Large Scale Graph Analytics with

JanusGraph

DataWorks Summit San Jose
June 13, 2017

P. Taylor Goetz, Hortonworks
@ptgoetz

About Me

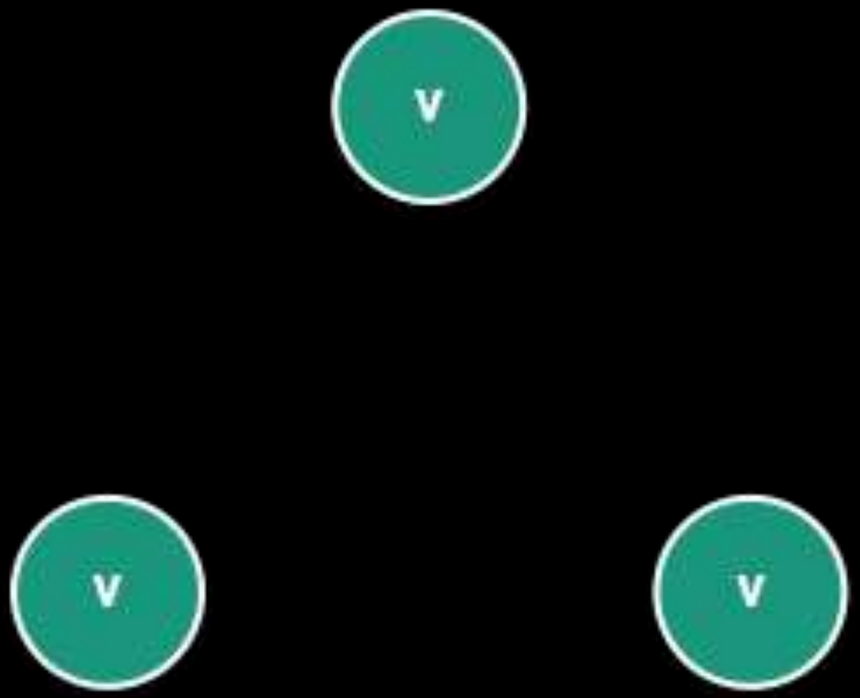
- Tech Staff @ Hortonworks
- TSC Member, JanusGraph
- PMC Chair, Apache Storm
- ASF Member
- PMC: Apache Incubator, Apache Arrow, Apache Kylin, Apache Apex, Apache Eagle, Apache Metron

What is a Graph Database?

“In computing, a graph database is a database that uses ***graph structures*** for semantic queries with ***nodes, edges and properties*** to represent and store data. A key concept of the system is the graph (or edge or relationship), which directly relates data items in the store. ***The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation***.”

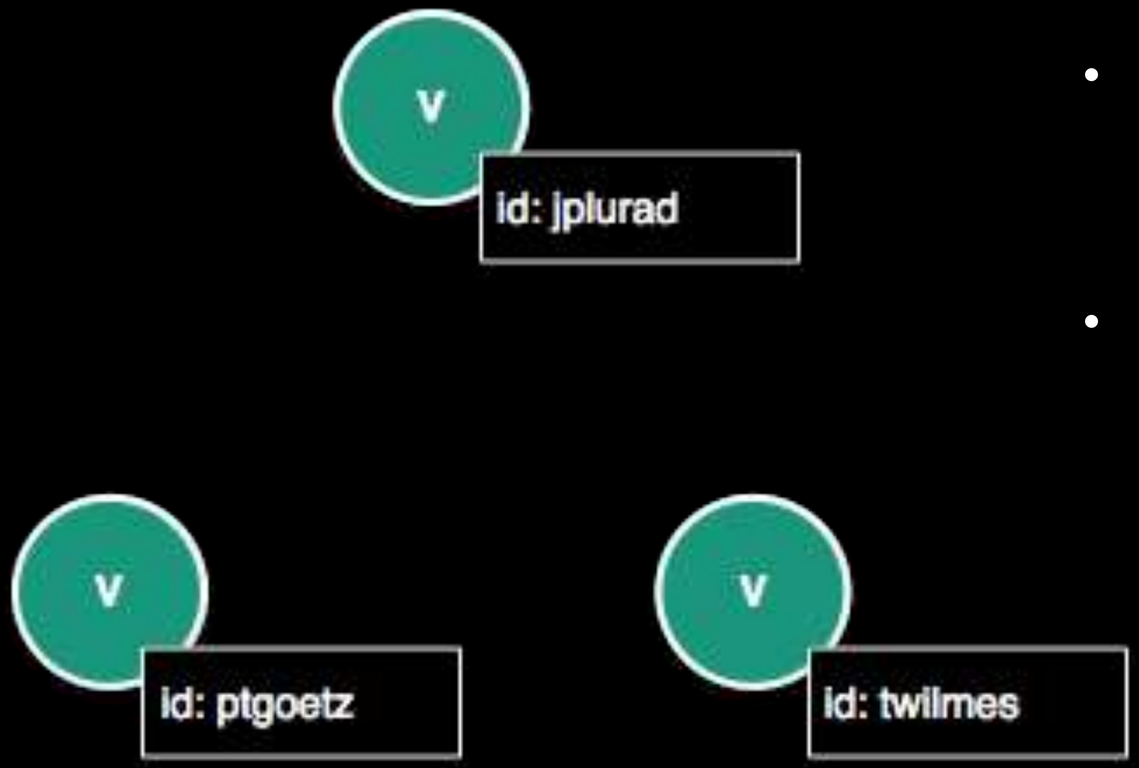
–Wikipedia

Graph Structures - Vertices



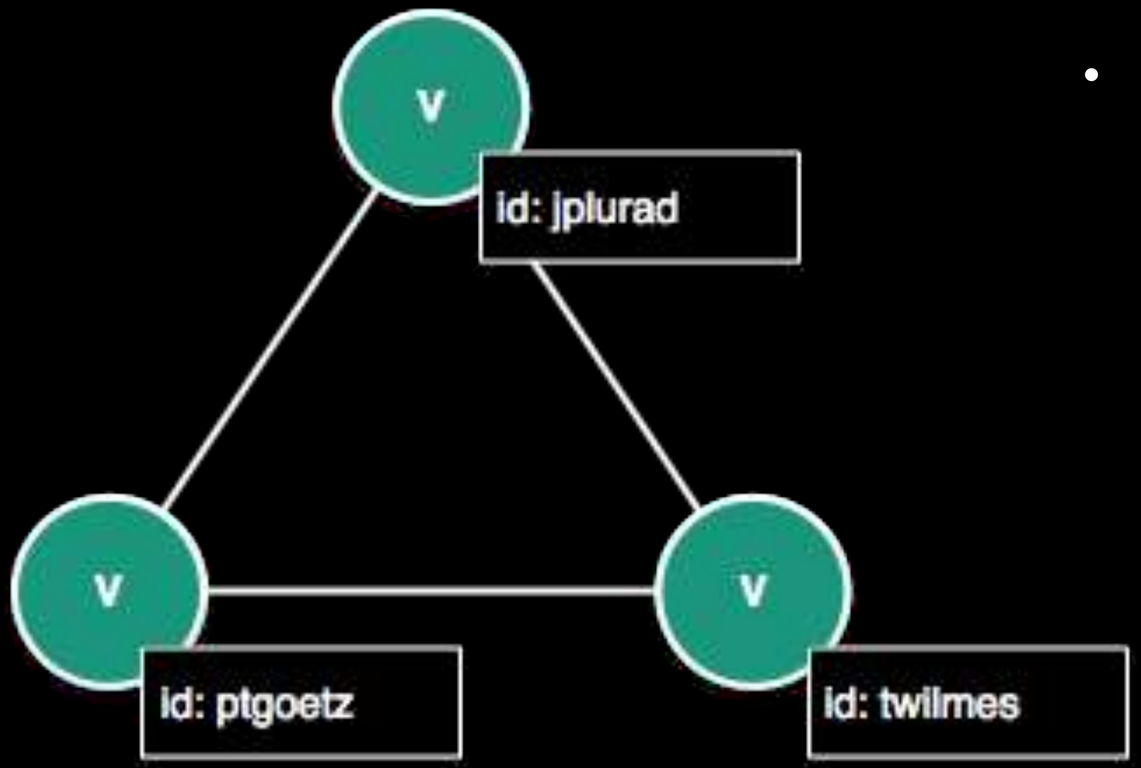
- Vertices are the ***nodes*** or ***points*** in a graph structure

Graph Structures - Vertices



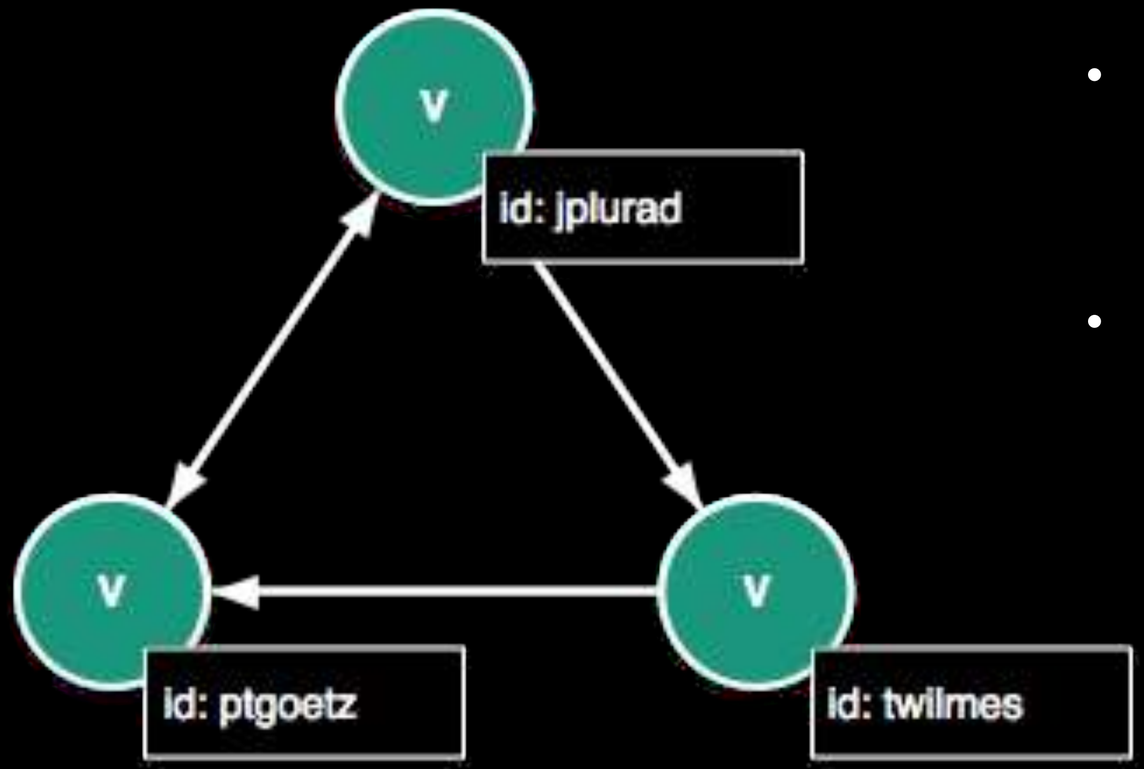
- Vertices are the *nodes* or *points* in a graph structure
- Vertices can be associated with a set of ***properties*** (key-value pairs)

