Graph Databases
Titan

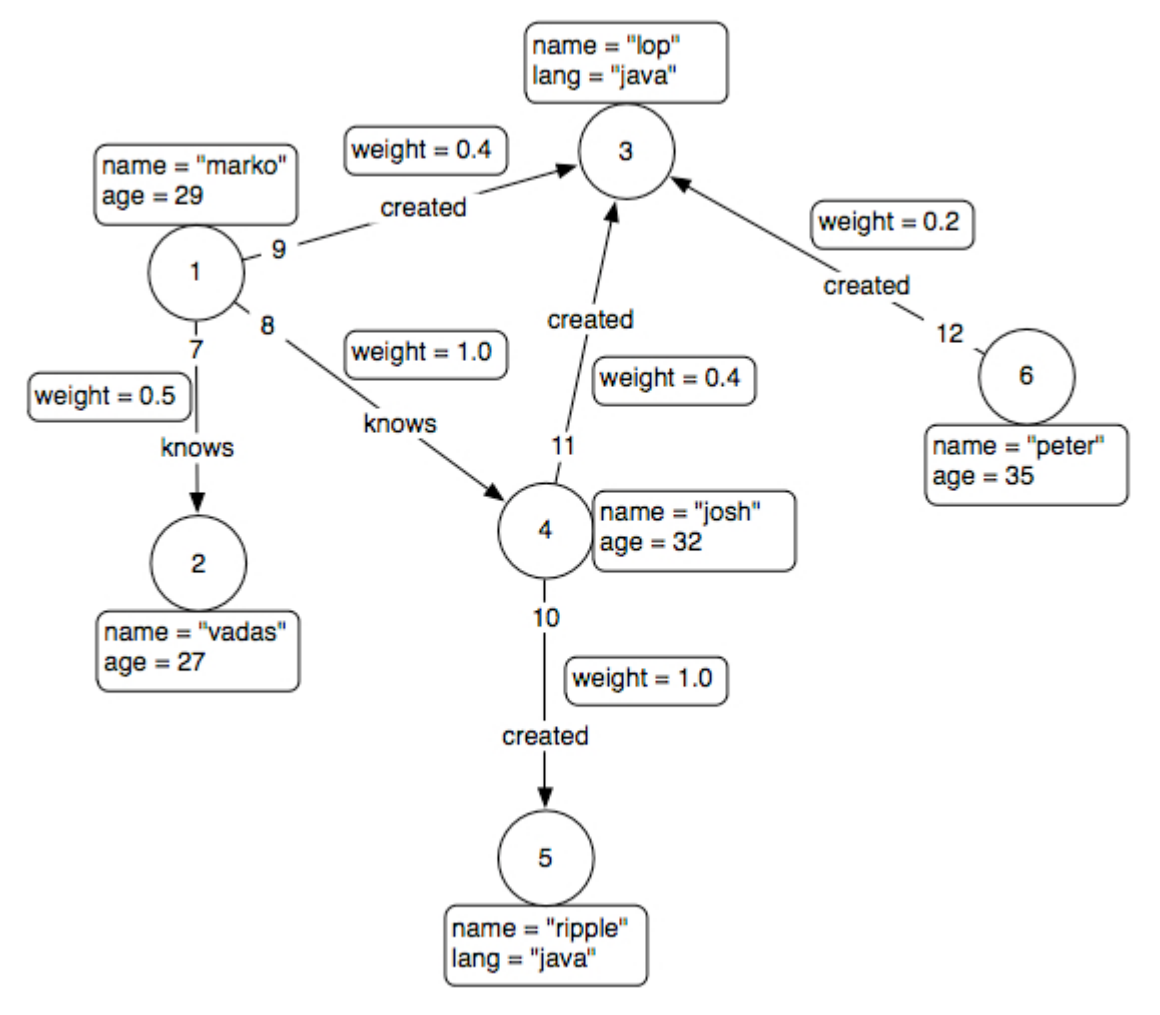
Raqueline Penteado

Definições

- *Graph database*
  - Um SGBD em Grafo é um sistema de gerenciamento de dados online com métodos CRUD (Create, Read, Update, and Delete - CRUD) para manipular um modelo de dados em grafo.

  - G = (V, E)

Definições


name = "lop"
lang = "java"
3
weight = 0.4
created
name = "marko"
age = 29
1
9
weight = 0.2
created
12
6
name = "peter"
age = 35
created
8
weight = 1.0
weight = 0.4
7
weight = 0.5
knows
knows
11
4
name = "josh"
age = 32
2
name = "vadas"
age = 27
10
weight = 1.0
created
5
name = "ripple"
lang = "java"

Definições

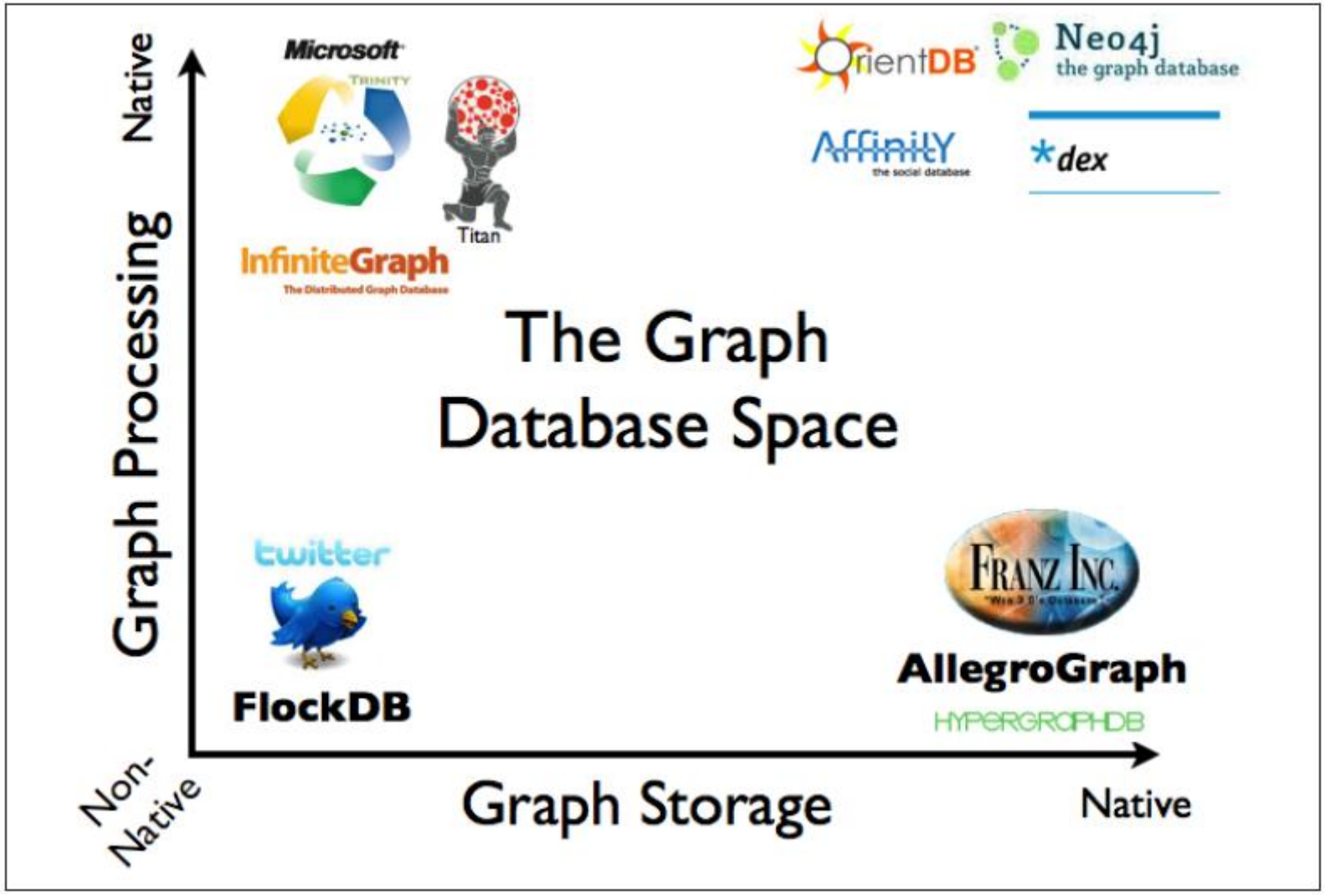
- Alguns pontos importantes
  - Armazenamento físico (*underlying storage*)
    - Nativo
    - Relacional…
  - Processamento (*processing engine*)
    - Tipos de consultas
      - Algoritmos
    - Índices
  - Distribuição
    - Dados
    - Consulta

