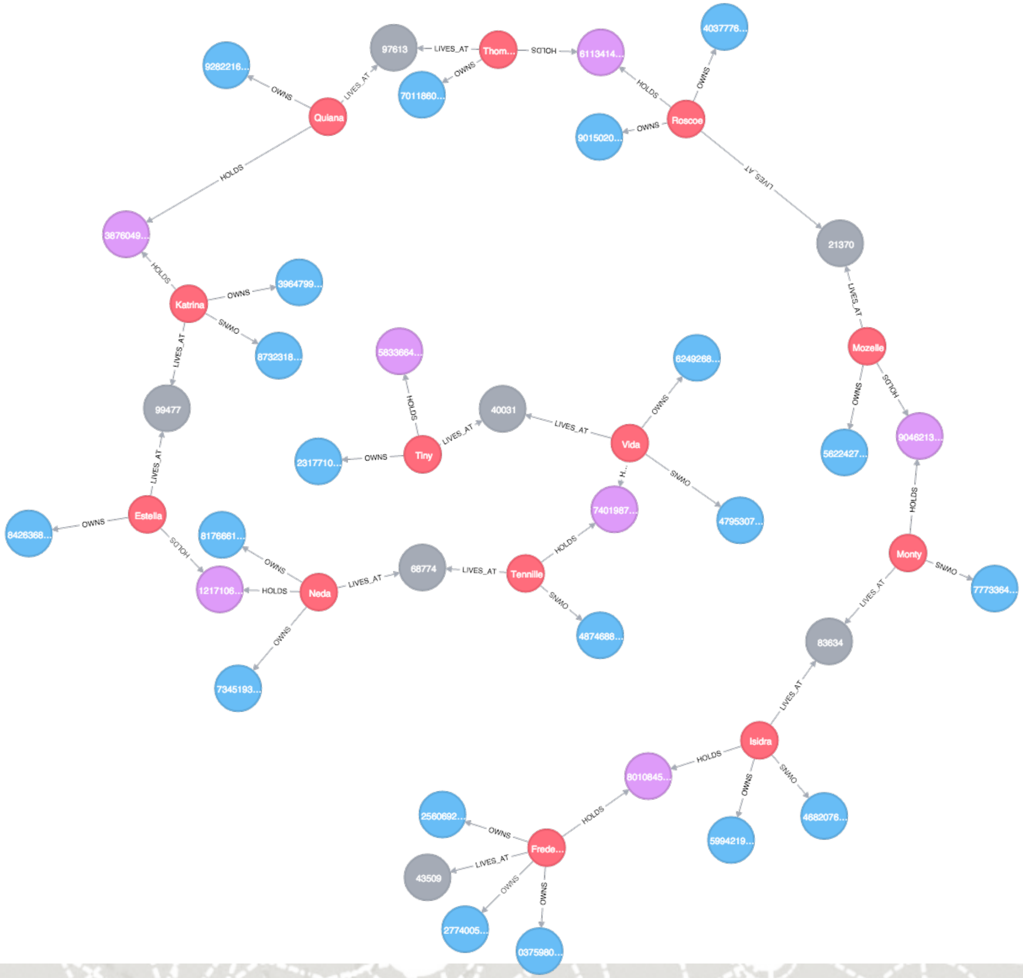
```
$ MATCH (p1:Person)-[:HOLDS|LIVES_AT*]->()<-[:HOLDS|LIVES_AT*]-(p2:Person) WHERE p1 <> p2 RETURN p1 LIMIT 10
```

Person **name:** Julianna

Probable Cohabiters Query

```
MATCH (p1:Person)-[:HOLDS|LIVES_AT*]->()
    <-[:HOLDS|LIVES_AT*]-(p2:Person)
WHERE p1 <> p2
RETURN DISTINCT p1
```

Dodgy-Looking Chain



Risky People

```
MATCH (p1:Person)-[:HOLDS|LIVES_AT]->()
  <-[:HOLDS|LIVES_AT]-(p2:Person)
  -[:HOLDS|LIVES_AT]->()
  <-[:HOLDS|LIVES_AT]-(p3:Person)
WHERE p1 <> p2 AND p2 <> p3 AND p3 <> p1
WITH collect (p1.name) + collect(p2.name) +
     collect(p3.name) AS names
UNWIND names AS fraudster
RETURN DISTINCT fraudster
```

Pretty quick...

```
Number of people: [5163]
Number of fraudsters: [40]
Time taken: [2495] ms
```

Localise the focus

```
MATCH (p1:Person {name:'Sol'})-[:HOLDS|LIVES_AT]->()…
```

```
Number of fraudsters: [5]
Time taken: [431] ms
```

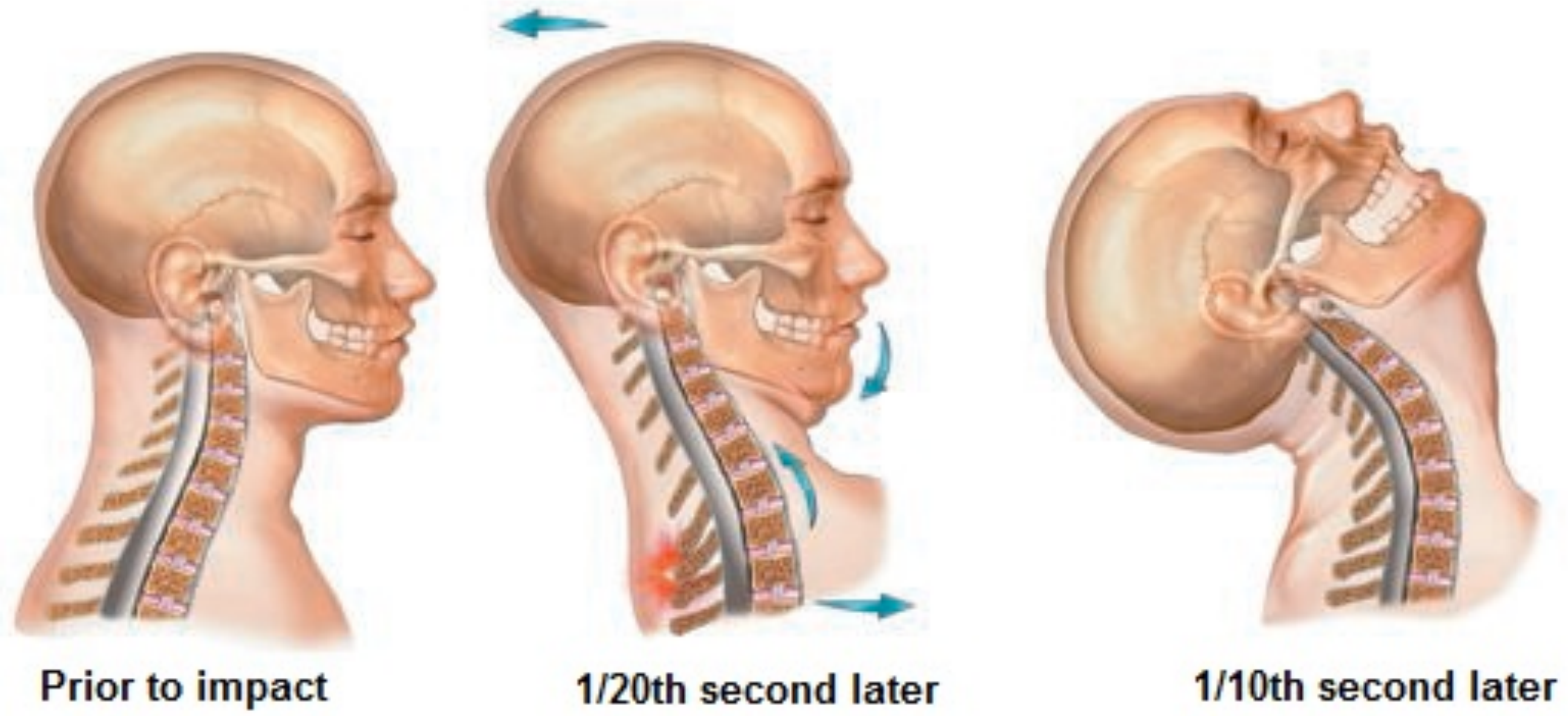
Stop a bust-out inms.

Quickly Revoke Cards in Bust-Out

```
MATCH (p1:Person)-[:HOLDS|LIVES_AT]->()
  <-[:HOLDS|LIVES_AT]-(p2:Person)
  -[:HOLDS|LIVES_AT]->()
  <-[:HOLDS|LIVES_AT]-(p3:Person)
WHERE p1 <> p2 AND p2 <> p3 AND p3 <> p1
WITH collect (p1) + collect(p2)+ collect(p3)
 AS names
UNWIND names AS fraudster
MATCH (fraudster)-[o:OWNS]->(card:CreditCard)
DELETE o, card
```

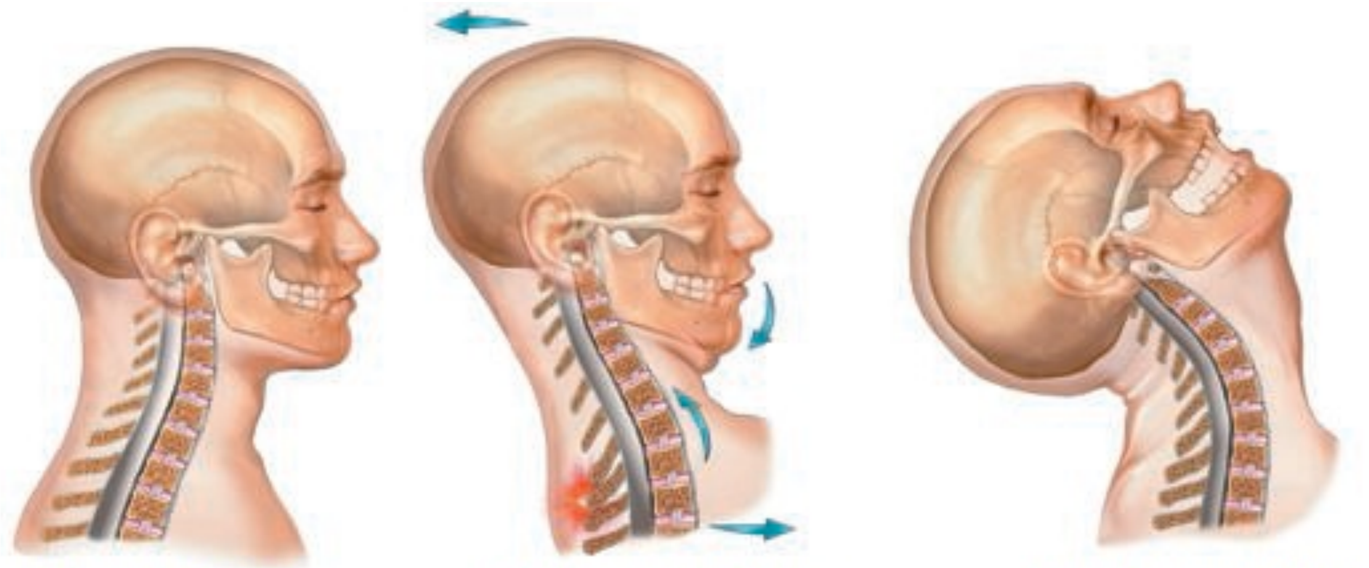
“Auto Fraud”

