Gremlin

$$G = (V, E)$$

Michal Červeňanský
Miriama Jánošová
Petra Rebrošová

Gremlin

1. *No bright light*
2. *Do not get him wet*
3. *Do not feed him after midnight, no matter how much he begs*

Gremlin

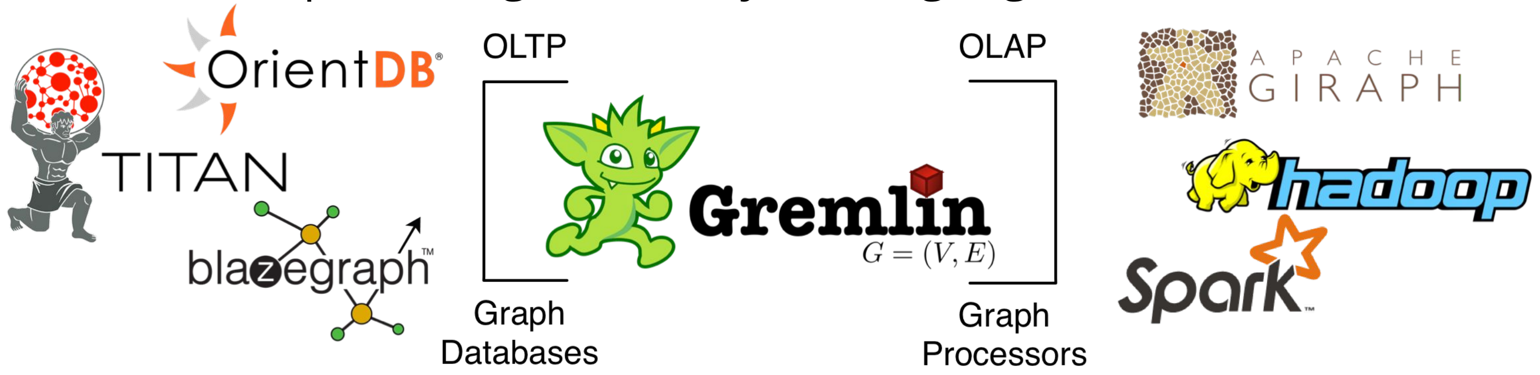
- Developed by *Apache TinkerPop*™ of the *Apache Software Foundation*
  - graph computing framework for both graph databases (OLTP) and graph analytic systems (OLAP)
- Since 2009
- Current stable release: *Gremlin 3.4.4* (14 Oct 2019)
- Cross-platform
- Graph traversal and query language for working with property graphs
- Traverse a graph looking for values, patterns and relationships
  - create sub-graphs
  - add or delete vertices and edges...
- Not widely used (1. Cypher... SPARQL, GraphQL, Gremlin)

Gremlin

- Gremlin is a functional, data-flow **language** and a **traversal machine**
- = virtual machine (instruction set + execution engine)
- Every step is either:
  - *map*-step (transforming the objects in the stream)
  - *filter*-step (removing objects from the stream)
  - *sideEffect*-step (computing statistics about the stream).
- Graph-based virtual machine coordinates the execution of a multi-machine graph traversal

OLTP and OLAP

- *"write once, run anywhere"* -philosophy
    - All TinkerPop-enabled graph systems execute Gremlin traversals (OLTP)
    - Every Gremlin traversal can be evaluated as either a real-time database query or as a batch analytics query (OLAP)
    - User does not need to learn both database query language and domain-specific BigData analytics language



Gremlin usage and support

- **Gremlin Console**
- Data System Providers:
  - Amazon Neptune
  - Hadoop
  - Apache Spark
  - Azure Cosmos DB
  - **Neo4j**
  - Orient DB
  - etc.

- Query Language Providers:
  - SQL, SPARQL (own compilers)
  - Gremlin-Python
  - Gremlin-Java
  - Gremlin.Net
  - Gremlin-Groovy
  - Gremlin-Scala
  - Orge (version for Clojure)
  - etc.

Gremlin Console

- Gremlin Console -> ./bin/gremlin.sh
- Set up the configuration:
  - conf = new BaseConfiguration()
  - conf.setProperty('gremlin.neo4j.directory',
    Neo4j_database_with_loaded_data.db');
  - graph = Neo4jGraph.open(conf);
  - g = graph.traversal();
  - ... use queries over g ...

