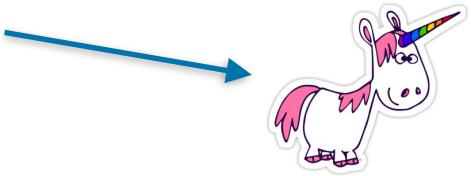
ScaDS
DRESDEN LEIPZIG

THE MAGIC OF GRAPH DATABASES

FABIAN GÄRTNER

www.scads.de

Norbert E. Horn
https://twitter.com/Norbert_E_Horn



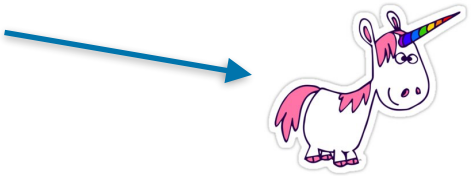
ScaDS
DRESDEN LEIPZIG

THE MAGIC OF GRAPH DATABASES

FABIAN GÄRTNER

www.scads.de

Norbert E. Horn
https://twitter.com/Norbert_E_Horn



ScaDS
DRESDEN LEIPZIG

Gremlin

THE MAGIC OF GRAPH DATABASES

FABIAN GÄRTNER

www.scads.de

Norbert E. Horn
https://twitter.com/Norbert_E_Horn

ScaDS

Gremlin

Game of Thrones
Season 6

www.scads.de

ScaDS DRESDEN LEIPZIG

MOTIVATION

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project
  - A Supergenome is a common coordinate system for all genomes in a multiple alignment.

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project
  - A Supergenome is a common coordinate system for all genomes in a multiple alignment.
- Data is a graph structure

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project
  - A Supergenome is a common coordinate system for all genomes in a multiple alignment.
- Data is a graph structure
  - 100 Way Vertebrata
    - 109,850,411 Vertices
    - 4,062,653,410 Edges

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project
  - A Supergenome is a common coordinate system for all genomes in a multiple alignment.
- Data is a graph structure
  - 100 Way Vertebrata
    - 109,850,411 Vertices
    - 4,062,653,410 Edges

My graph is bigger!

ScaDS DRESDEN LEIPZIG

MOTIVATION

- Supergenome project
  - A Supergenome is a common coordinate system for all genomes in a multiple alignment.
- Data is a graph structure
  - 100 Way Vertebrata
    - 109,850,411 Vertices
    - 4,062,653,410 Edges
- Operations are graph-operations

My graph is bigger!

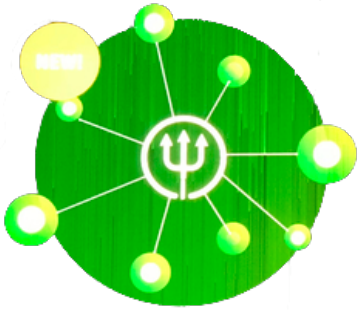
ScaDS DRESDEN LEIPZIG

GRAPH DATABASES

neo4j

Azure Cosmos DB

IBM Graph™

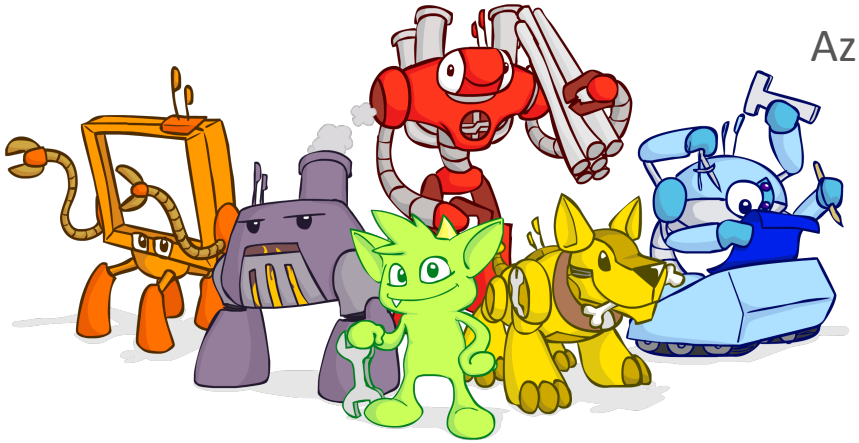


Amazon Neptune

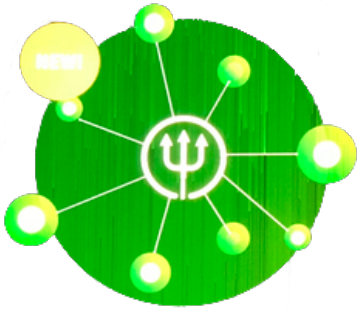
JanusGraph

GRAPH DATABASES

Azure Cosmos DB



Apache TinkerPop



Amazon Neptune

ScaDS DRESDEN LEIPZIG

APACHE TINKERPOP

Apache TinkerPop™ is a graph computing framework
for both graph databases (OLTP)and graph analytic systems (OLAP).

ScaDS DRESDEN LEIPZIG

APACHE TINKERPOP

Apache TinkerPop™ is a graph computing framework
for both graph databases (OLTP)and graph analytic systems (OLAP).

- Open source

ScaDS DRESDEN LEIPZIG

APACHE TINKERPOP

Apache TinkerPop™ is a graph computing framework
for both graph databases (OLTP)and graph analytic systems (OLAP).

- Open source
- Vertex based query language is Gremlin:

ScaDS DRESDEN LEIPZIG

APACHE TINKERPOP

Apache TinkerPop™ is a graph computing framework
for both graph databases (OLTP)and graph analytic systems (OLAP).

- Open source
- Vertex based query language is Gremlin:
  - python
  - ruby
  - **java**
  - js
  - go
  - .NET - C#
  - php
  - scala
  - typescript

ScaDS DRESDEN LEIPZIG

GREMLIN-JAVA

GREMLIN-JAVA

MAVEN

```
<dependencies>
  <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>tinkergraph-gremlin</artifactId>
      <version>3.3.1</version>
  </dependency>
  <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>neo4j-gremlin</artifactId>
      <version>3.3.1</version>
  </dependency>
  <dependency>
      <groupId>org.neo4j</groupId>
      <artifactId>neo4j-tinkerpop-api-impl</artifactId>
      <version>0.7-3.2.3</version>
  </dependency>
</dependencies>
```

GREMLIN-JAVA

MAVEN

```xml
<dependencies>
  <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>tinkergraph-gremlin</artifactId>
      <version>3.3.1</version>
  </dependency>
  <dependency>
      <groupId>org.apache.tinkerpop</groupId>
      <artifactId>neo4j-gremlin</artifactId>
      <version>3.3.1</version>
  </dependency>
  <dependency>
      <groupId>org.neo4j</groupId>
      <artifactId>neo4j-tinkerpop-api-impl</artifactId>
      <version>0.7-3.2.3</version>
  </dependency>
</dependencies>
```

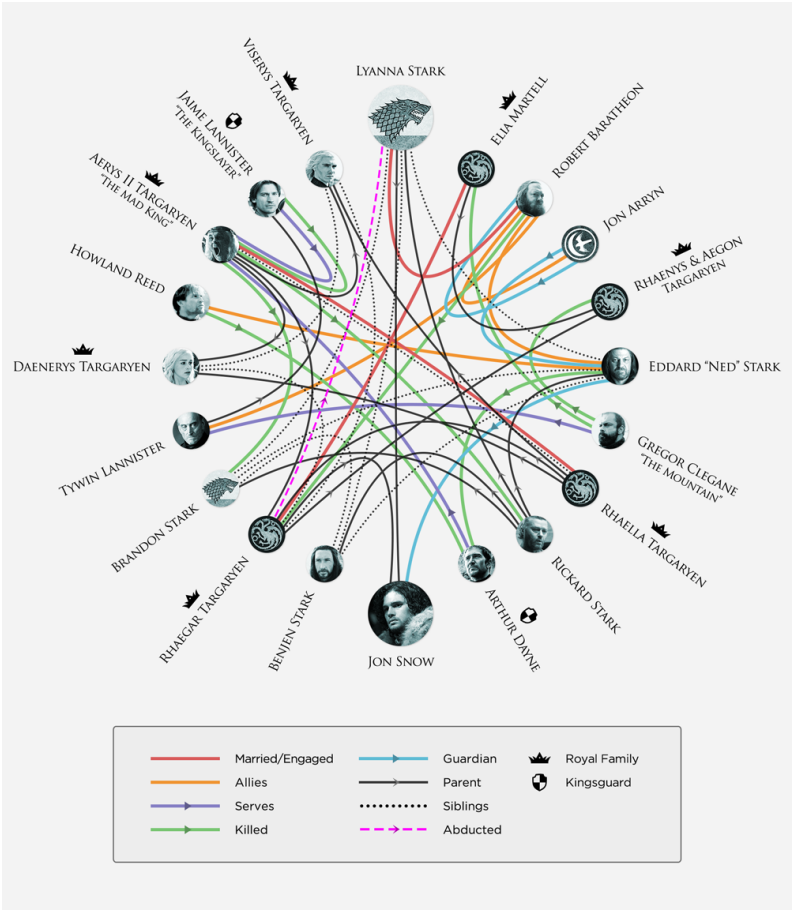
```java
public class GraphExample {

    public static void main(String[] args) throws Exception {
        //open Graph
        Graph g = Neo4jGraph.open("example/graph.db/");
        GraphTraversalSource t = g.traversal();

        //TODO add something

        //TODO query and print something

        //Commit changes
        g.tx().commit();
        //Close Graph
        g.close();
    }

    public static void print(Iterator<Vertex> it ) {
        for(Vertex v: (Iterable<Vertex>) () -> it) {
            System.out.println((String)v.value("name"));
        }
    }

}
```

