THE PATH FORWARD

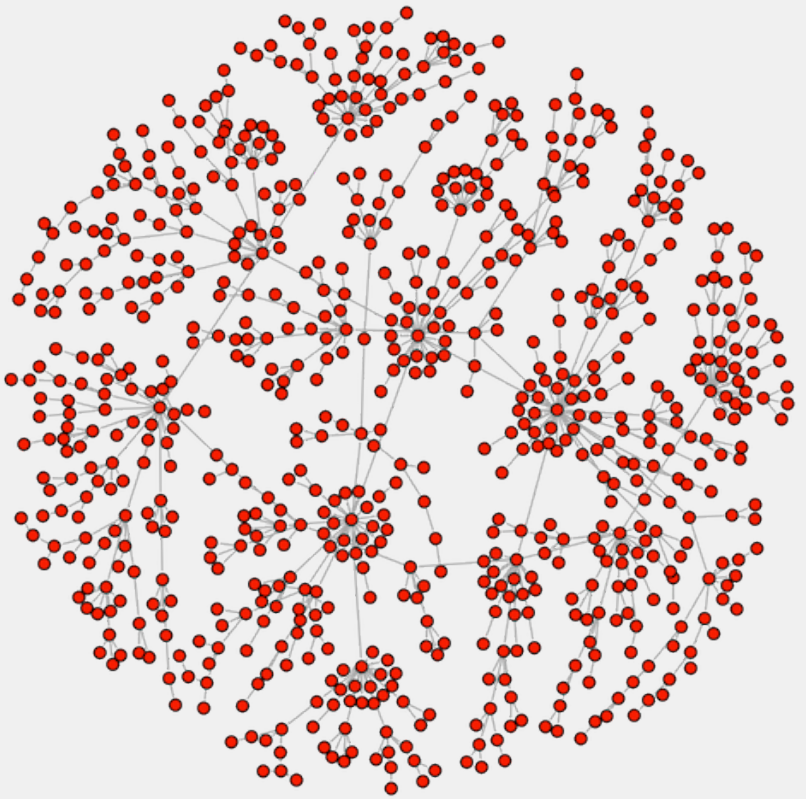
TITAN 1.0 + TINKERPOP3

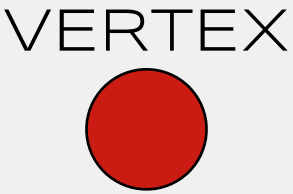
MARKO A. RODRIGUEZ

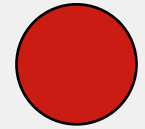
HTTP://THINKAURELIUS.COM
HTTP://TINKERPOP.COM

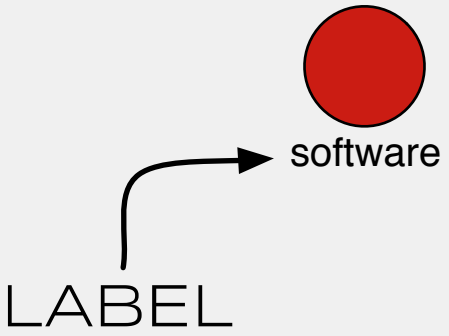
AURELIUS

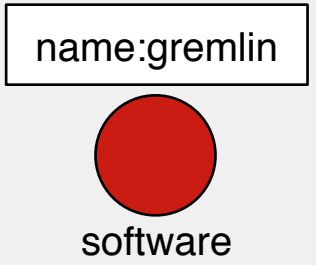
PART 1:
INTRODUCTION TO GRAPHS

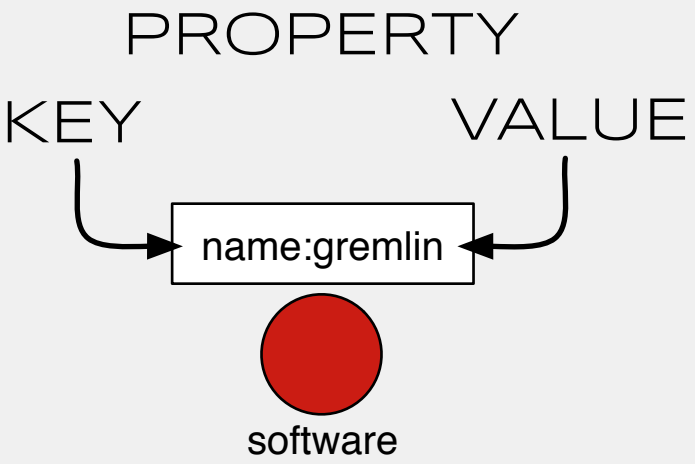


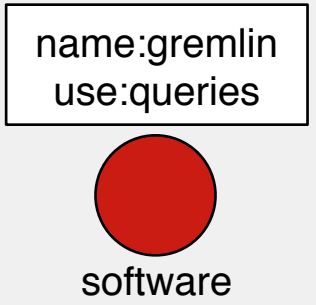

VERTEX

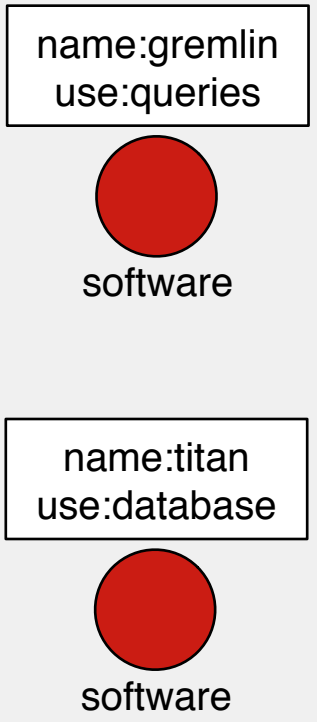

software

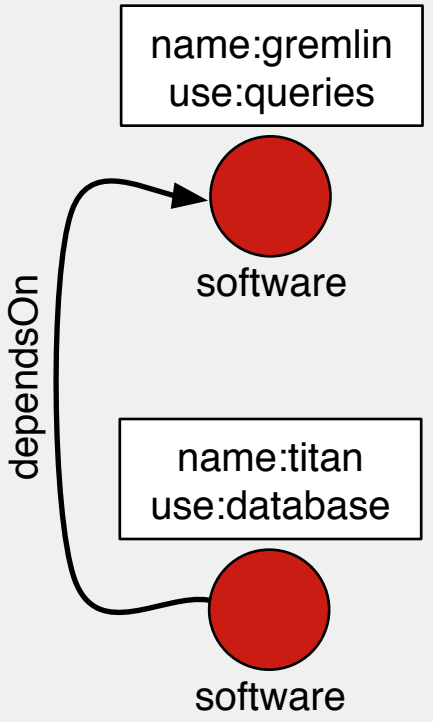

software
LABEL

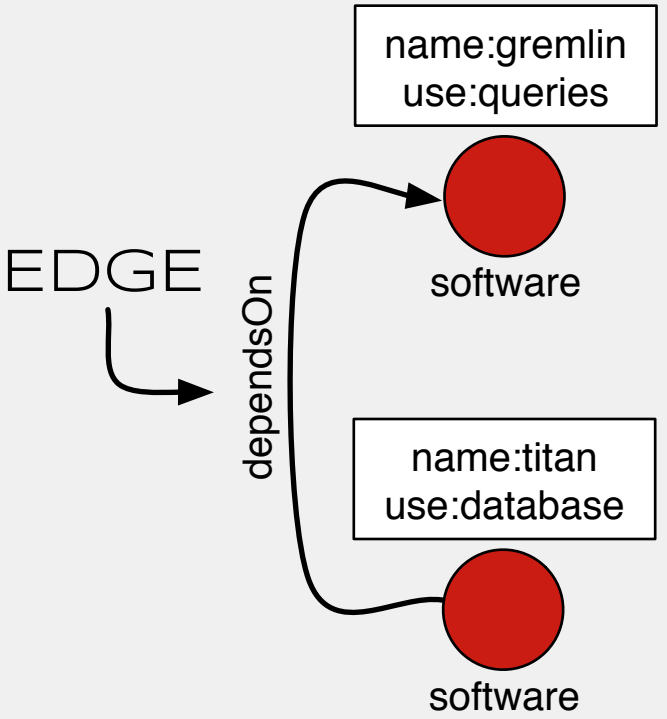

name:gremlin
software

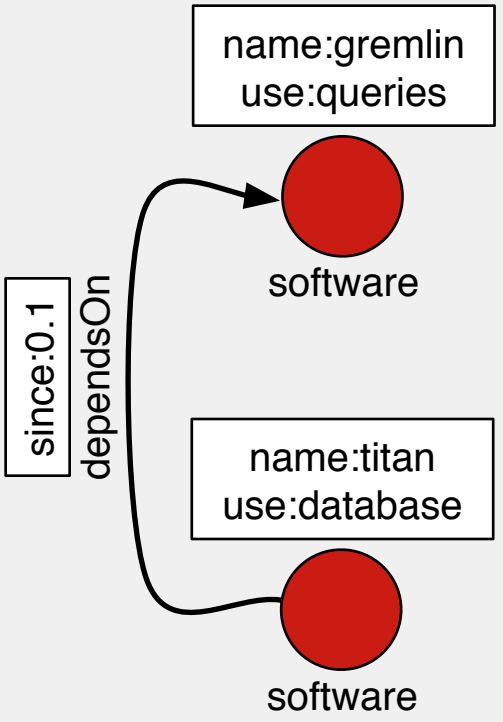

PROPERTY
KEY
VALUE
name:gremlin
software


name:gremlin
use:queries
software

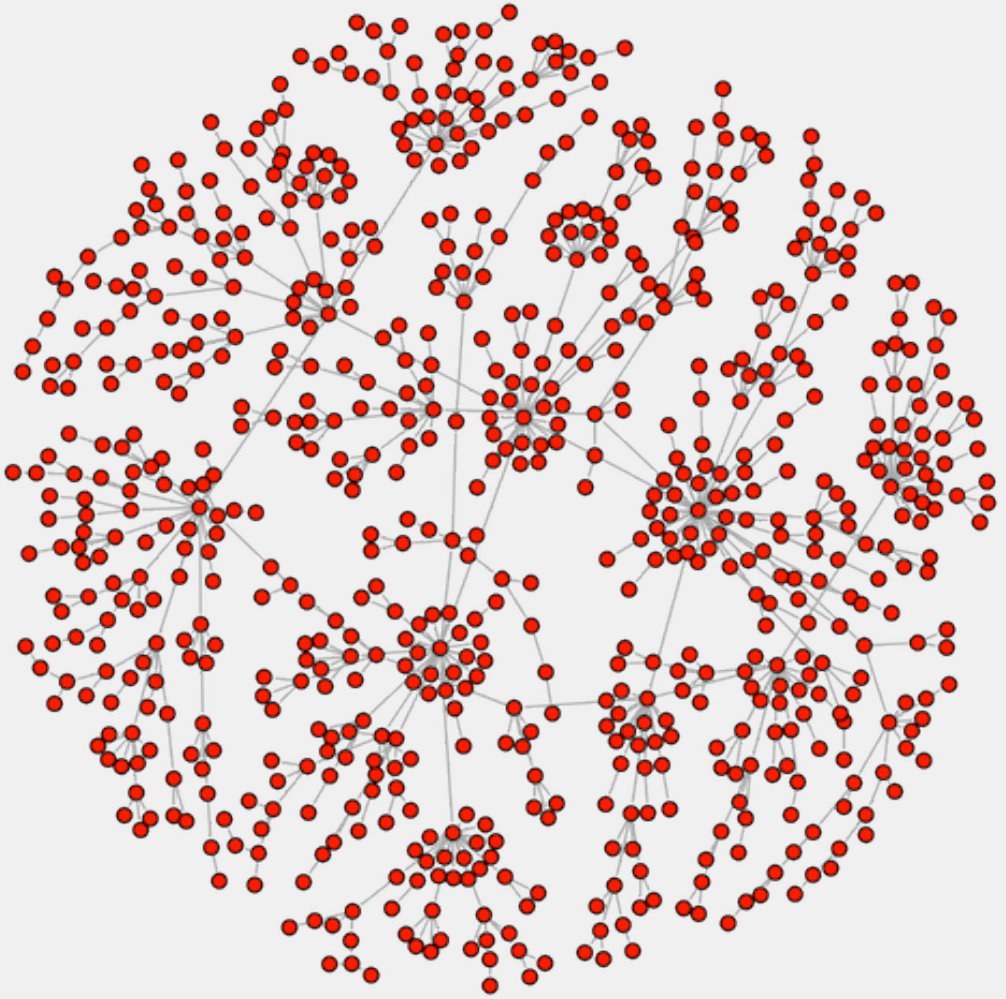

name:gremlin
use:queries
software
name:titan
use:database
software

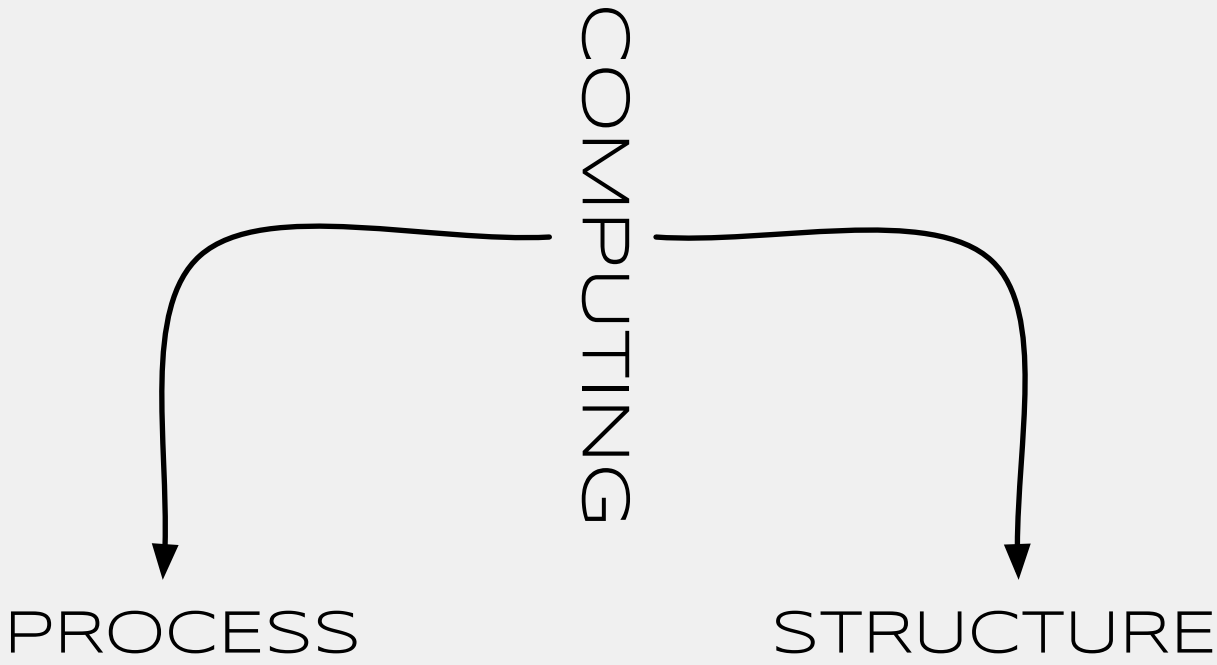

name:gremlin
use:queries
software
dependsOn
name:titan
use:database
software


name:gremlin
use:queries
EDGE
dependsOn
software
name:titan
use:database
software


name:gremlin
use:queries
software
since:0.1
dependsOn
name:titan
use:database
software

GRAPH

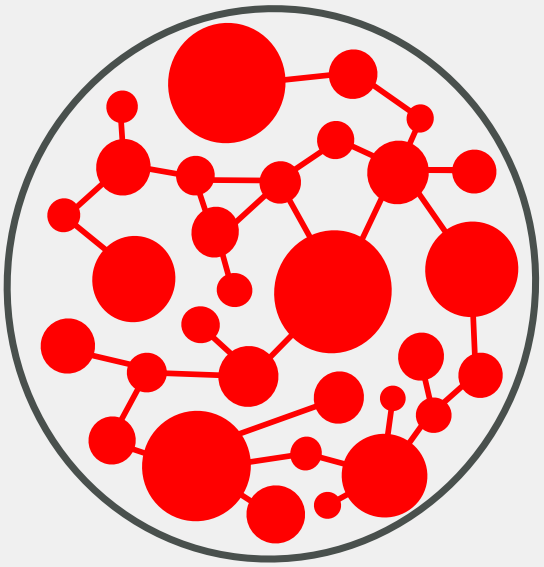



COMPUTING
PROCESS
STRUCTURE

COMPUTING

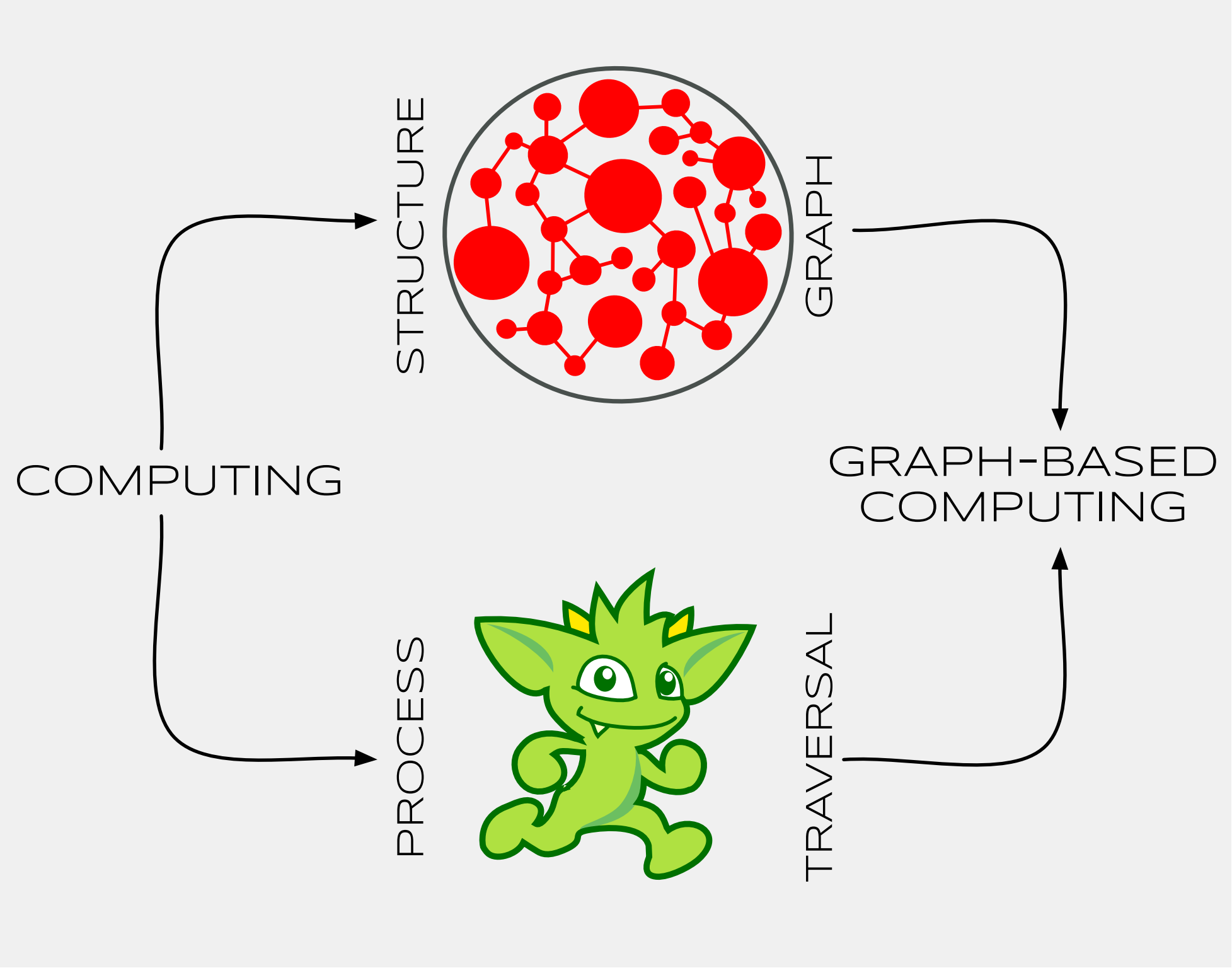
PROCESS

STRUCTURE



TRAVERSAL

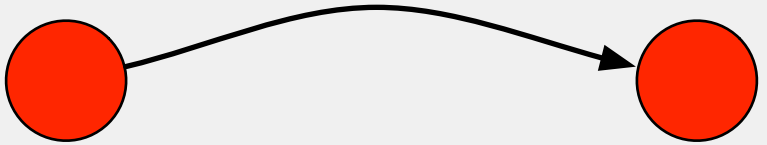
GRAPH


COMPUTING
STRUCTURE
GRAPH
GRAPH-BASED
COMPUTING
PROCESS
TRAVERSAL

WHY GRAPH-BASED COMPUTING?

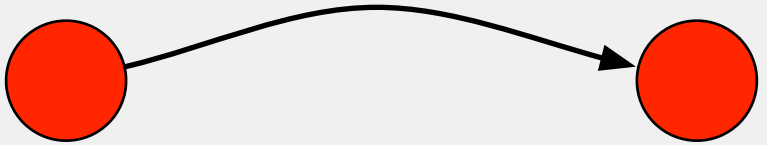
WHY GRAPH-BASED COMPUTING?

INTUITIVE MODELING

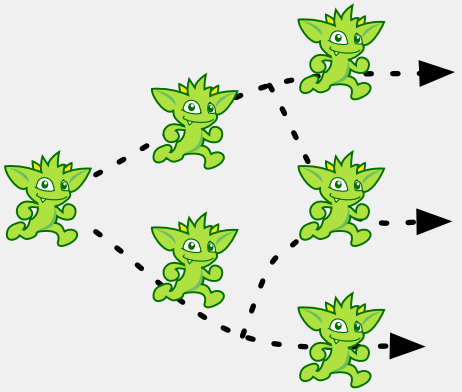


WHY GRAPH-BASED COMPUTING?

INTUITIVE MODELING

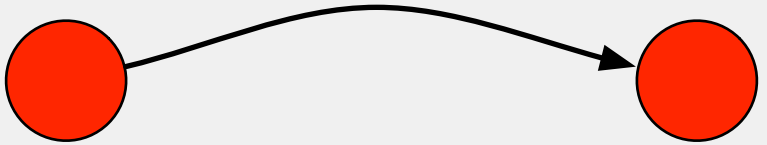


EXPRESSIVE QUERYING

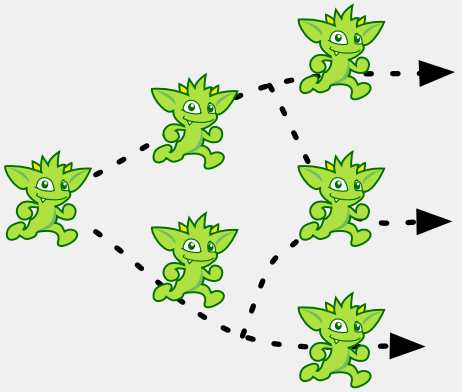


WHY GRAPH-BASED COMPUTING?

INTUITIVE MODELING



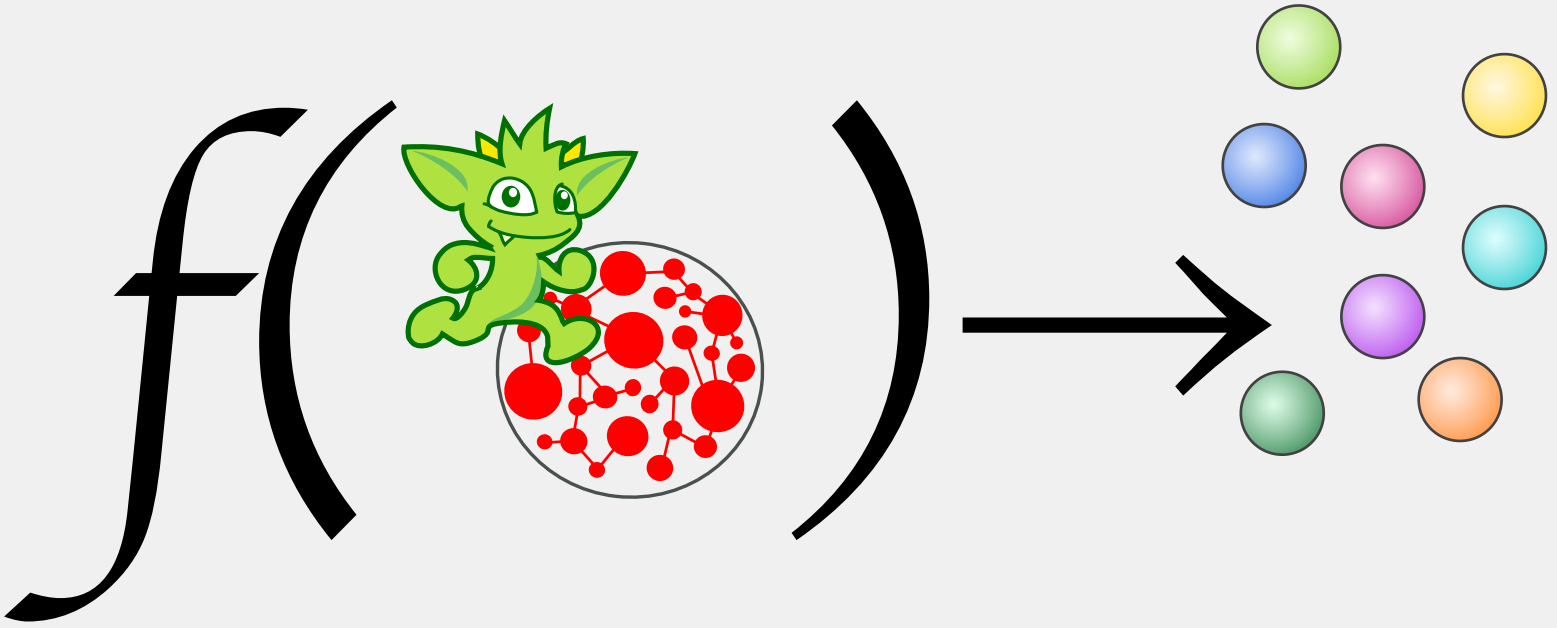
EXPRESSIVE QUERYING



NUMEROUS ANALYSES

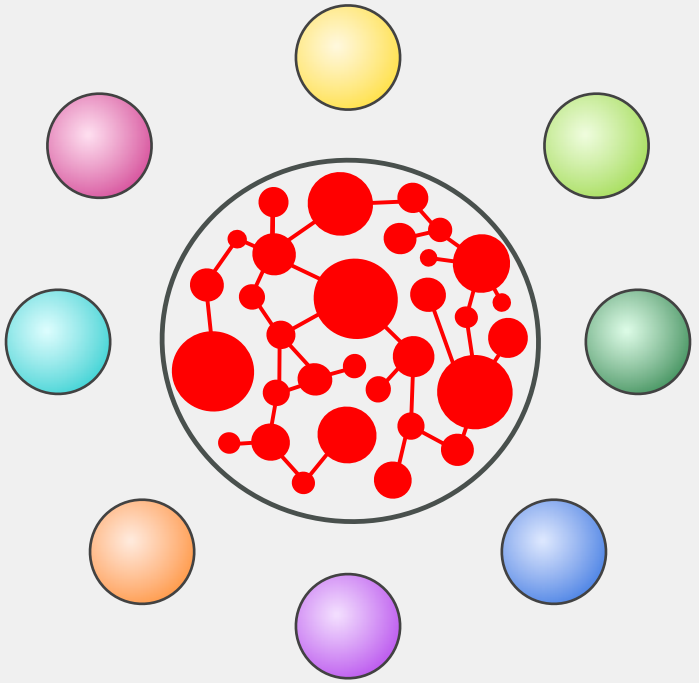
- Inference
- Mixing Patterns
- Ranking
- Path Expressions
- Motifs
- Centrality
- Scoring
- Geodesics

ANALYSES ARE THE EPIPHENOMENA OF TRAVERSAL



WHAT IS THE SIGNIFICANCE OF GRAPH ANALYSIS?

ANALYSES YIELD INSIGHTS ABOUT THE MODEL



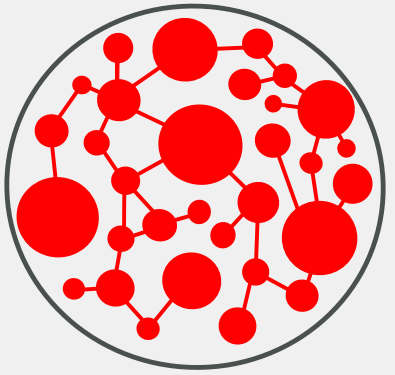
=

DATA PRODUCTS

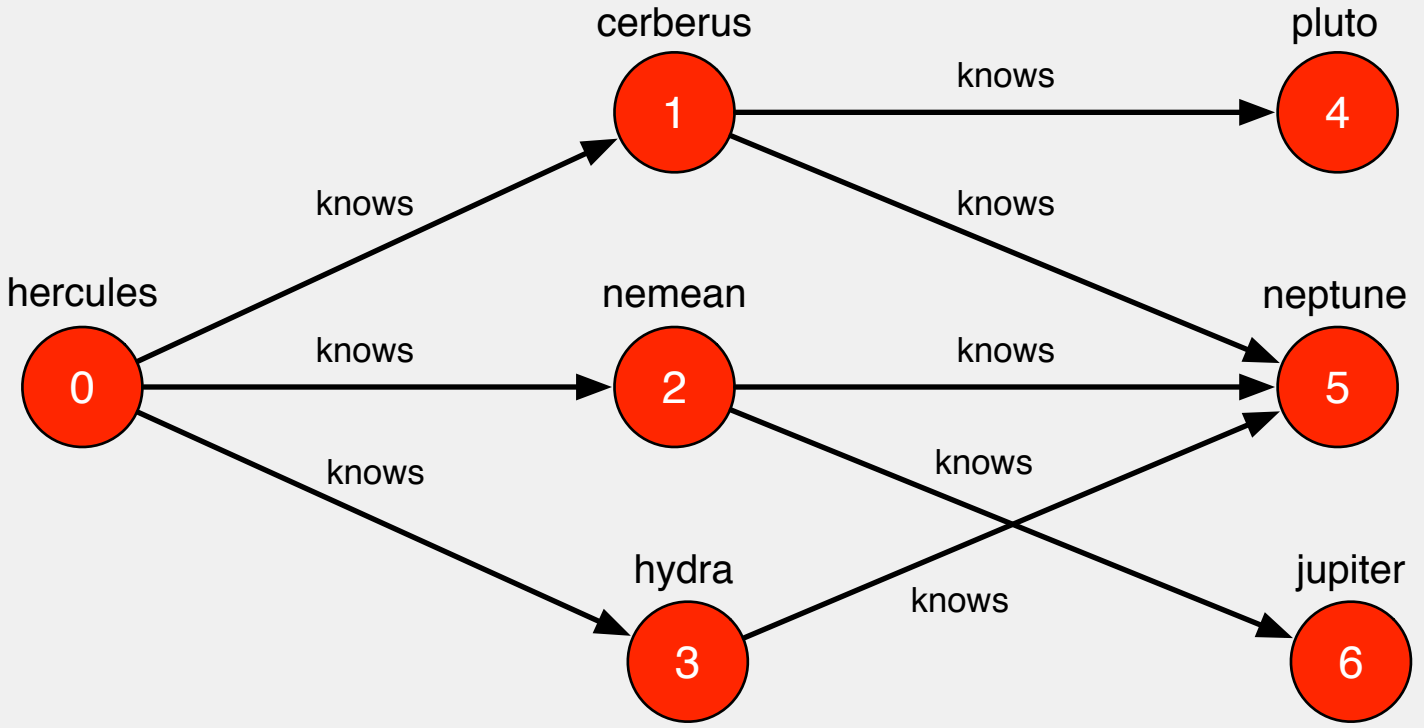
DATA-DRIVEN DECISION SUPPORT

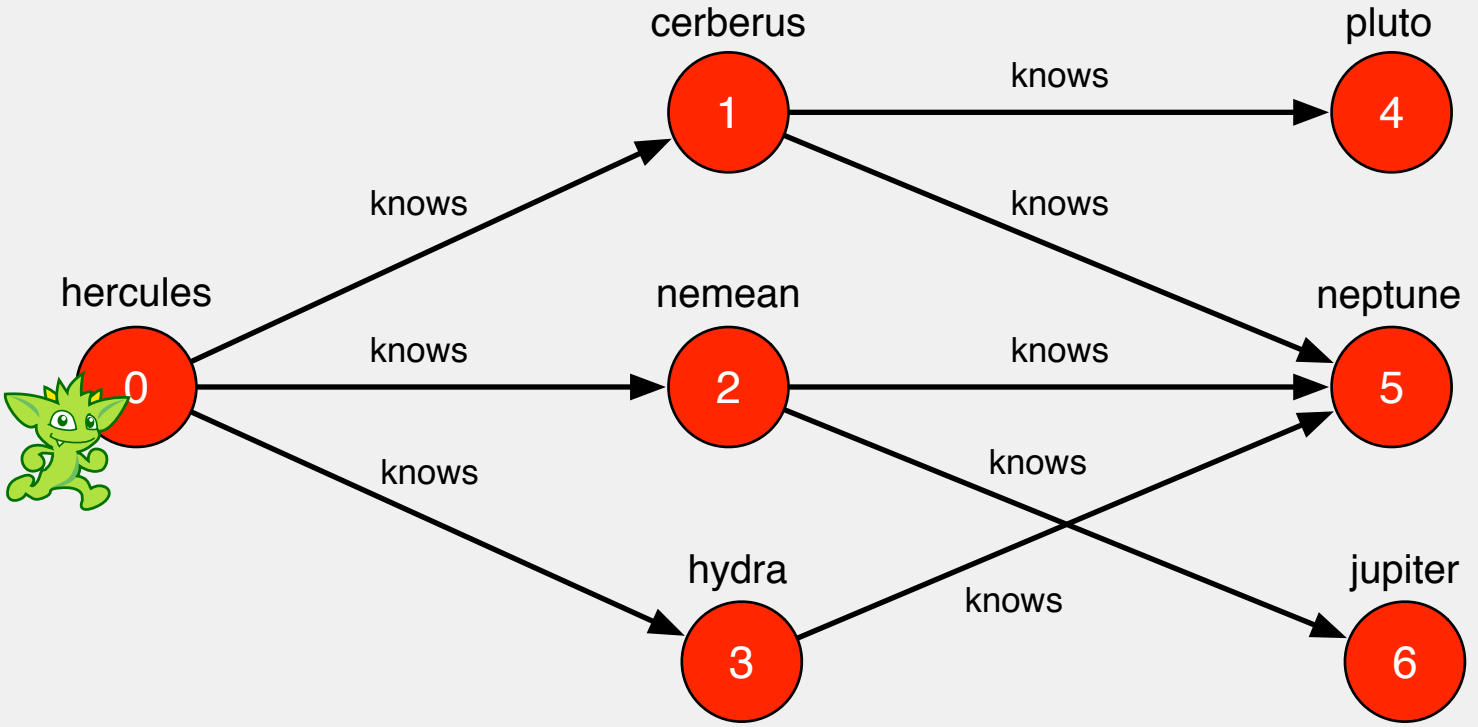
RECOMMENDATION

- People you may know. **SOCIAL GRAPH**
- Products you might like. **RATINGS GRAPH**
- Movies you should watch and the friends you should watch them with. **SOCIAL+RATINGS GRAPH**

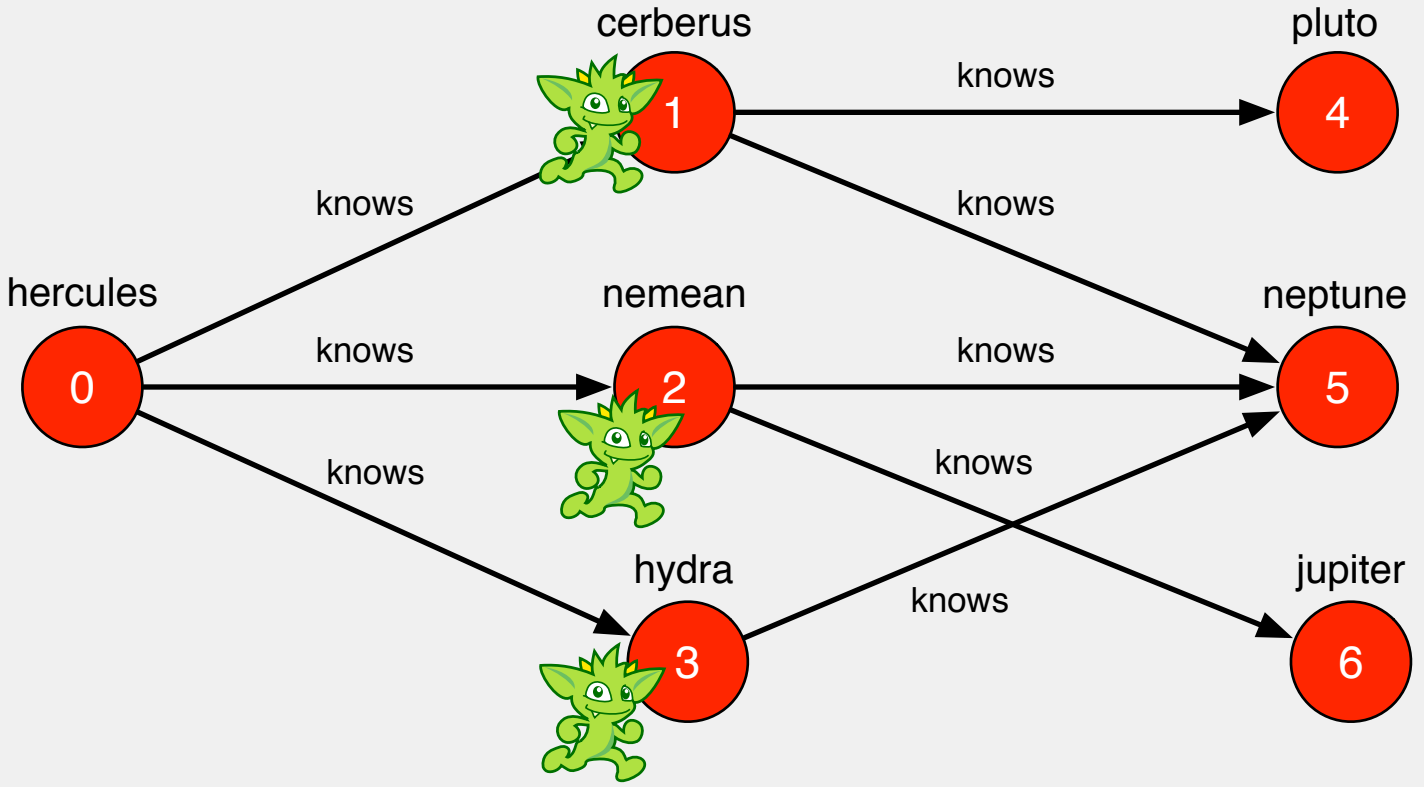


WHO ELSE MIGHT HERCULES KNOW?