Traversals

1. Imperative *(procedural)*
    - How to proceed at each step of traversal - the order of operations
2. Declarative *(descriptive)*
    - Allows each traverser to select pattern to execute from a collection of patterns
    - Runtime query planner that chooses which traversal pattern to execute next based on the historic statistics of each pattern
      - favoring those patterns which tend to reduce/filter the most data
3. Hybrid
    - Combination of imperative and declarative

Imperative vs. Declarative Traversals

```
g.V()
.hasLabel('Tag')
.has('tagId', 'gremlin')
.inE().outV()
.hasLabel('Post')
.inE().outV()
.hasLabel('User')
.dedup();
```

```
 g.V()
.match(
        __.as('t').has('tagId', 'gremlin'),
        __.as('t').in('HAS_TAG').as('p'),
        __.as('p').in('POSTED').as('u'))
.select('u').dedup();
```

- Looks similar to Cypher

Neo4j

- The most popular Graph database
- Query language: Cypher

- Also Cypher on Gremlin
  - Execution on TinkerPop engine
  - Mapping Cypher queries to Gremlin - not very efficient

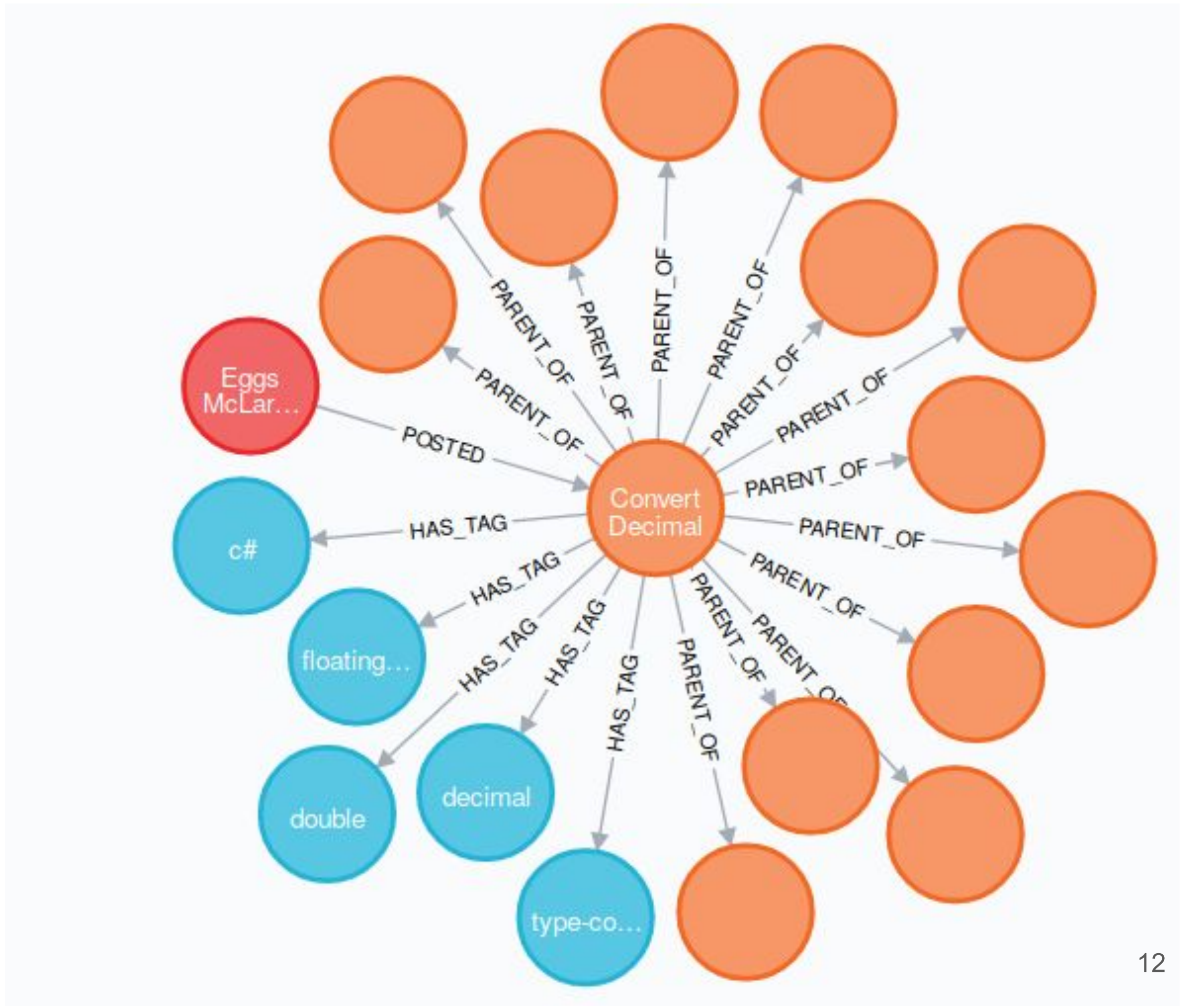
Our Data

- Source: https://archive.org/details/stackexchange
- Data in XML: 85.3 GB
- Data in CSV: 10.88 GB
- DB: 43.6 GB
- *Stack Overflow*
  - Posts
  - Tags
  - Users
  - + relations between Posts
  - + relations between Posts and Tags
  - + relations between Posts and Users