ScaDS
DRESDEN LEIPZIG

CREATE GRAPH

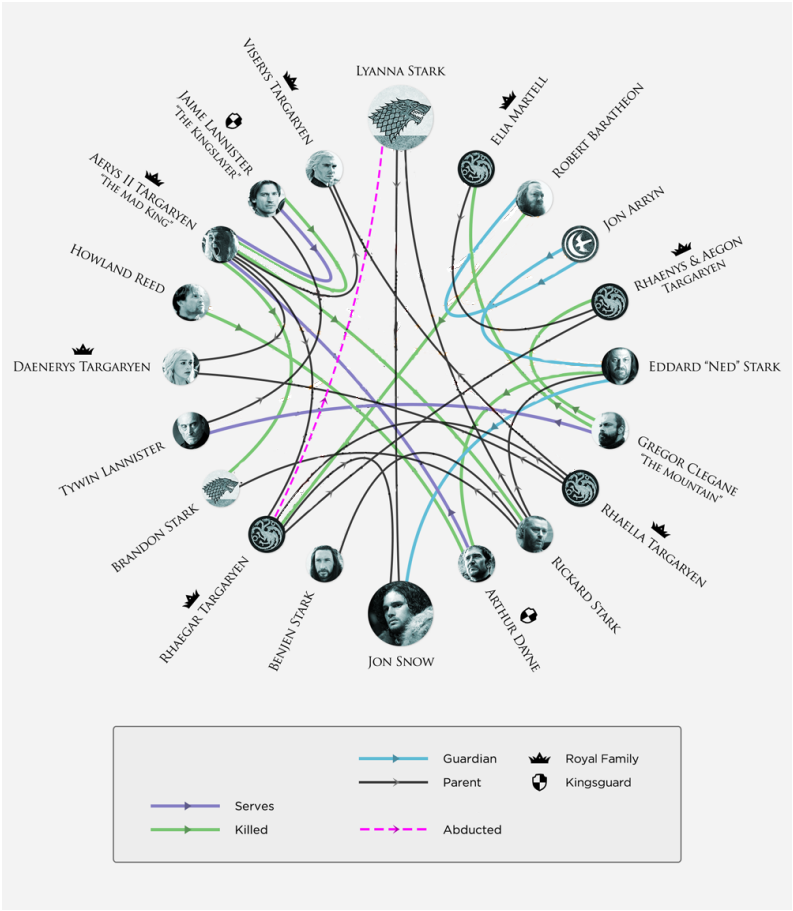
ScaDS DRESDEN LEIPZIG

CREATE GRAPH

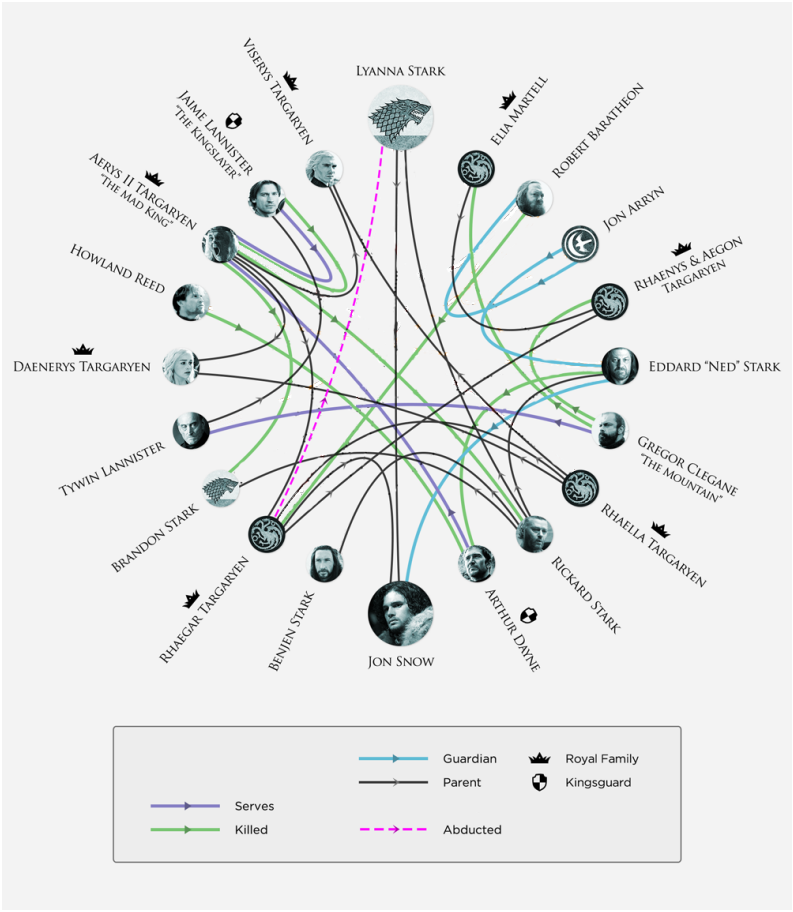


ScaDS DRESDEN LEIPZIG

CREATE GRAPH



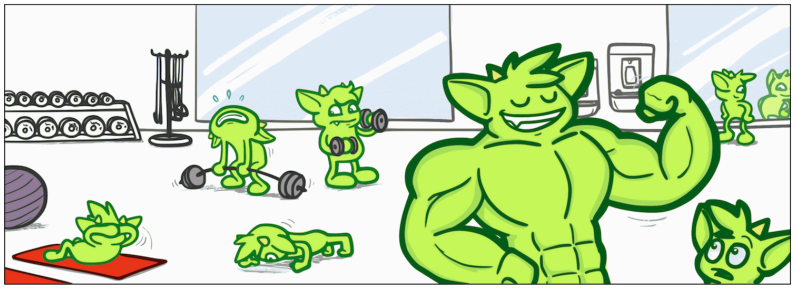
CREATE GRAPH

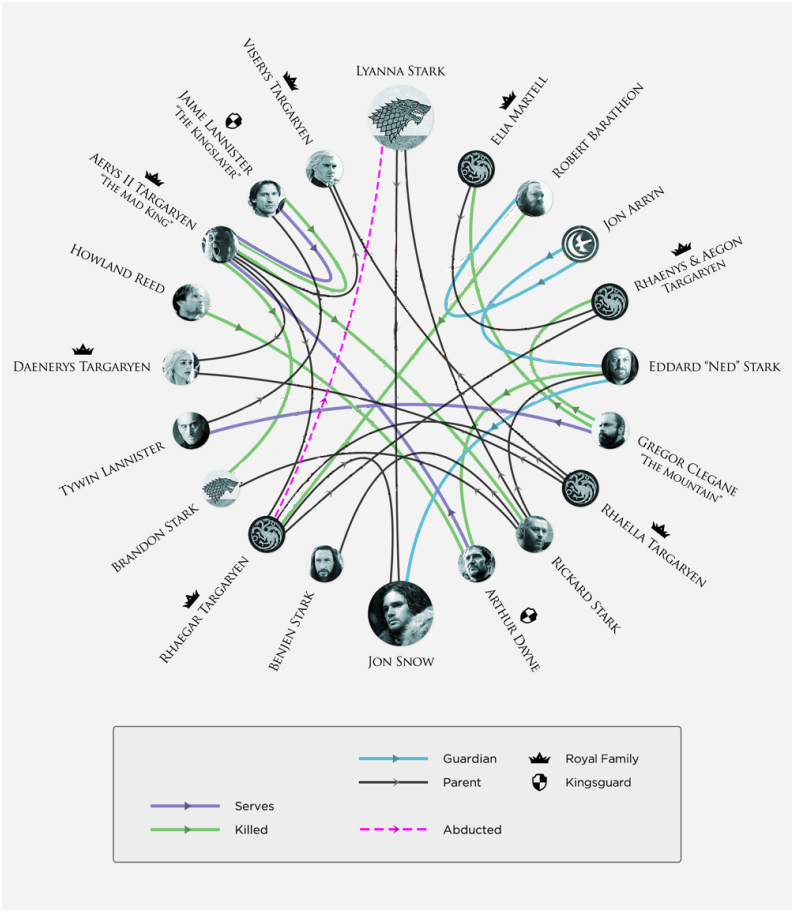



```
//TODO add something
t
    .addV("Person").property("name","Lyanna Stark").as("1")
    .addV("Person").property("name","Elia Martell").property("Royal Family",true).as("2")
    .addV("Person").property("name","Robert Baratheon").as("3")
    .addV("Person").property("name","John Arryn").as("4")
    .addE("Guardian").from("4").to("3")
    .addV("Person").property("name","Rhaenys & Aegon Targaryen").property("Royal Family",true).as("5")
    .addE("Parent").from("2").to("5")
    .addV("Person").property("name","Eddard \"Ned\" Stark").as("6")
    .addE("Guardian").from("4").to("6")
    .addV("Person").property("name","Gregor Clegane \"The Mountain\"").as("7")
    .addE("Killed").from("7").to("2")
    .addE("Killed").from("7").to("5")
    .addV("Person").property("name","Rhaella Targaryen").property("Royal Family",true).as("8")
    .addV("Person").property("name","Rickard Stark").as("9")
    .addE("Parent").from("9").to("1")
    .addE("Parent").from("9").to("6")
    .addV("Person").property("name","Arthur Dayne").property("Kingsguard",true).as("10")
    .addE("Killed").from("6").to("10")
    .addV("Person").property("name","Jon Snow").property("age", 23).as("11")
    .addE("Parent").from("1").to("11")
    .addE("Guardian").from("6").to("11")
    .addV("Person").property("name","Benjen Stark").as("12")
    .addE("Parent").from("9").to("12")
    .addV("Person").property("name","Rhaegar Targaryen").property("Royal Family",true).as("13")
    .addE("Killed").from("3").to("13")
    .addE("Parent").from("8").to("13")
    .addE("Abducted").from("13").to("1")
    .addE("Parent").from("13").to("5")
    .addE("Parent").from("13").to("11")
    .addV("Person").property("name","Brandon Stark").as("14")
    .addE("Parent").from("9").to("14")
    .addV("Person").property("name","Tywin Lannister").as("15")
    .addE("Serves").from("7").to("15")
    .addV("Person").property("name","Daenerys Targaryen").property("Royal Family",true).property("age",22).as("16")
    .addE("Parent").from("8").to("16")
    .addV("Person").property("name","Howland Reed").as("17")
    .addE("Killed").from("17").to("10")
    .addV("Person").property("name","Aerys II Targaryen \"The Mad King\"").property("Royal Family",true).as("18")
    .addE("Serves").from("10").to("18")
    .addE("Killed").from("18").to("9")
    .addE("Parent").from("18").to("13")
    .addE("Killed").from("18").to("14")
    .addE("Parent").from("18").to("16")
    .addV("Person").property("name","Jaime Lannister").property("Kingsguard",true).property("age",43).as("19")
    .addE("Parent").from("15").to("19")
    .addE("Killed").from("19").to("18")
    .addE("Serves").from("19").to("18")
    .addV("Person").property("name","Visrys Targaryen").property("Royal Family",true).as("20")
    .addE("Parent").from("8").to("20")
    .addE("Parent").from("18").to("20")
    .iterate();
```