Whiplash



http://georgia-clinic.com/blog/wp-content/uploads/2013/10/whiplash.jpg

Whiplash for Cash



Prior to impact

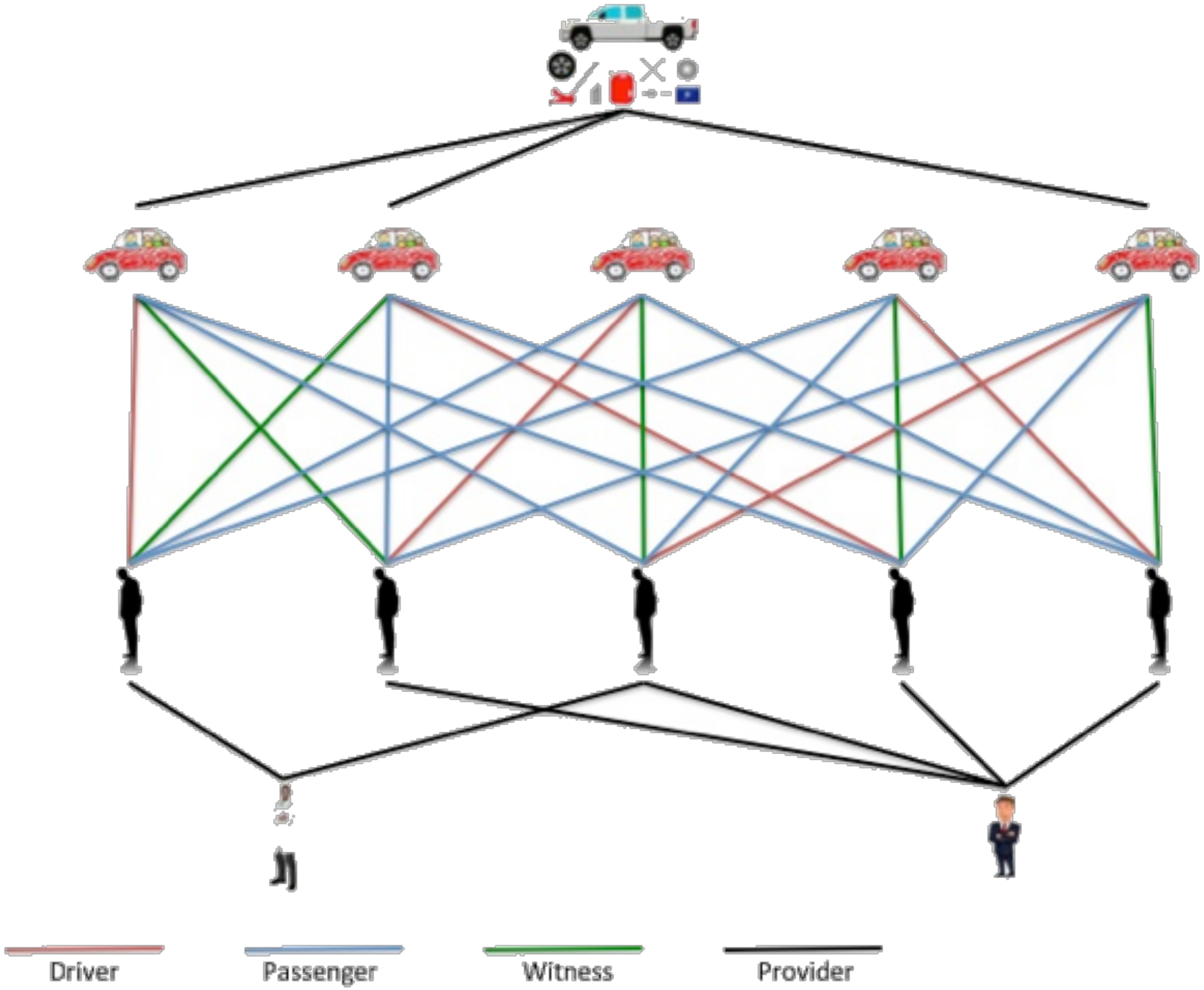
1/20th second later

1/10th second later

6 months later

http://georgia-clinic.com/blog/wp-content/uploads/2013/10/whiplash.jpg

http://cdn2.holytaco.com/wp-content/uploads/2012/06/lottery-winner.jpg

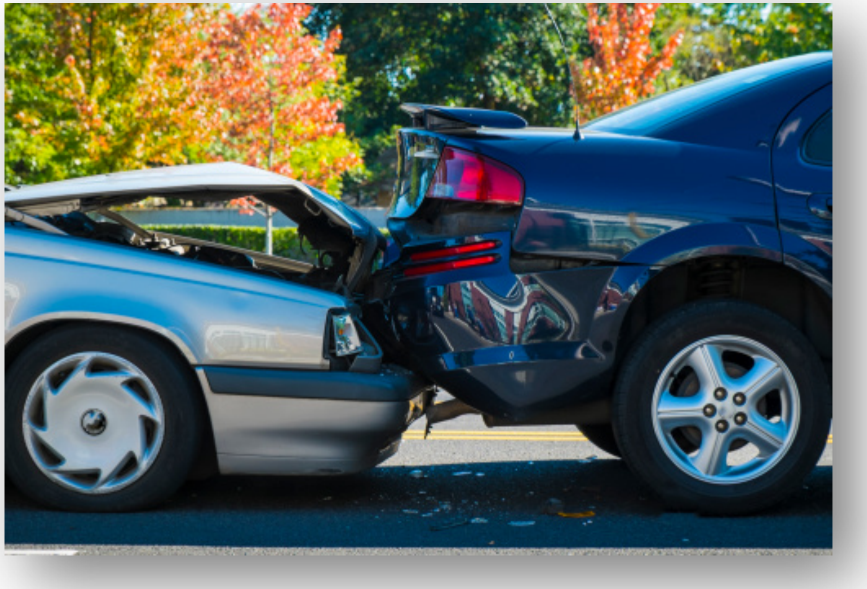

Driver
Passenger
Witness
Provider

Risk

- **$80,000,000,000** annually on auto insurance fraud and growing
  - Even small % reductions are worthwhile!
- British policyholders pay **~£100** per year to cover fraud
- US drivers pay **$200-$300** per year according to US National Insurance Crime Bureau

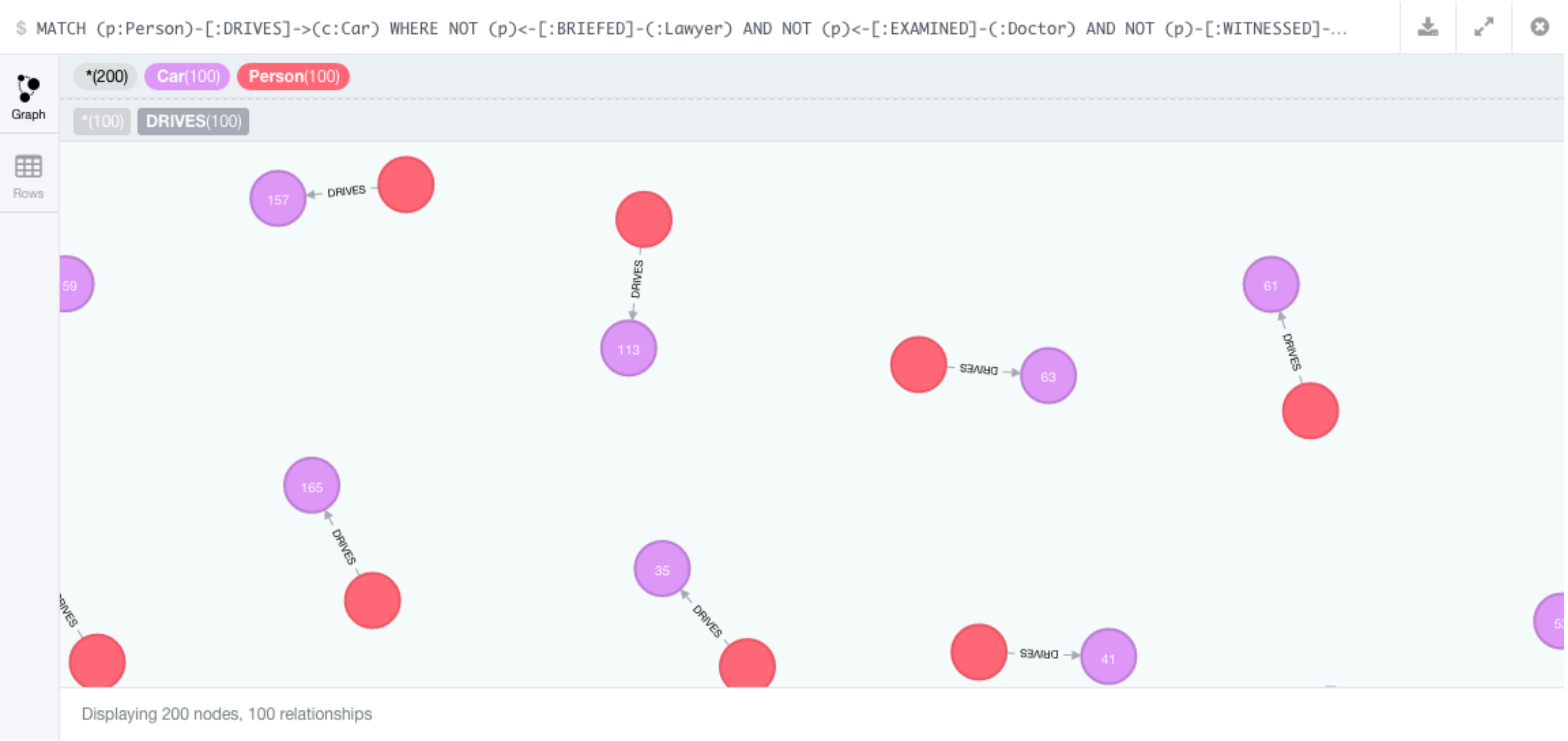
How?

“Flash for Cash”



“Crash for Cash”

Regular Drivers



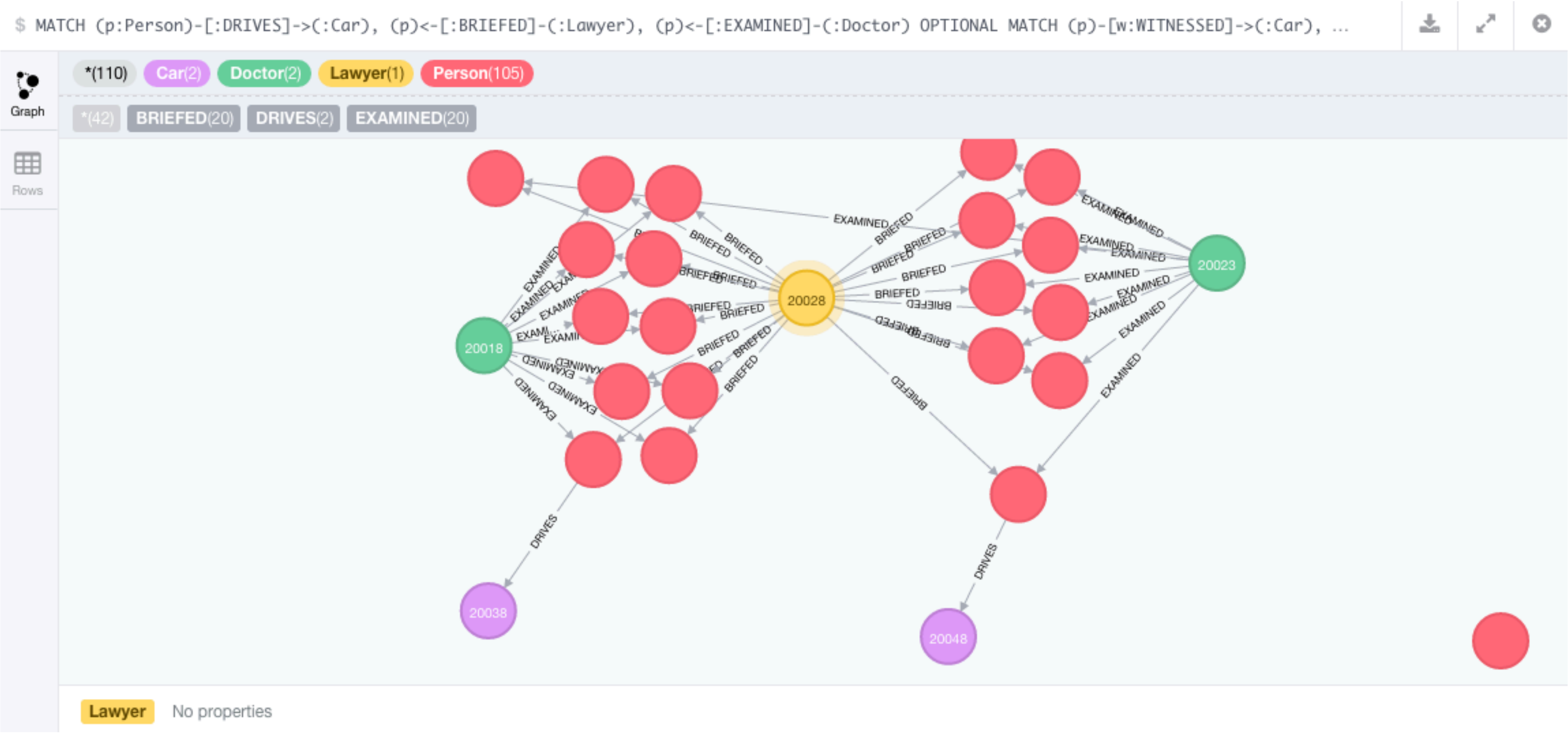
```
$ MATCH (p:Person)-[:DRIVES]->(c:Car) WHERE NOT (p)<-[:BRIEFED]-(:Lawyer) AND NOT (p)<-[:EXAMINED]-(:Doctor) AND NOT (p)-[:WITNESSED]-...
```