Graph Structures - Edges



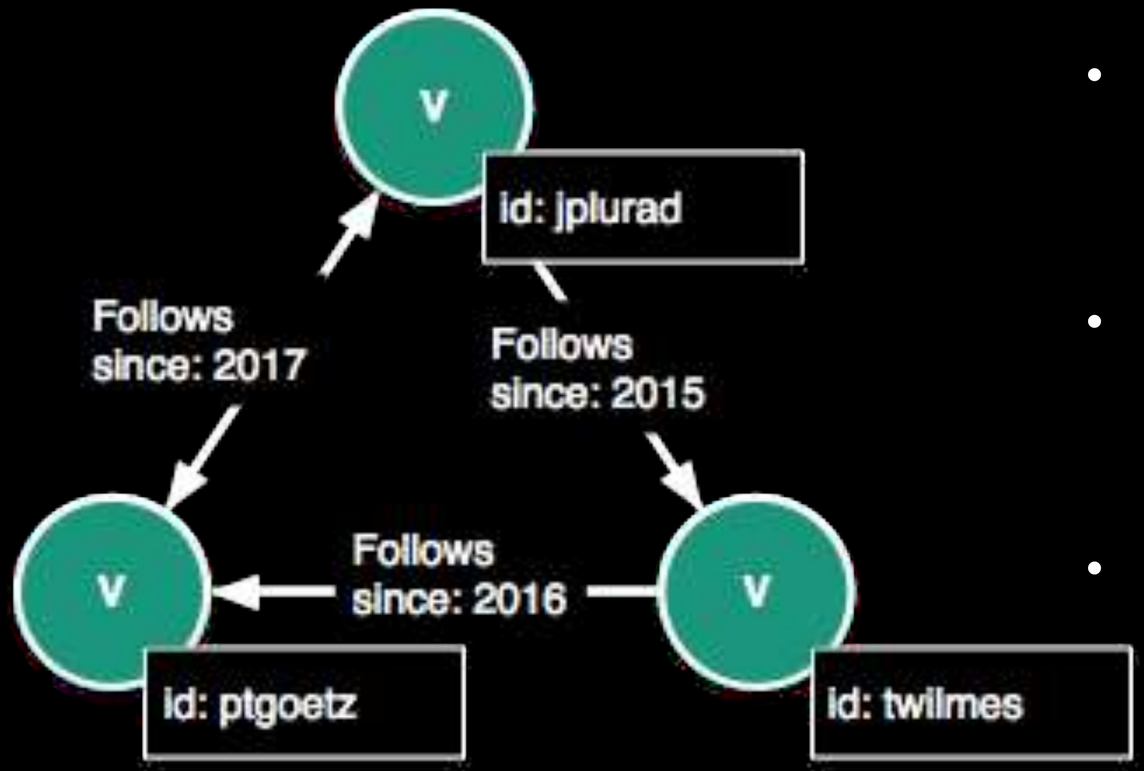
- Edges are the ***connections*** between the vertices in a graph

Graph Structures - Edges



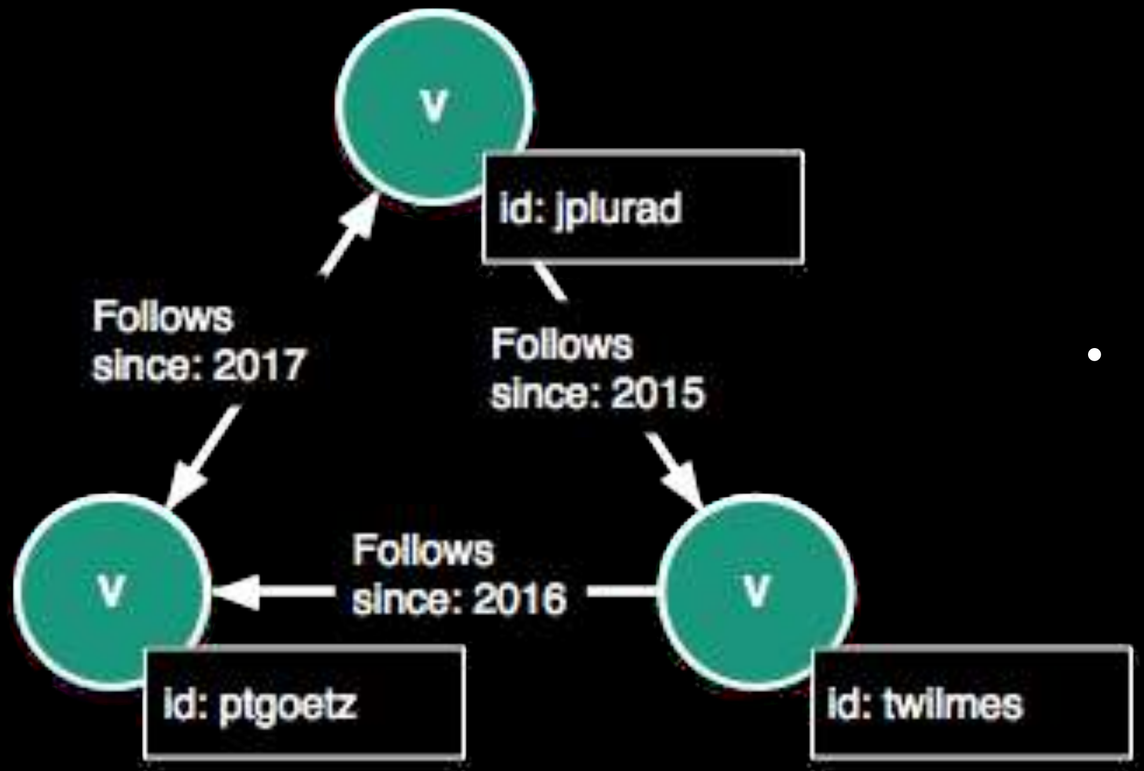
- Edges are the ***connections*** between the vertices in a graph
- Edges can be ***non-directional***, ***directional***, or ***bi-directional***

Graph Structures - Edges



- Edges are the ***connections*** between the vertices in a graph
- Edges can be ***non-directional***, ***directional***, or ***bi-directional***
- Edges can be named and like vertices can have ***properties***

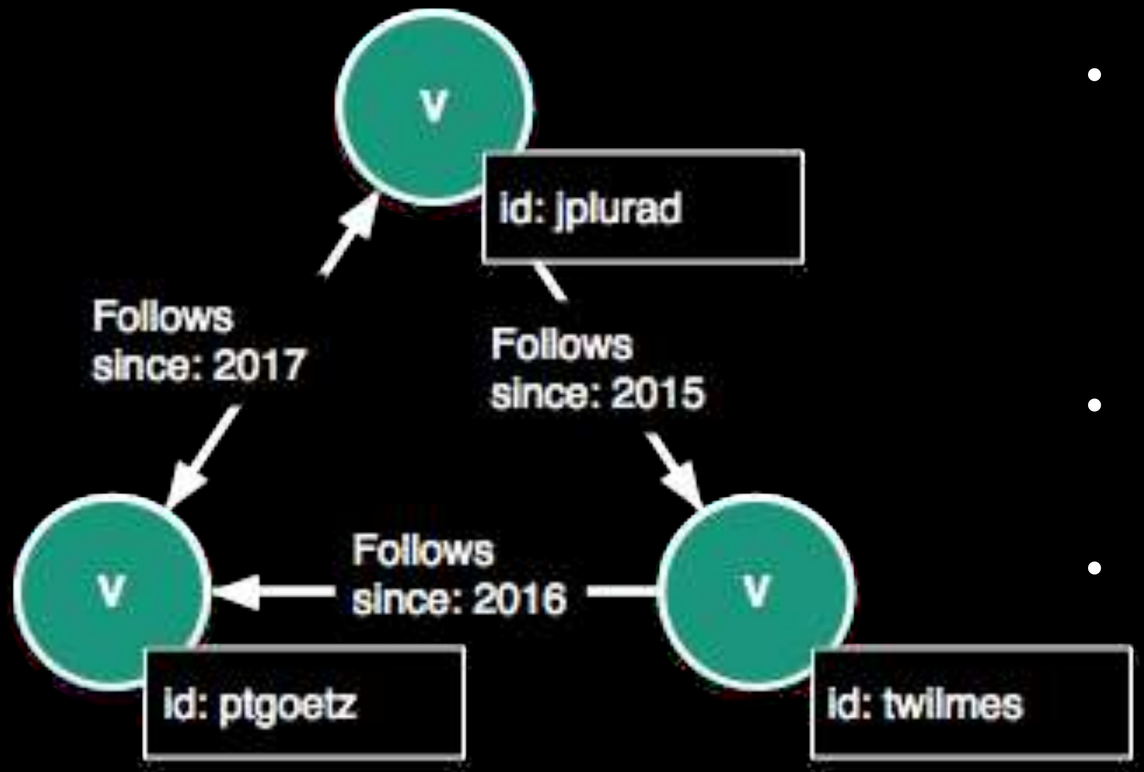
Graph Structures - Graph



G = (V, E)

- The ***graph*** is the collection of ***vertices***, ***edges***, and associated ***properties***

What is a Graph Database?