Titan

- Classificação/características
- Arquiteturas
- Formas de armazenamento
- Indexação
- Consultas
- Transações
- Comparações

**Cassandra

Exemplos

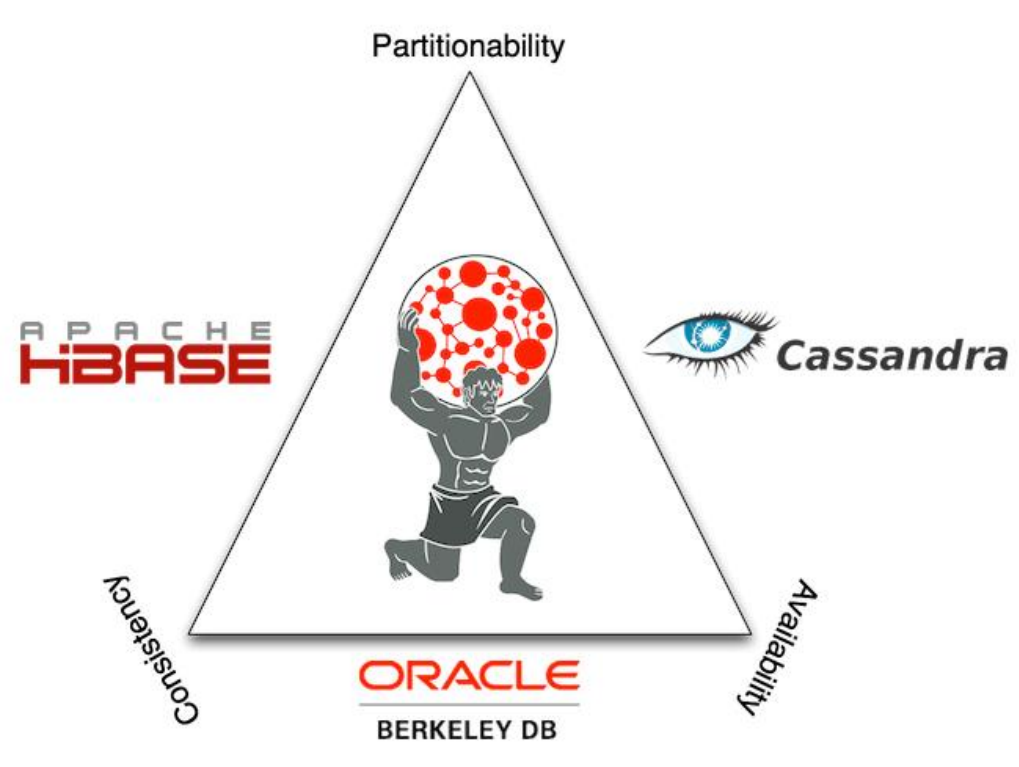

Graph Processing
Native
Non-Native
Graph Storage
Native
The Graph Database Space
Microsoft
TRINITY
Titan
InfiniteGraph
The Distributed Graph Database
OrientDB
Neo4j
the graph database
Affinity
the social database
*dex
twitter
FlockDB
FRANZ INC.
AllegroGraph
HYPERGRAPHDB

Características

- Transacional
- Suporte a grande volume de dados
- Escalável
  - Usuários/Dados
    - Distribuição e replicação de dados
- Licença
  - Open source com licença Apache 2

Características

- Formas de armazenamento
  - Cassandra , Hbase e Oracle BerkeleyDB

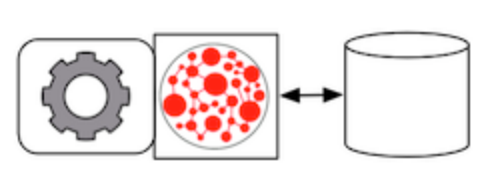


Características

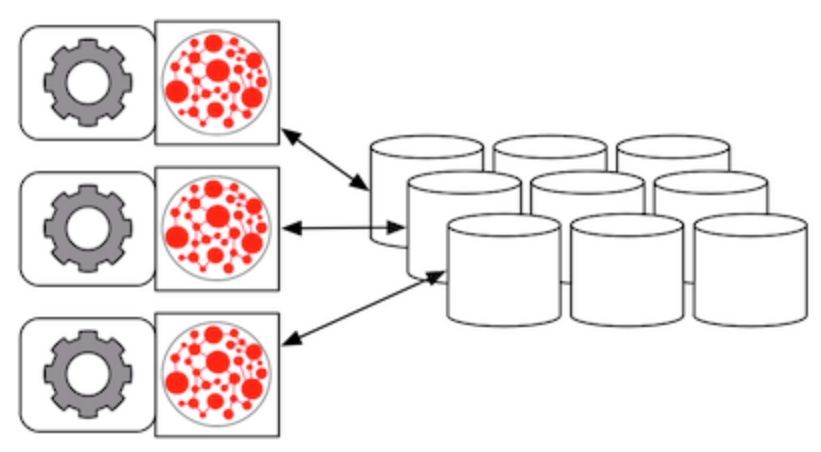
- Integração nativa com Tinkerpop
  - Gremlin: linguagem de consulta
  - Frames: mapeia objeto-para-grafo
  - Rexster: servidor de grafos
  - Blueprints: API padrão para banco de dados em grafos
  - Suporta
    - Neo4j, Titan, OrientDB, DEX, TinkerGraph