Displaying 200 nodes, 100 relationships

Regular Drivers Query

```
MATCH (p:Person)-[:DRIVES]->(c:Car)
WHERE NOT (p)<-[:BRIEFED]-(:Lawyer)
  AND NOT (p)<-[:EXAMINED]-(:Doctor)
  AND NOT (p)-[:WITNESSED]->(:Car)
  AND NOT (p)-[:PASSENGER_IN]->(:Car)
RETURN p,c LIMIT 100
```

Genuine Claimants

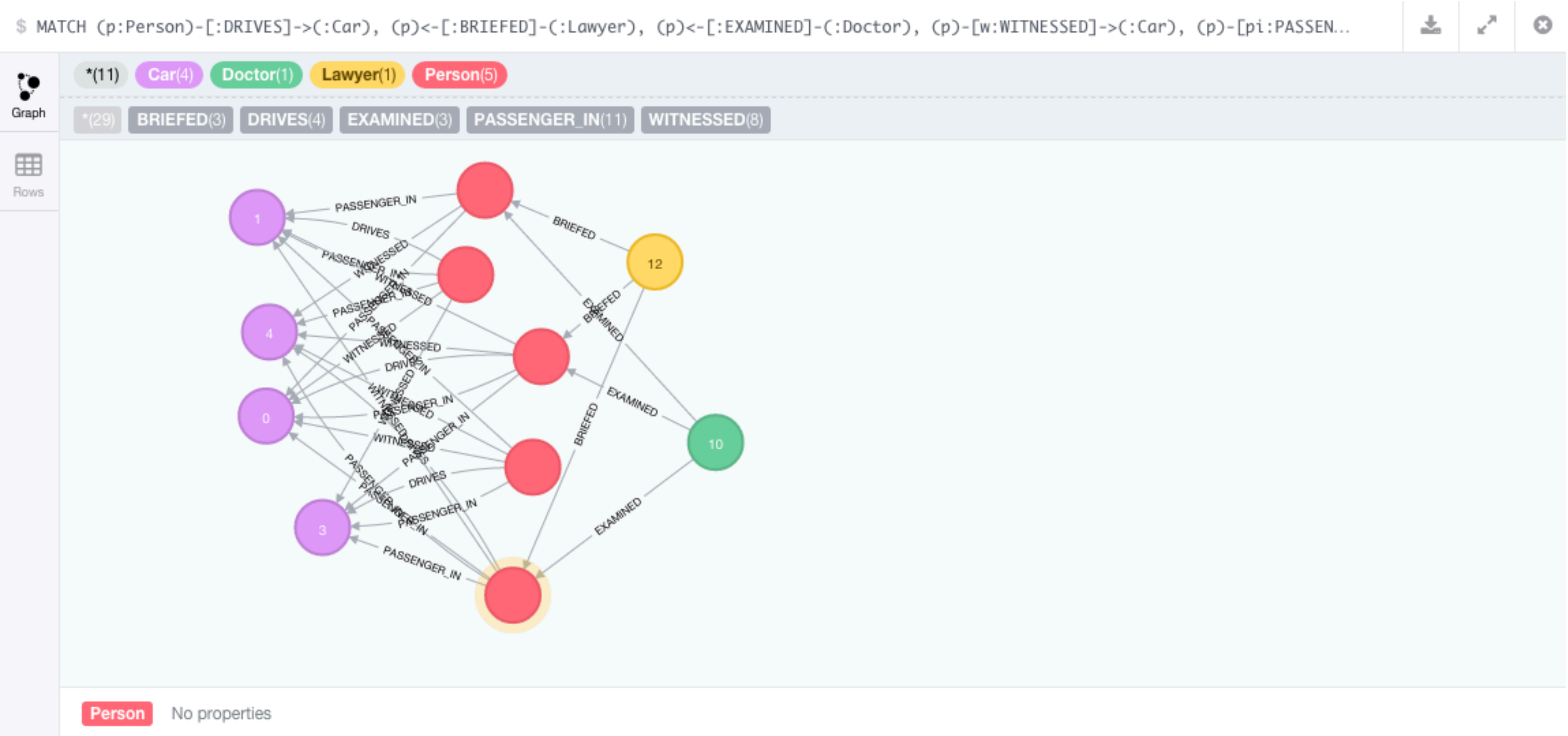


```
$ MATCH (p:Person)-[:DRIVES]->(:Car), (p)<-[:BRIEFED]-(:Lawyer), (p)<-[:EXAMINED]-(:Doctor) OPTIONAL MATCH (p)-[w:WITNESSED]->(:Car), ...
```

Genuine Claimants Query

```
MATCH (p:Person)-[:DRIVES]->(:Car),
  (p)<-[:BRIEFED]-(:Lawyer),
  (p)<-[:EXAMINED]-(:Doctor)
OPTIONAL MATCH (p)-[w:WITNESSED]->(:Car),
  (p)-[pi:PASSENGER_IN]->(:Car)
WITH p, count(w) AS noWitnessed,
  count(pi) as noPassengerIn
```

Fraudsters



```
$ MATCH (p:Person)-[:DRIVES]->(:Car), (p)<-[:BRIEFED]-(:Lawyer), (p)<-[:EXAMINED]-(:Doctor), (p)-[w:WITNESSED]->(:Car), (p)-[pi:PASSEN...
```

Person No properties

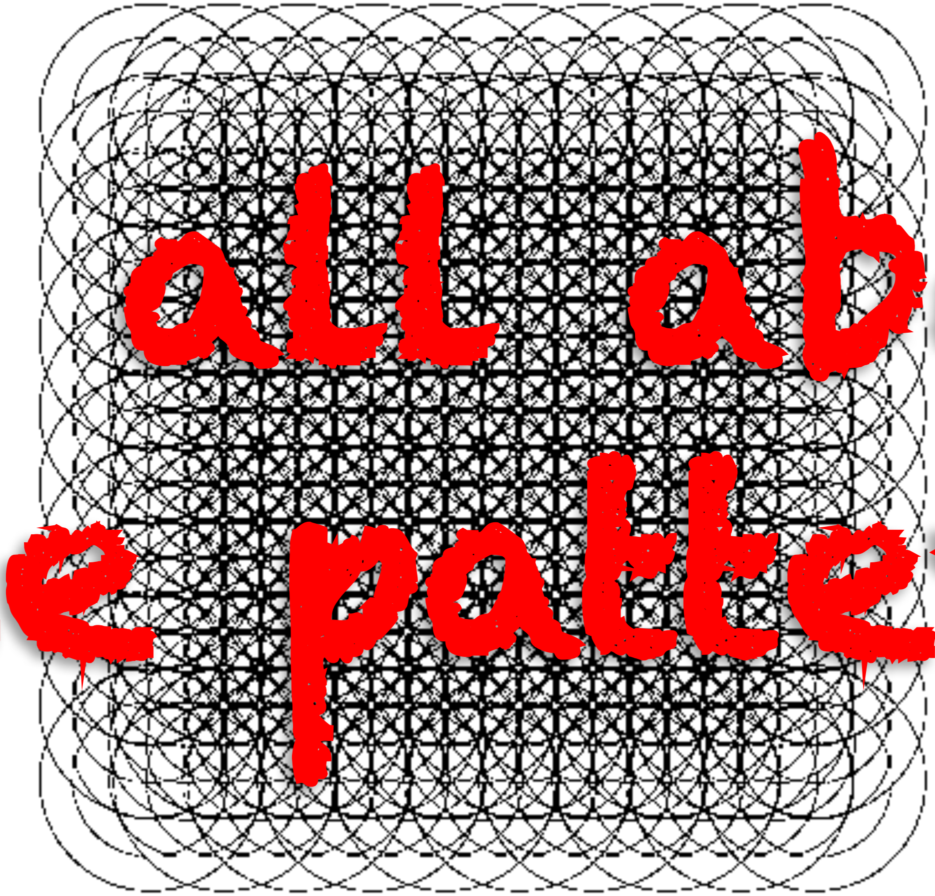
Fraudsters

```
MATCH (p:Person)-[:DRIVES]->(:Car),
  (p)<-[:BRIEFED]-(:Lawyer),
  (p)<-[:EXAMINED]-(:Doctor),
  (p)-[w:WITNESSED]->(:Car),
  (p)-[pi:PASSENGER_IN]->(:Car)
WITH p, count(w) AS noWitnessed,
  count(pi) as noPassengerIn
WHERE noWitnessed > 1 OR noPassengerIn > 1
RETURN p
```

Auto-fraud Graph

- Once you have the fraudsters, finding their support team is easy.
  - `(fraudster)<-[:EXAMINED]-(d:Doctor)`
  - `(fraudster)<-[:BRIEFED]-(l:Lawyer)`
- And it’s also easy to find their passengers
  - `(fraudster)-[:DRIVES]->(:Car)<-[:PASSENGER_IN]-(p)`
- And easy to find other places where they’ve maybe committed fraud
  - `(fraudster)-[:WITNESSED]->(:Car)`
  - `(fraudster)-[:PASSENGER_IN]->(:Car)`
- And you can see this in milliseconds!

It's all about the patterns



“Phoney Persona”

Online Payments Fraud (First-Party)

- Stealing credentials is commonplace
  - Phishing, malware, simple naïve users
- Buying stolen credit card numbers is easy

- How should one protect against seemingly fine credentials?
- And valid credit card numbers?