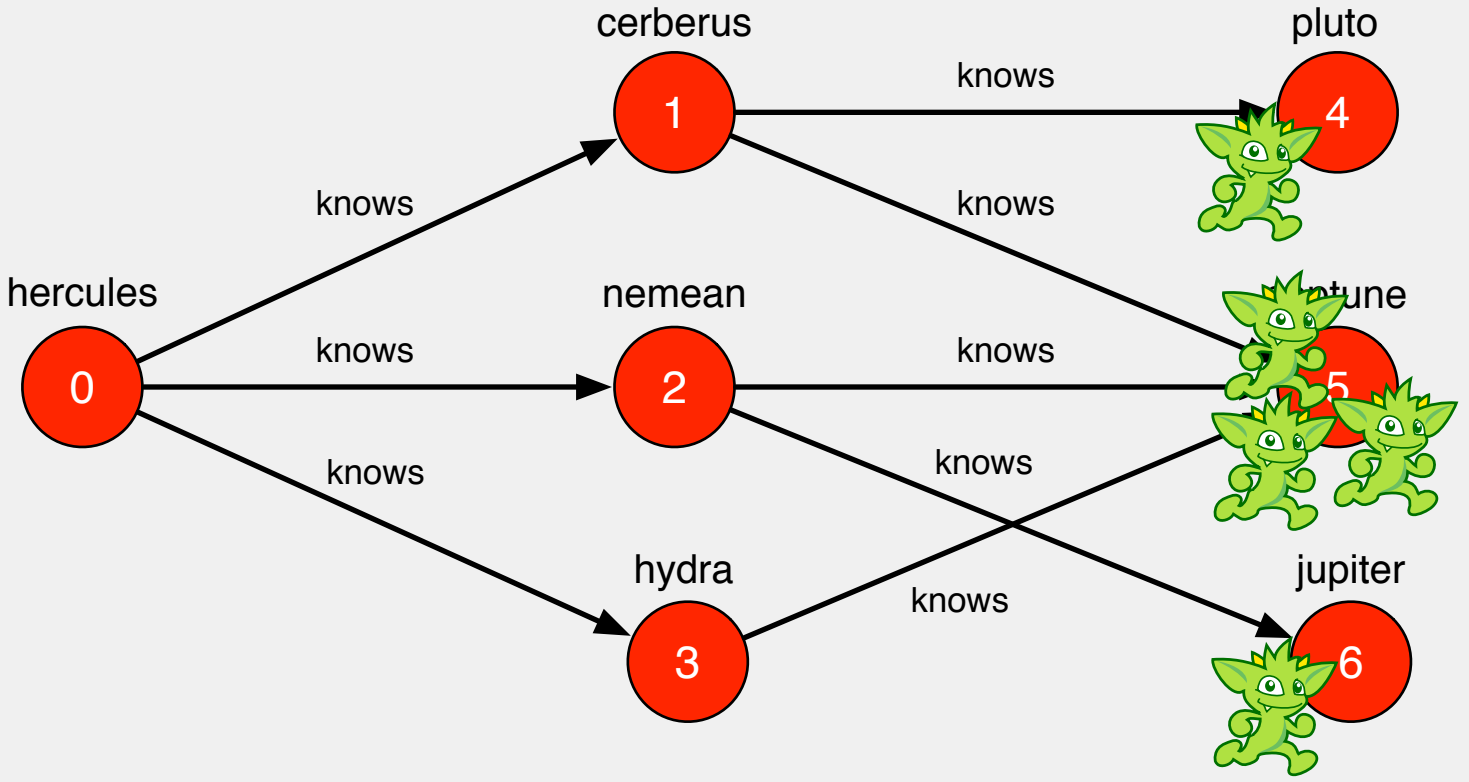




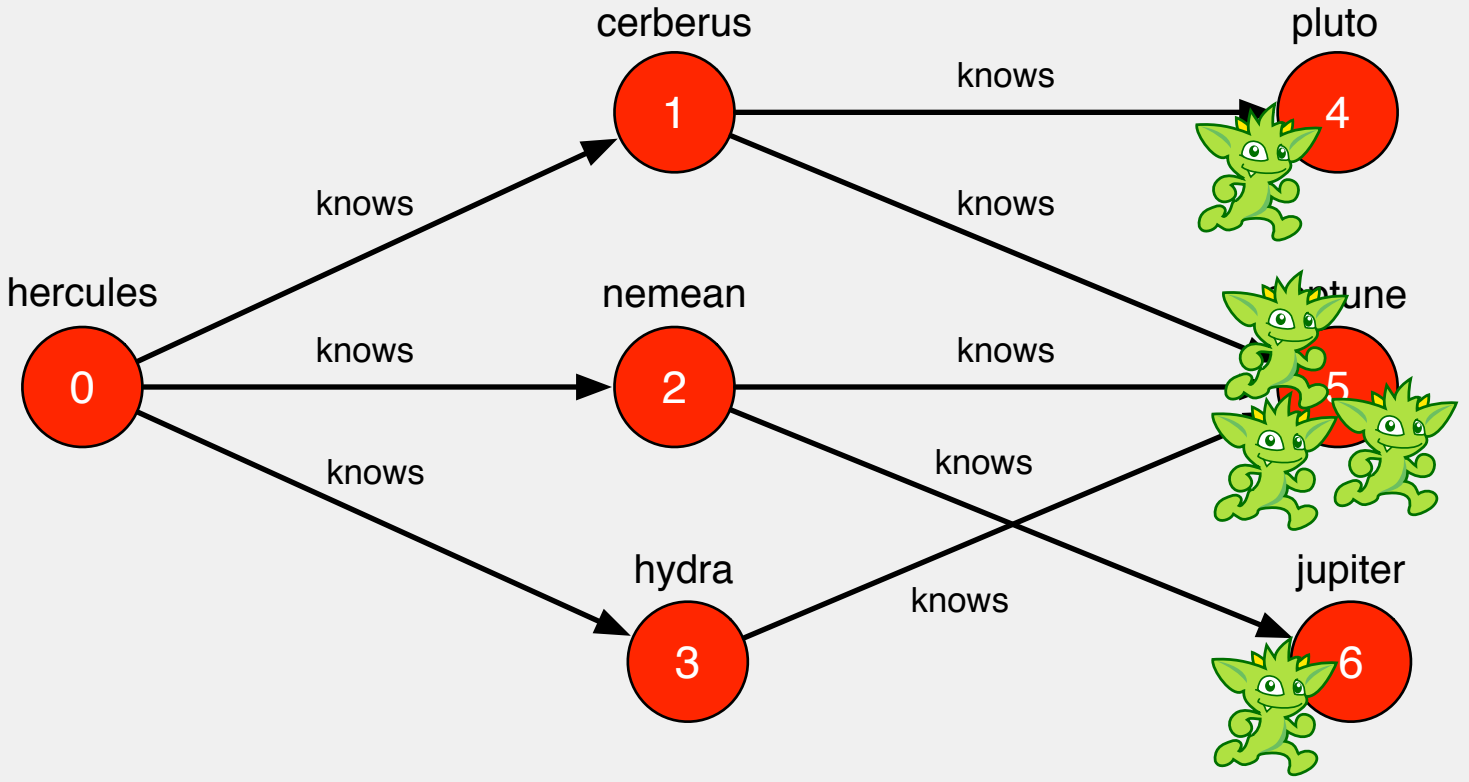

```
gremlin> hercules
==>v[0]
```




```
gremlin> hercules.out('knows')
==>v[1]
==>v[2]
==>v[3]
```

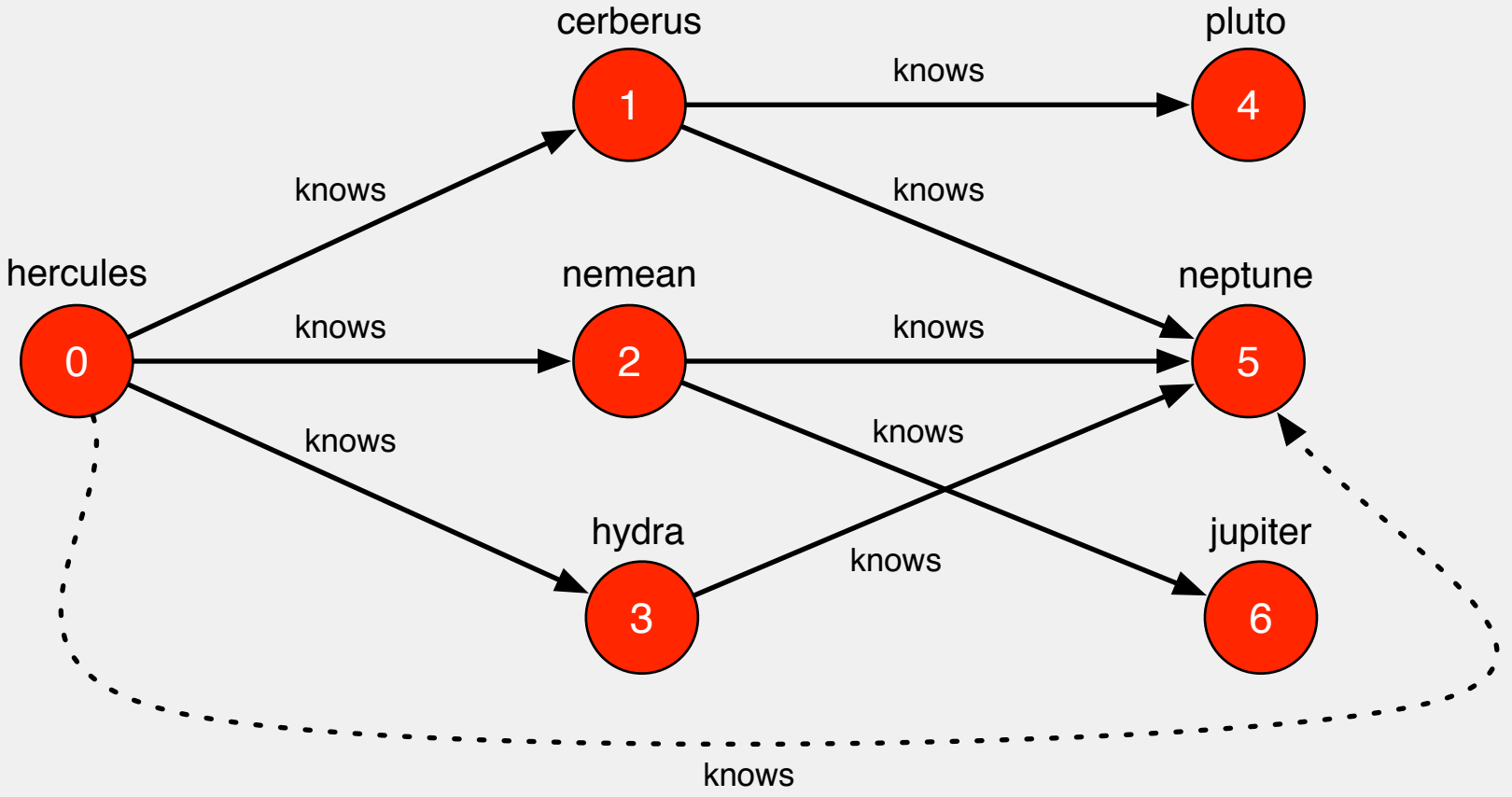



```
gremlin> hercules.out('knows').out('knows')
==>v[4]
==>v[5]
==>v[5]
==>v[6]
==>v[5]
```

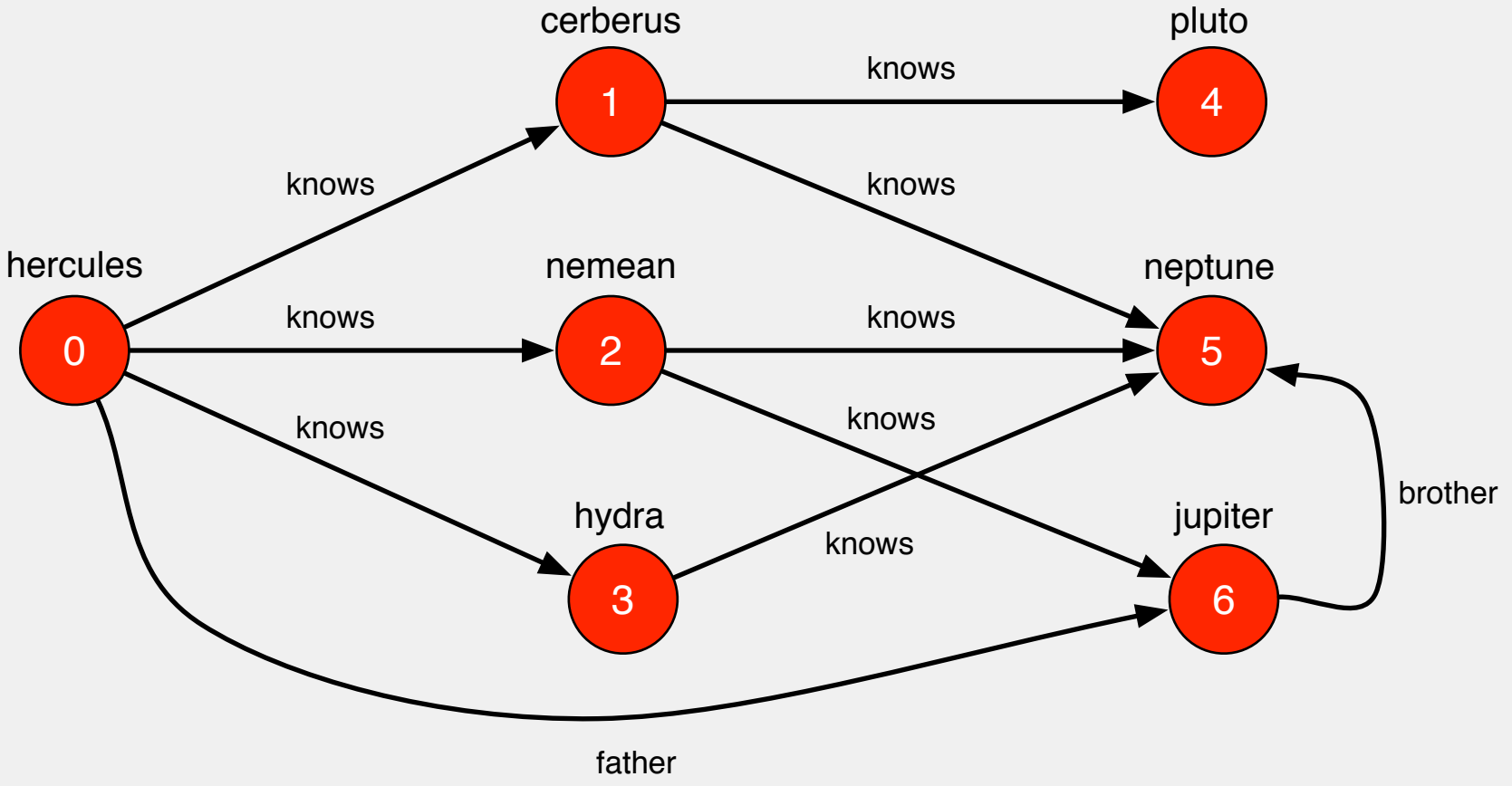



```
gremlin> hercules.out('knows').out('knows').groupCount()
==>v[4]=1
==>v[5]=3
==>v[6]=1
```

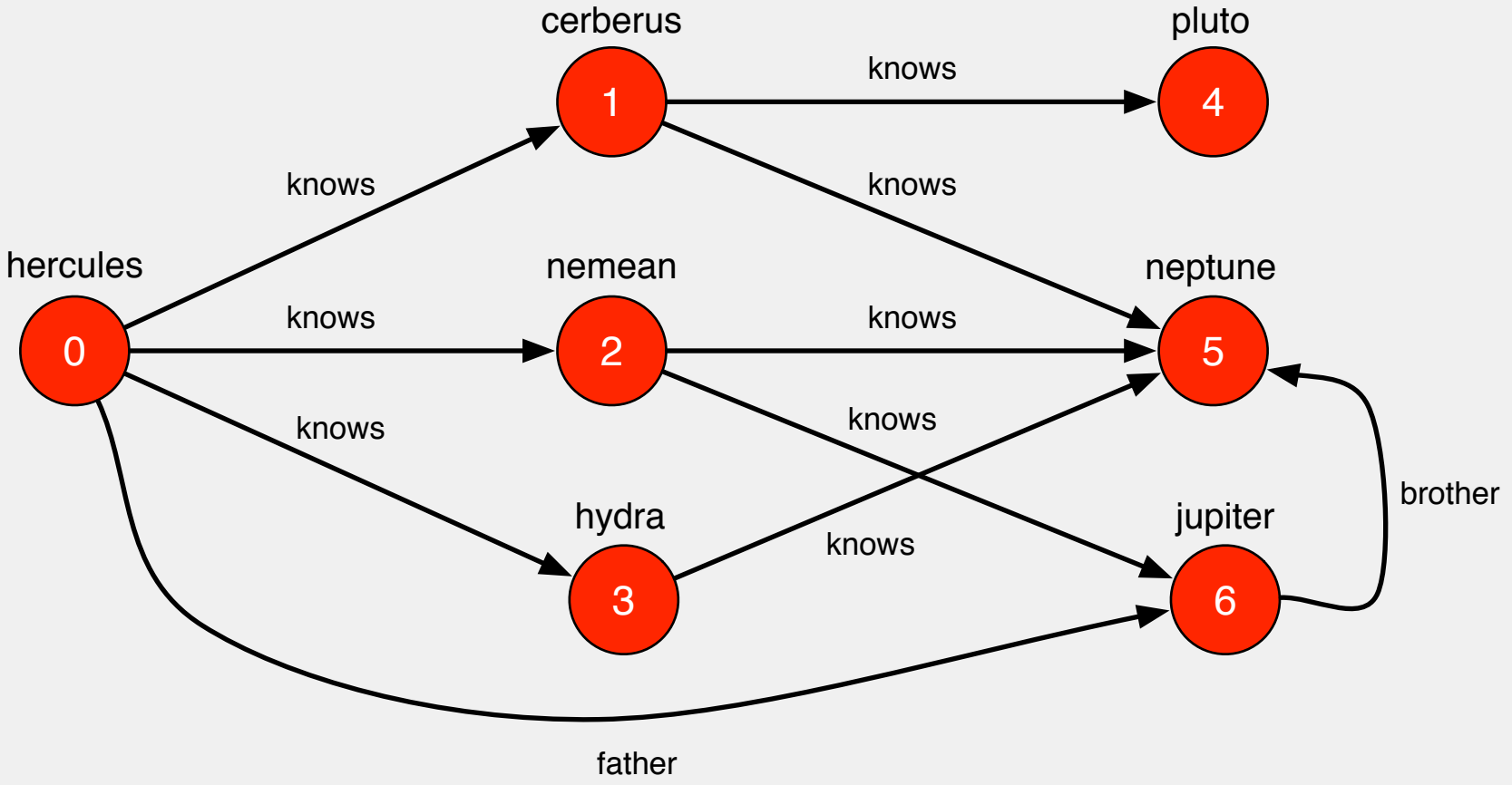
HERCULES PROBABLY KNOWS NEPTUNE

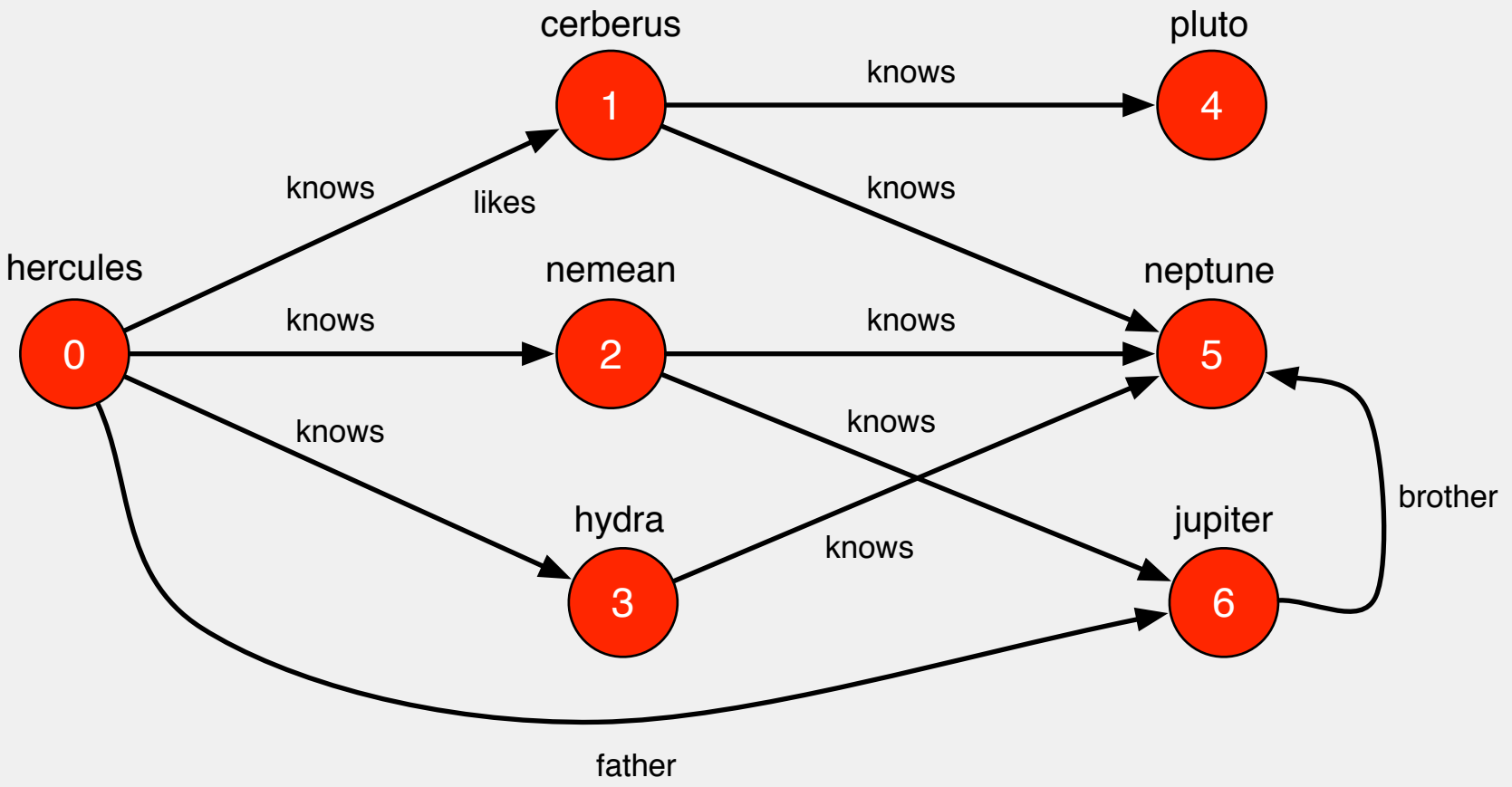


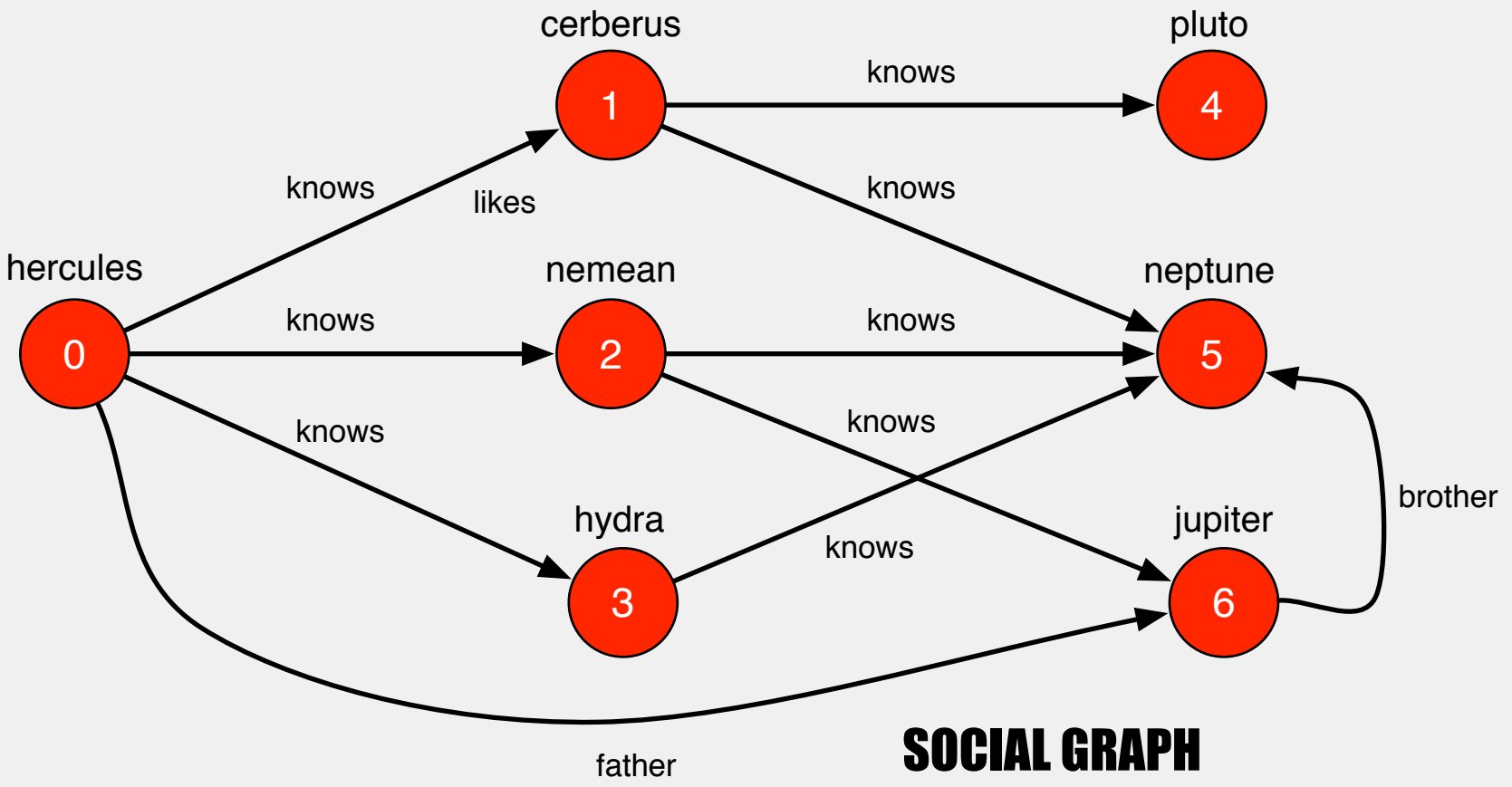
HERCULES PROBABLY KNOWS NEPTUNE

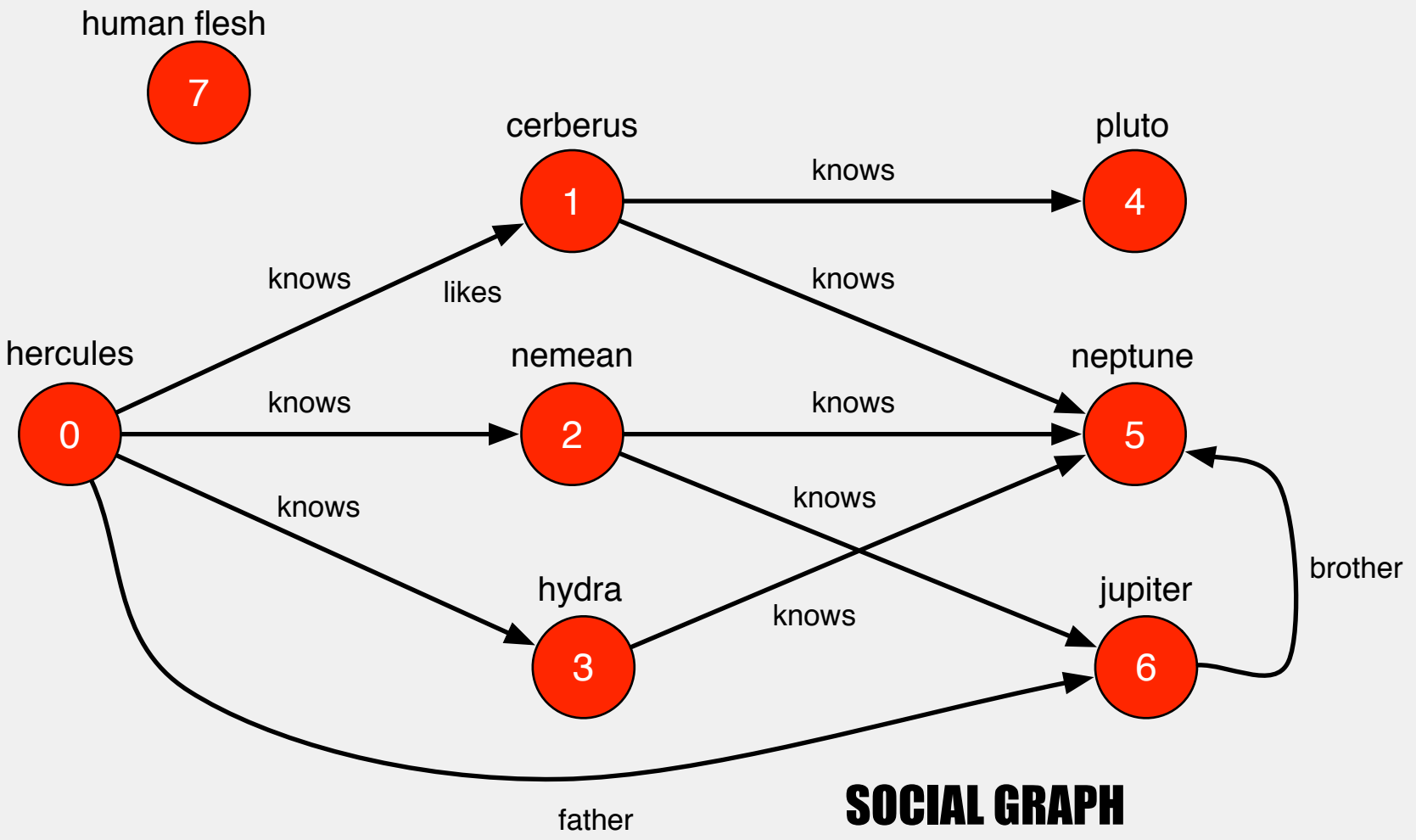


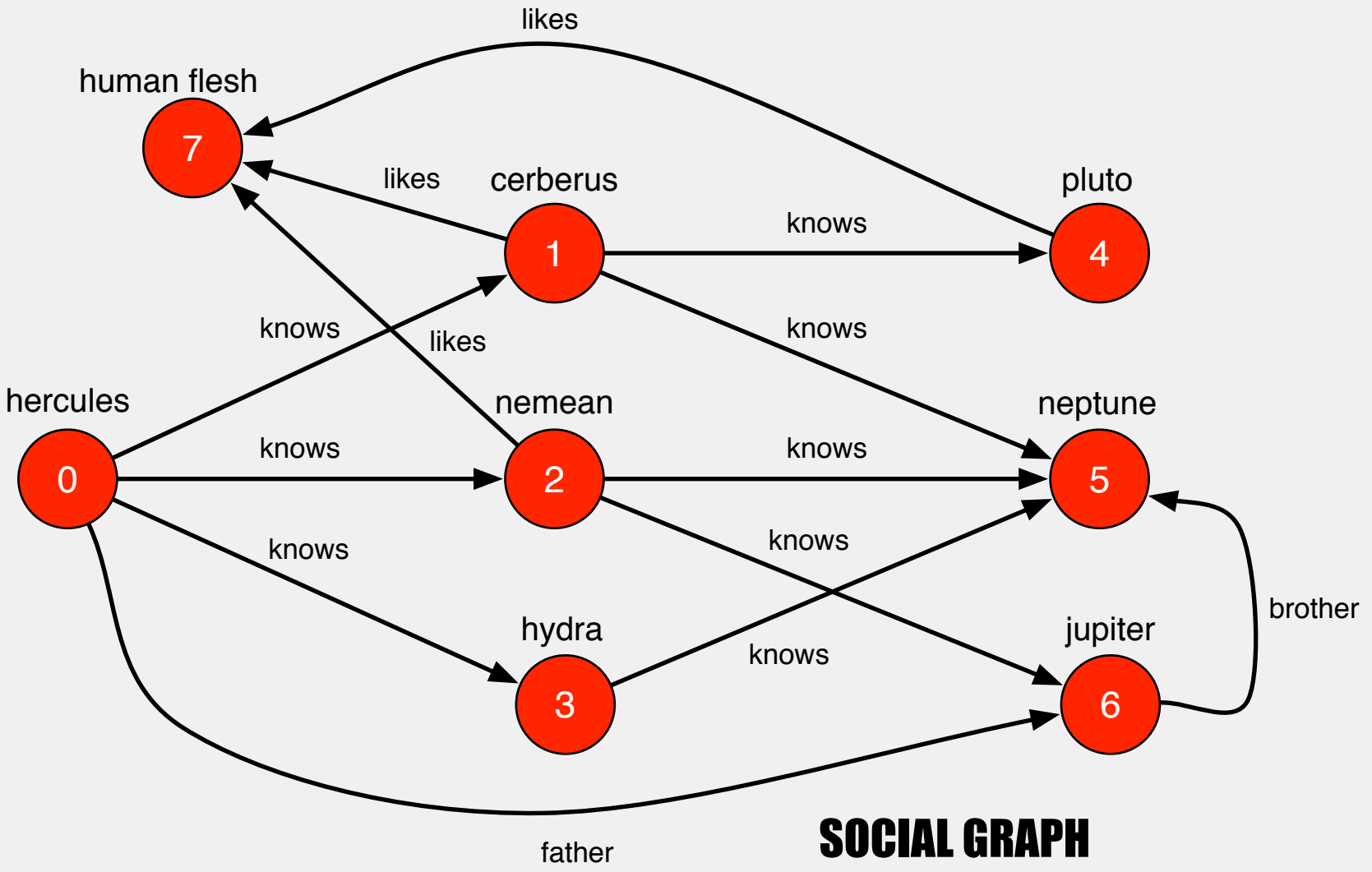
...PROBABLY MORE SO WHEN OTHER TYPES OF EDGES ARE ANALYZED