Entities - *Post*

46,947,633

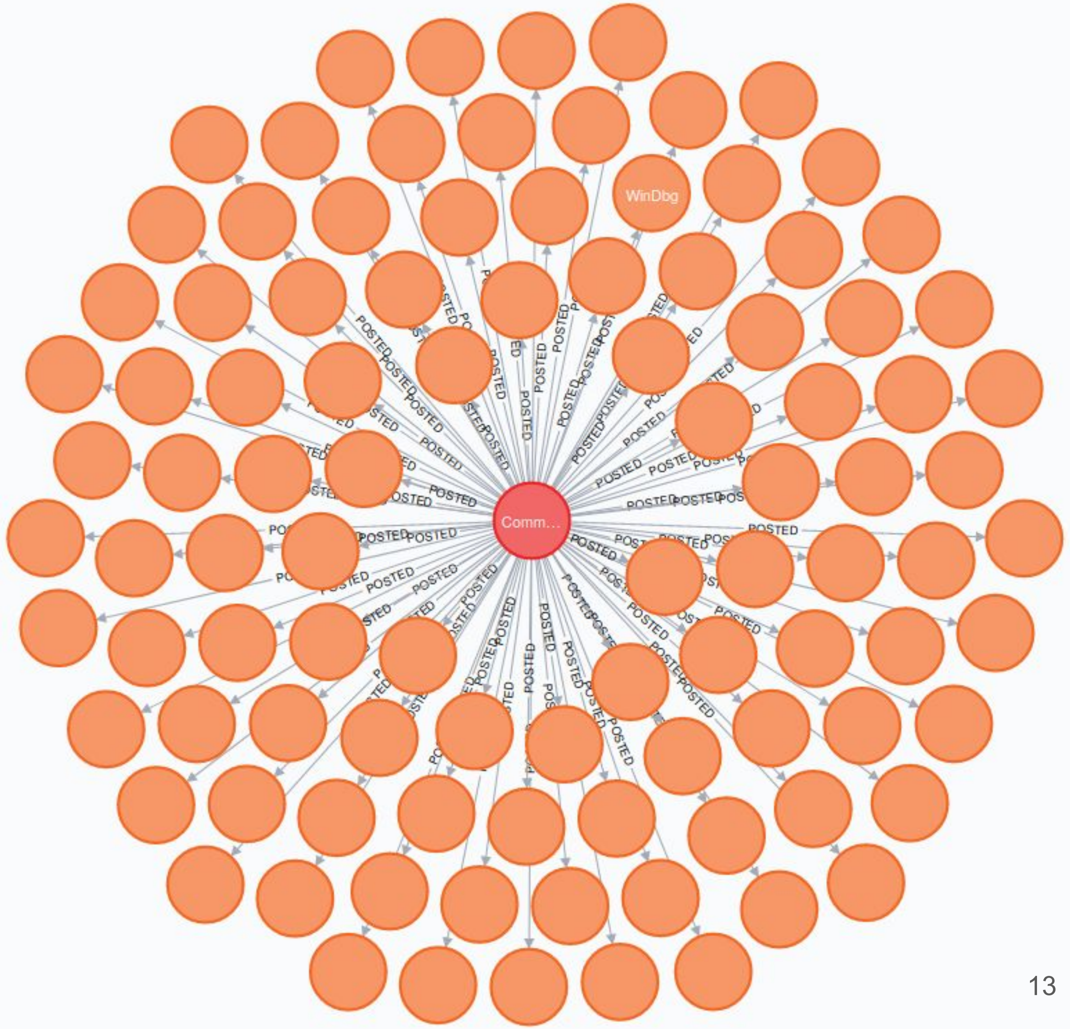
- postId:ID(Post)
- title
- body
- score
- views
- comments