ScaDS DRESDEN LEIPZIG

ACCESS VERTICES

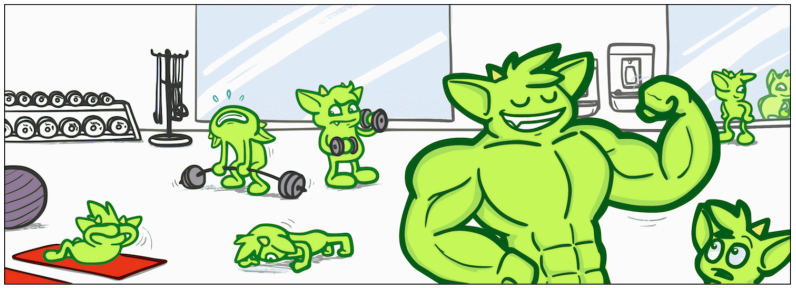


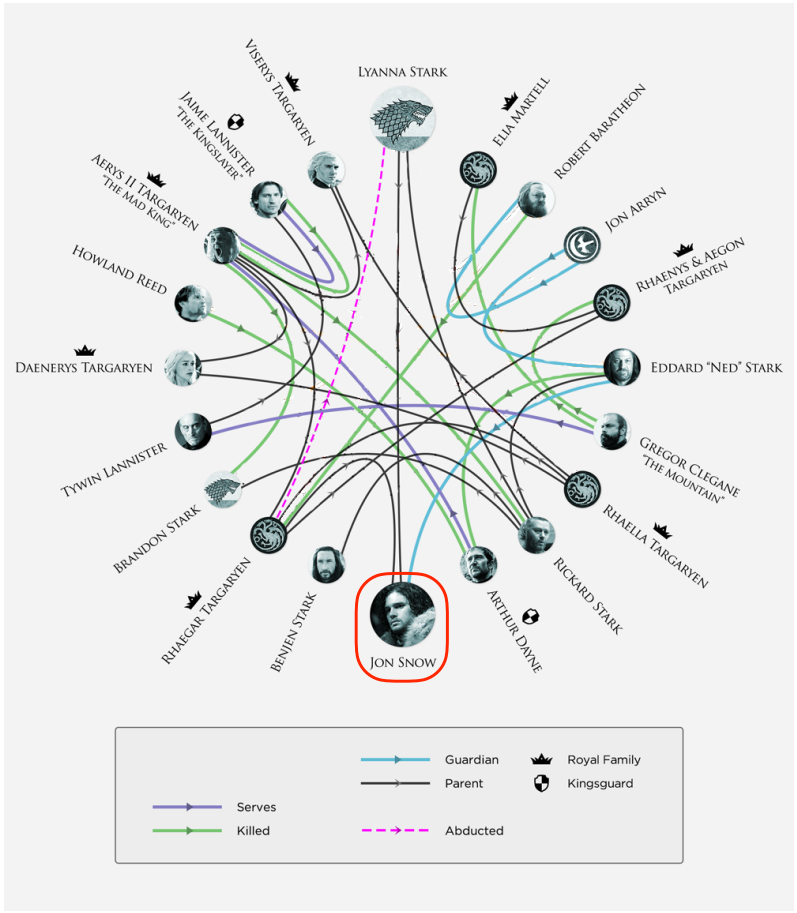



```
//TODO query and print something

Vertex jon = t.V().has("name","Jon Snow").next();

System.out.println(t.V(jon).values("age").next());


print(t.V().has("Royal Family"));


print(t.V().has("age",P.lt(25)));


P<String> nonTargarian = new P<String>((x, y) -> (x).indexOf(y) < 0 , "Targaryen");

print(t.V().has("Royal Family").has("name",nonTargarian));
```

ScaDS DRESDEN LEIPZIG

ACCESS VERTICES





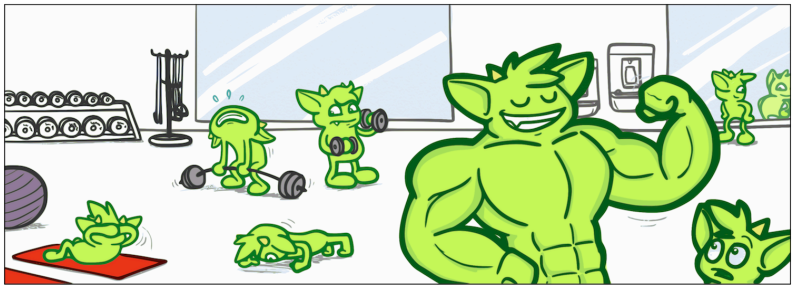

```
//TODO query and print something

Vertex jon = t.V().has("name","Jon Snow").next();

System.out.println(t.V(jon).values("age").next());

print(t.V().has("Royal Family"));

print(t.V().has("age",P.lt(25)));

P<String> nonTargarian = new P<String>((x, y) -> (x).indexOf(y) < 0 , "Targaryen");

print(t.V().has("Royal Family").has("name",nonTargarian));
```

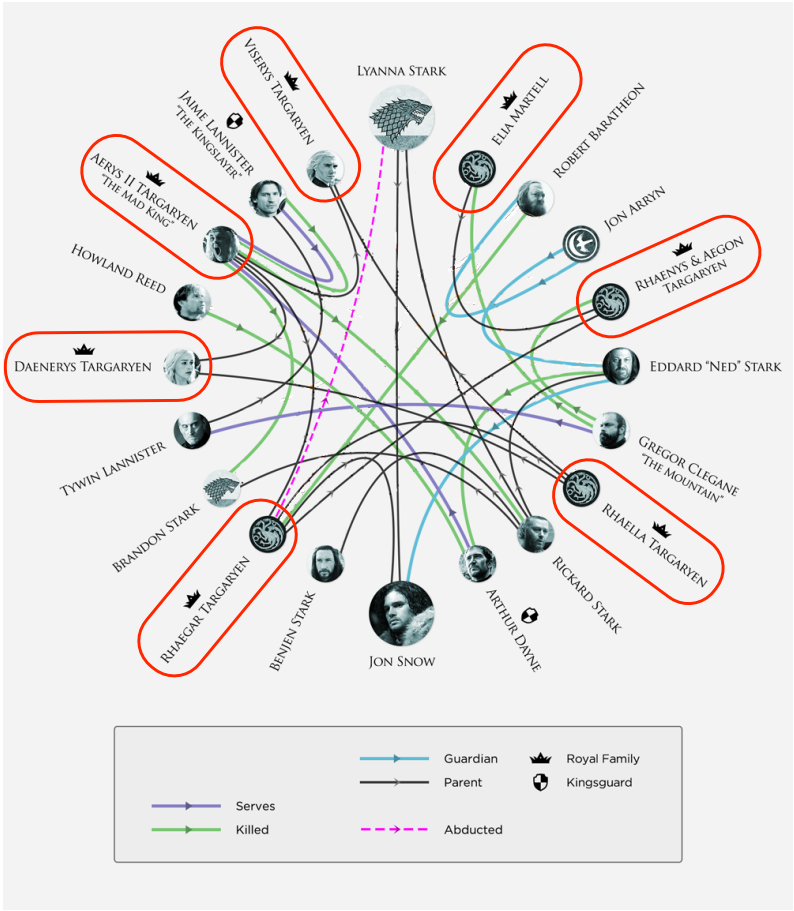
23

How old is
“Jon Snow”?