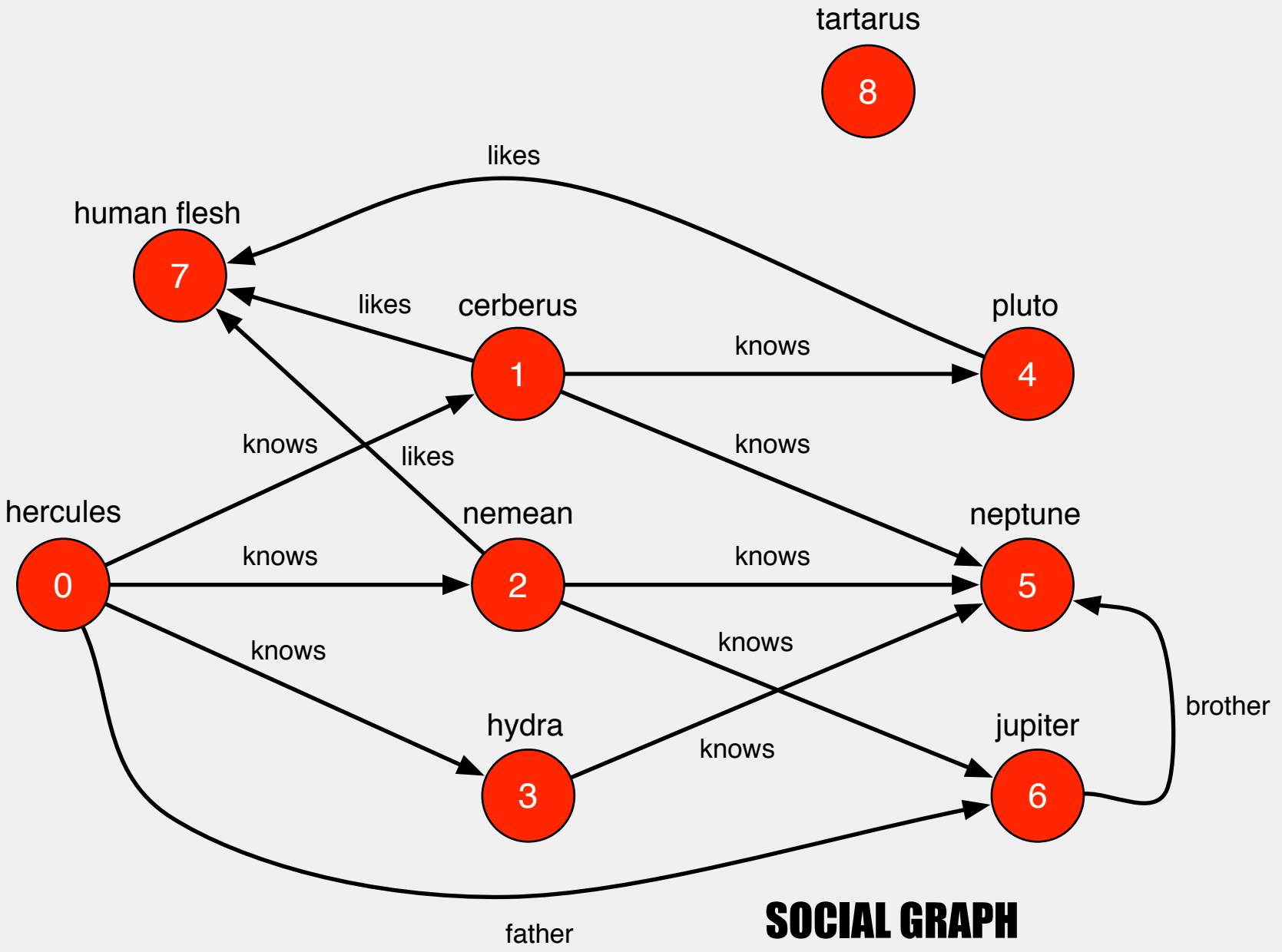

cerberus
pluto
1
knows
4
knows
knows
hercules
nemean
neptune
0
knows
2
knows
5
knows
knows
brother
hydra
jupiter
knows
3
6
father

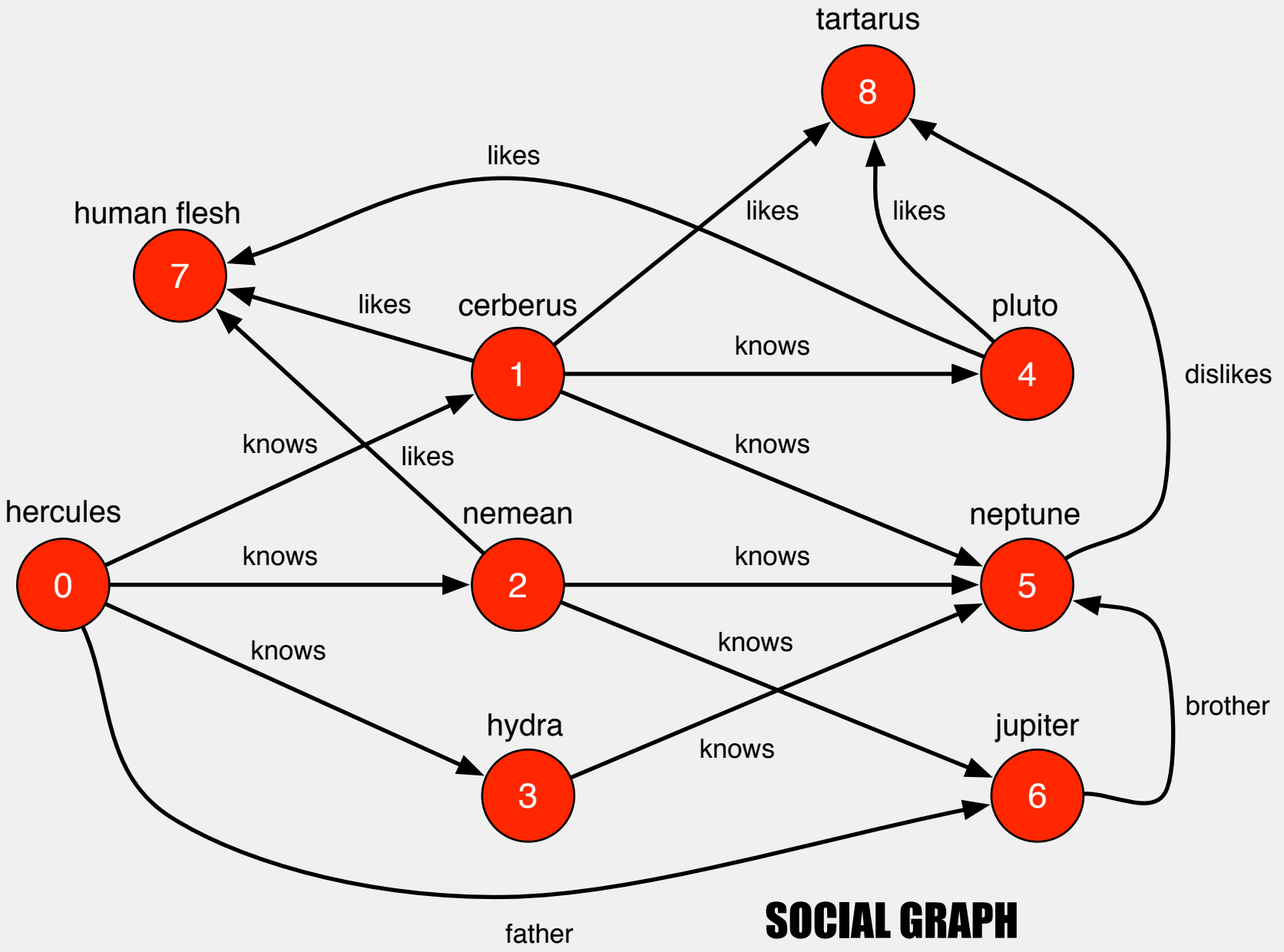

cerberus
pluto
knows
1
4
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father

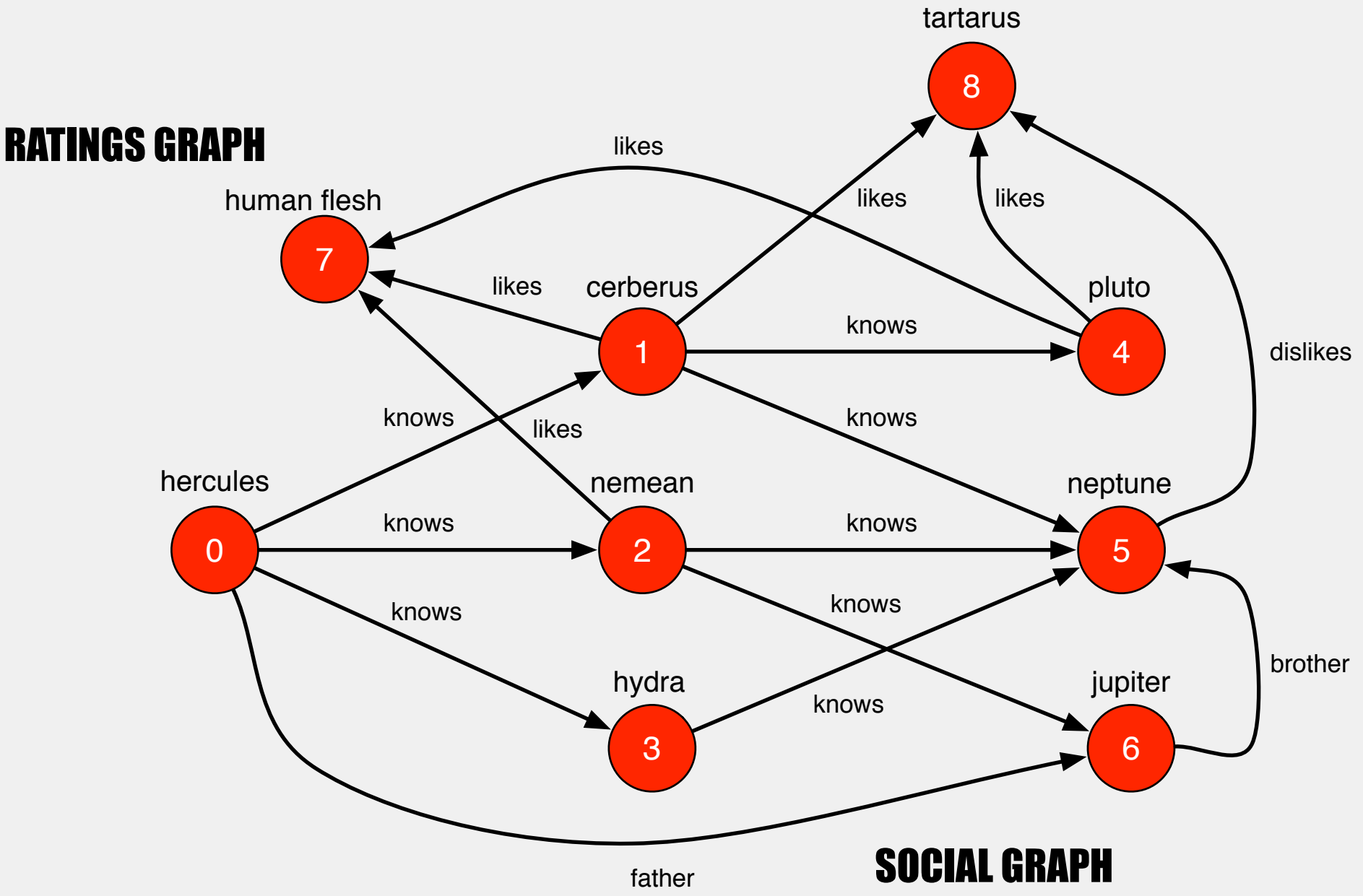

cerberus
pluto
knows
1
4
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father
SOCIAL GRAPH


human flesh
7
cerberus
1
pluto
4
knows
knows
likes
knows
hercules
0
nemean
2
neptune
5
knows
knows
knows
knows
hydra
3
jupiter
6
knows
brother
father
SOCIAL GRAPH

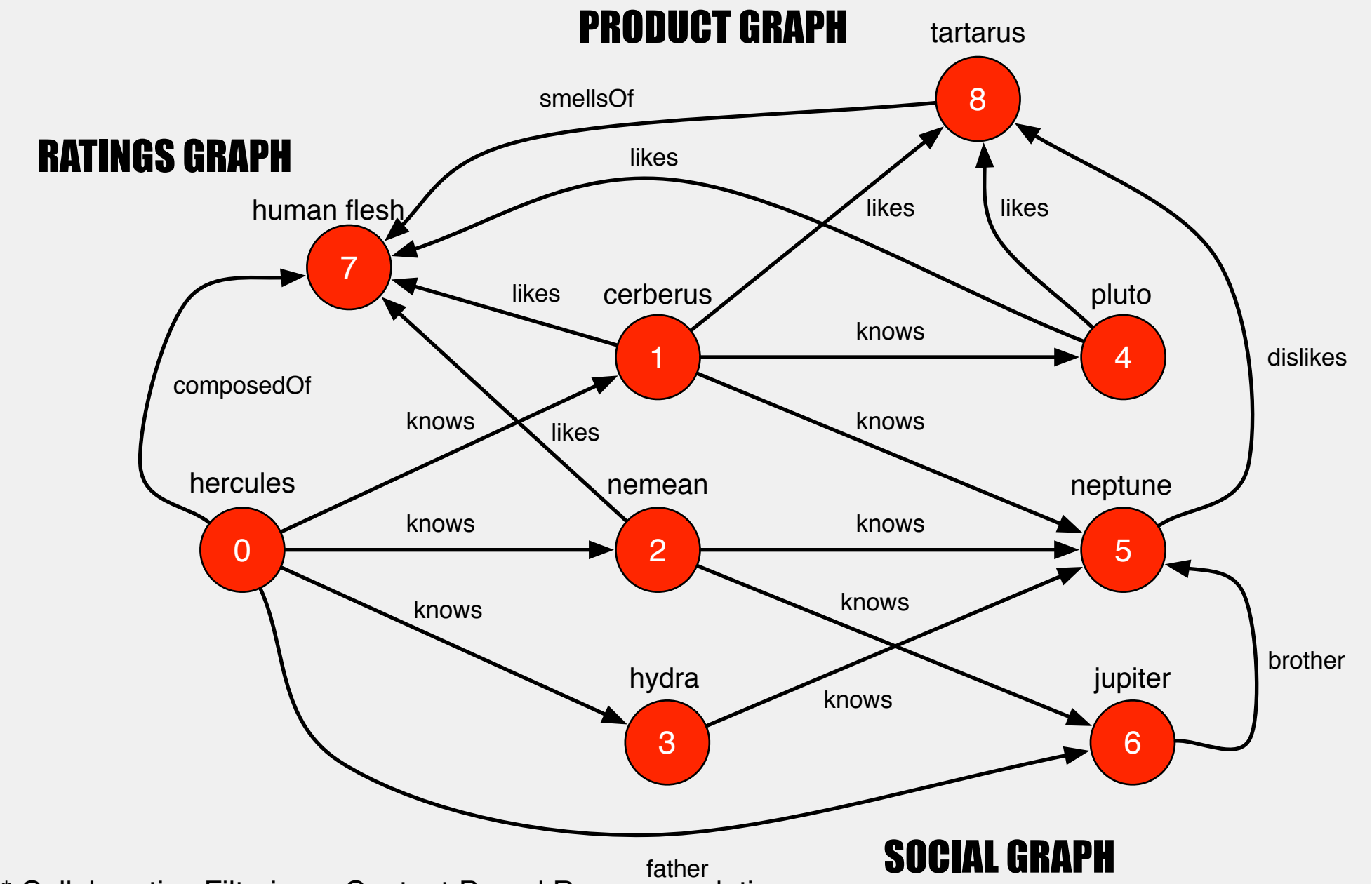

likes
human flesh
7
likes
cerberus
pluto
knows
1
4
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father
SOCIAL GRAPH


tartarus
8
likes
human flesh
7
likes
cerberus
pluto
knows
1
4
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father
SOCIAL GRAPH


tartarus
8
likes
human flesh
7
likes
likes
likes
cerberus
pluto
knows
1
4
dislikes
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father
SOCIAL GRAPH


RATINGS GRAPH
tartarus
8
likes
human flesh
7
likes
likes
likes
cerberus
pluto
knows
1
4
dislikes
knows
likes
knows
hercules
nemean
neptune
knows
knows
0
2
5
knows
knows
brother
hydra
jupiter
knows
3
6
father
SOCIAL GRAPH

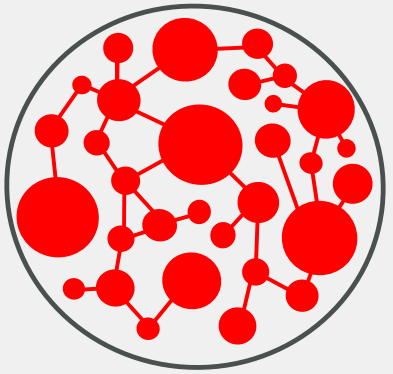
NEMEAN MIGHT LIKE TARTARUS



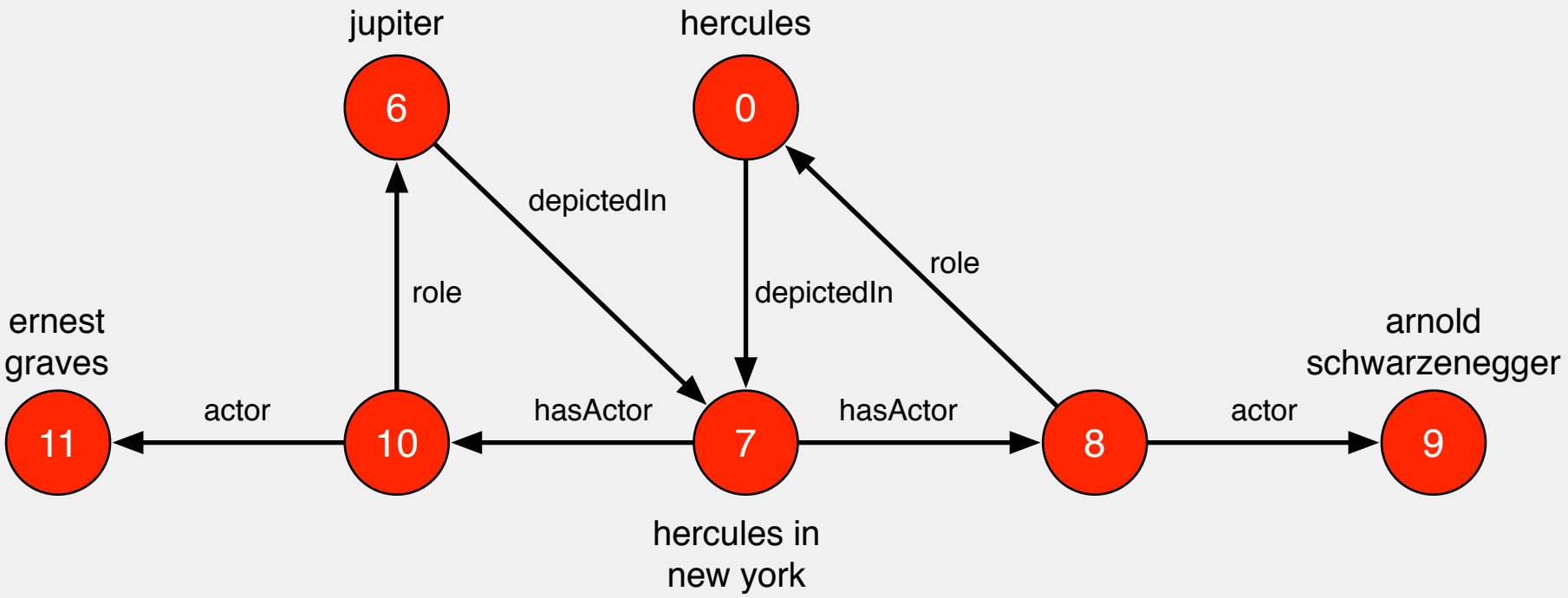
* Collaborative Filtering + Content-Based Recommendation

PATH FINDING

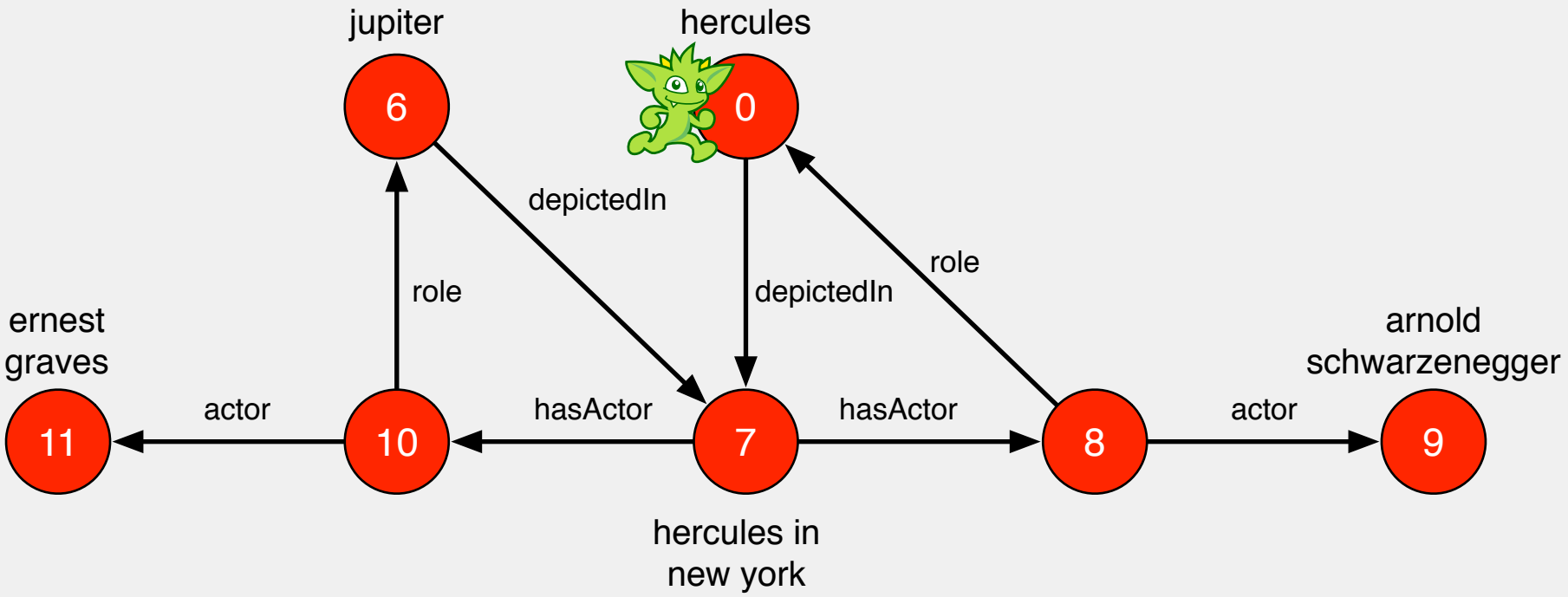
- How is this person related to this film? **MOVIE GRAPH**
- Which authors of this book also wrote a New York Times bestseller? **BOOK GRAPH**
- Which movies are based on a book by a New York Times bestseller? **MOVIE+BOOK GRAPH**



WHO PLAYED HERCULES IN WHAT MOVIE?

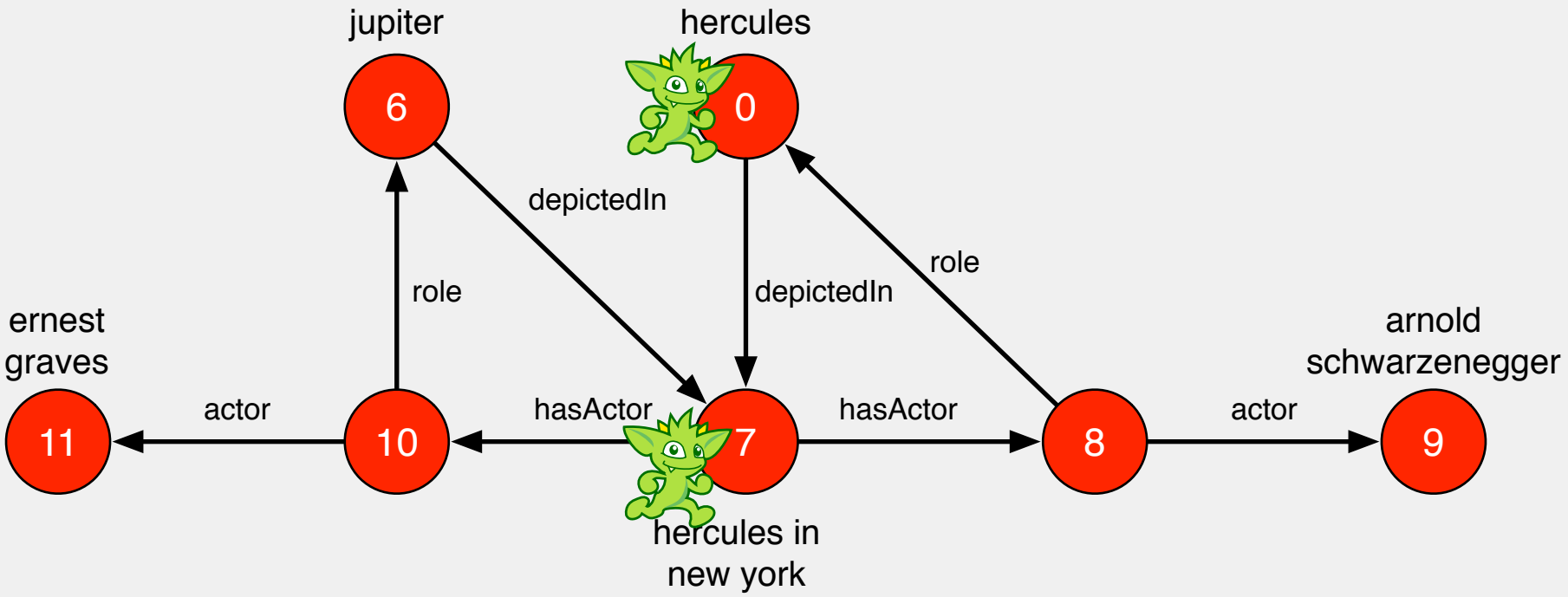


WHO PLAYED HERCULES IN WHAT MOVIE?



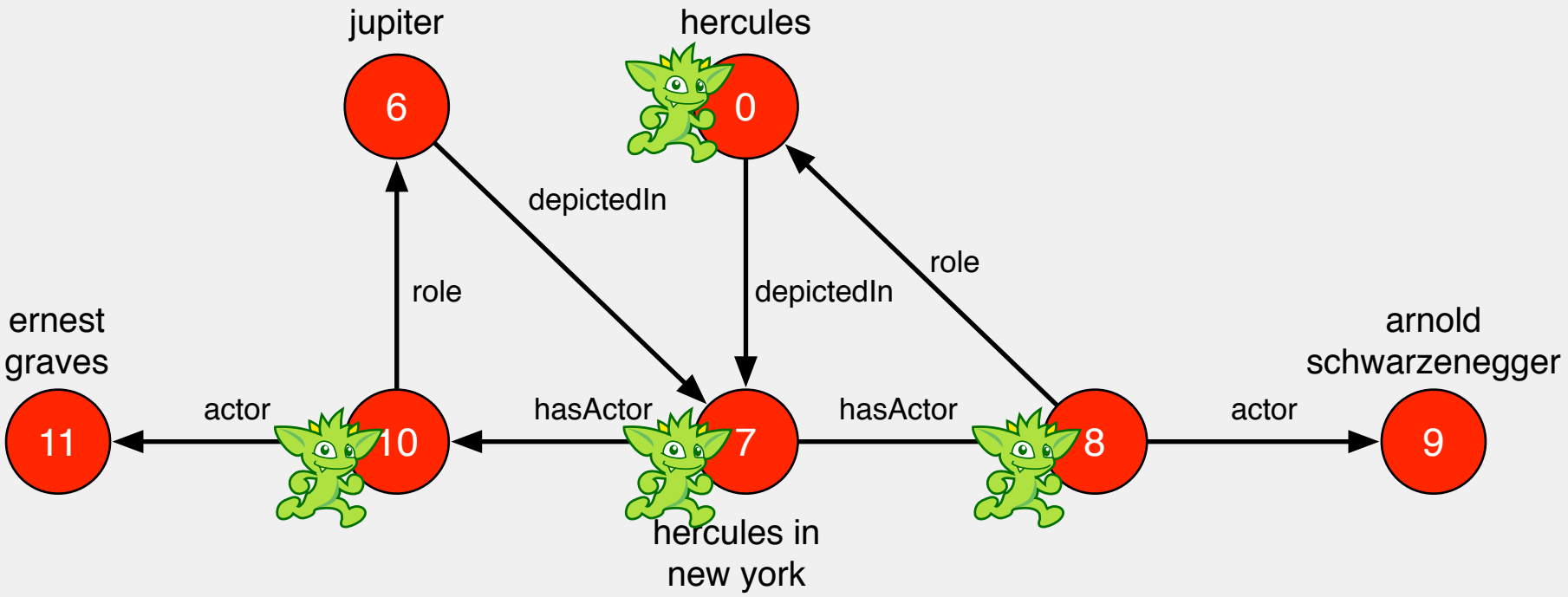

```
gremlin> hercules.match('hercules',
```

WHO PLAYED HERCULES IN WHAT MOVIE?



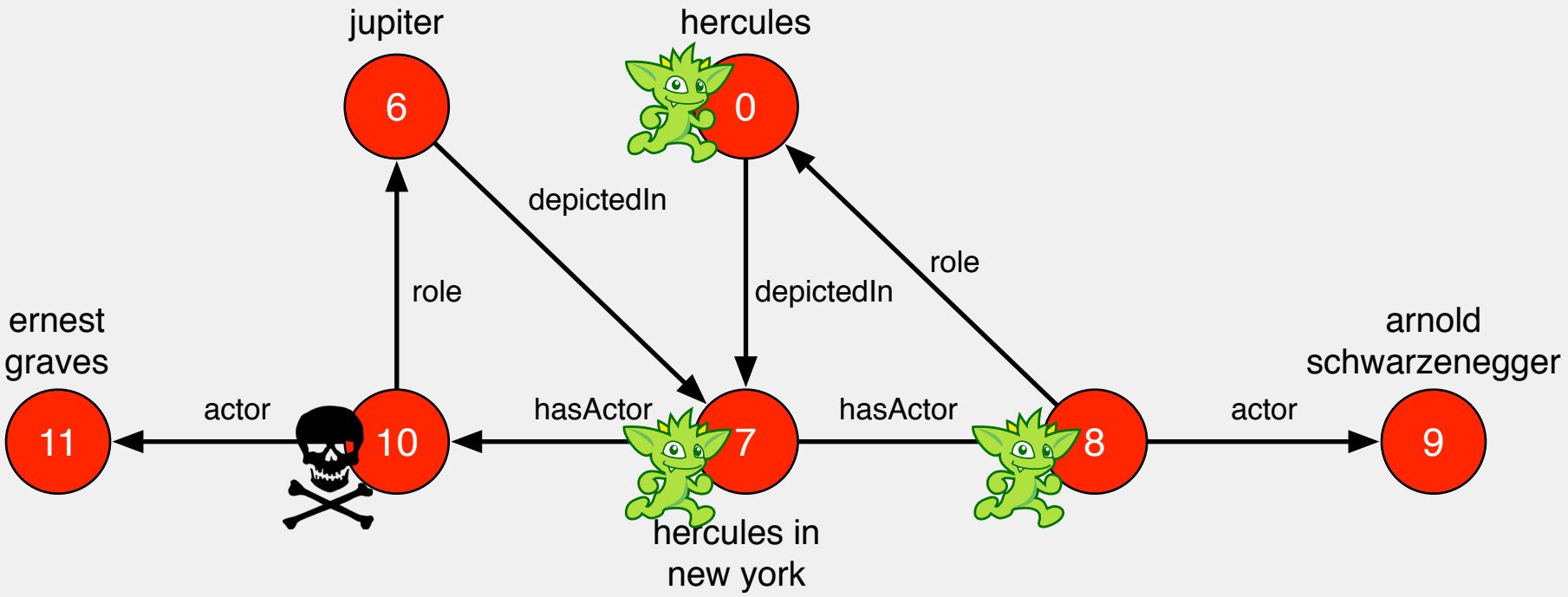

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie'),
```

WHO PLAYED HERCULES IN WHAT MOVIE?



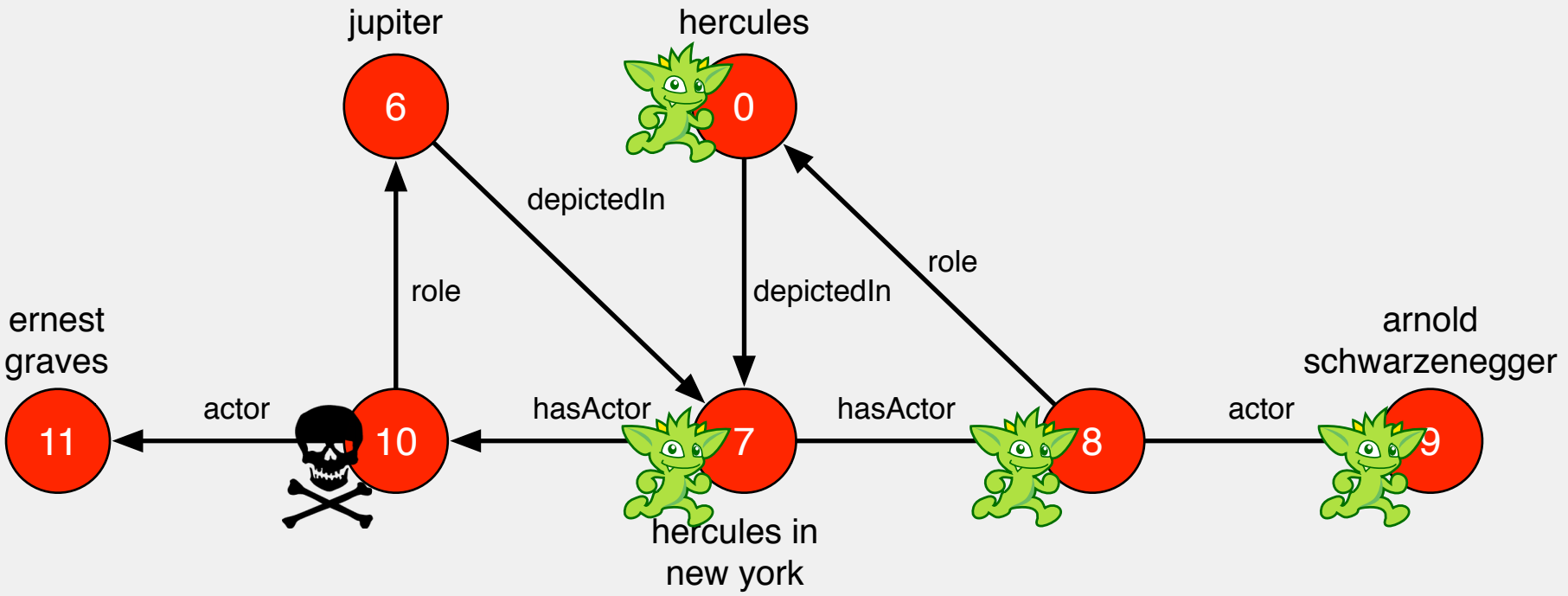

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie'),
  g.of().as('movie').out('hasActor').as('actor'),
```

WHO PLAYED HERCULES IN WHAT MOVIE?