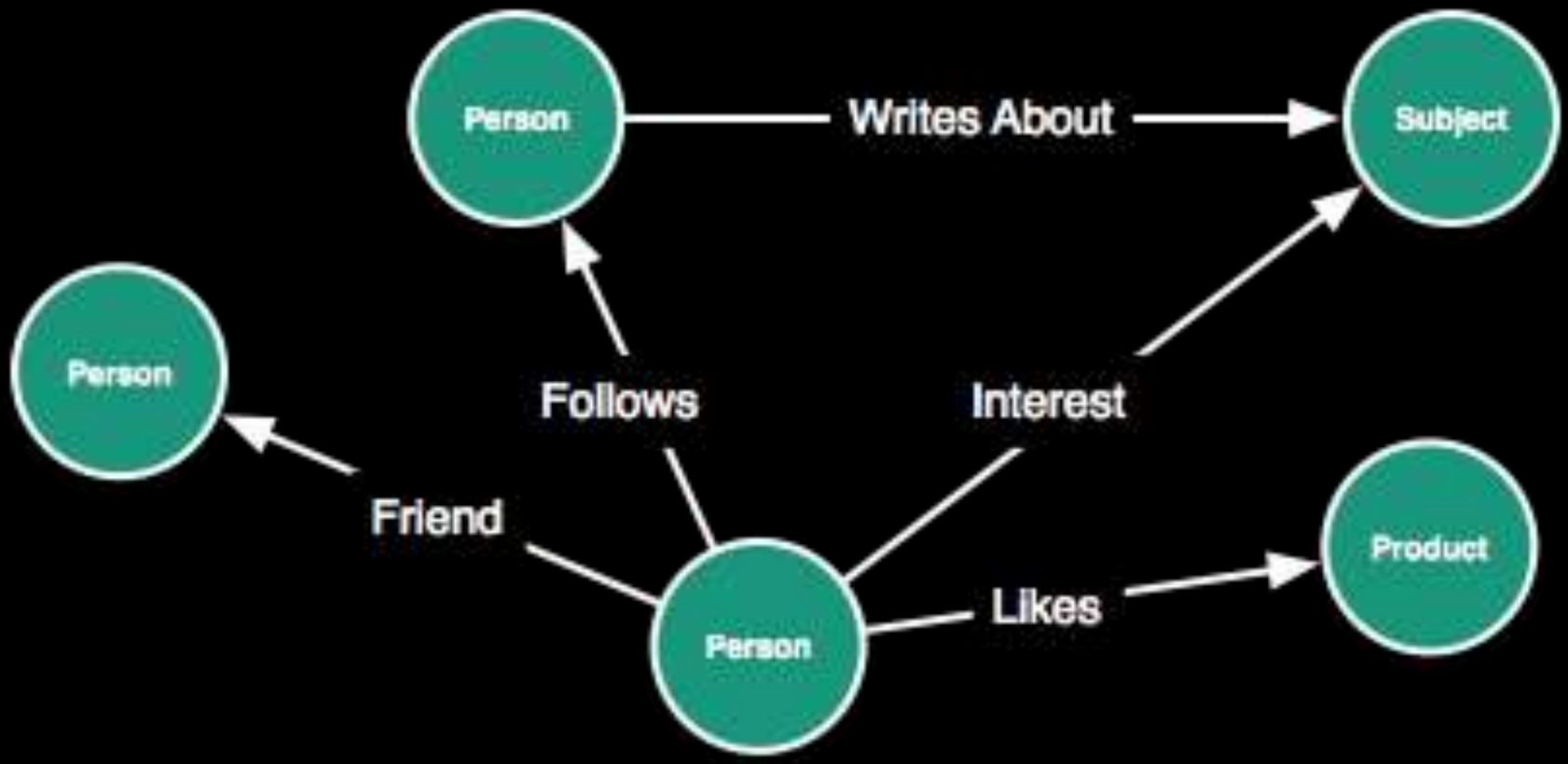


- A ***graph database*** is a datastore optimized for storing and querying graph structures
- Distinct from relational databases
- Focus in terms of storage and queries is on ***relationships***

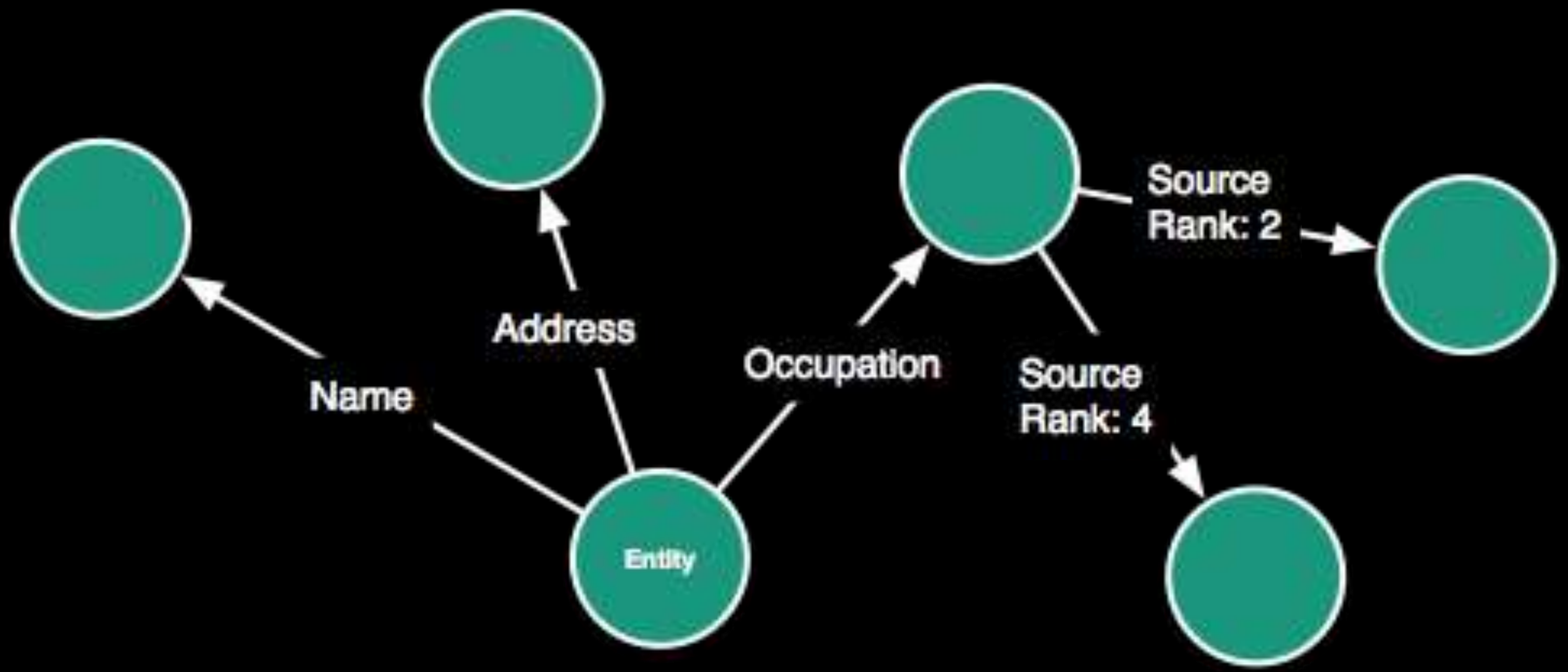
Common Use Cases

Anywhere relationship modeling and analysis can provide insight or value.

Social Media



Master Data Management



Common Use Cases

- Social Networks
- Master Data Management
- Fraud Detection
- Cybersecurity
- Identity and Access Management
- Recommendation Engines

Common Use Cases

- Social Networks
- Master Data Management
- Fraud Detection
- Cybersecurity
- Identity and Access Management
- Recommendation Engines

Many of these can overlap and be combined to provide new insights.

The Power of Relationships

The Power of Relationships

- Harness the value of interconnectedness
- “Paths to Insight”
- Traversal vs. Traditional Query: Join Reduction
- “If you can whiteboard it, you can graph it.”

A little history and the importance of OSS licensing.

JanusGraph

Titan DB

- Large scale graph db developed by Aurelius
- Licensed under ALv2 (this is important)
- Aurelius acquired by DataStax Feb. 2015
- 1.0 released Sept. 19, 2015

GitHub Contributions to Titan

DataStax Aurelius
Acquisition Feb. 2015

GitHub Contributions to Titan

0.9.0-M2
Jun. 9, 2015

DataStax Aurelius
Acquisition Feb. 2015

GitHub Contributions to Titan

0.9.0-M2
Jun. 9, 2015

1.0
Sept. 19, 2015

DataStax Aurelius
Acquisition Feb. 2015

GitHub Contributions to Titan

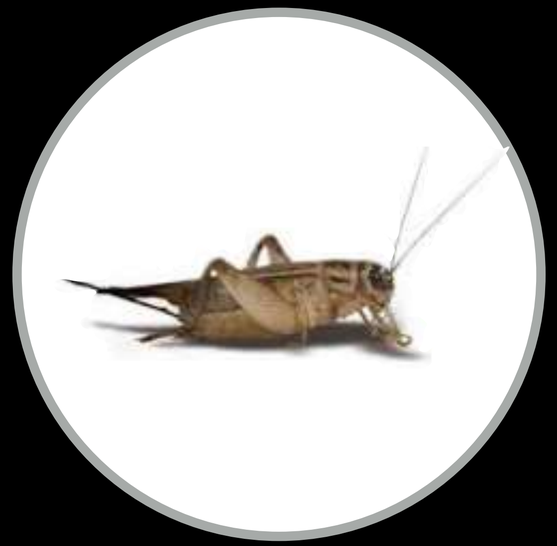
0.9.0-M2
Jun. 9, 2015

1.0
Sept. 19, 2015

DataStax Aurelius
Acquisition Feb. 2015

When will X be fixed?

When will this pull request get merged?



Where does that leave community, users?

Next version?

Security vulnerabilities?

ALv2 to the Rescue!

Empowering Communities

ALv2 to the Rescue!

Empowering Communities

“We can do this. What’s the next step?”

“Apache Olympian?”

What is a “hostile fork?”

A "hostile fork" is a fork of a project that goes against the wishes of the copyright holders and/or community.

“DataStax does not approve of and objects to the proposed forking of Titan into Olympian or any other ASF project.”

–DataStax counsel on Apache Incubator mailing list

“Apache Olympian?”