Arquiteturas

- Modo servidor local

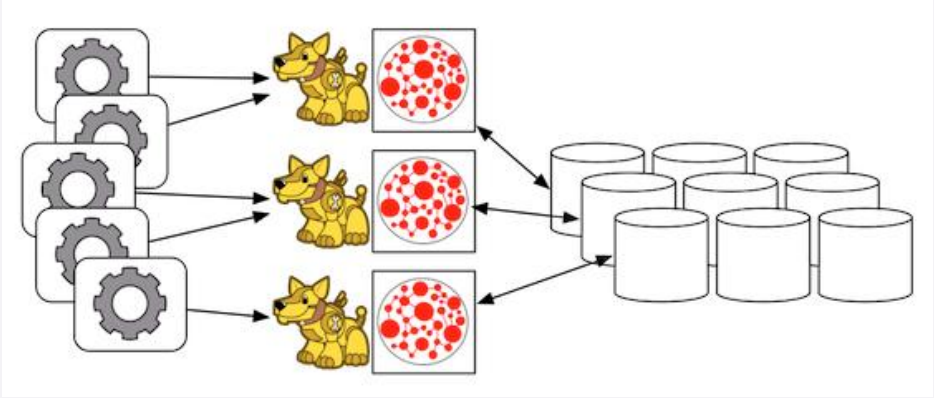


- Modo servidor remoto

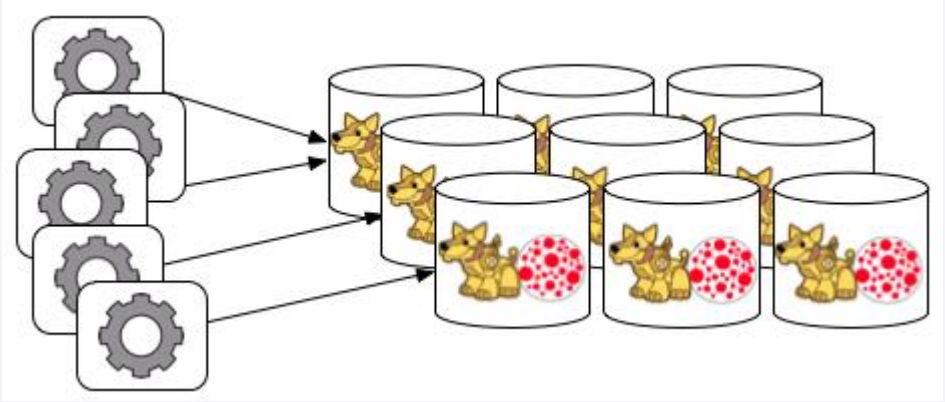


Arquiteturas

- Modo servidor remoto com Rexster

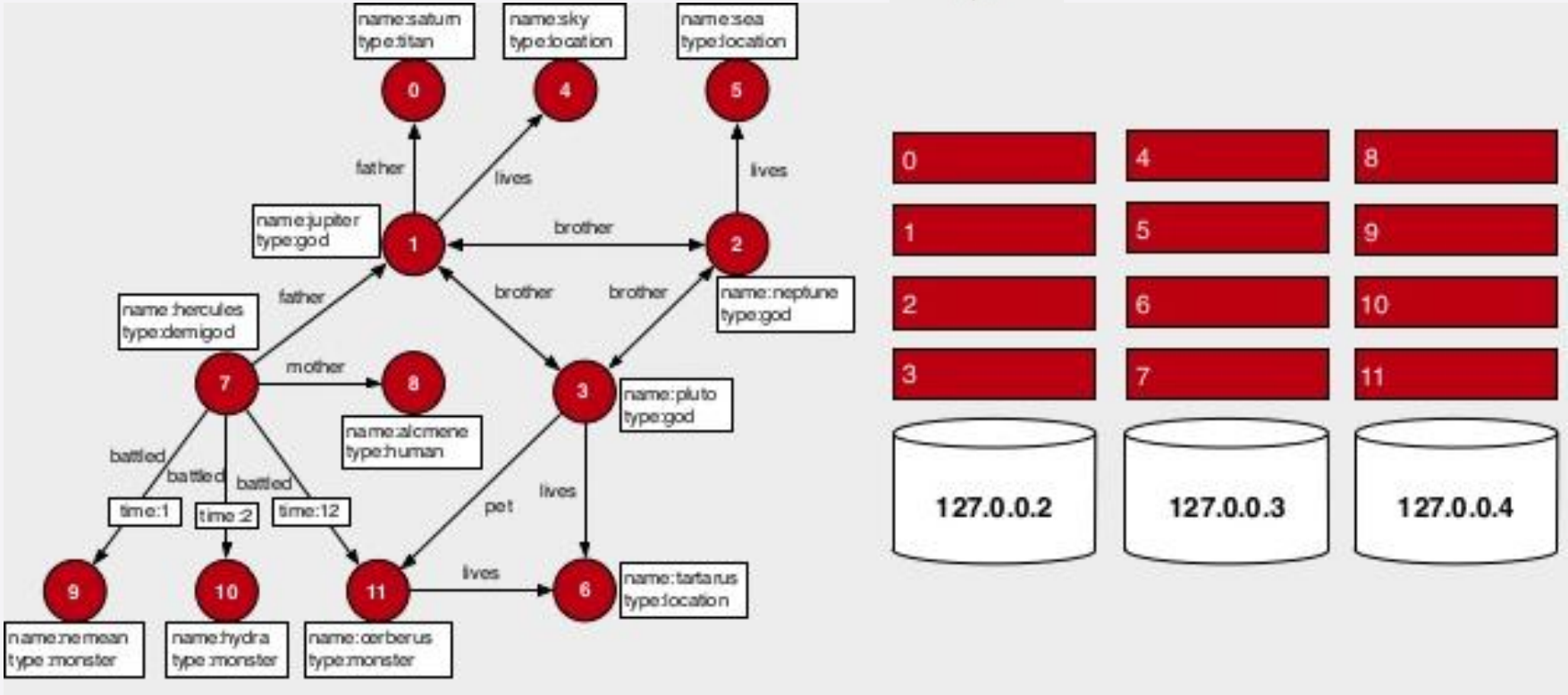


- Modo embutido



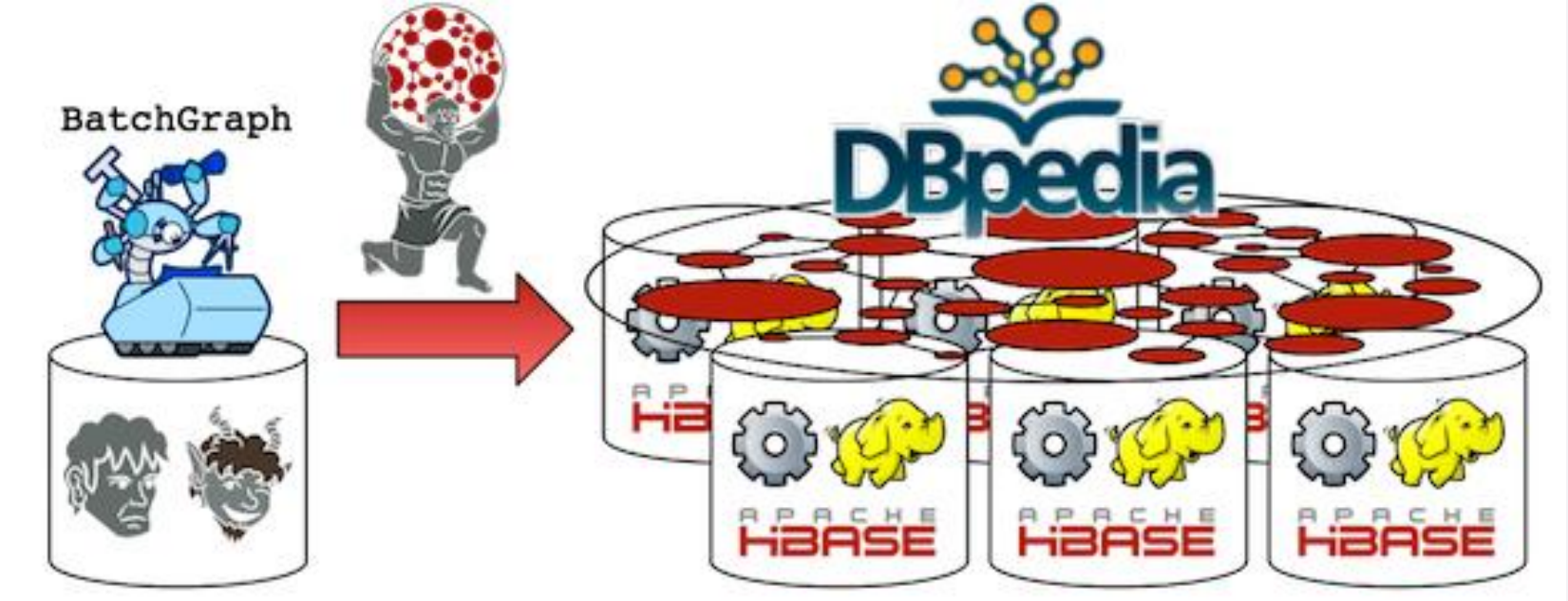
Arquiteturas

- Operações globais – Faunus

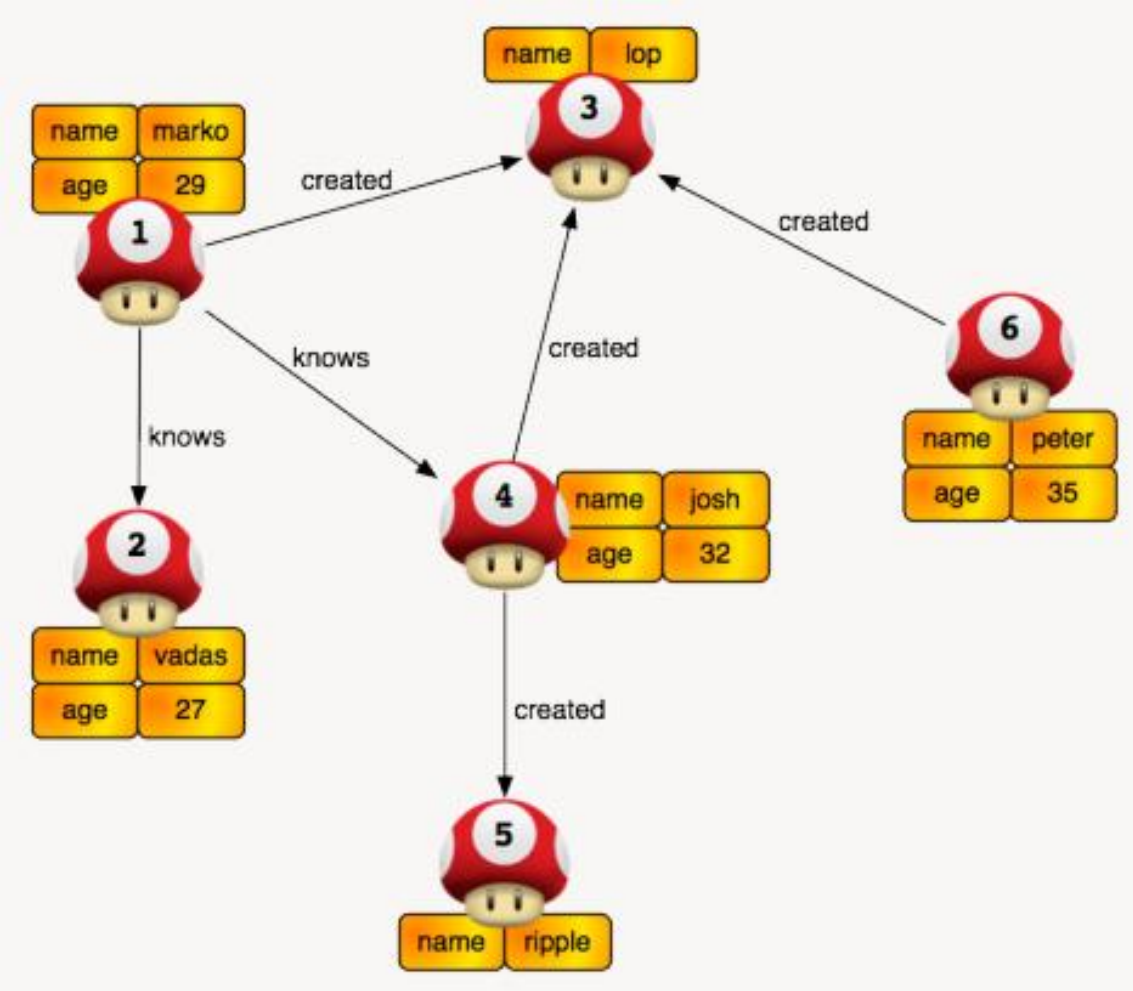


Arquiteturas

- Operações globais – Faunus
  - OLAP
  - RDF



Armazenando



Armazenando

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
    <key id="weight" for="edge" attr.name="weight" attr.type="float"/>
    <key id="name" for="node" attr.name="name" attr.type="string"/>
    <key id="age" for="node" attr.name="age" attr.type="int"/>
    <key id="lang" for="node" attr.name="lang" attr.type="string"/>
    <graph id="G" edgedefault="directed">
        <node id="1">
            <data key="name">marko</data>
            <data key="age">29</data>
        </node>
        <node id="2">
            <data key="name">vadas</data>
            <data key="age">27</data>
        </node>
        <node id="3">
            <data key="name">lop</data>
            <data key="lang">java</data>
        </node>
        <node id="4">
            <data key="name">josh</data>
            <data key="age">32</data>
        </node>
```

Armazenando