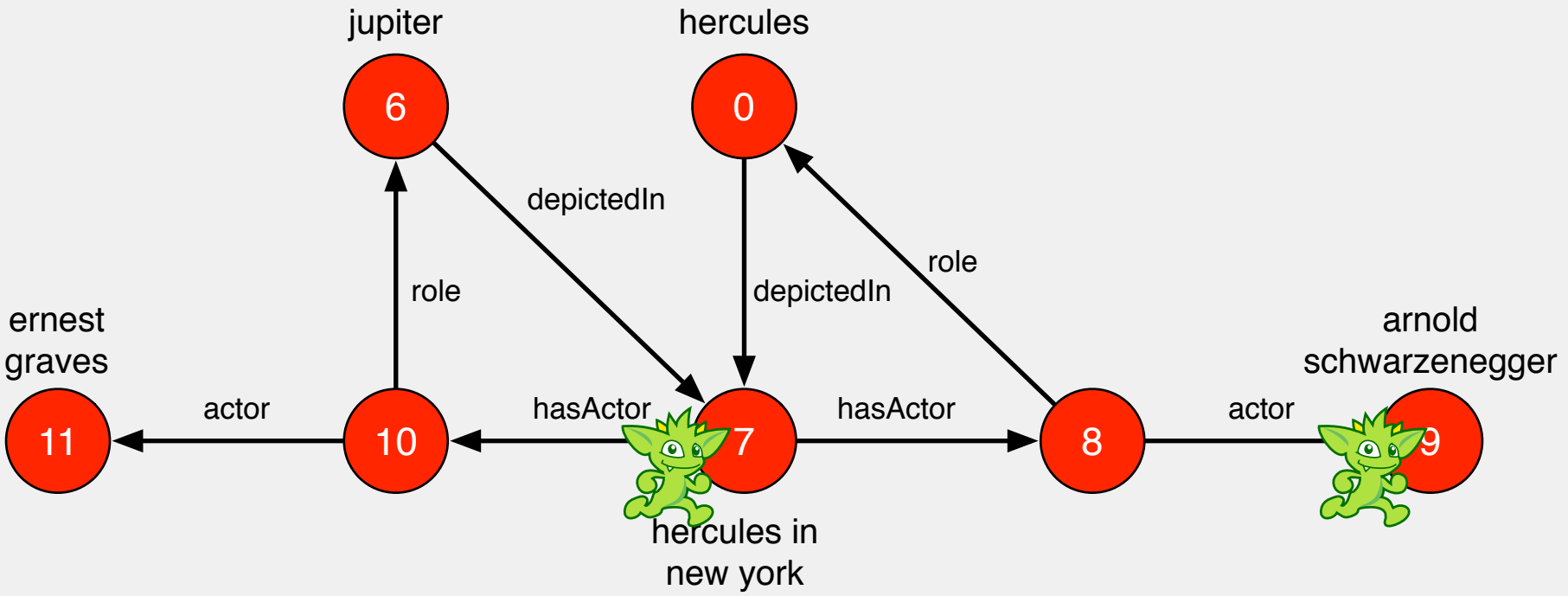

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie'),
  g.of().as('movie').out('hasActor').as('actor'),
  g.of().as('actor').out('role').as('hercules'),
```

WHO PLAYED HERCULES IN WHAT MOVIE?



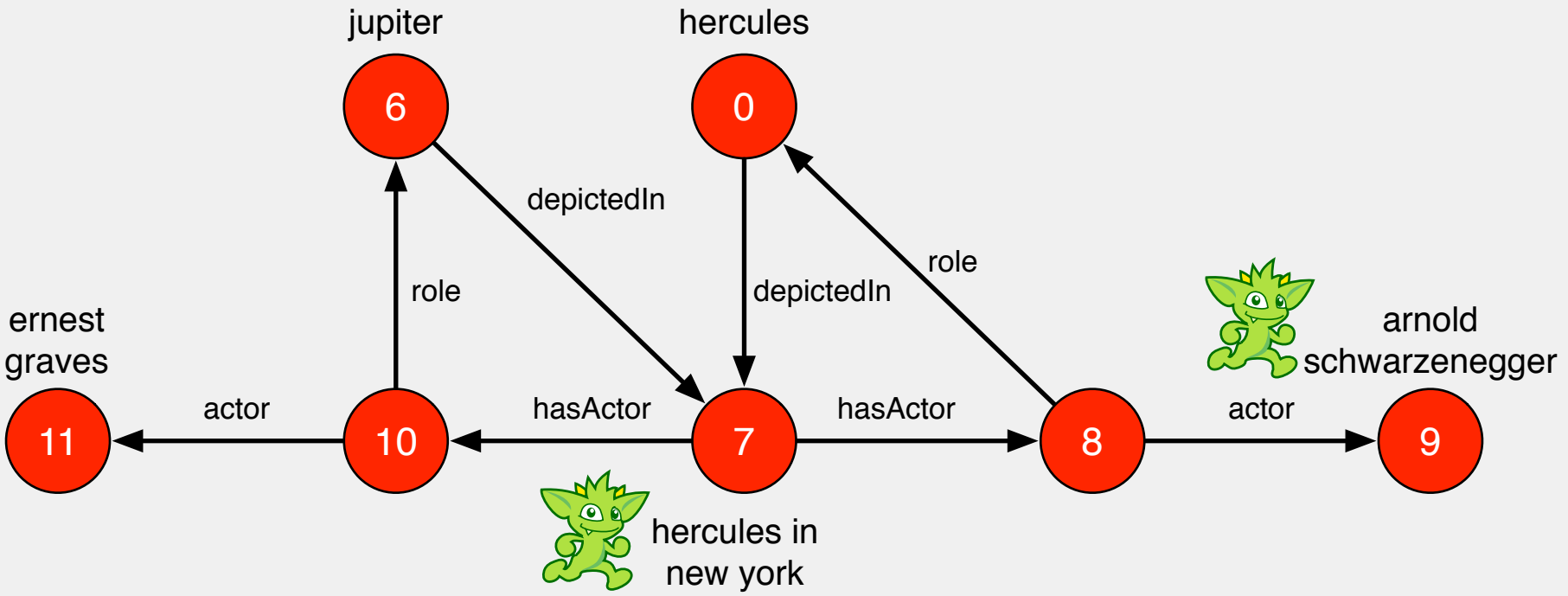

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie'),
  g.of().as('movie').out('hasActor').as('actor'),
  g.of().as('actor').out('role').as('hercules'),
  g.of().as('actor').out('actor').as('star'))
```

WHO PLAYED HERCULES IN WHAT MOVIE?

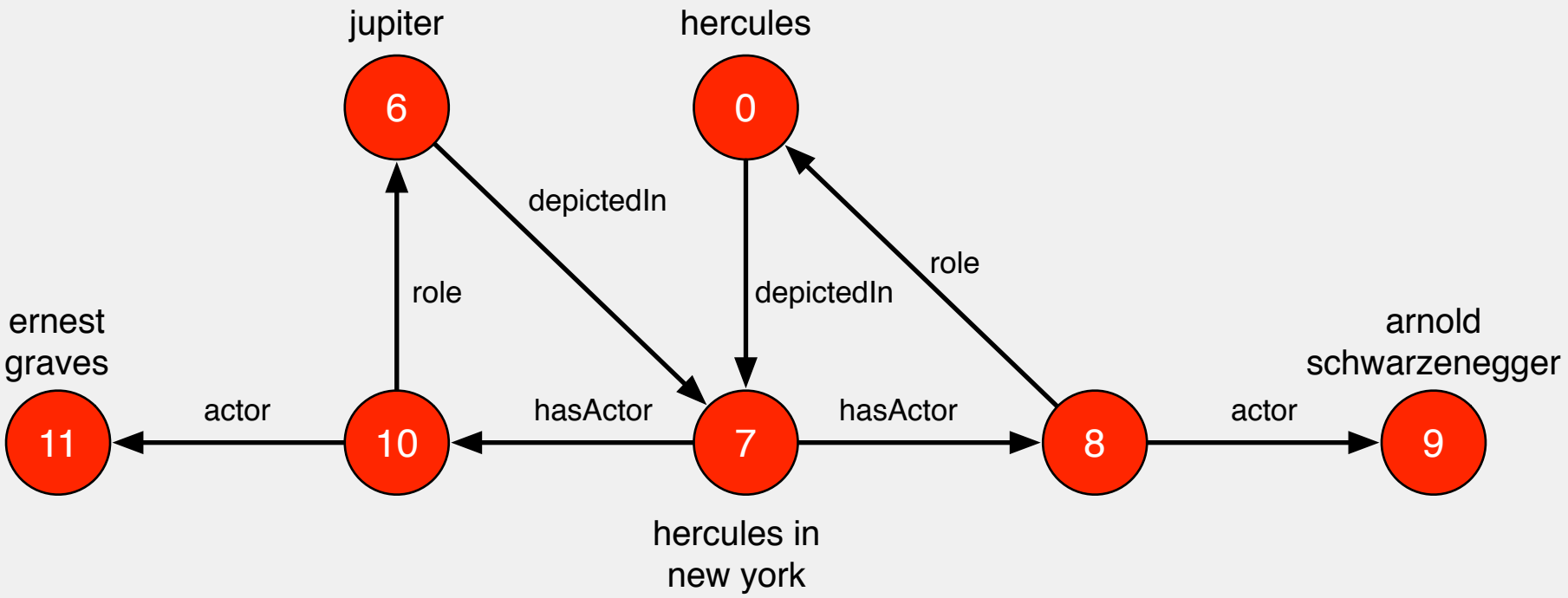


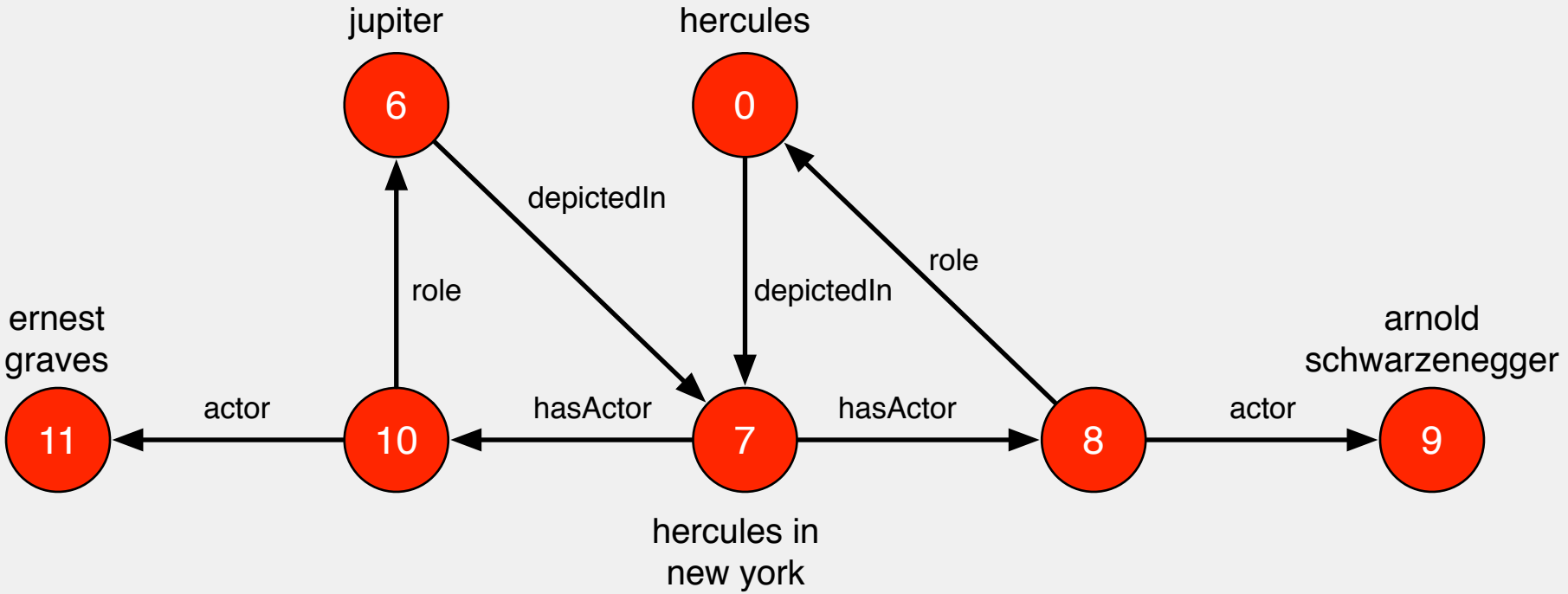

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie'),
  g.of().as('movie').out('hasActor').as('actor'),
  g.of().as('actor').out('role').as('hercules'),
  g.of().as('actor').out('actor').as('star'))
.select(['movie','star'])
==>[movie:v[7], star:v[9]]
```

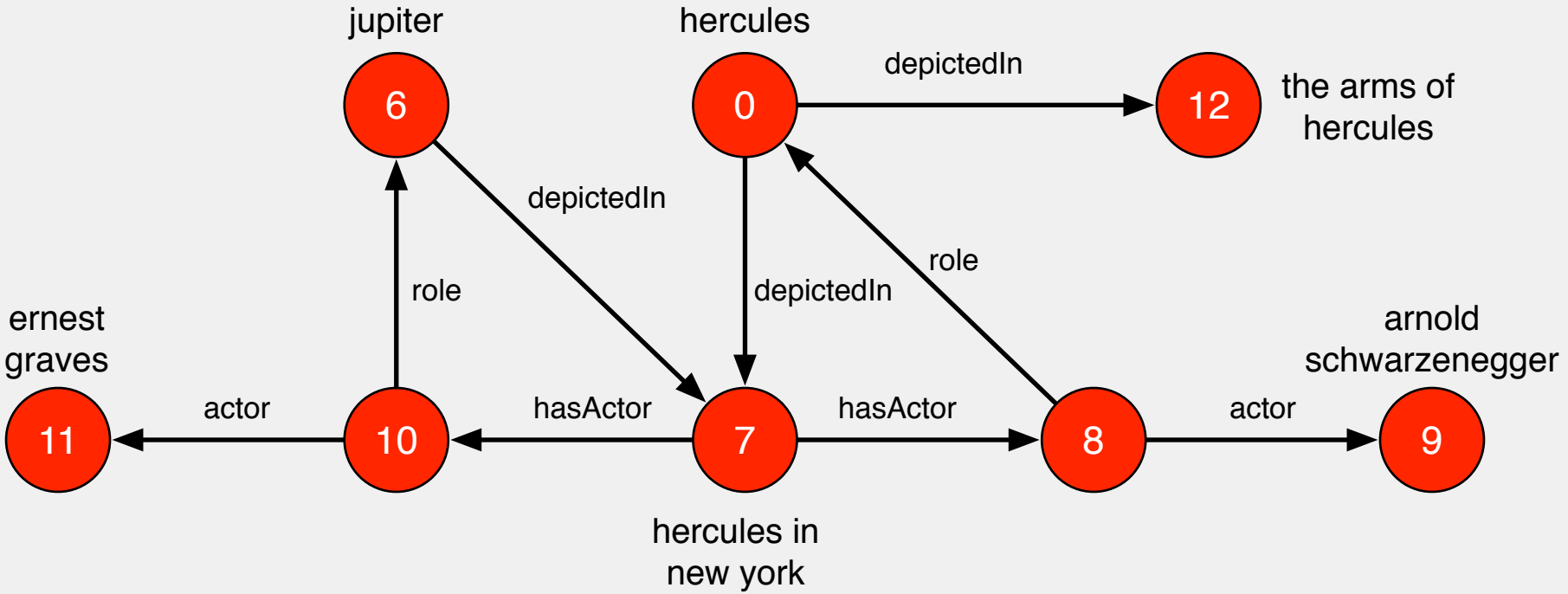
WHO PLAYED HERCULES IN WHAT MOVIE?

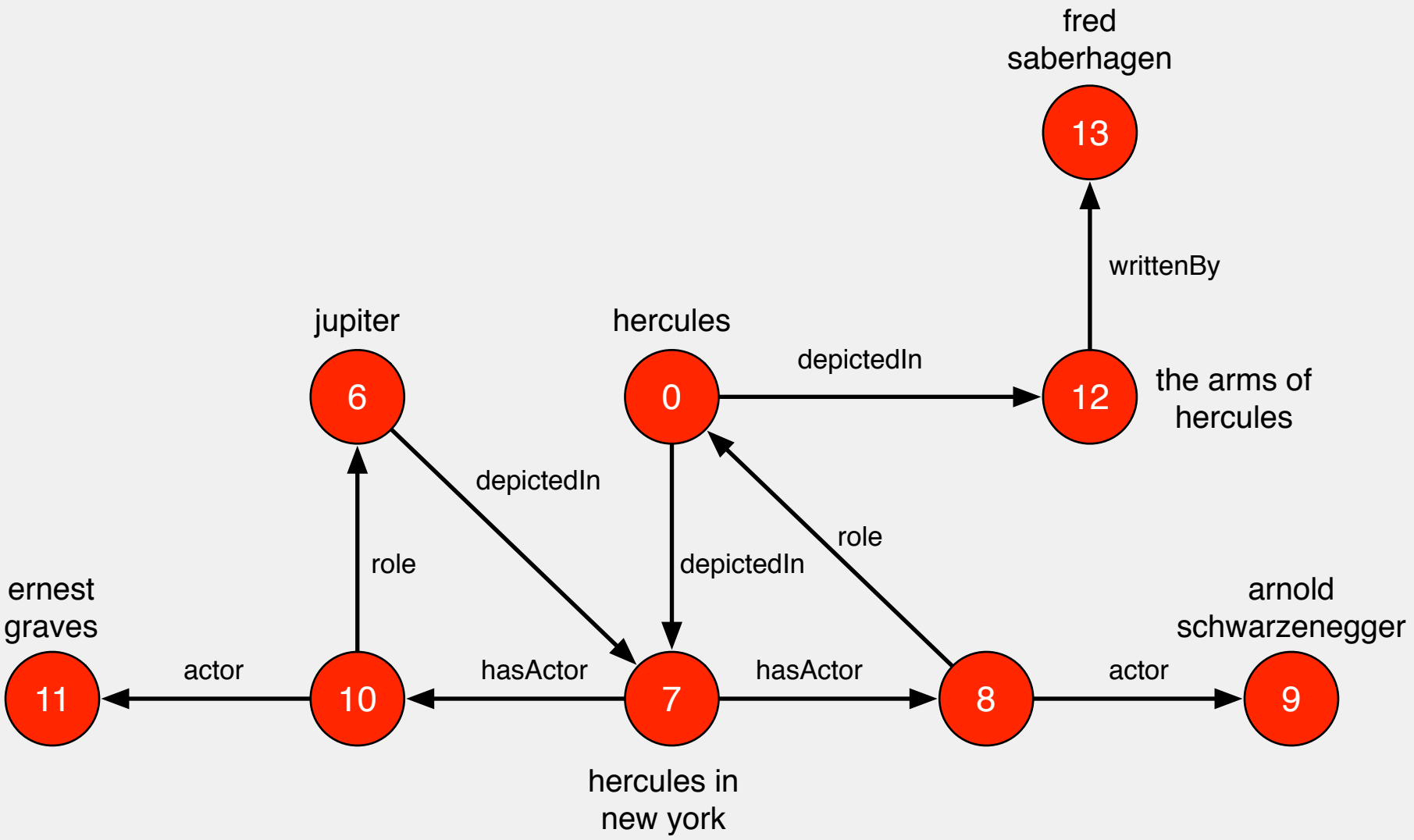


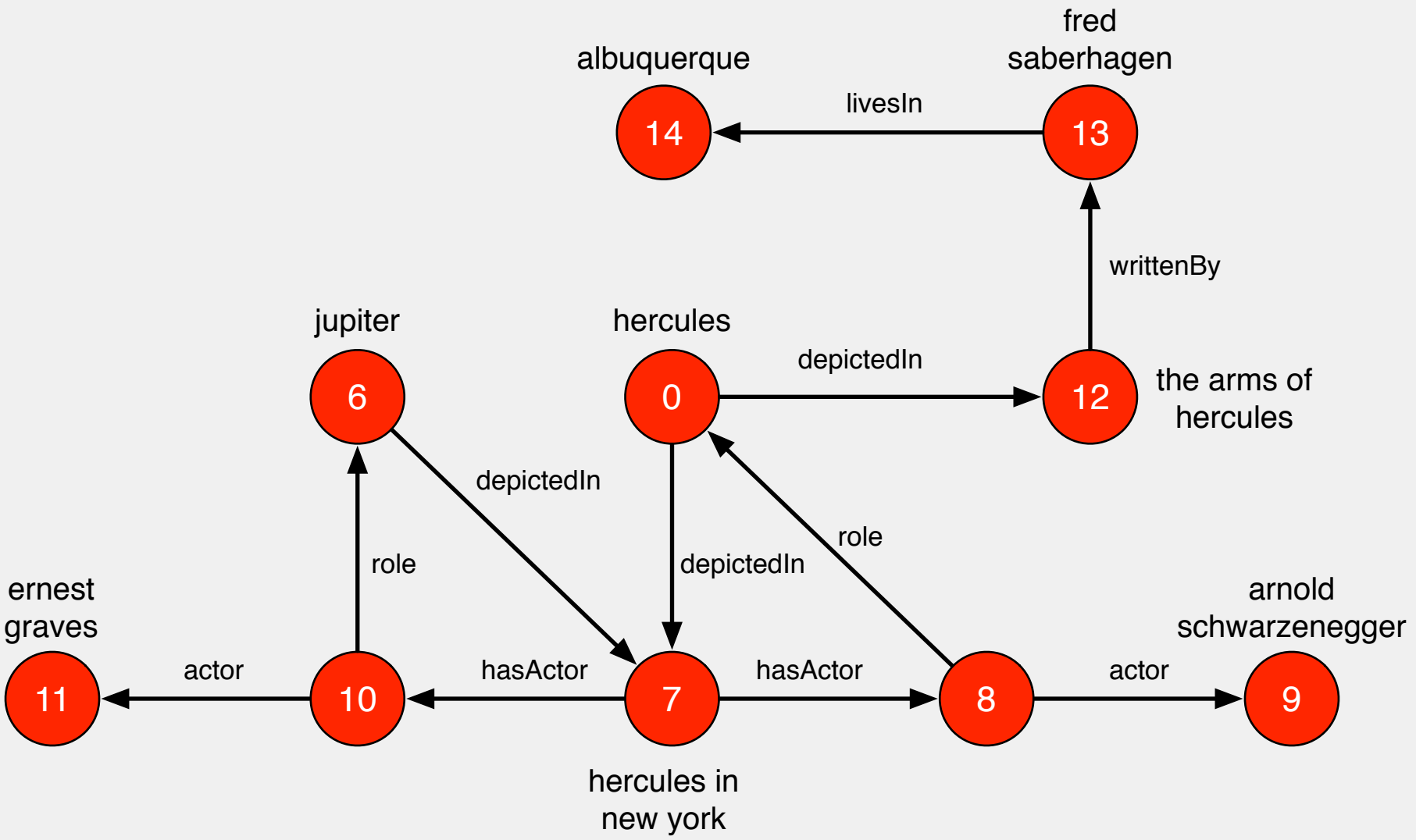

