Graph querying and traversal
Graph querying and traversal

Graph Databases

OLTP

TITAN

blazedgraph

OLAP

Gremlin

$G = (V, E)$

OLAP

Graph Processors

Apache Giraph

Apache Spark

Hadoop
Language agnostic: Write once, run everywhere (so long as it’s Java)
Language agnostic: Write once, run everywhere (so long as it’s Java)
Gremlin traverser

- Traversal location
- Local memory
- Graph location
Operation in a distributed system
Gremlin syntax: a couple simple examples

What are the names of Gremlin's friends' friends?
1. Get the vertex with name "gremlin."
2. Traverse to the people that Gremlin knows.
3. Traverse to the people those people know.
4. Get those people's names.

What are the names of the projects created by two friends?
1. ...there exists some "a" who knows "b".
2. ...there exists some "a" who created "c".
3. ...there exists some "b" who created "c".
4. ...there exists some "c" created by 2 people.
5. Get the name of all matching "c" projects.
Gremlin syntax: a couple complex examples

Get the managers from Gremlin to the CEO in the hierarchy.
1. Get the vertex with the name "gremlin."
2. Traverse up the management chain...
3. ...until a person with the title of CEO is reached.
4. Get name of the managers in the path traversed.

```
g.V().has("name","gremlin").
repeat(in("manages"),).
  until(has("title","ceo"),).
path().by("name")
```

Get the distribution of titles amongst Gremlin's collaborators.
1. Get the vertex with the name "gremlin" and label it "a."
2. Get Gremlin's created projects and then who created them...
3. ...that are not Gremlin.
4. Group count those collaborators by their titles.

```
g.V().has("name","gremlin").as("a").
  out("created").in("created").
    where(neq("a"),).
groupCount().by("title")
```
Gremlin syntax: a couple more complex examples

**Get a ranked list of relevant products for Gremlin to purchase.**
1. Get the vertex with the name "gremlin."
2. Get the products Gremlin has purchased and save as "stash."
3. Who else bought those products and what else did they buy...
4. ...that Gremlin has not already purchased.
5. Group count the products and order by their relevance.

```java
g.V().has("name", "gremlin").
    out("bought").aggregate("stash").
    in("bought").out("bought").
    where(not(within("stash")).
    groupCount().order(local).by(values,desc)
```

**Get the 10 most central people in the knows-graph.**
1. Get all people vertices.
2. Calculate their PageRank using knows-edges.
3. Order the people by their friendRank score.
4. Get the top 10 ranked people.

```java
g.V().hasLabel("person").
    pageRank().
    by("friendRank").
    by(outE("knows").
    order().by("friendRank", desc).
    limit(10)
```
Which song writers wrote songs that were sung by Jerry Garcia and performed by the Grateful Dead more than 300 times?

This query is expressed in Gremlin-Java as:

```java
g.V().match(
    as("song").out("sungBy").as("artist"),
    as("song").out("writtenBy").as("writer"),
    as("artist").has("name", "Garcia"),
    where(as("song").values("performances").is(gt(300)))
).select("song", "writer").by("name")
```
Jerry Garcia

```bash
$ bin/gremlin.sh

// ...
(o o)

Plugin activated: aurorast titan
Plugin activated: tinkerpop.server
Plugin activated: tinkerpop.utilities
Plugin activated: tinkerpop.hadoop
Plugin activated: tinkerpop.tinkergraph
gremlin: >install com.datastax:spargi-gremlin:0.1
>>>Load: {com.datastax, spargi-gremlin, 0.1}
gremlin: >plugin use datastax.spargi
>>>datastax.spargi activated

gremlin: >RemoteFactory.open("conf/titan-cassandra.properties")
<<<StandardGraph[cassandra thrift: [127.0.0.1]]
gremlin: >remote connect datastax.spargi graph

gremlin: >query = ""
  SELECT `songName`, `writerName` WHERE {
    `song`-[:writtenBy]->`artist` ,
    `song`-[:writtenBy]->`writer` ,
    `song`-[:name]->`songName` ,
    `writer`-[:name]->`writerName` ,
    `artist`-[:name]->`artist` ,
    `song`-[:performance]->`performances`-[:performance] FILTER (`performances` > 300)
  }
```

```