```
        <node id="5">
            <data key="name">ripple</data>
            <data key="lang">java</data>
        </node>
        <node id="6">
            <data key="name">peter</data>
            <data key="age">35</data>
        </node>
        <edge id="7" source="1" target="2" label="knows">
            <data key="weight">0.5</data>
        </edge>
        <edge id="8" source="1" target="4" label="knows">
            <data key="weight">1.0</data>
        </edge>
        <edge id="9" source="1" target="3" label="created">
            <data key="weight">0.4</data>
        </edge>
        <edge id="10" source="4" target="5" label="created">
            <data key="weight">1.0</data>
        </edge>
        <edge id="11" source="4" target="3" label="created">
            <data key="weight">0.4</data>
        </edge>
        <edge id="12" source="6" target="3" label="created">
            <data key="weight">0.2</data>
        </edge>
    </graph>
</graphml>
```

Armazenando

- Memória (TinkerGraph)

```
gremlin$ ./gremlin.sh

         \,,,/
         (o o)
-----oOOo-(_)-oOOo-----
gremlin> g = new TinkerGraph()
==>tinkergraph[vertices:0 edges:0]
gremlin> g.loadGraphML('data/graph-example-1.xml')
==>null
gremlin>
```

```
gremlin> g = new TinkerGraph()
==>tinkergraph[vertices:0 edges:0]
gremlin> v1 = g.addVertex(100)
==>v[100]
gremlin> v2 = g.addVertex(200)
==>v[200]
gremlin> g.addEdge(v1,v2,'friend')
==>e[0][100-friend->200]
gremlin> g.addEdge(1000,v1,v2,'buddy')
==>e[1000][100-buddy->200]
gremlin> g.addEdge(null,v1,v2,'pal',[weight:0.75f])
==>e[1][100-pal->200]
```

Armazenando

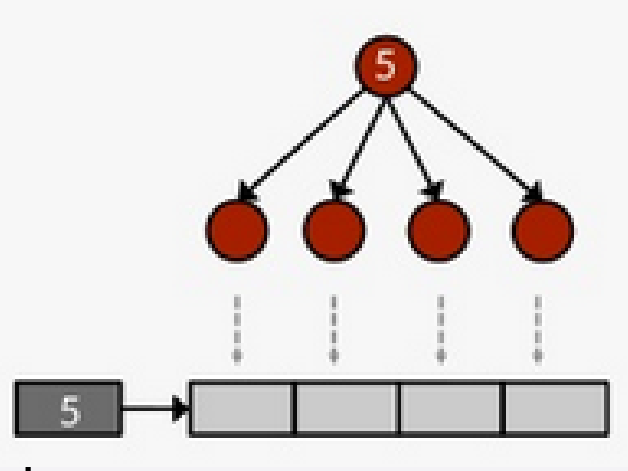
- Cassandra
  - Via Gremlin

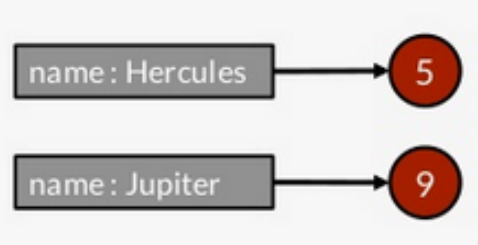
```
Configuration conf = new BaseConfiguration();
conf.setProperty("storage.backend","cassandra");
conf.setProperty("storage.hostname","127.0.0.1");
TitanGraph g = TitanFactory.open(conf);
```