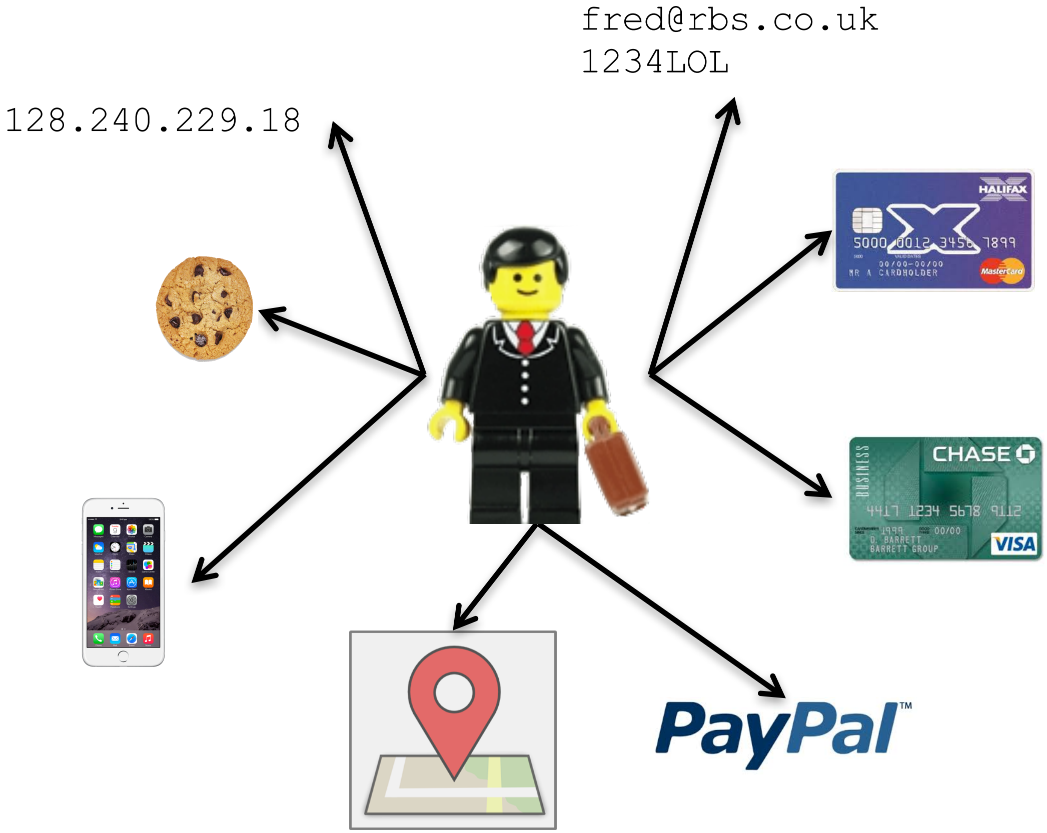
We are all little stars

- Username and passwords
- Two-factor auth
- IP addresses, cookies
- Credit card, paypal account

- Some gaming sites already do some of this

- Arts and Crafts platform Etsy already embraced the idea of *graph of identity*

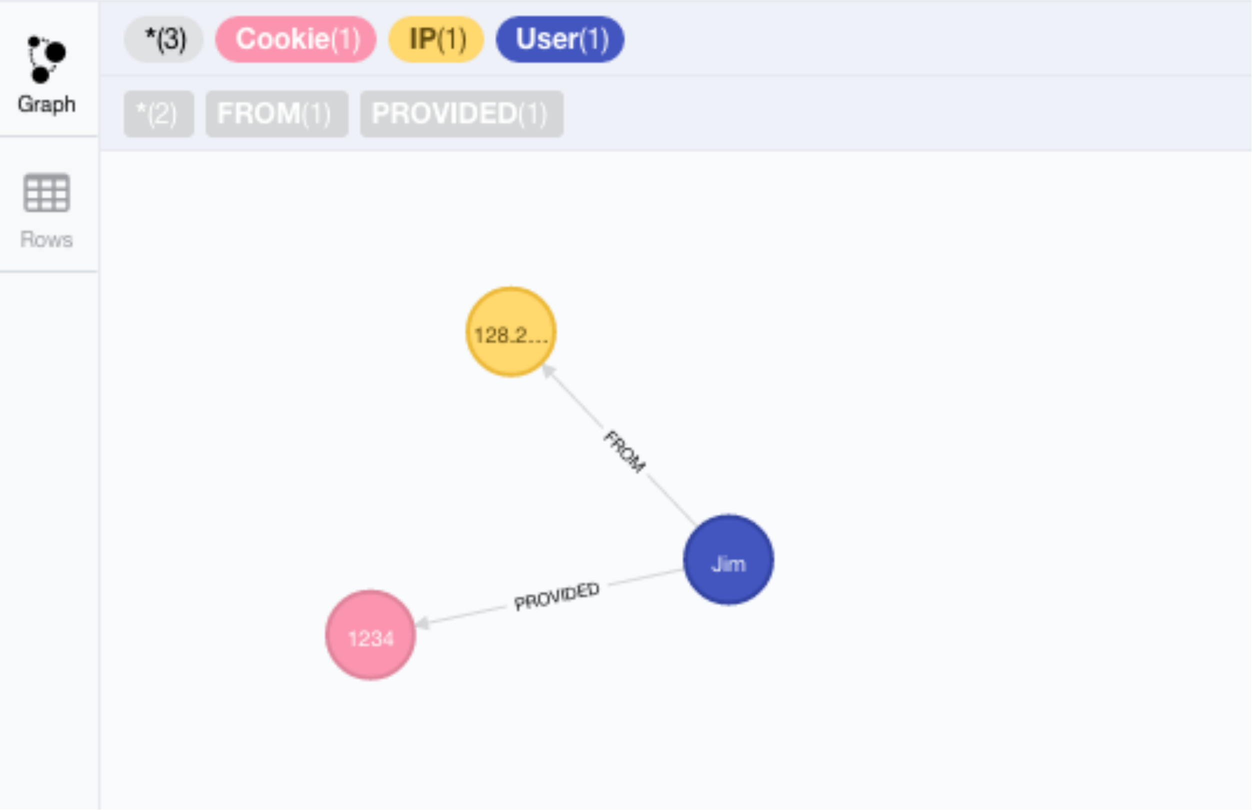
An Individual Identity Subgraph



fred@rbs.co.uk
1234LOL

128.240.229.18

We are all made of stars...



Specific Weighted Identity Query

```
MATCH (u:User {username:'Jim', password: 'secret'})
```

Bare Minimum

```
OPTIONAL MATCH
   (u) -[cookie:PROVIDED]->(:Cookie {id:'1234'})
```

Other Specific Considerations

```
OPTIONAL MATCH
   (u)-[address:FROM]->(:IP {network:'128.240.0.0'})
```

Other Specific Considerations

```
RETURN  SUM(cookie.weighting) + SUM(address.weighting)
  AS score
```

Final Decision

General Weighted Identity Query

```
MATCH (u:User {username:'Jim', password: 'secret'})
```

Bare Minimum

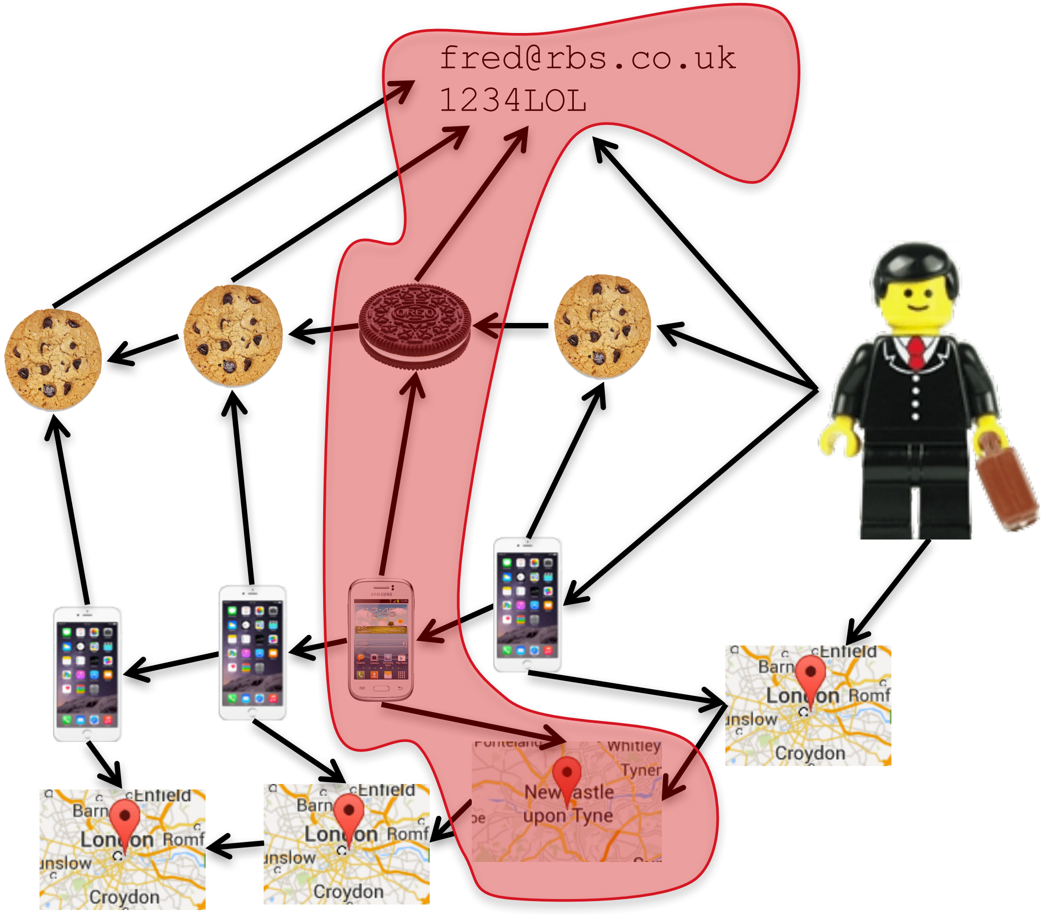
```
OPTIONAL MATCH (u)-[rel]->()
WHERE has(rel.weighting)
```