Entities - *User*

11,376,305

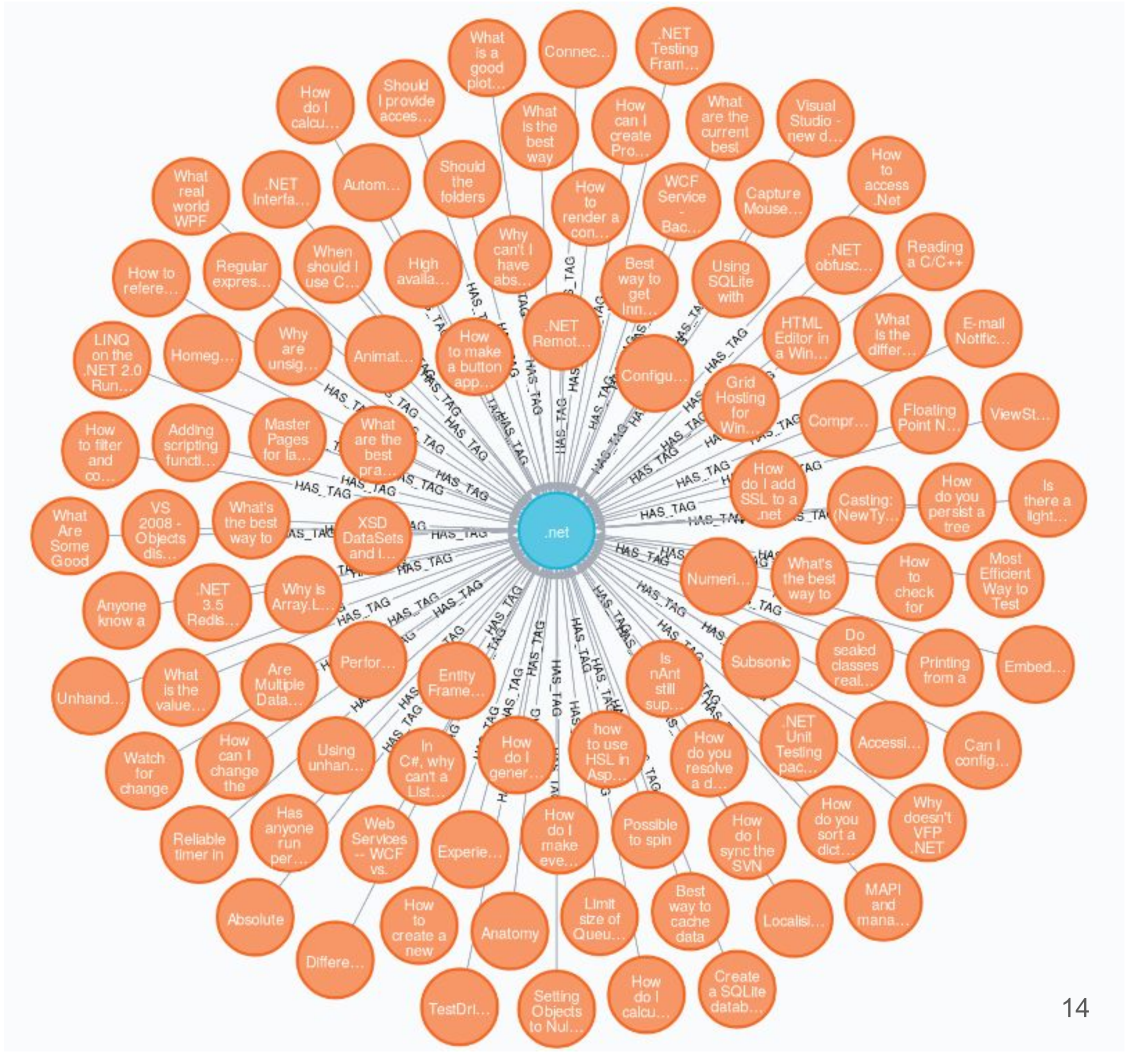
- userId:ID(User)
- displayname
- aboutme
- websiteurl
- location
- profileimageurl
- views
- upvotes
- downvotes



Entities - *Tag*

56,525

- tagId:ID(Tag)



Relationships

- HAS_TAG:
  - 55,078,412
  - Post -[:HAS_TAG]-> Tag
- POSTED:
  - 46,383,097
  - User -[:POSTED]-> Post
- PARENT_OF:
  - 28,248,207
  - Post -[:PARENT_OF]-> Post

Indexing

- Index created on the DB neo4j, not by Gremlin language
- Create index (*Cypher*):
  - CREATE INDEX ON :Post(views);
  - CREATE INDEX ON :Tag(tagId);
  - CREATE INDEX ON :User(reputation)
  - CREATE INDEX ON :Post(postId) - FAILED, run out of RAM

- Indexing failed -> next run -> Neo4j continues with indexing

- Matching with index was slower than without it

Indexing in Gremlin

```
// in Gremlin Console

graph = Neo4jGraph.open(conf);

graph.createIndex('postId', Post.class)

g = graph.traversal()

...
```