Next stop…

Introducing…

- Spearheaded by Google, IBM, Hortonworks, Expero, GRAKN.AI
- Contributors from Netflix, Amazon, Uber, Orchestral Developments
- Sponsored by the Linux Foundation

Introducing…

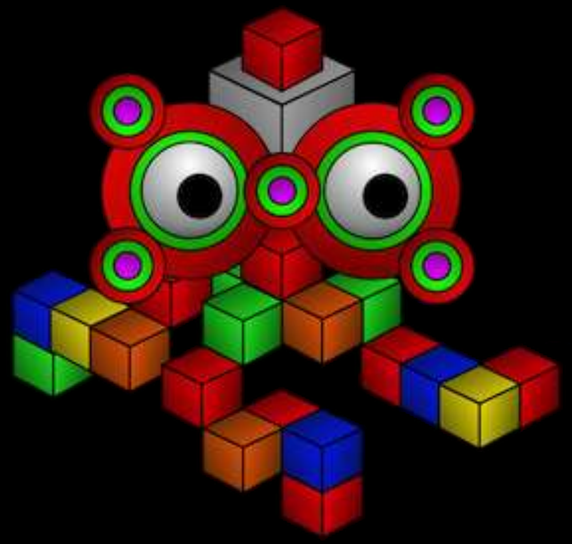
- ALv2 License
- Apache style governance model
- Source code, issues hosted on GitHub
- Mailing lists on Google Groups
- Chat on Gitter

Technical Dive

JanusGraph

- Optimized for storing/querying ***billions of vertices and edges***
- Supports ***thousands of concurrent users***
- Can execute ***local queries*** (OLTP) or cross-cluster ***distributed queries*** (OLAP)

Apache Tinkerpop

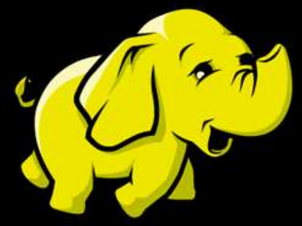


- THE framework and API for graph manipulation and traversal
- Open source, vendor agnostic
- Supported by a number of Graph DBs
- Promotes portability

Gremlin Query Language

- DSL for graph traversal and manipulation
- Fluent style API
- Multi-language support (Java, Scala, Groovy, Python, Ruby, etc.)

OLAP Integration

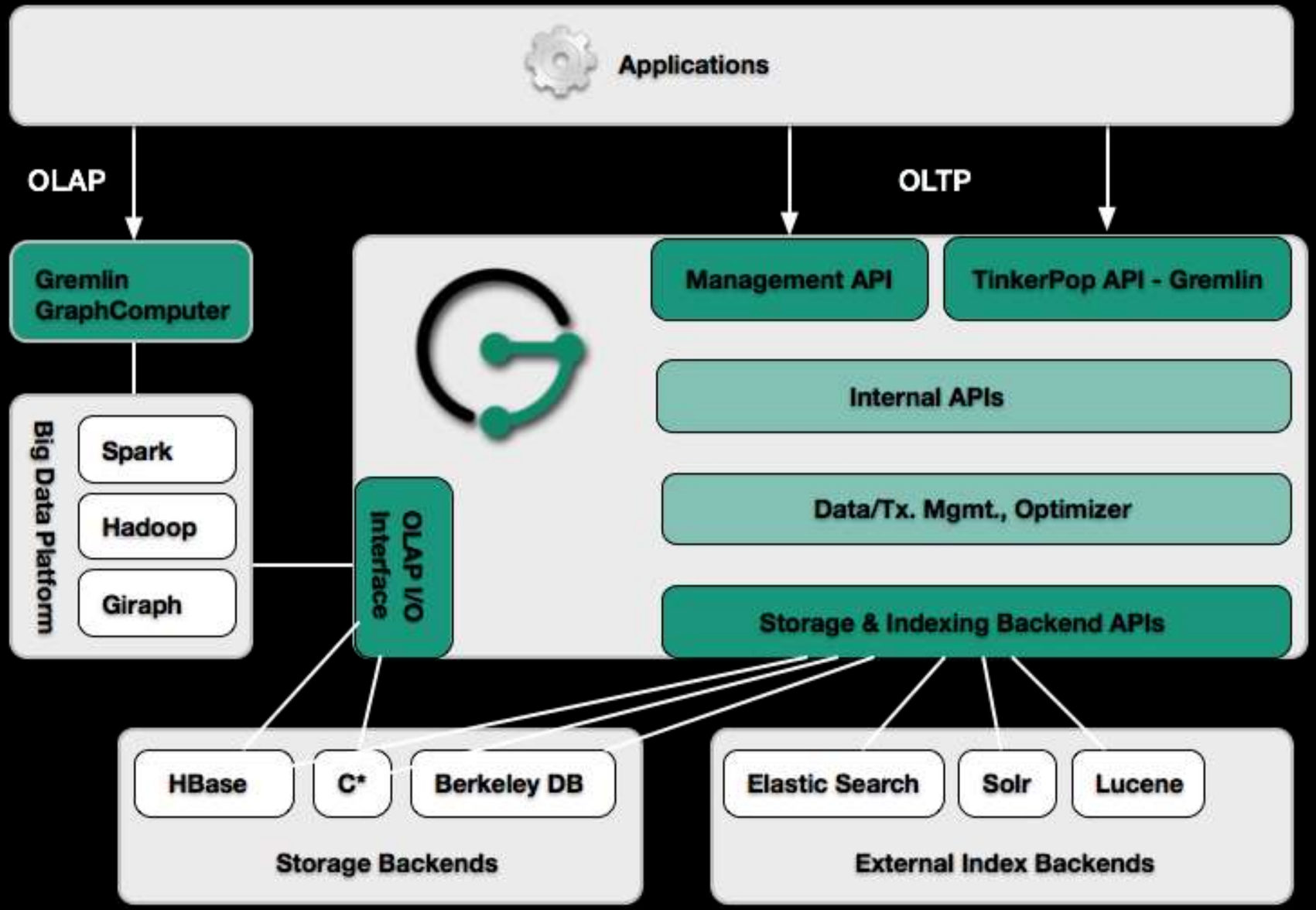


- Apache Hadoop
- Apache Spark
- Apache Giraph

JanusGraph

- ACID compliant (depending on backend)
- Supports very many concurrent transactions
- Embedded, Single Node, or Scale out

JanusGraph Architectural Overview



Storage Backends

- Well defined storage API allows for easily pluggable implementations
- Choose the backend best for your use case and architecture
- Options include: Apache HBase, Apache Cassandra, Google Cloud Bigtable, Berkeley DB
- More on the way…

Choose Your Own [CAP] Adventure

Consistency

Apache
HBase

Google Cloud
Bigtable

Berkeley DB

Partition Tolerance

Availability

Apache
Cassandra

Scylla DB

JanusGraph External Indices

- Secondary to primary graph storage
- Provide a means to speed up graph traversal and information retrieval
- Two types:
  - Graph Index
  - Vertex-centric Index

Graph Indices

- Global index structures across entire graph
- Efficient retrieval of vertices and edges based on associated ***properties***
- Eliminates need to do a full graph scan
- When querying, JanusGraph will typically warn when a full scan is necessary
- New indexes take effect immediately, but reindexing may be required

Vertex-Centric Indexes

- Local index structures built per-vertex
- Eliminates the need to load all vertices from the graph for filtering

Pluggable Index Backends

- Elastic Search
- Apache Solr
- Apache Lucene

Schema and Data Modeling

- Consist of edge labels, property keys, vertex labels
- Explicit or Implicit
- Can evolve over time w/out database downtime
- Edge label multiplicity, Property keys, Key cardinality, Vertex labels

Schema - Edge Label Multiplicity

- **MULTI**: Multiple edges of the same label between vertices
- **SIMPLE**: One edge with that label (unique per label)
- **MANY2ONE**: One *outgoing* edge with that label (mother/children)
- **ONE2MANY**: One *incoming* edge with that label
- **ONE2ONE**: One incoming, one outgoing edge with that label

Schema - Property Key Data Types

Name	Description
String	Character sequence
Character	Individual character
Boolean	true or false
Byte	byte value
Short	short value
Integer	integer value
Long	long value
Float	4 byte floating point number
Double	8 byte floating point number
Decimal	Number with 3 decimal digits
Precision	Number with 6 decimal digits
Date	Date
Geoshape	Geographic shape like point, circle or box
UUID	UUID

Schema - Property Key Cardinality

- **SINGLE:** At most one value per element.
- **LIST:** Arbitrary number of values per element. Allows duplicates.
- **SET:** Multiple values, but no duplicates.

Graph Traversal with Gremlin

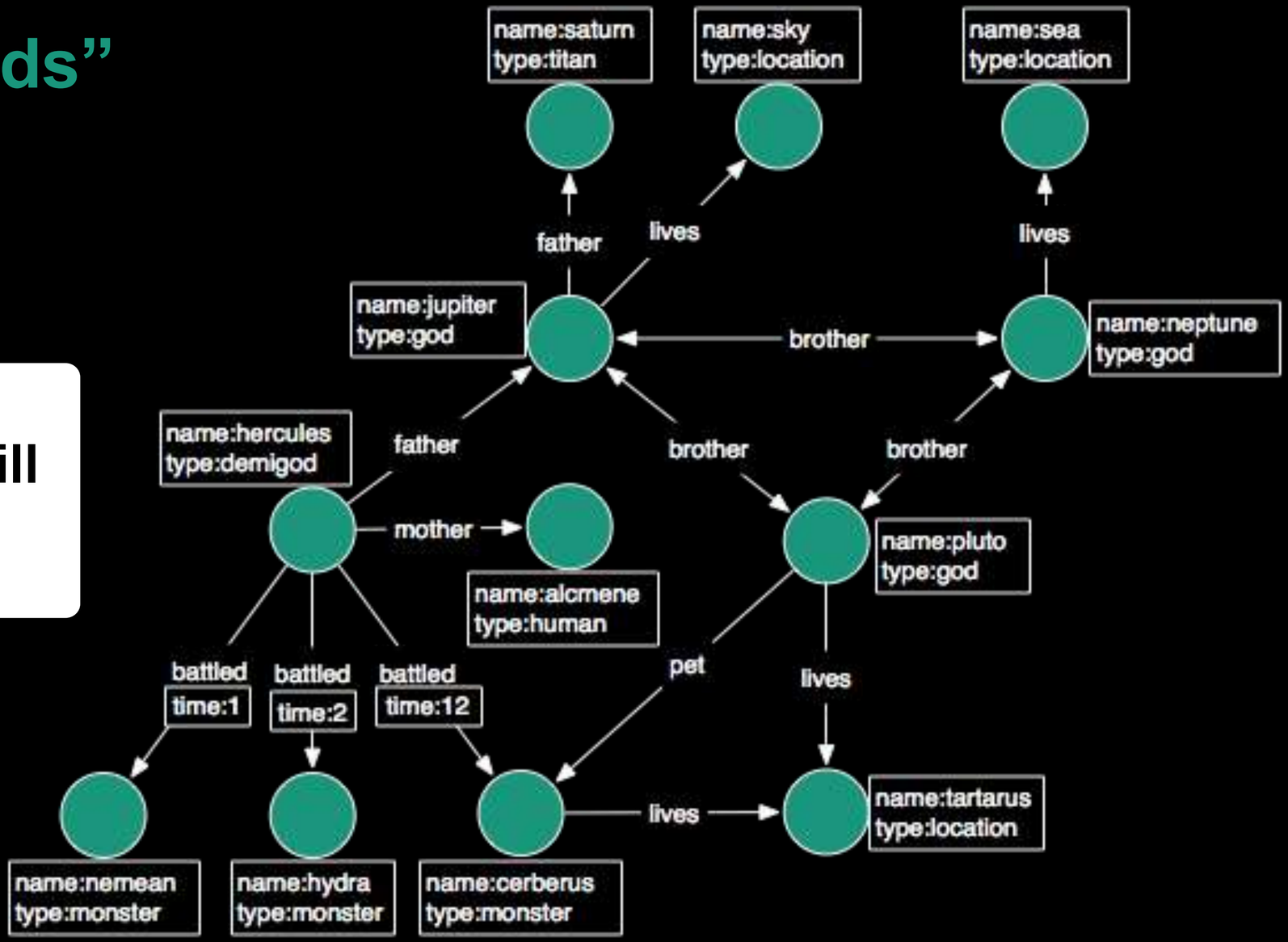
- Gremlin console:
  - Groovy-based REPL for exploring the graph
- Pre-defined convenience variables, expandable by plugins. E.g.:
  - “g” — represents the entire graph
  - “hdfs” — access to hdfs provided by the TinkerPop Hadoop plugin
- Local or remote