All Available Weightings

```
RETURN SUM(rel.weighting) AS score
```

Final Decision

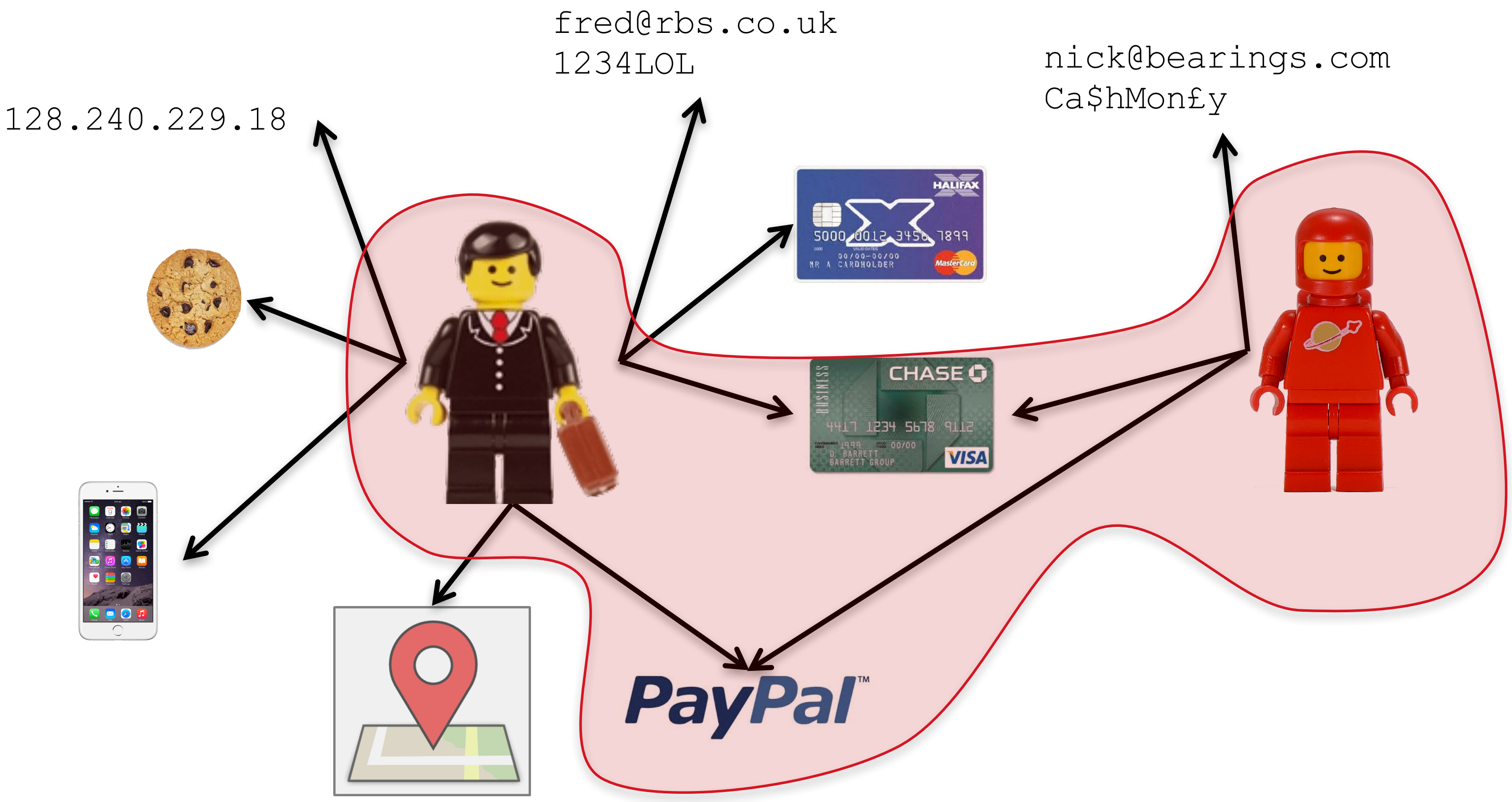
An Individual Login History



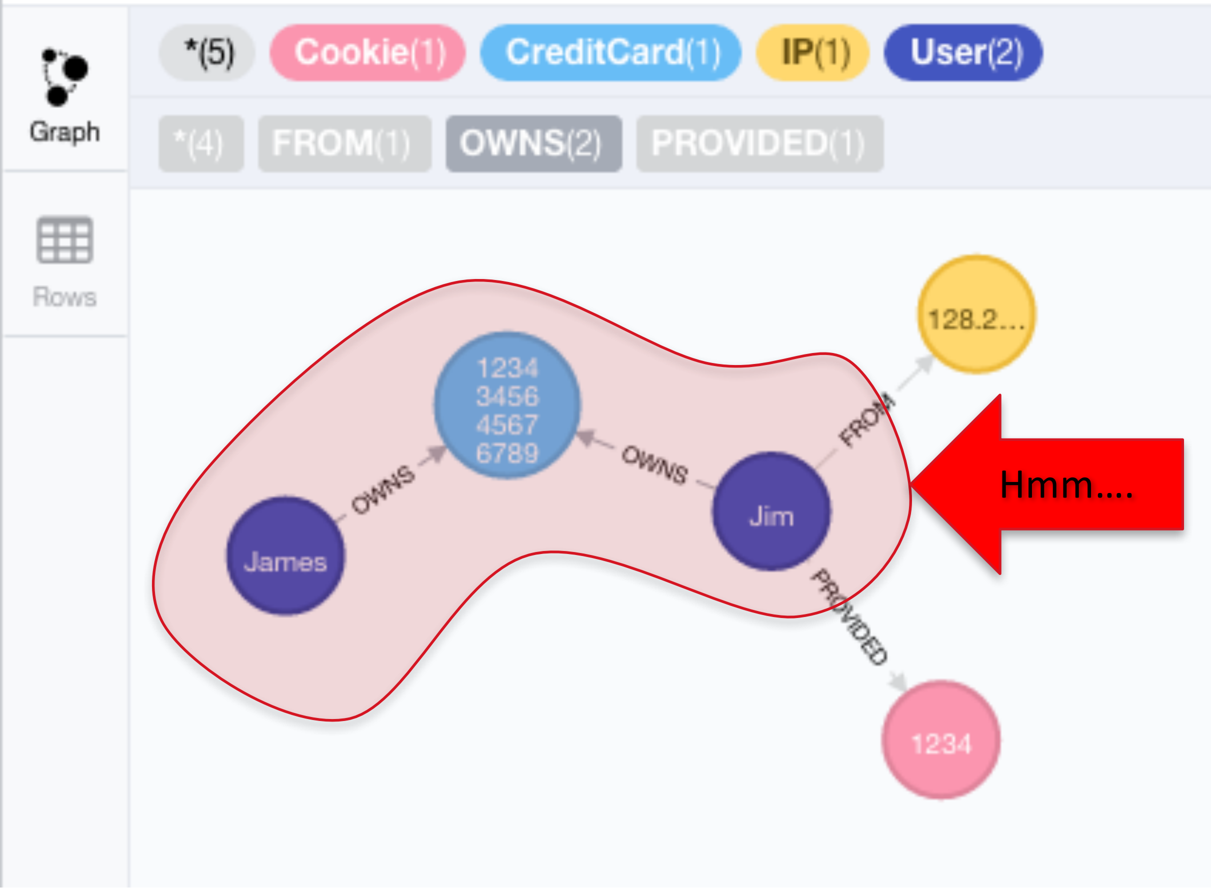
From 1st to 3rd Party

- The 1st party identity graph can easily be extended to 3rd party fraud
- Like in the bank fraud ring, fraudsters can mix-n-match claims
- Start with a few phished accounts and expand from there!

Shared Connections



Graphing Shared Connections



Hmm….

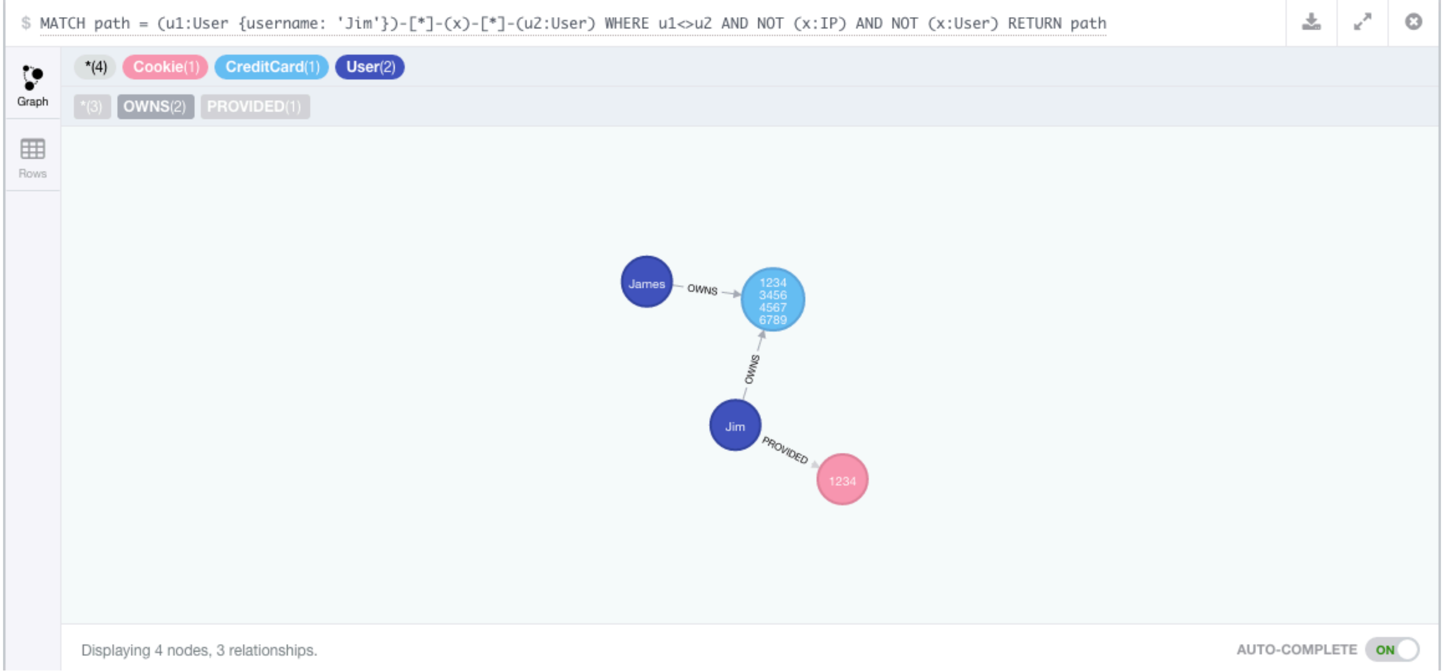
Scan for Potential Fraudsters

```
MATCH (u1:User)--(x)--(u2:User)
WHERE u1 <> u2 AND NOT (x:IP)
RETURN x
```

Network in common is OK

Stop specific fraudster network, quickly

```
MATCH path =
   (u1:User {username: 'Jim'})-[*]-(x)-[*]-(u2:User)
WHERE u1<>u2 AND NOT (x:IP) AND NOT (x:User)
RETURN path
```

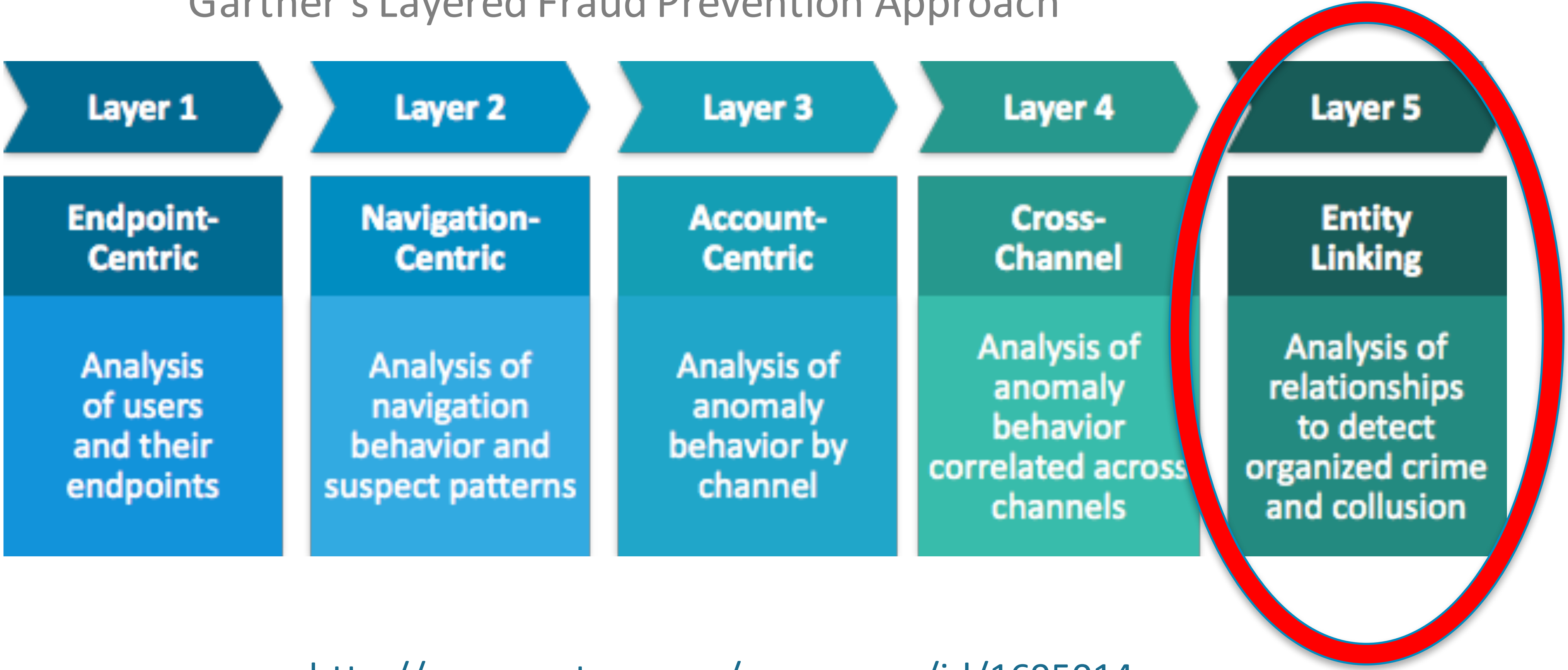


```
$ MATCH path = (u1:User {username: 'Jim'})-[*]-(x)-[*]-(u2:User) WHERE u1<>u2 AND NOT (x:IP) AND NOT (x:User) RETURN path
```

Displaying 4 nodes, 3 relationships.

How do these fit with traditional fraud prevention?