Queries - behind the scenes

- Neo4j shows time of execution in its results
- Difficult to measure time in Gremlin queries
  - Clock() does not work - time of saving query into a variable
  - script needed for query execution time

    = (Gremlin DB setup + query)

    - time of setup *(4 commands)*

Comparison of queries - Neo4j and Gremlin

- General querying: Cypher is enough and **generally faster**
- Gremlin:
  - better in **high-level traversing**
  - define exact traversal pattern
  - more execution control
- Cypher
  - the best traversing solution on its own
  - problem: multiple conditions

Comparison of queries - Neo4j and Gremlin

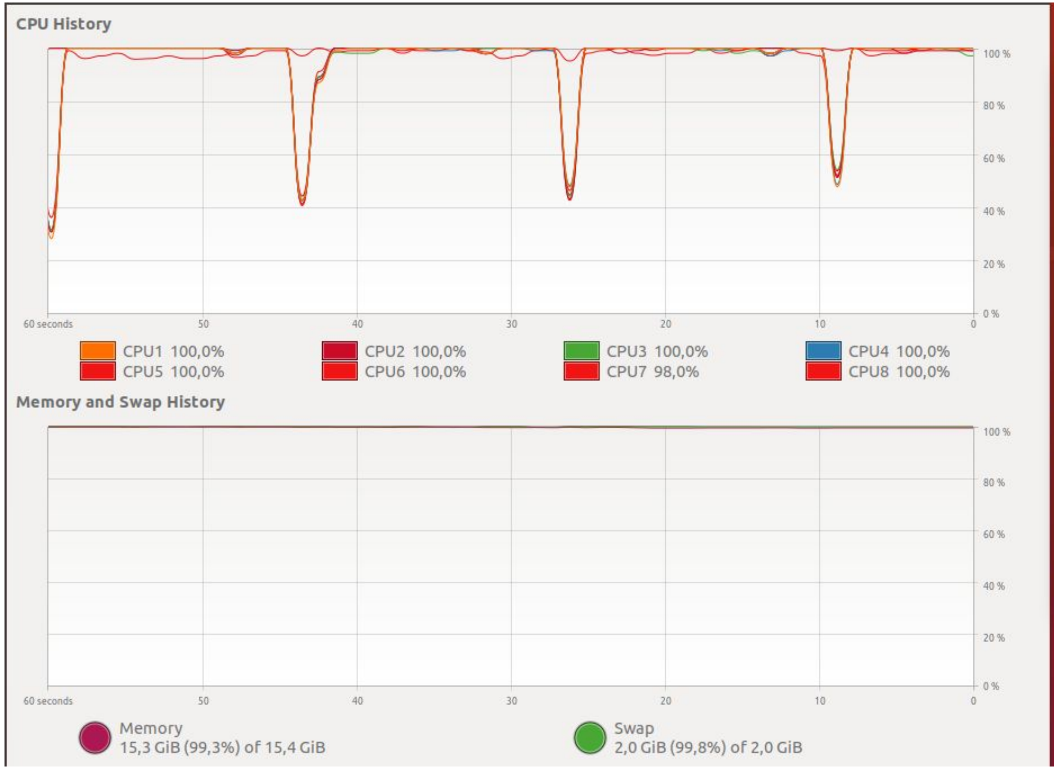
Find top 10 mostly viewed posts.

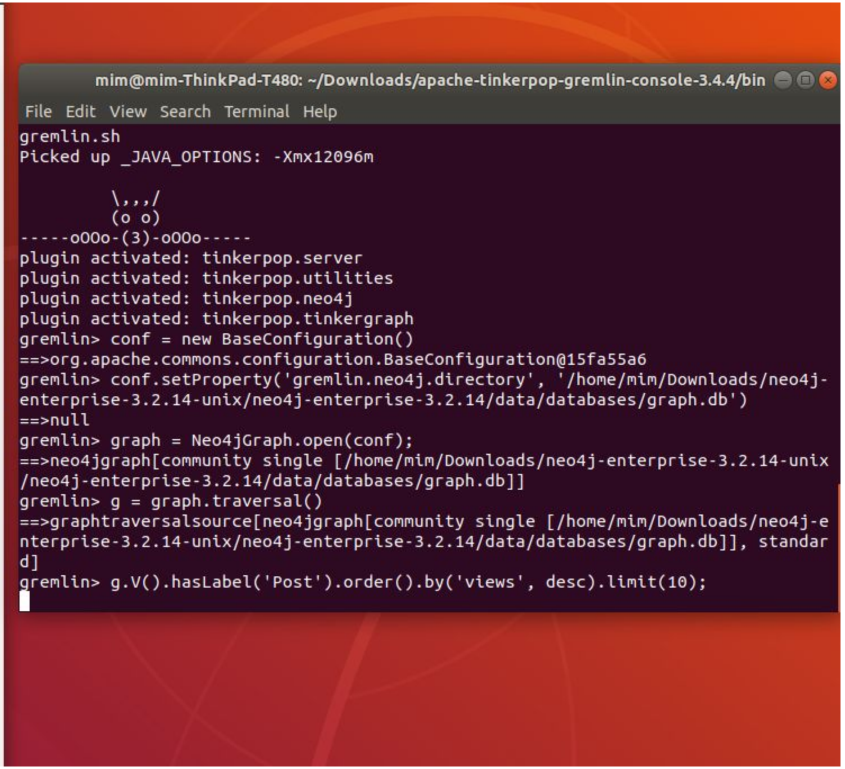
- Neo4j (24.22 s)

```
MATCH (p:Post)
RETURN p
ORDER BY p.views DESC
LIMIT 10;
```