ScaDS DRESDEN LEIPZIG

ACCESS VERTICES






```
//TODO query and print something
Vertex jon = t.V().has("name","Jon Snow").next();
System.out.println(t.V(jon).values("age").next());
```

23

```
print(t.V().has("Royal Family"));
```

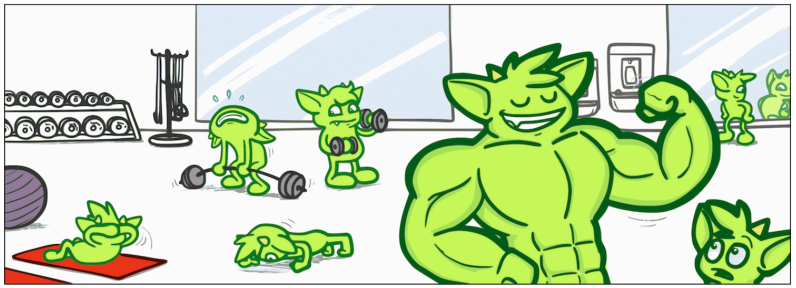
```
print(t.V().has("age",P.lt(25)));
```

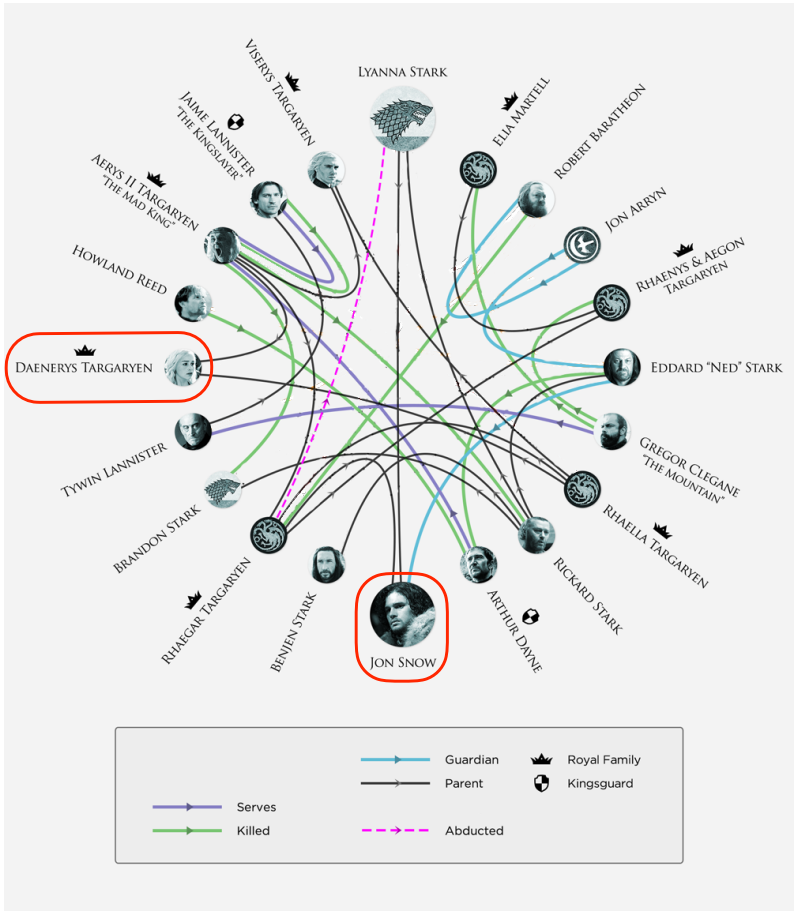
```
P<String> nonTargarian = new P<String>((x, y) -> (x).indexOf(y) < 0 , "Targaryen");
print(t.V().has("Royal Family").has("name",nonTargarian));
```

Who is member of the royal family?

ScaDS DRESDEN LEIPZIG

ACCESS VERTICES






```
//TODO query and print something

Vertex jon = t.V().has("name","Jon Snow").next();

System.out.println(t.V(jon).values("age").next());
```

23

```
print(t.V().has("Royal Family"));
```

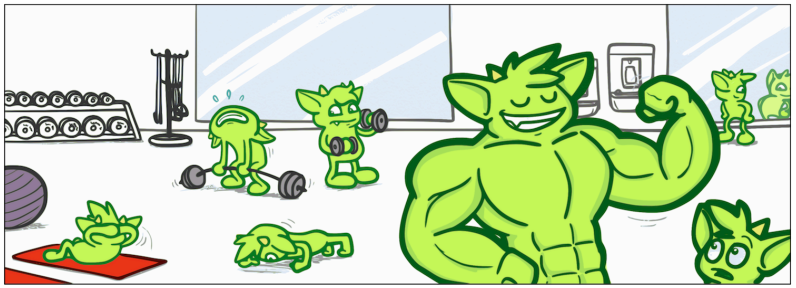
```
print(t.V().has("age",P.lt(25)));
```

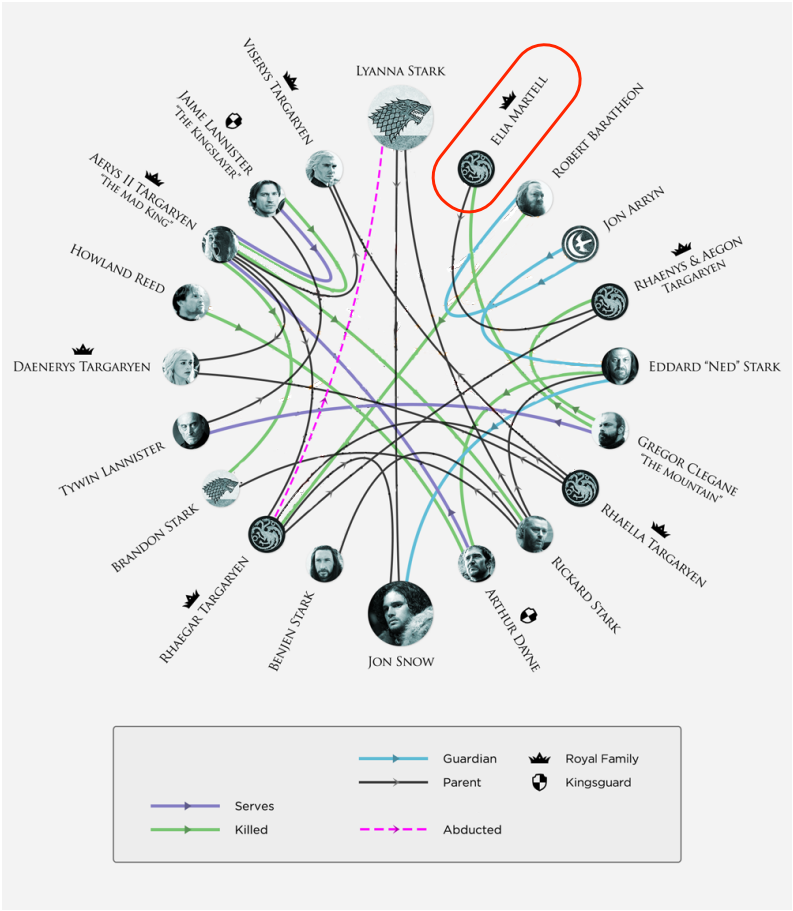
```
P<String> nonTargarian = new P<String>((x, y) -> (x).indexOf(y) < 0 , "Targaryen");

print(t.V().has("Royal Family").has("name",nonTargarian));
```

Who is younger
then 25 years?

ScaDS DRESDEN LEIPZIG

ACCESS VERTICES






```
//TODO query and print something

Vertex jon = t.V().has("name","Jon Snow").next();

System.out.println(t.V(jon).values("age").next());
```

23

```
print(t.V().has("Royal Family"));
```

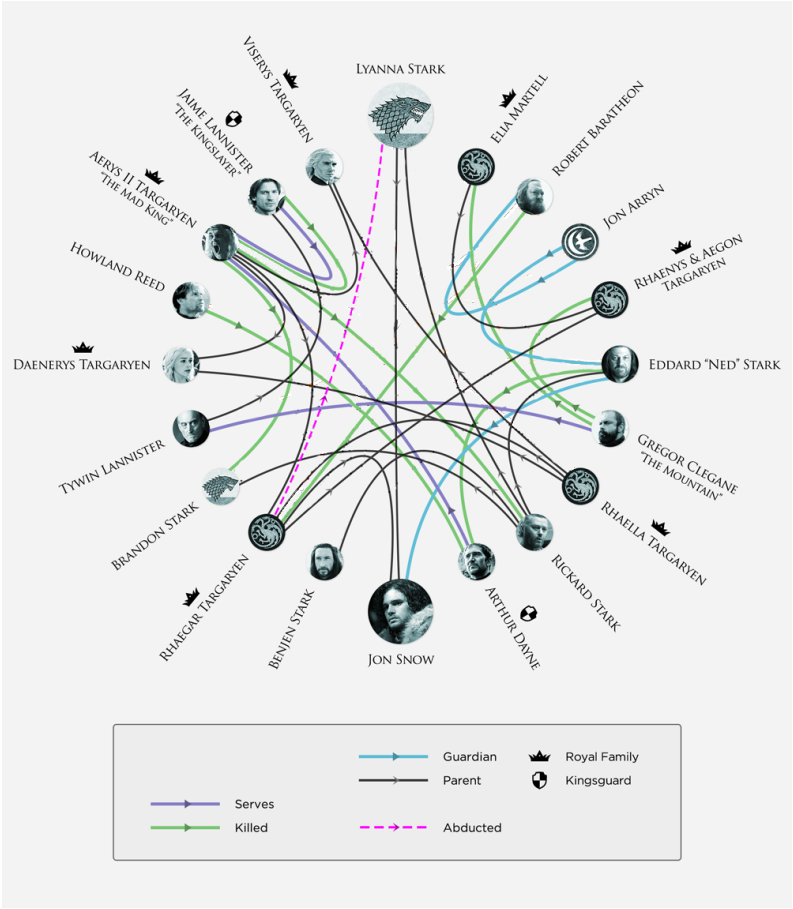
```
print(t.V().has("age",P.lt(25)));
```

```
P<String> nonTargarian = new P<String>((x, y) -> (x).indexOf(y) < 0 , "Targaryen");

print(t.V().has("Royal Family").has("name",nonTargarian));
```

Who is in the royal family
but is not named “Targaryen”?