Graph Traversal with Gremlin

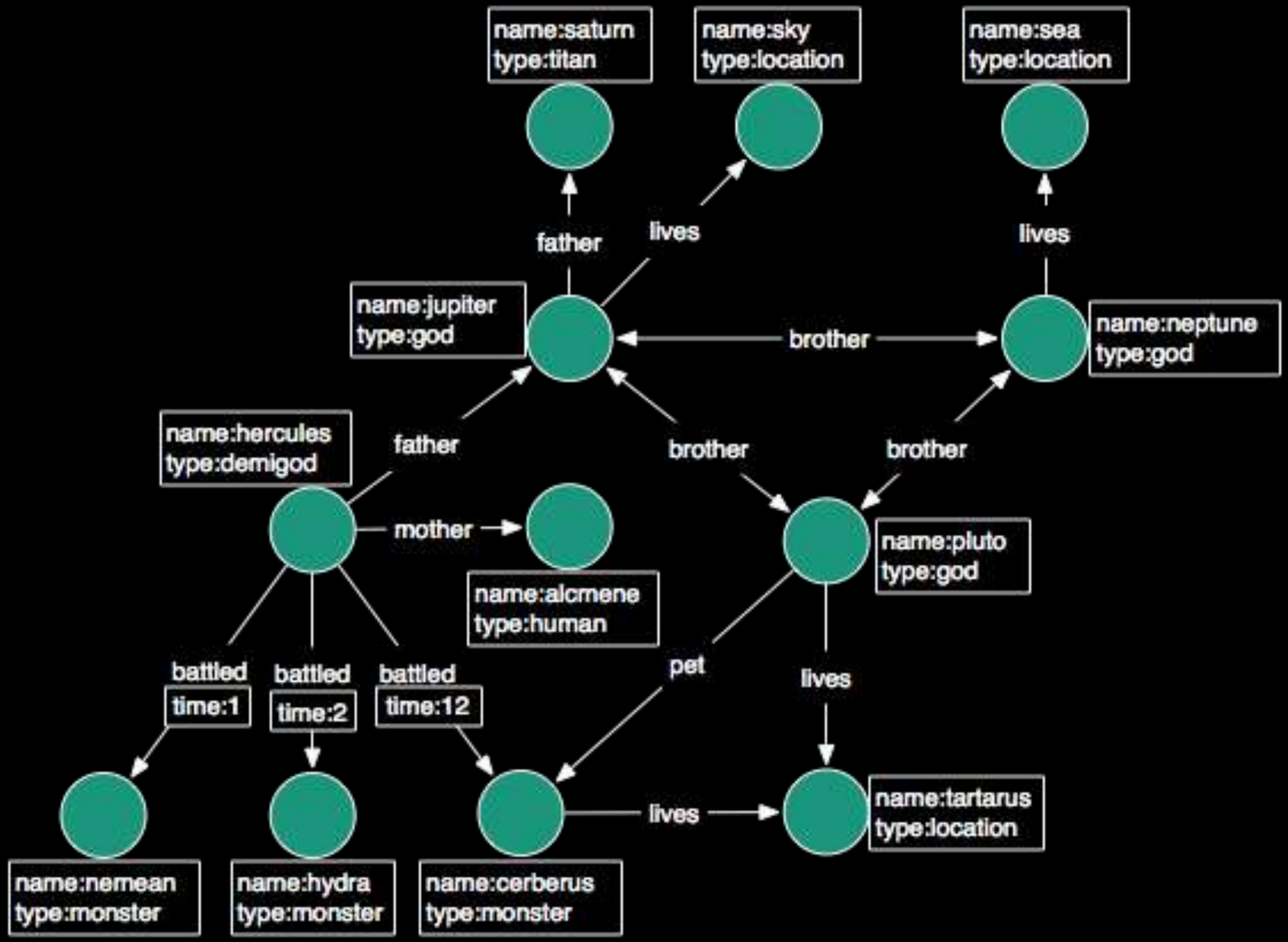
```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
09:12:24 INFO  org.apache.tinkerpop.gremlin.hadoop.structure.H
plugin activated: tinkerpop.hadoop
plugin activated: janusgraph.imports
gremlin>
```

“Graph of the Gods”

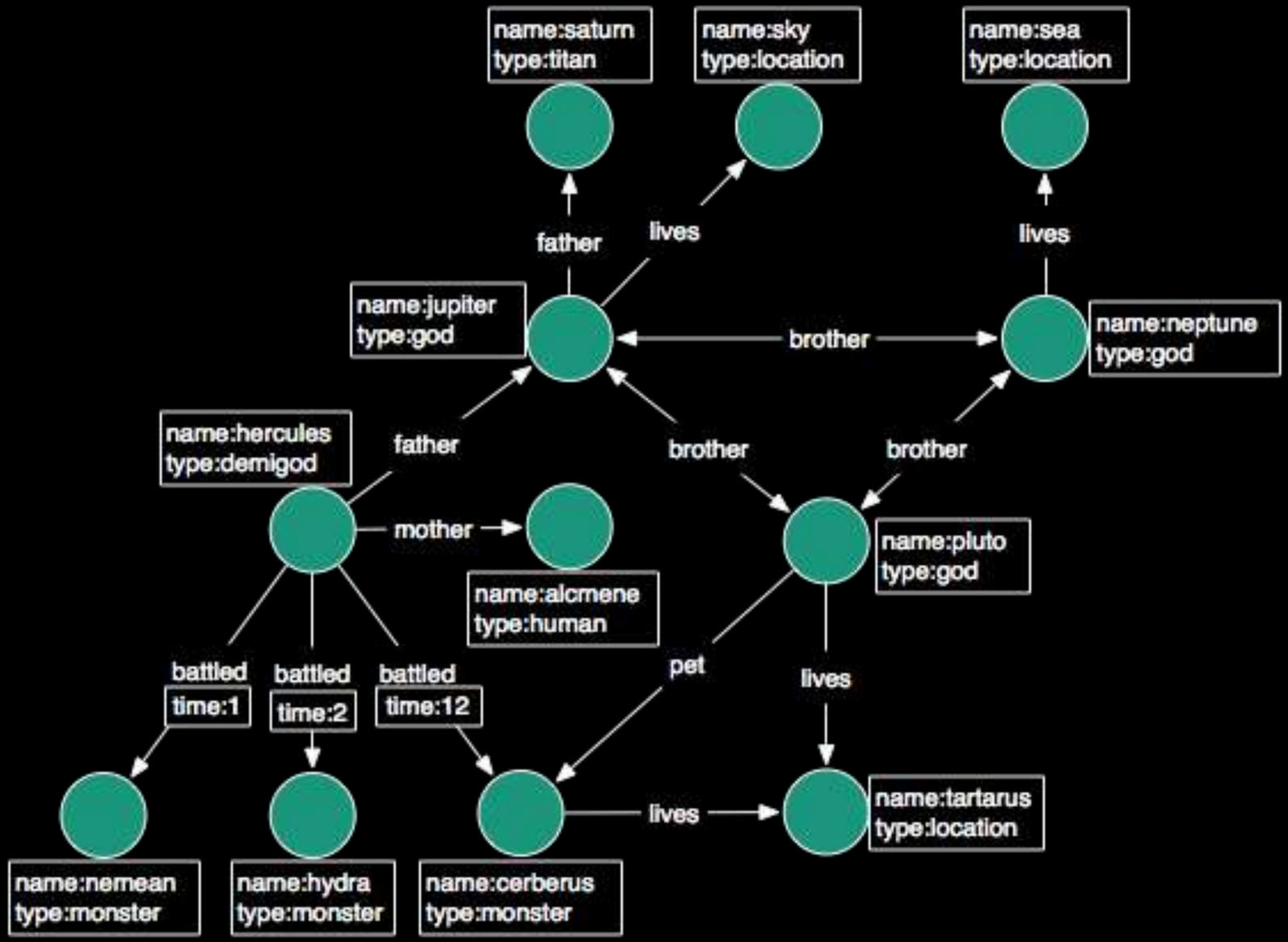
What path will we be taking today?



Who is Hercules' grandfather?