- Gremlin (**FAILED**)

```
g.V()
.hasLabel('Post')
.order().by('views', desc)
.limit(10);
```




mim@mim-ThinkPad-T480: ~/Downloads/apache-tinkerpop-gremlin-console-3.4.4/bin
File Edit View Search Terminal Help

```
gremlin.sh
Picked up _JAVA_OPTIONS: -Xmx12096m

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.neo4j
plugin activated: tinkerpop.tinkergraph
gremlin> conf = new BaseConfiguration()
==>org.apache.commons.configuration.BaseConfiguration@15fa55a6
gremlin> conf.setProperty('gremlin.neo4j.directory', '/home/mim/Downloads/neo4j-
enterprise-3.2.14-unix/neo4j-enterprise-3.2.14/data/databases/graph.db')
==>null
gremlin> graph = Neo4jGraph.open(conf);
==>neo4jgraph[community single [/home/mim/Downloads/neo4j-enterprise-3.2.14-unix
/neo4j-enterprise-3.2.14/data/databases/graph.db]]
gremlin> g = graph.traversal()
==>graphtraversalsource[neo4jgraph[community single [/home/mim/Downloads/neo4j-e
nterprise-3.2.14-unix/neo4j-enterprise-3.2.14/data/databases/graph.db]], standar
d]
gremlin> g.V().hasLabel('Post').order().by('views', desc).limit(10);
```

Comparison of queries - Neo4j and Gremlin

Select all users, who posted Posts with tag ‘gremlin’. (1167 results)

- Neo4j (11ms)

```
MATCH
(t:Tag {tagId:'gremlin'}) <-[:HAS_TAG]-(p:Post)
<-[:POSTED] - (u:User)
RETURN DISTINCT u;
```

- Gremlin (80 ms)

```
g.V()
.hasLabel('Tag')
.has('tagId', 'gremlin')
.inE().outV()
.hasLabel('Post')
.inE().outV()
.hasLabel('User')
.dedup();
```

Comparison of queries - Neo4j and Gremlin

Return top 10 trolls and count of their posts.

- Neo4j (65 552 results, 24.5 s)

```
MATCH (u:User) WITH u
ORDER BY u.downvotes DESC
LIMIT 10
MATCH (u)-[:POSTED]->(p:Post)
RETURN COUNT (p);
```

- Gremlin (65 552 results, 45.7 s)

```
g.V()
.hasLabel('User')
.order().by('downvotes', desc)
.limit(10)
.outE()
.hasLabel('POSTED')
.outV()
.count();
```

Difficult queries (1)

Top 10 user whose posts are the most heterogeneous and has better score than 300.

- Neo4j (6.1 min)

```
MATCH (t:Tag)<-[:HAS_TAG]-(p:Post)
<-[:POSTED]-(u:User) where p.score > 300
RETURN distinct u,
COLLECT(distinct t) as tags,
count(distinct t) as ctags
ORDER BY SIZE(tags) DESC
Limit 10;
```

- Gremlin (3.85 min)

```
g.V.().hasLabel(‘User’).as(‘users’)
.outE().outV().hasLabel('Post')
.has('score', gt(300)).as(‘post’)
.map {
        def t = g.V(it.get())
                .out(‘HAS_TAG’).count()
}.as(‘counts’)
.select(‘users’, ‘counts’)
.by(‘counts’, desc).limit(10)
```

Difficult queries (2)

Select and order by reputation all users that answered a question with tags “gremlin” and “neo4j”. (118 results)

- Neo4j (8.21min)