```
Configuration conf = new BaseConfiguration();
conf.setProperty("storage.backend","cassandra");
conf.setProperty("storage.hostname","77.77.77.77");
TitanGraph g = TitanFactory.open(conf);
```

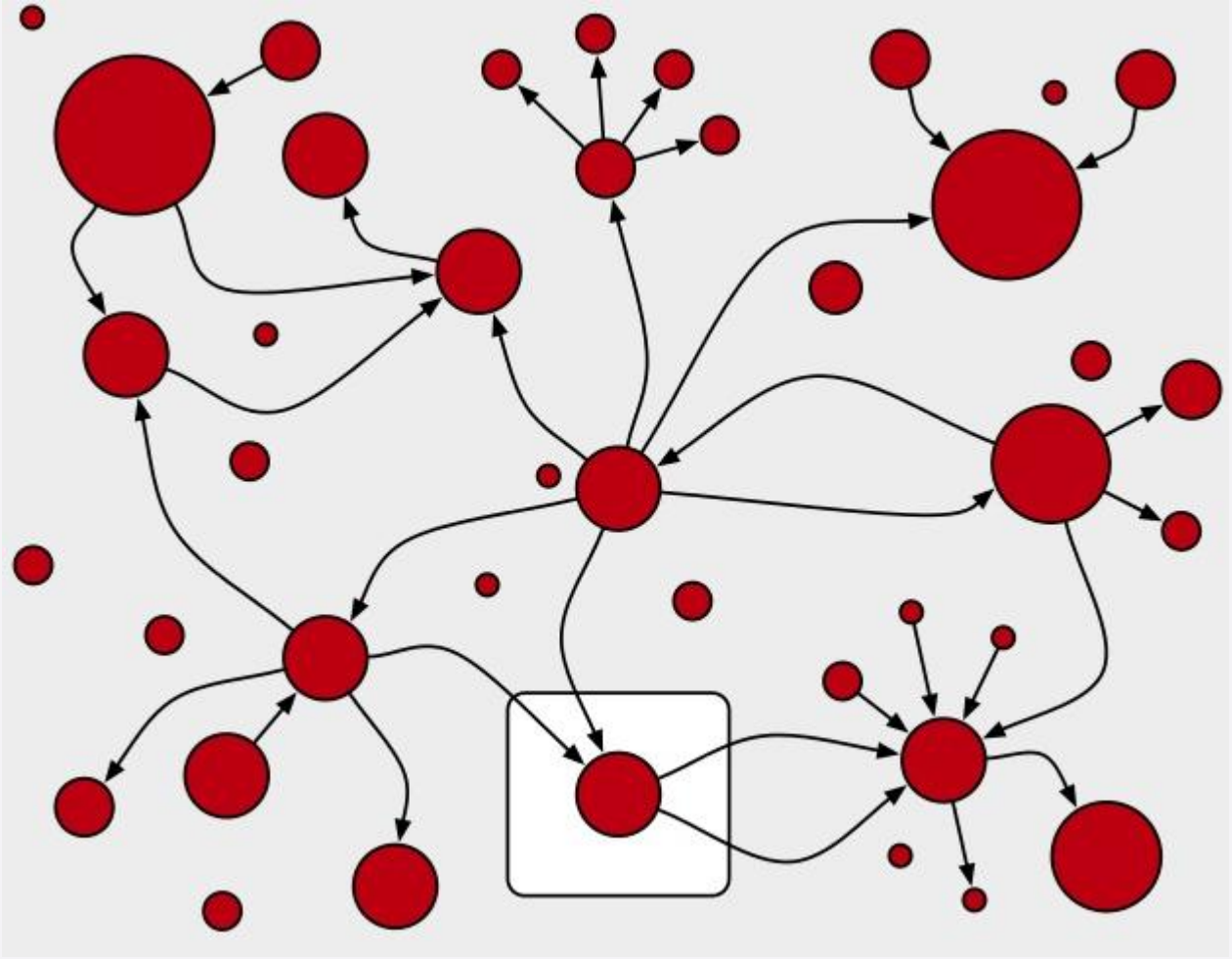
Armazenando

- Cassandra
  - Lista de adjacência em uma família de colunas
  - Chave (key) da linha = Id_vértice
  - Cada propriedade e aresta é uma coluna
  - Índices
    - Vértices
      - .index()
    - Arestas
      - Primarykey

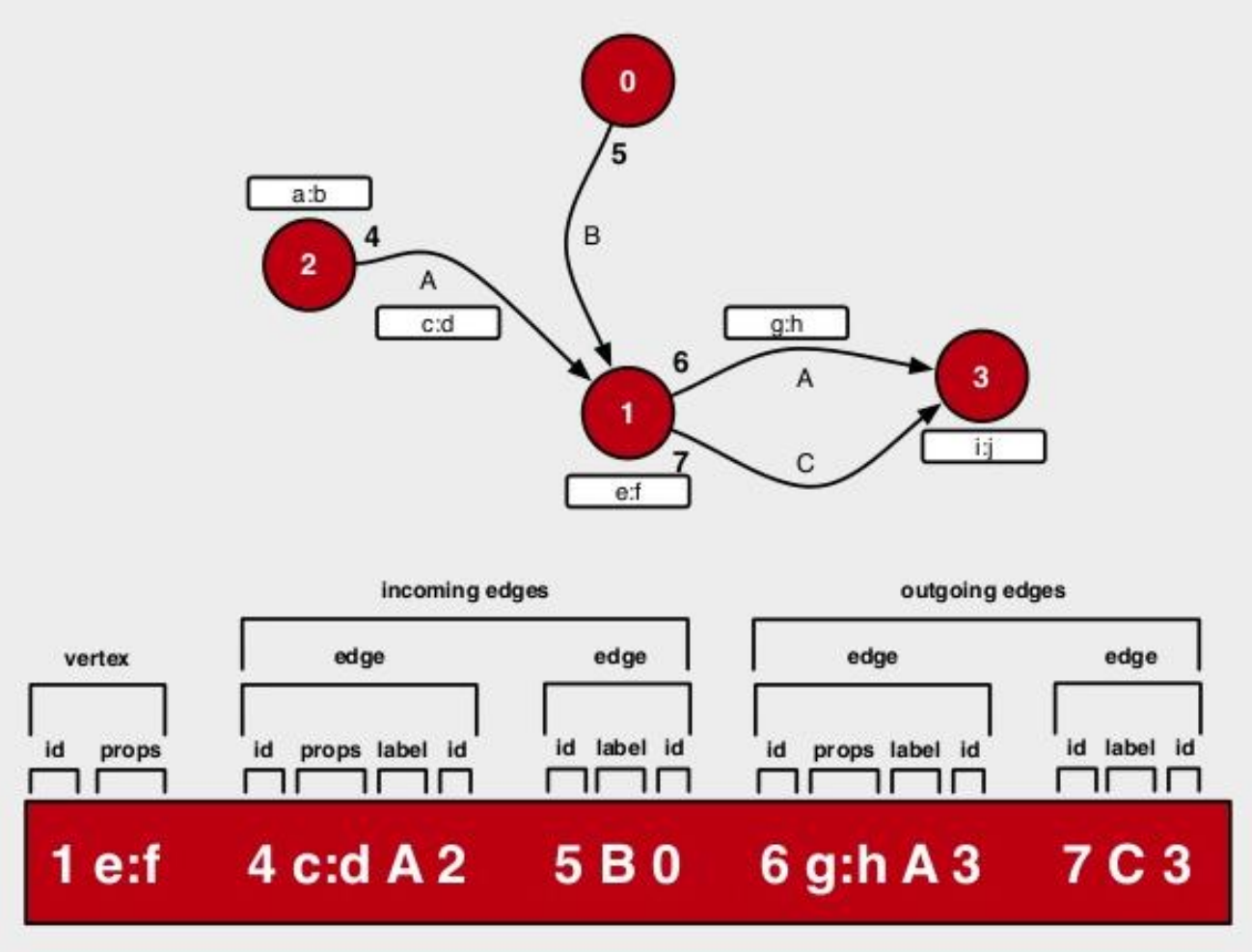




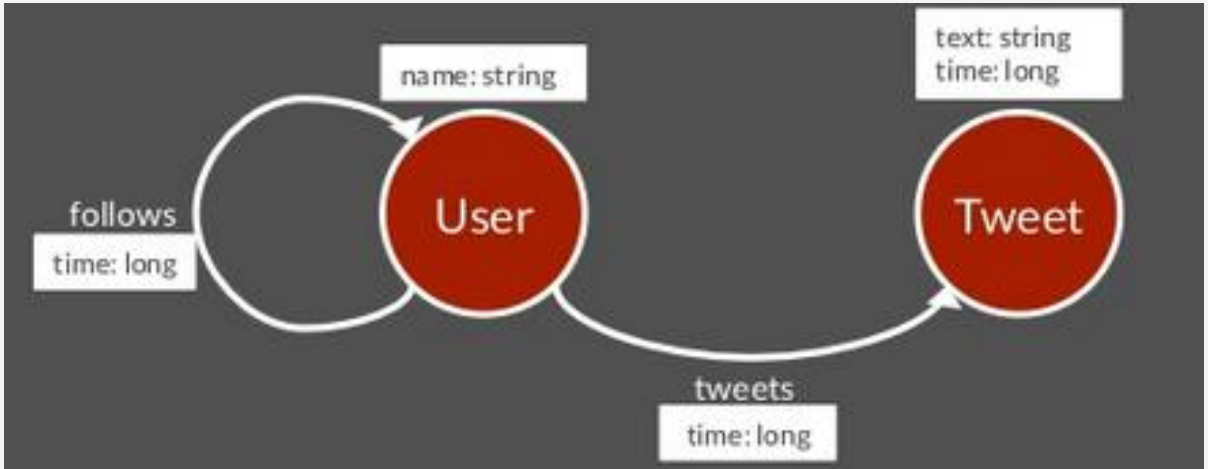
Armazenando



Armazenando



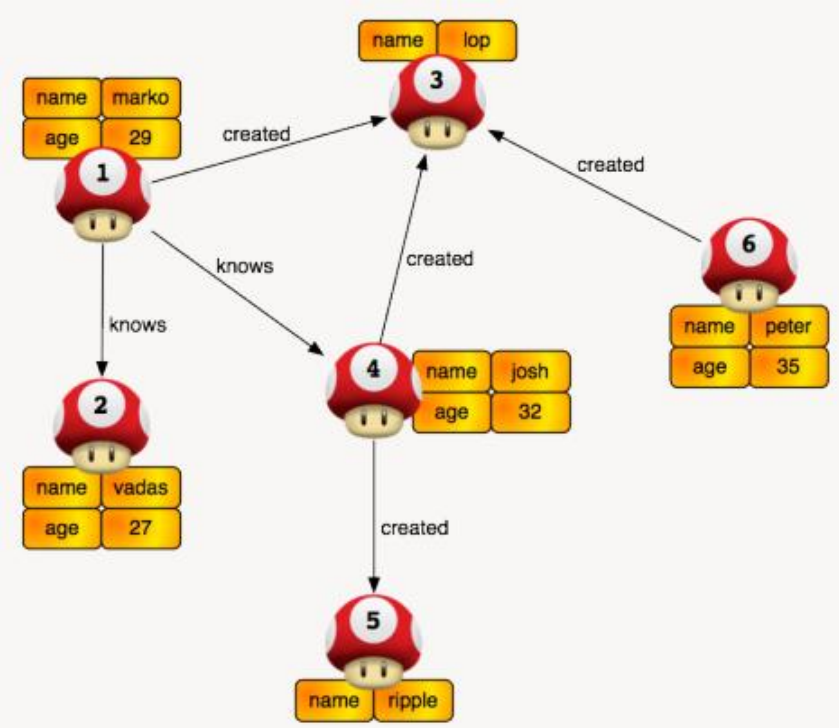
Indexando