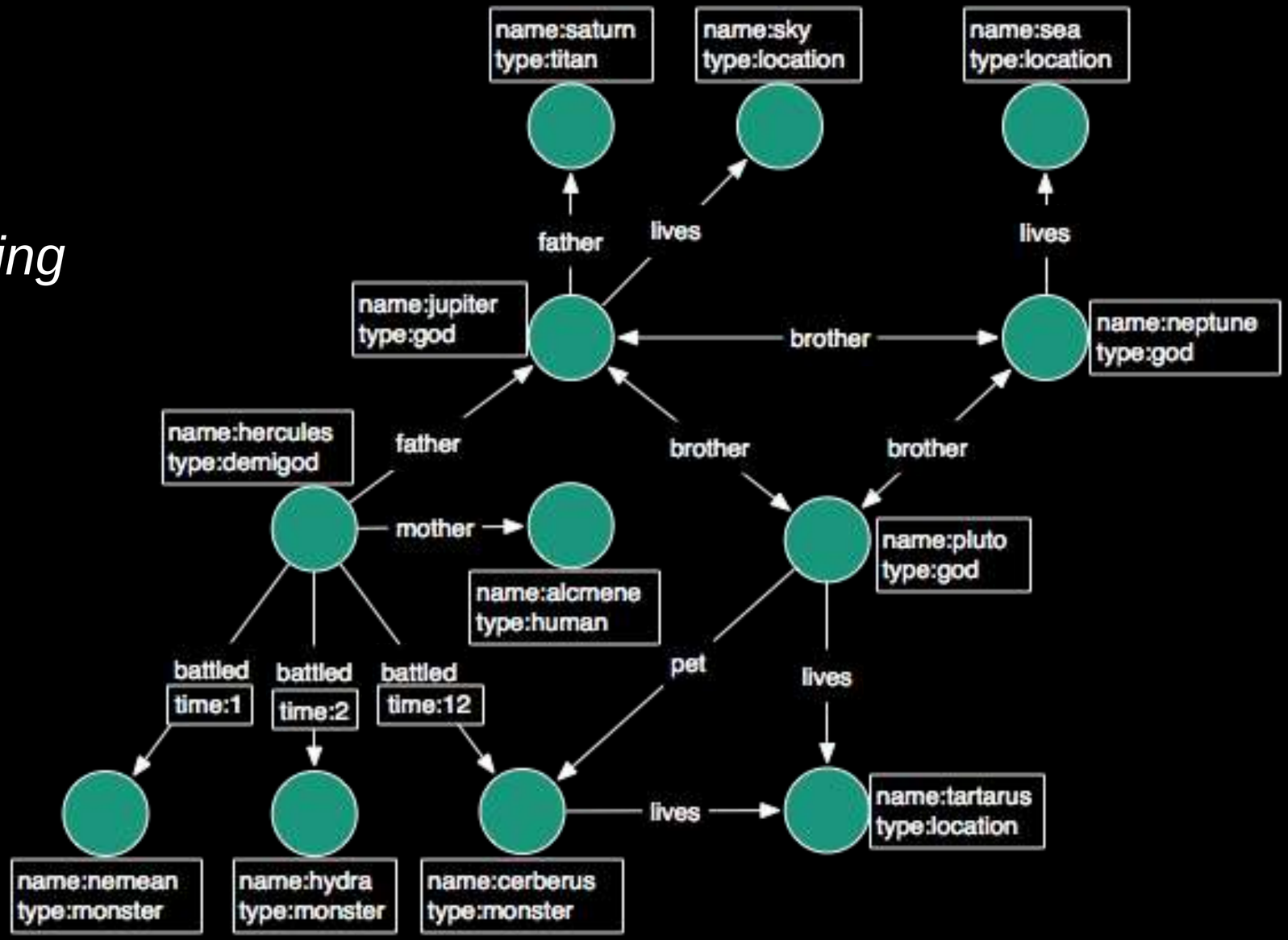
gremlin>


name:saturn
type:titan
name:sky
type:location
name:sea
type:location
father
lives
lives
name:jupiter
type:god
brother
name:neptune
type:god
name:hercules
type:demigod
father
brother
brother
mother
name:alcmene
type:human
name:pluto
type:god
battled
time:1
battled
time:2
battled
time:12
pet
lives
lives
name:tartarus
type:location
name:nemean
type:monster
name:hydra
type:monster
name:cerberus
type:monster

```
gremlin>
```