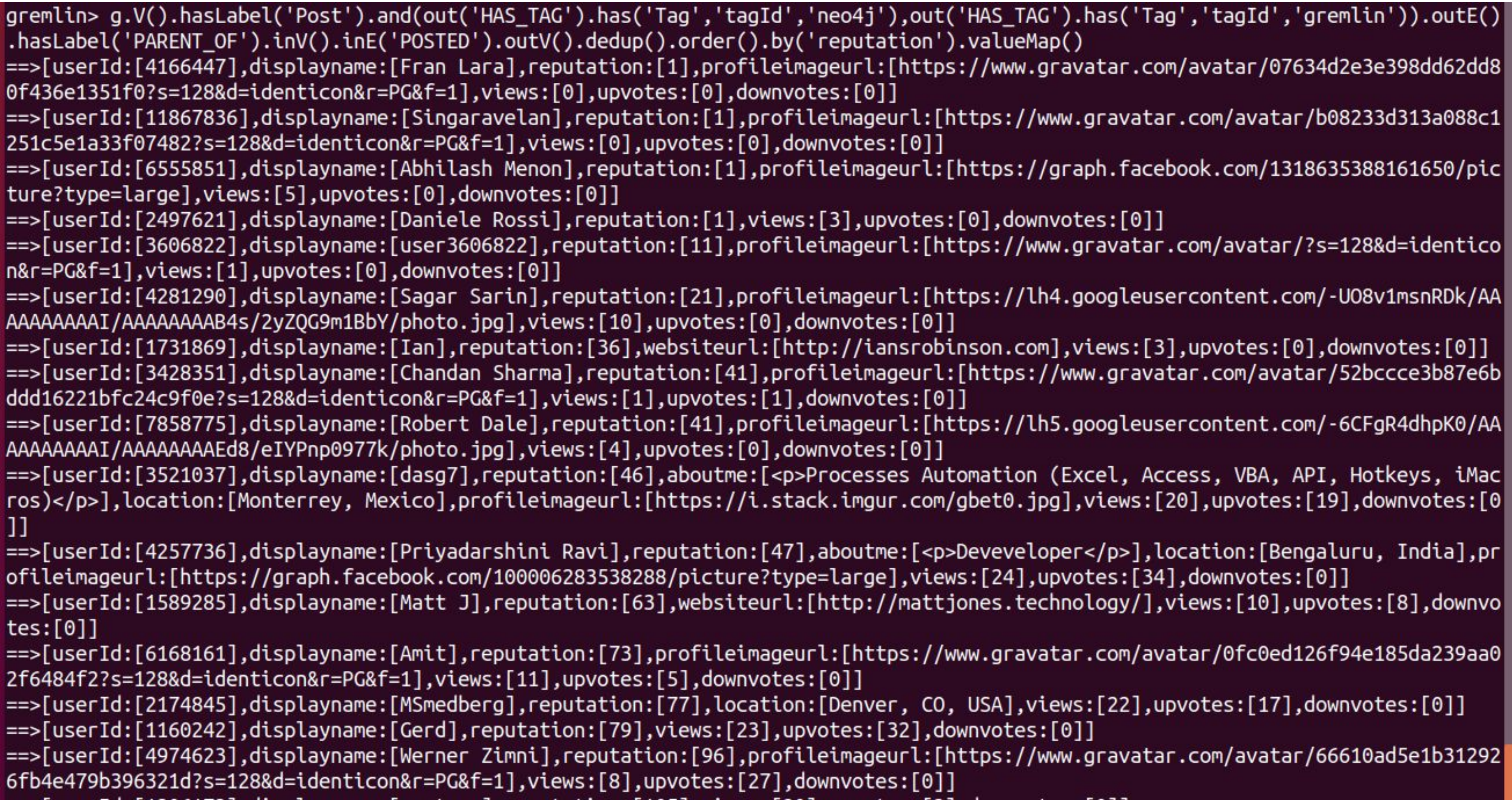
```
with ['neo4j', 'gremlin'] as tags match (t:Tag)
where t.tagId in tags
with collect(t) as taglist
match (p:Post)
where all (t in taglist where
(p)-[:HAS_TAG]->(t))
match (u:User) - [:POSTED] -> (ans:Post) <- [:PARENT_OF] - (p)
return u order by toInteger(u.reputation) desc;
```

Difficult queries (2 cont.)

Select and order by reputation all users that answered a question with tags “gremlin” and “neo4j”.