```
gremlin> g.makeType().name("time").
         dataType(Long.class).
         functional().
         makePropertyKey();
gremlin> g.makeType().name("text").dataType(String.class).
         functional().makePropertyKey();
gremlin> g.makeType().name("name").dataType(String.class).
         indexed().
         unique().
         functional().makePropertyKey();
gremlin> g.makeType().name("follows").
         primaryKey(time).
         makeEdgeLabel();
gremlin> g.makeType().name("tweets").
         primaryKey(time).makeEdgeLabel();
```

Consultando

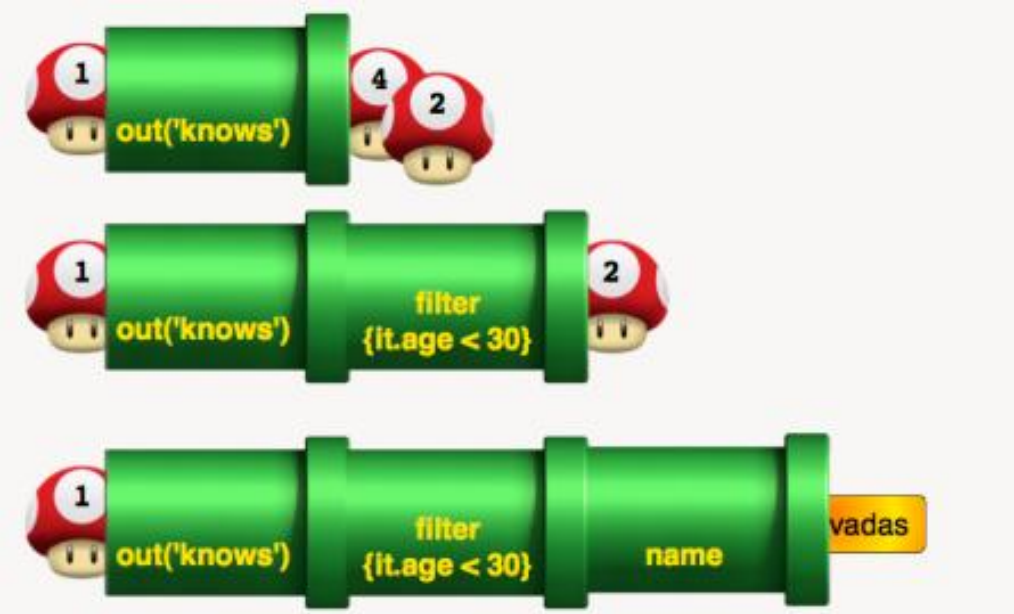
- Gremlin




```
gremlin> g.v(1).out.name
==>vadas
==>lop
==>josh
```


```
gremlin> g.v(1).out
==>v[2]
==>v[3]
==>v[4]
gremlin> g.v(1).out.name
==>vadas
==>lop
==>josh
```

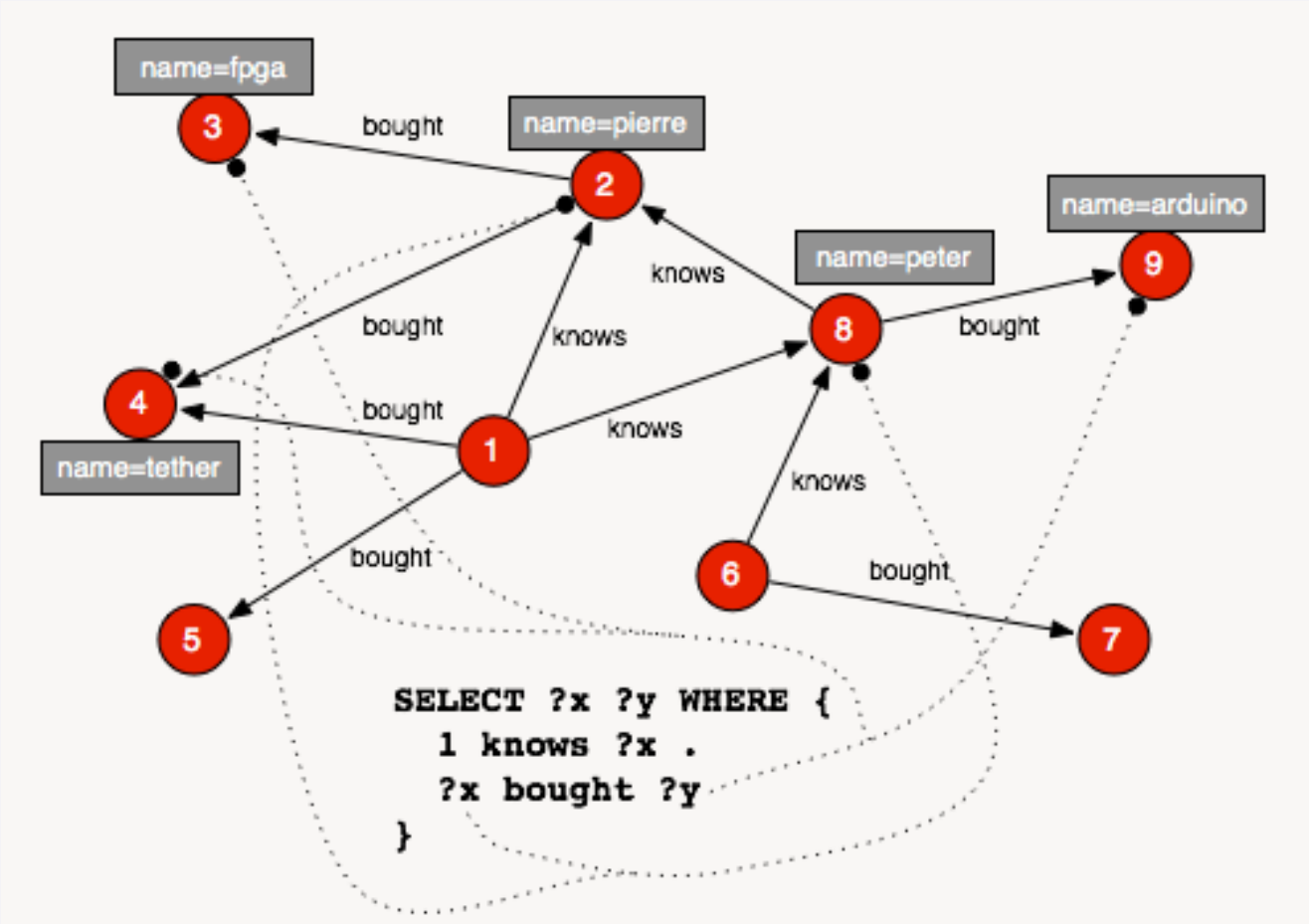
Consultando