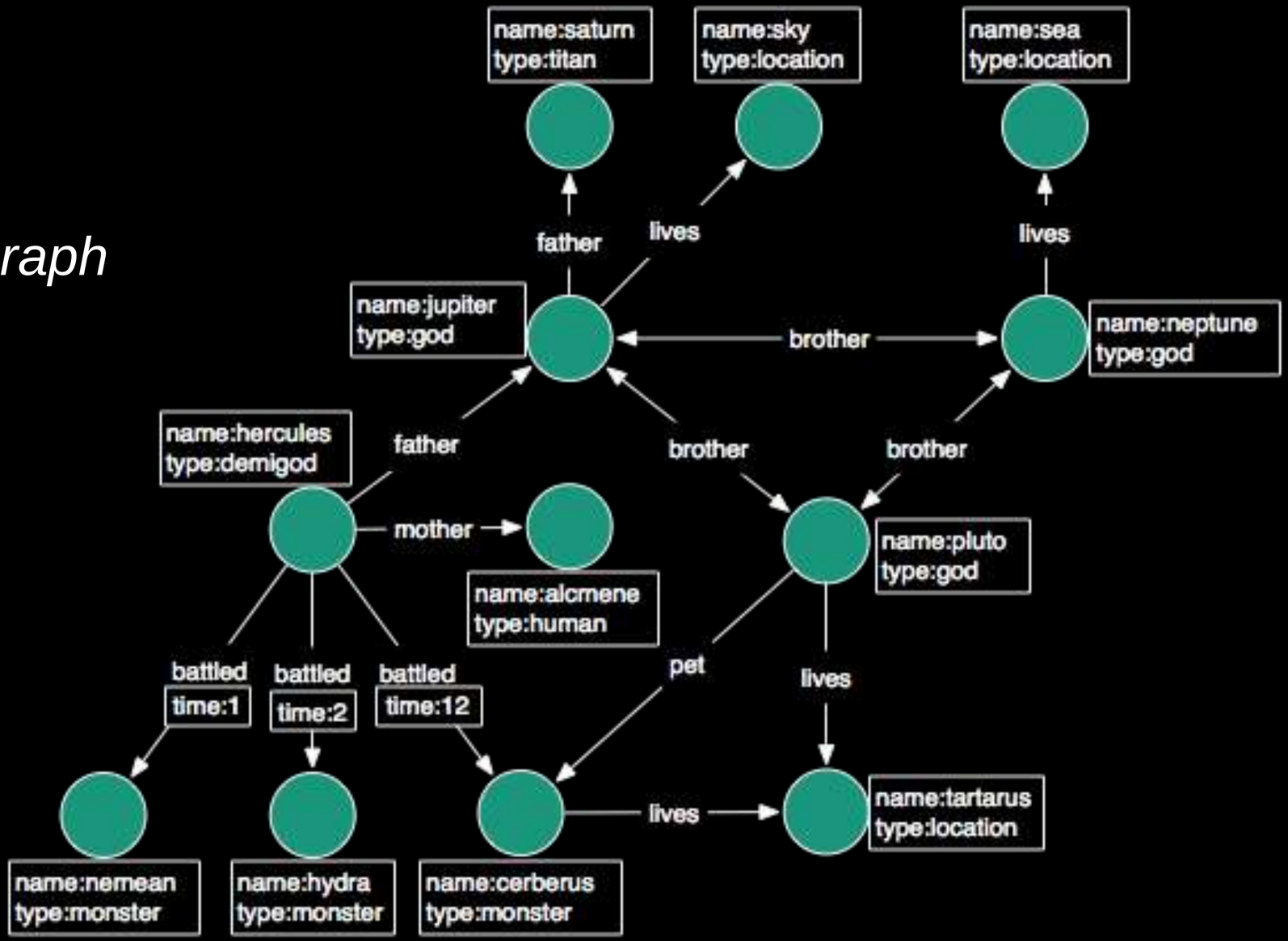
```
g
```

Global variable representing the entire graph



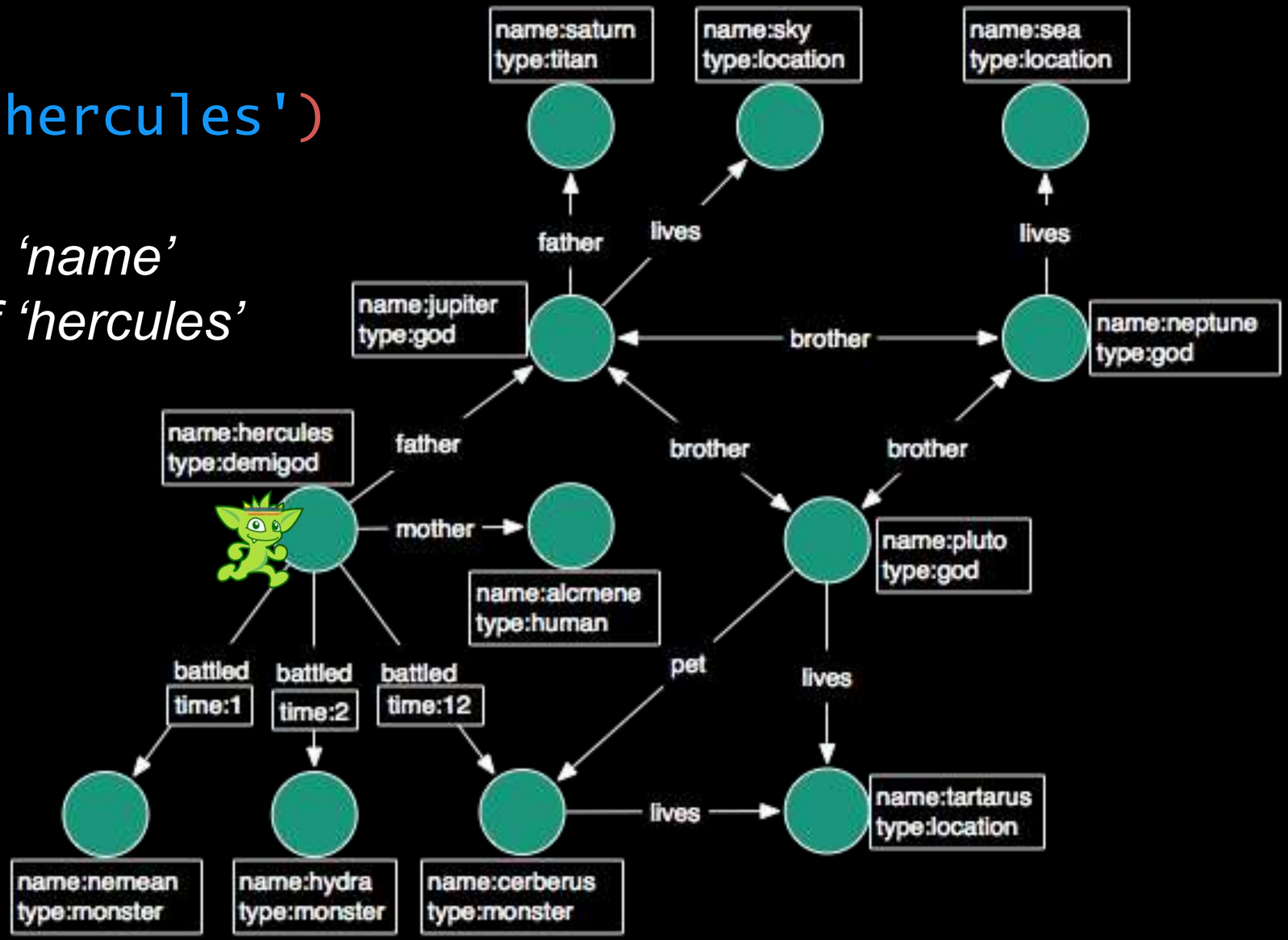
```
gremlin>
 g.V()
```

Select all vertices in the graph



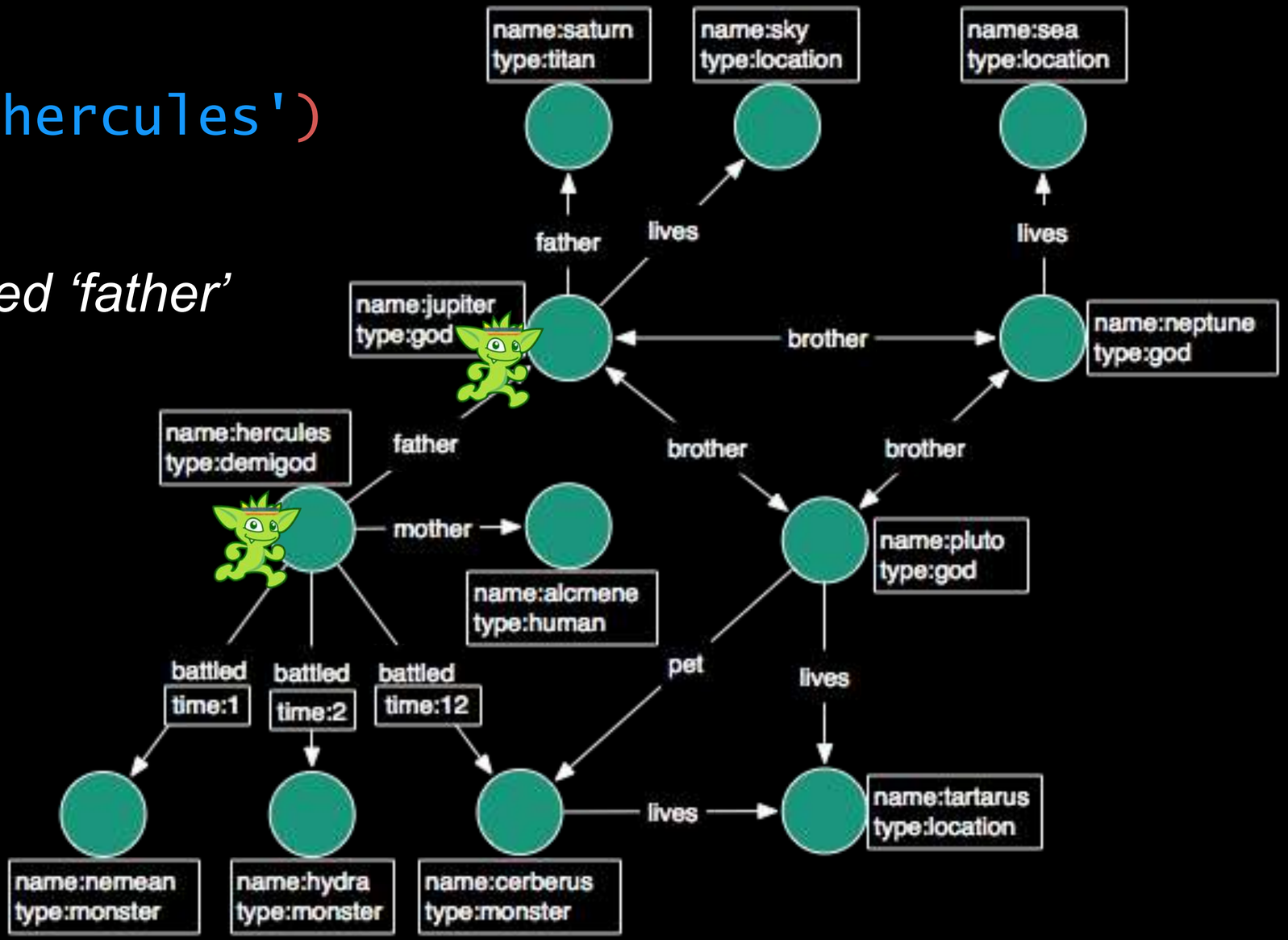
```
gremlin>
 g.V().has('name', ‘hercules')
```

Find the vertex that has a ‘name’
Property with the value of ‘hercules’



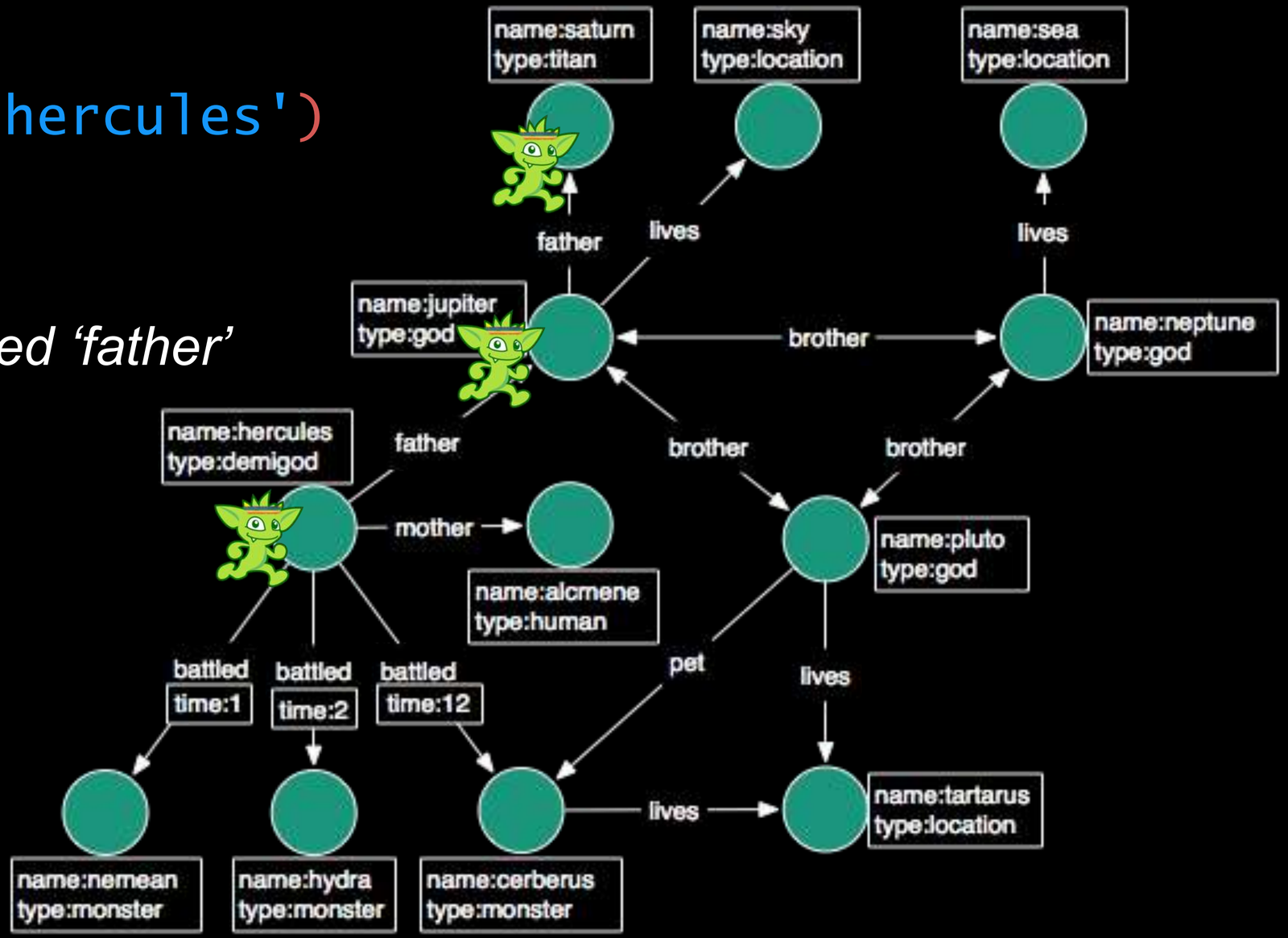
```
gremlin>
 g.V().has('name', ‘hercules')
    .out(‘father')
```

Follow outbound edge named ‘father’ to the connected vertex



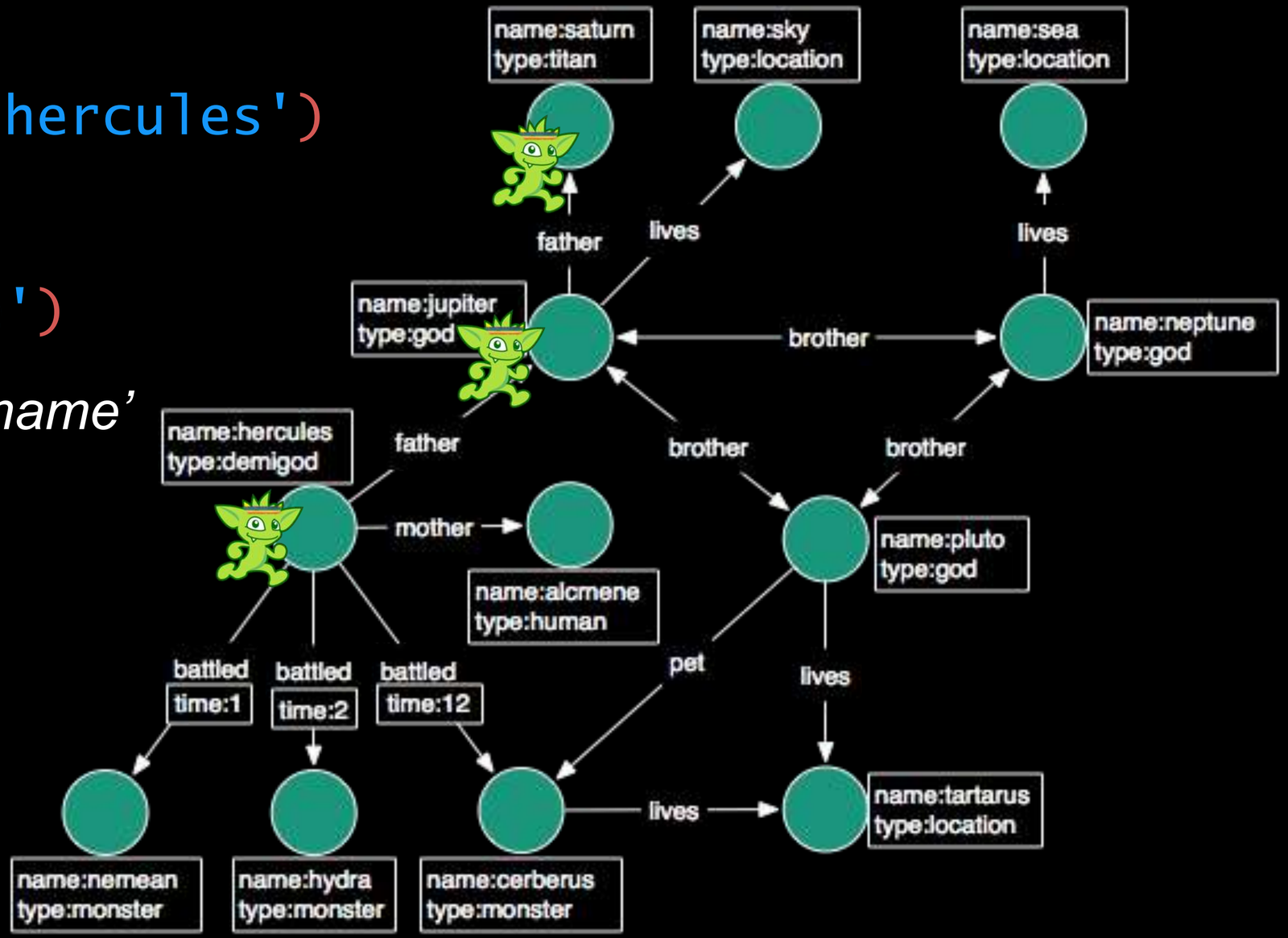
```
gremlin>
 g.V().has('name', ‘hercules')
    .out(‘father')
       .out(‘father')
```