```
gremlin> hercules.match('hercules',
  g.of().as('hercules').out('depictedIn').as('movie')
  g.of().as('movie').out('hasActor').as('actor')
  g.of().as('actor').out('role').as('hercules')
  g.of().as('actor').out('actor').as('star'))
.select(['movie','star']){it.name}
==>[movie:hercules in new york, star:arnold schwarzenegger]
```

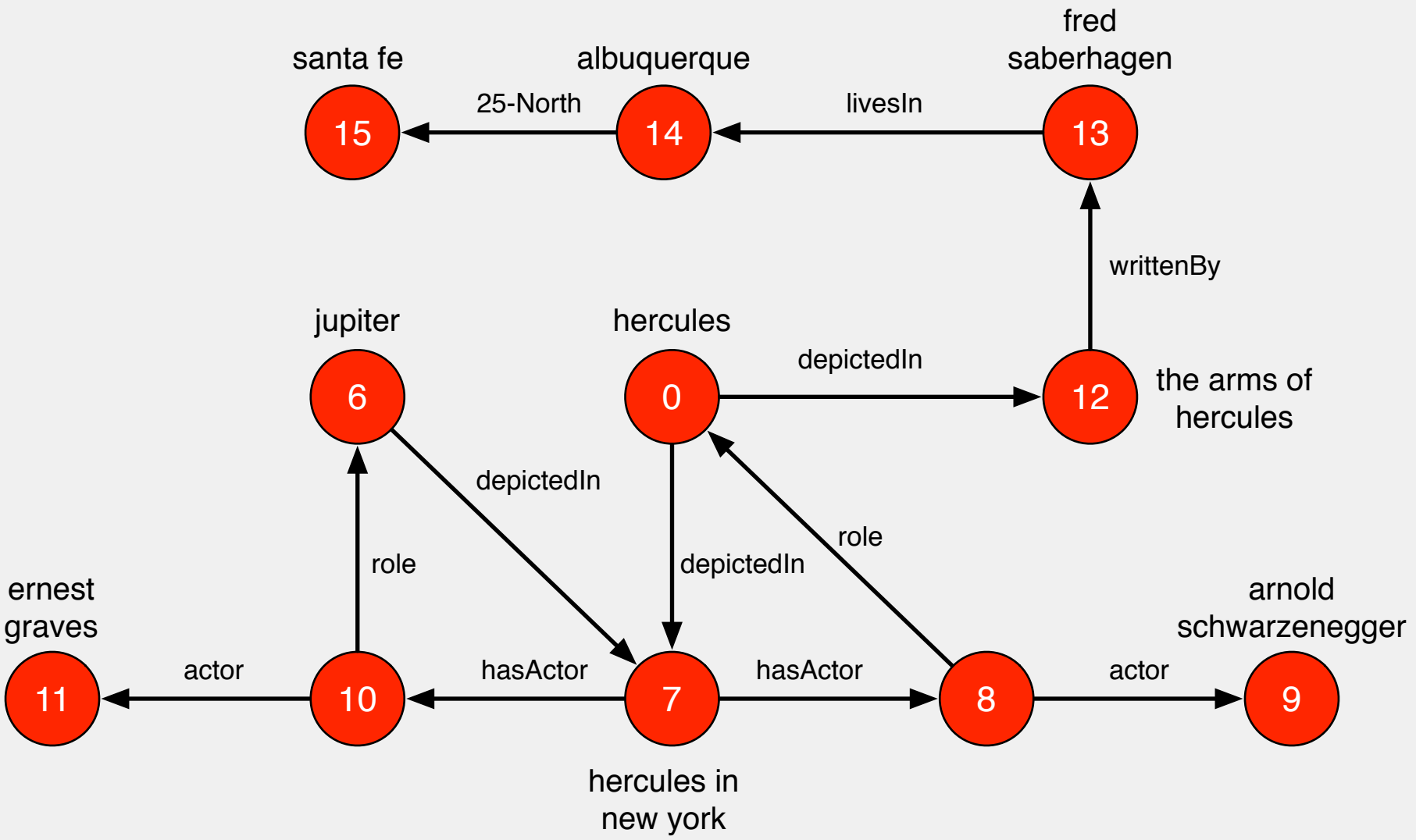

jupiter
hercules
6
0
depictedIn
role
depictedIn
role
ernest
graves
arnold
schwarzenegger
actor
hasActor
hasActor
actor
11
10
7
8
9
hercules in
new york

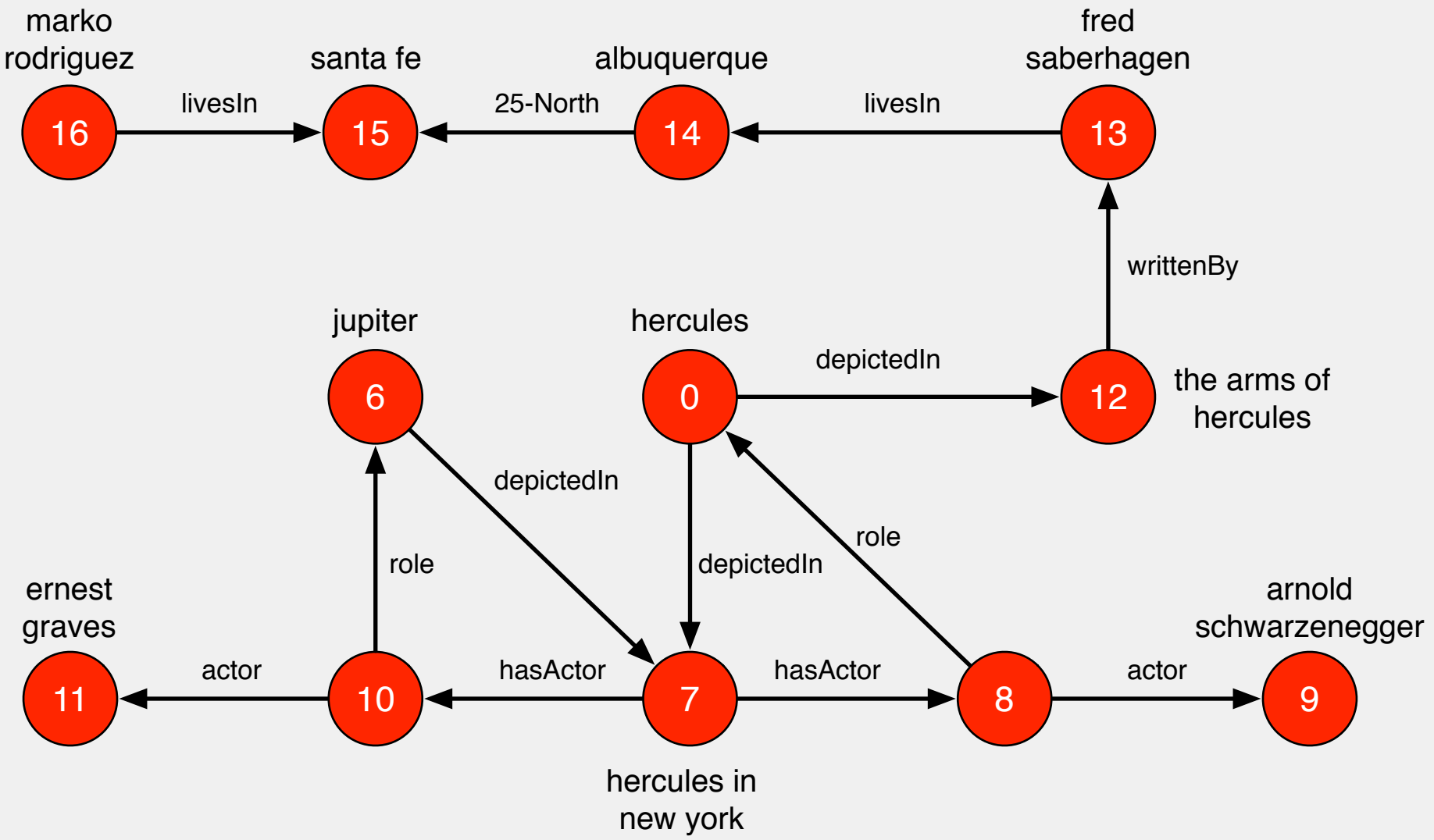

jupiter
hercules
6
0
depictedIn
role
depictedIn
role
ernest
graves
arnold
schwarzenegger
actor
hasActor
hasActor
actor
11
10
7
8
9
hercules in
new york

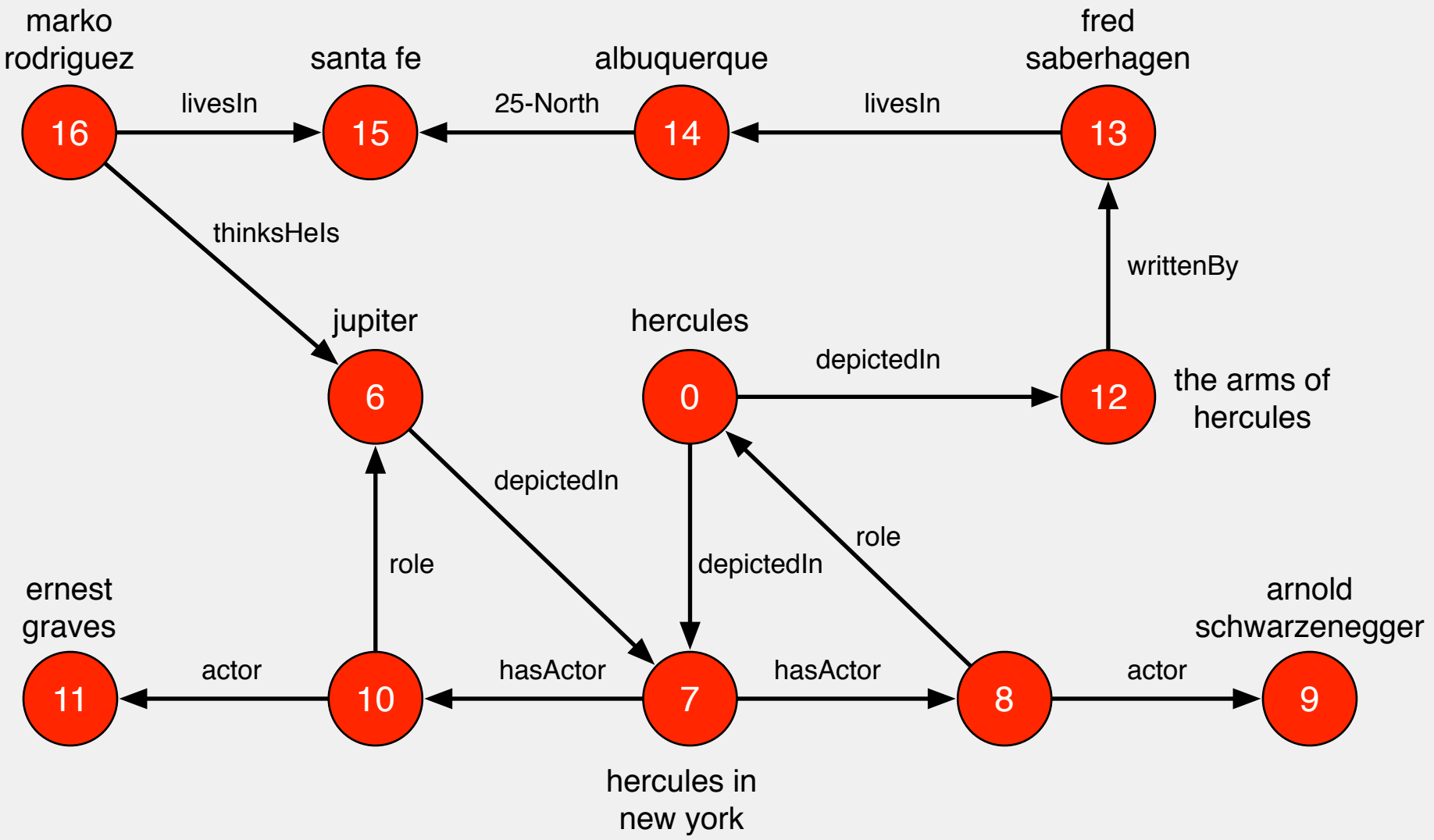

jupiter
hercules
6
0
depictedIn
12
the arms of hercules
depictedIn
role
depictedIn
role
ernest graves
arnold schwarzenegger
11
actor
10
hasActor
7
hasActor
8
actor
9
hercules in new york

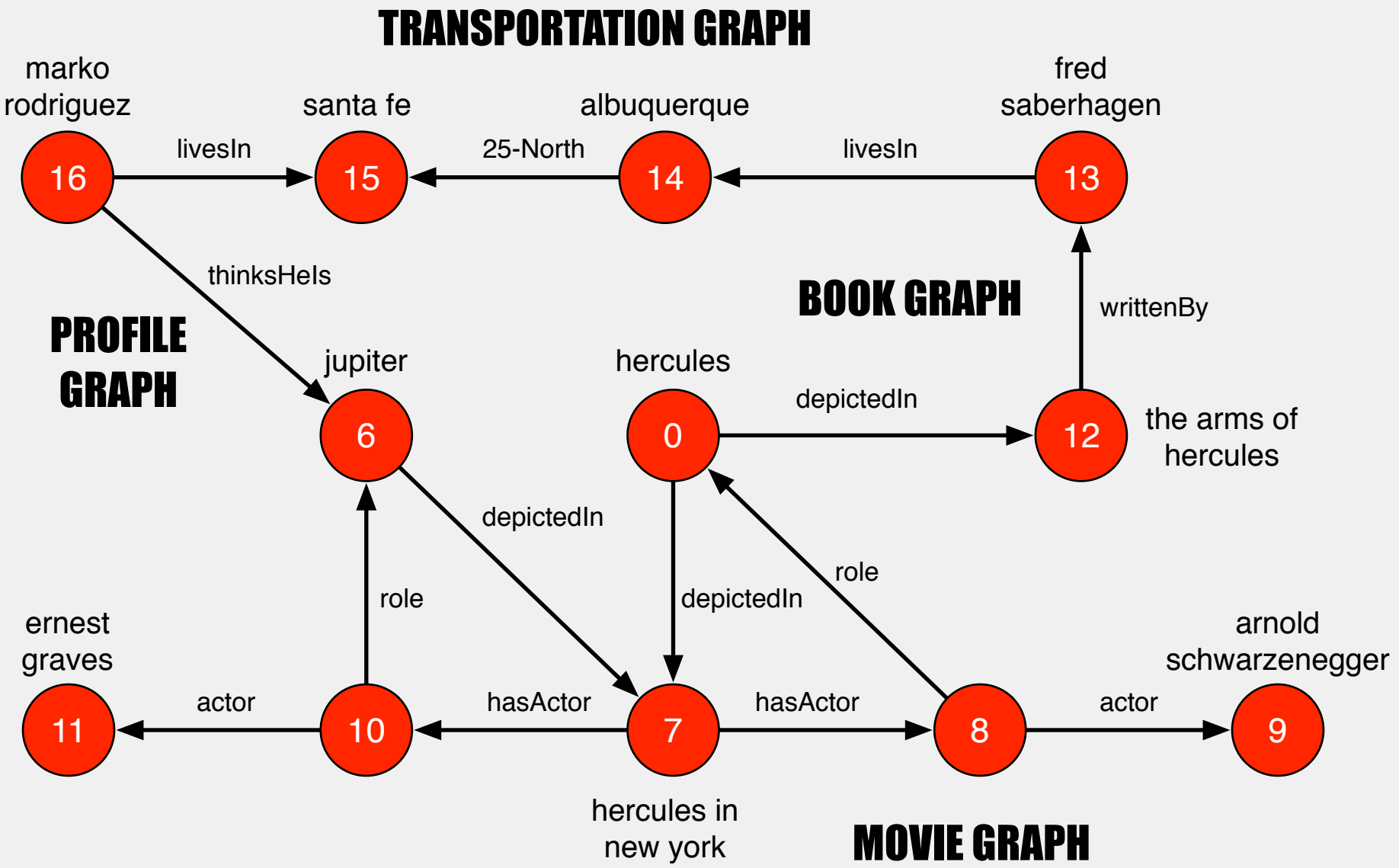

fred
saberhagen
13
writtenBy
jupiter
hercules
depictedIn
6
0
12
the arms of
hercules
depictedIn
role
depictedIn
role
ernest
graves
arnold
schwarzenegger
actor
hasActor
hasActor
actor
11
10
7
8
9
hercules in
new york


albuquerque
14
livesIn
fred
saberhagen
13
writtenBy
jupiter
6
hercules
0
depictedIn
12
the arms of
hercules
depictedIn
role
depictedIn
role
ernest
graves
11
actor
10
hasActor
7
hasActor
8
actor
arnold
schwarzenegger
9
hercules in
new york


santa fe
albuquerque
fred saberhagen
15
25-North
14
livesIn
13
writtenBy
jupiter
hercules
6
depictedIn
0
12
the arms of hercules
depictedIn
role
depictedIn
role
ernest graves
arnold schwarzenegger
11
actor
10
hasActor
7
hasActor
8
actor
9
hercules in new york


marko rodriguez
16
livesIn
santa fe
15
25-North
albuquerque
14
livesIn
fred saberhagen
13
writtenBy
jupiter
6
hercules
0
depictedIn
12
the arms of hercules
depictedIn
role
depictedIn
role
ernest graves
11
actor
10
hasActor
7
hasActor
8
actor
arnold schwarzenegger
9
hercules in new york

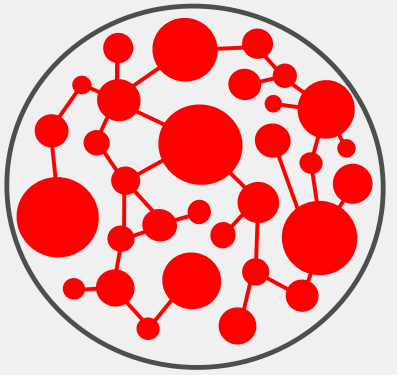

marko rodriguez
16
livesIn
santa fe
15
25-North
albuquerque
14
livesIn
fred saberhagen
13
thinksHeIs
writtenBy
jupiter
6
hercules
0
depictedIn
12
the arms of hercules
depictedIn
role
depictedIn
role
ernest graves
11
actor
10
hasActor
7
hasActor
8
actor
arnold schwarzenegger
9
hercules in new york


TRANSPORTATION GRAPH
marko rodriguez
santa fe
albuquerque
fred saberhagen
16
livesIn
15
25-North
14
livesIn
13
thinksHeIs
PROFILE GRAPH
BOOK GRAPH
writtenBy
jupiter
hercules
depictedIn
6
0
12
the arms of hercules
depictedIn
role
depictedIn
role
ernest graves
arnold schwarzenegger
actor
hasActor
hasActor
actor
11
10
7
8
9
hercules in new york
MOVIE GRAPH

SOCIAL INFLUENCE

- Who are the most influential people in java, mathematics, art, surreal art, politics, ...?
- Which region of the social graph will propagate this advertisement this furthest?
- Which 3 experts should review this submitted article?
- Which people should I talk to at the upcoming conference and what topics should I talk to them about?

SOCIAL + COMMUNICATION + EXPERTISE + EVENT GRAPH



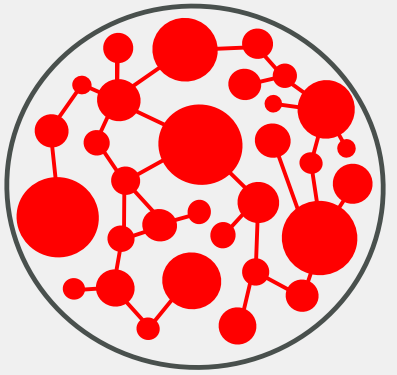
PATTERN IDENTIFICATION

- This connectivity pattern is a sign of financial fraud. When this motif is found, a red flag will be raised.

TRANSACTION GRAPH

- Healthy discourse is typified by a discussion board with a branch factor in this range and a concept clique score in this range.

DISCUSSION GRAPH



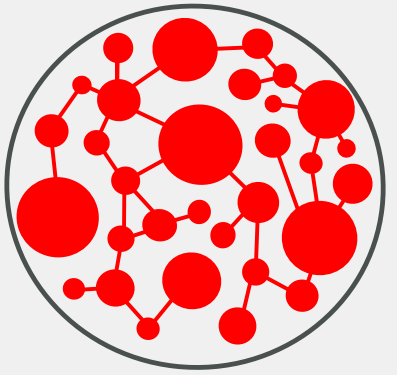
KNOWLEDGE DISCOVERY

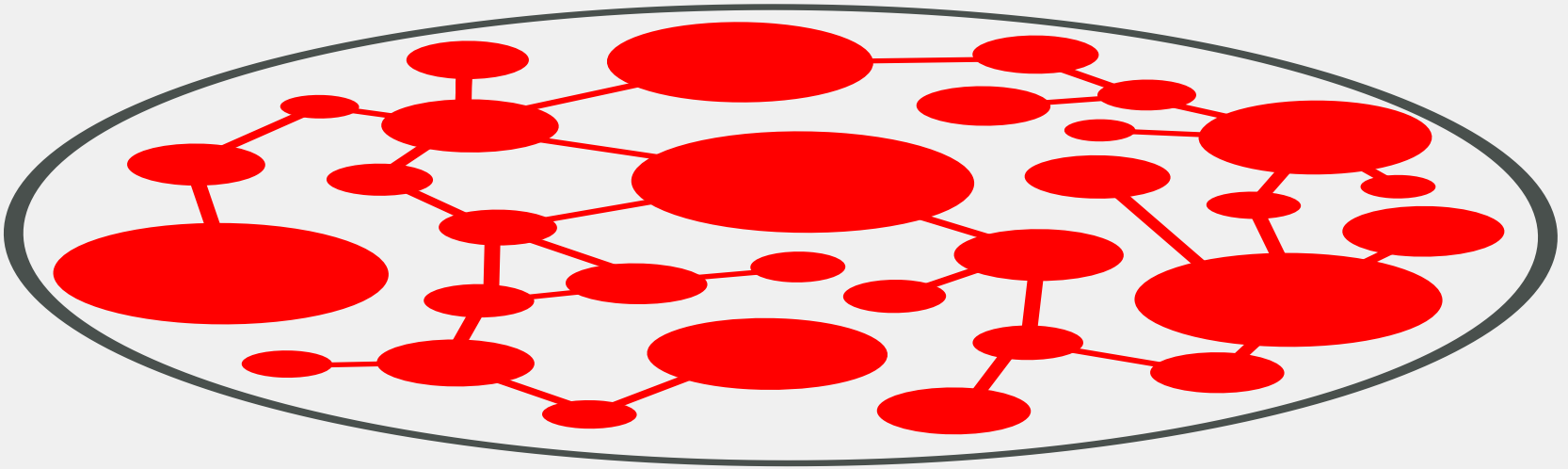
- The terms "ice", "fans", "stanley cup," are classified as "sports"

WIKIPEDIA GRAPH

- Given that all identified birds fly, it can be deduced that all birds fly. If contrary evidence is provided, then this "fact" can be retracted.

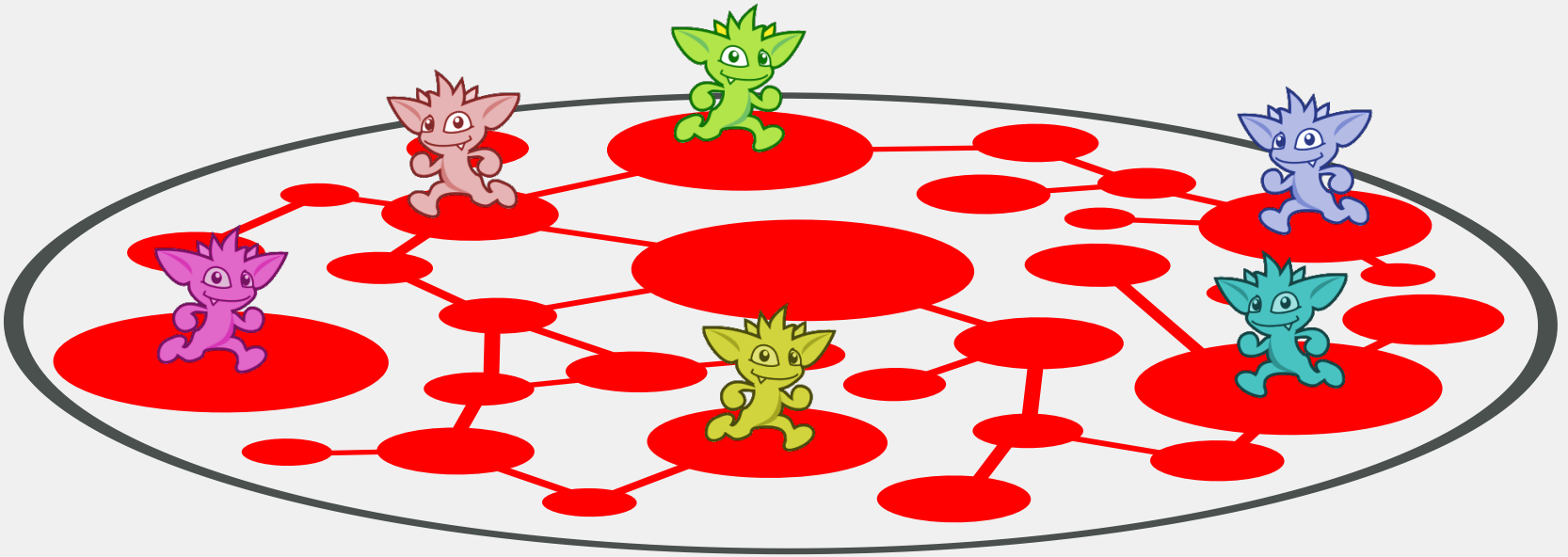
EVIDENTIAL LOGIC GRAPH



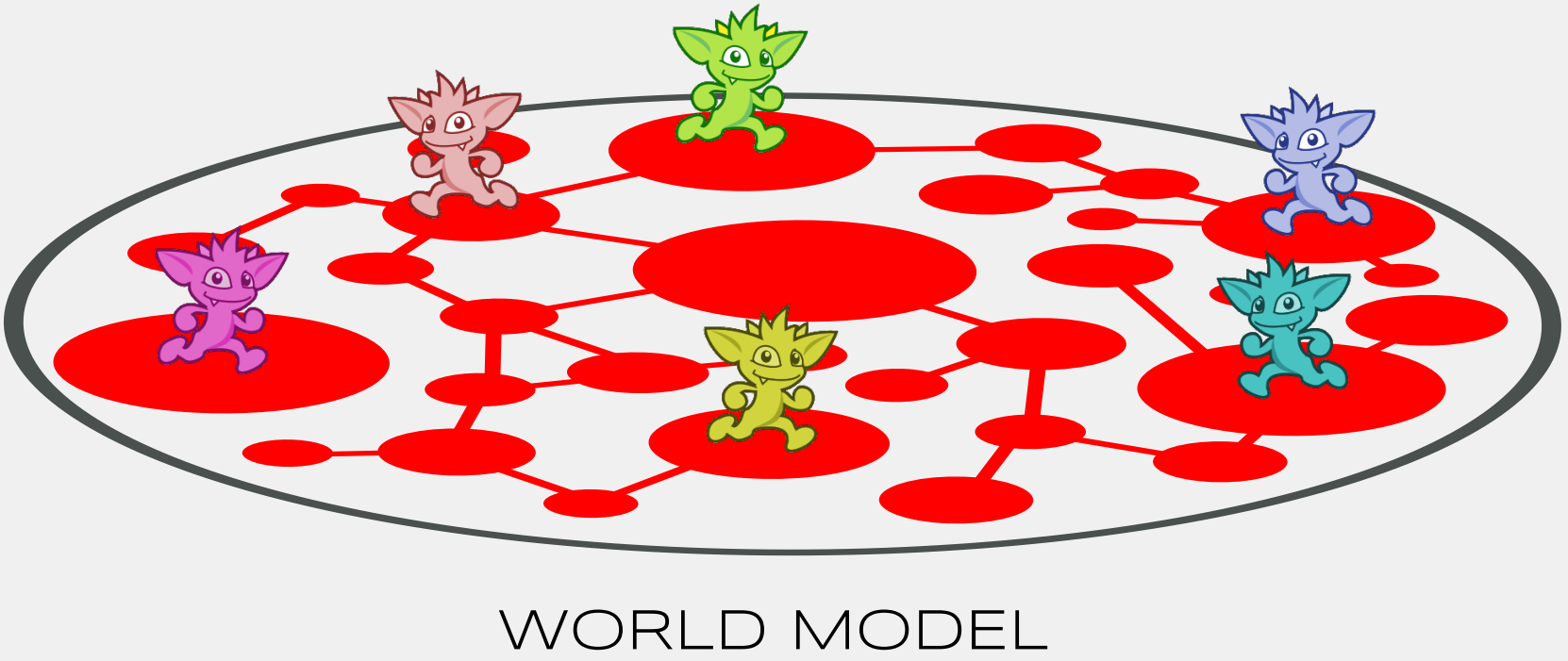


WORLD MODEL

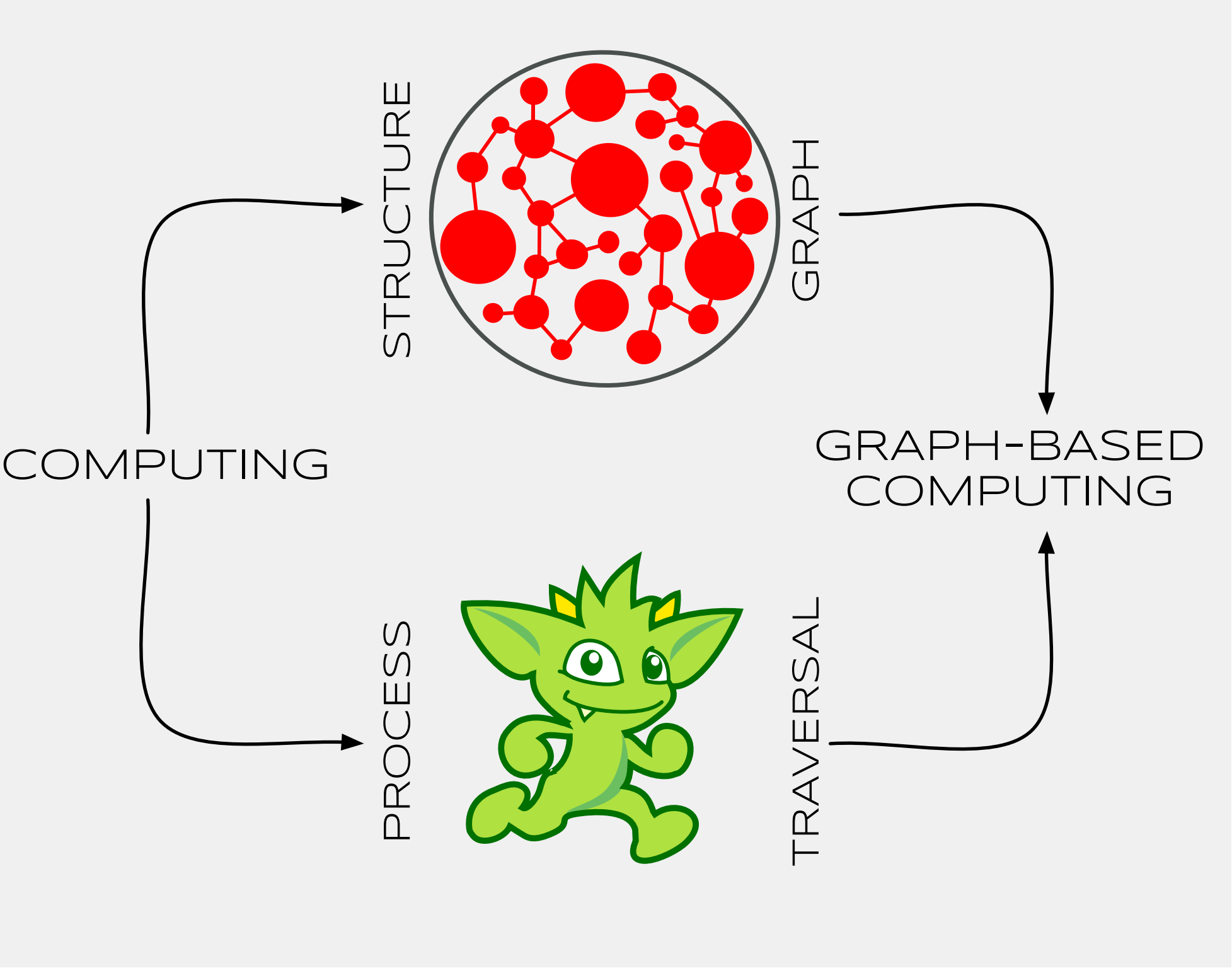
WORLD PROCESSES



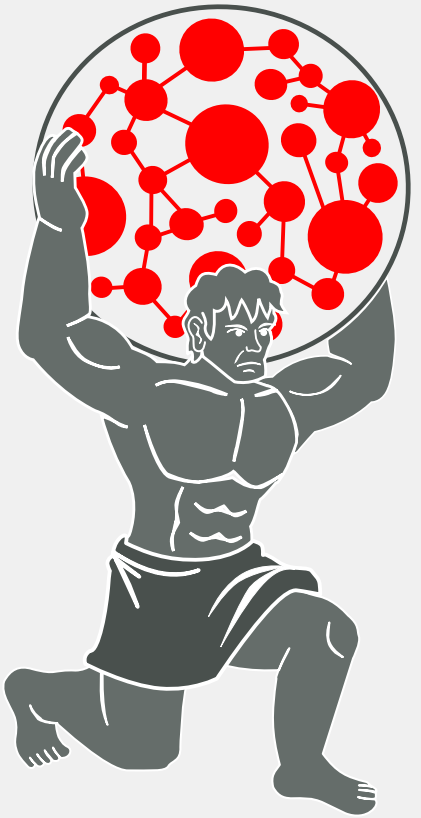
WORLD MODEL



A single world model and various types of traversers moving through that model to solve problems.


STRUCTURE
GRAPH
COMPUTING
GRAPH-BASED
COMPUTING
PROCESS
TRAVERSAL

PART 2:
THE PATH TO TITAN 1.0



WHY CREATE TITAN?

A number of Aurelius' clients...

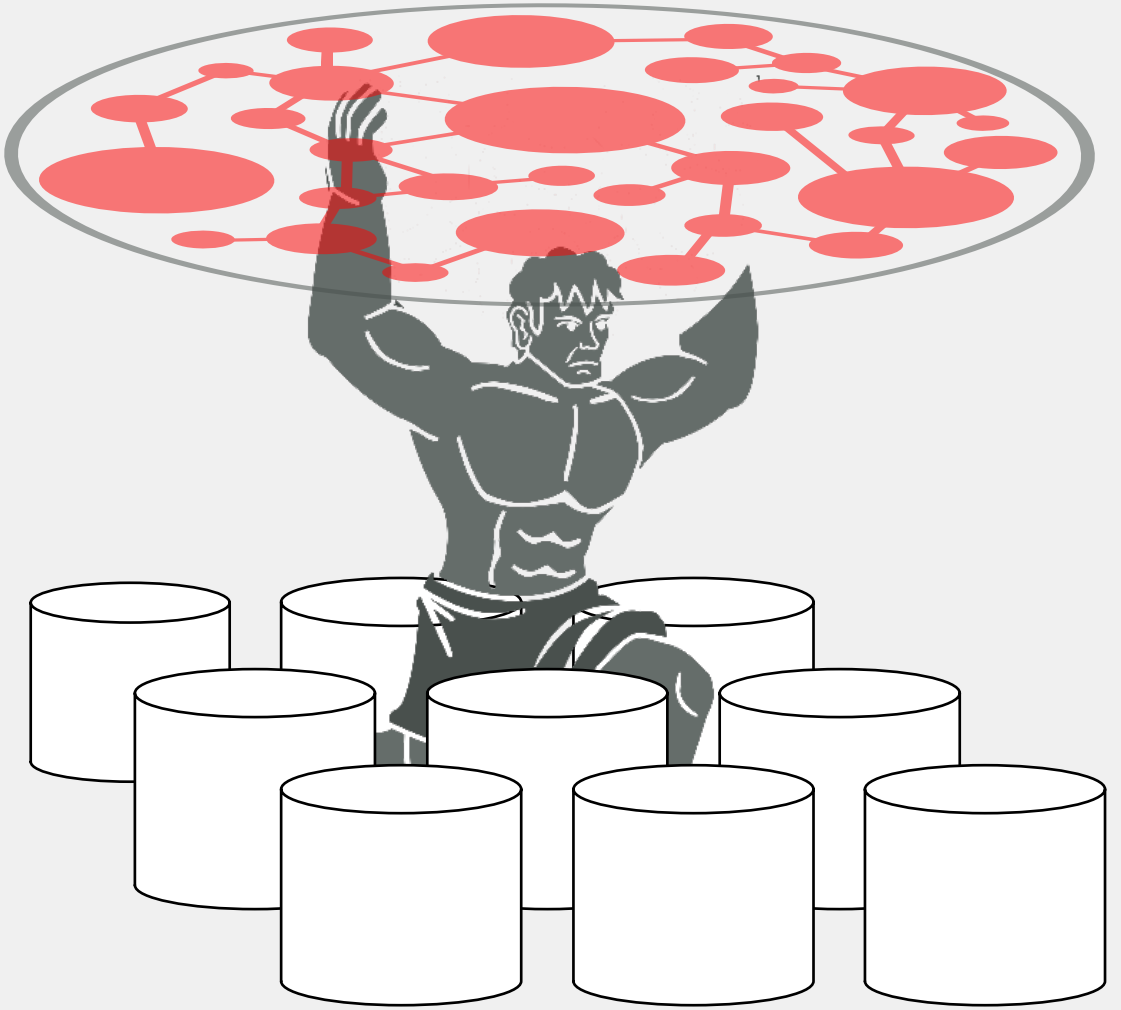
- ...need to represent and process graphs at the 100+ billion edge scale w/ thousands of concurrent transactions.
- ...need both local graph traversals (OLTP) and batch graph processing (OLAP).
- ...desire a free, open source distributed graph database.

TITAN'S KEY FEATURES

Titan provides...

- ..."infinite size" graphs and "unlimited" users by means of a distributed storage engine.
- ...real-time local traversals (OLTP) and support for global batch processing via Hadoop (OLAP).
- ...distribution via the liberal, free, open source Apache2 license.

BACKEND AGNOSTIC



Cassandra

-OR-

APACHE HBASE

TITAN DISTRIBUTED VIA CASSANDRA


```
titan$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(_)-oOOo-----
gremlin> conf = new BaseConfiguration();
==>org.apache.commons.configuration.BaseConfiguration@763861e6
gremlin> conf.setProperty("storage.backend","cassandra");
gremlin> conf.setProperty("storage.hostname","77.77.77.77");
gremlin> g = TitanFactory.open(conf);
==>titangraph[cassandra:77.77.77.77]
gremlin>
```

INHERITED FEATURES

Cassandra

- Continuously available with no single point of failure.
- No write bottlenecks to the graph as there is no master/slave architecture.
- Built-in replication ensures data is available during machine failure.
- Caching layer ensures that continuously accessed data is available in memory.
- Elastic scalability allows for the introduction and removal of machines.

Cassandra available at http://cassandra.apache.org/

TITAN DISTRIBUTED VIA HBASE

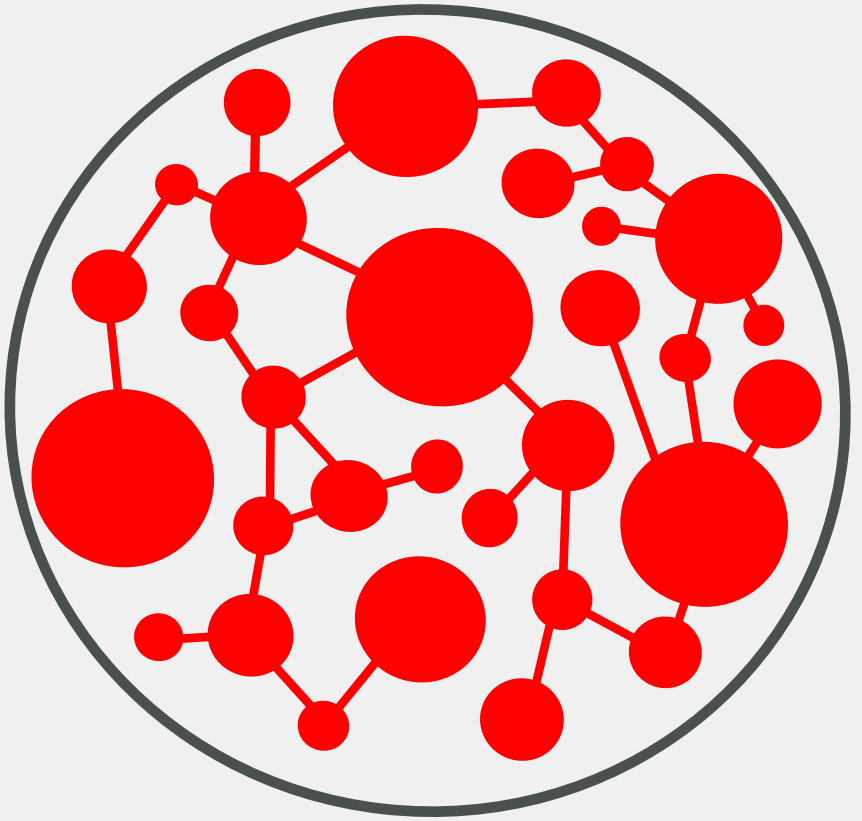