```
gremlin> g.v(1).out('knows')
==>v[2]
==>v[4]
gremlin> g.v(1).out('knows').filter{it.age < 30}
==>v[2]
gremlin> g.v(1).out('knows').filter{it.age < 30}.name
==>vadas
```

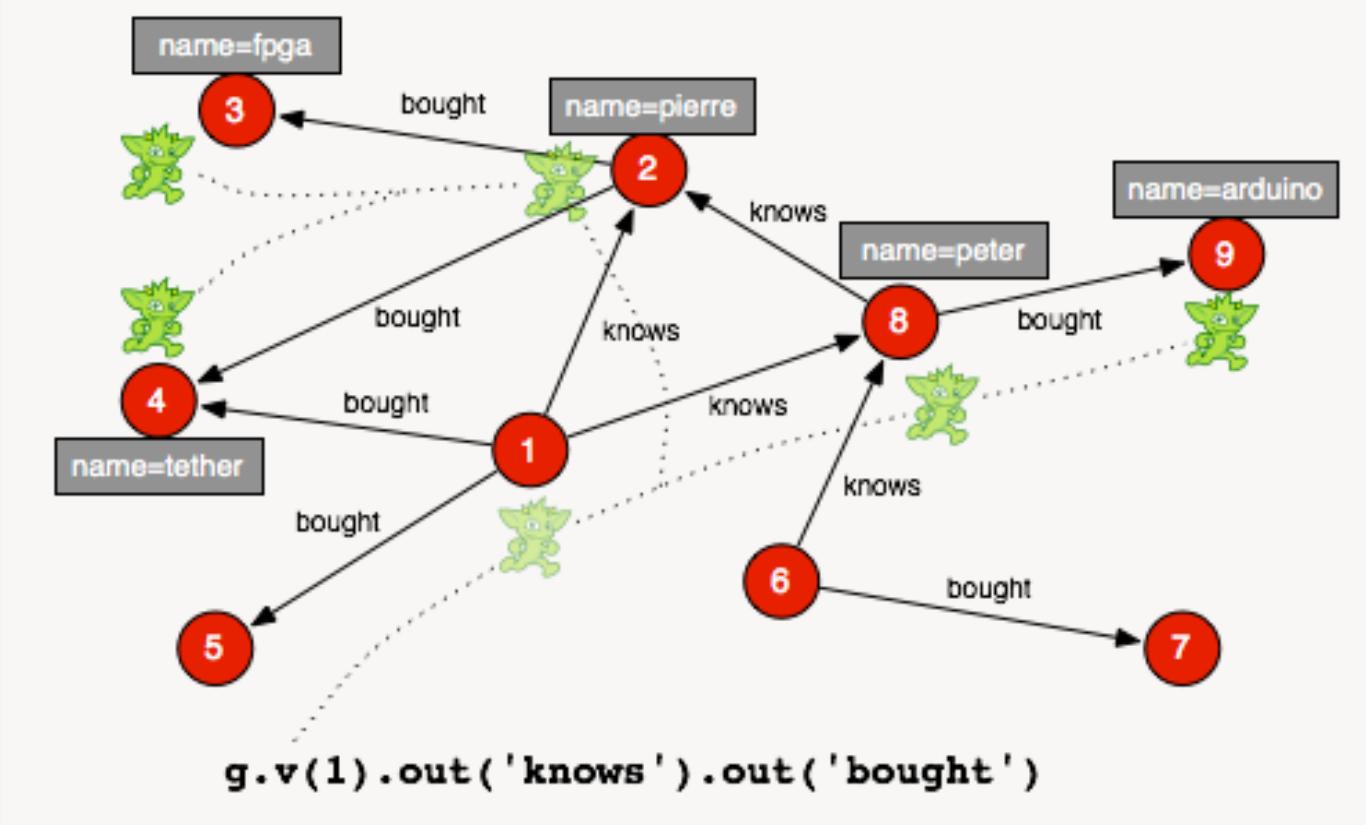
Consultando

- Combinando padrões de triplas



Consultando

- Combinando padrões de triplas



Consultando

- Combinando padrões de triplas

```
SELECT ?x ?y WHERE {
  1 knows ?x .
  ?x bought ?y
}

--------------
| ?x   | ?y   |
|-------------|
| v[8] | v[9] |
| v[2] | v[4] |
| v[2] | v[3] |
 -------------
```

```
gremlin> t = new Table()
gremlin> g.v(1).out('knows').as('x').out('bought').as('y').table(t)
==>v[9]
==>v[4]
==>v[3]
gremlin> t
==>[x:v[8], y:v[9]]
==>[x:v[2], y:v[4]]
==>[x:v[2], y:v[3]]
```

Consultando

- Scripts
  - Conjunto de operações
    - Ler um grafo
    - Consultar o grafo
    - Armazenar o resultado em um arquivo
    - Ler outro grafo
    - Consulta o grafo
    - Armazenar o resultado em um arquivo
    - Trabalhar com o resultado do arquivo
    - Enviar resposta ao usuário

Transações

- Isolamento, concorrência, consistência eventual
- Uma operação inicia a transação (primeira)
- A transação é finalizada explicitamente com commit(), roolback() ou shutdown() (commit automaticamente)

```
gremlin> g = TitanFactory.open('bin/cassandra.local')
==>titangraph[cassandra:127.0.0.1]
gremlin> g.loadGraphML('data/graph-of-the-gods.xml')
==>null
gremlin> g.commit()
```

Comparando

	Handle dynamic graph	Native graph model	Distributed storage	Distributed query process-ing	Memory based explo-ration	Transaction and index support
Neo4J	Yes	Yes	No	No	No	Yes
Titan	Yes	Yes	Yes	No	No	Yes
GBase	No	No	Yes	Yes	No	No
Trinity	Yes	Yes	Yes	Yes	Yes	No
*im*Graph	Yes	Yes	Yes	Yes	Yes	Yes

Salim Jouili e Aldemar Reynaga. *imGraph: A distributed in-memory graph database.* ASE/IEEE International Conference on Big Data. 2013.