Gartner’s Layered Fraud Prevention Approach



Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Endpoint-Centric	Navigation-Centric	Account-Centric	Cross-Channel	Entity Linking
Analysis of users and their endpoints	Analysis of navigation behavior and suspect patterns	Analysis of anomaly behavior by channel	Analysis of anomaly behavior correlated across channels	Analysis of relationships to detect organized crime and collusion

http://www.gartner.com/newsroom/id/1695014

“Chronic Master Data”

Master Data Management

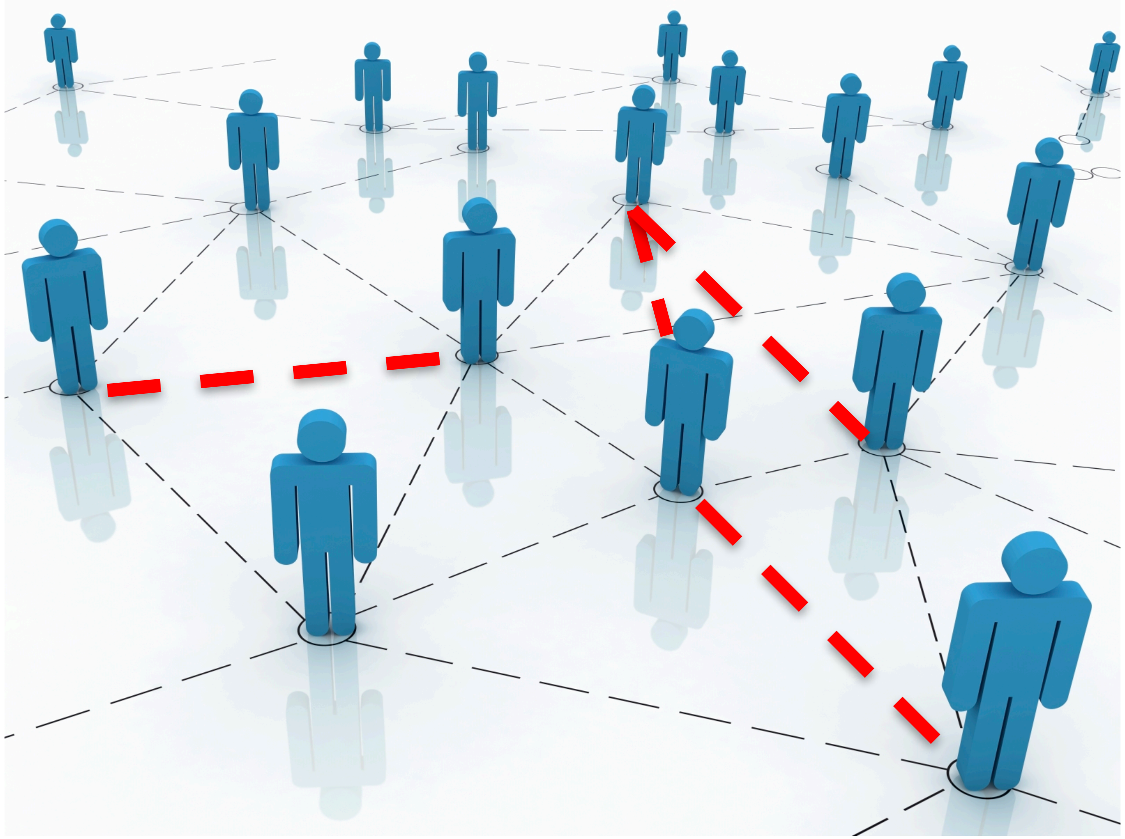
- Provide high quality, joined up data to the right process at the right time
- Bridge silos, leverage all data (including legacy)

- Database point of view: fancy indexes
- Graph database point of view: a Web of data
  - Multidimensional, path-centric index

Master Data Management Examples

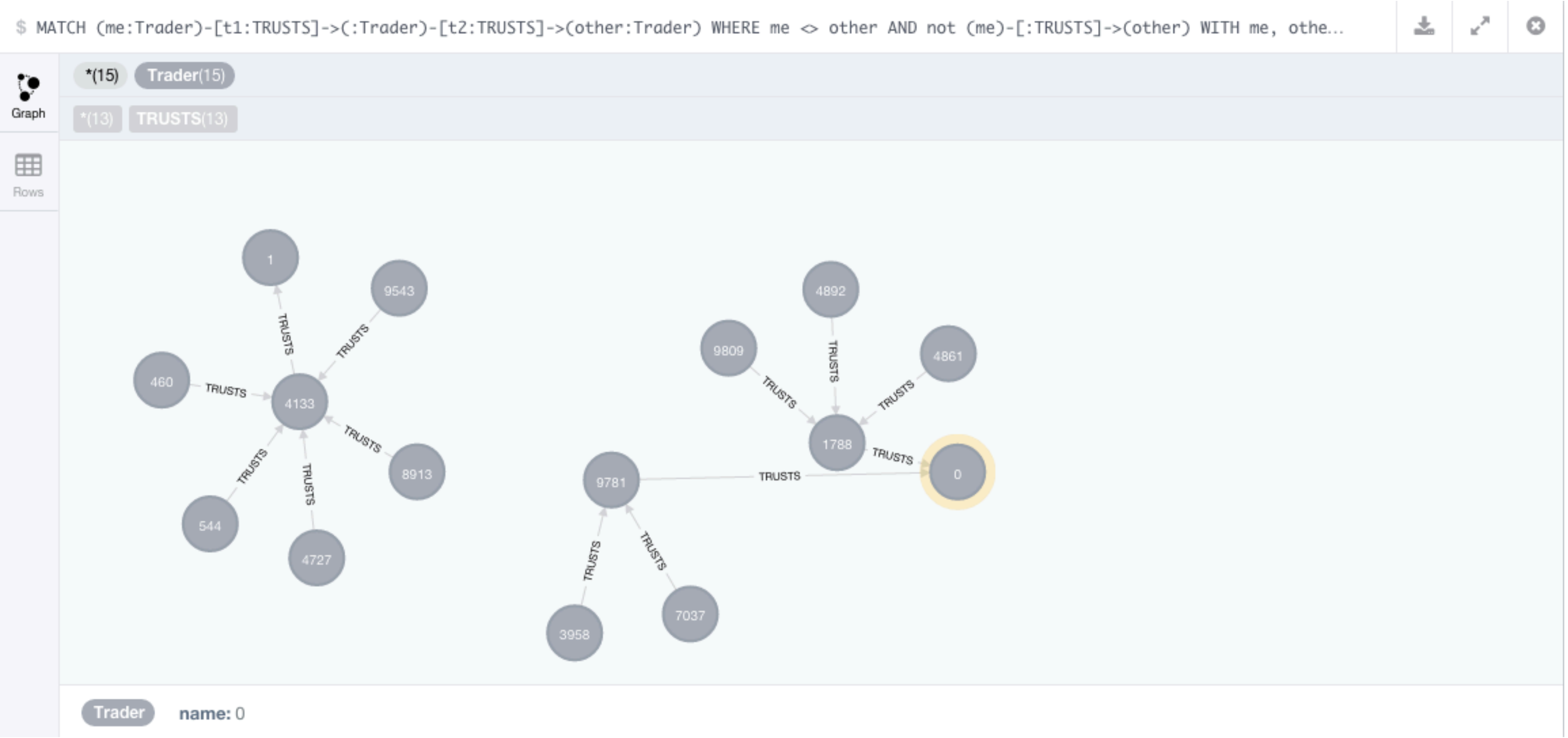
- Adidas: Shared Metadata Service
  - 360 degree view of data via the graph
  - Without disturbing existing (valuable) systems!
- ICE: Global directory for participants, market makers, investment funds etc.
  - Futures and trading house
  - Social network for brokers
    - Recommendations for the right broker means more business!
    - Recommendations are trivial in a graph
- Pitney Bowes productised platform on top of Neo4j
  - Materially affected their stock rating
  - http://www.zacks.com/stock/news/157741/pitney-bowes-selects-neo4j-to-develop-graphbased-mdm

Easy Recommendations: Triadic Closure



http://www.isciencemag.co.uk/blog/are-you-a-social-network-junkie/

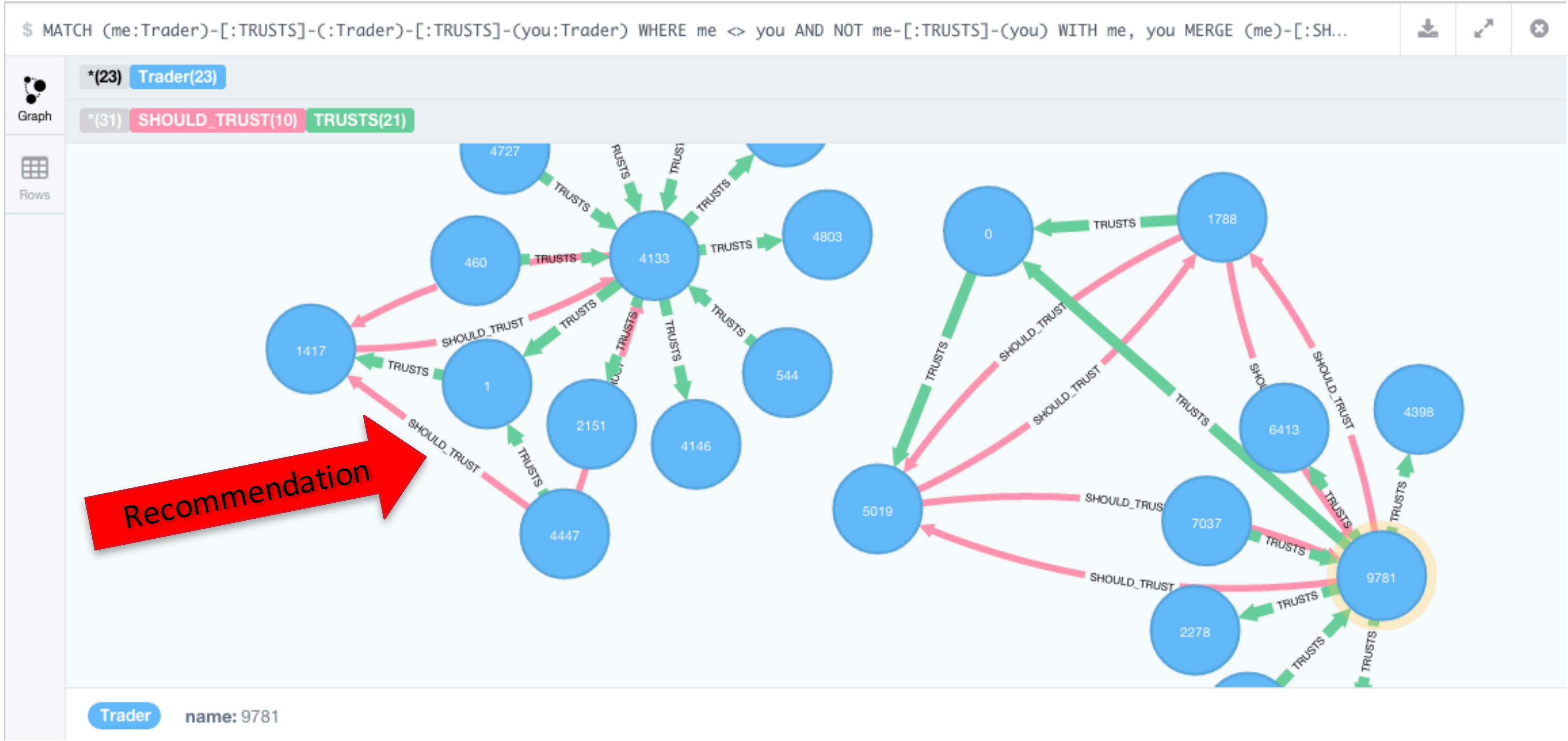
Triadic Closure (1)



```
$ MATCH (me:Trader)-[t1:TRUSTS]->(:Trader)-[t2:TRUSTS]->(other:Trader) WHERE me <> other AND not (me)-[:TRUSTS]->(other) WITH me, othe…
```