ScaDS DRESDEN LEIPZIG

ACCESS EDGES




```
print(t.V().has("Royal Family").out("Killed"));
```

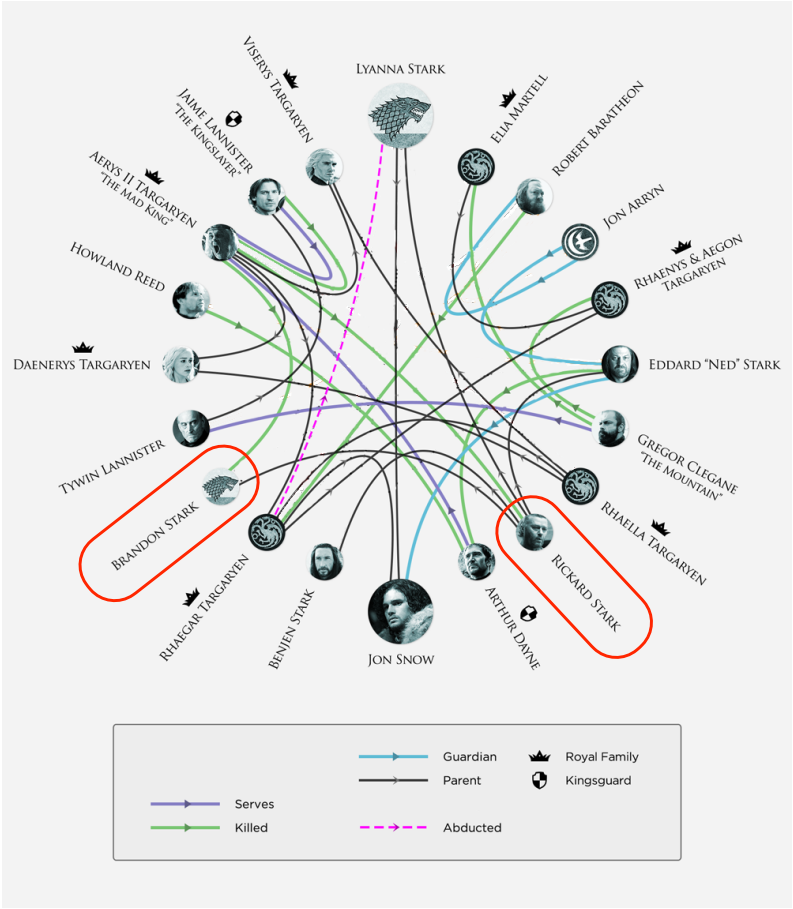
```
print(t.V().has("Royal Family").in("Killed").dedup());
```

```
print(t.V().has("Royal Family").where(__.in("Killed")));
```

```
print(t.V().as("q").out("Killed").in("Serves").where(P.eq("q")));
```

ScaDS DRESDEN LEIPZIG

ACCESS EDGES




```
print(t.V().has("Royal Family").out("Killed"));
```

```
print(t.V().has("Royal Family").in("Killed").dedup());
```

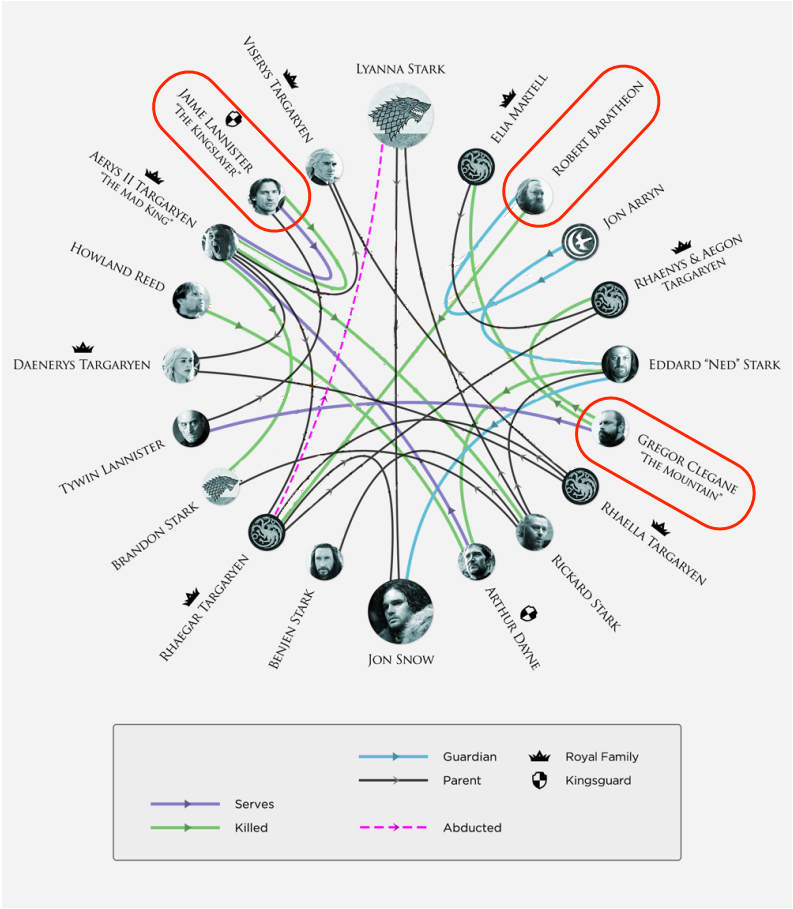
```
print(t.V().has("Royal Family").where(__.in("Killed")));
```

```
print(t.V().as("q").out("Killed").in("Serves").where(P.eq("q")));
```

Who got killed by a member of the royal family?

ScaDS DRESDEN LEIPZIG

ACCESS EDGES




```
print(t.V().has("Royal Family").out("Killed"));
```

```
print(t.V().has("Royal Family").in("Killed").dedup());
```

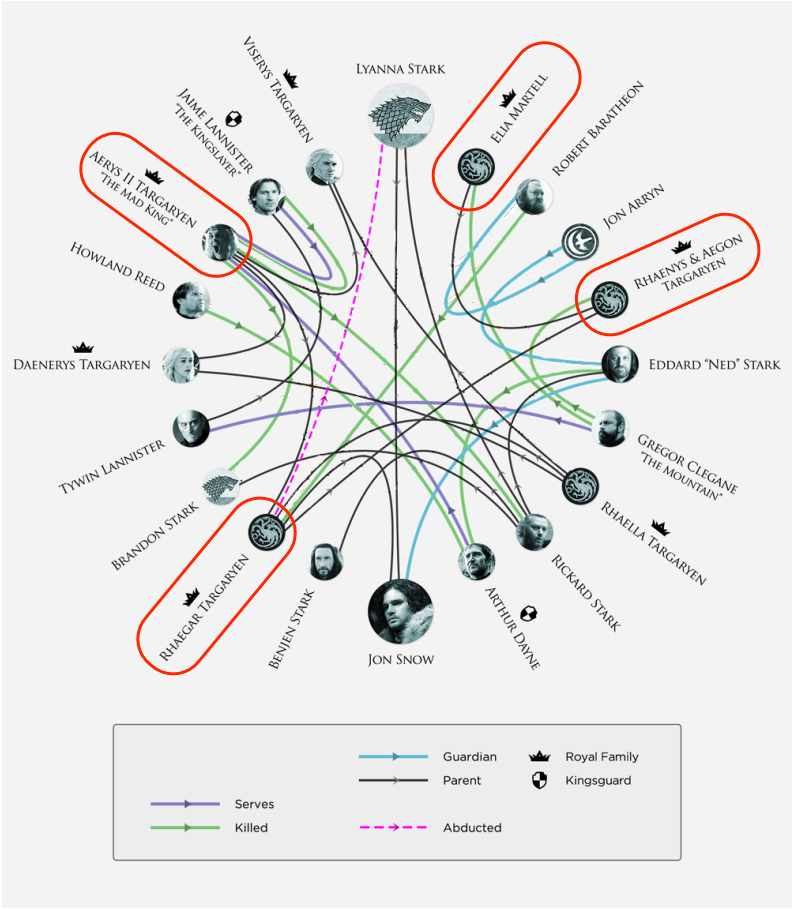
```
print(t.V().has("Royal Family").where(__.in("Killed")));
```

```
print(t.V().as("q").out("Killed").in("Serves").where(P.eq("q")));
```

Who killed at least one member of the royal family?