```
titan$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(_)-oOOo-----
gremlin> conf = new BaseConfiguration();
==>org.apache.commons.configuration.BaseConfiguration@763861e6
gremlin> conf.setProperty("storage.backend","hbase");
gremlin> conf.setProperty("storage.hostname","77.77.77.77");
gremlin> g = TitanFactory.open(conf);
==>titangraph[hbase:77.77.77.77]
gremlin>
```

INHERITED FEATURES

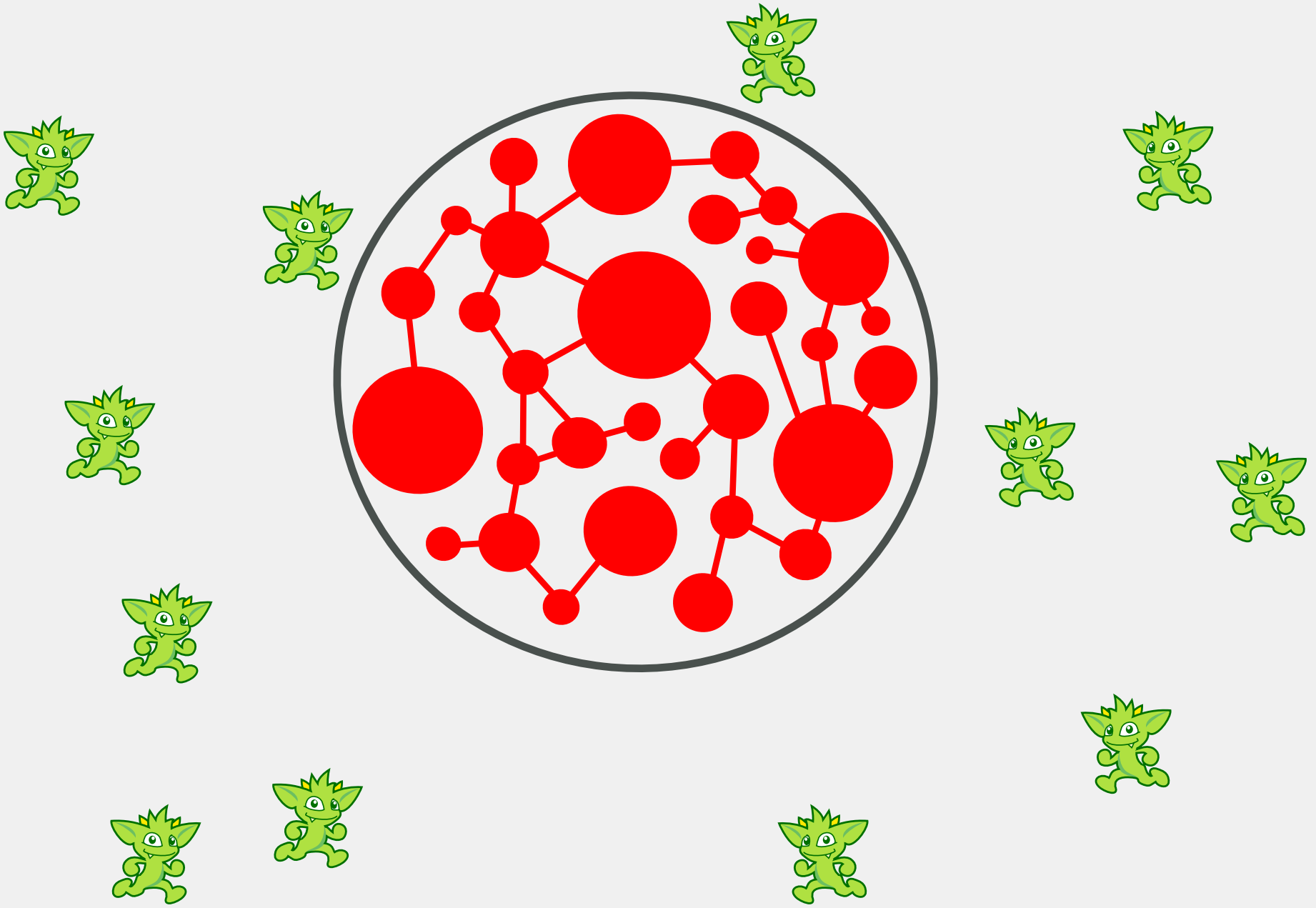
- Strictly consistent reads and writes.
- Linear scalability with the addition of machines.
- Base classes for backing Hadoop MapReduce jobs with HBase tables.
- HDFS-based data replication.
- Generally good integration with the tools in the Hadoop ecosystem.

HBase available at http://hbase.apache.org/

Titan is all about ...

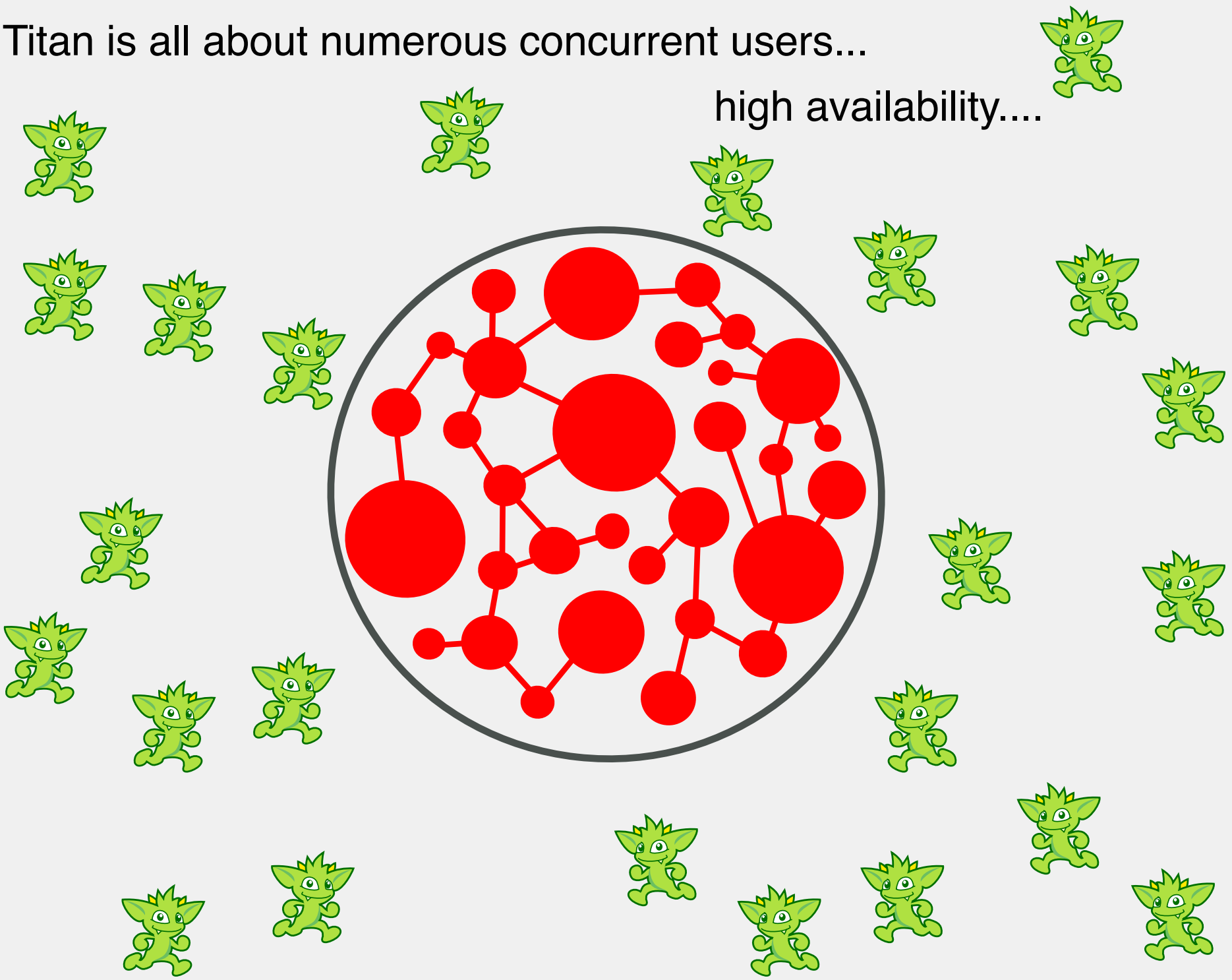


Titan is all about numerous concurrent users...



Titan is all about numerous concurrent users...

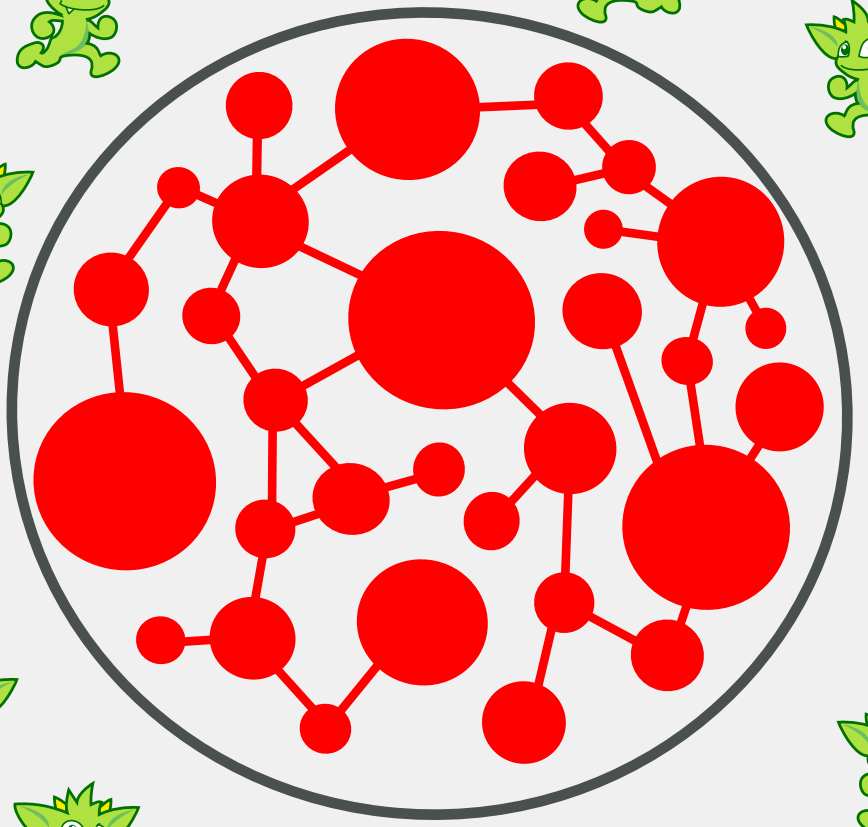
high availability....



Titan is all about numerous concurrent users...

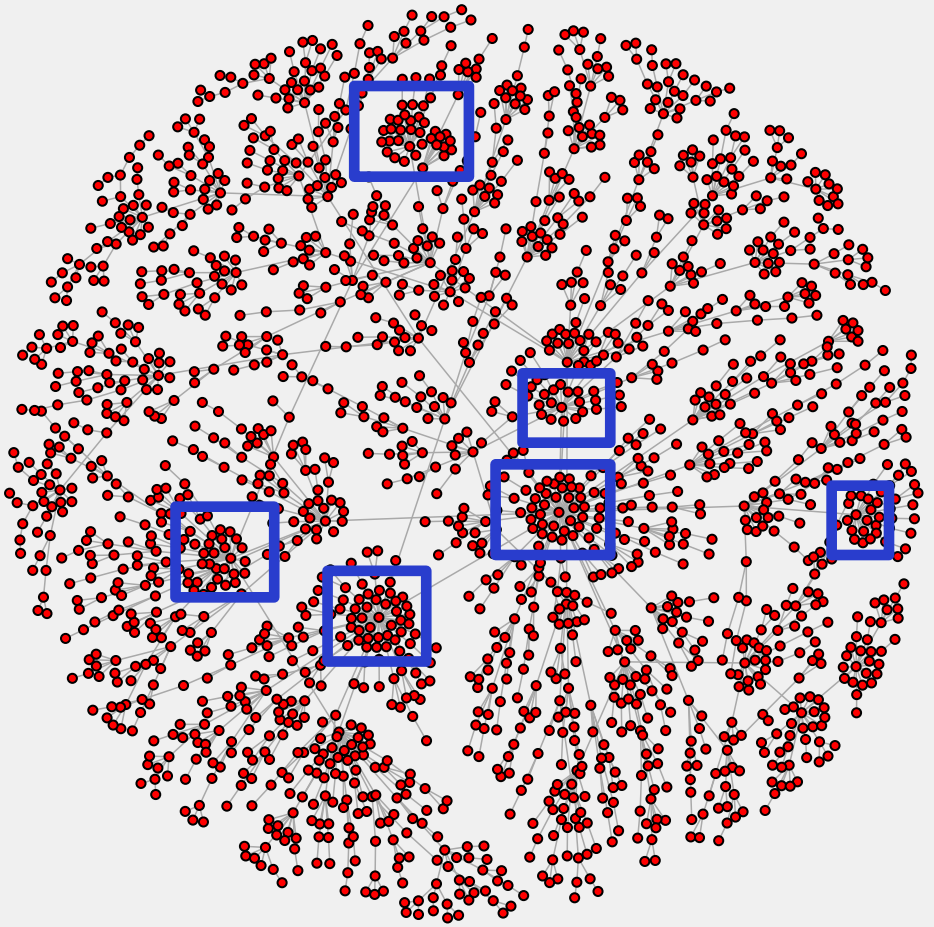
high availability....

dynamic scalability...



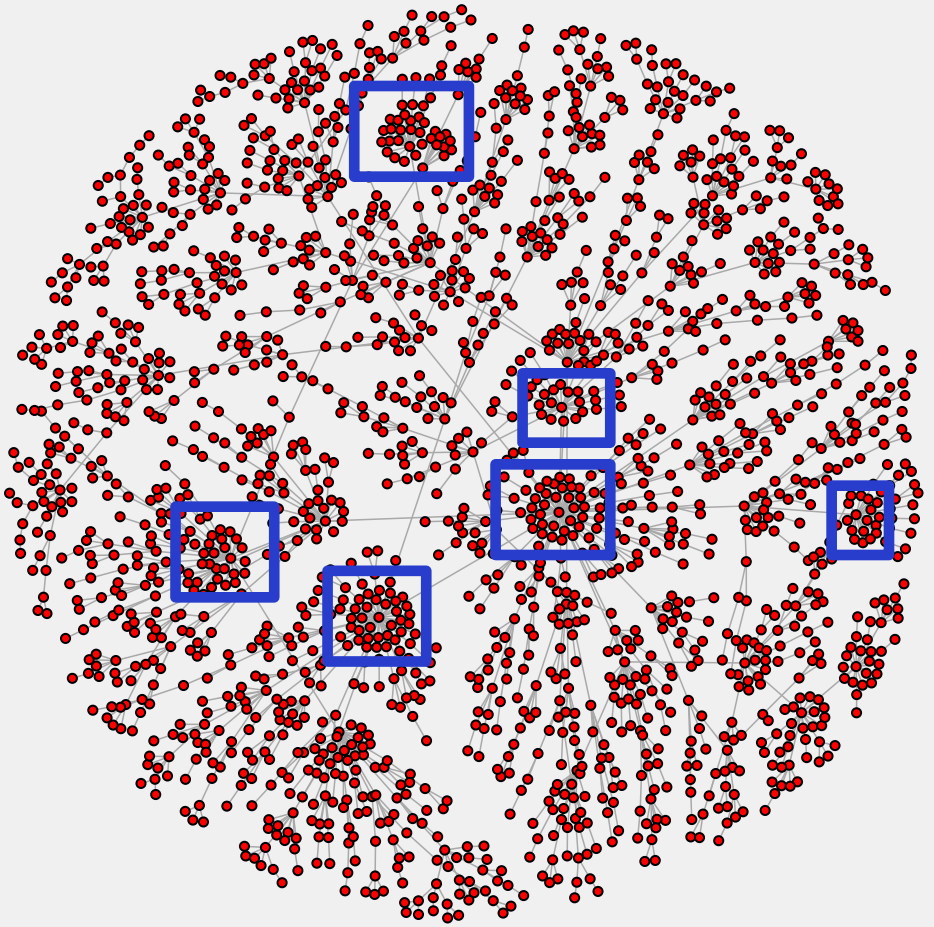
VERTEX-CENTRIC INDICES
THE SUPER NODE PROBLEM

- Natural, real-world graphs contain vertices of high degree.
- Even if rare, their degree ensures that they exist on many paths.
- Traversing a high degree vertex means touching numerous incident edges and potentially touching most of the graph in only a few steps.



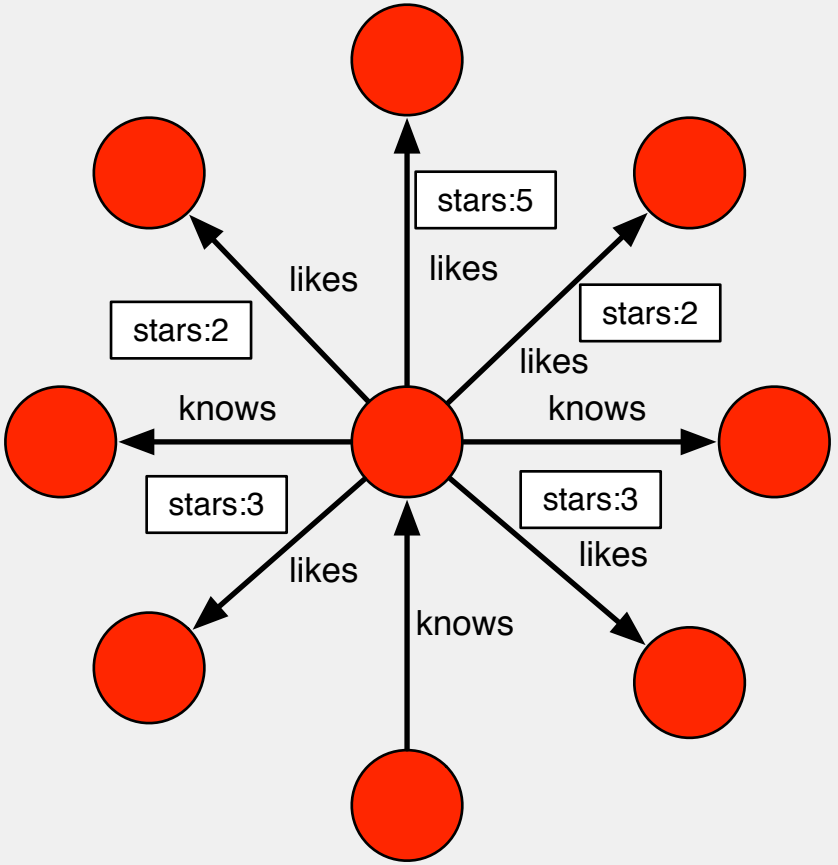
VERTEX-CENTRIC INDICES
A SUPER NODE SOLUTION

- A "super node" only exists from the vantage point of classic "textbook style" graphs.
- In the world of property graphs, intelligent disk-level filtering can interpret a "super node" as a more manageable low-degree vertex.
- Vertex-centric querying utilizes sort orders for speedy lookup of incident edges with particular qualities.



VERTEX-CENTRIC INDICES
PUSHDOWN PREDICATES

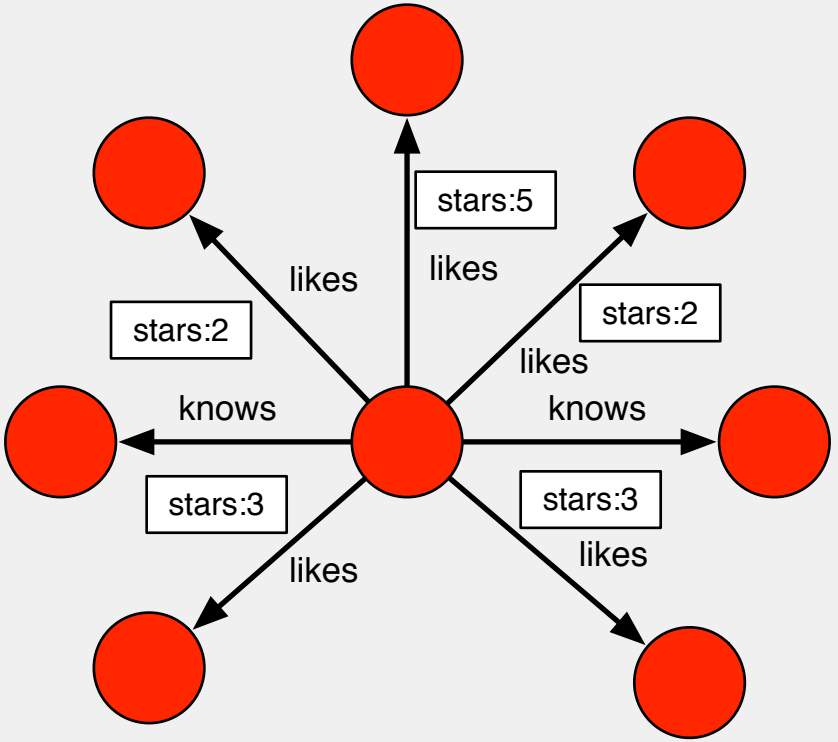
```
vertex.bothE()
```



8 edges

VERTEX-CENTRIC INDICES
PUSHDOWN PREDICATES

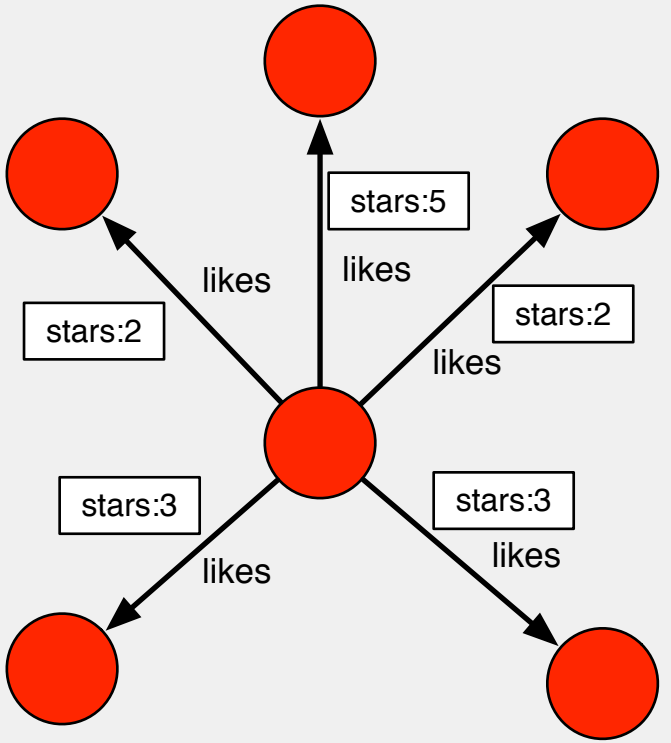
```
vertex.outE()
```



7 edges

VERTEX-CENTRIC INDICES
PUSHDOWN PREDICATES

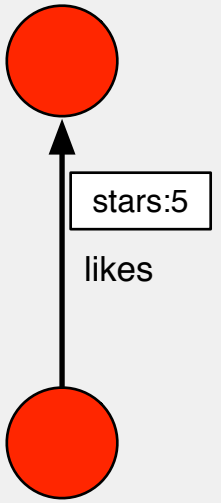
```
vertex.outE("likes")
```



5 edges

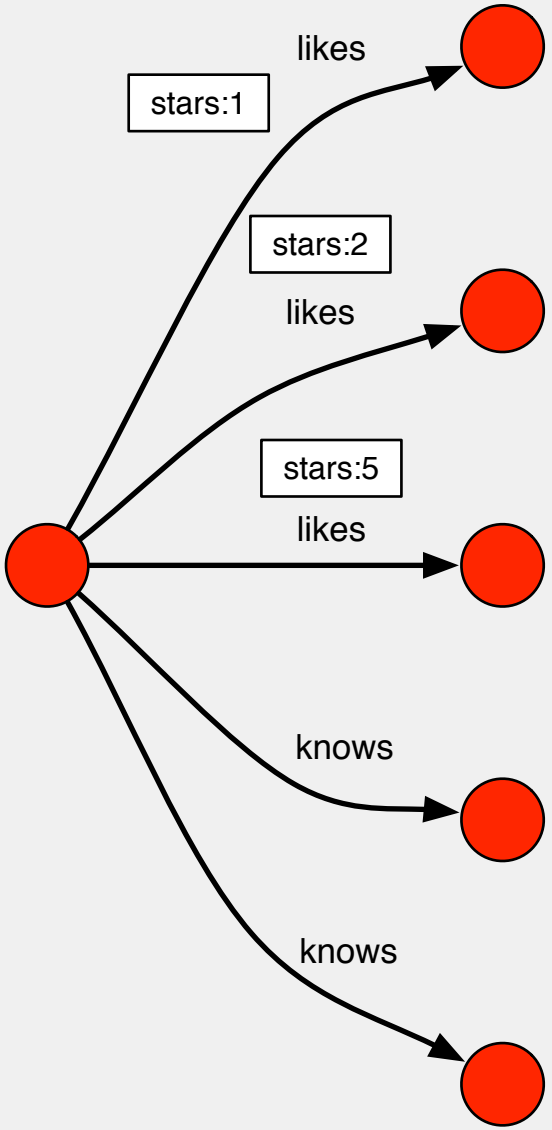
VERTEX-CENTRIC INDICES
PUSHDOWN PREDICATES

```
vertex.outE("likes").has("stars",5)
```

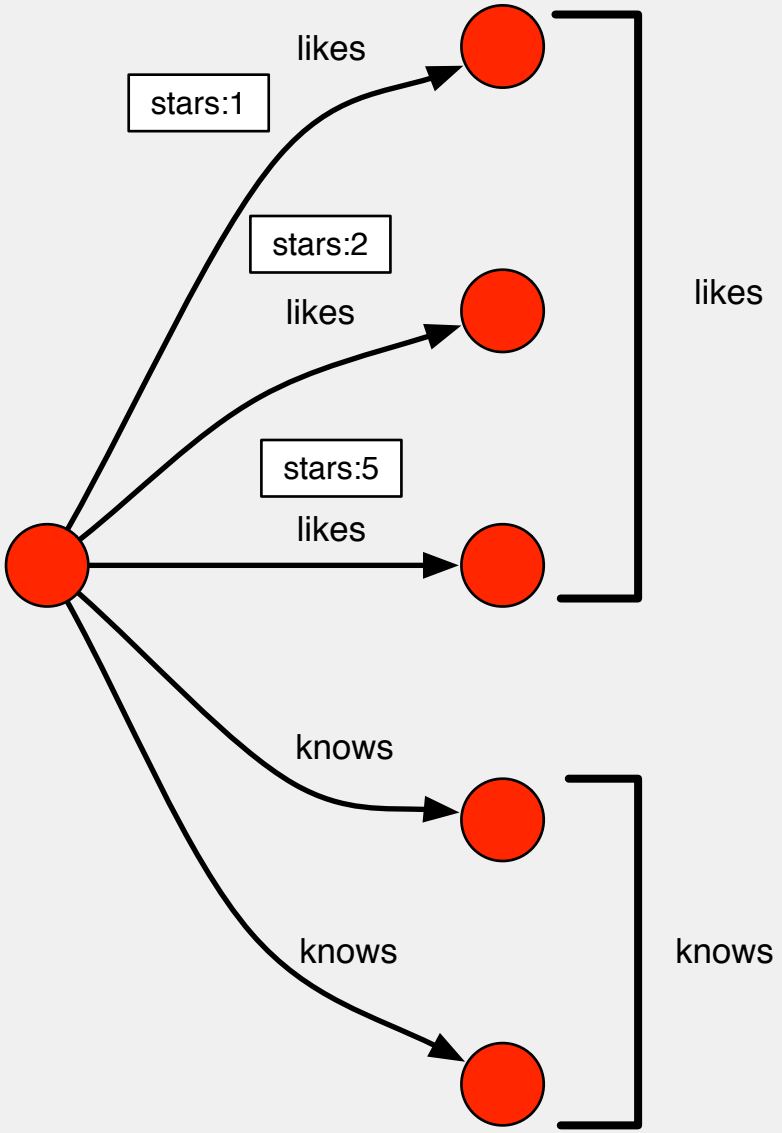


1 edge

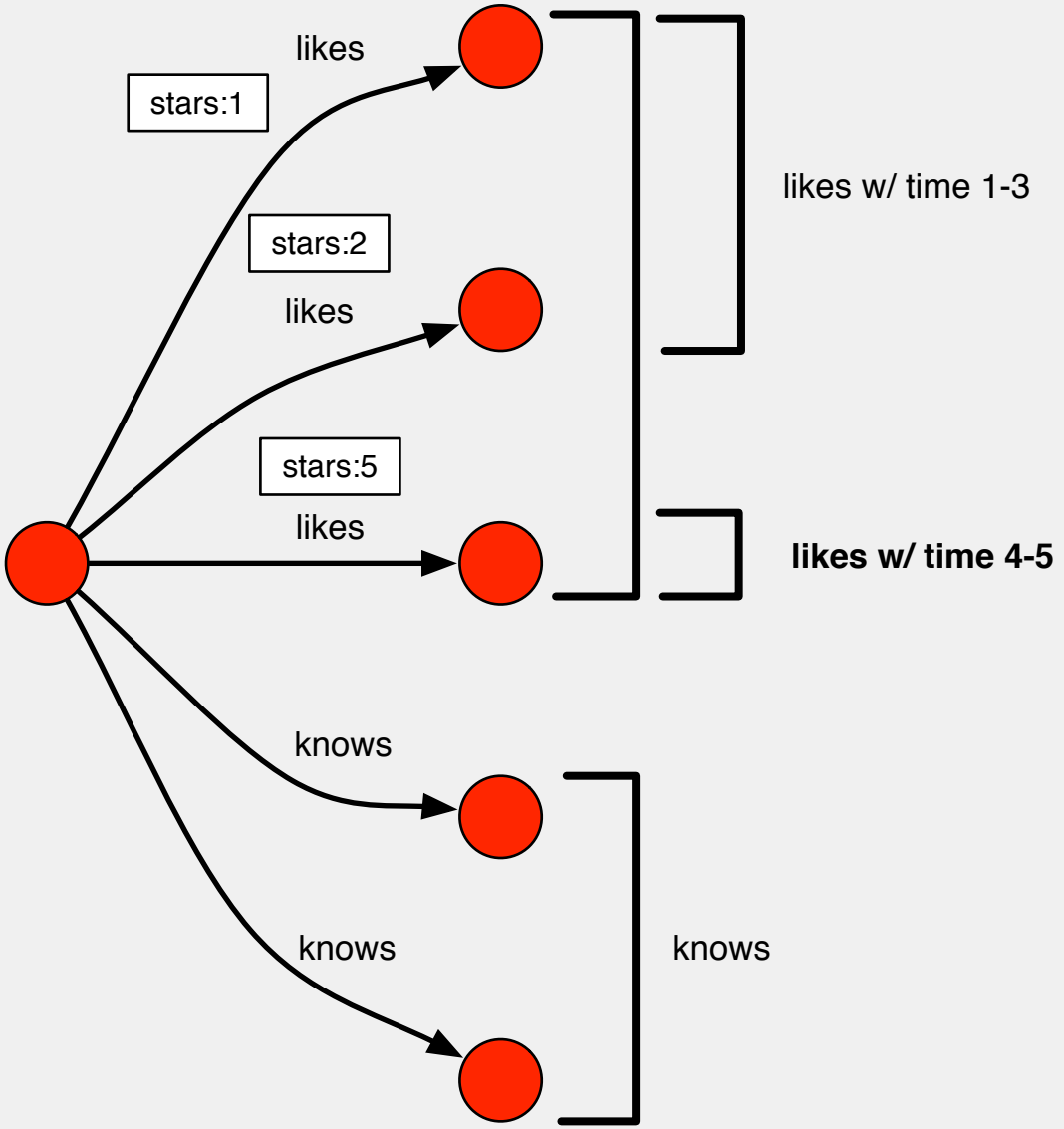
VERTEX-CENTRIC INDICES
DISK-LEVEL SORTING/INDEXING



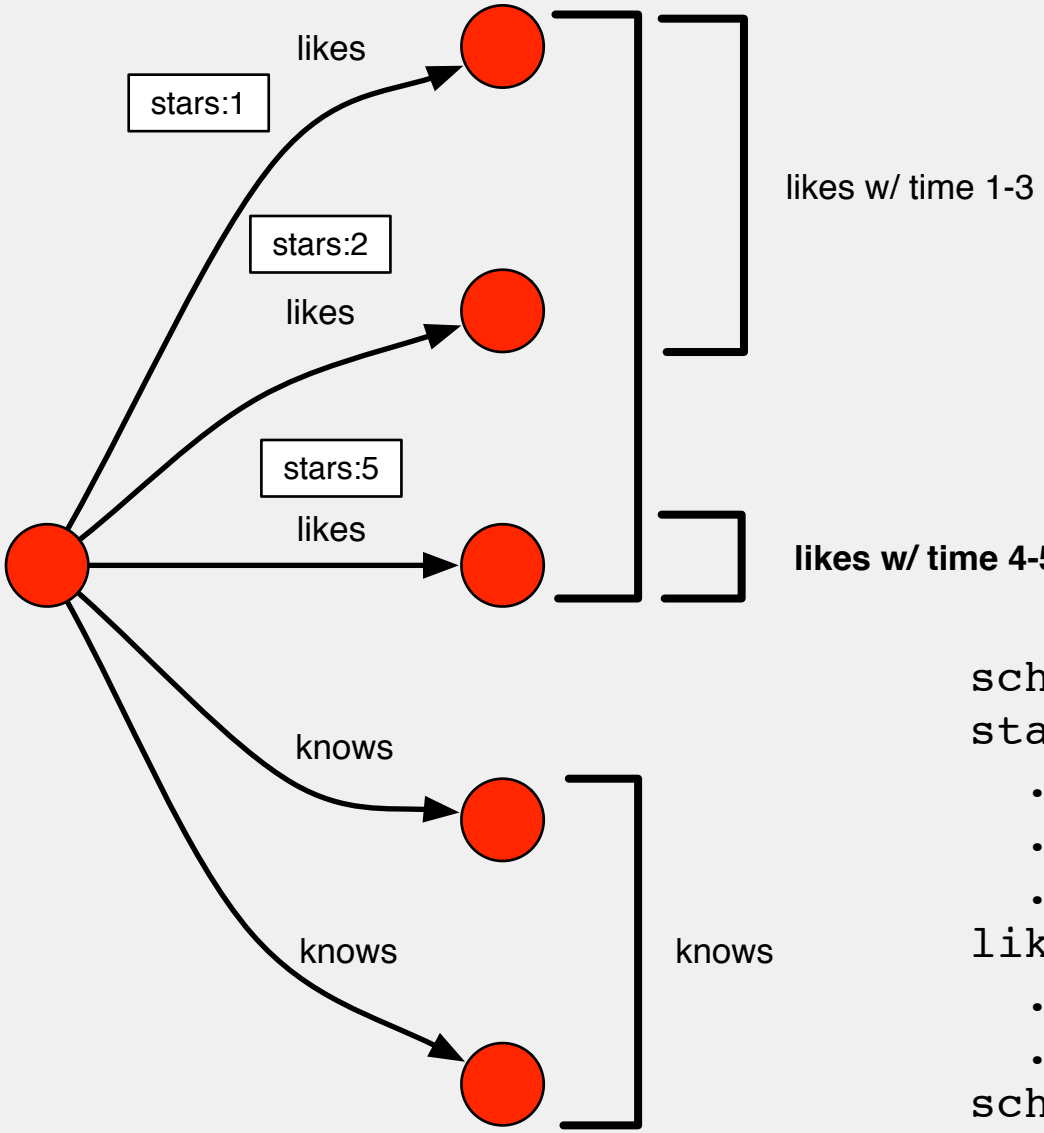
VERTEX-CENTRIC INDICES
DISK-LEVEL SORTING/INDEXING



VERTEX-CENTRIC INDICES
DISK-LEVEL SORTING/INDEXING

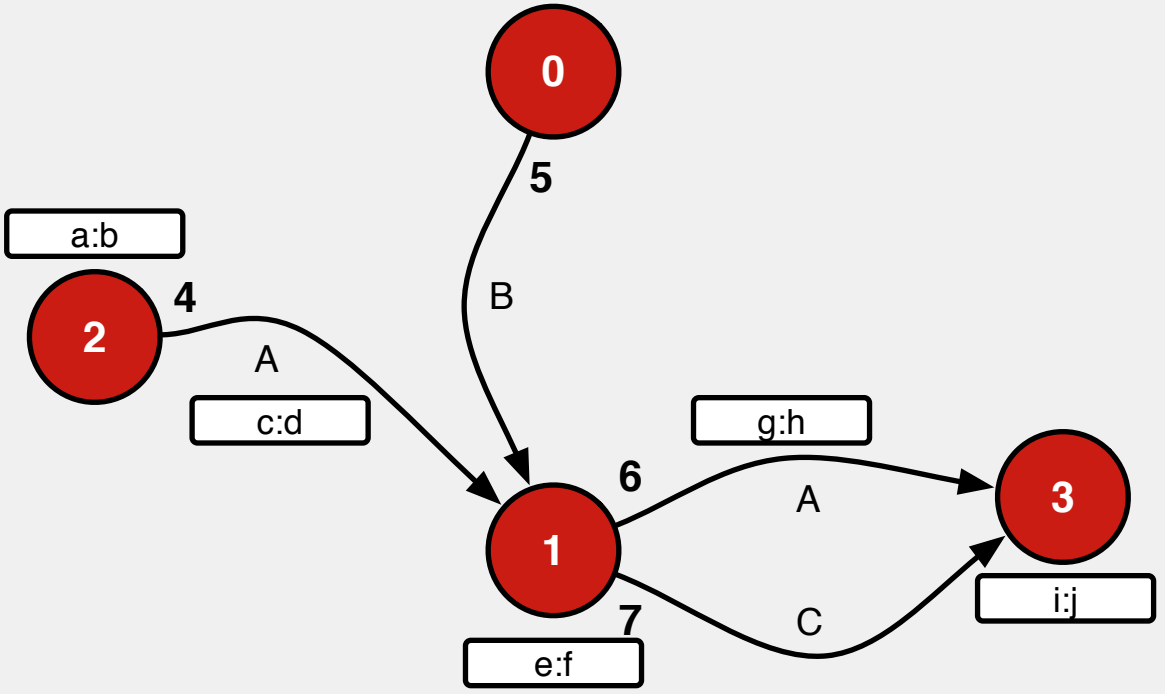


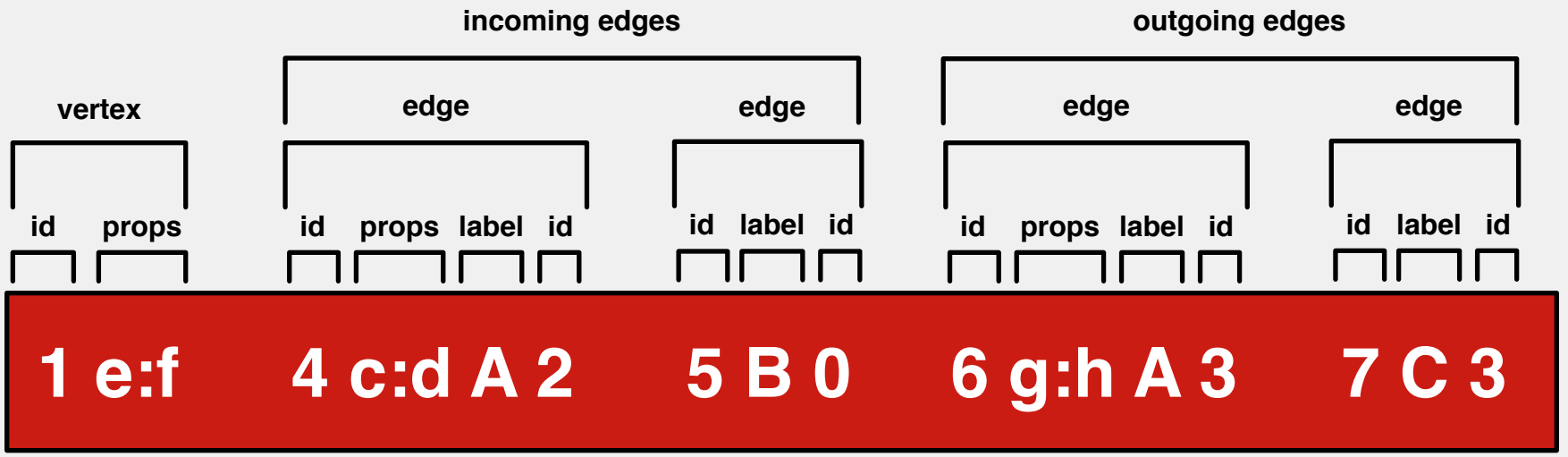
VERTEX-CENTRIC INDICES
DISK-LEVEL SORTING/INDEXING



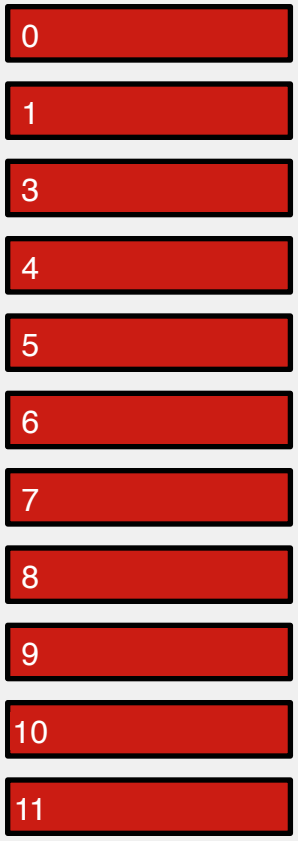