- Gremlin (3.31 min)

```
g.V()
.hasLabel('Post')
.and(out('HAS_TAG')
.has('Tag','tagId','neo4j'),out('HAS_TAG')
.has('Tag','tagId','gremlin'))
.outE().hasLabel('PARENT_OF')
.inV().inE('POSTED').outV().dedup()
.order().by('reputation').values()
```



```
gremlin> g.V().hasLabel('Post').and(out('HAS_TAG').has('Tag','tagId','neo4j'),out('HAS_TAG').has('Tag','tagId','gremlin')).outE()
.hasLabel('PARENT_OF').inV().inE('POSTED').outV().dedup().order().by('reputation').valueMap()
==>[userId:[4166447],displayname:[Fran Lara],reputation:[1],profileimageurl:[https://www.gravatar.com/avatar/07634d2e3e398dd62dd8
0f436e1351f0?s=128&d=identicon&r=PG&f=1],views:[0],upvotes:[0],downvotes:[0]]
==>[userId:[11867836],displayname:[Singaravelan],reputation:[1],profileimageurl:[https://www.gravatar.com/avatar/b08233d313a088c1
251c5e1a33f07482?s=128&d=identicon&r=PG&f=1],views:[0],upvotes:[0],downvotes:[0]]
==>[userId:[6555851],displayname:[Abhilash Menon],reputation:[1],profileimageurl:[https://graph.facebook.com/1318635388161650/pic
ture?type=large],views:[5],upvotes:[0],downvotes:[0]]
==>[userId:[2497621],displayname:[Daniele Rossi],reputation:[1],views:[3],upvotes:[0],downvotes:[0]]
==>[userId:[3606822],displayname:[user3606822],reputation:[11],profileimageurl:[https://www.gravatar.com/avatar/?s=128&d=identico
n&r=PG&f=1],views:[1],upvotes:[0],downvotes:[0]]
==>[userId:[4281290],displayname:[Sagar Sarin],reputation:[21],profileimageurl:[https://lh4.googleusercontent.com/-UO8v1msnRDk/AA
AAAAAAAAI/AAAAAAAAB4s/2yZQG9m1BbY/photo.jpg],views:[10],upvotes:[0],downvotes:[0]]
==>[userId:[1731869],displayname:[Ian],reputation:[36],websiteurl:[http://iansrobinson.com],views:[3],upvotes:[0],downvotes:[0]]
==>[userId:[3428351],displayname:[Chandan Sharma],reputation:[41],profileimageurl:[https://www.gravatar.com/avatar/52bccce3b87e6b
ddd16221bfc24c9f0e?s=128&d=identicon&r=PG&f=1],views:[1],upvotes:[1],downvotes:[0]]
==>[userId:[7858775],displayname:[Robert Dale],reputation:[41],profileimageurl:[https://lh5.googleusercontent.com/-6CFgR4dhpK0/AA
AAAAAAAAI/AAAAAAAAEd8/eIYPnp0977k/photo.jpg],views:[4],upvotes:[0],downvotes:[0]]
==>[userId:[3521037],displayname:[dasg7],reputation:[46],aboutme:[<p>Processes Automation (Excel, Access, VBA, API, Hotkeys, iMac
ros)</p>],location:[Monterrey, Mexico],profileimageurl:[https://i.stack.imgur.com/gbet0.jpg],views:[20],upvotes:[19],downvotes:[0
]]
==>[userId:[4257736],displayname:[Priyadarshini Ravi],reputation:[47],aboutme:[<p>Deveveloper</p>],location:[Bengaluru, India],pr
ofileimageurl:[https://graph.facebook.com/100006283538288/picture?type=large],views:[24],upvotes:[34],downvotes:[0]]
==>[userId:[1589285],displayname:[Matt J],reputation:[63],websiteurl:[http://mattjones.technology/],views:[10],upvotes:[8],downvo
tes:[0]]
==>[userId:[6168161],displayname:[Amit],reputation:[73],profileimageurl:[https://www.gravatar.com/avatar/0fc0ed126f94e185da239aa0
2f6484f2?s=128&d=identicon&r=PG&f=1],views:[11],upvotes:[5],downvotes:[0]]
==>[userId:[2174845],displayname:[MSmedberg],reputation:[77],location:[Denver, CO, USA],views:[22],upvotes:[17],downvotes:[0]]
==>[userId:[1160242],displayname:[Gerd],reputation:[79],views:[23],upvotes:[32],downvotes:[0]]
==>[userId:[4974623],displayname:[Werner Zimni],reputation:[96],profileimageurl:[https://www.gravatar.com/avatar/66610ad5e1b31292
6fb4e479b396321d?s=128&d=identicon&r=PG&f=1],views:[8],upvotes:[27],downvotes:[0]]
```

Demo

Thank you for your attention

Do you have any questions?