ScaDS DRESDEN LEIPZIG

ACCESS EDGES




```
print(t.V().has("Royal Family").out("Killed"));
```

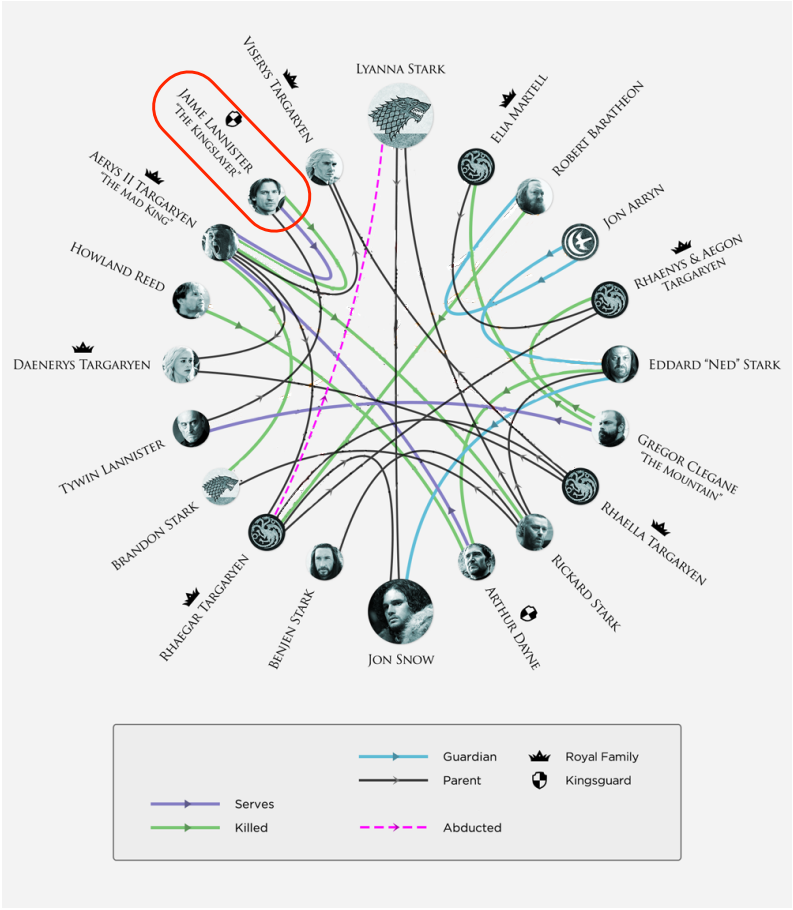
```
print(t.V().has("Royal Family").in("Killed").dedup());
```

```
print(t.V().has("Royal Family").where(__.in("Killed")));
```

```
print(t.V().as("q").out("Killed").in("Serves").where(P.eq("q")));
```

Who of the royal family
got killed?

ACCESS EDGES




```
print(t.V().has("Royal Family").out("Killed"));
```

```
print(t.V().has("Royal Family").in("Killed").dedup());
```

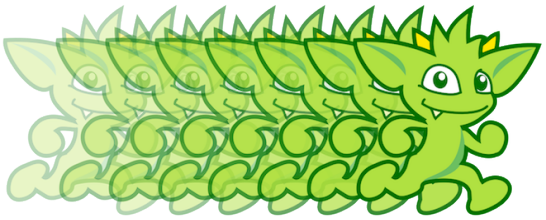
```
print(t.V().has("Royal Family").where(__.in("Killed")));
```

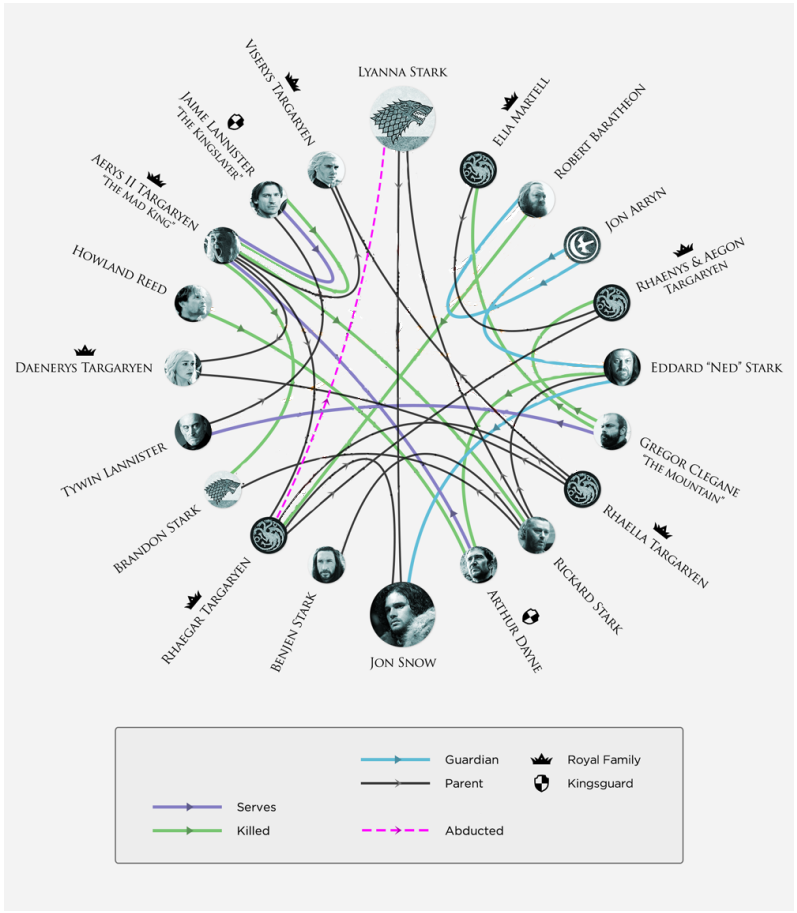
```
print(t.V().as("q").out("Killed").in("Serves").where(P.eq("q")));
```

Who killed the person
that he serves?

ScaDS DRESDEN LEIPZIG

REPEAT





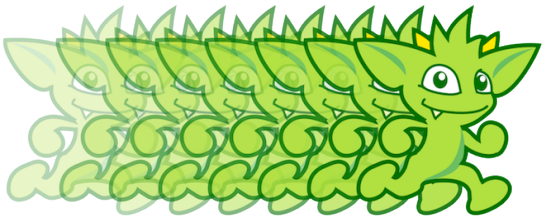

```
print(t.V(jon).repeat(__.in("Parent")).times(2));
```

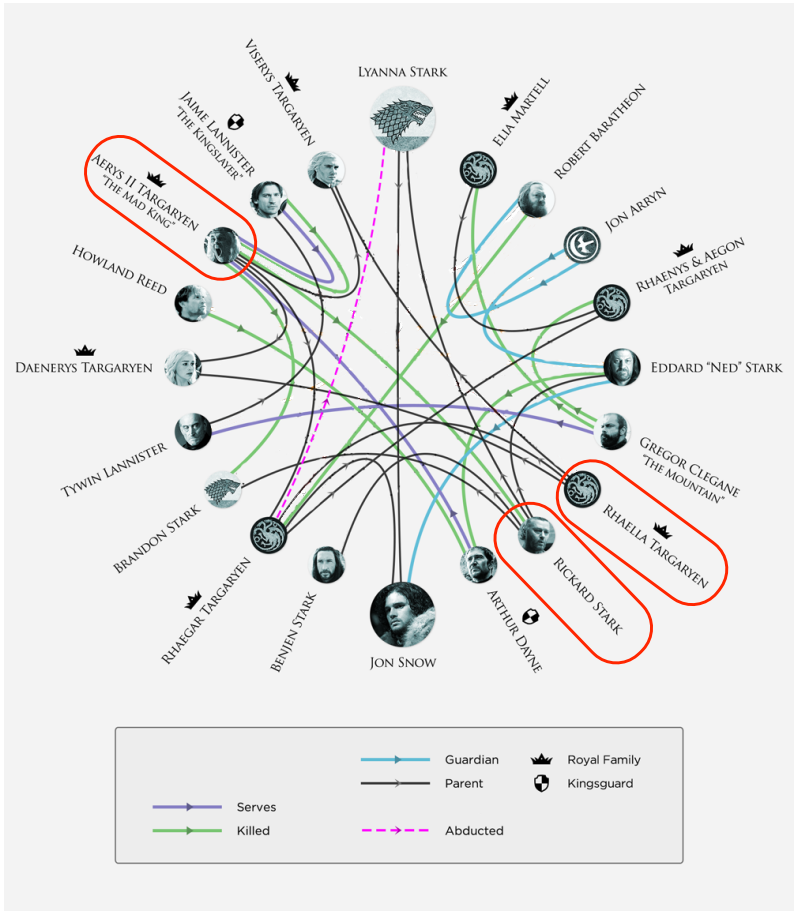
```
print(t.V(jon).repeat(__.in("Parent").store("Parents"))
        .cap("Parents").unfold());
```

Who are the grandparents of “Jon Snow”?

ScaDS DRESDEN LEIPZIG

REPEAT





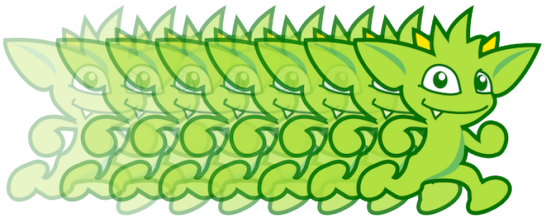

```
print(t.V(jon).repeat(__.in("Parent")).times(2));
```

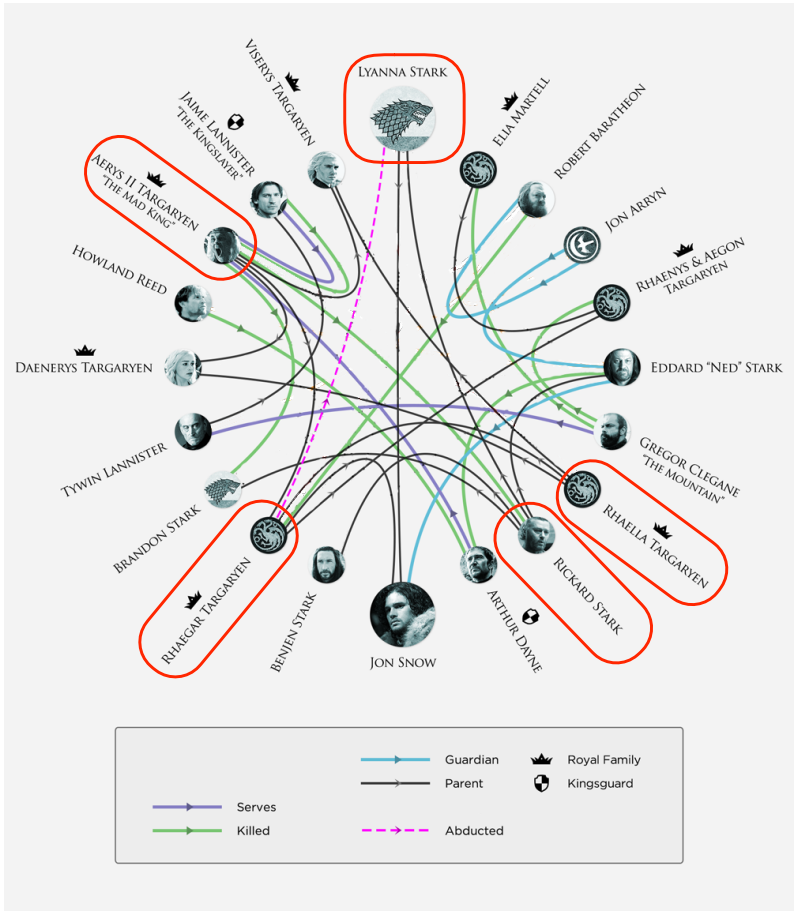
```
print(t.V(jon).repeat(__.in("Parent").store("Parents"))
        .cap("Parents").unfold());
```

Who are the grandparents of “Jon Snow”?