```
schema = g.openManagementSystem()
stars = schema
  .makePropertyKey("stars")
  .dataType(Integer.class)
  .make()
likes = schema
  .makeEdgeLabel('likes')
  .make()
schema.buildEdgeIndex(likes,
  'timeIdx',BOTH,DESC,stars)
```

THE TITAN OLAP STORY

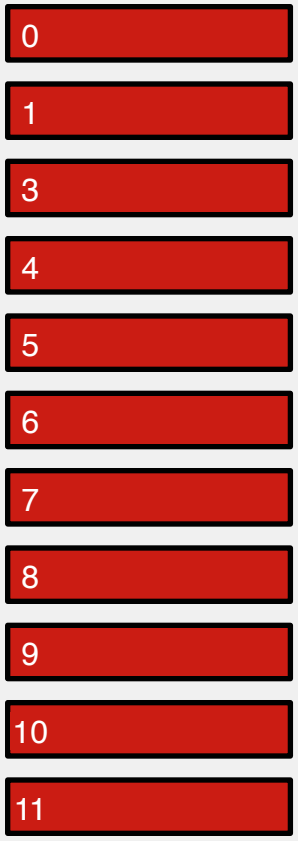


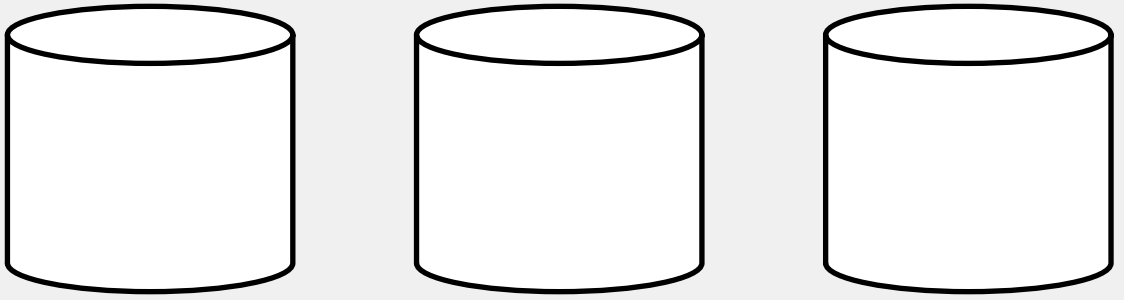


AN ADJACENCY LIST

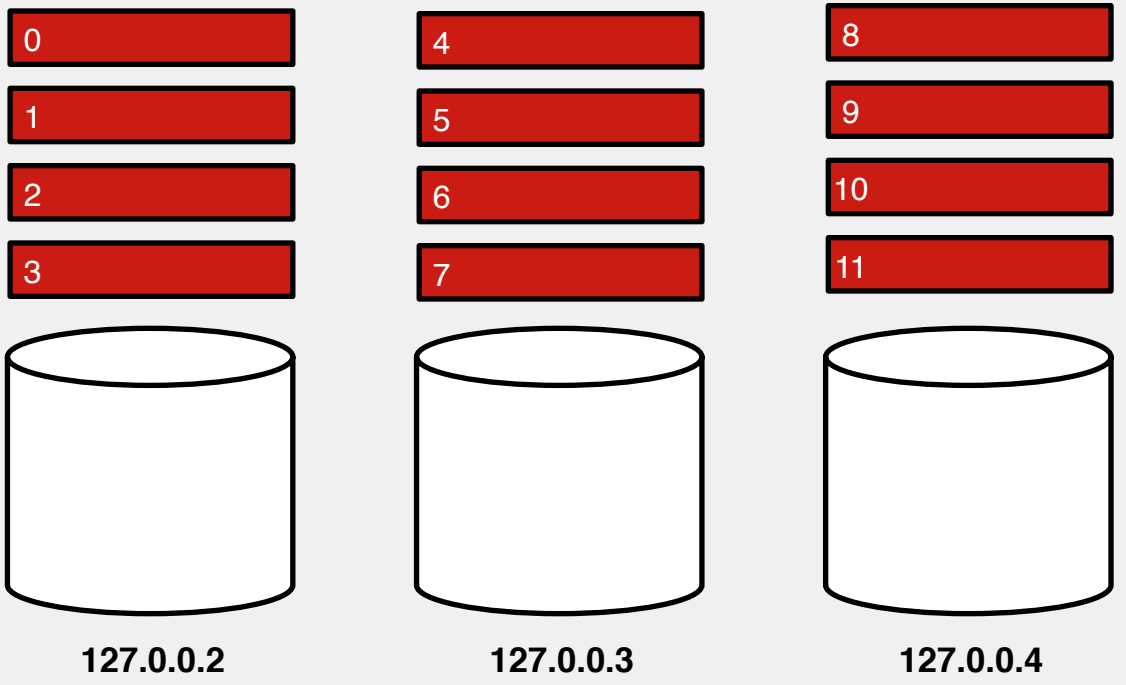


AN ADJACENCY LIST
+
CLUSTER





A DISTRIBUTED ADJACENCY LIST



HADOOP

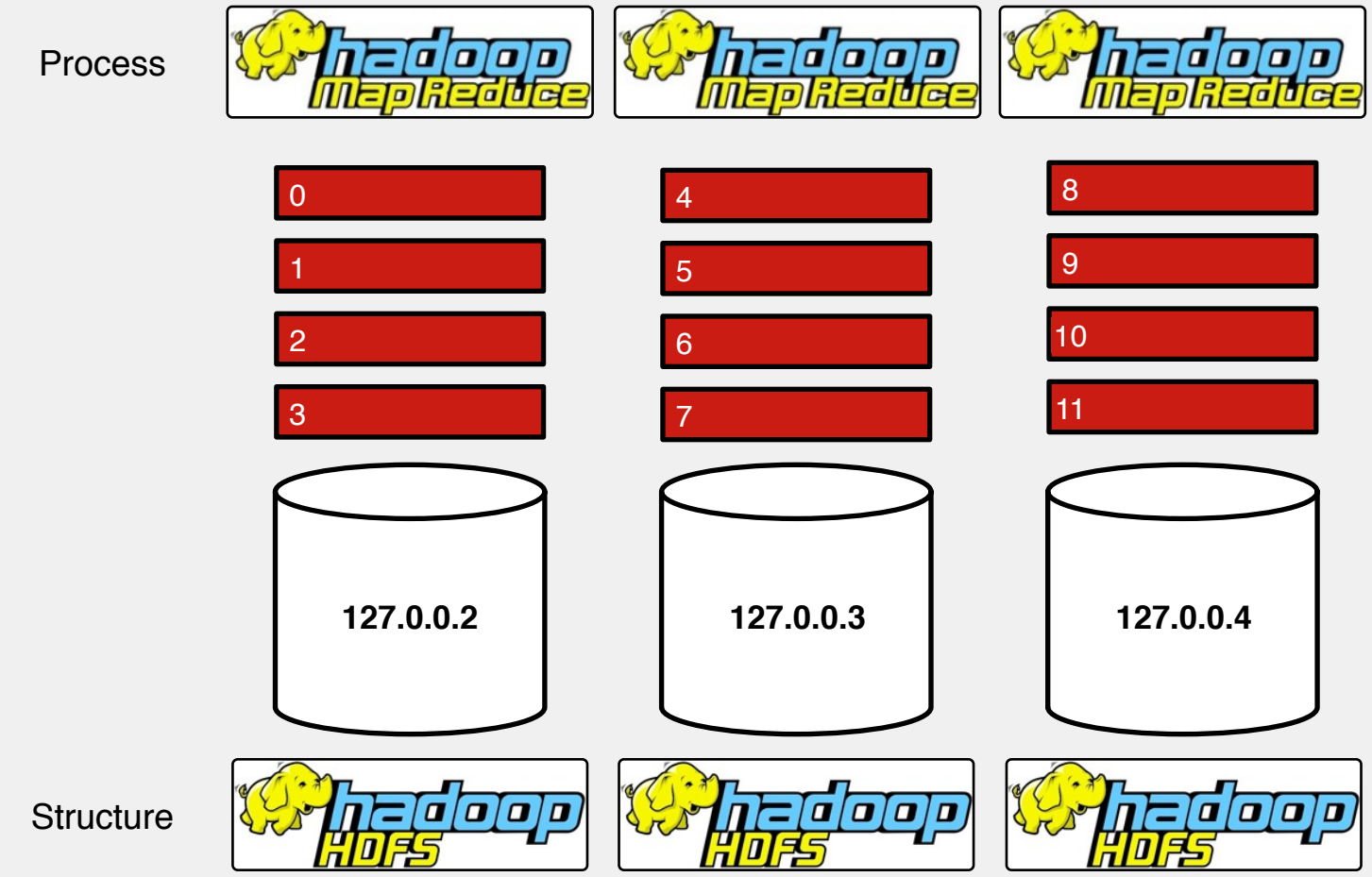
Hadoop is a distributed computing platform composed of two key components:

HDFS:
A distributed file system that stores arbitrarily large files within a cluster.

MapReduce:
A parallel functional computing model for key/value pair data.

http://hadoop.apache.org

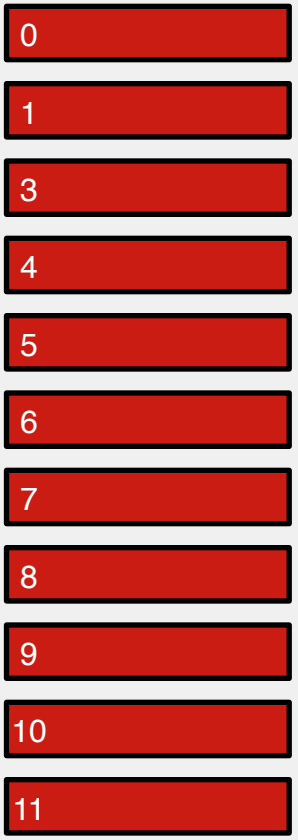
FAUNUS AND HADOOP



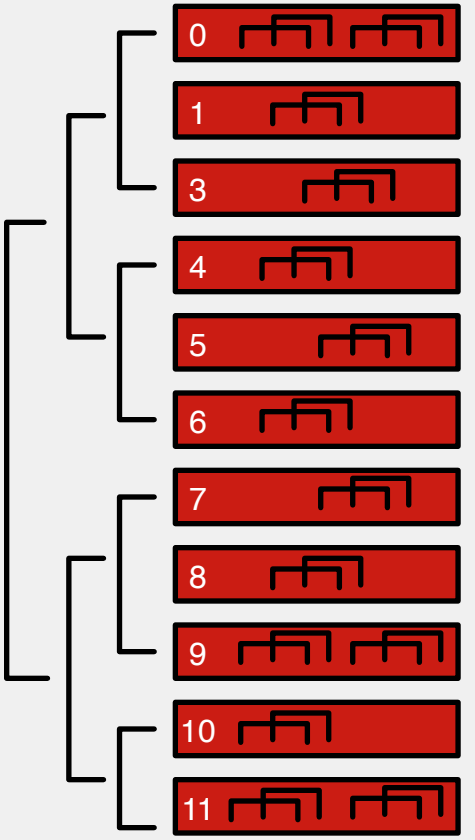
Faunus provides graph input/output formats (structure) and a traversal language for graphs (process).

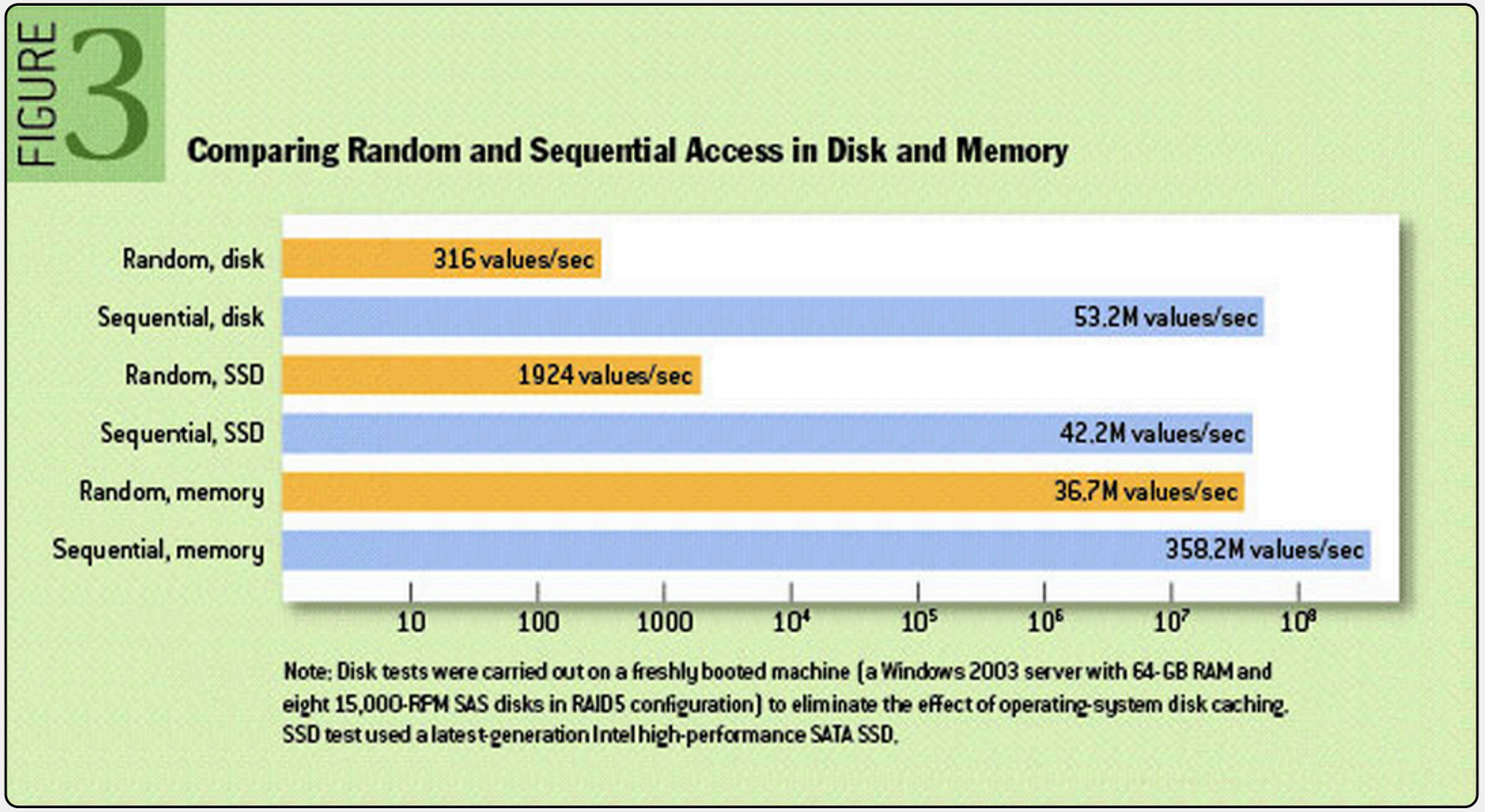
THE TITAN OLAP STORY

Serial Key/Value Data Structure



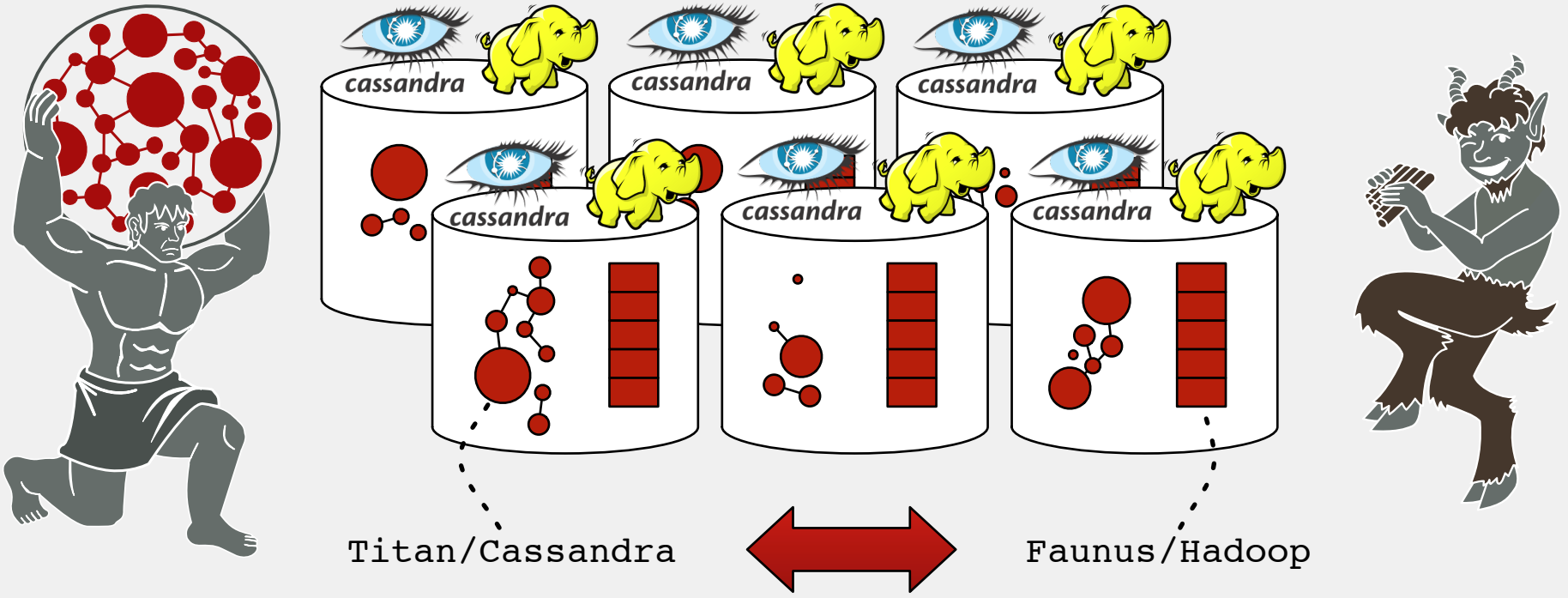
Indexed Key/Indexed Value Data Structure





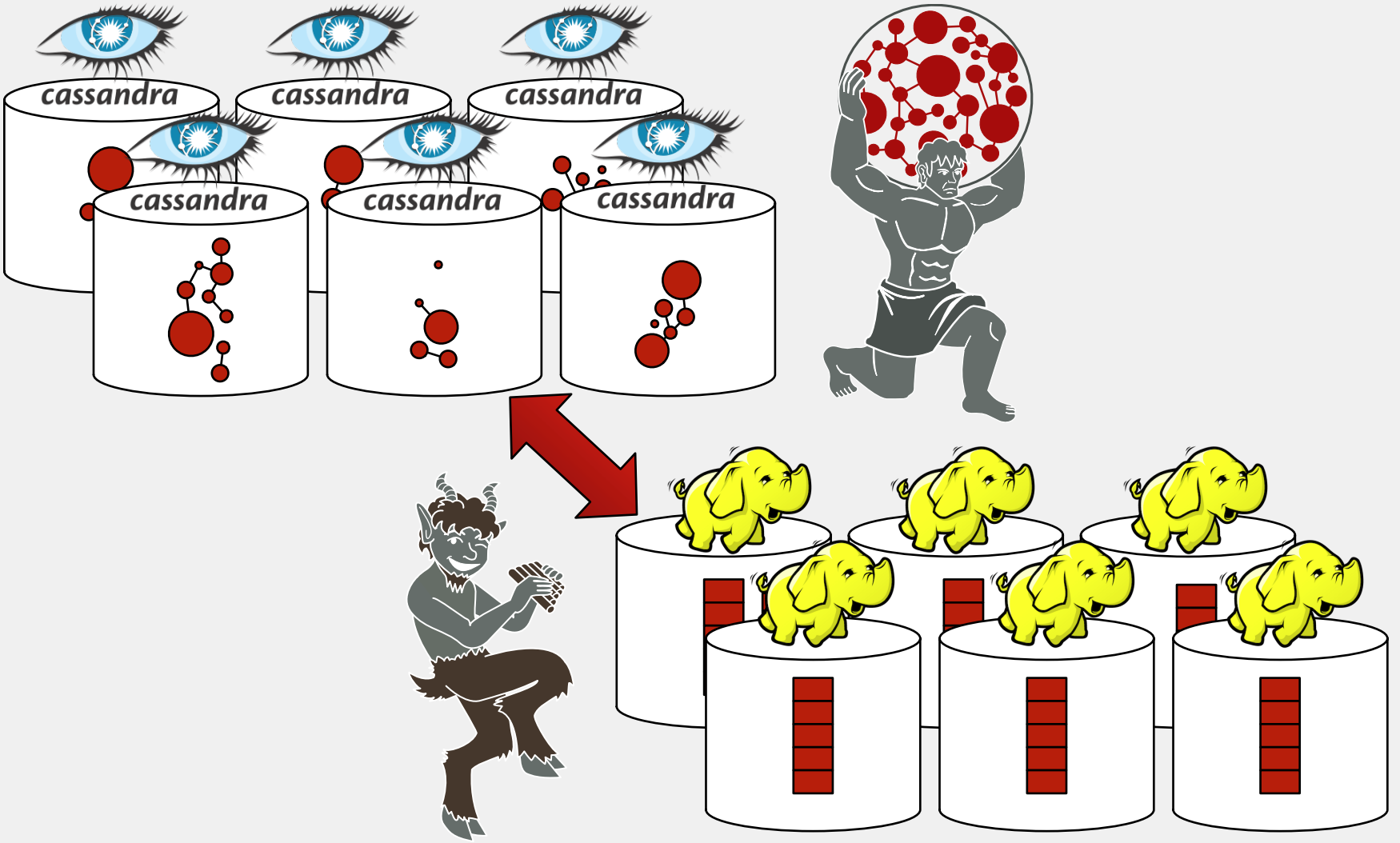
Adam Jacobs. 2009. The Pathologies of Big Data. Communications of the ACM 52, 8 (August 2009), 36-44. doi:10.1145/1536616.1536632 http://doi.acm.org/10.1145/1536616.1536632

INTRA-CLUSTER CONFIGURATION



- Data is processed on the machine where it is located.
- Limited network communication.

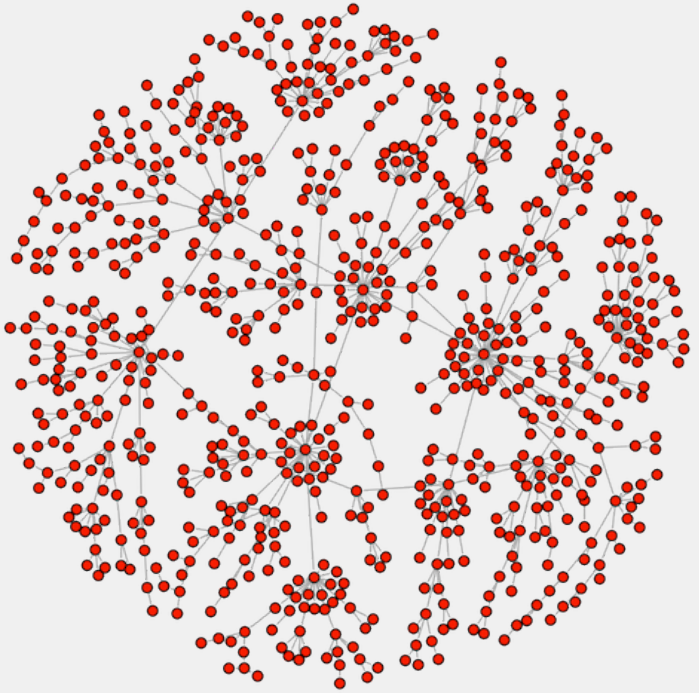
INTER-CLUSTER CONFIGURATION



- Graph data is offloaded to another cluster.
- Repeated analysis does not interfere with production graph database.

THE TITAN OLAP STORY

Titan currently provides a Hadoop OLAP engine
Titan-Hadoop (nicknamed **Faunus**)



Titan 1.0 will also provide an in-memory, off head, single-machine OLAP engine.
Titan-Memory (nicknamed **Fulgora**)

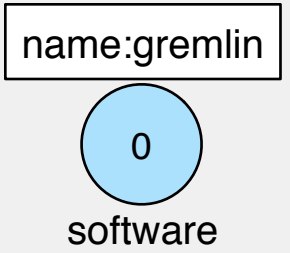
PART 3:
THE PATH TO TINKERPOP3

TINKERPOP3 FEATURES

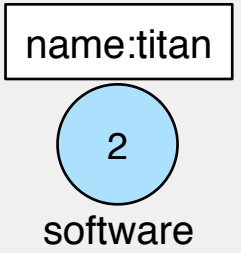
- A standard, vendor-agnostic graph query language.
- Both OLTP and OLAP engines for evaluating queries.
- A way for developers to create custom, domain specific variants.
- A server system for pushing queries to the server for evaluation.
- A vertex-centric computing framework for distributed graph algorithms.

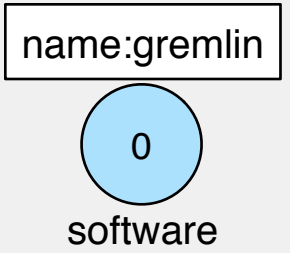
```
gremlin>
```

```
gremlin> gremlin = g.addVertex(label,'software','name','gremlin')
==>v[0]
```



```
gremlin> gremlin = g.addVertex(label,'software','name','gremlin')
==>v[0]
gremlin> titan = g.addVertex(label,'software','name','titan')
==>v[2]
```





```
gremlin> gremlin = g.addVertex(label,'software','name','gremlin')
==>v[0]
gremlin> titan = g.addVertex(label,'software','name','titan')
==>v[2]
gremlin> titan.addEdge('dependsOn',gremlin,'since','0.1')
==>e[4][2-dependsOn->0]
```

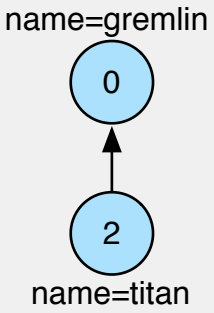
Gremlin is a style of graph traversal that is heavily influenced by functional programming.

```
gremlin> gremlin = g.addVertex(label,'software','name','gremlin')
==>v[0]
gremlin> titan = g.addVertex(label,'software','name','titan')
==>v[2]
gremlin> titan.addEdge('dependsOn',gremlin,'since','0.1')
==>e[4][2-dependsOn->0]

////////////////////////////////////////////////////////////////

gremlin> g.V
==>v[0]
==>v[2]
gremlin> g.V.has('name','titan')
==>v[2]
gremlin> g.V.has('name','titan').outE
==>e[4][2-dependsOn->0]
gremlin> g.V.has('name','titan').outE.since
==>0.1
gremlin> g.V.has('name','titan').out.name
==>gremlin
```

```
g.V.has('name','titan').out.name
```

What are the names of the vertices outgoing adjacent to Titan?

```
g.V.has('name','titan').out.name
```



For the graph 'g'...

Traversals are defined using fluent, function composition.

```
g.V.has('name','titan').out.name
```

name=gremlin

name=titan

…get all the vertices...

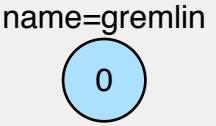
Objects flow through the traversal in a lazy evaluation manner.

```
g.V.has('name','titan').out.name
```

…filter all vertices that don't have the name 'titan'…

The general functions are map(), flatMap(), filter(), sideEffect(), and branch().

```
g.V.has('name','titan').out.name
```



…get all outgoing adjacent vertices…

Any programming language can provide a Gremlin implementation.

```
g.V.has('name','titan').out.name
```

gremlin

…get the names of those vertices...

Any graph system vendor can provide a binding to Gremlin.

THE GENERIC STEPS

$S \rightarrow S \cup \emptyset$


```
has()
retain()
except()
interval()
simplePath()
dedup()
timeLimit()
...
```

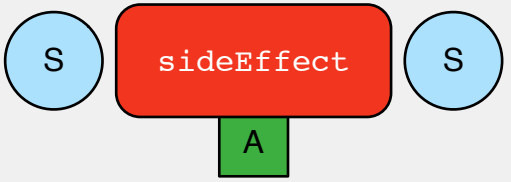
$S \rightarrow E$


```
back()
fold()
valueMap()
match()
orderBy()
shuffle()
...
```

$S \rightarrow E^*$

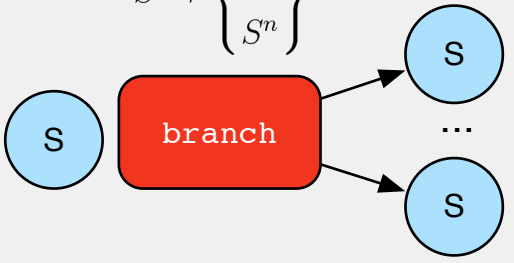

```
out()
outE()
inV()
both()
values()
unfold()
...
```

$S \xrightarrow{A} S$



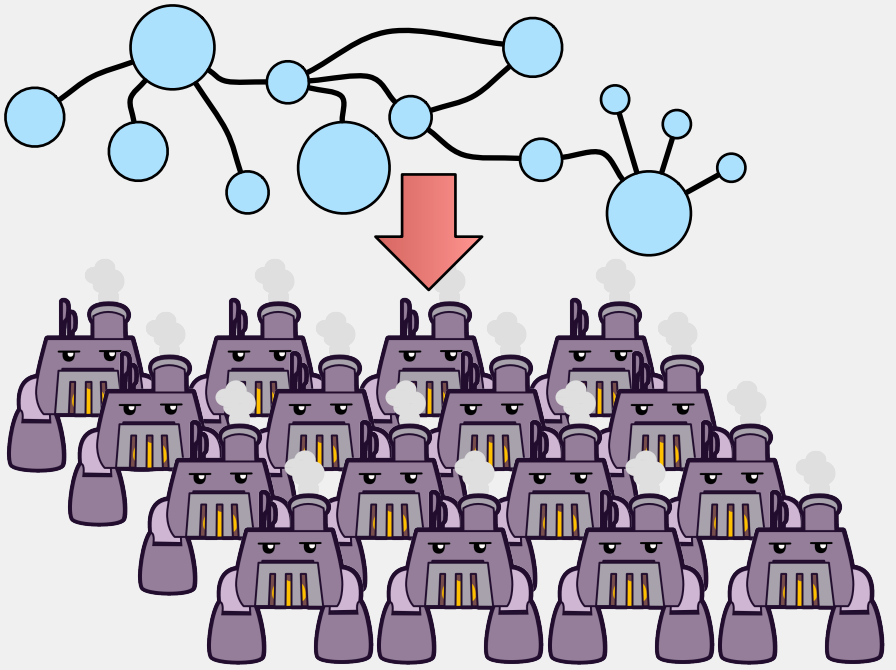

```
aggregate()
store()
groupCount()
groupBy()
tree()
count()
subgraph()
...
```

$S \rightarrow \left\{ \begin{matrix} S^1 \\ \cdots \\ S^n \end{matrix} \right\}$




```
jump()
until()
choose()
union()
...
```

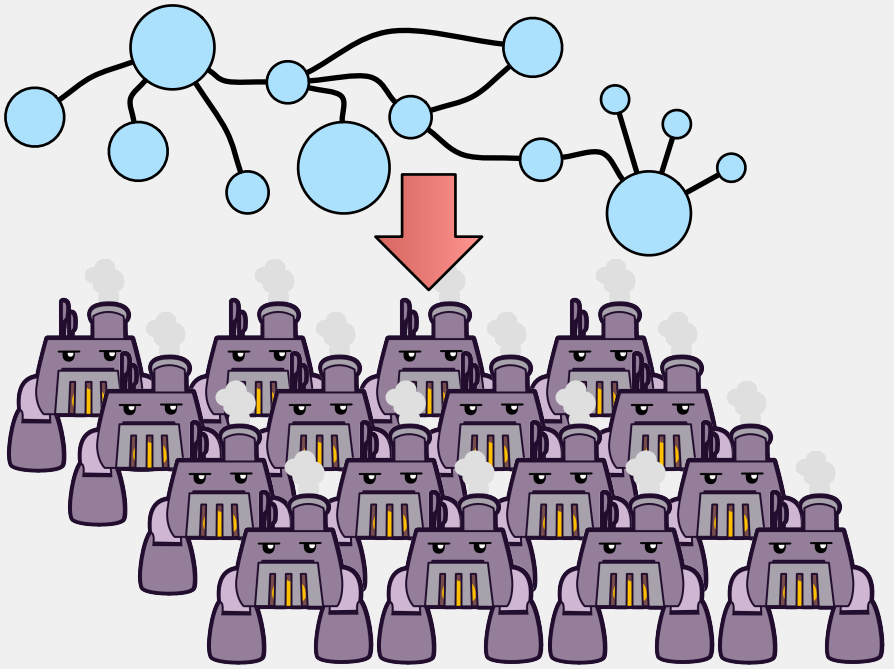
TINKERPOP'S OLAP STORY

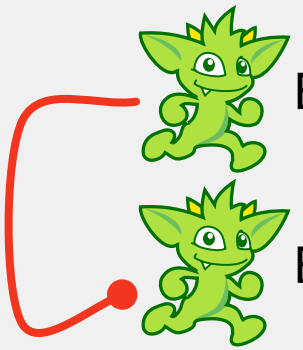


Execute parallel, distributed Gremlin traversals.

Execute vertex-centric message passing algorithms.

TINKERPOP'S OLAP STORY

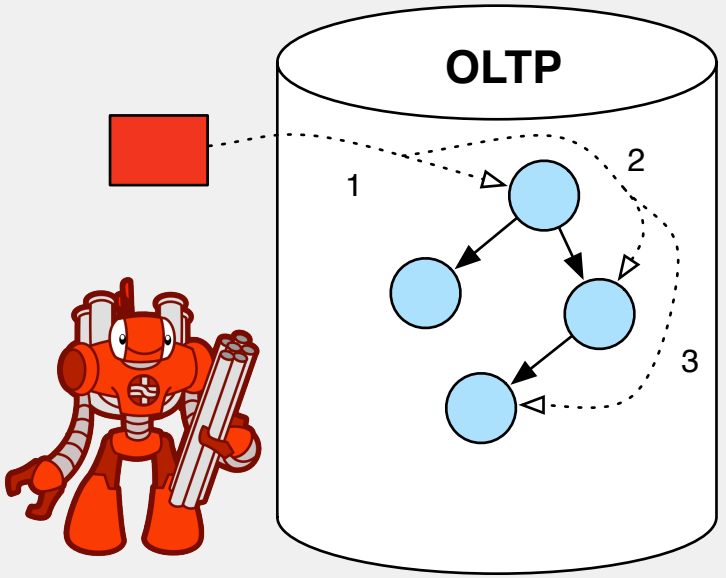


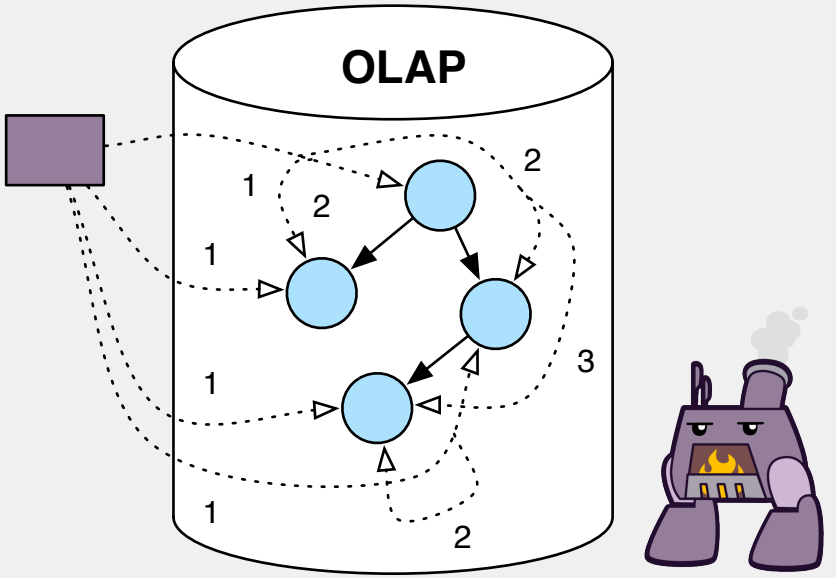


Execute parallel, distributed Gremlin traversals.

Execute vertex-centric message passing algorithms.

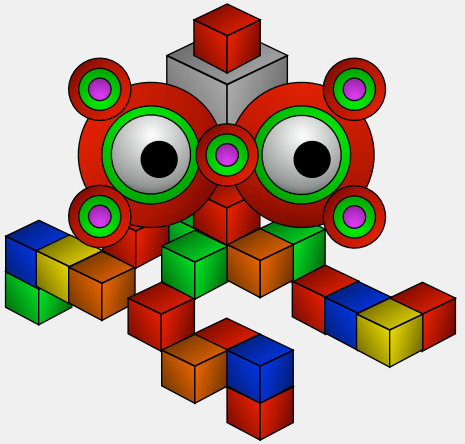
traversers are the messages!

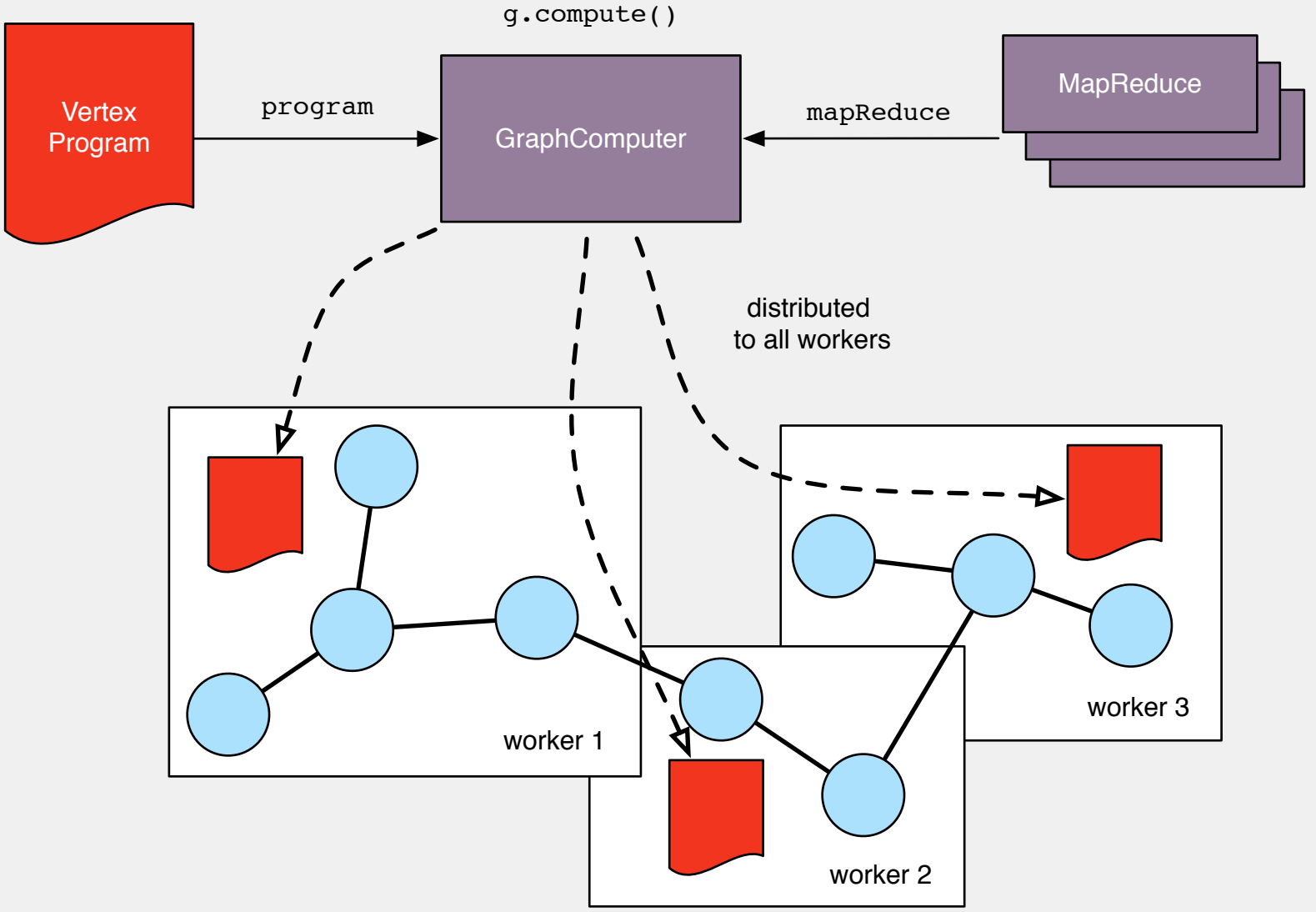




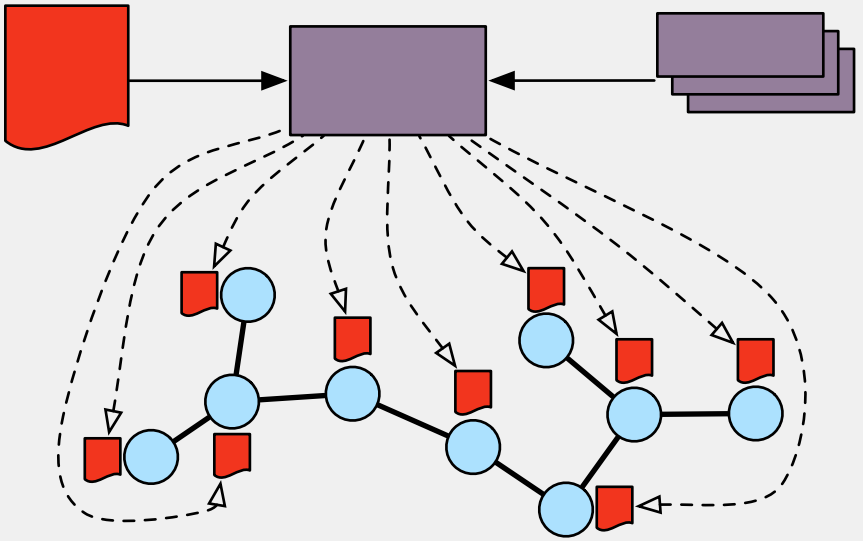
- ✓ Real-time queries.
- ✓ Touching a subset of the graph.
- ✓ Numerous concurrent queries.
- ✓ Good for local graph mutations.

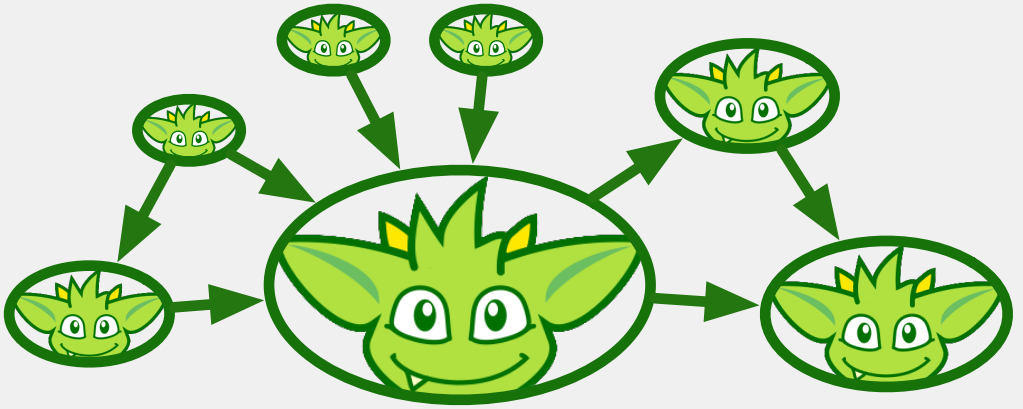
- ✓ Long running queries.
- ✓ Touching the entire of the graph.
- ✓ Single user.
- ✓ Good for bulk loading/mutations.





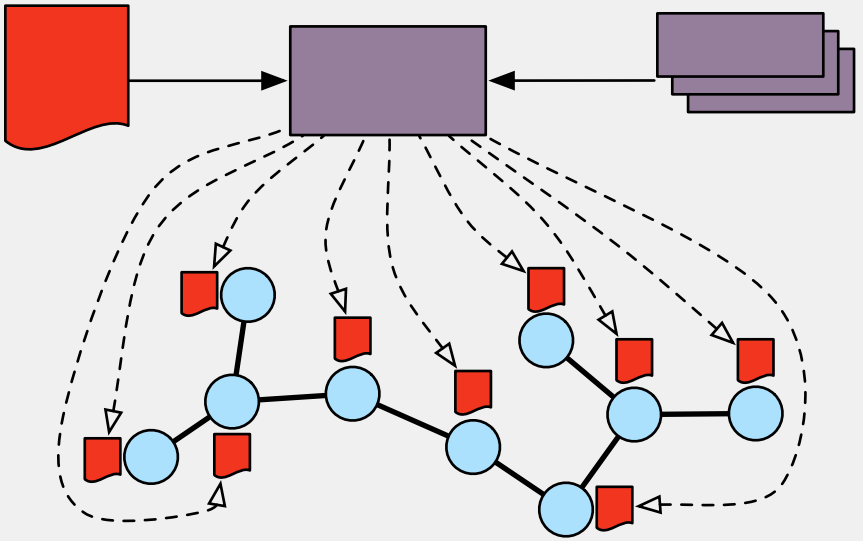
1. Logically distribute the vertex program to all the vertices in the graph.
2. The vertices execute the vertex program sending and receiving messages on each iteration.
3. Any number of MapReduce jobs can follow to aggregate the computed data in the graph.

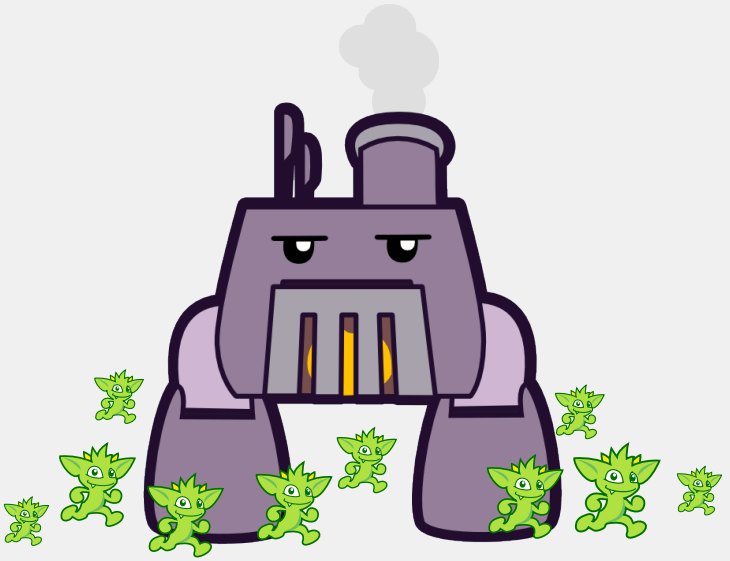




```
gremlin> result = g.compute().program(PageRankVertexProgram).submit()
==>result[tinkergraph[vertices:6 edges:6],memory[size:0]]
gremlin> result.memory.iteration
==>30
gremlin> result.graph
==>tinkergraph[vertices:6 edges:6]
gremlin> result.graph.V.map{ [it.id,it.pageRank]}
==>[1, 0.15000000000000002]
==>[2, 0.19250000000000003]
==>[3, 0.4018125]
==>[4, 0.19250000000000003]
==>[5, 0.23181250000000003]
==>[6, 0.15000000000000002]
```

* Minor tweaks to example to make it fit nicely on a single line.