Trader name: 0

Triadic Closure (2)



```
$ MATCH (me:Trader)-[:TRUSTS]-(:Trader)-[:TRUSTS]-(you:Trader) WHERE me <> you AND NOT me-[:TRUSTS]-(you) WITH me, you MERGE (me)-[:SH…
```

*(23) Trader(23)

*(31) SHOULD_TRUST(10) TRUSTS(21)

Trader name: 9781

Recommendation

Easy Global Query

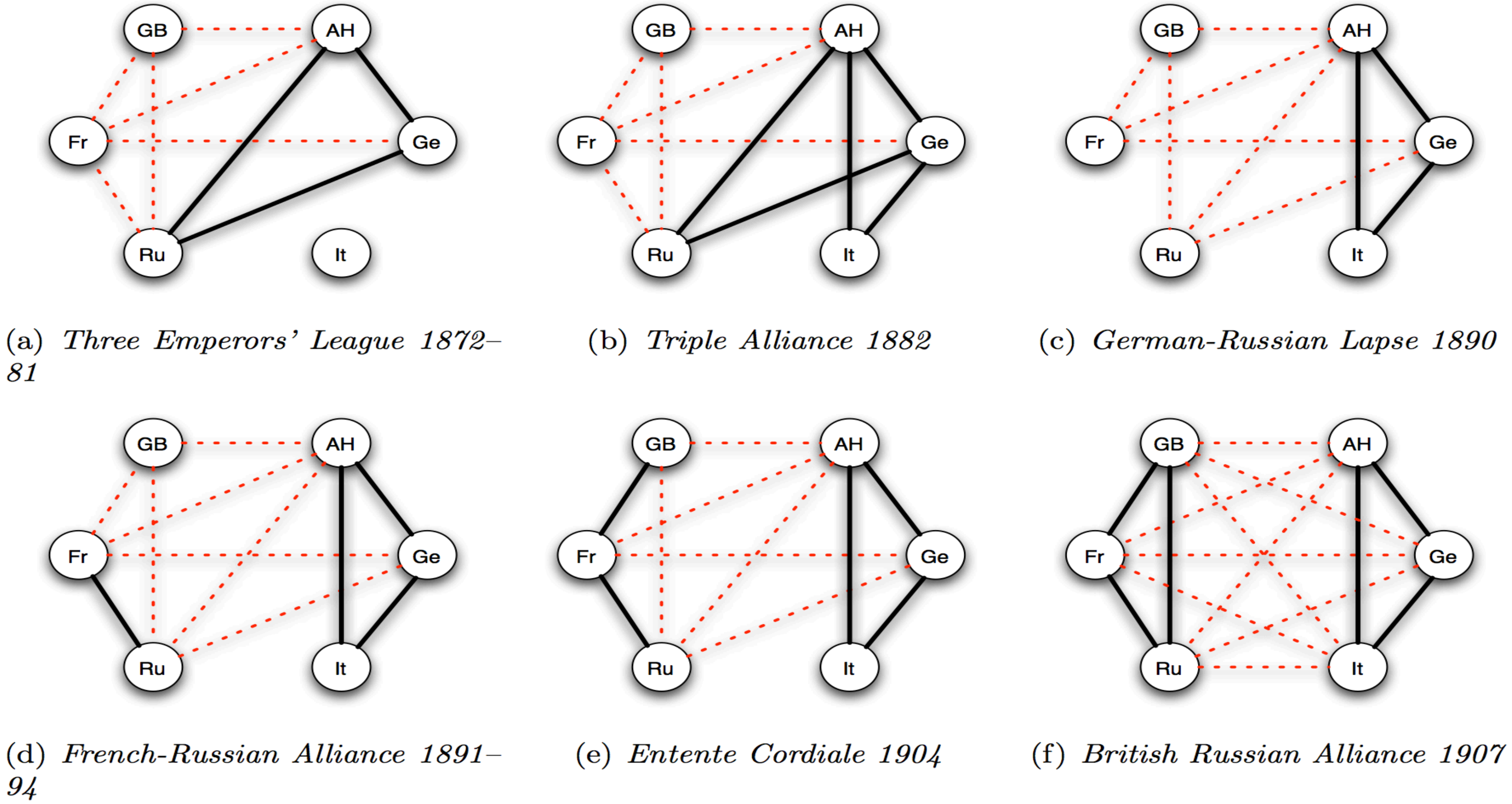
```
MATCH (me:Trader)-[:TRUSTS]-
        (:Trader)-[:TRUSTS]-(you:Trader)
WHERE me <> you AND NOT me-[:TRUSTS]-(you)
WITH me, you
MERGE (me)-[:TRUSTS]->(you)
RETURN me, you
```

Or Super-fast Local Query

```
MATCH (me:Trader name:'Ed')-[:TRUSTS]-
        (:Trader)-[:TRUSTS]-(you:Trader)
WHERE me <> you AND NOT me-[:TRUSTS]-(you)
WITH me, you
MERGE (me)-[:TRUSTS]->(you)
RETURN me, you
```

Side note: Triadic Closures Predict 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*

What has this to do with stopping fraud?

neo4j

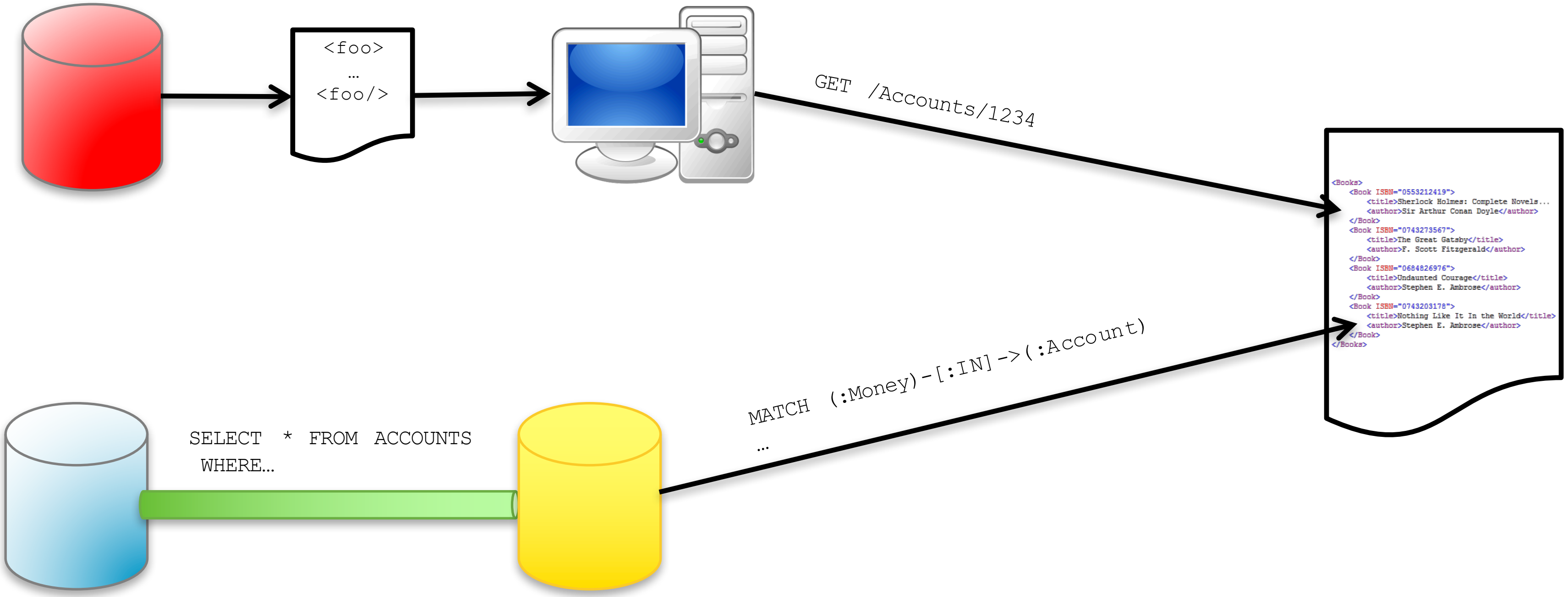
- Recommendations are a positive version of anti-recommendations
- Identifying fraud is an anti-recommendation
- So you can use triadic closure to try to identify networks of fraudsters and their targets via transitive relations

“False Provenance”

Provenance

- Banks are awash with data
  - And spend a lot of time moving and transforming it
- Where did **this** data come from?
  - Compliance and auditors want to know
- How do I show how this data got computed/transformed/moved?

It's a graph!