ScaDS DRESDEN LEIPZIG

REPEAT





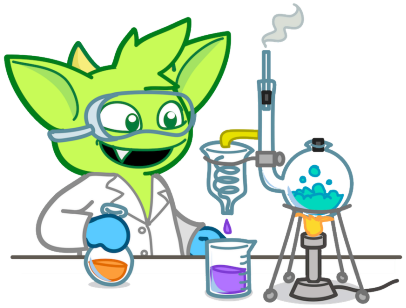

```
print(t.V(jon).repeat(__.in("Parent")).times(2));
```

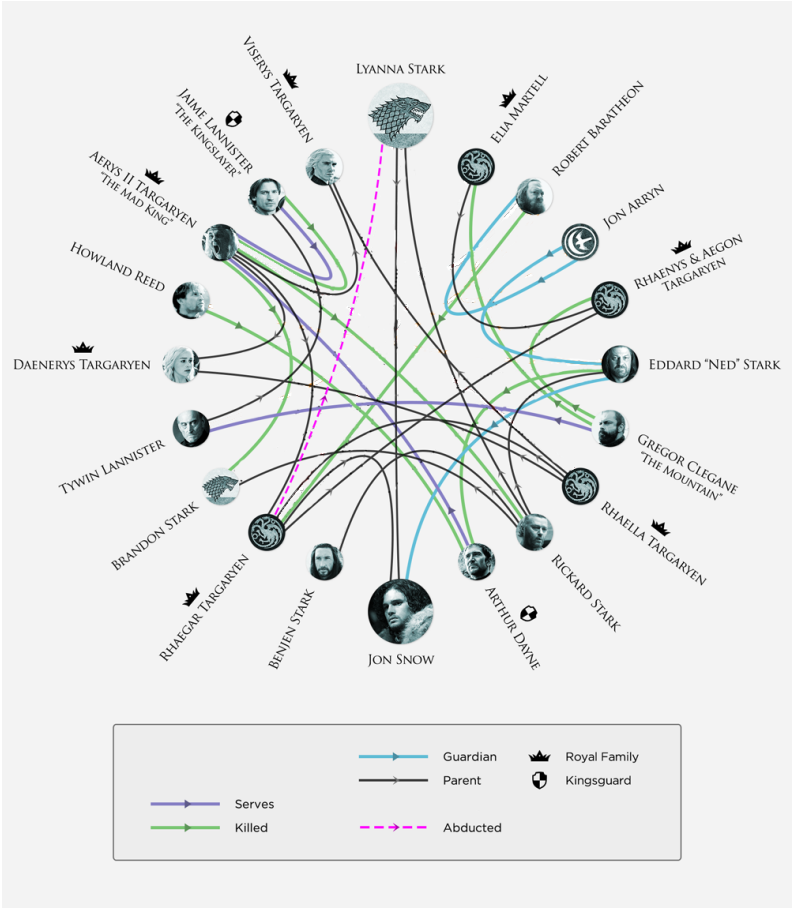
```
print(t.V(jon).repeat(__.in("Parent").store("Parents"))
        .cap("Parents").unfold());
```

Who are the parents and grandparents of “Jon Snow”?

ScaDS DRESDEN LEIPZIG

STORED DATA

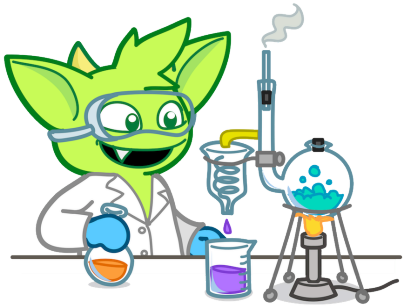


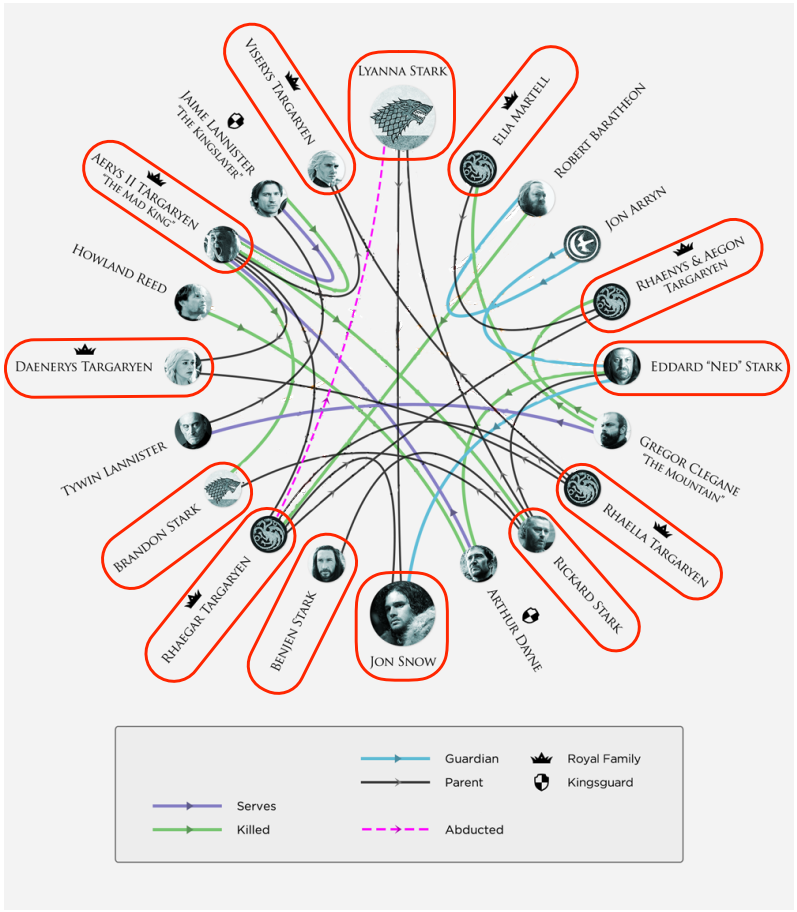



```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").unfold());
```

```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").<Vertex>unfold()
        .where(__.out("Killed").where(P.within("Family"))));
```

STORED DATA





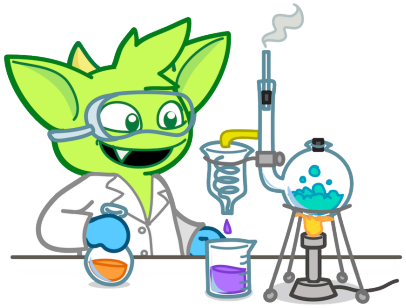

```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").unfold());
```

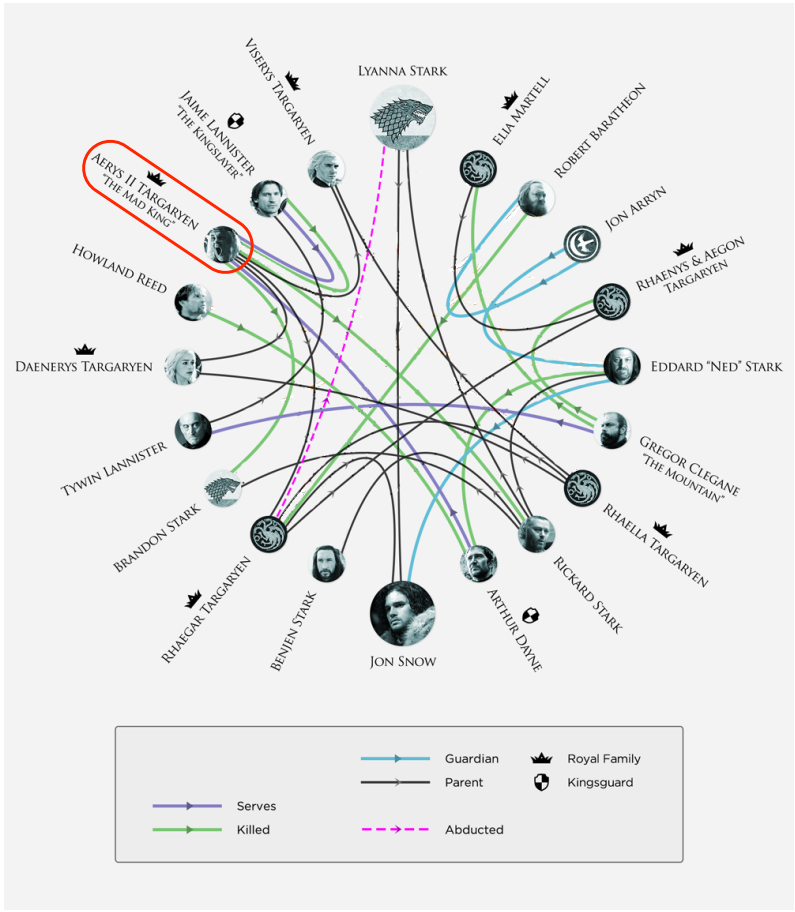
```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").<Vertex>unfold()
        .where(__.out("Killed").where(P.within("Family"))));
```

Who are “Jon Snow”s relatives?