```
gremlin> g.V.both.both.name.groupCount()
==>[ripple:3, peter:3, vadas:3, josh:7, lop:7, marko:7]
```

OLTP

```
gremlin> g.V.both.both.name.groupCount().submit(g.compute())
==>[ripple:3, peter:3, vadas:3, josh:7, lop:7, marko:7]
```

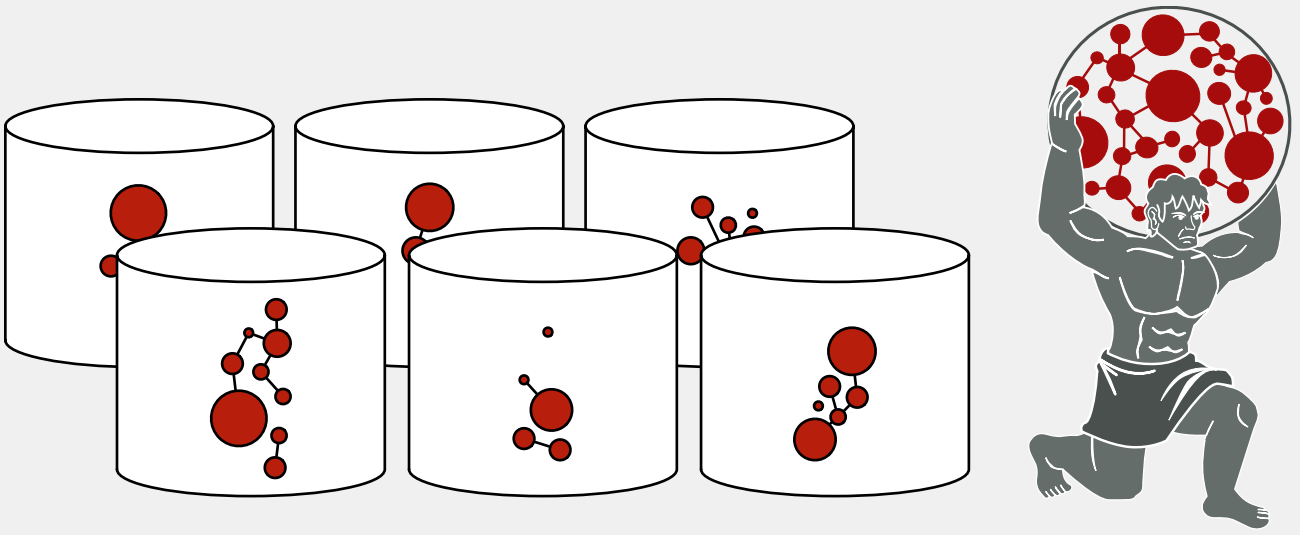
OLAP

MULTIPLE GRAPH COMPUTERS FOR THE SAME GRAPH

```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
gremlin> g.compute(FaunusGraphComputer)
           .program(PageRankVertexProgram)
           .submit()
==>result[faunusgraph,memory[size:0]]
```

```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
gremlin> g.compute(FulgoraGraphComputer)
           .program(PageRankVertexProgram)
           .submit()
==>result[fulgoragraph,memory[size:0]]
```

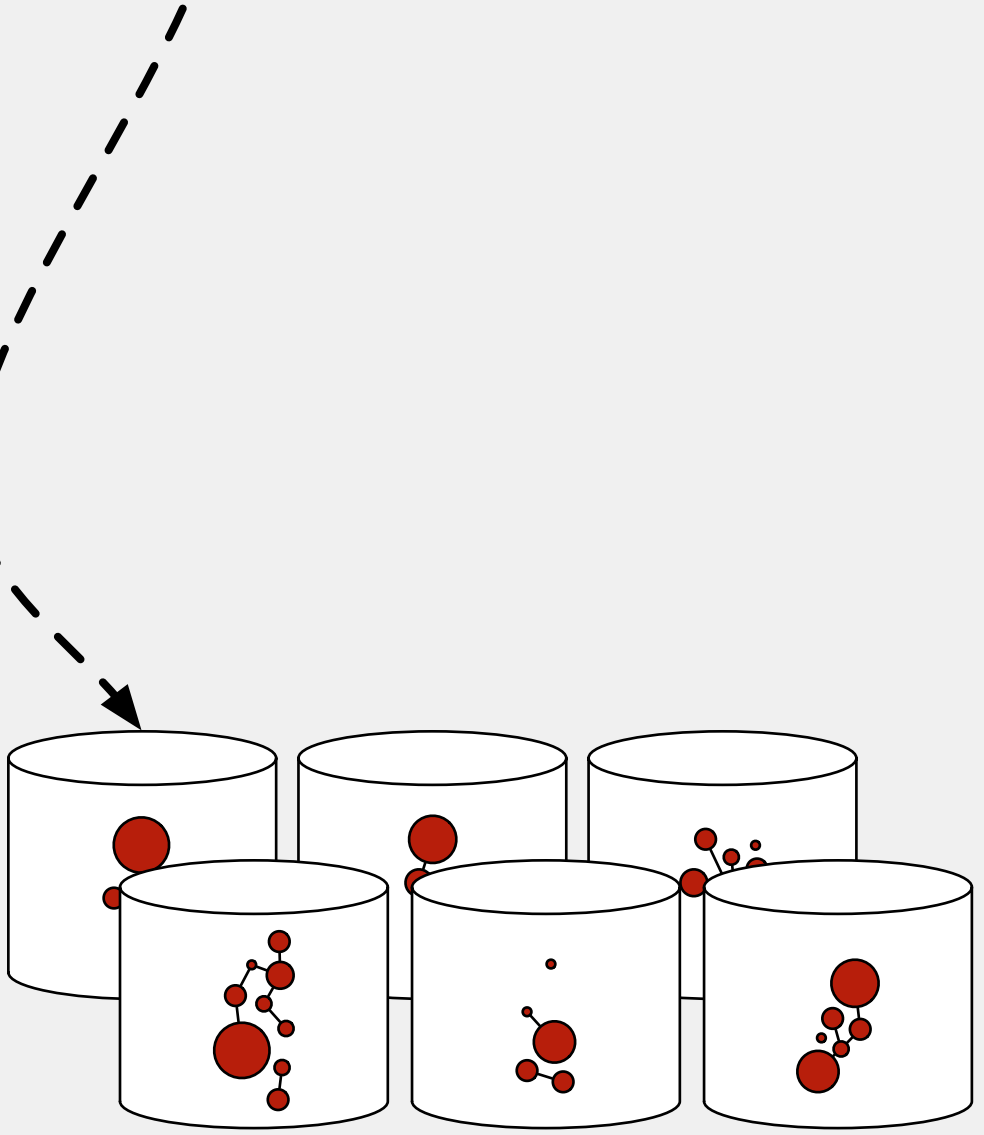
GREMLIN SERVER



GREMLIN SERVER

```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
```

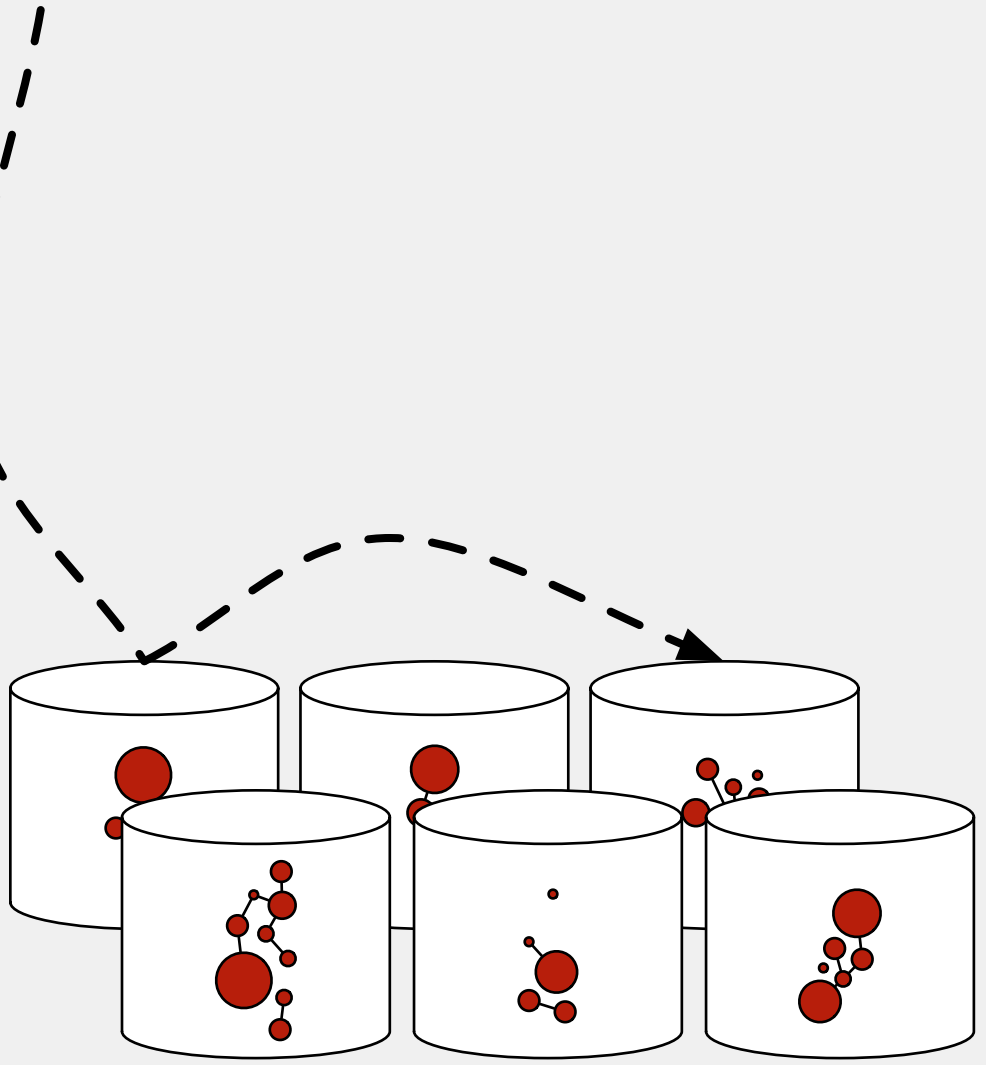
localhost

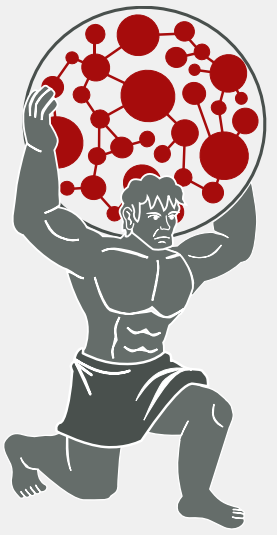


GREMLIN SERVER

```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
gremlin> g.V.has('name','titan')
==>v[2]
```

localhost



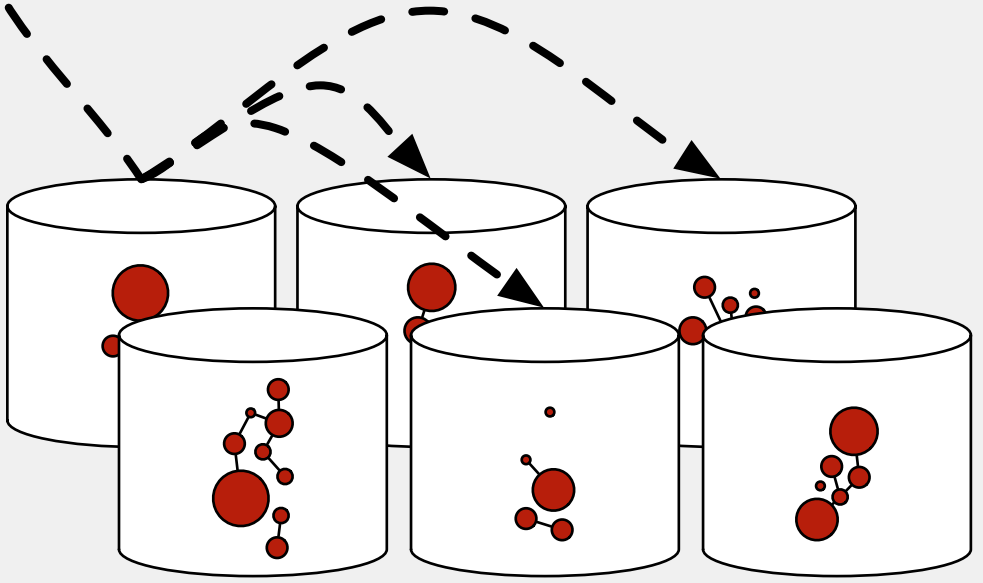


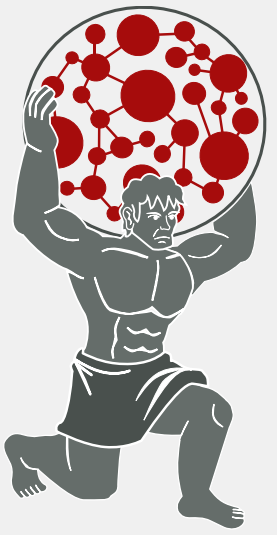
GREMLIN SERVER

```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
gremlin> g.V.has('name','titan')
==>v[2]
gremlin> g.V.has('name','titan').out
```

localhost

1. get the titan vertex
2. get titan's adjacents





GREMLIN SERVER

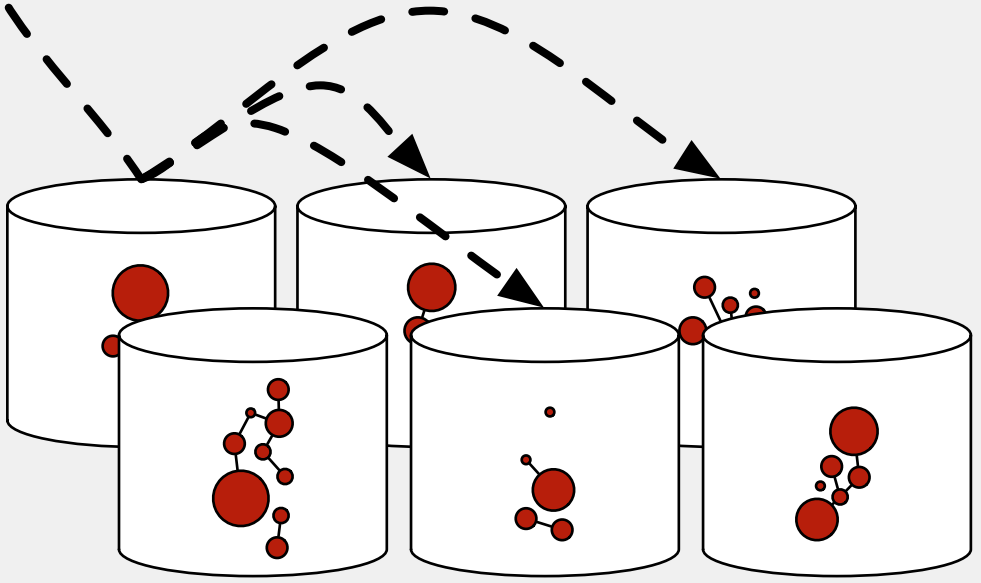
```
gremlin> g = TitanFactory.open(conf)
==>titangraph[66.77.88.99]
gremlin> g.V.has('name','titan')
==>v[2]
gremlin> g.V.has('name','titan').out
```

localhost

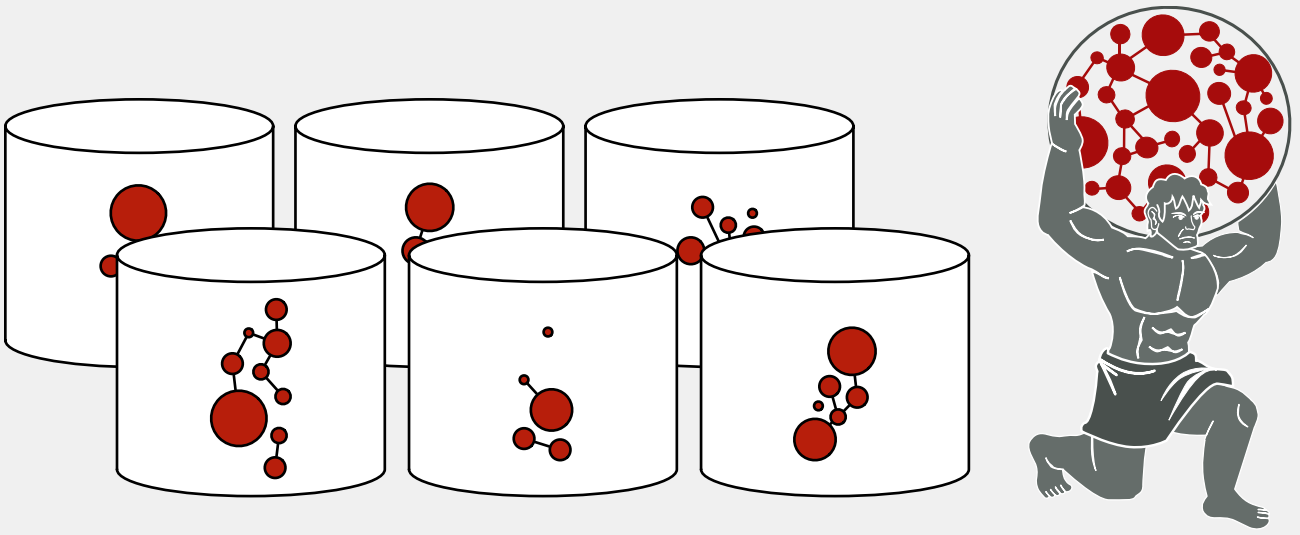
1. get the titan vertex
2. get titan's adjacents

CHATTY

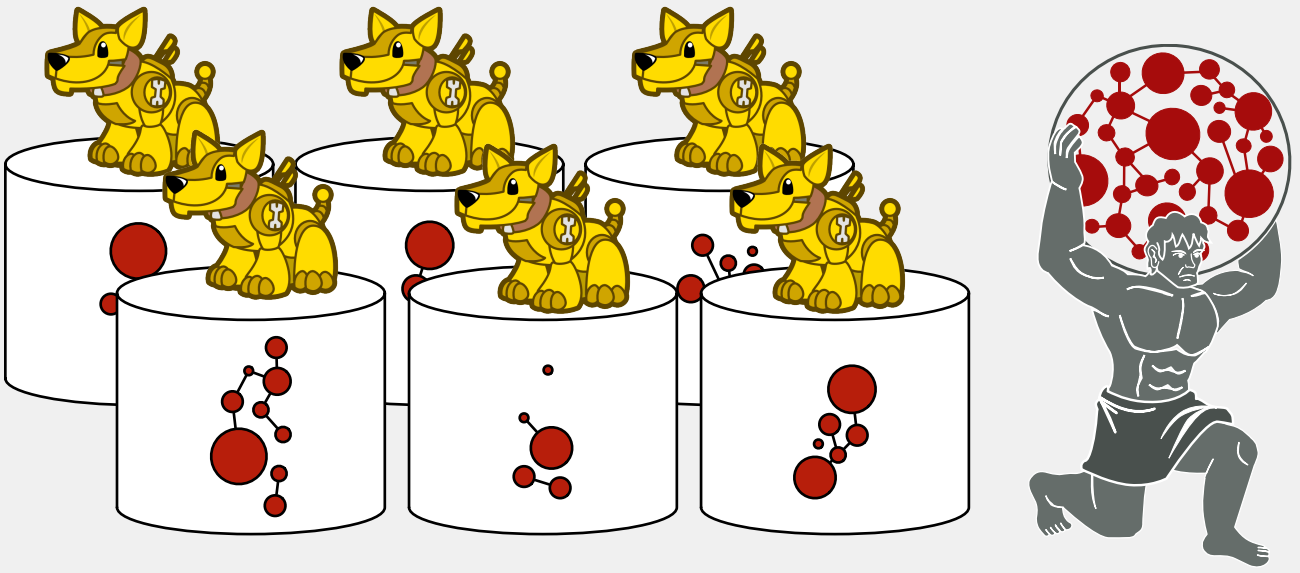
Each step in the traversal is a round trip



GREMLIN SERVER



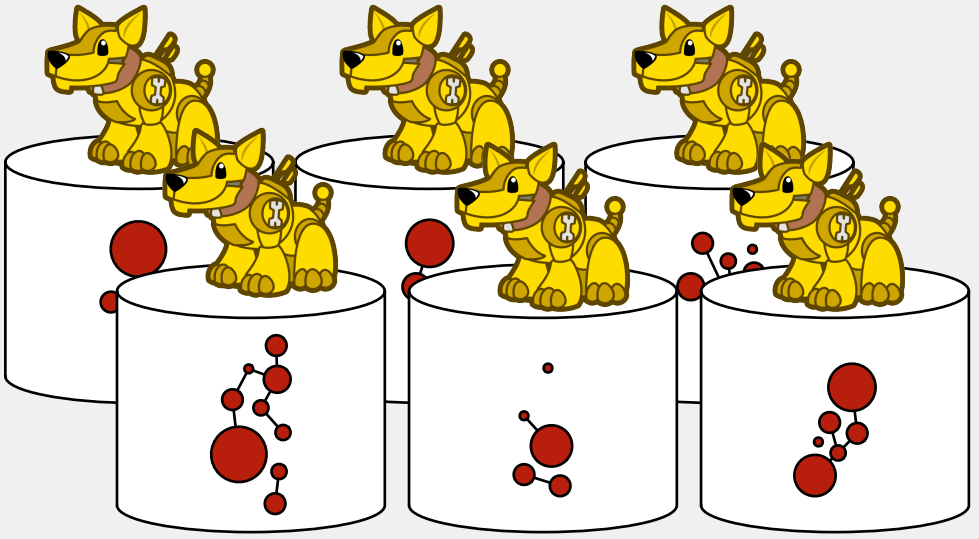
GREMLIN SERVER



GREMLIN SERVER

```
gremlin> :remote connect tinkerpop.server titan.yaml
==>connected - 66.77.88.99:8182,11.22.33.44:8182,...
```

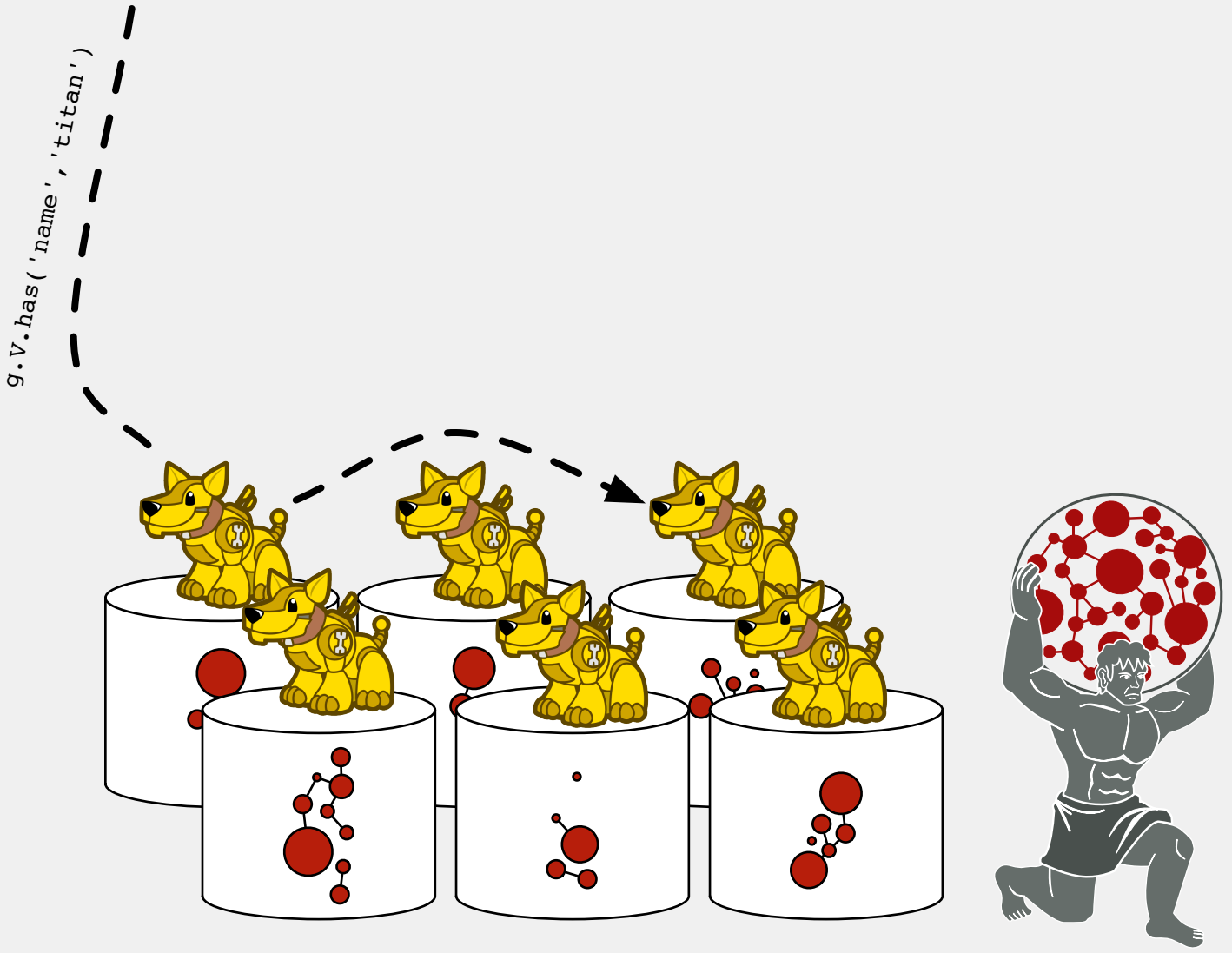
localhost



GREMLIN SERVER

```
gremlin> :remote connect tinkerpop.server titan.yaml
==>connected - 66.77.88.99:8182,11.22.33.44:8182,...
gremlin> :> g.V.has('name','titan')
==>v[2]
```

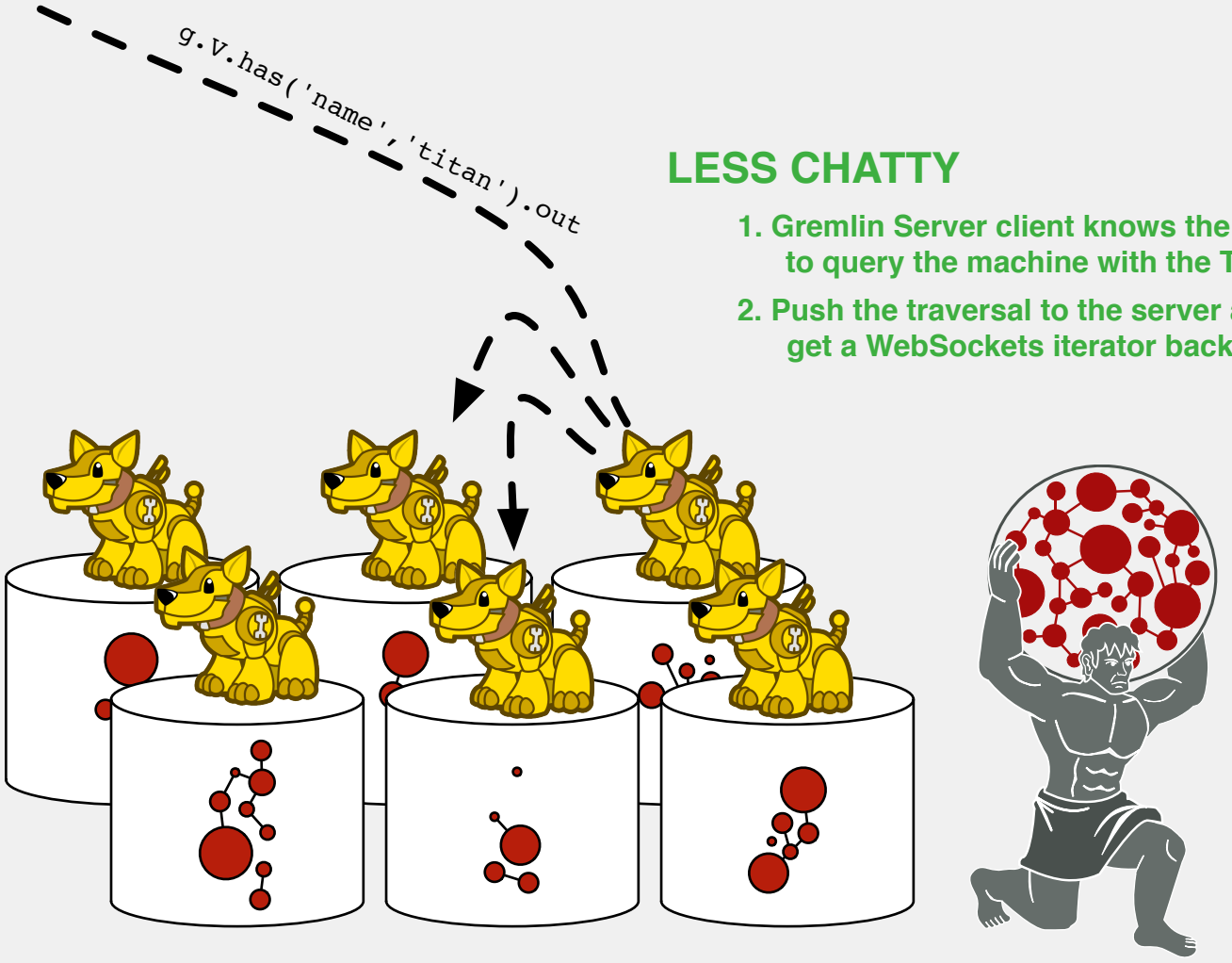
localhost



GREMLIN SERVER

```
gremlin> :remote connect tinkerpop.server titan.yaml
==>connected - 66.77.88.99:8182,11.22.33.44:8182,...
gremlin> :> g.V.has('name','titan')
==>v[2]
gremlin> :> g.V.has('name','titan').out
```

localhost



LESS CHATTY

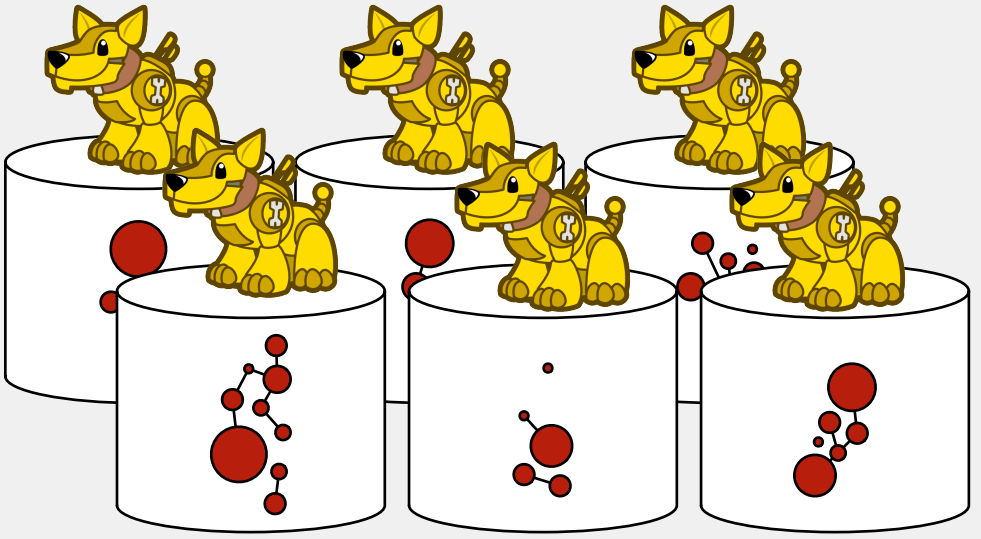
1. Gremlin Server client knows the hash function to query the machine with the Titan vertex
2. Push the traversal to the server and get a WebSockets iterator back

GREMLIN SERVER

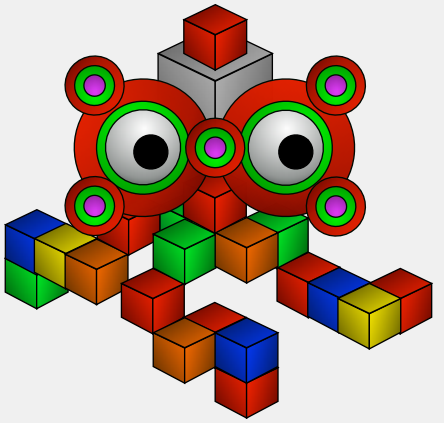
Allows non-JVM languages to interact with a remote graph.

Supports stored procedures -- MyAlgorithms.class

Supports WebSockets, REST, and various serialization mechanisms.



CREDITS



PRESENTED BY
MARKO A. RODRIGUEZ

MANY THANKS TO
MATTHIAS BRÖCHELER
STEPHEN MALLETTE
DAN LAROCQUE
DANIEL KUPPITZ
BOB BRIODY
BRYN COOKE
JOSH SHINAVIER
PAVEL YASKEVICH
AURELIUS COMMUNITY
TINKERPOP COMMUNITY
KETRINA YIM