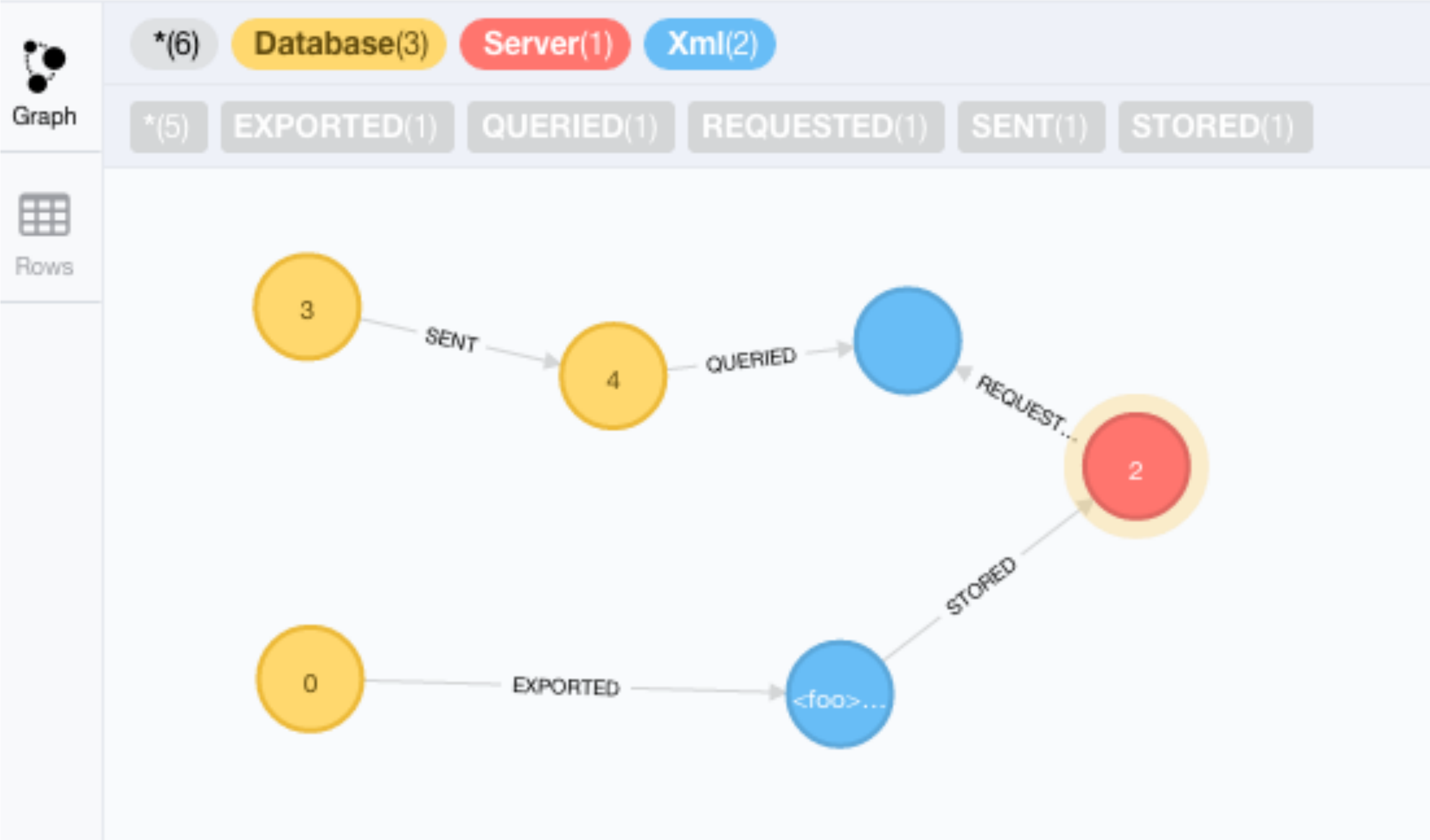
```
<foo>
…
<foo/>
```

GET /Accounts/1234

SELECT * FROM ACCOUNTS
WHERE…

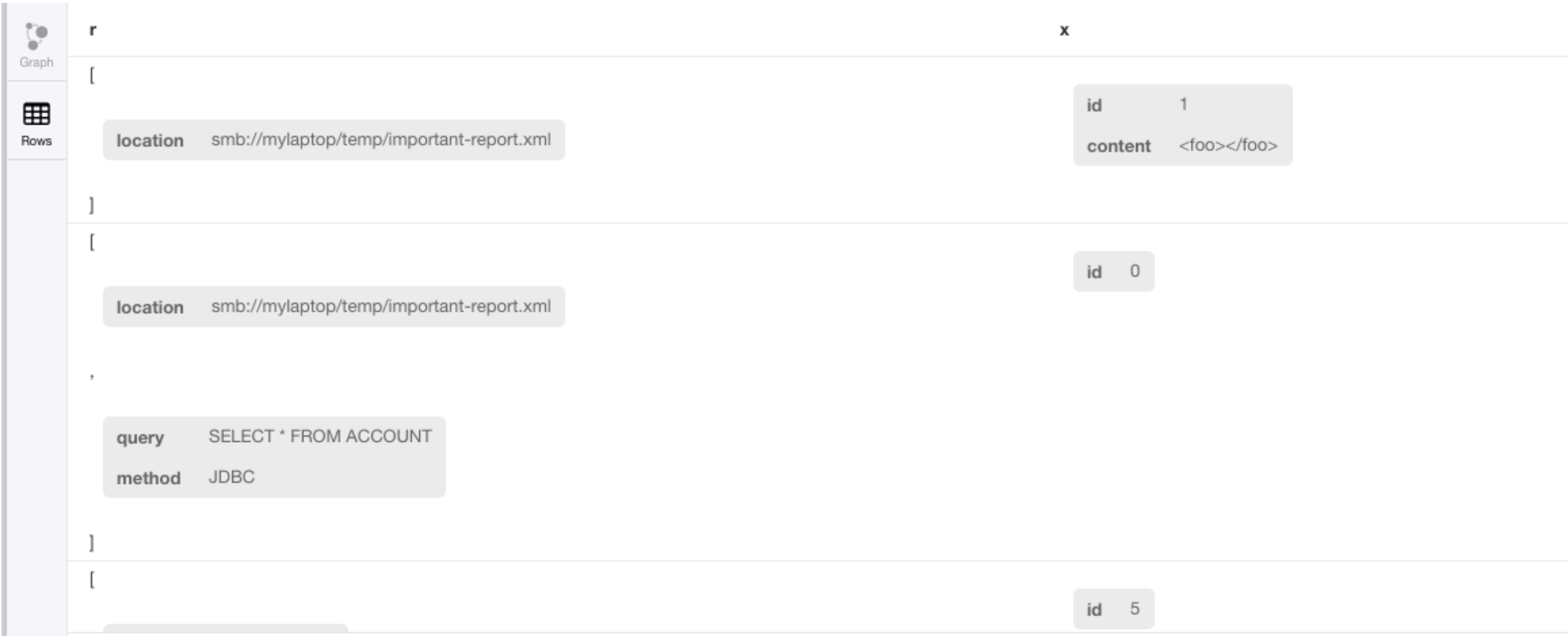
MATCH (:Money)-[:IN]->(:Account)
…

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```


*(6)
Database(3)
Server(1)
Xml(2)
*(5)
EXPORTED(1)
QUERIED(1)
REQUESTED(1)
SENT(1)
STORED(1)
Graph
Rows
3
SENT
4
QUERIED
REQUEST...
2
STORED
0
EXPORTED
<foo>...

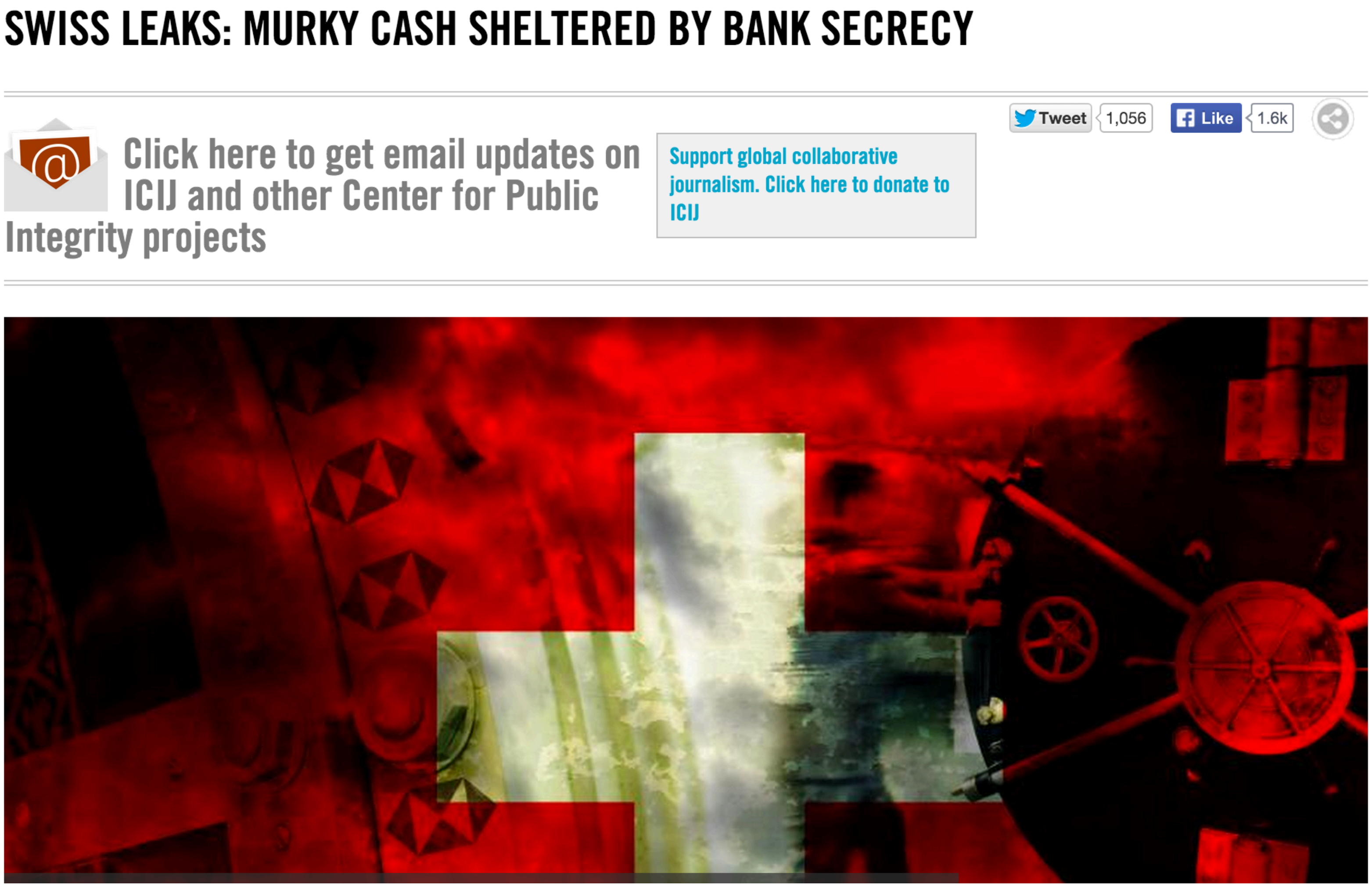
Detailed Provenance

```
MATCH (:Server {id: 2})-[r*]-(x)
RETURN x, r
```



```
r                                                          x
[
  location  smb://mylaptop/temp/important-report.xml       id       1
                                                           content  <foo></foo>
]
[
  location  smb://mylaptop/temp/important-report.xml       id  0
,
  query   SELECT * FROM ACCOUNT
  method  JDBC
]
[
                                                           id  5
```

“Lack of Governance”



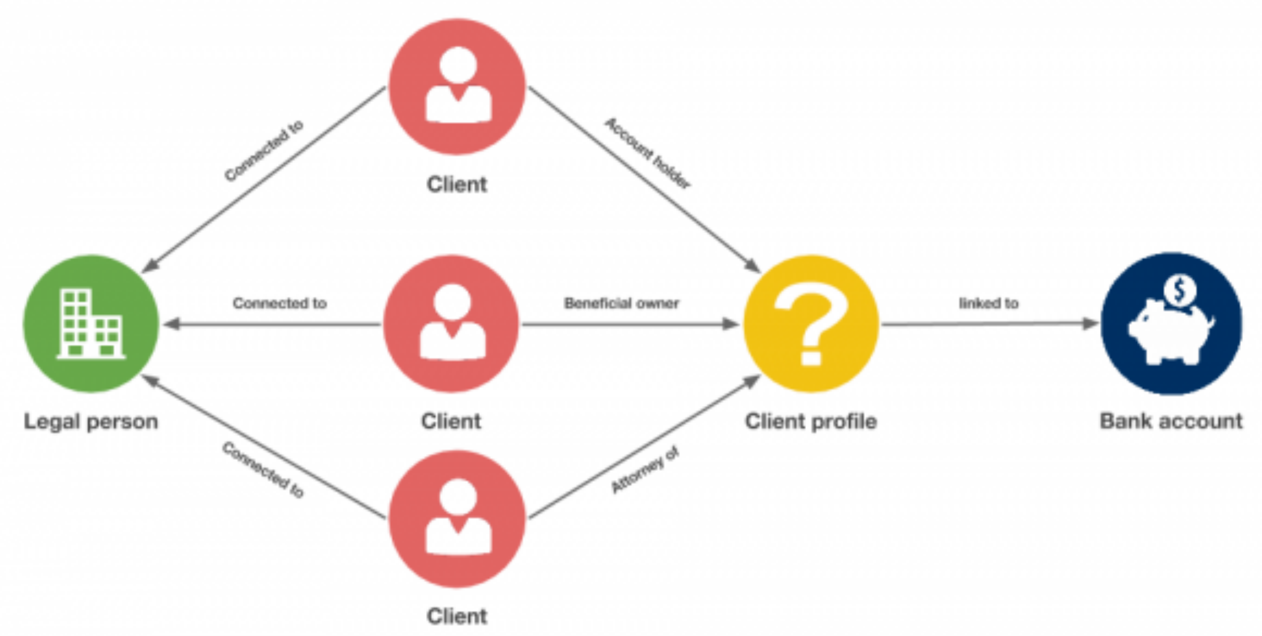
SWISS LEAKS: MURKY CASH SHELTERED BY BANK SECRECY

Click here to get email updates on ICIJ and other Center for Public Integrity projects

Support global collaborative journalism. Click here to donate to ICIJ

Poor Governance needs Good Graphs

- The Swissleaks episode caused substantial reputational harm to HSBC
  - Loss of revenue, legal costs
- Banks live and die on having a trustworthy reputation
- Compliance officers are overwhelmed by volume and traditional methods



Good data, Great Journalism

- Swissleaks may have been great journalism
  - It was! They’re heroes.
- But the tools that used could have been used to stop illegal behaviour long before it reached the press
- Neo4j **should** be used by every compliance office in every bank

- The ICIJ is like Jepsen for businesses.
- You should run the tools on your business before they do it for you!

Thanks for listening

@jimwebber

neo4j