ScaDS DRESDEN LEIPZIG

STORED DATA





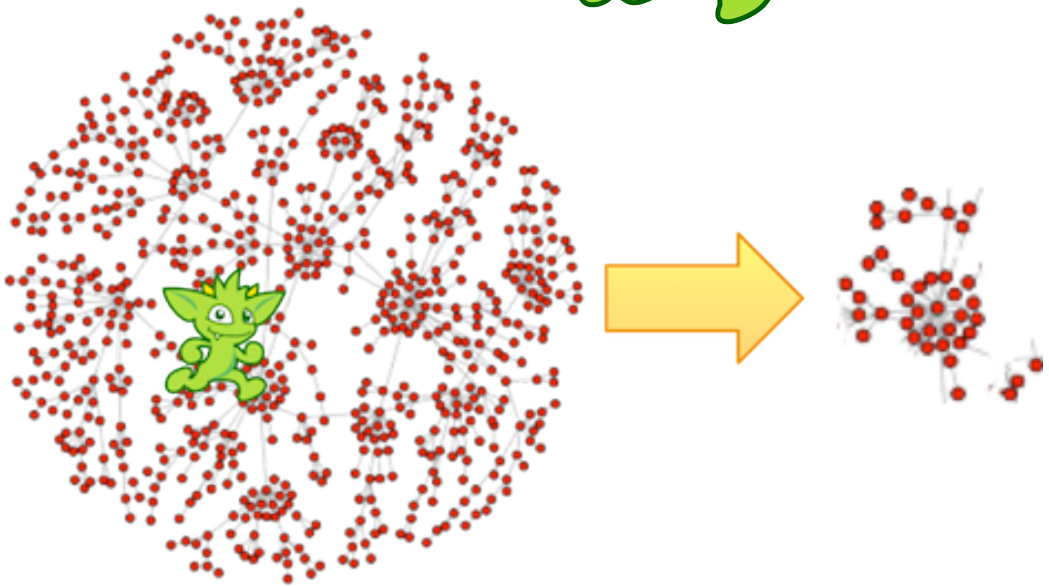

```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").unfold());
```

```
print(t.V(jon).store("Family")
        .repeat(__.both("Parent").dedup()
                .where(P.without("Family")).store("Family")
        )
        .cap("Family").<Vertex>unfold()
        .where(__.out("Killed").where(P.within("Family"))));
```

Who of “Jon Snow”s relatives killed an other relativ?

OTHER POSSIBILITIES

- choose
- project
- sack
- order
- math
- sideEffect
- subgraph

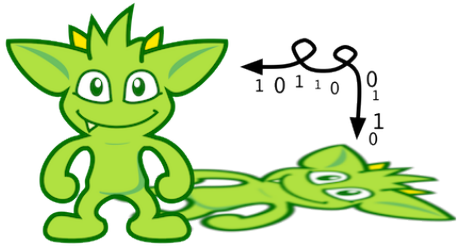


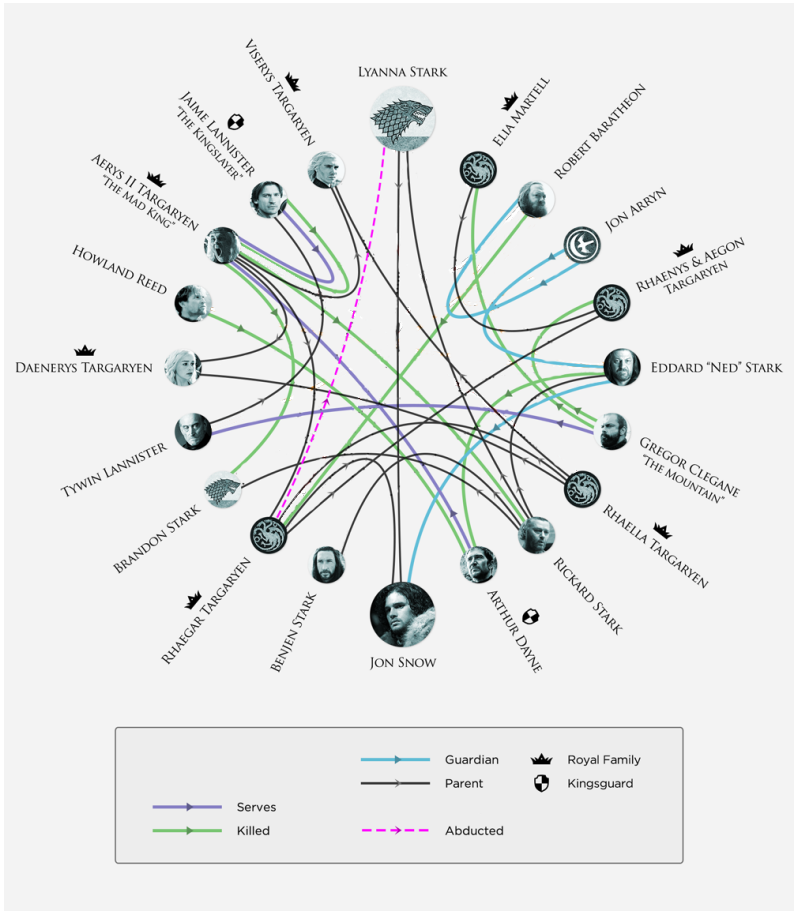
ScaDS
DRESDEN LEIPZIG

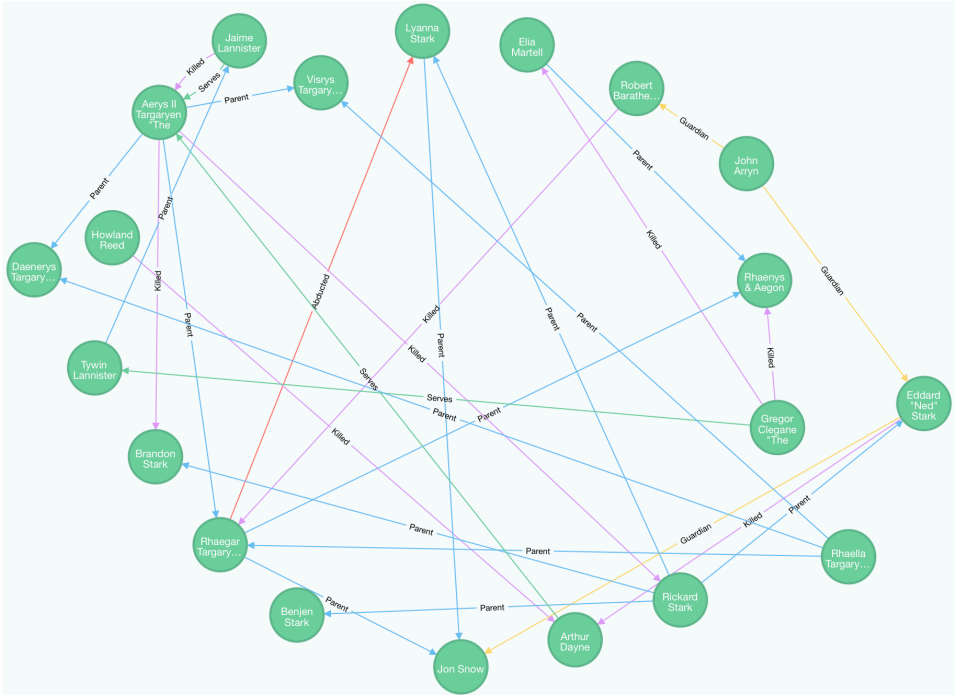
http://tinkerpop.apache.org

THANK YOU FOR YOUR ATTENTION

THE GRAPH


1 0 1 1 0
0
1
1
0


Lyanna Stark
Viserys Targaryen
Jaime Lannister "The Kingslayer"
Aerys II Targaryen "The Mad King"
Howland Reed
Daenerys Targaryen
Tywin Lannister
Brandon Stark
Rhaegar Targaryen
Benjen Stark
Jon Snow
Arthur Dayne
Rickard Stark
Rhaella Targaryen
Gregor Clegane "The Mountain"
Eddard "Ned" Stark
Rhaenys & Aegon Targaryen
Jon Arryn
Robert Baratheon
Elia Martell
Guardian
Parent
Serves
Killed
Royal Family
Kingsguard
Abducted


Jaime Lannister
Lyanna Stark
Elia Martell
Visrys Targary…
Robert Barathe…
Aerys II Targaryen "The
John Arryn
Howland Reed
Daenerys Targary…
Rhaenys & Aegon
Tywin Lannister
Gregor Clegane "The
Eddard "Ned" Stark
Brandon Stark
Rhaegar Targary…
Rhaella Targary…
Rickard Stark
Benjen Stark
Arthur Dayne
Jon Snow
Killed
Serves
Parent
Guardian
Abducted

ScaDS DRESDEN LEIPZIG

APACHE TINKERPOP