Follow outbound edge named ‘father’ to the connected vertex



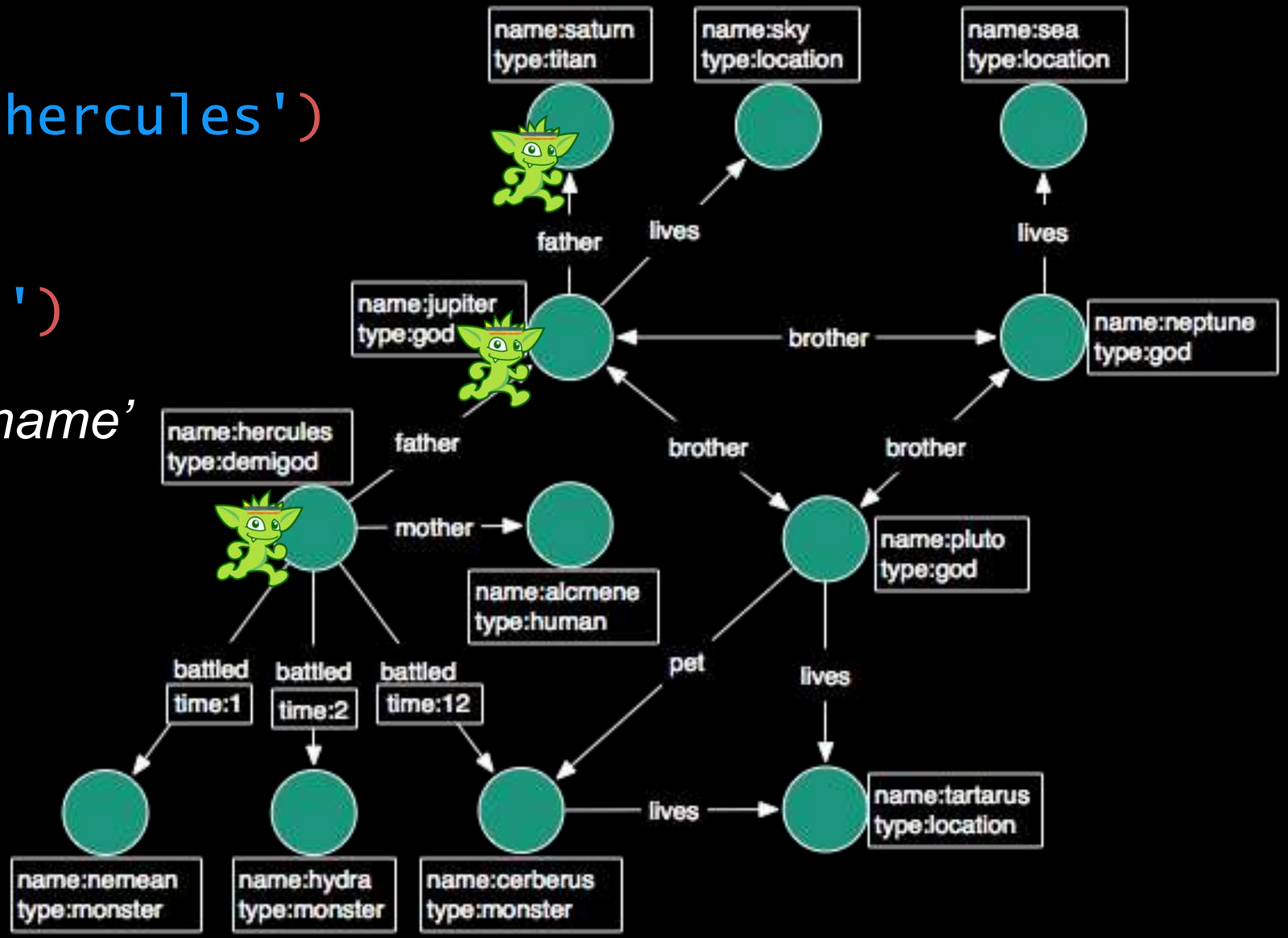
```
gremlin>
 g.V().has('name', ‘hercules')
    .out(‘father')
       .out(‘father')
          .values('name')
```

Select the vertex property ‘name’

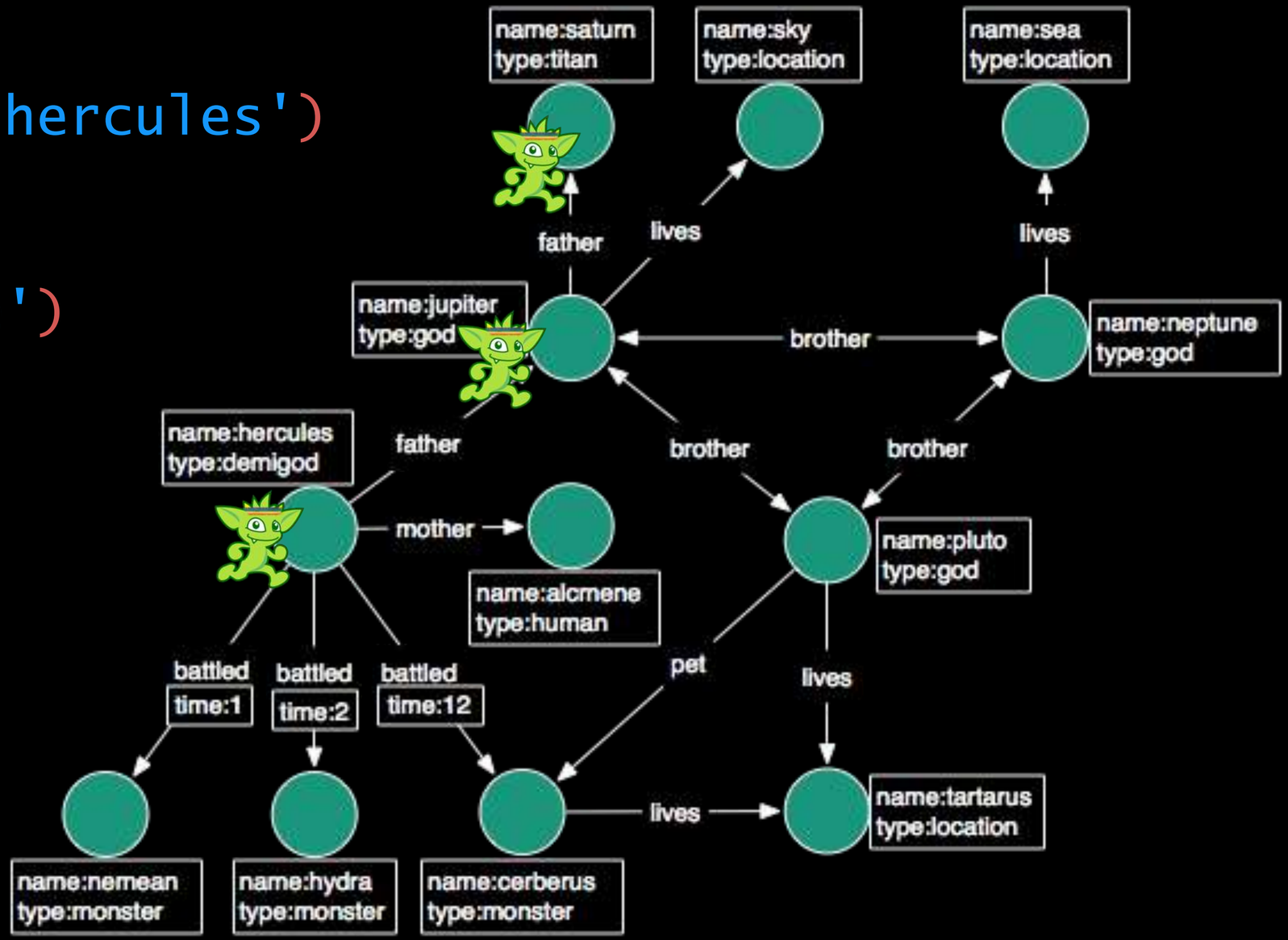


```
gremlin>
 g.V().has('name', ‘hercules')
    .out(‘father')
       .out(‘father')
          .values('name')
```

Select the vertex property ‘name’



```
gremlin>
 g.V().has('name', ‘hercules')
    .out(‘father')
       .out(‘father')
          .values('name')
==> saturn
```



What’s in a version number?

1.1
Unreleased

JanusGraph

0.1.1
May 16, 2017

Contributions Welcome!

- Website: http://janusgraph.org
- GitHub Organization: https://github.com/JanusGraph
- User Mailing List: janusgraph-user@googlegroups.com
- Developer Mailing List: janusgraph-dev@googlegroups.com

Thank you!

Questions?

P. Taylor Goetz, Hortonworks
@ptgoetz